NORMA
BRASILEIRA

ABNT NBR 15606-6

Primeira edição
16.07.2010

Válida a partir de
16.08.2010

Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital
Parte 6: Java DTV 1.3

Digital terrestrial television – Data coding and transmission specification for digital broadcasting
Part 6: Java DTV 1.3

ICS 33.160.01

ISBN 978-85-07- 02188-9

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS

Número de referência
ABNT NBR 15606-6-2010
860 páginas

ABNT
Av.Treze de Maio, 13 - 28º andar
20031-901 - Rio de Janeiro - RJ
Tel.: + 55 21 3974-2300
Fax: + 55 21 3974-2346
abnt@abnt.org.br
www.abnt.org.br

Sumário

Página

Prefácio

A Associação Brasileira de Normas Técnicas (ABNT) é o Foro Nacional de Normalização. As Normas Brasileiras, cujo conteúdo é de responsabilidade dos Comitês Brasileiros (ABNT/CB), dos Organismos de Normalização Setorial (ABNT/ONS) e das Comissões de Estudo Especiais (ABNT/CEE), são elaboradas por Comissões de Estudo (CE), formadas por representantes dos setores envolvidos, delas fazendo parte: produtores, consumidores e neutros (universidades, laboratórios e outros).

Os Documentos Técnicos ABNT são elaborados conforme as regras das Diretivas ABNT, Parte 2.

A Associação Brasileira de Normas Técnicas (ABNT) chama atenção para a possibilidade de que alguns dos elementos deste documento podem ser objeto de direito de patente. A ABNT não deve ser considerada responsável pela identificação de quaisquer direitos de patentes.

A ABNT NBR 15606-6 foi elaborada pela Comissão de Estudo de Televisão Digital (ABNT/CEE-85). O Projeto circulou em Consulta Nacional conforme Edital nº 05, de 12.05.2010 a 12.07.2010, com o número de Projeto 85:000.00-006/6.

A especificação da tecnologia Java DTV 1.3 é fruto da celebração de um contrato de cooperação técnica entre o Fórum do Sistema Brasileiro de Televisão Digital Terrestre e a Sun Microsystem (agora parte da Oracle® Corporation) em 15 de maio de 2008. Esta especificação está em total observância às diretrizes do Decreto Presidencial nº 5.820, de 29.06.2006, e com os entendimentos dos especialistas da ABNT/CEE-85.

A ABNT NBR 15606, sob o título geral “Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital”, tem previsão de conter as seguintes partes:

— Parte 1: Codificação de dados;

— Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações;

— Parte 3: Especificação de transmissão de dados;

— Parte 4: Ginga-J – Ambiente para a execução de aplicações procedurais;

— Parte 5: Ginga-NCL para receptores portáteis – Linguagem de aplicação XML para codificação de aplicações.

— Parte 6: Java DTV 1.3

O desenvolvimento dessa especificação foi regido por um acordo que estabeleceu a não utilização de componentes cuja propriedade intelectual estivesse sujeita ao pagamento de royalties e a gratuidade das tecnologias desenvolvidas.

Esta parte da ABNT NBR 15606 inclui termos utilizados pela linguagem Java para a descrição de pacotes, classes, métodos, interfaces, variáveis. Estes termos estão grafados em fonte Courier New. O emprego dessa fonte visa destacar claramente os termos que devem ser usados em todas as implementações desta parte da ABNT NBR 15606 exatamente como aparecem, por serem termos técnicos significativos para tais implementações.

De forma análoga, para os trechos de código fonte em linguagens de programação (Java, XML, etc.), emprega-se fonte Courier New com 9 pts. O uso desse formato objetiva a diferenciação dos exemplos de código fonte que podem ser implementados seguindo as regras desta especificação.

Excepcionalmente, nas seções que descrevem pacotes e classes da linguagem assim como nas diversas seções de índices e detalhes, nomes de classes e pacotes assim como listas de nomes de classes, métodos, exceções, campos etc. são representados em Arial Negrito usando formatos padronizados da linguagem Java, o qual consiste em agregar diversas palavras, sem espaços e iniciando cada uma delas com letra maiúscula exceto a primeira que inicia com letra minúscula.

Os verbos DEVE(M), CONVÉM(ÊM), PODE(M) e suas negativas (NÃO PODE(M), NÃO CONVÉM(ÊM)), quando necessitam de especial destaque, são apresentados em caixa alta, seguindo as regras da RFC 2119 usada na versão em inglês e tornando o texto preparado para o processamento com ferramentas automáticas especiais que se guiam pelo uso desses tempos verbais em maiúsculas para achar os casos de especial destaque descritos na especificação.

Java é uma marca registrada da Oracle® Corporation e/ou suas afiliadas.

O Escopo desta Norma Brasileira em inglês é o seguinte:

Scope

This part of ABNT NBR 15606 specifies the requirements for the basic core of the procedural part of the middleware for the Brazilian digital terrestrial televison system (SBTVD).

Introdução

A Associação Brasileira de Normas Técnicas (ABNT) chama atenção para o fato de que a exigência de conformidade com este Documento Técnico ABNT pode envolver o uso de patentes relativas à especificação JavaDTV constante no corpo deste documento.

A ABNT não se posiciona a respeito de evidências, validade e escopo deste direito de patente.

O proprietário deste direito de patente assegurou à ABNT que está preparado para negociar licenças sobre termos e condições razoáveis e não discriminatórias com os solicitantes. Sobre isto, uma declaração do proprietário desta patente está registrada com a ABNT. Informações podem ser obtidas com:

Oracle Corporation

Rua Alexandre Dumas, 2016

São Paulo - SP - Brazil 04717-004

A ABNT chama atenção para a possibilidade de que alguns dos elementos deste Documento Técnico ABNT podem ser objeto de outros direitos de patente além dos identificados acima. A ABNT não pode ser considerada responsável pela identificação de quaisquer direitos de patente.

O Anexo A apresenta detalhes sobre os direitos de patente da Oracle Corporation.

A Figura 1 representa o contexto em que esta parte da ABNT NBR 15606 é alocado.

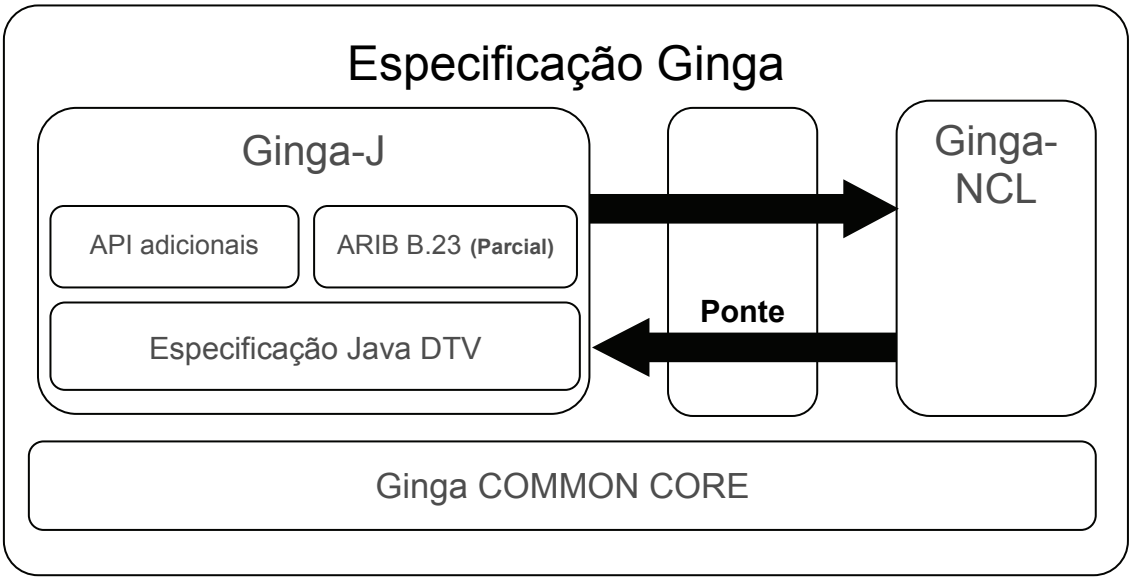


Figura 1 — Contexto Java DTV

Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 6: Java DTV 1.3

1 Escopo

Esta parte da ABNT NBR 15606 especifica os requerimentos do núcleo que serve de base para a parte procedural do *middleware* para o sistema brasileiro de televisão digital terrestre (SBTVD).

2 Referências normativas

Os documentos relacionados a seguir são indispensáveis à aplicação deste documento. Para referências datadas, aplicam-se somente as edições citadas. Para referências não datadas, aplicam-se as edições mais recentes do referido documento (incluindo emendas).

ABNT NBR 15602-1, *Televisão digital terrestre — Codificação de vídeo, codificação de áudio e multiplexação — Parte 1: Codificação de vídeo*

ABNT NBR 15603-2, *Televisão digital terrestre — Multiplexação e serviços de informação (SI) — Parte 2: Estrutura de dados e definições de informações básicas da SI*

ABNT NBR 15606-1, *Televisão digital terrestre — Codificação de dados e especificação de transmissão para transmissão digital — Parte 1: Especificação de codificação de dados*

ABNT NBR 15606-3, *Televisão digital terrestre — Codificação de dados e especificação de transmissão para transmissão digital — Parte 3: Especificação de transmissão de dados*

ARIB STD-B24 Version 5.1-E1, Data Coding and Transmission Specification for Digital Broadcasting - Volume 1

ETR 154 (ETSI TS 101 154 V1.5.1), *Digital Video Broadcasting (DVB); Implementation guidelines for the use of Video and Audio Coding in Broadcasting Applications based on the MPEG-2 Transport Stream*

ETSI EN 301 790, *Digital Video Broadcasting (DVB); Interaction channel for satellite distribution*

ETSI EN 50221, *Common interface specification for conditional access and other Digital Video Broadcasting decoder applications*

ISO/IEC 8859-1, *Information technology — 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1*

ISO 639-2, *Codes for the representation of names of languages - Part 2: Alpha-3 code*

ISO 3166-1, *Codes for the representation of names of countries and their subdivisions – Part 1: Country codes*

CDC, *Connected Device Configuration 1.1 - http://jcp.org/en/jsr/detail?id=218*

FP, *Foundation Profile 1.1 - http://jcp.org/en/jsr/detail?id=219*

PBP, *Personal Basis Profile 1.1 - http://jcp.org/en/jsr/detail?id=217*

JavaTV, *JavaTV 1.1, disponível em http://jcp.org/en/jsr/detail?id=927*

SATSA, *Security and Trust Services API para J2ME, disponível em http://jcp.org/en/jsr/detail?id=177*

JAR, *Especificação de Arquivo JAR: - http://java.sun.com/javase/6/docs/technotes/guides/jar/jar.html*

FIPS 180-1, *Federal Information Processing Standards Publication 180-1 - Secure Hash Standard - http://www.itl.nist.gov/fipspubs/fip180-1.htm*

RFC 1630, *Universal Resource Identifiers in WWW - http://www.ietf.org/rfc/rfc1630.txt*

RFC 1738, *Uniform Resource Locators (URL) - http://www.ietf.org/rfc/rfc1738.txt*

RFC 1952, *GZIP file format specification version 4.3 - http://www.ietf.org/rfc/rfc1921.txt*

RFC 2246, *The TLS Protocol Version 1.0 - http://www.ietf.org/rfc/rfc2246.txt*

RFC 5246, *The TLS Protocol Version 1.2 - http://www.ietf.org/rfc/rfc5246.txt*

RFC 2313 PKCS #1: *RSA Encryption Versão 1.5 - http://www.ietf.org/rfc/rfc2313.txt*

RFC 2315 PKCS #7: *Cryptographic Message Syntax Versão 1.5 - http://www.ietf.org/rfc/rfc2315.txt*

RFC 3280, *X.509 Internet Public Key Infrastructure - Certificate and Certificate Revocation List (CRL) Profile - http://www.ietf.org/rfc/rfc3280.txt*

RFC 2560, *X.509 Internet Public Key Infrastructure - Online Certificate Status Protocol - OCSP - http://www.ietf.org/rfc/rfc2560.txt*

3 Termos e definições

Para os efeitos desta parte da ABNT NBR 15606, aplicam-se os seguintes termos e definições.

3.1

Java
linguagem de programação orientada a objeto desenvolvida pela Oracle Corporation, onde seus códigos são compilados para um código intermediário (*bytecode*), o qual é executado por uma máquina virtual

3.2

Java Plataform Micro Edition
Java ME
J2ME
tecnologia para desenvolvimento baseada em Java para sistemas e aplicações embarcados em um dispositivos compactos, como celulares, PDAs, controles remotos, dentre outros. Consiste em uma coleção de pacotes Java definidas através do JCP (Java Community Proccess)

3.3

Java Platform
Java Virtual Machine
JVM
componente de *software* que carrega e executa aplicativos escritos utilizando a linguagem de programação Java, interpretando seus *bytecodes* (códigos compilados) e convertendo-os em código executável de máquina

3.4

Java Runtime
ambiente de execução que inclui a JVM e outros componentes e API que servem como base de execução para aplicativos Java

3.5

Java DTV Runtime
Ambiente de execução do Java DTV
ambiente de execução que inclui a JVM e outros componentes e API que servem como base de execução para aplicativos Java DTV

3.6

SWING
uma ferramenta Java para o desenvolvimento de interfaces gráficas de usuário (GUIs). Inclui elementos como menus, barras de ferramentas e caixas de diálogo. O Swing é escrito em Java e asim sendo é independented e plataforma

3.7

RGB
ARGB
ARGB8888
YUV
conjuntos de cores representados em sistemas específicos (chamados "espaços de cores") associados a modelos matemáticos que descrevem modos como as cores são representadas como "t-uplas" de valores numéricos

4 Abreviaturas

Para os efeitos desta parte da ABNT NBR 15606, aplicam-se as seguintes abreviaturas:

AFD	Active Format Description
AID	Application IDentifier
API	Application Programming Interface
APDU	Application Protocol Data Unit
ARIB	Association of Radio Industries and Businesses
ATSC	Advanced Television System Committee
ATR	Answer To Reset
AVR	Automated Voice Response
AWT	Abstract Window Toolkit
CDC	Connected Device Configuration
CRC	Cyclic Redundancy Check
DTD	Document Type Definition
DTMF	Dual-tone Multi-Frequency Signaling
DVB	Digital Video Broadcasting
EDT	Event Dispatch Thread
EIT	Event Information Table
FP	Foundation Profile

GIF	Graphics Interchange Format
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IP	Internet Protocol
IXC	Inter Xlet Communication
JAR	Java Archive
JMF	Java Media Framework
JPEG	Joint Photographic Experts Group
JSR	Java Specification Request
JSSE	Java Secure Socket Extension
LCR	Lista de Certificados Revogados
LWUIT	Lightweight User Interface Toolkit
MHP	Multimedia Home Platform
MIDP	Mobile Information Device Profile
MPEG	Moving Pictures Expert Group
MVC	Model View Controller
NVOD	Near Video On Demand
OCSP	Online Certificate Status Protocol
OTA	Over The Air
PBP	Personal Basis Profile
PIN	Personal Identification Number
PID	Packet Identifier
PKCS	Public-Key Cryptography Standards
PKI	Public Key Infrastructure
PNG	Portable Network Graphics
PSTN	Public Switched Telephone Network
SDT	Service Descriptor Table
SF	Signature File
SHA	Secure Hash Algorithm
SI	Service Information
SVG	Scalable Vectorial Graphics
TLS/SSL	Transport Layer Security/Secure Sockets Layer
UI	User Interface
URL	Uniform Resource Locator
USB	Universal Serial Bus
UTF	Unicode Transformation Format
XML	eXtended Markup Language

5 Requisitos gerais

5.1 Tipos de conteúdo suportados

Os tipos de conteúdos que DEVEM ser suportados estão estabelecidos na ABNT NBR 15606-1.

5.2 Empacotamento, autenticação e autorização de conteúdo e aplicativos

O modelo de conteúdo e aplicativos do Java DTV é designado para definir o empacotamento de conteúdo e aplicativos, para possibilitar a autenticação de ambos e para autorizar aplicativos a acessar funções e API que são de outro modo restritos. Esse mecanismo de pacote também pode ser usado para aplicativos não-Java.

5.3 Gerenciamento de certificados

O gerenciamento de certificados é designado para definir o perfil certificado e a lista de certificados revogados para essa especificação.

5.4 Integração entre o Java TV SI e o SI de baixo nível

A seção de integração entre o Java TV SI e o SI de baixo nível descreve a integração da API de serviços de informação Java TV e a API de serviços de informação de baixo nível.

5.5 Armazenamento persistente de arquivos

O armazenamento persistente de arquivos é disponibilizado para aplicativos autorizados.

5.6 Segurança do canal de retorno

A segurança do canal de retorno especifica a versão TLS segura, o pacote obrigatório de cifras e a API para prover suporte TLS/SSL.

5.7 Tocador de mídia

A seleção de serviços com componentes de serviço conflituosos, por exemplo, seleção simultânea de mais de uma legenda ou áudio em línguas diferentes, deve ser resolvida levando-se em conta que o último componente do serviço especificado é selecionado.

Se um aplicativo excede os recursos do sistema (por exemplo, requisita múltiplos *streams*) a resolução é feita de que a última seleção válida reflete a intenção do aplicativo.

Se um aplicativo está perdendo recursos temporariamente, por exemplo, o decodificador de mídia é usado por um aplicativo diferente, o estado anterior não é restaurado quando o recurso estiver disponível novamente.

5.8 Domínios

O domínio `com.sun.dtv` é protegido. Uma implementação de acordo com esta especificação não pode extender a API definida nesta especificação. Classes e interfaces adicionais só podem ser adicionadas fora destes pacotes.

6 Visão geral da arquitetura

6.1 Considerações gerais

A especificação Java DTV consiste na API Java DTV e na API Java TV acrescentadas à base comum dos componentes do *Java Runtime*, incluindo o *Connected Device Configuration*, o *Foundation Profile* e o *Personal Basis Profile*.

A Figura 2 ilustra a estrutura da API Java DTV.

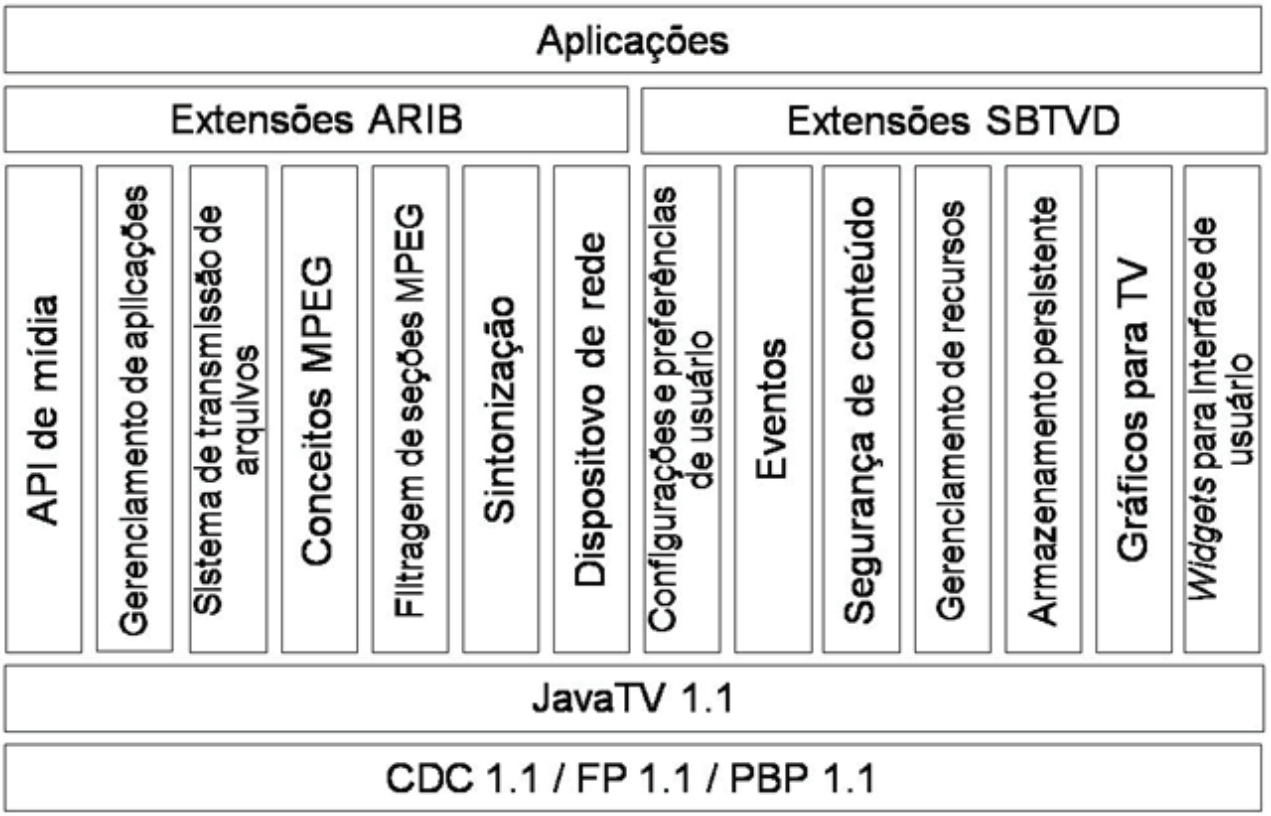


Figura 2 – Estrutura da API Java DTV

A Figura 3 mostra a estrutura dos pacotes da API Java DTV.

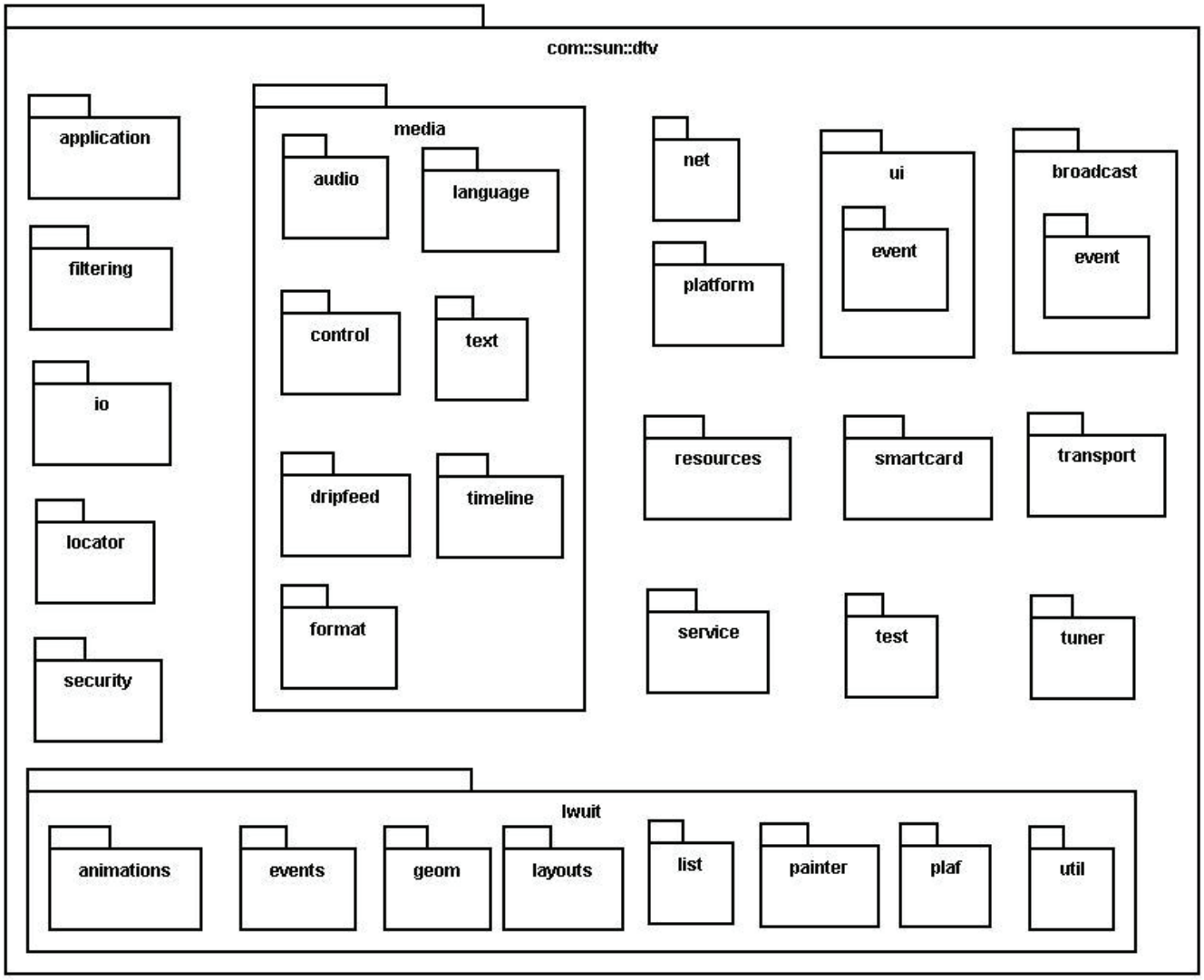


Figura 3 – Estrutura dos pacotes da API Java DTV

6.2 Componentes do Java DTV Runtime

Os componentes do *Java runtime* para o Java DTV são:

- *Java Virtual Machine* como requerida pelo CDC;
- *Connected Device Configuration* 1.1 (JSR 218) ou versão superior;
- *Foundation Profile* 1.1 (JSR 219, incluindo SecOp JSSE 1.0.1 pacote opcional) ou versão superior;
- *Personal Basis Profile* 1.1 (JSR 217) ou versão superior;
- *Java TV* 1.1 (JSR 927) ou versão superior;
- *SATSA - Security and Trust Services API for J2ME* (JSR 177) ou versão superior.

7 Esclarecimentos para as especificações usadas pelo Java DTV

7.1 Java Virtual Machine

A JVM DEVE suportar arquivos de classe Java cujo número da versão está entre 45.3 a 47.

7.2 Connected Device Configuration

7.2.1 Horas e fuso horário

A granularidade do `System.currentTimeMillis()` é menor ou igual a 10 ms.

Os terminais devem manter as horas e data tão precisas quanto possível, inicializando o relógio a partir de entradas de horas em tabelas de SI como as disponíveis na rede.

O fuso horário deve ser retirado de uma entrada de fuso horário em uma tabela de SI ou definido pelo usuário.

Uma chamada para `Object.wait(long timeout)` aguarda pelo menos 10 ms sempre.

7.2.2 java.lang.System

Os aplicativos devem poder usar `System.out` e `System.err`, mas NÃO PODEM usar `System.in`.

Não existe comportamento garantido de `System.gc()`.

7.2.3 java.lang.Runtime

Os métodos `Runtime.traceInstructions()` e `Runtime.traceMethodCalls()` podem ser usados para depuração sem impacto negativo na execução de aplicativos. Essas API não interferem com outras API e a saída não é exibida ao usuário final.

Não há comportamento garantido de `Runtime.gc()`.

7.2.4 java.lang.SecurityManager

Implementadores da especificação Java DTV são aconselhados a usar mecanismos em `java.lang.SecurityManager.checkPackageDefinition` e `java.lang.SecurityManager.checkPackageAccess` para prevenir aplicativos de acessar classes de sistemas de implementação onde a segurança de um terminal esteja violada.

Todos os pacotes de implementação devem ser definidos com nomes de pacotes próprios, o nome de pacote vazio ("") não pode ser usado.

7.2.5 Codificação de caracteres

A codificação UTF-8 DEVE ser suportada.

Latin1 DEVE ser suportado como definido pela ISO 8859-1.

Não há requerimento de que finalizadores sejam executados em qualquer momento definido e confiável. Os aplicativos NÂO PODEM depender de finalizadores para limpeza ou liberação de recursos.

7.2.6 Rede

O canal de transmissão só pode ser usado para receber pacotes de IP. Qualquer tentativa de enviar um pacote de IP pelo canal de transmissão DEVE lançar uma `IOException`. Isso inclui todos os métodos de envio de classes `java.net` que DEVEM lançar um `IOException` quando usados sobre um canal de transmissão.

`java.net.URLClassLoader` DEVE suportar URL definidas pelo Java DTV `URLLocators` para conseguir carregar classes ou recursos do `BroadcastFilesystem`, canal de retorno e outras conexões de rede.

Os `URLLocators` usados para acessar os *streams* de transporte NÂO PODEM modificar a sintonia. Se o `URLLocator` referir-se a um canal que não estiver sintonizado, o acesso deve falhar.

O comportamento da implementação `java.net` em respeito ao controle de conectividade além de limites da plataforma está descrito em `NetworkDevice`.

7.2.7 E/S de arquivos para arquivos de transmissão

Considerações especiais para a operação de entrada/saída de arquivos de transmissão estão documentadas em `com.sun.broadcast.BroadcastFile`. Estes esclarecimentos também se aplicam ao `java.io.File` e operações subseqüentes de entrada/saída utilizando outras classes em `java.io`.

7.2.8 Foundation Profile 1.1.

Não há considerações adicionais para o *Foundation Profile 1.1.*

7.2.9 Personal Basis Profile 1.1.

Os modelos de cores indexado e direto DEVEM ser suportados.

Todas as instâncias de `javax.microedition.xlet.XletContext` DEVEM também implementar `javax.tv.xlet.XletContext`. As interfaces têm métodos comuns e comportamentos prescritos. Isso possibilita que aplicativos usem a instância com um lançamento explícito para `javax.tv.xlet.XletContext` em chamadas para `javax.tv.service.selection.ServiceContextFactory.getServiceContext(xletContext)`.

7.2.10 API Java TV

Essa especificação usa a API Java TV, um pacote opcional do *Java Micro Edition* (Java ME) que fornece controle sobre funcionalidades únicas para receptores de televisão. Algumas das funções que a API Java TV fornece são: acesso ao banco de dados de serviços de informação, seleção de conteúdo, controle de tocador de mídia específico para televisão e acesso a dados que são transmitidos com o sinal de televisão.

As considerações para o uso do Java TV são:

- o pacote `javax.tv.xlet` está obsoleto. Todos os aplicativos Java DTV usam o ciclo de vida do pacote `javax.microedition.xlet.Xlet` definido no PBP;
- o pacote `javax.tv.graphics` está obsoleto. O PBP fornece o container do AWT do `XletContext.getContainer` e suporta as cores via `java.awt.Color`;
- a granularidade mínima para o `javax.tv.util.TVTimer` é menor ou igual a 10 ms, o intervalo de repetição mínimo é menor ou igual a 40 ms. Caso não estejam disponíveis mais temporizadores, uma chamada para `TVTimer.scheduleTimerSpec` pode acarretar `TVTimerScheduleFailedException`;
- para uma instância `javax.media.Player` que estiver apresentando um serviço, as seguintes regras se aplicam:
    - para *streams* de áudio em línguas diferentes presentes na mídia, as preferências do usuário DEVEM ser usadas para selecionar o *stream* correto;
    - para *streams* de textos sobrepostos em línguas diferentes presentes na mídia, as preferências do usuário DEVEM ser usadas para selecionar *stream* correto;
    - o primeiro *stream* para uma rede DEVE ser exibido;
- quando o conteúdo sob controle do tocador houver mudado a lista de controles retornada do `getControls` DEVE ser renovada;
- os seguintes controles têm de ser suportados por um tocador sempre:
    - `javax.tv.media.AWTVideoSizeControl`
    - `com.sun.dtv.media.format.VideoPresentationControl`
    - `com.sun.dtv.media.format.BackgroundVideoPresentationControl`
- o tempo de mídia de um `javax.media.Player` ao tocar conteúdo entregue por um *stream* de transporte é apenas um valor temporal que progride à medida que o *stream* de mídia é tocado. O valor não tem nenhuma relação direta com o conteúdo relógio do sistema em *stream* no receptor;
- para *video drips*, os controles seguintes têm de ser suportados:
    - `javax.tv.media.AWTVideoSizeControl`

      `com.sun.dtv.media.format.BackgroundVideoPresentationControl`

7.2.11 *Security and Trust Services API* (SATSA) para Java ME, Versão 1.0.1

O pacote `javax.microedition.apdu` DEVE ser suportado.

Suporte para o "*(U)SIM Application Toolkit*" no pacote `javax.microedition.apdu` NÃO CONVÉM estar presente.

8 Empacotamento, autenticação e autorização de conteúdo e aplicativos

8.1 Considerações gerais

O pacote de aplicativos é construído para permitir a distribuição compacta de conteúdos e aplicativos, para possibilitar a autenticação do conteúdo ou aplicativo e para autorizar aplicativos a acessar funções e API normalmente negadas ou restritas. A autenticação de conteúdo é usada para autenticar arquivos avulsos entregues separadamente no sistema de arquivos de transmissão. O conteúdo que é tratado como dado pelo aplicativo NÃO PODE ser entregue ao aplicativo se se existir uma autenticação e se a autenticação falhar. Para o ambiente de execução do Java DTV , o `BroadcastFilesystem` DEVE autenticar cada arquivo, se o metadado de autenticação estiver disponível.

Os aplicativos pertencem a duas categorias, assinados e não-assinados. Aplicativos assinados são aplicativos que podem ser autenticados em um conjunto de certificados raiz confiáveis. A segurança dos aplicativos Java DTV consiste em políticas separadas para aplicativos assinados e não-assinados. Aplicativos não-assinados usam uma política padrão que protege o aparelho e o usuário de ações maliciosas. Aplicativos assinados podem requisitar uma política por aplicativo que concede acesso a funções e API adicionais.

8.2 Empacotamento de aplicativos

Cada aplicativo é contido dentro de um ou mais JAR [JAR]. O JAR principal contém os arquivos de aplicativos, arquivos de recurso e um manifesto que descreve o aplicativo e seus requerimentos. Para aplicativos Java DTV JAR adicionais podem ser adicionados ao *classpath* (caminho de classe) usando o atributo `Class-Path` do manifesto.

O dispositivo DEVE suportar dois mecanismos complementares para criação de JAR compactos. No primeiro, cada arquivo dentro de um JAR é comprimido individualmente e a forma comprimida é incluída no JAR como especificado em *[Application Note on the .ZIP file format]*. Em acréscimo, todo o JAR pode ser comprimido usando o GZip. [GZip]. Os JAR comprimidos usando o GZip DEVEM possuir o sufixo `.jar.gz`. Os dispositivos DEVEM reconhecer os JAR Gzipados e suportar descompressão conforme necessária.

Para arquivos JAR assinados, o metadado usado para autenticar o conteúdo é armazenado na raiz do JAR no diretório `/META-INF/` como especificado na especificação do JAR e descrito em 8.3. Os passos para assinar são mostrados no exemplo em 9.

8.3 Autenticação de conteúdo

A autenticação de qualquer conteúdo ou aplicativo é especificada usando metadados que são assinados e podem ser autenticados no dispositivo. O mecanismo é usado para armazenagem hierárquica de arquivos e protege os arquivos, nomeados com caminhos relativos, dentro de uma sub-árvore da hierarquia ao colocar metadados no topo da sub-árvore. Os mecanismos são usados para JAR assinados e adaptados para os casos em que a armazenagem hierárquica de arquivos não é provida pelo formato zip. Para um arquivo dentro de uma hierarquia de arquivos, o metadado da autenticação pode ser encontrado procurando-se pelo diretório `./META-INF/`. A busca DEVE começar no mesmo diretório do arquivo e buscar acima na hierarquia até que o metadado seja encontrado, mas não além da raiz do sistema de arquivos. Evidentemente, uma implementação pode ser otimizada para evitar a busca e usar outros algoritmos para avaliar a informação de autenticação e registrar na *cache* até que o arquivo ou o metadado mude.

A especificação JAR define a sintaxe, codificação e armazenagem do metadado usado para a autenticação. A informação é estocada em três arquivos em um diretório `./META-INF/` relativo a árvore com conteúdo a ser autenticado. Os arquivos de metadados são:

- `MANIFEST.MF` - uma entrada principal e entradas por arquivo fornecem informação sobre cada arquivo, incluindo o *hash* seguro do arquivo;
- *xxx.SF* - uma entrada principal e entradas por arquivo fornecem informação sobre cada arquivo e o *hash* correspondente a cada entrada do manifesto.
- ***xxx***.DSA or ***xxx***.RSA - Uma assinatura PKCS#7 para a entrada ***xxx***.SF. Ambas as assinaturas RSA e DSA podem ser usadas. O prefixo do nome do arquivo deve corresponder entre os arquivos .SF e .RSA

ou .DSA. O *xxx* pode ser escolhido para refletir o assinante. Assinantes múltiplos podem ser usados para autenticar o .SF.

Os arquivos podem ser assinados por um ou mais assinantes, usando diferentes algoritmos e chaves. De acordo com a especificação do JAR cada assinatura produz dois arquivos que ficam localizados no diretório "META-INF". O arquivo de assinatura, XXX.SF, por exemplo, contém uma síntese de informação sobre cada entrada no manifesto para cada arquivo dentro do JAR que deve ser autenticado. O arquivo bloqueador de assinatura, XXX.DSA ou XXX.RSA, por exemplo, contém a assinatura PKCS#7 do arquivo XXX.SF correspondente. Os dois arquivos DEVEM ter o mesmo nome com o sufixo requerido. O arquivo bloqueador de assinatura DEVE conter o certificado de assinatura e todos os certificados intermediários necessários até, mas não incluindo, o certificado raiz. Para cada assinante, cada arquivo do JAR que deve ser autenticado, uma entrada deve ser incluída no arquivo XXX.SF que contém a síntese da entrada no arquivo manifesto. Apenas um dos assinantes disponíveis é usado para autenticar o manifesto e os arquivos.

O arquivo assinatura DEVE conter a seção principal, as seções por entrada e atributos como requerido pela especificação de assinatura do JAR. A seção principal DEVE conter o atributo SHA1-Digest-Manifest e o valor é o *hash* SHA1 de MANIFEST.MF. Para cada arquivo a ser autenticado, o manifesto DEVE incluir uma entrada que inclua o atributo SHA1-Digest com o valor *hash*. O valor *digest* DEVE usar a função de *hash* seguro SHA1 como definido pelo FIPS 180-1. Se o manifesto puder ser autenticado, então o *hash* para cada arquivo com uma entrada pode ser usado para autenticar o arquivo.

O dispositivo DEVE suportar o algoritmo SHA1 para *hashs* seguros. Tanto o algoritmo de criptografia DSA quanto o RSA DEVEM ser suportados para certificados e assinatura de JAR.

Certificados PKI necessários para verificar assinaturas podem ser incluídos com o conteúdo incluindo-os no diretório "META-INF". Os certificados devem estar em formato PKCS#7 e ser nomeados CERT-nn.CRT, onde o primeiro valor de *nn* é 00. Para arquivos de certificado adicionais o valor *nn* é incrementado em 1, permitindo valores de 00 a 99.

8.4 Autenticação de aplicativos

Os arquivos de conteúdo são autenticados usando os mecanismos descritos na especificação do JAR [JAR]. O MANIFEST.MF é autenticado verificando-se o seu *hash* contra o atributo SHA1-Digest-Manifest no arquivo assinatura. Se o atributo não estiver presente, a autenticação falha e a autenticação não é possível. O MANIFEST.MF é autenticado por um único *hash* de todo o seu conteúdo. O arquivo de assinatura é autenticado usando o arquivo bloqueador de assinatura. O arquivo bloqueador de assinatura contém certificados intermediários que são necessários para verificar a assinatura de certificados raiz encontrados no aparelho.

8.5 Política por aplicativo

A política por aplicativo é concedida apenas se o aplicativo e a política forem assinados e autenticados com sucesso. A política por aplicativo DEVE estar incluída no JAR e ser autenticada com os mesmos certificados usados para autenticar os aplicativos componentes do JAR. O arquivo de política por aplicativo DEVE ser nomeado/META-INF/POLICY.PAP. Para aplicativos Java DTV, o mapeamento da política de aplicativo para a política de segurança Java e os detalhes de empacotamento de aplicativo são encontrados em Java DTV Application Packaging.

8.6 Esquema da política por aplicativo

8.6.1 Considerações gerais

A política de aplicativo (conhecida por arquivo de requisição de permissão) define como um aplicativo requisita acesso às funções que são concedidas apenas aos aplicativos confiáveis. A política é formatada usando um esquema XML definido no DTD Application Policy v1.1. Arquivos da política de aplicativos deveriam usar o seguinte DOCTYPE:

```
<!DOCTYPE ApplicationPolicy PUBLIC
           "-//Oracle Corporation.//DTD Java DTV Application Policy 1.1//EN"
           "http://java.sun.com/dtd/dtv-applicationpolicy-1.1.dtd">
```

Um modelo de política pode parecer-se com o seguinte template:

```
<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE ApplicationPolicy PUBLIC
            "-//Oracle Corporation.//DTD Java DTV Application Policy 1.0//EN"
            "http://java.sun.com/dtd/dtv-applicationpolicy-1.1.dtd">

<ApplicationPolicy orgid="0x00000000" appid="0x0000">
    <file value="true"/>
    <applifecycle value="true"/>
    <networkdevice>
        <defaultisp/>
        <phonenumber> +18005551221</phonenumber>
    </networkdevice>
    <serviceselect value="true"/>
    <userproperty read="true" write="false"/>
    <network>
        <host action="connect">host1
        </host>
    </network>
    <dripfeed value="false"/>
    <ixcaccess value="true"/>
    <persistentfilecredential>
        <grantoridentifier id="x0001" />
        <expirationdate date="01/01/2008"/>
        <filename read="true" write="false">3/32/com/sun/example/abc
        </filename>
        <signature>123123123123123123123123123123123</signature>
        <certchainfileid> bigca</certchainfileid>
    </persistentfilecredential>
</ApplicationPolicy>
```

A Tabela 1 descreve as funções estendidas que podem ser permitidas pela política de aplicativo.

Tabela 1 – Funções estendidas que podem ser permitidas pela política de aplicativo

Elemento	Descrição	Não-assinado	Assinatura padrão
file	Acesso ao sistema de arquivos	Sem acesso	Verdadeiro
applifecycle	Controle de aplicativos não inicializados por esse aplicativo	Falso	Falso
networkdevice	Acesso ao dispositivo de rede	Falso	Falso
dripfeed	Acesso a *dripfeed controls*	Falso	Verdadeiro
scarceresource	Acesso a recursos escassos	Nenhum	Nenhum
tuning	Acesso a funções de sintonização	Falso	Falso
smartcardaccess	Acesso às API *smartcard*	Falso	Falso
serviceselect	Acesso a um serviço selecionado	Falso	Verdadeiro
userproperty	Acesso as propriedades de usuário	Ler: falso Escrever: falso Permitido sempre: língua do usuário, classificação indicativa, Tamanho padrão de fonte, código do país	Ler: verdadeiro Escrever: falso
network	Acesso aos hospedeiros nomeados de rede	Nenhum	Nenhum
ixcaccess	Acesso as funções de comunicação inter-aplicativos	Falso	Falso
persistentfilecredential	Credencias provendo acesso a arquivos de outro aplicativo	Nenhum	Nenhum

8.6.2 Acesso a arquivos

O elemento `file` requisita acesso a arquivos dentro do sistema de arquivos do dispositivo sujeito as proteções contra acesso recebidas pelos arquivos pelos aplicativos que os possuem. A estrutura do sistema de arquivos inclui áreas dedicadas a cada organização e cada aplicativo dentro da organização.

As regras de acesso a arquivos são:

- aplicativos não-assinados NÃO PODEM receber acesso ao sistema de arquivos;
- todos os aplicativos assinados DEVEM receber acesso irrestrito ao seus próprios arquivos;
- a elemento acesso aos arquivos é usado para possibilitar acesso dos aplicativos assinados a arquivos de outros aplicativos usando os mecanismos de controle de acesso das plataformas;
- se o valor for `true`, o acesso DEVE ser concedido;
- se o valor for *`false`*, o acesso NÃO PODE ser concedido.

Para o ambiente de execução do Java DTV, o aplicativo define os modos de acesso à plataforma para arquivos

que possui usando com.sun.dtv.io.FileProperties. O acesso de aplicativos a arquivos é via `java.io.File` e classes relacionadas.

8.6.3 Acesso ao ciclo de vida dos aplicativos

O elemento `applifecycle` requisita acesso para controlar outros aplicativos dentro do serviço atual conforme segue:

- aplicativos não-assinados DEVEM poder inicializar qualquer aplicativo listado como visível no serviço atual;
- todo aplicativo DEVE poder controlar o lifecycle, início, parar, pause, resumo de outros aplicativos que ele tiver iniciado;
- os aplicativos NÃO PODEM ser capazes de controlar o lifecycle de aplicativos que eles não inicializaram;
- aplicativos assinados recebem acesso para controlar aplicativos que eles não inicializaram;
- se o valor for `true`, o acesso DEVE ser concedido;
- se o valor for *`false`*, o acesso NÃO PODE ser concedido.

8.6.4 Acesso ao dispositivo de rede

O elemento `networkdevice` requisita acesso ao dispositivo de rede que provê acesso ao aplicativo a serviços fora do dispositivo. O acesso ao dispositivo de rede é restrito conforme segue:

- aplicativos não assinados NÃO PODEM ter acesso permitido ao dispositivo de rede;
- aplicativos assinados DEVEM ter acesso permitido ao dispositivo de rede apenas quando requisitado e como definido pelos parâmetros;
- se **`defaultisp`** estiver presente, o aplicativo DEVE ter acesso permitido ao dispositivo de rede configurado pelo dispositivo;
- para cada padrão de número de telefone provido o aplicativo DEVE ter acesso permitido ao dispositivo de rede usando o número de telefone. Qualquer número de telefone que comece com um dos padrões fornecidos é permitido. A linha vazia pode ser usada para permitir acesso a números de telefone arbitrários.

Para o ambiente de execução do Java DTV, o dispositivo de rede é acessado usando `com.sun.dtv.net.NetworkDevice`.

8.6.5 Acesso a *dripfeeds*

O elemento `dripfeed` requisita acesso para usar controles *dripfeed*. O acesso aos *dripfeeds* é restrito conforme segue:

- aplicativos não-assinados NÃO PODEM ser capazes de usar controles *dripfeed*;
- aplicativos assinados recebem acesso ao *dripfeed*:
  - se o valor for **`true`**, o acesso DEVE ser concedido;
  - se o valor for ***`false`***, o acesso NÃO PODE ser concedido.

Para o ambiente de execução do Java DTV, o *dripfeed* é acessado usando `com.sun.dtv.media.dripfeed.DripFeedControl`.

8.6.6 Acesso a sintonização

O elemento `tuning` requisita acesso às funções de sintonização. O acesso a sintonização é restrito conforme segue:

- Aplicativos não-assinados NÃO PODEM ser capazes de usar o sintonizador;
- Aplicativos assinados recebem acesso ao sintonizador:
  - Se o valor for ***`true`***, o acesso DEVE ser concedido;
  - Se o valor for ***`false`***, o acesso NÃO PODE ser concedido.

Para o ambiente de execução do Java DTV , o sintonizador é acessado usando `com.sun.dtv.tuner.Tuner`.

8.6.7 Recursos escassos

O elemento `scarceresource` requisita acesso a recursos nomeados com as ações especificadas. O acesso a recursos escassos é restrito conforme segue:

- aplicativos não-assinados NÃO PODEM receber acesso a quaisquer recursos escassos;
- aplicativos assinados DEVEM receber acesso aos recursos escassos listados com as ações requisitadas. As ações DEVEM ser formatadas de acordo com as conveções de `com.sun.dtv.resources.ScarceResourcePermission`;

Para o ambiente de execução do Java DTV , o acesso a recursos escassos, incluindo o aparelho de rede e o(s) sintonizador(es), é normal via elementos `networkdevice` e `tuning`. Se o aplicativo precisa ser capaz de forçar recursos a serem liberados, ele DEVE requisitar acesso ao recurso com um elemento `scarceresource` com o nome do recurso e a ação "`force`".

8.6.8 Acesso ao *smartcard*

O elemento `smartcard` requisita acesso às funções APDU para *smartcard* . O acesso ao *smartcard* tem as seguintes restrições:

- aplicativos não-assinados NÃO PODEM ser capazes de usar o *smartcard*.
- aplicativos assinados recebem acesso ao *smartcard*:
    - se o elemento `smartcardaccess` estiver presente.

Para o ambiente de execução do Java DTV , o *smartcard* é acessado usando *Security and Trust Services API for J2ME (JSR 177)*.

8.6.9 Acesso a seleção de serviço

As requisições de acesso do `serviceselect` às funções da seleção de serviço são definidas para permitir alguns aplicativos selecionar o próximo serviço a ser apresentado. O acesso a seleção de serviço é restrita conforme segue:

- aplicativos não-assinados NÃO PODEM ser capazes de selecionar um novo serviço;
- aplicativos assinados recebem acesso a seleção de serviço:
    - Se o valor for `true`, o acesso DEVE ser concedido;
    - Se o valor for *`false`*, o acesso NÃO PODE ser concedido;
    - Se o elemento *serviceselect* for omitido, o acesso DEVE ser concedido.

O acesso ao serviço é via *locator*, a política normal para acesso através do *locator* DEVE ser aplicada.

Para o ambiente de execução do Java DTV , os serviços são selecionados usando `java.tv.service.selection.ServiceContext` da API Java TV.

8.6.10 Acesso a propriedades do usuário

O elemento `userproperty` requisita acesso a propriedades do usuário que são definidas para permitir aplicativos a se personalizarem. O acesso a propriedades do usuário é restrito conforme segue:

- todos os aplicativos recebem acesso de leitura para:
    - língua do usuário;
    - classificação indicativa;
    - tamanho padrão do fonte;
    - código do país;
- aplicativos não-assinados NÃO PODEM receber acesso a quaisquer outras propriedades;
- aplicativos assinados recebem acesso a:
    - acesso de leitura a todas as propriedades;
        - se o valor de `read` for `true`, o acesso de leitura DEVE ser concedido;

- se o valor de `read` for *`false`*, o acesso de leitura NÃO PODE ser concedido;
- se os elementos `userproperty` ou `read` estiverem omissos, o acesso de leitura NÃO PODE ser concedido;

- acesso de escrita a todas as propriedades;
  - se o valor de `write` for `true`, o acesso de escrita DEVE ser concedido;
  - se o valor de `write` for *`false`*, o acesso de escrita NÃO PODE ser concedido;
  - se os elementos `userproperty` ou `write` estiverem omissos, o acesso de escrita NÃO PODE ser concedido.

Para o ambiente de execução do Java DTV , o acesso a propriedades do usuário é via pacote `com.sun.dtv.platform`.

8.6.11 Acesso a rede

O elemento `network` requisita acesso as conexões da rede a hospedeiros via dispositivo de rede. O acesso a hospedeiros é restrito conforme segue:

- aplicativos não-assinados NÃO PODEM receber acesso a quaisquer hospedeiros;
- aplicativos assinados DEVEM receber acesso apenas aos hospedeiros listados com as ações requisitadas. As ações DEVEM ser formatadas de acordo com as conveções de `java.net.SocketPermission`.

Para o ambiente de execução do Java DTV, acesso a rede é conseguido usando as API de `com.sun.dtv.net.NetworkDevice`.

8.6.12 Acesso a comunicação inter-aplicativos

O elemento `ixcaccess` requisita acesso as funções de comunicação inter-aplicativos. O acesso ao `ixc` é restrito conforme segue:

- aplicativos não-assinados NÃO PODEM receber acesso;
- aplicativos assinados DEVEM receber acesso apenas se requisitado.

Para o ambiente de execução do Java DTV, acesso inter-aplicativo é conseguido usando as API de `javax.microedition.xlet.ixc`, usando o domínio definido para aplicativos DTV assinados.

8.6.13 Acesso a credencial de persistência de dados

O elemento `persistentfilecredential` requisita a um aplicativo (doador) acesso aos seus próprios recursos, comumente arquivos, e concede acesso a outro aplicativo específico (donatário). A autorização assinada para conceder acesso é criada e assinada pelo doador, mas é entregue ao dispositivo pelo donatário embutindo a informação de acesso ao arquivo na política por aplicativo.

A Tabela 2 descreve os elementos para a informação de acesso a arquivos como incluídas na política por aplicativo.

A Tabela 2 - Elementos para a informação de acesso a arquivos como incluídas na política por aplicativo

Elemento	Descrição
grantoridentifier	A organização de identidade do proprietário do recurso
expirationdate	O acesso pode ser concedido até a data de expiração, mas não além. A data deve estar formatada como *mm/dd/yyyy* onde *mm, dd, yyyy* são números decimais representando o mês, dia e ano; tanto mês quanto dia começam do 1
filename	O nome do caminho relativo a raiz persistente com a seguinte sintaxe: `filename = persistentfilename` `| persistentpath "/" persistentfilename` `| persistentpath "/" "*"` `| persistentpath "/" "-"` `persistentpath = name` `| persistentpath "/" name` `persistentfilename = name |` `name "." suffix` `name = 8*fchar ; comprimento é limitado a 8 caracteres` `suffix = 3*fchar ; comprimento é limitado a 3 caracteres` `fchar = "a".."z"	"A".."Z"	"0".."9"	"_"` onde: "/" é o separador do caminho. "*" usado no fim indica todos os arquivos contidos no diretório. "-" usado no fim indica todos os arquivos no diretório e subdiretórios existentes recursivamente no momento em que a requisição é analisada Os atributos dos nomes dos arquivos são "`read`" e "`write`", os valores DEVEM ser `true` ou *`false`*
signature	A informação de acesso ao arquivo é concatenada como definido abaixo. O *stream* binário é assinado pelo doador e então codificado pela Base64
certchainfileid	O identificador usado para localizar certificados no diretório */META-INF* é usado para autenticar a assinatura

A Tabela 3 documenta a ordem em que a informação de acesso a arquivos é concatenada para ser assinada.

Tabela 3 – Ordem de concatenação de informação de acesso a arquivos para ser assinada

Campos	Codificação binária
grantee.organization_id	O identificador da organização do donatário em 32 bits
grantee.application_id	O identificador do aplicativo do donatário em 16 bits
grantor.organization_id	O identificador da organização do doador em 32 bits
Data de expiração	Os caracteres em ASCII. 10 caracteres, por exemplo: "01/01/2008"
filenames e *actions* para cada (*filename*) { `true` ou *`false`* para leitura `true` ou *`false`* para escrita do *filename* }	Na mesma ordem em que os elementos aparecem no XML 4 caracteres para `true`; 5 caracteres para *`false`* o nome do arquivo e todos os caracteres em ASCII

O acesso aos arquivos listados é concedido apenas se:

- a política do aplicativo esteja assinada, autenticada E;
- a informação de acesso a arquivos estiver concatenada como definido acima e o *hash* resultante seja igual ao *hash* na assinatura E;
- a assinatura seja autenticada usando os certificados localizados usando a informação `certchainfileid`.

9 Exemplo de aplicativo Java DTV assinado

Supondo que um simples aplicativo "Hello" Java DTV:

```
package hello;

import javax.microedition.xlet.*;

public class HelloXlet implements Xlet {
    XletContext ctx;

    public void destroyXlet(boolean arg0) throws XletStateChangeException {
        // Empty
    }

    public void initXlet(XletContext ctx) throws XletStateChangeException {
        this.ctx = ctx;
    }

    public void pauseXlet() {
        // Empty
    }

    public void startXlet() throws XletStateChangeException {
        System.out.println("Hello Java DTV.");
        ctx.notifyDestroyed();
    }}
```

Usar um MANIFEST.MF simples que identifique a classe de entrada principal.

```
Main-Class: hello.HelloXlet
```

Criar o arquivo JAR usando a ferramenta regular *jar*. Assinar com a ferramenta *jarsigner* usando o certificado no arquivo-chave com o pseudônimo *mykey*.

```
% jar -cfm hello.jar manifest.mf -C classes hello
% jarsigner hello.jar mykey -storepass xxx
```

O hello.jar contém, agora, os arquivos de classes e autenticação.

```
META-INF/MANIFEST.MF
META-INF/MYKEY.SF
META-INF/MYKEY.DSA
META-INF/
hello/
hello/HelloXlet.class
```

O MANIFEST.MF gerado contém os atributos principais e entradas para cada arquivo.

```
Manifest-Version: 1.0
Created-By: 1.5.0_16-133
Main-Class:  hello.HelloXlet

Name: hello/HelloXlet.class
SHA1-Digest: IAhtIdnN9ITtPFY2nIb8NHjXYC0=
```

O arquivo *mykey.sf* contém:

```
Signature-Version: 1.0
Created-By: 1.5.0_16
SHA1-Digest-Manifest-Main-Attributes: 80psXuLFfIvDp8Uj/StiEb4uVEs=
SHA1-Digest-Manifest: gGhmIOAW3WJ2cmhf+XsPVg8F+uk=

Name: hello/HelloXlet.class
SHA1-Digest: BgKfgpXw6c/geXKxU9SyI3h/VUQ=
```

10 DTD Application Policy v1.1

```
<?xml version='1.0' encoding='UTF-8' ?>
<!--
Este é o XML DTD para a Política de Aplicativos Java DTV.
Todas as políticas de aplicativo devem incluir um DOCTYPE da seguinte forma:

    <!DOCTYPE ApplicationPolicy PUBLIC
            "-//Orcale Corporation.//DTD Java DTV Application Policy 1.0//EN"
            "http://java.sun.com/dtd/dtv-applicationpolicy-1.1.dtd"
```

```
    >
-->

<!ELEMENT ApplicationPolicy
    (file?,applifecycle?,networkdevice?,scarceresource*,tuning?,
    serviceselect?,userproperty?,network?, dripfeed?,
    ixcaccess?, persistentfilecredential*,
    smartcardaccess?)>

<!ATTLIST ApplicationPolicy
     orgid CDATA #REQUIRED
     appid CDATA #REQUIRED
>
<!ELEMENT file EMPTY>
<!ATTLIST file
    value (true|false) "true"
>

<!ELEMENT applifecycle EMPTY>
<!ATTLIST applifecycle
    value (true|false) "true"
>

<!ELEMENT networkdevice (defaultisp?,phonenumber*)>
<!ELEMENT defaultisp EMPTY>
<!ELEMENT phonenumber (#PCDATA)>

<!ELEMENT dripfeed EMPTY>
<!ATTLIST dripfeed
    value (true|false) "true"
>

<!ELEMENT scarceresource (resource,action)>
<!ELEMENT resource (#PCDATA)>
<!ELEMENT action (#PCDATA)>

<!ELEMENT tuning EMPTY>
<!ATTLIST tuning
    value (true|false) "true"
>

<!ELEMENT serviceselect EMPTY>
<!ATTLIST serviceselect
    value (true|false) "true"
>

<!ELEMENT userproperty EMPTY>
<!ATTLIST userproperty
    write (true|false) "false"
    read (true|false) "true"
>
<!ELEMENT network (host)+>
<!ELEMENT host (#PCDATA)>
<!ATTLIST host
    action CDATA #REQUIRED
```

```
>

<!ELEMENT ixcaccess EMPTY>
<!ATTLIST ixcaccess
    value (true|false) "true"
>

<!ELEMENT                                                    persistentfilecredential
(grantoridentifier,expirationdate,filename+,signature,certchainfileid)>

<!ELEMENT grantoridentifier EMPTY>
<!ATTLIST grantoridentifier
    id CDATA #REQUIRED
>

<!ELEMENT expirationdate EMPTY>
<!ATTLIST expirationdate
    date CDATA #REQUIRED
>

<!ELEMENT filename (#PCDATA)>
<!ATTLIST filename
    write (true|false) "true"
    read (true|false) "true"
>

<!ELEMENT signature (#PCDATA)>
<!ELEMENT certchainfileid (#PCDATA)>

<!ELEMENT smartcardaccess EMPTY>
```

11 Modelos de cor Java DTV

11.1 Considerações gerais

As API de gráficos do Java DTV são providas pelo PBP no pacote `java.awt`. O `GraphicsEnvironment` prove uma enumeração dos `GraphicsDevice`s disponíveis. Para cada dispositivo, o `GraphicsConfiguration` fornece informação sobre os limites da superfície de exibição, os modelos de cor e as capacidades de imagem. Os modelos de cor predefinidos são `IndexColorModel` e `DirectColorModel`.

O modelo `java.awt.Color` define o espectro de valores para valores alfa, vermelho, verde e azul variando de 0.0 – 1.0 ou 0 – 255. O espaço para cores padrão é sRGB, ver [sRGB].

Os dispositivos mapeiam esses valores nas profundidades disponíveis nos dispositivos para apresentação.

11.2 Modelo de cores indexado

O `IndexColorModel` DEVE ser suportado com uma tabela de cores predefinida, contendo as entradas como descritas na Tabela 4 que permite variações selecionadas para cores em opaco, parcialmente transparente, completamente transparentes e escalas de cinza opacas.

Tabela 4 – Entradas para tabela de cores

Transparência	Valores vermelho	Valores verde	Valores azul
Opaco; alfa = 255	0, 63, 127, 191, 255	0, 31, 63, 95, 127, 159, 191, 223, 255	0, 127, 255
30; alfa = 179	0, 85, 170, 255	0, 51, 102, 153, 204, 255	0, 255
Transparente; alfa = 0	Não disponível	Não disponível	Não disponível

Opaco; alfa = 255	42, 85, 170, 212	Idêntico a vermelho	Idêntico a vermelho

11.3 Modelo de cores direto

O `DirectColorModel` DEVE ser suportado com pelo menos 4 bits por pixel para alfa, vermelho, verde e azul.

O valores alfa DEVEM suportar 16 níveis de transparência de completamente transparente a completamente opaco. Os valores vermelho, verde e azul DEVEM suportar 16 níveis de nada a completamente saturado.

12 Gerenciamento de certificados

12.1 Considerações gerais

O gerenciamento de certificados define o perfil certificado, o gerenciamento de certificados raiz e certificados teste e a distribuição da lista de certificados revogados (LCR).

12.2 Perfil certificado

Todo certificado usado no contexto desta especificação DEVE seguir as definições da RFC 3280.

12.3 Certificados raiz

Todo dispositivo conforme com esta especificação deve suportar o armazenamento de certificados raiz em memória não volátil.

12.4 Certificados teste

O dispositivo DEVE suportar um mecanismo para instalar certificados raiz para o fim de testar o dispositivo. Certificados raiz de teste DEVEM expirar após o período de teste e todos os objetos assinados com um certificado raiz de teste NÃO PODEM ser usados com outros fins que não de teste.

12.5 Gerenciamento de certificados raiz

O gerenciamento de certificados raiz é particular ao contexto do sistema e envolve tanto a definição de um repositório para tal, quanto o mecanismo para sua manutenção.

12.6 Lista de certificados revogados (LCR)

Toda LCR usada no contexto desta especificação DEVE seguir o perfil LCR e as políticas definidas na RFC 3280.

12.7 Distribuição da lista de certificados revogados

A distribuição da lista de certificados revogados é feita de tal maneira que estas são distribuídas junto com as aplicações assinadas.

12.8 Retenção da LCR

LCR devem ser mantidas em memória não-volátil no dispositivo.

12.9 Processamento de LCR

Durante a validação de uma cadeia de certificados, a LCR de cada certificado de autoridade no caminho do certificado deve ser verificada.

12.10 Protocolo de estado *online* do certificado

Dispositivos que suportam um canal de retorno CONVÊM suportar o protocolo OCSP definido na RFC 2560 para determinar o status atual de um certificado sem requerer o armazenamento das LCR no dispositivo.

13 Integração entre Java TV SI e SI de baixo nível

13.1 Considerações gerais

Em sistemas DTV típicos um conjunto específico de API é definido para encapsular o banco de dados de SI de baixo nível, o qual é usado no padrão específico. Essas API definem classes e interfaces para representar elementos das tabelas SI como disponíveis na rede. As API Java TV SI descrevem um modelo abstrato para manusear serviços de informação. Se essa API é usado com um banco de dados SI concreto, um mapeamento das classes e métodos para os campos concretos do banco de dados SI precisa ser definido, o que garante que ambas as API de acesso a SI trabalhem juntas.

13.2 Integração do SI do Java TV e da API de SI de baixo nível

Enquanto a API de SI independente de protocolo como definido no JavaTV oferece acesso a serviços de informação de alto nível, as API dependentes de transporte são usados para obter informações adicionais dependentes de protocolo sobre os elementos na rede. Para alcançar uma coesão entre essas duas API, é requerido que os objetos representando entidades de informação SI, que estão disponíveis em ambos os conjuntos de API, estejam implementando as interfaces correspondentes de ambos os conjuntos de API.

As interfaces de ambas API devem ser implementadas de acordo com a Tabela 5.

Tabela 5 – API de Integração entre o Java TV SI e o SI de baixo nível

Interface JavaTV SI	Interface depentede de protocolo
`javax.tv.service.transport.Network`	`SINetwork`
`javax.tv.service.transport.Bouquet`	`SIBouquet`
`javax.tv.service.transport.TransportStream`	`SITransportStreamNIT`
`javax.tv.service.guide.ProgramEvent`	`SIEvent`
`javax.tv.service.navigation.ServiceDetails`	`SIService`

13.3 Mapeamento da API Java TV SI no banco de dados SI

13.3.1 Considerações gerais

Para todos os métodos `getter`, que retornam um nome ou descritor correspondente de um campo de nomes multilíngüe da tabela SI subjacente, as seguintes regras se aplicam:

- se a língua de um campo descritor de nomes multilíngüe na tabela SI corresponder a língua retornada pelo `javax.tv.service.SIManager.getPreferredLanguage`, o nome do campo descritor é retornado;
- em qualquer outro caso, o nome retornado é uma escolha dependente de implementação do valores disponíveis nos descritores.

13.3.2 javax.tv.service.transport.Bouquet

Essa interface representa um *buquê* de SI. Instâncias de `Bouquet` devem também implementar a interface do `CAIdentification`.
Um localizador retornado pelo método `getLocator` é um objeto `javax.tv.locator.Locator` dependente de implementação, onde nenhuma forma externa padronizada é requerida.

13.3.3 javax.tv.service.transport.Network

Essa interface representa uma rede SI. Um localizador retornado pelo método `getLocator` é um objeto `javax.tv.locator.Locator` dependente de implementação, onde nenhuma forma externa padronizada é requerida.

13.3.4 javax.tv.service.transport.Transport

Instâncias de objetos que implementem a interface `Transport` devem também implementar as interfaces `BouquetCollection` e `NetworkCollection`.

13.3.5 javax.tv.service.transport.TransportStream

Para sistemas SI onde nenhuma descrição de *stream* de transporte é definida, uma *string* vazia é retornada pelo método `getDescription`.

13.3.6 javax.tv.service.guide.ProgramEvent

Objetos estão usando essa interface para eventos de SI e devem também implementar a interface `CAIdentification`. Os valores de return para `getStartTime`, `getDuration` e `getEndTime` são tomados de ou calculados a partir dos valores de tempo correspondentes da entrada do evento no corpo da tabela EIT ou de sua tabela SI equivalente. O método `ProgramEventDescription#getProgramEventDescription()` deveria retornar uma descrição textual para um dado evento sempre que estiver disponível. Os caracteres código 0x8a e 0xe08a são substituídos por um nova linha Java (\n).

13.3.7 javax.tv.service.Service

A interface serviço representa os serviços disponíveis na rede. Objetos implementando a interface `Service` podem também implementar a interface `ServiceNumber`.

13.3.8 javax.tv.service.ServiceType

O mapeamento de tipos de serviço para Java TV service types é definido na Tabela 6.

Tabela 6 – Mapeamento de tipos de serviço para tipos de serviço Java TV

Service_type	Descrição	Java TV service type
0x00	Reservado para uso futuro	UNKNOWN
0x01	Serviço de televisão digital	DIGITAL_TV
0x02	Serviço de áudio digital	DIGITAL_RADIO
0x03	Serviço de teletexto	DATA_BROADCAST
0x04	Serviço de referência NVOD	NVOD_REFERENCE
0x05	Serviço *time-shifted* NVOD	NVOD_TIME_SHIFTED
0x06	Serviço de mosaico	DIGITAL_TV
0x07 – 0x09	Reservado para uso futuro	UNKNOWN
0x0A	Codificação avançada para serviço de rádio digital	DIGITAL_RADIO
0x0B	Codificação avançada para serviço de mosaico	DIGITAL_TV
0x0C	Serviço de transmissão de dados	DATA_BROADCAST
0x0D	Reservado para interface de uso comum (ver EN 50221)	UNKNOWN
0x0E	RCS *map* (ver EN 301 790)	UNKNOWN
0x0F	RCS FLS (ver EN 301 790)	UNKNOWN
0x10	Serviço DVB MHP	DATA_APPLICATION
0x11	Serviço de televisão digital MPEG2 HD	DIGITAL_TV
0x12 – 0x15	Reservado para uso futuro	UNKNOWN
0x16	Codificação avançada de serviço de televisão digital SD	DIGITAL_TV
0x17	Codificação avançada de serviço de televisão digital SD NVOD *time-shifted*	NVOD_TIME_SHIFTED
0x18	Codificação avançada de serviço de referência em televisão digital NVOD SD	NVOD_REFERENCE
0x19	Codificação avançada de serviço de televisão digital HD	NVOD_TIME_SHIFTED
0x1A	Codificação avançada de serviço de televisão digital NVOD HD *time-shifted*	NVOD_TIME_SHIFTED
0x1B	Codificação avançada de serviço de referência NVOD HD	NVOD_REFERENCE
0x1C – 0x7F	Reservado para uso futuro	UNKNOWN
0x80 – 0xA0	Definido pelo provedor do serviço	–
0xA1	Serviço de vídeo especial	DIGITAL_TV
0xA2	Serviço de áudio especial	DIGITAL_RADIO
0xA3	Serviço de dados especial	DATA_BROADCAST
0xA4	Serviço de engenharia	UNKNOWN
0xA5	Serviço de vídeo promocional	DIGITAL_TV
0xA6	Serviço de áudio promocional	DIGITAL_RADIO
0xA7	Serviço de dados promocional	DATA_BROADCAST
0xA8	Serviço de dados para armazenagem antecipado	DATA_BROADCAST
0xA9	Serviço de dados exclusivos para armazenagem	DATA_BROADCAST
0xAA	Lista de serviços de marcadores	DATA_BROADCAST
0xAB	Serviço de servidor simultâneo	UNKNOWN
0xAC	Serviço de arquivos independentes	DATA_BROADCAST
0xAD – 0xBF	Não definido (variação definida pela organização de padronização)	UNKNOWN
0xC0	Serviço de dados	DATA_BROADCAST
0xC1 – 0xFF	Não definido	UNKNOWN

13.3.9 javax.tv.service.navigation.ServiceDetails

Esta interface representa informação representando os detalhes de um serviço proveniente das tabelas EIT e SDT ou de suas tabelas SI equivalentes.

Instâncias de `ServiceDetails` devem também implementar a interface `CAIdentification` e não podem implementar a interface `ServiceNumber`.

Em sistemas SI, onde não há descrição de serviço disponível nas tabelas SI subjacentes, o método `retrieveServiceDescription` deve falhar com `SIRequestFailureType(DATA_UNAVAILABLE)`.

13.3.10 javax.tv.service.navigation.ServiceComponent

A interface `ServiceComponent` é mapeada para a EIT e representa informação contida nos descritores dos componentes, descritores de componentes multilíngües ou descritores de transmissão de dados.

13.3.11 javax.tv.service.SIElement

Esta interface é implementada por objetos implementando as interfaces `ServiceDetails`, `ServiceComponent`, `ProgramEvent`, `Network`, `Bouquet` e `TransportStream`.

13.3.12 javax.tv.service.SIManager

Os valores de return de `getSupportedDimensions` e `getRatingDimension` dependem do sistema de classificação indicativa do esquema SI específico.

Considerações:

- o método `retrieveSIElement` deve retornar um objeto implementando a interface `ServiceDetails`, quando chamado com um localizador que aponte um serviço. Localizadores para eventos de programas falham com `SIRequestFailureType(INSUFFICIENT_RESOURCES)`
- o objeto retornado pelo método `getTransports` deve implementar a interface `javax.tv.service.transport.Transport`
- filtro de serviços é suportado com `javax.tv.service.navigation.SIElementFilter`.
- todas as chamadas para o método `retrieveProgramEvent` devem falhar com um `SIRequestFailureType(INSUFFICIENT_RESOURCES)`

13.3.13 javax.tv.service.navigation.SIElementFilter

O `SIElementFilter` deve ser suportado para os objetos `TransportStream` e `Network`. Se não estiver disponível para outros objetos `SIElement`, o construtor pode lançar um `FilterNotSupportedException`. Em sistemas em que nenhum acesso condicional é usado, o método `getCASystemIDs` CONVÉM retornar um conjunto vazio e o método isFree() CONVÉM retornar sempre *true*.

13.3.13.1 javax.tv.service.navigation.CAIdentification

Esta interface é implementada por objetos implementando as interfaces `ServiceDetails`, `ProgramEvent` ou `Bouquet`.

13.3.14 javax.tv.service.navigation.ServiceProviderInformation

Esta interface é implementada por objetos implementando a interface `ServiceDetails`.

13.3.15 javax.tv.service.guide.ContentRatingAdvisory

O `ContentRatingAdvisory` depende do esquema de classificação indicativa do banco de dados SI concreto.

13.3.16 javax.tv.service.RatingDimension

Esta interface é mapeada no descritor de classificação indicativa como definido no esquema de classificação indicativa da rede SI.

13.3.17 javax.tv.service.navigation.StreamType

O mapeamento de `stream_content` e `component_type` é definido na Tabela 7. Se um componente não tem um descritor de componente associado, mas um descritor de transmissão de dados, o tipo de *stream* `DATA` deve ser usado.

Tabela 7 – Mapeamento de *stream_content* e *component_type*

stream_content	*component_type*	*JavaTV stream type*
0x01	0x00…0xff	`VIDEO`
0x02	0x00…0xff	`AUDIO`
0x05	0x00…0xff	`VIDEO`
0x06	0x00…0xff	`AUDIO`

14 Armazenamento persistente de arquivos

Aplicativos Java DTV têm acesso ao sistema de dados acessível através das API `java.io.File`, `java.net.URL` com o protocolo "`file:`" e `javax.microedition.io`, se presentes.

Observa-se que a persistência de um arquivo é garantida sobre o tempo de execução do aplicativo que criou o arquivo. Quando o aplicativo termina, o arquivo pode estar sujeito a remoção, isto é, no caso de carência de espaço de armazenamento. Também significa que um dado arquivo NÃO PODE ser removido enquanto o aplicativo que o criou estiver sendo executado, estejam o arquivo e o seu correspondente manuseador abertos ainda ou não.

A hierarquia de arquivos é estruturada para simplificar o acesso a arquivos de aplicativos conforme segue:

- Uma única raiz do sistema de arquivos deve ser definida para todos os aplicativos pela implementação. Para o ambiente de execução do Java DTV, a raiz é acessível via propriedade "`com.sun.dtv.persistent.root`" disponível em `java.lang.System.getProperty`.
- Acesso a quaisquer arquivos nesse diretório raiz DEVE resultar em uma violação de segurança, no caso do ambiente de execução do Java DTV em uma `java.lang.SecurityException`.
- Para cada organização para a qual existe um aplicativo, um subdiretório do diretório raiz é definido usando o identificador da organização formatado como uma *string* hexadecimal usando apenas números e letras em caixa baixa ('0'..'9','a'..'f'). A *string* NÃO PODE ter nenhum caractere '0' à esquerda.
- Para cada aplicativo um subdiretório do diretório da organização é definido usando o identificador do aplicativo formatado como uma *string* hexadecimal usando apenas números e letras em caixa baixa ('0'..'9','a'..'f'). A *string* NÃO PODE ter nenhum caracter '0' à esquerda.
- Para cada aplicativo o "diretório atual" é definido para o diretório do aplicativo quando o aplicativo é inicializado. O diretório atual é mudado apenas pelo aplicativo.
- Os diretórios são criados automaticamente, como segue:
    - Para a organização abaixo da raiz e diretório do aplicativo abaixo do diretório da organização. `<root>/<organization_id>/<application_id>/`, por exemplo.
    - Para diretórios no caminho referido por qualquer credencial autenticada na política por aplicativo até um curinga ("*" ou "-").
    - Antes ou durante o primeiro acesso ao diretório ou qualquer arquivo no diretório.
    - O proprietário de diretórios criados abaixo de um diretório de aplicativo DEVE ser o mesmo proprietário do diretório do aplicativo para que o proprietário sempre tenha acesso ao seu próprio diretório e os arquivos nele contidos. O proprietário DEVE ser a organização concatenada de 48 bits e o identificador do aplicativo.
    - O grupo de diretórios criados abaixo do diretório de uma organização DEVE ser o mesmo que o grupo do diretório da organização para que os aplicativos com a mesma organização possam

compartilhar acesso ao seu grupo de diretórios e arquivos neles contidos dependendo dos direitos de acesso do grupo.

- O proprietário do diretório da organização DEVE ser a plataforma.
- Quando criados, os direitos de acesso do usuário DEVEM conceder acesso de leitura, escrita e busca ao proprietário.
- Quando criados, os direitos de acesso do grupo NÃO PODEM conceder acesso de leitura, escrita ou busca para outros membros do grupo ou qualquer outro aplicativo. O aplicativo pode mudar os direitos quando necessário.
- Na situação incomum em que o diretório abaixo do diretório do aplicativo não está em posse do aplicativo, esse diretório DEVE ser deletado e recriado como um diretório vazio com o aplicativo como proprietário.

- Todos os aplicativos assinados e autenticados que requisitarem acesso a arquivos na política por aplicativo DEVEM receber acesso como descrito na Tabela 8.

Tabela 8 – Acesso a ser concedido a aplicativos assinados e autenticados

Caminho	Acesso do proprietário	Acesso do grupo	Acesso mundial
<root>/*	ler	ler	ler
<root>/<organization_id>/*	ler/escrever	ler	-
<root>/<organization_id>/<application_id>/*	ler/escrever	-	-

15 Segurança do canal de retorno

A segurança do canal de retorno é provida pelo suporte do TLS 1.2 definido em RFC 5246. Um terminal conforme com essa especificação DEVE implementar o pacote opcional JSSE 1.01 encontrado no Foundation Profile 1.1 (JSR219) para prover aos aplicativos acesso a funcionalidade SSL/TLS A implementação TLS DEVE suportar os seguintes pacotes de cifras:

- `TLS_NULL_WITH_NULL_NULL`
- `TLS_RSA_WITH_NULL_MD5`
- `TLS_RSA_WITH_NULL_SHA`
- `TLS_RSA_EXPORT_WITH_DES40_CBC_SHA`
- `TLS_RSA_WITH_DES_CBC_SHA`
- `TLS_RSA_WITH_3DES_EDE_CBC_SHA`
- `TLS_RSA_WITH_AES_128_CBC_SHA`
- `TLS_RSA_WITH_AES_256_CBC_SHA`

Mais informações em relação a esses pacotes de cifras podem ser encontradas na RFC 5246.

16 Especificação de arquivos de recursos

16.1 Considerações gerais

Estas definições fazem parte da especificação Java DTV e documentam o formato dos arquivos de recurso como

requerido pela classe `com.sun.dtv.lwuit.util.Resources`.

16.2 Recursos

16.2.1 Descrição

O arquivo de recurso é composto de blocos, cada tipo de bloco é descrito na Tabela 9. O primeiro bloco no arquivo de recurso DEVE ser do tipo *header* (cabeçalho) o qual descreve o arquivo de recurso.

Observa-se que embora os arquivos de recurso liberem espaço para armazenar metadados, é aconselhado que se faça pouco uso de tais facilidades para evitar aumento de tamanho e exposição de informação. Os arquivos de recurso são projetados para serem enviados com um aplicativo e, portanto, deveriam manter-se compactados com apenas o metadado requerido pelo usuário final permanecendo dentro do aplicativo.

Arquivos de recurso são planejados para serem lidos ou gravados usando ferramentas Java e são projetados em torno da arquitetura do `DataInputStream/DataOutputStream`. Isso tem várias implicações em relação a especificação e ao formato do arquivo:

- elementos UTF na especificação se referem aos métodos `readUTF/writeUTF` do `DataInputStream` e não a *strings* UTF terminadas em `null` no estilo da linguagem C;
- o arquivo usa *big endian* para representar todos os tipos;
- codificação de cores usa o modelo ARGB.

16.2.2 Formato geral

O formato geral é descrito na Tabela 9.

Tabela 9 – Formato Geral

O quê	Tipo	Tamanho Bytes	Fator de iteração
Identificador"mágico" “LWUITRF\0”	`BYTE[]`	8	-
Número de blocos > 1	`SHORT`	2	-
Blocos, primeiro bloco deve ser do tipo *header*	-	Individual	Para qualquer recurso

16.2.3 Blocos

O formato do bloco é descrito na Tabela 10.

Tabela 10 – Formato do bloco

O quê	Tipo	Tamanho Bytes	Fator de iteração
Tipos de recursos Tema: 0xF2 - L10N: 0xF9 - Dado: 0xFA - Fonte: 0xFC - Imagem: 0xFD - *Header*: 0xFF	`BYTE`	1	-
Nome dos recursos	`UTF`	Individual	-
Dados dos recursos	Ver itens 16.3 a 16.8	Individual	-

16.3 Header

O *header* (cabeçalho) deve ser o primeiro bloco do arquivo a permitir ferramentas e o aplicativo para identificar detalhes sobre o arquivo de recurso e fornecer ferramentas para estensibilidade futura. Seu formato é descrito na Tabela 11.

A versão atual da especificação tem versão principal 1 e versão secundária 3.

Tabela 11 – Formato do *Header*

O quê	Tipo	Tamanho Bytes	Fator de iteração
Tamanho do *Header*	`SHORT`	2	-
Versão principal	`SHORT`	2	-
Versão secundária	`SHORT`	2	-
Contagem de metadados	`SHORT`	2	-
Strings de metadados	`UTF[]`	2	-

16.4 Bloco Tema

16.4.1 Descrição

Temas são um conjunto de pares de chaves-valores onde a chave representa um *string* bastante conhecido usando o formato de `[ComponentUIID.]attribute`. Os nomes do `ComponentUIID` são definidos pelo usuário, os atributos, por outro lado, são bem conhecidos e determinam o tipo de valor armazenado no arquivo de recurso.

O mapeamento dos atributos como descrito na Tabela 12 aplica-se ao tema.

Tabela 12 – Mapeamento dos atributos

Atributo	Tipo
`fgColor, bgColor, fgSelectionColor, bgSelectionColor`	`Color`
`font`	`Font`
`padding, margin`	`Spacing`
`transparency`	`Transparency`
`Background, selectionBackground`	`Background`
`border`	`Border`

16.4.2 Estrutura do recurso de tema

A estrutura do recurso de tema está descrita na Tabela 13.

Tabela 13 – Estrutura do recurso de tema

O quê	Tipo	Tamanho Bytes	Fator de iteração
Número de Propriedades	`SHORT`	2	-

Par de propriedades	Ver 10.3.2	Individual	Para qualquer propriedade

16.4.3 Par de propriedades de tema

A estrutura de um par de propriedades de tema está descrita na Tabela 14.

Tabela 14 – Estrutura de par de propriedades de tema

O quê	Tipo	Tamanho Bytes	Fator de Iteração
Chave da propriedade	UTF	Individual	-
Valor da propriedade	Ver 16.4.4	Individual	-

16.4.4 Valores de propriedade de tema

16.4.4.1 Valor de propriedade de tema: COLOR

O componente alfa da cor é ignorado independente do seu valor. O valor de propriedade de tema COLOR é estruturado como descrito na Tabela 15.

Tabela 15 – Estrutura do valor de propriedade de tema COLOR

O quê	Tipo	Tamanho Bytes	Fator de iteração
Valor da cor	INT	4	-

16.4.4.2 Valores de propriedade de tema: TRANSPARENCY

Representa o valor de transparência alfa entre 0x00 - 0xff. O valor de propriedade de tema TRANSPARENCY é estruturado como descrito na Tabela 16.

Tabela 16 – Estrutura do valor de propriedade de tema TRANSPARENCY

O quê	Tipo	Tamanho Bytes	Fator de Iteração
Valor da transparência	BYTE	1	-

16.4.4.3 Valores de propriedade de tema: SPACING

O valor de propriedade de tema SPACING é estruturado como descrito na Tabela 17.

Tabela 17 – Estrutura do valor de propriedade de tema SPACING

O quê	Tipo	Tamanho Bytes	Fator de iteração
Espaçamento superior	BYTE	1	-
Espaçamento inferior	BYTE	1	-
Espaçamento esquerdo	BYTE	1	-
Espaçamento direito	BYTE	1	-

16.4.4.4 Valores de propriedade de tema: FONT

Refere-se a um bloco de fontes pelo nome. O valor de propriedade de tema FONT é estruturado como descrito na Tabela 18.

Tabela 18 – Estrutura do valor de propriedade de tema FONT

O quê	Tipo	Tamanho Bytes	Fator de iteração
Nova fonte	BOOLEAN	1	-
Nome da fonte	UTF	Individual	Apenas se nova fonte for = *true*
Face da fonte	BYTE	1	Apenas se nova fonte for = *false*
Estilo da fonte	BYTE	1	Somente se nova fonte for = *false*
Tamanho da fonte	BYTE	1	Somente se nova fonte for = *false*

16.4.4.5 Valores de propriedade de tema: BACKGROUND

Representa o desenho de plano de fundo para um componente, comportamento da imagem, gradiente, etc.

Gradientes radiais têm uma posição central para o núcleo do efeito radial, esse centro é determinado relativamente usando as variáveis x relativo/y relativo. Essas variáveis estão no intervalo entre 0.0 a 1.0 e elas representam a posição dentro do componente relativa ao seu tamanho, onde 0.5/0.5 é o centro exato do componente. O gradiente radial deve ser desenhado exatamente nessa variação.

O valor de propriedade de tema BACKGROUND é estruturado como descrito na Tabela 19.

Tabela 19 – Estrutura do valor de propriedade de tema BACKGROUND

O quê	Tipo	Tamanho Bytes	Fator de iteração
Tipo Imagem escalonada: 0xF1 Imagem em lado a lado vertical: 0xF2 Imagem em lado a lado horizontal: 0xF3 Imagem em lado a lado: 0xF4 Imagem alinhada 0xF5 Gradiente linear horizontal: 0xF6 Gradiente linear vertical: 0xF7 Gradiente radial: 0xF8	`BYTE`	1	-
Imagem	`UTF` (identificador de referência da imagem)	individual	Apenas para imagens relacionadas aos tipos 0xf1-0xf5
Alinhamento Superior: 0xf1 Inferior: 0xf2 Centro: 0xf3 Esquerda: 0xf4 Direita: 0xf5	`BYTE`	1	Apenas para imagens lado a lado verticais/horizontais e imagens alinhadas.
Cor inicial	`INT`	4	Apenas para tipos gradiente
Cor final	`INT`	4	Apenas para tipos gradiente
X relativo	`FLOAT`	4	Apenas para tipos gradiente
Y relativo	`FLOAT`	4	Apenas para tipos gradiente
Tamanho relativo	`FLOAT`	4	Apenas para tipos gradiente

16.4.4.6 Valores de propriedade de tema: BORDER

16.4.4.6.1 Descrição

A propriedade borda permite a definição de um componente borda, correspondendo ao comportamento da classe LWUIT `Border`. Isso inclui alguns tipos de borda integrados que podem ser aumentados em versões futuras do LWUIT.

Cada tipo de borda tem um conjunto diferente de variáveis para indicar sua aparência.

Algumas bordas suportam cores determinadas pelo usuário como uma opção, a borda pode receber uma *flag* indicando se a cor deve ser extraída do tema ou se deve ser especificada especialmente para essa instância de

borda.

16.4.4.6.2 Estrutura BORDER

O valor de propriedade de tema BORDER é estruturado como descrito na Tabela 20.

Tabela 20 – Estrutura do valor de propriedade de tema BORDER

O quê	Tipo	Tamanho Bytes	Fator de iteração
Tipo de borda: Nenhum: 0xff01 Linha: 0xff02 Arredondada: 0xff03 Gravada rebaixada: 0xff04 Gravada em relevo: 0xff05 Bisel rebaixado: 0xff06 Bisel em relevo: 0xff07 Imagem: 0xff08	SHORT	2	-
Dado	Dado da borda	Individual (sem conteúdo no caso de nenhuma borda)	-

16.4.4.6.3 Dados de BORDER: Linha

Os dados de bordas do tipo linha são estrurados como descrito na Tabela 21.

Tabela 21 – Estrutura de dados de borda do tipo linha

O quê	Tipo	Tamanho Bytes	Fator de iteração
Cores do tema	BOOLEAN	1	-
Espessura	BYTE	1	-
Cor	INT	4	Apenas se cores do tema for *false*

16.4.4.6.4 Dados de BORDER: Arredondada

Os dados de borda do tipo arredondada são estruturados como descrito na Tabela 22.

Tabela 22 – Estrutura de dados de borda do tipo arredondada

O quê	Tipo	Tamanho Bytes	Fator de iteração
Cores do tema	BOOLEAN	1	-
Largura do arco	BYTE	1	-
Altura do arco	BYTE	1	
Cor	INT	4	Apenas se cores do tema for *false*

16.4.4.6.5 Dados de BORDER: Gravada rebaixada/em relevo

Os dados de bordas do tipo gravado rebaixado/em relevo são estruturados como descrito na Tabela 23.

Tabela 23 – Estrutura de dados de borda do tipo gravada rebaixada/em relevo

O quê	Tipo	Tamanho Bytes	Fator de iteração
Cores do tema	`BOOLEAN`	1	-
Cor de destaque	`INT`	4	Apenas se cores do tema for *false*
Cor de sombra	`INT`	4	Apenas se cores do tema for *false*

16.4.4.6.6 Dados de BORDER: bisel abaixado/levantado

Os dados de bordas do tipo bisel E são estruturados como descrito na Tabela 24.

Tabela 24 – Estrutura de dados de borda do tipo bisel

O quê	Tipo	Tamanho Bytes	Fator de Iteração
Cores de tema	`BOOLEAN`	1	-
Destacar cor exterior	`INT`	4	Apenas se cores do tema for *false*
Destacar cor interior	`INT`	4	Apenas se cores do tema for *false*
Sombrear cor exterior	`INT`	4	Apenas se cores do tema for *false*
Sombrear cor interior	`INT`	4	Apenas se cores do tema for *false*

16.4.4.6.7 Dados de BORDER: Imagem

Um borda de imagem suporta dois padrões, 9 imagens ou 3 imagens, a última dessas imagens pode sernula, portanto a borda também pode suportar 2 ou 8 imagens.

Os dados de borda do tipo Imagem são estruturados como descrito na Tabela 25.

Tabela 25 – Estrutura de dados de borda do tipo imagem

O quê	Tipo	Tamanho Bytes	Fator de iteração
Contagem de imagens	`BYTE`	1	-
Imagens	`UTF` (referência ao bloco image)	Individual * contagem de imagem	-

NOTA O uso do sinal "*" em expressões como a acima denota uma operação matemática de multiplicação.

16.5 Bloco Imagem

16.5.1 Descrição

Um recurso de imagem também pode representar uma animação simples dentro do arquivo de recursos. Há diversos tipos de imagem suportados dentro do bloco imagem:

16.5.2 Tipos de imagem

A Tabela 26 lista os tipos de imagem disponíveis.

Tabela 26 – Tipos de imagem

O quê	Tipo	Tamanho Bytes	Fator de iteração
Tipo de imagem PNG: 0xf1 JPEG: 0xf2 Indexada: 0xf3 Animação: 0xf4 SVG: 0xf5	BYTE	1	-
Imagem	-	Individual	-

A estrutura de descrição de imagens pode ser vista na Tabela 27 para imagens PNG/JPEG, na Tabela 28 para imagens indexadas, na Tabela 29 para imagens animadas e na Tabela 34 para imagens SVG.

16.5.3 Imagem PNG/JPEG

A estrutura de dados de imagens PNG/JPEG é descrita na Tabela 27.

Tabela 27 – Imagens PNG/JPEG

O quê	Tipo	Tamanho Bytes	Fator de iteração
Comprimento dos dados da imagem	INT	4	-
Dados da imagem	-	Individual	-

16.5.4 Imagem indexada

Imagens indexadas são baseadas em *bitmaps* de um valor de paleta de consulta. A estrutura de dados de imagens indexadas é descrita na Tabela 28.

Tabela 28 – Imagens indexadas

O quê	Tipo	Tamanho Bytes	Fator de iteração
Tamanho da paleta de cores	BYTE	1	-
Cor	INT[]	4	Para qualquer cor na paleta (0 indica 256 cores)
Largura da imagem	SHORT	2	-
Altura da imagem	SHORT	2	-
Valor de cor por pixel da imagem	BYTE	1	Para qualquer pixel da imagem

16.5.5 Animação da imagem

16.5.5.1 Formato geral

A estrutura de dados de imagens de animação é descrita na Tabela 29.

Tabela 29 – Imagens de animação

O quê	Tipo	Tamanho Bytes	Fator de iteração
Tamanho da paleta de cores	BYTE	1	-
Cor	INT[]	4	Para qualquer cor na paleta (0 indica 256 cores)
Largura da animação	SHORT	2	-
Altura da animação	SHORT	2	-
Número de quadros da animação	BYTE	1	-
Tempo total da animação	INT	4	-
Indicador do *loop* da animação	BOOLEAN	1	-
Dados do quadro da animação	Ver 16.5.5.2	Individual	Para qualquer quadro da animação

16.5.5.2 Dados do quadro da animação

Os dados do quadro da animação representam um quadro dentro da animação, o número de quadros depende da complexidade da animação. Os quadros deveriam ser lidos um por um com exceção do primeiro quadro, o qual é sempre apenas um *bitmap*. A Tabela 30 mostra a estrutura.

Tabela 30 – Estrutura de dado de quadro de animação

O quê	Tipo	Tamanho Bytes	Fator de iteração
Primeiro quadro	Primeiro quadro	Individual	1

Quadro	Quadro	Individual	Número de quadros da animação - 1

16.5.5.3 Primeiro quadro

O primeiro quadro de uma animação é sempre um quadro chave e não precisa de marca de tempo. Ele é um *bitmap* da paleta da primeira imagem dentro da animação. A Tabela 31 mostra como o quadro 1 é estruturado.

Tabela 31 – Estrutura do primeiro quadro

O quê	Tipo	Tamanho Bytes	Fator de iteração
Índice da paleta de cores	`BYTE[]`	1	Largura da animação * altura da animação

16.5.5.4 Quadro

O quadro chave pinta toda a animação, é um *bitmap* e, portanto, dispendioso em termos de memória/armazenamento. A Tabela 32 mostra como os quadros, exceto o primeiro, são estruturados.

Tabela 32 – Estrutura dos quadros, exceto o primeiro

O quê	Tipo	Tamanho Bytes	Fator de iteração
Marca de tempo	`INT`	4	-
Indicador de quadro chave	`BOOLEAN`	1	-
Quadro chave (índices de paleta)	`BYTE[]`	Largura da animação * altura da animação	Apenas se o indicador quadro chave for *true*
Desenho do quadro anterior	`BOOLEAN`	1	Apenas se o indicador quadro chave for *false*
Linhas alteradas	Linhas alteradas	Individual	Apenas se o indicador quadro-chave for *false*

16.5.5.5 Linhas alteradas

A Tabela 33 descreve a estrutura das linhas alteradas em uma imagem.

Tabela 33 – Estrutura de linhas alteradas

O quê	Tipo	Tamanho Bytes	Fator de iteração
Deslocamento da linha	`SHORT`	2	Repetir sobre as linhas até que o deslocamento da linha seja -1
Dados da linha (índices de paleta)	`BYTE[]`	Largura da animação	-

16.5.6 Imagem SVG

Uma imagem SVG pode ser processada em dispositivos que tem suporte para imagens SVG integrado, quando tal suporte não está disponível a imagem reserva é mostrada. Implementações que não são voltadas para ambientes não-SVG podem definir o comprimento da reserva para 0. Uma reserva tem uma razão largura/altura em ponto

flutuante representando a razão entre a imagem reserva e o tamanho da tela. Essa proporção permite que as ferramentas adaptem o arquivo de recursos para múltiplas resoluções de tela enquanto preserva a proporção relativa da imagem.

A URL base é usada para buscar recursos requeridos pelo analisador SVG, se a URL é uma *string* vazia, a implementação pode procurar no interior do próprio arquivo de recursos, ou no *classpath*.

Observa-se que o dado da imagem reserva pode ser 0, indicando que não há imagem reserva.

Tabela 34 – Imagens SVG

O quê	Tipo	Tamanho Bytes	Fator de iteração
Comprimento do arquivo SVG	`INT`	4	-
Dados do SVG	`BYTE[]`	Individual	-
URL base	`UTF`	Individual	-
Animada	`BOOLEAN`	1	-
Largura da imagem reserva	`FLOAT`	4	-
Altura da imagem reserva	`FLOAT`	4	-
Comprimento dos dados da imagem reserva	`INT`	4	-
Dados da imagem reserva	`BYTE[]`	Individual	-

16.6 Bloco Fonte

16.6.1 Formato geral

Uma fonte contém um conjunto de valores reserva opcionais ordenados por prioridade e sempre terminando com uma definição de fonte *system*. Se uma plataforma não suporta ou não deveria suportar um dado tipo de fonte, ela se move para a próxima fonte disponível na cadeia.

Há 4 tipos de fonte, algumas das quais podem não ser suportadas em uma dada plataforma. O único tipo de fonte requerido é a fonte *system*, a qual deve trabalhar corretamente para todas as plataformas/codificações de caracteres e serve como reserva:

- Fonte *system*: define uma fonte baseada em atributos do menor denominador comum que trabalharia em todos os dispositivos.
- Fonte *bitmap*: representa um mapeamento de *bitmap* e de conjunto de caracteres para permitir o desenho da fonte em algumas plataformas que não suportam fontes *true type* ou uma tabela de *lookup*;
- Fonte *true type*: suporta o embutimento de uma fonte *true type* no arquivo de recurso;
- Fonte *lookup*: uma *string* representando uma fonte utilizando uma sintaxe de *lookup* de fonte específica, por exemplo, `Arial-Bold-16`. É um conjunto de *strings* separado por vírgula que permite ao LWUIT tentar diversos *strings* baseado na disponibilidade de fontes da plataforma.

As fontes são carregadas como uma cadeia reserva onde cada um dos tipos de fonte é armazenado por ordem de qualidade, a primeira fonte estocada no arquivo e suportada pela plataforma é usada. A ordem de armazenagem/escolha de fontes é:

- Fonte *true type*
- Fonte *lookup*

- Fonte *bitmap*
- Fonte *system*

16.6.2 Estrutura

A Tabela 35 descreve a estrutura de um bloco fonte.

Tabela 35 – Estrutura de bloco fonte

O quê	Tipo	Tamanho Bytes	Fator de iteração
systemFontFallback Valor *bitwise*: - MONOSPACE : 32 - PROPORATIONAL : 64 - SYSTEM : 0 - BOLD: 1 - ITALIC: 2 - PLAIN: 0 - LARGE: 16 - SMALL: 8 - MEDIUM: 0	BYTE	1	-
isTrueTypeFontIncluded	BOOLEAN	1	-
Tamanho *true type*	INT	4	Apenas se a fonte true type estiver incluída
Dados *true type*	BYTE[]	Tamanho *TrueType*	Apenas se a fonte true type estiver incluída
isLookupIncluded	BOOLEAN	1	-
Nome da fonte *lookup*	UTF		Apenas se a fonte *lookup* estiver incluída
isBitmapIncluded	BOOLEAN	1	-
Imagem	Imagem (ver seção 16.5)	Individual	Apenas se a fonte *bitmap* estiver incluída
Contagem de caracteres	SHORT	2	Apenas se a fonte *bitmap* estiver incluída
Deslocamentos de corte	SHORT	Contagem de caracteres	Apenas se a fonte *bitmap* estiver incluída
Largura de caracteres	BYTE	Contagem de caracteres	Apenas se a fonte *bitmap* estiver incluída
Conjunto de caracteres	UTF	Individual	Apenas se a fonte *bitmap* estiver incluída
Dica de processamento	BYTE	1	Apenas se a fonte *bitmap* estiver incluída

16.6.3 Instância da fonte

16.6.3.1 Formato geral

As fontes na tabela são organizadas por plataforma que represente um conjunto descritivo de nomes para identificar uma plataforma específica, por exemplo, MIDP, PBP, RIM, etc. A plataforma reserva padrão terá uma *string* de comprimento 0 e deve sempre ser tentada caso não se encontrem as outras *strings* de plataforma. As plataformas são organizadas por prioridade como, por exemplo, sendo executada em um dispositivo RIM a plataforma RIM seria procurada primeiro, depois a plataforma MIDP.

Os nomes das plataformas estão sujeitos a mudanças e estensões à medida que novas plataformas LWUIT forem adicionadas. Um nome de plataforma deve incluir apenas caracteres alfa-numéricos ([A-Z][a-z][0-9]) e é tratado como insensível a caixa. Os nomes das plataformas podem ser delimitados por vírgulas para associar uma única fonte com múltiplas plataformas.

A estrutura geral de uma instância de fonte é descrita na Tabela 36. A Tabela 37 descreve a estrutura de uma fonte *system*, a Tabela 38 mostra uma fonte *bitmap*. A estrutura de uma fonte *true type* pode ser vista na Tabela 39, a estrutura de uma fonte *lookup* na Tabela 40.

Tabela 36 – Estrutura geral de uma instância de fonte

O quê	Tipo	Tamanho Bytes	Fator de iteração
Fonte *system* padrão	Fonte *system*	1	-
Tamanho da tabela	`BYTE`	1	-
Plataforma	`UTF`	Individual para todos os elementos da tabela (tamanho da tabela)	-
Fonte	`UTF`	Individual para todos os elementos da tabela (tamanho da tabela)	-

Um modelo de tabela pode parecer-se com isto:

```
Fonte System: BOLD

Tamanho da Tabela: 3

Plataforma: "RIM,MIDP"

Fonte: Bitmap Font (binary data)

Plataforma: "PBP"

Fonte: "Arial-Bold-14"

Plataforma: "HDTV"

Fonte: "Arial-Bold-30"
```

Uma vez que as tabelas devem ser processadas por prioridade, uma plataforma de televisão deve escolher a fonte de 30 pixels em vez da fonte de 14 pixels, a qual deve ser apropriada também para plataformas PBP. Plataformas MIDP/RIM podem ignorar essa fonte completamente.

Observa-se que a fonte *system* é a padrão e é sempre requerida como reserva válida por todas as plataformas.

16.6.3.2 Instância de fonte *system*

A Tabela 37 descreve a estrutura de uma fonte *system.*

Tabela 37 – Estrutura de uma fonte *system*

O quê	Tipo	Tamanho Bytes	Fator de iteração
Valor *bitwise*: - MONOSPACE : 32 - PROPORATIONAL : 64 - SYSTEM : 0 - BOLD: 1 - ITALIC: 2 - PLAIN: 0 - LARGE: 16 - SMALL: 8 - MEDIUM: 0	BYTE	1	-

16.6.3.3 Instância de fonte *bitmap*

A fonte *bitmap* é essencialmente uma imagem PNG horizontal com pontos de corte, é pintada em tons de vermelho que são manipulados pelo código de processamento da fonte.

A fonte possui uma propriedade geradora de fonte que permite que as ferramentas do editor regenerem a fonte usando as informações de nome/tamanho/estilo da fonte em questão codificadas no interior daquela *string*.

Tabela 38 – Estrutura de uma fonte *bitmap*

O quê	Tipo	Tamanho Bytes	Fator de iteração
Imagem	Imagem (ver seção 16.5)	Individual	-
Contagem de caracteres	SHORT	2	-
Deslocamentos de corte	SHORT	Contagem de caracteres * 2	-
Largura de caracteres	BYTE	Contagem de caracteres	-
Conjunto de caracteres	UTF	Individual	-
Dica de processamento	BYTE	1	-
Fonte geradora	UTF	Individual	-

16.6.3.4 Instância de fonte *true type*

A estrutura geral de uma fonte *true type* é mostrada na Tabela 39.

Tabela 39 – Estrutura de uma fonte *true type*

O quê	Tipo	Tamanho	Fator de iteração

		Bytes	
Tamanho	INT	4	-
Arquivo	BYTE[]	Individual	-

16.6.3.5 Instância de fonte *lookup*

A estrutura geral de uma fonte *lookup* é mostrada na Tabela 40.

Tabela 40 – Estrutura de uma fonte *lookup*

O quê	Tipo	Tamanho Bytes	Fator de iteração
Lookup	UTF	Individual	-

16.7 Bloco L10N

16.7.1 Recurso L10N

A estrutura geral de um bloco L10N é descrita na Tabela 41.

Tabela 41 – Estrutura geral de um bloco L10N

O quê	Tipo	Tamanho Bytes	Fator de Iteração
Número de propriedades chave L10N	SHORT	2	-
Número de línguas L10N	SHORT	2	-
Propriedade chave L10N	UTF	Individual	Para qualquer propriedade L10N
Valores de propriedades L10N	Ver 16.7.2	Individual	Para qualquer língua L10N

16.7.2 Valores de propriedades L10N

Os valores das propriedades L10N estão descritos na Tabela 42.

Tabela 42 – Estrutura do valor de propriedades L10N

O quê	Tipo	Tamanho Bytes	Fator de iteração
Língua L10N	UTF	Individual	-
Valor da propriedade L10N para esta língua	UTF	Individual	Para qualquer propriedade L10N

16.8 Bloco dados

A Tabela 43 mostra a estrutura geral de um bloco de dados.

Tabela 43 – Estrutura de um bloco de dados

O quê	Tipo	Tamanho Bytes	Fator de iteração
Comprimento do dado	INT	4	-
Dado	-	Individual	-

17 Índice de pacotes

A Tabela 44 fornece um índice de todos os pacotes.

Tabela 44 – Índice de todos os pacotes Java DTV

Nome do pacote	Descrição resumida do pacte
com.sun.dtv.application	Gerenciamento de aplicativos e controle do ciclo de vida
com.sun.dtv.broadcast	Acesso a arquivos de transmissão e *streams*
com.sun.dtv.broadcast.event	Acesso a eventos de transmissão
com.sun.dtv.filtering	Provê suporte para filtro de seções MPEG
com.sun.dtv.io	Estende o pacote java.io provendo direitos de acesso e propriedades aos arquivos
com.sun.dtv.locator	Esse pacote define localizadores para uso através de todo o sistema
com.sun.dtv.lwuit	Pacote *widget* principal contendo o "composto" componente/container similares tanto em terminologia quanto em projeto para *Swing/AWT*
com.sun.dtv.lwuit.animations	Todos os componentes são animáveis por animações potenciais e adicionais (não-relacionadas a um componente específico) podem ser instalados instantaneamente, transições entre formas também são tratadas como parte desse pacote
com.sun.dtv.lwuit.events	Padrão observável de receptores de evento no espírito da arquitetura despachadora de eventos do AWT1.1, todos os eventos são despachados na EDT (*Event Dispatch Thread*)
com.sun.dtv.lwuit.geom	Contém classes relacionadas aos locais geométricos e cálculos tais como retângulo e tamanho
com.sun.dtv.lwuit.layouts	Gerenciadores de layout permitem que um *container* organize seus componentes por um conjunto de regras que seriam adaptadas para tamanhos específicos de fonte/tela
com.sun.dtv.lwuit.list	As listas são altamente personalizáveis e servem como base para o *ComboBox* e outros componentes (tais como carrossel, etc.). Elas empregam uma abordagem MVC similar ao SWING incluindo o padrão renderizador
com.sun.dtv.lwuit.painter	*Painter* permite desenhar elementos arbitrários de gráficos provenientes de imagens simples/escalonadas/lado a lado a gradientes e praticamente todas as formas de desenho gráfico que pudermos imaginar
com.sun.dtv.lwuit.plaf	A aparência do aplicativo pode ser inteiramente personalizada por meio desse pacote, ele representa uma camada processadora que pode ser plugada separadamente em runtime e tematizado para prover qualquer aparência personalizada
com.sun.dtv.lwuit.util	Funções de utilitários que são ou muito específicas de um domínio ou não se "encaixam" em nenhum outro pacote
com.sun.dtv.media	Pacote para funcionalidade básica relevante de mídia e controles congela / restaura
com.sun.dtv.media.audio	Pacotes para controle de língua do áudio
com.sun.dtv.media.control	Controles adicionais para obter informações sobre a mídia apresentada
com.sun.dtv.media.dripfeed	Controle para prover dado de imagem fixa para um tocador JMF, os dados ficam sob controle do aplicativo

Tabela 44 (continuação)

Nome do pacote	Descrição resumida do pacte
com.sun.dtv.media.format	Controle para o formato do vídeo
com.sun.dtv.media.language	Provê a funcionalidade básica de informar-se sobre e definir a língua do áudio, das legendas e do *closed caption*
com.sun.dtv.media.text	Controle das legendas e do *closed caption*
com.sun.dtv.media.timeline	Controle da linha de tempo da mídia
com.sun.dtv.net	Estende o pacote java.net com controle de dispositivo de comunicação extensiva
com.sun.dtv.platform	Provê classes que são específicas da plataforma Java DTV, particularmente classes que são específicas para o consumidor
com.sun.dtv.resources	Provê um quadro básico de recursos escassos
com.sun.dtv.security	Pacote para a funcionalidade adicional de segurança
com.sun.dtv.service	Esse pacote provê uma interface para acessar o banco de dados SI de baixo nível
com.sun.dtv.smartcard	Pacote que provê funcionalidade *smartcard* adicional
com.sun.dtv.test	Provê um suporte extensivo de controle de teste
com.sun.dtv.transport	Provê acesso a entidades contidas em um *stream* de transporte
com.sun.dtv.tuner	Provê API para acesso a e controle de interface de redes de transmissão (ou “sintonizador”) usado para recepção de *streams* de transporte
com.sun.dtv.ui	Funcionalidade UI específica da televisão
com.sun.dtv.ui.event	Subpacote de evento de funcionalidade UI específica da TV

18 Pacote com.sun.dtv.application

18.1 Descrição do pacote

Gerenciamento de aplicativos e controle de ciclo de vida.

18.2 Índice de interfaces

Application

Obtenha atributos de um aplicativo.

ApplicationProxy

Controle um aplicativo por meio do gerenciador de aplicativos.

AppManagerListener

Listener a ser notificado de mudanças em aplicativos disponíveis.

AppProxyListener

Listener a ser notificado de mudanças no estado dos aplicativos.

18.3 Índice de classes

AppFilter

`AppFilter` é chamado para aceitar um aplicativo em um filtro ou não.

AppManager

`AppManager` provê controle de e acesso a aplicativos.

AppManagerPermission

`AppManagerPermission` é necessário para informar-se sobre ou controlar aplicativos.

18.4 Aplicativos Java DTV

18.4.1 Considerações gerais

Aplicativos Java DTV rodam em um ambiente direcionado por serviço por um gerenciador de aplicativos que cobre todo o sistema. O ambiente de execução do Java DTV é um dos aplicativos runtime no dispositivo gerenciado pelo gerenciador de aplicativos. Aplicativos Java DTV rodam em um ambiente multi-tarefa dinâmico direcionado por e respondendo a eventos da mídia em transmissão, entrada do usuário e sintonizadores.

O gerenciador de aplicativos é um componente do sistema fora do ambiente de execução do Java DTV que monitora e controla a execução de todos os aplicativos. O gerenciador de aplicativos gerencia os contextos de serviço e o serviço selecionado em cada contexto. Cada serviço define os aplicativos que têm permissão para rodar dentro do seu contexto. O gerenciador de aplicativos inicia e interrompe aplicativos para responder a mudanças no contexto do serviço pelo dispositivo e pela interface do usuário. No caso de assinalar com `KILL` o aplicativo é parado usando `Xlet.destroyApp(`*`false`*`)`; o aplicativo pode escolher continuar. No caso de assinalar com `DESTROY`, o aplicativo é para usando `Xlet.destroyApp(true)`; o aplicativo é destruído incondicionalmente.

O gerenciador de dispositivos usa o aplicativo da mídia em transmissão assinalando para para identificar o conjunto de aplicativos que são considerados estar disponíveis para cada serviço. A informação proveniente da mídia de transporte é definida pela ABNT NBR15606-3, 12.18. Essa informação é posta em prática pelo gerenciador de aplicativos para iniciar e parar aplicativos, fazer com que estejam disponíveis ou não e fazer com que estejam visíveis ao usuário ou outros aplicativos ou não.

O gerenciador de aplicativos coleta e mantém informações sobre os aplicativos que estão disponíveis e que estão rodando. Os aplicativos podem acessar a lista de aplicativos disponíveis e ativos usando a classe `AppManager`. Os atributos de cada aplicativo estão disponíveis na classe `Application`. Monitoramento e controle de aplicativos é por meio da interface `ApplicationProxy`. Monitoramento e controle de aplicativos são permitidos apenas se a política de segurança permitir o `AppManagerPermission`.

A Figura 4 mostra a estrutura do pacote `Application` do Java DTV.

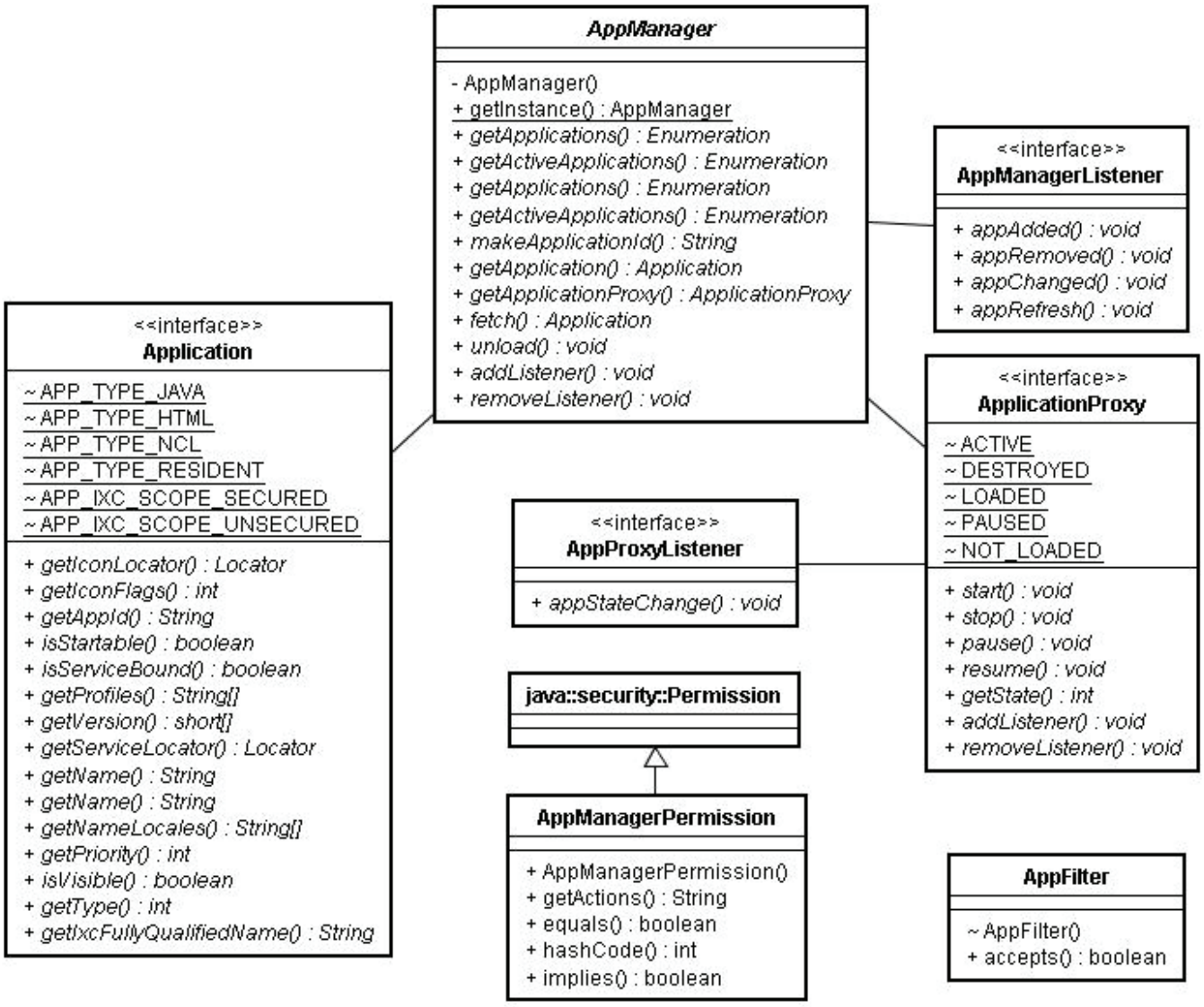


Figura 4 – Aplicações no Java DTV

O modelo de aplicações do Java DTV inclui:

- Empacotamento de aplicações Java DTV - Java ARchive (JAR)
- Ambiente de execução Java DTV - *classpath*, bibliotecas de sistema etc.
- Ciclo de vida Java DTV - `Xlet` e `XletContext`
- Autenticação Java DTV - JAR signing
- Autorização de aplicações Java DTV - Política de segurança
- Contexto de serviços Java DTV - Ligação ao serviço

18.4.2 Empacotamento de aplicações Java DTV

Cada aplicativo é empacotado, autenticado e autorizado de acordo com o DTV Application Packaging and Security comum. Requisitos adicionais especificam o comportamento de aplicativos Java DTV.

Cada aplicativo é contido dentro de um ou mais arquivos JAR [JAR]. O JAR principal contém os arquivos de classe de aplicativos, arquivos de recurso e um manifesto que descreve o aplicativo e seus requerimentos. O mecanismo de assinatura JAR permite que o JAR seja autenticado.

A especificação JAR inclui atributos para controlar como o aplicativo é executado. A especificação de arquivo JAR [JAR] define os atributos e sua interpretação. Os atributos relevantes estão descritos na Tabela 45.

Tabela 45 – Atributos relevantes da especificação de arquivo JAR

Atributo do manifesto	Descrição
`Main-Class`	A classe do aplicativo implementando a interface
`Class-Path`	Refere-se a JAR adicionais que são acrescentados ao *classpath* e são disponibilizados no ambiente de execução

18.4.3 Ambiente de execução Java DTV

O ambiente de execução do Java DTV é instituído quando um aplicativo precisa ser iniciado e é descartado quando o aplicativo é destruído. O *ambiente de execução* existe dentro de uma Java VM e provê o ambiente para um aplicativo Java DTV.

O *ambiente de execução* provê isolamento limitado entre aplicativos. Toda a interação entre aplicativos DEVE ser apenas por meio do uso explícito de API. Aplicativos NÃO PODEM sincronizar-se em classes `System` ou objetos estáticos do sistema, uma vez que o comportamento entre aplicativos é imprevisível. NÃO PODE haver qualquer compartilhamento do estado de classes de aplicativo ou objetos entre ambientes de execução. *`ClassLoaders`* provêem isolamento suficiente das classes e objetos do aplicativo. Implementações do ambiente de execução NÃO PODEM sincronizar-se em objetos visíveis aos aplicativos.

Aplicativos NÃO PODEM ser inicializados se outra instância do aplicativo já estiver ativa no serviço. O sistema deve suportar multi-tarefas com pelo menos dois aplicativos rodando concorrentemente.

18.4.4 Propriedades do sistema

As proriedades do sistema definidas estão listadas na Tabela 46.

Tabela 46 – Propriedades do sistema definidas

Nome da propriedade	Tipo da propriedade	Valor da propriedade
`com.sun.dtv.version`	`String`	O número da versão do ambiente de execução do Java DTV, a versão desta especificação é "`1.0`".

18.4.5 Contexto do Xlet para cada aplicação

O ambiente de execução do Java DTV DEVE prover cada aplicativo com um `javax.microedition.xlet.XletContext` incluindo um conjunto de propriedades Java DTV específicas como descrito na Tabela 47.

Tabela 47 – Propriedades Java DTV específicas

Nome da propriedade	Tipo da propriedade	Valor da propriedade
`com.sun.dtv.persistent.root`	`String`	A raiz do sistema de arquivos Java DTV
`com.sun.dtv.orgid`	`Integer`	Identificador da organização do aplicativo atual proveniente do gerenciador de aplicativos
`com.sun.dtv.appid`	`Integer`	Identificador do aplicativo atual proveniente do gerenciador de aplicativos
`javax.microedition.xlet.XletContext.ARGS`	`String[]`	Argumentos para o aplicativo por meio do gerenciador de aplicativos como provido pelo assinalamento do aplicativo ou proveniente de uma invocação de `ApplicationProxy.start()`

18.4.6 Classes e *classpath*

O JAR do aplicativo deve ser a primeira entrada no caminho da classe para o aplicativo. Todas as classes e recursos que ele contém são disponibilizadas no ambiente de execução.

O atributo `Class-Path` na seção principal do manifesto contém uma lista de arquivos JAR adicionais. Cada um dos arquivos JAR deve ser adicionado no caminho da classe. Todas as classes e recursos que ele contém são disponibilizadas no ambiente de execução.

18.4.7 Processos, *threads* e grupos de *threads*

Cada aplicativo é iniciado no seu próprio grupo de *threads*. O grupo de *threads* DEVE ter seu `maxPriority` inicializado para `java.lang.Thread.NORM_PRIORITY`.

18.4.8 Locale

O suporte para *locale* dentro do ambiente de execução considera que:

- DEVE suportar o *locale* `EN.UK`
- PODE suportar qualquer número de outros *locale*
- o *locale* padrão é `pt_BR`

18.4.9 Codificação de textos

Suporte para codificação de textos considera que:

- DEVE suportar UTF-8 como definido por `DataOutput.writeUTF()`
- DEVE ser suportado Latin1 como definido pela ISO 8859-1
- PODE suportar codificações de texto adicionais

18.4.10 Ciclo de vida de aplicações JavaDTV

A execução de um aplicativo é gerenciada pelo gerenciador de aplicativos e é coordenada com o aplicativo usando interfaces `Xlet` e `XletContext` definidas em `javax.microedition.xlet package`.

O gerenciador de aplicativos DEVE definir o *classpath* antes do aplicativo ser iniciado para incluir o arquivo JAR contendo o aplicativo. Para todo arquivo JAR, incluindo o JAR principal, o manifesto DEVE ser examinado e para cada atributo `Class-Path` o arquivo JAR referenciado deve ser incluído. Arquivos JAR são examinados em detalhes antes de adicionar quaisquer JAR referenciados. Cada JAR DEVE aparecer apenas uma vez no *classpath* ao ser referenciado pela primeira vez. No runtime o aplicativo usa o `URLClassLoader` para especificar locais (por meio de localizadores) de onde as classes podem ser carregadas dinamicamente.

O aplicativo DEVE prover uma classe implementando a interface javax.microedition.xlet.Xlet e identificar essa classe como o ponto de entrada do aplicativo usando o atributo `Main-Class` no manifesto. Para inicializar o aplicativo, uma instância da classe é criada usando o construtor e no método `initXlet` é passado o `XletContext`. O `XletContext` tem informação do contexto do ambiente de execução para o aplicativo, incluindo propriedades e mecanismos para relatar mudanças de estado inciadas pelo aplicativo. Os estados de aplicativo carregado, pausado, ativo e destruído são completamente descritos pela especificação do pacote `javax.microedition.xlet package`.

O pacote `javax.tv.xlet` está obsoleto. O pacote javax.microedition.xlet é usado para gerenciamento de aplicativos de forma que o IXC possa ser usado sem duplicar as API ou funcionalidades. Todas as instâncias de `javax.microedition.xlet.XletContext` DEVEM ser também implementar `javax.tv.xlet.XletContext`. As interfaces devem possuir métodos comuns e comportamentos prescritos. Isso possibilita que aplicativos usem a instância com um lançamento explícito para `javax.tv.xlet.XletContext` em chamadas para `javax.tv.service.selection.ServiceContextFactory.getServiceContext(xletContext)`..

Aplicativos Java DTV devem fazer uma limpeza antes de saírem, tanto quando `Xlet.destroyXlet` é chamado quanto antes do aplicativo chamar `XletContext.NotifyDestroyed`.

Os seguintes passos devem ser seguidos:

- cada *thread* criado pelo aplicativo deve terminar;
- cada *player* JMF criado pelo aplicativo deve ser parado e fechado;
- cada JavaTV `ServiceContext` criado pelo aplicativo deve ser parado e fechado;
- cada `ScarceResource` reservado pelo aplicativo deve ser liberado;
- cada imagem com dados não liberados deve ser liberada;
- o aplicativo deve fazer a limpeza e retornar de `Xlet.destroyXlet` tão rápido quanto possível;
- cada *listener* de evento registrado deve ser removido.

Independente do aplicativo Java DTV ter feito a limpeza requerida quando um Xlet é destruído, é responsabilidade da implementação liberar todos `ScarceResources` e outros recursos, recursos relacionados a mídias, por exemplo, usados pelo Xlet, e notificar outros aplicativos recebendo esses recursos usando as API e *listeners* definidos em cada API.

18.4.11 Comunicação inter-aplicativo

Os aplicativos Java DTV podem comunicar-se uns com os outros usando este pacote. A comunicação de um aplicativo para o outro é estabilizada ao ligar um objeto a um nome no `IxcRegistry` e outro aplicativo procurando o nome e invocando os métodos do objeto. O domínio é estruturado para permitir acesso comum a todos os nomes definidos por aplicativos assinados e outros e ainda mais todos os nomes definidos para uma organização e todos os nomes definidos por um aplicativo. Cada aplicativo é provido de uma parte do domínio IXC para si, usando o `appId` que consiste de um identificador de organização e um identificador de aplicativo. A construção dos nomes é auxiliada usando o método `Application.getIxcFullyQualifiedName`.

O Java DTV define a estrutura de nomes IXC para aplicativos para facilitar comunicações seguras:

- seguro: aplicativos assinados podem se comunicar por meio do domínio `/dtv/signed/<appId>/`. Cada aplicativo pode ligar seus próprios objetos na sua parte do domínio e pode consultar objetos de qualquer aplicativo assinado;
- não-seguro: qualquer aplicativo pode se comunicar por meio do domínio `/dtv/ixc/<appId>/`. Cada aplicativo não assinado pode ligar seus próprios objetos na sua parte do domínio e pode consultar objetos

de qualquer aplicativo.

Essa estrutura permite que cada aplicativo comunique-se seguramente com seus pares por uma maneira controlada. O acesso as partes do domínio é controlado pelas políticas de aplicações assinadas e não-assinadas.

18.5 Armazenamento persistente

O armazenagem persistente pode ser acessível a aplicativos. Os controles de acesso e estrutura estão definidos em 14 (Armazenamento Persistente de Arquivos). A raiz do sistema de arquivos se encontra na propriedade de sistema `com.sun.dtv.persistent.root`. O acesso a arquivos abaixo da raiz é ditado pela política de segurança. O acesso a arquivos acima da raiz DEVE causar o lançamento de um `java.lang.SecurityException`.

As API `FileProperties` podem ser usadas para mudar os direitos de acesso a arquivos que pertencem a um aplicativo.

18.6 Política de segurança recomendada para o Java DTV

A política de segurança para aplicativos Java DTV consiste em recomendações para aplicativos não-assinados e aplicativos assinados. Aplicativos assinados podem requisitar permissões adicionais usando a política por aplicativo. Os aplicativos Java DTV não são autorizados a usar as funções extendidas e API do dispositivo a menos que o aplicativo possa ser autenticado para um certificado raiz válido no dispositivo. A política de empacotamento e segurança aplica-se a todos os aplicativos Java DTV.

18.6.1 Política para aplicativos assinados

A política de segurança é aplicada a aplicativos assinados e é combinada a política por aplicativo apenas se o indivíduo for provido de um aplicativo assinado e autenticado. Os detalhes desta política estão listados na Tabela 48.

Tabela 48 – Política para aplicativos assinados

Permissão	Descrição
`java.util.PropertyPermission(<propriedades de sistema definidas pelo Java DTV>, "read")`	Permite acesso de leitura às propriedades de sistema definidas pelo Java DTV pela `java.lang.System.getProperty`
`java.io.FilePermission(BroadcastFilesystem.getBroadcastRoot(), "read")`	Permite acesso de leitura ao sistema de arquivos de transmissão
`javax.tv.media.MediaSelectPermission("*")`	Permite acesso a seleção de novos clipes de mídia do *stream* de transporte atual
`javax.tv.service.ReadPermission("*")`	Permite acesso a outros serviços
`javax.tv.service.selection.ServiceContextPermission("getServiceContentHandlers", "own")`	Permite acesso a todos os manuseadores de conteúdo para o serviço do próprio aplicativo
`javax.tv.service.selection.ServiceContextPermission("access", "own")`	Permite acesso ao serviço do próprio aplicativo. Restringe acesso a outros serviços
`Não permite java.awt.AWTPermission`	Restringe o controle de baixo nível pelo AWT
`Não permite java.util.PropertyPermission("user.language", "write")`	Restringe mudança do *locale*
`Não permite java.lang.RuntimePermission`	Restringe ações do ambiente de execução
`Não permite java.io.SerializablePermission`	Restringe comportamento de serialização

	primordial

Para o ambiente de execução do Java DTV a política por aplicação é mapeada para a política de segurança do Java DTV. Se o elemento da política estiver presente, a(s) permissão(ões) correspondente(s) descrita(s) na Tabela 49 estarão incluídas na política, com quaisquer parâmetros presentes no elemento.

Tabela 49 – Permissões correspondentes aos elementos da política

Elemento da política DTV	Permissão	Descrição
`file`	`FilePermission(<ownerfiles>, "read,write")`	`ownerfiles` é a concatenação da propriedade `com.sun.dtv.persistent.root` com o valor da propriedade `com.sun.dtv.orgid` separados pelo delimitador apropriado do sistema de arquivos
`applifecycle`	`AppManagerPermission("*", "read,manage")`	Permite acesso para gerenciar todos os aplicativos no serviço atual
`networkdevice`	`NetworkDevicePermission("networkdevice.any", "dial=tel")` `ScarceResourcePermission("networkdevice.any", "reserve")`	Permite acesso ao dispositivo de rede, a rede padrão e todos os números de telefone
`dripfeed`	`DripFeedPermission("dripfeed://")`	Permite acesso ao controle de *dripfeed*
`scarceresource`	`ScarceResourcePermission(name, actions)`	Permite acesso aos recursos escassos nomeados com ação(ões)
`tuning`	`ScarceResourcePermission("tuner.any", "reserve")`	Permite acesso a todos os sintonizadores
`smartcardaccess`	`javax.microedition.apdu.APDUPermission("aid")`	Permite acesso a API de *smartcard*
`serviceselect`	`SelectPermission("*", "*")`	Permite acesso a todos os serviços
`userproperty`	`UserPropertyPermission("*", "read,remove,write")`	Permite acesso as ações de propriedades do usuário ler, remover e escrever
`network`	`SocketPermission(host, actions)`	Permite acesso para cada hospedeiro requisitado
`ixcaccess`	`IxcPermission("/dtv/signed/<appId>/*", "bind")`	Permite ligar nomes no escopo do domínio <`appID`> dos aplicativos assinados
`ixcaccess`	`IxcPermission("/dtv/signed/*", "lookup")`	Permite consulta de nomes no escopo do domínio de qualquer aplicativo assinado
`ixcaccess`	`IxcPermission("/dtv/ixc/<appId>/*", "bind")`	Permite ligar nomes no escopo não-seguro do domínio <`appID`>
`ixcaccess`	`IxcPermission("/dtv/ixc/*", "lookup")`	Permite consulta de nomes no escopo não-seguro do domínio
`persistentfilecredential`	`FilePermission(filename, read/write mode)`	Permite acesso a cada nome de arquivo e modo de leitura ou escrita

18.6.2 Política para aplicativos não-assinados

A Tabela 50 descreve a política de segurança que se aplica a aplicativos não-assinados.

Tabela 50 – Política para aplicativos não-assinados

Permissão	Descrição
java.util.PropertyPermission(<propriedades de sistema definidas pelo Java DTV>*, "read")*	Permite acesso de leitura às propriedades de sistema definidas pelo Java DTV
java.io.FilePermission(BroadcastFilesystem.getBroadcastRoot(), "read")	Permite acesso de leitura ao sistema de arquivos de transmissão
javax.tv.media.MediaSelectPermission("")*	Permite acesso a seleção de novos clipes de mídia do *stream* de transporte atual
javax.tv.service.ReadPermission("")*	Permite acesso a todos os serviços
javax.tv.service.selection.ServiceContextPermission("getServiceContentHandlers", "own")	Permite acesso a todos os manuseadores de conteúdo para o serviço do próprio aplicativo
javax.tv.service.selection.ServiceContextPermission("access", "own")	Permite acesso ao serviço do próprio aplicativo. Restringe acesso a outros serviços.
Não permite `com.sun.dtv.media.dripfeed.DripFeedPermission`	Restringe o controle do *dripfeed*
Não permite `java.awt.AWTPermission`	Restringe o controle AWT de baixo nível
Não permite `java.net.SocketPermission`	Restringe conexões de rede
Não permite `java.util.PropertyPermission("user.language", "write")`	Restringe mudança do *locale*
Não permite `java.lang.RuntimePermission`	Restringe ações do ambiente de execução
Não permite `java.io.SerializablePermission`	Restringe comportamento de serialização primordial

18.7 Classe AppFilter

18.7.1 Descrição da Classe

`com.sun.dtv.application`

```
java.lang.Object
  └com.sun.dtv.application.AppFilter
```

```
public class AppFilter
extends Object
```

`AppFilter` é chamada para aceitar um aplicativo em um filtro ou não. O comportamento dessa classe, até ser invalidada, é selecionar aplicativos que são definidos no serviço atual, não incluindo aplicativos declarados como externos e permitidos a continuar rodando quando o serviço muda. Subclasses invalidam o método `accepts()` para aplicar seu próprio critério para aceitar aplicativos.

Relaciona-se com:

`AppManager.getActiveApplications(com.sun.dtv.application.AppFilter)`,
`AppManager.getApplications(com.sun.dtv.application.AppFilter)`

18.7.2 Índice de métodos

`boolean` **`accepts`**`(Application application)`

Determina se um Aplicativo deveria ser aceito no filtro ou não.

18.7.3 Detalhe dos métodos

accepts

`public boolean accepts(Application application)`

Determina se um aplicativo deve ser aceito no filtro ou não. Esse método retorna `true` se, e apenas se, o aplicativo estiver disponível no serviço atual.

Subclasses podem chamar `super.accepts(application)` para determinar se o aplicativo está disponível no serviço atual.

Subclasses devem suprir suas próprias condições e retornar *true* para aceitar o aplicativo no filtro.

Parâmetros:

`application` - o aplicativo a ser filtrado

Retorna:

true se o `application` estiver disponível no serviço atual, se não, *false*.

18.8 Interface Application

18.8.1 Descrição da interface

`com.sun.dtv.application`

`public interface Application`

Obtém atributos de um aplicativo. Para cada aplicativo os atributos podem ser consultados, incluindo o appId, tipo, o nome do aplicativo em um ou mais locales, a prioridade e visibilidade, o serviço, informação sobre a versão e ícone.

Relaciona-se com:

`AppManager`

18.8.2 Índice de campos

`int` **`APP_IXC_SCOPE_SECURED`**

Escopo global para comunicação inter-xlet.

`int` **`APP_IXC_SCOPE_UNSECURED`**

Escopo não-seguro para comunicação inter-xlet.

`int` **`APP_TYPE_HTML`**

Tipo de aplicativo para um aplicativo HTML.

`int` **`APP_TYPE_JAVA`**

Tipo de aplicativo para um aplicativo Java DTV.

`int` **`APP_TYPE_NCL`**

Tipo de aplicativo para NCL.

`int` **`APP_TYPE_RESIDENT`**

Tipo de aplicativo para aplicativos nativos ou residentes.

18.8.3 Índice de métodos

`String` **`getAppId`**`()`

Obtém o `appId` como uma *string*.

`int` **`getIconFlags`**`()`

Obtém as *flags* que descrevem os tamanhos disponíveis dos ícones.

`Locator` **`getIconLocator`**`()`

Obtém o localizador para o ícone.

`String` **`getIxcFullyQualifiedName`**`(String name, int scope)`

Gera uma referência qualificada para o objeto `name` em um dado domínio de aplicativo com propósito de comunicação inter-xlet (IXC) dentro de um dado escopo.

`String` **`getName`**`()`

Obtém o nome do aplicativo no locale padrão.

`String` **`getName`**`(String locale)`

Obtém o nome do aplicativo no locale requisitado.

`String[]` **`getNameLocales`**`()`

Obtém o locale para o qual o aplicativo tem nomes específicos de locale.

`int` **`getPriority`**`()`

Obtém a prioridade do aplicativo.

`String[]` **`getProfiles`**`()`

Obtém os perfis do aplicativo requeridos para rodar o aplicativo.

`Service` **`getService`**`()`

Obtém o serviço do aplicativo.

`int` **`getType`**`()`

Obtém o tipo do aplicativo.

`short[]` **`getVersion`**`(String appProfile)`

Obtém o número da versão do aplicativo no perfil requisitado.

`boolean` **`isServiceBound`**`()`

Obtém a ligação do aplicativo ao contexto do serviço.

`boolean` **`isStartable`**`()`

Obtém a *flag* iniciável para o aplicativo.

`boolean` **`isVisible`**`()`

Obtém a visibilidade para usuários do aplicativo.

18.8.4 Detalhe dos campos

APP_TYPE_JAVA

```
public static final int APP_TYPE_JAVA = 1
```

Tipo de aplicativo para um aplicativo Java DTV. O valor é `1` como definido pela ABNT NBR 15606, Tabela 45.

APP_TYPE_HTML

```
public static final int APP_TYPE_HTML = 2
```

Tipo de aplicativo para um aplicativo HTML. O valor é `2` como definido pela ABNT NBR 15606, Tabela 45.

APP_TYPE_NCL

```
public static final int APP_TYPE_NCL = 9
```

Tipo de aplicativo para NCL. O valor é `9` como definido pela ABNT NBR 15606-3, Tabela 45.

APP_TYPE_RESIDENT

```
public static final int APP_TYPE_RESIDENT = 32768
```

Tipo de aplicativo para aplicativos nativos ou residentes. O valor é `32768`.

APP_IXC_SCOPE_SECURED

```
public static final int APP_IXC_SCOPE_SECURED = 1
```

Escopo global para comunicação inter-xlet. Neste escopo, comunicação inter-xlet é restrita a aplicativos da mesma classe: Aplicativos assinados só podem compartilhar objetos com outros aplicativos assinados e aplicativos não-assinados só podem compartilhar objetos com outros aplicativos não-assinados.

Relaciona-se com:

`getIxcFullyQualifiedName()`

APP_IXC_SCOPE_UNSECURED

```
public static final int APP_IXC_SCOPE_UNSECURED = 2
```

Escopo não-seguro para comunicação inter-xlet. Isso permite que os aplicativos se comuniquem com qualquer outro aplicativo. Isso inclui também aplicativos assinados comunicando-se com aplicativos não-assinados.

Relaciona-se com:

`getIxcFullyQualifiedName()`

18.8.5 Detalhe dos métodos

getIconLocator

```
Locator getIconLocator()
```

Obtém o localizador para o ícone.

Retorna:

Obtém o localizador para os ícones.

getIconFlags

`int getIconFlags()`

Obtém as *flags* que descrevem os tamanhos disponíveis dos ícones.

Retorna:

Um inteiro com *flags* para cada tamanho disponível de ícone. Consultar a ABNT NBR 15606-3, Tabela 54, para descrição dos bits das *flags* de ícone.

getAppId

`String getAppId()`

Obtém o `appId` como uma *string*.

Retorna:

O identificador do aplicativo como uma *string*.

Relaciona-se com:

`AppManager.makeApplicationId(int, int)`

isStartable

`boolean isStartable()`

Obtém a *flag startable* para o aplicativo.

Retorna:

`true` se o aplicativo puder ser iniciado, se não, *`false`*.

isServiceBound

`boolean isServiceBound()`

Obtém a ligação do aplicativo ao contexto do serviço.

Retorna:

`true` se o aplicactivo estiver ligado ao serviço, se não, *`false`*.

getProfiles

`String[] getProfiles()`

Obtém os perfis do aplicativo requeridos para rodar o aplicativo.

Retorna:

Um *array* dos perfis requeridos por esse aplicativo; retorna um *array* de comprimento 0, se não houver nenhum.

getVersion

`short[] getVersion(String appProfile)`

Obtém o número da versão do aplicativo no perfil requisitado.

Parâmetros:

`appProfile` – o perfil para o qual obter a versão do aplicativo.

Retorna:

Um *array* de três elementos, contendo os números de versão *major*, *minor* e *micro*.

getService

```
Service getService()
```

Obtém o serviço do aplicativo. O serviço é o serviço do qual o aplicativo foi reavido.

Retorna:

O serviço de que esse aplicativo faz parte pode ser `null`.

getName

```
String getName()
```

Obtém o nome do aplicativo no *locale* padrão.

Retorna:

O nome do aplicativo no *locale* padrão.

getName

```
String getName(String locale)
```

Obtém o nome do aplicativo no *locale* requisitado.

Parâmetros:

`locale` – o *locale* no qual obter o nome do aplicativo.

Retorna:

O nome do aplicativo no *locale* requisitado, `null` se não estiver disponível.

getNameLocales

```
String[] getNameLocales()
```

Obtém o *locale* para o qual o aplicativo tem nomes específicos de locale.

Retorna:

um *array* dos *locales* para os quais o aplicativo tem nomes localizados, se não houver nenhum, um *array* de comprimento zero é retornado.

getPriority

```
int getPriority()
```

Obtém a prioridade do aplicativo.

Retorna:

o valor do aplicativo: o valor é maior que ou igual a zero e menor que ou igual a 255.

isVisible

```
boolean isVisible()
```

Obtém a visibilidade para usuários do aplicativo.

Retorna:

Visibilidade do aplicativo: `true` se o aplicativo for visível ao usuário, se não, *`false`*.

getType

```
int getType()
```

Obtém o tipo do aplicativo.

Retorna:

tipo do aplicativo: um dos tipos de aplicativo `APP_TYPE_JAVA`, `APP_TYPE_NCL`, `APP_TYPE_RESIDENT`, ou da ABNT NBR 15606-3, Tabela 45.

getIxcFullyQualifiedName

```
String getIxcFullyQualifiedName(String name,
                                int scope)
                         throws NullPointerException,
                                IllegalArgumentException,
                                IllegalStateException
```

Gera uma referência qualificada para o objeto `name` em um dado domínio de aplicativo com propósito de comunicação inter-xlet (IXC) dentro de um dado escopo. Consultar a seção 18.4.11 (Comunicação inter-aplicativo) para mais detalhes. Essa referência pode ser usada como argumento ao chamar quaisquer métodos da classe `IxcRegistry`.

Dependendo do valor de `scope`, a implementação gera o localizador de recurso correspondente.

`APP_IXC_SCOPE_SECURED`: usado ao compartilhar objetos entre aplicativos assinados. Neste escopo, se o aplicativo for não-assinado, esse método lança um `IllegalStateException`. Se o aplicativo é assinado e verificado como tal, o valor return é: "`/dtv/signed/`*`appId`*`/name`".

`APP_IXC_SCOPE_UNSECURED`: usado ao compartilhar objetos entre quaisquer tipos de aplicativos independentemente deles serem assinados ou não. Neste escopo, a referência qualificada IXC é definida como se segue:

"`/dtv/ixc/`*`appId`*`/name`".

Onde `appId` é o valor conforme com o valor retornado por `getAppId()`.

Um aplicativo com intenção de construir uma referência para um objeto pertencente a outro aplicativo pode proceder como se segue:

```
String appid = ... a              aplicação              comunica              com;
String fqn = AppManager
             .getAppManager()
             .getApplication(appid)
             .getIxcFullyQualifiedName("obj", APP_IXC_SCOPE_SECURED);
```

A comunicação ocorre como se segue:

```
IxcRegistry ixc = IxcRegistry.getRegistry(xletContext);
ixc.bind(fqn, obj);
```

Parâmetros:

`name` - o nome identificando o objeto que é comparilhado.

`scope` - qualquer uma das constantes `APP_IXC_SCOPE_` definidas por essa interface.

Retorna:

A referência qualificada IXC para `name`.

Lança:

`NullPointerException` - Se o nome for `null`.

`IllegalArgumentException` - Se o escopo não é nenhuma das constantes começando com `APP_IXC_SCOPE_` definidas por essa interface.

`IllegalStateException` - Se o escopo é estabelecido para `APP_IXC_SCOPE_SECURED` embora o aplicativo não seja assinado (ou verificado como tal).

Relaciona-se com:

`javax.microedition.xlet.ixc.IxcRegistry`

18.9 Interface ApplicationProxy

18.9.1 Descrição da interface

`com.sun.dtv.application`

`public interface ApplicationProxy`

Controle um aplicativo por meio do gerenciador de aplicativos. O gerenciador de aplicativos de um dispositivo controla os aplicativos rodando no dispositivo. Essa interface permite que um aplicativo com as permissões apropriadas monitore aplicativos rodando, que inicie um aplicativo e que requisite mudanças para o ciclo de vida do aplicativo.

18.9.2 Índice de campos

`int` **`ACTIVE`**

O aplicativo está ativo.

`int` **`DESTROYED`**

O aplicativo foi destruído.

`int` **`LOADED`**

O aplicativo foi carregado.

`int` **`NOT_LOADED`**

O aplicativo é conhecido pelo gerenciador de aplicativos, mas não foi carregado.

`int` **`PAUSED`**

O aplicativo foi pausado.

18.9.3 Índice de métodos

`void` **`addListener`**`(AppProxyListener listener)`

Adiciona um *listener* para mudanças no estado do aplicativo.

`int` **`getState`**`()`

Obtém o estado do aplicativo.

`void` **`pause`**`()`

Requisita que o aplicativo seja pausado.

`void` **`removeListener`**`(AppProxyListener listener)`

Remove um *listener* registrado para mudanças.

`void` **`resume`**`()`

Requisita que o aplicativo seja resumido.

`void` **`start`**`(String[] parameters)`

Requisita que o aplicativo esteja `ACTIVE` com os parâmetros especificados.

`void` **`stop`**`()`

Requisita que o aplicativo seja parado.

18.9.4 Detalhe dos campos

ACTIVE

`public static final int ACTIVE = 0`

O aplicativo está ativo. O valor é `0`.

Relaciona-se com:

`getState(),Xlet states`

DESTROYED

`public static final int DESTROYED = 1`

O aplicativo foi destruído. O valor é `1`.

Relaciona-se com:

`getState(),Xlet states`

LOADED

`public static final int LOADED = 2`

O aplicativo foi carregado. O valor é `2`.

Relaciona-se com:

`getState(),Xlet states`

PAUSED

`public static final int PAUSED = 3`

O aplicativo foi pausado. O valor é `3`.

Relaciona-se com:

`getState(),Xlet states`

NOT_LOADED

`public static final int NOT_LOADED = 4`

O aplicativo é conhecido pelo gerenciador de aplicativos, mas não foi carregado. O valor é `4`.

Relaciona-se com:

`getState()`

18.9.5 Detalhe dos métodos

start

```
void start(String[] parameters)
```

Requisita que o aplicativo esteja `ACTIVE` com os parâmetros especificados. O número de parâmetros pode ser zero ou mais. O aplicativo é requisitado a ser iniciado se já não estiver ativo. Se o aplicativo alvo é um aplicativo Java DTV, os argumentos DEVEM ser disponibilizados como o `javax.microedition.xlet.XletContext.ARGS`.

Parâmetros:

`parameters` - o *array* de parâmetros a serem passados ao aplicativo; `null` DEVE ter o mesmo comportamento que passar um *array* de comprimento zero.

Lança:

`IllegalStateException` - se o aplicativo já estiver `ACTIVE`

stop

```
void stop()
```

Requisita que o aplicativo seja parado.

Lança:

`IllegalStateException` - se o aplicativo não estiver `ACTIVE`

pause

```
void pause()
```

Requisita que o aplicativo seja pausado.

Lança:

`IllegalStateException` - se o aplicativo não estiver `ACTIVE`

resume

```
void resume()
```

Requisita que o aplicativo seja resumido.

Lança:

`IllegalStateException` - se o aplicativo não estiver `PAUSED`.

getState

```
int getState()
```

Obtém o estado do aplicativo.

Retorna:

um dos `LOADED, PAUSED, ACTIVE, DESTROYED,` ou `NOT_LOADED`.

addListener

```
void addListener(AppProxyListener listener)
```

Adiciona um *listener* para mudanças no estado do aplicativo.

Parâmetros:

`listener` - o *listener* a ser adicionado.

Relaciona-se com:

`removeListener()`

removeListener

```
void removeListener(AppProxyListener listener)
```

Remove um *listener* registrado para mudanças. Se o *listener* não for adicionado, nenhuma ação é tomada.

Parâmetros:

`listener` – o *listener* a ser removido.

Relaciona-se com:

`addListener()`

18.10 Classe AppManager

18.10.1 Descrição da Classe

`com.sun.dtv.application`

```
java.lang.Object
  └com.sun.dtv.application.AppManager
```

```
abstract public class AppManager
extends Object
```

`AppManager` provê controle de e acesso a aplicativos. O `AppManager` pode listar os aplicativos conhecidos pelo ambiente de execução do Java DTV tanto ativos quanto inativos. O monitoramento de mudanças nos aplicativos disponíveis e em seus atributos é por meio do `AppManagerListener` usado no método `addListener`.

Para cada aplicativo vários atributos podem ser consultados usando a classe Application, incluindo o identificador do aplicativo, o localizador usado para reaver o aplicativo, o nome do aplicativo em um ou mais *locales*, a prioridade e visibilidade, informação sobre a versão e ícones. O método `getApplications` é usado neste caso.

Aplicativos podem ser controlados usando a classe `ApplicationProxy`, provendo métodos para iniciar, parar, pausar e resumir o aplicativo por meio do método `getActiveApplications`. O monitoramento de aplicativos é por meio do `AppProxyListener` por meio do método `ApplicationProxy.addListener`.

Essa API tem segurança *multi-thread*, significando que a API pode ser chamada por muitas *threads* concurrentemente e não ocorre corrupção do banco de dados do aplicativo.

18.10.2 Exemplo de código

Exemplo usando o `AppManager` para encontrar informação sobre o aplicativo atual. O `XletContext` contém o

identificador da organização e o identificador do aplicativo que são usados para reaver a instância aplicativo.

```
import com.sun.dtv.application.AppManager;
import com.sun.dtv.application.Application;
import javax.microedition.xlet.XletContext;

    XletContext xc = ...;
    AppManager mgr = AppManager.getInstance();
    Integer aid = (Integer)xc.getXletProperty("com.sun.dtv.appid");
    Integer oid = (Integer)xc.getXletProperty("com.sun.dtv.orgid");
    String id = mgr.makeApplicationId(oid.intValue(), aid.intValue());
    Application app = mgr.getApplication(id);
```

18.10.3 Índice de métodos

`abstract void` **`addListener`**`(AppManagerListener listener)`

Adiciona um *listener* para mudanças nos aplicativos disponíveis.

`abstract Application` **`fetch`**`(String locator)`

Busca um aplicativo de um localizador.

`abstract Enumeration` **`getActiveApplications`**`()`

Retorna uma enumeração dos `AppProxys` para aplicativos ativos no serviço atual.

`abstract Enumeration` **`getActiveApplications`**`(AppFilter appFilter)`

Retorna uma enumeração dos `AppProxys` para aplicativos ativos filtrados pelo `AppFilter`.

`abstract Application` **`getApplication`**`(String appId)`

Obtém o aplicativo para o `appId` requisitado.

`abstract ApplicationProxy` **`getApplicationProxy`**`(String appId)`

Obtém o Proxy do aplicativo para o aplicativo requisitado.

`abstract Enumeration` **`getApplications`**`()`

Retorna uma enumeração dos aplicativos disponíveis no serviço atual.

`abstract Enumeration` **`getApplications`**`(AppFilter appFilter)`

Retorna uma enumeração dos aplicativos filtrados pelo `AppFilter`.

`static AppManager` **`getInstance`**`()`

Obtém o *singleton* `AppManager`.

`abstract String` **`makeApplicationId`**`(int organization, int applicationid)`

Faz um `appId` para o aplicativo com a identificador da organização e identificador do aplicativo indicados.

`abstract void` **`removeListener`**`(AppManagerListener listener)`

Remove um *listener* registrado para mudanças nos aplicativos disponíveis.

`abstract void` **`unload`**`(String appId)`

Descarrega o aplicativo, liberando quaisquer recursos que possam ser adquiridos, se necessário.

18.10.4 Detalhe dos métodos

getInstance

`public static AppManager getInstance()`

Obtém o *singleton* `AppManager`.

Retorna:

A instância *singleton* `AppManager`.

Lança:

`SecurityException` - se o Aplicativo não tem `AppManagerPermission("*", "read")`.

getApplications

`public abstract Enumeration getApplications()`

Retorna uma enumeração dos aplicativos disponíveis no serviço atual. Os resultados são os mesmos que:

`getApplications(new AppFilter)`.

Retorna:

Uma enumeração de `Application`s.

getActiveApplications

`public abstract Enumeration getActiveApplications()`

Retorna uma enumeração dos `AppProxys` para aplicativos ativos no serviço atual. Os resultados são os mesmos que:

`getApplications(new AppFilter)`

Retorna:

Uma enumeração de `ApplicationProxys`.

Lança:

`SecurityException` - se o aplicativo não tem `AppManagerPermission("*", "manage")`.

getApplications

`public abstract Enumeration getApplications(AppFilter appFilter)`

Retorna uma enumeração dos aplicativos filtrados pelo `AppFilter`. Todos os aplicativos conhecidos são passados ao filtro e o filtro os aceita (usando o método `accepts`) ou não.

Parâmetros:

`appFilter` - um `AppFilter` usado para selecionar aplicativos.

Retorna:

Uma enumeração de `Applications`.

getActiveApplications

`public abstract Enumeration getActiveApplications(AppFilter appFilter)`

Retorna uma enumeração dos `AppProxys` para aplicativos ativos filtrados pelo `AppFilter`. Todos os aplicativos conhecidos são passados ao filtro e o filtro os aceita (usando o método `accepts`) ou não.

Parâmetros:

`appFilter` - um `AppFilter` usado para selecionar aplicativos.

Retorna:

Uma enumeração de `Applications`.

Lança:

`SecurityException` - se o aplicativo não tem `AppManagerPermission("*", "manage")`.

makeApplicationId

```
public abstract String makeApplicationId(int organization,
                                         int applicationid)
```

Faz um `appId` para o aplicativo com o identificador da organização indicado e o identificador do aplicativo. O identificador da organização e o identificador do aplicativo são concatenados para formar um valor de 48 bits não assinado. Todos os 32 bits do identificador da organização são usados e se tornam os bits mais significantes do resultado. Apenas os 16 bits menos significantes do identificador do aplicativo são usados e se tornam os bits menos significantes do resultado. O valor é convertido em uma *string* hexadecimal usando apenas números e letras em caixa baixa ('0'..'9','a'..'f'). A *string* NÃO PODE ter nenhum caractere '0' à esquerda.

Parâmetros:

`organization` - um identificador de organização, como um int

`applicationid` – um identificador de aplicativo, como um int.

Retorna:

uma *string* formatada apropriadamente com o identificador da organização e o identificador do aplicativo.

getApplication

```
public abstract Application getApplication(String appId)
```

Obtém o aplicativo para o `appId` requisitado.

Parâmetros:

`appId` - a *string* `appId`

Retorna:

o aplicativo com o `appId` correspondente; caso contrário `null`.

Relaciona-se com:

`makeApplicationId(int, int)`

getApplicationProxy

```
public abstract ApplicationProxy getApplicationProxy(String appId)
```

Obtém o Proxy do aplicativo para o aplicativo requisitado.

Parâmetros:

`appId` - a *string* `appId`

Retorna:

o `ApplicationProxy` de um aplicativo com o `appId` correspondente; caso contrário `null`.

Lança:

`SecurityException` - se o Aplicativo não tem `AppManagerPermission("*", "manage")`.

Relaciona-se com:

`makeApplicationId(int, int)`

fetch

```
public abstract Application fetch(String locator)
                           throws IOException
```

Busca um aplicativo de um localizador. O método bloqueia a *thread* de chamadas até que haja suficiente informação disponível para o aplicativo. O aplicativo é inscrito no banco de dados. Chamadas para `fetch` são sincronizadas internamente para que o aplicativo seja buscado apenas uma vez e o mesmo objeto do aplicativo é retornado.

Parâmetros:

`locator` – referenciando um aplicativo.

Retorna:

a referência do aplicativo.

Lança:

`IOException` - se tentativas de buscar o aplicativo resultarem em uma exceção.

`SecurityException` - se o aplicativo não tem `AppManagerPermission("*", "fetch")`, se o aplicativo não tem permissão para usar o localizador para buscar o aplicativo.

unload

```
public abstract void unload(String appId)
```

Descarrega o aplicativo, liberando quaisquer recursos que possam ser adquiridos, se necessário. Se o aplicativo estiver ativo, o descarregamento é deferido até que o aplicativo não esteja mais ativo. A requisição de descarregamento é cancelada se o aplicativo for reiniciado.

Parâmetros:

`appId` - o appId do aplicativo para descarregar

Lança:

`SecurityException` - se o Aplicativo não tem `AppManagerPermission("*", "manage")`.

addListener

```
public abstract void addListener(AppManagerListener listener)
```

Adiciona um *listener* para mudanças nos aplicativos disponíveis.

Parâmetros:

`listener` - o *listener* a ser adicionado.

Relaciona-se com:

`removeListener()`

removeListener

```
public abstract void removeListener(AppManagerListener listener)
```

Remove um *listener* registrado para mudanças nos aplicativos disponíveis. Se o *listener* não for adicionado, nenhuma ação é tomada.

Parâmetros:

`listener` – o *listener* a ser removido.

Relaciona-se com:

addListener()

18.11 Interface AppManagerListener

18.11.1 Descrição da Interface

18.11.2 com.sun.dtv.application

public interface AppManagerListener

Listener a ser notificado de mudanças em aplicativos disponíveis.

Relaciona-se com:

AppManager.addListener()

18.11.3 Índice de métodos

void **appAdded**(Application app)

Chamada quando um novo aplicativo está disponível

void **appChanged**(Application app)

Chamada quando os atributos de aplicativo mudam.

void **appRefresh**()

Chamada quando toda a lista de aplicativos disponíveis muda.

void **appRemoved**(Application app)

Chamada quando um aplicativo é removido.

18.11.4 Detalhe dos métodos

appAdded

void appAdded(Application app)

Chamada quando um novo aplicativo está disponível

Parâmetros:

app - o novo aplicativo.

appRemoved

void appRemoved(Application app)

Chamada quando um aplicativo é removido.

Parâmetros:

app - o aplicativo que foi removido.

appChanged

void appChanged(Application app)

Chamada quando os atributos de aplicativo mudam.

Parâmetros:

app - o aplicativo com os atributos mudados.

appRefresh

`void appRefresh()`

Chamado quando toda a lista de aplicativos disponíveis muda. *Listeners* individuais adicionados/removidos/mudados não podem ser chamados.

18.12 Classe AppManagerPermission

18.12.1 Descrição da Classe

`com.sun.dtv.application`

```
java.lang.Object
  └java.security.Permission
      └com.sun.dtv.application.AppManagerPermission
```

Todas as interfaces implementadas

`Guard, Serializable`

```
public class AppManagerPermission
extends Permission
```

`AppManagerPermission` é necessário para informar-se sobre ou controlar aplicativos. Para consultar os aplicativos disponíveis, a ação "*`read`*" é requerida. Para gerenciar o ciclo de vida de outro aplicativo, a ação "*`manage`*" é requerida. Para buscar novos aplicativos, a ação "*`fetch`*" é requerida.

18.12.2 Índice de construtores

`AppManagerPermission``(String name, String actions)`

Constrói uma nova `AppManagerPermission` com o recurso e ações requisitados.

18.12.3 Índice de métodos

`boolean equals(Object obj)`

Verifica se há igualdade em dois objetos `AppManagerPermission`.

`String getActions()`

Retorna a "representação canônica" da *string* das ações.

`int hashCode()`

Retorna o valor do código *hash* para esse objeto.

`boolean implies(Permission p)`

Verifica se esse objeto `AppManagerPermission` “implica” a permissão especificada.

18.12.4 Detalhe dos construtores

AppManagerPermission

```
public AppManagerPermission(String name,
```

```
                                String actions)
```

Constrói uma nova `AppManagerPermission` com o recurso e ações requisitados. O nome do recurso é o `appId` do aplicativo a ser gerenciado. Pode ser "*" para permitir acesso a todos aplicativos.

Parâmetros:

`name` - o nome do aplicativo para se permitir acesso ou "*" para permitir acesso a todos aplicativos.

`actions` – contém uma lista separada por vírgulas das ações desejadas concedidas. Ações possíver são *`read`*, *`fetch`*, e *`manage`*.

18.12.5 Detalhe dos métodos

getActions

```
public String getActions()
```

Retorna a "representação canônica" da *string* das ações. Ou seja, esse método sempre retorna ações presentes na seguinte ordem: *`read, fetch, manage`*. Por exemplo, se o objeto `AppManagerPermission` permite tanto ações *`manage`*, quanto *`fetch`* e *`read`*, uma chamada para `getActions` irá retornar a *string* "*`read,fetch,manage`*".

Substituições:

`getActions` na classe `Permission`

Retorna:

as ações em representação da *string* canônica.

equals

```
public boolean equals(Object obj)
```

Verifica se há igualdade em dois objetos `AppManagerPermission`. Verifica que *obj* é uma `AppManagerPermission` e tem o mesmo nome e ações que esse objeto.

Substituições:

`equals` na classe `Permission`

Parâmetros:

`obj` - o objeto a ser testado para detectar igualdade com esse objeto.

Retorna:

`true` se obj é uma `AppManagerPermission` e tem o mesmo nome e as mesmas ações que esse objeto `AppManagerPermission`.

hashCode

```
public int hashCode()
```

Retorna o valor do código *hash* para esse objeto. O código *hash* usado é o código *hash* do nome dessas permissões mais o código *hash* das ações, isto é, `getName().hashCode() + getActions.hashCode()`, onde getName é proveniente da superclasse permissão e getActions está nessa classe.

Substituições:

`hashCode` na classe `Permission`

Retorna:

um valor de código *hash* para esse objeto.

implies

`public boolean implies(Permission p)`

Verifica se esse objeto `AppManagerPermission` “implica” na permissão especificada. Mais especificamente, esse método retorna true se:

"p" é uma instância de `AppManagerPermission`, e

as ações de "p" são um subconjunto das ações do objeto, e

o nome desse objeto é igual a "*" ou o nome de "p" é igual ao nome desse objeto.

Substituições:

`implies` na classe `Permission`

Parâmetros:

`p` - a permissão a ser verificada.

Retorna:

`true` se essa permissão "implicar" na permissão "`p`".

18.13 Interface AppProxyListener

18.13.1 Descrição da interface

`com.sun.dtv.application`

`public interface AppProxyListener`

Listener a ser notificado de mudanças no estado dos aplicativos.

Relaciona-se com:

`ApplicationProxy`

18.13.2 Índice de métodos

`void appStateChange(Application app, int oldstate, int newstate, boolean failed)`

Chamada quando um aplicativo muda de estado.

18.13.3 Detalhe dos métodos

appStateChange

```
void appStateChange(Application app,
                    int oldstate,
                    int newstate,
                    boolean failed)
```

Chamada quando um aplicativo muda de estado.

Parâmetros:

`app` - o aplicativo mudando de estado.

`oldstate` - o estado anterior.

`newstate` - o novo estado.

`failed` - `true` se a requisição de mudança de estado falhar, caso contrário *`false`*.

19 Pacote com.sun.dtv.broadcast

19.1 Descrição do Pacote

Provê acesso a arquivos de transmissão e *streams* por meio de API que trabalham em conjunção com o pacote `java.io`. A API Java DTV modela transmissões como discos de sistema de arquivos convencionais somente leitura com grandes latências de acesso. Para limitar as latências de acesso para um número mínimo é fortemente recomendado que todos os arquivos de transmissão recebidos sejam registrados em cache. A maioria das interações com os protocolos de transporte específicos são manuseadas pela implementação da API Java TV em vez de pelo aplicativo. A classe `BroadcastFilesystem` autentica seu conteúdo como definido pelo modelo de conteúdo e aplicação. Se a informação de autenticação estiver disponível para um arquivo, o arquivo DEVE ser autenticado antes de se tornar visível por meio da API `BroadcastFile`. Se a autenticação falhar, o arquivo é tratado como se ele não existisse.

A classe java.io.File representa arquivos genéricos para todos os dados, tais como arquivos e *streams*. Esclarecimentos especiais como descritos em `com.sun.dtv.broadcast.BroadcastFile` também aplicar-se-ão quando usados diretamente por meio do `java.io.File`.

A classe da API Java DTV `com.sun.dtv.broadcast.BroadcastFile` torna o `java.io.File` em subclasse para manusear acesso a arquivos de transmissão, acrescentando a habilidade de:

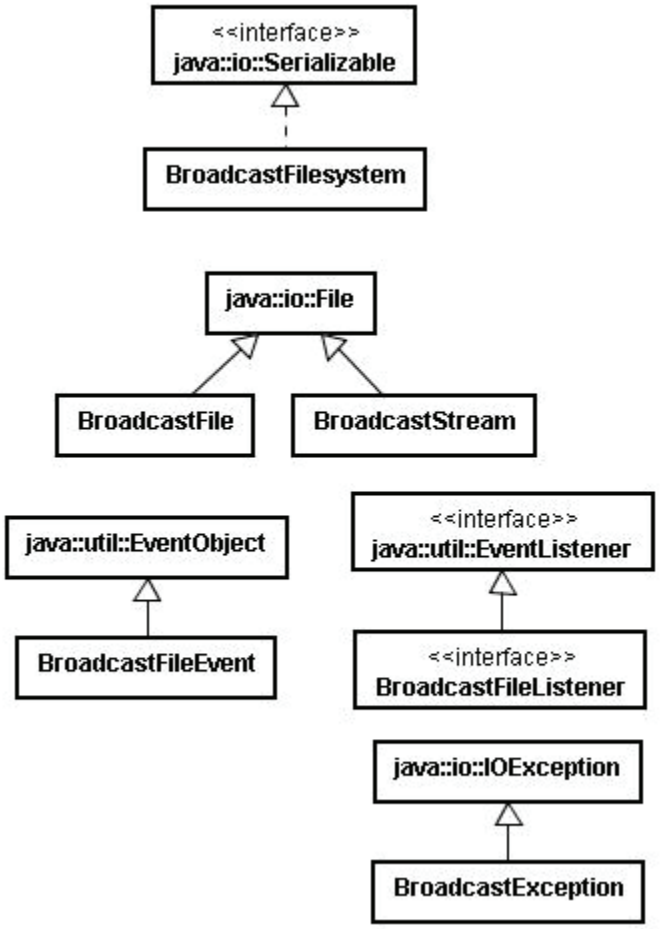
- consultar entidades em sistema de arquivos de transmissão usando instâncias de `javax.tv.locator.Locator`;
- notificar aplicativos de atualizações para diretórios individuais e arquivos na transmissão.

A classe da API Java DTV `com.sun.dtv.broadcast.BroadcastStream` torna o `java.io.File` em subclasse para manusear o acesso ao *stream* de transmissão, acrescentado habilidade de:

- obter a duração do *stream* de transmissão.

Aplicativos usam classes de entrada de arquivo convencionais do pacote (isto é, `java.io.FileInputStream`, `java.io.FileReader`, e `java.io.RandomAccessFile`) para ler de um objeto `BroadcastFile` ou `BroadcastStream`.

A Figura 5 mostra a estrutura do pacote `Broadcast` do Java DTV.



NOTA Este pacote está especificado desde o Java DTV 1.0.

Figura 5 – Pacote Broadcast

19.2 Índice de interfaces

BroadcastFileListener

A interface `BroadcastFileListener` é implementada por classes de aplicativo as quais requerem notificação das mudanças para dados `BroadcastFile`.

19.3 Índice de classes

BroadcastFile

A classe `BroadcastFile` representa dados de arquivos ou diretórios obtidos de sistemas de arquivos de transmissão.

BroadcastFileEvent

O `BroadcastFileEvent` provê notificação de mudança para o dado `BroadcastFile`.

BroadcastFilesystem

Essa classe representa instâncias de sistemas de arquivos de transmissão montados no sistema de arquivos local.

BroadcastStream

A classe `BroadcastStream` representa o *stream* obtido a partir do sistema de arquivos de transmissão.

19.4 Índice de exceções

BroadcastException

Todas as exceções relacionadas a transmissão DEVEM usar essa subclasse de `java.io.IOException` para fazer a diferente razão mais fácil de ser distinguida.

NOTA Este pacote está especificado desde o Java DTV 1.0.

19.5 Classe BroadcastException

19.5.1 Descrição da Classe

com.sun.dtv.broadcast

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └java.io.IOException
              └com.sun.dtv.broadcast.BroadcastException
```

Todas as interfaces implementadas

`Serializable`

```
public class BroadcastException
extends IOException
```

Todas as exceções relacionadas a transmissão DEVEM usar essa subclasse de `java.io.IOException` para tornar a razão diferenciada mais fácil de ser distinguida.

19.5.2 Índice de campos

`static int` **`REASON_INCONSISTENT_MESSAGE`**

Esse código razão é usado quando uma mensagem inconsistente é recebida do *stream* de transmissão.

`static int` **`REASON_INVALID_OPERATION`**

Esse código razão é usado se essa operação for inválida.

`static int` **`REASON_INVALID_PATH`**

Esse código razão é usado se o nome do caminho requisitado não existir no *stream* de transmissão.

`static int` **`REASON_NONEXISTENT_OBJECT`**

Esse código razão é usado se o objeto requisitado não existir no *stream* de transmissão.

`static int` **`REASON_OTHER`**

Esse código razão é usado se nenhum outro código razão se aplicar.

`static int` **`REASON_TIMEOUT`**

Esse código razão é usado se ocorrer temporização enquanto lendo do *stream* de transmissão.

`static int` **`REASON_UNSUPPORTED`**

Esse código razão é usado se essa operação não for suportada.

19.5.3 Índice de construtores

BroadcastException()

19.5.4 Índice de métodos

int **getReason**()

Restaura a razão dessa exceção.

19.5.5 Detalhe dos campos

REASON_INCONSISTENT_MESSAGE

```
public static final int REASON_INCONSISTENT_MESSAGE = 1
```

Esse código razão é usado quando uma mensagem inconsistente é recebida do *stream* de transmissão.

REASON_NONEXISTENT_OBJECT

```
public static final int REASON_NONEXISTENT_OBJECT = 2
```

Esse código razão é usado se o objeto requisitado não existir no *stream* de transmissão.

REASON_INVALID_PATH

```
public static final int REASON_INVALID_PATH = 3
```

Esse código razão é usado se o nome do caminho requisitado não existir no *stream* de transmissão.

REASON_TIMEOUT

```
public static final int REASON_TIMEOUT = 4
```

Esse código razão é usado se ocorrer temporização enquanto lendo do *stream* de transmissão.

REASON_UNSUPPORTED

```
public static final int REASON_UNSUPPORTED = 5
```

Esse código razão é usado se essa operação não for suportada.

REASON_INVALID_OPERATION

```
public static final int REASON_INVALID_OPERATION = 6
```

Esse código razão é usado se essa operação for inválida.

REASON_OTHER

```
public static final int REASON_OTHER = -1
```

Esse código razão é usado se nenhum outro código razão se aplicar. Informação adicional deveria ser restaurada

a partir da mensagem de texto detalhada.

19.5.6 Detalhe dos construtores

BroadcastException

```
public BroadcastException()
```

19.5.7 Detalhe dos métodos

getReason

```
public int getReason()
```

Restaura a razão dessa exceção.

Retorna:

a razão

19.6 Classe BroadcastFile

19.6.1 Descrição da Classe

com.sun.dtv.broadcast

```
java.lang.Object
  └java.io.File
      └com.sun.dtv.broadcast.BroadcastFile
public class BroadcastFile
extends File
```

A classe `BroadcastFile` representa dados de arquivos ou diretórios obtidos de sistemas de arquivos de transmissão. Se a informação de autenticação for provida no sistema de arquivos, o conteúdo do arquivo é autenticado antes de se tornar disponível para o aplicativo. Se a autenticação falhar, DEVE parecer para o aplicativo que o arquivo não existe.

Uma instância do `BroadcastFile` pode ser construída a partir de uma instância `Locator` ou através de construtores similares àqueles do `java.io.File`. Este fato também significa que todos aquels construtores utilizam o mesmo caracter para a separação de nome a caminhos, como os outros arquivos. Por outro lado o localizador com base no construtor usa localizadores possivelmente criados a partir de URL seguindo o esquema comum baseado em URL no qual os nomes dos caminhos são separados por "/".

Classes de aplicativos implementando a interface `BroadcastFileListener` podem subscrever-se com o `BroadcastFile` para receber notificação de mudanças no arquivo na transmissão. Após a ocorrência de uma mudança, o `BroadcastFile` notifica instâncias `BroadcastFileListener` subscritas por meio de objetos `BroadcastFileChangeEvent`.

Instanciação bem-sucedida de um objeto `BroadcastFile` causa a "montagem" dinâmica do seu sistema de arquivos de transmissão no sistema de arquivos local. O ponto preciso de montagem pode ser determinado chamando `getBroadcastFilesystem()`. Deve ser observado que um sistema de arquivos de transmissão também pode ser explicitamente montado usando instâncias de `BroadcastFilesystem`.

A construção de um objeto causa o carregamento assíncrono (registrado em cache) do seu conteúdo a partir do *stream* de transmissão. Tentativas subseqüentes de ler os dados de um objeto `BroadcastFile` bloquear-se-ão até que seu conteúdo esteja carregado. Isso também significa que informações sobre o arquivo (por exemplo, comprimento) podem não ser precisas até `isCached()` retornar `true`. Qualquer operação lança uma `java.io.IOException` uma vez que arquivos de transmissão são somente leitura e só podem ser atualizados por meio do canal de transmissão.

Após completar o carregamento de um arquivo para o cache do sistema, todas operações de entrada e saída

subseqüentes trabalham no arquivo carregado. Isso significa que quaisquer atualizações não aparecerão nessas operações. Atualizações podem ser requisitadas explicitamente por `refreshCache()`.

Sempre que um `BroadcastFile` é inicialmente carregado ou atualizado no cache local a partir do canal de transmissão a informação `lastModified` é atualizada até o momento em que foi carregada do canal de transmissão.

Perda da transmissão de um arquivo de transmissão resulta em todos os arquivos que já estão completos no cache continuarem acessíveis. Quaisquer operações bloqueadas no carregamento de operações causa o lançamento de uma `IOException` e a construção ou novo carregamento de arquivos.

Todos os construtores e operações de entrada e saída subseqüentes devem usar `BroadcastException` no caso de precisarem lançar exceções específicas de transmissão.

Sistemas de arquivos de transmissão para os quais não há instâncias `BroadcastFile` restantes ou descritores de arquivos abertos têm admissão para descarregar do cache. Sistemas de arquivos de transmissão para os quais não há instâncias `BroadcastFile` restantes ou descritores de arquivos abertos têm admissão para desmontarem-se do sistema de arquivos local.

Implementações Java DTV API que não suportam acesso a sistema de arquivos de transmissão lançarão `UnsupportedOperationException` após qualquer tentativa de construir um objeto `BroadcastFile`.

Todas as interfaces implementadas

`Comparable, Serializable`

19.6.2 Índice de construtores

`BroadcastFile``(File dir, String name)`

Cria uma instância `BroadcastFile` que representa o arquivo com o nome especificado no diretório de transmissão especificado.

`BroadcastFile``(String path)`

Cria uma instância `BroadcastFile` que representa o arquivo cujo nome do caminho na transmissão é o argumento do caminho.

`BroadcastFile``(String path, String name)`

Cria uma instância `BroadcastFile` que representa o arquivo com o nome especificado no diretório de transmissão especificado.

`BroadcastFile``(Locator locator)`

Cria uma instância `BroadcastFile` que representa o arquivo referenciado pelo `Locator`.

19.6.3 Índice de métodos

`void` **`addListener`**`(BroadcastFileListener listener)`

Subscreve um `BroadcastFileListener` para receber notificações de mudanças nesse `BroadcastFile`.

`BroadcastFilesystem` **`getBroadcastFilesystem`**`()`

Retorna o `BroadcastFileSystem` a que esse `BroadcastFile` pertence.

`Locator` **`getLocator`**`()`

Retorna um `Locator` identificando esse `BroadcastFile`.

`String` **`getType`**`()`

Restaura o tipo mime de um `BroadcastFile`.

`boolean` **`isCached`**`()`

Restaura independente de um `BroadcastFile` estar completamente disponível em cache do sistema de arquivos localmente no sistema.

`String[]` **`listDirectory`**`()`

Lista o conteúdo do diretório desse objeto `BroadcastFile`.

`void` **`refreshCache`**`()`

Requisita que o conteúdo em cache desse `BroadcastFile` seja atualizado com a versão atualmente no *stream* de transmissão.

`void` **`removeListener`**`(BroadcastFileListener listener)`

Retira a subscrição de um `BroadcastFileListener` para receber notificações de mudanças nesse `BroadcastFile`.

19.6.4 Detalhe dos construtores

BroadcastFile

```
public BroadcastFile(Locator locator)
              throws InvalidLocatorException,
                     IOException
```

Cria uma instância `BroadcastFile` que representa o arquivo referenciado pelo `Locator`. Esse método não inicia o carregamento de ou bloqueia arquivos sendo carregados da transmissão.

Esse construtor lança `java.io.IOException` se isso determinar imediatamente que o arquivo de transmissão requisitado não pode ser acessado. Uma vez que esse construtor pode completar seu trabalho assincronamente, a ausência de um `IOException` não é garantia de que o arquivo de transmissão requisitado esteja acessível.

Parâmetros:

`locator` - Um `Locator` referenciando a fonte do `BroadcastFile`.

Lança:

`InvalidLocatorException` - Se o localizador não consultar um arquivo de transmissão.

`IOException` - Se o arquivo de transmissão requisitado não pode ser acessado ou os dados recebidos são inválidos.

`UnsupportedOperationException` - se o sistema de arquivos de transmissão não é suportado.

BroadcastFile

```
public BroadcastFile(String path)
              throws IOException
```

Cria uma instância `BroadcastFile` que representa o arquivo cujo nome do caminho na transmissão é o argumento do caminho.

Esse construtor lança `java.io.IOException` se isso determinar imediatamente que o arquivo de transmissão requisitado não pode ser acessado. Uma vez que esse construtor pode completar seu trabalho assincronamente, a ausência de um `IOException` não é garantia de que o arquivo de transmissão requisitado esteja acessível.

Parâmetros:

`path` - o nome do caminho do arquivo.

Lança:

`IOException` - Se o arquivo de transmissão requisitado não pode ser acessado ou os dados recebidos são

inválidos.

`UnsupportedOperationException` - Se o sistema de arquivos de transmissão não é suportado.

BroadcastFile

```
public BroadcastFile(File dir,
                     String name)
              throws IOException
```

Cria uma instância `BroadcastFile` que representa o arquivo com o nome especificado no diretório de transmissão especificado.

Esse construtor lança `java.io.IOException` se isso determinar imediatamente que o arquivo de transmissão requisitado não pode ser acessado. Uma vez que esse construtor pode completar seu trabalho assincronamente, a ausência de um `IOException` não é garantia de que o arquivo de transmissão requisitado esteja acessível.

Parâmetros:

`dir` - o diretório.

`name` - o nome do arquivo.

Lança:

`IOException` - se o arquivo de transmissão requisitado não pode ser acessado ou os dados recebidos são inválidos.

`UnsupportedOperationException` - se o sistema de arquivos de transmissão não é suportado.

BroadcastFile

```
public BroadcastFile(String path,
                     String name)
              throws IOException
```

Cria uma instância `BroadcastFile` que representa o arquivo com o nome especificado no diretório de transmissão especificado.

Esse construtor lança `java.io.IOException` se isso determinar imediatamente que o arquivo de transmissão requisitado não pode ser acessado. Uma vez que esse construtor pode completar seu trabalho assincronamente, a ausência de um `IOException` não é garantia de que o arquivo de transmissão requisitado esteja acessível.

Parâmetros:

`path` - o nome do caminho do diretório.

`name` - o nome do arquivo.

Lança:

`IOException` - se o arquivo de transmissão requisitado não pode ser acessado ou os dados recebidos são inválidos.

`UnsupportedOperationException` - Se o sistema de arquivos de transmissão não é suportado.

19.6.5 Detalhe dos métodos

listDirectory

```
public String[] listDirectory()
                       throws IOException
```

Lista o conteúdo do diretório desse objeto `BroadcastFile`. Essa lista não inclui o diretório atual ou principal. Essa lista pode também estar incorreta no caso de todas as entradas nesse diretório ainda não estarem carregadas.

Retorna:

Um *array* de nomes de arquivos contidos no diretório especificado por esse objeto `BroadcastFile`. Se esse objeto `BroadcastFile` não consultar um diretório, esse método retorna `null`.

Lança:

`IOException` - Se o diretório não pode ser acessado ou os dados recebidos são inválidos.

`SecurityException` - se um gerenciador de segurança existir e seu método `SecurityManager.checkRead(String)` negar acesso de leitura ao arquivo.

addListener

```
public void addListener(BroadcastFileListener listener)
                 throws IOException
```

Subscreve um `BroadcastFileListener` para receber notificações de mudanças nesse `BroadcastFile`. Se o *listener* especificado estiver correntemente subscrito, nenhuma ação é tomada.

Parâmetros:

`listener` - o `BroadcastFileListener` a ser notificado.

Lança:

`IOException` - Se houver recursos insuficientes para suportar esse *listener*.

`SecurityException` - Se um gerenciador de segurança existir e seu método `SecurityManager.checkRead(java.lang.String)` negar acesso de leitura ao arquivo.

Relaciona-se com:

`removeListener()`

removeListener

```
public void removeListener(BroadcastFileListener listener)
```

Retira a subscrição de um `BroadcastFileListener` para receber notificações de mudanças nesse `BroadcastFile`. Se o dado `BroadcastFileListener` não estiver correntemente subscrito para notificação, nenhuma ação é tomada.

Parâmetros:

`listener` - Um `BroadcastFileListener` correntemente registrado.

Relaciona-se com:

`addListener()`

getBroadcastFilesystem

```
public BroadcastFilesystem getBroadcastFilesystem()
```

Retorna o `BroadcastFileSystem` a que esse `BroadcastFile` pertence.

Retorna:

Um `BroadcastFilesystem` a que esse `BroadcastFile` pertence.

getLocator

`public Locator getLocator()`

Retorna um `Locator` identificando esse `BroadcastFile`.

Retorna:

Um `Locator` identificando esse `BroadcastFile`.

isCached

`public boolean isCached()`

Restaura independente de um `BroadcastFile` estar completamente disponível em cache do sistema de arquivos localmente no sistema.

Retorna:

`true` se o `BroadcastFile` estiver disponível no cache do sistema de arquivos

getType

`public String getType()`

Restaura o tipo mime de um `BroadcastFile`.

Retorna:

O *mimetype* do `BroadcastFile`, `null` se não estiver disponível.

refreshCache

```
public void refreshCache()
                  throws SecurityException
```

Requisita que o conteúdo em cache desse `BroadcastFile` seja atualizado com a versão atualmente no *stream* de transmissão. Se o dado `BroadcastFile` não residir correntemente no *stream* de transporte, tentativas subseqüentes de acessar seu conteúdo falharão.

Lança:

`SecurityException` - se um gerenciador de segurança existir e seu método `SecurityManager.checkRead(java.lang.String)` negar acesso de leitura ao arquivo.

19.7 Classe BroadcastFileEvent

19.7.1 Descrição da classe

`com.sun.dtv.broadcast`

```
java.lang.Object
  └java.util.EventObject
      └com.sun.dtv.broadcast.BroadcastFileEvent
```

Todas as interfaces implementadas

`Serializable`

```
public class BroadcastFileEvent
extends EventObject
```

O `BroadcastFileEvent` provê notificação de mudança para o dado `BroadcastFile`.

19.7.2 Índice de construtores

`BroadcastFileEvent``(BroadcastFile source)`

Cria um `BroadcastFileEvent` indicando que o `BroadcastFile` especificado mudou.

19.7.3 Índice de métodos

`BroadcastFile` **`getBroadcastFile`**`()`

Relata o `BroadcastFile` cujo conteúdo mudou.

19.7.4 Detalhe dos construtores

BroadcastFileEvent

```
public BroadcastFileEvent(BroadcastFile source)
```

Cria um `BroadcastFileEvent` indicando que o `BroadcastFile` especificado mudou.

Parâmetros:

`source` - o `BroadcastFile` cujo conteúdo mudou.

19.7.5 Detalhe dos métodos

getBroadcastFile

```
public BroadcastFile getBroadcastFile()
```

Relata o `BroadcastFile` cujo conteúdo mudou.

Retorna:

O `BroadcastFile` cujo conteúdo mudou.

19.8 Interface BroadcastFileListener

19.8.1 Descrição da interface

`com.sun.dtv.broadcast`

Todas as super-interfaces

`EventListener`

```
public interface BroadcastFileListener
extends EventListener
```

A interface `BroadcastFileListener` é implementada por classes de aplicativo as quais requerem notificação

das mudanças para dados `BroadcastFile`.

19.8.2 Índice de métodos

`void` **`broadcastFileChanged`**`(BroadcastFileEvent event)`

Notifica o `BroadcastFileListener` que o `BroadcastFile` mudou na transmissão.

`void` **`broadcastFileLost`**`(BroadcastFileEvent event)`

Notifica o `BroadcastFileListener` que o `BroadcastFile` foi perdido na transmissão.

19.8.3 Detalhe dos métodos

broadcastFileChanged

`void broadcastFileChanged(BroadcastFileEvent event)`

Notifica o `BroadcastFileListener` que o `BroadcastFile` mudou na transmissão. Se o conteúdo de um `BroadcastFile` muda enquanto um aplicativo estiver lendo seus dados a partir do cache local, os dados em cache devem ou (a) permanecer completamente não-modificados ou (b) ser liberado do cache. Se os dados são liberados do cache, tentativas de ler a partir desse `BroadcastFile` usando objetos leitores de arquivos pré-existentes (por exemplo, `FileInputStream`, `FileReader` ou `RandomAccessFile)` falham.

Para ler novos dados, o aplicativo deve criar um novo objeto leitor de arquivos. Para certificar-se de que esses dados são a versão mais recente da transmissão, o aplicativo deveria primeiro invocar o método `BroadcastFile.refreshCache()`.

Não são dadas garantias concernindo a habilidade do *listener* de detectar mudanças no `BroadcastFile` da transmissão ou a latência de notificação de eventos, se uma mudança for detectada.

Parâmetros:

`event` - Evento indicando `BroadcastFile` que mudou.

Relaciona-se com:

`BroadcastFile.refreshCache()`

broadcastFileLost

`void broadcastFileLost(BroadcastFileEvent event)`

Notifica o `BroadcastFileListener` que o `BroadcastFile` foi perdido na transmissão. A razão de perder `BroadcastFile` na transmissão pode ser a seleção de outro serviço, erros de transmissão, descontinuação da transmissão ou outros. Se o conteúdo de um `BroadcastFile` é perdido enquanto um aplicativo estiver lendo seus dados a partir do cache local, os dados em cache devem ou (a) permanecer disponíveis ou (b) ser limpados do cache. Se os dados são limpos do cache, tentativas de ler a partir desse `BroadcastFile` usando objetos leitores de arquivos pré-existentes (por exemplo, `FileInputStream`, `FileReader` ou `RandomAccessFile)` falharão. A criação de um novo objeto leitor de arquivos também DEVE falhar. Para certificar-se de que a informação sobre disponibilidade de dados é a mais recente da transmissão, o aplicativo deveria primeiro invocar o método `BroadcastFile.refreshCache()` no diretório principal.

Não são dadas garantias concernindo a habilidade do *listener* de detectar mudanças no `BroadcastFile` da transmissão ou a latência de notificação de eventos, se uma perda for detectada.

Parâmetros:

`event` - Evento indicando `BroadcastFile` que mudou.

Relaciona-se com:

```
BroadcastFile.refreshCache()
```

19.9 Classe BroadcastFilesystem

19.9.1 Descrição da Classe

com.sun.dtv.broadcast

```
java.lang.Object
  └com.sun.dtv.broadcast.BroadcastFilesystem
```

Todas as interfaces implementadas

```
Serializable
```

```
public class BroadcastFilesystem
extends Object
implements Serializable
```

Essa classe representa instâncias de sistemas de arquivos de transmissão montados no sistema de arquivos local. Isso significa que todo serviço transmitindo um sistema de arquivos de transmissão cria seu próprio sistema de arquivos, dependendo da definição do *stream* de transporte.

19.9.2 Índice de construtores

BroadcastFilesystem(URL url)

Cria e monta um sistema de arquivos de transmissão definido pela URL específica.

BroadcastFilesystem(Locator l)

Cria e monta um sistema de arquivos de transmissão definido pelo *listener* dependente de transporte específico.

19.9.3 Índice de métodos

static BroadcastFile **getBroadcastRoot**()

Restaura ponto de montagem raiz específico do sistema de todos os sistemas de arquivos de transmissão no sistema de arquivos local.

static String **getBroadcastRootPath**()

Restaura ponto de montagem raiz específico do sistema de todos os sistemas de arquivos de transmissão no sistema de arquivos local.

BroadcastFile **getMountLocation**()

Restaura a raiz desse sistema de arquivos de transmissão montado.

void **unmount**()

Desmonta esse sistema de arquivos de transmissão do sistema de arquivos local.

19.9.4 Detalhe dos construtores

BroadcastFilesystem

```
public BroadcastFilesystem(Locator l)
                    throws IOException,
                           IllegalArgumentException
```

Cria e monta um sistema de arquivos de transmissão definido pelo *listener* dependente de transporte específico.

Parâmetros:

`l` - o localizador do sistema de arquivos de transmissão a ser montado.

Lança:

`IOException` - se nenhum sistema de arquivos de transmissão existir no dado local.

`IllegalArgumentException` - se o dado localizador não puder ser usado para identificar um sistema de arquivos de transmissão.

BroadcastFilesystem

```
public BroadcastFilesystem(URL url)
                    throws IOException,
                           IllegalArgumentException
```

Cria e monta um sistema de arquivos de transmissão definido pela URL específica. A URL provida tem de seguir o "arquivo:" Protocolo como definido em RFC 1630 e RFC 1738.

Parâmetros:

`url` - A URL do sistema de arquivos de transmissão a ser montado

Lança:

`IOException` - se nenhum sistema de arquivos de transmissão existir no dado local.

`IllegalArgumentException` - se a dada URL não puder ser usada para identificar um sistema de arquivos de transmissão.

19.9.5 Detalhe dos métodos

unmount

```
public void unmount()
```

Desmonta esse sistema de arquivos de transmissão do sistema de arquivos local.

getMountLocation

```
public BroadcastFile getMountLocation()
                                throws IOException
```

Restaura a raiz desse sistema de arquivos de transmissão montado.

Retorna:

a localização da raiz desse sistema de arquivos de transmissão.

Lança:

`IOException` - lançada se o sistema de arquivos for desmontado.

getBroadcastRoot

```
public static BroadcastFile getBroadcastRoot()
```

Restaura ponto de montagem raiz específico do sistema de todos os sistemas de arquivos de transmissão no sistema de arquivos local. Uma lista de todos os sistemas de arquivos de transmissão disponíveis (não importa se

montados ou não) pode ser restaurada a partir do `BroadcastFile.listDirectory()` retornado.

Retorna:

A localização da raiz da transmissão

getBroadcastRootPath

`public static String getBroadcastRootPath()`

Restaura ponto de montagem raiz específico do sistema de todos os sistemas de arquivos de transmissão no sistema de arquivos local.

Retorna:

o caminho do ponto de montagem raiz

19.10 Classe BroadcastStream

19.10.1 Descrição da Classe

`com.sun.dtv.broadcast`

```
java.lang.Object
  └java.io.File
      └com.sun.dtv.broadcast.BroadcastStream
```

`public class BroadcastStream`

`extends File`

A classe `BroadcastStream` representa o *stream* obtido a partir do sistema de arquivos de transmissão. Uma instância do `BroadcastStream` pode ser construída a partir de uma instância `Locator` ou através de construtores similares àqueles do `java.io.File`. Este fato também significa que todos aquels construtores utilizam o mesmo caracter para a separação de nome a caminhos, como os outros arquivos. Por outro lado o localizador com base no construtor usa localizadores possivelmente criados a partir de URLs seguindo o esquema comum baseado em URL no qual os nomes dos caminhos são separados por "/".

Instanciação bem-sucedida de um objeto `BroadcastStream` causa a "montagem" dinâmica do seu *stream* de transmissão no sistema de arquivos local. O ponto preciso de montagem pode ser determinado chamando `getBroadcastFilesystem()`. Deveria ser observado que um sistema de arquivos de transmissão também pode ser explicitamente montado usando instâncias de `BroadcastFilesystem`.

A construção de um objeto `BroadcastStream` não causa o carregamento automático de seu conteúdo a partir do *stream* de transmissão. O conteúdo só deve ser carregado quando um `FileInputStream` for criado a partir desse `BroadcastStream`. Tentativas subsequentes de ler os dados de um objeto `BroadcastStream` devem ser bloqueadas até que seu conteúdo esteja carregado.

Todos os construtores e operações de entrada e saída subseqüentes devem usar `BroadcastException` para lançar exceções específicas de transmissão.

Sistemas de arquivos de transmissão para os quais não há instâncias `BroadcastStream` restantes ou descritores de arquivos abertos têm admissão para desmontarem-se do sistema de arquivos local.

Implementações de API Java DTV que não suportam acesso a sistema de arquivos de transmissão lançarão `UnsupportedOperationException` após qualquer tentativa de construir um objeto `BroadcastStream`.

Todas as interfaces implementadas

Comparable, Serializable

19.10.2 Índice de construtores

BroadcastStream(File dir, String name)

Cria uma instância BroadcastStream que representa o *stream* com o nome especificado no diretório de transmissão especificado.

BroadcastStream(String path)

Cria uma instância BroadcastStream que representa o *stream* cujo nome do caminho na transmissão é o argumento do caminho.

BroadcastStream(String path, String name)

Cria uma instância BroadcastStream que representa o *stream* com o nome especificado no diretório de transmissão especificado.

BroadcastStream(Locator locator)

Cria uma instância BroadcastStream que representa o arquivo referenciado pelo Locator.

19.10.3 Índice de métodos

BroadcastFilesystem **getBroadcastFilesystem**()

Retorna o BroadcastFileSystem a que esse BroadcastStream pertence.

long **getCurrentTime**()

Retorna o tempo do próximo bloco em BroadcastStream.

long **getDuration**()

Retorna a duração desse BroadcastStream.

Locator **getLocator**()

Retorna um Locator identificando esse BroadcastStream.

String **getType**()

Restaura o tipo mime de um BroadcastStream.

19.10.4 Detalhe dos construtores

BroadcastStream

```
public BroadcastStream(Locator locator)
                throws InvalidLocatorException,
                       IOException
```

Cria uma instância BroadcastStream que representa o arquivo referenciado pelo Locator.

Esse construtor lança java.io.IOException se isso determinar imediatamente que o *stream* de transmissão requisitado não pode ser acessado. Uma vez que esse construtor pode completar seu trabalho assincronamente, a ausência de um IOException não é garantia de que o *stream* de transmissão requisitado esteja acessível.

Parâmetros:

locator - Um Locator referenciando a fonte do BroadcastStream.

Lança:

InvalidLocatorException - Se o localizador não consultar um *stream* de transmissão.

IOException - Se o *stream* de transmissão requisitado não pode ser acessado ou os dados recebidos são inválidos.

`UnsupportedOperationException` - Se o sistema de arquivos de transmissão não é suportado.

BroadcastStream

```
public BroadcastStream(String path)
                throws IOException
```

Cria uma instância `BroadcastStream` que representa o *stream* cujo nome do caminho na transmissão é o argumento do caminho.

Esse construtor lança `java.io.IOException` se isso determinar imediatamente que o *stream* de transmissão requisitado não pode ser acessado. Uma vez que esse construtor pode completar seu trabalho assincronamente, a ausência de um `IOException` não é garantia de que o *stream* de transmissão requisitado esteja acessível.

Parâmetros:

`path` - O nome do caminho do *stream*.

Lança:

`IOException` - Se o *stream* de transmissão requisitado não pode ser acessado ou os dados recebidos são inválidos.

`UnsupportedOperationException` - Se o sistema de arquivos de transmissão não é suportado.

BroadcastStream

```
public BroadcastStream(File dir,
                       String name)
                throws IOException
```

Cria uma instância `BroadcastStream` que representa o *stream* com o nome especificado no diretório de transmissão especificado.

Esse construtor lança `java.io.IOException` se isso determinar imediatamente que o *stream* de transmissão requisitado não pode ser acessado. Uma vez que esse construtor pode completar seu trabalho assincronamente, a ausência de um `IOException` não é garantia de que o *stream* de transmissão requisitado esteja acessível.

Parâmetros:

`dir` - O diretório.

`name` - O nome do *stream*.

Lança:

`IOException` - se o *stream* de transmissão requisitado não pode ser acessado ou os dados recebidos são inválidos.

`UnsupportedOperationException` - se o sistema de arquivos de transmissão não é suportado.

BroadcastStream

```
public BroadcastStream(String path,
                       String name)
                throws IOException
```

Cria uma instância `BroadcastStream` que representa o *stream* com o nome especificado no diretório de transmissão especificado.

Esse construtor lança `java.io.IOException` se isso determinar imediatamente que o *stream* de transmissão requisitado não pode ser acessado. Uma vez que esse construtor pode completar seu trabalho assincronamente,

a ausência de um `IOException` não é garantia de que o *stream* de transmissão requisitado esteja acessível.

Parâmetros:

`path` - o nome do caminho do diretório.

`name` - o nome do *stream*.

Lança:

`IOException` - se o *stream* de transmissão requisitado não pode ser acessado ou os dados recebidos são inválidos.

`UnsupportedOperationException` - Se o sistema de arquivos de transmissão não é suportado.

19.10.5 Detalhe dos métodos

getType

```
public String getType()
```

Restaura o tipo mime de um `BroadcastStream`.

Retorna:

o tipo mime do `BroadcastStream`, `null` se não estiver disponível.

getDuration

```
public long getDuration()
```

Retorna a duração desse `BroadcastStream`.

Retorna:

A duração desse *stream* de transmissão em micro segundos. Retorna -1 se for desconhecido.

getCurrentTime

```
public long getCurrentTime()
```

Retorna o tempo do próximo bloco em `BroadcastStream`.

Retorna:

o tempo do próximo bloco em micro segundos. Retorna -1 se for desconhecido.

getBroadcastFilesystem

```
public BroadcastFilesystem getBroadcastFilesystem()
```

Retorna o `BroadcastFileSystem` a que esse `BroadcastStream` pertence.

Retorna:

Um `BroadcastFileSystem` a que esse `BroadcastStream` pertence.

getLocator

```
public Locator getLocator()
```

Retorna um `Locator` identificando esse `BroadcastStream`.

Retorna:

Um `Locator` identificando esse `BroadcastStream`.

20 Pacote com.sun.dtv.broadcast.event

20.1 Descrição do pacote

Provê acesso a eventos de transmissão.

A classe da API Java DTV `com.sun.dtv.broadcast.event.BroadcastEventManager` é usada para manusear eventos de transmissão os quais são entregues por meio da interface `BroadcastEventListener`.

A Figura 6 mostra a estrutura do pacote `Broadcast.Event` do Java DTV.

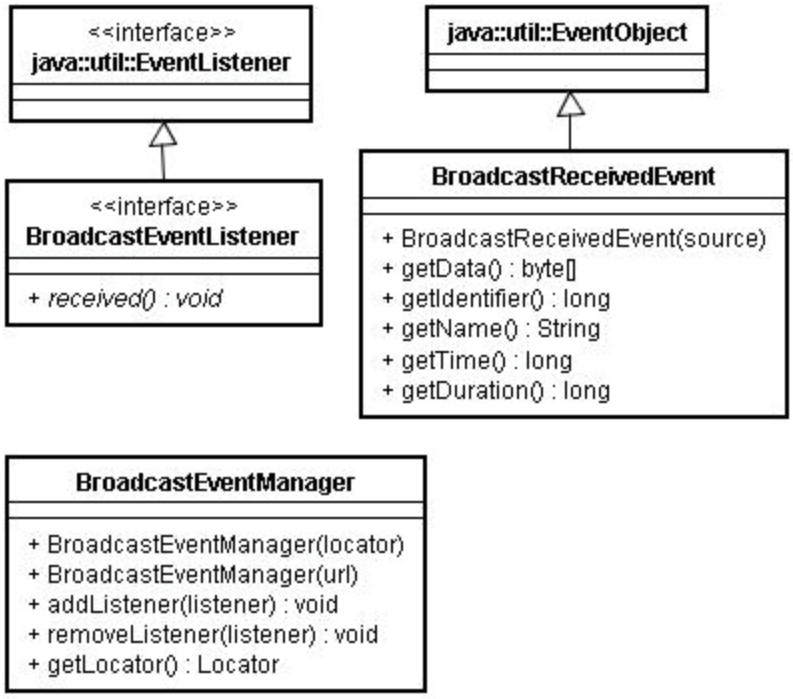


Figura 6 – Pacote `Broadcast.Event`

NOTA Este pacote está especificado desde o Java DTV 1.0.

20.2 Índice de interfaces

BroadcastEventListener

A interface `BroadcastEventListener` é implementada por classes de aplicativo que requerem notificação de dados `BroadcastEvent` recebidos.

20.3 Índice de classes

BroadcastEventManager

A classe `BroadcastEventManager` representa eventos obtidos do sistema de arquivos de transmissão.

BroadcastReceivedEvent

Essa classe representa eventos que foram recebidos por meio de transmissão.

20.4 Interface BroadcastEventListener

20.4.1 Descrição da Interface

com.sun.dtv.broadcast.event

```
public interface BroadcastEventListener
extends EventListener
```

A interface `BroadcastEventListener` é implementada por classes de aplicativo que requerem notificação de dados `BroadcastEvent` recebidos.

Todas as super-interfaces

`EventListener`

20.4.2 Índice de métodos

`void` **`received`**`(BroadcastReceivedEvent event)`

Notifica o `BroadcastEventListener` que o `BroadcastEvent` foi recebido na transmissão.

20.4.3 Detalhe dos métodos

received

```
void received(BroadcastReceivedEvent event)
```

Notifica o `BroadcastEventListener` que o `BroadcastEvent` foi recebido na transmissão.

Não são dadas garantias concernindo a habilidade do *listener* de receber todos `BroadcastEvent` no *stream* de transporte, nem a latência de modificação de evento após a recepção do evento no *stream*.

Parâmetros:

`event` - evento indicando que `BroadcastEvent` foi recebido.

20.5 Classe BroadcastEventManager

20.5.1 Descrição da classe

com.sun.dtv.broadcast.event

```
java.lang.Object
  └com.sun.dtv.broadcast.event.BroadcastEventManager
```

```
public class BroadcastEventManager
extends Object
```

A classe `BroadcastEventManager` representa eventos obtidos do sistema de arquivos de transmissão. Uma instância de `BroadcastEventManager` pode ser construída a partir de uma instância `Locator`.

Eventos chegando no canal de transmissão devem ser entregues como `BroadcastReceivedEvent` usando o `BroadcastEventListener`.

Todos os construtores devem usar `BroadcastException` para lançar exceções específicas de transmissão. Implementações Java DTV API que não suportam acesso eventos de transmissão lançam `UnsupportedOperationException` após qualquer tentativa de construir um objeto `BroadcastEvent`.

20.5.2 Índice de construtores

BroadcastEventManager(String url)

Cria uma instância BroadcastEventManager que representa os eventos cujos nome do caminho na transmissão é o argumento do caminho.

BroadcastEventManager(Locator locator)

Cria uma instância BroadcastEventManager que representa os eventos referenciados pelo dado Locator.

20.5.3 Índice de métodos

void **addListener**(BroadcastEventListener listener)

Subscreve um BroadcastEventListener para recbeber notificações de BroadcastReceivedEvent recebidos.

Locator **getLocator**()

Retorna um Locator identificando esse BroadcastEvent.

void **removeListener**(BroadcastEventListener listener)

Retira a subscrição de um BroadcastEventListener para receber notificações de BroadcastEvent recebidos.

20.5.4 Detalhe dos construtores

BroadcastEventManager

```
public BroadcastEventManager(Locator locator)
                      throws InvalidLocatorException,
                             IOException
```

Cria uma instância BroadcastEventManager que representa os eventos referenciados pelo dado Locator.

Esse construtor lança java.io.IOException se isso determinar imediatamente que os eventos de transmissão requisitados não podem ser acessados. Uma vez que esse construtor pode completar seu trabalho assincronamente, a ausência de um IOException não é garantia de que os eventos de transmissão requisitados estejam acessíveis.

Parâmetros:

locator - Um Locator referenciando a fonte do BroadcastEvent.

Lança:

InvalidLocatorException - se o localizador não consultar eventos de transmissão.

IOException - se os eventos de transmissão requisitados não puderem ser acessados.

UnsupportedOperationException - se o sistema de arquivos de transmissão não é suportado.

BroadcastEventManager

```
public BroadcastEventManager(String url)
                      throws IOException
```

Cria uma instância BroadcastEventManager que representa os eventos cujos nome do caminho na transmissão é o argumento do caminho.

Esse construtor lança `java.io.IOException` se isso determinar imediatamente que os eventos de transmissão requisitados não podem ser acessados. Uma vez que esse construtor pode completar seu trabalho assincronamente, a ausência de um `IOException` não é garantia de que os eventos de transmissão requisitados estejam acessíveis.

Parâmetros:

`url` – o nome do caminho do *stream*.

Lança:

`IOException` - se os eventos de transmissão requisitados não puderem ser acessados.

`UnsupportedOperationException` - se o sistema de arquivos de transmissão não é suportado.

20.5.5 Detalhe dos métodos

addListener

```
public void addListener(BroadcastEventListener listener)
                throws IOException,
                       SecurityException
```

Subscreve um `BroadcastEventListener` para recbeber notificações de `BroadcastReceivedEvent` recebidos. Se o *listener* especificado estiver correntemente subscrito, nenhuma ação é tomada.

Parâmetros:

`listener` - o `BroadcastEventListener` a ser notificado.

Lança:

`IOException` - se houver recursos insuficientes para suportar esse *listener*.

`SecurityException` - Se um gerenciador de segurança existir e seu método `SecurityManager.checkRead(java.lang.String)` negar acesso de leitura ao arquivo.

Relaciona-se com:

`removeListener()`

removeListener

```
public void removeListener(BroadcastEventListener listener)
```

Retira a subscrição de um `BroadcastEventListener` para receber notificações de `BroadcastEvent` recebidos. Se o dado `BroadcastFileListener` não estiver correntemente subscrito para notificação, nenhuma ação é tomada.

Parâmetros:

`listener` - um `BroadcastEventListener` correntemente registrado.

Relaciona-se com:

`addListener()`

getLocator

```
public Locator getLocator()
```

Retorna um `Locator` identificando esse `BroadcastEvent`.

Retorna:

Um `Locator` identificando esse `BroadcastEvent`.

20.6 Classe BroadcastReceivedEvent

20.6.1 Descrição da classe

com.sun.dtv.broadcast.event

```
java.lang.Object
  └java.util.EventObject
      └com.sun.dtv.broadcast.event.BroadcastReceivedEvent
public class BroadcastReceivedEvent
extends EventObject
```

Essa classe representa eventos que foram recebidos por meio de transmissão.

Todas as interfaces implementadas:

`Serializable`

20.6.2 Índice de construtores

`BroadcastReceivedEvent``(BroadcastEventManager source)`

Cria um `BroadcastFileEvent` indicando que o `BroadcastFile` especificado mudou.

20.6.3 Índice de métodos

`byte[]` **`getData`**`()`

Restaura o bloco de dados recebidos do `BroadcastReceivedEvent`.

`long` **`getDuration`**`()`

Retorna a duração desse `BroadcastReceivedEvent`.

`long` **`getIdentifier`**`()`

Retorna o identificador desse `BroadcastReceivedEvent`.

`String` **`getName`**`()`

Retorna o nome desse `BroadcastReceivedEvent`.

`long` **`getTime`**`()`

Retorna o tempo desse `BroadcastReceivedEvent`.

20.6.4 Detalhe dos construtores

BroadcastReceivedEvent

`public BroadcastReceivedEvent(BroadcastEventManager source)`

Cria um `BroadcastFileEvent` indicando que o `BroadcastFile` especificado mudou.

Parâmetros:

`source` - O `BroadcastFile` cujo conteúdo mudou.

20.6.5 Detalhe dos métodos

getData

`public byte[] getData()`

Restaura o bloco de dados recebidos do `BroadcastReceivedEvent`.

Retorna:

os dados

getIdentifier

`public long getIdentifier()`

Retorna o identificador desse `BroadcastReceivedEvent`.

Retorna:

o identificador desse `BroadcastReceivedEvent`. Retorna -1 se for desconhecido.

getName

`public String getName()`

Retorna o nome desse `BroadcastReceivedEvent`.

Retorna:

o nome desse `BroadcastReceivedEvent`. Retorna `null` se for desconhecido.

getTime

`public long getTime()`

Retorna o tempo desse `BroadcastReceivedEvent`.

Retorna:

o tempo desse `BroadcastReceivedEvent` em micro segundos. Retorna -1 se for desconhecido.

getDuration

`public long getDuration()`

Retorna a duração desse `BroadcastReceivedEvent`.

Retorna:

a duração desse evento transmissão em micro segundos. Retorna -1 se for desconhecido.

21 Pacote com.sun.dtv.filtering

21.1 Descrição do pacote

Provê suporte para filtragem de seções MPEG.

Descrição

A filtragem de seções MPEG provê acesso a seções MPEG-2 TS que combinam uma coleção de filtros providos pelo aplicativo. A API usa assincronamente uma interface de *listener* para ser notificado de eventos tais como a disponibilidade de seções ou que um erro ocorreu.

Diferentes mecanismos e comportamentos de filtro podem ser definidos usando `DataSectionFilter` e suas subclasses `CircularFilter`, `ListFilter` e `SingleFilter`.

A Figura 7 mostra a estrutura do pacote `Filtering` do Java DTV.

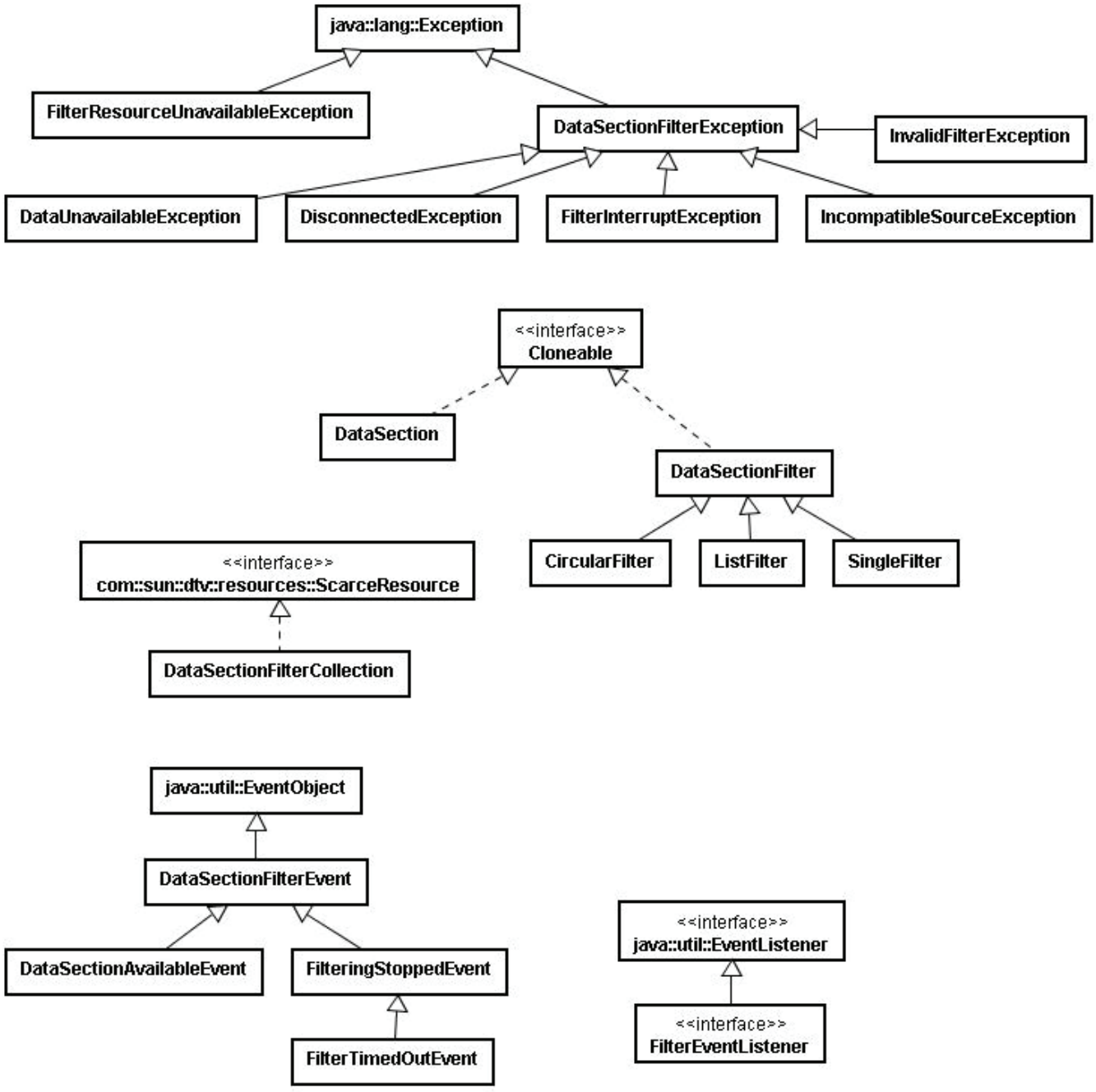


Figura 7 – Pacote `Filtering`

NOTA Este pacote está especificado desde o Java DTV 1.0.

21.2 Índice de interfaces

FilterEventListener

A interface desse *listener* pode ser implementada por classes requerendo eventos filtradores.

21.3 Índice de classes

CircularFilter

Essa classe define um filtro de seção com intenção de ser usado para capturar um *stream* contínuo de seções de dados sem necessitar reiniciar continuamente o filtro.

DataSection

Essa classe encapsula uma seção de dados do Fluxo de Transporte

DataSectionAvailableEvent

Esse evento relata que uma seção completa de dados está sendo filtrada.

DataSectionDataBuffer

Essa classe encapsula uma porção de uma seção de carga útil de Fluxo de Transporte.

DataSectionFilter

Essa classe é a raiz de todas as classes de filtro.

DataSectionFilterCollection

Essa classe representa uma coleção de filtros de seções de dados a serem ativados e desativados como uma operação atômica.

DataSectionFilterEvent

Essa classe é a classe base para Eventos na API filtro de seções.

FilteringStoppedEvent

Essa classe é usada pra relatar o fim de uma operação de filtragem com uma exceção: Não é gerado quando o filtramento pára para um `SimpleSectionFilter` sob circunstâncias normais.

FilterTimedOutEvent

Esse evento é gerado se o tempo de espera das operações do filtro de seção esgotar dentro do período especificado pelo método `setTimeOut()`.

ListFilter

Essa classe define um filtro de seção que irá filtrar um conjunto inteiro de segmentos de seção de dados que compõem uma única tabela de seções.

SingleFilter

Essa classe define um filtro de seção com intenção de ser usado para capturar uma seção de dados avulsa.

21.4 Índice de exceções

DataSectionFilterException

A classe base para exceções lançadas pelas API de filtragem de seções de dados.

DataUnavailableException

Indica que não há dados disponíveis em um objeto `DataSection`.

DisconnectedException

Indica que um `DataSectionFilterCollection` perdeu sua conexão, causando a falha de uma chamada startFiltering.

FilterInterruptException

Indica que um filtro foi interrompido antes que a quantidade requerida de dados fosse filtrada.

FilterResourceUnavailableException

Indica que requerimentos de recursos não podem ser alcançados por uma operação de filtro.

IncompatibleSourceException

Indica que o *stream* fonte provido é incompatível.

InvalidFilterException

Indica que um filtro foi definido inapropriadamente.

21.5 Classe CircularFilter

21.5.1 Descrição da classe

`com.sun.dtv.filtering`

```
java.lang.Object
  └com.sun.dtv.filtering.DataSectionFilter
      └com.sun.dtv.filtering.CircularFilter
```

```
public class CircularFilter
extends DataSectionFilter
```

Essa classe define um filtro de seção com intenção de ser usado para capturar um *stream* contínuo de seções de dados sem necessitar reiniciar continuamente o filtro.

Um `DataSectionAvailableEvent` deve ser gerado cada vez que um `DataSection` for capturado. Um objeto `CircularFilter` tem um número pré-definidos de objetos `DataSection` definidos no momento da construção. Seções de dados filtradas com sucesso são carregadas em objetos `DataSection` disponíveis seqüencialmente.

A filtragem deve ser parada apenas se o tempo de espera esgotar desde a última seção filtrada com sucesso ou se não houver mais objetos `DataSection` disponíveis para escrita.

Aplicativos querendo que a filtragem proceda indefinidamente devem usar o método release do `DataSection` para que eles sejam re-usados em novas combinações de filtragem de seções.

Relaciona-se com:

`DataSection`, `DataSection.release()`, `DataSectionFilter`

Todas as interfaces implementadas

`Cloneable`

21.5.2 Índice de métodos

`DataSection[]` **`getSections`**`()`

Esse método retorna o *array* do `DataSection` para o `CircularFilter`. O aplicativo precisa verificar quais objetos contém dados válidos.

Métodos herdados da classe `com.sun.dtv.filtering.`**`DataSectionFilter`**

addSectionFilterListener, clearGreaterThanFilter, clearLessThanFilter, clearPositiveFilter, clearTableId, clearXorFilter, removeSectionFilterListener, setGreaterThanFilter, setLessThanFilter, setPid, setPositiveFilter, setTableId, setTimeOut, setXorFilter, startFiltering, stopFiltering

21.5.3 Detalhe dos métodos

getSections

```
public DataSection[] getSections()
```

Esse método retorna o *array* do `DataSection` para o `CircularFilter`. O aplicativo precisa verificar quais objetos contém dados válidos.

Retorna:

O *array* de objetos `DataSection`

21.6 Classe DataSection

21.6.1 Descrição da classe

com.sun.dtv.filtering

```
java.lang.Object
  └com.sun.dtv.filtering.DataSection
```

```
public class DataSection
extends Object
implements Cloneable
```

Essa classe encapsula uma seção de dados do *stream* de transporte.

Esse objeto é clonável. A modificação de um objeto clonado não modifica o objeto original. Em acréscimo, acesso a seções de dados brutos é também provida por meio de um mecanismo de cópia. A edição de tais dados não afetará dados internos dos objetos da Seção, nem afetará os conteúdos de um transporte ou *stream* elementar.

Todas as interfaces implementadas

`Cloneable`

21.6.2 Índice de métodos

`Object` **`clone`**`()`

Cria uma cópia desse objeto `DataSection`.

`boolean` **`getAvailable`**`()`

Restaura independente de um objeto `DataSection` conter dados válidos.

`boolean` **`getCurrentNextIndicator`**`()`

Restaura o valor do campo correspondente da seção de dados *header*.

`int` **`getLastSectionNumber`**`()`

Restaura o valor do campo correspondente da seção de dados *header*.

`boolean` **`getPrivateIndicator`**`()`

Restaura o valor do campo correspondente da seção de dados *header*.

`byte` **`getSectionByte`**`(int index)`

Restaura um byte de dados avulso no índice provido a partir da seção filtrada.

`DataSectionDataBuffer` **`getSectionData`**`()`

Restaura os dados da seção filtrada no objeto `DataSection`.

`DataSectionDataBuffer` **`getSectionData`**`(int index, int length)`

Restaura dados provenientes da seção filtrada usando o índice e comprimento providos.

`int` **`getSectionLength`**`()`

Restaura o valor do campo correspondente da seção de dados *header*.

`int` **`getSectionNumber`**`()`

Restaura o valor do campo correspondente da seção de dados *header*.

`boolean` **`getSectionSyntaxIndicator`**`()`

Restaura o valor do campo correspondente da seção de dados *header*.

`int` **`getTableId`**`()`

Restaura o valor do campo correspondente de uma seção de dados *header*.

`int` **`getTableIdExtension`**`()`

Restaura o valor do campo correspondente da seção de dados *header*.

`short` **`getVersionNumber`**`()`

Restaura o valor do campo correspondente da seção de dados *header*.

`void` **`release`**`()`

Esse método marca objetos das seções de dados para uso, porque os dados não são mais úteis ou foram clonados.

21.6.3 Detalhe dos métodos

getSectionData

```
public DataSectionDataBuffer getSectionData()
                                    throws DataUnavailableException
```

Restaura os dados da seção filtrada no objeto `DataSection`. Uma nova cópia da seção de dados deve ser criada para cada chamada (isto é, tudo após o campo comprimento, incluindo o *header* da seção e não incluindo uma verificação de CRC).

Retorna:

Um `DataSectionDataBuffer` contendo os dados requisitados.

Lança:

`DataUnavailableException` - se não houver dados válidos disponíveis.

getSectionData

```
public DataSectionDataBuffer getSectionData(int index,
                                            int length)
                                    throws DataUnavailableException,
                                           IndexOutOfBoundsException
```

Restaura dados provenientes da seção filtrada usando o índice e comprimento providos. Uma nova cópia da seção de dados deve ser criada para cada chamada (isto é, tudo após o campo comprimento, não incluindo uma verificação de CRC).

Parâmetros:

`index` - menu iniciar (o primeiro byte da seção (o campo `table_id`) tem índice 1)

`length` - o comprimento

Retorna:

Um `DataSectionDataBuffer` contendo os dados requisitados.

Lança:

`DataUnavailableException` - se não houver dados válidos disponíveis.

`IndexOutOfBoundsException` - se os dados requisitados não estiverem no limite de dados da seção.

getSectionByte

```
public byte getSectionByte(int index)
                    throws DataUnavailableException,
                           IndexOutOfBoundsException
```

Restaura um byte de dados avulso no índice provido a partir da seção filtrada.

Parâmetros:

`index` - o índice (o primeiro byte da seção (o campo `table_id`) tem índice 1)

Retorna:

o valor de byte no índice especificado da seção de dados.

Lança:

`DataUnavailableException` - se não houver dados válidos disponíveis.

`IndexOutOfBoundsException` - se os dados requisitados não estiverem no limite de dados da seção.

getTableId

```
public int getTableId()
```

Restaura o valor do campo correspondente de uma seção de dados *header*.

Observação: Não são lançadas exceções uma vez que esse campo deveria estar SEMPRE disponível.

Retorna:

o valor de `table_id`

getSectionSyntaxIndicator

```
public boolean getSectionSyntaxIndicator()
```

Restaura o valor do campo correspondente da seção de dados *header*.

Observação: Não são lançadas exceções uma vez que esse campo deveria estar SEMPRE disponível.

Retorna:

se o indicador de sintaxe da seção está estabelecido na seção de dados.

getPrivateIndicator

```
public boolean getPrivateIndicator()
```

Restaura o valor do campo correspondente da seção de dados *header*.

Observação: Não são lançadas exceções uma vez que esse campo deveria estar SEMPRE disponível.

Retorna:

se a *flag* indicador privada está estabelecida na seção de dados.

getSectionLength

```
public int getSectionLength()
```

Restaura o valor do campo correspondente da seção de dados *header*.

Observação: Não são lançadas exceções uma vez que esse campo deveria estar SEMPRE disponível.

Retorna:

o campo comprimento da seção

getTableIdExtension

```
public int getTableIdExtension()
                        throws DataUnavailableException
```

Restaura o valor do campo correspondente da seção de dados *header*.

Retorna:

o campo `table_id_extension`

Lança:

`DataUnavailableException` - se não houver dados válidos disponíveis.

getVersionNumber

```
public short getVersionNumber()
                       throws DataUnavailableException
```

Restaura o valor do campo correspondente da seção de dados *header*.

Retorna:

o campo número da versão.

Lança:

`DataUnavailableException` - se não houver dados válidos disponíveis.

getCurrentNextIndicator

```
public boolean getCurrentNextIndicator()
                                 throws DataUnavailableException
```

Restaura o valor do campo correspondente da seção de dados *header*.

Retorna:

se a *flag* current_next_indicator está estabelecida na seção de dados.

Lança:

`DataUnavailableException` - se não houver dados válidos disponíveis.

getSectionNumber

```
public int getSectionNumber()
                     throws DataUnavailableException
```

Restaura o valor do campo correspondente da seção de dados *header*.

Retorna:

o número da seção

Lança:

`DataUnavailableException` - se não houver dados válidos disponíveis.

getLastSectionNumber

```
public int getLastSectionNumber()
                         throws DataUnavailableException
```

Restaura o valor do campo correspondente da seção de dados *header*.

Retorna:

o campo `last_section_number`.

Lança:

`DataUnavailableException` - se não houver dados válidos disponíveis.

getAvailable

```
public boolean getAvailable()
```

Restaura independente de um objeto `DataSection` conter dados válidos.

Retorna:

se os dados da seção estão válidos e disponíveis para leitura.

release

```
public void release()
```

Esse método marca objetos das seções de dados para uso, porque os dados não são mais úteis ou foram clonados. Isso permite objetos `RingFilter` a re-usar o objeto da seção de dados para uma combinação de filtros subseqüente.

clone

```
public Object clone()
```

Cria uma cópia desse objeto `DataSection`. Quaisquer modificações nesse objeto copiado não afetarão o objeto original que foi clonado.

Substituições:

`clone` na classe `Object`

Retorna:

o objeto clonado.

21.7 Classe DataSectionAvailableEvent

21.7.1 Descrição da classe

com.sun.dtv.filtering

```
java.lang.Object
  └java.util.EventObject
      └com.sun.dtv.filtering.DataSectionFilterEvent
          └com.sun.dtv.filtering.DataSectionAvailableEvent
```

```
public class DataSectionAvailableEvent
extends DataSectionFilterEvent
```

Esse evento relata que uma seção completa de dados está sendo filtrada. É emitido pelos objetos `SingleFilter`, `ListFilter` e `CircularFilter` quando uma seção de dados combinando o filtro definido é descoberta e analisada.

Todas as interfaces implementadas

`Serializable`

21.7.2 Índice de construtores

`DataSectionAvailableEvent``(DataSectionFilter f, Object refObj)`

Isso constrói um `DataSectionAvailableEvent` para o `DataSectionFilter` especificado.

21.7.3 Índice de métodos

`Object` **`getSource`**`()`

Restaura o objeto `DataSectionFilter` filtrado na seção de dados.

Métodos herdados da classe `com.sun.dtv.filtering.`**`DataSectionFilterEvent`**

`getrefObj`

21.7.4 Detalhe dos construtores

DataSectionAvailableEvent

```
public DataSectionAvailableEvent(DataSectionFilter f,
                                 Object refObj)
```

Isso constrói um `DataSectionAvailableEvent` para o `DataSectionFilter` especificado.

Parâmetros:

`f` - a fonte do evento

`refObj` - o objeto retorno de chamada registrado na chamada `startFiltering` original.

21.7.5 Detalhe dos métodos

getSource

`public Object getSource()`

Restaura o objeto `DataSectionFilter` filtrado na seção de dados.

Substituições:

`getSource` na classe `DataSectionFilterEvent`

Retorna:

o objeto `DataSectionFilter`

21.8 Classe DataSectionDataBuffer

21.8.1 Descrição da classe

`com.sun.dtv.filtering`

```
java.lang.Object
  └com.sun.dtv.filtering.DataSectionDataBuffer
```

`public class DataSectionDataBuffer`

`extends Object`

Essa classe encapsula uma porção de uma seção de carga útil de Fluxo de Transporte.

Relaciona-se com:

`DataSection`

21.8.2 Índice de campos

`protected byte[] **data**`

O *array* de bytes que segura o bloco media data para esse `Buffer`.

`static int **FLAG_IS_COPY**`

Indica que esse dado é uma cópia ou clone e mudá-lo não afetará os dados dos quais ele foi gerado.

`protected int ***flags***`

Uma *flag* máscara que descreve os atributos boolean disponíveis para esse `Buffer`.

`protected int **length**`

Informa quantas unidades são válidas no *array*.

`protected int **offset**`

Indica o ponto de início (ajuste) no *array* onde os dados válidos começam.

21.8.3 Índice de construtores

`DataSectionDataBuffer``()`

21.8.4 Índice de métodos

`Object` **`clone`**`()`

Clona um buffer.

`void` **`copy`**`(DataSectionDataBuffer buffer)`

Copia os atributos do `Buffer` especificado nesse `Buffer`.

`void` **`copy`**`(DataSectionDataBuffer buffer, boolean swapData)`

Copia os atributos do `Buffer` especificado nesse `Buffer`.

`byte[]` **`getData`**`()`

Obtém o *array* de bytes internos para esse `Buffer`.

`int` **`getFlags`**`()`

Obtém a máscara das *flag*s estabelecidas para esse `Buffer`.

`int` **`getLength`**`()`

Obtém o comprimento dos dados válidos nesse Buffer.

`int` **`getOffset`**`()`

Obtém o ajuste dos dados válidos nesse `Buffer`.

`void` **`setData`**`(byte[] data)`

Define o *array* de bytes internos para esse `Buffer`.

`void` **`setFlags`**`(int` *`flags`*`)`

Define a *flag* máscara para esse `Buffer`.

`void` **`setLength`**`(int length)`

Define o comprimento dos dados válidos nesse `Buffer`.

`void` **`setOffset`**`(int offset)`

Define o ajuste dos dados válidos nesse `Buffer`.

`String` **`toString`**`()`

A representação `String` do objeto.

21.8.5 Detalhe dos campos

***flag*s**

`protected int` *`flags`*

Uma *flag* máscara que descreve os atributos boolean disponíveis para esse `Buffer`. Essa máscara é definida pela soma lógica de todas as *flag*s definidas.

Relaciona-se com:

`FLAG_IS_COPY`

data

`protected byte[] data`

O *array* de bytes que segura o bloco media data para esse `Buffer`.

length

`protected int length`

Informa quantas unidades são válidas no *array*. (o próprio *array* pode ser maior que o comprimento).

offset

`protected int offset`

Indica o ponto de início (ajuste) no *array* onde os dados válidos começam.

FLAG_IS_COPY

`public static final int FLAG_IS_COPY = 1`

Indica que esse dado é uma cópia ou clone e mudá-lo não afetará os dados dos quais ele foi gerado.

21.8.6 Detalhe dos construtores

DataSectionDataBuffer

`public DataSectionDataBuffer()`

21.8.7 Detalhe dos métodos

getFlags

`public int getFlags()`

Obtém a máscara das *flag*s estabelecidas para esse `Buffer`. O valor inteiro da máscara é igual à soma lógica das *flag*s estabelecidas.

Retorna:

as *flag*s correntemente ativas

Relaciona-se com:

`FLAG_IS_COPY`, `setFlags()`

setFlags

`public void setFlags(int flags)`

Define a *flag* máscara para esse `Buffer`. O valor inteiro da máscara é igual à soma lógica das *flag*s estabelecidas.

Parâmetros:

`flags` - As *flag*s específicas a serem estabelecidas para o objeto buffer.

Relaciona-se com:

`FLAG_IS_COPY`, `getFlags()`

getData

`public byte[] getData()`

Obtém o *array* de bytes internos para esse `Buffer`.

Retorna:

O *array* de bytes para esse `Buffer`.

Relaciona-se com:

`data`, `setData()`

setData

```
public void setData(byte[] data)
```

Define o *array* de bytes internos para esse `Buffer`.

Parâmetros:

`data` - o *array* de bytes a ser definido

Relaciona-se com:

`data`, `getData()`

getLength

```
public int getLength()
```

Obtém o comprimento dos dados válidos nesse Buffer.

Retorna:

O comprimento dos dados válidos no *array* de bytes para esse `Buffer`.

Relaciona-se com:

`length`, `setLength()`

setLength

```
public void setLength(int length)
```

Define o comprimento dos dados válidos nesse `Buffer`.

Parâmetros:

`length` - O comprimento dos dados válidos no *array* de bytes de dados.

Relaciona-se com:

`length`, `getLength()`

getOffset

```
public int getOffset()
```

Obtém o ajuste dos dados válidos nesse `Buffer`.

Retorna:

O ponto de início dos dados válidos no *array* de bytes de dados.

Relaciona-se com:

`offset`, `setOffset()`

setOffset

```
public void setOffset(int offset)
```

Define o ajuste dos dados válidos nesse `Buffer`.

Parâmetros:

`offset` - O ponto de início dos dados válidos no *array* de bytes de dados.

Relaciona-se com:

`offset`, `getOffset()`

copy

```
public void copy(DataSectionDataBuffer buffer)
```

Copia os atributos do `Buffer` especificado nesse `Buffer`.

Parâmetros:

`buffer` - O `Buffer` de entrada copia os atributos de.

copy

```
public void copy(DataSectionDataBuffer buffer,
                 boolean swapData)
```

Copia os atributos do `Buffer` especificado nesse `Buffer`. Se swapData for true, os valores dos dados são trocados entre os buffers, caso contrário, os valores dos dados são copiados.

Parâmetros:

`buffer` - O `Buffer` de entrada copia os atributos de.

`swapData` – Especifica-se os objetos de dados que devem ser trocados.

clone

```
public Object clone()
```

Clona um buffer.

Substituições:

`clone` na classe `Object`

Retorna:

o objeto clonado

toString

```
public String toString()
```

A representação `String` do objeto.

Substituições:

`toString` na classe `Object`

Retorna:

a representação `string`

21.9 Classe DataSectionFilter

21.9.1 Descrição da classe

com.sun.dtv.filtering

```
java.lang.Object
  └com.sun.dtv.filtering.DataSectionFilter
```

```
public class DataSectionFilter
extends Object
implements Cloneable
```

Essa classe é a raiz de todas as classes de filtro.

Todas as interfaces implementadas

`Cloneable`

Subclasses diretas conhecidas

`CircularFilter, ListFilter, SingleFilter`

21.9.2 Índice de métodos

`void addSectionFilterListener(FilterEventListener listener)`

Adiciona um `FilterEventListener` para receber eventos do objeto `DataSectionFilter`.

`void clearGreaterThanFilter()`

Limpa e desativa qualquer conjunto previamente estabelecido de parâmetros greaterThan do filtro.

`void clearLessThanFilter()`

Limpa e desativa qualquer conjunto previamente estabelecido de parâmetros lessThan do filtro.

`void clearPositiveFilter()`

Limpa e desativa qualquer conjunto previamente estabelecido de parâmetros positivos do filtro.

`void clearTableId()`

Limpa qualquer valor `table_id` previamente estabelecido.

`void clearXorFilter()`

Limpa e desativa qualquer conjunto previamente estabelecido de parâmetros xor do filtro.

`void removeSectionFilterListener(FilterEventListener listener)`

Remove o `FilterEventListener` especificado, o qual não irá mais remover eventos do `DataSectionFilter`.

`void setGreaterThanFilter(int offset, byte[] greaterThanFilterDef, byte[] greaterThanFilterMask)`

Define um filtro baseado em especificar um conjunto de posições de bits os quais, quando os valores correspondentes na seção de dados estiverem concatenados e interpretados como um valor inteiro, devem ser MAIORES QUE o inteiro gerado a partir do *array* correspondente de mesma definição.

`void setLessThanFilter(int offset, byte[] lessThanFilterDef, byte[] lessThanFilterMask)`

Define um filtro baseado em especificar um conjunto de posições de bits os quais, quando os valores correspondentes na seção de dados estiverem concatenados e interpretados como um valor inteiro, devem ser MENOS QUE o inteiro gerado a partir do *array* correspondente de mesma definição.

`void setPid(int pid)`

Define ou modifica o valor do PID para uso em pacotes de filtragem.

`void setPositiveFilter(int offset, byte[] positiveFilterDef, byte[] positiveFilterMask)`

Define um filtro baseado em especificar um conjunto de posições de bits os quais, quando os valores correspondentes na seção de dados estiverem concatenados e interpretados como um valor inteiro, devem ser IGUAIS AO inteiro gerado a partir do *array* correspondente de mesma definição.

`void setTableId(int table_id)`

Define ou modifica o valor do `table_id` para uso em pacotes de filtragem.

`void setTimeOut(long milliseconds)`

Define o tempo de espera do `DataSectionFilter`.

`void setXorFilter(int offset, byte[] xorData, byte[] xorZeroMask, byte[] xorOneMask)`

Define dados *xor* e *array*s de máscara de tal forma que bits no conteúdo da seção especificado pelo `xorZeroMask` irá avaliar em zero quando operados por *xor* com bits `xorData` correspondentes, e bits especificados pelo `xorOneMask` devem ser avaliados em um quando operados por xor com os bits `xorData` correspondentes.

`void startFiltering(Object refObj)`

Coloca um filtro no estado “started”.

`void stopFiltering()`

Para a filtragem desse `DataSectionFilter` no principal `DataSectionFilterCollection`.

21.9.3 Detalhe dos métodos

setPid

```
public void setPid(int pid)
            throws InvalidFilterException
```

Define ou modifica o valor do PID para uso em pacotes de filtragem. Esse método DEVE ser chamado pelo menos uma vez antes que o filtro seja usado.

Parâmetros:

`pid` - o valor do PID a ser filtrado nas seções de entrada. Deve ser um valor 13 *bit field* entre 0 e 0x1FFF

Lança:

`InvalidFilterException` - se o PID não for um valor válido entre 0 e 0x1FFF

setTableId

```
public void setTableId(int table_id)
                throws InvalidFilterException
```

Define ou modifica o valor do `table_id` para uso em pacotes de filtragem.

Parâmetros:

`table_id` - o valor do `table_id` a ser filtrado nas seções de entrada deve ser um valor 8 bit field entre 0 e 0xFF

Lança:

`InvalidFilterException` - se o `table_id` não for um valor válido entre 0 e 0xFF

clearTableId

```
public void clearTableId()
                  throws InvalidFilterException
```

Limpa qualquer valor `table_id` previamente estabelecido.

Se o filtro estiver anexado a um *stream* de transporte, o filtro não irá mais filtrar no filtro `table_id` de seções de dados.

Lança:

`InvalidFilterException` - Um `TableSectionFilter` deve ter o campo `table_id` definido.

setPositiveFilter

```
public void setPositiveFilter(int offset,
                              byte[] positiveFilterDef,
                              byte[] positiveFilterMask)
                       throws InvalidFilterException
```

Define um filtro baseado em especificar um conjunto de posições de bits os quais, quando os valores correspondentes na seção de dados estiverem concatenados e interpretados como um valor inteiro, devem ser IGUAIS AO inteiro gerado a partir do *array* correspondente de mesma definição.

Se o `DataSectionFilterCollection` associado estiver conectado a um `TransportStream`, a filtragem irá começar imediatamente a usar esses parâmetros maiores que de filtro.

Parâmetros:

`offset` - define o ajuste dentro da seção em que primeiro byte dos *array*s `positiveFilterDef` e `positiveFilterMask` têm que corresponder. O ajuste deve ser maior que ou igual a 3 e menor que 31

`positiveFilterDef` - define um valor inteiro gerado pela seleção de bits de acordo com a máscara suprida e concatenando-os.

`positiveFilterMask` - define quais bits no `positiveFilterDef` deveriam ser selecionados para a comparação igual a.

Lança:

`InvalidFilterException` - se o ajuste não for um valor válido, se os *array*s não forem do mesmo comprimento, se o comprimento do *array* e do ajuste fariam com que o filtro se estendesse além da capacidade do filtro nativo ou se as posições dos bits selecionados nas máscaras já houverem sido escolhidas em um diferente *array* máscara de filtro.

clearPositiveFilter

```
public void clearPositiveFilter()
```

Limpa e desativa qualquer conjunto previamente estabelecido de parâmetros positivos do filtro.

Se o filtro estiver conectado a um *stream* de transporte, os parâmetros positivos do filtro não são mais usados para combinar seções de dados.

setGreaterThanFilter

```
public void setGreaterThanFilter(int offset,
                                 byte[] greaterThanFilterDef,
```

```
                                byte[] greaterThanFilterMask)
                         throws InvalidFilterException
```

Define um filtro baseado em especificar um conjunto de posições de bits os quais, quando os valores correspondentes na seção de dados estiverem concatenados e interpretados como um valor inteiro, devem ser maiores que o inteiro gerado a partir do *array* correspondente de mesma definição.

Se o `DataSectionFilterCollection` associado estiver conectado a um `TransportStream`, a filtragem irá começar imediatamente a usar esses parâmetros maiores que de filtro.

Parâmetros:

`offset` - define o ajuste dentro da seção em que o primeiro byte dos *array*s `greaterThanFilterDef` e `greaterThanFilterMask` têm que corresponder. O ajuste deve ser maior que ou igual a 3 e menor que 31

`greaterThanFilterDef` - define um valor inteiro gerado pela seleção de bits de acordo com a máscara suprida e concatenando-os.

`greaterThanFilterMask` - define quais bits no `greaterThanFilterDef` deveriam ser selecionados para a comparação maior que.

Lança:

`InvalidFilterException` - se o ajuste não for um valor válido, se os *array*s não forem do mesmo comprimento, se o comprimento do *array* e do ajuste fariam com que o filtro se estendesse além da capacidade do filtro nativo ou se as posições dos bits selecionados nas máscaras já houverem sido escolhidas em um diferente *array* máscara de filtro.

clearGreaterThanFilter

```
public void clearGreaterThanFilter()
```

Limpa e desativa qualquer conjunto previamente estabelecido de parâmetros `greaterThan` do filtro.

Se o filtro estiver conectado a um *stream* de transporte, os parâmetros `greaterThan` do filtro não são mais usados para combinar seções de dados.

setLessThanFilter

```
public void setLessThanFilter(int offset,
                              byte[] lessThanFilterDef,
                              byte[] lessThanFilterMask)
                       throws InvalidFilterException
```

Define um filtro baseado em especificar um conjunto de posições de bits os quais, quando os valores correspondentes na seção de dados estiverem concatenados e interpretados como um valor inteiro, devem ser menos que o inteiro gerado a partir do *array* correspondente de mesma definição.

Se o `DataSectionFilterCollection` associado estiver conectado a um `TransportStream`, a filtragem irá começar imediatamente a usar esses parâmetros menores que de filtro.

Parâmetros:

`offset` – define o ajuste dentro da seção em que primeiro byte dos *array*s `lessThanFilterDef` e `lessThanFilterMask` têm que corresponder. O ajuste deve ser menor que ou igual a 3 e menor que 31

`lessThanFilterDef` – define um valor inteiro gerado pela seleção de bits de acordo com a máscara suprida e concatenando-os.

`lessThanFilterMask` - define quais bits no `lessThanFilterDef` deveriam ser selecionados para comparação menor que.

Lança:

`InvalidFilterException` - se o ajuste não for um valor válido, se os *array*s não forem do mesmo comprimento, se o comprimento do *array* e do ajuste fariam com que o filtro se estendesse além da capacidade do

filtro nativo ou se as posições dos bits selecionados nas máscaras já houverem sido escolhidas em um diferente *array* máscara de filtro.

clearLessThanFilter

```
public void clearLessThanFilter()
```

Limpa e desativa qualquer conjunto previamente estabelecido de parâmetros lessThan do filtro.

Se o filtro estiver conectado a um *stream* de transporte, os parâmetros lessThan do filtro não devem ser mais usados para combinar seções de dados.

setXorFilter

```
public void setXorFilter(int offset,
                         byte[] xorData,
                         byte[] xorZeroMask,
                         byte[] xorOneMask)
                  throws InvalidFilterException
```

Define dados *xor* e *array*s de máscara de tal forma que bits no conteúdo da seção especificado pelo `xorZeroMask` irá avaliar em zero quando operados por *xor* com bits `xorData` correspondentes, e bits especificados pelo `xorOneMask` devem ser avaliados em um quando operados por xor com os bits `xorData` correspondentes.

Os *array*s de máscara selecionam bits com valor 0. Bits na máscara com valor 1 não podem ser operados por *xor* e não terão influência sobre uma seção combinando o filtro.

O pseudo código a seguir ilustra como essas operações lógicas combinam uma seção.

```
 if ((sectionData ^ xorData) & ~xorZeroMask == 0 &&
    ((sectionData ^ xorData) & ~xorOneMask) ^ ~xorOneMask > 0) {
    //SECTION MATCH FOUND
 }
```

Observar que para evitar um conflito óbvio, o conteúdo de `xorZeroMask` e `xorOneMask` não deve selecionar os mesmos bits para verificação *xor* (uma vez que um resultado *xor* não pode ser simultaneamente um e zero) Em efeito, isso reforça o requerimento lógico: `(~xorOneMask) & (~xorOneMask) = 0`.

Se o `DataSectionFilterCollection` associado estiver conectado a um `TransportStream`, a filtragem irá começar imediatamente a usar esses parâmetros *xor* de filtro.

Parâmetros:

`offset` – define o ajuste dentro da seção em que primeiro byte dos *array*s `xorData` e `xorZeroMask` têm que corresponder. O ajuste deve ser maior que ou igual a 3 e menor que 31

`xorData` - Define valores que deveriam ser estabelecidos "0" quando bits especificados pelo `xorZeroMask` são operados por *xor* com a seção de dados e estabelecidos "1" quando bits especificados pelo `xorOneMask` são operados por *xor* com a seção de dados.

`xorZeroMask` - Define quais bits na seção devem ser estabelecidos "0" quando operados por *xor* com o `xorData`. Um bit com valor “0" marca o bit a ser operados por *xor*.

`xorZeroMask` – Define quais bits na seção devem ser estabelecidos "1" quando operados por *xor* com o `xorData`. Um bit com valor “0" marca o bit a ser operados por *xor*.

Lança:

`InvalidFilterException` - Se os *array*s não forem do mesmo comprimento, se as `xormasks` sobreporem, se o ajuste não for um valor válido, se o comprimento do *array* e do ajuste fariam com que o filtro se estendesse além

da capacidade do filtro nativo ou se as posições dos bits selecionados nas máscaras já houverem sido escolhidas em um diferente *array* máscara de filtro.

clearXorFilter

```
public void clearXorFilter()
```

Limpa e desativa qualquer conjunto previamente estabelecido de parâmetros *xor* do filtro.

Se o filtro estiver conectado a um *stream* de transporte, os parâmetros *xor* do filtro não podem ser mais usados para combinar seções de dados.

startFiltering

```
public void startFiltering(Object refObj)
                    throws FilterResourceUnavailableException,
                           ConditionalAccessDeniedException,
                           InvalidFilterException,
                           DisconnectedException
```

Coloca um filtro no estado "`started`". Uma vez que o filtro for anexado a um *stream* de transporte por meio de um método `DataFilterCollection` chamando `connect()`, os recursos nativos do filtro são alocados e a filtragem começa com os parâmetros definidos por esse filtro. Se o filtro já é parte de um `DataSectionFilterCollection` anexado, os parâmetros desse filtro se tornam ativos.

Parâmetros:

`refObj` - Um objeto suprido pelo aplicativo para uso como referência interna dentro do aplicativo para identificar unicamente a operação do filtro. Se não for `null`, o objeto deve ser provido com todas as entregas.`DataSectionFilterEvent`.

Lança:

`FilterResourceUnavailableException` - se todo o número de `DataSectionFilter`s iniciados para o `DataSectionFilterCollection` contido já for igual ao número dos filtros de seções reservados para o `DataSectionFilterCollection` quando ele foi criado.

`ConditionalAccessDeniedException` - se a informação requisitada estiver misturada e a permissão para separá-la for negada.

`InvalidFilterException` - Parâmetros insuficientes de filtro foram supridos para permitir que o filtro começasse. Um `ListFilter` deve ter o campo `table_id` definido. Lançado também se o ajuste e comprimento dos *array*s `xorData`, `xorZeroMask` e `xorOneMask` fizerem com que filtro se estenda além da capacidade do filtro nativo.

`DisconnectedException` - se o `DataSectionFilterCollection` principal houver perdido sua conexão com um *stream* de transporte desde que foi conectado.

stopFiltering

```
public void stopFiltering()
```

Para a filtragem desse `DataSectionFilter` no principal `DataSectionFilterCollection`. Conectar o `DataSectionFilterCollection` principal a um *stream* de transporte não iniciará a filtragem desse `DataSectionFilter`.

setTimeOut

```
public void setTimeOut(long milliseconds)
                throws IllegalArgumentException
```

Define o tempo de espera do `DataSectionFilter`. Um `TimeOutEvent` deve ser enviado para todos

FilterEventListener, quando o tempo de espera ocorrer. A filtragem também deve ser parada.

Para um SingleFilter isso deve ser gerado se não chegarem seções dentro do período especificado. Para um ListFilter, isso deve ser gerado se a tabela completa não chegar dentro do tempo especificado. Para um CircularFilter, isso deve ser gerado se o tempo especificado houver decorrido desde a chegada da última seção filtrada com sucesso.

Especificar um tempo de espera de 0 remove o tempo de espera, resultando em nenhum TimeOutEvent adicional nas ativações de filtro subseqüentes, mas não nas ativações de filtro que já estiverem em progresso. O valor de tempo de espera padrão é 0.

Parâmetros:

milliseconds - O período do tempo de espera.

Lança:

IllegalArgumentException - Se o parâmetro 'milissegundos' estiver negativo.

addSectionFilterListener

```
public void addSectionFilterListener(FilterEventListener listener)
```

Adiciona um FilterEventListener para receber eventos do objeto DataSectionFilter.

Parâmetros:

listener - O FilterEventListener para receber os eventos.

Relaciona-se com:

removeSectionFilterListener()

removeSectionFilterListener

```
public void removeSectionFilterListener(FilterEventListener listener)
```

Remove o FilterEventListener especificado, o qual não irá mais remover eventos do DataSectionFilter.

Parâmetros:

listener - o FilterEventListener a ser removido.

Relaciona-se com:

addSectionFilterListener().

21.10 Classe DataSectionFilterCollection

21.10.1 Descrição da classe

com.sun.dtv.filtering

```
java.lang.Object
  └com.sun.dtv.filtering.DataSectionFilterCollection
```

```
public class DataSectionFilterCollection
extends Object
implements ScarceResource
```

Essa classe representa uma coleção de filtros de seções de dados a serem ativados e desativados como uma operação atômica. O propósito dessa classe é de minimizar o potencial para impasses de recurso entre peças independentes do(s) aplicativo(s).

Observar que devido ao propósito particular dessa classe e embora esteja herdando de `ScarceResource`, ela não provê quaisquer métodos `getInstances()` ou `reserveOne()` como descrito na descrição da interface `ScarceResource`.

Todas as interfaces implementadas:

`ScarceResource`

21.10.2 Exemplo de código

Criando um conjunto de filtros com um filtro *XOR* circular a iniciando a filtragem.

```
CircularFilter circularFilter;


 void demo()
      throws IllegalArgumentException,
             TimeoutException,
             IllegalStateException,
             IncompatibleSourceException,
             TuningException,
             InvalidFilterException {
     com.sun.dtv.transport.TransportStream transportStream = null;


     // cria       uma       coleção       de       filtros       com       3       filtros
     DataSectionFilterCollection collection = new DataSectionFilterCollection(1);

     collection.reserve(true, 10000, new ScarceResourceListener() {

         public boolean releaseRequested(ScarceResource resource) {
             return true;
         }

         public void releaseForced(ScarceResource resource) {
         }

         public void released(ScarceResource resource) {
         }
     });

     // agora cria um filtro circular XOR

     circularFilter = collection.newCircularFilter(1);
     byte[] xorData = { 0x01, 0x02, 0x03, 0x04 };
     byte[] xorZeroMask = { (byte) 0xFF, (byte) 0x00, (byte) 0xFF, (byte) 0x00};
     byte[] xorOneMask = { (byte) 0x00, (byte) 0xF0, (byte) 0x00, (byte) 0xFF};
     circularFilter.setXorFilter(10, xorData, xorZeroMask, xorOneMask);

     // cria um audio para filtro de eventos

     circularFilter.addSectionFilterListener(new FilterEventListener() {

         public void dataSectionFilterUpdate(DataSectionFilterEvent event) {
             if (event instanceof DataSectionAvailableEvent) {
                   DataSection section = circularFilter.getSections()[0];
                   // faz       alguma       coisa       com       a       seção       filtrada
               }
           }
     });
```

```
    // conecta com o stream de transporte e iniciará a filtragem
    // (transportStream pode ser recuperado do sintonizador ou aplicação)
    collection.connect(transportStream, this);
    circularFilter.startFiltering(this);
}
```

Relaciona-se com:

`DataSection`, `DataSectionFilter`

21.10.3 Índice de construtores

`DataSectionFilterCollection``(TransportStream stream, int numberOfFilters)`

Cria um objeto conjunto de filtros de seções com um número associado de filtros de seções de dados no *stream* de transporte especificado que necessita ser reservado quando o objeto estiver para ser conectado a uma fonte ativa de dados de seções.

`DataSectionFilterCollection``(TransportStream stream, int numberOfFilters, boolean highPriority)`

Cria um objeto conjunto de filtros de seções com um número associado de filtros de seções de dados no *stream* de transporte especificado que necessita ser reservado quando o objeto estiver para ser conectado a uma fonte ativa de seções de dados.

21.10.4 Índice de métodos

`static void addResourceTypeListener(ResourceTypeListener listener)`

Adiciona um `ResourceTypeListener` a implementação.

`void` **`connect`**`(Object requestData)`

Conecta um `DataSectionFilterCollection` ao *stream* de transporte.

`void disconnect()`

Retorna um `DataSectionFilterCollection` ao estado desconectado.

`TransportStream getSource()`

Retorna o *stream* de transporte no qual um `DataSectionFilterCollection` é usado para filtrar.

`boolean isAvailable()`

Verifica se o dado recurso está correntemente disponível para reserva.

`CircularFilter newCircularFilter(int numberOfEntries)`

Cria um novo filtro de cadeia de seções dentro do conjunto de filtros de seções principal.

`CircularFilter newCircularFilter(int numberOfEntries, int sectionSize)`

Cria um novo filtro de cadeia de seções dentro do conjunto de filtros de seções principal.

`ListFilter newListFilter()`

Cria um novo objeto tabela de filtros de seções dentro do conjunto de filtros de seções principal.

`ListFilter newListFilter(int sectionSize)`

Cria um novo objeto tabela de filtros de seções dentro do conjunto de filtros de seções principal.

`SingleFilter newSingleFilter()`

Cria um novo objeto filtro de seção única dentro do conjunto de filtros de seções principal.

```
SingleFilter newSingleFilter(int sectionSize)
```

Cria um novo objeto filtro de seção simples dentro do conjunto de filtros de seções principal.

```
void release()
```

Libera esse recurso.

```
static void removeResourceTypeListener(ResourceTypeListener listener)
```

Remove um *listener* previamente anexado.

```
void reserve(boolean force, long timeoutms, ScarceResourceListener listener)
```

Requisita reserva da dada instância de recursos escassos.

21.10.5 Detalhe dos construtores

DataSectionFilterCollection

```
public DataSectionFilterCollection(TransportStream stream,
                                   int numberOfFilters)
```

Cria um objeto conjunto de filtros de seções com um número associado de filtros de seções de dados no *stream* de transporte especificado que necessita ser reservado quando o objeto estiver para ser conectado a uma fonte ativa de dados de seções.

O objeto terá uma prioridade alta de recursos padrão para o caso do número de filtros de seção disponíveis ao pacote se tornar insuficiente.

Parâmetros:

`stream` - especifica a fonte de seções de dados a serem filtrados.

`numberOfFilters` - o número de filtros de seções necessário para o objeto.

DataSectionFilterCollection

```
public DataSectionFilterCollection(TransportStream stream,
                                   int numberOfFilters,
                                   boolean highPriority)
```

Cria um objeto conjunto de filtros de seções com um número associado de filtros de seções de dados no *stream* de transporte especificado que necessita ser reservado quando o objeto estiver para ser conectado a uma fonte ativa de seções de dados.

Parâmetros:

`stream` - especifica a fonte de seções de dados a serem filtrados

`numberOfFilters` - o número de filtros de seções necessário para o objeto.

`highPriority` - A prioridade de recursos do objeto para o caso do número de filtros de seções disponíveis ao pacote se tornar insuficiente. Prioridade alta é indicada por `true` e prioridade baixa por *`false`*.

21.10.6 Detalhe dos métodos

newSingleFilter

```
public SingleFilter newSingleFilter()
```

Cria um novo objeto filtro de seção única dentro do conjunto de filtros de seções principal. Na ativação (`startFiltering` bem sucedido) o objeto `SingleFilter` usará filtros de seções do local especificado quando o `DataSectionFilterCollection` principal foi criado. O objeto filtro de seções terá um buffer adequado para manter uma seção longa padrão `newSimpleSectionFilter`.

Retorna:

o novo objeto `SingleFilter`.

newSingleFilter

```
public SingleFilter newSingleFilter(int sectionSize)
```

Cria um novo objeto filtro de seção simples dentro do conjunto de filtros de seções principal. Na ativação (`startFiltering` bem sucedido) o objeto `SingleFilter` usará filtros de seções do local especificado quando o `DataSectionFilterCollection` principal foi criado.

Parâmetros:

`sectionSize` - especifica o tamanho em bytes do buffer a ser criado para manter dados capturados pelo `DataSectionFilter`. Se seções filtradas que são maiores que isso, os dados extras devem ser descartados e a filtragem continuará sem qualquer notificação para o aplicativo.

Retorna:

o novo objeto `SingleFilter`.

newCircularFilter

```
public CircularFilter newCircularFilter(int numberOfEntries)
```

Cria um novo filtro de cadeia de seções dentro do conjunto de filtros de seções principal. Na ativação (`startFiltering` bem sucedido) o novo objeto `CircularFilter` usará filtros de seções do local especificado quando o `DataSectionFilterCollection` principal foi criado.

Parâmetros:

`numberOfEntries` - O número de objetos `DataSection` a serem criados para uso no buffer circular.

Retorna:

o novo objeto `CircularFilter`.

newCircularFilter

```
public CircularFilter newCircularFilter(int numberOfEntries,
                                        int sectionSize)
```

Cria um novo filtro de cadeia de seções dentro do conjunto de filtros de seções principal. Na ativação (`startFiltering` bem sucedido) o novo objeto `CircularFilter` usará filtros de seções do local especificado quando o `DataSectionFilterCollection` principal foi criado.

Parâmetros:

`numberOfEntries` – O número de objetos `DataSection` a serem criados para uso no buffer circular.

`sectionSize` - o tamanho em bytes do buffer para cada objeto Seção. Se seções filtradas que são maiores que isso, os dados extras devem ser descartados e a filtragem continuará sem qualquer notificação para o aplicativo.

Retorna:

o novo objeto `CircularFilter`.

newListFilter

```
public ListFilter newListFilter()
```

Cria um novo objeto tabela de filtros de seções dentro do conjunto de filtros de seções principal. Na ativação (`startFiltering` bem sucedido) o novo objeto `ListFilter` usará filtros de seções do local especificado quando o `DataSectionFilterCollection` principal foi criado. Cada `DataSection` criado para objeto tabela de filtros de seções terá um buffer adequado para manter um seção longa padrão.

Retorna:

o novo objeto `ListFilter`.

newListFilter

```
public ListFilter newListFilter(int sectionSize)
```

Cria um novo objeto tabela de filtros de seções dentro do conjunto de filtros de seções principal. Na ativação (`startFiltering` bem sucedido) o novo objeto `ListFilter` usará filtros de seções do local especificado quando o `DataSectionFilterCollection` principal foi criado.

Parâmetros:

`sectionSize` - especifica o tamanho em bytes do buffer a ser criado para manter dados capturados pelo `DataSectionFilter`. Quando a primeira seção for capturada e o número total de seções na tabela for conhecido, cada `DataSection` criado terá um buffer desse tamanho. Se seções filtradas que são maiores que isso, os dados extras devem ser descartados e a filtragem continuará sem qualquer notificação para o aplicativo.

Retorna:

o novo objeto `ListFilter`.

connect

```
public void connect(Object requestData)
             throws IllegalStateException,
                    IncompatibleSourceException,
                    TuningException
```

Conecta um `DataSectionFilterCollection` ao *stream* de transporte. O `DataSectionFilterCollection` tem de reservar o número de filtros de seções no *stream* de transporte especificado antes. Quaisquer objetos `DataSectionFilter` que são partes do conjunto concernente e cuja filtragem foi iniciada se tornará ativo e iniciará a filtragem da fonte em busca de seções correspondendo aos padrões especificados. Uma chamada para `connect` em um `DataSectionFilterCollection` conectado deve ser tratada como uma desconexão seguida pela nova conexão.

Parâmetros:

`requestData` - dados específicos do aplicativo para uso da API de notificação de recursos.

Lança:

`IllegalStateException` - se o conjunto especificado de filtros de seções não houver sido reservado antes.

`IncompatibleSourceException` - se a fonte não é uma fonte válida de seções de dados.

`TuningException` - se a fonte não estiver correntemente sintonizada para

disconnect

```
public void disconnect()
                throws IllegalStateException
```

Retorna um `DataSectionFilterCollection` ao estado desconectado. Quando chamado para um `DataSectionFilterCollection` no estado conectado, desconecta um `DataSectionFilterCollection` de uma fonte de seções de dados. Os filtros de seções mantidos pelo `DataSectionFilterCollection` não podem ser liberados de volta ao ambiente. Quaisquer operações de filtragem rodando devem ser terminadas. Esse método não terá efeito para `DataSectionFilterCollection`s que já estiverem no estado desconectado.

Lança:

`IllegalStateException` - se o `DataSectionFilterCollection` não houver sido conectado antes.

getSource

```
public TransportStream getSource()
```

Retorna o *stream* de transporte no qual um `DataSectionFilterCollection` é usado para filtrar.

Retorna:

o Fluxo de Transporte a ser filtrado

reserve

```
public void reserve(boolean force,
                    long timeoutms,
                    ScarceResourceListener listener)
             throws IllegalArgumentException,
                    TimeoutException,
                    SecurityException
```

Requisita reserva da dada instância de recursos escassos. O método bloquear-se-á até que a instância esteja disponível. O método retorna normalmente apenas se a reserva for bem sucedida. Todos os outros casos são manuseados por exceções.

Primeiro, se houver um gerenciador de segurança, seu método `checkPermission` é chamado com a permissão `ScarceResourcePermission(name, "reserve")`. Se `force` é além disso definido `true`, a permissão também verificado no `ScarceResourcePermission(name, "force")`.

Durante o processo de reserva, se a instância do recurso já estiver alocada, a implementação deve notificar o proprietário atual do recurso sobre a requisição de reserva por meio da interface `ScarceResourceListener`:

ou por `ScarceResourceListener.releaseRequested()` se forçar for *false*,

ou por `ScarceResourceListener.releaseForced()` no outro caso.

O *listener* deve ser usado para tais notificações apenas até o recurso ser liberado. Após a liberação, a implementação não deve chamar quaisquer dos métodos da interface do *listener*.

Após esse primeiro evento, a implementação procederá de acordo e liberará (ou não) o recurso requisitado. No caso da implementação liberar o recurso, desencadeará um evento `ScarceResourceListener.released()` ao proprietário *listener* original do recurso informando-o que o recurso não lhe pertence mais.

O aplicativo pode controlar o tempo de espera para que tal recurso esteja disponível ao definir `timeoutms`. No caso da duração da reserva exceder o tempo expresso em `timeoutms`, um `TimeoutException` é lançado.

Sob operação normal, recursos são liberados usando o método `release`. De qualquer forma, no caso do aplicativo não liberar recursos quando requisitado ou o aplicativo ser terminado, a implementação deve liberar todos os recursos alocados para o aplicativo para permitir que outros aplicativos sejam notificados de mudanças na alocação de recursos e poderem reservá-los. Ver a seção Resource Cleanup do ciclo de vida do aplicativo.

Especificado por:

`reserve` na interface `ScarceResource`

Parâmetros:

`force` - Se `true`, esse método retirará o recurso do proprietário atual. Se *`false`*, a implementação bloquear-se-á e esperará até que o recurso esteja disponível (usando `release()`) ou até `timeoutms`.

`timeoutms` - Uma quantidade positiva de milissegundos que esse método esperará para que o recurso seja

liberado pelo proprietário atual. O valor `-1` indica que a implementação esperará para sempre.

`listener` - O *listener* a ser anexado para receber notificação sobre o status do recurso.

Lança:

`IllegalArgumentException` - Se o *listener* estiver `null` ou se o valor especificado em `timeoutms` não for válido, isto é, nem um inteiro positivo, nem -1.

`TimeoutException` - Se o tempo especificado em `timeoutms` houver acabado e o recurso não houver podido ser reservado.

`SecurityException` - Se o aplicativo não tiver permissão para ação de reserva para o recurso que está prestes a reservar. Também lançado se forçar for definido true, mas o aplicativo não tiver permissão para a ação forçar.

release

```
public void release()
```

Libera esse recurso.

O recurso deve ser disponibilizado para outro aplicativo através da plataforma. Após chamar esse método, não é mais possível interagir com o dado recurso: chamadas para métodos críticos desses recursos escassos devem ser ignorados e lançar `IllegalStateException`. Essa afirmação é válida e é o comportamento requerido de qualquer classe implementando a interface `ScarceResource`. Para interagir novamente com o dado recurso, o aplicativo deve chamar o método `reserve()` e se tornar o proprietário novamente.

A implementação pode organizar quaisquer recursos do sistema que esse objeto estiver usando. Depois a implementação não deve chamar quaisquer dos métodos do *listener* que estava anexado no momento da reserva.

Se o recurso já estava disponível (isto é, não reservado), esse método não faz nada.

Especificado por:

`release` na interface `ScarceResource`

isAvailable

```
public boolean isAvailable()
```

Verifica se o dado recurso está correntemente disponível para reserva. O valor retornado dá a situação atual e não garante que o recurso ainda estará disponível em um momento posterior, por exemplo, no momento da reserva: outro aplicativo pode ter tomado aquele recurso no tempo decorrido.

Especificado por:

`isAvailable` na interface `ScarceResource`

Retorna:

Um boolean definido `true` se o dado recurso estiver atualmente disponível para reserva.

addResourceTypeListener

```
public static void addResourceTypeListener(ResourceTypeListener listener)
                                    throws NullPointerException
```

Adiciona um `ResourceTypeListener` a implementação. Sempre que um `reserve()` ou um `release()` for chamado para quaisquer recursos do mesmo tipo, a implementação chamará os métodos correspondentes do *listener*.

Parâmetros:

`listener` - O *listener* desencadeado para eventos em recursos do mesmo tipo.

Lança:

`NullPointerException` - Se o *listener* estiver `null`.

Relaciona-se com:

`removeResourceTypeListener()`

removeResourceTypeListener

```
public static void removeResourceTypeListener(ResourceTypeListener listener)
                                        throws NullPointerException
```

Remove um *listener* previamente anexado. Se o *listener* não foi anexado antes, esse método não faz nada..

Parâmetros:

`listener` - O *listener* desencadeado para eventos em recursos do mesmo tipo.

Lança:

`NullPointerException` - Se o *listener* estiver `null`.

Relaciona-se com:

`addResourceTypeListener()`

21.11 Classe DataSectionFilterEvent

21.11.1 Descrição da classe

`com.sun.dtv.filtering`

```
java.lang.Object
  └java.util.EventObject
      └com.sun.dtv.filtering.DataSectionFilterEvent
```

```
public class DataSectionFilterEvent
extends EventObject
```

Essa classe é a classe base para Eventos na API filtro de seções.

Todas as interfaces implementadas

`Serializable`

Subclasses diretas conhecidas

`DataSectionAvailableEvent, FilteringStoppedEvent`

21.11.2 Índice de construtores

`DataSectionFilterEvent``(DataSectionFilter f, Object refObj)`

Constrói um `DataSectionFilterEvent` para o objeto `DataSectionFilter` especificado.

21.11.3 Índice de métodos

`Object` **`getrefObj`**`()`

Retorna os dados do aplicativo que foram passados ao método `startFiltering`.

`Object` **`getSource`**`()`

Isso retorna o objeto `DataSectionFilter` que foi a fonte do evento.

21.11.4 Detalhe dos construtores

DataSectionFilterEvent

```
public DataSectionFilterEvent(DataSectionFilter f,
                              Object refObj)
```

Constrói um `DataSectionFilterEvent` para o objeto `DataSectionFilter` especificado.

Parâmetros:

`f` - o objeto `DataSectionFilter` onde o evento foi originado.

`refObj` - dados do aplicativo que foram passados ao método `startFiltering`.

21.11.5 Detalhe dos métodos

getSource

```
public Object getSource()
```

Isso retorna o objeto `DataSectionFilter` que foi a fonte do evento.

Substituições:

`getSource` na classe `EventObject`

Retorna:

o `DataSectionFilter`

getrefObj

```
public Object getrefObj()
```

Retorna os dados do aplicativo que foram passados ao método `startFiltering`.

Retorna:

os dados do aplicativo.

21.12 Classe DataSectionFilterException

21.12.1 Descrição da classe

`com.sun.dtv.filtering`

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └com.sun.dtv.filtering.DataSectionFilterException
```

```
public class DataSectionFilterException
```

`extends Exception`

A classe base para exceções lançadas pelas API de filtragem de seções de dados.

Todas as interfaces implementadas

`Serializable`

Subclasses diretas conhecidas

`DataUnavailableException, DisconnectedException, FilterInterruptException, IncompatibleSourceException, InvalidFilterException`

21.12.2 Índice de construtores

`DataSectionFilterException``()`

Constrói uma `DataSectionFilterException`.

`DataSectionFilterException``(String s)`

Constrói um `DataSectionFilterException` com uma mensagem detalhada.

21.12.3 Detalhe dos construtores

DataSectionFilterException

`public DataSectionFilterException()`

Constrói uma `DataSectionFilterException`.

DataSectionFilterException

`public DataSectionFilterException(String s)`

Constrói um `DataSectionFilterException` com uma mensagem detalhada.

Parâmetros:

`s` - a mensagem

21.13 Classe DataUnavailableException

21.13.1 Descrição da classe

`com.sun.dtv.filtering`

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └com.sun.dtv.filtering.DataSectionFilterException
              └com.sun.dtv.filtering.DataUnavailableException
```

```
public class DataUnavailableException
extends DataSectionFilterException
```

Indica que não há dados disponíveis em um objeto `DataSection`.

Todas as interfaces implementadas

`Serializable`

21.13.2 Índice de construtores

`DataUnavailableException``()`

Constrói um `DataUnavailableException` sem mensagem detalhada.

`DataUnavailableException``(String s)`

Constrói um `DataUnavailableException` com a mensagem detalhada especificada.

21.13.3 Detalhe dos construtores

DataUnavailableException

```
public DataUnavailableException()
```

Constrói um `DataUnavailableException` sem mensagem detalhada.

DataUnavailableException

```
public DataUnavailableException(String s)
```

Constrói um `DataUnavailableException` com a mensagem detalhada especificada.

Parâmetros:

`s` - a mensagem detalhada.

21.14 Classe DisconnectedException

21.14.1 Descrição da classe

`com.sun.dtv.filtering`

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └com.sun.dtv.filtering.DataSectionFilterException
              └com.sun.dtv.filtering.DisconnectedException
public class DisconnectedException
extends DataSectionFilterException
```

Indica que um `DataSectionFilterCollection` perdeu sua conexão, causando a falha de uma chamada startFiltering. Só é gerada para `DataSectionFilterCollections` que estiver desconectado.

Todas as interfaces implementadas

`Serializable`

21.14.2 Índice de construtores

`DisconnectedException``()`

Constrói um `DisconnectedException`.

`DisconnectedException``(String s)`

Constrói um `DisconnectedException` com uma mensagem detalhada.

21.14.3 Detalhe dos construtores

DisconnectedException

`public DisconnectedException()`

Constrói um `DisconnectedException`.

DisconnectedException

`public DisconnectedException(String s)`

Constrói um `DisconnectedException` com uma mensagem detalhada.

Parâmetros:

`s` - a mensagem

21.15 Interface FilterEventListener

21.15.1 Descrição da interface

`com.sun.dtv.filtering`

`public interface FilterEventListener`

`extends EventListener`

A interface desse *listener* pode ser implementada por classes requerendo eventos filtradores.

Todas as super-interfaces

`EventListener`

21.15.2 Índice de métodos

`void` **`dataSectionFilterUpdate`**`(DataSectionFilterEvent event)`

Esse método deve ser chamado para todos os *listeners* registrados com um filtro, suprindo-os com o evento específico que houver ocorrido.

21.15.3 Detalhe dos métodos

dataSectionFilterUpdate

`void dataSectionFilterUpdate(DataSectionFilterEvent event)`

Esse método deve ser chamado para todos os *listeners* registrados com um filtro, suprindo-os com o evento específico que houver ocorrido.

Parâmetros:

`event` - o evento

21.16 Classe FilteringStoppedEvent

21.16.1 Descrição da classe

`com.sun.dtv.filtering`

```
java.lang.Object
  └java.util.EventObject
      └com.sun.dtv.filtering.DataSectionFilterEvent
          └com.sun.dtv.filtering.FilteringStoppedEvent
```

`public class FilteringStoppedEvent`

`extends DataSectionFilterEvent`

Essa classe é usada pra relatar o fim de uma operação de filtragem com uma exceção: Não é gerado quando a filtragem pára para um `SimpleSectionFilter` sob circunstâncias normais (isto é, após uma seção ser filtrada com sucesso).

Relaciona-se com:

`DataSectionFilterEvent`

Todas as interfaces implementadas

`Serializable`

Subclasses diretas conhecidas:

`FilterTimedOutEvent`

21.16.2 Índice de campos

`static int` **`CAUSE_FILTER_INCOMPLETE`**

Causa de filtro incompleto

`static int` **`CAUSE_FILTER_RESOURCE_LOST`**

Os recursos de filtros de seções são removidos de um `DataSectionFilterCollection` conectado.

`static int` **`CAUSE_FILTER_TIMEOUT`**

Essa causa é gerada se o tempo de espera das operações do filtro de seção esgotar dentro do período

especificado pelo método `setTimeOut()`.

`static int` **`CAUSE_TRANSPORT_STREAM_DISCONNECTED`**

Um `TransportStream` que estivava anexado a um `DataSectionFilterCollection` se desconecta ou se torna indisponível, fazendo com que a filtragem pare.

`static int` **`CAUSE_UNKNOWN`**

Causa desconhecida

`static int` **`CAUSE_VERSION_CHANGE_DETECTED`**

Essa causa é usada por `ListFilter` para relatar que uma seção foi encontrada e tem um version_number diferente de seções anteriores.

21.16.3 Índice de construtores

`FilteringStoppedEvent``(DataSectionFilter f, Object refObj)`

Isso constrói um `FilteringStoppedEvent` para o objeto `DataSectionFilter` especificado.

`FilteringStoppedEvent``(DataSectionFilter f, Object refObj, int cause)`

Isso constrói um `FilteringStoppedEvent` para o objeto `DataSectionFilter` especificado.

21.16.4 Índice de métodos

`int` **`getCause`**`()`

Define a causa da operação de filtragem ter parado.

Métodos herdados da classe `com.sun.dtv.filtering.`**`DataSectionFilterEvent`**

`getrefObj, getSource`

21.16.5 Detalhe dos campos

CAUSE_UNKNOWN

```
public static final int CAUSE_UNKNOWN = 0
```

Causa desconhecida

CAUSE_FILTER_INCOMPLETE

```
public static final int CAUSE_FILTER_INCOMPLETE = 1
```

Causa de filtro incompleto

CAUSE_FILTER_TIMEOUT

```
public static final int CAUSE_FILTER_TIMEOUT = 2
```

Essa causa é gerada se o tempo de espera das operações do filtro de seção esgotar dentro do período especificado pelo método `setTimeOut()`.

Para um `SimpleSectionFilter` isso deve ser gerado se não chegarem seções dentro do período especificado.

Para um `TableSectionFilter`, isso deve ser gerado se a tabela completa não chegar dentro do tempo

especificado.

Para um `RingSectionFilter`, isso deve ser gerado se o tempo especificado houver decorrido desde a chegada da última seção filtrada com sucesso.

CAUSE_VERSION_CHANGE_DETECTED

```
public static final int CAUSE_VERSION_CHANGE_DETECTED = 3
```

Essa causa é usada por `ListFilter` para relatar que uma seção foi encontrada e tem um version_number diferente de seções anteriores. Só é gerado uma vez por ação de filtragem. A seção com um version_number diferente é ignorada.

CAUSE_FILTER_RESOURCE_LOST

```
public static final int CAUSE_FILTER_RESOURCE_LOST = 4
```

Os recursos de filtros de seções são removidos de um `DataSectionFilterCollection` conectado.

CAUSE_TRANSPORT_STREAM_DISCONNECTED

```
public static final int CAUSE_TRANSPORT_STREAM_DISCONNECTED = 5
```

Um `TransportStream` que estivava anexado a um `DataSectionFilterCollection` se desconecta ou se torna indisponível, fazendo com que a filtragem pare.

21.16.6 Detalhe dos construtores

FilteringStoppedEvent

```
public FilteringStoppedEvent(DataSectionFilter f,
                             Object refObj)
```

Isso constrói um `FilteringStoppedEvent` para o objeto `DataSectionFilter` especificado.

Parâmetros:

`f` - o objeto `DataSectionFilter` onde o evento se originou

`refObj` - dados do aplicativo que foram passados ao método `startFiltering`

FilteringStoppedEvent

```
public FilteringStoppedEvent(DataSectionFilter f,
                             Object refObj,
                             int cause)
```

Isso constrói um `FilteringStoppedEvent` para o objeto `DataSectionFilter` especificado.

Esse construtor permite que a causa do filtro ter parado seja especificada

Parâmetros:

`f` - o objeto `DataSectionFilter` onde o evento se originou

`refObj` - dados do aplicativo que foram passados ao método `startFiltering`

`cause` - a causa do filtro ter parado

21.16.7 Detalhe dos métodos

getCause

```
public int getCause()
```

Define a causa da operação de filtragem ter parado.

Retorna:

a enumeração da causa

21.17 Classe FilterInterruptException

21.17.1 Descrição da classe

com.sun.dtv.filtering

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └com.sun.dtv.filtering.DataSectionFilterException
              └com.sun.dtv.filtering.FilterInterruptException
```

```
public class FilterInterruptException
extends DataSectionFilterException
```

Indica que um filtro foi interrompido antes que a quantidade requerida de dados fosse filtrada.

Todas as interfaces implementadas

```
Serializable
```

21.17.2 Índice de construtores

`FilterInterruptException``()`

Constrói um `FilterInterruptException`.

`FilterInterruptException``(String s)`

Constrói um `FilterInterruptException` com uma mensagem detalhada.

21.17.3 Detalhe dos construtores

FilterInterruptException

```
public FilterInterruptException()
```

Constrói um `FilterInterruptException`.

FilterInterruptException

```
public FilterInterruptException(String s)
```

Constrói um `FilterInterruptException` com uma mensagem detalhada.

Parâmetros:

s - a mensagem.

21.18 Classe FilterResourceUnavailableException

21.18.1 Descrição da classe

com.sun.dtv.filtering

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └com.sun.dtv.filtering.FilterResourceUnavailableException
```

```
public class FilterResourceUnavailableException
extends Exception
```

Indica que requerimentos de recursos não podem ser alcançados por uma operação de filtro. Pode ser lançada se um `DataSectionFilterCollection` estiver conectado ou não.

Todas as interfaces implementadas

`Serializable`

21.18.2 Índice de construtores

`FilterResourceUnavailableException``()`

Constrói um `FilterResourceUnavailableException`.

`FilterResourceUnavailableException``(String s)`

Constrói um `FilterResourceUnavailableException` com uma mensagem detalhada.

21.18.3 Detalhe dos construtores

FilterResourceUnavailableException

```
public FilterResourceUnavailableException()
```

Constrói um `FilterResourceUnavailableException`.

FilterResourceUnavailableException

```
public FilterResourceUnavailableException(String s)
```

Constrói um `FilterResourceUnavailableException` com uma mensagem detalhada.

Parâmetros:

`s` - a mensagem.

21.19 Classe FilterTimedOutEvent

21.19.1 Descrição da classe

com.sun.dtv.filtering

```
java.lang.Object
  └java.util.EventObject
      └com.sun.dtv.filtering.DataSectionFilterEvent
          └com.sun.dtv.filtering.FilteringStoppedEvent
              └com.sun.dtv.filtering.FilterTimedOutEvent
```

```
public class FilterTimedOutEvent
extends FilteringStoppedEvent
```

Esse evento é gerado se o tempo de espera das operações do filtro de seção esgotar dentro do período especificado pelo método `setTimeOut()`.

Para um `SingleFilter` isso deve ser gerado se seções de dados insuficientes forem filtrados dentro do período do tempo de espera originalmente especificado.

Para um `ListFilter` isso deve ser gerado se o conjunto completo de seções de dados não for filtrado dentro do período do tempo de espera originalmente especificado.

Para um `CircularFilter` isso deve ser gerado se uma seção de dados completa não for filtrada dentro do período do tempo de espera originalmente especificado.

Relaciona-se com:

`DataSectionFilter`, `FilteringStoppedEvent`, `SingleFilter`, `ListFilter`, `CircularFilter`

Campos herdados da classe `com.sun.dtv.filtering.`**`FilteringStoppedEvent`**

`CAUSE_FILTER_INCOMPLETE, CAUSE_FILTER_RESOURCE_LOST, CAUSE_FILTER_TIMEOUT, CAUSE_TRANSPORT_STREAM_DISCONNECTED, CAUSE_UNKNOWN, CAUSE_VERSION_CHANGE_DETECTED`

Todas as interfaces implementadas

`Serializable`

21.19.2 Índice de construtores

`FilterTimedOutEvent``(DataSectionFilter f, Object refObj)`

Isso constrói um `FilterTimedOutEvent` para o `DataSectionFilter` suprido.

Métodos herdados da classe `com.sun.dtv.filtering.`**`FilteringStoppedEvent`**

`getCause`

Métodos herdados da classe `com.sun.dtv.filtering.`**`DataSectionFilterEvent`**

`getrefObj, getSource`

21.19.3 Detalhe dos construtores

FilterTimedOutEvent

```
public FilterTimedOutEvent(DataSectionFilter f,
                           Object refObj)
```

Isso constrói um `FilterTimedOutEvent` para o `DataSectionFilter` suprido.

Parâmetros:

`f` - a fonte do evento

`refObj` - o objeto retorno de chamada registrado na chamada startFiltering original.

21.20 Classe IncompatibleSourceException

21.20.1 Descrição da classe

`com.sun.dtv.filtering`

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └com.sun.dtv.filtering.DataSectionFilterException
              └com.sun.dtv.filtering.IncompatibleSourceException
```

```
public class IncompatibleSourceException
extends DataSectionFilterException
```

Indica que o *stream* fonte provido é incompatível.

Todas as interfaces implementadas

`Serializable`

21.20.2 Índice de construtores

`IncompatibleSourceException``()`

Constrói uma `IncompatibleSourceException`.

`IncompatibleSourceException``(String s)`

Constrói uma `IncompatibleSourceException` com uma mensagem detalhada.

21.20.3 Detalhe dos construtores

IncompatibleSourceException

```
public IncompatibleSourceException()
```

Constrói uma `IncompatibleSourceException`.

IncompatibleSourceException

`public IncompatibleSourceException(String s)`

Constrói uma `IncompatibleSourceException` com uma mensagem detalhada.

Parâmetros:

`s` - a mensagem.

21.21 Classe InvalidFilterException

21.21.1 Descrição da classe

`com.sun.dtv.filtering`

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └com.sun.dtv.filtering.DataSectionFilterException
              └com.sun.dtv.filtering.InvalidFilterException
```

`public class InvalidFilterException`

`extends DataSectionFilterException`

Indica que um filtro foi definido inapropriadamente.

Todas as interfaces implementadas

`Serializable`

21.21.2 Índice de construtores

`InvalidFilterException``()`

Constrói uma `InvalidFilterException`.

`InvalidFilterException``(String s)`

Constrói uma `InvalidFilterException` com uma mensagem detalhada.

21.21.3 Detalhe dos construtores

InvalidFilterException

`public InvalidFilterException()`

Constrói uma `InvalidFilterException`.

InvalidFilterException

`public InvalidFilterException(String s)`

Constrói uma `InvalidFilterException` com uma mensagem detalhada.

Parâmetros:

`s` - a mensagem.

21.22 Classe ListFilter

21.22.1 Descrição da classe

`com.sun.dtv.filtering`

```
java.lang.Object
  └com.sun.dtv.filtering.DataSectionFilter
      └com.sun.dtv.filtering.ListFilter
```

```
public class ListFilter
extends DataSectionFilter
```

Essa classe define um filtro de seção que irá filtrar um conjunto inteiro de segmentos de seção de dados que compõem uma única tabela de seções. O campo `last_section_number` da seção de dados deve ser usado para definir quantos objetos `DataSection` requer-se quem sejam alocados para popular toda a tabela de seções.

Um `DataSectionAvailableEvent` deve ser gerado cada vez que um `DataSection` for capturado. Uma `EndOfFilteringEvent` deve ser gerado quando tabela completa houver sido capturada. Um `FilterTimedOutEvent` deve ser gerado se a tabela completa não houver sido filtrada dentro do tempo especificado por `DataSectionFilter.setTimeOut`.

O version_number de todas as seções da tabela deve ser o mesmo. Seções com um número de versão diferente da primeira seção devem ser ignoradas. E um `FilteringStoppedEvent` avulso com `CAUSE_VERSION_CHANGE_DETECTED` deve ser despachado (e nenhum a mais). Se nenhuma seção adicional for restaurada com o primeiro número de versão observado, o `FilterTimedOutEvent` deve ser eventualmente despachado se o filtro não estiver parado.

Relaciona-se com:

`DataSection`, `DataSectionFilter`, `DataSectionFilter.setTimeOut(long)`

Todas as interfaces implementadas

`Cloneable`

21.22.2 Índice de métodos

`DataSection[]` **`getSections`**`()`

Esse método retorna um *array* de objetos `DataSection` correspondentes às seções na tabela.

Métodos herdados da classe `com.sun.dtv.filtering.`**`DataSectionFilter`**

`addSectionFilterListener, clearGreaterThanFilter, clearLessThanFilter, clearPositiveFilter, clearTableId, clearXorFilter, removeSectionFilterListener, setGreaterThanFilter, setLessThanFilter, setPid, setPositiveFilter, setTableId, setTimeOut, setXorFilter, startFiltering, stopFiltering`

21.22.3 Detalhe dos métodos

getSections

```
public DataSection[] getSections()
                            throws FilterInterruptException
```

Esse método retorna um *array* de objetos `DataSection` correspondentes às seções na tabela. As seções no *array* devem ser ordenadas de acordo com o seu section_number. Quaisquer seções da fonte que não houverem sido filtradas terão a entrada correspondente no *array* definida para `null`. Se nenhuma seção houver sido filtrada, esse método bloquear-se-á até que pelo menos uma seção esteja disponível ou a filtragem pare.

Chamadas repetidas para esse método retornarão o mesmo *array*, dado que nenhuma nova chamada para `startFiltering` tenha sido feita no ínterim. Cada vez que uma nova operação de filtragem é iniciada, um novo *array* de objetos `DataSection` deve ser criado. Todas as referências, exceto as do aplicativo, ao *array* e objetos `DataSection` anteriores devem ser removidos. Todos os métodos para acessar dados nos objetos `DataSection` anteriores lançarão um `DataUnavailableException`.

Retorna:

O *array* de objetos `DataSection`

Lança:

`FilterInterruptException` - se a filtragem parar antes de um seção se tornar disponível

21.23 Classe SingleFilter

21.23.1 Descrição da classe

`com.sun.dtv.filtering`

```
java.lang.Object
  └com.sun.dtv.filtering.DataSectionFilter
      └com.sun.dtv.filtering.SingleFilter
```

```
public class SingleFilter
extends DataSectionFilter
```

Essa classe define um filtro de seção com intenção de ser usado para capturar uma seção de dados avulsa. Quando uma seção correspondendo ao padrão de filtro especificado é encontrada, objetos `SingleFilter` pararão. Um `DataSectionAvailableEvent` deve ser gerado quando um `DataSection` for capturado. Os filtros irão, então, parar.

Relaciona-se com:

`DataSection`, `DataSectionFilter`

Todas as interfaces implementadas

`Cloneable`

21.23.2 Índice de métodos

`DataSection` **`getSection`**`()`

Esse método restaura um objeto `DataSection` avulso descrevendo uma seção que tenha correspondido a uma definição de filtro ativo.

Métodos herdados da classe `com.sun.dtv.filtering.`**`DataSectionFilter`**

`addSectionFilterListener, clearGreaterThanFilter, clearLessThanFilter, clearPositiveFilter, clearTableId, clearXorFilter, removeSectionFilterListener, setGreaterThanFilter, setLessThanFilter, setPid, setPositiveFilter, setTableId, setTimeOut, setXorFilter, startFiltering, stopFiltering`

21.23.3 Detalhe dos métodos

getSection

```
public DataSection getSection()
                      throws FilterInterruptException
```

Esse método restaura um objeto `DataSection` avulso descrevendo uma seção que tenha correspondido a uma definição de filtro ativo. Esse método nunca bloquear-se-á e retornará `null` se nenhuma seção tiver correspondido ainda.

Cada vez que uma nova operação de filtragem é iniciada, um novo objeto `DataSection` deve ser criado. Todas as referências, exceto as do aplicativo, aos objetos `DataSection` anteriores devem ser removidas. Todos os método para acessar dados no objeto `DataSection` lançarão um `NoDataAvailableException`.

Retorna:

O objeto `DataSection` avulso correspondente ou `null`.

Lança:

`FilterInterruptException` - se o filtro foi interrompido antes que a quantidade requerida de dados fosse filtrada.

22 Pacote com.sun.dtv.io

22.1 Descrição do pacote

Estende o pacote `java.io` provendo direitos de acesso e propriedades aos arquivos.

Esse pacote provê uma API que permite aplicativos interativos a estocar dados na memória não-volátil da plataforma e controlar acesso a esses dados:

`FileAccessRights`: oferece um meio de controlar acesso a um arquivo em diferentes granularidades. Atualmente três grupos são definidos: Global, grupo e aplicativo (usuário). O modelo é muito similar ao encontrado nas plataformas Unix;

`FileProperties`: oferece um meio de associar propriedades (meta informação) a arquivos específicos. A especificação define atualmente três propriedades obrigatórias que permitem um mecanismo de cache avançado.

A Figura 8 mostra a estrutura do pacote I/O do Java DTV.

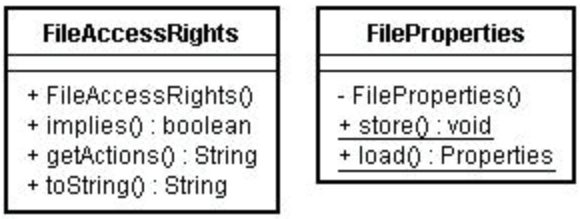


Figura 8 – Pacote I/O

22.2 Índice de classes

FileAccessRights

Provê um meio para definir direitos de níveis de acesso a grupos diferentes a um arquivo ou diretório

FileProperties

Usado para associar propriedades (ou metadados) com um arquivo identificado pelo seu nome do caminho em um dado sistema de arquivos.

NOTA Este pacote está especificado desde o Java DTV 1.0.

22.3 Classe FileAccessRights

22.3.1 Descrição da classe

com.sun.dtv.io

```
java.lang.Object
  └com.sun.dtv.io.FileAccessRights
```

```
final public class FileAccessRights
extends Object
```

Provê um meio para definir direitos de níveis de acesso a grupos diferentes a um arquivo ou diretório Um `FileAccessRights` consiste em um nome do caminho e um conjunto de ações válidas para esse nome do caminho. Nome do caminho é o nome do caminho do arquivo ou diretório concedido pelas ações especificadas.

Esse pacote define classes que estendem o comportamento da implementação do pacote `java.io`. Por exemplo: a implementação do `java.io` em uma dada plataforma que segue esta especificação deve reforçar o modelo de acessos e direitos definidas no `FileAccessRights`.

22.3.2 Nome do Caminho

Um nome do caminho que termina em "`/*`" (onde "`/`" é o caractere separador de arquivos, `File.separatorChar`) indica um diretório e todos os arquivos contidos nesse diretório. Um nome do caminho que termine em "`/-`" indica um diretório e (recursivamente) todos os arquivos e subdiretórios contidos nesse diretório. O nome do caminho especial "`<<ALL FILES>>`" corresponde a todos os arquivos.

Um nome do caminho consistindo de um único "`*`" indica todos os arquivos no diretório corrente, enquanto um nome do caminho consistindo de um único "`-`" indica todos os arquivos no diretório atual e (recursivamente) todos os arquivos e subdiretórios contidos no diretório atual. Diretório atual está seguindo a mesma definição que "`.`"

22.3.3 Ações

As ações a serem concedidas são passadas ao construtor em uma *string* contendo uma lista de zero ou mais conjuntos de *flags* separadas por ponto e vírgula. Três grupos são definidos e mapeados para os grupos de permissão de arquivos UNIX equivalentes: *user*, *group* e *world*. A interpretação desses três grupos é, todavia, deixada para a implementação: outras plataformas como conversores (set top box) podem mapear os grupos para *application, organization* e *world*, respectivamente.

A *string* de ação é descrito pela seguinte gramática (essa definição também dá a representação canônica):

```
action   ::= perm_set [';' perm_set [';' perm_set ] ]
perm_set ::= read write execute
read     ::= 'r' | '-' | nil
write    ::= 'w' | '-' | nil
execute  ::= 'x' | '-' | nil
```

A ordem dos `perm_sets` em uma *string* `action` é *user*, *group* e *world* respectivamente.

Os símbolos de direitos de acesso são definidos como se segue:

`r`

o *flag* para ler. Permite explicitamente que o dado grupo leia o arquivo associado.

`w`

o *flag* para escrever. Permite explicitamente que o dado grupo escreva no arquivo associado.

`x`

o *flag* para executar. Permite explicitamente que o dado grupo execute o arquivo associado.

`-`

é usado para expressar acesso negado para um dado direito de acesso. É equivalente a `nil`.

Exemplos de ações válidas

`"rwx;rw-;r"`

Torna um arquivo disponível para leitura para todos os grupos, para escrita para usuário e grupo e executável para usuário apenas.

`"rw;rw;---"`

Torna um arquivo disponível para escrita para usuário e grupo.

`"rw;rw"`

É equivalente a definição anterior.

`"r;;"`

Torna um arquivo disponível para leitura para usuário e negue qualquer outra ação para quaisquer outros grupos.

`"r"`

É equivalente a definição anterior.

`";;rx"`

È válido e sua representação canônica equivalente é "`rx;rx;rx`" permitindo todos a ler e executar esse arquivo.

`""`

Torna um arquivo inacessível e inútil.

22.3.4 Representação canônica e verificação de consistência

A representação canônica dos direitos de acesso a arquivos é definida pela representação *string* mínima: para o direito de acesso que segue a gramática definida na seção actions, que é completamente coerente com a intenção.

Como uma conseqüência e devido à definição dada na seção 22.3.3 que uma `action` representa o direito de acesso dado a três grupos que estão incluídos uns nos outros; qualquer direito de acesso concedido a um grupo maior implica que ele também foi concedido a um grupo menor: `user` ⊃ `group` ⊃ `world`.

A Tabela 51 lista diferentes exemplos e sua representação canônica padronizada:

Tabela 51 – Exemplos de direitos de acesso a arquivos

Direitos de Acesso	Canônico	Descrição
"r;w;x"	"rwx;wx;x"	Devido a herança de grupos incluídos, todo grupo menor herda os direitos do grupo maior
";;rx"	"rx;rx;rx"	Direitos de acesso para `world` também se aplicam a cadeias mais baixas (menores) de grupos
`";r;"`	`"r;r"`	Direitos de acesso para `group` também se aplicam a `user`. Além disso, o último ponto e vírgula é removido porque não há direitos de acesso para `world`
`";;"`	""	Pontos e vírgulas são removidos, porque não há direitos de acesso para nenhum grupo

Relaciona-se com:

`FileProperties`

22.3.5 Índice de construtores

`FileAccessRights``(String path, String actions)`

Cria um novo objeto `FileAccessRights` com as ações especificadas.

22.3.6 Índice de métodos

`String` **`getActions`**`()`

Retorna a "representação *string* canônica" de outras ações.

`boolean` **`implies`**`(FileAccessRights other)`

Checa se esse objeto `FileAccessRights` “implica” os direitos de acesso especificados.

`String` **`toString`**`()`

Retorna uma *string* descrevendo os direitos de acesso.

22.3.7 Detalhe dos construtores

FileAccessRights

```
public FileAccessRights(String path,
                        String actions)
                 throws NullPointerException
```

Cria um novo objeto `FileAccessRights` com as ações especificadas.

O `path` e `actions` seguem as definições descritas na seção visão geral

Parâmetros:

`path` - O nome do caminho do arquivo/diretório.

`actions` - A *string* de ação.

Lança:

`NullPointerException` - se qualquer parte do argumento for `null`.

22.3.8 Detalhe dos métodos

implies

```
public boolean implies(FileAccessRights other)
```

Checa se esse objeto `FileAccessRights` "implica" os direitos de acesso especificados.

Mais especificamente, esse método retorna `true` se:

outro é uma instância de `FileAccessRights`,

As ações de `other` são um subconjunto próprio das ações do objeto, e

O nome do caminho de `other` é implicado pelo nome do caminho desse objeto. Por exemplo: "`/tmp/*`" implica "`/tmp/foo`", já que "`/tmp/*`" engloba o diretório "`/tmp`" e todos os arquivos nesse diretório, incluindo o arquivo chamado "`foo`".

Parâmetros:

`other` - Os direitos de acesso a confrontar.

Retorna:

`true` se os direitos de acesso especificados estiverem implicados nesse objeto; *`false`* se não.

getActions

```
public String getActions()
```

Retorna a "representação *string* canônica" de outras ações. Ou seja, esse método sempre retorna ações presentes na ordem e formato descritos na gramática encontrada na seção 22.3.3 acima. A representação deveria também ser coerente e minimalista como descrito na seção 22.3.4 acima.

Retorna:

O a representação canônica de *string* das ações.

toString

```
public String toString()
```

Retorna um *string* descrevendo os direitos de acesso. A convenção é especificar o nome da classe, o caminho e as ações (como retornadas por `getActions()`) no seguinte formato: `'("ClassName" "name" "actions")'`.

Substituições:

`toString` na classe `Object`

Retorna:

Informações sobre esse objeto `FileAccessRights`.

22.4 Classe FileProperties

22.4.1 Descrição da classe

com.sun.dtv.io

```
java.lang.Object
  └com.sun.dtv.io.FileProperties
```

```
final public class FileProperties
extends Object
```

Usado para associar propriedades (ou metadados) com um arquivo identificado pelo seu nome do caminho em um dado sistema de arquivos. Como definido em `Properties`, propriedades são um conjunto de (chave, valor) pares de `String`.

Esse pacote define classes que estendem o comportamento da implementação do pacote `java.io`. Por exemplo, a implementação do `java.io` em uma dada plataforma que segue esta especificação deve reforçar o modelo de acessos e gerenciamento de arquivos definidos em `FileProperties` e `FileAccessRights`.

Nome do Caminho

Um nome do caminho que termina em "`/*`" (onde "`/`" é o caractere separador de arquivos, `File.separatorChar`) indica um diretório e todos os arquivos contidos nesse diretório. Um nome do caminho que termine em "`/-`" indica um diretório e (recursivamente) todos os arquivos e subdiretórios contidos nesse diretório. O nome do caminho especial "`<<ALL FILES>>`" corresponde a todos os arquivos.

Um nome do caminho consistindo de um único "`*`" indica todos os arquivos no diretório corrente, enquanto um nome do caminho consistindo de um único "`-`" indica todos os arquivos no diretório atual e (recursivamente) todos os arquivos e subdiretórios contidos no diretório atual.

Propriedades

As seguintes propriedades devem estar disponíveis para todos os arquivos na plataforma que estão acessíveis a aplicativos e ser suportadas pela implementação:

`com.sun.dtv.access-rights`

Particularmente na plataforma que não suporta ou que tem suporte básico de direitos de acesso, essa propriedade permite associar um conjunto de informações avançadas relacionadas a direito de acesso com um arquivo. valor é um String representando os direitos de acesso cujo formato deve corresponder a saída de `FileAccessRights.getActions()`
padrão: "`rw`"
mínimo denominador comum: o mínimo de cada direito dentro de um conjunto de direitos de acesso. Decisões detalhadas são encontradas em `FileAccessRights`.

`com.sun.dtv.expiration-date`

Em plataformas com estocagem limitada de recursos ou onde áreas específicas de armazenagem podem atuar

como cache, essa propriedade permite definir uma data após a qual o arquivo associado é considerado como expirado e não mais disponível. O valor é uma *string* representando um inteiro positivo correspondente ao número de milissegundos desde a base padrão conhecida como "*the epoch*", especificamente `January 1, 1970, 00:00:00 GMT`. Esse número deve ser lançado pela implementação como um `long`. Esse valor corresponde à saída de `Date.getTime()` e à entrada de `Date.Date(long)`. O valor pode também ser a *string* vazio (`""`) para expressar "nunca" (isto é, uma data indefinida no futuro).
default: `""`
mínimo denominador comum: A última das datas dentro de um conjunto de datas de expiração.
Regra de gerenciamento: um arquivo que houver expirado não pode ser acessível e deveria ser deletado assim que houver a primeira tentativa de acessar o arquivo após a expiração.

`com.sun.dtv.persistence-level`

Em plataformas com recursos de armazenamento limitados ou onde áreas de armazenamento específicas podem atuar como *cache*, essa propriedade permite definir um nível de retenção em relação à escassez de armazenamento. O valor é uma *string* representando um valor inteiro positivo entre 0 e `Integer.MAX_VALUE` onde quanto mais alto o valor, mais provavelmente o arquivo associado deve ser retido isto é, arquivos em níveis mais baixos são removidos primeiros. Todavia, resta à plataforma definir a política de retenção atual baseada no valor definido por essa propriedade.

padrão `"0"`
mínimo denominador comum: o máximo de todos os níveis de persistência.
Regra de gerenciamento: em caso de escassez de armazenagem, a implementação deve remover o arquivo com menor valor de nível de persistência. As plataformas podem aplicar algoritmos de decisão de granularidade fina em caso de múltiplos arquivos se encontrarem no menor nível: por exemplo, uma seleção baseada no tamanho do arquivo ou baseada na freqüência com que um dado aplicativo é usado ou baseado nas datas de expirações previstas para o futuro próximo. Resta à implementação definir razoavelmente quando houver escassez de armazenagem.

Essa árvore de propriedades pode ser estendida a aplicativos e bibliotecas de terceiros. O limite de número de propriedades é específico da plataforma. Novas propriedades devem seguir a mesma convenção para seus nomes e para criação de nome de um novo pacote Java, que é: considerando o nomde de domínio de internet da organização, como `example.com`. Então reverter este nome, componente por componente, anexando o prefixo da árvore ou do nome do pacote: `com.example.voting`. Convenções usadas além do nome de domínio revertido são deixadas a critério de qualquer companhia.

O nome de domínio `com.sun.dtv` é reservado para propriedades definidas nesta especificação.

Observar que propriedades definidas por aplicativos e bibliotecas de terceiros não podem correntemente definir um *mínimo denominador comum* que pudesse ser usado pelo método `load()`. Portanto, ao aplicar `load()` em um conjunto, pode acontecer de propriedades cujos valores diferem entre dois ou mais arquivos/diretórios dentro daquele conjunto, não serem retornadas. Consulte a descrição do método `load`.

Singularidade

Todos as coleções de propriedades associadas a um dado arquivo devem garantir a singularidade de todas as chaves. Isso é feito pela classe `Properties` na qual a classe `FileProperties` é baseada.

Do ponto de vista da implementação, todos os arquivos identificados por um caminho devem ter no máximo uma coleção de propriedades associada.

Do ponto de vista do aplicativo, todos os arquivos identificados por um caminho devem ter uma, e apenas uma, coleção de propriedades associada. Isso é feito ao deixar a implementação provendo uma coleção de propriedades padrão quando é requisitada a restaurar as propriedades de um arquivo para o qual não foram armazenadas propriedades (ver `load(String)`).

Regra de gerenciamento

Regra de gerenciamento como descrita para algumas propriedades, define como a dada propriedade influência a implementação em relação à escassez de estocagem e alguns estoques precisarem ser liberados. Essas regras devem ser respeitadas como descritas e combinadas como solicitado. Consultar também Persistent Storage para

mais informações.

Relaciona-se com:

`Properties`, `FileAccessRights`

22.4.2 Índice de métodos

`static Properties` **`load`**`(String path)`

Carrega uma cópia da coleção de propriedades associadas a um (conjunto de) arquivo/diretório identificado pelo `path`.

`static void` **`store`**`(String path, Properties properties)`

Associa a coleção de propriedades passada em `properties` ao arquivo identificado pelo `path` e estoca-o persistentemente.

22.4.3 Detalhe dos métodos

store

```
public static void store(String path,
                         Properties properties)
                  throws NullPointerException,
                         IOException,
                         ClassCastException,
                         FileNotFoundException
```

Associa a coleção de propriedades passada em `properties` ao arquivo identificado pelo `path` e estoca-o persistentemente.

O argumento `path` é o nome do caminho de um (conjunto de) arquivo/diretório ao qual as `properties` são associadas. Qualquer associação de propriedades subseqüente deve sobrescrever o conjunto de propriedades prévio de forma que todos os `path` estejam associados unicamente a um conjunto de propriedades.

Assim que esse método retornar, a plataforma deve ter armazenado as propriedades persistentemente. A implementação deveria usar o método `Properties.store(OutputStream, String)` correspondente para processar essa requisição.

Se o `path` não existe, a associação não pode ser concluída e a implementação deve indicar isso lançando um `FileNotFoundException`.

Se a coleção de propriedades estiver incompleta no que diz respeito às propriedades obrigatórias descritas acima, resta à implementação gerá-las na estocagem ou no momento de restauração.

Parâmetros:

`path` – o nome do caminho do arquivo/diretório.

`properties` - as propriedades a serem associadas aquele arquivo/diretório

Lança:

`NullPointerException` - se qualquer parte do argumento passado for `null`.

`IOException` - se escrever essa lista de propriedades para o *stream* de saída especificado lançar uma `IOException`.

`ClassCastException` - se quaisquer propriedades armazenadas tiverem uma chave ou um valor que não é uma *string*.

`FileNotFoundException` - se o arquivo/diretório referenciado pelo caminho não existir.

load

```
public static Properties load(String path)
                        throws NullPointerException,
                               FileNotFoundException,
                               IllegalArgumentException
```

Carrega uma cópia da coleção de propriedades associadas a um (conjunto de) arquivo/diretório identificado pelo `path`.

O argumento do caminho segue a mesma definição de `store(String, Properties)`.

Quando esse método é aplicado a um conjunto de arquivos/diretórios, ele deve retornar o mínimo denominador comum desse conjunto de propriedades. Isso só se aplica a propriedades para as quais um mínimo denominador comum é definido. Para qualquer propriedade para a qual um mínimo denominador comum não foi definido, esse método deve incluir na coleção retornada apenas aqueles cujos valores correspondem a cada arquivo/diretório dentro desse conjunto.

Se um arquivo correspondendo a um dado `path` não tiver nenhuma coleção de prioridades associadas ainda, a implementação deve prover transparentemente um novo objeto `Properties` onde os valores das propriedades são definidos como os valores padrão definidos na descrição dessa classe.

Parâmetros:

`path` – o nome do caminho do arquivo/diretório.

Retorna:

Uma cópia das propriedades para o dado `path`.

Lança:

`NullPointerException` - se o caminho for `null`.

`FileNotFoundException` - se o caminho não indicar nenhum arquivo existente.

`IllegalArgumentException` - se o caminho for um caminho inválido e não estiver conforme o formato dado na descrição da classe.

23 Pacote com.sun.dtv.locator

23.1 Descrição do pacote

Esse pacote define localizadores para uso através de todo o sistema.

`URLLocator` é um localizador criado a partir de URLs.

`EntityLocator` é um localizador usado para representar entidades em *streams* de transporte. Essas são o próprio *stream* de transporte assim como serviços, *streams* elementares e eventos.

`TransportDependentLocator` é uma interface tag para identificar localizadores que são dependentes de transporte. Todas as implementações de localizadores que são dependentes de transporte devem implementar essa interface.

`NetworkBoundLocator` é um localizador usado para representar entidades ligadas por rede.

A Figura 9 mostra a estrutura do pacote `Locator` do Java DTV.

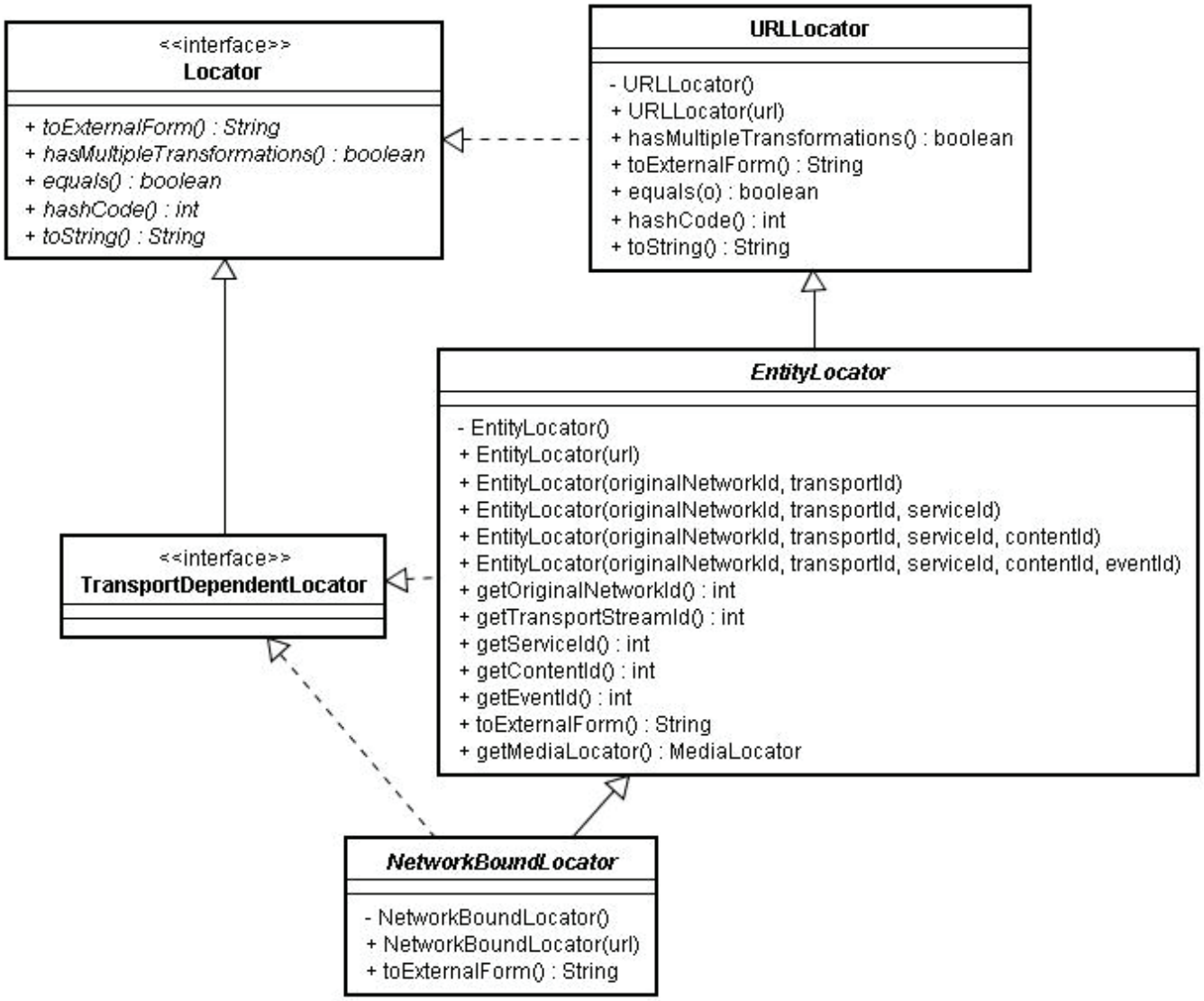


Figura 9 – Pacote Locator

NOTA Este pacote está especificado desde o Java DTV 1.0.

23.2 Índice de interfaces

TransportDependentLocator

Localizador referenciando entidades em um *stream* de transporte.

23.3 Índice de classes

EntityLocator

Localizador para entidades no *stream* de transporte.

NetworkBoundLocator

Localizador referenciando entidades que estão ligadas por rede.

URLLocator

Localizador baseado em uma URL.

23.4 Classe EntityLocator

23.4.1 Descrição da classe

com.sun.dtv.locator

```
java.lang.Object
  └com.sun.dtv.locator.URLLocator
      └com.sun.dtv.locator.EntityLocator
```

```
abstract public class EntityLocator
extends URLLocator
implements TransportDependentLocator
```

Localizador para entidades no *stream* de transporte. Essas entidades podem ser *streams* de transporte, serviços, *streams* elementares e eventos.

Todas as interfaces implementadas

`Locator, TransportDependentLocator`

Subclasses diretas conhecidas

`NetworkBoundLocator`

23.4.2 Índice de construtores

`EntityLocator``(int originalNetworkId, int transportId)`

Crie um localizador para um *stream* de transporte.

`EntityLocator``(int originalNetworkId, int transportId, int serviceId)`

Crie um localizador para um serviço.

`EntityLocator``(int originalNetworkId, int transportId, int serviceId, int contentId)`

Crie um localizador para um *stream* elementar.

`EntityLocator``(int originalNetworkId, int transportId, int serviceId, int contentId, int eventId)`

Crie um localizador para um evento.

`EntityLocator``(String url)`

Crie um localizador a partir de uma url específica.

23.4.3 Índice de métodos

`int` **`getContentId`**`()`

Obtém o Id de conteúdo desse localizador.

`int` **`getEventId`**`()`

Obtém o Id de evento desse localizador.

`MediaLocator` **`getMediaLocator`**`()`

Provê a representação `MediaLocator` se essa instância for um localizador para um `TransportStream` ou um `ElementaryStream` em um formato que é suportado pela implementação JMF.

`int` **`getOriginalNetworkId`**`()`

Obtém o Id de rede original desse localizador.

`int` **`getServiceId`**`()`

Obtém o Id de serviço desse localizador.

`int` **`getTransportStreamId`**`()`

Obtém o Id de *stream* de transporte desse localizador.

`String` **`toExternalForm`**`()`

Gera uma representação canônica baseada em *string* desse `Locator`.

Métodos herdados da classe `com.sun.dtv.locator.`**`URLLocator`**

`equals, hashCode, hasMultipleTransformations, toString`

23.4.4 Detalhe dos construtores

EntityLocator

```
public EntityLocator(String url)
              throws InvalidLocatorException
```

Crie um localizador a partir de uma url específica.

Parâmetros:

`url` - a *string* `url`

Lança:

`InvalidLocatorException` - se o localizador não pode ser criado a partir dos parâmetros providos.

EntityLocator

```
public EntityLocator(int originalNetworkId,
                     int transportId)
              throws InvalidLocatorException
```

Crie um localizador para um *stream* de transporte.

Parâmetros:

`originalNetworkId` – Id de rede original; “-1” deve ser usado para especificar o Id de rede padrão.

`transportId` – Id do *stream* de transporte.

Lança:

`InvalidLocatorException` - se o localizador não pode ser criado a partir dos parâmetros providos.

EntityLocator

```
public EntityLocator(int originalNetworkId,
```

```
                    int transportId,
                    int serviceId)
             throws InvalidLocatorException
```

Crie um localizador para um serviço.

Parâmetros:

`originalNetworkId` – Id de rede original; “-1” deve ser usado para especificar o Id de rede padrão.

`transportId` – Id do *stream* de transporte.

`serviceId` - Id do serviço

Lança:

`InvalidLocatorException` - se o localizador não pode ser criado a partir dos parâmetros providos.

EntityLocator

```
public EntityLocator(int originalNetworkId,
                    int transportId,
                    int serviceId,
                    int contentId)
             throws InvalidLocatorException
```

Crie um localizador para um *stream* elementar.

Parâmetros:

`originalNetworkId` – identificador de rede original; “-1” deve ser usado para especificar o identificador de rede padrão.

`transportId` – identifricador do *stream* de transporte.

`serviceId` - identificador do serviço

`contentId` - identificador do conteúdo do *stream* elementar.

Lança:

`InvalidLocatorException` - se o localizador não pode ser criado a partir dos parâmetros providos.

EntityLocator

```
public EntityLocator(int originalNetworkId,
                    int transportId,
                    int serviceId,
                    int contentId,
                    int eventId)
             throws InvalidLocatorException
```

Crie um localizador para um evento.

Parâmetros:

`originalNetworkId` – identificador de rede original; “-1” deve ser usado para especificar o identificador de rede padrão.

`transportId` - identificador do *stream* de transporte.

`serviceId` - o identificador do serviço.

`contentId` – identificador do conteúdo do *stream* elementar.

`eventId` - identificador do evento.

Lança:

`InvalidLocatorException` - se o localizador não pode ser criado a partir dos parâmetros providos.

23.4.5 Detalhe dos métodos

getOriginalNetworkId

```
public int getOriginalNetworkId()
```

Obtém o identificador de rede original desse localizador.

Retorna:

O identificador original de rede “-1”, caso não esteja disponível.

getTransportStreamId

```
public int getTransportStreamId()
```

Obtém o identificador de *stream* de transporte desse localizador.

Retorna:

O identificador original de *stream* de transporte “-1”, caso não esteja disponível.

getServiceId

```
public int getServiceId()
```

Obtém o identificador de serviço desse localizador.

Retorna:

O identificador de serviço “-1”, caso não esteja disponível.

getContentId

```
public int getContentId()
```

Obtém o identificador de conteúdo desse localizador.

Retorna:

O identificador de conteúdo “-1”, caso não esteja disponível.

getEventId

```
public int getEventId()
```

Obtém o identificador de evento desse localizador.

Retorna:

O identificador de evento “-1”, caso não esteja disponível.

toExternalForm

```
public String toExternalForm()
```

Gera uma representação canônica baseada em *string* desse `Locator`. O *string* retornado pode ser inteiramente dependente de plataforma. Se dois localizadores têm formas externas idênticas, eles consultam o mesmo recurso. Contudo, dois localizadores que consultam o mesmo recurso podem ter diferentes formas externas.

Esse método retorna a forma canônica do *string* que foi usado para criar o localizador (por meio de `LocatorFactory.createLocator()`). Para gerar formas externas canônicas, a implementação irá dar o máximo de si para dividir localizadores em relações um-a-um com os recursos que eles consultam.

O resultado desse método pode ser usado para criar novas instâncias `Locator`, assim como outros tipos de localizadores, tais como como `MediaLocator`s JMF e `URL`s.

Especificado por:

`toExternalForm` na interface `Locator`

Substituições:

`toExternalForm` na classe `URLLocator`

Retorna:

Uma representação baseada em *string* desse Localizador.

getMediaLocator

`public MediaLocator getMediaLocator()`

Provê a representação `MediaLocator` se essa instância for um localizador para um `TransportStream` ou um `ElementaryStream` em um formato que é suportado pela implementação JMF.

Retorna:

o `MediaLocator` se esse `EntityLocator` é um localizador para mídias que podem ser exibidas por uma instância de tocador JMF; caso contrário `null`.

23.5 Classe NetworkBoundLocator

23.5.1 Descrição da classe

`com.sun.dtv.locator`

```
java.lang.Object
  └com.sun.dtv.locator.URLLocator
      └com.sun.dtv.locator.EntityLocator
          └com.sun.dtv.locator.NetworkBoundLocator
```

`abstract public class NetworkBoundLocator`

`extends EntityLocator`

`implements TransportDependentLocator`

Localizador referenciando entidades que estão ligadas por rede.

Todas as interfaces implementadas

`Locator, TransportDependentLocator`

23.5.2 Índice de construtores

`NetworkBoundLocator``(String url)`

Cria um localizador a partir de uma `url` específica.

23.5.3 Índice de métodos

`String` **`toExternalForm`**`()`

Gera uma representação canônica baseada em *string* desse `Locator`.

Métodos herdados da classe `com.sun.dtv.locator.`**`EntityLocator`**

`getContentId, getEventId, getMediaLocator, getOriginalNetworkId, getServiceId, getTransportStreamId`

Métodos herdados da classe `com.sun.dtv.locator.`**`URLLocator`**

`equals, hashCode, hasMultipleTransformations, toString`

23.5.4 Detalhe dos construtores

NetworkBoundLocator

```
public NetworkBoundLocator(String url)
                    throws InvalidLocatorException
```

Crie um localizador a partir de uma url específica.

Parâmetros:

`url` - a *string* `url`

Lança:

`InvalidLocatorException` - se o localizador não pode ser criado a partir dos parâmetros providos.

23.5.5 Detalhe dos métodos

toExternalForm

```
public String toExternalForm()
```

Gera uma representação canônica baseada em *string* desse `Locator`. O *string* retornado pode ser inteiramente dependente de plataforma. Se dois localizadores têm formas externas idênticas, eles consultam o mesmo recurso. Contudo, dois localizadores que consultam o mesmo recurso podem ter diferentes formas externas.

Esse método retorna a forma canônica do *string* que foi usado para criar o localizador (por meio de `LocatorFactory.createLocator()`). Para gerar formas externas canônicas, a implementação irá dar o máximo de si para dividir localizadores em relações um-a-um com os recursos que eles consultam.

O resultado desse método pode ser usado para criar novas instâncias `Locator`, assim como outros tipos de localizadores, tais como como `MediaLocators` JMF e `URLs`.

Especificado por:

`toExternalForm` na interface `Locator`

Substituições:

`toExternalForm` na classe `EntityLocator`

Retorna:

Uma representação baseada em *string* desse localizador.

23.6 Interface TransportDependentLocator

23.7 Descrição da interface

com.sun.dtv.locator

```
public interface TransportDependentLocator
extends Locator
```

Localizador referenciando entidades em um *stream* de transporte. O localizador pode consultar o próprio *stream* de transporte, assim como entidades contidas nele (tais como serviço, *stream* elementar, evento e componente).

Todas as super-interfaces

```
Locator
```

Todas as classes implementadoras conhecidas

```
EntityLocator, NetworkBoundLocator
```

23.8 Classe URLLocator

23.8.1 Descrição da classe

com.sun.dtv.locator

```
java.lang.Object
  └com.sun.dtv.locator.URLLocator
```

```
public class URLLocator
extends Object
implements Locator
```

Localizador baseado em uma URL. Esse localizador pode ser usado como dependente de transporte assim como localizadores independentes de transporte.

As seguintes formas de localizadores DEVEM ser usadas para criar um localizador:

para *streams* de transporte:
`dtv://<original_network_id>.<transport_stream_id> or`
`dtv://<frequency>:<modulation>.<symbol_rate>`

para serviços:
`dtv://<original_network_id>.<transport_stream_id>.<service_id>`

para domínios de serviços:
`dtv://<original_network_id>.<transport_stream_id>.<service_id>[;<content_id>][.<event_id>]/<component_tag>`

para eventos de programas:
`dtv://<original_network_id>.<transport_stream_id>.<service_id>.<event_id>`

para *streams* elementares MPEG:
`dtv://<original_network_id>.<transport_stream_id>.<service_id>[;<content_id>][.<event_id>]/<component_tag>[;<channel_id>]{&<component_tag>[;<channel_id>]}`

para arquivos e diretórios:
URL de "`arquivo:`", "`http:`" e "`https:`".

Deveria ser observado que todos os arquivos de transmissão podem ser chamados de "`file:`" usando o local no sistema de arquivos local.

Todos os identificadores nas URLs DEVEM ser providas em forma hexadecimal, usando apenas dígitos hexadecimais ('0'-'9','a'-'f','A'-'F'). “0” a esquerda é permitido para preencher *string* de comprimento fixo.

Valores de freqüência DEVEM ser dados como números decimais em kiloHertz (kHz).

A modulação DEVE ser “`DQPSK`”, “`QPSK`”, “`16QAM`” or “`64QAM`”. Caixa baixa é permitida para todas as letras.

`Symbol_rate` DEVE ser “`12`” para 1/2, “`23`” para 2/3, “`34`” para 3/4, “`56`” para 5/6 ou “`78`” para 7/8.

Todas as interfaces implementadas

`Locator`

Subclasses diretas conhecidas

`EntityLocator`

23.8.2 Índice de construtores

`URLLocator``(String url)`

Construtor para um localizador baseado em URL.

23.8.3 Índice de métodos

`boolean` **`equals`**`(Object o)`

Compara esse `Locator` ao objeto especificado em busca de igualdade.

`int` **`hashCode`**`()`

Gera um valor de código *hash* para esse `Locator`.

`boolean` **`hasMultipleTransformations`**`()`

Indica se esse localizador tem mapeamento para transportes múltiplos.

`String` **`toExternalForm`**`()`

Gera uma representação canônica baseada em *string* desse `Locator`.

`String` **`toString`**`()`

Retorna a *string* usada para criar esse localizador.

23.8.4 Detalhe dos construtores

URLLocator

```
public URLLocator(String url)
           throws InvalidLocatorException
```

Construtor para um localizador baseado em URL.

Parâmetros:

`url` - a *string* URL

Lança:

`InvalidLocatorException` - se o localizador não pode ser criado a partir dos parâmetros providos.

23.8.5 Detalhe dos métodos

hasMultipleTransformations

```
public boolean hasMultipleTransformations()
```

Indica se esse localizador tem mapeamento para transportes múltiplos.

Especificado por:

`hasMultipleTransformations` na interface `Locator`

Retorna:

true se existirem transformações múltiplas para esse Localizador; *false* caso contrário.

Relaciona-se com:

`Locator.hasMultipleTransformations()`

toExternalForm

```
public String toExternalForm()
```

Gera uma representação canônica baseada em *string* desse `Locator`. O *string* retornado pode ser inteiramente dependente de plataforma. Se dois localizadores têm formas externas idênticas, eles consultam o mesmo recurso. Contudo, dois localizadores que consultam o mesmo recurso podem ter diferentes formas externas.

Esse método retorna a forma canônica do *string* que foi usado para criar o localizador (por meio de `LocatorFactory.createLocator()`). Para gerar formas externas canônicas, a implementação irá dar o máximo de si para dividir localizadores em relações um-a-um com os recursos que eles consultam.

O resultado desse método pode ser usado para criar novas instâncias `Locator`, assim como outros tipos de localizadores, tais como como `MediaLocator`s JMF e `URL`s.

Especificado por:

`toExternalForm` na interface `Locator`

Retorna:

Uma representação baseada em *string* desse Localizador.

Relaciona-se com:

`Locator.toExternalForm()`

equals

```
public boolean equals(Object o)
```

Compara esse `Locator` ao objeto especificado em busca de igualdade. O resultado é `true` se, e apenas se, o objeto especificado também for um `Locator` e tiver uma forma externa idêntica à desse `Locator`.

Especificado por:

`equals` na interface `Locator`

Substituições:

`equals` na classe `Object`

Parâmetros:

`o` - O objeto contra o qual confrontar esse `Locator`.

Retorna:

`true` se o objeto especificado for igual a esse `Locator`.

Relaciona-se com:

`String.equals(Object)`

hashCode

`public int hashCode()`

Gera um valor de código *hash* para esse `Locator`. Duas instâncias `Locator` para as quais `Locator.equals()` é `true` terão valores de código *hash* idênticos.

Especificado por:

`hashCode` na interface `Locator`

Substituições:

`hashCode` na classe `Object`

Retorna:

O valor de código *hash* para esse `Locator`.

Relaciona-se com:

`equals(Object)`

toString

`public String toString()`

Retorna a *string* usado para criar esse localizador.

Especificado por:

`toString` na interface `Locator`

Substituições:

`toString` na classe `Object`

Retorna:

A *string* usado para criar esse localizador.

24 Pacote com.sun.dtv.lwuit

24.1 Descrição do pacote

Pacote principal de *widget*s contendo a composição componente/container similares tanto em terminologia quanto em projeto ao Swing/AWT. Ao contrário do Swing/AWT um sistema completo de janelas não é aplicável nesse caso e as formas são colocadas usando um `DTVContainer` passado em abstrações `Plane` e `Screen`.

Os componentes são colocados em um container com gerenciadores de *layout* que são usados parar determinar o posicionamento de componente `Layout`, containers podem ser aninhados profundamente de maneira similar a Swing/AWT. Todos os componentes são simples e desenhados pelo `UIManager` o que permite tematizar tudo usando estilos. Também permite personalização de UI elaborada ao se obter o `LookAndFeel` e sobrepondo-se métodos específicos para desenhar/dimensionar componentes.

A Figura 10 mostra a estrutura do pacote LWUIT do Java DTV.

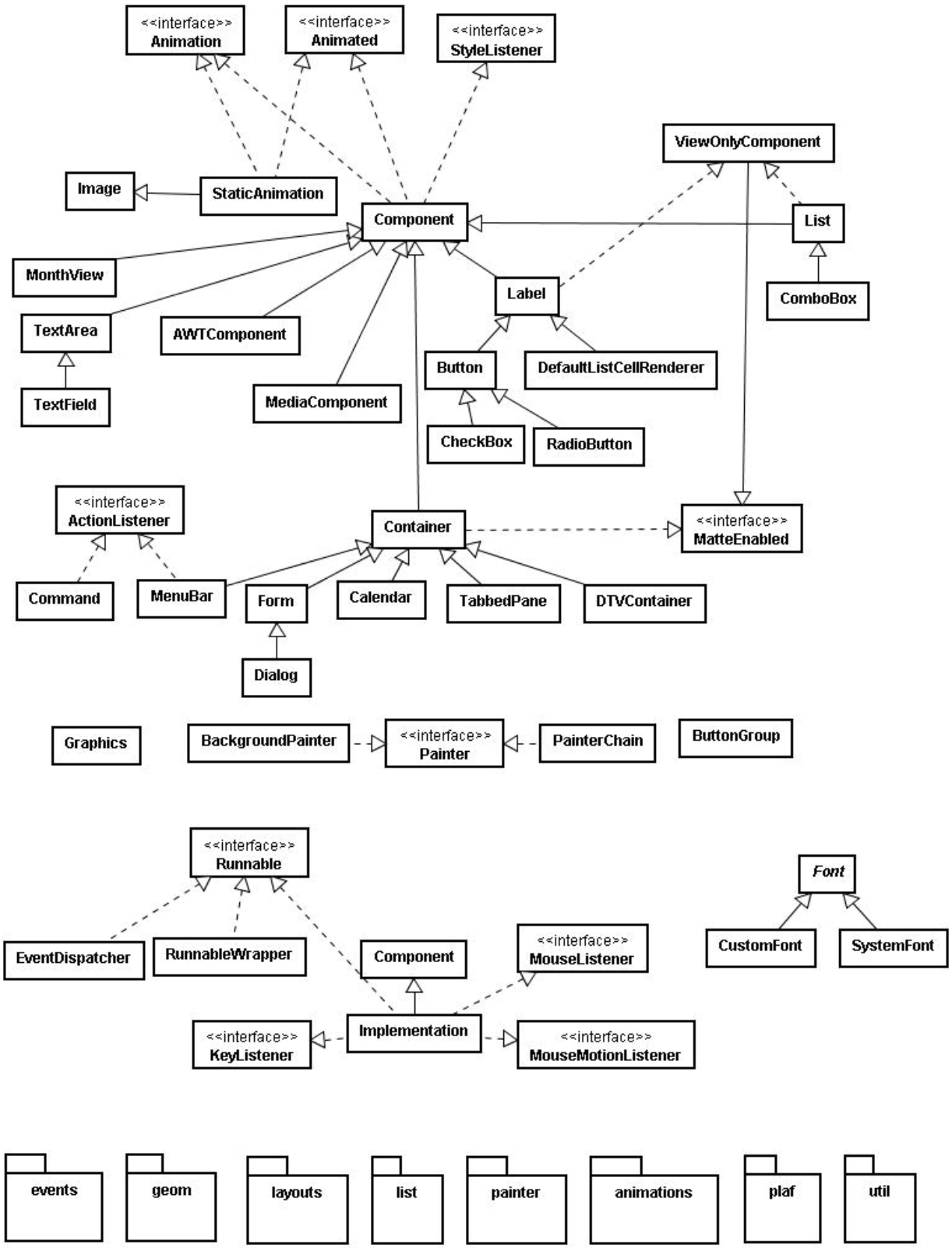


Figura 10 – Pacote LWUIT

NOTA Este pacote está especificado desde o Java DTV 1.0.

24.2 Índice de interfaces

Painter

O `Painter` pode ser usado para desenhar em componentes plano de fundo.

24.3 Índice de classes

AWTComponent

Essa classe estende a classe `Component` como uma variável especial, que delega a função de pintar a um `java.awt.Component` encapsulado.

Button

Botão é a classe base para vários UI *widget*s permitindo clicabilidade.

ButtonGroup

Essa classe é usada para um escopo de exclusão-múltipla para um conjunto de `RadioButtons`.

Calendar

Date o *widget* para selecionar um valor data/hora.

CheckBox

Checkbox é um botão que pode ser selecionado e desselecionado e que exibe seu estado ao usuário.

ComboBox

Um combo Box é uma lista que permite apenas uma seleção por vez, quando um usuário clica no combo box, um botão popup com a lista completa de elementos permite a seleção de um único elemento.

Command

A ação colocada nos botões soft e no menu em dispositivos, similares ao comando de abstração do MIDP e às Ações do Swing.

Component

Classe base para todos os *widget*s no kit de ferramentas usando o padrão composto em uma maneira similar a relação `Container AWT/Component`.

Container

Um padrão composto com `Component`, permite aninhamento e a organização de componentes múltiplos usando uma arquitetura de gerenciador de layout plugável.

Dialog

Um diálogo é um formulário que ocupa parte da tela e aparece como uma entidade modal ao desenvolvedor.

Font

Uma simples abstração de fontes de plataforma e fontes de bibliotecas que possibilita a biblioteca usar fontes mais elaboradas não-suportadas por um dispositivo específico.

Form

componente de nível mais alto que serve como a entidade mais visível no UI (diretamente acoplado no `DTVContainer`, esse `Container` manuseia os menus e título enquanto coloca conteúdo entre eles.

Graphics

Abstrai o contexto gráfico da plataforma subjacente, permitindo, então, que alcancemos portabilidade entre dispositivos MIDP e dispositivos CDC.

Image

Abstrai as imagens da plataforma subjacente, permitindo que as tratemos como um objeto uniforme.

Label `()`

Permite a exibição de etiquetas e imagens com diferentes opções de alinhamento, essa classe é uma classe base para vários componentes, permitindo-lhes a declarar o visual do ícone/alinhamento de uma maneira similar.

List

Um conjunto de elementos que são processados usando um `ListCellRenderer` e são extraídos por meio do `ListModel`.

MediaComponent

Um componente permitindo acoplar e controlar conteúdo *rich media*.

RadioButton

`RadioButton` é um `Button` que mantém um estado de seleção exclusivamente dentro de um `ButtonGroup` específico .

StaticAnimation

Uma Imagem capaz de animação.

TabbedPane

Um componente que deixa o usuário trocar entre um grupo de componentes ao clicar em uma aba com um dado título e/ou ícone.

TextArea

Uma região editável multi-linhas opcional que pode exibir texto e permitir um usuário a editá-lo.

TextField

Permite edição no local usando uma API simples sem necessariamente mover-se a caixa de texto nativa externa.

24.4 Classe AWTComponent

24.4.1 Descrição da classe

`com.sun.dtv.lwuit`

```
java.lang.Object
  └com.sun.dtv.lwuit.Component
      └com.sun.dtv.lwuit.AWTComponent
```

```
final public class AWTComponent
extends Component
```

Esta classe estende-se à classe `Component` como uma variante especial, que delega a pintura a um `java.awt.Component`. encapsulado. Enquanto em geral o LWUIT possui sues próprios modelos de componente sem o uso de mecanismos AWT, esta classe abre a possibilidade de integrar os componentes AWT no container controlado LWUIT

Esteja ciente de que permanece sendo não recomendado misturar componentes LWUIT e AWT em um aplicativo, já que isso pode causar problemas sérios de layout e é difícil de manusear. Por outro lado, pode haver situações que façam com que a integração de componentes AWT seja necessária, por exemplo, para o uso do legado de

componentes AWT. O código snippet a seguir ilustra como manusear essas situações:

```
import com.legacy.LegacyAWTComponent;
import com.sun.dtv.lwuit.Container;
import com.sun.dtv.lwuit.AWTComponent;

...
Container lwuitContainer = new COntainer(...);
...

LegacyAWTComponent legacyCmp = new LegacyAWTComponent();
lwuitCOntainer.addComponent(new AWTComponent(legacyCmp));
```

Campos herdados da classe com.sun.dtv.lwuit.**Component**

BOTTOM, BRB_CENTER_OFFSET, BRB_CONSTANT_ASCENT, BRB_CONSTANT_DESCENT, BRB_OTHER, CENTER, LEFT, RIGHT, TOP

Campos herdados da interface com.sun.dtv.ui.**Animated**

ALTERNATING, LOOP, REPEATING

Todas as interfaces implementadas

Animated, Animation, StyleListener

24.4.2

24.4.3 Índice de construtores

AWTComponent(Component component)

Constrói uma instância de AWTComponent, cobrindo o java.awt.Component especificado.

Métodos herdados da classe com.sun.dtv.lwuit.**Component**

addFocusListener, animate, calcPreferredSize, contains, deinitialize, getAbsoluteX, getAbsoluteY, getAnimationMode, getBaseline, getBaselineResizeBehavior, getBottomGap, getBounds, getClientProperty, getComponentForm, getDelay, getHeight, getNextFocusDown, getNextFocusLeft, getNextFocusRight, getNextFocusUp, getParent, getPosition, getPreferredSize, getRepetitionMode, getScrollAnimationSpeed, getScrollX, getScrollY, getSideGap, getStyle, getUIID, getWidth, getX, getY, handlesInput, hasFocus, initialize, isEnabled, isFocusable, isFocusPainted, isInitialized, isRunning, isScrollableX, isScrollableY, isScrollVisible, isSmoothScrolling, isVisible, jumpTo, keyPressed, keyReleased, keyRepeated, longKeyPress, paint, paintBackgrounds, paintComponent, paintComponent, pointerDragged, pointerPressed, pointerReleased, putClientProperty, refreshTheme, removeFocusListener, repaint, repaint, requestFocus, scrollRectToVisible, setAnimationMode, setCellRenderer, setDelay, setEnabled, setFocus, setFocusable, setFocusPainted, setHandlesInput, setHeight, setInitialized, setIsScrollVisible, setNextFocusDown, setNextFocusLeft, setNextFocusRight, setNextFocusUp, setPreferredSize, setRepetitionMode, setScrollAnimationSpeed, setScrollX, setScrollY, setShouldCalcPreferredSize, setSize, setSmoothScrolling, setStyle, setUIID, setVisible, setWidth, setX, setY, start, stop, styleChanged, toString

24.4.4 Detalhe dos construtores

AWTComponent

`public AWTComponent(Component component)`

Constrói uma instância de `AWTComponent`, cobrindo o `java.awt.Component` especificado.

Parâmetros:

`component` - o componente java.awt. a ser encapsulado

24.5 Classe Button

24.5.1 Descrição da classe

`com.sun.dtv.lwuit`

```
java.lang.Object
  └com.sun.dtv.lwuit.Component
      └com.sun.dtv.lwuit.Label
          └com.sun.dtv.lwuit.Button
```

`public class Button`

`extends Label`

Botão é a classe base para vários *UI widgets* permitindo clicabilidade. Ela possui 3 estados: rolagem, pressionado e o estado padrão, também pode ter `ActionListeners` que reagem quando o botão é clicado.

Todas as interfaces implementadas

`Animated, Animation, MatteEnabled, StyleListener, ViewOnlyComponent`

Subclasses diretas conhecidas

`CheckBox, RadioButton`

24.5.2 Índice de campos

`static int` **`STATE_DEFAULT`**

Indica o estado padrão de um botão que não está nem pressionado, nem destacado.

`static int` **`STATE_PRESSED`**

Indica o estado pressionado de um botão.

`static int` **`STATE_ROLLOVER`**

Indica o estado rolagem de um botão que equivale a destacado para a maioria dos usos.

Campos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`BOTTOM, BRB_CENTER_OFFSET, BRB_CONSTANT_ASCENT, BRB_CONSTANT_DESCENT, BRB_OTHER, CENTER, LEFT, RIGHT, TOP`

Campos herdados da interface `com.sun.dtv.ui.`**`Animated`**

ALTERNATING, LOOP, REPEATING

Campos herdados da interface com.sun.dtv.ui.**ViewOnlyComponent**

HORIZONTAL_ALIGN_CENTER, HORIZONTAL_ALIGN_JUSTIFIED, HORIZONTAL_ALIGN_LEFT, HORIZONTAL_ALIGN_RIGHT, SCALE_ASPECT_PROOF, SCALE_NO, SCALE_NO_ASPECT_PROOF, STATE_DISABLED, STATE_ENABLED, VERTICAL_ALIGN_BOTTOM, VERTICAL_ALIGN_CENTER, VERTICAL_ALIGN_JUSTIFIED, VERTICAL_ALIGN_TOP

24.5.3

24.5.4 Índice de construtores

Button()

Constrói um botão com um *string* vazio para seu texto.

Button(Command cmd)

Permite ligar um comando a um botão para facilitar o uso.

Button(Image icon)

Constrói um botão com a imagem especificada.

Button(String text)

Constrói um botão com o texto especificado.

Button(String text, Image icon)

Constrói um botão com texto e imagem.

24.5.5 Índice de métodos

void **addActionListener**(ActionListener l)

Adiciona um *listener* ao botão, o que causa que um evento seja despachado ao clicar.

Image **getPressedIcon**()

Indica o ícone que é exibido no botão, quando o botão está no estado pressionado.

Image **getRolloverIcon**()

Indica o ícone que é exibido no botão, quando o botão está no estado sobreposto.

int **getState**()

Retorna o estado do botão.

void **keyPressed**(int keyCode)

Se esse componente estiver destacado, o evento chave pressionado chamará esse método.

void **keyReleased**(int keyCode)

Se esse componente estiver destacado, o evento chave liberado chamará esse método.

void **paint**(Graphics g)

Método para pintar o componente.

void **pointerPressed**(int x, int y)

Se esse componente estiver destacado, o evento indicador pressionado chamará esse método.

void **pointerReleased**(int x, int y)

Se esse componente estiver destacado, o evento indicador liberado chamará esse método.

`void` **`removeActionListener`**`(ActionListener l)`

Remove um dado *listener* de ação do botão.

`void` **`setPressedIcon`**`(Image pressedIcon)`

Indica o ícone que é exibido no botão, quando o botão está no estado pressionado.

`void` **`setRolloverIcon`**`(Image rolloverIcon)`

Indica o ícone que é exibido no botão, quando o botão está no estado sobreposto.

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Label`**

`animate, getAlignment, getAnimateContent, getBaselineResizeBehavior, getGap, getGraphicContent, getHorizontalAlignment, getIcon, getInteractionState, getMatte, getScalingMode, getShiftText, getText, getTextContent, getTextLayoutManager, getTextPosition, getVerticalAlignment, isDoubleBuffered, isEndsWith3Points, isOpaque, isTickerEnabled, isTickerRunning, processEvent, setAlignment, setAnimateContent, setEndsWith3Points, setGap, setGraphicContent, setHorizontalAlignment, setIcon, setInteractionState, setMatte, setScalingMode, setShiftText, setText, setTextContent, setTextLayoutManager, setTextPosition, setTickerEnabled, setVerticalAlignment, startTicker, stopTicker`

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`addFocusListener, calcPreferredSize, contains, deinitialize, getAbsoluteX, getAbsoluteY, getAnimationMode, getBaseline, getBottomGap, getBounds, getClientProperty, getComponentForm, getDelay, getHeight, getNextFocusDown, getNextFocusLeft, getNextFocusRight, getNextFocusUp, getParent, getPosition, getPreferredSize, getRepetitionMode, getScrollAnimationSpeed, getScrollX, getScrollY, getSideGap, getStyle, getUIID, getWidth, getX, getY, handlesInput, hasFocus, initialize, isEnabled, isFocusable, isFocusPainted, isInitialized, isRunning, isScrollableX, isScrollableY, isScrollVisible, isSmoothScrolling, isVisible, jumpTo, keyRepeated, longKeyPress, paintBackgrounds, paintComponent, paintComponent, pointerDragged, putClientProperty, refreshTheme, removeFocusListener, repaint, repaint, requestFocus, scrollRectToVisible, setAnimationMode, setCellRenderer, setDelay, setEnabled, setFocus, setFocusable, setFocusPainted, setHandlesInput, setHeight, setInitialized, setIsScrollVisible, setNextFocusDown, setNextFocusLeft, setNextFocusRight, setNextFocusUp, setPreferredSize, setRepetitionMode, setScrollAnimationSpeed, setScrollX, setScrollY, setShouldCalcPreferredSize, setSize, setSmoothScrolling, setStyle, setUIID, setVisible, setWidth, setX, setY, start, stop, styleChanged, toString`

24.5.6 Detalhe dos campos

STATE_ROLLOVER

`public static final int STATE_ROLLOVER = 0`

Indica o estado rolagem de um botão que equivale a destacado para a maioria dos usos.

STATE_PRESSED

`public static final int STATE_PRESSED = 1`

Indica o estado pressionado de um botão.

STATE_DEFAULT

```
public static final int STATE_DEFAULT = 2
```

Indica o estado padrão de um botão que não está nem pressionado, nem destacado.

24.5.7 Detalhe dos construtores

Button

```
public Button()
```

Constrói um botão com um *string* vazio para seu texto.

Button

```
public Button(String text)
```

Constrói um botão com o texto especificado.

Parâmetros:

`text` - etiqueta aparecendo no botão

Button

```
public Button(Command cmd)
```

Permite ligar um comando a um botão para facilitar o uso.

Parâmetros:

`cmd` - comando cujo texto deve ser usado para o botão e receberá eventos de ação a partir do botão

Button

```
public Button(Image icon)
```

Constrói um botão com a imagem especificada.

Parâmetros:

`icon` - aparecendo no botão

Button

```
public Button(String text,
              Image icon)
```

Constrói um botão com texto e imagem.

Parâmetros:

`text` - etiqueta aparecendo no botão

`icon` - imagem aparecendo no botão

24.5.8 Detalhe dos métodos

getState

`public int getState()`

Retorna o estado do botão.

Retorna:

Um dos valores `STATE_ROLLOVER, STATE_DEAFULT, STATE_PRESSED`

getPressedIcon

`public Image getPressedIcon()`

Indica o ícone que é exibido no botão, quando o botão está no estado pressionado.

Retorna:

ícones usados

Relaciona-se com:

`setPressedIcon(com.sun.dtv.lwuit.Image), STATE_PRESSED`

getRolloverIcon

`public Image getRolloverIcon()`

Indica o ícone que é exibido no botão, quando o botão está no estado sobreposto.

Retorna:

ícones usados

Relaciona-se com:

`setRolloverIcon(com.sun.dtv.lwuit.Image), STATE_ROLLOVER`

setRolloverIcon

`public void setRolloverIcon(Image rolloverIcon)`

Indica o ícone que é exibido no botão, quando o botão está no estado sobreposto.

Parâmetros:

`rolloverIcon` - ícone a ser usado

Relaciona-se com:

`getRolloverIcon(), STATE_ROLLOVER`

setPressedIcon

`public void setPressedIcon(Image pressedIcon)`

Indica o ícone que é exibido no botão, quando o botão está no estado pressionado.

Parâmetros:

`pressedIcon` - ícone usado

Relaciona-se com:

`getPressedIcon(),STATE_PRESSED`

addActionListener

`public void addActionListener(ActionListener l)`

Adiciona um *listener* ao botão, o que causa que um evento seja despachado ao clicar.

Parâmetros:

`l` – implementação da interface de *listener* de ação

Relaciona-se com:

`removeActionListener(com.sun.dtv.lwuit.events.ActionListener)`

removeActionListener

`public void removeActionListener(ActionListener l)`

Remove um dado *listener* de ação do botão.

Parâmetros:

`l` – implementação da interface de *listener* de ação

Relaciona-se com:

`addActionListener(com.sun.dtv.lwuit.events.ActionListener)`

keyPressed

`public void keyPressed(int keyCode)`

Se esse componente estiver destacado, o evento chave pressionado chamará esse método.

Substituições:

`keyPressed` na classe `Component`

Parâmetros:

`keyCode` - o valor do código chave a indicar a chave física.

keyReleased

`public void keyReleased(int keyCode)`

Se esse componente estiver destacado, o evento chave liberado chamará esse método.

Substituições:

`keyReleased` na classe `Component`

Parâmetros:

`keyCode` - o valor do código chave a indicar a chave física.

pointerPressed

```
public void pointerPressed(int x,
                           int y)
```

Se esse componente estiver destacado, o evento indicador pressionado chamará esse método.

Substituições:

`pointerPressed` na classe `Component`

Parâmetros:

`x` - a coordenada do indicador x

`y` - a coordenada do indicador y

pointerReleased

```
public void pointerReleased(int x,
                            int y)
```

Se esse componente estiver destacado, o evento indicador liberado chamará esse método.

Substituições:

`pointerReleased` na classe `Component`

Parâmetros:

`x` - a coordenada do indicador x

`y` - a coordenada do indicador y

paint

```
public void paint(Graphics g)
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Método para pintar o componente.

Especificado por:

`paint` na interface `ViewOnlyComponent`

`paint` na interface `Animation`

Substituições:

`paint` na classe `Label`

Parâmetros:

`g` - o contexto gráfico a ser usado para pintar.

24.6 Classe ButtonGroup

24.6.1 Descrição da classe

`com.sun.dtv.lwuit`

```
java.lang.Object
  └com.sun.dtv.lwuit.ButtonGroup
```

```
public class ButtonGroup
extends Object
```

Essa classe é usada para um escopo de exclusão-múltipla para um conjunto de `RadioButtons`. A criação de um conjunto de `RadioButtons` com o mesmo objeto `ButtonGroup` significa que apenas um `RadioButton` pode ser selecionado entre o `ButtonGroup`. Inicialmente todos `RadioButtons` são desselecionados.

24.6.2 Índice de construtores

ButtonGroup()

Cria uma nova instância de ButtonsGroup.

24.6.3 Índice de métodos

void **add**(RadioButton rb)

Adiciona um RadioButton ao grupo.

void **clearSelection**()

Limpa a seleção de tal forma que nenhum dos botões no ButtonGroup esteja selecionado.

int **getButtonCount**()

Retorna o número de botões no grupo.

RadioButton **getRadioButton**(int index)

Retorna o botão de rádio a um dado índice de grupo.

int **getSelectedIndex**()

Retorna o índice do botão selecionado dentro do grupo.

boolean **isSelected**()

Retorna se um botão de rádio no grupo está selecionado.

void **remove**(RadioButton rb)

Remove um RadioButton do grupo.

void **setSelected**(RadioButton rb)

Seleciona um dado botão de rádio.

void **setSelected**(int index)

Define o botão de Rádio selecionado por índice.

24.6.4 Detalhe dos construtores

ButtonGroup

public ButtonGroup()

Cria uma nova instância de ButtonsGroup.

24.6.5 Detalhe dos métodos

add

public void add(RadioButton rb)

Adiciona um RadioButton ao grupo.

Parâmetros:

rb - um RadioButton a ser adicionado

remove

`public void remove(RadioButton rb)`

Remove um `RadioButton` do grupo.

Parâmetros:

`rb` - um `RadioButton` a ser removido

clearSelection

`public void clearSelection()`

Limpa a seleção de tal forma que nenhum dos botões no `ButtonGroup` esteja selecionado.

getButtonCount

`public int getButtonCount()`

Retorna o número de botões no grupo.

Retorna:

número de botões de rádio no grupo.

isSelected

`public boolean isSelected()`

Retorna se um botão de rádio no grupo está selecionado.

Retorna:

true se uma seleção houver sido feita no grupo de botões de rádio

getSelectedIndex

`public int getSelectedIndex()`

Retorna o índice do botão selecionado dentro do grupo.

Retorna:

o índice do botão selecionado dentro do grupo.

getRadioButton

`public RadioButton getRadioButton(int index)`

Retorna o botão de rádio a um dado índice de grupo.

Parâmetros:

`index` - ajuste dentro do grupo começando com 0 e não maior que `getButtonCount()`

Retorna:

A instância botão de rádio

setSelected

`public void setSelected(RadioButton rb)`

Seleciona um dado botão de rádio.

Parâmetros:

`rb` - o botão de rádio a definir como selecionado

setSelected

`public void setSelected(int index)`

Define o botão de Rádio selecionado por índice.

Parâmetros:

`index` - o índice do botão de rádio a ser marcado como selecionado

24.7 Classe Calendar

24.7.1 Descrição da classe

`com.sun.dtv.lwuit`

```
java.lang.Object
  └com.sun.dtv.lwuit.Component
      └com.sun.dtv.lwuit.Container
          └com.sun.dtv.lwuit.Calendar
```

`public class Calendar`

`extends Container`

Data o *widget* para selecionar um valor data/hora.

Para localizar *strings* por nomes de meses use os valores "`Calendar.Month`" no local onde se encontra o recurso, por exemplo: "`Calendar.Jan`", "`Calendar.Feb`", etc...

Campos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`BOTTOM, BRB_CENTER_OFFSET, BRB_CONSTANT_ASCENT, BRB_CONSTANT_DESCENT, BRB_OTHER, CENTER, LEFT, RIGHT, TOP`

Campos herdados da interface `com.sun.dtv.ui.`**`Animated`**

`ALTERNATING, LOOP, REPEATING`

Todas as interfaces implementadas

`Animated, Animation, MatteEnabled, StyleListener`

24.7.2 Índice de construtores

`Calendar``()`

Constrói um calendário com a data atual e as horas do dia.

Calendar(long time)

Cria uma nova instância de Calendário definida pela data especificada baseada no tempo desde a epoch (a convenção java.util.Date).

24.7.3 Índice de métodos

void **addActionListener**(ActionListener l)

É ativado quando uma mudança é feita na visualização de mês desse componente.

Date **getDate**()

Retorna o objeto data correspondente a seleção atual.

Style **getMonthViewStyle**()

Define o estilo do componente visualização do mês dentro do calendário.

long **getSelectedDay**()

Retorna a hora do dia para o calendário atual.

void **paint**(Graphics g)

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

void **refreshTheme**()

Certifica-se de que o componente está atualizado com o objeto estilo atual.

void **removeActionListener**(ActionListener l)

É ativado quando uma mudança é feita na visualização de mês desse componente.

void **setMonthViewStyle**(Style s)

Define o estilo do componente visualização do mês dentro do calendário.

void **setStyle**(Style s)

Muda o estilo do componente trocando-o por um dado estilo.

Métodos herdados da classe com.sun.dtv.lwuit.**Container**

addComponent, addComponent, addComponent, contains, doLayout, getComponentAt, getComponentAt, getComponentCount, getComponentIndex, getLayout, getLayoutHeight, getLayoutWidth, getMatte, invalidate, isScrollableX, isScrollableY, layoutContainer, pointerPressed, removeAll, removeComponent, replace, revalidate, scrollComponentToVisible, setCellRenderer, setLayout, setMatte, setScrollable, setScrollableX, setScrollableY

Métodos herdados da classe com.sun.dtv.lwuit.**Component**

addFocusListener, animate, calcPreferredSize, contains, deinitialize, getAbsoluteX, getAbsoluteY, getAnimationMode, getBaseline, getBaselineResizeBehavior, getBottomGap, getBounds, getClientProperty, getComponentForm, getDelay, getHeight, getNextFocusDown, getNextFocusLeft, getNextFocusRight, getNextFocusUp, getParent, getPosition, getPreferredSize, getRepetitionMode, getScrollAnimationSpeed, getScrollX, getScrollY, getSideGap, getStyle, getUIID, getWidth, getX, getY, handlesInput, hasFocus, initialize, isEnabled, isFocusable, isFocusPainted, isInitialized, isRunning, isScrollVisible, isSmoothScrolling, isVisible, jumpTo, keyPressed, keyReleased, keyRepeated, longKeyPress, paintBackgrounds, paintComponent,

paintComponent, pointerDragged, pointerReleased, putClientProperty, removeFocusListener, repaint, repaint, requestFocus, scrollRectToVisible, setAnimationMode, setDelay, setEnabled, setFocus, setFocusable, setFocusPainted, setHandlesInput, setHeight, setInitialized, setIsScrollVisible, setNextFocusDown, setNextFocusLeft, setNextFocusRight, setNextFocusUp, setPreferredSize, setRepetitionMode, setScrollAnimationSpeed, setScrollX, setScrollY, setShouldCalcPreferredSize, setSize, setSmoothScrolling, setUIID, setVisible, setWidth, setX, setY, start, stop, styleChanged, toString

24.7.4 Detalhe dos construtores

Calendar

```
public Calendar(long time)
```

Cria uma nova instância de Calendário definida pela data especificada baseada no tempo desde a epoch (a convenção java.util.Date).

Parâmetros:

`time` - tempo desde a epoch

Calendar

```
public Calendar()
```

Constrói um calendário com a data atual e as horas do dia.

24.7.5 Detalhe dos métodos

getSelectedDay

```
public long getSelectedDay()
```

Retorna a hora do dia para o calendário atual.

Retorna:

a hora do dia para o calendário atual.

getDate

```
public Date getDate()
```

Retorna o objeto data correspondente a seleção atual.

Retorna:

o objeto data correspondente a seleção atual.

paint

```
public void paint(Graphics g)
```

Descrição copiada da interface: **`Animation`**

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

Especificado por:

`paint` na interface `Animation`

Substituições:

`paint` na classe `Container`

Parâmetros:

`g` - os gráficos do componente

refreshTheme

```
public void refreshTheme()
```

Certifica-se de que o componente está atualizado com o objeto estilo atual.

Substituições:

`refreshTheme` na classe `Container`

setStyle

```
public void setStyle(Style s)
```

Muda o estilo do componente trocando-o por um dado estilo.

Substituições:

`setStyle` na classe `Component`

setMonthViewStyle

```
public void setMonthViewStyle(Style s)
```

Define o estilo do componente visualização do mês dentro do calendário.

Parâmetros:

`s` - um objeto estilo especificando a o valor do estilo da visualização do mês

Relaciona-se com:

`getMonthViewStyle()`

getMonthViewStyle

```
public Style getMonthViewStyle()
```

Define o estilo do componente visualização do mês dentro do calendário.

Retorna:

um objeto estilo representando o valor do estilo da visualização do mês

Relaciona-se com:

`setMonthViewStyle(com.sun.dtv.lwuit.plaf.Style)`

addActionListener

```
public void addActionListener(ActionListener l)
```

É ativado quando uma mudança é feita na visualização de mês desse componente.

Parâmetros:

`l` - um objeto `ActionListener`

Relaciona-se com:

removeActionListener(com.sun.dtv.lwuit.events.ActionListener)

removeActionListener

public void removeActionListener(ActionListener l)

É ativado quando uma mudança é feita na visualização de mês desse componente.

Parâmetros:

l - um objeto ActionListener

Relaciona-se com:

addActionListener(com.sun.dtv.lwuit.events.ActionListener)

24.8 Classe CheckBox

24.8.1 Descrição da classe

com.sun.dtv.lwuit

java.lang.Object

└com.sun.dtv.lwuit.Component

└com.sun.dtv.lwuit.Label

└com.sun.dtv.lwuit.Button

└com.sun.dtv.lwuit.CheckBox

public class CheckBox

extends Button

Checkbox é um botão que pode ser selecionado e desselecionado e que exibe seu estado ao usuário.

Campos herdados da classe com.sun.dtv.lwuit.**Button**

STATE_DEFAULT, STATE_PRESSED, STATE_ROLLOVER

Campos herdados da classe com.sun.dtv.lwuit.**Component**

BOTTOM, BRB_CENTER_OFFSET, BRB_CONSTANT_ASCENT, BRB_CONSTANT_DESCENT, BRB_OTHER, CENTER, LEFT, RIGHT, TOP

Campos herdados da interface com.sun.dtv.ui.**Animated**

ALTERNATING, LOOP, REPEATING

Campos herdados da interface com.sun.dtv.ui.**ViewOnlyComponent**

HORIZONTAL_ALIGN_CENTER, HORIZONTAL_ALIGN_JUSTIFIED, HORIZONTAL_ALIGN_LEFT, HORIZONTAL_ALIGN_RIGHT, SCALE_ASPECT_PROOF, SCALE_NO, SCALE_NO_ASPECT_PROOF, STATE_DISABLED, STATE_ENABLED, VERTICAL_ALIGN_BOTTOM, VERTICAL_ALIGN_CENTER, VERTICAL_ALIGN_JUSTIFIED, VERTICAL_ALIGN_TOP

Todas as interfaces implementadas

`Animated, Animation, MatteEnabled, StyleListener, ViewOnlyComponent`

24.8.2 Índice de construtores

`CheckBox``()`

Constrói um checkbox sem texto.

`CheckBox``(Image icon)`

Constrói um checkbox com um dado ícone.

`CheckBox``(String text)`

Constrói um checkbox com um dado texto.

`CheckBox``(String text, Image icon)`

Constrói um checkbox com um dado ícone e texto.

24.8.3 Índice de métodos

`boolean` **`isSelected`**`()`

Retorna true se o checkbox for selecionado.

`void` **`paint`**`(Graphics g)`

Método para pintar o componente.

`void` **`setSelected`**`(boolean selected)`

Seleciona o checkbox corrente.

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Button`**

`addActionListener, getPressedIcon, getRolloverIcon, getState, keyPressed, keyReleased, pointerPressed, pointerReleased, removeActionListener, setPressedIcon, setRolloverIcon`

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Label`**

`animate, getAlignment, getAnimateContent, getBaselineResizeBehavior, getGap, getGraphicContent, getHorizontalAlignment, getIcon, getInteractionState, getMatte, getScalingMode, getShiftText, getText, getTextContent, getTextLayoutManager, getTextPosition, getVerticalAlignment, isDoubleBuffered, isEndsWith3Points, isOpaque, isTickerEnabled, isTickerRunning, processEvent, setAlignment, setAnimateContent, setEndsWith3Points, setGap, setGraphicContent, setHorizontalAlignment, setIcon, setInteractionState, setMatte, setScalingMode, setShiftText, setText, setTextContent, setTextLayoutManager, setTextPosition, setTickerEnabled, setVerticalAlignment, startTicker, stopTicker`

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`addFocusListener, calcPreferredSize, contains, deinitialize, getAbsoluteX, getAbsoluteY, getAnimationMode, getBaseline, getBottomGap, getBounds, getClientProperty, getComponentForm, getDelay, getHeight, getNextFocusDown, getNextFocusLeft, getNextFocusRight, getNextFocusUp, getParent, getPosition,`

getPreferredSize, getRepetitionMode, getScrollAnimationSpeed, getScrollX, getScrollY, getSideGap, getStyle, getUIID, getWidth, getX, getY, handlesInput, hasFocus, initialize, isEnabled, isFocusable, isFocusPainted, isInitialized, isRunning, isScrollableX, isScrollableY, isScrollVisible, isSmoothScrolling, isVisible, jumpTo, keyRepeated, longKeyPress, paintBackgrounds, paintComponent, paintComponent, pointerDragged, putClientProperty, refreshTheme, removeFocusListener, repaint, repaint, requestFocus, scrollRectToVisible, setAnimationMode, setCellRenderer, setDelay, setEnabled, setFocus, setFocusable, setFocusPainted, setHandlesInput, setHeight, setInitialized, setIsScrollVisible, setNextFocusDown, setNextFocusLeft, setNextFocusRight, setNextFocusUp, setPreferredSize, setRepetitionMode, setScrollAnimationSpeed, setScrollX, setScrollY, setShouldCalcPreferredSize, setSize, setSmoothScrolling, setStyle, setUIID, setVisible, setWidth, setX, setY, start, stop, styleChanged, toString

24.8.4 Detalhe dos construtores

CheckBox

```
public CheckBox(String text)
```

Constrói um checkbox com um dado texto.

Parâmetros:

`text` - para exibir próximo ao checkbox

CheckBox

```
public CheckBox()
```

Constrói um checkbox sem texto.

CheckBox

```
public CheckBox(Image icon)
```

Constrói um checkbox com um dado ícone.

Parâmetros:

`icon` - ícone para exibir próximo ao checkbox

CheckBox

```
public CheckBox(String text,
                Image icon)
```

Constrói um checkbox com um dado ícone e texto.

Parâmetros:

`text` - para exibir próximo ao checkbox

`icon` - ícone para exibir próximo ao texto

24.8.5 Detalhe dos métodos

isSelected

`public boolean isSelected()`

Retorna true se o checkbox for selecionado.

Retorna:

true se o checkbox for selecionado.

setSelected

`public void setSelected(boolean selected)`

Seleciona o checkbox corrente.

Parâmetros:

`selected` - valor para seleção

paint

`public void paint(Graphics g)`

Descrição copiada da interface: **`ViewOnlyComponent`**

Método para pintar o componente.

Especificado por:

`paint` na interface `ViewOnlyComponent`

`paint` na interface `Animation`

Substituições:

`paint` na classe `Button`

Parâmetros:

`g` - o contexto gráfico a ser usado para pintar.

24.9 Classe ComboBox

24.9.1 Descrição da classe

`com.sun.dtv.lwuit`

```
java.lang.Object
  └com.sun.dtv.lwuit.Component
      └com.sun.dtv.lwuit.List
          └com.sun.dtv.lwuit.ComboBox
```

`public class ComboBox`

`extends List`

Um combo Box é uma lista que permite apenas uma seleção por vez, quando um usuário clica no combo box, um botão popup com a lista completa de elementos permite a seleção de um único elemento. A combo Box é guiada pelo modelo de lista que também permite todas as funções exibidoras da Lista.

Relaciona-se com:

`List`

Campos herdados da classe `com.sun.dtv.lwuit.`**`List`**

FIXED_CENTER, FIXED_LEAD, FIXED_NONE, FIXED_NONE_CYCLIC, FIXED_NONE_ONE_ELEMENT_MARGIN_FROM_EDGE, FIXED_TRAIL, HORIZONTAL, VERTICAL

Campos herdados da classe com.sun.dtv.lwuit.**Component**

BOTTOM, BRB_CENTER_OFFSET, BRB_CONSTANT_ASCENT, BRB_CONSTANT_DESCENT, BRB_OTHER, CENTER, LEFT, RIGHT, TOP

Campos herdados da interface com.sun.dtv.ui.**Animated**

ALTERNATING, LOOP, REPEATING

Campos herdados da interface com.sun.dtv.ui.**ViewOnlyComponent**

HORIZONTAL_ALIGN_CENTER, HORIZONTAL_ALIGN_JUSTIFIED, HORIZONTAL_ALIGN_LEFT, HORIZONTAL_ALIGN_RIGHT, SCALE_ASPECT_PROOF, SCALE_NO, SCALE_NO_ASPECT_PROOF, STATE_DISABLED, STATE_ENABLED, VERTICAL_ALIGN_BOTTOM, VERTICAL_ALIGN_CENTER, VERTICAL_ALIGN_JUSTIFIED, VERTICAL_ALIGN_TOP

Todas as interfaces implementadas

Animated, Animation, MatteEnabled, StyleListener, ViewOnlyComponent

24.9.2 Índice de construtores

ComboBox()

Constrói um combo box vazio.

ComboBox(ListModel model)

Cria uma nova instância de ComboBox.

ComboBox(Object[] items)

Cria uma nova instância de ComboBox.

ComboBox(Vector items)

Cria uma nova instância de ComboBox.

24.9.3 Índice de métodos

int **getBaseline**(int width, int height)

A base para o texto do componente de acordo com o qual deveria ser alinhado com outros componentes para um visual melhor.

void **paint**(Graphics g)

Método para pintar o componente.

void **pointerReleased**(int x, int y)

Se esse componente estiver destacado, o evento indicador liberado chamará esse método.

void **setHandlesInput**(boolean handlesInput)

Previne que eventos chave sejam tomados para foco transversal.

`void` **`setModel`**`(ListModel m)`

Substitui/define o modelo subjacente à lista.

`void` **`setSelectedIndex`**`(int selection)`

Define o atual ajuste selecionado na lista, padronizadamente essa implementação rolará a lista até a seleção, se a seleção estiver fora da tela.

`void` **`setSelectedIndex`**`(int selection, boolean scroll)`

Define o atual ajuste selecionado na lista.

Métodos herdados da classe `com.sun.dtv.lwuit.`**`List`**

`addActionListener, addItem, addSelectionListener, animate, getAnimateContent, getBorderGap, getFixedSelection, getGraphicContent, getHorizontalAlignment, getInteractionState, getItemGap, getMatte, getModel, getOrientation, getRenderer, getRenderingPrototype, getScalingMode, getSelectedIndex, getSelectedItem, getTextContent, getTextLayoutManager, getVerticalAlignment, isDoubleBuffered, isNumericKeyActions, isOpaque, isScrollableX, isScrollableY, keyPressed, keyReleased, pointerDragged, processEvent, refreshTheme, removeActionListener, removeSelectionListener, scrollRectToVisible, setAnimateContent, setBorderGap, setFixedSelection, setGraphicContent, setHorizontalAlignment, setInputOnFocus, setInteractionState, setItemGap, setListCellRenderer, setMatte, setNumericKeyActions, setOrientation, setPaintFocusBehindList, setRenderingPrototype, setScalingMode, setSelectedItem, setTextContent, setTextLayoutManager, setVerticalAlignment, size`

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`addFocusListener, calcPreferredSize, contains, deinitialize, getAbsoluteX, getAbsoluteY, getAnimationMode, getBaselineResizeBehavior, getBottomGap, getBounds, getClientProperty, getComponentForm, getDelay, getHeight, getNextFocusDown, getNextFocusLeft, getNextFocusRight, getNextFocusUp, getParent, getPosition, getPreferredSize, getRepetitionMode, getScrollAnimationSpeed, getScrollX, getScrollY, getSideGap, getStyle, getUIID, getWidth, getX, getY, handlesInput, hasFocus, initialize, isEnabled, isFocusable, isFocusPainted, isInitialized, isRunning, isScrollVisible, isSmoothScrolling, isVisible, jumpTo, keyRepeated, longKeyPress, paintBackgrounds, paintComponent, paintComponent, pointerPressed, putClientProperty, removeFocusListener, repaint, repaint, requestFocus, scrollRectToVisible, setAnimationMode, setCellRenderer, setDelay, setEnabled, setFocus, setFocusable, setFocusPainted, setHeight, setInitialized, setIsScrollVisible, setNextFocusDown, setNextFocusLeft, setNextFocusRight, setNextFocusUp, setPreferredSize, setRepetitionMode, setScrollAnimationSpeed, setScrollX, setScrollY, setShouldCalcPreferredSize, setSize, setSmoothScrolling, setStyle, setUIID, setVisible, setWidth, setX, setY, start, stop, styleChanged, toString`

24.9.4 Detalhe dos construtores

ComboBox

`public ComboBox(Vector items)`

Cria uma nova instância de `ComboBox`.

Parâmetros:

`items` - conjunto de itens colocados no modelo de combo box

ComboBox

```
public ComboBox(Object[] items)
```

Cria uma nova instância de `ComboBox`.

Parâmetros:

`items` - conjunto de itens colocados no modelo de combo box

ComboBox

```
public ComboBox()
```

Constrói um combo box vazio.

ComboBox

```
public ComboBox(ListModel model)
```

Cria uma nova instância de `ComboBox`.

Parâmetros:

`model` - o modelo de elementos e seleção combo box

24.9.5 Detalhe dos métodos

getBaseline

```
public int getBaseline(int width,
                       int height)
```

A base para o texto do componente de acordo com o qual deveria ser alinhado com outros componentes para um visual melhor.

Substituições:

`getBaseline` na classe `Component`

Parâmetros:

`width` - a largura do componente

`height` - a altura do componente

Retorna:

Valor de base do topo do componente.

setSelectedIndex

```
public void setSelectedIndex(int selection)
```

Define o atual ajuste selecionado na lista, padronizadamente essa implementação rolará a lista até a seleção, se a seleção estiver fora da tela.

Substituições:

`setSelectedIndex` na classe `List`

setSelectedIndex

```
public void setSelectedIndex(int selection,
```

```
                                boolean scroll)
```

Define o atual ajuste selecionado na lista.

Substituições:

`setSelectedIndex` na classe `List`

setModel

```
public void setModel(ListModel m)
```

Substitui/define o modelo subjacente à lista.

Substituições:

`setModel` na classe `List`

setHandlesInput

```
public void setHandlesInput(boolean handlesInput)
```

Previne que as teclas de evento fiquem fixas em foco horizontal.Por exemplo: um componente da lista pode usar as teclas direcionais para navegação interna, portanto, ele mudará essa *flag* para true para prevenir que o gerenciador de foco se mova ao próximo componente.

Substituições:

`setHandlesInput` na classe `List`

Parâmetros:

`handlesInput` - indica se eventos chaves podem ser tomados para foco transversal

pointerReleased

```
public void pointerReleased(int x,
                            int y)
```

Se esse componente estiver destacado, o evento indicador liberado chamará esse método.

Substituições:

`pointerReleased` na classe `List`

Parâmetros:

`x` - a coordenada do indicador x

`y` - a coordenada do indicador y

paint

```
public void paint(Graphics g)
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Método para pintar o componente.

Especificado por:

`paint` na interface `ViewOnlyComponent`

`paint` na interface `Animation`

Substituições:

`paint` na classe `List`

Parâmetros:

`g` - o contexto gráfico a ser usado para pintar.

24.10 Classe Command

24.10.1 Descrição da classe

`com.sun.dtv.lwuit`

```
java.lang.Object
  └com.sun.dtv.lwuit.Command
```

```
public class Command
extends Object
implements ActionListener
```

A ação colocada nos botões soft e no Menu em dispositivos, similares ao comando de abstração do MIDP e às Ações do Swing. Ao contrário da abstração do MIDP, essa classe pode ser obtida para implementar o comportamento correto

Todas as interfaces implementadas

`ActionListener`

24.10.2 Índice de construtores

`Command``(String command)`

Cria um nova instância de Comando.

`Command``(String command, Image icon)`

Cria um nova instância de Comando.

`Command``(String command, Image icon, int id)`

Cria um nova instância de Comando.

`Command``(String command, int id)`

Cria um nova instância de Comando.

24.10.3 Índice de métodos

`void` **`actionPerformed`**`(ActionEvent evt)`

Esse método é chamado quanto o item Menu/botão soft é clicado.

`boolean` **`equals`**`(Object obj)`

Compara dois comandos.

`String` **`getCommandName`**`()`

Obtém o Nome do Comando.

`Image` **`getIcon`**`()`

Retorna o ícone representando o comando.

`int` **`getId`**`()`

Retorna o identificador do comando.

`int` **`hashCode`**`()`

`String` **`toString`**`()`

Retorna uma representação *string* do objeto.

24.10.4 Detalhe dos construtores

Command

```
public Command(String command)
```

Cria um nova instância de Comando.

Parâmetros:

`command` - a *string* que deve ser colocado nos botões Soft\Menu

Command

```
public Command(String command,
               Image icon)
```

Cria um nova instância de Comando.

Parâmetros:

`command` - a *string* que deve ser colocado nos botões Soft\Menu

`icon` - O ícone representando o comando

Command

```
public Command(String command,
               int id)
```

Cria um nova instância de Comando.

Parâmetros:

`command` - a *string* que deve ser colocado nos botões Soft\Menu

`id` - identificador definido pelo usuário para um comando simplificando a troca de código statement trabalhando com um comando

Command

```
public Command(String command,
               Image icon,
               int id)
```

Cria um nova instância decomando.

Parâmetros:

`command` - a *string* que deve ser colocado nos botões *Soft\Menu*

`icon` - O ícone representando o comando

`id` - identificador defino pelo usuário para um comando simplificando a troca de código *statement* trabalhando com um comando

24.10.5 Detalhe dos métodos

getId

`public int getId()`

Retorna o identificador do comando.

Retorna:

o identificador do comando

getCommandName

`public String getCommandName()`

Obtém o Nome do Comando.

Retorna:

o nome do Comando.

getIcon

`public Image getIcon()`

Retorna o ícone representando o comando.

Retorna:

um ícone representando o comando

toString

`public String toString()`

Retorna uma representação *string* do objeto.

Substituições:

`toString` na classe `Object`

Retorna:

Retorna uma representação *string* do objeto.

equals

`public boolean equals(Object obj)`

Compara dois comandos.

Substituições:

`equals` na classe `Object`

Parâmetros:

`obj` - um Objeto Comando a ser comparado

Retorna:

true se o obj tiver o mesmo nome de comando

hashCode

`public int hashCode()`

Substituições:

`hashCode` na classe `Object`

actionPerformed

`public void actionPerformed(ActionEvent evt)`

Esse método é chamado quanto o item Menu/botão soft é clicado.

Especificado por:

`actionPerformed` na interface `ActionListener`

Parâmetros:

`evt` - o Objeto Evento

24.11 Classe Component

24.11.1 Descrição da classe

`com.sun.dtv.lwuit`

```
java.lang.Object
  └com.sun.dtv.lwuit.Component
```

```
public class Component
extends Object
implements Animation, Animated, StyleListener
```

Classe base para todos os *widget*s no kit de ferramentas usando o padrão composto em uma maneira similar a relação `Container AWT/Component`. Todos os componentes são potencialmente animados (precisa ser registrado em `Form`).

Relaciona-se com:

`Form.registerAnimation(com.sun.dtv.lwuit.animations.Animation)`

Todas as interfaces implementadas

`Animated, Animation, StyleListener`

Subclasses diretas conhecidas

`AWTComponent, Container, Label, List, MediaComponent, TextArea`

24.11.2 Índice de campos

`static int` **`BOTTOM`**

A constante de orientação por caixa usado para especificar o fundo de uma caixa.

`static int` **`BRB_CENTER_OFFSET`**

Constante de comportamento de redimensionamento de base usado para alinhar componentes apropriadamente.

`static int` **`BRB_CONSTANT_ASCENT`**

Constante de comportamento de redimensionamento de base usado para alinhar componentes apropriadamente.

`static int` **`BRB_CONSTANT_DESCENT`**

Constante de comportamento de redimensionamento de base usado para alinhar componentes apropriadamente.

`static int` **`BRB_OTHER`**

Constante de comportamento de redimensionamento de base usado para alinhar componentes apropriadamente.

`static int` **`CENTER`**

Indica o centro de alinhamento de um componente.

`static int` **`LEFT`**

A constante de orientação por caixa usada para especificar o lado esquerdo de uma caixa.

`static int` **`RIGHT`**

A constante de orientação por caixa usada para especificar o lado direito de uma caixa.

`static int` **`TOP`**

A constante de orientação por caixa usada para especificar o topo de uma caixa.

Campos herdados da interface `com.sun.dtv.ui.`**`Animated`**

`ALTERNATING, LOOP, REPEATING`

24.11.3 Índice de construtores

`protected` **`Component`**`()`

Cria uma nova instância de componente.

24.11.4 Índice de métodos

`void addFocusListener(FocusListener l)`

Registra interesse em receber chamada de retorno para eventos focusgained; um evento de foco é invocado quando o componente aceitam o foco.

`boolean animate()`

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*.

`protected Dimension calcPreferredSize()`

Calcula o tamanho preferível baseado no conteúdo do componente.

`boolean contains(int x, int y)`

Retorna true se uma dada coordenada absoluta estiver contida no componente.

`protected void deinitialize()`

Invocado para indicar que a inicialização do componente está sendo revertida desde que o componente foi removido da hierarquia do container.

`int getAbsoluteX()`

Retorna o local X absoluto baseado na hierarquia do componente; esse método calcula o local exato na tela do componente em vez de um local relativo como retornado por getX().

```
int getAbsoluteY()
```

Retorna o local Y absoluto baseado na hierarquia do componente; esse método calcula o local exato na tela do componente em vez de um local relativo como retornado por getY().

```
int getAnimationMode()
```

Obtém o modo de animação para essa animação.

```
int getBaseline(int width, int height)
```

A base para o texto do componente de acordo com o qual deveria ser alinhado com outros componentes para um visual melhor.

```
int getBaselineResizeBehavior()
```

Retorna uma constante indicando como a base varia com o tamanho do componente.

```
int getBottomGap()
```

Retorna o vão a ser deixado na barra de rolagem inferior no eixo X.

```
Rectangle getBounds()
```

Retorna as ligações do componente, o que, às vezes, é mais conveniente do que invocar getX/Y/Width/Height.

```
Object getClientProperty(String key)
```

Propriedades do cliente permitem a associação de metadados a um componente; isso é útil para alguns aplicativos que constroem GUIs dinamicamente e precisam rastrear a conexão entre o UI e os dados.

```
Form getComponentForm()
```

Retorna o formulário do componente ou `null`, se esse componente não for adicionado a um formulário.

```
int getDelay()
```

Obtém o atraso após cada imagem dessa animação ao rodar.

```
int getHeight()
```

Retorna a altura do componente

```
Component getNextFocusDown()
```

Permite determinar qual componente receberá destaque a seguir ao atravessar com a tecla direcional para baixo.

```
Component getNextFocusLeft()
```

Permite determinar qual componente receberá destaque a seguir ao atravessar com a tecla direcional esquerda.

```
Component getNextFocusRight()
```

Permite determinar qual componente receberá destaque a seguir ao atravessar com a tecla direcional direita.

```
Component getNextFocusUp()
```

Permite determinar qual componente receberá destaque a seguir ao atravessar com a tecla direcional para cima.

```
Container getParent()
```

Retorna o container no qual esse componente está contido.

```
int getPosition()
```

Obtém a posição atual que a animação ocupa no momento da chamada do método.

```
Dimension getPreferredSize()
```

Retorna o tamanho preferível do componente, sem garantia de que o componente deve ser dimensionado para seu tamanho preferível.

```
int getRepetitionMode()
```

Retorna o modo de repetição dessa animação na forma de um número.

```
int getScrollAnimationSpeed()
```

Coloca em barra de rolagem a velocidade da animação em milissegundos, permitindo que um desenvolvedor diminua ou acelere o modo de animação suave.

```
int getScrollX()
```

Indica a posição X da barra de rolagem; esse número é relativo à posição do componente e, portanto, uma posição 0 indicaria a posição x do componente.

```
int getScrollY()
```

Indica a posição Y da barra de rolagem; esse número é relativo à posição do componente e, portanto, uma posição 0 indicaria a posição x do componente.

```
int getSideGap()
```

Retorna o vão a ser deixado no barra de rolagem inferior no eixo Y.

```
Style getStyle()
```

Retorna o estilo do componente, permitindo manipular o visual do componente.

```
String getUIID()
```

Restaura um identificador único para um componente, deve ser sobreposto por um componente de forma que um estilo possa ser aplicado ao componente

```
int getWidth()
```

Retorna a largura do componente.

```
int getX()
```

Retorna o local x do componente atual relativo ao seu container principal.

```
int getY()
```

Retorna o local y do componente atual relativo ao seu container principal.

```
boolean handlesInput()
```

Previne que eventos chave sejam tomados para foco transversal.

```
boolean hasFocus()
```

Retorna true se o componente tiver foco.

```
protected void initialize()
```

Permite que subclasses liguem funcionalidades que se apóiam em estados de componentes totalmente inicializados e "prontos para ação"

```
boolean isEnabled()
```

Indica se o componente está disponível ou indisponível, permitindo, então, prevenir que um componente receba eventos de entrada e o indica visualmente.

```
boolean isFocusable()
```

Retorna true se esse componente puder receber destaque e estiver disponível.

```
boolean isFocusPainted()
```

Indica se o destaque deveria ser desenhado em volta do componente ou se ele manuseará seu próprio desenho de destaque.

```
protected boolean isInitialized()
```

Indica se o componente está no estado inicializado; um componente está inicializado quando seu método `initialize` foi invocado.

`boolean isRunning()`

Obtém o modo rodando de uma animação.

`boolean isScrollableX()`

Indica se o componente deveria/poderia rolar no eixo X.

`boolean isScrollableY()`

Indica se o componente deveria/poderia rolar no eixo Y.

`boolean isScrollVisible()`

Indica se a barra de rolagem desse componente é visível.

`boolean isSmoothScrolling()`

Indica que essa rolagem pelo componente deveria funcionar como uma animação.

`boolean isVisible()`

Retorna se o componente está visível ou não.

`void jumpTo(int position)`

Força uma animação a saltar para a posição indicada pelo parâmetro.

`void keyPressed(int keyCode)`

Se esse componente estiver destacado, o evento chave pressionado chamará esse método.

`void keyReleased(int keyCode)`

Se esse componente estiver destacado, o evento chave liberado chamará esse método.

`void keyRepeated(int keyCode)`

Se esse componente estiver destacado, o evento chave repetir chamará esse método.

`void longKeyPress(int keyCode)`

Se esse componente estiver destacado, o método é invocado quando o usuário pressionar e mantém a tecla pressionada.

`void paint(Graphics g)`

Esse método pinta o componente na tela, ele deveria ser sobreposto por subclasses para executar desenho personificado ou invocar as APIs de UI para deixar o `PLAF` executar a renderização.

`void paintBackgrounds(Graphics g)`

Esse método pinta todos os planos de fundo dos componentes principais.

`void paintComponent(Graphics g)`

Pinta esse componente como uma raiz indo a todos os componentes principais e definindo a tradução absoluta baseada em coordenadas e status da barra de rolagem.

`void paintComponent(Graphics g, boolean background)`

Pinta esse componente como uma raiz indo a todos os componentes principais e definindo a tradução absoluta baseada em coordenadas e status da barra de rolagem.

`void pointerDragged(int x, int y)`

Se esse componente estiver destacado, o evento indicador arrastado chamará esse método.

`void pointerPressed(int x, int y)`

Se esse componente estiver destacado, o evento indicador pressionado chamará esse método.

`void pointerReleased(int x, int y)`

Se esse componente estiver destacado, o evento indicador liberado chamará esse método.

```
void putClientProperty(String key, Object value)
```

Propriedades do cliente permitem a associação de metadados a um componente; isso é útil para alguns aplicativos que constroem GUIs dinamicamente e precisam rastrear a conexão entre o UI e os dados.

```
void refreshTheme()
```

Certifica-se de que o componente está atualizado com o objeto estilo atual.

```
void removeFocusListener(FocusListener l)
```

Desregistra interesse em receber chamadas de retorno para eventos focusgained.

```
void repaint()
```

Repinta esse componente, a chamada de repintura causa uma chamada de retorno do método pintar na EDT.

```
void repaint(int x, int y, int w, int h)
```

Repinta a área especificada desse componente.

```
void requestFocus()
```

Muda o componente atual para o componente destacado; funcionará apenas para um componente que pertença à forma principal.

```
protected void scrollRectToVisible(Rectangle rect, Component coordinateSpace)
```

Certifica-se de que o componente está visível na barra de rolagem, se esse container for rolável

```
void setAnimationMode(int mode)
```

Define o modo de animação para essa animação.

```
void setCellRenderer(boolean cellRenderer)
```

Usado como uma otimização para marcar que esse componente está correntemente sendo usado como um renderizador de células.

```
void setDelay(int n)
```

Determina o atraso após cada imagem dessa animação ao rodar.

```
void setEnabled(boolean enabled)
```

Indica se o componente está disponível ou indisponível, permitindo, então, prevenir que um componente receba eventos de entrada e o indica visualmente.

```
void setFocus(boolean focused)
```

Essa *flag* não destaca de fato, é um estado que determina quais cores do estilo deveriam ser usadas ao pintar o componente.

```
void setFocusable(boolean focusable)
```

Um definidor simples que determina se esse componente pode ser destacado.

```
void setFocusPainted(boolean focusPainted)
```

Indica se o destaque deveria ser desenhado em volta do componente ou se ele manuseará seu próprio desenho de destaque.

```
void setHandlesInput(boolean handlesInput)
```

Previne que eventos chave sejam tomados para foco transversal.

```
void setHeight(int height)
```

Define a altura do componente; esse método é exposto para fim de gerenciadores de layout externos e não

deveria ser invocado diretamente.

Se um usuário deseja efetuar o tamanho do componente, `setPreferredSize` deveria ser usado.

`protected void setInitialized(boolean initialized)`

Necessário para suportar o conceito de inicialização.

`void setIsScrollVisible(boolean isScrollVisible)`

Define se a barra de rolagem desse componente está visível.

`void setNextFocusDown(Component nextFocusDown)`

Permite determinar qual componente receberá destaque a seguir ao atravessar com a tecla direcional para baixo.

`void setNextFocusLeft(Component nextFocusLeft)`

Permite determinar qual componente receberá destaque a seguir ao atravessar com a tecla direcional esquerda; esse método não afeta o comportamento de destaque geral.

`void setNextFocusRight(Component nextFocusRight)`

Permite determinar qual componente receberá destaque a seguir ao atravessar com a tecla direcional direita.

`void setNextFocusUp(Component nextFocusUp)`

Permite determinar qual componente receberá destaque a seguir ao atravessar com a tecla direcional para cima; esse método não afeta o comportamento de destaque geral.

`void setPreferredSize(Dimension d)`

Define o tamanho preferível do componente, sem garantia de que o componente deve ser dimensionado para seu tamanho preferível.

`void setRepetitionMode(int n)`

Determina quão freqüentemente a animação é repetida, uma vez que for iniciada.

`void setScrollAnimationSpeed(int animationSpeed)`

Coloca em barra de rolagem a velocidade da animação em milissegundos, permitindo que um desenvolvedor diminua ou acelere o modo de animação suave.

`protected void setScrollX(int scrollX)`

Indica a posição X da barra de rolagem; esse número é relativo à posição do componente e, portanto, uma posição 0 indicaria a posição x do componente.

`protected void setScrollY(int scrollY)`

Indica a posição Y da barra de rolagem; esse número é relativo à posição do componente e, portanto, uma posição 0 indicaria a posição y do componente.

`protected void setShouldCalcPreferredSize(boolean shouldCalcPreferredSize)`

Indica que os valores dentro do componente mudaram e o tamanho preferencial deveria ser recalculado

`void setSize(Dimension d)`

Define o tamanho do componente; esse método é exposto para fim de gerenciadores de layout externos e não deveria ser invocado diretamente.
Se um usuário deseja efetuar o tamanho do componente, `setPreferredSize` deveria ser usado.

`void setSmoothScrolling(boolean smoothScrolling)`

Indica que essa rolagem pelo componente deveria funcionar como uma animação.

`void setStyle(Style style)`

Muda o estilo do componente trocando-o por um dado estilo.

`void setUIID(String uiid)`

Define um identificador único para um componente, assim que é restaurado usando `getUIID`.

`void setVisible(boolean visible)`

Muda a visibilidade do componente.

`void setWidth(int width)`

Define a largura do componente; esse método é exposto para fim de gerenciadores de layout externos e não deveria ser invocado diretamente.

Se um usuário deseja efetuar o tamanho do componente, `setPreferredSize` deveria ser usado.

`void setX(int x)`

Define o local x relativo do componente; esse método é exposto para fim de gerenciadores de layout externos e não deveria ser invocado diretamente.

`void setY(int y)`

Define o local y relativo do componente; esse método é exposto para fim de gerenciadores de layout externos e não deveria ser invocado diretamente.

`void start()`

Inicia uma animação.

`void stop()`

Pára uma animação.

`void styleChanged(String propertyName, Style source)`

Invocado para indicar uma mudança em um `propertyName` de um estilo.

`String toString()`

Sobreposto para retornar um valor útil para fins de depuração.

24.11.5 Detalhe dos campos

BRB_CONSTANT_ASCENT

`public static final int BRB_CONSTANT_ASCENT = 1`

Constante de comportamento de redimensionamento de base usado para alinhar componentes apropriadamente. Indica à medida que o tamanho do componente muda que a base permanece a uma distância fixa do topo do componente.

Relaciona-se com:

`getBaselineResizeBehavior()`

BRB_CONSTANT_DESCENT

`public static final int BRB_CONSTANT_DESCENT = 2`

Constante de comportamento de redimensionamento de base usado para alinhar componentes apropriadamente. Indica à medida que o tamanho do componente muda que a base permanece a uma distância fixa do fundo do componente.

Relaciona-se com:

`getBaselineResizeBehavior()`

BRB_CENTER_OFFSET

```
public static final int BRB_CENTER_OFFSET = 3
```

Constante de comportamento de redimensionamento de base usado para alinhar componentes apropriadamente. Indica à medida que o tamanho do componente muda que a base permanece a uma distância fixa do centro do componente.

Relaciona-se com:

```
getBaselineResizeBehavior()
```

BRB_OTHER

```
public static final int BRB_OTHER = 4
```

Constante de comportamento de redimensionamento de base usado para alinhar componentes apropriadamente. Indica à medida que o tamanho do componente muda que a base não pode ser determinada usando uma das outras constantes.

Relaciona-se com:

```
getBaselineResizeBehavior()
```

CENTER

```
public static final int CENTER = 4
```

Indica o centro de alinhamento de um componente.

TOP

```
public static final int TOP = 0
```

A constante de orientação por caixa usada para especificar o topo de uma caixa.

LEFT

```
public static final int LEFT = 1
```

A constante de orientação por caixa usada para especificar o lado esquerdo de uma caixa.

BOTTOM

```
public static final int BOTTOM = 2
```

A constante de orientação por caixa usado para especificar o fundo de uma caixa.

RIGHT

```
public static final int RIGHT = 3
```

A constante de orientação por caixa usada para especificar o lado direito de uma caixa.

24.11.6 Detalhe dos construtores

Component

```
protected Component()
```

Cria uma nova instância de componente.

24.11.7 Detalhe dos métodos

getX

```
public int getX()
```

Retorna o local x do componente atual relativo ao seu container principal.

Retorna:

A coordenada atual de x da origem dos componentes

Relaciona-se com:

`setX(int)`

getY

```
public int getY()
```

Retorna o local y do componente atual relativo ao seu container principal.

Retorna:

A coordenada atual de y da origem dos componentes

Relaciona-se com:

`setY(int)`

isVisible

```
public boolean isVisible()
```

Retorna se o componente está visível ou não.

Retorna:

true se o componente estiver visível; caso contrário *false*

getClientProperty

```
public Object getClientProperty(String key)
```

Propriedades do cliente permitem a associação de metadados a um componente; isso é útil para alguns aplicativos que constroem GUIs dinamicamente e precisam rastrear a conexão entre o UI e os dados.

Parâmetros:

`key` - a chave usada para `putClientProperty`

Retorna:

O valor definido para `putClientProperty` ou `null`, se nenhum valor for definido a propriedade.

putClientProperty

```
public void putClientProperty(String key,
                              Object value)
```

Propriedades do cliente permitem a associação de metadados a um componente; isso é útil para alguns

aplicativos que constroem GUI's dinamicamente e precisam rastrear a conexão entre o UI e os dados. Definir o valor para `null` remove a propriedade de cliente do componente.

Parâmetros:

`key` - chave arbitrária para a propriedade

`value` - o valor designado para uma dada propriedade de cliente

setVisible

```
public void setVisible(boolean visible)
```

Muda a visibilidade do componente.

Parâmetros:

`visible` - true se o componente estiver visível; caso contrário *false*

getWidth

```
public int getWidth()
```

Retorna a largura do componente.

Retorna:

a largura do componente

Relaciona-se com:

`setWidth(int)`

getHeight

```
public int getHeight()
```

Retorna a altura do componente

Retorna:

a altura do componente

Relaciona-se com:

`setHeight(int)`

setX

```
public void setX(int x)
```

Define o local x relativo do componente; esse método é exposto para fim de gerenciadores de layout externos e não deveria ser invocado diretamente.

Parâmetros:

`x` - a coordenada x atual da origem dos componentes

Relaciona-se com:

`getX()`

setY

```
public void setY(int y)
```

Define o local y relativo do componente; esse método é exposto para fim de gerenciadores de layout externos e

não deveria ser invocado diretamente.

Parâmetros:

`y` - a coordenada y atual da origem dos componentes

Relaciona-se com:

`getY()`

getBaseline

```
public int getBaseline(int width,
                       int height)
```

A base para o texto do componente de acordo com o qual deveria ser alinhado com outros componentes para um visual melhor.

Parâmetros:

`width` - a largura do componente

`height` - a altura do componente

Retorna:

Valor de base do topo do componente.

getBaselineResizeBehavior

```
public int getBaselineResizeBehavior()
```

Retorna uma constante indicando como a base varia com o tamanho do componente.

Retorna:

uma das constantes `BRB_CONSTANT_ASCENT`, `BRB_CONSTANT_DESCENT`, `BRB_CENTER_OFFSET` ou `BRB_OTHER`

setPreferredSize

```
public void setPreferredSize(Dimension d)
```

Define o tamanho preferível do componente, sem garantia de que o componente deve ser dimensionado para seu tamanho preferencial. O tamanho final do componente pode ser menor que seu tamanho preferível ou mesmo maior.
O gerenciador de layout pode levar esse valor em consideração, mas não há garantia ou requerimento.

Parâmetros:

`d` - a dimensão do componente

Relaciona-se com:

`getPreferredSize()`

getPreferredSize

```
public Dimension getPreferredSize()
```

Retorna o tamanho preferível do componente, sem garantia de que o componente deve ser dimensionado para seu tamanho preferível. O tamanho final do componente pode ser menor que seu tamanho preferível ou mesmo

maior.
O gerenciador de layout pode levar esse valor em consideração, mas não há garantia ou requerimento.

Retorna:

o tamanho preferível do componente

Relaciona-se com:

`setPreferredSize(com.sun.dtv.lwuit.geom.Dimension)`

calcPreferredSize

`protected Dimension calcPreferredSize()`

Calcula o tamanho preferível baseado no conteúdo do componente. Esse método é invocado vagarosamente por `getPreferredSize`.

Retorna:

O tamanho preferível calculado baseado no conteúdo do componente.

setWidth

`public void setWidth(int width)`

Define a largura do componente; esse método é exposto para fim de gerenciadores de layout externos e não deveria ser invocado diretamente.

Se um usuário deseja efetuar o tamanho do componente, `setPreferredSize` deveria ser usado.

Parâmetros:

`width` - a largura do componente

Relaciona-se com:

`getWidth()`, `setPreferredSize()`

setHeight

`public void setHeight(int height)`

Define a altura do componente; esse método é exposto para fim de gerenciadores de layout externos e não deveria ser invocado diretamente.

Se um usuário deseja efetuar o tamanho do componente, `setPreferredSize` deveria ser usado.

Parâmetros:

`height` - a altura do componente

Relaciona-se com:

`getHeight()`, `setPreferredSize()`

setSize

`public void setSize(Dimension d)`

Define o tamanho do componente; esse método é exposto para fim de gerenciadores de layout externos e não deveria ser invocado diretamente.

Se um usuário deseja efetuar o tamanho do componente, `setPreferredSize` deveria ser usado.

Parâmetros:

`d` - a dimensão do componente

Relaciona-se com:

`setPreferredSize()`

getParent

`public Container getParent()`

Retorna o container no qual esse componente está contido.

Retorna:

o container principal no qual esse componente está contido.

addFocusListener

`public void addFocusListener(FocusListener l)`

Registra interesse em receber chamada de retorno para eventos focusgained; um evento de foco é invocado quando o componente aceitam o foco. Existe um caso especial para o Form que envia destaque para cada seleção dentro do formulário.

Parâmetros:

`l` - interface do *listener* implementando o padrão observável

Relaciona-se com:

`removeFocusListener(com.sun.dtv.lwuit.events.FocusListener)`

removeFocusListener

`public void removeFocusListener(FocusListener l)`

Desregistra interesse em receber chamadas de retorno para eventos focusgained.

Parâmetros:

`l` - interface do *listener* implementando o padrão observável

Relaciona-se com:

`addFocusListener(com.sun.dtv.lwuit.events.FocusListener)`

paintBackgrounds

`public void paintBackgrounds(Graphics g)`

Esse método pinta todos os planos de fundo dos componentes principais.

Parâmetros:

`g` - o objeto gráficos

getAbsoluteX

`public int getAbsoluteX()`

Retorna o local X absoluto baseado na hierarquia do componente; esse método calcula o local exato na tela do componente em vez de um local relativo como retornado por `getX()`.

Retorna:

o local x absoluto do componente

Relaciona-se com:

`getX()`

getAbsoluteY

```
public int getAbsoluteY()
```

Retorna o local Y absoluto baseado na hierarquia do componente; esse método calcula o local exato na tela do componente em vez de um local relativo como retornado por getY().

Retorna:

o local y absoluto do componente

Relaciona-se com:

`getY()`

paintComponent

```
public final void paintComponent(Graphics g)
```

Pinta esse componente como uma raiz indo a todos os componentes principais e definindo a tradução absoluta baseada em coordenadas e status da barra de rolagem. Restaura a tradução quando a pintura é acabada.

Parâmetros:

`g` - os gráficos com os quais pintar esse componente

paintComponent

```
public final void paintComponent(Graphics g,
                                 boolean background)
```

Pinta esse componente como uma raiz indo a todos os componentes principais e definindo a tradução absoluta baseada em coordenadas e status da barra de rolagem. Restaura a tradução quando a pintura é acabada.

Parâmetros:

`g` - os gráficos com os quais pintar esse componente

`background` - se true pintar todos os planos de fundo principais

paint

```
public void paint(Graphics g)
```

Esse método pinta o componente na tela, ele deveria ser sobreposto por subclasses para executar desenho personificado ou invocar as API de UI para deixar o PLAF executar a renderização.

Especificado por:

`paint` na interface `Animation`

Parâmetros:

`g` - os gráficos do componente

isScrollableX

```
public boolean isScrollableX()
```

Indica se o componente deveria/poderia rolar no eixo X.

Retorna:

se o componente é rolável no eixo X

isScrollableY

```
public boolean isScrollableY()
```

Indica se o componente deveria/poderia rolar no eixo Y.

Retorna:

se o componente é rolável no eixo X

getScrollX

```
public int getScrollX()
```

Indica a posição X da barra de rolagem; esse número é relativo à posição do componente e, portanto, uma posição 0 indicaria a posição x do componente.

Retorna:

a posição X da barra de rolagem

getScrollY

```
public int getScrollY()
```

Indica a posição Y da barra de rolagem; esse número é relativo à posição do componente e, portanto, uma posição 0 indicaria a posição x do componente.

Retorna:

a posição Y da barra de rolagem

getBottomGap

```
public int getBottomGap()
```

Retorna o vão a ser deixado na barra de rolagem inferior no eixo X. Esse método é usado por gerenciadores de layout para determinar o espaço que deveriam deixar para a barra de rolagem

Retorna:

o vão a ser deixado na barra de rolagem inferior no eixo X.

getSideGap

```
public int getSideGap()
```

Retorna o vão a ser deixado no barra de rolagem inferior no eixo Y. Esse método é usado por gerenciadores de layout para determinar o espaço que deveriam deixar para a barra de rolagem. (observação: é usado barra de rolagem lateral em vez de barra de rolagem esquerda para uma versão futura que suportará bidi).

Retorna:

o vão a ser deixado na barra de rolagem lateral no eixo Y.

contains

```
public boolean contains(int x,
                        int y)
```

Retorna true se uma dada coordenada absoluta estiver contida no componente.

Parâmetros:

`x` - uma dada coordenada x absoluta

`y` - uma dada coordenada y absoluta

Retorna:

true se uma dada coordenada absoluta estiver contida no componente; caso contrário *false*

getBounds

```
public Rectangle getBounds()
```

Retorna as ligações do componente, o que, às vezes, é mais conveniente do que invocar getX/Y/Width/Height. Ligações são relativas ao container principal. Valores em mudança dentro das ligações podem levar a um comportamento imprevisto.

Retorna:

as ligações do componente

Relaciona-se com:

`getX(),getY()`

isFocusable

```
public boolean isFocusable()
```

Retorna true se esse componente puder receber destaque e estiver disponível.

Retorna:

true se esse componente puder receber destaque; caso contrário *false*

setFocusable

```
public void setFocusable(boolean focusable)
```

Um definidor simples que determina se esse componente pode ser destacado.

Parâmetros:

`focusable` - indica se esse componente pode ser destacado

isFocusPainted

```
public boolean isFocusPainted()
```

Indica se o destaque deveria ser desenhado em volta do componente ou se ele manuseará seu próprio desenho de destaque.

Retorna:

true se o destaque deveria ser desenhado em volta do componente; caso contrário *false*

setFocusPainted

```
public void setFocusPainted(boolean focusPainted)
```

Indica se o destaque deveria ser desenhado em volta do componente ou se ele manuseará seu próprio desenho de destaque.

Parâmetros:

`focusPainted` - indica se o destaque deveria ser desenhado em volta do componente

handlesInput

```
public boolean handlesInput()
```

Previne que as teclas de evento fiquem fixas em foco horizontal.Por exemplo: um componente da lista pode usar as teclas direcionais para navegação interna, portanto, ele mudará essa *flag* para *true* para prevenir que o gerenciador de foco se mova ao próximo componente.

Retorna:

true se eventos chave estiverem sendo usados para foco transversal; caso contrário *false*

setHandlesInput

```
public void setHandlesInput(boolean handlesInput)
```

Previne que as teclas de evento fiquem fixas em foco horizontal.Por exemplo: um componente da lista pode usar as teclas direcionais para navegação interna, portanto, ele mudará essa *flag* para *true* para prevenir que o gerenciador de foco se mova ao próximo componente.

Parâmetros:

`handlesInput` - indica se eventos chaves podem ser tomados para foco transversal

hasFocus

```
public boolean hasFocus()
```

Retorna *true* se o componente tiver foco.

Retorna:

true se o componente estiver focado; caso contrário *false*

Relaciona-se com:

`setFocus()`

setFocus

```
public void setFocus(boolean focused)
```

Essa *flag* não destaca de fato, é um estado que determina quais cores do estilo deveriam ser usadas ao pintar o componente. O foco é determinado pelo Container principal.

Parâmetros:

`focused` - define o estado que determina quais cores do estilo deveriam ser usadas ao pintar um componente focado

Relaciona-se com:

`requestFocus()`

getComponentForm

```
public Form getComponentForm()
```

Retorna o formulário do componente ou `null`, se esse componente não for adicionado a um formulário.

Retorna:

o formulário do componente

repaint

```
public void repaint()
```

Repinta esse componente, a chamada de repintura causa uma chamada de retorno do método pintar na EDT.

repaint

```
public void repaint(int x,
                    int y,
                    int w,
                    int h)
```

Repinta a área especificada desse componente.

Parâmetros:

`x` - a coordenada x do canto superior esquerdo

`y` - a coordenada y do canto superior esquerdo

`w` - a largura da área

`h` - a altura da área

longKeyPress

```
public void longKeyPress(int keyCode)
```

Se esse componente estiver destacado, o método é invocado quando o usuário pressionar e mantém a tecla pressionada.

Parâmetros:

`keyCode` - o valor do código chave a indicar a chave física.

keyPressed

```
public void keyPressed(int keyCode)
```

Se esse componente estiver destacado, o evento chave pressionado chamará esse método.

Parâmetros:

`keyCode` - o valor do código chave a indicar a chave física.

keyReleased

```
public void keyReleased(int keyCode)
```

Se esse componente estiver destacado, o evento chave liberado chamará esse método.

Parâmetros:

`keyCode` - o valor do código chave a indicar a chave física.

keyRepeated

```
public void keyRepeated(int keyCode)
```

Se esse componente estiver destacado, o evento chave repetir chamará esse método. Chama teclas pressionadas/liberadas automaticamente.

Parâmetros:

`keyCode` - o valor do código chave a indicar a chave física.

getScrollAnimationSpeed

```
public int getScrollAnimationSpeed()
```

Coloca em barra de rolagem a velocidade da animação em milissegundos, permitindo que um desenvolvedor diminua ou acelere o modo de animação suave.

Retorna:

Barra de rolagem da velocidade da animação em milissegundos

Relaciona-se com:

`setScrollAnimationSpeed(int)`

setScrollAnimationSpeed

```
public void setScrollAnimationSpeed(int animationSpeed)
```

Coloca em barra de rolagem a velocidade da animação em milissegundos, permitindo que um desenvolvedor diminua ou acelere o modo de animação suave.

Parâmetros:

`animationSpeed` - barra de rolagem da velocidade da animação em milissegundos

Relaciona-se com:

`getScrollAnimationSpeed()`

isSmoothScrolling

```
public boolean isSmoothScrolling()
```

Indica que essa rolagem pelo componente deveria funcionar como uma animação.

Retorna:

se esse componente usa rolagem suave

setSmoothScrolling

```
public void setSmoothScrolling(boolean smoothScrolling)
```

Indica que essa rolagem pelo componente deveria funcionar como uma animação.

Parâmetros:

`smoothScrolling` - indica se um componente usa rolagem suave

pointerDragged

```
public void pointerDragged(int x,
                           int y)
```

Se esse componente estiver destacado, o evento indicador arrastado chamará esse método.

Parâmetros:

`x` - a coordenada do indicador x

`y` - a coordenada do indicador y

pointerPressed

```
public void pointerPressed(int x,
                           int y)
```

Se esse componente estiver destacado, o evento indicador pressionado chamará esse método.

Parâmetros:

`x` - a coordenada do indicador x

`y` - a coordenada do indicador y

pointerReleased

```
public void pointerReleased(int x,
                            int y)
```

Se esse componente estiver destacado, o evento indicador liberado chamará esse método.

Parâmetros:

`x` - a coordenada do indicador x

`y` - a coordenada do indicador y

getStyle

```
public Style getStyle()
```

Retorna o estilo do componente, permitindo manipular o visual do componente.

Retorna:

O objeto estilo do componente

Relaciona-se com:

`setStyle(com.sun.dtv.lwuit.plaf.Style)`

setStyle

```
public void setStyle(Style style)
```

Muda o estilo do componente trocando-o por um dado estilo.

Parâmetros:

`style` - o objeto estilo do componente

Relaciona-se com:

`getStyle()`

requestFocus

```
public void requestFocus()
```

Muda o componente atual para o componente destacado; funcionará apenas para um componente que pertença à forma principal.

toString

`public String toString()`

Sobreposto para retornar um valor útil para fins de depuração. Sobrepõe `Object.toString`.

Substituições:

`toString` na classe `Object`

Retorna:

uma representação *string* desse componente

refreshTheme

`public void refreshTheme()`

Certifica-se de que o componente está atualizado com o objeto estilo atual.

animate

`public boolean animate()`

Descrição copiada da interface: **`Animation`**

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*. É chamado uma vez para cada quadro.

Especificado por:

`animate` na interface `Animation`

Retorna:

true se uma repintura for desejada ou *false* se a repintura não for necessária

setCellRenderer

`public void setCellRenderer(boolean cellRenderer)`

Usado como uma otimização para marcar que esse componente está correntemente sendo usado como um renderizador de células.

Parâmetros:

`cellRenderer` - indica se esse componente está correntemente sendo usado como um renderizador de células

isScrollVisible

`public boolean isScrollVisible()`

Indica se a barra de rolagem desse componente é visível.

Retorna:

true se a barra de rolagem do componente estiver visível; caso contrário *false*

setIsScrollVisible

`public void setIsScrollVisible(boolean isScrollVisible)`

Define se a barra de rolagem desse componente está visível.

Parâmetros:

`isScrollVisible` - indica se a barra de rolagem do componente está visível

styleChanged

```
public void styleChanged(String propertyName,
                         Style source)
```

Descrição copiada da interface: **`StyleListener`**

Invocado para indicar uma mudança em um `propertyName` de um estilo.

Especificado por:

`styleChanged` na interface `StyleListener`

Parâmetros:

`propertyName` - o nome da propriedade que foi mudada

`source` - o objeto estilo mudado

getNextFocusDown

`public Component getNextFocusDown()`

Permite determinar qual componente receberá destaque a seguir ao atravessar com a tecla direcional para baixo.

Retorna:

o componente receberá foco a seguir ao atravessar com a tecla direcional para baixo.

Relaciona-se com:

`setNextFocusDown(com.sun.dtv.lwuit.Component)`

setNextFocusDown

`public void setNextFocusDown(Component nextFocusDown)`

Permite determinar qual componente receberá destaque a seguir ao atravessar com a tecla direcional para baixo.

Parâmetros:

`nextFocusDown` - o componente receberá foco a seguir ao atravessar com a tecla direcional para baixo.

Relaciona-se com:

`getNextFocusDown()`

getNextFocusUp

`public Component getNextFocusUp()`

Permite determinar qual componente receberá destaque a seguir ao atravessar com a tecla direcional para cima.

Retorna:

o componente receberá foco a seguir ao atravessar com a tecla direcional para cima.

Relaciona-se com:

`setNextFocusUp(com.sun.dtv.lwuit.Component)`

setNextFocusUp

`public void setNextFocusUp(Component nextFocusUp)`

Permite determinar qual componente receberá destaque a seguir ao atravessar com a tecla direcional para cima; esse método não afeta o comportamento de destaque geral.

Parâmetros:

`nextFocusUp` - o componente receberá foco a seguir ao atravessar com a tecla direcional para cima.

Relaciona-se com:

`getNextFocusUp()`

getNextFocusLeft

`public Component getNextFocusLeft()`

Permite determinar qual componente receberá destaque a seguir ao atravessar com a tecla direcional esquerda.

Retorna:

o componente receberá foco a seguir ao atravessar com a tecla direcional esquerda.

Relaciona-se com:

`setNextFocusLeft(com.sun.dtv.lwuit.Component)`

setNextFocusLeft

`public void setNextFocusLeft(Component nextFocusLeft)`

Permite determinar qual componente receberá destaque a seguir ao atravessar com a tecla direcional esquerda; esse método não afeta o comportamento de destaque geral.

Parâmetros:

`nextFocusLeft` - o componente receberá foco a seguir ao atravessar com a tecla direcional esquerda.

Relaciona-se com:

`getNextFocusLeft()`

getNextFocusRight

`public Component getNextFocusRight()`

Permite determinar qual componente receberá destaque a seguir ao atravessar com a tecla direcional direita.

Retorna:

o componente receberá foco a seguir ao atravessar com a tecla direcional direita.

Relaciona-se com:

`setNextFocusRight(com.sun.dtv.lwuit.Component)`

setNextFocusRight

`public void setNextFocusRight(Component nextFocusRight)`

Permite determinar qual componente receberá destaque a seguir ao atravessar com a tecla direcional direita.

Parâmetros:

`nextFocusRight` - o componente receberá foco a seguir ao atravessar com a tecla direcional direita.

Relaciona-se com:

`getNextFocusRight()`

isEnabled

```
public boolean isEnabled()
```

Indica se o componente está disponível ou indisponível, permitindo, então, prevenir que um componente receba eventos de entrada e o indica visualmente.

Retorna:

true se esse componente estiver disponível; caso contrário *false*

setEnabled

```
public void setEnabled(boolean enabled)
```

Indica se o componente está disponível ou indisponível, permitindo, então, prevenir que um componente receba eventos de entrada e o indica visualmente.

Parâmetros:

`enabled` - boolean para indicar se o componente deveria estar disponível ou não

start

```
public void start()
```

Inicia uma animação. Considere que uma animação fica em modo parado automaticamente, portanto deve ser iniciada explicitamente uma vez que é criada. No caso da animação já estar rodando, uma chamada para `start()` faz com que ela reinicie.

Especificado por:

`start` na interface `Animated`

stop

```
public void stop()
```

Pára uma animação. No caso da animação já ter parado, uma chamada para `stop()` não tem efeito.

Especificado por:

`stop` na interface `Animated`

isRunning

```
public boolean isRunning()
```

Obtém o modo rodando de uma animação. Retorna *true* se a animação já estiver rodando; caso contrário *false*.

Especificado por:

`isRunning` na interface `Animated`

Retorna:

true se a animação já estiver rodando; caso contrário *false*.

jumpTo

```
public void jumpTo(int position)
```

Força uma animação a saltar para a posição indicada pelo parâmetro. Uma animação pode ser considerada como uma fila de imagens, o parâmetro desse método provê o índice dentro desse fila. Se a animação for parada, uma chamada para esse método faz com que a animação mostre a imagem na posição indicada, mas não que inicie. Se a animação estiver rodando, uma chamada para esse método faz com que a animação salte para a imagem na posição indicada e continue a rodar imediatamente.

Especificado por:

`jumpTo` na interface `Animated`

Parâmetros:

`position` - o índice na fila de imagens fazendo a animação para o qual a animação é forçada a saltar

getPosition

```
public int getPosition()
```

Obtém a posição atual que a animação ocupa no momento da chamada do método. Uma animação pode ser considerada como uma fila de imagens, esse método provê o índice dentro desse fila.

Especificado por:

`getPosition` na interface `Animated`

Retorna:

a posição atual da imagem dentro da fila de imagens fazendo a animação

setRepetitionMode

```
public void setRepetitionMode(int n)
```

Determina quão freqüentemente a animação é repetida, uma vez que for iniciada.

Especificado por:

`setRepetitionMode` na interface `Animated`

Parâmetros:

`n` - número de repetições a ser determinado. Pode ser um número maior que zero ou `LOOP` alternativamente. LOOP significa que a animação roda em um *loop* infinito até que o método `stop()` seja chamado.

Relaciona-se com:

`getRepetitionMode()`

getRepetitionMode

```
public int getRepetitionMode()
```

Retorna o modo de repetição dessa animação na forma de um número. O número lê quantas repetições foram determinadas para essa animação.

Especificado por:

`getRepetitionMode` na interface `Animated`

Retorna:

número de repetições determinado para essa animação ou `LOOP`, o que significa que a animação roda em um *loop* infinito até que o método `stop()` seja chamado.

Relaciona-se com:

`setRepetitionMode(int)`

setDelay

`public void setDelay(int n)`

Determina o atraso após cada imagem dessa animação ao rodar.

Especificado por:

`setDelay` na interface `Animated`

Parâmetros:

`n` - atraso em milissegundos

Relaciona-se com:

`getDelay()`

getDelay

`public int getDelay()`

Obtém o atraso após cada imagem dessa animação ao rodar.

Especificado por:

`getDelay` na interface `Animated`

Retorna:

o atraso em milissegundos

Relaciona-se com:

`setDelay(int)`

setAnimationMode

`public void setAnimationMode(int mode)`

Define o modo de animação para essa animação. O modo de animação determina como a animação está rodando: sempre para frente, ou para frente e para trás alternando.

Especificado por:

`setAnimationMode` na interface `Animated`

Parâmetros:

`mode` - o modo da animação. Valores possíveis são: `REPEATING` and `ALTERNATING`.

Relaciona-se com:

`getAnimationMode()`

getAnimationMode

`public int getAnimationMode()`

Obtém o modo de animação para essa animação. O modo de animação determina como a animação está rodando: sempre para frente, ou para frente e para trás alternando.

Especificado por:

`getAnimationMode` na interface `Animated`

Retorna:

O modo da animação. Valores possíveis são: `REPEATING` and `ALTERNATING`.

Relaciona-se com:

`setAnimationMode(int)`

initialize

```
protected void initialize()
```

Permite que subclasses liguem funcionalidades que se apóiam em estados de componentes totalmente inicializados e "prontos para ação"

deinitialize

```
protected void deinitialize()
```

Invocado para indicar que a inicialização do componente está sendo revertida desde que o componente foi removido da hierarquia do container. Isso permite que o componente desregistre animadores e faça uma limpeza em si mesmo. Esse método é o oposto do método `initialize`.

setInitialized

```
protected void setInitialized(boolean initialized)
```

Necessário para suportar o conceito de inicialização. Ver o método `initialize` para detalhes.

Parâmetros:

`initialized` - estado de inicialização a ser definido

isInitialized

```
protected boolean isInitialized()
```

Indica se o componente está no estado inicializado; um componente está inicializado quando seu método `initialize` foi invocado. Esse método tem de ser invocado antes de mostrar o componente ao usuário.

Retorna:

true se o componente estiver no estado inicializado

getUIID

```
public String getUIID()
```

Restaura um identificador único para um componente, deve ser sobreposto por um componente de forma que um estilo possa ser aplicado ao componente

Retorna:

string único identificando esse componente para a folha de estilo

setUIID

```
public void setUIID(String uiid)
```

Define um identificador único para um componente, assim que é restaurado usando `getUIID`.

Parâmetros:

`uiid` - o `uiid` a ser definido

scrollRectToVisible

```
protected void scrollRectToVisible(Rectangle rect,
                                   Component coordinateSpace)
```

Certifica-se de que o componente está visível na barra de rolagem, se esse container for rolável

Parâmetros:

`rect` - o retângulo que precisa estar visível

`coordinateSpace` - o componente de acordo com as coordenadas do qual o retângulo é definido. O x/y do retângulo são relativos a esse componente (não são absolutos).

setScrollX

```
protected void setScrollX(int scrollX)
```

Indica a posição X da barra de rolagem; esse número é relativo à posição do componente e, portanto, uma posição 0 indicaria a posição x do componente.

Parâmetros:

`scrollX` - a posição X da rolagem

setScrollY

```
protected void setScrollY(int scrollY)
```

Indica a posição Y da barra de rolagem; esse número é relativo à posição do componente e, portanto, uma posição 0 indicaria a posição y do componente.

Parâmetros:

`scrollY` - a posição Y da rolagem

setShouldCalcPreferredSize

```
protected void setShouldCalcPreferredSize(boolean shouldCalcPreferredSize)
```

Indica que os valores dentro do componente mudaram e o tamanho preferencial deveria ser recalculado

Parâmetros:

`shouldCalcPreferredSize` - indica se esse componente necessita recalcular seu tamanho preferencial

24.12 Classe Container

24.12.1 Descrição da classe

`com.sun.dtv.lwuit`

`java.lang.Object`

```
└com.sun.dtv.lwuit.Component
    └com.sun.dtv.lwuit.Container
```

```
public class Container
extends Component
implements MatteEnabled
```

Um padrão composto com `Component`, permite aninhamento e a organização de componentes múltiplos usando uma arquitetura de gerenciador de layout plugável. `Containers` podem ser aninhados uns dentro dos outros para formar interfaces de usuário (UI) elaboradas.

Relaciona-se com:

`Layout`, `Component`

Campos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`BOTTOM, BRB_CENTER_OFFSET, BRB_CONSTANT_ASCENT, BRB_CONSTANT_DESCENT, BRB_OTHER, CENTER, LEFT, RIGHT, TOP`

Campos herdados da interface `com.sun.dtv.ui.`**`Animated`**

`ALTERNATING, LOOP, REPEATING`

Todas as interfaces implementadas

`Animated, Animation, MatteEnabled, StyleListener`

Subclasses diretas conhecidas

`Calendar, DTVContainer, Form, TabbedPane`

24.12.2 Índice de construtores

`Container``()`

Constrói um novo `Container` com um `FlowLayout`.

`Container``(Layout layout)`

Constrói um novo `Container` com um novo gerenciador de *layout*.

24.12.3 Índice de métodos

`void addComponent(Component cmp)`

Adiciona um componente ao *container*.

`void addComponent(int index, Component cmp)`

Esse método adiciona o componente ao local específico do índice no *array* de componentes do *container*.

`void addComponent(Object constraints, Component cmp)`

Adiciona um componente ao *container*.

`boolean contains(Component cmp)`

Retorna *true* se um dado componente estiver dentro da hierarquia desse *container*.

`void doLayout()`

Esquematiza o *container*.

`Component getComponentAt(int index)`

Retorna o componente em um dado índice.

`Component getComponentAt(int x, int y)`

Retorna um componente que existe nas dadas coordenadas x e y ao atravessar objetos do componente e ao invocar *containers*.

`int getComponentCount()`

Retorna o número de componentes.

`int getComponentIndex(Component cmp)`

Retorna o índice do componente no *container*.

`Layout getLayout()`

Retorna o gerenciador de *layout* responsável por organizar esse *container*.

`int getLayoutHeight()`

Retorna a altura para atender a fins do gerenciador de layout; leva em consideração a rolagem, ao contrário do método getWidth.

`int getLayoutWidth()`

Retorna a largura para atender a fins do gerenciador de layout; leva em consideração a rolagem, ao contrário do método getWidth.

`Matte getMatte()`

Retorna o `Matte` atualmente associado ao componente implementando essa interface.

`void invalidate()`

O mesmo que `setShouldCalcPreferredSize(`*`true`*`)`, mas acessível a gerenciadores de layout.

`boolean isScrollableX()`

Indica se o componente deveria/poderia rolar no eixo X.

`boolean isScrollableY()`

Indica se o componente deveria/poderia rolar no eixo Y.

`void layoutContainer()`

Executa o layout do container, se um layout for necessário.

`void paint(Graphics g)`

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

`void pointerPressed(int x, int y)`

Se esse componente estiver destacado, o evento indicador pressionado chamará esse método.

`void refreshTheme()`

Certifica-se de que o componente está atualizado com o objeto estilo atual.

`void removeAll()`

Remove todos os componentes de um container.

`void removeComponent(Component cmp)`

Remove um componente do container.

`void replace(Component current, Component next, Transition t)`

Esse método substitui o componente atual pelo próximo componente.

`void revalidate()`

Re-planifica o container; é útil quando modifica-se a hierarquia do container e é necessário refazer o layout.

`protected void scrollComponentToVisible(Component c)`

Certifica-se de que o componente está visível na barra de rolagem, se esse container for rolável

`void setCellRenderer(boolean cellRenderer)`

Usado como uma otimização para marcar que esse componente está correntemente sendo usado como um renderizador de células.

`void setLayout(Layout layout)`

Define o gerenciador de layout responsável por organizar esse container.

`void setMatte(Matte matte)`

Adiciona um *matte* ao componente implementando essa interface para possibilitar composição de *matte*.

`void setScrollable(boolean scrollable)`

O equivalente de chamar `setScrollableY` e `setScrollableX`.

`void setScrollableX(boolean scrollableX)`

Define se o componente deveria/poderia rolar no eixo X.

`void setScrollableY(boolean scrollableY)`

Define se o componente deveria/poderia rolar no eixo Y.

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`addFocusListener, animate, calcPreferredSize, contains, deinitialize, getAbsoluteX, getAbsoluteY, getAnimationMode, getBaseline, getBaselineResizeBehavior, getBottomGap, getBounds, getClientProperty, getComponentForm, getDelay, getHeight, getNextFocusDown, getNextFocusLeft, getNextFocusRight, getNextFocusUp, getParent, getPosition, getPreferredSize, getRepetitionMode, getScrollAnimationSpeed, getScrollX, getScrollY, getSideGap, getStyle, getUIID, getWidth, getX, getY, handlesInput, hasFocus, initialize, isEnabled, isFocusable, isFocusPainted, isInitialized, isRunning, isScrollVisible, isSmoothScrolling, isVisible, jumpTo, keyPressed, keyReleased, keyRepeated, longKeyPress, paintBackgrounds, paintComponent, paintComponent, pointerDragged, pointerReleased, putClientProperty, removeFocusListener, repaint, repaint, requestFocus, scrollRectToVisible, setAnimationMode, setDelay, setEnabled, setFocus, setFocusable, setFocusPainted, setHandlesInput, setHeight, setInitialized, setIsScrollVisible, setNextFocusDown, setNextFocusLeft, setNextFocusRight, setNextFocusUp, setPreferredSize, setRepetitionMode, setScrollAnimationSpeed, setScrollX, setScrollY, setShouldCalcPreferredSize, setSize, setSmoothScrolling, setStyle, setUIID, setVisible, setWidth, setX, setY, start, stop, styleChanged, toString`

24.12.4 Detalhe dos construtores

Container

`public Container(Layout layout)`

Constrói um novo Container com um novo gerenciador de layout.

Parâmetros:

`layout` - o gerenciador de layout especificado

Container

`public Container()`

Constrói um novo `Container` com um `FlowLayout`.

24.12.5 Detalhe dos métodos

getLayout

`public Layout getLayout()`

Retorna o gerenciador de layout responsável por organizar esse *container*.

Retorna:

o gerenciador de layout do *container*

Relaciona-se com:

`setLayout(com.sun.dtv.lwuit.layouts.Layout)`

setLayout

`public void setLayout(Layout layout)`

Define o gerenciador de layout responsável por organizar esse *container*.

Parâmetros:

`layout` - o gerenciador de layout especificado

Relaciona-se com:

`getLayout()`

invalidate

`public void invalidate()`

O mesmo que `setShouldCalcPreferredSize(true)`, mas acessível a gerenciadores de layout.

getLayoutWidth

`public int getLayoutWidth()`

Retorna a largura para atender a fins do gerenciador de layout; leva em consideração a rolagem, ao contrário do método `getWidth`.

Retorna:

a largura do layout

getLayoutHeight

```
public int getLayoutHeight()
```

Retorna a altura para atender a fins do gerenciador de layout; leva em consideração a rolagem, ao contrário do método `getWidth`.

Retorna:

a altura do layout

addComponent

```
public void addComponent(Component cmp)
```

Adiciona um componente ao *container*.

Parâmetros:

`cmp` - o componente a ser adicionado

addComponent

```
public void addComponent(Object constraints,
                         Component cmp)
```

Adiciona um componente ao *container*.

Parâmetros:

`constraints` - esse método é útil quando o *layout* requer uma limitação como o `BorderLayout`. Nesse caso é preciso especificar um dado adicional quando se adiciona um componente, como `"CENTER", "NORTH"`...

`cmp` - componente a adicionar

addComponent

```
public void addComponent(int index,
                         Component cmp)
```

Esse método adiciona o componente ao local específico do índice no *array* de componentes do *container*.

Parâmetros:

`index` - local onde inserir o componente

`cmp` - o componente a ser adicionado

Lança:

`ArrayIndexOutOfBoundsException` - se o índice estiver fora dos limites

`IllegalArgumentException` - se o componente já estiver contido ou o `cmp` for um `Form Component`

replace

```
public void replace(Component current,
                    Component next,
                    Transition t)
```

Esse método substitui o componente atual pelo próximo componente. O componente atual deve estar contido

nesse *container*. Esse método retorna imediatamente.

Parâmetros:

`current` - um componente a ser removido do *container*

`next` - um componente que substitui o componente atual

`t` - uma transição entre o adicionamento e remoção de componentes; uma transição pode ser `null`

removeComponent

```
public void removeComponent(Component cmp)
```

remove um componente do *container*.

Parâmetros:

`cmp` - o componente removido

removeAll

```
public void removeAll()
```

remove todos os componentes de um *container*.

revalidate

```
public void revalidate()
```

Re-planifica o container; é útil quando modifica-se a hierarquia do *container* e é necessário refazer o layout.

paint

```
public void paint(Graphics g)
```

Descrição copiada da interface: **`Animation`**

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

Especificado por:

`paint` na interface `Animation`

Substituições:

`paint` na classe `Component`

Parâmetros:

`g` - os gráficos do componente

layoutContainer

```
public void layoutContainer()
```

Executa o *layout* do *container*, se um *layout* for necessário.

doLayout

```
public void doLayout()
```

Esquematiza o *container*.

getComponentCount

`public int getComponentCount()`

Retorna o número de componentes.

Retorna:

a contagem de componentes

getComponentAt

`public Component getComponentAt(int index)`

Retorna o componente em um dado índice.

Parâmetros:

`index` - do componente que se deseja obter

Retorna:

um componente

Lança:

`ArrayIndexOutOfBoundsException` - se um índice inválido foi dado.

getComponentIndex

`public int getComponentIndex(Component cmp)`

Retorna o índice do componente no *container*.

Parâmetros:

`cmp` - o componente a ser buscado

Retorna:

o índice do componente no *container* ou -1 se não for encontrado

contains

`public boolean contains(Component cmp)`

Retorna *true* se um dado componente estiver dentro da hierarquia desse *container*.

Parâmetros:

`cmp` - um componente a ser verificado

Retorna:

true se esse componente estiver contido nesse *container.*

getComponentAt

```
public Component getComponentAt(int x,
                                int y)
```

Retorna um componente que existe nas dadas coordenadas x e y ao atravessar objetos do componente e ao

invocar containers.

Parâmetros:

`x` - localização absoluta na tela

`y` - localização absoluta na tela

Retorna:

um componente, se encontrado; caso contrário `null`

Relaciona-se com:

`Component.contains()`

pointerPressed

```
public void pointerPressed(int x,
                           int y)
```

Se esse componente estiver destacado, o evento indicador pressionado chamará esse método.

Substituições:

`pointerPressed` na classe `Component`

Parâmetros:

`x` - a coordenada do indicador x

`y` - a coordenada do indicador y

refreshTheme

```
public void refreshTheme()
```

Certifica-se de que o componente está atualizado com o objeto estilo atual.

Substituições:

`refreshTheme` na classe `Component`

isScrollableX

```
public boolean isScrollableX()
```

Indica se o componente deveria/poderia rolar no eixo X.

Substituições:

`isScrollableX` na classe `Component`

Retorna:

se o componente é rolável no eixo X

isScrollableY

```
public boolean isScrollableY()
```

Indica se o componente deveria/poderia rolar no eixo Y.

Substituições:

`isScrollableY` na classe `Component`

Retorna:

se o componente é rolável no eixo X

setScrollableX

`public void setScrollableX(boolean scrollableX)`

Define se o componente deveria/poderia rolar no eixo X.

Parâmetros:

`scrollableX` - se o componente deveria/poderia rolar no eixo X

setScrollableY

`public void setScrollableY(boolean scrollableY)`

Define se o componente deveria/poderia rolar no eixo Y.

Parâmetros:

`scrollableY` - se o componente deveria/poderia rolar no eixo Y

setScrollable

`public void setScrollable(boolean scrollable)`

O equivalente de chamar `setScrollableY` e `setScrollableX`.

Parâmetros:

`scrollable` - se o componente deveria/poderia rolar no eixo X e Y

setCellRenderer

`public void setCellRenderer(boolean cellRenderer)`

Usado como uma otimização para marcar que esse componente está correntemente sendo usado como um renderizador de células.

Substituições:

`setCellRenderer` na classe `Component`

Parâmetros:

`cellRenderer` - indica se esse componente está correntemente sendo usado como um renderizador de células

setMatte

```
public void setMatte(Matte matte)
              throws MatteException
```

Descrição copiada da interface: **`MatteEnabled`**

Adiciona um `Matte` ao componente implementando essa interface para possibilitar composição de `matte`. Se já houver um `Matte` designado para esse componente e esse `Matte` for animado, ele tem de ser parado antes de qualquer chamada para esse método.

Especificado por:

`setMatte` na interface `MatteEnabled`

Parâmetros:

matte - o Matte a ser designado ao componente. Todo o tempo, só pode haver um matte associado ao componente, devido a isso qualquer matte associado anteriormente deve ser sobreposto por uma chamada para esse método. O parâmetro Matte pode também ser null, nesse caso não há matte associado ao componente após a chamada, mesmo se houvesse um antes.

Lança:

MatteException - se o Matte tem um tipo não-suportado, a plataforma não suporta mattes em nenhuma circunstância ou um matte animado está associado ao componente e ainda está rodando

getMatte

```
public Matte getMatte()
```

Descrição copiada da interface: **MatteEnabled**

Retorna o Matte atualmente associado ao componente implementando essa interface.

Especificado por:

getMatte na interface MatteEnabled

Retorna:

o Matte atualmente associado ao componente ou null se não houver nenhum

scrollComponentToVisible

```
protected void scrollComponentToVisible(Component c)
```

Certifica-se de que o componente está visível na barra de rolagem, se esse container for rolável

Parâmetros:

c - o componente que estará rolando para obter visibilidade

24.13 Classe Dialog

24.13.1 Descrição da classe

com.sun.dtv.lwuit

```
java.lang.Object
  └com.sun.dtv.lwuit.Component
      └com.sun.dtv.lwuit.Container
          └com.sun.dtv.lwuit.Form
              └com.sun.dtv.lwuit.Dialog
```

Todas as interfaces implementadas

```
Animated, Animation, MatteEnabled, StyleListener
```

```
public class Dialog
extends Form
```

Um diálogo é um formulário que ocupa parte da tela e aparece como uma entidade modal ao desenvolvedor. Dialogs permitem prontificar usuários para informações e confiar na informação sendo disponibilizada na próxima

linha após o método de exibição.

Modalidade indica que um diálogo bloqueará a *thread* chamadora, mesmo se essa *thread* for a EDT. Observar que um diálogo não liberará o bloqueio até que o `dispose` seja chamado, mesmo se um `show()` de outro formulário for chamado.

Para determinar o tamanho do diálogo, use o método de exibição que aceita quatro valores inteiros; observar que esses valores aceitam margem dos quatro lados, em vez de valores x, y, largura e altura.

Para estilizar o diálogo, seria desejado estilizar o painel de conteúdo em vez do próprio diálogo.

24.13.2 Índice de campos

`static int` **`TYPE_ALARM`**

Constante indicando o tipo de alerta para indicar o som a ser tocado ou ícone, se nenhum for definido explicitamente.

`static int` **`TYPE_CONFIRMATION`**

Constante indicando o tipo de alerta para indicar o som a ser tocado ou ícone, se nenhum for definido explicitamente.

`static int` **`TYPE_ERROR`**

Constante indicando o tipo de alerta para indicar o som a ser tocado ou ícone, se nenhum for definido explicitamente.

`static int` **`TYPE_INFO`**

Constante indicando o tipo de alerta para indicar o som a ser tocado ou ícone, se nenhum for definido explicitamente.

`static int` **`TYPE_WARNING`**

Constante indicando o tipo de alerta para indicar o som a ser tocado ou ícone, se nenhum for definido explicitamente.

Campos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`BOTTOM, BRB_CENTER_OFFSET, BRB_CONSTANT_ASCENT, BRB_CONSTANT_DESCENT, BRB_OTHER, CENTER, LEFT, RIGHT, TOP`

Campos herdados da interface `com.sun.dtv.ui.`**`Animated`**

`ALTERNATING, LOOP, REPEATING`

24.13.3 Índice de construtores

`Dialog``()`

Constrói um diálogo com um título.

`Dialog``(String title)`

Constrói um diálogo com um título.

24.13.4 Índice de métodos

`boolean animate()`

Permite a animação reduzir chamadas para "repintura", quando retorna *false*.

```
void dispose()
```

Fecha o formulário atual e retorne ao formulário anterior, liberando a EDT no processo.

```
static String getDefaultDialogPosition()
```

Posição de orientação de tela padrão para o próximo diálogo.

```
Style getDialogStyle()
```

Coletor simples que obtém o estilo do diálogo.

```
boolean isAutoDispose()
```

Determina se a execução de um comando nesse diálogo dispõe implicitamente o diálogo.

```
void refreshTheme()
```

Certifica-se de que o componente está atualizado com o objeto estilo atual.

```
void setAutoDispose(boolean autoDispose)
```

Determina se a execução de um comando nesse diálogo dispõe implicitamente o diálogo.

```
static void setDefaultDialogPosition(String p)
```

Posição de orientação de tela padrão para o próximo diálogo.

```
void setDialogStyle(Style style)
```

Definidor simples para definir o estilo do diálogo.

```
void setTimeout(long time)
```

Indica o tempo (em milissegundos) após o qual o diálogo deve ser disposto implicitamente.

```
void show()
```

A versão padrão de exibição modal exibe o diálogo ocupando a porção central da tela.

```
Command show(int top, int bottom, int left, int right, boolean includeTitle)
```

Esse método exibe o formulário como um alerta modal permitindo produzir um comportamento de caixa de alerta/diálogo.

```
Command show(int top, int bottom, int left, int right, boolean includeTitle, boolean
modal)
```

Esse método exibe o formulário como um alerta modal permitindo produzir um comportamento de caixa de alerta/diálogo.

```
static Command show(String title, Component body, Command defaultCommand, Command[]
cmds, int type, Image icon, long timeout, Transition transition)
```

Exibe um diálogo modal com um dado componente como seu “corpo” posicionado no centro.

```
static Command show(String title, Component body, Command[] cmds)
```

Exibe um diálogo modal com um dado componente como seu “corpo” posicionado no centro.

```
static Command show(String title, Component body, Command[] cmds, int type, Image
icon)
```

Exibe um diálogo modal com um dado componente como seu “corpo” posicionado no centro.

```
static Command show(String title, Component body, Command[] cmds, int type, Image
icon, long timeout)
```

Exibe um diálogo modal com um dado componente como seu “corpo” posicionado no centro.

```
static Command show(String title, Component body, Command[] cmds, int type, Image
icon, long timeout, Transition transition)
```

Exibe um diálogo modal com um dado componente como seu “corpo” posicionado no centro.

```
static Command show(String title, String text, Command defaultCommand, Command[] cmds, int type, Image icon, long timeout)
```

Exibe um diálogo de prontificação modal com um dado título e texto.

```
static Command show(String title, String text, Command defaultCommand, Command[] cmds, int type, Image icon, long timeout, Transition transition)
```

Exibe um diálogo de prontificação modal com um dado título e texto.

```
static Command show(String title, String text, Command[] cmds, int type, Image icon, long timeout)
```

Exibe um diálogo de prontificação modal com um dado título e texto.

```
static Command show(String title, String text, Command[] cmds, int type, Image icon, long timeout, Transition transition)
```

Exibe um diálogo de prontificação modal com um dado título e texto.

```
static boolean show(String title, String text, int type, Image icon, String okText, String cancelText)
```

Exibe um diálogo de prontificação modal com um dado título e texto.

```
static boolean show(String title, String text, int type, Image icon, String okText, String cancelText, long timeout)
```

Exibe um diálogo de prontificação modal com um dado título e texto.

```
static boolean show(String title, String text, String okText, String cancelText)
```

Exibe um diálogo de prontificação modal com um dado título e texto.

```
Command showDialog()
```

Exibe um diálogo modal e retorna o comando pressionado dentro do diálogo modal.

```
void showModeless()
```

Exibe um diálogo não-modal que é útil para alguns casos mais simples tais como indicação de progresso e etc.

```
Command showPacked(String position, boolean modal)
```

Método de conveniência para exibir um diálogo dimensionado para corresponder ao seu conteúdo.

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Form`**

```
addCommand, addCommand, addComponent, addComponent, addComponent, addGameKeyListener, addKeyListener, deregisterAnimation, getBackCommand, getClearCommand, getCommand, getCommandCount, getComponentForm, getContentPane, getDefaultCommand, getFocused, getGlassPane, getScrollAnimationSpeed, getSoftButton, getSoftButtonCount, getSoftButtonStyle, getTintColor, getTitle, getTitleComponent, getTitleStyle, getTransitionInAnimator, getTransitionOutAnimator, hideNotify, isSingleFocusMode, isSmoothScrolling, keyPressed, keyReleased, keyRepeated, longKeyPress, onShow, paint, paintBackground, pointerDragged, pointerPressed, pointerReleased, registerAnimation, removeAll, removeAllCommands, removeCommand, removeComponent, removeGameKeyListener, removeKeyListener, replace, scrollComponentToVisible, setBackCommand, setBgImage, setClearCommand, setCommandListener, setDefaultCommand, setFocused, setGlassPane, setLayout, setMenuCellRenderer, setMenuTransitions, setScrollable, setScrollableX, setScrollableY, setScrollAnimationSpeed, setSmoothScrolling, setSoftButtonStyle, setTintColor, setTitle, setTitleStyle, setTransitionInAnimator, setTransitionOutAnimator, setVisible, showMenuDialog, showNotify, sizeChanged
```

Métodos herdados da classe `com.sun.dtv.lwuit.Container`

`contains, doLayout, getComponentAt, getComponentAt, getComponentCount, getComponentIndex, getLayout, getLayoutHeight, getLayoutWidth, getMatte, invalidate, isScrollableX, isScrollableY, layoutContainer, revalidate, setCellRenderer, setMatte`

Métodos herdados da classe `com.sun.dtv.lwuit.Component`

`addFocusListener, calcPreferredSize, contains, deinitialize, getAbsoluteX, getAbsoluteY, getAnimationMode, getBaseline, getBaselineResizeBehavior, getBottomGap, getBounds, getClientProperty, getDelay, getHeight, getNextFocusDown, getNextFocusLeft, getNextFocusRight, getNextFocusUp, getParent, getPosition, getPreferredSize, getRepetitionMode, getScrollX, getScrollY, getSideGap, getStyle, getUIID, getWidth, getX, getY, handlesInput, hasFocus, initialize, isEnabled, isFocusable, isFocusPainted, isInitialized, isRunning, isScrollVisible, isVisible, jumpTo, paintBackgrounds, paintComponent, paintComponent, putClientProperty, removeFocusListener, repaint, repaint, requestFocus, scrollRectToVisible, setAnimationMode, setDelay, setEnabled, setFocus, setFocusable, setFocusPainted, setHandlesInput, setHeight, setInitialized, setIsScrollVisible, setNextFocusDown, setNextFocusLeft, setNextFocusRight, setNextFocusUp, setPreferredSize, setRepetitionMode, setScrollX, setScrollY, setShouldCalcPreferredSize, setSize, setStyle, setUIID, setWidth, setX, setY, start, stop, styleChanged, toString`

24.13.5 Detalhe dos campos

TYPE_ALARM

```
public static final int TYPE_ALARM = 1
```

Constante indicando o tipo de alerta para indicar o som a ser tocado ou ícone, se nenhum for definido explicitamente.

TYPE_CONFIRMATION

```
public static final int TYPE_CONFIRMATION = 2
```

Constante indicando o tipo de alerta para indicar o som a ser tocado ou ícone, se nenhum for definido explicitamente.

TYPE_ERROR

```
public static final int TYPE_ERROR = 3
```

Constante indicando o tipo de alerta para indicar o som a ser tocado ou ícone, se nenhum for definido explicitamente.

TYPE_INFO

```
public static final int TYPE_INFO = 4
```

Constante indicando o tipo de alerta para indicar o som a ser tocado ou ícone, se nenhum for definido explicitamente.

TYPE_WARNING

```
public static final int TYPE_WARNING = 5
```

Constante indicando o tipo de alerta para indicar o som a ser tocado ou ícone, se nenhum for definido explicitamente.

24.13.6 Detalhe dos construtores

Dialog

```
public Dialog(String title)
```

Constrói um `Dialog` com um título.

Parâmetros:

`title` - o título do diálogo

Dialog

```
public Dialog()
```

Constrói um `Dialog` com um título.

24.13.7 Detalhe dos métodos

setDialogStyle

```
public void setDialogStyle(Style style)
```

Definidor simples para definir o estilo do diálogo. Existe um diferença significativa entre o propósito desse método e o método `setStyle` herdado de `Component`. Ao passo em que `setStyle` influencia o estilo de toda a tela, esse método consulta o estilo painel de conteúdo, onde todos os elementos do diálogo estão visíveis.

Parâmetros:

`style` - o estilo

Relaciona-se com:

`getDialogStyle()`, `Component.setStyle(com.sun.dtv.lwuit.plaf.Style)`

getDialogStyle

```
public Style getDialogStyle()
```

Coletor simples que obtém o estilo do diálogo. Existe um diferença significativa entre o propósito desse método e o método `getStyle` herdado de `Component`. Ao passo em que `getStyle` retorna o estilo de toda a tela, esse método referencia o estilo do painel de conteúdo, onde todos os elementos do diálogo estão visíveis.

Retorna:

o estilo

Relaciona-se com:

`setDialogStyle(com.sun.dtv.lwuit.plaf.Style)`, `Component.getStyle()`

show

```
public Command show(int top,
                    int bottom,
                    int left,
```

```
                    int right,
                    boolean includeTitle)
```

Esse método exibe o formulário como um alerta modal permitindo produzir um comportamento de caixa de alerta/diálogo. Esse método irá chamar a *thread* chamador, mesmo que essa *thread* seja a EDT. Observar que esse método não liberará o bloqueio até que seja chamado dispor, mesmo se um `show()` de outro formulário for chamado.

Diálogos modais permitem os formulários de “conteúdo” a “pairar no ar", o que é especialmente útil para diálogos dos quais se desejaria que o formulário subjacente "olhasse" por trás do formulário.

Parâmetros:

`top` - espaço em pixels entre o topo da tela e o formulário

`bottom` - espaço em pixels entre a parte inferior da tela e o formulário

`left` - espaço em pixels entre o lado esquerdo da tela e o formulário

`right` - espaço em pixels entre o lado direito da tela e o formulário

`includeTitle` - se o título deveria pender no topo da tela ou ser colado ao painel de conteúdo

Retorna:

o último comando pressionado pelo usuário, se tal comando existir.

show

```
public Command show(int top,
                    int bottom,
                    int left,
                    int right,
                    boolean includeTitle,
                    boolean modal)
```

Esse método exibe o formulário como um alerta modal permitindo produzir um comportamento de caixa de alerta/diálogo. Esse método irá chamar a *thread* chamadora, mesmo que essa *thread* seja a EDT. Observar que esse método não liberará o bloqueio até que seja chamado `dispose()`, mesmo se um `show()` de outro formulário for chamado.

Diálogos modais permitem os formulários de “conteúdo” a “pairar no ar", o que é especialmente útil para diálogos dos quais se desejaria que o formulário subjacente "olhasse" por trás do formulário.

Parâmetros:

`top` - espaço em pixels entre o topo da tela e o formulário

`bottom` - espaço em pixels entre a parte inferior da tela e o formulário

`left` - espaço em pixels entre o lado esquerdo da tela e o formulário

`right` -espaço em pixels entre o lado direito da tela e o formulário

`includeTitle` - se o título deveria pender no topo da tela ou ser colado ao painel de conteúdo

`modal` - indica que o diálogo deveria ser modal definido como falso para diálogo não-modal, que é útil para alguns casos

Retorna:

o último comando pressionado pelo usuário, se tal comando existir.

setTimeout

```
public void setTimeout(long time)
```

Indica o tempo (em milissegundos) após o qual o diálogo deve ser disposto implicitamente.

Parâmetros:

`time` - um tempo em milissegundos usado para dispor o diálogo

show

```
public static boolean show(String title,
                           String text,
                           int type,
                           Image icon,
                           String okText,
                           String cancelText)
```

Exibe um diálogo de prontificação modal com um dado título e texto.

Parâmetros:

`title` - O título para o diálogo, opcionalmente `null`;

`text` - o texto exibido no diálogo

`type` - o tipo do alerta, um dos `TYPE_WARNING`, `TYPE_INFO`, `TYPE_ERROR`, `TYPE_CONFIRMATION` ou `TYPE_ALARM`

`icon` - o ícone para o diálogo; pode ser `null`

`okText` - o texto que aparecerá no comando dispensando o diálogo

`cancelText` - opcionalmente `null` para um texto que aparecerá no comando cancelar para cancelar o diálogo

Retorna:

`True` se o comando ok for pressionado ou se `cancelText` for `null`. `False` caso contrário.

show

```
public static boolean show(String title,
                           String text,
                           int type,
                           Image icon,
                           String okText,
                           String cancelText,
                           long timeout)
```

Exibe um diálogo de prontificação modal com um dado título e texto.

Parâmetros:

`title` - O título para o diálogo, opcionalmente `null`;

`text` - o texto exibido no diálogo

`type` - o tipo do alerta, um dos `TYPE_WARNING`, `TYPE_INFO`, `TYPE_ERROR`, `TYPE_CONFIRMATION` ou `TYPE_ALARM`

`icon` - o ícone para o diálogo; pode ser `null`

`okText` - o texto que aparecerá no comando dispensando o diálogo

`cancelText` - opcionalmente `null` para um texto que aparecerá no comando cancelar para cancelar o diálogo

`timeout` - um tempo de espera após o qual `null` seria retornado; se o tempo de espera for 0, é usado tempo infinito

Retorna:

true se o comando ok for pressionado ou se cancelText for `null`. False caso contrário.

show

```
public static Command show(String title,
                           String text,
                           Command[] cmds,
                           int type,
                           Image icon,
                           long timeout)
```

Exibe um diálogo de prontificação modal com um dado título e texto.

Parâmetros:

`title` - O título para o diálogo, opcionalmente `null`;

`text` - o texto exibido no diálogo

`cmds` - comandos que são adicionados ao formulário; qualquer clique em qualquer comando irá dispor o formulário

`type` - o tipo do alerta, um dos TYPE_WARNING, TYPE_INFO, TYPE_ERROR, TYPE_CONFIRMATION ou TYPE_ALARM

`icon` - o ícone para o diálogo; pode ser `null`

`timeout` - um tempo de espera após o qual `null` seria retornado; se o tempo de espera for 0, é usado tempo infinito

Retorna:

o comando pressionado pelo usuário

show

```
public static Command show(String title,
                           String text,
                           Command defaultCommand,
                           Command[] cmds,
                           int type,
                           Image icon,
                           long timeout)
```

Exibe um diálogo de prontificação modal com um dado título e texto.

Parâmetros:

`title` - O título para o diálogo, opcionalmente `null`;

`text` - o texto exibido no diálogo

`defaultCommand` - comando a ser designado como comando padrão ou `null`

`cmds` - comandos que são adicionados ao formulário; qualquer clique em qualquer comando irá dispor o formulário

`type` - o tipo do alerta, um dos `TYPE_WARNING`, `TYPE_INFO`, `TYPE_ERROR`, `TYPE_CONFIRMATION` ou `TYPE_ALARM`

`icon` - o ícone para o diálogo; pode ser `null`

`timeout` - um tempo de espera após o qual `null` seria retornado; se o tempo de espera for 0, é usado tempo infinito

Retorna:

o comando pressionado pelo usuário

show

```
public static Command show(String title,
                           String text,
                           Command[] cmds,
                           int type,
                           Image icon,
                           long timeout,
                           Transition transition)
```

Exibe um diálogo de prontificação modal com um dado título e texto.

Parâmetros:

`title` - O título para o diálogo, opcionalmente `null`;

`text` - o texto exibido no diálogo

`cmds` - comandos que são adicionados ao formulário; qualquer clique em qualquer comando irá dispor o formulário

`type` - o tipo do alerta, um dos `TYPE_WARNING`, `TYPE_INFO`, `TYPE_ERROR`, `TYPE_CONFIRMATION` ou `TYPE_ALARM`

`icon` - o ícone para o diálogo; pode ser `null`

`timeout` - um tempo de espera após o qual `null` seria retornado; se o tempo de espera for 0, é usado tempo infinito

`transition` - a transição instalada quando o diálogo entra/sai

Retorna:

o comando pressionado pelo usuário

show

```
public static Command show(String title,
                           String text,
                           Command defaultCommand,
                           Command[] cmds,
                           int type,
                           Image icon,
                           long timeout,
                           Transition transition)
```

Exibe um diálogo de prontificação modal com um dado título e texto.

Parâmetros:

`title` - O título para o diálogo, opcionalmente `null`;

`text` - o texto exibido no diálogo

`defaultCommand` - comando a ser designado como comando padrão ou `null`

`cmds` - comandos que são adicionados ao formulário; qualquer clique em qualquer comando irá dispor o formulário

`type` - o tipo do alerta, um dos `TYPE_WARNING`, `TYPE_INFO`, `TYPE_ERROR`, `TYPE_CONFIRMATION` ou `TYPE_ALARM`

`icon` - o ícone para o diálogo; pode ser `null`

`timeout` - um tempo de espera após o qual `null` seria retornado; se o tempo de espera for 0, é usado tempo

infinito

`transition` - a transição instalada quando o diálogo entra/sai

Retorna:

o comando pressionado pelo usuário

show

```
public static boolean show(String title,
                           String text,
                           String okText,
                           String cancelText)
```

Exibe um diálogo de prontificação modal com um dado título e texto.

Parâmetros:

`title` - O título para o diálogo, opcionalmente `null`;

`text` - o texto exibido no diálogo

`okText` - o texto que aparecerá no comando dispensando o diálogo

`cancelText` - opcionalmente `null` para um texto que aparecerá no comando cancelar para cancelar o diálogo

Retorna:

True se o comando *ok* for pressionado ou se `cancelText` for `null`. *False* caso contrário.

show

```
public static Command show(String title,
                           Component body,
                           Command[] cmds)
```

Exibe um diálogo modal com um dado componente como seu “corpo” posicionado no centro.

Parâmetros:

`title` - título para o diálogo

`body` - componente posicionado no centro do diálogo

`cmds` - comandos que são adicionados ao formulário; qualquer clique em qualquer comando irá dispor o formulário

Retorna:

o comando pressionado pelo usuário

show

```
public static Command show(String title,
                           Component body,
                           Command[] cmds,
                           int type,
                           Image icon)
```

Exibe um diálogo modal com um dado componente como seu “corpo” posicionado no centro.

Parâmetros:

`title` - título para o diálogo

`body` - componente posicionado no centro do diálogo

`cmds` - comandos que são adicionados ao formulário; qualquer clique em qualquer comando irá dispor o formulário

type - o tipo do alerta, um dos TYPE_WARNING, TYPE_INFO, TYPE_ERROR, TYPE_CONFIRMATION ou TYPE_ALARM

icon - o ícone para o diálogo; pode ser null

Retorna:

o comando pressionado pelo usuário

show

```
public static Command show(String title,
                           Component body,
                           Command[] cmds,
                           int type,
                           Image icon,
                           long timeout)
```

Exibe um diálogo modal com um dado componente como seu “corpo” posicionado no centro.

Parâmetros:

title - título para o diálogo

body - componente posicionado no centro do diálogo

cmds - comandos que são adicionados ao formulário; qualquer clique em qualquer comando irá dispor o formulário

type - o tipo do alerta, um dos TYPE_WARNING, TYPE_INFO, TYPE_ERROR, TYPE_CONFIRMATION ou TYPE_ALARM

icon - o ícone para o diálogo; pode ser null

timeout - um tempo de espera após o qual null seria retornado; se o tempo de espera for 0, é usado tempo infinito

Retorna:

o comando pressionado pelo usuário

show

```
public static Command show(String title,
                           Component body,
                           Command[] cmds,
                           int type,
                           Image icon,
                           long timeout,
                           Transition transition)
```

Exibe um diálogo modal com um dado componente como seu “corpo” posicionado no centro.

Parâmetros:

title - título para o diálogo

body - componente posicionado no centro do diálogo

cmds - comandos que são adicionados ao formulário; qualquer clique em qualquer comando irá dispor o formulário

type - o tipo do alerta, um dos TYPE_WARNING, TYPE_INFO, TYPE_ERROR, TYPE_CONFIRMATION ou TYPE_ALARM

icon - o ícone para o diálogo; pode ser null

`timeout` - um tempo de espera após o qual `null` seria retornado; se o tempo de espera for 0, é usado tempo infinito

`transition` - a transição instalada quando o diálogo entra/sai

Retorna:

o comando pressionado pelo usuário

show

```
public static Command show(String title,
                           Component body,
                           Command defaultCommand,
                           Command[] cmds,
                           int type,
                           Image icon,
                           long timeout,
                           Transition transition)
```

Exibe um diálogo modal com um dado componente como seu “corpo” posicionado no centro.

Parâmetros:

`title` - título para o diálogo

`body` - componente posicionado no centro do diálogo

`defaultCommand` - comando a ser designado como comando padrão ou `null`

`cmds` - comandos que são adicionados ao formulário; qualquer clique em qualquer comando irá dispor o formulário

`type` - o tipo do alerta, um dos `TYPE_WARNING`, `TYPE_INFO`, `TYPE_ERROR`, `TYPE_CONFIRMATION` ou `TYPE_ALARM`

`icon` - o ícone para o diálogo; pode ser `null`

`timeout` - um tempo de espera após o qual `null` seria retornado; se o tempo de espera for 0, é usado tempo infinito

`transition` - a transição instalada quando o diálogo entra/sai

Retorna:

o comando pressionado pelo usuário

show

```
public void show()
```

A versão padrão de exibição modal exibe o diálogo ocupando a porção central da tela.

Substituições:

`show` na classe `Form`

showModeless

```
public void showModeless()
```

Exibe um diálogo não-modal que é útil para alguns casos mais simples tais como indicação de progresso e etc.

showPacked

```
public Command showPacked(String position,
                          boolean modal)
```

Método de conveniência para exibir um diálogo dimensionado para corresponder ao seu conteúdo.

Parâmetros:

`position` - um dos valores provenientes da classe `BorderLayout`, por exemplo: `BorderLayout.CENTER`, `BorderLayout.NORTH` etc.

`modal` - se o diálogo deveria ser modal

Retorna:

o comando selecionado se o diálogo for modal e disposto por meio de um comando

dispose

```
public void dispose()
```

Fecha o formulário atual e retorne ao formulário anterior, liberando a EDT no processo.

refreshTheme

```
public void refreshTheme()
```

Certifica-se de que o componente está atualizado com o objeto estilo atual.

Substituições:

`refreshTheme` na classe `Form`

showDialog

```
public Command showDialog()
```

Exibe um diálogo modal e retorna o comando pressionado dentro do diálogo modal.

Retorna:

último comando pressionado no diálogo modal

animate

```
public boolean animate()
```

Descrição copiada da interface: **`Animation`**

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*. É chamado uma vez para cada quadro.

Especificado por:

`animate` na interface `Animation`

Substituições:

`animate` na classe `Component`

Retorna:

true se uma repintura for desejada ou *false* se a repintura não for necessária

isAutoDispose

`public boolean isAutoDispose()`

Determina se a execução de um comando nesse diálogo dispõe implicitamente o diálogo. Isso é *true* padronizadamente, o que é um padrão razoável para diálogos simples

Retorna:

se a execução de um comando nesse diálogo dispõe implicitamente o diálogo

setAutoDispose

`public void setAutoDispose(boolean autoDispose)`

Determina se a execução de um comando nesse diálogo dispõe implicitamente o diálogo. Isso é *true* padronizadamente, o que é um padrão razoável para diálogos simples

Parâmetros:

`autoDispose` - estado auto-disposto

setDefaultDialogPosition

`public static void setDefaultDialogPosition(String p)`

Posição de orientação de tela padrão para o próximo diálogo. Padronizadamente o diálogo deve ser exibido em coordenadas irrevogáveis, esse método permite empacotar o diálogo apropriadamente em um dos locais baseados em layout de borda; consulte a classe `BorderLayout` para detalhes adicionais.

Parâmetros:

`p` - posição do diálogo padrão

Relaciona-se com:

`getDefaultDialogPosition()`

getDefaultDialogPosition

`public static String getDefaultDialogPosition()`

Posição de orientação de tela padrão para o próximo diálogo. Padronizadamente o diálogo deve ser exibido em coordenadas irrevogáveis, esse método permite empacotar o diálogo apropriadamente em um dos locais baseados em layout de borda; consulte a classe `BorderLayout` para detalhes adicionais.

Retorna:

posição do diálogo padrão

Relaciona-se com:

`setDefaultDialogPosition(java.lang.String)`

24.14 Classe Font

24.14.1 Descrição da classe

`com.sun.dtv.lwuit`

```
java.lang.Object
  └com.sun.dtv.lwuit.Font
```

```
abstract public class Font
extends Object
```

Uma simples abstração de fontes de plataforma e fontes de bibliotecas que possibilita a biblioteca usar fontes mais elaboradas não-suportadas por um dispositivo específico. Essa abstração também suporta fontes *bitmap*.

Para o uso do LWUIT dentro da especificação Java DTV, um objeto `Font` sempre empacota uma instância de `java.awt.Font`. O acesso do objeto empacotado é estimulado apenas se realmente necessário para manusear um LWUIT específico vs. Uma questão de integração. Não deveria ser usado para tarefas comuns.

24.14.2 Índice de campos

`static int` **`FACE_MONOSPACE`**

Constante permitindo designar sistemas de fontes portáteis.

`static int` **`FACE_PROPORTIONAL`**

Constante permitindo designar sistemas de fontes portáteis.

`static int` **`FACE_SYSTEM`**

Constante permitindo designar sistemas de fontes portáteis.

`static int` **`SIZE_LARGE`**

Constante permitindo designar sistemas de fontes portáteis.

`static int` **`SIZE_MEDIUM`**

Constante permitindo designar sistemas de fontes portáteis.

`static int` **`SIZE_SMALL`**

Constante permitindo designar sistemas de fontes portáteis.

`static int` **`STYLE_BOLD`**

Constante permitindo designar sistemas de fontes portáteis.

`static int` **`STYLE_ITALIC`**

Constante permitindo designar sistemas de fontes portáteis.

`static int` **`STYLE_PLAIN`**

Constante permitindo designar sistemas de fontes portáteis.

`static int` **`STYLE_UNDERLINED`**

Constante permitindo designar sistemas de fontes portáteis.

24.14.3 Índice de construtores

`protected` **`Font`**`()`
Constrói um objeto fonte.

`Font``(Font f)`
Cria um objeto fonte do LWUIT usando uma instância de fonte AWT a ser encapsulada.

`Font``(String name, int style, int size)`
Constrói um objeto fonte.

24.14.4 Índice de métodos

`void` **`addContrast`**`(byte value)`

Aumenta o contraste da fonte *bitmap* para renderização no topo de uma superfície cuja cor for mais escura.

`int` **`charsWidth`**`(char[] ch, int offset, int length)`

Retorna a largura de caracteres específicos nessa instância de fonte.

`abstract int` **`charWidth`**`(char ch)`

Retorna a largura de um caractere específico ao ser renderizado sozinho.

`static void` **`clearBitmapCache`**`()`

Fontes *bitmap* são registradas em cache.

`static Font` **`createBitmapFont`**`(Image bitmap, int[] cutOffsets, int[] charWidth, String charsets)`

Cria uma fonte *bitmap* com os dados argumentos.

`static Font` **`createBitmapFont`**`(String name, Image bitmap, int[] cutOffsets, int[] charWidth, String charsets)`

Cria uma fonte *bitmap* com os dados argumentos e coloca a tal fonte no cache.

`static Font` **`createSystemFont`**`(int face, int style, int size)`

Cria um fonte *system* nativa de forma similar a fontes MIDP comuns.

`static Font` **`decode`**`(String str)`

Retorna a `Font` descrita pelo argumento.

`abstract void` **`drawChar`**`(Graphics g, char character, int x, int y)`

Desenha um dado caractere usando a fonte atual e a cor nas coordenadas x e y.

`boolean` **`equals`**`(Object obj)`

Compara esse objeto fonte ao fonte especificado.

`Map` **`getAttributes`**`()`

Retorna um mapa de atributos disponíveis nessa fonte.

`Font` **`getAWTFont`**`()`

Obtém o objeto `java.awt.Font` encapsulado por essa instância de fonte.

`static Font` **`getBitmapFont`**`(String fontName)`

Retorna uma fonte *bitmap* carregada previamente do cache.

`String` **`getCharset`**`()`

Retorna um *string* contendo todos os caracteres suportados por essa fonte.

`static Font` **`getDefaultFont`**`()`

Retorna a instância de fonte padrão global.

`int` **`getFace`**`()`

Retorna uma operação opcional retornando a *face* do fonte para fontes *system*.

`String` **`getFamily`**`()`

Retorna o nome da família dessa fonte.

`static Font` **`getFont`**`(String nm)`

Retorna um objeto `Font` a partir da lista de propriedades do sistema.

`static Font` **`getFont`**`(String nm, Font font)`

Obtém a `Font` especificada do sistema.

`static Font` **`getFont`**`(Map attributes)`

Retorna uma `Font` apropriada a esse conjunto de atributos.

`int abstrato` **`getHeight`**`()`

Retorna a altura total da fonte.

`String` **`getName`**`()`

Retorna o nome lógico dessa fonte.

`int` **`getSize`**`()`

Retorna uma operação opcional retornando o tamanho da fonte para fontes system.

`int` **`getStyle`**`()`

Retorna uma operação opcional retornando o estilo da fonte para fontes system.

`int` **`hashCode`**`()`

Retorna um código *hash* para essa fonte.

`boolean` **`isBold`**`()`

Indica se essa fonte está em `BOLD`.

`boolean` **`isItalic`**`()`

Indica se essa fonte está em `ITALIC`.

`boolean` **`isPlain`**`()`

Indica se essa fonte está `PLAIN`.

`static void` **`setDefaultFont`**`(Font f)`

Configura a instância de fonte padrão global.

`int` ***`string`*****`Width`**`(String str)`

Retorna a largura de um dado *string* nessa instância fonte.

`int` **`sub`*****`string`*****`Width`**`(String str, int offset, int len)`

Retorna a largura de um dado *string* sub-agrupado nessa instância de fonte.

`String` **`toString`**`()`

Converte esse objeto fonte em uma *string*.

24.14.5 Detalhe dos campos

FACE_MONOSPACE

`public static final int FACE_MONOSPACE = 32`

Constante permitindo designar sistemas de fontes portáteis.

FACE_PROPORTIONAL

`public static final int FACE_PROPORTIONAL = 64`

Constante permitindo designar sistemas de fontes portáteis.

FACE_SYSTEM

```
public static final int FACE_SYSTEM = 0
```

Constante permitindo designar sistemas de fontes portáteis.

SIZE_LARGE

```
public static final int SIZE_LARGE = 16
```

Constante permitindo designar sistemas de fontes portáteis.

SIZE_MEDIUM

```
public static final int SIZE_MEDIUM = 0
```

Constante permitindo designar sistemas de fontes portáteis.

SIZE_SMALL

```
public static final int SIZE_SMALL = 8
```

Constante permitindo designar sistemas de fontes portáteis.

STYLE_BOLD

```
public static final int STYLE_BOLD = 1
```

Constante permitindo designar sistemas de fontes portáteis.

STYLE_ITALIC

```
public static final int STYLE_ITALIC = 2
```

Constante permitindo designar sistemas de fontes portáteis.

STYLE_UNDERLINED

```
public static final int STYLE_UNDERLINED = 4
```

Constante permitindo designar sistemas de fontes portáteis.

STYLE_PLAIN

```
public static final int STYLE_PLAIN = 0
```

Constante permitindo designar sistemas de fontes portáteis.

24.14.6 Detalhe dos construtores

Fonte

```
protected Font()
```

Constrói um objeto fonte.

Fonte

```
public Font(String name,
            int style,
            int size)
```

Constrói um objeto fonte.

Parâmetros:

`name` - o nome da fonte

`style` - o estilo da fonte

`size` - o tamanho da fonte

Fonte

```
public Font(Font f)
```

Cria um objeto fonte LWUIT usando uma instância de fonte AWT a ser encapsulada.

Parâmetros:

`f` - uma instância de fonte AWT a ser encapsulada.

24.14.7 Detalhe dos métodos

getBitmapFont

```
public static Font getBitmapFont(String fontName)
```

Retorna uma fonte *bitmap* carregada previamente do cache.

Parâmetros:

`fontName` - o nome da fonte é o nome lógico da fonte

Retorna:

um objeto fonte.

Relaciona-se com:

`clearBitmapCache()`

clearBitmapCache

```
public static void clearBitmapCache()
```

Fontes *bitmap* são registradas em cache. Esse método permite liberar o *cache*, permitindo então que recarreguemos uma fonte.

addContrast

```
public void addContrast(byte value)
```

Aumenta o contraste da fonte *bitmap* para renderização no topo de uma superfície cuja cor for mais escura. Isso é útil ao desenhar fontes *bitmap anti-aliasing* usando uma cor leve (por exemplo, branco) sobre uma superfície escura (por exemplo, preto). A fonte costuma falhar se o contraste não for aumentando, devido à maneira que a mistura alfa é vista pelos olhos.

Observar que esse método só funciona em um sentido, o contraste não pode ser diminuído apropriadamente em uma fonte e ela deveria ser limpa e recarregada com um mudança de Visual e Sensação.

Parâmetros:

`value` - o valor a ser aumentado

createBitmapFont

```
public static Font createBitmapFont(String name,
                                     Image bitmap,
                                     int[] cutOffsets,
                                     int[] charWidth,
                                     String charsets)
```

Cria uma fonte *bitmap* com os dados argumentos e coloca a tal fonte no cache.

Parâmetros:

`name` - o nome para essa fonte no *cache*

`bitmap` - um mapa de transparência em vermelho e preto que indica os caracteres

`cutOffsets` - ajustes de caracteres correspondendo aos pixels do *bitmap* e caracteres na fonte

`charWidth` - A largura do caractere ao desenhar. Não convém ser confundido com o número de `cutOffset[o + 1] - cutOffset[o]`. Eles são completamente diferentes, uma vez que um caractere pode ser mais “amplo” e “penetrar” na próxima região. Isso é especialmente verdadeiro para caracteres em itálico, os quais "se inclinam" para fora de seus limites.

`charsets` - o conjunto de caracteres da fonte

Retorna:

um objeto fonte para desenhar fontes *bitmap*

createBitmapFont

```
public static Font createBitmapFont(Image bitmap,
                                     int[] cutOffsets,
                                     int[] charWidth,
                                     String charsets)
```

Cria uma fonte *bitmap* com os dados argumentos.

Parâmetros:

`bitmap` - um mapa de transparência em vermelho e preto que indica os caracteres

`cutOffsets` - ajustes de caracteres correspondendo aos pixels do *bitmap* e caracteres na fonte

`charWidth` - A largura do caractere ao desenhar... não convém ser confundida com o número de `cutOffset[o + 1] - cutOffset[o]`. Eles são completamente diferentes, uma vez que um caractere pode ser mais “amplo” e “profundo” na próxima região. Isso é especialmente verdadeiro para caracteres em itálico, os quais "se inclinam" para fora de seus limites.

`charsets` - o conjunto de caracteres da fonte.

Retorna:

um objeto fonte para desenhar fontes *bitmap.*

createSystemFont

```
public static Font createSystemFont(int face,
                                     int style,
                                     int size)
```

Cria um fonte system nativa de forma similar a fontes MIDP comuns.

Parâmetros:

`face` - Algum desses: `FACE_SYSTEM, FACE_PROPORTIONAL, FACE_MONOSPACE`

`style` - Algum desses: `STYLE_PLAIN, STYLE_ITALIC, STYLE_BOLD`

`size` - Algum desses `SIZE_SMALL, SIZE_MEDIUM, SIZE_LARGE`

Retorna:

Uma instância fonte *system* recém-criada

charsWidth

```
public int charsWidth(char[] ch,
                      int offset,
                      int length)
```

Retorna a largura de caracteres específicos nessa instância fonte.

Parâmetros:

`ch` - *array* de caracteres

`offset` - deslocamento de caracteres

`length` - comprimento dos caracteres

Retorna:

a largura de caracteres específicos nessa instância de fonte

sub*string*Width

```
public int substringWidth(String str,
                          int offset,
                          int len)
```

Retorna a largura de um dado *string* sub-agrupado nessa instância de fonte.

Parâmetros:

`str` - a *string*

`offset` - o deslocamento da *string*

`len` - o tamanho da *string*

Retorna:

largura de uma dada *string* sub-agrupada nessa instância de fonte

***string*Width**

```
public int stringWidth(String str)
```

Retorna a largura de uma dada *string* nessa instância de fonte.

Parâmetros:

`str` - a *string* dada

Retorna:

a largura de uma dada *string* nessa instância de fonte

charWidth

```
public abstract int charWidth(char ch)
```

Retorna a largura de um caractere específico ao ser renderizado sozinho.

Parâmetros:

`ch` - um caractere específico

Retorna:

a largura de um caractere específico ao ser renderizado sozinho

getHeight

```
public abstract int getHeight()
```

Retorna a altura total da fonte.

Retorna:

a altura total da fonte

drawChar

```
public abstract void drawChar(Graphics g,
                              char character,
                              int x,
                              int y)
```

Desenha um dado caractere usando a fonte atual e a cor nas coordenadas x e y.

Parâmetros:

`g` - o objeto gráficos

`character` - o caractere dado

`x` - a coordenada x onde desenhar o caractere

`y` - a coordenada y onde desenhar o caractere

getDefaultFont

```
public static Font getDefaultFont()
```

Retorna a instância de fonte padrão global.

Retorna:

a instância de fonte padrão global

Relaciona-se com:

`setDefaultFont(com.sun.dtv.lwuit.Font)`

setDefaultFont

```
public static void setDefaultFont(Font f)
```

Configura a instância de fonte padrão global.

Parâmetros:

`f` - a instância de fonte padrão global

Relaciona-se com:

`getDefaultFont()`

getFace

`public int getFace()`

Retorna uma operação opcional retornando a *face* da fonte para fontes *system*.

Retorna:

Uma operação opcional retornando a *face* da fonte para fontes *system.*

getSize

`public int getSize()`

Retorna uma operação opcional retornando o tamanho da fonte para fontes system.

Retorna:

Uma operação opcional retornando o tamanho da fonte para fontes system.

getStyle

`public int getStyle()`

Retorna uma operação opcional retornando o estilo da fonte para fontes system.

Retorna:

Uma operação opcional retornando o estilo da fonte para fontes system

getCharset

`public String getCharset()`

Retorna um *string* contendo todos os caracteres suportados por essa fonte. Retornará `null` para fontes system.

Retorna:

String contendo os caracteres suportado por uma fonte *bitmap*; caso contrário `null`.

getAWTFont

`public Font getAWTFont()`

Obtém o objeto `java.awt.Font` encapsulado por essa instância de fonte.

Retorna:

O objeto `java.awt.Font` encapsulado

decode

`public static Font decode(String str)`

Retorna a `Font` descrita pelo argumento.

NOTA Na implementação:
Delega ao objeto fonte AWT encapsulado.

Parâmetros:
`str` - a descrição
Retorna:
a `Font` a ser encontrada

equals
`public boolean equals(Object obj)`
Compara esse objeto fonte ao fonte especificado.

NOTA Na implementação:
Delega ao objeto fonte AWT encapsulado.

Substituições:
`equals` na classe `Object`
Parâmetros:
`obj` - o objeto a ser confrontado.
Retorna:
`true` se igual, *`false`* caso contrário.

getAttributes
`public Map getAttributes()`
Retorna um mapa de atributos disponíveis nessa fonte.

NOTA Na implementação:
Delega ao objeto fonte AWT encapsulado.

Retorna:
o mapa de atributos

getFamily
`public String getFamily()`
Retorna o nome da família dessa fonte.

NOTA Na implementação:
Delega ao objeto fonte AWT encapsulado.

Retorna:

o nome da família

getFont

```
public static Font getFont(Map attributes)
```

Retorna uma fonte apropriada a esse conjunto de atributos.

NOTA Na implementação:

Delega ao objeto fonte AWT encapsulado.

Parâmetros:

`attributes` - o conjunto de atributos

Retorna:

a fonte apropriada

getFont

```
public static Font getFont(String nm)
```

Retorna um objeto fonte a partir da lista de propriedades do sistema.

NOTA Na implementação:

Delega ao objeto fonte AWT encapsulado.

Parâmetros:

`nm` - o nome da propriedade

Retorna:

a fonte apropriada

getFont

```
public static Font getFont(String nm,
                           Font font)
```

Obtém a fonte especificada do sistema.

NOTA Na implementação:

Delega ao objeto fonte AWT encapsulado.

Parâmetros:

`nm` - o nome da propriedade

`font` - uma fonte padrão a ser retornada se o nome da propriedade não existir

Retorna:

a fonte encontrada

getName

`public String getName()`

Retorna o nome lógico dessa fonte.

NOTA Na implementação:

Delega ao objeto fonte AWT encapsulado.

Retorna:

o nome lógico

hashCode

`public int hashCode()`

Retorna um código *hash* para essa fonte.

NOTA Na implementação:

Delega ao objeto fonte AWT encapsulado.

Substituições:

`hashCode` na classe `Object`

Retorna:

o `hashcode`

isBold

`public boolean isBold()`

Indica se essa fonte está em *bold*.

NOTA Na implementação:

Delega ao objeto fonte AWT encapsulado.

Retorna:

`true` se a `Font` estiver em *bold*; caso contrário *`false`*

isItalic

`public boolean isItalic()`

Indica se essa `Font` está em `ITALIC`.

NOTA Na implementação:
Delega ao objeto fonte AWT encapsulado.

Retorna:
true se a `Font` estiver em `ITALIC`; caso contrário *false*

isPlain

```
public boolean isPlain()
```

Indica se essa fonte está `PLAIN`.

NOTA Na implementação:
Delega ao objeto fonte AWT encapsulado.

Retorna:
true se a fonte estiver `PLAIN`; caso contrário *false*

toString

```
public String toString()
```

Converte esse objeto `Font` em uma *string*.

NOTA Na implementação:
Delega ao objeto fonte AWT encapsulado.

Substituições:
`toString` na classe `Object`
Retorna:
uma representação dessa fonte em uma *string*.

24.15 Classe Form

24.15.1 Descrição da classe

`com.sun.dtv.lwuit`

```
java.lang.Object
  └com.sun.dtv.lwuit.Component
      └com.sun.dtv.lwuit.Container
          └com.sun.dtv.lwuit.Form
```

Todas as interfaces implementadas
`Animated, Animation, MatteEnabled, StyleListener`

Subclasses diretas conhecidas

`Dialog`

`public class Form`

`extends Container`

componente de nível mais alto que serve como a entidade mais visível no UI (diretamente acoplado no `DTVContainer`, esse `Container` manuseia os menus e título enquanto coloca conteúdo entre eles. Por padrão, o conteúdo central de um formuláro (o painel de conteúdo) é rolável. O formulário contém barra de título, barra de menu e um painel de conteúdo. A chamada de `addComponent` no formulário é delegada ao `contentPane.addComponent`.

Campos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`BOTTOM, BRB_CENTER_OFFSET, BRB_CONSTANT_ASCENT, BRB_CONSTANT_DESCENT, BRB_OTHER, CENTER, LEFT, RIGHT, TOP`

Campos herdados da interface `com.sun.dtv.ui.`**`Animated`**

`ALTERNATING, LOOP, REPEATING`

24.15.2 Índice de construtores

`Form``()`

Um construtor padrão cria um formulário simples.

`Form``(String title)`

Define o título após invocar o construtor.

24.15.3 Índice de métodos

`void addCommand(Command cmd)`

Adiciona um comando às *softkeys* da barra de menu.

`void addCommand(Command cmd, int offset)`

Adiciona um comando às *softkeys* da barra de menu ou no diálogo do menu; esse tipo de adicionamento permite colocar um comando em local arbitrário.

`void addComponent(Component cmp)`

Adiciona componente ao painel de conteúdo do formulário.

`void addComponent(int index, Component cmp)`

Adiciona componente ao painel de conteúdo do formulário.

`void addComponent(Object constraints, Component cmp)`

Adiciona componente ao painel de conteúdo do formulário.

`void addGameKeyListener(int keyCode, ActionListener listener)`

Adiciona um *listener* de tecla de jogo a uma dada tecla de jogo para uma chamada de retorno quando a tecla é liberada.

`void addKeyListener(int keyCode, ActionListener listener)`

Adiciona um *listener* tecla a um dado keycode para uma chamada de retorno quando a tecla é liberada.

`void deregisterAnimation(Animation cmp)`

Indica que a `cmp` não está mais recebendo eventos de animação.

`Command getBackCommand()`

Indica o comando definido como o comando voltar nesse formuláro.

`Command getClearCommand()`

Indica o comando definido como o comando limpar nesse formulário.

`Command getCommand(int index)`

Retorna o comando ocupando um dado índice.

`int getCommandCount()`

Um método de ajuda para verificar a quantidade de comandos no menu do formulário.

`Form getComponentForm()`

Retorna o formulário do componente ou `null`, se esse componente não for adicionado a um formulário.

`Container getContentPane()`

Esse método retorna a instância do painel de conteúdo.

`Command getDefaultCommand()`

Comando padrão é invocado quando um usuário pressiona ativar; essa funcionalidade funciona bem em algumas situações, mas pode colidir com elementos como navegação e combo box.

`Component getFocused()`

Retorna o componente de foco atual para esse formulário.

`Painter getGlassPane()`

Permite que um desenvolvedor que não recebe do formulário a desenhar no topo da forma independente de animações ou mudanças subjacentes.

`int getScrollAnimationSpeed()`

Coloca em barra de rolagem a velocidade da animação em milissegundos, permitindo que um desenvolvedor diminua ou acelere o modo de animação suave.

`Button getSoftButton(int offset)`

Retorna o botão representando o *softbutton*; isso permite modificar os atributos e comportamento do softbutton programaticamente, em vez de usando a API de comando.

`int getSoftButtonCount()`

Retorna o número de botões na barra de menu para uso com `getSoftButton()`.

`Style getSoftButtonStyle()`

Restaura o estilo da barra de menu programaticamente.

`Color getTintColor()`

A tonalidade padrão para cor da tela, quando um diálogo ou menu é exibido.

`String getTitle()`

Retorna o texto título do formulário.

`Label getTitleComponent()`

Permite modificar os atributos de título além de estilo (como definição de ícone/alinhamento e etc.).

`Style getTitleStyle()`

Retorna o estilo do título.

```
Transition getTransitionInAnimator()
```

Essa propriedade permite definir uma animação que irá ilustrar a transição para entrar nesse formulário.

```
Transition getTransitionOutAnimator()
```

Essa propriedade permite definir uma animação que irá ilustrar a transição para sair desse formulário.

```
protected void hideNotify()
```

Esse método só é invocado quando o ambiente gráfico para o formulário está oculto.

```
boolean isSingleFocusMode()
```

Retorna *true* se houver apenas um membro focável nesse formulário.

```
boolean isSmoothScrolling()
```

Indica que essa rolagem pelo componente deveria funcionar como uma animação.

```
void keyPressed(int keyCode)
```

Se esse componente estiver destacado, o evento chave pressionado chamará esse método.

```
void keyReleased(int keyCode)
```

Se esse componente estiver destacado, o evento chave liberado chamará esse método.

```
void keyRepeated(int keyCode)
```

Se esse componente estiver destacado, o evento chave repetir chamará esse método.

```
void longKeyPress(int keyCode)
```

Se esse componente estiver destacado, o método é invocado quando o usuário pressionar e mantém a tecla pressionada.

```
protected void onShow()
```

Permite subclasses a ligar funcionalidade que ocorrem imediatamente após um diálogo ou forma específica aparecer na tela

```
void paint(Graphics g)
```

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

```
void paintBackground(Graphics g)
```

Exposição da pintura de plano de fundo para benefício das animações.

```
void pointerDragged(int x, int y)
```

Se esse componente estiver destacado, o evento indicador arrastado chamará esse método.

```
void pointerPressed(int x, int y)
```

Se esse componente estiver destacado, o evento indicador pressionado chamará esse método.

```
void pointerReleased(int x, int y)
```

Se esse componente estiver destacado, o evento indicador liberado chamará esse método.

```
void refreshTheme()
```

Certifica-se de que o componente está atualizado com o objeto estilo atual.

```
void registerAnimation(Animation cmp)
```

Um dado componente está interessado em animar sua aparência e começará a receber chamadas de retornos

quando estiver visível no formulário, permitindo que sua aparência seja animada.

```
void removeAll()
```

Remove todos os componentes do painel de conteúdo.

```
void removeAllCommands()
```

Limpa os comandos do menu da barra de menu.

```
void removeCommand(Command cmd)
```

Remove o comando das *softkeys* da barra de menu.

```
void removeComponent(Component cmp)
```

Remove um componente do painel de conteúdo do formulário.

```
void removeGameKeyListener(int keyCode, ActionListener listener)
```

Remove um *listener* tecla de jogo de um dado keycode de jogo.

```
void removeKeyListener(int keyCode, ActionListener listener)
```

Remove um *listener* tecla de um dado keycode.

```
void replace(Component current, Component next, Transition t)
```

Esse método substitui o componente atual pelo próximo componente.

```
void scrollComponentToVisible(Component c)
```

Certifica-se de que o componente está visível na barra de rolagem, se esse container for rolável.

```
void setBackCommand(Command backCommand)
```

Indica o comando definido como o comando voltar nesse formuláro.

```
void setBgImage(Image bgImage)
```

Define a imagem de plano de fundo a ser exibida atrás do formulário.

```
void setClearCommand(Command clearCommand)
```

Indica o comando definido como o comando limpar nesse formulário.

```
void setCommandListener(ActionListener commandListener)
```

Um *listener* que é invocado quando um comando é clicado, permitindo que múltiplos comandos sejam manuseados por um único bloco.

```
void setDefaultCommand(Command defaultCommand)
```

Comando padrão é invocado quando um usuário pressiona ativar; essa funcionalidade funciona bem em algumas situações, mas pode colidir com elementos como navegação e combo box.

```
void setFocused(Component focused)
```

Define o componente focado e ativa os eventos apropriados para fazer com que assim seja.

```
void setGlassPane(Painter glassPane)
```

Permite que um desenvolvedor que não recebe do formulário a desenhar no topo da forma independente de animações ou mudanças subjacentes.

```
void setLayout(Layout layout)
```

Define o gerenciador de layout responsável por organizar esse container.

```
void setMenuCellRenderer(ListCellRenderer menuCellRenderer)
```

Determina o renderizador de células usado para renderizar elementos de menu para tematizar a aparência das opções do menu.

```
void setMenuTransitions(Transition transitionIn, Transition transitionOut)
```

Define as transições do menu para exibir/esconder o menu.

`void setScrollable(boolean scrollable)`

O equivalente de chamar `setScrollableY` e `setScrollableX`.

`void setScrollableX(boolean scrollableX)`

Define se o componente deveria/poderia rolar no eixo X.

`void setScrollableY(boolean scrollableY)`

Define se o componente deveria/poderia rolar no eixo Y.

`void setScrollAnimationSpeed(int animationSpeed)`

Coloca em barra de rolagem a velocidade da animação em milissegundos, permitindo que um desenvolvedor diminua ou acelere o modo de animação suave.

`void setSmoothScrolling(boolean smoothScrolling)`

Indica que essa rolagem pelo componente deveria funcionar como uma animação.

`void setSoftButtonStyle(Style s)`

Define o estilo da barra de menu programaticamente.

`void setTintColor(Color tintColor)`

A tonalidade padrão para cor da tela, quando um diálogo ou menu é exibido.

`void setTitle(String title)`

Define o título do Form para um dado texto.

`void setTitleStyle(Style s)`

Define o estilo do título programaticamante.

`void setTransitionInAnimator(Transition transitionInAnimator)`

Essa propriedade permite definir uma animação que irá ilustrar a transição para entrar nesse formulário.

`void setTransitionOutAnimator(Transition transitionOutAnimator)`

Essa propriedade permite definir uma animação que irá ilustrar a transição para sair desse formulário.

`void setVisible(boolean visible)`

Muda a visibilidade do componente.

`void show()`

Exibe o formulário atual na tela.

`protected Command showMenuDialog(Dialog menu)`

Se um menu é implementado como um diálogo, esse método permite sobrepor a exibição do diálogo para personalizar o menu do diálogo de várias maneiras

`protected void showNotify()`

Esse método só é invocado quando o ambiente gráfico para o formulário for exibido.

`protected void sizeChanged(int w, int h)`

Esse método só é invocado quando o ambiente gráfico para o formulário recebe um evento de tamanho modificado.

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Container`**

contains, doLayout, getComponentAt, getComponentAt, getComponentCount, getComponentIndex, getLayout, getLayoutHeight, getLayoutWidth, getMatte, invalidate, isScrollableX, isScrollableY, layoutContainer, revalidate, setCellRenderer, setMatte

Métodos herdados da classe com.sun.dtv.lwuit.**Component**

addFocusListener, animate, calcPreferredSize, contains, deinitialize, getAbsoluteX, getAbsoluteY, getAnimationMode, getBaseline, getBaselineResizeBehavior, getBottomGap, getBounds, getClientProperty, getDelay, getHeight, getNextFocusDown, getNextFocusLeft, getNextFocusRight, getNextFocusUp, getParent, getPosition, getPreferredSize, getRepetitionMode, getScrollX, getScrollY, getSideGap, getStyle, getUIID, getWidth, getX, getY, handlesInput, hasFocus, initialize, isEnabled, isFocusable, isFocusPainted, isInitialized, isRunning, isScrollVisible, isVisible, jumpTo, paintBackgrounds, paintComponent, paintComponent, putClientProperty, removeFocusListener, repaint, repaint, requestFocus, scrollRectToVisible, setAnimationMode, setDelay, setEnabled, setFocus, setFocusable, setFocusPainted, setHandlesInput, setHeight, setInitialized, setIsScrollVisible, setNextFocusDown, setNextFocusLeft, setNextFocusRight, setNextFocusUp, setPreferredSize, setRepetitionMode, setScrollX, setScrollY, setShouldCalcPreferredSize, setSize, setStyle, setUIID, setWidth, setX, setY, start, stop, styleChanged, toString

24.15.4 Detalhe dos construtores

Form

```
public Form()
```

Um construtor padrão cria um formulário simples.

Form

```
public Form(String title)
```

Define o título após invocar o construtor.

Parâmetros:

`title` - o título do formulário

24.15.5 Detalhe dos métodos

setSoftButtonStyle

```
public void setSoftButtonStyle(Style s)
```

Define o estilo da barra de menu programaticamente.

Parâmetros:

`s` - novo estilo

Relaciona-se com:

`getSoftButtonStyle()`

getSoftButtonStyle

```
public Style getSoftButtonStyle()
```

Restaura o estilo da barra de menu programaticamente.

Retorna:

Um objeto estilo representando o valor de estilo dos botões soft

Relaciona-se com:

`setSoftButtonStyle(com.sun.dtv.lwuit.plaf.Style)`

setGlassPane

```
public void setGlassPane(Painter glassPane)
```

Permite que um desenvolvedor que não recebe do formulário a desenhar no topo da forma independente de animações ou mudanças subjacentes. Isso é útil para marcas d'água ou efeitos especiais (como tinting), também é útil para desenhos genéricos de erros de validação e etc. Uma painel de vidro é, geralmente, transparente ou translúcido e permite que o UI abaixo seja visto.

Parâmetros:

`glassPane` - um novo painel de vidro a ser instalado. É recomendado que se use uma corrente de pintores se mais de um pintor for requerido.

Relaciona-se com:

`getGlassPane()`

getGlassPane

```
public Painter getGlassPane()
```

Permite que um desenvolvedor que não recebe do formulário a desenhar no topo da forma independente de animações ou mudanças subjacentes. Isso é útil para marcas d'água ou efeitos especiais (como "*tinting*"), também é útil para desenhos genéricos de erros de validação e etc. Uma painel de vidro é, geralmente, transparente ou translúcido e permite que o UI abaixo seja visto.

Retorna:

a instância do painel de vidro para esse formulário

Relaciona-se com:

`setGlassPane(com.sun.dtv.lwuit.Painter)`, `PainterChain.installGlassPane(Form, com.sun.dtv.lwuit.Painter)`

setTitleStyle

```
public void setTitleStyle(Style s)
```

Define o estilo do título programaticamante.

Parâmetros:

`s` - novo estilo

Relaciona-se com:

`getTitleStyle()`

getTitleComponent

```
public Label getTitleComponent()
```

Permite modificar os atributos de título além de estilo (como definição de ícone/alinhamento e etc).

Retorna:

o componente representando o título para o formulário

addKeyListener

```
public void addKeyListener(int keyCode,
                           ActionListener listener)
```

Adiciona um *listener* de tecla a um dado código de tecla para uma chamada de retorno quando a tecla é liberada.

Parâmetros:

`keyCode` - código por meio do qual enviar o evento

`listener` - *listener* a invocar quando o keycode for liberado.

Relaciona-se com:

`removeKeyListener(int, com.sun.dtv.lwuit.events.ActionListener)`

removeKeyListener

```
public void removeKeyListener(int keyCode,
                              ActionListener listener)
```

Remove um *listener* de tecla de um dado keycode.

Parâmetros:

`keyCode` - código por meio do qual o evento é enviado

`listener` - instância de *listener* a ser removida

Relaciona-se com:

`addKeyListener(int, com.sun.dtv.lwuit.events.ActionListener)`

removeGameKeyListener

```
public void removeGameKeyListener(int keyCode,
                                  ActionListener listener)
```

Remove um *listener* de tecla de jogo de um dado *keycode* de jogo.

Parâmetros:

`keyCode` - código por meio do qual o evento é enviado

`listener` - instância de *listener* a ser removida

Relaciona-se com:

`addGameKeyListener(int, com.sun.dtv.lwuit.events.ActionListener)`

addGameKeyListener

```
public void addGameKeyListener(int keyCode,
                               ActionListener listener)
```

Adiciona um *listener* de tecla de jogo a uma dada tecla de jogo para uma chamada de retorno quando a tecla é liberada.

Parâmetros:

`keyCode` - código por meio do qual enviar o evento

`listener` - *listener* a invocar quando o *keycode* for liberado.

Relaciona-se com:

`removeGameKeyListener(int, com.sun.dtv.lwuit.events.ActionListener)`

getSoftButtonCount

`public int getSoftButtonCount()`

Retorna o número de botões na barra de menu para uso com `getSoftButton()`.

Retorna:

um inteiro representando o valor de contagem do `softbutton`.

getSoftButton

`public Button getSoftButton(int offset)`

Retorna o botão representando o softbutton; isso permite modificar os atributos e comportamento do softbutton programaticamente, em vez de usando a API de comando. Observar que o comportamento desse API é frágil, uma vez que o botão mapeado para um ajuste específico pode mudar baseado na API de comando.

Parâmetros:

`offset` - o ajuste para o qual o botão soft é mapeado

Retorna:

um botão que pode ser manipulado

getTitleStyle

`public Style getTitleStyle()`

Retorna o estilo do título.

Retorna:

um objeto estilo representando o valor de estilo do título

Relaciona-se com:

`setTitleStyle(com.sun.dtv.lwuit.plaf.Style)`

setDefaultCommand

`public void setDefaultCommand(Command defaultCommand)`

Comando padrão é invocado quando um usuário pressiona ativar; essa funcionalidade funciona bem em algumas situações, mas pode colidir com elementos como navegação e combo box. Use com cuidado.

Parâmetros:

`defaultCommand` - um objeto Comando especificando o valor de comando padrão

Relaciona-se com:

`getDefaultCommand()`

getDefaultCommand

`public Command getDefaultCommand()`

Comando padrão é invocado quando um usuário pressiona ativar; essa funcionalidade funciona bem em algumas situações, mas pode colidir com elementos como navegação e combo box. Use com cuidado.

Retorna:

um objeto Comando representando o valor de comando padrão

Relaciona-se com:

`setDefaultCommand(com.sun.dtv.lwuit.Command)`

setClearCommand

```
public void setClearCommand(Command clearCommand)
```

Indica o comando definido como o comando limpar nesse formulário. Um comando limpar pode ser usado tanto para mapear quanto para "limpar" um botão de hardware, se houver um desses botões.

Parâmetros:

`clearCommand` - um objeto Comando especificando o valor do comando limpar

Relaciona-se com:

`getClearCommand()`

getClearCommand

```
public Command getClearCommand()
```

Indica o comando definido como o comando limpar nesse formulário. Um comando limpar pode ser usado tanto para mapear quanto para "limpar" um botão de hardware, se houver um desses botões.

Retorna:

um objeto Comando representando o valor do comando limpar

Relaciona-se com:

`setClearCommand(com.sun.dtv.lwuit.Command)`

setBackCommand

```
public void setBackCommand(Command backCommand)
```

Indica o comando definido como o comando voltar nesse formuláro. Um Comando voltar pode ser usado tanto para mapear quanto para um botão de hardware (por exemplo, nos dispositivos Sony Ericsson) e por elementos, como transições e etc., para mudar o comportamento baseado em direção (por exemplo, deslizar para a esquerda para entrar na tela e deslizar para a direita para sair com voltar).

Parâmetros:

`backCommand` - um objeto Comando especificando o valor do comando voltar

Relaciona-se com:

`getBackCommand()`

getBackCommand

```
public Command getBackCommand()
```

Indica o comando definido como o comando voltar nesse formuláro. Um Comando voltar pode ser usado tanto

para mapear quanto para um botão de hardware (por exemplo, nos dispositivos Sony Ericsson) e por elementos, como transições e etc., para mudar o comportamento baseado em direção (por exemplo, deslizar para a esquerda para entrar na tela e deslizar para a direita para sair com voltar).

Retorna:

um objeto Comando representando o valor do comando voltar

Relaciona-se com:

`setBackCommand(com.sun.dtv.lwuit.Command)`

getContentPane

`public Container getContentPane()`

Esse método retorna a instância do painel de conteúdo.

Retorna:

uma instância painel de conteúdo

removeAll

`public void removeAll()`

Remove todos os componentes do painel de conteúdo.

Substituições:

`removeAll` na classe `Container`

setBgImage

`public void setBgImage(Image bgImage)`

Define a imagem de plano de fundo a ser exibida atrás do formulário.

Parâmetros:

`bgImage` - a imagem de plano de fundo

setLayout

`public void setLayout(Layout layout)`

Define o gerenciador de layout responsável por organizar esse container.

Substituições:

`setLayout` na classe `Container`

Parâmetros:

`layout` - o gerenciador de layout especificado

setTitle

`public void setTitle(String title)`

Define o título do Form para um dado texto.

Parâmetros:

`title` - o título do formulário

Relaciona-se com:

`getTitle()`

getTitle

```
public String getTitle()
```

Retorna o texto título do Form.

Retorna:

um String representando o valor do título

Relaciona-se com:

`setTitle(java.lang.String)`

addComponent

```
public void addComponent(Component cmp)
```

Adiciona componente ao painel de conteúdo do formulário.

Substituições:

`addComponent` na classe `Container`

Parâmetros:

`cmp` - o parâmetro adicionado

addComponent

```
public void addComponent(Object constraints,
                         Component cmp)
```

Adiciona componente ao painel de conteúdo do formulário.

Substituições:

`addComponent` na classe `Container`

Parâmetros:

`constraints` - esse método é útil quando o layout requer uma limitação como o `BorderLayout`. Nesse caso é preciso especificar um dado adicional quando se adiciona um componente, como `"CENTER", "NORTH"`...

`cmp` - componente a adicionar

addComponent

```
public void addComponent(int index,
                         Component cmp)
```

Adiciona componente ao painel de conteúdo do formulário.

Substituições:

`addComponent` na classe `Container`

Parâmetros:

`index` - índice onde adicionar o componente

`cmp` - o parâmetro adicionado

replace

```
public void replace(Component current,
                    Component next,
                    Transition t)
```

Esse método substitui o componente atual pelo próximo componente. O componente atual deve estar contido nesse *container*. Esse método retorna imediatamente.

Substituições:

`replace` na classe `Container`

Parâmetros:

`current` - um componente a ser removido do *container*

`next` - um componente que substitui o componente atual

`t` - uma transição entre o adicionamento e remoção de componentes; uma transição pode ser `null`

removeComponent

```
public void removeComponent(Component cmp)
```

Remove um componente do painel de conteúdo do formulário.

Substituições:

`removeComponent` na classe `Container`

Parâmetros:

`cmp` - o componente a ser removido

registerAnimation

```
public void registerAnimation(Animation cmp)
```

Um dado componente está interessado em animar sua aparência e começará a receber chamadas de retornos quando estiver visível no formulário, permitindo que sua aparência seja animada. Esse método não registraria uma instância componente mais de uma vez.

Parâmetros:

`cmp` - componente que seria animado

deregisterAnimation

```
public void deregisterAnimation(Animation cmp)
```

Indica que o cmp não está mais recebendo eventos de animação.

Parâmetros:

`cmp` - componente que não mais receberá eventos animados

refreshTheme

```
public void refreshTheme()
```

Certifica-se de que o componente está atualizado com o objeto estilo atual.

Substituições:

`refreshTheme` na classe `Container`

paintBackground

```
public void paintBackground(Graphics g)
```

Exposição da pintura de plano de fundo para benefício das animações.

Parâmetros:

`g` - os gráficos do formulário

getTransitionInAnimator

```
public Transition getTransitionInAnimator()
```

Essa propriedade permite definir uma animação que irá ilustrar a transição para entrar nesse formulário. Uma transição é uma animação que ocorre na mudança de um formulário para outro.

Retorna:

o formulário em transição

Relaciona-se com:

```
setTransitionInAnimator(com.sun.dtv.lwuit.animations.Transition)
```

setTransitionInAnimator

```
public void setTransitionInAnimator(Transition transitionInAnimator)
```

Essa propriedade permite definir uma animação que irá ilustrar a transição para entrar nesse formulário. Uma transição é uma animação que ocorre na mudança de um formulário para outro.

Parâmetros:

`transitionInAnimator` – o formulário em transição

Relaciona-se com:

```
getTransitionInAnimator()
```

getTransitionOutAnimator

```
public Transition getTransitionOutAnimator()
```

Essa propriedade permite definir uma animação que irá ilustrar a transição para sair desse formulário. Uma transição é uma animação que ocorre na mudança de um formulário para outro.

Retorna:

o formulário fora de transição

Relaciona-se com:

```
setTransitionOutAnimator(com.sun.dtv.lwuit.animations.Transition)
```

setTransitionOutAnimator

```
public void setTransitionOutAnimator(Transition transitionOutAnimator)
```

Essa propriedade permite definir uma animação que irá ilustrar a transição para sair desse formulário. Uma transição é uma animação que ocorre na mudança de um formulário para outro.

Parâmetros:

`transitionOutAnimator` - o formulário fora de transição

Relaciona-se com:

`getTransitionOutAnimator()`

setCommandListener

`public void setCommandListener(ActionListener commandListener)`

Um *listener* que é invocado quando um comando é clicado, permitindo que múltiplos comandos sejam manuseados por um único bloco.

Parâmetros:

`commandListener` - o *listener* de comando ação

show

`public void show()`

Exibe o formulário atual na tela.

setSmoothScrolling

`public void setSmoothScrolling(boolean smoothScrolling)`

Indica que essa rolagem pelo componente deveria funcionar como uma animação.

Substituições:

`setSmoothScrolling` na classe `Component`

Parâmetros:

`smoothScrolling` - indica se um componente usa rolagem suave

isSmoothScrolling

`public boolean isSmoothScrolling()`

Indica que essa rolagem pelo componente deveria funcionar como uma animação.

Substituições:

`isSmoothScrolling` na classe `Component`

Retorna:

se esse componente usa rolagem suave

getScrollAnimationSpeed

`public int getScrollAnimationSpeed()`

Coloca em barra de rolagem a velocidade da animação em milissegundos, permitindo que um desenvolvedor diminua ou acelere o modo de animação suave.

Substituições:

`getScrollAnimationSpeed` na classe `Component`

Retorna:

Barra de rolagem da velocidade da animação em milissegundos

Relaciona-se com:

`setScrollAnimationSpeed()`

setScrollAnimationSpeed

```
public void setScrollAnimationSpeed(int animationSpeed)
```

Coloca em barra de rolagem a velocidade da animação em milissegundos, permitindo que um desenvolvedor diminua ou acelere o modo de animação suave.

Substituições:

`setScrollAnimationSpeed` na classe `Component`

Parâmetros:

`animationSpeed` - barra de rolagem da velocidade da animação em milissegundos

Relaciona-se com:

`getScrollAnimationSpeed()`

getComponentForm

```
public final Form getComponentForm()
```

Retorna o formulário do componente ou `null`, se esse componente não for adicionado a um formulário.

Substituições:

`getComponentForm` na classe `Component`

Retorna:

o formulário do componente

setFocused

```
public void setFocused(Component focused)
```

Define o componente focado e ativa os eventos apropriados para fazer com que assim seja.

Parâmetros:

`focused` - o novo componente focado ou `null`, se não houver foco

Relaciona-se com:

`getFocused()`

getFocused

```
public Component getFocused()
```

Retorna o componente de foco atual para esse formulário.

Retorna:

o componente de foco atual para esse formulário

Relaciona-se com:

`setFocused(com.sun.dtv.lwuit.Component)`

longKeyPress

```
public void longKeyPress(int keyCode)
```

Se esse componente estiver destacado, o método é invocado quando o usuário pressionar e mantém a tecla pressionada.

Substituições:

`longKeyPress` na classe `Component`

Parâmetros:

`keyCode` - o valor do código chave a indicar a chave física.

keyPressed

```
public void keyPressed(int keyCode)
```

Se esse componente estiver destacado, o evento chave pressionado chamará esse método.

Substituições:

`keyPressed` na classe `Component`

Parâmetros:

`keyCode` - o valor do código chave a indicar a chave física.

keyReleased

```
public void keyReleased(int keyCode)
```

Se esse componente estiver destacado, o evento chave liberado chamará esse método.

Substituições:

`keyReleased` na classe `Component`

Parâmetros:

`keyCode` - o valor do código chave a indicar a chave física.

keyRepeated

```
public void keyRepeated(int keyCode)
```

Se esse componente estiver destacado, o evento chave repetir chamará esse método. Chama teclas pressionadas/liberadas automaticamente.

Substituições:

`keyRepeated` na classe `Component`

Parâmetros:

`keyCode` - o valor do código chave a indicar a chave física.

pointerPressed

```
public void pointerPressed(int x,
                           int y)
```

Se esse componente estiver destacado, o evento indicador pressionado chamará esse método.

Substituições:

`pointerPressed` na classe `Container`

Parâmetros:

`x` - a coordenada do indicador x

`y` - a coordenada do indicador y

pointerDragged

```
public void pointerDragged(int x,
                           int y)
```

Se esse componente estiver destacado, o evento indicador arrastado chamará esse método.

Substituições:

`pointerDragged` na classe `Component`

Parâmetros:

`x` - a coordenada do indicador x

`y` - a coordenada do indicador y

isSingleFocusMode

```
public boolean isSingleFocusMode()
```

Retorna *true* se houver apenas um membro focável nesse formulário. Isso é útil, portanto, `setHandlesInput` sempre será *true* para esse caso.

Retorna:

true se houver um componente focável nesse formulário; *false* para 0 ou mais

pointerReleased

```
public void pointerReleased(int x,
                            int y)
```

Se esse componente estiver destacado, o evento indicador liberado chamará esse método.

Substituições:

`pointerReleased` na classe `Component`

Parâmetros:

`x` - a coordenada do indicador x

`y` - a coordenada do indicador y

setScrollableY

```
public void setScrollableY(boolean scrollableY)
```

Define se o componente deveria/poderia rolar no eixo Y.

Substituições:

`setScrollableY` na classe `Container`

Parâmetros:

`scrollableY` - se o componente deveria/poderia rolar no eixo Y

setScrollableX

```
public void setScrollableX(boolean scrollableX)
```

Define se o componente deveria/poderia rolar no eixo X.

Substituições:

`setScrollableX` na classe `Container`

Parâmetros:

`scrollableX` - se o componente deveria/poderia rolar no eixo X

addCommand

```
public void addCommand(Command cmd,
                       int offset)
```

Adiciona um comando às *softkeys* da barra de menu ou no diálogo do menu; esse tipo de adicionamento permite colocar um comando em local arbitrário. Isso permite forçar um comando para as *softkeys* quando a ordem de adicionamento de comandos não puder ser mudada.

Parâmetros:

`cmd` - o comando de formulário a ser adicionado

`offset` - posição onde o comando é adicionado

getCommandCount

```
public int getCommandCount()
```

Um método de ajuda para verificar a quantidade de comandos no menu do formulário.

Retorna:

um int representando o valor de contagem do comando

getCommand

```
public Command getCommand(int index)
```

Retorna o comando ocupando um dado índice.

Parâmetros:

`index` - ajuste do comando

Retorna:

o comando no índice específico

addCommand

```
public void addCommand(Command cmd)
```

Adiciona um comando às *softkeys* da barra de menu. Os comandos são colocados na ordem em que são adicionados. Se o formulário tem um comando, ele deve ser colocado à direita. Se o formulário tem dois comandos, o primeiro a ser adicionado deve ser colocado à direita e o segundo deve ser colocado à esquerda. Se o formulário tem mais de dois comandos, o primeiro permanecerá à esquerda e um menu deve ser adicionado

com todos os comandos restantes.

Parâmetros:

`cmd` - o comando de formulário a ser adicionado

removeCommand

```
public void removeCommand(Command cmd)
```

Remove o comando das *softkeys* da barra de menu.

Parâmetros:

`cmd` - o comando de formulário a ser removido

scrollComponentToVisible

```
public void scrollComponentToVisible(Component c)
```

Certifica-se de que o componente está visível na barra de rolagem, se esse container for rolável.

Substituições:

`scrollComponentToVisible` na classe `Container`

Parâmetros:

`c` - o componente a se tornar visível

setMenuCellRenderer

```
public void setMenuCellRenderer(ListCellRenderer menuCellRenderer)
```

Determina o renderizador de células usado para renderizar elementos de menu para tematizar a aparência das opções do menu.

Parâmetros:

`menuCellRenderer` - o renderizador de células do menu

removeAllCommands

```
public void removeAllCommands()
```

Limpa os comandos do menu da barra de menu.

paint

```
public void paint(Graphics g)
```

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

Especificado por:

`paint` na interface `Animation`

Substituições:

`paint` na classe `Container`

Parâmetros:

`g` - os gráficos do componente

setScrollable

```
public void setScrollable(boolean scrollable)
```

O equivalente de chamar `setScrollableY` e `setScrollableX`.

Substituições:

`setScrollable` na classe `Container`

Parâmetros:

`scrollable` - se o componente deveria/poderia rolar no eixo X e Y

setVisible

```
public void setVisible(boolean visible)
```

Muda a visibilidade do componente.

Substituições:

`setVisible` na classe `Component`

Parâmetros:

`visible` - *true* se o componente estiver visível; caso contrário *false*

getTintColor

```
public Color getTintColor()
```

A tonalidade padrão para cor da tela, quando um diálogo ou menu é exibido.

Retorna:

a tonalidade de cor quando um diálogo ou menu é exibido

Relaciona-se com:

`setTintColor(java.awt.Color)`

setTintColor

```
public void setTintColor(Color tintColor)
```

A tonalidade padrão para cor da tela, quando um diálogo ou menu é exibido.

Parâmetros:

`tintColor` - a tonalidade de cor quando um diálogo ou menu é exibido

Relaciona-se com:

`getTintColor()`

setMenuTransitions

```
public void setMenuTransitions(Transition transitionIn,
                               Transition transitionOut)
```

Define as transições do menu para exibir/esconder o menu. Pode ser `null`.

Parâmetros:

`transitionIn` - a transição que deve ser executada quando o menu aparecer

`transitionOut` - a transição que deve ser executada quando o menu for compactado

onShow

```
protected void onShow()
```

Permite subclasses a ligar funcionalidade que ocorrem imediatamente após um diálogo ou forma específica aparecer na tela

sizeChanged

```
protected void sizeChanged(int w,
                           int h)
```

Esse método só é invocado quando o ambiente gráfico para o formulário recebe um evento de tamanho modificado. Esse método desencadeará uma re-planificação do formulário. Esse método obterá a chamada de retorno apenas se esse formulário for o atual.

Parâmetros:

`w` - a nova largura do Form

`h` - a nova altura do Form

hideNotify

```
protected void hideNotify()
```

Esse método só é invocado quando o ambiente gráfico para o formulário está oculto. Esse método não é chamado para eventos baseados em forma e geralmente é útil para comportamento baseado em suspender/resumir.

showNotify

```
protected void showNotify()
```

Esse método só é invocado quando o ambiente gráfico para o formulário for exibido. Esse método não é chamado para eventos baseados em forma e geralmente é útil para comportamento baseado em suspender/resumir.

showMenuDialog

```
protected Command showMenuDialog(Dialog menu)
```

Se um menu é implementado como um diálogo, esse método permite sobrepor a exibição do diálogo para personalizar o menu do diálogo de várias maneiras

Parâmetros:

`menu` - um diálogo contendo opções de menu que podem ser personalizadas

Retorna:

o comando selecionado pelo usuário no diálogo (não no menu), ou `Select` ou `Cancel`

24.16 Classe Graphics

24.16.1 Descrição da classe

`com.sun.dtv.lwuit`

```
java.lang.Object
  └com.sun.dtv.lwuit.Graphics
```

```
final public class Graphics
extends Object
```

Abstrai o contexto gráfico da plataforma subjacente, permitindo, então, que alcancemos portabilidade entre dispositivos MIDP e dispositivos CDC. Esta abstração simplifica e unifica as implementações de gráficos de várias plataformas. Para o uso de LWUIT dentro da especificação Java DTV, um objeto `Graphics` sempre encapsula uma instância de `java.awt.Graphics2D`. O acesso ao objeto encapsulado só é encorajado se isto for realmente necessário para lidar com um problema específico de integração entre LWUIT e AWT. Não deveria ser usado para tarefas comuns.

24.16.2 Índice de construtores

`Graphics``(Graphics2D g)`

Criar um objeto `Graphics` do LWUIT usando uma instância `Graphics` da biblioteca de gráficos interna a ser encapsulada.

24.16.3 Índice de métodos

`void` **`clearRect`**`(int x, int y, int width, int height)`

Limpa o retângulo especificado por preenchê-lo com a cor de fundo da superfície do desenho atual.

`void` **`clipRect`**`(int x, int y, int width, int height)`

Recorta o retângulo especificado fazendo uma intersecção com a região de recorte atual. Esse método pode apenas diminuir e nunca aumentar a região de recorte.

`void` **`copyArea`**`(int x, int y, int width, int height, int dx, int dy)`

Copia uma área do componente a uma distância especificada pelos parâmetros `dx` e `dy`.

`void` **`darkerColor`**`(int factor)`

Escurece suavemente a cor atual. Isso é útil para muitos efeitos visuais.

`void` **`draw3DRect`**`(int x, int y, int width, int height, boolean raised)`

Desenha um contorno destacado em 3D do retângulo especificado.

`void` **`drawArc`**`(int x, int y, int width, int height, int startAngle, int arcAngle)`

Desenha um arco circular ou elíptico baseado nos ângulos e na caixa delimitadora inseridos.

`void` **`drawBytes`**`(byte[] data, int offset, int length, int x, int y)`

Desenha o texto especificado pelo *array* de byte determinado usando as fontes e cores atuais desse contexto gráfico.

`void` **`drawChar`**`(char character, int x, int y)`

Desenha um dado caractere usando a fonte atual e a cor nas coordenadas x e y.

`void` **`drawChars`**`(char[] data, int offset, int length, int x, int y)`

Desenha o *array* de char determinado usando as fontes e cores atuais nas coordenadas x, y.

`void` **`drawImage`**`(Image img, int x, int y)`

Desenha a imagem de forma que sua coordenada superior esquerda corresponda a x/y.

`void` **`drawLine`**`(int x1, int y1, int x2, int y2)`

Desenha uma linha entre as 2 coordenadas X/Y.

`void` **`drawOval`**`(int x, int y, int width, int height)`

Desenha o contorno de uma forma oval nos limites do retângulo determinado.

`void` **`drawPolygon`**`(int[] xPoints, int[] yPoints, int nPoints)`

Desenha um polígono fechado definido pelos *array*s das coordenadas x e y.

`void` **`drawPolyline`**`(int[] xPoints, int[] yPoints, int nPoints)`

Desenha uma sequência de linhas conectadas definida pelos *array*s das coordenadas x e y.

`void` **`drawRect`**`(int x, int y, int width, int height)`

Desenha um retângulo nas coordenadas determinadas.

`void` **`drawRoundRect`**`(int x, int y, int width, int height, int arcWidth, int arcHeight)`

Desenha um retângulo com cantos arredondados nas coordenadas determinadas com o arcWidth/height combinando com os dois últimos argumentos respectivamente.

`void` **`drawString`**`(String str, int x, int y)`

Desenha um *string* usando as fontes e cores atuais nas coordenadas x, y.

`void` **`fill3DRect`**`(int x, int y, int width, int height, boolean raised)`

Preenche um retângulo 3D destacado preenchido com a cor atual.

`void` **`fill3DRect`**`(int x, int y, int width, int height, boolean raised, byte alpha)`

Preenche um retângulo 3D destacado preenchido com uma cor de preenchimento opcionalmente translúcida.

`void` **`fillArc`**`(int x, int y, int width, int height, int startAngle, int arcAngle)`

Preenche um arco circular ou elíptico baseado nos ângulos e na caixa delimitadora inseridos.

`void` **`fillArc`**`(int x, int y, int width, int height, int startAngle, int arcAngle, byte alpha)`

Preenche um arco circular ou elíptico baseado nos ângulos e na caixa delimitadora inseridos com uma cor opcionalmente translúcida.

`void` **`fillLinearGradient`**`(Color startColor, Color endColor, int x, int y, int width, int height, boolean horizontal)`

Desenha um gradiente linear nas coordenadas determinadas com as cores especificadas. Não considera o alpha ao desenhar o gradiente.

`void` **`fillOval`**`(int x, int y, int width, int height)`

Preenche uma forma oval delimitada pelo retângulo determinado com a cor atual.

`void` **`fillOval`**`(int x, int y, int width, int height, byte alpha)`

Preenche uma forma oval delimitada pelo retângulo determinado com uma cor de preenchimento opcionalmente translúcida.

`void` **`fillPolygon`**`(int[] xPoints, int[] yPoints, int nPoints)`

Preenche um polígono fechado definido pelos *array*s determinados das coordenadas x e y com a cor atual do contexto gráfico.

`void` **`fillPolygon`**`(int[] xPoints, int[] yPoints, int nPoints, byte alpha)`

Preenche um polígono fechado definido pelos *array*s determinados das coordenadas x e y com uma cor de preenchimento opcionalmente translúcida.

`void` **`fillRadialGradient`**`(Color startColor, Color endColor, int x, int y, int width, int height)`

Desenha um gradiente nas coordenadas determinadas com as cores especificadas. Não considera o alpha ao desenhar o gradiente.

`void **fillRect**(int x, int y, int width, int height)`

Preenche o retângulo a partir da posição determinada de acordo com a largura/altura menos 1 pixel, conforme a convenção em Java.

`void **fillRect**(int x, int y, int w, int h, byte alpha)`

Preenche um retângulo com uma cor de preenchimento opcionalmente translúcida.

`void **fillRoundRect**(int x, int y, int width, int height, int arcWidth, int arcHeight)`

Preenche um retângulo arredondado da mesma maneira que `drawRoundRect`.

`void **fillRoundRect**(int x, int y, int width, int height, int arcWidth, int arcHeight, byte alpha)`

Preenche um retângulo arredondado da mesma maneira que `drawRoundRect` com uma cor de preenchimento opcionalmente translúcida.

`void **fillTriangle**(int x1, int y1, int x2, int y2, int x3, int y3)`

Desenha um retângulo preenchido com as coordenadas determinadas.

`void **fillTriangle**(int x1, int y1, int x2, int y2, int x3, int y3, byte alpha)`

Desenha um triângulo preenchido com as coordenadas determinadas com uma cor de preenchimento opcionalmente translúcida.

`Graphics2D **getAWTGraphics**()`

Obtém o objeto java.awt.Graphics2D envolvido por esta instância Graphics.

`Rectangle **getClipBounds**()`

Retorna o retângulo delimitado da atual área de recorte.

`Rectangle **getClipBounds**(Rectangle r)`

Retorna o retângulo delimitado da atual área de recorte.

`int **getClipHeight**()`

Retorna a altura de recorte.

`int **getClipWidth**()`

Retorna a largura de recorte.

`int **getClipX**()`

Retorna a posição de recorte x.

`int **getClipY**()`

Retorna a posição de recorte y.

`Color **getColor**()`

Retorna a cor atual.

`Font **getFont**()`

Retorna a fonte usada com as chamadas do método drawString.

`int **getTranslateX**()`

Retorna o valor atual de conversão x.

`int **getTranslateY**()`

Retorna o valor atual de conversão y.

`boolean **hitClip**(int x, int y, int width, int height)`

Retorna *true* se a área retangular determinada puder cruzar com a atual área de recorte.

`void **lighterColor**(int factor)`

Esclarece suavemente a cor atual. Isso é útil para muitos efeitos visuais.

`void **setClip**(int x, int y, int width, int height)`

Atualiza a região de recorte para corresponder exatamente à região determinada.

`void **setColor**(Color color)`

Define a cor rgb atual ao mesmo tempo que ignora qualquer componente alpha em potencial dentro do valor da cor citado.

`void **setFont**(Font font)`

Define a fonte a ser usada com as chamadas do método drawString.

`void **translate**(int x, int y)`

Converte a localização X/Y para desenhar na superfície subjacente.

24.16.4 Detalhe dos construtores

Graphics

```
public Graphics(Graphics2D g)
```

Criar um objeto Graphics LWUIT usando uma instância Graphics da biblioteca subjacente Graphics a ser envolvida.

Parâmetros:

`g` – uma instância Graphics da biblioteca Graphics subjacente.

24.16.5 Detalhe dos métodos

translate

```
public void translate(int x,
                      int y)
```

Converte a localização X/Y para desenhar na superfície subjacente. A conversão é incremental para que o novo valor seja adicionado à atual conversão, e para redefinir a conversão temos que executar o comando `translate(-getTranslateX(), -getTranslateY())`

Parâmetros:

`x` – a coordenada x

`y` – a coordenada y

getTranslateX

```
public int getTranslateX()
```

Retorna o valor atual de conversão x.

Retorna:

um int representa o valor de conversão x

getTranslateY

`public int getTranslateY()`

Retorna o valor atual de conversão y.

Retorna:

um int representa o valor de conversão y

getColor

`public Color getColor()`

Retorna a cor atual.

Retorna:

a cor atual

Relaciona-se com:

`setColor(java.awt.Color)`

setColor

`public void setColor(Color color)`

Define a cor rgb atual ao mesmo tempo que ignora qualquer componente alpha em potencial dentro do valor da cor citado.

Parâmetros:

`color` - a cor a ser definida.

Relaciona-se com:

`getColor()`

getFont

`public Font getFont()`

Retorna a fonte usada com as chamadas do método drawString.

Retorna:

a fonte atual

Relaciona-se com:

`setFont(Font)`

setFont

`public void setFont(Font font)`

Define a fonte a ser usada com as chamadas do método drawString.

Parâmetros:

`font` - a fonte a ser definida

Relaciona-se com:

`getFont()`

getClipX

```
public int getClipX()
```

Retorna a posição de recorte x.

Retorna:

um int que representa o valor x do recorte

getClipY

```
public int getClipY()
```

Retorna a posição de recorte y.

Retorna:

um int que representa o valor y do recorte

getClipWidth

```
public int getClipWidth()
```

Retorna a largura de recorte.

Retorna:

um int que representa o valor da largura do recorte

getClipHeight

```
public int getClipHeight()
```

Retorna a altura de recorte.

Retorna:

um int representa o valor da altura do recorte

clipRect

```
public void clipRect(int x,
                     int y,
                     int width,
                     int height)
```

Recorta o retângulo especificado fazendo uma intersecção com a região de recorte atual. Esse método pode apenas diminuir e nunca aumentar a região de recorte.

Parâmetros:

`x` - a coordenada x do retângulo determinado

`y` - a coordenada y do retângulo determinado

`width` - a largura do retângulo determinado

`height` - a altura do retângulo determinado

setClip

```
public void setClip(int x,
                    int y,
                    int width,
                    int height)
```

Atualiza a região de recorte para corresponder exatamente à região determinada.

Parâmetros:

`x` - a coordenada x do retângulo determinado

`y` - a coordenada y do retângulo determinado

`width` - a largura do retângulo determinado

`height` - a altura do retângulo determinado

drawLine

```
public void drawLine(int x1,
                     int y1,
                     int x2,
                     int y2)
```

Desenha uma linha entre as 2 coordenadas X/Y.

Parâmetros:

`x1` – primeira posição x

`y1` – primeira posição y

`x2` – segunda posição x

`y2` – segunda posição y

fillRect

```
public void fillRect(int x,
                     int y,
                     int width,
                     int height)
```

Preenche o retângulo a partir da posição determinada de acordo com a largura/altura menos 1 pixel, conforme a convenção em Java.

Parâmetros:

`x` - a coordenada x do retângulo determinado

`y` - a coordenada y do retângulo determinado

`width` - a largura do retângulo determinado

`height` - a altura do retângulo determinado

fillRadialGradient

```
public void fillRadialGradient(Color startColor,
                               Color endColor,
                               int x,
                               int y,
                               int width,
```

```
                                int height)
```

Desenha um gradiente nas coordenadas determinadas com as cores especificadas. Não considera o alpha ao desenhar o gradiente. Note que um gradiente radial resultará em uma forma circular. Para criar um quadrado, use fillRect ou desenhe uma forma maior e recorte o tamanho apropriado.

Parâmetros:

`startColor` – a cor RGB inicial

`startColor` – a cor RGB final

`x` – a coordenada x

`y` – a coordenada y

`width` - a largura da região a ser preenchida

`height` – a altura da região a ser preenchida

fillLinearGradient

```
public void fillLinearGradient(Color startColor,
                               Color endColor,
                               int x,
                               int y,
                               int width,
                               int height,
                               boolean horizontal)
```

Desenha um gradiente linear nas coordenadas determinadas com as cores especificadas. Não considera o alpha ao desenhar o gradiente.

Parâmetros:

`startColor` – a cor RGB inicial

`startColor` – a cor RGB final

`x` – a coordenada x

`y` – a coordenada y

`width` - a largura da região a ser preenchida

`height` – a altura da região a ser preenchida

`horizontal` – indica se é um preenchimento horizontal ou vertical

drawRect

```
public void drawRect(int x,
                     int y,
                     int width,
                     int height)
```

Desenha um retângulo nas coordenadas determinadas.

Parâmetros:

`x` - a coordenada x do retângulo determinado

`y` - a coordenada y do retângulo determinado

`width` - a largura do retângulo determinado

`height` - a altura do retângulo determinado

drawRoundRect

```
public void drawRoundRect(int x,
                          int y,
                          int width,
                          int height,
                          int arcWidth,
                          int arcHeight)
```

Desenha um retângulo com cantos arredondados nas coordenadas determinadas com o arcWidth/height combinando com os dois últimos argumentos respectivamente.

Parâmetros:

`x` - a coordenada x do retângulo determinado

`y` - a coordenada y do retângulo determinado

`width` - a largura do retângulo determinado

`height` - a altura do retângulo determinado

`arcWidth` - a largura do arco do retângulo determinado

`arcHeight` - a altura do arco do retângulo determinado

fillRoundRect

```
public void fillRoundRect(int x,
                          int y,
                          int width,
                          int height,
                          int arcWidth,
                          int arcHeight)
```

Preenche um retângulo arredondado da mesma maneira que `drawRoundRect`.

Parâmetros:

`x` - a coordenada x do retângulo determinado

`y` - a coordenada y do retângulo determinado

`width` - a largura do retângulo determinado

`height` - a altura do retângulo determinado

`arcWidth` - a largura do arco do retângulo determinado

`arcHeight` - a altura do arco do retângulo determinado

Relaciona-se com:

`drawRoundRect()`

fillRoundRect

```
public void fillRoundRect(int x,
                          int y,
                          int width,
                          int height,
                          int arcWidth,
                          int arcHeight,
```

```
                              byte alpha)
```

Preenche um retângulo arredondado da mesma maneira que `drawRoundRect` com uma cor de preenchimento opcionalmente translúcida.

Parâmetros:

`x` - a coordenada x do retângulo determinado

`y` - a coordenada y do retângulo determinado

`width` - a largura do retângulo determinado

`height` - a altura do retângulo determinado

`arcWidth` - a largura do arco do retângulo determinado

`arcHeight` - a altura do arco do retângulo determinado

`alpha` - valor composto alpha

Relaciona-se com:

`drawRoundRect()`

fillArc

```
public void fillArc(int x,
                    int y,
                    int width,
                    int height,
                    int startAngle,
                    int arcAngle)
```

Preenche um arco circular ou elíptico baseado nos ângulos e na caixa delimitadora inseridos.

Parâmetros:

`x` - a coordenada x do arco determinado

`y` - a coordenada y do arco determinado

`width` - a largura do arco determinado

`height` - a altura do arco determinado

`startAngle` – o ângulo inicial do arco determinado

`arcAngle` - o ângulo final do arco determinado

fillArc

```
public void fillArc(int x,
                    int y,
                    int width,
                    int height,
                    int startAngle,
                    int arcAngle,
                    byte alpha)
```

Preenche um arco circular ou elíptico baseado nos ângulos e na caixa delimitadora inseridos com uma cor opcionalmente translúcida.

Parâmetros:

`x` - a coordenada x do arco determinado

`y` - a coordenada y do arco determinado

`width` - a largura do arco determinado

`height` - a altura do arco determinado

`startAngle` – o ângulo inicial do arco determinado

`arcAngle` - o ângulo final do arco determinado

`alpha` - valor composto alpha

drawArc

```
public void drawArc(int x,
                    int y,
                    int width,
                    int height,
                    int startAngle,
                    int arcAngle)
```

Desenha um arco circular ou elíptico baseado nos ângulos e na caixa delimitadora inseridos.

Parâmetros:

`x` - a coordenada x do arco determinado

`y` - a coordenada y do arco determinado

`width` - a largura do arco determinado

`height` - a altura do arco determinado

`startAngle` – o ângulo inicial do arco determinado

`arcAngle` - o ângulo final do arco determinado

drawString

```
public void drawString(String str,
                       int x,
                       int y)
```

Desenha um *string* usando as fontes e cores atuais nas coordenadas x, y. A fonte é desenhada a partir da posição superior e não da linha de base.

Parâmetros:

`str` – a *string*

`x` - a coordenada x da *string*

`y` - a coordenada y da *string*

drawChar

```
public void drawChar(char character,
                     int x,
                     int y)
```

Desenha um dado caractere usando a fonte atual e a cor nas coordenadas x e y. A fonte é desenhada a partir da posição superior e não da linha de base.

Parâmetros:

`character` – um char

`x` – coordenada x

`y` – coordenada y

drawChars

```
public void drawChars(char[] data,
                      int offset,
                      int length,
                      int x,
                      int y)
```

Desenha o *array* de char determinado usando as fontes e cores atuais nas coordenadas x, y. A fonte é desenhada a partir da posição superior e não da linha de base.

Parâmetros:

`data` – o *array* de char determinado

`offset` – compensação do *array*

`length` – comprimento do *array*

`x` – coordenada x da posição

`y` - coordenada y da posição

drawImage

```
public void drawImage(Image img,
                      int x,
                      int y)
```

Desenha a imagem de forma que sua coordenada superior esquerda corresponda a x/y.

Parâmetros:

`img` – um objeto Image

`x` – coordenada x

`y` – coordenada y

fillTriangle

```
public void fillTriangle(int x1,
                         int y1,
                         int x2,
                         int y2,
                         int x3,
                         int y3)
```

Desenha um retângulo preenchido com as coordenadas determinadas.

Parâmetros:

`x1` – coordenada x do ponto 1

`y1` – coordenada y do ponto 1

`x2` – coordenada x do ponto 2

`y2` – coordenada y do ponto 2

`x3` – coordenada x do ponto 3

`Y3` – coordenada y do ponto 3

fillTriangle

```
public void fillTriangle(int x1,
                         int y1,
                         int x2,
                         int y2,
                         int x3,
                         int y3,
                         byte alpha)
```

Desenha um triângulo preenchido com as coordenadas determinadas com uma cor de preenchimento opcionalmente translúcida.

Parâmetros:

`x1` – coordenada x do ponto 1

`y1` – coordenada y do ponto 1

`x2` – coordenada x do ponto 2

`y2` – coordenada y do ponto 2

`x3` – coordenada x do ponto 3

`Y3` – coordenada y do ponto 3

`alpha` - valor composto alpha

fillRect

```
public void fillRect(int x,
                     int y,
                     int w,
                     int h,
                     byte alpha)
```

Preenche um retângulo com uma cor de preenchimento opcionalmente translúcida.

Parâmetros:

`x` – coordenada x do canto superior esquerdo

`y` – coordenada y do canto superior esquerdo

`w` – a largura

`h` - a altura

`alpha` - valor composto alpha

lighterColor

```
public void lighterColor(int factor)
```

Esclarece suavemente a cor atual. Isso é útil para muitos efeitos visuais.

Parâmetros:

`factor` – o grau de clareamento de uma cor por canal; um número de 1 a 255

darkerColor

```
public void darkerColor(int factor)
```

Escurece suavemente a cor atual. Isso é útil para muitos efeitos visuais.

Parâmetros:

`factor` – o grau de clareamento de uma cor por canal; um número de 1 a 255

getAWTGraphics

```
public Graphics2D getAWTGraphics()
```

Obtém o objeto java.awt.Graphics2D envolvido por esta instância Graphics.

Retorna:

o objeto `java.awt.Graphics2D` envolvido

clearRect

```
public void clearRect(int x,
                      int y,
                      int width,
                      int height)
```

Limpa o retângulo especificado por preenchê-lo com a cor de fundo da superfície do desenho atual.

NOTA Na implementação:

Delega ao objeto AWT Graphics2D envolvido.

Parâmetros:

`x` - a coordenada x do retângulo determinado

`y` - a coordenada y do retângulo determinado

`width` - a largura do retângulo determinado

`height` - a altura do retângulo determinado

copyArea

```
public void copyArea(int x,
                     int y,
                     int width,
                     int height,
                     int dx,
                     int dy)
```

Copia uma área do componente a uma distância especificada pelos parâmetros `dx` e `dy`.

NOTA Na implementação:

Delega ao objeto AWT Graphics2D envolvido.

Parâmetros:

`x` - a coordenada x da área determinada

`y` - a coordenada y da área determinada

`width` - a largura da área determinada

`height` - a altura da área determinada

`dx` – a coordenada x da distância

`dy` - a coordenada y da distância

draw3DRect

```
public void draw3DRect(int x,
                       int y,
                       int width,
                       int height,
                       boolean raised)
```

Desenha um contorno destacado em 3D do retângulo especificado.

NOTA Na implementação:

Delega ao objeto AWT Graphics2D envolvido.

Parâmetros:

`x` - a coordenada x do retângulo determinado

`y` - a coordenada y do retângulo determinado

`width` - a largura do retângulo determinado

`height` - a altura do retângulo determinado

`raised` – *true* se o retângulo parecer estar em alto relevo na superfície, ou *false* se parecer estar em baixo relevo na superfície

drawBytes

```
public void drawBytes(byte[] data,
                      int offset,
                      int length,
                      int x,
                      int y)
```

Desenha o texto especificado pelo *array* de byte determinado usando as fontes e cores atuais desse contexto gráfico.

NOTA Na implementação:

Delega ao objeto AWT Graphics2D envolvido.

Parâmetros:

`data` - o *array* de byte determinado

`offset` – a compensação do *array*

`length` – o comprimento do *array*

`x` – a coordenada x da posição

`y` - a coordenada y da posição

drawOval

```
public void drawOval(int x,
                     int y,
                     int width,
                     int height)
```

Desenha o contorno de uma forma oval nos limites do retângulo determinado.

NOTA Na implementação:

Delega ao objeto AWT Graphics2D envolvido.

Parâmetros:

`x` - a coordenada x do retângulo determinado

`y` - a coordenada y do retângulo determinado

`width` - a largura do retângulo determinado

`height` - a altura do retângulo determinado

drawPolygon

```
public void drawPolygon(int[] xPoints,
                        int[] yPoints,
                        int nPoints)
```

Desenha um polígono fechado definido pelos *array*s das coordenadas x e y.

NOTA Na implementação:

Delega ao objeto AWT Graphics2D envolvido.

Parâmetros:

`xPoints` – *array* das coordenadas x

`yPoints` – *array* das coordenadas y

`nPoints` – número de pontos determinados

drawPolyline

```
public void drawPolyline(int[] xPoints,
                         int[] yPoints,
                         int nPoints)
```

Desenha uma sequência de linhas conectadas definida pelos *array*s das coordenadas x e y.

NOTA Na implementação:

Delega ao objeto AWT Graphics2D envolvido.

Parâmetros:

`xPoints` – *array* das coordenadas x

`yPoints` – *array* das coordenadas y

`nPoints` – número de pontos determinados

fill3DRect

```
public void fill3DRect(int x,
                       int y,
                       int width,
                       int height,
                       boolean raised)
```

Preenche um retângulo 3D destacado preenchido com a cor atual.

NOTA Na implementação:

Delega ao objeto AWT Graphics2D envolvido.

Parâmetros:

`x` - a coordenada x do retângulo determinado

`y` - a coordenada y do retângulo determinado

`width` - a largura do retângulo determinado

`height` - a altura do retângulo determinado

`raised` – *true* se o retângulo parecer estar em alto relevo na superfície, ou *false* se parecer estar em baixo relevo na superfície

fill3DRect

```
public void fill3DRect(int x,
                       int y,
                       int width,
                       int height,
                       boolean raised,
                       byte alpha)
```

Preenche um retângulo 3D destacado preenchido com uma cor de preenchimento opcionalmente translúcida.

NOTA Na implementação:

Delega ao objeto AWT Graphics2D envolvido.

Parâmetros:

`x` - a coordenada x do retângulo determinado

`y` - a coordenada y do retângulo determinado

`width` - a largura do retângulo determinado

`height` - a altura do retângulo determinado

`raised` – *true* se o retângulo parecer estar em alto relevo na superfície, ou *false* se parecer estar em baixo relevo na superfície

`alpha` - valor composto alpha

fillOval

```
public void fillOval(int x,
                     int y,
                     int width,
                     int height)
```

Preenche uma forma oval delimitada pelo retângulo determinado com a cor atual.

NOTA Na implementação:

Delega ao objeto AWT Graphics2D envolvido.

Parâmetros:

`x` - a coordenada x do retângulo determinado

`y` - a coordenada y do retângulo determinado

`width` - a largura do retângulo determinado

`height` - a altura do retângulo determinado

fillOval

```
public void fillOval(int x,
                     int y,
                     int width,
                     int height,
                     byte alpha)
```

Preenche uma forma oval delimitada pelo retângulo determinado com uma cor de preenchimento opcionalmente translúcida.

NOTA Na implementação:

Delega ao objeto AWT Graphics2D envolvido.

Parâmetros:

`x` - a coordenada x do retângulo determinado

`y` - a coordenada y do retângulo determinado

`width` - a largura do retângulo determinado

`height` - a altura do retângulo determinado

`alpha` - valor composto alpha

fillPolygon

```
public void fillPolygon(int[] xPoints,
                        int[] yPoints,
                        int nPoints)
```

Preenche um polígono fechado definido pelos *array*s determinados das coordenadas x e y com a cor atual do contexto gráfico.

NOTA Na implementação:

Delega ao objeto AWT Graphics2D envolvido.

Parâmetros:

`xPoints` – *array* das coordenadas x

`yPoints` – *array* das coordenadas y

`nPoints` – número de pontos determinados

fillPolygon

```
public void fillPolygon(int[] xPoints,
                        int[] yPoints,
                        int nPoints,
                        byte alpha)
```

Preenche um polígono fechado definido pelos *array*s determinados das coordenadas x e y com uma cor de preenchimento opcionalmente translúcida.

NOTA Na implementação:

Delega ao objeto AWT Graphics2D envolvido.

Parâmetros:

`xPoints` – *array* das coordenadas x

`yPoints` – *array* das coordenadas y

`nPoints` – número de pontos determinados

`alpha` - valor composto alpha

getClipBounds

```
public Rectangle getClipBounds()
```

Retorna o retângulo delimitado da atual área de recorte.

NOTA Na implementação:

Delega ao objeto AWT Graphics2D envolvido.

Retorna:

o retângulo delineado

getClipBounds

```
public Rectangle getClipBounds(Rectangle r)
```

Retorna o retângulo delimitado da atual área de recorte. Um retângulo existente é usado em vez de estabelecer um novo.

NOTA Na implementação:

Delega ao objeto AWT Graphics2D envolvido.

Parâmetros:

`r` – o retângulo determinado existente

Retorna:

o retângulo delineado

hitClip

```
public boolean hitClip(int x,
                       int y,
                       int width,
                       int height)
```

Retorna *true* se a área retangular determinada puder cruzar com a atual área de recorte.

NOTA Na implementação:

Delega ao objeto AWT Graphics2D envolvido.

Parâmetros:

`x` - a coordenada x do retângulo determinado

`y` - a coordenada y do retângulo determinado

`width` - a largura do retângulo determinado

`height` - a altura do retângulo determinado

Retorna:

`true` se a área determinada cruza com a área de recorte, caso contrário, *`false`*

24.17 Classe Image

24.17.1 Descrição da classe

`com.sun.dtv.lwuit`

```
java.lang.Object
  └com.sun.dtv.lwuit.Image
```

Subclasses diretas conhecidas

StaticAnimation

public class Image

extends Object

Abstrai as imagens da plataforma subjacente, permitindo que sejam tratadas como um objeto uniforme. Na versão Java DTV do LWUIT, uma imagem sempre envolve uma instância de `java.awt.Image`. Como consequência, uma funcionalidade adicional foi adicionada para evidenciar a instância `java.awt.Image` envolvida quando necessário. O acesso ao objeto envolvido só é encorajado se isto for realmente necessário para lidar com um problema específico de integração entre LWUIT e AWT. Não deveria ser usado para tarefas comuns.

24.17.2 Índice de construtores

Image(Image image)

Cria uma nova instância de Imagedo `java.awt.Image` a ser envolvido.

24.17.3 Índice de métodos

static Image **createImage**(byte[] bytes, int offset, int len)

Cria uma imagem a partir dos dados de *array* de byte determinados.

static Image **createImage**(int width, int height)

Cria uma imagem buferizada.

static Image **createImage**(int[] rgb, int width, int height)

Cria uma imagem a partir de uma imagem RGB.

static Image **createImage**(Image image)

Cria uma imagem a partir do `java.awt.Image` a ser envolvido.

static Image **createImage**(InputStream stream)

Cria uma imagem a partir de um `InputStream`.

static Image **createImage**(String path)

Cria uma imagem a partir do caminho determinado.

Image **getAWTImage**()

Obtém o java.awt.Image envolvido por esta Image.

Graphics **getGraphics**()

Se esta for uma imagem alterável, um objeto gráfico que permite desenhar nele é retornado.

int **getHeight**()

Retorna a altura da imagem.

int[] **getRGB**()

Retorna o conteúdo dessa imagem como um *array* ARGB recém-criado.

int **getWidth**()

Retorna a largura da imagem.

boolean **isAnimation**()

Retorna *true* se essa for uma imagem animada.

boolean **isOpaque**()

Indica se essa imagem é opaca ou não.

```
Image modifyAlpha(byte alpha)
```

Cria uma nova instância de imagem com o canal alpha de pixels opacos/translúcidos na imagem usando o novo valor alpha.

```
Image modifyAlpha(byte alpha, Color removeColor)
```

Cria uma nova instância de imagem com o canal alpha de pixels opacos/translúcidos na imagem usando o novo valor alpha.

```
Image rotate(int degrees)
```

Retorna uma instância dessa imagem girada pelo número de graus determinado.

```
Image scaled(int width, int height)
```

Retorna uma versão redimensionada desta imagem usando a largura e altura determinadas. É um algoritmo rápido que preserva a informação de translucidez.

```
Image scaledHeight(int height)
```

Redimensiona a imagem para a altura determinada enquanto atualiza a largura com base na proporção de tela da altura.

```
Image scaledSmallerRatio(int width, int height)
```

Redimensiona a imagem enquanto mantém a proporção de tela com a imagem de menor tamanho.

```
Image scaledWidth(int width)
```

Redimensiona a imagem para a largura determinada enquanto atualiza a altura com base na proporção de tela da largura.

```
Image subImage(int x, int y, int width, int height, boolean processAlpha)
```

Remove uma subimagem da imagem determinada permitindo dividir uma única imagem grande em várias pequenas imagens em RAM. Isso, na verdade, cria uma versão autônoma da imagem para uso.

24.17.4 Detalhe dos construtores

Imagem

```
public Image(Image image)
```

Cria uma nova instância de Imagedo `java.awt.Image` a ser envolvido.

Parâmetros:

`image` – a `java.awt.Image` a ser envolvida

24.17.5 Detalhe dos métodos

subImage

```
public Image subImage(int x,
                      int y,
                      int width,
                      int height,
                      boolean processAlpha)
```

Remove uma subimagem da imagem determinada permitindo dividir uma única imagem grande em várias pequenas imagens em RAM. Isso, na verdade, cria uma versão autônoma da imagem para uso.

Parâmetros:

`x` – coordenada x do canto superior esquerdo

`y` – coordenada y do canto superior esquerdo

`width` – a largura de imagens internas

`height` – a altura de imagens internas

`processAlpha` - valor alphaComposite

Retorna:

Um *array* de todas as imagens possíveis que podem ser criadas a partir da fonte

Lança:

`IOException`

rotate

```
public Image rotate(int degrees)
```

Retorna uma instância dessa imagem girada pelo número de graus determinado. Por padrão, divisões de ângulos de 90 graus são suportadas. Todo o restante é dependente de implementação. Este método assume uma imagem quadrada. Note que é ineficaz na atual implementação alterar para ângulos não quadrados,

Ex.: alterar uma imagem para 45, 90 e 135 graus é ineficiente. Use girar para 45, 90 e então girar para 45 para que outro 90 graus alcance o mesmo efeito com menos memória.

Parâmetros:

`degrees` – Um grau no ângulo direito deve ser maior que 0 e ter até 359 graus

Retorna:

nova instância de imagem com a rotação mais próxima possível

modifyAlpha

```
public Image modifyAlpha(byte alpha)
```

Cria uma nova instância de imagem com o canal alpha de pixels opacos/translúcidos na imagem usando o novo valor alpha. Os pixels transparentes (alpha == 0) permanecem transparentes. Todos os outros pixels terão o novo valor alpha.

Parâmetros:

`alpha` - Novo valor para todo canal alpha

Retorna:

Imagem translúcida/opaca com base no valor alpha e nos pixels dessa imagem

modifyAlpha

```
public Image modifyAlpha(byte alpha,
                         Color removeColor)
```

Cria uma nova instância de imagem com o canal alpha de pixels opacos/translúcidos na imagem usando o novo valor alpha. Os pixels transparentes (alpha == 0) permanecem transparentes. Todos os outros pixels terão o novo valor alpha.

Parâmetros:

`alpha` - Novo valor para todo canal alpha

`removeColor` – os pixels que combinam com esta cor passam a ser transparentes (alpha channel ignored)

Retorna:

Imagem translúcida/opaca com base no valor alpha e nos pixels dessa imagem

createImage

```
public static Image createImage(String path)
                         throws IOException
```

Cria uma imagem a partir do caminho determinado.

Parâmetros:

`path` – o caminho onde encontrar informações da imagem

Retorna:

objeto Image recém-criado

Lança:

`IOException`

createImage

```
public static Image createImage(InputStream stream)
                         throws IOException
```

Cria uma imagem a partir de um `InputStream`.

Parâmetros:

`stream` – um `InputStream` determinado

Retorna:

um objeto Image

Lança:

`IOException`

createImage

```
public static Image createImage(int[] rgb,
                                int width,
                                int height)
```

Cria uma imagem a partir de uma imagem RGB.

Parâmetros:

`rgb` – dados do *array* da imagem RGB

`width` – a largura da imagem

`height` – a altura da imagem

Retorna:

uma imagem de uma imagem RGB

createImage

```
public static Image createImage(int width,
```

```
                                int height)
```

Cria uma imagem buferizada.

Parâmetros:

`width` – a largura da imagem

`height` – a altura da imagem

Retorna:

uma imagem em uma dimensão de largura e altura determinada

createImage

```
public static Image createImage(byte[] bytes,
                                int offset,
                                int len)
```

Cria uma imagem a partir dos dados de *array* de byte determinados.

Parâmetros:

`bytes` – o *array* de dados da imagem em um formato de imagem suportado

`offset` – a compensação do início dos dados no *array*

`len` – o comprimento dos dados no *array*

Retorna:

um objeto Image

createImage

```
public static Image createImage(Image image)
```

Cria uma imagem a partir do `java.awt.Image` a ser envolvido.

Parâmetros:

`image` – a `java.awt.Image` a ser envolvida

Retorna:

a instância criada da Image

getGraphics

```
public Graphics getGraphics()
```

Se esta for uma imagem alterável, um objeto gráfico que permite desenhar nele é retornado.

Retorna:

Objeto gráfico que nos permite manipular o conteúdo de uma imagem alterável

getWidth

```
public int getWidth()
```

Retorna a largura da imagem.

Retorna:

a largura da imagem

getHeight

```
public int getHeight()
```

Retorna a altura da imagem.

Retorna:

a altura da imagem

getRGB

```
public int[] getRGB()
```

Retorna o conteúdo dessa imagem como um *array* ARGB recém-criado.

Retorna:

nova instância do *array* contendo os dados ARGB dentro dessa imagem

scaledWidth

```
public Image scaledWidth(int width)
```

Redimensiona a imagem para a largura determinada enquanto atualiza a altura com base na proporção de tela da largura.

Parâmetros:

`width` – a largura determinada da imagem nova

Retorna:

um objeto Image

scaledHeight

```
public Image scaledHeight(int height)
```

Redimensiona a imagem para a altura determinada enquanto atualiza a largura com base na proporção de tela da altura.

Parâmetros:

`height` – a altura determinada da imagem nova

Retorna:

um objeto Image

scaledSmallerRatio

```
public Image scaledSmallerRatio(int width,
                                int height)
```

Redimensiona a imagem enquanto mantém a proporção de tela com a imagem de menor tamanho.

Parâmetros:

`width` – a largura determinada da imagem nova

`height` – a altura determinada da imagem nova

Retorna:

um objeto Image

scaled

```
public Image scaled(int width,
                    int height)
```

Retorna uma versão redimensionada desta imagem usando a largura e altura determinadas. É um algoritmo rápido que preserva a informação de translucidez.

Parâmetros:

`width` – largura para o redimensionamento

`height` – a altura da imagem redimensionada

Retorna:

instância da nova imagem redimensionada para a altura e largura determinadas

getAWTImage

```
public Image getAWTImage()
```

Obtém o java.awt.Image envolvido por esta Image.

Retorna:

a java.awt.Image envolvida

isAnimation

```
public boolean isAnimation()
```

Retorna *true* se essa for uma imagem animada.

Retorna:

true se a imagem for uma animação

isOpaque

```
public boolean isOpaque()
```

Indica se essa imagem é opaca ou não.

Retorna:

true se a imagem for completamente opaca, o que permite algumas grandes otimizações

24.18 Classe Label

24.18.1 Descrição da classe

com.sun.dtv.lwuit

```
java.lang.Object
  └com.sun.dtv.lwuit.Component
      └com.sun.dtv.lwuit.Label
```

Todas as interfaces implementadas

`Animated, Animation, MatteEnabled, StyleListener, ViewOnlyComponent`

Subclasses diretas conhecidas

`Button, DefaultListCellRenderer`

```
public class Label
extends Component
implements ViewOnlyComponent
```

Permite a exibição de etiquetas e imagens com diferentes opções de alinhamento, essa classe é uma classe base para vários componentes, permitindo-lhes a declarar o visual do ícone/alinhamento de uma maneira similar.

Campos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`BOTTOM, BRB_CENTER_OFFSET, BRB_CONSTANT_ASCENT, BRB_CONSTANT_DESCENT, BRB_OTHER, CENTER, LEFT, RIGHT, TOP`

Campos herdados da interface `com.sun.dtv.ui.`**`Animated`**

`ALTERNATING, LOOP, REPEATING`

Campos herdados da interface `com.sun.dtv.ui.`**`ViewOnlyComponent`**

`HORIZONTAL_ALIGN_CENTER, HORIZONTAL_ALIGN_JUSTIFIED, HORIZONTAL_ALIGN_LEFT, HORIZONTAL_ALIGN_RIGHT, SCALE_ASPECT_PROOF, SCALE_NO, SCALE_NO_ASPECT_PROOF, STATE_DISABLED, STATE_ENABLED, VERTICAL_ALIGN_BOTTOM, VERTICAL_ALIGN_CENTER, VERTICAL_ALIGN_JUSTIFIED, VERTICAL_ALIGN_TOP`

24.18.2 Índice de construtores

`Label()`

Cria um label vazio.

`Label``(Image icon)`

Cria um novo label com o ícone determinado.

`Label``(String text)`

Cria um novo label com um *string* do texto determinado, justificado à esquerda.

24.18.3 Índice de métodos

`boolean` **`animate`**`()`

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*.

`int` **`getAlignment`**`()`

Retorna o alinhamento do Label.

`Image[] **getAnimateContent**(int state)`

Retorna o conteúdo animado desse componente, dependendo do estado atual.

`int **getBaselineResizeBehavior**()`

Retorna uma constante indicando como a base varia com o tamanho do componente.

`int **getGap**()`

Retorna o intervalo de pixels entre o ícone/texto aos limites do Label.

`Image **getGraphicContent**(int state)`

Retorna conteúdo gráfico desse componente, dependendo do estado atual.

`int **getHorizontalAlignment**()`

Restaura o alinhamento horizontal de qualquer conteúdo baseado em estado nesse componente.

`Image **getIcon**()`

Retorna os ícones dos labels.

`int **getInteractionState**()`

Retorna o estado de interação atual do componente.

`Matte **getMatte**()`

Retorna o `Matte` atualmente associado ao componente implementando essa interface.

`int **getScalingMode**()`

Restaura o modo de redimensionamento desse componente.

`int **getShiftText**()`

`Getter` simples para retornar a quantidade de pixels para alterar o texto dentro do *label*.

`String **getText**()`

Retorna o texto do label.

`String **getTextContent**(int state)`

Retorna conteúdo de texto desse componente, dependendo do estado atual.

`TextLayoutManager **getTextLayoutManager**()`

Obtém o gerenciador de layout de texto que está em uso para o layout de texto nesse componente.

`int **getTextPosition**()`

Retorna a posição do texto relativo ao ícone.

`int **getVerticalAlignment**()`

Restaura o alinhamento vertical de qualquer conteúdo com base em estado nesse componente.

`boolean **isDoubleBuffered**()`

Retorna *true* se o buffering duplo estiver disponível e ativado.

`boolean **isEndsWith3Points**()`

`Getter` simples.

`boolean **isOpaque**()`

Retorna *true* se toda a área do componente (conforme retornado pelo método `com.sun.dtv.lwuit.Component#getBounds`, for opaco.

`boolean **isTickerEnabled**()`

Esse método retorna *true* se o ticker estiver habilitado nesse Label.

`boolean` **`isTickerRunning`**`()`

Retorna *true* se o ticker estiver funcionando.

`void` **`paint`**`(Graphics g)`

Método para pintar o componente.

`void` **`processEvent`**`(AWTEvent event)`

Lida com o `AWTEvent` determinado.

`void` **`setAlignment`**`(int align)`

Método de comodidade, semelhante ao `#setHorizontalAlignment`

`void` **`setAnimateContent`**`(Image[] images, int state)`

Designa um *array* de conteúdo gráfico para ser usado em animação desse componente, que é dependente do estado.

`void` **`setEndsWith3Points`**`(boolean endsWith3Points)`

Se o texto do Label for muito longo, inclua o texto no *widget* e adicione "..." pontos no final.

`void` **`setGap`**`(int gap)`

Define o intervalo de pixels entre o ícone/texto aos limites do Label.

`void` **`setGraphicContent`**`(Image image, int state)`

Designa o conteúdo gráfico a esse componente que é dependente do estado.

`void` **`setHorizontalAlignment`**`(int alignment)`

Define o alinhamento horizontal para qualquer conteúdo baseado no estado desse componente.

`void` **`setIcon`**`(Image icon)`

Define o ícone do Label.

`void` **`setInteractionState`**`(int state)`

Define o estado atual do componente com relação a interação.

`void` **`setMatte`**`(Matte matte)`

Adiciona um `Matte` ao componente implementando essa interface para possibilitar composição de matte.

`void` **`setScalingMode`**`(int scaling)`

Define o modo de redimensionamento para esse componente.

`void` **`setShiftText`**`(int shiftText)`

Esse método altera o texto de sua posição em pixels.

`void` **`setText`**`(String text)`

Define o texto do Label.

`void` **`setTextContent`**`(String text, int state)`

Designa o conteúdo do texto a esse componente que é dependente do estado.

`void` **`setTextLayoutManager`**`(TextLayoutManager manager)`

Define o gerenciador de layout de texto que deve ser usado para o layout de texto nesse componente.

`void` **`setTextPosition`**`(int textPosition)`

Define a posição do texto em relação ao ícone, se houver.

`void` **`setTickerEnabled`**`(boolean tickerEnabled)`

Define o Label para permitir a sinalização do texto.

`void` **`setVerticalAlignment`**`(int valign)`

Define o alinhamento vertical de qualquer conteúdo com base em estado desse componente.

`void` **`startTicker`**`(long delay, boolean rightToLeft)`

Esse método iniciará o sinalizador de texto.

`void` **`stopTicker`**`()`

Para o sinalizador de texto.

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`addFocusListener, calcPreferredSize, contains, deinitialize, getAbsoluteX, getAbsoluteY, getAnimationMode, getBaseline, getBottomGap, getBounds, getClientProperty, getComponentForm, getDelay, getHeight, getNextFocusDown, getNextFocusLeft, getNextFocusRight, getNextFocusUp, getParent, getPosition, getPreferredSize, getRepetitionMode, getScrollAnimationSpeed, getScrollX, getScrollY, getSideGap, getStyle, getUIID, getWidth, getX, getY, handlesInput, hasFocus, initialize, isEnabled, isFocusable, isFocusPainted, isInitialized, isRunning, isScrollableX, isScrollableY, isScrollVisible, isSmoothScrolling, isVisible, jumpTo, keyPressed, keyReleased, keyRepeated, longKeyPress, paintBackgrounds, paintComponent, paintComponent, pointerDragged, pointerPressed, pointerReleased, putClientProperty, refreshTheme, removeFocusListener, repaint, repaint, requestFocus, scrollRectToVisible, setAnimationMode, setCellRenderer, setDelay, setEnabled, setFocus, setFocusable, setFocusPainted, setHandlesInput, setHeight, setInitialized, setIsScrollVisible, setNextFocusDown, setNextFocusLeft, setNextFocusRight, setNextFocusUp, setPreferredSize, setRepetitionMode, setScrollAnimationSpeed, setScrollX, setScrollY, setShouldCalcPreferredSize, setSize, setSmoothScrolling, setStyle, setUIID, setVisible, setWidth, setX, setY, start, stop, styleChanged, toString`

24.18.4 Detalhe dos construtores

Label

`public Label(String text)`

Cria um novo label com um *string* do texto determinado, justificado à esquerda.

Parâmetros:

`text` – a *string* que o Label apresenta.

Label

`public Label()`

Cria um label vazio.

Label

`public Label(Image icon)`

Cria um novo label com o ícone determinado.

Parâmetros:

`icon` – a imagem que o label apresenta.

24.18.5 Detalhe dos métodos

getBaselineResizeBehavior

`public int getBaselineResizeBehavior()`

Retorna uma constante indicando como a base varia com o tamanho do componente.

Substituições:

`getBaselineResizeBehavior` na classe `Component`

Retorna:

um dos `BRB_CONSTANT_ASCENT, BRB_CONSTANT_DESCENT, BRB_CENTER_OFFSET` ou `BRB_OTHER`

setText

`public void setText(String text)`

Define o texto do Label. Método de comodidade, semelhante ao `#setTextContent.`

Parâmetros:

`text` – a *string* que o Label apresenta.

Relaciona-se com:

`getText()`

getText

`public String getText()`

Retorna o texto do label. Método de comodidade, semelhante ao `#getTextContent.`

Retorna:

o texto do label

Relaciona-se com:

`setText(java.lang.String)`

setIcon

`public void setIcon(Image icon)`

Define o ícone do Label. Método de comodidade, semelhante ao `#setGraphicsContent.`

Parâmetros:

`icon` – a imagem que o label apresenta.

Relaciona-se com:

`getIcon()`

getIcon

`public Image getIcon()`

Retorna os ícones dos labels. Método de comodidade, semelhante ao `#getGraphicsContent`.

Retorna:

o ícone dos labels

Relaciona-se com:

`setIcon(com.sun.dtv.lwuit.Image)`

setAlignment

```
public void setAlignment(int align)
```

Método de comodidade, semelhante ao `#setHorizontalAlignment`.

setVerticalAlignment

```
public void setVerticalAlignment(int valign)
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Define o alinhamento vertical de qualquer conteúdo com base em estado desse componente.

Especificado por:

`setVerticalAlignment` na interface `ViewOnlyComponent`

getVerticalAlignment

```
public int getVerticalAlignment()
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Restaura o alinhamento vertical de qualquer conteúdo com base em estado nesse componente.

Especificado por:

`getVerticalAlignment` na interface `ViewOnlyComponent`

Retorna:

O modo de alinhamento horizontal atual, um de `ViewOnlyComponent.VERTICAL_ALIGN_TOP`, `ViewOnlyComponent.VERTICAL_ALIGN_CENTER`, `ViewOnlyComponent.VERTICAL_ALIGN_BOTTOM` ou `ViewOnlyComponent.VERTICAL_ALIGN_JUSTIFIED`.

getAlignment

```
public int getAlignment()
```

Retorna o alinhamento do Label. Método de comodidade, semelhante ao `#getHorizontalAlignment`.

Retorna:

o modo de alinhamento atual, um de `ViewOnlyComponent.HORIZONTAL_ALIGN_LEFT`, `ViewOnlyComponent.HORIZONTAL_ALIGN_CENTER`, `ViewOnlyComponent.HORIZONTAL_ALIGN_RIGHT` ou `ViewOnlyComponent.HORIZONTAL_ALIGN_JUSTIFIED`.

Relaciona-se com:

`setAlignment(int)`

setTextPosition

```
public void setTextPosition(int textPosition)
```

Define a posição do texto em relação ao ícone, se houver.

Parâmetros:

`textPosition` – valor do alinhamento (LEFT, RIGHT, BOTTOM ou TOP)

Relaciona-se com:

`getTextPosition()`, `Component.LEFT`, `Component.RIGHT`, `Component.BOTTOM`, `Component.TOP`

getTextPosition

```
public int getTextPosition()
```

Retorna a posição do texto relativo ao ícone.

Retorna:

A posição do texto em relação ao ícone, um de: LEFT, RIGHT, BOTTOM, TOP

Relaciona-se com:

`setTextPosition(int)`, `Component.LEFT`, `Component.RIGHT`, `Component.BOTTOM`, `Component.TOP`

setGap

```
public void setGap(int gap)
```

Define o intervalo de pixels entre o ícone/texto aos limites do Label.

Parâmetros:

`gap` – o intervalo de pixels

Relaciona-se com:

`getGap()`

getGap

```
public int getGap()
```

Retorna o intervalo de pixels entre o ícone/texto aos limites do Label.

Retorna:

o intervalo de pixels entre o ícone/texto aos limites do Label

Relaciona-se com:

`setGap(int)`

paint

```
public void paint(Graphics g)
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Método para pintar o componente.

Especificado por:

`paint` na interface `ViewOnlyComponent`

`paint` na interface `Animation`

Substituições:

`paint` na classe `Component`

Parâmetros:

`g` - o contexto gráfico a ser usado para pintar.

getShiftText

```
public int getShiftText()
```

Getter simples para retornar a quantidade de pixels para alterar o texto dentro do Label.

Retorna:

número de pixels para alterar

Relaciona-se com:

`setShiftText(int)`

setShiftText

```
public void setShiftText(int shiftText)
```

Esse método altera o texto de sua posição em pixels. O valor pode ser positivo/negativo para mover o texto para a direita/esquerda.

Parâmetros:

`shiftText` – O número de pixels para mover o texto

Relaciona-se com:

`getShiftText()`

startTicker

```
public void startTicker(long delay,
                        boolean rightToLeft)
```

Esse método iniciará o sinalizador de texto.

Parâmetros:

`delay` – o atraso em milissegundos entre os intervalos de animação

`rightToLeft` – se *true*, move o texto para a esquerda

stopTicker

```
public void stopTicker()
```

Para o sinalizador de texto.

isTickerRunning

```
public boolean isTickerRunning()
```

Retorna *true* se o ticker estiver funcionando.

Retorna:

true se o sinalizador estiver funcionando

setTickerEnabled

`public void setTickerEnabled(boolean tickerEnabled)`

Define o Label para permitir a sinalização do texto. Por padrão, é *true*.

Parâmetros:

`tickerEnabled` – indica se o sinalizador está habilitado

isTickerEnabled

`public boolean isTickerEnabled()`

Esse método retorna *true* se o ticker estiver habilitado nesse Label.

Retorna:

true se o sinalizador estiver habilitado, caso contrário, *false*

setEndsWith3Points

`public void setEndsWith3Points(boolean endsWith3Points)`

Se o texto do Label for muito longo, inclua o texto no *widget* e adicione "..." pontos no final. Por padrão, é definido para *true*.

Parâmetros:

`endsWith3Points` – *true* se o texto adicionar "..." no final

isEndsWith3Points

`public boolean isEndsWith3Points()`

`Getter` simples.

Retorna:

true se esse Label adicionar "..." quando o texto for muito longo

animate

`public boolean animate()`

Descrição copiada da interface: **`Animation`**

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*. É chamado uma vez para cada quadro.

Especificado por:

`animate` na interface `Animation`

Substituições:

`animate` na classe `Component`

Retorna:

true se uma repintura for desejada ou *false* se a repintura não for necessária

setTextContent

```
public void setTextContent(String text,
                           int state)
                    throws IllegalArgumentException
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Designa o conteúdo do texto a esse componente que é dependente do estado.

Especificado por:

`setTextContent` na interface `ViewOnlyComponent`

Parâmetros:

`text` – o conteúdo do texto, ou `null` (remove o conteúdo do texto que foi designado antes)

`state` - o estado do componente para o qual esse conteúdo deve ser exibido; `STATE_ENABLED` ou `STATE_DISABLED`

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

setGraphicContent

```
public void setGraphicContent(Image image,
                              int state)
                       throws IllegalArgumentException
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Designa o conteúdo gráfico a esse componente que é dependente do estado.

Especificado por:

`setGraphicContent` na interface `ViewOnlyComponent`

Parâmetros:

`image` - o conteúdo gráfico, ou `null` (remove o conteúdo gráfico que foi designado antes)

`state` - o estado do componente para o qual esse conteúdo deve ser exibido; `STATE_ENABLED` ou `STATE_DISABLED`

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

setAnimateContent

```
public void setAnimateContent(Image[] images,
                              int state)
                       throws IllegalArgumentException
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Designa um *array* de conteúdo gráfico para ser usado em animação desse componente, que é dependente do estado.

Especificado por:

`setAnimateContent` na interface `ViewOnlyComponent`

Parâmetros:

`images` - o conteúdo gráfico de um *array*, ou `null` (remove qualquer conteúdo gráfico que foi designado antes)

`state` - o estado do componente para o qual esse conteúdo deve ser exibido; `STATE_ENABLED` ou `STATE_DISABLED`

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

getTextContent

```
public String getTextContent(int state)
                      throws IllegalArgumentException
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Retorna conteúdo de texto desse componente, dependendo do estado atual.

Especificado por:

`getTextContent` na interface `ViewOnlyComponent`

Parâmetros:

`state` - o estado do componente para o qual esse conteúdo deve ser restaurado; `STATE_ENABLED` ou `STATE_DISABLED`

Retorna:

O conteúdo do texto associado ao estado determinado, ou `null` se não houver nenhum.

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

getGraphicContent

```
public Image getGraphicContent(int state)
                        throws IllegalArgumentException
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Retorna conteúdo gráfico desse componente, dependendo do estado atual.

Especificado por:

`getGraphicContent` na interface `ViewOnlyComponent`

Parâmetros:

`state` - o estado do componente para o qual esse conteúdo deve ser restaurado; `STATE_ENABLED` ou `STATE_DISABLED`

Retorna:

O conteúdo gráfico associado ao estado determinado, ou `null` se não houver nenhum.

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

getAnimateContent

```
public Image[] getAnimateContent(int state)
                          throws IllegalArgumentException
```

Descrição copiada da interface: **ViewOnlyComponent**

Retorna o conteúdo animado desse componente, dependendo do estado atual.

Especificado por:

`getAnimateContent` na interface `ViewOnlyComponent`

Parâmetros:

`state` - o estado do componente para o qual esse conteúdo deve ser restaurado; `STATE_ENABLED` ou `STATE_DISABLED`

Retorna:

O conteúdo gráfico associado ao estado determinado, ou `null` se não houver nenhum.

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

setInteractionState

```
public void setInteractionState(int state)
                         throws IllegalArgumentException
```

Descrição copiada da interface: **ViewOnlyComponent**

Define o estado atual do componente com relação a interação.

Especificado por:

`setInteractionState` na interface `ViewOnlyComponent`

Parâmetros:

`state` – o estado de interação para esse conteúdo; `STATE_ENABLED` ou `STATE_DISABLED`

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

getInteractionState

```
public int getInteractionState()
```

Descrição copiada da interface: **ViewOnlyComponent**

Retorna o estado de interação atual do componente.

Especificado por:

`getInteractionState` na interface `ViewOnlyComponent`

Retorna:

o estado de interação atual do componente.

setTextLayoutManager

```
public void setTextLayoutManager(TextLayoutManager manager)
```

Descrição copiada da interface: **ViewOnlyComponent**

Define o gerenciador de layout de texto que deve ser usado para o layout de texto nesse componente.

Especificado por:

`setTextLayoutManager` na interface `ViewOnlyComponent`

Parâmetros:

`manager` – o `TextLayoutManager`

getTextLayoutManager

`public TextLayoutManager getTextLayoutManager()`

Descrição copiada da interface: **`ViewOnlyComponent`**

Obtém o gerenciador de layout de texto que está em uso para o layout de texto nesse componente.

Especificado por:

`getTextLayoutManager` na interface `ViewOnlyComponent`

Retorna:

o `TextLayoutManager`

setHorizontalAlignment

`public void setHorizontalAlignment(int alignment)`

Descrição copiada da interface: **`ViewOnlyComponent`**

Define o alinhamento horizontal para qualquer conteúdo baseado no estado desse componente.

Especificado por:

`setHorizontalAlignment` na interface `ViewOnlyComponent`

Parâmetros:

`alignment` – o modo de alinhamento horizontal, um de `ViewOnlyComponent.HORIZONTAL_ALIGN_LEFT`, `ViewOnlyComponent.HORIZONTAL_ALIGN_CENTER`, `ViewOnlyComponent.HORIZONTAL_ALIGN_RIGHT` ou `ViewOnlyComponent.HORIZONTAL_ALIGN_JUSTIFIED`.

getHorizontalAlignment

`public int getHorizontalAlignment()`

Descrição copiada da interface: **`ViewOnlyComponent`**

Restaura o alinhamento horizontal de qualquer conteúdo baseado em estado nesse componente.

Especificado por:

`getHorizontalAlignment` na interface `ViewOnlyComponent`

Retorna:

o modo de alinhamento horizontal atual, um de `ViewOnlyComponent.HORIZONTAL_ALIGN_LEFT`, `ViewOnlyComponent.HORIZONTAL_ALIGN_CENTER`, `ViewOnlyComponent.HORIZONTAL_ALIGN_RIGHT` ou `ViewOnlyComponent.HORIZONTAL_ALIGN_JUSTIFIED`.

setScalingMode

`public void setScalingMode(int scaling)`

Descrição copiada da interface: **`ViewOnlyComponent`**

Define o modo de redimensionamento para esse componente.

Especificado por:

setScalingMode na interface ViewOnlyComponent

Parâmetros:

scaling – o modo de redimensionamento, ViewOnlyComponent.SCALE_NO, ViewOnlyComponent.SCALE_NO_ASPECT_PROOF ou ViewOnlyComponent.SCALE_ASPECT_PROOF.

getScalingMode

public int getScalingMode()

Descrição copiada da interface: **ViewOnlyComponent**

Restaura o modo de redimensionamento desse componente.

Especificado por:

getScalingMode na interface ViewOnlyComponent

Retorna:

o modo de redimensionamento atual; ViewOnlyComponent.SCALE_NO, ViewOnlyComponent.SCALE_NO_ASPECT_PROOF ou ViewOnlyComponent.SCALE_ASPECT_PROOF.

isDoubleBuffered

public boolean isDoubleBuffered()

Descrição copiada da interface: **ViewOnlyComponent**

Retorna *true* se o buffering duplo estiver disponível e ativado.

Especificado por:

isDoubleBuffered na interface ViewOnlyComponent

Retorna:

true se o buffering duplo estiver disponível e ativado, caso contrário, *false*

isOpaque

public boolean isOpaque()

Descrição copiada da interface: **ViewOnlyComponent**

Retorna *true* se toda a área do componente (conforme retornado pelo método com.sun.dtv.lwuit.Component#getBounds, for opaco.

Especificado por:

isOpaque na interface ViewOnlyComponent

Retorna:

true se todos os pixels dentro da área determinada pelo método com.sun.dtv.lwuit.Component#getBounds forem opacos, ou seja, todos os pixels estiverem pintados com uma cor opaca, caso contrário, *false*.

processEvent

public void processEvent(AWTEvent event)

Descrição copiada da interface: **ViewOnlyComponent**

Lida com o AWTEvent determinado.

Especificado por:

`processEvent` na interface `ViewOnlyComponent`

Parâmetros:

`event` – um `java.awt.AWTEvent` para lidar.

setMatte

```
public void setMatte(Matte matte)
              throws MatteException
```

Descrição copiada da interface: **`MatteEnabled`**

Adiciona um `Matte` ao componente implementando essa interface para possibilitar composição de matte. Se já houver um Matte designado para esse componente e esse Matte for animado, ele tem de ser parado antes de qualquer chamada para esse método.

Especificado por:

`setMatte` na interface `MatteEnabled`

Parâmetros:

`matte` - o `Matte` a ser designado ao componente. Todo o tempo, só pode haver um `matte` associado ao componente, devido a isso qualquer `matte` associado anteriormente deve ser sobreposto por uma chamada para esse método. O parâmetro `Matte` pode também ser `null`, nesse caso não há `matte` associado ao componente após a chamada, mesmo se houvesse um antes.

Lança:

`MatteException` - se o `Matte` tem um tipo não-suportado, a plataforma não suporta `mattes` em nenhuma circunstância ou um `matte` animado está associado ao componente e ainda está rodando

getMatte

```
public Matte getMatte()
```

Descrição copiada da interface: **`MatteEnabled`**

Retorna o `Matte` atualmente associado ao componente implementando essa interface.

Especificado por:

`getMatte` na interface `MatteEnabled`

Retorna:

o `Matte` atualmente associado ao componente ou `null` se não houver nenhum

24.19 Classe List

24.19.1 Descrição da classe

`com.sun.dtv.lwuit`

```
java.lang.Object
  └com.sun.dtv.lwuit.Component
      └com.sun.dtv.lwuit.List
```

Todas as interfaces implementadas

Animated, Animation, MatteEnabled, StyleListener, ViewOnlyComponent

Subclasses diretas conhecidas

ComboBox

public class List

extends Component

implements ViewOnlyComponent

Um conjunto de elementos que são processados usando um ListCellRenderer e são extraídos por meio do ListModel.

Uma lista pode representar muitos conceitos de interface do usuário que variam de um carrossel a uma lista de verificação de "tarefas". Isso é possível graças ao amplo uso do estilo MVC do Swing. Especificamente, um componente da lista é relativamente simples. Executa o modelo para extrair a informação exibida/selecionada e a mostra para o usuário invocando o renderizador de célula.

A classe List em si é completamente separada de tudo. Deste modo, permite extrair seu conteúdo de qualquer fonte (ex.: rede, armazenamento, etc.) e exibe a informação em qualquer formulário (ex.: ícones, elementos com caixa de seleção, etc.).

Relaciona-se com:

ListModel

24.19.2 Índice de campos

static int **FIXED_CENTER**

Indica que a seleção da lista está fixada no local no centro da lista.

static int **FIXED_LEAD**

Indica que a seleção da lista está fixada no local no topo da lista ou à esquerda da lista.

static int **FIXED_NONE**

Indica que a lista não está fixada e que a seleção é móvel.

static int **FIXED_NONE_CYCLIC**

Indica que a lista não está fixada no lugar, mas move seus elementos em ciclos.

static int **FIXED_NONE_ONE_ELEMENT_MARGIN_FROM_EDGE**

Indica que a seleção de lista só não alcançará o fim quando não houver mais elementos na lista.

static int **FIXED_TRAIL**

Indica que a seleção da lista está fixada no local no fim da lista ou à direita da lista.

static int **HORIZONTAL**

Indica que a orientação da lista é HORIZONTAL.

static int **VERTICAL**

Indica que a orientação da lista é VERTICAL.

Campos herdados da classe com.sun.dtv.lwuit.**Component**

BOTTOM, BRB_CENTER_OFFSET, BRB_CONSTANT_ASCENT, BRB_CONSTANT_DESCENT, BRB_OTHER, CENTER, LEFT, RIGHT, TOP

Campos herdados da interface com.sun.dtv.ui.**Animated**

`ALTERNATING, LOOP, REPEATING`

Campos herdados da interface `com.sun.dtv.ui.`**`ViewOnlyComponent`**

`HORIZONTAL_ALIGN_CENTER, HORIZONTAL_ALIGN_JUSTIFIED, HORIZONTAL_ALIGN_LEFT, HORIZONTAL_ALIGN_RIGHT, SCALE_ASPECT_PROOF, SCALE_NO, SCALE_NO_ASPECT_PROOF, STATE_DISABLED, STATE_ENABLED, VERTICAL_ALIGN_BOTTOM, VERTICAL_ALIGN_CENTER, VERTICAL_ALIGN_JUSTIFIED, VERTICAL_ALIGN_TOP`

24.19.3 Índice de construtores

`List``()`

Cria uma nova instância da Lista com um modelo padrão vazio.

`List``(ListModel model)`

Cria uma nova instância da Lista com o modelo determinado.

`List``(Object[] items)`

Cria uma nova instância da Lista.

`List``(Vector items)`

Cria uma nova instância da Lista.

24.19.4 Índice de métodos

`void` **`addActionListener`**`(ActionListener l)`

Permite vincular um *listener* às ações de seleção do usuário.

`void` **`addItem`**`(Object item)`

Permite adicionar um elemento a uma lista se o modelo subjacente o suportar. Note que essa é uma operação opcional, e se o modelo não a suportar (o modelo de lista padrão suporta), então a operação poderá falhar.

`void` **`addSelectionListener`**`(SelectionListener l)`

Executado para indicar interesse em eventos de seleção futuros.

`boolean` **`animate`**`()`

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*.

`Image[]` **`getAnimateContent`**`(int state)`

Retorna o conteúdo animado desse componente, dependendo do estado atual.

`int` **`getBorderGap`**`()`

Recebendo o intervalo da borda circundante.

`int` **`getFixedSelection`**`()`

Indica se a seleção pode ser fixada no lugar. Nesse caso, todos os elementos da lista se movem e a seleção fica no lugar.

`Image` **`getGraphicContent`**`(int state)`

Retorna conteúdo gráfico desse componente, dependendo do estado atual.

`int` **`getHorizontalAlignment`**`()`

Restaura o alinhamento horizontal de qualquer conteúdo baseado em estado nesse componente.

`int` **`getInteractionState`**`()`

Retorna o estado de interação atual do componente.

`int` **`getItemGap`**`()`

Retorna o intervalo entre os itens.

`Matte` **`getMatte`**`()`

Retorna o `Matte` atualmente associado ao componente implementando essa interface.

`ListModel` **`getModel`**`()`

Retorna o modelo subjacente à lista.

`int` **`getOrientation`**`()`

Retorna a orientação da lista.

`ListCellRenderer` **`getRenderer`**`()`

Retorna o renderizador usado para desenhar elementos da lista.

`Object` **`getRenderingPrototype`**`()`

Consulte **`setRenderingPrototype`**.

`int` **`getScalingMode`**`()`

Restaura o modo de redimensionamento desse componente.

`int` **`getSelectedIndex`**`()`

Retorna a compensação atual selecionada na lista.

`Object` **`getSelectedItem`**`()`

Retorna o item atual selecionado na lista ou `null` para nenhuma seleção.

`String` **`getTextContent`**`(int state)`

Retorna conteúdo de texto desse componente, dependendo do estado atual.

`TextLayoutManager` **`getTextLayoutManager`**`()`

Obtém o gerenciador de layout de texto que está em uso para o layout de texto nesse componente.

`int` **`getVerticalAlignment`**`()`

Restaura o alinhamento vertical de qualquer conteúdo com base em estado nesse componente.

`boolean` **`isDoubleBuffered`**`()`

Retorna *true* se o buffering duplo estiver disponível e ativado.

`boolean` **`isNumericKeyActions`**`()`

Indica se pressionar as teclas de números ativa uma ação.

`boolean` **`isOpaque`**`()`

Retorna *true* se toda a área do componente (conforme retornado pelo método `com.sun.dtv.lwuit.Component#getBounds`, for opaco.

`boolean` **`isScrollableX`**`()`

Indica se o componente deveria/poderia rolar no eixo X.

`boolean` **`isScrollableY`**`()`

Indica se o componente deveria/poderia rolar no eixo Y.

`void` **`keyPressed`**`(int keyCode)`

Se esse componente estiver destacado, o evento chave pressionado chamará esse método.

`void` **`keyReleased`**`(int keyCode)`

Se esse componente estiver destacado, o evento chave liberado chamará esse método.

`void` **`paint`**`(Graphics g)`

Método para pintar o componente.

`void` **`pointerDragged`**`(int x, int y)`

Se esse componente estiver destacado, o evento indicador arrastado chamará esse método.

`void` **`pointerReleased`**`(int x, int y)`

Se esse componente estiver destacado, o evento indicador liberado chamará esse método.

`void` **`processEvent`**`(AWTEvent event)`

Lida com o AWTEvent determinado.

`void` **`refreshTheme`**`()`

Certifica-se de que o componente está atualizado com o objeto estilo atual.

`void` **`removeActionListener`**`(ActionListener l)`

Permite vincular um *listener* às ações de seleção do usuário.

`void` **`removeSelectionListener`**`(SelectionListener l)`

Executado para indicar nenhum interesse em eventos de seleção futuros.

`void` **`scrollRectToVisible`**`(Rectangle rect)`

Garante que o índice selecionado esteja visível. Se não estiver no rect de visualização a lista irá rolar para que apareça.

`void` **`setAnimateContent`**`(Image[] images, int state)`

Designa um *array* de conteúdo gráfico para ser usado em animação desse componente, que é dependente do estado.

`void` **`setBorderGap`**`(int borderGap)`

Configurando o intervalo da borda circundante.

`void` **`setFixedSelection`**`(int fixedSelection)`

Indica se a seleção pode ser fixada no lugar. Nesse caso, todos os elementos da lista se movem e a seleção fica no lugar.

`void` **`setGraphicContent`**`(Image image, int state)`

Designa o conteúdo gráfico a esse componente que é dependente do estado.

`void` **`setHandlesInput`**`(boolean b)`

Previne que eventos chave sejam tomados para foco transversal.

`void` **`setHorizontalAlignment`**`(int alignment)`

Define o alinhamento horizontal para qualquer conteúdo baseado no estado desse componente.

`void` **`setInputOnFocus`**`(boolean inputOnFocus)`

A lista pode começar tratar a entrada implicitamente ao ganhar foco. Isso pode colaborar para uma interface de usuário mais intuitiva quando não houver nenhum outro elemento de foco ou quando seu “caso de uso” não for frequente.

`void` **`setInteractionState`**`(int state)`

Define o estado atual do componente com relação a interação.

`void` **`setItemGap`**`(int itemGap)`

Define o intervalo entre os itens.

`void` **`setListCellRenderer`**`(ListCellRenderer renderer)`

Define o renderizador usado para desenhar elementos da lista.

`void` **`setMatte`**`(Matte matte)`

Adiciona um `Matte` ao componente implementando essa interface para possibilitar composição de matte.

`void` **`setModel`**`(ListModel model)`

Substitui/define o modelo subjacente à lista.

`void` **`setNumericKeyActions`**`(boolean numericKeyActions)`

Indica se pressionar as teclas de números ativa uma ação.

`void` **`setOrientation`**`(int orientation)`

Define a orientação da lista HORIZONTAL ou VERTICAL.

`void` **`setPaintFocusBehindList`**`(boolean paintFocusBehindList)`

Este método determina se o foco animado é desenhado no topo da Lista ou atrás da Lista quando está em movimento.

`void` **`setRenderingPrototype`**`(Object renderingPrototype)`

O `renderingPrototype` é usado opcionalmente em cálculos do tamanho da Lista e é recomendado por razões de performance.

`void` **`setScalingMode`**`(int scaling)`

Define o modo de redimensionamento para esse componente.

`void` **`setSelectedIndex`**`(int index)`

Define o atual ajuste selecionado na lista, padronizadamente essa implementação rolará a lista até a seleção, se a seleção estiver fora da tela.

`void` **`setSelectedIndex`**`(int index, boolean scrollToSelection)`

Define o atual ajuste selecionado na lista.

`void` **`setSelectedItem`**`(Object item)`

Define o item atual selecionado na lista.

`void` **`setTextContent`**`(String text, int state)`

Designa o conteúdo do texto a esse componente que é dependente do estado.

`void` **`setTextLayoutManager`**`(TextLayoutManager manager)`

Define o gerenciador de layout de texto que deve ser usado para o layout de texto nesse componente.

`void` **`setVerticalAlignment`**`(int valign)`

Define o alinhamento vertical de qualquer conteúdo com base em estado desse componente.

`int` **`size`**`()`

Retorna o número de elementos na lista, forma abreviada de getModel().getSize().

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`addFocusListener, calcPreferredSize, contains, deinitialize, getAbsoluteX, getAbsoluteY, getAnimationMode, getBaseline, getBaselineResizeBehavior, getBottomGap, getBounds, getClientProperty, getComponentForm, getDelay, getHeight, getNextFocusDown, getNextFocusLeft, getNextFocusRight, getNextFocusUp, getParent,`

```
getPosition, getPreferredSize, getRepetitionMode, getScrollAnimationSpeed,
getScrollX, getScrollY, getSideGap, getStyle, getUIID, getWidth, getX, getY,
handlesInput, hasFocus, initialize, isEnabled, isFocusable, isFocusPainted,
isInitialized, isRunning, isScrollVisible, isSmoothScrolling, isVisible, jumpTo,
keyRepeated, longKeyPress, paintBackgrounds, paintComponent, paintComponent,
pointerPressed, putClientProperty, removeFocusListener, repaint, repaint,
requestFocus, scrollRectToVisible, setAnimationMode, setCellRenderer, setDelay,
setEnabled, setFocus, setFocusable, setFocusPainted, setHeight, setInitialized,
setIsScrollVisible, setNextFocusDown, setNextFocusLeft, setNextFocusRight,
setNextFocusUp, setPreferredSize, setRepetitionMode, setScrollAnimationSpeed,
setScrollX, setScrollY, setShouldCalcPreferredSize, setSize, setSmoothScrolling,
setStyle, setUIID, setVisible, setWidth, setX, setY, start, stop, styleChanged,
toString
```

24.19.5 Detalhe dos campos

FIXED_NONE

```
public static final int FIXED_NONE = 0
```

Indica que a lista não está fixada e que a seleção é móvel.

FIXED_NONE_CYCLIC

```
public static final int FIXED_NONE_CYCLIC = 1
```

Indica que a lista não está fixada no lugar, mas move seus elementos em ciclos.

FIXED_NONE_ONE_ELEMENT_MARGIN_FROM_EDGE

```
public static final int FIXED_NONE_ONE_ELEMENT_MARGIN_FROM_EDGE = 2
```

Indica que a seleção de lista só não alcançará o fim quando não houver mais elementos na lista.

FIXED_LEAD

```
public static final int FIXED_LEAD = 10
```

Indica que a seleção da lista está fixada no local no topo da lista ou à esquerda da lista.

FIXED_TRAIL

```
public static final int FIXED_TRAIL = 11
```

Indica que a seleção da lista está fixada no local no fim da lista ou à direita da lista.

FIXED_CENTER

```
public static final int FIXED_CENTER = 12
```

Indica que a seleção da lista está fixada no local no centro da lista.

VERTICAL

```
public static final int VERTICAL = 0
```

Indica que a orientação da lista é VERTICAL.

HORIZONTAL

```
public static final int HORIZONTAL = 1
```

Indica que a orientação da lista é HORIZONTAL.

24.19.6 Detalhe dos construtores

List

```
public List(Vector items)
```

Cria uma nova instância da Lista.

Parâmetros:

`items` – conjunto de itens inseridos no modelo de lista

List

```
public List(Object[] items)
```

Cria uma nova instância da Lista.

Parâmetros:

`items` – conjunto de itens inseridos no modelo de lista

List

```
public List()
```

Cria uma nova instância da Lista com um modelo padrão vazio.

List

```
public List(ListModel model)
```

Cria uma nova instância da Lista com o modelo determinado.

Parâmetros:

`model` – a instância do modelo

24.19.7 Detalhe dos métodos

isScrollableY

```
public boolean isScrollableY()
```

Indica se o componente deveria/poderia rolar no eixo Y.

Substituições:

`isScrollableY` na classe `Component`

Retorna:

se o componente é rolável no eixo X

isScrollableX

```
public boolean isScrollableX()
```

Indica se o componente deveria/poderia rolar no eixo X.

Substituições:

`isScrollableX` na classe `Component`

Retorna:

se o componente é rolável no eixo X

size

```
public int size()
```

Retorna o número de elementos na lista, forma abreviada de getModel().getSize().

Retorna:

o número de elementos na lista

getSelectedIndex

```
public int getSelectedIndex()
```

Retorna a compensação atual selecionada na lista.

Retorna:

a compensação atual selecionada na lista

Relaciona-se com:

`setSelectedIndex(int)`, `setSelectedIndex(int, boolean)`

setSelectedIndex

```
public void setSelectedIndex(int index)
```

Define o atual ajuste selecionado na lista, padronizadamente essa implementação rolará a lista até a seleção, se a seleção estiver fora da tela.

Parâmetros:

`index` - a compensação atual selecionada na lista

Relaciona-se com:

`getSelectedIndex()`

setSelectedIndex

```
public void setSelectedIndex(int index,
                             boolean scrollToSelection)
```

Define o atual ajuste selecionado na lista.

Parâmetros:

`index` - a compensação atual selecionada na lista

`scrollToSelection` – indica se o rolamento para a seleção deve ocorrer se a seleção estiver fora do campo de visão

Relaciona-se com:

`getSelectedIndex()`

getSelectedItem

`public Object getSelectedItem()`

Retorna o item atual selecionado na lista ou `null` para nenhuma seleção.

Retorna:

o item atual selecionado na lista

Relaciona-se com:

`setSelectedItem(java.lang.Object)`

setSelectedItem

`public void setSelectedItem(Object item)`

Define o item atual selecionado na lista.

Parâmetros:

`item` – o item atual selecionado na lista

Relaciona-se com:

`getSelectedItem()`

getModel

`public ListModel getModel()`

Retorna o modelo subjacente à lista.

Retorna:

o modelo subjacente à lista

Relaciona-se com:

`setModel(com.sun.dtv.lwuit.list.ListModel)`

setModel

`public void setModel(ListModel model)`

Substitui/define o modelo subjacente à lista.

Parâmetros:

O novo modelo subjacente à lista

Relaciona-se com:

`getModel()`

isNumericKeyActions

`public boolean isNumericKeyActions()`

Indica se pressionar as teclas de números ativa uma ação.

Retorna:

true se pressionar as teclas de números ativar uma ação, caso contrário, *false*

setNumericKeyActions

`public void setNumericKeyActions(boolean numericKeyActions)`

Indica se pressionar as teclas de números ativa uma ação.

Parâmetros:

`numericKeyActions` – *true* para ativar uma ação nas teclas de números

setListCellRenderer

`public void setListCellRenderer(ListCellRenderer renderer)`

Define o renderizador usado para desenhar elementos da lista.

Parâmetros:

`renderer` – instância do renderizador da célula

getRenderer

`public ListCellRenderer getRenderer()`

Retorna o renderizador usado para desenhar elementos da lista.

Retorna:

o renderizador que é usado para desenhar os elementos da lista

getOrientation

`public int getOrientation()`

Retorna a orientação da lista.

Retorna:

a orientação da lista HORIZONTAL ou VERTICAL

Relaciona-se com:

`setOrientation(int)`, `HORIZONTAL`, `VERTICAL`

refreshTheme

`public void refreshTheme()`

Certifica-se de que o componente está atualizado com o objeto estilo atual.

Substituições:

`refreshTheme` na classe `Component`

setOrientation

```
public void setOrientation(int orientation)
```

Define a orientação da lista HORIZONTAL ou VERTICAL.

Parâmetros:

`orientation` - a orientação da lista HORIZONTAL ou VERTICAL

Relaciona-se com:

`getOrientation()`, `HORIZONTAL`, `VERTICAL`

scrollRectToVisible

```
public void scrollRectToVisible(Rectangle rect)
```

Garante que o índice selecionado esteja visível. Se não estiver no rect de visualização a lista irá rolar para que apareça.

Parâmetros:

`rect` – a área retangular para rolagem

setHandlesInput

```
public void setHandlesInput(boolean b)
```

Previne que as teclas de evento fiquem fixas em foco horizontal.Por exemplo: um componente da lista pode usar as teclas direcionais para navegação interna, portanto, ele mudará essa *flag* para *true* para prevenir que o gerenciador de foco se mova ao próximo componente.

Substituições:

`setHandlesInput` na classe `Component`

keyReleased

```
public void keyReleased(int keyCode)
```

Se esse componente estiver destacado, o evento chave liberado chamará esse método.

Substituições:

`keyReleased` na classe `Component`

Parâmetros:

`keyCode` - o valor do código chave a indicar a chave física.

keyPressed

```
public void keyPressed(int keyCode)
```

Se esse componente estiver destacado, o evento chave pressionado chamará esse método.

Substituições:

`keyPressed` na classe `Component`

Parâmetros:

`keyCode` - o valor do código chave a indicar a chave física.

paint

`public void paint(Graphics g)`

Descrição copiada da interface: **`ViewOnlyComponent`**

Método para pintar o componente.

Especificado por:

`paint` na interface `ViewOnlyComponent`

`paint` na interface `Animation`

Substituições:

`paint` na classe `Component`

Parâmetros:

`g` - o contexto gráfico a ser usado para pintar.

addSelectionListener

`public void addSelectionListener(SelectionListener l)`

Executado para indicar interesse em eventos de seleção futuros.

Parâmetros:

`l` – o *listener* da seleção a ser adicionado

Relaciona-se com:

`removeSelectionListener(com.sun.dtv.lwuit.events.SelectionListener)`

removeSelectionListener

`public void removeSelectionListener(SelectionListener l)`

Executado para indicar nenhum interesse em eventos de seleção futuros.

Parâmetros:

`l` – o *listener* da seleção a ser removido

Relaciona-se com:

`addSelectionListener(com.sun.dtv.lwuit.events.SelectionListener)`

addActionListener

`public void addActionListener(ActionListener l)`

Permite vincular um *listener* às ações de seleção do usuário.

Parâmetros:

`l` – o *listener* da ação a ser adicionado

Relaciona-se com:

`removeActionListener(com.sun.dtv.lwuit.events.ActionListener)`

removeActionListener

`public void removeActionListener(ActionListener l)`

Permite vincular um *listener* às ações de seleção do usuário.

Parâmetros:

`l` – o *listener* da ação a ser removido

Relaciona-se com:

`addActionListener(com.sun.dtv.lwuit.events.ActionListener)`

setInputOnFocus

`public void setInputOnFocus(boolean inputOnFocus)`

A lista pode começar tratar a entrada implicitamente ao ganhar foco. Isso pode colaborar para uma interface de usuário mais intuitiva quando não houver nenhum outro elemento de foco ou quando seu “caso de uso” não for frequente. No entanto, pode ser estranho em alguns casos nos quais a lista "rouba" o foco.

Parâmetros:

`inputOnFocus` – *true* se uma lista puder tratar a entrada implicitamente ao ganhar foco

setPaintFocusBehindList

`public void setPaintFocusBehindList(boolean paintFocusBehindList)`

Este método determina se o foco animado é desenhado no topo da Lista ou atrás da Lista quando está em movimento.

Parâmetros:

`paintFocusBehindList` – *true* para atrás, *false* para no topo

getItemGap

`public int getItemGap()`

Retorna o intervalo entre os itens.

Retorna:

o intervalo entre os itens

Relaciona-se com:

`setItemGap(int)`

setItemGap

`public void setItemGap(int itemGap)`

Define o intervalo entre os itens.

Parâmetros:

`itemGap` - o intervalo entre os itens

Relaciona-se com:

`getItemGap()`

setRenderingPrototype

```
public void setRenderingPrototype(Object renderingPrototype)
```

O `renderingPrototype` é usado opcionalmente em cálculos do tamanho da Lista e é recomendado por razões de performance. Deve ser executado com um objeto representando um valor teórico na lista que deve ser usada para calcular o tamanho necessário para cada elemento na lista.

Isto permite que os cálculos do tamanho da lista funcionem com várias aparências e que os desenvolvedores predeterminem o tamanho dos elementos da lista.

Ex.: Para uma lista de *strings* que você deseja que tenha sempre 5 caracteres, você pode usar um *prototype* "XXXXX" que usaria o tamanho preferido da *string* "XXXXX" para determinar o tamanho do elemento da lista.

```
EXEMPLO: Para uma lista de datas, pode-se usar uma nova Data (30, 12, 00) etc...
```

Parâmetros:

`renderingPrototype` – um valor que pode ser passado para o renderizador para indicar o tamanho preferido de um componente da lista.

Relaciona-se com:

```
getRenderingPrototype()
```

getRenderingPrototype

```
public Object getRenderingPrototype()
```

Consulte o método `setRenderingPrototype`.

Retorna:

o valor do `renderingPrototype`

Relaciona-se com:

```
setRenderingPrototype(java.lang.Object)
```

pointerDragged

```
public void pointerDragged(int x,
                           int y)
```

Se esse componente estiver destacado, o evento indicador arrastado chamará esse método.

Substituições:

`pointerDragged` na classe `Component`

Parâmetros:

`x` - a coordenada do indicador x

`y` - a coordenada do indicador y

pointerReleased

```
public void pointerReleased(int x,
                            int y)
```

Se esse componente estiver destacado, o evento indicador liberado chamará esse método.

Substituições:

`pointerReleased` na classe `Component`

Parâmetros:

`x` - a coordenada do indicador x

`y` - a coordenada do indicador y

addItem

```
public void addItem(Object item)
```

Permite adicionar um elemento a uma lista se o modelo subjacente o suportar. Note que essa é uma operação opcional, e se o modelo não a suportar (o modelo de lista padrão suporta), então a operação poderá falhar.

Parâmetros:

`item` – o item a ser adicionado a um modelo de lista

getFixedSelection

```
public int getFixedSelection()
```

Indica se a seleção pode ser fixada no lugar. Nesse caso, todos os elementos da lista se movem e a seleção fica no lugar.

Retorna:

um de: `FIXED_NONE, FIXED_TRAIL, FIXED_LEAD, FIXED_CENTER, FIXED_NONE_CYCLIC`

Relaciona-se com:

`setFixedSelection(int)`

setFixedSelection

```
public void setFixedSelection(int fixedSelection)
```

Indica se a seleção pode ser fixada no lugar. Nesse caso, todos os elementos da lista se movem e a seleção fica no lugar.

Parâmetros:

`fixedSelection` - um de: `FIXED_NONE, FIXED_TRAIL, FIXED_LEAD, FIXED_CENTER, FIXED_NONE_CYCLIC`

Relaciona-se com:

`getFixedSelection()`

animate

```
public boolean animate()
```

Descrição copiada da interface: **`Animation`**

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*. É chamado uma vez para cada quadro.

Especificado por:

`animate` na interface `Animation`

Substituições:

`animate` na classe `Component`

Retorna:

true se uma repintura for desejada ou *false* se a repintura não for necessária

setBorderGap

```
public void setBorderGap(int borderGap)
```

Configurando o intervalo da borda circundante.

Parâmetros:

`borderGap` – número de pixels para o intervalo

Relaciona-se com:

`getBorderGap()`

getBorderGap

```
public int getBorderGap()
```

Recebendo o intervalo da borda circundante.

Retorna:

intervalo da borda em pixels

Relaciona-se com:

`setBorderGap(int)`

setTextContent

```
public void setTextContent(String text,
                           int state)
                    throws IllegalArgumentException
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Designa o conteúdo do texto a esse componente que é dependente do estado.

Especificado por:

`setTextContent` na interface `ViewOnlyComponent`

Parâmetros:

`text` – o conteúdo do texto, ou `null` (remove o conteúdo do texto que foi designado antes)

`state` - o estado do componente para o qual esse conteúdo deve ser exibido; `STATE_ENABLED` ou `STATE_DISABLED`

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

setGraphicContent

```
public void setGraphicContent(Image image,
                              int state)
```

```
                        throws IllegalArgumentException
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Designa o conteúdo gráfico a esse componente que é dependente do estado.

Especificado por:

`setGraphicContent` na interface `ViewOnlyComponent`

Parâmetros:

`image` - o conteúdo gráfico, ou `null` (remove o conteúdo gráfico que foi designado antes)

`state` - o estado do componente para o qual esse conteúdo deve ser exibido; `STATE_ENABLED` ou `STATE_DISABLED`

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

setAnimateContent

```
public void setAnimateContent(Image[] images,
                              int state)
                       throws IllegalArgumentException
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Designa um *array* de conteúdo gráfico para ser usado em animação desse componente, que é dependente do estado.

Especificado por:

`setAnimateContent` na interface `ViewOnlyComponent`

Parâmetros:

`images` - o conteúdo gráfico de um *array*, ou `null` (remove qualquer conteúdo gráfico que foi designado antes)

`state` - o estado do componente para o qual esse conteúdo deve ser exibido; `STATE_ENABLED` ou `STATE_DISABLED`

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

getTextContent

```
public String getTextContent(int state)
                      throws IllegalArgumentException
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Retorna conteúdo de texto desse componente, dependendo do estado atual.

Especificado por:

`getTextContent` na interface `ViewOnlyComponent`

Parâmetros:

`state` - o estado do componente para o qual esse conteúdo deve ser restaurado; `STATE_ENABLED` ou `STATE_DISABLED`

Retorna:

O conteúdo do texto associado ao estado determinado, ou `null` se não houver nenhum.

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

getGraphicContent

```
public Image getGraphicContent(int state)
                        throws IllegalArgumentException
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Retorna conteúdo gráfico desse componente, dependendo do estado atual.

Especificado por:

`getGraphicContent` na interface `ViewOnlyComponent`

Parâmetros:

`state` - o estado do componente para o qual esse conteúdo deve ser restaurado; `STATE_ENABLED` ou `STATE_DISABLED`

Retorna:

O conteúdo gráfico associado ao estado determinado, ou `null` se não houver nenhum.

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

getAnimateContent

```
public Image[] getAnimateContent(int state)
                          throws IllegalArgumentException
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Retorna o conteúdo animado desse componente, dependendo do estado atual.

Especificado por:

`getAnimateContent` na interface `ViewOnlyComponent`

Parâmetros:

`state` - o estado do componente para o qual esse conteúdo deve ser restaurado; `STATE_ENABLED` ou `STATE_DISABLED`

Retorna:

O conteúdo gráfico associado ao estado determinado, ou `null` se não houver nenhum.

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

setInteractionState

```
public void setInteractionState(int state)
                          throws IllegalArgumentException
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Define o estado atual do componente com relação a interação.

Especificado por:

setInteractionState na interface ViewOnlyComponent

Parâmetros:

state – o estado de interação para esse conteúdo; STATE_ENABLED ou STATE_DISABLED

Lança:

IllegalArgumentExceptionIllegalArgumentException – se o parâmetro do estado não representar um estado

getInteractionState

public int getInteractionState()

Descrição copiada da interface: **ViewOnlyComponent**

Retorna o estado de interação atual do componente.

Especificado por:

getInteractionState na interface ViewOnlyComponent

Retorna:

o estado de interação atual do componente.

setTextLayoutManager

public void setTextLayoutManager(TextLayoutManager manager)

Descrição copiada da interface: **ViewOnlyComponent**

Define o gerenciador de layout de texto que deve ser usado para o layout de texto nesse componente.

Especificado por:

setTextLayoutManager na interface ViewOnlyComponent

Parâmetros:

manager – o TextLayoutManager

getTextLayoutManager

public TextLayoutManager getTextLayoutManager()

Descrição copiada da interface: **ViewOnlyComponent**

Obtém o gerenciador de layout de texto que está em uso para o layout de texto nesse componente.

Especificado por:

getTextLayoutManager na interface ViewOnlyComponent

Retorna:

o TextLayoutManager

setHorizontalAlignment

public void setHorizontalAlignment(int alignment)

Descrição copiada da interface: **ViewOnlyComponent**

Define o alinhamento horizontal para qualquer conteúdo baseado no estado desse componente.

Especificado por:

setHorizontalAlignment na interface ViewOnlyComponent

Parâmetros:

alignment – o modo de alinhamento horizontal, um de ViewOnlyComponent.HORIZONTAL_ALIGN_LEFT, ViewOnlyComponent.HORIZONTAL_ALIGN_CENTER, ViewOnlyComponent.HORIZONTAL_ALIGN_RIGHT ou ViewOnlyComponent.HORIZONTAL_ALIGN_JUSTIFIED.

getHorizontalAlignment

public int getHorizontalAlignment()

Descrição copiada da interface: **ViewOnlyComponent**

Restaura o alinhamento horizontal de qualquer conteúdo baseado em estado nesse componente.

Especificado por:

getHorizontalAlignment na interface ViewOnlyComponent

Retorna:

o modo de alinhamento horizontal atual, um de ViewOnlyComponent.HORIZONTAL_ALIGN_LEFT, ViewOnlyComponent.HORIZONTAL_ALIGN_CENTER, ViewOnlyComponent.HORIZONTAL_ALIGN_RIGHT ou ViewOnlyComponent.HORIZONTAL_ALIGN_JUSTIFIED.

setScalingMode

public void setScalingMode(int scaling)

Descrição copiada da interface: **ViewOnlyComponent**

Define o modo de redimensionamento para esse componente.

Especificado por:

setScalingMode na interface ViewOnlyComponent

Parâmetros:

scaling – o modo de redimensionamento, ViewOnlyComponent.SCALE_NO, ViewOnlyComponent.SCALE_NO_ASPECT_PROOF ou ViewOnlyComponent.SCALE_ASPECT_PROOF.

getScalingMode

public int getScalingMode()

Descrição copiada da interface: **ViewOnlyComponent**

Restaura o modo de redimensionamento desse componente.

Especificado por:

getScalingMode na interface ViewOnlyComponent

Retorna:

o modo de redimensionamento atual; ViewOnlyComponent.SCALE_NO, ViewOnlyComponent.SCALE_NO_ASPECT_PROOF ou ViewOnlyComponent.SCALE_ASPECT_PROOF.

isDoubleBuffered

public boolean isDoubleBuffered()

Descrição copiada da interface: **ViewOnlyComponent**

Retorna *true* se o buffering duplo estiver disponível e ativado.

Especificado por:

`isDoubleBuffered` na interface `ViewOnlyComponent`

Retorna:

`true` se o buffering duplo estiver disponível e ativado, caso contrário, *false*

isOpaque

```
public boolean isOpaque()
```

Descrição copiada da interface: **ViewOnlyComponent**

Retorna *true* se toda a área do componente (conforme retornado pelo método `com.sun.dtv.lwuit.Component#getBounds`, for opaco.

Especificado por:

`isOpaque` na interface `ViewOnlyComponent`

Retorna:

`true` se todos os pixels dentro da área determinada pelo método `com.sun.dtv.lwuit.Component#getBounds` forem opacos, ou seja, todos os pixels estiverem pintados com uma cor opaca, caso contrário, *`false`*.

processEvent

```
public void processEvent(AWTEvent event)
```

Descrição copiada da interface: **ViewOnlyComponent**

Processa o `AWTEvent` determinado.

Especificado por:

`processEvent` na interface `ViewOnlyComponent`

Parâmetros:

`event` – um `java.awt.AWTEvent` para ser processado.

setMatte

```
public void setMatte(Matte matte)
              throws MatteException
```

Descrição copiada da interface: **MatteEnabled**

Adiciona um `Matte` ao componente implementando essa interface para possibilitar composição de `matte`. Se já houver um `Matte` designado para esse componente e esse `Matte` for animado, ele tem de ser parado antes de qualquer chamada para esse método.

Especificado por:

`setMatte` na interface `MatteEnabled`

Parâmetros:

`matte` - o `Matte` a ser designado ao componente. Todo o tempo, só pode haver um `matte` associado ao componente, devido a isso qualquer `matte` associado anteriormente deve ser sobreposto por uma chamada para esse método. O parâmetro `Matte` pode também ser `null`, nesse caso não há `matte` associado ao componente após a chamada, mesmo se houvesse um antes.

Lança:

`MatteException` - se o `Matte` tem um tipo não-suportado, a plataforma não suporta `mattes` em nenhuma circunstância ou um `matte` animado está associado ao componente e ainda está rodando

getMatte

```
public Matte getMatte()
```

Descrição copiada da interface: **`MatteEnabled`**

Retorna o `Matte` atualmente associado ao componente implementando essa interface.

Especificado por:

`getMatte` na interface `MatteEnabled`

Retorna:

o `Matte` atualmente associado ao componente ou `null` se não houver nenhum

setVerticalAlignment

```
public void setVerticalAlignment(int valign)
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Define o alinhamento vertical de qualquer conteúdo com base em estado desse componente.

Especificado por:

`setVerticalAlignment` na interface `ViewOnlyComponent`

getVerticalAlignment

```
public int getVerticalAlignment()
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Restaura o alinhamento vertical de qualquer conteúdo com base em estado nesse componente.

Especificado por:

`getVerticalAlignment` na interface `ViewOnlyComponent`

Retorna:

o modo de alinhamento horizontal atual, um de `ViewOnlyComponent.VERTICAL_ALIGN_TOP`, `ViewOnlyComponent.VERTICAL_ALIGN_CENTER`, `ViewOnlyComponent.VERTICAL_ALIGN_BOTTOM` ou `ViewOnlyComponent.VERTICAL_ALIGN_JUSTIFIED`.

24.20 Classe MediaComponent

24.20.1 Descrição da classe

`com.sun.dtv.lwuit`

```
java.lang.Object
  └com.sun.dtv.lwuit.Component
      └com.sun.dtv.lwuit.MediaComponent
```

Todas as interfaces implementadas

`Animated, Animation, StyleListener`

`public class MediaComponent`

`extends Component`

Um componente permitindo acoplar e controlar conteúdo *rich media*.

Campos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`BOTTOM, BRB_CENTER_OFFSET, BRB_CONSTANT_ASCENT, BRB_CONSTANT_DESCENT, BRB_OTHER, CENTER, LEFT, RIGHT, TOP`

Campos herdados da interface `com.sun.dtv.ui.`**`Animated`**

`ALTERNATING, LOOP, REPEATING`

24.20.2 Índice de construtores

`MediaComponent``(Player player)`

Cria uma nova instância do `MediaComponent`.

24.20.3 Índice de métodos

`void` **`paint`**`(Graphics g)`

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

`void` **`paintBackgrounds`**`(Graphics g)`

Esse método pinta todos os planos de fundo dos componentes principais.

`void` **`setHeight`**`(int height)`

Define a altura do componente; esse método é exposto para fim de gerenciadores de layout externos e não deveria ser invocado diretamente.

Se um usuário deseja efetuar o tamanho do componente, `setPreferredSize` deveria ser usado.

`void` **`setVisible`**`(boolean visible)`

Exibe o componente de media integrado.

`void` **`setWidth`**`(int width)`

Define a largura do componente; esse método é exposto para fim de gerenciadores de layout externos e não deveria ser invocado diretamente.

Se um usuário deseja efetuar o tamanho do componente, `setPreferredSize` deveria ser usado.

`void` **`setX`**`(int x)`

Define o local x relativo do componente; esse método é exposto para fim de gerenciadores de layout externos e não deveria ser invocado diretamente.

`void` **`setY`**`(int y)`

Define o local y relativo do componente; esse método é exposto para fim de gerenciadores de layout externos e não deveria ser invocado diretamente.

`void` **`startMedia`**`()`

Inicia playback da mídia implicitamente configurando o componente para visível.

`void` **`stopMedia`**`()`

Para o *playback* da mídia.

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`addFocusListener, animate, calcPreferredSize, contains, deinitialize, getAbsoluteX, getAbsoluteY, getAnimationMode, getBaseline, getBaselineResizeBehavior, getBottomGap, getBounds, getClientProperty, getComponentForm, getDelay, getHeight, getNextFocusDown, getNextFocusLeft, getNextFocusRight, getNextFocusUp, getParent, getPosition, getPreferredSize, getRepetitionMode, getScrollAnimationSpeed, getScrollX, getScrollY, getSideGap, getStyle, getUIID, getWidth, getX, getY, handlesInput, hasFocus, initialize, isEnabled, isFocusable, isFocusPainted, isInitialized, isRunning, isScrollableX, isScrollableY, isScrollVisible, isSmoothScrolling, isVisible, jumpTo, keyPressed, keyReleased, keyRepeated, longKeyPress, paintComponent, paintComponent, pointerDragged, pointerPressed, pointerReleased, putClientProperty, refreshTheme, removeFocusListener, repaint, repaint, requestFocus, scrollRectToVisible, setAnimationMode, setCellRenderer, setDelay, setEnabled, setFocus, setFocusable, setFocusPainted, setHandlesInput, setInitialized, setIsScrollVisible, setNextFocusDown, setNextFocusLeft, setNextFocusRight, setNextFocusUp, setPreferredSize, setRepetitionMode, setScrollAnimationSpeed, setScrollX, setScrollY, setShouldCalcPreferredSize, setSize, setSmoothScrolling, setStyle, setUIID, start, stop, styleChanged, toString`

24.20.4 Detalhe dos construtores

MediaComponent

`public MediaComponent(Player player)`

Cria uma nova instância do `MediaComponent`.

Parâmetros:

`player` – o *player* de mídia

24.20.5 Detalhe dos métodos

paint

`public void paint(Graphics g)`

Descrição copiada da interface: **`Animation`**

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

Especificado por:

`paint` na interface `Animation`

Substituições:

`paint` na classe `Component`

Parâmetros:

g - os gráficos do componente

paintBackgrounds

`public void paintBackgrounds(Graphics g)`

Esse método pinta todos os planos de fundo dos componentes principais.

Substituições:

`paintBackgrounds` na classe `Component`

Parâmetros:

`g` - o objeto gráficos

setX

`public void setX(int x)`

Define o local x relativo do componente; esse método é exposto para fim de gerenciadores de layout externos e não deveria ser invocado diretamente.

Substituições:

`setX` na classe `Component`

Parâmetros:

`x` - a coordenada x atual da origem dos componentes

setY

`public void setY(int y)`

Define o local y relativo do componente; esse método é exposto para fim de gerenciadores de layout externos e não deveria ser invocado diretamente.

Substituições:

`setY` na classe `Component`

Parâmetros:

`y` - a coordenada y atual da origem dos componentes

setWidth

`public void setWidth(int width)`

Define a largura do componente; esse método é exposto para fim de gerenciadores de layout externos e não deveria ser invocado diretamente.

Se um usuário deseja efetuar o tamanho do componente, `setPreferredSize` deveria ser usado.

Substituições:

`setWidth` na classe `Component`

Parâmetros:

`width` - a largura do componente

setHeight

```
public void setHeight(int height)
```

Define a altura do componente; esse método é exposto para fim de gerenciadores de layout externos e não deveria ser invocado diretamente.

Se um usuário deseja efetuar o tamanho do componente, `setPreferredSize` deveria ser usado.

Substituições:

`setHeight` na classe `Component`

Parâmetros:

`height` - a altura do componente

setVisible

```
public void setVisible(boolean visible)
```

Exibe o componente de media integrado.

Substituições:

`setVisible` na classe `Component`

Parâmetros:

`visible` – *true* para exibir, *false* para ocultar

startMedia

```
public void startMedia()
                throws MediaException
```

Inicia *playback* da mídia implicitamente configurando o componente para visível.

Lança:

`MediaException` – se a inicialização da mídia falhar

stopMedia

```
public void stopMedia()
               throws MediaException
```

Para media playback.

Lança:

`MediaException` – se a paralização da mídia falhar

24.21 Interface Painter

24.21.1 Descrição da interface

`com.sun.dtv.lwuit`

Todas as classes implementadoras conhecidas

`BackgroundPainter, PainterChain`

```
public interface Painter
```

O `Painter` pode ser usado para desenhar em componentes plano de fundo. O uso desse painter permite a reutilização de painters de plano de fundo para vários componentes. Note que, para visualizar desenho do painter,

o componente precisa ter algum nível de transparência.

24.21.2 Índice de métodos

`void` **`paint`**`(Graphics g, Rectangle rect)`

Desenha dentro da área de recorte do retângulo determinado.

24.21.3 Detalhe dos métodos

paint

```
void paint(Graphics g,
           Rectangle rect)
```

Desenha dentro da área de recorte do retângulo determinado.

Parâmetros:

`g` – o objeto Graphics

`rect` - a área de recorte do retângulo determinado

24.22 Classe RadioButton

24.22.1 Descrição da classe

`com.sun.dtv.lwuit`

```
java.lang.Object
  └com.sun.dtv.lwuit.Component
      └com.sun.dtv.lwuit.Label
          └com.sun.dtv.lwuit.Button
              └com.sun.dtv.lwuit.RadioButton
```

Todas as interfaces implementadas

`Animated, Animation, MatteEnabled, StyleListener, ViewOnlyComponent`

```
public class RadioButton
extends Button
```

`RadioButton` é um `Button` que mantém um estado de seleção exclusivamente dentro de um `ButtonGroup` específico .

Campos herdados da classe `com.sun.dtv.lwuit.`**`Button`**

`STATE_DEFAULT, STATE_PRESSED, STATE_ROLLOVER`

Campos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`BOTTOM, BRB_CENTER_OFFSET, BRB_CONSTANT_ASCENT, BRB_CONSTANT_DESCENT, BRB_OTHER, CENTER, LEFT, RIGHT, TOP`

Campos herdados da interface com.sun.dtv.ui.**Animated**

ALTERNATING, LOOP, REPEATING

Campos herdados da interface com.sun.dtv.ui.**ViewOnlyComponent**

HORIZONTAL_ALIGN_CENTER, HORIZONTAL_ALIGN_JUSTIFIED, HORIZONTAL_ALIGN_LEFT, HORIZONTAL_ALIGN_RIGHT, SCALE_ASPECT_PROOF, SCALE_NO, SCALE_NO_ASPECT_PROOF, STATE_DISABLED, STATE_ENABLED, VERTICAL_ALIGN_BOTTOM, VERTICAL_ALIGN_CENTER, VERTICAL_ALIGN_JUSTIFIED, VERTICAL_ALIGN_TOP

24.22.2 Índice de construtores

RadioButton()

Cria um botão de seleção vazio.

RadioButton(Image icon)

Cria uma seleção com o ícone determinado.

RadioButton(String text)

Cria uma seleção com o texto determinado.

RadioButton(String text, Image icon)

Cria uma seleção com o ícone e o texto determinados.

24.22.3 Índice de métodos

boolean **isSelected**()

Retorna *true* se o botão de seleção estiver selecionado.

void **paint**(Graphics g)

Método para pintar o componente.

void **setSelected**(boolean selected)

Seleciona o atual botão de seleção.

String **toString**()

Sobreposto para retornar um valor útil para fins de depuração.

Métodos herdados da classe com.sun.dtv.lwuit.**Button**

addActionListener, getPressedIcon, getRolloverIcon, getState, keyPressed, keyReleased, pointerPressed, pointerReleased, removeActionListener, setPressedIcon, setRolloverIcon

Métodos herdados da classe com.sun.dtv.lwuit.**Label**

animate, getAlignment, getAnimateContent, getBaselineResizeBehavior, getGap, getGraphicContent, getHorizontalAlignment, getIcon, getInteractionState, getMatte, getScalingMode, getShiftText, getText, getTextContent, getTextLayoutManager, getTextPosition, getVerticalAlignment, isDoubleBuffered, isEndsWith3Points, isOpaque, isTickerEnabled, isTickerRunning, processEvent, setAlignment, setAnimateContent, setEndsWith3Points, setGap, setGraphicContent, setHorizontalAlignment, setIcon, setInteractionState, setMatte, setScalingMode, setShiftText, setText, setTextContent,

`setTextLayoutManager, setTextPosition, setTickerEnabled, setVerticalAlignment, startTicker, stopTicker`

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`addFocusListener, calcPreferredSize, contains, deinitialize, getAbsoluteX, getAbsoluteY, getAnimationMode, getBaseline, getBottomGap, getBounds, getClientProperty, getComponentForm, getDelay, getHeight, getNextFocusDown, getNextFocusLeft, getNextFocusRight, getNextFocusUp, getParent, getPosition, getPreferredSize, getRepetitionMode, getScrollAnimationSpeed, getScrollX, getScrollY, getSideGap, getStyle, getUIID, getWidth, getX, getY, handlesInput, hasFocus, initialize, isEnabled, isFocusable, isFocusPainted, isInitialized, isRunning, isScrollableX, isScrollableY, isScrollVisible, isSmoothScrolling, isVisible, jumpTo, keyRepeated, longKeyPress, paintBackgrounds, paintComponent, paintComponent, pointerDragged, putClientProperty, refreshTheme, removeFocusListener, repaint, repaint, requestFocus, scrollRectToVisible, setAnimationMode, setCellRenderer, setDelay, setEnabled, setFocus, setFocusable, setFocusPainted, setHandlesInput, setHeight, setInitialized, setIsScrollVisible, setNextFocusDown, setNextFocusLeft, setNextFocusRight, setNextFocusUp, setPreferredSize, setRepetitionMode, setScrollAnimationSpeed, setScrollX, setScrollY, setShouldCalcPreferredSize, setSize, setSmoothScrolling, setStyle, setUIID, setVisible, setWidth, setX, setY, start, stop, styleChanged`

24.22.4 Detalhe dos construtores

RadioButton

```
public RadioButton(String text)
```

Cria uma seleção com o texto determinado.

Parâmetros:

`text` - para exibir o próximo botão

RadioButton

```
public RadioButton()
```

Cria um botão de seleção vazio.

RadioButton

```
public RadioButton(Image icon)
```

Cria uma seleção com o ícone determinado.

Parâmetros:

`icon` – ícone para exibir o próximo botão

RadioButton

```
public RadioButton(String text,
                   Image icon)
```

Cria uma seleção com o ícone e o texto determinados.

Parâmetros:

`text` - para exibir o próximo botão

`icon` – ícone para exibir o próximo botão

24.22.5 Detalhe dos métodos

toString

```
public String toString()
```

Sobreposto para retornar um valor útil para fins de depuração. Sobrepõe `Object.toString`.

Substituições:

`toString` na classe `Component`

Retorna:

uma representação *string* desse componente

isSelected

```
public boolean isSelected()
```

Retorna *true* se o botão de seleção estiver selecionado.

Retorna:

true se o botão de seleção estiver selecionado

setSelected

```
public void setSelected(boolean selected)
```

Seleciona o atual botão de seleção.

Parâmetros:

`selected` - valor para seleção

paint

```
public void paint(Graphics g)
```

Descrição copiada da interface: **`ViewOnlyComponent`**

Método para pintar o componente.

Especificado por:

`paint` na interface `ViewOnlyComponent`

`paint` na interface `Animation`

Substituições:

`paint` na classe `Button`

Parâmetros:

`g` - o contexto gráfico a ser usado para pintar.

24.23 Classe StaticAnimation

24.23.1 Descrição da classe

com.sun.dtv.lwuit

```
java.lang.Object
  └com.sun.dtv.lwuit.Image
      └com.sun.dtv.lwuit.StaticAnimation
```

Todas as interfaces implementadas

Animated, Animation

```
public class StaticAnimation
extends Image
implements Animation, Animated
```

Uma Imagem capaz de animação.

Campos herdados da interface com.sun.dtv.ui.**Animated**

ALTERNATING, LOOP, REPEATING

24.23.2 Índice de métodos

boolean **animate**()

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*.

static StaticAnimation **createAnimation**(DataInputStream data)

Cria uma animação a partir do *stream* determinado; esse método é usado internamente pelos recursos. Normalmente, você não deve criar animações estáticas manualmente.

int **getAnimationMode**()

Obtém o modo de animação para essa animação.

int **getDelay**()

Obtém o atraso após cada imagem dessa animação ao rodar.

int **getFrameCount**()

Retorna o número de frames na animação incluindo o frame inicial.

int **getFrameTime**(int frame)

O tempo no qual o frame determinado deve aparecer.

int **getPosition**()

Obtém a posição atual que a animação ocupa no momento da chamada do método.

int **getRepetitionMode**()

Retorna o modo de repetição dessa animação na forma de um número.

`int` **`getTotalAnimationTime`**`()`

Retorna a duração de toda animação usada para determinar quando executar o *loop*.

`boolean` **`isAnimation`**`()`

Retorna *true* se essa for uma imagem animada.

`boolean` **`isLoop`**`()`

Indica se a animação deve ser executada em um *loop* ou apenas uma vez.

`boolean` **`isRunning`**`()`

Obtém o modo rodando de uma animação.

`void` **`jumpTo`**`(int position)`

Força uma animação a saltar para a posição indicada pelo parâmetro.

`void` **`paint`**`(Graphics g)`

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

`void` **`restart`**`()`

Reinicia a animação.

`Image` **`scaled`**`(int width, int height)`

Retorna uma versão redimensionada desta imagem usando a largura e altura determinadas. É um algoritmo rápido que preserva a informação de translucidez.

`void` **`setAnimationMode`**`(int mode)`

Define o modo de animação para essa animação.

`void` **`setDelay`**`(int n)`

Determina o atraso após cada imagem dessa animação ao rodar.

`void` **`setLoop`**`(boolean loop)`

Indica se a animação deve ser executada em um *loop* ou apenas uma vez.

`void` **`setRepetitionMode`**`(int n)`

Determina quão freqüentemente a animação é repetida, uma vez que for iniciada.

`void` **`start`**`()`

Inicia uma animação.

`void` **`stop`**`()`

Pára uma animação.

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Image`**

`createImage, createImage, createImage, createImage, createImage, createImage, getAWTImage, getGraphics, getHeight, getRGB, getWidth, isOpaque, modifyAlpha, modifyAlpha, rotate, scaledHeight, scaledSmallerRatio, scaledWidth, subImage`

24.23.3 Detalhe dos métodos

getFrameCount

`public int getFrameCount()`

Retorna o número de frames na animação incluindo o frame inicial.

Retorna:

a contagem do frame

getFrameTime

```
public int getFrameTime(int frame)
```

O tempo no qual o frame determinado deve aparecer.

Parâmetros:

`frame` – deve ser um número maior que -1 e menor que `getFrameCount()`

Retorna:

o tempo em milissegundos para o frame aparecer

getTotalAnimationTime

```
public int getTotalAnimationTime()
```

Retorna a duração de toda animação usada para determinar quando executar o *loop*.

Retorna:

tempo total de animação em milissegundos

createAnimation

```
public static StaticAnimation createAnimation(DataInputStream data)
                                       throws IOException
```

Cria uma animação a partir do *stream* determinado; esse método é usado internamente pelos recursos. Normalmente, você não deve criar animações estáticas manualmente.

Parâmetros:

`data` – *stream* de entrada do local de onde a animação é carregada

Retorna:

Uma instância de uma animação estática

Lança:

`IOException` – se o *stream* não puder ser aberto

animate

```
public boolean animate()
```

Descrição copiada da interface: **`Animation`**

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*. É chamado uma vez para cada quadro.

Especificado por:

`animate` na interface `Animation`

Retorna:

true se uma repintura for desejada ou *false* se a repintura não for necessária

restart

```
public void restart()
```

Reinicia a animação.

paint

```
public void paint(Graphics g)
```

Descrição copiada da interface: **`Animation`**

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

Especificado por:

`paint` na interface `Animation`

isLoop

```
public boolean isLoop()
```

Indica se a animação deve ser executada em um *loop* ou apenas uma vez.

Retorna:

true se a animação executar em *loop*, caso contrário, *false*

setLoop

```
public void setLoop(boolean loop)
```

Indica se a animação deve ser executada em um *loop* ou apenas uma vez.

Parâmetros:

`loop` – indica se a animação deve executar em *loop* ou não

scaled

```
public Image scaled(int width,
                    int height)
```

Retorna uma versão redimensionada desta imagem usando a largura e altura determinadas. É um algoritmo rápido que preserva a informação de translucidez.

Substituições:

`scaled` na classe `Image`

Parâmetros:

`width` – largura para o redimensionamento

`height` – a altura da imagem redimensionada

Retorna:

instância da nova imagem redimensionada para a altura e largura determinadas

isAnimation

```
public boolean isAnimation()
```

Retorna *true* se essa for uma imagem animada.

Substituições:

`isAnimation` na classe `Image`

Retorna:

true se a imagem for uma animação

start

```
public void start()
```

Inicia uma animação. Considere que uma animação fica em modo parado automaticamente, portanto deve ser iniciada explicitamente uma vez que é criada. No caso da animação já estar rodando, uma chamada para `start()` faz com que ela reinicie.

Especificado por:

`start` na interface `Animated`

stop

```
public void stop()
```

Pára uma animação. No caso da animação já ter parado, uma chamada para `stop()` não tem efeito.

Especificado por:

`stop` na interface `Animated`

isRunning

```
public boolean isRunning()
```

Obtém o modo rodando de uma animação. Retorna *true* se a animação já estiver rodando; caso contrário *false*.

Especificado por:

`isRunning` na interface `Animated`

Retorna:

true se a animação já estiver rodando; caso contrário *false*.

jumpTo

```
public void jumpTo(int position)
```

Força uma animação a saltar para a posição indicada pelo parâmetro. Uma animação pode ser considerada como uma fila de imagens, o parâmetro desse método provê o índice dentro desse fila. Se a animação for parada, uma chamada para esse método faz com que a animação mostre a imagem na posição indicada, mas não que inicie. Se a animação estiver rodando, uma chamada para esse método faz com que a animação salte para a imagem na posição indicada e continue a rodar imediatamente.

Especificado por:

`jumpTo` na interface `Animated`

Parâmetros:

`position` - o índice na fila de imagens fazendo a animação para o qual a animação é forçada a saltar

getPosition

`public int getPosition()`

Obtém a posição atual que a animação ocupa no momento da chamada do método. Uma animação pode ser considerada como uma fila de imagens, esse método provê o índice dentro desse fila.

Especificado por:

`getPosition` na interface `Animated`

Retorna:

a posição atual da imagem dentro da fila de imagens fazendo a animação

setRepetitionMode

`public void setRepetitionMode(int n)`

Determina quão freqüentemente a animação é repetida, uma vez que for iniciada.

Especificado por:

`setRepetitionMode` na interface `Animated`

Parâmetros:

`n` - número de repetições a ser determinado. Pode ser um número maior que zero ou `LOOP` alternativamente. LOOP significa que a animação roda em um *loop* infinito até que o método `stop()` seja chamado.

Relaciona-se com:

`getRepetitionMode()`

getRepetitionMode

`public int getRepetitionMode()`

Retorna o modo de repetição dessa animação na forma de um número. O número lê quantas repetições foram determinadas para essa animação.

Especificado por:

`getRepetitionMode` na interface `Animated`

Retorna:

número de repetições determinado para essa animação ou `LOOP`, o que significa que a animação roda em um *loop* infinito até que o método `stop()` seja chamado.

Relaciona-se com:

`setRepetitionMode(int)`

setDelay

`public void setDelay(int n)`

Determina o atraso após cada imagem dessa animação ao rodar.

Especificado por:

`setDelay` na interface `Animated`

Parâmetros:

`n` - atraso em milissegundos

Relaciona-se com:

`getDelay()`

getDelay

`public int getDelay()`

Obtém o atraso após cada imagem dessa animação ao rodar.

Especificado por:

`getDelay` na interface `Animated`

Retorna:

o atraso em milissegundos

Relaciona-se com:

`setDelay(int)`

setAnimationMode

`public void setAnimationMode(int mode)`

Define o modo de animação para essa animação. O modo de animação determina como a animação está rodando: sempre para frente, ou para frente e para trás alternando.

Especificado por:

`setAnimationMode` na interface `Animated`

Parâmetros:

`mode` - o modo da animação. Valores possíveis são: `REPEATING` and `ALTERNATING`.

Relaciona-se com:

`getAnimationMode()`

getAnimationMode

`public int getAnimationMode()`

Obtém o modo de animação para essa animação. O modo de animação determina como a animação está rodando: sempre para frente, ou para frente e para trás alternando.

Especificado por:

`getAnimationMode` na interface `Animated`

Retorna:

O modo da animação. Valores possíveis são: `REPEATING` and `ALTERNATING`.

Relaciona-se com:

`setAnimationMode(int)`

24.24 Classe TabbedPane

24.24.1 Descrição da classe

`com.sun.dtv.lwuit`

```
java.lang.Object
  └com.sun.dtv.lwuit.Component
      └com.sun.dtv.lwuit.Container
          └com.sun.dtv.lwuit.TabbedPane
```

Todas as interfaces implementadas

`Animated, Animation, MatteEnabled, StyleListener`

```
public class TabbedPane
extends Container
```

Um componente que deixa o usuário trocar entre um grupo de componentes ao clicar em uma aba com um dado título e/ou ícone.

Guias/componentes são adicionados a um objeto `TabbedPane` utilizando os métodos `addTab` e `insertTab`. Uma guia é representada por um índice correspondente à posição na qual foi adicionada, onde a primeira guia tem um índice igual a 0 e a última tem um índice igual à contagem da guia menos 1.

O `TabbedPane` usa um `SingleSelectionModel` para representar o conjunto de índices das guias e o índice atualmente selecionado. Se a contagem da guia for maior que 0, sempre existirá um índice selecionado que, por padrão, deve ser iniciado na primeira guia. Se a contagem da guia for 0, o índice selecionado deve ser -1.

Campos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`BOTTOM, BRB_CENTER_OFFSET, BRB_CONSTANT_ASCENT, BRB_CONSTANT_DESCENT, BRB_OTHER, CENTER, LEFT, RIGHT, TOP`

Campos herdados da interface `com.sun.dtv.ui.`**`Animated`**

`ALTERNATING, LOOP, REPEATING`

24.24.2 Índice de construtores

`TabbedPane``()`

Cria um `TabbedPane` vazio com um posicionamento da guia padrão do `Component.TOP`.

`TabbedPane``(int tabPlacement)`

Cria um `TabbedPane` vazio com o posicionamento determinado da guia de: `Component.TOP`, `Component.BOTTOM`, `Component.LEFT`, ou `Component.RIGHT`.

24.24.3 Índice de métodos

`void` **`addTab`**`(String title, Component component)`

Adiciona um `component` representado por um `title` e não um `icon`.

`void` **`addTab`**`(String title, Image icon, Component component)`

Adiciona um `component` representado por um `title` e/ou um `icon`, que podem ser `null`.

`void` **`addTabsListener`**`(SelectionListener listener)`

Esse método adiciona um *listener* às guias.

`int` **`getSelectedIndex`**`()`

Retorna o índice atualmente selecionado para este tabbedpane.

`int` **`getTabbedPaneBorderWidth`**`()`

A largura da borda circundada do `TabbedPane`.

`Component` **`getTabComponentAt`**`(int index)`

Retorna a guia no `index`.

`int` **`getTabCount`**`()`

Retorna o número de guias nesse `tabbedpane`.

`int` **`getTabPlacement`**`()`

Retorna o posicionamento das guias nesse tabbedpane.

`Transition` **`getTransitionLeft`**`()`

Indica a transição para usar ao alterar entre as guias, da direita para a esquerda.

`Transition` **`getTransitionRight`**`()`

Indica a transição para usar ao alterar entre as guias, da esquerda para a direita.

`int` **`indexOfComponent`**`(Component component)`

Retorna o índice da guia para um componente determinado.

`void` **`insertTab`**`(String title, Image icon, Component component, int index)`

Insere um `component` no `index`, representado por um `title` e/ou `icon`, que podem ser `null`.

`void` **`paint`**`(Graphics g)`

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

`void` **`refreshTheme`**`()`

Certifica-se de que o componente está atualizado com o objeto estilo atual.

`void` **`removeTabAt`**`(int index)`

Remove a guia no `index`.

`void` **`requestFocus`**`()`

Muda o componente atual para o componente destacado; funcionará apenas para um componente que pertença à forma principal.

`void` **`setFocusable`**`(boolean b)`

Um definidor simples que determina se esse componente pode ser destacado.

`void` **`setPadding`**`(int orientation, int gap)`

Define o padding do tabbed pane.

`void` **`setPadding`**`(int top, int bottom, int left, int right)`

Define o padding do tabbed pane.

`void` **`setSelectedIndex`**`(int index)`

Retorna o índice selecionado para esse tabbedpane.

`void` **`setStyle`**`(Style style)`

Muda o estilo do componente trocando-o por um dado estilo.

`void` **`setTabbedPaneBorderWidth`**`(int tabbedPaneBorderWidth)`

Definindo a largura da borda circundada do `TabbedPane`.

`void` **`setTabPlacement`**`(int tabPlacement)`

Define localização da guia para esse tabbedpane.

`void` **`setTabTitle`**`(String title, Image icon, int index)`

Atualiza a informação sobre os detalhes da guia.

`void` **`setTabTitlePrototype`**`(String title)`

O prototype é usado opcionalmente em cálculos do tamanho de uma guia individual e é recomendado por razões de performance.

`void` **`setTransitionLeft`**`(Transition transitionLeft)`

Indica a transição para usar ao alterar entre as guias, da direita para a esquerda.

`void` **`setTransitionRight`**`(Transition transitionRight)`

Indica a transição para usar ao alterar entre as guias, da esquerda para a direita.

`String` **`toString`**`()`

Sobreposto para retornar um valor útil para fins de depuração.

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Container`**

`addComponent, addComponent, addComponent, contains, doLayout, getComponentAt, getComponentAt, getComponentCount, getComponentIndex, getLayout, getLayoutHeight, getLayoutWidth, getMatte, invalidate, isScrollableX, isScrollableY, layoutContainer, pointerPressed, removeAll, removeComponent, replace, revalidate, scrollComponentToVisible, setCellRenderer, setLayout, setMatte, setScrollable, setScrollableX, setScrollableY`

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`addFocusListener, animate, calcPreferredSize, contains, deinitialize, getAbsoluteX, getAbsoluteY, getAnimationMode, getBaseline, getBaselineResizeBehavior, getBottomGap, getBounds, getClientProperty, getComponentForm, getDelay, getHeight, getNextFocusDown, getNextFocusLeft, getNextFocusRight, getNextFocusUp, getParent, getPosition, getPreferredSize, getRepetitionMode, getScrollAnimationSpeed, getScrollX, getScrollY, getSideGap, getStyle, getUIID, getWidth, getX, getY, handlesInput, hasFocus, initialize, isEnabled, isFocusable, isFocusPainted, isInitialized, isRunning, isScrollVisible, isSmoothScrolling, isVisible, jumpTo, keyPressed, keyReleased, keyRepeated, longKeyPress, paintBackgrounds, paintComponent, paintComponent, pointerDragged, pointerReleased, putClientProperty, removeFocusListener, repaint, repaint, scrollRectToVisible, setAnimationMode, setDelay, setEnabled, setFocus, setFocusPainted, setHandlesInput, setHeight, setInitialized, setIsScrollVisible, setNextFocusDown, setNextFocusLeft, setNextFocusRight, setNextFocusUp, setPreferredSize, setRepetitionMode, setScrollAnimationSpeed, setScrollX, setScrollY, setShouldCalcPreferredSize, setSize, setSmoothScrolling, setUIID, setVisible, setWidth, setX, setY, start, stop, styleChanged`

24.24.4 Detalhe dos construtores

TabbedPane

`public TabbedPane()`

Cria um `TabbedPane` vazio com um posicionamento da guia padrão do `Component.TOP`.

TabbedPane

```
public TabbedPane(int tabPlacement)
```

Cria um `TabbedPane` vazio com o posicionamento determinado da guia de: `Component.TOP`, `Component.BOTTOM`, `Component.LEFT`, ou `Component.RIGHT`.

Parâmetros:

`tabPlacement` – a localização das guias em relação ao conteúdo

24.24.5 Detalhe dos métodos

setTransitionLeft

```
public void setTransitionLeft(Transition transitionLeft)
```

Indica a transição para usar ao alterar entre as guias, da direita para a esquerda.

Parâmetros:

`transitionLeft` – transição para usar ao alterar guias ou `null` para evitar a transição

Relaciona-se com:

`getTransitionLeft()`

getTransitionLeft

```
public Transition getTransitionLeft()
```

Indica a transição para usar ao alterar entre as guias, da direita para a esquerda.

Retorna:

transição para usar ao alterar guias

Relaciona-se com:

`setTransitionLeft(com.sun.dtv.lwuit.animations.Transition)`

setTransitionRight

```
public void setTransitionRight(Transition transitionRight)
```

Indica a transição para usar ao alterar entre as guias, da esquerda para a direita.

Parâmetros:

`transitionRight` – transição para usar ao alterar guias ou `null` para evitar a transição

Relaciona-se com:

`getTransitionRight()`

getTransitionRight

```
public Transition getTransitionRight()
```

Indica a transição para usar ao alterar entre as guias, da esquerda para a direita.

Retorna:

transição para usar ao alterar guias

Relaciona-se com:

`setTransitionRight(com.sun.dtv.lwuit.animations.Transition)`

setFocusable

```
public void setFocusable(boolean b)
```

Um definidor simples que determina se esse componente pode ser destacado.

Substituições:

`setFocusable` na classe `Component`

addTabsListener

```
public void addTabsListener(SelectionListener listener)
```

Esse método adiciona um *listener* às guias.

Parâmetros:

`listener` – um *listener* de seleção para obter os eventos de seleção

requestFocus

```
public void requestFocus()
```

Muda o componente atual para o componente destacado; funcionará apenas para um componente que pertença à forma principal.

Substituições:

`requestFocus` na classe `Component`

setTabPlacement

```
public void setTabPlacement(int tabPlacement)
```

Define localização da guia para esse tabbedpane. Valores possíveis são:

Component.TOP

Component.BOTTOM

Component.LEFT

Component.RIGHT

Se o valor padrão não for definido, deve ser `Component.TOP`.

Parâmetros:

`tabPlacement` – a localização das guias em relação ao conteúdo

Relaciona-se com:

`getTabPlacement()`

addTab

```
public void addTab(String title,
                   Image icon,
```

```
                    Component component)
```

Adiciona um `component` representado por um `title` e/ou um `icon`, que podem ser `null`. Método cover para `insertTab`.

Parâmetros:

`title` – o título a ser exibido nessa guia

`icon` – o ícone a ser exibido nessa guia

`component` – o componente a ser exibido quando essa guia for clicada

Relaciona-se com:

`insertTab()`, `removeTabAt()`

addTab

```
public void addTab(String title,
                   Component component)
```

Adiciona um `component` representado por um `title` e não um `icon`. Método cover para `insertTab`.

Parâmetros:

`title` – o título a ser exibido nessa guia

`component` – o componente a ser exibido quando essa guia for clicada

Relaciona-se com:

`insertTab()`, `removeTabAt()`

insertTab

```
public void insertTab(String title,
                      Image icon,
                      Component component,
                      int index)
```

Insere um `component` no `index`, representado por um `title` e/ou `icon`, que podem ser `null`. Usa java.util.Vector internamente. Ver `insertElementAt` para detalhes de convenções de inserção.

Parâmetros:

`title` – o título a ser exibido nessa guia

`icon` – o ícone a ser exibido nessa guia

`component` – O componente a ser exibido quando essa guia for clicada.

`index` – a posição para inserir essa nova guia

Relaciona-se com:

`addTab()`, `removeTabAt()`

setTabTitle

```
public void setTabTitle(String title,
                        Image icon,
                        int index)
```

Atualiza a informação sobre os detalhes da guia.

Parâmetros:

`title` – o título a ser exibido nessa guia

`icon` – o ícone a ser exibido nessa guia

`index` – a posição para inserir essa nova guia

removeTabAt

```
public void removeTabAt(int index)
```

Remove a guia no `index`. Após o componente associado ao `index` ser removido, sua visibilidade é redefinida para *true* para garantir que ficará visível se adicionado a outros receptáculos.

Parâmetros:

`index` – o índice da guia a ser removido

Lança:

`IndexOutOfBoundsException` – se o índice estiver fora de alcance `(index 0 || index >= tab count)`

Relaciona-se com:

`addTab(),insertTab()`

getTabComponentAt

```
public Component getTabComponentAt(int index)
```

Retorna a guia no `index`.

Parâmetros:

`index` – o índice da guia a ser removido

Retorna:

a guia no `index`.

Lança:

`IndexOutOfBoundsException` – se o índice estiver fora de alcance `(index 0 || index >= tab count)`

Relaciona-se com:

`addTab(),insertTab()`

indexOfComponent

```
public int indexOfComponent(Component component)
```

Retorna o índice da guia para um componente determinado. Retorna -1 se não existir guia para esse componente.

Parâmetros:

`component` - o componente para a guia

Retorna:

a primeira guia que combina esse componente, ou -1 se não existir guia para esse componente

getTabCount

```
public int getTabCount()
```

Retorna o número de guias nesse `tabbedpane`.

Retorna:

um número inteiro especificando o número das páginas tabuladas

getSelectedIndex

`public int getSelectedIndex()`

Retorna o índice atualmente selecionado para este tabbedpane. Retorna -1 se não existir guia atual selecionada.

Retorna:

O índice da guia selecionada

Relaciona-se com:

`setSelectedIndex(int)`

setTabTitlePrototype

`public void setTabTitlePrototype(String title)`

O prototype é usado opcionalmente em cálculos do tamanho de uma guia individual e é recomendado por razões de performance. Deve ser executado com um String representando a largura/altura que deve ser usada para calcular o tamanho necessário de cada elemento da lista.\

Essa operação não é essencial e se não for executada o tamanho das primeiras guias deve ser usado para determinar o tamanho total de uma guia.

Isto permite que os cálculos de tamanho funcionem com várias aparências e que os desenvolvedores predeterminem o tamanho das guias.

Ex.: Para guias que você gostaria que sempre tivessem 5 caracteres, você pode usar um prototype “XXXXX” que usaria o tamanho preferido do String XXXXX para determinar o tamanho das guias.

Parâmetros:

`title` – uma *string* para determinar o tamanho.

toString

`public String toString()`

Sobreposto para retornar um valor útil para fins de depuração. Sobrepõe Object.toString.

Substituições:

`toString` na classe `Component`

Retorna:

uma representação *string* desse componente

paint

`public void paint(Graphics g)`

Descrição copiada da interface: **`Animation`**

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

Especificado por:

`paint` na interface `Animation`

Substituições:

`paint` na classe `Container`

Parâmetros:

`g` - os gráficos do componente

setStyle

`public void setStyle(Style style)`

Muda o estilo do componente trocando-o por um dado estilo.

Substituições:

`setStyle` na classe `Component`

Parâmetros:

`style` - o objeto estilo do componente

refreshTheme

`public void refreshTheme()`

Certifica-se de que o componente está atualizado com o objeto estilo atual.

Substituições:

`refreshTheme` na classe `Container`

getTabPlacement

`public int getTabPlacement()`

Retorna o posicionamento das guias nesse tabbedpane.

Retorna:

o posicionamento das guias nesse tabbedpane

Relaciona-se com:

`setTabPlacement()`

getTabbedPaneBorderWidth

`public int getTabbedPaneBorderWidth()`

A largura da borda circundada do `TabbedPane`.

Retorna:

A largura da borda circundada no `TabbedPane`

Relaciona-se com:

`setTabbedPaneBorderWidth(int)`

setTabbedPaneBorderWidth

`public void setTabbedPaneBorderWidth(int tabbedPaneBorderWidth)`

Definindo a largura da borda circundada do `TabbedPane`.

Parâmetros:

`tabbedPaneBorderWidth` - Largura da borda circundada no `TabbedPane`

Relaciona-se com:

`getTabbedPaneBorderWidth()`

setPadding

```
public void setPadding(int top,
                       int bottom,
                       int left,
                       int right)
```

Define o padding do tabbed pane.

Parâmetros:

`top` - valor para o topo

`bottom` – valor para o fim

`left` - valor para o lado esquerdo

`right` - valor para o lado direito

setPadding

```
public void setPadding(int orientation,
                       int gap)
```

Define o padding do tabbed pane.

Parâmetros:

`orientation` – orientação do padding

`gap` – intervalo do padding

setSelectedIndex

```
public void setSelectedIndex(int index)
```

Retorna o índice selecionado para esse tabbedpane. O índice deve ser um índice de guia válido.

Parâmetros:

`index` – o índice a ser selecionado

Lança:

`IndexOutOfBoundsException` – se o índice estiver fora de alcance `(index 0 || index >= tab count)`

Relaciona-se com:

`getSelectedIndex()`

24.25 Classe TextArea

24.25.1 Descrição da classe

`com.sun.dtv.lwuit`

```
java.lang.Object
  └com.sun.dtv.lwuit.Component
```

└com.sun.dtv.lwuit.TextArea

Todas as interfaces implementadas

Animated, Animation, StyleListener

Subclasses diretas conhecidas

TextField

public class TextArea

extends Component

Uma região editável multi-linhas opcional que pode exibir texto e permitir um usuário a editá-lo. Dependendo da plataforma, a edição pode ocorrer em uma nova tela. Note que ao criar uma área de texto com uma linha, ela se comportará como um campo de texto e nunca aumentará mais do que isso. No entanto, ao designar um número maior de linhas, a área de texto terá múltiplas linhas com um número mínimo de linhas visíveis; a área de texto aumentará de acordo com seu conteúdo.

24.25.2 Índice de campos

static int **ANY**

Permite qualquer tipo de entrada no campo de texto. Se uma restrição não for suportada por uma implementação subjacente, isso deve ser o padrão.

static int **DECIMAL**

O usuário tem permissão de inserir valores numéricos com frações decimais ideais, por exemplo “-123”, “0.123", ou “.5”.

static int **EMAILADDR**

O usuário tem permissão de inserir um endereço de e-mail.

static int **INITIAL_CAPS_SENTENCE**

Esta *flag* é uma dica para a implementação de que, durante a edição de texto, a letra inicial de cada frase deve estar em caixa-alta.

static int **INITIAL_CAPS_WORD**

Esta *flag* é uma dica para a implementação de que, durante a edição de texto, a palavra inicial de cada frase deve estar em caixa-alta.

static int **NON_PREDICTIVE**

Indica que o texto inserido não consiste em palavras que provavelmente são encontradas em dicionários normalmente usados por esquemas de entrada previsíveis.

static int **NUMERIC**

O usuário tem permissão de inserir apenas um valor inteiro.

static int **PASSWORD**

Indica que o texto inserido é um dado confidencial que deve ser ocultado sempre que possível.

static int **PHONENUMBER**

O usuário tem permissão de inserir um número de telefone.

static int **SENSITIVE**

Indica que o texto inserido é um dado confidencial que a implementação não deve nunca armazenar em um dicionário ou uma tabela para o uso em esquemas de entrada previsíveis, tipo “autocompletar” ou outros

esquemas de entrada acelerados.

`static int` **`UNEDITABLE`**

Indica que a edição não é permitida no momento.

`static int` **`URL`**

O usuário tem permissão de inserir um URL.

Campos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`BOTTOM, BRB_CENTER_OFFSET, BRB_CONSTANT_ASCENT, BRB_CONSTANT_DESCENT, BRB_OTHER, CENTER, LEFT, RIGHT, TOP`

Campos herdados da interface `com.sun.dtv.ui.`**`Animated`**

`ALTERNATING, LOOP, REPEATING`

24.25.3 Índice de construtores

`TextArea``()`

Cria uma área de texto vazia. Esse construtor criará uma área de texto com uma única linha similar ao campo de texto.

`TextArea``(int rows, int columns)`

Cria uma área com as linhas e colunas determinadas.

`TextArea``(int rows, int columns, int constraint)`

Cria uma área com as linhas, colunas e restrições determinadas.

`TextArea``(String text)`

Cria uma área com o texto determinado. Esse construtor criará uma área de texto com uma única linha semelhante a um campo de texto.

`TextArea``(String text, int maxSize)`

Cria uma área com o texto determinado e tamanho máximo. Esse construtor criará uma área de texto com uma única linha semelhante a um campo de texto.

`TextArea``(String text, int rows, int columns)`

Cria uma área com texto, linhas e colunas determinados.

`TextArea``(String text, int rows, int columns, int constraint)`

Cria uma área com texto, linhas, colunas e restrição determinados.

24.25.4 Índice de métodos

`void` **`addActionListener`**`(ActionListener a)`

Adiciona um *listener* da ação que é executado quando a área de texto foi modificada não durante a modificação.

`int` **`getColumns`**`()`

Retorna o número de colunas na área de texto.

`int` **`getConstraint`**`()`

Retorna o valor restrito de edição.

`int` **`getLines`**`()`

Retorna o número de linhas de texto na `TextArea`.

`int` **`getMaxSize`**`()`

Retorna o tamanho máximo para a área de texto.

`int` **`getRows`**`()`

Retorna o número de linhas na área de texto.

`int` **`getRowsGap`**`()`

Obtém o número de intervalos de pixels entre as linhas.

`String` **`getText`**`()`

Retorna o texto na área de texto.

`String` **`getTextAt`**`(int line)`

Retorna o texto na linha determinada da caixa de texto.

`String` **`getUnsupportedChars`**`()`

Caracteres não suportados são um *string* que contém caracteres que causam problemas ao renderizar algumas fontes problemáticas.

`static boolean` **`isAutoDegradeMaxSize`**`()`

Indica se um valor alto para o maxSize padrão deve ser reduzido a um valor menor se a plataforma subjacente abrir uma exceção.

`boolean` **`isEditable`**`()`

Retorna *true* se essa área for editável.

`boolean` **`isGrowByContent`**`()`

Indica que a área de texto deve “aumentar” em altura com base no conteúdo além dos limites indicados pela variável das linhas.

`boolean` **`isScrollableY`**`()`

Indica se o componente deveria/poderia rolar no eixo Y.

`void` **`keyPressed`**`(int keyCode)`

Se esse componente estiver destacado, o evento chave pressionado chamará esse método.

`void` **`keyReleased`**`(int keyCode)`

Se esse componente estiver destacado, o evento chave liberado chamará esse método.

`void` **`paint`**`(Graphics g)`

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

`void` **`pointerReleased`**`(int x, int y)`

Se esse componente estiver destacado, o evento indicador liberado chamará esse método.

`void` **`removeActionListener`**`(ActionListener a)`

Remove um *listener* da ação.

`static void` **`setAutoDegradeMaxSize`**`(boolean value)`

Indica se um valor alto para o maxSize padrão deve ser reduzido a um valor menor se a plataforma subjacente abrir uma exceção.

`void` **`setColumns`**`(int columns)`

Define o número de colunas na área de texto.

`void` **`setConstraint`**`(int constraint)`

Define a restrição.

`static void` **`setDefaultMaxSize`**`(int value)`

Define o limite padrão para o tamanho da caixa de texto original.

`void` **`setEditable`**`(boolean b)`

Define essa área de texto para somente leitura ou editável.

`void` **`setGrowByContent`**`(boolean growByContent)`

Indica que a área de texto deve "aumentar" em altura com base no conteúdo além dos limites indicados pela variável das linhas.

`void` **`setMaxSize`**`(int maxSize)`

Define o tamanho máximo da área de texto.

`void` **`setRows`**`(int rows)`

Define o número de linhas na área de texto.

`void` **`setRowsGap`**`(int rowsGap)`

O intervalo de pixels entre as linhas.

`void` **`setText`**`(String t)`

Define o texto dentro desta área de texto.

`void` **`setUnsupportedChars`**`(String unsupportedChars)`

Caracteres não suportados são um *string* que contém caracteres que causam problemas ao renderizar algumas fontes problemáticas.

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`addFocusListener, animate, calcPreferredSize, contains, deinitialize, getAbsoluteX, getAbsoluteY, getAnimationMode, getBaseline, getBaselineResizeBehavior, getBottomGap, getBounds, getClientProperty, getComponentForm, getDelay, getHeight, getNextFocusDown, getNextFocusLeft, getNextFocusRight, getNextFocusUp, getParent, getPosition, getPreferredSize, getRepetitionMode, getScrollAnimationSpeed, getScrollX, getScrollY, getSideGap, getStyle, getUIID, getWidth, getX, getY, handlesInput, hasFocus, initialize, isEnabled, isFocusable, isFocusPainted, isInitialized, isRunning, isScrollableX, isScrollVisible, isSmoothScrolling, isVisible, jumpTo, keyRepeated, longKeyPress, paintBackgrounds, paintComponent, paintComponent, pointerDragged, pointerPressed, putClientProperty, refreshTheme, removeFocusListener, repaint, repaint, requestFocus, scrollRectToVisible, setAnimationMode, setCellRenderer, setDelay, setEnabled, setFocus, setFocusable, setFocusPainted, setHandlesInput, setHeight, setInitialized, setIsScrollVisible, setNextFocusDown, setNextFocusLeft, setNextFocusRight, setNextFocusUp, setPreferredSize, setRepetitionMode, setScrollAnimationSpeed, setScrollX, setScrollY, setShouldCalcPreferredSize, setSize, setSmoothScrolling, setStyle, setUIID, setVisible, setWidth, setX, setY, start, stop, styleChanged, toString`

24.25.5 Detalhe dos campos

ANY

```
public static final int ANY = 0
```

Permite qualquer tipo de entrada no campo de texto. Se uma restrição não for suportada por uma implementação subjacente, isso deve ser o padrão.

EMAILADDR

```
public static final int EMAILADDR = 1
```

O usuário tem permissão de inserir um endereço de e-mail.

NUMERIC

```
public static final int NUMERIC = 2
```

O usuário tem permissão de inserir apenas um valor inteiro.

PHONENUMBER

```
public static final int PHONENUMBER = 3
```

O usuário tem permissão de inserir um número de telefone.

URL

```
public static final int URL = 4
```

O usuário tem permissão de inserir um URL.

DECIMAL

```
public static final int DECIMAL = 5
```

O usuário tem permissão de inserir valores numéricos com frações decimais ideais, por exemplo “-123”, “0.123", ou “.5”.

PASSWORD

```
public static final int PASSWORD = 65536
```

Indica que o texto inserido é um dado confidencial que deve ser ocultado sempre que possível.

UNEDITABLE

```
public static final int UNEDITABLE = 131072
```

Indica que a edição não é permitida no momento.

SENSITIVE

```
public static final int SENSITIVE = 262144
```

Indica que o texto inserido é um dado confidencial que a implementação não deve nunca armazenar em um dicionário ou uma tabela para o uso em esquemas de entrada previsíveis, tipo “autocompletar” ou outros esquemas de entrada acelerados.

NON_PREDICTIVE

```
public static final int NON_PREDICTIVE = 524288
```

Indica que o texto inserido não consiste em palavras que provavelmente são encontradas em dicionários normalmente usados por esquemas de entrada previsíveis.

INITIAL_CAPS_WORD

```
public static final int INITIAL_CAPS_WORD = 1048576
```

Esta *flag* é uma dica para a implementação de que, durante a edição de texto, a palavra inicial de cada frase deve estar em caixa-alta.

INITIAL_CAPS_SENTENCE

```
public static final int INITIAL_CAPS_SENTENCE = 2097152
```

Esta *flag* é uma dica para a implementação de que, durante a edição de texto, a letra inicial de cada frase deve estar em caixa-alta.

24.25.6 Detalhe dos construtores

TextArea

```
public TextArea(int rows,
                int columns)
```

Cria uma área com as linhas e colunas determinadas.

Parâmetros:

`rows` – o número de linhas

`columns` - o número de colunas

TextArea

```
public TextArea(int rows,
                int columns,
                int constraint)
```

Cria uma área com as linhas, colunas e restrições determinadas.

Parâmetros:

`rows` – o número de linhas

`columns` - o número de colunas

`constraint` – `ANY, EMAILADDR, NUMERIC, PHONENUMBER, URL, DECIMAL` pode ser lógica binária ou `PASSWORD, UNEDITABLE, SENSITIVE, NON_PREDICTIVE, INITIAL_CAPS_SENTENCE, INITIAL_CAPS_WORD.` `Ex.: ANY | PASSWORD.`

TextArea

```
public TextArea(String text,
                int rows,
                int columns)
```

Cria uma área com texto, linhas e colunas determinados.

Parâmetros:

`text` – o texto a ser exibido; se o texto for `null`, a *string* vazia “”deve ser exibida

`rows` – o número de linhas

`columns` - o número de colunas

TextArea

```
public TextArea(String text,
                int rows,
                int columns,
                int constraint)
```

Cria uma área com texto, linhas, colunas e restrição determinados.

Parâmetros:

`text` – o texto a ser exibido; se o texto for `null`, a *string* vazia “”deve ser exibida

`rows` – o número de linhas

`columns` - o número de colunas

`constraint` – `ANY, EMAILADDR, NUMERIC, PHONENUMBER, URL, DECIMAL` pode ser lógica binária ou `PASSWORD, UNEDITABLE, SENSITIVE, NON_PREDICTIVE, INITIAL_CAPS_SENTENCE, INITIAL_CAPS_WORD`. Ex.: `ANY | PASSWORD`.

TextArea

```
public TextArea(String text,
                int maxSize)
```

Cria uma área com o texto determinado e tamanho máximo. Esse construtor criará uma área de texto com uma única linha semelhante a um campo de texto.

Parâmetros:

`text` – o texto a ser exibido; se o texto for `null`, a *string* vazia “”deve ser exibida

`maxSize` – o tamanho máximo da área de texto

TextArea

```
public TextArea(String text)
```

Cria uma área com o texto determinado. Esse construtor criará uma área de texto com uma única linha semelhante a um campo de texto.

Parâmetros:

`text` – o texto a ser exibido; se o texto for `null`, a *string* vazia “”deve ser exibida

TextArea

```
public TextArea()
```

Cria uma área de texto vazia. Esse construtor criará uma área de texto com uma única linha similar ao campo de texto.

24.25.7 Detalhe dos métodos

setConstraint

```
public void setConstraint(int constraint)
```

Define a restrição.

Parâmetros:

`constraint` – `ANY, EMAILADDR, NUMERIC, PHONENUMBER, URL, DECIMAL` pode ser lógica binária ou `PASSWORD, UNEDITABLE, SENSITIVE, NON_PREDICTIVE, INITIAL_CAPS_SENTENCE, INITIAL_CAPS_WORD. Ex.: ANY | PASSWORD.`

Relaciona-se com:

`getConstraint()`

getConstraint

`public int getConstraint()`

Retorna o valor restrito de edição.

Retorna:

o valor restrito de edição

Relaciona-se com:

`setConstraint()`

setText

`public void setText(String t)`

Define o texto dentro desta área de texto.

Parâmetros:

`t` – o novo valor para a área de texto

Relaciona-se com:

`getText()`

getText

`public String getText()`

Retorna o texto na área de texto.

Retorna:

o texto na área de texto

Relaciona-se com:

`setText(java.lang.String)`

isEditable

`public boolean isEditable()`

Retorna *true* se essa área for editável.

Retorna:

true se essa área for editável

setEditable

`public void setEditable(boolean b)`

Define essa área de texto para somente leitura ou editável.

Parâmetros:

`b` – *true* se a área de texto for editável, caso contrário, *false*

getMaxSize

`public int getMaxSize()`

Retorna o tamanho máximo para a área de texto.

Retorna:

o tamanho máximo para a área de texto

Relaciona-se com:

`setMaxSize(int)`

setMaxSize

`public void setMaxSize(int maxSize)`

Define o tamanho máximo da área de texto.

Parâmetros:

`maxSize` - o tamanho máximo da área de texto.

Relaciona-se com:

`getMaxSize()`

keyPressed

`public void keyPressed(int keyCode)`

Se esse componente estiver destacado, o evento chave pressionado chamará esse método.

Substituições:

`keyPressed` na classe `Component`

Parâmetros:

`keyCode` - o valor do código chave a indicar a chave física.

keyReleased

`public void keyReleased(int keyCode)`

Se esse componente estiver destacado, o evento chave liberado chamará esse método.

Substituições:

`keyReleased` na classe `Component`

Parâmetros:

`keyCode` - o valor do código chave a indicar a chave física.

isScrollableY

```
public boolean isScrollableY()
```

Indica se o componente deveria/poderia rolar no eixo Y.

Substituições:

`isScrollableY` na classe `Component`

Retorna:

se o componente é rolável no eixo X

pointerReleased

```
public void pointerReleased(int x,
                            int y)
```

Se esse componente estiver destacado, o evento indicador liberado chamará esse método.

Substituições:

`pointerReleased` na classe `Component`

Parâmetros:

`x` - a coordenada do indicador x

`y` - a coordenada do indicador y

getColumns

```
public int getColumns()
```

Retorna o número de colunas na área de texto.

Retorna:

o número de colunas na área de texto

Relaciona-se com:

`setColumns(int)`

getRows

```
public int getRows()
```

Retorna o número de linhas na área de texto.

Retorna:

o número de linhas na área de texto

Relaciona-se com:

`setRows(int)`

setColumns

```
public void setColumns(int columns)
```

Define o número de colunas na área de texto.

Parâmetros:

`columns` - número de colunas

Relaciona-se com:

`getColumns()`

setRows

`public void setRows(int rows)`

Define o número de linhas na área de texto.

Parâmetros:

`rows` – número de linhas

Relaciona-se com:

`getRows()`

getLines

`public int getLines()`

Retorna o número de linhas de texto na `TextArea`.

Retorna:

o número de linhas de texto na `TextArea`

getTextAt

`public String getTextAt(int line)`

Retorna o texto na linha determinada da caixa de texto.

Parâmetros:

`line` – o número da linha na caixa de texto

Retorna:

o texto na linha determinada

getRowsGap

`public int getRowsGap()`

Obtém o número de intervalos de pixels entre as linhas.

Retorna:

o intervalo entre as linhas em pixels

Relaciona-se com:

`setRowsGap(int)`

setRowsGap

`public void setRowsGap(int rowsGap)`

O intervalo de pixels entre as linhas.

Parâmetros:

`rowsGap` - número de pixels do intervalo entre as linhas

Relaciona-se com:

`getRowsGap()`

paint

`public void paint(Graphics g)`

Descrição copiada da interface: **`Animation`**

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

Especificado por:

`paint` na interface `Animation`

Substituições:

`paint` na classe `Component`

Parâmetros:

`g` - os gráficos do componente

addActionListener

`public void addActionListener(ActionListener a)`

Adiciona um *listener* da ação que é executado quando a área de texto foi modificada não durante a modificação. O **campo** de texto pode nunca iniciar um evento de ação se for editado no local e o usuário nunca sair do campo de texto.

Parâmetros:

`a` - `actionListener`

Relaciona-se com:

`removeActionListener(com.sun.dtv.lwuit.events.ActionListener)`

removeActionListener

`public void removeActionListener(ActionListener a)`

Remove um *listener* da ação.

Parâmetros:

`a` - `actionListener`

Relaciona-se com:

`addActionListener(com.sun.dtv.lwuit.events.ActionListener)`

setDefaultMaxSize

`public static void setDefaultMaxSize(int value)`

Define o limite padrão para o tamanho da caixa de texto original.

Parâmetros:

`value` – valor padrão para o tamanho da caixa de texto original

isGrowByContent

`public boolean isGrowByContent()`

Indica que a área de texto deve “aumentar” em altura com base no conteúdo além dos limites indicados pela variável das linhas.

Retorna:

true se o componente do texto aumentar e, caso o contrário, *false*

setGrowByContent

```
public void setGrowByContent(boolean growByContent)
```

Indica que a área de texto deve “aumentar” em altura com base no conteúdo além dos limites indicados pela variável das linhas.

Parâmetros:

`growByContent` - *true* se o componente do texto aumentar, caso o contrário, *false*

setAutoDegradeMaxSize

```
public static void setAutoDegradeMaxSize(boolean value)
```

Indica se um valor alto para o maxSize padrão deve ser reduzido a um valor menor se a plataforma subjacente abrir uma exceção.

Parâmetros:

`value` - se *true*, a plataforma diminuirá automaticamente.

isAutoDegradeMaxSize

```
public static boolean isAutoDegradeMaxSize()
```

Indica se um valor alto para o maxSize padrão deve ser reduzido a um valor menor se a plataforma subjacente abrir uma exceção.

Retorna:

a atual configuração para diminuir automaticamente.

getUnsupportedChars

```
public String getUnsupportedChars()
```

Caracteres não suportados são um *string* que contém caracteres que causam problemas ao renderizar algumas fontes problemáticas. O mecanismo de renderização pode, assim, removê-los quando estiver desenhando.

Retorna:

String contendo os caracteres não suportados

Relaciona-se com:

`setUnsupportedChars(java.lang.String)`

setUnsupportedChars

```
public void setUnsupportedChars(String unsupportedChars)
```

Caracteres não suportados são um *string* que contém caracteres que causam problemas ao renderizar algumas fontes problemáticas. O mecanismo de renderização pode, assim, removê-los quando estiver desenhando.

Parâmetros:

`unsupportedChars` - String contendo os caracteres não suportados

Relaciona-se com:

`getUnsupportedChars()`

24.26 Classe TextField

24.26.1 Descrição da classe

`com.sun.dtv.lwuit`

```
java.lang.Object
  └com.sun.dtv.lwuit.Component
      └com.sun.dtv.lwuit.TextArea
          └com.sun.dtv.lwuit.TextField
```

Todas as interfaces implementadas

`Animated, Animation, StyleListener`

```
public class TextField
extends TextArea
```

Permite edição no local usando uma API simples sem necessariamente mover-se a caixa de texto nativa externa. O principal defeito nessa abordagem é que a edição não pode suportar recursos como o T9 e pode não ter o mesmo mapeamento principal ou comportamento da entrada do texto original.

Note que, devido a limitações da área de texto e no campo de texto, os modos de entrada na área de texto não são devidamente suportados, já que não funcionarão devidamente em vários aparelhos. Para limitar modos de entrada, use o método `setInputModeOrder`. Toda as constantes declaradas na `TextArea` são ignoradas com exceção de `PASSWORD`.

Campos herdados da classe `com.sun.dtv.lwuit.`**`TextArea`**

`ANY, DECIMAL, EMAILADDR, INITIAL_CAPS_SENTENCE, INITIAL_CAPS_WORD, NON_PREDICTIVE, NUMERIC, PASSWORD, PHONENUMBER, SENSITIVE, UNEDITABLE, URL`

Campos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`BOTTOM, BRB_CENTER_OFFSET, BRB_CONSTANT_ASCENT, BRB_CONSTANT_DESCENT, BRB_OTHER, CENTER, LEFT, RIGHT, TOP`

Campos herdados da interface `com.sun.dtv.ui.`**`Animated`**

`ALTERNATING, LOOP, REPEATING`

24.26.2 Índice de construtores

`TextField``()`

Construtor padrão.

`TextField``(int columns)`

Cria um campo de texto com espaço reservado para colunas.

`TextField(String text)`

Cria um campo de texto.

`TextField(String text, int columns)`

Cria um campo de texto.

24.26.3 Índice de métodos

`void addDataChangeListener(DataChangedListener d)`

Adiciona um *listener* para eventos de alteração de dados. Deve ser executado a cada alteração feita no campo de texto.

`static void addInputMode(String name, Hashtable values, boolean firstUpcase)`

Adiciona uma nova *hashtable* do modo de entrada com o nome determinado e o conjunto de valores.

`boolean animate()`

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*.

`int getCommitTimeout()`

O total de tempo em milissegundos que levará para a alteração ser confirmada no campo.

`int getCursorBlinkTimeOff()`

O total de tempo em milissegundos que o cursor fica invisível.

`int getCursorBlinkTimeOn()`

O total de tempo em milissegundos que o cursor fica visível.

`int getCursorPosition()`

Retorna a posição do cursor.

`static String[] getDefaultInputModeOrder()`

Retorna a ordem na qual os modos de entrada são alterados por padrão.

`String getInputMode()`

Retorna o modo atual de entrada selecionado.

`String[] getInputModeOrder()`

Retorna a ordem na qual os modos de entrada são alterados.

`static char[] getSymbolTable()`

Retorna a tabela de símbolos para o dispositivo.

`boolean isPendingCommit()`

Retorna *true* se o campo de texto estiver esperando por uma confirmação na edição.

`static boolean isQwertyAutoDetect()`

Indica se o campo de texto deve tentar detectar automaticamente o qwerty e trocar a *flag* do dispositivo qwerty implicitamente.

`static boolean isQwertyDevice()`

O valor padrão para a *flag* do qwerty para que ele não precise ser configurado para cada campo de texto individualmente.

`boolean` **`isQwertyInput`**`()`

True se for um dispositivo qwerty ou um dispositivo que estiver atualmente em modo qwerty.

`boolean` **`isReplaceMenu`**`()`

Indica se o menu do formulário deve ser substituído pelos comandos T9/Clear durante a interatividade do campo de texto.

`static boolean` **`isReplaceMenuDefault`**`()`

Indica se o menu do formulário deve ser substituído pelos comandos T9/Clear durante a interatividade do campo de texto.

`boolean` **`isUseSoftkeys`**`()`

Quando configurado para *true*, as *softkeys* são usadas para excluir a funcionalidade.

`void` **`keyPressed`**`(int keyCode)`

Se esse componente estiver destacado, o evento chave pressionado chamará esse método.

`void` **`keyReleased`**`(int keyCode)`

Se esse componente estiver destacado, o evento chave liberado chamará esse método.

`void` **`paint`**`(Graphics g)`

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

`void` **`pointerReleased`**`(int x, int y)`

Se esse componente estiver destacado, o evento indicador liberado chamará esse método.

`void` **`removeDataChangeListener`**`(DataChangedListener d)`

Remove o *listener* para eventos de alteração de dados.

`static void` **`setClearText`**`(String text)`

Define o texto que deve aparecer na soft key clear.

`void` **`setCommitTimeout`**`(int commitTimeout)`

O total de tempo em milissegundos que levará para a alteração ser confirmada no campo.

`void` **`setCursorBlinkTimeOff`**`(int time)`

O total de tempo em milissegundos que o cursor fica invisível.

`void` **`setCursorBlinkTimeOn`**`(int time)`

O total de tempo em milissegundos que o cursor fica visível.

`void` **`setCursorPosition`**`(int pos)`

Define a posição do cursor.

`static void` **`setDefaultInputModeOrder`**`(String[] order)`

Define a ordem na qual os modos de entrada são alterados por padrão e permite desabilitar/ocultar um modo de entrada.

`void` **`setEditable`**`(boolean b)`

Define essa área de texto para somente leitura ou editável.

`void` **`setInputMode`**`(String mode)`

Define o atual modo de entrada selecionado combinando um dos modos de entrada existentes.

`void` **`setInputModeOrder`**`(String[] order)`

Define a ordem na qual os modos de entrada são alterados e permite desabilitar/ocultar um modo de entrada.

`static void` **`setQwertyAutoDetect`**`(boolean v)`

Indica se o campo de texto deve tentar detectar automaticamente o qwerty e trocar a *flag* do dispositivo qwerty implicitamente.

`static void` **`setQwertyDevice`**`(boolean v)`

O valor padrão para a *flag* do qwerty para que ele não precise ser configurado para cada campo de texto individualmente.

`void` **`setQwertyInput`**`(boolean qwerty)`

Define esse dispositivo para operar ou não operar mais no modo qwerty.

`void` **`setReplaceMenu`**`(boolean replaceMenu)`

Indica se o menu do formulário deve ser substituído pelos comandos T9/Clear durante a interatividade do campo de texto.

`static void` **`setReplaceMenuDefault`**`(boolean replaceMenu)`

Indica se o menu do formulário deve ser substituído pelos comandos T9/Clear durante a interatividade do campo de texto.

`static void` **`setSymbolTable`**`(char[] table)`

Define a tabela de símbolos para mostrar quando o usuário clicar no botão da tabela de símbolos.

`static void` **`setT9Text`**`(String text)`

Define o texto que deve aparecer na soft key T9.

`void` **`setText`**`(String text)`

Define o texto dentro desta área de texto.

`void` **`setUseSoftkeys`**`(boolean useSoftkeys)`

Quando configurado para *true*, as *softkeys* são usadas para excluir a funcionalidade.

Métodos herdados da classe `com.sun.dtv.lwuit.`**`TextArea`**

`addActionListener, getColumns, getConstraint, getLines, getMaxSize, getRows, getRowsGap, getText, getTextAt, getUnsupportedChars, isAutoDegradeMaxSize, isEditable, isGrowByContent, isScrollableY, removeActionListener, setAutoDegradeMaxSize, setColumns, setConstraint, setDefaultMaxSize, setGrowByContent, setMaxSize, setRows, setRowsGap, setUnsupportedChars`

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`addFocusListener, calcPreferredSize, contains, deinitialize, getAbsoluteX, getAbsoluteY, getAnimationMode, getBaseline, getBaselineResizeBehavior, getBottomGap, getBounds, getClientProperty, getComponentForm, getDelay, getHeight, getNextFocusDown, getNextFocusLeft, getNextFocusRight, getNextFocusUp, getParent, getPosition, getPreferredSize, getRepetitionMode, getScrollAnimationSpeed, getScrollX, getScrollY, getSideGap, getStyle, getUIID, getWidth, getX, getY, handlesInput, hasFocus, initialize, isEnabled, isFocusable, isFocusPainted, isInitialized, isRunning, isScrollableX, isScrollVisible, isSmoothScrolling, isVisible, jumpTo, keyRepeated, longKeyPress, paintBackgrounds, paintComponent, paintComponent, pointerDragged, pointerPressed, putClientProperty, refreshTheme, removeFocusListener, repaint, repaint, requestFocus, scrollRectToVisible, setAnimationMode, setCellRenderer, setDelay, setEnabled, setFocus, setFocusable,`

setFocusPainted, setHandlesInput, setHeight, setInitialized, setIsScrollVisible, setNextFocusDown, setNextFocusLeft, setNextFocusRight, setNextFocusUp, setPreferredSize, setRepetitionMode, setScrollAnimationSpeed, setScrollX, setScrollY, setShouldCalcPreferredSize, setSize, setSmoothScrolling, setStyle, setUIID, setVisible, setWidth, setX, setY, start, stop, styleChanged, toString

24.26.4 Detalhe dos construtores

TextField

```
public TextField()
```

Construtor padrão.

TextField

```
public TextField(int columns)
```

Cria um campo de texto com espaço reservado para colunas.

Parâmetros:

`columns` - número de colunas

TextField

```
public TextField(String text)
```

Cria um campo de texto.

Parâmetros:

`text` – conteúdo do textfield

TextField

```
public TextField(String text,
                 int columns)
```

Cria um campo de texto.

Parâmetros:

`text` – conteúdo do campo de texto

`columns` - número de colunas

24.26.5 Detalhe dos métodos

setClearText

```
public static void setClearText(String text)
```

Define o texto que deve aparecer na soft key clear.

Parâmetros:

`text` – o texto da soft key clear

setT9Text

```
public static void setT9Text(String text)
```

Define o texto que deve aparecer na *softkey* T9.

Parâmetros:

`text` – o texto para a *softkey* T9

isPendingCommit

`public boolean isPendingCommit()`

Retorna *true* se o campo de texto estiver esperando por uma confirmação na edição.

Retorna:

true se houver uma confirmação pendente, caso contrário, *false*

setCommitTimeout

`public void setCommitTimeout(int commitTimeout)`

O total de tempo em milissegundos que levará para a alteração ser confirmada no campo.

Parâmetros:

`commitTimeout` – indica o total de tempo que deve passar para uma confirmação para ocorrer automaticamente

Relaciona-se com:

`getCommitTimeout()`

getCommitTimeout

`public int getCommitTimeout()`

O total de tempo em milissegundos que levará para a alteração ser confirmada no campo.

Retorna:

o total de tempo que deve passar para uma confirmação ocorrer automaticamente

Relaciona-se com:

`setCommitTimeout(int)`

setInputMode

`public void setInputMode(String mode)`

Define o atual modo de entrada selecionado combinando um dos modos de entrada existentes.

Parâmetros:

`mode` - o nome de exibição do modo de entrada. Por padrão, os seguintes modos são suportados: Abc, ABC, abc, 123

Relaciona-se com:

`getInputMode()`

getInputMode

`public String getInputMode()`

Retorna o modo atual de entrada selecionado.

Retorna:

o nome de exibição do modo de entrada. Por padrão, os seguintes modos são suportados: Abc, ABC, abc, 123

Relaciona-se com:

`setInputMode(java.lang.String)`

addInputMode

```
public static void addInputMode(String name,
                                 Hashtable values,
                                 boolean firstUpcase)
```

Adiciona uma nova *hashtable* do modo de entrada com o nome determinado e o conjunto de valores.

Parâmetros:

`name` – um nome de exibição único para o modo de entrada, por ex.: ABC, 123 etc...

`values` – O botão da *hashtable* é um keyCode íntegro, e o valor é um String contendo os caracteres para alterar entre o keycode determinado

`firstUpcase` – indica se esse modo de entrada é usado para o caso especial em que a primeira letra é uma letra em caixa-alta

getInputModeOrder

```
public String[] getInputModeOrder()
```

Retorna a ordem na qual os modos de entrada são alterados.

Retorna:

a ordem na qual os modos de entrada são alterados

Relaciona-se com:

`setInputModeOrder(java.lang.String[])`

setInputModeOrder

```
public void setInputModeOrder(String[] order)
```

Define a ordem na qual os modos de entrada são alterados e permite desabilitar/ocultar um modo de entrada.

Parâmetros:

`order` - a ordem para os modos de entrada nesse campo

Relaciona-se com:

`getInputModeOrder()`

getDefaultInputModeOrder

```
public static String[] getDefaultInputModeOrder()
```

Retorna a ordem na qual os modos de entrada são alterados por padrão.

Retorna:

a ordem padrão para os modos de entrada nesse campo

Relaciona-se com:

`setDefaultInputModeOrder(java.lang.String[])`

setDefaultInputModeOrder

`public static void setDefaultInputModeOrder(String[] order)`

Define a ordem na qual os modos de entrada são alterados por padrão e permite desabilitar/ocultar um modo de entrada.

Parâmetros:

`order` - a ordem para os modos de entrada em todos os campos criados no futuro

Relaciona-se com:

`getDefaultInputModeOrder()`

setCursorPosition

`public void setCursorPosition(int pos)`

Define a posição do cursor.

Parâmetros:

`pos` - a posição do cursor

Relaciona-se com:

`getCursorPosition()`

getCursorPosition

`public int getCursorPosition()`

Retorna a posição do cursor.

Retorna:

a posição do cursor

Relaciona-se com:

`setCursorPosition(int)`

setText

`public void setText(String text)`

Define o texto dentro desta área de texto.

Substituições:

`setText` na classe `TextArea`

isQwertyInput

`public boolean isQwertyInput()`

True se for um dispositivo qwerty ou um dispositivo que estiver atualmente em modo qwerty.

Retorna:

padrões atuais para *false*

setQwertyInput

`public void setQwertyInput(boolean qwerty)`

Define esse dispositivo para operar ou não operar mais no modo qwerty.

Parâmetros:

`qwerty` – indica se o dispositivo deve ser definido para o modo qwerty ou não

keyReleased

`public void keyReleased(int keyCode)`

Se esse componente estiver destacado, o evento chave liberado chamará esse método.

Substituições:

`keyReleased` na classe `TextArea`

Parâmetros:

`keyCode` - o valor do código chave a indicar a chave física.

getSymbolTable

`public static char[] getSymbolTable()`

Retorna a tabela de símbolos para o dispositivo.

Retorna:

tabela de símbolo do dispositivo

Relaciona-se com:

`setSymbolTable(char[])`

setSymbolTable

`public static void setSymbolTable(char[] table)`

Define a tabela de símbolos para mostrar quando o usuário clicar no botão da tabela de símbolos.

Parâmetros:

`table` – a tabela de símbolo

Relaciona-se com:

`getSymbolTable()`

setEditable

`public void setEditable(boolean b)`

Define essa área de texto para somente leitura ou editável.

Substituições:

`setEditable` na classe `TextArea`

Parâmetros:

`b` – *true* se a área de texto for editável, caso contrário, *false*

keyPressed

`public void keyPressed(int keyCode)`

Se esse componente estiver destacado, o evento chave pressionado chamará esse método.

Substituições:

`keyPressed` na classe `TextArea`

Parâmetros:

`keyCode` - o valor do código chave a indicar a chave física.

paint

`public void paint(Graphics g)`

Descrição copiada da interface: **`Animation`**

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

Especificado por:

`paint` na interface `Animation`

Substituições:

`paint` na classe `TextArea`

Parâmetros:

`g` - os gráficos do componente

setCursorBlinkTimeOn

`public void setCursorBlinkTimeOn(int time)`

O total de tempo em milissegundos que o cursor fica visível.

Parâmetros:

`time` - tempo em milissegundos que o cursor fica visível

Relaciona-se com:

`getCursorBlinkTimeOn()`

setCursorBlinkTimeOff

`public void setCursorBlinkTimeOff(int time)`

O total de tempo em milissegundos que o cursor fica invisível.

Parâmetros:

`time` - tempo em milissegundos que o cursor fica invisível

Relaciona-se com:

`getCursorBlinkTimeOff()`

getCursorBlinkTimeOn

```
public int getCursorBlinkTimeOn()
```

O total de tempo em milissegundos que o cursor fica visível.

Retorna:

tempo em milissegundos que o cursor fica visível

Relaciona-se com:

`setCursorBlinkTimeOn(int)`

getCursorBlinkTimeOff

```
public int getCursorBlinkTimeOff()
```

O total de tempo em milissegundos que o cursor fica invisível.

Retorna:

tempo em milissegundos que o cursor fica invisível

Relaciona-se com:

`setCursorBlinkTimeOff(int)`

animate

```
public boolean animate()
```

Descrição copiada da interface: **`Animation`**

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*. É chamado uma vez para cada quadro.

Especificado por:

`animate` na interface `Animation`

Substituições:

`animate` na classe `Component`

Retorna:

true se uma repintura for desejada ou *false* se a repintura não for necessária

pointerReleased

```
public void pointerReleased(int x,
                            int y)
```

Se esse componente estiver destacado, o evento indicador liberado chamará esse método.

Substituições:

`pointerReleased` na classe `TextArea`

Parâmetros:

`x` - a coordenada do indicador x

`y` - a coordenada do indicador y

isUseSoftkeys

```
public boolean isUseSoftkeys()
```

Quando configurado para *true*, as *softkeys* são usadas para excluir a funcionalidade.

Retorna:

true se as *softkeys* são usadas para habilitar a funcionalidade de exclusão

setUseSoftkeys

```
public void setUseSoftkeys(boolean useSoftkeys)
```

Quando configurado para *true*, as *softkeys* são usadas para excluir a funcionalidade.

Parâmetros:

`useSoftkeys` – valor indicado se as *softkeys* são usados para habilitar a funcionalidade de exclusão

addDataChangeListener

```
public void addDataChangeListener(DataChangedListener d)
```

Adiciona um *listener* para eventos de alteração de dados. Deve ser executado a cada alteração feita no campo de texto.

Parâmetros:

`d` – o `DataChangedListener`

Relaciona-se com:

`removeDataChangeListener(com.sun.dtv.lwuit.events.DataChangedListener)`

removeDataChangeListener

```
public void removeDataChangeListener(DataChangedListener d)
```

Remove o *listener* para eventos de alteração de dados.

Parâmetros:

`d` – o `DataChangedListener`

Relaciona-se com:

`addDataChangeListener(com.sun.dtv.lwuit.events.DataChangedListener)`

isReplaceMenu

```
public boolean isReplaceMenu()
```

Indica se o menu do formulário deve ser substituído pelos comandos `T9/Clear` durante a interatividade do campo de texto.

Retorna:

true se o menu do formulário deve ser substituído pelos comandos `T9/Clear` durante a interatividade com o campo de texto

setReplaceMenu

```
public void setReplaceMenu(boolean replaceMenu)
```

Indica se o menu do formulário deve ser substituído pelos comandos `T9/Clear` durante a interatividade do campo

de texto.

Parâmetros:

`replaceMenu` - *true* se o menu do formulário deve ser substituído pelos comandos `T9/Clear` durante a interatividade com o campo de texto

isReplaceMenuDefault

```
public static boolean isReplaceMenuDefault()
```

Indica se o menu do formulário deve ser substituído pelos comandos `T9/Clear` durante a interatividade do campo de texto.

Retorna:

valor padrão para o valor de retorno de `isReplaceMenu()`

Relaciona-se com:

`isReplaceMenu()`, `setReplaceMenuDefault(boolean)`

setReplaceMenuDefault

```
public static void setReplaceMenuDefault(boolean replaceMenu)
```

Indica se o menu do formulário deve ser substituído pelos comandos `T9/Clear` durante a interatividade do campo de texto.

Parâmetros:

`replaceMenu` - valor padrão do valor de retorno de `isReplaceMenu()`

Relaciona-se com:

`isReplaceMenu()`, `isReplaceMenuDefault()`

setQwertyAutoDetect

```
public static void setQwertyAutoDetect(boolean v)
```

Indica se o campo de texto deve tentar detectar automaticamente o qwerty e trocar a *flag* do dispositivo qwerty implicitamente.

Parâmetros:

`v` – valor que indica se o campo de texto deve tentar detectar automaticamente o qwerty e trocar a *flag* do dispositivo qwerty implicitamente

setQwertyDevice

```
public static void setQwertyDevice(boolean v)
```

O valor padrão para a *flag* do qwerty para que ele não precise ser configurado para cada campo de texto individualmente.

Parâmetros:

`v` - valor padrão para a *flag* do qwerty para que ele não precise ser configurado para cada campo de texto individualmente

isQwertyAutoDetect

```
public static boolean isQwertyAutoDetect()
```

Indica se o campo de texto deve tentar detectar automaticamente o qwerty e trocar a *flag* do dispositivo qwerty implicitamente.

Retorna:

valor que indica se o campo de texto deve tentar detectar automaticamente o qwerty e trocar a *flag* do dispositivo qwerty implicitamente

isQwertyDevice

```
public static boolean isQwertyDevice()
```

O valor padrão para a *flag* do qwerty para que ele não precise ser configurado para cada campo de texto individualmente.

Retorna:

valor padrão para a *flag* do qwerty para que ele não precise ser configurado para cada campo de texto individualmente.

25 Pacote com.sun.dtv.lwuit.animations

25.1 Descrição do pacote

Todos os componentes são animáveis por animações potenciais e adicionais (não-relacionadas a um componente específico) podem ser instalados instantaneamente, transições entre formas também são tratadas como parte desse pacote. O tratamento e a pintura da sequência de animação são executados de maneira uniforme para evitar a complexidade do sequenciamento e sua potencial penalidade de performance em dispositivos pequenos.

NOTA Este pacote está especificado desde o Java DTV 1.0.

25.2 Índice de interfaces

Animation

Permite que qualquer objeto reaja aos eventos e desenhe uma animação em um intervalo fixo.

25.3 Índice de classes

CommonTransitions

Contém animações de transição comuns incluindo as seguintes.

Motion

Abstrai a noção de movimento físico ao longo do tempo de um local numérico para outro.

Transition

Representa a animação de transição entre dois formulários. Essa classe é usada internamente pelo `DTVContainer` para executar uma animação ao mover de um formulário a outro.

25.4 Interface Animation

25.4.1 Descrição da interface

`com.sun.dtv.lwuit.animations`

Todas as classes implementadoras conhecidas

`AnimatedMatte, AWTComponent, Button, Calendar, CheckBox, ComboBox, CommonTransitions, Component, Container, DefaultListCellRenderer, Dialog, DTVContainer, Form, Label, List, MediaComponent, RadioButton, StaticAnimation, TabbedPane, TextArea, TextField, Transition`

`public interface Animation`

Permite que qualquer objeto reaja aos eventos e desenhe uma animação em um intervalo fixo. Todos os métodos de animação são executados na EDT. Para fins de simplicidade, todos os componentes são animáveis. No entanto, nenhuma animação aparecerá a não ser que seja explicitamente registrada no formulário principal. Para parar o callbacks de animação, a animação deve ser explicitamente removida do formulário (note que isso é diferente de remover o componente do formulário.).

25.4.2 Índice de métodos

`boolean` **`animate`**`()`

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*.

`void` **`paint`**`(Graphics g)`

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

25.4.3 Detalhe dos métodos

animate

`boolean animate()`

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*. É chamado uma vez para cada quadro.

Retorna:

true se uma repintura for desejada ou *false* se a repintura não for necessária

paint

`void paint(Graphics g)`

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

25.5 Classe CommonTransitions

25.5.1 Descrição da classe

`com.sun.dtv.lwuit.animations`

```
java.lang.Object
  └com.sun.dtv.lwuit.animations.Transition
      └com.sun.dtv.lwuit.animations.CommonTransitions
```

Todas as interfaces implementadas

Animated, Animation

```
final public class CommonTransitions
extends Transition
```

Contém animações de transição comuns incluindo as seguintes.

Deslizar: o formulário existente desliza para sair da tela enquanto o formulário novo desliza para entrar nela.

Dissolver: os componentes dissolvem para entrar na tela ou sair dela.

Instâncias dessa classe são criadas usando métodos de fabricação.

25.5.2 Índice de campos

`static int` **`SLIDE_HORIZONTAL`**

Deslizar a transição horizontalmente.

`static int` **`SLIDE_VERTICAL`**

Deslizar a transição verticalmente.

Campos herdados da interface `com.sun.dtv.ui.`**`Animated`**

`ALTERNATING, LOOP, REPEATING`

25.5.3 Índice de métodos

`boolean` **`animate`**`()`

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*.

`void` **`cleanup`**`()`

Operação opcional para limpar o lixo deixado por uma transição em andamento.

`Transition` **`copy`**`()`

Copia uma transição.

`static CommonTransitions` **`createEmpty`**`()`

Cria uma transição vazia que não faz nada.

`static CommonTransitions` **`createFade`**`(int duration)`

Cria uma transição de um formulário original para outro formulário por dissolução.

`static CommonTransitions` **`createSlide`**`(int type, boolean forward, int duration)`

Cria uma transição por deslizamento com duração e direção determinadas.

`static CommonTransitions` **`createSlide`**`(int type, boolean forward, int duration, boolean drawDialogMenu)`

Cria uma transição por deslizamento com duração e direção determinadas.

`Motion` **`getMotion`**`()`

Motion representa o movimento físico em uma transição. Pode ser substituído pelo um usuário para proporcionar uma sensação física mais apropriada.

`void` **`initTransition`**`()`

Callback que é executado antes das transições começarem. O formulário fonte pode ser `null` para o primeiro

formulário do aplicativo.

```
void paint(Graphics g)
```

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

```
void setMotion(Motion motion)
```

Motion representa o movimento físico em uma transição. Pode ser substituído pelo um usuário para proporcionar uma sensação física mais apropriada.

Métodos herdados da classe `com.sun.dtv.lwuit.animations.Transition`

```
getAnimationMode, getDelay, getDestination, getPosition, getRepetitionMode,
getSource, init, isRunning, jumpTo, setAnimationMode, setDelay, setRepetitionMode,
start, stop
```

25.5.4 Detalhe dos campos

SLIDE_HORIZONTAL

```
public static final int SLIDE_HORIZONTAL = 0
```

Deslizar a transição horizontalmente.

Relaciona-se com:

```
createSlide()
```

SLIDE_VERTICAL

```
public static final int SLIDE_VERTICAL = 1
```

Deslizar a transição verticalmente.

Relaciona-se com:

```
createSlide()
```

25.5.5 Detalhe dos métodos

createEmpty

```
public static CommonTransitions createEmpty()
```

Cria uma transição vazia que não faz nada. Isso tem o mesmo efeito que configurar uma transição para `null`.

Retorna:

um objeto `CommonTransitions` vazio

createSlide

```
public static CommonTransitions createSlide(int type,
                                            boolean forward,
                                            int duration)
```

Cria uma transição por deslizamento com duração e direção determinadas.

Parâmetros:

`type` – o tipo pode ser vertical ou horizontal, o que se refere à direção de movimento da transição

`forward` - avançar é um valor boleano; representa as direções de alterações de formulários, por exemplo, para

um tipo horizontal. True significa movimento horizontal para a direita.

`duration` – representa o tempo que a transição deve levar em milissegundos

Retorna:

um objeto de transição

createSlide

```
public static CommonTransitions createSlide(int type,
                                            boolean forward,
                                            int duration,
                                            boolean drawDialogMenu)
```

Cria uma transição por deslizamento com duração e direção determinadas.

Parâmetros:

`type` – o tipo pode ser vertical ou horizontal, o que se refere à direção de movimento da transição

`forward` - avançar é um valor boleano; representa as direções de alterações de formulários, por exemplo, para um tipo horizontal. True significa movimento horizontal para a direita.

`duration` – representa o tempo que a transição deve levar em milissegundos

`drawDialogMenu` – indica que o menu da caixa de diálogo deve ser mantido durante uma transição de slides. Isso só é relevante para transições de entrada/saída de caixas de diálogo.

Retorna:

um objeto de transição

createFade

```
public static CommonTransitions createFade(int duration)
```

Cria uma transição de um formulário original para outro formulário por dissolução.

Parâmetros:

`duration` – representa o tempo que a transição deve levar em milissegundos

Retorna:

um objeto de transição

initTransition

```
public void initTransition()
```

Callback que é executado antes das transições começarem. O formulário fonte pode ser `null` para o primeiro formulário do aplicativo.

Substituições:

`initTransition` na classe `Transition`

animate

```
public boolean animate()
```

Descrição copiada da interface: **`Animation`**

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*. É chamado uma vez para cada quadro.

Especificado por:

`animate` na interface `Animation`

Substituições:

`animate` na classe `Transition`

Retorna:

true se uma repintura for desejada ou *false* se a repintura não for necessária

paint

```
public void paint(Graphics g)
```

Descrição copiada da interface: **`Animation`**

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

Especificado por:

`paint` na interface `Animation`

Substituições:

`paint` na classe `Transition`

cleanup

```
public void cleanup()
```

Operação opcional para limpar o lixo deixado por uma transição em andamento.

Substituições:

`cleanup` na classe `Transition`

getMotion

```
public Motion getMotion()
```

`Motion` representa o movimento físico em uma transição. Pode ser substituído pelo um usuário para proporcionar uma sensação física mais apropriada.

Retorna:

a instância da classe `motion` usada por essa transição

Relaciona-se com:

`setMotion(com.sun.dtv.lwuit.animations.Motion)`

setMotion

```
public void setMotion(Motion motion)
```

`Motion` representa o movimento físico em uma transição. Pode ser substituído pelo um usuário para proporcionar uma sensação física mais apropriada.

Parâmetros:

`motion` – nova instância da classe motion que deve ser usada pela transição

Relaciona-se com:

`getMotion()`

copy

`public Transition copy()`

Copia uma transição.

Substituições:

`copy` na classe `Transition`

Retorna:

a transição copiada

25.6 Classe Motion

25.6.1 Descrição da classe

`com.sun.dtv.lwuit.animations`

```
java.lang.Object
  └com.sun.dtv.lwuit.animations.Motion
```

`public class Motion`

`extends Object`

Abstrai a noção de movimento físico ao longo do tempo de um local numérico para outro. Esta classe pode ser subclasse para implementar qualquer equação de movimento para efeitos de física adequados.

Essa classe é baseada no método `System.currentTimeMillis()` para possibilitar transições entre as coordenadas. O `motion` pode ser subclassificado para fornecer todo tipo de sensação de movimento, de parabólica a curva e linear. A implementação padrão fornece um único algoritmo que proporciona a sensação de aceleração e desaceleração.

25.6.2 Índice de construtores

`protected` **`Motion`**`(int sourceValue, float initVelocity, float friction)`

Cria um motion de velocidade.

`protected` **`Motion`**`(int sourceValue, int destinationValue, int duration)`

Cria um motion de ponto/destino.

25.6.3 Índice de métodos

`static Motion` **`createFrictionMotion`**`(int sourceValue, float initVelocity, float friction)`

Cria um motion de fricção que inicia na fonte com a velocidade inicial e fricção.

`static Motion` **`createLinearMotion`**`(int sourceValue, int destinationValue, int duration)`

Cria um motion linear que inicia no valor fonte e vai até o valor de destino.

`static Motion` **`createSplineMotion`**`(int sourceValue, int destinationValue, int duration)`

Cria um motion curvo que inicia no valor fonte e vai até o valor de destino.

`int` **`getSourceValue`**`()`

O número a partir do qual estamos iniciando (normalmente indica a posição inicial da animação).

`int` **`getValue`**`()`

Retorna o valor do motion para o horário atual.

`boolean` **`isFinished`**`()`

Retorna *true* se o `motion` tiver executado seu curso e terminado, significando que o horário atual é maior que startTime + duração.

`void` **`setSourceValue`**`(int sourceValue)`

O número a partir do qual estamos iniciando (normalmente indica a posição inicial da animação).

`void` **`start`**`()`

Define o horário de início para o horário atual.

25.6.4 Detalhe dos construtores

Motion

```
protected Motion(int sourceValue,
                 int destinationValue,
                 int duration)
```

Cria um `motion` de ponto/destino.

Parâmetros:

`sourceValue` – valor inicial

`destinationValue` – valor de destino

`duration` – duração do `motion`

Motion

```
protected Motion(int sourceValue,
                 float initVelocity,
                 float friction)
```

Cria um motion de velocidade.

Parâmetros:

`sourceValue` – valor inicial

`initVelocity` - velocidade inicial

`friction` - grau de fricção

25.6.5 Detalhe dos métodos

createLinearMotion

```
public static Motion createLinearMotion(int sourceValue,
                                        int destinationValue,
                                        int duration)
```

Cria um motion linear que inicia no valor fonte e vai até o valor de destino.

Parâmetros:

`sourceValue` - o número a partir do qual estamos iniciando (normalmente indica a posição inicial da animação)

`destinationValue` – o número ao qual estamos conduzindo (normalmente indica o destino da animação)

`duration` – o comprimento do motion em milissegundos (o tempo que leva do `sourceValue` até o `destinationValue`)

Retorna:

um objeto `Motion`

createSplineMotion

```
public static Motion createSplineMotion(int sourceValue,
                                        int destinationValue,
                                        int duration)
```

Cria um motion curvo que inicia no valor fonte e vai até o valor de destino.

Parâmetros:

`sourceValue` - o número a partir do qual estamos iniciando (normalmente indica a posição inicial da animação)

`destinationValue` – o número ao qual estamos conduzindo (normalmente indica o destino da animação)

`duration` – o comprimento do motion em milissegundos (o tempo que leva do `sourceValue` até o `destinationValue`)

Retorna:

um objeto `Motion`

createFrictionMotion

```
public static Motion createFrictionMotion(int sourceValue,
                                          float initVelocity,
                                          float friction)
```

Cria um motion de fricção que inicia na fonte com a velocidade inicial e fricção.

Parâmetros:

`sourceValue` - o número a partir do qual estamos iniciando (normalmente indica a posição inicial da animação)

`initVelocity` - a velocidade inicial

`friction` - a fricção do `motion`

Retorna:

um objeto `Motion`

start

```
public void start()
```

Define o horário de início para o horário atual.

isFinished

```
public boolean isFinished()
```

Retorna *true* se o `motion` tiver executado seu curso e terminado, significando que o horário atual é maior que `startTime` + duração.

Retorna:

true se `System.currentTimeMillis()` > duração + `startTime`

getValue

```
public int getValue()
```

Retorna o valor do `motion` para o horário atual. O valor é dependente do tipo de `motion`.

Retorna:

um valor que é relativo ao valor fonte

getSourceValue

```
public int getSourceValue()
```

O número a partir do qual estamos iniciando (normalmente indica a posição inicial da animação).

Retorna:

a posição inicial

Relaciona-se com:

`setSourceValue(int)`

setSourceValue

```
public void setSourceValue(int sourceValue)
```

O número a partir do qual estamos iniciando (normalmente indica a posição inicial da animação).

Parâmetros:

`sourceValue` - a posição inicial

Relaciona-se com:

`getSourceValue()`

25.7 Classe Transition

25.7.1 Descrição da classe

`com.sun.dtv.lwuit.animations`

```
java.lang.Object
  └com.sun.dtv.lwuit.animations.Transition
```

Todas as interfaces implementadas

Animated, Animation

Subclasses diretas conhecidas

`CommonTransitions`

```
abstract public class Transition
```

`extends Object`

`implements Animation, Animated`

Representa a animação de transição entre dois formulários. Essa classe é usada internamente pelo `DTVContainer` para executar uma animação ao mover de um formulário a outro. Uma transição pode ser instalada em um objeto `Form` usando as transições de entrada/saída. Para fácil utilização, `LookAndFeel` tem suporte para transições padrão.

Campos herdados da interface `com.sun.dtv.ui.`**`Animated`**

`ALTERNATING, LOOP, REPEATING`

25.7.2 Índice de construtores

`Transition``()`

25.7.3 Índice de métodos

`abstract boolean` **`animate`**`()`

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*.

`void` **`cleanup`**`()`

Operação opcional para limpar o lixo deixado por uma transição em andamento.

`abstract Transition` **`copy`**`()`

Copia uma transição.

`int` **`getAnimationMode`**`()`

Obtém o modo de animação para essa animação.

`int` **`getDelay`**`()`

Obtém o atraso após cada imagem dessa animação ao rodar.

`Component` **`getDestination`**`()`

Retorna o formulário de destino que deve ser definido uma vez que a animação estiver completa.

`int` **`getPosition`**`()`

Obtém a posição atual que a animação ocupa no momento da chamada do método.

`int` **`getRepetitionMode`**`()`

Retorna o modo de repetição dessa animação na forma de um número.

`Component` **`getSource`**`()`

Retorna o formulário fonte que é o formulário a partir do qual a animação está iniciando.

`void` **`init`**`(Component source, Component destination)`

Executado para definir os formulários fonte e de destino.

`void` **`initTransition`**`()`

Callback que é executado antes das transições começarem. O formulário fonte pode ser `null` para o primeiro formulário do aplicativo.

`boolean` **`isRunning`**`()`

Obtém o modo rodando de uma animação.

`void` **`jumpTo`**`(int position)`

Força uma animação a saltar para a posição indicada pelo parâmetro.

`abstract void` **`paint`**`(Graphics g)`

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

`void` **`setAnimationMode`**`(int mode)`

Define o modo de animação para essa animação.

`void` **`setDelay`**`(int n)`

Determina o atraso após cada imagem dessa animação ao rodar.

`void` **`setRepetitionMode`**`(int n)`

Determina quão freqüentemente a animação é repetida, uma vez que for iniciada.

`void` **`start`**`()`

Inicia uma animação.

`void` **`stop`**`()`

Pára uma animação.

25.7.4 Detalhe dos construtores

Transition

```
public Transition()
```

25.7.5 Detalhe dos métodos

init

```
public final void init(Component source,
                       Component destination)
```

Executado para definir os formulários fonte e de destino. Esse método não pode ser executado pelos desenvolvedores.

Parâmetros:

`source` – o formulário fonte a partir do qual a transição se origina

`destination` - o formulário de destino ao qual a transição conduzirá

initTransition

```
public void initTransition()
```

Callback que é executado antes das transições começarem. O formulário fonte pode ser `null` para o primeiro formulário do aplicativo.

getDestination

```
public final Component getDestination()
```

Retorna o formulário de destino que deve ser definido uma vez que a animação estiver completa.

Retorna:

o destino

getSource

```
public final Component getSource()
```

Retorna o formulário fonte que é o formulário a partir do qual a animação está iniciando. Pode ser `null` para o primeiro formulário no aplicativo.

Retorna:

a fonte

cleanup

```
public void cleanup()
```

Operação opcional para limpar o lixo deixado por uma transição em andamento.

copy

```
public abstract Transition copy()
```

Copia uma transição.

Retorna:

a transição copiada

animate

```
public abstract boolean animate()
```

Descrição copiada da interface: **`Animation`**

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*. É chamado uma vez para cada quadro.

Especificado por:

`animate` na interface `Animation`

Retorna:

true se uma repintura for desejada ou *false* se a repintura não for necessária

paint

```
public abstract void paint(Graphics g)
```

Descrição copiada da interface: **`Animation`**

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

Especificado por:

`paint` na interface `Animation`

start

`public void start()`

Inicia uma animação. Considere que uma animação fica em modo parado automaticamente, portanto deve ser iniciada explicitamente uma vez que é criada. No caso da animação já estar rodando, uma chamada para `start()` faz com que ela reinicie.

Especificado por:

`start` na interface `Animated`

stop

`public void stop()`

Pára uma animação. No caso da animação já ter parado, uma chamada para `stop()` não tem efeito.

Especificado por:

`stop` na interface `Animated`

isRunning

`public boolean isRunning()`

Obtém o modo rodando de uma animação. Retorna *true* se a animação já estiver rodando; caso contrário *false*.

Especificado por:

`isRunning` na interface `Animated`

Retorna:

true se a animação já estiver rodando; caso contrário *false*.

jumpTo

`public void jumpTo(int position)`

Força uma animação a saltar para a posição indicada pelo parâmetro. Uma animação pode ser considerada como uma fila de imagens, o parâmetro desse método provê o índice dentro desse fila. Se a animação for parada, uma chamada para esse método faz com que a animação mostre a imagem na posição indicada, mas não que inicie. Se a animação estiver rodando, uma chamada para esse método faz com que a animação salte para a imagem na posição indicada e continue a rodar imediatamente.

Especificado por:

`jumpTo` na interface `Animated`

Parâmetros:

`position` - o índice na fila de imagens fazendo a animação para o qual a animação é forçada a saltar

getPosition

`public int getPosition()`

Obtém a posição atual que a animação ocupa no momento da chamada do método. Uma animação pode ser considerada como uma fila de imagens, esse método provê o índice dentro desse fila.

Especificado por:

`getPosition` na interface `Animated`

Retorna:

a posição atual da imagem dentro da fila de imagens fazendo a animação

setRepetitionMode

`public void setRepetitionMode(int n)`

Determina quão freqüentemente a animação é repetida, uma vez que for iniciada.

Especificado por:

`setRepetitionMode` na interface `Animated`

Parâmetros:

`n` - número de repetições a ser determinado. Pode ser um número maior que zero ou `LOOP` alternativamente. `LOOP` significa que a animação roda em um *loop* infinito até que o método `stop()` seja chamado.

Relaciona-se com:

`getRepetitionMode()`

getRepetitionMode

`public int getRepetitionMode()`

Retorna o modo de repetição dessa animação na forma de um número. O número lê quantas repetições foram determinadas para essa animação.

Especificado por:

`getRepetitionMode` na interface `Animated`

Retorna:

número de repetições determinado para essa animação ou `LOOP`, o que significa que a animação roda em um *loop* infinito até que o método `stop()` seja chamado.

Relaciona-se com:

`setRepetitionMode(int)`

setDelay

`public void setDelay(int n)`

Determina o atraso após cada imagem dessa animação ao rodar.

Especificado por:

`setDelay` na interface `Animated`

Parâmetros:

`n` - atraso em milissegundos

Relaciona-se com:

`getDelay()`

getDelay

`public int getDelay()`

Obtém o atraso após cada imagem dessa animação ao rodar.

Especificado por:

`getDelay` na interface `Animated`

Retorna:

o atraso em milissegundos

Relaciona-se com:

`setDelay(int)`

setAnimationMode

```
public void setAnimationMode(int mode)
```

Define o modo de animação para essa animação. O modo de animação determina como a animação está rodando: sempre para frente, ou para frente e para trás alternando.

Especificado por:

`setAnimationMode` na interface `Animated`

Parâmetros:

`mode` - o modo da animação. Valores possíveis são: `REPEATING` and `ALTERNATING`.

Relaciona-se com:

`getAnimationMode()`

getAnimationMode

```
public int getAnimationMode()
```

Obtém o modo de animação para essa animação. O modo de animação determina como a animação está rodando: sempre para frente, ou para frente e para trás alternando.

Especificado por:

`getAnimationMode` na interface `Animated`

Retorna:

O modo da animação. Valores possíveis são: `REPEATING` and `ALTERNATING`.

Relaciona-se com:

`setAnimationMode(int)`

26 Pacote com.sun.dtv.lwuit.events

26.1 Descrição do pacote

Listeners de eventos de padrões observáveis no espírito da arquitetura despachadora de eventos do AWT1.1. Todos os eventos são despachados na EDT.

NOTA Este pacote está especificado desde o Java DTV 1.0.

26.2 Índice de interfaces

ActionListener

Interface de callback do evento executada quando uma ação do componente ocorre.

DataChangedListener

Interface de callback do evento executada quando um `ListModel` altera seu estado, indicando assim a visualização que deve atualizar.

FocusListener

Observa eventos de alteração do foco de um formulário determinado e executa os callbacks para nos permitir designar uma funcionalidade baseada no componente enfocado atual.

SelectionListener

Executado para indicar uma alteração de seleção no modelo de lista.

StyleListener

Executado para indicar uma alteração em uma propriedade `Style`.

26.3 Índice de classes

ActionEvent

Objeto do evento fornecido quando um callback de `ActionListener` é executado.

26.4 Classe ActionEvent

26.4.1 Descrição da classe

`com.sun.dtv.lwuit.events`

```
java.lang.Object
  └com.sun.dtv.lwuit.events.ActionEvent
```

```
public class ActionEvent
extends Object
```

Objeto do evento fornecido quando um callback de `ActionListener` é executado.

26.4.2 Índice de construtores

`ActionEvent``(Object source)`

Cria uma nova instância de `ActionEvent`.

`ActionEvent``(Object source, int keyEvent)`

Cria uma nova instância de `ActionEvent`.

26.4.3 Índice de métodos

`void` **`consume`**`()`

Consuma o evento indicando que foi tratado, evitando assim que outros *listeners* de ação tratem/recebam o evento.

`Command` **`getCommand`**`()`

Se esse evento foi enviado com resultado de uma ação de comando, esse método retorna aquele comando.

`int` **`getKeyEvent`**`()`

Se esse evento foi acionado por um botão, esse método retornará o keycode apropriado.

`Object` **`getSource`**`()`

O elemento que acionou o evento da ação, útil para separar o código de tratamento de evento.

`boolean` **`isConsumed`**`()`

Retorna *true* se o evento foi consumado, indicando assim que isto foi tratado.

26.4.4 Detalhe dos construtores

ActionEvent

```
public ActionEvent(Object source)
```

Cria uma nova instância de `ActionEvent`.

Parâmetros:

`source` - elemento do evento de ação.

ActionEvent

```
public ActionEvent(Object source,
                   int keyEvent)
```

Cria uma nova instância de `ActionEvent`.

Parâmetros:

`source` - elemento do evento de ação

`keyEvent` – o botão que acionou o evento

26.4.5 Detalhe dos métodos

getSource

```
public Object getSource()
```

O elemento que acionou o evento da ação, útil para separar o código de tratamento de evento.

Retorna:

o elemento que acionou o evento de ação

getKeyEvent

```
public int getKeyEvent()
```

Se esse evento foi acionado por um botão, esse método retornará o keycode apropriado.

Retorna:

o botão que acionou o evento.

getCommand

```
public Command getCommand()
```

Se esse evento foi enviado com resultado de uma ação de comando, esse método retorna aquele comando.

Retorna:

a ação do comando que acionou o evento de ação.

consume

`public void consume()`

Consuma o evento indicando que foi tratado, evitando assim que outros *listeners* de ação tratem/recebam o evento.

isConsumed

`public boolean isConsumed()`

Retorna *true* se o evento foi consumado, indicando assim que isto foi tratado. Isto evita que outros *listeners* de ação tratem/recebam o evento.

Retorna:

True se o evento for consumado, caso contrário, *false*

26.5 Interface ActionListener

26.5.1 Descrição da interface

`com.sun.dtv.lwuit.events`

Todas as classes implementadoras conhecidas

Command

`public interface ActionListener`

Interface de callback do evento executada quando uma ação do componente ocorre.

26.5.2 Índice de métodos

`void` **`actionPerformed`**`(ActionEvent evt)`

Executado quando uma ação ocorre em um componente.

26.5.3 Detalhe dos métodos

actionPerformed

`void actionPerformed(ActionEvent evt)`

Executado quando uma ação ocorre em um componente.

Parâmetros:

`evt` - objeto do evento descrevendo a fonte da ação, bem como seu acionador

26.6 Interface DataChangedListener

26.6.1 Descrição da interface

`com.sun.dtv.lwuit.events`

`public interface DataChangedListener`

Interface de callback do evento executada quando um `ListModel` altera seu estado, indicando assim a visualização que deve atualizar.

26.6.2 Índice de campos

`int` **`ADDED`**

Valor do tipo para dados adicionados na `ListModel`.

`int` **`CHANGED`**

Valor do tipo para dados alterados na `ListModel`.

`int` **`REMOVED`**

Valor do tipo para dados excluídos na `ListModel`.

26.6.3 Índice de métodos

`void` **`dataChanged`**`(int type, int index)`

Executado quando havia uma alteração no modelo subjacente.

26.6.4 Detalhe dos campos

REMOVED

```
public static final int REMOVED = 0
```

Valor do tipo para dados excluídos na `ListModel`.

ADDED

```
public static final int ADDED = 1
```

Valor do tipo para dados adicionados na `ListModel`.

CHANGED

```
public static final int CHANGED = 2
```

Valor do tipo para dados alterados na `ListModel`.

26.6.5 Detalhe dos métodos

dataChanged

```
void dataChanged(int type,
                 int index)
```

Executado quando havia uma alteração no modelo subjacente.

Parâmetros:

`type` – a alteração de dados do tipo; `REMOVED, ADDED` ou `CHANGED`

`index` – índice do item em um modelo de lista

26.7 Interface FocusLstener

26.7.1 Descrição da interface

`com.sun.dtv.lwuit.events`

Todas as classes implementadoras conhecidas

`DefaultLookAndFeel`

```
public interface FocusListener
```

Observa eventos de alteração do foco de um formulário determinado e executa os callbacks para nos permitir designar uma funcionalidade baseada no componente enfocado atual.

26.7.2 Índice de métodos

`void` **`focusGained`**`(Component cmp)`

Executado quando o componente ganha foco.

`void` **`focusLost`**`(Component cmp)`

Executado quando o componente perde foco.

26.7.3 Detalhe dos métodos

focusGained

```
void focusGained(Component cmp)
```

Executado quando o componente ganha foco.

Parâmetros:

`cmp` – o componente que ganha foco

focusLost

```
void focusLost(Component cmp)
```

Executado quando o componente perde foco.

Parâmetros:

`cmp` – o componente que perde foco

26.8 Interface SelectionListener

26.8.1 Descrição da interface

`com.sun.dtv.lwuit.events`

```
public interface SelectionListener
```

Executado para indicar uma alteração de seleção no modelo de lista.

26.8.2 Índice de métodos

`void` **`selectionChanged`**`(int oldSelected, int newSelected)`

Indica que a seleção mudou no modelo de lista subjacente.

26.8.3 Detalhe dos métodos

selectionChanged

```
void selectionChanged(int oldSelected,
                      int newSelected)
```

Indica que a seleção mudou no modelo de lista subjacente.

Parâmetros:

`oldSelected` – índice antigo selecionado no modelo de lista

`newSelected` – novo índice selecionado no modelo de lista

26.9 Interface StyleListener

26.9.1 Descrição da interface

`com.sun.dtv.lwuit.events`

Todas as classes implementadoras conhecidas

`AWTComponent, Button, Calendar, CheckBox, ComboBox, Component, Container, DefaultListCellRenderer, Dialog, DTVContainer, Form, Label, List, MediaComponent, RadioButton, TabbedPane, TextArea, TextField`

```
public interface StyleListener
```

Executado para indicar uma alteração em uma propriedade estilo.

26.9.2 Índice de métodos

`void` **`styleChanged`**`(String propertyName, Style source)`

Invocado para indicar uma mudança em um `propertyName` de um estilo.

26.9.3 Detalhe dos métodos

styleChanged

```
void styleChanged(String propertyName,
                  Style source)
```

Invocado para indicar uma mudança em um `propertyName` de um estilo.

Parâmetros:

`propertyName` - o nome da propriedade que foi mudada

`source` - o objeto estilo mudado

27 Pacote com.sun.dtv.lwuit.geom

27.1 Descrição do pacote

Contém classes relacionadas aos locais geométricos e cálculos tais como retângulo e tamanho.

NOTA Este pacote está especificado desde o Java DTV 1.0.

27.2 Índice de classes

Dimension

Classe de utilidade que contém uma largura e altura que representa uma dimensão de um componente ou elemento.

Point

Um ponto que representa um local em espaço de coordenada (x, y), especificado com precisão.

Rectangle

Representa uma posição (x,y) de retângulo e dimensão (largura, altura). Isso é útil para medir coordenadas dentro do aplicativo.

27.3 Classe Dimension

27.3.1 Descrição da classe

com.sun.dtv.lwuit.geom

```
java.lang.Object
  └com.sun.dtv.lwuit.geom.Dimension
```

```
public class Dimension
extends Object
```

Classe de utilidade que contém uma largura e altura que representa uma dimensão de um componente ou elemento.

27.3.2 Índice de construtores

Dimension()

Cria uma nova instância de `Dimension`.

Dimension(Dimension d)

Cria uma nova instância de `Dimension` com uma dimensão predefinida.

Dimension(int width, int height)

Cria uma nova instância de `Dimension` com largura e altura.

Dimension(Dimension d)

Cria uma nova instância de `Dimension` com parâmetros retirados de uma instância do `java.awt.Dimension`.

27.3.3 Índice de métodos

int **getHeight**()

Retorna a altura da dimensão.

int **getWidth**()

Retorna a largura da dimensão.

`void` **`setHeight`**`(int height)`

Define a altura da dimensão.

`void` **`setWidth`**`(int width)`

Define a largura da dimensão.

`String` **`toString`**`()`

Retorna o formulário imprimível dessa dimensão.

27.3.4 Detalhe dos construtores

Dimension

`public Dimension()`

Cria uma nova instância de `Dimension`.

Dimension

`public Dimension(Dimension d)`

Cria uma nova instância de `Dimension` com uma dimensão predefinida.

Parâmetros:

`d` – `Dimension` para copiar.

Dimension

`public Dimension(Dimension d)`

Cria uma nova instância de `Dimension` com parâmetros obtidos de uma instância do `java.awt.Dimension`. Isso é um criador de conveniência para aprimoramento da integração AWT.

Parâmetros:

`d` – instância `java.awt.Dimension` da qual obter dados.

Dimension

```
public Dimension(int width,
                 int height)
```

Cria uma nova instância de `Dimension` com largura e altura.

Parâmetros:

`width` – a largura da dimensão.

`height` – a altura da dimensão.

27.3.5 Detalhe dos métodos

setWidth

`public void setWidth(int width)`

Define a largura da dimensão.

Parâmetros:

`width` – a largura da dimensão.

Relaciona-se com:

`getWidth()`

setHeight

`public void setHeight(int height)`

Define a altura da dimensão.

Parâmetros:

`height` – a altura da dimensão.

Relaciona-se com:

`getHeight()`

getWidth

`public int getWidth()`

Retorna a largura da dimensão.

Retorna:

largura da dimensão.

Relaciona-se com:

`setWidth(int)`

getHeight

`public int getHeight()`

Retorna a altura da dimensão.

Retorna:

altura da dimensão.

Relaciona-se com:

`setHeight(int)`

toString

`public String toString()`

Retorna o formulário imprimível dessa dimensão.

Substituições:

`toString` na classe `Object`

Retorna:

uma representação em *string* dessa dimensão.

27.4 Classe Point

27.4.1 Descrição da classe

`com.sun.dtv.lwuit.geom`

```
java.lang.Object
  └com.sun.dtv.lwuit.geom.Point
```

```
public class Point
extends Object
```

Um ponto que representa um local em espaço de coordenada (x, y), especificado com precisão.

27.4.2 Índice de construtores

`Point``()`

Cria e inicializa um ponto na origem (0, 0) do espaço de coordenada.

`Point``(Point p)`

Cria e inicializa um ponto com a mesma localização que o objeto `Point` especificado.

`Point``(int x, int y)`

Cria e inicializa um ponto no local determinado (x, y) no espaço de coordenada.

27.4.3 Índice de métodos

`Object` **`clone`**`()`

Cria um novo objeto da mesma classe e com os mesmos conteúdos que esse objeto.

`boolean` **`equals`**`(Object obj)`

Determina se dois pontos são iguais ou não.

`Point` **`getLocation`**`()`

Retorna o local desse ponto.

`int` **`getX`**`()`

Retorna a coordenada x desse ponto.

`int` **`getY`**`()`

Retorna a coordenada y desse ponto.

`int` **`hashCode`**`()`

Retorna o *hashcode* desse ponto.

`void` **`move`**`(int x, int y)`

Move esse ponto para o local determinado no plano de coordenada (x, y).

`void` **`setLocation`**`(Point p)`

Define o local do ponto para o local determinado.

`void` **`setLocation`**`(int x, int y)`

Altera o ponto para ter o local determinado.

`String` **`toString`**`()`

Retorna uma representação da *string* desse ponto e seu local no espaço de coordenada (x, y)

`void` **`translate`**`(int dx, int dy)`

Transforma esse ponto, no local (x, y), por dx junto ao eixo x e dy junto ao eixo y para que represente o ponto (x + dx, y + dy).

27.4.4 Detalhe dos construtores

Point

```
public Point()
```

Cria e inicializa um ponto na origem (0, 0) do espaço de coordenada.

Point

```
public Point(int x,
             int y)
```

Cria e inicializa um ponto no local determinado (x, y) no espaço de coordenada.

Parâmetros:

`x` – a coordenada x

`y` – a coordenada y

Point

```
public Point(Point p)
```

Cria e inicializa um ponto com a mesma localização que o objeto `Point` especificado.

Parâmetros:

`p` – o objeto `Point` determinado

27.4.5 Detalhe dos métodos

getX

```
public int getX()
```

Retorna a coordenada x desse ponto.

Retorna:

a coordenada x desse ponto.

getY

```
public int getY()
```

Retorna a coordenada y desse ponto.

Retorna:

a coordenada y desse ponto.

clone

```
public Object clone()
```

Cria um novo objeto da mesma classe e com os mesmos conteúdos que esse objeto.

Substituições:

`clone` na classe `Object`

Retorna:

o novo Point

equals

```
public boolean equals(Object obj)
```

Determina se dois pontos são iguais ou não.

Substituições:

`equals` na classe `Object`

Parâmetros:

`obj` - o ponto a ser comparado

Retorna:

true se os pontos forem iguais, caso contrário, *false*

getLocation

```
public Point getLocation()
```

Retorna o local desse ponto.

Retorna:

o local

Relaciona-se com:

`setLocation()`

hashCode

```
public int hashCode()
```

Retorna o *hashcode* desse ponto.

Substituições:

`hashCode` na classe `Object`

Retorna:

o *hashcode*

move

```
public void move(int x,
                 int y)
```

Move esse ponto para o local determinado no plano de coordenada (x, y).

Parâmetros:

`x` – a coordenada x do local determinado

`y` - a coordenada y do local determinado

setLocation

```
public void setLocation(int x,
                        int y)
```

Altera o ponto para ter o local determinado.

Parâmetros:

`x` – a coordenada x do local determinado

`y` - a coordenada y do local determinado

Relaciona-se com:

`getLocation()`

setLocation

```
public void setLocation(Point p)
```

Define o local do ponto para o local determinado.

Parâmetros:

`p` – o ponto do local determinado

Relaciona-se com:

`getLocation()`

toString

```
public String toString()
```

Retorna uma representação da *string* desse ponto e seu local no espaço de coordenada (x, y)

Substituições:

`toString` na classe `Object`

Retorna:

a representação da *string*

translate

```
public void translate(int dx,
                      int dy)
```

Transforma esse ponto, no local (x, y), por dx junto ao eixo x e dy junto ao eixo y para que represente o ponto (x + dx, y + dy).

Parâmetros:

`dx` – valor da conversão na direção x

`dy` – valor da conversão na direção y

27.5 Classe Rectangle

27.5.1 Descrição da classe

`com.sun.dtv.lwuit.geom`

`java.lang.Object`

└com.sun.dtv.lwuit.geom.Rectangle

```
public class Rectangle
extends Object
```

Representa uma posição (x,y) de retângulo e dimensão (largura, altura). Isso é útil para medir coordenadas dentro do aplicativo.

27.5.2 Índice de construtores

`Rectangle``()`

Cria uma nova instância do retângulo.

`Rectangle``(Point p, int w, int h)`

Cria uma nova instância do retângulo em uma posição determinada pelo parâmetro `Point`, e com largura e altura predefinidas.

`Rectangle``(Rectangle rect)`

Um criador de cópia.

`Rectangle``(int x, int y, Dimension size)`

Cria uma nova instância de retângulo na posição (x, y) com dimensão predefinida.

`Rectangle``(int x, int y, int w, int h)`

Cria uma nova instância de retângulo na posição (x, y) com largura e altura predefinidas.

`Rectangle``(Rectangle r)`

Cria uma nova instância de retângulo com parâmetros retirados de uma instância do `java.awt.Rectangle`.

27.5.3 Índice de métodos

`boolean` **`contains`**`(Rectangle rect)`

Verifica se este retângulo contém inteiramente o retângulo determinado no parâmetro de entrada.

`boolean` **`contains`**`(int rX, int rY)`

Verifica se esse retângulo contém o ponto no local determinado (rX, rY).

`boolean` **`contains`**`(int rX, int rY, int rWidth, int rHeight)`

Verifica se este retângulo contém inteiramente o retângulo no local especificado (rX, rY) com as dimensões especificadas (`rWidth, rHeight`).

`Dimension` **`getSize`**`()`

Retorna a dimensão do retângulo.

`int` **`getX`**`()`

Retorna a coordenada x do retângulo.

`int` **`getY`**`()`

Retorna a coordenada y do retângulo.

`boolean` **`intersects`**`(Rectangle rect)`

Determina se esse retângulo e o local do retângulo determinado (x, y) com dimensões especificadas (largura, altura) se cruzam.

`boolean` **`intersects`**`(int x, int y, int width, int height)`

Determina se esse retângulo e o local do retângulo determinado (x, y) com dimensões especificadas (largura, altura) se cruzam.

```
static boolean intersects(int tx, int ty, int tw, int th, int x, int y, int width, int height)
```

O método *helper* permite determinar se dois conjuntos de coordenadas se cruzam.

```
void setX(int x)
```

Define a posição x do retângulo.

```
void setY(int y)
```

Define a posição y do retângulo.

```
String toString()
```

27.5.4 Detalhe dos construtores

Rectangle

```
public Rectangle()
```

Cria uma nova instância de retângulo.

Rectangle

```
public Rectangle(int x,
                 int y,
                 Dimension size)
```

Cria uma nova instância de retângulo na posição (x, y) com dimensão predefinida.

Parâmetros:

`x` – a coordenada x do retângulo.

`y` - a coordenada y do retângulo.

`size` - a dimensão do retângulo.

Rectangle

```
public Rectangle(int x,
                 int y,
                 int w,
                 int h)
```

Cria uma nova instância de retângulo na posição (x, y) com largura e altura predefinidas.

Parâmetros:

`x` – a coordenada x do retângulo.

`y` - a coordenada y do retângulo.

`w` - a largura do retângulo.

`h` - a altura do retângulo.

Rectangle

```
public Rectangle(Point p,
                 int w,
                 int h)
```

Cria uma nova instância de retângulo em uma posição determinada pelo parâmetro `Point`, e com largura e altura predefinidas.

Parâmetros:

`p` - o ponto especificando o canto superior esquerdo do retângulo

`w` - a largura do retângulo.

`h` - a altura do retângulo.

Rectangle

```
public Rectangle(Rectangle rect)
```

Um criador de cópia.

Parâmetros:

`rect` – o retângulo para copiar.

Rectangle

```
public Rectangle(Rectangle r)
```

Cria uma nova instância de retângulo com parâmetros obtidos de uma instância do `java.awt.Dimension`. Este é um criador de conveniência para aprimoramento da integração AWT.

Parâmetros:

`r` – instância do `java.awt.Rectangle` de onde obter dados.

27.5.5 Detalhe dos métodos

getSize

```
public Dimension getSize()
```

Retorna a dimensão do retângulo.

Retorna:

o tamanho do retângulo.

getX

```
public int getX()
```

Retorna a coordenada x do retângulo.

Retorna:

a coordenada x do retângulo.

Relaciona-se com:

`setX(int)`

getY

```
public int getY()
```

Retorna a coordenada y do retângulo.

Retorna:

a coordenada y do retângulo.

Relaciona-se com:

`setY(int)`

toString

```
public String toString()
```

Substituições:

`toString` na classe `Object`

setX

```
public void setX(int x)
```

Define a posição x do retângulo.

Parâmetros:

`x` – a coordenada x do retângulo.

Relaciona-se com:

`getX()`

setY

```
public void setY(int y)
```

Define a posição y do retângulo.

Parâmetros:

`y` - a coordenada y do retângulo.

Relaciona-se com:

`getY()`

contains

```
public boolean contains(Rectangle rect)
```

Verifica se este retângulo contém inteiramente o retângulo determinado.

Parâmetros:

`rect` – o retângulo determinado.

Retorna:

true se o retângulo estiver contido inteiramente dentro desse retângulo, caso contrário, *false*.

contains

```
public boolean contains(int rX,
                        int rY,
```

```
                        int rWidth,
                        int rHeight)
```

Verifica se este retângulo contém inteiramente o retângulo no local especificado (rX, rY) com as dimensões especificadas (`rWidth, rHeight`).

Parâmetros:

`rX` – a coordenada x determinada.

`rY` – a coordenada y determinada.

`rWidth` - a largura do retângulo.

`rHeight` - a altura do retângulo.

Retorna:

true se o retângulo determinado por (`rX, rWidth, rHeight`) estiver inteiramente dentro desse retângulo, caso contrário, *false*.

contains

```
public boolean contains(int rX,
                        int rY)
```

Verifica se esse retângulo contém o ponto no local determinado (rX, rY).

Parâmetros:

`rX` – a coordenada x determinada.

`rY` – a coordenada y determinada.

Retorna:

true se o ponto (rX, rY) estiver dentro desse retângulo, caso contrário, *false*.

intersects

```
public boolean intersects(int x,
                          int y,
                          int width,
                          int height)
```

Determina se esse retângulo e o local do retângulo determinado (x, y) com dimensões especificadas (largura, altura) se cruzam. Dois retângulos se cruzam se sua interseção não for vazia.

Parâmetros:

`X` – a coordenada x determinada.

`Y` – a coordenada y determinada.

`width` - a largura do retângulo.

`height` - a altura do retângulo.

Retorna:

true se o retângulo determinado e este retângulo se cruzarem, caso contrário, *false*.

intersects

```
public boolean intersects(Rectangle rect)
```

Determina se esse retângulo e o local do retângulo determinado (x, y) com dimensões especificadas (largura, altura) se cruzam. Dois retângulos se cruzam se sua interseção não for vazia.

Parâmetros:

`rect` – o retângulo para verificar a interseção com

Retorna:

true se o retângulo determinado e este retângulo se cruzarem, caso contrário, *false*.

intersects

```
public static boolean intersects(int tx,
                                 int ty,
                                 int tw,
                                 int th,
                                 int x,
                                 int y,
                                 int width,
                                 int height)
```

O método *helper* permite determinar se dois conjuntos de coordenadas se cruzam. Isso exclui a necessidade de criar um objeto do retângulo para um cálculo rápido.

Parâmetros:

`tx` – a coordenada x do canto superior esquerdo do retângulo 1

`ty` – a coordenada y do canto superior esquerdo do retângulo 1

`tw` – largura do retângulo 1

`th` – altura do retângulo 1

`x` – a coordenada x do canto superior esquerdo do retângulo 2

`y` – a coordenada y do canto superior esquerdo do retângulo 2

`width` - largura do retângulo 2

`height` - altura do retângulo 2

Retorna:

true se os retângulos se cruzam

28 Pacote com.sun.dtv.lwuit.layouts

28.1 Descrição do pacote

Gerenciadores de layout permitem que um `Container` organize seus componentes por um conjunto de regras que seriam adaptadas para tamanhos específicos de fonte/tela. Um gerenciador de layout é um algoritmo de organização encapsulado por uma interface. A implementação da interface insere componentes absolutamente com base nas "dicas" recebidas.

NOTA Este pacote está especificado desde o Java DTV 1.0.

28.2 Índice de classes

BorderLayout

Um layout da borda esquematiza um receptáculo, dispondo e redimencionando seus componentes para se ajustar em cinco regiões: norte, sul, leste, oeste, e centro.

BoxLayout

O gerenciador de layout que insere elementos em uma linha ou coluna de acordo com a orientação da caixa.

CoordinateLayout

Permite esquematizar componentes com base em posições/tamanhos absolutos que são adaptados com base no espaço disponível para o layout.

FlowLayout

Faz o *stream* de elementos em uma linha para que eles possam derramar quando chegarem ao final.

GridLayout

Os componentes são dispostos em um grid de lados iguais com base no espaço disponível.

GroupLayout

`GroupLayout` é um `LayoutManager` que agrupa hierarquicamente os componentes para obter layouts comuns e não tão comuns.

GroupLayout.Group

O grupo oferece semelhança entre dois tipos de operações suportadas pelo `GroupLayout`: esquematizando componentes um depois do outro (`SequentialGroup`) ou um sobre o outro (`ParallelGroup`).

GroupLayout.ParallelGroup

Um `Group` que esquematiza seus elementos um sobre o outro.

GroupLayout.SequentialGroup

Um `Group` que esquematiza seus elementos sequencialmente, um depois do outro.

GroupLayout.Spring

`Spring` consiste em uma variação: min, pref e max um valor em algum lugar no meio daquilo e um local.

Layout

Classe abstrata que pode ser usada para organizar os componentes em um container usando um algoritmo pré-definidos.

LayoutStyle

`LayoutStyle` é usado para determinar quanto espaço deve inserir entre os componentes durante o layout.

28.3 Classe BorderLayout

28.3.1 Descrição da classe

`com.sun.dtv.lwuit.layouts`

```
java.lang.Object
  └com.sun.dtv.lwuit.layouts.Layout
      └com.sun.dtv.lwuit.layouts.BorderLayout
```

```
public class BorderLayout
extends Layout
```

Um layout da borda esquematiza um receptáculo, dispondo e redimencionando seus componentes para se ajustar em cinco regiões: norte, sul, leste, oeste, e centro. Cada região pode conter não mais que um componente e é identificada por uma constante correspondente: `NORTH`, `SOUTH`, `EAST`, `WEST` e `CENTER`. Ao adicionar um componente a um receptáculo com um layout de borda, use uma dessas cinco constantes.

28.3.2 Índice de campos

`static String` **`CENTER`**

A restrição do layout do centro (meio do receptáculo).

`static String` **`EAST`**

A restrição do layout do leste (direita do receptáculo).

`static String` **`NORTH`**

A restrição do layout do norte (parte superior do receptáculo).

`static String` **`SOUTH`**

A restrição do layout do sul (parte inferior do receptáculo).

`static String` **`WEST`**

A restrição do layout do oeste (esquerda do receptáculo).

28.3.3 Índice de construtores

`BorderLayout``()`

Cria uma nova instância do `BorderLayout`.

28.3.4 Índice de métodos

`void` **`addLayoutComponent`**`(Object name, Component comp, Container c)`

Alguns layouts podem opcionalmente monitorar a adição de elementos com metadados que permitem que o usuário dê “dicas” do posicionamento do objeto.

`Object` **`getComponentConstraint`**`(Component comp)`

Retorna a restrição do componente.

`Dimension` **`getPreferredSize`**`(Container parent)`

Retorna o tamanho preferido do receptáculo.

`void` **`layoutContainer`**`(Container target)`

Esquematiza o secundário do receptáculo principal determinando.

`void` **`removeLayoutComponent`**`(Component comp)`

Remove o componente do layout. Essa operação é somente útil se o layout mantiver referências aos componentes dentro dele.

Métodos herdados da classe `com.sun.dtv.lwuit.layouts.`**`Layout`**

`isOverlapSupported`

28.3.5 Detalhe dos campos

NORTH

```
public static final String NORTH = "North"
```

A restrição do layout do norte (parte superior do receptáculo).

SOUTH

```
public static final String SOUTH = "South"
```

A restrição do layout do sul (parte inferior do receptáculo).

CENTER

```
public static final String CENTER = "Center"
```

A restrição do layout do centro (meio do receptáculo).

WEST

```
public static final String WEST = "West"
```

A restrição do layout do oeste (esquerda do receptáculo).

EAST

```
public static final String EAST = "East"
```

A restrição do layout do leste (direita do receptáculo).

28.3.6 Detalhe dos construtores

BorderLayout

```
public BorderLayout()
```

Cria uma nova instância do `BorderLayout`.

28.3.7 Detalhe dos métodos

addLayoutComponent

```
public void addLayoutComponent(Object name,
                               Component comp,
                               Container c)
```

Alguns layouts podem opcionalmente monitorar a adição de elementos com metadados que permitem que o usuário dê “dicas” do posicionamento do objeto.

Substituições:

`addLayoutComponent` na classe `Layout`

Parâmetros:

`comp` – o componente adicionado ao layout

`c` – o receptáculo principal

removeLayoutComponent

```
public void removeLayoutComponent(Component comp)
```

Remove o componente do layout. Essa operação é somente útil se o layout mantiver referências aos componentes dentro dele.

Substituições:

`removeLayoutComponent` na classe `Layout`

Parâmetros:

`comp` – o componente removido do layout

getComponentConstraint

`public Object getComponentConstraint(Component comp)`

Retorna a restrição do componente.

Substituições:

`getComponentConstraint` na classe `Layout`

Parâmetros:

`comp` – o componente

Retorna:

a restrição do componente

layoutContainer

`public void layoutContainer(Container target)`

Esquematiza o secundário do receptáculo principal determinando.

Substituições:

`layoutContainer` na classe `Layout`

getPreferredSize

`public Dimension getPreferredSize(Container parent)`

Retorna o tamanho preferido do receptáculo.

Substituições:

`getPreferredSize` na classe `Layout`

Parâmetros:

`parent` – o receptáculo principal

Retorna:

o tamanho preferido do receptáculo

28.4 Classe BoxLayout

28.4.1 Descrição da classe

`com.sun.dtv.lwuit.layouts`

```
java.lang.Object
  └com.sun.dtv.lwuit.layouts.Layout
      └com.sun.dtv.lwuit.layouts.BoxLayout
```

`public class BoxLayout`

`extends Layout`

O gerenciador de layout que insere elementos em uma linha ou coluna de acordo com a orientação da caixa.

28.4.2 Índice de campos

`static int` **`X_AXIS`**

Layout horizontal onde os componentes são organizados da esquerda para a direita.

`static int` **`Y_AXIS`**

Layout vertical onde os componentes são organizados de cima para baixo.

28.4.3 Índice de construtores

`BoxLayout``(int axis)`

Cria uma nova instância do `BoxLayout`.

28.4.4 Índice de métodos

`Dimension` **`getPreferredSize`**`(Container parent)`

Retorna o tamanho preferido do receptáculo.

`void` **`layoutContainer`**`(Container parent)`

Esquematiza o secundário do receptáculo principal determinando.

Métodos herdados da classe `com.sun.dtv.lwuit.layouts.`**`Layout`**

`addLayoutComponent, getComponentConstraint, isOverlapSupported, removeLayoutComponent`

28.4.5 Detalhe dos campos

X_AXIS

```
public static final int X_AXIS = 1
```

Layout horizontal onde os componentes são organizados da esquerda para a direita.

Y_AXIS

```
public static final int Y_AXIS = 2
```

Layout vertical onde os componentes são organizados de cima para baixo.

28.4.6 Detalhe dos construtores

BoxLayout

```
public BoxLayout(int axis)
```

Cria uma nova instância `do BoxLayout`.

Parâmetros:

`axis` – o eixo ao longo do qual esquematizar os componentes. Pode ser: `BoxLayout.X_AXIS` ou `BoxLayout.Y_AXIS`.

28.4.7 Detalhe dos métodos

layoutContainer

`public void layoutContainer(Container parent)`

Esquematiza o secundário do receptáculo principal determinando.

Substituições:

`layoutContainer` na classe `Layout`

Parâmetros:

`parent` – o receptáculo principal determinado

getPreferredSize

`public Dimension getPreferredSize(Container parent)`

Retorna o tamanho preferido do receptáculo.

Substituições:

`getPreferredSize` na classe `Layout`

Parâmetros:

`parent` – o receptáculo principal

Retorna:

o tamanho preferido do receptáculo

28.5 Classe CoordinateLayout

28.5.1 Descrição da classe

`com.sun.dtv.lwuit.layouts`

```
java.lang.Object
  └com.sun.dtv.lwuit.layouts.Layout
      └com.sun.dtv.lwuit.layouts.CoordinateLayout
```

`public class CoordinateLayout`

`extends Layout`

Permite esquematizar componentes com base em posições/tamanhos absolutos que são adaptados com base no espaço disponível para o layout.

28.5.2 Índice de construtores

`CoordinateLayout``(Dimension d)`

Este criador aceita um objeto `Dimension` para definir a relação de tela do receptáculo.

`CoordinateLayout``(int width, int height)`

Este criador aceita a largura e altura relativas usadas para definir a relação de tela do receptáculo.

28.5.3 Índice de métodos

`Dimension` **`getPreferredSize`**`(Container parent)`

Retorna o tamanho preferido do receptáculo.

`boolean` **`isOverlapSupported`**`()`

Esse método retorna *true* se o layout permitir que os componentes se sobreponham.

`void` **`layoutContainer`**`(Container parent)`

Esquematiza o secundário do receptáculo principal determinando.

Métodos herdados da classe `com.sun.dtv.lwuit.layouts.`**`Layout`**

`addLayoutComponent, getComponentConstraint, removeLayoutComponent`

28.5.4 Detalhe dos construtores

CoordinateLayout

```
public CoordinateLayout(int width,
                        int height)
```

Este criador aceita a largura e altura relativas usadas para definir a relação de tela do receptáculo.

Parâmetros:

`width` – a largura

`height` – a altura

CoordinateLayout

```
public CoordinateLayout(Dimension d)
```

Este criador aceita um objeto `Dimension` para definir a relação de tela do receptáculo.

Parâmetros:

`d` – a relação de tela de um receptáculo como um objeto `Dimension`

28.5.5 Detalhe dos métodos

layoutContainer

```
public void layoutContainer(Container parent)
```

Esquematiza o secundário do receptáculo principal determinando.

Substituições:

`layoutContainer` na classe `Layout`

Parâmetros:

`parent` – o receptáculo principal determinado

getPreferredSize

```
public Dimension getPreferredSize(Container parent)
```

Retorna o tamanho preferido do receptáculo.

Substituições:

`getPreferredSize` na classe `Layout`

Parâmetros:

`parent` – o receptáculo principal

Retorna:

o tamanho preferido do receptáculo

isOverlapSupported

`public boolean isOverlapSupported()`

Esse método retorna *true* se o layout permitir que os componentes se sobreponham.

Substituições:

`isOverlapSupported` na classe `Layout`

Retorna:

true se os componentes puderem cruzar nesse layout

28.6 Classe FlowLayout

28.6.1 Descrição da classe

`com.sun.dtv.lwuit.layouts`

```
java.lang.Object
  └com.sun.dtv.lwuit.layouts.Layout
      └com.sun.dtv.lwuit.layouts.FlowLayout
```

```
public class FlowLayout
extends Layout
```

Faz o *stream* de elementos em uma linha para que eles possam derramar quando chegarem ao final.

28.6.2 Índice de construtores

`FlowLayout``()`

Cria uma nova instância do `FlowLayout` com alinhamento à esquerda.

`FlowLayout``(int orientation)`

Cria uma nova instância do `FlowLayout` com uma orientação determinada `LEFT, RIGHT` ou `CENTER`.

28.6.3 Índice de métodos

`Dimension` **`getPreferredSize`**`(Container parent)`

Retorna o tamanho preferido do receptáculo.

`void` **`layoutContainer`**`(Container parent)`

Esquematiza o secundário do receptáculo principal determinando.

Métodos herdados da classe `com.sun.dtv.lwuit.layouts.`**`Layout`**

`addLayoutComponent, getComponentConstraint, isOverlapSupported, removeLayoutComponent`

28.6.4 Detalhe dos construtores

FlowLayout

`public FlowLayout()`

Cria uma nova instância do `FlowLayout` com alinhamento à esquerda.

FlowLayout

`public FlowLayout(int orientation)`

Cria uma nova instância do `FlowLayout` com uma orientação determinada `LEFT, RIGHT` ou `CENTER`.

Parâmetros:

`orientation` – o valor da orientação

28.6.5 Detalhe dos métodos

layoutContainer

`public void layoutContainer(Container parent)`

Esquematiza o secundário do receptáculo principal determinando.

Substituições:

`layoutContainer` na classe `Layout`

Parâmetros:

`parent` – o receptáculo principal determinado

getPreferredSize

`public Dimension getPreferredSize(Container parent)`

Retorna o tamanho preferido do receptáculo.

Substituições:

`getPreferredSize` na classe `Layout`

Parâmetros:

`parent` – o receptáculo principal

Retorna:

o tamanho preferido do receptáculo

28.7 Classe GridLayout

28.7.1 Descrição da classe

`com.sun.dtv.lwuit.layouts`

```
java.lang.Object
  └com.sun.dtv.lwuit.layouts.Layout
```

└com.sun.dtv.lwuit.layouts.GridLayout

public class GridLayout

extends Layout

Os componentes são dispostos em um grid de lados iguais com base no espaço disponível.

28.7.2 Índice de construtores

GridLayout(int rows, int columns)

Cria uma nova instância do GridLayout com as linhas e colunas determinadas.

28.7.3 Índice de métodos

Dimension **getPreferredSize**(Container parent)

Retorna o tamanho preferido do receptáculo.

void **layoutContainer**(Container parent)

Esquematiza o secundário do receptáculo principal determinando.

Métodos herdados da classe com.sun.dtv.lwuit.layouts.**Layout**

addLayoutComponent, getComponentConstraint, isOverlapSupported, removeLayoutComponent

28.7.4 Detalhe dos construtores

GridLayout

```
public GridLayout(int rows,
                  int columns)
```

Cria uma nova instância do GridLayout com as linhas e colunas determinadas.

Parâmetros:

rows - - as linhas, com o valor zero para representar qualquer número de linhas.

columns - - as colunas, com o valor zero para representar qualquer número de colunas.

28.7.5 Detalhe dos métodos

layoutContainer

```
public void layoutContainer(Container parent)
```

Esquematiza o secundário do receptáculo principal determinando.

Substituições:

layoutContainer na classe Layout

Parâmetros:

parent – o receptáculo principal determinado

getPreferredSize

```
public Dimension getPreferredSize(Container parent)
```

Retorna o tamanho preferido do receptáculo.

Substituições:

`getPreferredSize` na classe `Layout`

Parâmetros:

`parent` – o receptáculo principal

Retorna:

o tamanho preferido do receptáculo

28.8 Classe GroupLayout

28.8.1 Descrição da classe

`com.sun.dtv.lwuit.layouts`

```
java.lang.Object
  └com.sun.dtv.lwuit.layouts.Layout
      └com.sun.dtv.lwuit.layouts.GroupLayout
```

```
public class GroupLayout
extends Layout
```

`GroupLayout` é um `LayoutManager` que agrupa hierarquicamente os componentes para obter layouts comuns e não tão comuns. O agrupamento é feito pelas instâncias da classe `Group`. A Tabela 52 listas os tipos de grupos suportados por `GroupLayout`.

Tabela 52 - Tipos de grupos suportados por `GroupLayout`

Sequencial:	Um grupo sequencial posiciona seus elementos secundários sequencialmente, um depois do outro.
Paralelo:	Um grupo paralelo posiciona seus elementos secundários no mesmo espaço, um em cima do outro. Grupos paralelos também posem alinhar os elementos secundários ao longo de sua linha de base.

Cada grupo pode conter qualquer número dos grupos secundários, componentes ou intervalos. `GroupLayout` trata cada eixo independentemente. Isto é, existe um grupo representando o eixo horizontal e um grupo separado representando o eixo vertical. O grupo horizontal é responsável por definir o x e a largura de seus conteúdos, enquanto o grupo vertical é responsável por definir o y e a altura de seus conteúdos.

O seguinte código cria um layout simples que consiste em dois labels em uma coluna, seguidos de dois campos de texto na coluna seguinte:

```
Form panel = ...;
  GroupLayout layout = new GroupLayout(panel);
  panel.setLayout(layout);
  layout.setAutocreateGaps(true);
  layout.setAutocreateContainerGaps(true);
  GroupLayout.SequentialGroup hGroup = layout.createSequentialGroup();
  hGroup.add(layout.createParallelGroup().add(label1).add(label2)).
         add(layout.createParallelGroup().add(tf1).add(tf2));
```

```
layout.setHorizontalGroup(hGroup);
GroupLayout.SequentialGroup vGroup = layout.createSequentialGroup();
vGroup.add(layout.createParallelGroup(GroupLayout.BASELINE).add(label1).add(tf1)).
       add(layout.createParallelGroup(GroupLayout.BASELINE).add(label2).add(tf2));
layout.setVerticalGroup(vGroup);
```

Esse layout consiste no seguinte:

O eixo horizontal consiste no grupo sequencial que contém dois grupos paralelos. O primeiro grupo paralelo consiste nos labels, e o segundo grupo paralelo consiste nos campos de texto.

O eixo vertical também consiste em um grupo sequencial que contém dois grupos paralelos. Os grupos paralelos alinham seus conteúdos ao longo da linha de base. O primeiro grupo paralelo consiste no primeiro label e campo de texto, e o segundo grupo consiste no segundo label e campo de texto.

Existem algumas coisas para observar neste código:

Você não precisa adicionar explicitamente os componentes ao receptáculo. Isso é feito indiretamente por usar um dos métodos `add`.

Os vários métodos `add` de `Groups` retornam eles mesmos. Isto permite fácil encadeamento de execuções. Por exemplo, `group.add(label1).add(label2)` é equivalente a `group.add(label1);group.add(label2);`.

Não há construtores públicos para `Groups`. Em vez disso, use os métodos de `GroupLayout`.

`GroupLayout` oferece a habilidade de inserir automaticamente o intervalo apropriado entre os componentes. Isso pode ser ativado usando o método `setAutocreateGaps()`. Similarmente, você pode usar o método `setAutocreateContainerGaps()` para inserir intervalos entre os componentes e o receptáculo.

28.8.2 Índice de classes aninhadas

`abstract class` **`GroupLayout.Group`**

O grupo oferece semelhança entre dois tipos de operações suportadas pelo `GroupLayout`: esquematizando componentes um depois do outro (`SequentialGroup`) ou um sobre o outro (`ParallelGroup`).

`class` **`GroupLayout.ParallelGroup`**

Um `Group` que esquematiza seus elementos um sobre o outro.

`class` **`GroupLayout.SequentialGroup`**

Um `Group` que esquematiza seus elementos sequencialmente, um depois do outro.

`abstract class` **`GroupLayout.Spring`**

Spring consiste em uma variação: min, pref e max um valor em algum lugar no meio daquilo e um local.

28.8.3 Índice de campos

`static int` **`BASELINE`**

Possível tipo de alinhamento.

`static int` **`CENTER`**

Possível tipo de alinhamento.

`static int` **`DEFAULT_SIZE`**

Possível valor dos métodos add que obtêm um componente.

`static int` **`EAST`**

Direção leste (direita).

`static int` **`HORIZONTAL`**

Possível argumento ao unir tamanhos dos componentes.

`static int` **`LEADING`**

Possível tipo de alinhamento.

`static int` **`NORTH`**

Direção norte (acima).

`static int` **`PREFERRED_SIZE`**

Possível valor dos métodos add que obtêm um componente.

`static int` **`SOUTH`**

Direção sul (abaixo).

`static int` **`TRAILING`**

Possível tipo de alinhamento.

`static int` **`VERTICAL`**

Possível argumento ao unir tamanhos dos componentes.

`static int` **`WEST`**

Direção oeste (esquerda).

28.8.4 Índice de construtores

`GroupLayout``(Container host)`

Cria um `GroupLayout` para o `JComponent` determinado.

28.8.5 Índice de métodos

`GroupLayout.ParallelGroup` **`createBaselineGroup`**`(boolean resizable, boolean anchorBaselineToTop)`

Cria e retorna um `ParallelGroup` que alinha seus elementos ao longo da linha de base.

`GroupLayout.ParallelGroup` **`createParallelGroup`**`()`

Cria e retorna um `ParallelGroup` com um alinhamento `LEADING`.

`GroupLayout.ParallelGroup` **`createParallelGroup`**`(int alignment)`

Cria e retorna um `ParallelGroup`.

`GroupLayout.ParallelGroup` **`createParallelGroup`**`(int alignment, boolean resizable)`

Cria e retorna um `ParallelGroup`.

`GroupLayout.SequentialGroup` **`createSequentialGroup`**`()`

Cria e retorna um `SequentialGroup`.

`boolean` **`getAutocreateContainerGaps`**`()`

Retorna se intervalos entre o receptáculo e o primeiro/último componente devem ou não ser criados automaticamente.

`boolean` **`getAutocreateGaps`**`()`

Retorna *true* se os intervalos entre os componentes são criados automaticamente.

`boolean` **`getHonorsVisibility`**`()`

Retorna se a visibilidade do componente é considerada ao dimensionar e posicionar componentes.

`GroupLayout.Group` **`getHorizontalGroup`**`()`

Retorna o `Group` que é responsável pelo layout ao longo do eixo horizontal.

`LayoutStyle` **`getLayoutStyle`**`()`

Retorna a instância `LayoutStyle` para uso.

`Dimension` **`getPreferredSize`**`(Container parent)`

Retorna o tamanho preferido para receptáculo determinado.

`GroupLayout.Group` **`getVerticalGroup`**`()`

Retorna o `ParallelGroup` que é responsável pelo layout diante do eixo vertical.

`void` **`layoutContainer`**`(Container parent)`

Esquematiza o receptáculo determinado.

`void` **`linkSize`**`(Component[] components)`

Força o conjunto de componentes a ter o mesmo tamanho.

`void` **`linkSize`**`(Component[] components, int axis)`

Força o conjunto de componentes a ter o mesmo tamanho.

`void` **`removeLayoutComponent`**`(Component component)`

Notificação de que um `Component` foi removido do receptáculo principal.

`void` **`replace`**`(Component existingComponent, Component newComponent)`

Remove um componente existente, substituindo-o por um componente determinado.

`void` **`setAutocreateContainerGaps`**`(boolean autocreatePadding)`

Define se há ou não intervalos entre o receptáculo e o primeiro/último componente deve ser criado automaticamente.

`void` **`setAutocreateGaps`**`(boolean autocreatePadding)`

Define se espaços entre os componentes devem ser criados automaticamente.

`void` **`setHonorsVisibility`**`(boolean honorsVisibility)`

Define se a visibilidade do componente é considerada ao dimensionar e posicionar os componentes.

`void` **`setHonorsVisibility`**`(Component component, Boolean honorsVisibility)`

Define se a visibilidade do componente é considerada para dimensionamento e posicionamento.

`void` **`setHorizontalGroup`**`(GroupLayout.Group group)`

Define o `Group` que é responsável pelo layout ao longo do eixo horizontal.

`void` **`setLayoutStyle`**`(LayoutStyle layoutStyle)`

Define o `LayoutStyle` que esse `GroupLayout` deve usar.

`void` **`setVerticalGroup`**`(GroupLayout.Group group)`

Define o `Group` que é responsável pelo layout ao longo do eixo vertical.

`String` **`toString`**`()`

Retorna uma descrição textual desse `GroupLayout`.

Métodos herdados da classe `com.sun.dtv.lwuit.layouts.`**`Layout`**

`addLayoutComponent, getComponentConstraint, isOverlapSupported`

28.8.6 Detalhe dos campos

NORTH

```
public static final int NORTH = 1
```

Direção norte (acima).

EAST

```
public static final int EAST = 3
```

Direção leste (direita).

SOUTH

```
public static final int SOUTH = 5
```

Direção sul (abaixo).

WEST

```
public static final int WEST = 7
```

Direção oeste (esquerda).

HORIZONTAL

```
public static final int HORIZONTAL = 1
```

Possível argumento ao unir tamanhos dos componentes. Especifica os dois componentes que devem dividir o mesmo tamanho ao longo do eixo horizontal.

Relaciona-se com:

```
linkSize(Component[], int)
```

VERTICAL

```
public static final int VERTICAL = 2
```

Possível argumento ao unir tamanhos dos componentes. Especifica se os dois componentes devem dividir o mesmo tamanho ao longo do eixo vertical.

Relaciona-se com:

```
linkSize(Component[],int)
```

LEADING

```
public static final int LEADING = 1
```

Possível tipo de alinhamento. Indica se os elementos devem ser alinhados à origem. Para o eixo horizontal com uma orientação da esquerda para a direita, isso significa alinhado à esquerda.

Relaciona-se com:

```
createParallelGroup(int)
```

TRAILING

```
public static final int TRAILING = 2
```

Possível tipo de alinhamento. Indica se os elementos devem ser alinhados ao final. Para o eixo horizontal com uma orientação da esquerda para a direita, isso significa alinhado à direita.

Relaciona-se com:

`createParallelGroup(int)`

CENTER

```
public static final int CENTER = 4
```

Possível tipo de alinhamento. Indica se os elementos devem ser centralizados no espaço fornecido.

Relaciona-se com:

`createParallelGroup(int)`

BASELINE

```
public static final int BASELINE = 3
```

Possível tipo de alinhamento. Indica se os elementos devem ser alinhados ao longo de sua linha de base.

Relaciona-se com:

`createParallelGroup(int)`

DEFAULT_SIZE

```
public static final int DEFAULT_SIZE = -1
```

Possível valor dos métodos add que obtêm um componente. Indica se o tamanho do componente deve ser usado.

PREFERRED_SIZE

```
public static final int PREFERRED_SIZE = -2
```

Possível valor dos métodos add que obtêm um componente. Indica se o tamanho preferido deve ser usado.

28.8.7 Detalhe dos construtores

GroupLayout

```
public GroupLayout(Container host)
```

Cria um `GroupLayout` para o `JComponent` determinado.

Parâmetros:

`host` – o receptáculo para layout

Lança:

`IllegalArgumentException` - se o *host* for `null`

28.8.8 Detalhe dos métodos

setHonorsVisibility

```
public void setHonorsVisibility(boolean honorsVisibility)
```

Define se a visibilidade do componente é considerada ao dimensionar e posicionar os componentes. Um valor *true* indica que componentes invisíveis não podem ser tratados como parte do layout. Um valor *false* indica que componentes devem ser posicionados e dimencionados independentemente da visibilidade.

Um valor *false* é útil quando a visibilidade dos componentes for dinamicamente ajustada e você não quiser que os componentes circundantes e o tamanho sejam alterados.

O valor determinado é usado para componentes que não têm uma visibilidade explícita determinada.

O padrão é *true*.

Parâmetros:

`honorsVisibility` - se a visibilidade do componente é considerada ao dimensionar e posicionar componentes.

Relaciona-se com:

`setHonorsVisibility(Component,Boolean), getHonorsVisibility()`

getHonorsVisibility

```
public boolean getHonorsVisibility()
```

Retorna se a visibilidade do componente é considerada ao dimensionar e posicionar componentes.

Retorna:

se a visibilidade do componente é considerada ao dimensionar e posicionar componentes.

Relaciona-se com:

`setHonorsVisibility(boolean), setHonorsVisibility(com.sun.dtv.lwuit.Component, java.lang.Boolean)`

setHonorsVisibility

```
public void setHonorsVisibility(Component component,
                                Boolean honorsVisibility)
```

Define se a visibilidade do componente é considerada para dimensionamento e posicionamento. Um valor `Boolean.TRUE` indica que se `component` não estiver visível, não pode ser tratado como parte do layout. Um valor *false* indica que `component` é posicionado e dimencionado independentemente de sua visibilidade. Um valor `null` indica que o valor determinado por um único método de argumento `setHonorsVisibility` deve ser usado.

Se `component` não for secundário do `Container` que esse `GroupLayout` está gerenciando, deve ser adicionado ao `Container`.

Parâmetros:

`component` – o componente

`honorsVisibility` - se a visibilidade do `component` deve ser considerada para dimensionamento e posicionamento

Lança:

`IllegalArgumentException` - se o componente for `null`

Relaciona-se com:

`setHonorsVisibility(boolean), getHonorsVisibility()`

toString

`public String toString()`

Retorna uma descrição textual desse `GroupLayout`. O retorno do valor é programado somente para finalidade de depuração.

Substituições:

`toString` na classe `Object`

Retorna:

descrição textual desse `GroupLayout`

setAutocreateGaps

`public void setAutocreateGaps(boolean autocreatePadding)`

Define se espaços entre os componentes devem ser criados automaticamente. Por exemplo, se for *true* e você adicionar dois componentes a um `SequentialGroup`, um intervalo entre os dois deve ser automaticamente criado. O padrão é *false*.

Parâmetros:

`autocreatePadding` - criar ou não um intervalo automaticamente entre componentes e o receptáculo.

Relaciona-se com:

`getAutocreateGaps()`

getAutocreateGaps

`public boolean getAutocreateGaps()`

Retorna *true* se os intervalos entre os componentes são criados automaticamente.

Retorna:

true se os intervalos entre os componentes forem criados automaticamente

Relaciona-se com:

`setAutocreateGaps(boolean)`

setAutocreateContainerGaps

`public void setAutocreateContainerGaps(boolean autocreatePadding)`

Define se há ou não intervalos entre o receptáculo e o primeiro/último componente deve ser criado automaticamente. O padrão é *false*.

Parâmetros:

`autocreatePadding` - criar ou não automaticamente os espaços entre o receptáculo e o primeiro/último componente.

Relaciona-se com:

`getAutocreateContainerGaps()`

getAutocreateContainerGaps

`public boolean getAutocreateContainerGaps()`

Retorna se intervalos entre o receptáculo e o primeiro/último componente devem ou não ser criados automaticamente. O padrão é *false*.

Retorna:

se os intervalos entre o receptáculo e a primeiro/último componente devem ser criado automaticamente ou não

Relaciona-se com:

`setAutocreateContainerGaps(boolean)`

setHorizontalGroup

`public void setHorizontalGroup(GroupLayout.Group group)`

Define o `Group` que é responsável pelo layout ao longo do eixo horizontal.

Parâmetros:

`group` - `Group` responsável pelo layout AL longo do eixo horizontal

Lança:

`IllegalArgumentException` - se o grupo for `null`

Relaciona-se com:

`getHorizontalGroup()`

getHorizontalGroup

`public GroupLayout.Group getHorizontalGroup()`

Retorna o `Group` que é responsável pelo layout ao longo do eixo horizontal.

Retorna:

`ParallelGroup` responsável pelo layout ao longo do eixo horizontal.

Relaciona-se com:

`setHorizontalGroup()`

setVerticalGroup

`public void setVerticalGroup(GroupLayout.Group group)`

Define o `Group` que é responsável pelo layout ao longo do eixo vertical.

Parâmetros:

`group` - `Group` responsável pelo layout ao longo do eixo vertical.

Lança:

`IllegalArgumentException` - se o grupo for `null`.

Relaciona-se com:

`getVerticalGroup()`

getVerticalGroup

`public GroupLayout.Group getVerticalGroup()`

Retorna o `ParallelGroup` que é responsável pelo layout diante do eixo vertical.

Retorna:

`ParallelGroup` responsável pelo layout ao longo do eixo vertical.

Relaciona-se com:

`setVerticalGroup()`

createSequentialGroup

`public GroupLayout.SequentialGroup createSequentialGroup()`

Cria e retorna um `SequentialGroup`.

Retorna:

um novo `SequentialGroup`

createParallelGroup

`public GroupLayout.ParallelGroup createParallelGroup()`

Cria e retorna um `ParallelGroup` com um alinhamento `LEADING`. Esse é um método cover para o método mais geral `createParallelGroup(int)`.

Retorna:

um novo `ParallelGroup`

Relaciona-se com:

`createParallelGroup(int)`

createParallelGroup

`public GroupLayout.ParallelGroup createParallelGroup(int alignment)`

Cria e retorna um `ParallelGroup`. O alinhamento especifica como os elementos secundários devem estar posicionados quando for dado mais espaço do que o necessário ao grupo paralelo. Por exemplo, se a um `ParallelGroup` com um alinhamento de `TRAILING` forem dados 100 pixels, e um secundário só precisar de 50 pixels, este deve ser posicionado na posição 50.

Parâmetros:

`alignment` - alinhamento dos elementos do Grupo, `LEADING`, `TRAILING`, `CENTER` ou `BASELINE`.

Retorna:

um novo `ParallelGroup`

Lança:

`IllegalArgumentException` – se alinhamento não for `LEADING, TRAILING, CENTER` ou `BASELINE`

createParallelGroup

```
public GroupLayout.ParallelGroup createParallelGroup(int alignment,
                                                     boolean resizable)
```

Cria e retorna um `ParallelGroup`. O alinhamento especifica como os elementos secundários devem estar posicionados quando for dado mais espaço do que o necessário ao grupo paralelo. Por exemplo, se a um `ParallelGroup` com um alinhamento de `TRAILING` forem dados 100 pixels, e um secundário só precisar de 50 pixels, este deve ser posicionado na posição 50.

Parâmetros:

`alignment` - alinhamento dos elementos do Grupo, `LEADING`, `TRAILING`, `CENTER` ou `BASELINE`.

`resizable` – se o grupo é ou não redimensionável. Se o grupo não for redimensionável o tamanho min/max deve ser o mesmo que o preferido.

Retorna:

um novo `ParallelGroup`

Lança:

`IllegalArgumentException` – se alinhamento não for `LEADING, TRAILING, CENTER` ou `BASELINE`

createBaselineGroup

```
public GroupLayout.ParallelGroup createBaselineGroup(boolean resizable,
                                                     boolean anchorBaselineToTop)
```

Cria e retorna um `ParallelGroup` que alinha seus elementos ao longo da linha de base.

Parâmetros:

`resizable` – se o grupo é redimensionável.

`anchorBaselineToTop` – se a linha de base estiver ancorada no início ou no final do grupo

Retorna:

um `ParallelGroup` que alinha seus elementos ao longo da linha de base.

Relaciona-se com:

`createBaselineGroup(),GroupLayout.ParallelGroup`

linkSize

```
public void linkSize(Component[] components)
```

Força o conjunto de componentes a ter o mesmo tamanho. Pode ser usado várias vezes para forçar qualquer número de componentes a compartilhar o mesmo tamanho.

componentes unidos não são redimensionáveis.

Parâmetros:

`components` – componentes para forçar a ter o mesmo tamanho.

Lança:

`IllegalArgumentException` - se os componentes forem `null` ou contiverem `null`.

linkSize

```
public void linkSize(Component[] components,
                     int axis)
```

Força o conjunto de componentes a ter o mesmo tamanho. Pode ser usado várias vezes para forçar qualquer número de componentes a compartilhar o mesmo tamanho.

componentes unidos não são redimencionáveis.

Parâmetros:

`components` – componentes para forçar a ter o mesmo tamanho.

`axis` – Eixo para vincular o tamanho, de uma `HORIZONTAL, VERTICAL` ou `HORIZONTAL | VERTICAL`

Lança:

`IllegalArgumentException` – se os componentes forem `null` ou contiverem `null`, se o eixo não contiver `HORIZONTAL` ou `VERTICAL`

replace

```
public void replace(Component existingComponent,
                    Component newComponent)
```

Remove um componente existente, substituindo-o por um componente determinado.

Parâmetros:

`existingComponent` – o componente que deve ser removido e substituído por newComponent

`newComponent` – o componente para inserir no lugar de existingComponents

Lança:

`IllegalArgumentException` - se algum dos componentes for `null`, ou se o existingComponent não estiver sendo gerenciado por esse gerenciador de layout

setLayoutStyle

```
public void setLayoutStyle(LayoutStyle layoutStyle)
```

Define o `LayoutStyle` que esse `GroupLayout` deve usar. Um valor `null` pode ser usado para indicar a instância compartilhada do `LayoutStyle` que deve ser usada.

Parâmetros:

`layoutStyle` - o `LayoutStyle` a ser usado

Relaciona-se com:

`getLayoutStyle()`

getLayoutStyle

```
public LayoutStyle getLayoutStyle()
```

Retorna a instância `LayoutStyle` para uso.

Retorna:

a instância `LayoutStyle` a ser usada

Relaciona-se com:

`setLayoutStyle(com.sun.dtv.lwuit.layouts.LayoutStyle)`

removeLayoutComponent

`public void removeLayoutComponent(Component component)`

Notificação de que um `Component` foi removido do receptáculo principal. Você não deve executar esse método diretamente. Execute `removeComponent` no `Container` principal.

Substituições:

`removeLayoutComponent` na classe `Layout`

Parâmetros:

`component` – o componente a ser removido

Relaciona-se com:

`Container.removeComponent()`

getPreferredSize

`public Dimension getPreferredSize(Container parent)`

Retorna o tamanho preferido para receptáculo determinado.

Substituições:

`getPreferredSize` na classe `Layout`

Parâmetros:

`parent` - o receptáculo para o qual retornar o tamanho

Retorna:

o tamanho preferido

Lança:

`IllegalArgumentException` – se o principal não for o mesmo Receptáculo com que foi criado

`IllegalStateException` – se qualquer componente adicionado a esse *layout* não estiver em um grupo horizontal e vertical

Relaciona-se com:

`Component.getPreferredSize()`

layoutContainer

`public void layoutContainer(Container parent)`

Esquematiza o receptáculo determinado.

Substituições:

`layoutContainer` na classe `Layout`

Parâmetros:

`parent` – o receptáculo para ser esquematizado

Lança:

`IllegalStateException` – se qualquer componente adicionado a esse *layout* não estiver em um grupo

horizontal e vertical

28.9 Classe GroupLayout.Group

28.9.1 Descrição da classe

com.sun.dtv.lwuit.layouts

```
java.lang.Object
  └com.sun.dtv.lwuit.layouts.GroupLayout.Spring
      └com.sun.dtv.lwuit.layouts.GroupLayout.Group
```

Subclasses diretas conhecidas

`GroupLayout.ParallelGroup, GroupLayout.SequentialGroup`

Classe externa:

`GroupLayout`

```
abstract public class GroupLayout.Group
extends GroupLayout.Spring
```

O grupo oferece semelhança entre dois tipos de operações suportadas pelo `GroupLayout`: esquematizando componentes um depois do outro (`SequentialGroup`) ou um sobre o outro (`ParallelGroup`). Use `createSequentialGroup` ou `createParallelGroup` para criar uma.

28.10 Classe GroupLayout.ParallelGroup

28.10.1 Descrição da Classe

com.sun.dtv.lwuit.layouts

```
java.lang.Object
  └com.sun.dtv.lwuit.layouts.GroupLayout.Spring
      └com.sun.dtv.lwuit.layouts.GroupLayout.Group
          └com.sun.dtv.lwuit.layouts.GroupLayout.ParallelGroup
```

Classe externa:

`GroupLayout`

```
public class GroupLayout.ParallelGroup
extends GroupLayout.Group
```

Um `Group` que esquematiza seus elementos um sobre o outro. Se um elemento secundário for menor do que o espaço fornecido, ele deve ser alinhado com base no alinhamento do secundário (se determinado) ou no alinhamento do `ParallelGroup`.

Relaciona-se com:

`GroupLayout.createParallelGroup()`

28.10.2 Índice de métodos

`GroupLayout.ParallelGroup` **`add`**`(Component component)`

Adiciona o componente determinado.

`GroupLayout.ParallelGroup` **`add`**`(Component component, int min, int pref, int max)`

Adiciona o componente determinado.

`GroupLayout.ParallelGroup` **`add`**`(GroupLayout.Group group)`

Adiciona o `Group` determinado.

`GroupLayout.ParallelGroup` **`add`**`(int pref)`

Adiciona um intervalo rígido.

`GroupLayout.ParallelGroup` **`add`**`(int alignment, Component component)`

Adiciona o componente determinado.

`GroupLayout.ParallelGroup` **`add`**`(int alignment, Component component, int min, int pref, int max)`

Adiciona o componente determinado.

`GroupLayout.ParallelGroup` **`add`**`(int alignment, GroupLayout.Group group)`

Adiciona o `Group` determinado com um secundário desse grupo.

`GroupLayout.ParallelGroup` **`add`**`(int min, int pref, int max)`

Adiciona um intervalo com o tamanho determinado.

28.10.3 Detalhe dos métodos

add

`public GroupLayout.ParallelGroup add(GroupLayout.Group group)`

Adiciona o `Group` determinado.

Parâmetros:

`group` – o Grupo para adicionar

Retorna:

esse Grupo

add

`public GroupLayout.ParallelGroup add(Component component)`

Adiciona o componente determinado. Se o `min/max` do componente forem diferentes do `pref`, o componente deve ser redimensionável.

Parâmetros:

`component` - o componente para adicionar

Retorna:

esse `ParallelGroup`

add

```
public GroupLayout.ParallelGroup add(Component component,
                                     int min,
                                     int pref,
                                     int max)
```

Adiciona o componente determinado. `Min`, `pref` e `max` podem ser valores absolutos ou `DEFAULT_SIZE` ou `PREFERRED_SIZE`. Por exemplo, o seguinte:

```
   add(component, PREFERRED_SIZE, PREFERRED_SIZE, 1000);
```

Força um valor para `max` de 1000, com o `min` e `pref` igualando aquele do tamanho preferível do `component`.

Parâmetros:

`component` - o componente para adicionar

`min` – o tamanho mínimo

`pref` - o tamanho preferível

`max` – o tamanho máximo

Retorna:

esse `SequentialGroup`

Lança:

`IllegalArgumentException` – se `min`, `pref` ou `max` não for positivo e não for `PREFERRED_SIZE` ou `DEFAULT_SIZE`.

add

```
public GroupLayout.ParallelGroup add(int pref)
```

Adiciona um intervalo rígido.

Parâmetros:

`pref` - o tamanho do intervalo.

Retorna:

esse `ParallelGroup`

Lança:

`IllegalArgumentException` – se o `min` = 0, pref = 0 ou max = 0, ou se o seguinte for verdadeiro: `min` = `pref` = `max`.

add

```
public GroupLayout.ParallelGroup add(int min,
                                     int pref,
                                     int max)
```

Adiciona um intervalo com o tamanho determinado.

Parâmetros:

min - o tamanho mínimo do intervalo

pref - o tamanho preferível do intervalo

max - o tamanho máximo do intervalo

Retorna:

esse ParallelGroup

Lança:

IllegalArgumentException – se o min = 0, pref = 0 ou max = 0, ou se o seguinte for verdadeiro: min = pref = max.

add

```
public GroupLayout.ParallelGroup add(int alignment,
                                     GroupLayout.Group group)
```

Adiciona o Group determinado com um secundário desse grupo.

Parâmetros:

alignment – o alinhamento do grupo.

group – o grupo para adicionar

Retorna:

esse ParallelGroup

Lança:

IllegalArgumentException – se alinhamento não for LEADING, TRAILING ou CENTER

add

```
public GroupLayout.ParallelGroup add(int alignment,
                                     Component component)
```

Adiciona o componente determinado. Se o min/max do componente forem diferentes do pref, o componente deve ser redimensionável.

Parâmetros:

alignment – o alinhamento do componente

component - o componente para adicionar

Retorna:

esse grupo

Lança:

IllegalArgumentException – se alinhamento não for LEADING, TRAILING ou CENTER

add

```
public GroupLayout.ParallelGroup add(int alignment,
                                     Component component,
                                     int min,
                                     int pref,
                                     int max)
```

Adiciona o componente determinado. Min, pref e max podem ser valores absolutos ou DEFAULT_SIZE ou PREFERRED_SIZE. Por exemplo, o seguinte:

```
add(component, PREFERRED_SIZE, PREFERRED_SIZE, 1000);
```

Força um `max` de 1000, com o `min` e `pref` igualando o tamanho preferível do componente.

Parâmetros:

`alignment` – o alinhamento do componente.

`component` - o componente para adicionar

`min` – o tamanho mínimo

`pref` - o tamanho preferível

`max` – o tamanho máximo

Retorna:

esse grupo.

Lança:

`IllegalArgumentException` – se `min`, `pref` ou `max` não for positivo e não for `PREFERRED_SIZE` ou `DEFAULT_SIZE`.

28.11 Classe GroupLayout.SequentialGroup

28.11.1 Descrição da classe

`com.sun.dtv.lwuit.layouts`

```
java.lang.Object
  └com.sun.dtv.lwuit.layouts.GroupLayout.Spring
      └com.sun.dtv.lwuit.layouts.GroupLayout.Group
          └com.sun.dtv.lwuit.layouts.GroupLayout.SequentialGroup
```

Classe externa:

`GroupLayout`

```
public class GroupLayout.SequentialGroup
extends GroupLayout.Group
```

Um `Group` que esquematiza seus elementos sequencialmente, um depois do outro. Essa classe não tem criador público, use o método `createSequentialGroup` para criar um.

Relaciona-se com:

`GroupLayout.createSequentialGroup()`

28.11.2 Índice de métodos

`GroupLayout.SequentialGroup` **`add`**`(boolean useAsBaseline, Component component)`

Adiciona um `Component` a esse `Group`.

`GroupLayout.SequentialGroup` **`add`**`(boolean useAsBaseline, Component component, int min, int pref, int max)`

Adiciona um `Component` a esse `Group` com o tamanho determinado.

`GroupLayout.SequentialGroup` **`add`**`(boolean useAsBaseline, GroupLayout.Group group)`

Adiciona um `Group` a esse `Group`.

`GroupLayout.SequentialGroup` **`add`**`(Component component)`

Adiciona o componente determinado.

`GroupLayout.SequentialGroup` **`add`**`(Component component, int min, int pref, int max)`

Adiciona o componente determinado.

`GroupLayout.SequentialGroup` **`add`**`(GroupLayout.Group group)`

Adiciona um `Group` determinado a esse `SequentialGroup`.

`GroupLayout.SequentialGroup` **`add`**`(int pref)`

Adiciona um intervalo rígido.

`GroupLayout.SequentialGroup` **`add`**`(int min, int pref, int max)`

Adiciona um intervalo com o tamanho determinado.

`GroupLayout.SequentialGroup` **`addContainerGap`**`()`

Adiciona um elemento representando o intervalo preferido entre uma margem do receptáculo e o componente próximo/anterior.

`GroupLayout.SequentialGroup` **`addContainerGap`**`(int pref, int max)`

Adiciona um elemento representando o intervalo preferido entre uma margem do receptáculo e o componente próximo/anterior.

`GroupLayout.SequentialGroup` **`addPreferredGap`**`(Component comp1, Component comp2, int type)`

Adiciona um elemento representando o intervalo preferido entre os dois componentes.

`GroupLayout.SequentialGroup` **`addPreferredGap`**`(Component comp1, Component comp2, int type, boolean canGrow)`

Adiciona um elemento representando o intervalo preferido entre os dois componentes.

`GroupLayout.SequentialGroup` **`addPreferredGap`**`(int type)`

Adiciona um elemento representando o intervalo preferido entre os componentes mais próximos.

`GroupLayout.SequentialGroup` **`addPreferredGap`**`(int type, int pref, int max)`

Adiciona um elemento para o intervalo preferido entre os componentes mais próximos.

28.11.3 Detalhe dos métodos

add

```
public GroupLayout.SequentialGroup add(GroupLayout.Group group)
```

Adiciona um `Group` determinado a esse `SequentialGroup`.

Parâmetros:

`group` – o grupo para adicionar.

Retorna:

esse grupo.

add

```
public GroupLayout.SequentialGroup add(boolean useAsBaseline,
                                       GroupLayout.Group group)
```

Adiciona um `Group` a esse `Group`.

Parâmetros:

`useAsBaseline` – se o `Group` determinado deve ser usado para calcular a linha de base desse `Group`.

`group` – o Grupo para adicionar

Retorna:

esse Grupo

add

```
public GroupLayout.SequentialGroup add(Component component)
```

Adiciona o componente determinado. Se o min/max do componente forem diferentes do pref, o componente deve ser redimensionável.

Parâmetros:

`component` - o componente para adicionar

Retorna:

esse `SequentialGroup`

add

```
public GroupLayout.SequentialGroup add(boolean useAsBaseline,
                                       Component component)
```

Adiciona um `Component` a esse `Group`.

Parâmetros:

`useAsBaseline` – se o componente determinado deve ser usado para calcular a linha de base desse grupo.

`component` - o componente para adicionar

Retorna:

esse grupo.

add

```
public GroupLayout.SequentialGroup add(Component component,
                                       int min,
                                       int pref,
                                       int max)
```

Adiciona o componente determinado. Min, pref e max podem ser valores absolutos ou `DEFAULT_SIZE` ou `PREFERRED_SIZE`. Por exemplo, o seguinte:

```
add(component, PREFERRED_SIZE, PREFERRED_SIZE, 1000);
```

Força um max de 1000, com o min e preferred igualando aquele do tamanho preferido do `component`.

Parâmetros:

`component` - o componente para adicionar

`min` – o tamanho mínimo

`pref` - o tamanho preferido

max – o tamanho máximo

Retorna:

esse SequentialGroup.

Lança:

IllegalArgumentException – se min, pref ou max não for positivo e não for PREFERRED_SIZE ou DEFAULT_SIZE.

add

```
public GroupLayout.SequentialGroup add(boolean useAsBaseline,
                                       Component component,
                                       int min,
                                       int pref,
                                       int max)
```

Adiciona um componente a esse grupo com o tamanho determinado.

Parâmetros:

useAsBaseline – se o componente determinado deve ser usado para calcular a linha de base desse grupo.

component - o componente para adicionar

min – o tamanho mínimo ou DEFAULT_SIZE ou PREFERRED_SIZE

pref - o tamanho preferido ou DEFAULT_SIZE ou PREFERRED_SIZE

max - o tamanho máximo ou DEFAULT_SIZE ou PREFERRED_SIZE

Retorna:

esse grupo

add

```
public GroupLayout.SequentialGroup add(int pref)
```

Adiciona um intervalo rígido.

Parâmetros:

pref - o tamanho do intervalo.

Retorna:

esse SequentialGroup

Lança:

IllegalArgumentException – se pref = 0.

add

```
public GroupLayout.SequentialGroup add(int min,
                                       int pref,
                                       int max)
```

Adiciona um intervalo com o tamanho determinado.

Parâmetros:

min - o tamanho mínimo do intervalo ou PREFERRED_SIZE

pref - o tamanho preferido do intervalo

`max` - o tamanho máximo do intervalo ou `PREFERRED_SIZE`

Retorna:

esse `SequentialGroup`

Lança:

`IllegalArgumentException` – se `min` = 0, `pref` = 0 ou `max` = 0, ou o seguinte for verdadeiro `min = pref = max`.

addPreferredGap

```
public GroupLayout.SequentialGroup addPreferredGap(Component comp1,
                                                   Component comp2,
                                                   int type)
```

Adiciona um elemento representando o intervalo preferido entre os dois componentes.

Parâmetros:

`comp1` - o primeiro componente

`Comp2` - o segundo componente

`type` – o tipo de intervalo, uma das constantes definidas pelo `LayoutStyle`

Retorna:

esse `SequentialGroup`

Lança:

`IllegalArgumentException` – se o tipo não for uma constante `LayoutStyle` válida

Relaciona-se com:

`LayoutStyle`

addPreferredGap

```
public GroupLayout.SequentialGroup addPreferredGap(Component comp1,
                                                   Component comp2,
                                                   int type,
                                                   boolean canGrow)
```

Adiciona um elemento representando o intervalo preferido entre os dois componentes.

Parâmetros:

`comp1` - o primeiro componente

`Comp2` - o segundo componente

`type` – o tipo de intervalo, uma das constantes definidas pelo `LayoutStyle`

`canGrow` – *true* se o espaço puder aumentar se houver mais espaço disponível

Retorna:

esse `SequentialGroup`

Lança:

`IllegalArgumentException` – se o tipo não for uma constante `LayoutStyle` válida

Relaciona-se com:

LayoutStyle

addPreferredGap

```
public GroupLayout.SequentialGroup addPreferredGap(int type)
```

Adiciona um elemento representando o intervalo preferido entre os componentes mais próximos. Ou seja, durante o layout, os componentes vizinhos são encontrados, e o `min`, `pref` e `max` desse elemento é definido com base no intervalo preferível entre os componentes. Se nenhum componente vizinho for encontrado, o `min`, `pref` e `max` devem ser definidos como 0.

Parâmetros:

`type` – o tipo de intervalo, uma das constantes do `LayoutStyle`

Retorna:

esse `SequentialGroup`

Lança:

`IllegalArgumentException` – se o tipo não for `LayoutStyle.RELATED` ou `LayoutStyle.UNRELATED`

Relaciona-se com:

`LayoutStyle`

addPreferredGap

```
public GroupLayout.SequentialGroup addPreferredGap(int type,
                                                   int pref,
                                                   int max)
```

Adiciona um elemento para o intervalo preferido entre os componentes mais próximos. Ou seja, durante o layout, os componentes vizinhos são encontrados, e o `min` desse elemento é definido com base no intervalo preferido entre os componentes. Se nenhum componente vizinho for encontrado, o `min` é definido como 0. Esse método permite especificar os tamanhos preferível e máximo pela forma dos argumentos `pref` e `max`. Podem ser um valor >= 0, quando `pref` ou `max` são o máximo do argumento e o intervalo preferível, do `DEFAULT_VALUE`, quando o valor é o mesmo que o intervalo preferível.

Parâmetros:

`type` – o tipo de intervalo, `LayoutStyle.RELATED` ou `LayoutStyle.UNRELATED`

`pref` - o tamanho preferido; `DEFAULT_SIZE` ou um valor > 0

`pref` - o tamanho máximo; `DEFAULT_SIZE`, `PREFERRED_SIZE` ou um valor > 0

Retorna:

esse `SequentialGroup`.

Lança:

`IllegalArgumentException` – se o tipo não for `LayoutStyle.RELATED` ou `LayoutStyle.UNRELATED` ou se `pref/max != DEFAULT_SIZE` e `pref/max != 0`, ou `pref > max`.

Relaciona-se com:

`LayoutStyle`.

NOTA O uso do sinal "!=" em expressões como a acina denota uma relação matemática de desigualdade.

addContainerGap

```
public GroupLayout.SequentialGroup addContainerGap()
```

Adiciona um elemento representando o intervalo preferido entre uma margem do receptáculo e o componente próximo/anterior. Isto não terá efeito se o elemento próximo/anterior não for um componente e não tocar em nenhuma margem do receptáculo principal.

Retorna:

esse `SequentialGroup`.

addContainerGap

```
public GroupLayout.SequentialGroup addContainerGap(int pref,
                                                   int max)
```

Adiciona um elemento representando o intervalo preferido entre uma margem do receptáculo e o componente próximo/anterior. Isto não terá efeito se o elemento próximo/anterior não for um componente e não tocar em nenhuma margem do receptáculo principal.

Parâmetros:

`pref` - o tamanho preferível; `DEFAULT_SIZE` ou um valor > 0

`max` - o tamanho máximo; `DEFAULT_SIZE`, `PREFERRED_SIZE` ou um valor > 0.

Retorna:

esse `SequentialGroup`

Lança:

`IllegalArgumentException` – se `pref/max` for != `DEFAULT_SIZE` e 0, ou `pref > max`.

28.12 Classe GroupLayout.Spring

28.12.1 Descrição da classe

`com.sun.dtv.lwuit.layouts`

```
java.lang.Object
  └com.sun.dtv.lwuit.layouts.GroupLayout.Spring
```

Subclasses diretas conhecidas

`GroupLayout.Group`

Classe externa

`GroupLayout`

```
abstract public class GroupLayout.Spring
extends Object
```

`Spring` consiste em um intervalo: `min`, `pref` e `max`, um valor em algum lugar no meio deste intervalo e um local. As subclasses devem sobrescrever os métodos para obter o `min/max/pref` e provavelmente precisarão sobrescrever o método `setSize`. A `Spring` automaticamente armazena em *cache* o `min/max/pref`. Se o `min/pref/max` possuir alterações internas ou precisar ser atualizado, deve-se executar `clear()`.

28.13 Classe Layout

28.13.1 Descrição da classe

com.sun.dtv.lwuit.layouts

java.lang.Object

└com.sun.dtv.lwuit.layouts.Layout

Subclasses diretas conhecidas

BorderLayout, BoxLayout, CoordinateLayout, FlowLayout, GridLayout, GroupLayout

abstract public class Layout

extends Object

Classe abstrata que pode ser usada para organizar os componentes em um container usando um algoritmo pré-definidos. Esta classe pode ser implementada externamente e é similar aos gerenciadores de layout AWT/Swing.

28.13.2 Índice de construtores

Layout()

28.13.3 Índice de métodos

void **addLayoutComponent**(Object value, Component comp, Container c)

Alguns layouts podem opcionalmente monitorar a adição de elementos com metadados que permitem que o usuário dê “dicas” do posicionamento do objeto.

Object **getComponentConstraint**(Component comp)

Retorna a restrição do componente.

abstract Dimension **getPreferredSize**(Container parent)

Retorna o tamanho preferido do receptáculo.

boolean **isOverlapSupported**()

Esse método retorna *true* se o layout permitir que os componentes se sobreponham.

abstract void **layoutContainer**(Container parent)

Esquematiza o secundário do receptáculo principal determinando.

void **removeLayoutComponent**(Component comp)

Remove o componente do layout. Essa operação é somente útil se o layout mantiver referências aos componentes dentro dele.

28.13.4 Detalhe dos construtores

Layout

public Layout()

28.13.5 Detalhe dos métodos

layoutContainer

```
public abstract void layoutContainer(Container parent)
```

Esquematiza o secundário do receptáculo principal determinando.

Parâmetros:

`parent` – o receptáculo principal determinado

getPreferredSize

```
public abstract Dimension getPreferredSize(Container parent)
```

Retorna o tamanho preferido do receptáculo.

Parâmetros:

`parent` – o receptáculo principal

Retorna:

o tamanho preferido do receptáculo

addLayoutComponent

```
public void addLayoutComponent(Object value,
                               Component comp,
                               Container c)
```

Alguns layouts podem opcionalmente monitorar a adição de elementos com metadados que permitem que o usuário dê “dicas” do posicionamento do objeto.

Parâmetros:

`value` – informações de metadados opcionais, como a orientação de alinhamento

`comp` – o componente adicionado ao layout

`c` – o receptáculo principal

removeLayoutComponent

```
public void removeLayoutComponent(Component comp)
```

Remove o componente do layout. Essa operação é somente útil se o layout mantiver referências aos componentes dentro dele.

Parâmetros:

`comp` – o componente removido do layout

getComponentConstraint

```
public Object getComponentConstraint(Component comp)
```

Retorna a restrição do componente.

Parâmetros:

`comp` – o componente

Retorna:

a restrição do componente

isOverlapSupported

`public boolean isOverlapSupported()`

Esse método retorna *true* se o layout permitir que os componentes se sobreponham.

Retorna:

true se os componentes puderem cruzar nesse layout

28.14 Classe LayoutStyle

28.14.1 Descrição da classe

`com.sun.dtv.lwuit.layouts`

```
java.lang.Object
  └com.sun.dtv.lwuit.layouts.LayoutStyle
```

`public class LayoutStyle`

`extends Object`

`LayoutStyle` é usado para determinar quanto espaço deve inserir entre os componentes durante o layout. O `LayoutStyle` pode ser obtido para dois componentes ou para um componente relativo para uma margem do receptáculo principal. O total do espaço pode variar dependendo se os componentes forem logicamente agrupados (`RELATED`).

Essa classe é principalmente útil para JREs anterior a 1.6. Em 1.6 API, pois foi adicionada a Swing. Quando executada em um JRE 1.6 ou superior, chamará os métodos apropriados no Swing.

28.14.2 Índice de campos

`static int` **`INDENT`**

Possível argumento para `getPreferredGap`.

`static int` **`RELATED`**

Possível argumento para `getPreferredGap`.

`static int` **`UNRELATED`**

Possível argumento para `getPreferredGap`.

28.14.3 Índice de construtores

`LayoutStyle``()`

28.14.4 Índice de métodos

`int` **`getContainerGap`**`(Component component, int position, Container parent)`

Retorna o total de espaço para posicionar um componente dentro de seu principal.

`int` **`getPreferredGap`**`(Component component1, Component component2, int type, int position, Container parent)`

Retorna o total de espaço para usar entre dois componentes.

`static LayoutStyle` **`getSharedInstance`**`()`

Métodos de fabricação para obter o atual objeto `LayoutStyle` apropriado para a atual aparência.

`static void` **`setSharedInstance`**`(LayoutStyle layout)`

Define a instância `LayoutStyle` para usar nessa aparência.

28.14.5 Detalhe dos campos

RELATED

```
public static final int RELATED = 0
```

Possível argumento para `getPreferredGap`. Usado para indicar os dois componentes que são agrupados.

UNRELATED

```
public static final int UNRELATED = 1
```

Possível argumento para `getPreferredGap`. Usado para indicar os dois componentes que não são agrupados.

INDENT

```
public static final int INDENT = 3
```

Possível argumento para `getPreferredGap`. Usado para indicar a distância para indentar um componente que está sendo solicitado. Para indicar visualmente om conjunto de componentes relacionados, eles devem ser muitas vezes indentados horizontalmente, a constante `INDENT` para isso. Por exemplo, para indentar uma caixa de seleção relativa a um label, use essa constante para `getPreferredGap`.

28.14.6 Detalhe dos construtores

LayoutStyle

```
public LayoutStyle()
```

28.14.7 Detalhe dos métodos

setSharedInstance

```
public static void setSharedInstance(LayoutStyle layout)
```

Define a instância `LayoutStyle` para usar nessa aparência. Você normalmente não precisa executar isso, em vez disso, use o `getter` que retornará o `LayoutStyle` apropriado para a aparência atual.

Parâmetros:

`layout` – o `LayoutStyle` a ser usado. Um valor `null` indica que o padrão deve ser usado

Relaciona-se com:

`getSharedInstance()`

getSharedInstance

```
public static LayoutStyle getSharedInstance()
```

Métodos de fabricação para obter o atual objeto `LayoutStyle` apropriado para a atual aparência.

Retorna:

a instância atual do `LayoutStyle`

Relaciona-se com:

```
setSharedInstance(com.sun.dtv.lwuit.layouts.LayoutStyle)
```

getPreferredGap

```
public int getPreferredGap(Component component1,
                           Component component2,
                           int type,
                           int position,
                           Container parent)
```

Retorna o total de espaço para usar entre dois componentes. O valor de retorno indica a distância para inserir `component2` relativo ao `component1`. Por exemplo, o seguinte retorna o total de espaço para inserir entre `component2` e `component2` quando `component2` for inserido verticalmente acima do `component1`:

```
int gap = getPreferredGap(component1, component2,
                          LayoutStyle.RELATED,
                          SwingConstants.NORTH, parent);
```

O parâmetro `type` indica o tipo de *gap* sendo solicitado. A Tabela 53 lista tipos de *gap* suportados.

Tabela 53 - Tipos de *gap* suportados

`RELATED`	Se os dois componentes estiverem no mesmo principal e exibirem itens similares relacionados logicamente, use `RELATED`.
`UNRELATED`	Se os dois componentes estiverem no mesmo principal, mas exibirem itens não relacionados logicamente, use `UNRELATED`.
`INDENT`	Usado para obter a distância preferida para indentar um componente em relação a outro. Por exemplo, se você quiser indentar horizontalmente um `JcheckBox` em relação a um `Jlabel`, use `INDENT`. Isto é útil somente para o eixo horizontal.

É importante notar que algumas aparências podem não distinguir entre `RELATED` e `UNRELATED`.

O valor de retorno não é indentado para considerar o tamanho e posição atuais do `component2` ou `component1`. O valor do retorno pode levar em consideração várias propriedades dos componentes. Por exemplo, o espaço pode variar de acordo com o tamanho da fonte ou o tamanho preferido do componente.

Parâmetros:

`component1` – o `JComponent` `component2` está sendo inserido em relação a

`component2` - o `JComponent` sendo inserido

`type` – como os dois componentes estão sendo inseridos

`position` – a posição em que o `component2` está sendo inserido em relação ao `component1`; `SwingConstants.NORTH`, `SwingConstants.SOUTH`, `SwingConstants.EAST` ou `SwingConstants.WEST`

`parent` – o principal do `component2`; pode ser diferente do principal atual e pode ser `null`

Retorna:

o total de espaço para inserir entre os dois componentes

Lança:

`IllegalArgumentException` – se a posição não for `SwingConstants.NORTH`, `SwingConstants.SOUTH`, `SwingConstants.EAST` ou `SwingConstants.WEST`; se o tipo não for `INDENT`, `RELATED` ou `UNRELATED`; ou se o `component1` ou `component2` for `null`

getContainerGap

```
public int getContainerGap(Component component,
                           int position,
                           Container parent)
```

Retorna o total de espaço para posicionar um componente dentro de seu principal.

Parâmetros:

`component` - o `Component` sendo posicionado

`position` – o componente da posição está sendo inserido em relação aos seu principal; `SwingConstants.NORTH`, `SwingConstants.SOUTH`, `SwingConstants.EAST` ou `SwingConstants.WEST`

`parent` – o principal do `component`; pode ser diferente do principal atual e pode ser `null`

Retorna:

o total de espaço para inserir entre o componente e a margem determinada

Lança:

`IllegalArgumentException` – se a posição não for `SwingConstants.NORTH`, `SwingConstants.SOUTH`, `SwingConstants.EAST` ou `SwingConstants.WEST`; o componente é `null`

29 Pacote com.sun.dtv.lwuit.list

29.1 Descrição do pacote

As listas são altamente personalizáveis e servem como base para `ComboBox` e outros componentes (como carrosséis, etc.). Empregam uma abordagem MVC similar ao Swing incluindo o padrão de renderizador. `ListCellRenderer` permite personalizar a aparência de uma entrada da lista e funciona como um "*rubber stamp*" desenhando o componente renderizado e descartando seu estado, permitindo assim listas muito grandes com muito pouco overhead de estado de componentes.

`ListModel` permite representar o estrutura de dados subjacentes para a `List/ComboBox` sem exigir que todos os dados estejam na memória ou em uma estrutura específica. Isso permite que um modelo represente uma fonte de dados de qualquer tipo. Agrupada com o renderizador, a fonte de dados pode retornar em um estado de representação interna e ainda pode ser renderizada devidamente para a tela.

NOTA Este pacote está especificado desde o Java DTV 1.0.

29.2 Índice de interfaces

ListCellRenderer

Uma ferramenta "*rubber stamp*" que permite extrair um componente (geralmente a mesma instância do

componente para todas as execuções) que é inicializado para o valor do item atual extraído do modelo. Esse componente é desenhado da lista e descartado.

ListModel

Representa a estrutura de dados da lista, permitindo assim que uma lista represente qualquer fonte de dados potencial por referenciar diferentes implementações dessa interface.

29.3 Índice de classes

DefaultListCellRenderer

Implementação padrão do renderizador com base em um label. Consulte a interface `ListCellRenderer` para mais informações sobre o uso e o propósito dessa classe.

DefaultListModel

Implementação padrão do modelo de lista com base em um vetor de elementos.

29.4 Classe DefaultListCellRenderer

29.4.1 Descrição da classe

`com.sun.dtv.lwuit.list`

```
java.lang.Object
  └com.sun.dtv.lwuit.Component
      └com.sun.dtv.lwuit.Label
          └com.sun.dtv.lwuit.list.DefaultListCellRenderer
```

Todas as interfaces implementadas

`Animated, Animation, ListCellRenderer, MatteEnabled, StyleListener, ViewOnlyComponent`

```
public class DefaultListCellRenderer
extends Label
implements ListCellRenderer
```

Implementação padrão do renderizador com base em um label. Consulte a interface `ListCellRenderer` para mais informações sobre o uso e o propósito dessa classe.

Campos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`BOTTOM, BRB_CENTER_OFFSET, BRB_CONSTANT_ASCENT, BRB_CONSTANT_DESCENT, BRB_OTHER, CENTER, LEFT, RIGHT, TOP`

Campos herdados da interface `com.sun.dtv.ui.`**`Animated`**

`ALTERNATING, LOOP, REPEATING`

Campos herdados da interface `com.sun.dtv.ui.`**`ViewOnlyComponent`**

`HORIZONTAL_ALIGN_CENTER, HORIZONTAL_ALIGN_JUSTIFIED, HORIZONTAL_ALIGN_LEFT, HORIZONTAL_ALIGN_RIGHT, SCALE_ASPECT_PROOF, SCALE_NO, SCALE_NO_ASPECT_PROOF, STATE_DISABLED, STATE_ENABLED, VERTICAL_ALIGN_BOTTOM, VERTICAL_ALIGN_CENTER, VERTICAL_ALIGN_JUSTIFIED, VERTICAL_ALIGN_TOP`

29.4.2 Índice de construtores

DefaultListCellRenderer()

Cria uma nova instância do DefaultCellRenderer.

DefaultListCellRenderer(boolean showNumbers)

Cria uma nova instância do DefaultCellRenderer.

29.4.3 Índice de métodos

Component **getListCellRendererComponent**(List list, Object value, int index, boolean isSelected)

Retorna uma instância do componente que já é definida para renderizar “valor”.

Component **getListFocusComponent**(List list)

Retorna uma instância do componente que é pintada sob o renderizador atualmente focado e é animada para suavizar a rolagem.

int **getSelectionTransparency**()

O fator de transparência do plano de fundo para aplicar ao foco de seleção.

boolean **isShowNumbers**()

Indica se a numeração deve existir para o renderizador de célula padrão.

void **repaint**()

Cancelado para não fazer nada e remover um problema de performance quando as alterações do renderizador realizarem chamadas desnecessárias de repaint.

void **setSelectionTransparency**(int selectionTransparency)

O fator de transparência do plano de fundo para aplicar ao foco de seleção.

void **setShowNumbers**(boolean showNumbers)

Indica se a numeração deve existir para o renderizador de célula padrão.

Métodos herdados da classe com.sun.dtv.lwuit.**Label**

animate, getAlignment, getAnimateContent, getBaselineResizeBehavior, getGap, getGraphicContent, getHorizontalAlignment, getIcon, getInteractionState, getMatte, getScalingMode, getShiftText, getText, getTextContent, getTextLayoutManager, getTextPosition, getVerticalAlignment, isDoubleBuffered, isEndsWith3Points, isOpaque, isTickerEnabled, isTickerRunning, paint, processEvent, setAlignment, setAnimateContent, setEndsWith3Points, setGap, setGraphicContent, setHorizontalAlignment, setIcon, setInteractionState, setMatte, setScalingMode, setShiftText, setText, setTextContent, setTextLayoutManager, setTextPosition, setTickerEnabled, setVerticalAlignment, startTicker, stopTicker

Métodos herdados da classe com.sun.dtv.lwuit.**Component**

addFocusListener, calcPreferredSize, contains, deinitialize, getAbsoluteX, getAbsoluteY, getAnimationMode, getBaseline, getBottomGap, getBounds, getClientProperty, getComponentForm, getDelay, getHeight, getNextFocusDown, getNextFocusLeft, getNextFocusRight, getNextFocusUp, getParent, getPosition,

getPreferredSize, getRepetitionMode, getScrollAnimationSpeed, getScrollX, getScrollY, getSideGap, getStyle, getUIID, getWidth, getX, getY, handlesInput, hasFocus, initialize, isEnabled, isFocusable, isFocusPainted, isInitialized, isRunning, isScrollableX, isScrollableY, isScrollVisible, isSmoothScrolling, isVisible, jumpTo, keyPressed, keyReleased, keyRepeated, longKeyPress, paintBackgrounds, paintComponent, paintComponent, pointerDragged, pointerPressed, pointerReleased, putClientProperty, refreshTheme, removeFocusListener, repaint, requestFocus, scrollRectToVisible, setAnimationMode, setCellRenderer, setDelay, setEnabled, setFocus, setFocusable, setFocusPainted, setHandlesInput, setHeight, setInitialized, setIsScrollVisible, setNextFocusDown, setNextFocusLeft, setNextFocusRight, setNextFocusUp, setPreferredSize, setRepetitionMode, setScrollAnimationSpeed, setScrollX, setScrollY, setShouldCalcPreferredSize, setSize, setSmoothScrolling, setStyle, setUIID, setVisible, setWidth, setX, setY, start, stop, styleChanged, toString

29.4.4 Detalhe dos construtores

DefaultListCellRenderer

```
public DefaultListCellRenderer()
```

Cria uma nova instância do `DefaultCellRenderer`.

DefaultListCellRenderer

```
public DefaultListCellRenderer(boolean showNumbers)
```

Cria uma nova instância do `DefaultCellRenderer`.

Parâmetros:

`showNumbers` – indica se os números são mostrados

29.4.5 Detalhe dos métodos

getListCellRendererComponent

```
public Component getListCellRendererComponent(List list,
                                              Object value,
                                              int index,
                                              boolean isSelected)
```

Descrição copiada da interface: **`ListCellRenderer`**

Retorna uma instância do componente que já é definida para renderizar “valor”. Apesar de não ser uma exigência, muitos renderizadores geralmente se originam de um componente (como um label) e retornam “isto”. Perceba que um valor `null` para o argumento de valor pode ser enviado ao atualizar o tema da lista.

Especificado por:

`getListCellRendererComponent` na interface `ListCellRenderer`

Parâmetros:

`list` – o objeto Lista

`value` – um Objeto

`index` – o índice

`isSelected` – indica se o componente está selecionado

Retorna:

um objeto Component

getListFocusComponent

```
public Component getListFocusComponent(List list)
```

Descrição copiada da interface: **`ListCellRenderer`**

Retorna uma instância do componente que é pintada sob o renderizador atualmente focado e é animada para suavizar a rolagem. Quando a seleção se move, esse componente é desenhado acima/abaixo dos itens da lista. Recomenda-se dar a esse componente algum nível de transparência (Ver acima do exemplo do código). Esse método é opcional; uma implementação pode escolher retornar `null`.

Especificado por:

`getListFocusComponent` na interface `ListCellRenderer`

Parâmetros:

`list` – a lista

Retorna:

uma instância do componente que é pintada sob o renderizador atualmente focado e é animada para suavizar a rolagem.

repaint

```
public void repaint()
```

Cancelado para não fazer nada e remover um problema de performance quando as alterações do renderizador realizarem chamadas desnecessárias de repaint.

Substituições:

`repaint` na classe `Component`

isShowNumbers

```
public boolean isShowNumbers()
```

Indica se a numeração deve existir para o renderizador de célula padrão.

Retorna:

true se a numeração deve existir para o renderizador de célula padrão

setShowNumbers

```
public void setShowNumbers(boolean showNumbers)
```

Indica se a numeração deve existir para o renderizador de célula padrão.

Parâmetros:

`showNumbers` - indica se a numeração deve existir para o renderizador de célula padrão

getSelectionTransparency

```
public int getSelectionTransparency()
```

O fator de transparência do plano de fundo para aplicar ao foco de seleção.

Retorna:

o fator de transparência do plano de fundo

Relaciona-se com:

`setSelectionTransparency(int)`

setSelectionTransparency

`public void setSelectionTransparency(int selectionTransparency)`

O fator de transparência do plano de fundo para aplicar ao foco de seleção.

Parâmetros:

`selectionTransparency` - o fator de transparência do plano de fundo

Relaciona-se com:

`getSelectionTransparency()`

29.5 Classe DefaultListModel

29.5.1 Descrição da classe

`com.sun.dtv.lwuit.list`

```
java.lang.Object
  └com.sun.dtv.lwuit.list.DefaultListModel
```

Todas as interfaces implementadas

`ListModel`

```
public class DefaultListModel
extends Object
implements ListModel
```

Implementação padrão do modelo de lista com base em um vetor de elementos.

29.5.2 Índice de construtores

`DefaultListModel``()`

Cria uma nova instância do `DefaultListModel`.

`DefaultListModel``(Object[] items)`

Cria uma nova instância do `DefaultListModel`.

`DefaultListModel``(Vector items)`

Cria uma nova instância do `DefaultListModel`.

29.5.3 Índice de métodos

`void` **`addDataChangedListener`**`(DataChangedListener l)`

Executado para indicar interesse em eventos de alteração futuros.

`void` **`addItem`**`(Object item)`

Adiciona o item determinado ao final desta lista.

`void` **`addItemAtIndex`**`(Object item, int index)`

Adiciona um item à lista no índice determinado.

`void` **`addSelectionListener`**`(SelectionListener l)`

Executado para indicar interesse em eventos de seleção futuros.

`Object` **`getItemAt`**`(int index)`

Retorna o item na compensação determinada.

`int` **`getSelectedIndex`**`()`

Retorna a compensação da lista selecionada.

`int` **`getSize`**`()`

Retorna o número de itens na lista.

`void` **`removeAll`**`()`

Remove todos os elementos do modelo.

`void` **`removeDataChangedListener`**`(DataChangedListener l)`

Executado para não indicar mais nenhum interesse em eventos de alteração futuros.

`void` **`removeItem`**`(int index)`

Remove o item na posição determinada nessa lista.

`void` **`removeSelectionListener`**`(SelectionListener l)`

Executado para indicar nenhum interesse em eventos de seleção futuros.

`void` **`setItem`**`(int index, Object item)`

Altera o item no índice determinado.

`void` **`setSelectedIndex`**`(int index)`

Define se a compensação da lista selecionada pode ser definida para -1 para limpar a seleção.

29.5.4 Detalhe dos construtores

DefaultListModel

`public DefaultListModel()`

Cria uma nova instância do `DefaultListModel`.

DefaultListModel

`public DefaultListModel(Vector items)`

Cria uma nova instância do `DefaultListModel`.

Parâmetros:

`list` – os itens da lista

DefaultListModel

`public DefaultListModel(Object[] items)`

Cria uma nova instância do `DefaultListModel`.

Parâmetros:

`list` – os itens da lista

29.5.5 Detalhe dos métodos

getItemAt

`public Object getItemAt(int index)`

Descrição copiada da interface: **`ListModel`**

Retorna o item na compensação determinada.

Especificado por:

`getItemAt` na interface `ListModel`

Parâmetros:

`index` – um índice dentro dessa lista

Retorna:

o item no índice determinado

getSize

`public int getSize()`

Descrição copiada da interface: **`ListModel`**

Retorna o número de itens na lista.

Especificado por:

`getSize` na interface `ListModel`

Retorna:

o número de itens na lista

getSelectedIndex

`public int getSelectedIndex()`

Descrição copiada da interface: **`ListModel`**

Retorna a compensação da lista selecionada.

Especificado por:

`getSelectedIndex` na interface `ListModel`

Retorna:

o índice da lista selecionada

addItem

`public void addItem(Object item)`

Descrição copiada da interface: **`ListModel`**

Adiciona o item determinado ao final desta lista. Uma operação opcional para listas alteráveis. Pode abrir uma exceção de uma operação não suportada se um modelo da lista não for alterável.

Especificado por:

`addItem` na interface `ListModel`

Parâmetros:

`item` - o item a ser adicionado

setItem

```
public void setItem(int index,
                    Object item)
```

Altera o item no índice determinado.

Parâmetros:

`index` – o índice

`item` - o objeto a ser inserido como o novo item na posição determinada

addItemAtIndex

```
public void addItemAtIndex(Object item,
                           int index)
```

Adiciona um item à lista no índice determinado.

Parâmetros:

`item` - o item a ser adicionado

`index` - a posição do índice na lista

removeItem

```
public void removeItem(int index)
```

Descrição copiada da interface: **`ListModel`**

Remove o item na posição determinada nessa lista.

Especificado por:

`removeItem` na interface `ListModel`

Parâmetros:

`index` - o índice do item a ser removido

removeAll

```
public void removeAll()
```

Remove todos os elementos do modelo.

setSelectedIndex

```
public void setSelectedIndex(int index)
```

Descrição copiada da interface: **`ListModel`**

Define se a compensação da lista selecionada pode ser definida para -1 para limpar a seleção.

Especificado por:

`setSelectedIndex` na interface `ListModel`

Parâmetros:

`index` – um índice dentro dessa lista

addDataChangedListener

`public void addDataChangedListener(DataChangedListener l)`

Descrição copiada da interface: **`ListModel`**

Executado para indicar interesse em eventos de alteração futuros.

Especificado por:

`addDataChangedListener` na interface `ListModel`

Parâmetros:

`l` - um *listener* alterado por dado

removeDataChangedListener

`public void removeDataChangedListener(DataChangedListener l)`

Descrição copiada da interface: **`ListModel`**

Executado para não indicar mais nenhum interesse em eventos de alteração futuros.

Especificado por:

`removeDataChangedListener` na interface `ListModel`

Parâmetros:

`l` - um *listener* alterado por dado

addSelectionListener

`public void addSelectionListener(SelectionListener l)`

Descrição copiada da interface: **`ListModel`**

Executado para indicar interesse em eventos de seleção futuros.

Especificado por:

`addSelectionListener` na interface `ListModel`

Parâmetros:

`l` - um *listener* de seleção

removeSelectionListener

`public void removeSelectionListener(SelectionListener l)`

Descrição copiada da interface: **`ListModel`**

Executado para indicar nenhum interesse em eventos de seleção futuros.

Especificado por:

`removeSelectionListener` na interface `ListModel`

Parâmetros:

`l` - um *listener* de seleção

29.6 Interface ListCellRenderer

29.6.1 Descrição da interface

com.sun.dtv.lwuit.list

Todas as classes implementadoras conhecidas

DefaultListCellRenderer

public interface ListCellRenderer

Uma ferramenta "*rubber stamp*" que permite extrair um componente (geralmente a mesma instância do componente para todas as execuções) que é inicializado para o valor do item atual extraído do modelo. Esse componente é desenhado da lista e descartado. Nenhum estado do componente é mantido e o componente é essencialmente descartado.

Uma instância de um renderizador pode ser desenvolvida como:

```
public class MyYesNoRenderer extends Label implements ListCellRenderer
 {
      public Component getListCellRendererComponent(List list, Object value,
                                          int index, boolean isSelected)
      {
                                              if(((Boolean)value).booleanValue())    {
                                                                    setText("Yes");
                                                                       }       else
                                                                                  {
                                                                     setText("No");
                                                                                  }
                                                                 return       this;
                                                                                  }
                                    public    Component    getListFocusComponent(List    list)
                                                                                  {
                                                                 Label label = new label("");
                                                          label.getStyle().setBgTransparency(100);
                                                                       return      label;
                                                                                  }
}
```

Recomenda-se que o componente cujos os valores são manipulados não suportem recursos como repaint(). Isso é realizado pelo cancelamento do repaint na subclass com uma implementação vazia. Isso é sugerido por razões de performance, ou toda alteração feita ao componente poderia acionar um repaint que não faria nada, mas ainda custaria em termos de processamento.

29.6.2 Índice de métodos

Component **getListCellRendererComponent**(List list, Object value, int index, boolean isSelected)

Retorna uma instância do componente que já é definida para renderizar "valor".

Component **getListFocusComponent**(List list)

Retorna uma instância do componente que é pintada sob o renderizador atualmente focado e é animada para

suavizar a rolagem.

29.6.3 Detalhe dos métodos

getListCellRendererComponent

```
Component getListCellRendererComponent(List list,
                                       Object value,
                                       int index,
                                       boolean isSelected)
```

Retorna uma instância do componente que já é definida para renderizar “valor”. Apesar de não ser uma exigência, muitos renderizadores geralmente se originam de um componente (como um label) e retornam “isto”. Perceba que um valor `null` para o argumento de valor pode ser enviado ao atualizar o tema da lista.

Parâmetros:

`list` – o objeto Lista

`value` – um Objeto

`index` – o índice

`isSelected` – indica se o componente está selecionado

Retorna:

um objeto Component

getListFocusComponent

```
Component getListFocusComponent(List list)
```

Retorna uma instância do componente que é pintada sob o renderizador atualmente focado e é animada para suavizar a rolagem. Quando a seleção se move, esse componente é desenhado acima/abaixo dos itens da lista. Recomenda-se dar a esse componente algum nível de transparência (Ver acima do exemplo do código). Esse método é opcional; uma implementação pode escolher retornar `null`.

Parâmetros:

`list` – a lista

Retorna:

uma instância do componente que é pintada sob o renderizador atualmente focado e é animada para suavizar a rolagem.

Relaciona-se com:

`Component.setSmoothScrolling()`

29.7 Interface ListModel

29.7.1 Descrição da interface

`com.sun.dtv.lwuit.list`

Todas as classes implementadoras conhecidas

`DefaultListModel`

```
public interface ListModel
```

Representa a estrutura de dados da lista, permitindo assim que a lista represente qualquer fonte de dados potenciais por referenciar implementações diferentes dessa interface. Ex.: um modelo de lista pode ser implementado de forma a restaurar os dados diretamente a partir do storage (apesar de que seria recomendado

ocultar).

É responsabilidade da lista notificar observadores (especificamente a List de visualização) de quaisquer alterações em seu estado (itens removidos/adicionados/alterados etc.). Assim os dados seriam atualizados na visualização.

29.7.2 Índice de métodos

`void` **`addDataChangedListener`**`(DataChangedListener l)`

Executado para indicar interesse em eventos de alteração futuros.

`void` **`addItem`**`(Object item)`

Adiciona o item determinado ao final desta lista.

`void` **`addSelectionListener`**`(SelectionListener l)`

Executado para indicar interesse em eventos de seleção futuros.

`Object` **`getItemAt`**`(int index)`

Retorna o item na compensação determinada.

`int` **`getSelectedIndex`**`()`

Retorna a compensação da lista selecionada.

`int` **`getSize`**`()`

Retorna o número de itens na lista.

`void` **`removeDataChangedListener`**`(DataChangedListener l)`

Executado para não indicar mais nenhum interesse em eventos de alteração futuros.

`void` **`removeItem`**`(int index)`

Remove o item na posição determinada nessa lista.

`void` **`removeSelectionListener`**`(SelectionListener l)`

Executado para indicar nenhum interesse em eventos de seleção futuros.

`void` **`setSelectedIndex`**`(int index)`

Define se a compensação da lista selecionada pode ser definida para -1 para limpar a seleção.

29.7.3 Detalhe dos métodos

getItemAt

`Object getItemAt(int index)`

Retorna o item na compensação determinada.

Parâmetros:

`index` – um índice dentro dessa lista

Retorna:

o item no índice determinado

getSize

`int getSize()`

Retorna o número de itens na lista.

Retorna:

o número de itens na lista

getSelectedIndex

`int getSelectedIndex()`

Retorna a compensação da lista selecionada.

Retorna:

o índice da lista selecionada

Relaciona-se com:

`setSelectedIndex(int)`

setSelectedIndex

`void setSelectedIndex(int index)`

Define se a compensação da lista selecionada pode ser definida para -1 para limpar a seleção.

Parâmetros:

`index` – um índice dentro dessa lista

Relaciona-se com:

`getSelectedIndex()`

addDataChangedListener

`void addDataChangedListener(DataChangedListener l)`

Executado para indicar interesse em eventos de alteração futuros.

Parâmetros:

`l` - um *listener* alterado por dado

Relaciona-se com:

`removeDataChangedListener(com.sun.dtv.lwuit.events.DataChangedListener)`

removeDataChangedListener

`void removeDataChangedListener(DataChangedListener l)`

Executado para não indicar mais nenhum interesse em eventos de alteração futuros.

Parâmetros:

`l` - um *listener* alterado por dado

Relaciona-se com:

`addDataChangedListener(com.sun.dtv.lwuit.events.DataChangedListener)`

addSelectionListener

`void addSelectionListener(SelectionListener l)`

Executado para indicar interesse em eventos de seleção futuros.

Parâmetros:

`l` - um *listener* de seleção

Relaciona-se com:

`removeSelectionListener(com.sun.dtv.lwuit.events.SelectionListener)`

removeSelectionListener

`void removeSelectionListener(SelectionListener l)`

Executado para indicar nenhum interesse em eventos de seleção futuros.

Parâmetros:

`l` - um *listener* de seleção

Relaciona-se com:

`addSelectionListener(com.sun.dtv.lwuit.events.SelectionListener)`

addItem

`void addItem(Object item)`

Adiciona o item determinado ao final desta lista. Uma operação opcional para listas alteráveis. Pode abrir uma exceção de uma operação não suportada se um modelo da lista não for alterável.

Parâmetros:

`item` - o item a ser adicionado

Relaciona-se com:

`removeItem(int)`

removeItem

`void removeItem(int index)`

Remove o item na posição determinada nessa lista.

Parâmetros:

`index` - o índice do item a ser removido

Relaciona-se com:

`addItem(java.lang.Object)`

30 Pacote com.sun.dtv.lwuit.painter

30.1 Descrição do pacote

O pacote `Painter` permite desenhar elementos arbitrários de gráficos provenientes de imagens plain/scaled/lado a lado a gradientes e praticamente todas as formas de desenho gráfico que pudermos imaginar. Painters são “fixados” ao centro da API, permitindo assim personalizar todos os seus aspectos. Ver Component e Form.

NOTA Este pacote está especificado desde o Java DTV 1.0.

30.2 Índice de classes

BackgroundPainter

Um painter que desenha o plano de fundo de um componente baseado em seu estilo.

PainterChain

Uma cadeia de painter permite unir vários painters para fornecer um efeito “camada” onde cada painter desenha

somente um elemento.

30.3 Classe BackgroundPainter

30.3.1 Descrição da classe

com.sun.dtv.lwuit.painter

```
java.lang.Object
  └com.sun.dtv.lwuit.painter.BackgroundPainter
```

Todas as interfaces implementadas

Painter

```
public class BackgroundPainter
extends Object
implements Painter
```

Um painter que desenha o plano de fundo de um componente baseado em seu estilo.

30.3.2 Índice de construtores

BackgroundPainter(Component parent)

Cria um painter de plano de fundo para o componente determinado.

30.3.3 Índice de métodos

void **paint**(Graphics g, Rectangle rect)

Desenha dentro da área de recorte do retângulo determinado.

30.3.4 Detalhe dos construtores

BackgroundPainter

```
public BackgroundPainter(Component parent)
```

Cria um painter de plano de fundo para o componente determinado.

Parâmetros:

parent – o componente determinado

30.3.5 Detalhe dos métodos

paint

```
public void paint(Graphics g,
                  Rectangle rect)
```

Descrição copiada da interface: **Painter**

Desenha dentro da área de recorte do retângulo determinado.

Especificado por:

paint na interface Painter

Parâmetros:

g – o objeto Graphics

`rect` - a área de recorte do retângulo determinado

30.4 Classe PainterChain

30.4.1 Descrição da classe

`com.sun.dtv.lwuit.painter`

```
java.lang.Object
  └com.sun.dtv.lwuit.painter.PainterChain
```

Todas as interfaces implementadas

Painter

```
public class PainterChain
extends Object
implements Painter
```

Uma cadeia de painter permite unir vários painters para fornecer um efeito “camada” onde cada painter desenha somente um elemento.

30.4.2 Índice de construtores

`PainterChain``(Painter[] chain)`

Cria uma nova cadeia de painter que irá pintar todos os elementos na cadeia na sequência de 0 até o último elemento.

30.4.3 Índice de métodos

`PainterChain` **`addPainter`**`(Painter p)`

Cria uma nova cadeia baseada na cadeia existente com um novo elemento adicionado no final.

`static void` **`installGlassPane`**`(Form f, Painter p)`

Instala um painel de vidro no formulário determinado certificando-se de transformá-lo em uma cadeia de painter apenas se solicitado por um painter existente.

`void` **`paint`**`(Graphics g, Rectangle rect)`

Desenha dentro da área de recorte do retângulo determinado.

`PainterChain` **`prependPainter`**`(Painter p)`

Cria uma nova cadeia baseada na cadeia existente com um novo elemento adicionado no começo.

30.4.4 Detalhe dos construtores

PainterChain

`public PainterChain(Painter[] chain)`

Cria uma nova cadeia de painter que irá pintar todos os elementos na cadeia na sequência de 0 até o último elemento.

Parâmetros:

`chain` – os elementos da cadeia

30.4.5 Detalhe dos métodos

addPainter

```
public PainterChain addPainter(Painter p)
```

Cria uma nova cadeia baseada na cadeia existente com um novo elemento adicionado no final.

Parâmetros:

`p` – novo painter

Retorna:

novo elemento da cadeia

prependPainter

```
public PainterChain prependPainter(Painter p)
```

Cria uma nova cadeia baseada na cadeia existente com um novo elemento adicionado no começo.

Parâmetros:

`p` – novo painter

Retorna:

novo elemento da cadeia

paint

```
public void paint(Graphics g,
                  Rectangle rect)
```

Descrição copiada da interface: **`Painter`**

Desenha dentro da área de recorte do retângulo determinado.

Especificado por:

`paint` na interface `Painter`

Parâmetros:

`g` – o objeto Graphics

`rect` - a área de recorte do retângulo determinado

installGlassPane

```
public static void installGlassPane(Form f,
                                    Painter p)
```

Instala um painel de vidro no formulário determinado certificando-se de transformá-lo em uma cadeia de painter apenas se solicitado por um painter existente.

Parâmetros:

`f` - formulário no qual instalar a cadeia

`p` – painter a ser instalado

31 Pacote com.sun.dtv.lwuit.plaf

31.1 Descrição do pacote

A aparência do aplicativo pode ser inteiramente personalizada por meio desse pacote, ele representa uma camada processadora que pode ser plugada separadamente em runtime e tematizado para prover qualquer aparência personalizada. Diferentemente do pacote `PLAF` do Swing, essa camada não suporta nenhum aspecto de "*feel*", como no tratamento de um evento etc, visto que esses aspectos exigiriam uma camada muito maior e mais elaborada, inapropriada para fornecimento via OTA para dispositivos pequenos.

Tamanhos dos componentes são calculados também pelo `LookAndFeel`, visto que o tamanho é muito afetado pela aparência do aplicativo, por exemplo, a espessura de uma borda e o tamanho das fontes.

NOTA Este pacote está especificado desde o Java DTV 1.0.

31.2 Índice de classes

Border

Classe de base que permite renderizar uma borda para um componente. Uma borda é desenhada antes do componente e dentro da região de padding do componente.

DefaultLookAndFeel

Usado para renderizar a aparência padrão do LWUIT.

LookAndFeel

Permite que um desenvolvedor da interface do usuário personalize completamente a aparência do aplicativo por cancelar métodos de desenho/dimencionamento apropriadamente.

Style

Representa a aparência do componente determinado: cores, fontes, transparência, margem e padding & imagens.

UIManager

O *singleton* do ponto central gerenciando a aparência do aplicativo. Essa classe permite personalizar os estilos (temas) e a aparência da instância.

31.3 Classe Border

31.3.1 Descrição da classe

`com.sun.dtv.lwuit.plaf`

```
java.lang.Object
  └com.sun.dtv.lwuit.plaf.Border
```

```
public class Border
extends Object
```

Classe de base que permite renderizar uma borda para um componente. Uma borda é desenhada antes do componente e dentro da região de padding do componente. É responsabilidade do componente não desenhar fora da linha da borda.

Essa classe pode ser estendida para fornecer tipos de borda adicionais e tipos de borda personalizados.

Uma borda pode pintar opcionalmente o plano de fundo do componente. Isso depende do tipo de borda e é

normalmente exigido por bordas arredondadas que "conhecem" a área que deve ser preenchida.

31.3.2 Índice de construtores

Border()

31.3.3 Índice de métodos

static Border **createBevelLowered**()

Cria uma borda chanfrada diminuída com cores padrão, o realce é derivado do componente e a sombra é uma cor escura simples.

static Border **createBevelLowered**(Color highlightOuter, Color highlightInner, Color shadowOuter, Color shadowInner)

Cria uma borda chanfrada aumentada com cores determinadas.

static Border **createBevelRaised**()

Cria uma borda chanfrada diminuída com cores padrão, o realce é derivado do componente e a sombra é uma cor escura simples.

static Border **createBevelRaised**(Color highlightOuter, Color highlightInner, Color shadowOuter, Color shadowInner)

Cria uma borda chanfrada aumentada com cores determinadas.

static Border **createEmpty**()

Cria uma borda vazia. Isso é útil onde não queremos uma borda para um componente, mas queremos uma borda do foco etc...

static Border **createEtchedLowered**()

Cria uma borda entalhada diminuída com cores padrão, o realce é derivado do componente e a sombra é uma cor escura simples.

static Border **createEtchedLowered**(Color highlight, Color shadow)

Cria uma borda entalhada aumentada com cores determinadas.

static Border **createEtchedRaised**()

Cria uma borda entalhada diminuída com cores padrão, o realce é derivado do componente e a sombra é uma cor escura simples.

static Border **createEtchedRaised**(Color highlight, Color shadow)

Cria uma borda entalhada aumentada com cores determinadas.

static Border **createImageBorder**(Image top, Image topLeft, Image background)

As imagens determinadas são dispostas lado a lado apropriadamente em todo o lado correspondente da borda, giradas e posicionadas conforme esperado nos quatro cantos.

static Border **createImageBorder**(Image top, Image bottom, Image left, Image right, Image topLeft, Image topRight, Image bottomLeft, Image bottomRight, Image background)

As imagens determinadas são dispostas lado a lado apropriadamente em todo o lado correspondente da borda e posicionadas conforme esperado nos quatro cantos.

static Border **createLineBorder**(int thickness)

Cria uma borda da linha que usa a cor do primeiro plano do componente para desenhar.

static Border **createLineBorder**(int thickness, Color color)

Cria uma borda da linha que usa a cor determinada para o componente.

Border **createPressedVersion**()

Quando aplicadas a botões, as bordas produzem uma versão que reverte os efeitos da borda proporcionando uma sensação pressionada.

`static Border` **`createRoundBorder`**`(int arcWidth, int arcHeight)`

Cria uma borda de canto arredondado que usa a cor do primeiro plano do componente para desenhar.

`static Border` **`createRoundBorder`**`(int arcWidth, int arcHeight, Color color)`

Cria uma borda arredondada que usa a cor determinada para o componente.

`static Border` **`getDefaultBorder`**`()`

Recebe a borda padrão para o valor determinado.

`static Border` **`getEmpty`**`()`

Retorna uma borda vazia. Isso é geralmente útil para cancelar componentes que têm uma borda por padrão.

`Border` **`getFocusedInstance`**`()`

Retorna a versão focada da borda se houver uma instalada.

`Border` **`getPressedInstance`**`()`

Retorna a versão pressionada da borda se uma for definida pelo usuário.

`boolean` **`isBackgroundPainter`**`()`

Retorna *true* se instalar essa borda cancelar a pintura do plano de fundo do componente.

`void` **`paint`**`(Graphics g, Component c)`

Desenha a borda para o componente determinado, esse método é chamado antes que uma chamada para a pintura do plano de fundo seja feita.

`void` **`paintBorderBackground`**`(Graphics g, Component c)`

Tem efeito quando a borda exige responsabilidade para a pintura do plano de fundo. Normalmente o painter realiza esse trabalho, mas nesse caso a borda pode fazê-lo.

`static void` **`setDefaultBorder`**`(Border border)`

Define a borda padrão para o valor determinado.

`void` **`setFocusedInstance`**`(Border focused)`

Permite definir uma borda que agirá como a versão focada dessa borda.

`void` **`setPressedInstance`**`(Border pressed)`

Permite definir uma borda que agirá como a versão pressionada dessa borda.

31.3.4 Detalhe dos construtores

Borda

```
public Border()
```

31.3.5 Detalhe dos métodos

getEmpty

```
public static Border getEmpty()
```

Retorna uma borda vazia. Isso é geralmente útil para cancelar componentes que têm uma borda por padrão.

Retorna:

uma borda que não desenha nada

createEmpty

```
public static Border createEmpty()
```

Cria uma borda vazia. Isso é útil onde não queremos uma borda para um componente, mas queremos uma borda do foco etc...

Retorna:

uma borda que não desenha nada

createImageBorder

```
public static Border createImageBorder(Image top,
                                       Image bottom,
                                       Image left,
                                       Image right,
                                       Image topLeft,
                                       Image topRight,
                                       Image bottomLeft,
                                       Image bottomRight,
                                       Image background)
```

As imagens determinadas são dispostas lado a lado apropriadamente em todo o lado correspondente da borda e posicionadas conforme esperado nos quatro cantos. A imagem de plano de fundo é opcional e deve ser disposta lado a lado no plano de fundo, se necessário.

Por padrão, essa borda não cancela o plano de fundo a menos que uma imagem de plano de fundo seja determinada.

Parâmetros:

`top` - a imagem do lado de cima

`bottom` – a imagem do lado de baixo

`left` - a imagem do lado esquerdo

`right` – a imagem do lado direito

`topLeft` - a imagem do canto superior esquerdo

`topRight` – a imagem do canto superior direito

`bottomLeft` – a imagem do canto inferior esquerdo

`bottomRight` – a imagem do canto inferior direito

`background` – a imagem do plano de fundo

Retorna:

nova instância da borda

createImageBorder

```
public static Border createImageBorder(Image top,
                                       Image topLeft,
                                       Image background)
```

As imagens determinadas são dispostas lado a lado apropriadamente em todo o lado correspondente da borda, giradas e posicionadas conforme esperado nos quatro cantos. A imagem de plano de fundo é opcional e deve ser disposta lado a lado no plano de fundo, se necessário.

Por padrão, essa borda não cancela o plano de fundo a menos que uma imagem de plano de fundo seja determinada.

Note que essa versão do método é potencialmente muito mais eficiente, visto que as imagens são giradas internamente, e isso pode economizar bastante memória.

As imagens superior e superior esquerda devem ser quadradas! A largura e altura dessas imagens devem ser iguais, caso contrário, a rotação não funcionará conforme esperado.

Parâmetros:

`top` - a imagem do lado de cima

`topLeft` - a imagem do canto superior esquerdo

`background` – a imagem do plano de fundo

Retorna:

nova instância da borda

createLineBorder

```
public static Border createLineBorder(int thickness)
```

Cria uma borda da linha que usa a cor do primeiro plano do componente para desenhar.

Parâmetros:

`thickness` – espessura da borda em pixels

Retorna:

nova instância da borda

createLineBorder

```
public static Border createLineBorder(int thickness,
                                      Color color)
```

Cria uma borda da linha que usa a cor determinada para o componente.

Parâmetros:

`thickness` – espessura da borda em pixels

`color` – a cor da borda

Retorna:

nova instância da borda

createRoundBorder

```
public static Border createRoundBorder(int arcWidth,
                                       int arcHeight)
```

Cria uma borda de canto arredondado que usa a cor do primeiro plano do componente para desenhar. Devido a problemas técnicos (falta de recorte modelado), o overhead de performance e memória de bordas arredondadas pode ser baixo se usado com bgImage ou translucidez.

Essa borda cancela qualquer painter usado no componente e ignoraria tal painter.

Parâmetros:

`arcWidth` – o diâmetro horizontal do arco nos quatro cantos.

`arcHeight` - o diâmetro vertical do arco nos quatro cantos.

Retorna:

nova instância da borda

createRoundBorder

```
public static Border createRoundBorder(int arcWidth,
                                       int arcHeight,
                                       Color color)
```

Cria uma borda arredondada que usa a cor determinada para o componente. Devido a problemas técnicos (falta de recorte modelado), o overhead de performance e memória de bordas arredondadas pode ser baixo se usado com bgImage ou translucidez.

Essa borda cancela qualquer painter usado no componente e ignoraria tal painter.

Parâmetros:

`arcWidth` – o diâmetro horizontal do arco nos quatro cantos.

`arcHeight` - o diâmetro vertical do arco nos quatro cantos.

`color` – a cor da borda

Retorna:

nova instância da borda

createEtchedLowered

```
public static Border createEtchedLowered()
```

Cria uma borda entalhada diminuída com cores padrão, o realce é derivado do componente e a sombra é uma cor escura simples.

Retorna:

nova instância da borda

createEtchedLowered

```
public static Border createEtchedLowered(Color highlight,
                                         Color shadow)
```

Cria uma borda entalhada aumentada com cores determinadas.

Parâmetros:

`highlight` – o valor RGB da cor

`shadow` - o valor RGB da cor

Retorna:

nova instância da borda

createEtchedRaised

```
public static Border createEtchedRaised()
```

Cria uma borda entalhada diminuída com cores padrão, o realce é derivado do componente e a sombra é uma cor escura simples.

Retorna:

nova instância da borda

createEtchedRaised

```
public static Border createEtchedRaised(Color highlight,
                                        Color shadow)
```

Cria uma borda entalhada aumentada com cores determinadas.

Parâmetros:

`highlight` – o valor RGB da cor

`shadow` - o valor RGB da cor

Retorna:

nova instância da borda

isBackgroundPainter

```
public boolean isBackgroundPainter()
```

Retorna *true* se instalar essa borda cancelar a pintura do plano de fundo do componente.

Retorna:

true se for um painter do plano de fundo, caso contrário, *false*

createBevelLowered

```
public static Border createBevelLowered()
```

Cria uma borda chanfrada diminuída com cores padrão, o realce é derivado do componente e a sombra é uma cor escura simples.

Retorna:

nova instância da borda

createBevelLowered

```
public static Border createBevelLowered(Color highlightOuter,
                                        Color highlightInner,
                                        Color shadowOuter,
                                        Color shadowInner)
```

Cria uma borda chanfrada aumentada com cores determinadas.

Parâmetros:

`highlightOuter` – o RGB da margem exterior da área de realce

`highlightInner` - o RGB da margem interior da área de realce

`shadowOuter` - o RGB da margem exterior da área de sombreamento

`shadowInner` - o RGB da margem interior da área de sombreamento

Retorna:

nova instância da borda

createBevelRaised

```
public static Border createBevelRaised()
```

Cria uma borda chanfrada diminuída com cores padrão, o realce é derivado do componente e a sombra é uma cor escura simples.

Retorna:

nova instância da borda

createBevelRaised

```
public static Border createBevelRaised(Color highlightOuter,
                                       Color highlightInner,
                                       Color shadowOuter,
                                       Color shadowInner)
```

Cria uma borda chanfrada aumentada com cores determinadas.

Parâmetros:

`highlightOuter` – o RGB da margem exterior da área de realce

`highlightInner` - o RGB da margem interior da área de realce

`shadowOuter` - o RGB da margem exterior da área de sombreamento

`shadowInner` - o RGB da margem interior da área de sombreamento

Retorna:

nova instância da borda

setPressedInstance

```
public void setPressedInstance(Border pressed)
```

Permite definir uma borda que agirá como a versão pressionada dessa borda.

Parâmetros:

`pressed` – uma borda que deve ser retornada pelo método de versão pressionada

Relaciona-se com:

`getPressedInstance()`

setFocusedInstance

```
public void setFocusedInstance(Border focused)
```

Permite definir uma borda que agirá como a versão focada dessa borda.

Parâmetros:

`focused` - uma borda que deve ser retornada pelo método de versão focada

Relaciona-se com:

`getFocusedInstance()`

getFocusedInstance

```
public Border getFocusedInstance()
```

Retorna a versão focada da borda se houver uma instalada.

Retorna:

a versão focada da borda se houver uma instalada

Relaciona-se com:

```
setFocusedInstance(com.sun.dtv.lwuit.plaf.Border)
```

getPressedInstance

```
public Border getPressedInstance()
```

Retorna a versão pressionada da borda se uma for definida pelo usuário.

Retorna:

a versão pressionada da borda se uma for definida pelo usuário

Relaciona-se com:

```
setPressedInstance(com.sun.dtv.lwuit.plaf.Border)
```

createPressedVersion

```
public Border createPressedVersion()
```

Quando aplicadas a botões, as bordas produzem uma versão que reverte os efeitos da borda proporcionando uma sensação pressionada.

Retorna:

uma borda que dará a sensação de botão pressionado

paintBorderBackground

```
public void paintBorderBackground(Graphics g,
                                  Component c)
```

Tem efeito quando a borda exige responsabilidade para a pintura do plano de fundo. Normalmente o painter realiza esse trabalho, mas nesse caso a borda pode fazê-lo.

Parâmetros:

`g` – contexto gráfico no qual desenhar

`c` – componente cuja borda deve ser desenhada

paint

```
public void paint(Graphics g,
                  Component c)
```

Desenha a borda para o componente determinado, esse método é chamado antes que uma chamada para a pintura do plano de fundo seja feita.

Parâmetros:

`g` – contexto gráfico no qual desenhar

`c` – componente cuja borda deve ser desenhada

setDefaultBorder

`public static void setDefaultBorder(Border border)`

Define a borda padrão para o valor determinado.

Parâmetros:

`border` – novo valor da borda

Relaciona-se com:

`getDefaultBorder()`

getDefaultBorder

`public static Border getDefaultBorder()`

Recebe a borda padrão para o valor determinado.

Retorna:

a borda padrão para o valor determinado

Relaciona-se com:

`setDefaultBorder(com.sun.dtv.lwuit.plaf.Border)`

31.4 Classe DefaultLookAndFeel

31.4.1 Descrição da classe

`com.sun.dtv.lwuit.plaf`

```
java.lang.Object
  └com.sun.dtv.lwuit.plaf.LookAndFeel
      └com.sun.dtv.lwuit.plaf.DefaultLookAndFeel
```

Todas as interfaces implementadas

`FocusListener`

```
public class DefaultLookAndFeel
extends LookAndFeel
implements FocusListener
```

Usado para renderizar a aparência padrão do LWUIT.

31.4.2 Índice de construtores

`DefaultLookAndFeel``()`

Cria uma nova instância de `DefaultLookAndFeel`.

31.4.3 Índice de métodos

`void bind(Component cmp)`

Cada componente se amarra ao `LookAndFeel` permitindo que ele customize o componente.

```
void drawButton(Graphics g, Button b)
```

Chamado para desenhar um *widget* de botão.

```
void drawCheckBox(Graphics g, CheckBox cb)
```

Chamado para desenhar um *widget* de *checkbox*.

```
void drawComboBox(Graphics g, ComboBox cb)
```

Chamado para desenhar um *widget* de *combobox*.

```
void drawLabel(Graphics g, Label l)
```

Chamado para desenhar um *widget* de *label*.

```
void drawList(Graphics g, List l)
```

Chamado para desenhar um *widget* de lista.

```
void drawMonthView(Graphics g, Calendar cal, Component mv)
```

Chamado para desenhar um *widget* de visão de mês.

```
void drawRadioButton(Graphics g, RadioButton rb)
```

Chamado para desenhar um *widget* de *radio button*.

```
void drawTabbedPane(Graphics g, TabbedPane tp)
```

Chamado para desenhar um *widget* de painel de guias.

```
void drawTabbedPaneContentPane(TabbedPane tp, Graphics g, Rectangle rect, Dimension cellsPreferredSize, int numOfTabs, int selectedTabIndex, Dimension tabsSize, int cellOffsetX, int cellOffsetY)
```

Desenha e retorna o *painter* do painel de conteúdo `TabbedPane`.

```
void drawTextArea(Graphics g, TextArea ta)
```

Chamado para desenhar um *widget* de área de texto

```
void drawTextField(Graphics g, TextField ta)
```

Desenha o campo de texto sem seu cursor que é desenhado com um método diferente.

```
void drawTextFieldCursor(Graphics g, TextField ta)
```

Desenha o cursor do campo de texto, a intermitência é tratada simplesmente evitando-se uma chamada para esse método.

```
long findDayAt(int x, int y, Calendar cal, Component mv)
```

Retorna o dia do mês no `MonthView` no dado componente relativo aos deslocamentos X/Y.

```
void focusGained(Component cmp)
```

Executado quando o componente ganha foco.

```
void focusLost(Component cmp)
```

Executado quando o componente perde foco.

```
Dimension getButtonPreferredSize(Button b)
```

Retorna o tamanho preferido para o botão.

```
Dimension getCheckBoxPreferredSize(CheckBox cb)
```

Retorna o tamanho preferido para a *checkbox*.

`Dimension getComboBoxPreferredSize(ComboBox cb)`

Retorna o tamanho preferido para a *combobox*.

`Dimension getLabelPreferredSize(Label l)`

Retorna o tamanho escolhido para *label*.

`Dimension getListPreferredSize(List l)`

Retorna o tamanho preferido para lista.

`Dimension getMonthViewPreferredSize(Component mv)`

Retorna o tamanho preferido para o componente `MonthView`.

`Dimension getRadioButtonPreferredSize(RadioButton rb)`

Retorna o tamanho preferido para o *radio button*.

`Component getTabbedPaneCell(TabbedPane tp, String text, Image icon, boolean isSelected, boolean cellHasFocus, Style cellStyle, Style tabbedPaneStyle, int cellOffsetX, int cellOffsetY, Dimension cellsPreferredSize, Dimension contentPaneSize)`

Desenha e retorna o componente de célula `TabbedPane` (renderizador) de acordo com cada orientação de guia, as bordas estão obtém desenhos.

`Dimension getTextAreaPreferredSize(TextArea ta)`

Retorna o tamanho preferido para área de texto.

`Dimension getTextFieldPreferredSize(TextField ta)`

Retorna o tamanho preferido para campo de texto.

`long getTickerSpeed()`

Obtém a velocidade do *ticker*.

`void setCheckBoxImages(Image checked, Image unchecked)`

Configura imagens para os modos verificado/não-verificado da *checkbox*.

`void setComboBoxImage(Image picker)`

Configura imagem para o desenho suspenso da *combobox*.

`void setRadioButtonImages(Image selected, Image unselected)`

Configura imagens para os modos selecionado/não-selecionado do *radio button*.

`void setTickerSpeed(long tickerSpeed)`

Configura a velocidade do ticker.

`void setTickWhenFocused(boolean tickWhenFocused)`

Este método permite configurar todos `Labels`, `Buttons`, `CheckBoxes`, `RadioButtons` para iniciar verificação quando o texto é muito longo.

Métodos herdados da classe `com.sun.dtv.lwuit.plaf.`**`LookAndFeel`**

`drawHorizontalScroll, drawVerticalScroll, getDefaultDialogTransitionIn, getDefaultDialogTransitionOut, getDefaultFormTintColor, getDefaultFormTransitionIn, getDefaultFormTransitionOut, getDefaultMenuTransitionIn, getDefaultMenuTransitionOut, getDefaultSmoothScrollingSpeed, getDisableColor, getHorizontalScrollHeight, getMenuIcons, getMenuRenderer, getVerticalScrollWidth, isDefaultSmoothScrolling, isReverseSoftButtons, setDefaultDialogTransitionIn, setDefaultDialogTransitionOut, setDefaultFormTintColor, setDefaultFormTransitionIn, setDefaultFormTransitionOut, setDefaultMenuTransitionIn, setDefaultMenuTransitionOut, setDefaultSmoothScrolling,`

setDefaultSmoothScrollingSpeed, setDisableColor, setFG, setMenuIcons, setMenuRenderer, setReverseSoftButtons, uninstall

31.4.4 Detalhe dos construtores

DefaultLookAndFeel

public DefaultLookAndFeel()

Cria uma nova instância de DefaultLookAndFeel.

31.4.5 Detalhe dos métodos

bind

public void bind(Component cmp)

Cada componente se amarra ao LookAndFeel permitindo que ele customize o componente. A amarração ocorre no final do construtor quando o componente está em estado válido e pronto para ser utilizado. Observar que um componente pode ser amarrado duas vezes ou mais e é responsabilidade do LookAndFeel evitar que isso ocorra.

Substituições:

bind na classe LookAndFeel

Parâmetros:

cmp – instância de componente que pode ser customizada pelo LookAndFeel

getTickerSpeed

public long getTickerSpeed()

Obtém a velocidade do *ticker*.

Retorna:

velocidade do ticker em milissegundos

Relaciona-se com:

setTickerSpeed(long)

setTickerSpeed

public void setTickerSpeed(long tickerSpeed)

Configura a velocidade do *ticker*.

Parâmetros:

tickerSpeed - velocidade do ticker em milissegundos

Relaciona-se com:

getTickerSpeed()

setTickWhenFocused

public void setTickWhenFocused(boolean tickWhenFocused)

Este método permite configurar todos os `Labels`, `Buttons`, `CheckBoxes`, `RadioButtons` para iniciar verificação quando o texto é muito longo.

Parâmetros:

`tickWhenFocused` - boolean indicando se tais componentes devem começar a verificação caso seja necessário

setCheckBoxImages

```
public void setCheckBoxImages(Image checked,
                              Image unchecked)
```

Configura imagens para os modos verificado/não-verificado da *checkbox*.

Parâmetros:

`checked` – imagem para desenhar representando uma *checkbox* marcada

`unchecked` – imagem para desenhar representando uma *checkbox* não-marcada

setComboBoxImage

```
public void setComboBoxImage(Image picker)
```

Configura imagem para o desenho suspenso da *combobox*.

Parâmetros:

`picker` – selecionador de imagem

setRadioButtonImages

```
public void setRadioButtonImages(Image selected,
                                 Image unselected)
```

Configura imagens para os modos selecionado/não-selecionado do *radio button*.

Parâmetros:

`checked` – imagem para desenhar representando um *radio button* selecionado

`unselected` – imagem para desenhar representando um *radio button* não-selecionado

drawButton

```
public void drawButton(Graphics g,
                       Button b)
```

Chamado para desenhar um *widget* de botão.

Substituições:

`drawButton` na classe `LookAndFeel`

Parâmetros:

`g` – objetos `Graphics`

`b` – botão

drawCheckBox

```
public void drawCheckBox(Graphics g,
                         CheckBox cb)
```

Chamado para desenhar um *widget* de *checkbox*.

Substituições:

`drawCheckBox` na classe `LookAndFeel`

Parâmetros:

`g` – objetos `Graphics`

`cb` - `CheckBox`

drawLabel

```
public void drawLabel(Graphics g,
                      Label l)
```

Chamado para desenhar um *widget* de *label*.

Substituições:

`drawLabel` na classe `LookAndFeel`

Parâmetros:

`g` – objetos `Graphics`

`l` - `Label`

drawRadioButton

```
public void drawRadioButton(Graphics g,
                            RadioButton rb)
```

Chamado para desenhar um *widget* de *radio button*.

Substituições:

`drawRadioButton` na classe `LookAndFeel`

Parâmetros:

`g` – objetos `Graphics`

`rb` – `RadioButton`

drawComboBox

```
public void drawComboBox(Graphics g,
                         ComboBox cb)
```

Chamado para desenhar um *widget* de *combobox*.

Substituições:

`drawComboBox` na classe `LookAndFeel`

Parâmetros:

`g` – objetos `Graphics`

`cb` - `ComboBox`

drawList

```
public void drawList(Graphics g,
```

```
                        List l)
```

Chamado para desenhar um *widget* de lista.

Substituições:

`drawList` na classe `LookAndFeel`

Parâmetros:

`g` – objetos `Graphics`

`l` - `List`

drawMonthView

```
public void drawMonthView(Graphics g,
                          Calendar cal,
                          Component mv)
```

Chamado para desenhar um *widget* de visão de mês.

Substituições:

`drawMonthView` na classe `LookAndFeel`

Parâmetros:

`g` – objetos `Graphics`

`cal` - calendário

`mv` – componente `MonthView`

drawTextArea

```
public void drawTextArea(Graphics g,
                         TextArea ta)
```

Chamado para desenhar um *widget* de área de texto.

Substituições:

`drawTextArea` na classe `LookAndFeel`

Parâmetros:

`g` – objetos `Graphics`

`ta` – `TextArea`

getButtonPreferredSize

```
public Dimension getButtonPreferredSize(Button b)
```

Retorna o tamanho preferido para o botão.

Substituições:

`getButtonPreferredSize` na classe `LookAndFeel`

Parâmetros:

`b` – `Button`

Retorna:

tamanho preferido para o botão

getCheckBoxPreferredSize

`public Dimension getCheckBoxPreferredSize(CheckBox cb)`

Retorna o tamanho preferido para a *checkbox*.

Substituições:

`getCheckBoxPreferredSize` na classe `LookAndFeel`

Parâmetros:

`cb` - `CheckBox`

Retorna:

tamanho preferido para a *checkbox*

getLabelPreferredSize

`public Dimension getLabelPreferredSize(Label l)`

Retorna o tamanho preferido para o *label*.

Substituições:

`getLabelPreferredSize` na classe `LookAndFeel`

Parâmetros:

`l` - `Label`

Retorna:

tamanho preferido para o *label*

getListPreferredSize

`public Dimension getListPreferredSize(List l)`

Retorna o tamanho preferido para a lista.

Substituições:

`getListPreferredSize` na classe `LookAndFeel`

Parâmetros:

`l` - `List`

Retorna:

tamanho preferido para a lista

getMonthViewPreferredSize

`public Dimension getMonthViewPreferredSize(Component mv)`

Retorna o tamanho preferido para o componente `MonthView`.

Substituições:

`getMonthViewPreferredSize` na classe `LookAndFeel`

Parâmetros:

`mv` – componente `MonthView`

Retorna:

Retorna o tamanho preferido para o componente `MonthView`.

getRadioButtonPreferredSize

```
public Dimension getRadioButtonPreferredSize(RadioButton rb)
```

Retorna o tamanho preferido para o *radio button*.

Substituições:

`getRadioButtonPreferredSize` na classe `LookAndFeel`

Parâmetros:

`rb` – `RadioButton`

Retorna:

tamanho preferido para o *radio button*

getTextAreaPreferredSize

```
public Dimension getTextAreaPreferredSize(TextArea ta)
```

Retorna o tamanho preferido para a área de texto.

Substituições:

`getTextAreaPreferredSize` na classe `LookAndFeel`

Parâmetros:

`ta` – `TextArea`

Retorna:

tamanho preferido para a área de texto.

findDayAt

```
public long findDayAt(int x,
                      int y,
                      Calendar cal,
                      Component mv)
```

Retorna o dia do mês no `MonthView` no dado componente relativo dos deslocamentos X/Y. Isso é importante para interação em touch screen

Substituições:

`findDayAt` na classe `LookAndFeel`

Parâmetros:

`x` – coordenada x de deslocamento

`y` - coordenada y de deslocamento

`cal` - calendário

`mv` – componente `MonthView`

Retorna:

dia do mês

getComboBoxPreferredSize

```
public Dimension getComboBoxPreferredSize(ComboBox cb)
```

Retorna o tamanho preferido para a *combobox*.

Substituições:

`getComboBoxPreferredSize` na classe `LookAndFeel`

Retorna:

tamanho preferido para a *combobox*

drawTabbedPane

```
public void drawTabbedPane(Graphics g,
                           TabbedPane tp)
```

Chamado para desenhar um *widget* de painel de guias.

Substituições:

`drawTabbedPane` na classe `LookAndFeel`

Parâmetros:

`g` – objetos `Graphics`

`tp` - `TabbedPane`

getTabbedPaneCell

```
public Component getTabbedPaneCell(TabbedPane tp,
                                   String text,
                                   Image icon,
                                   boolean isSelected,
                                   boolean cellHasFocus,
                                   Style cellStyle,
                                   Style tabbedPaneStyle,
                                   int cellOffsetX,
                                   int cellOffsetY,
                                   Dimension cellsPreferredSize,
                                   Dimension contentPaneSize)
```

Desenha e retorna o componente de célula `TabbedPane` (renderizador) de acordo com cada orientação de guia, as bordas estão obtém desenhos.

Substituições:

`getTabbedPaneCell` na classe `LookAndFeel`

Parâmetros:

`tp` - `TabbedPane` (painel de guias)

`text` – texto da célula

`icon` – imagem do ícone da célula

`isSelected` – célula que está selecionada

`cellHasFocus` – célula em foco

`cellStyle` – objeto estilo da célula

`tabbedPaneStyle` – objeto estilo do painel de guias

`cellOffsetX` - deslocamento quando a célula está ACIMA ou ABAIXO

`cellOffsetY` - deslocamento quando a célula está à ESQUERDA ou DIRETA

`cellsPreferredSize` – tamanho preferível total da célula

`contentPaneSize` - tamanho do painel de conteúdo

Retorna:

componente da célula do painel de guias

drawTabbedPaneContentPane

```
public void drawTabbedPaneContentPane(TabbedPane tp,
                                      Graphics g,
                                      Rectangle rect,
                                      Dimension cellsPreferredSize,
                                      int numOfTabs,
                                      int selectedTabIndex,
                                      Dimension tabsSize,
                                      int cellOffsetX,
                                      int cellOffsetY)
```

Desenha e retorna o *painter* do painel de conteúdo `TabbedPane`.

Substituições:

`drawTabbedPaneContentPane` na classe `LookAndFeel`

Parâmetros:

`tp` - `TabbedPane` (painel de guias)

`g` – objeto `Graphics` do painel de conteúdo

`rect` – área retangular de desenho do painel de conteúdo

`cellsPreferredSize` – tamanho preferível total da célula

`numOfTabs` – número de guias

`selectedTabIndex` – índice da guia selecionada

`tabsSize` – tamanho das guias

`cellOffsetX` - deslocamento quando a célula está ACIMA ou ABAIXO

`cellOffsetY` - deslocamento quando a célula está à ESQUERDA ou DIRETA

drawTextField

```
public void drawTextField(Graphics g,
                          TextField ta)
```

Desenha o campo de texto sem seu cursor que é desenhado com um método diferente. A indicação de modo de input também pode ser desenhada utilizando esse método.

Substituições:

`drawTextField` na classe `LookAndFeel`

Parâmetros:

`g` – objeto `Graphics`

getTextFieldPreferredSize

```
public Dimension getTextFieldPreferredSize(TextField ta)
```

Retorna o tamanho preferível para o campo de texto.

Substituições:

`getTextFieldPreferredSize` na classe `LookAndFeel`

Retorna:

tamanho preferível para o campo de texto.

drawTextFieldCursor

```
public void drawTextFieldCursor(Graphics g,
                                TextField ta)
```

Desenha o cursor do campo de texto, a intermitência é tratada simplesmente evitando-se uma chamada para esse método.

Substituições:

`drawTextFieldCursor` na classe `LookAndFeel`

Parâmetros:

`g` – objetos `Graphics`.

focusGained

```
public void focusGained(Component cmp)
```

Executado quando o componente ganha foco.

Especificado por:

`focusGained` na interface `FocusListener`

Parâmetros:

`cmp` – o componente que ganha foco.

focusLost

```
public void focusLost(Component cmp)
```

Executado quando o componente perde foco.

Especificado por:

`focusLost` na interface `FocusListener`

Parâmetros:

`cmp` – o componente que perde foco.

31.5 Classe LookAndFeel

31.5.1 Descrição da classe

com.sun.dtv.lwuit.plaf

java.lang.Object

└com.sun.dtv.lwuit.plaf.LookAndFeel

Subclasses diretas conhecidas

DefaultLookAndFeel

abstract public class LookAndFeel

extends Object

Permite que um desenvolvedor da interface do usuário personalize completamente a aparência do aplicativo por cancelar métodos de desenho/dimencionamento apropriadamente.

31.5.2 Índice de construtores

LookAndFeel()

31.5.3 Índice de métodos

void **bind**(Component cmp)

Cada componente se amarra ao LookAndFeel permitindo que ele customize o componente.

abstract void **drawButton**(Graphics g, Button b)

Chamado para desenhar um *widget* de botão.

abstract void **drawCheckBox**(Graphics g, CheckBox cb)

Chamado para desenhar um *widget* de *checkbox*.

abstract void **drawComboBox**(Graphics g, ComboBox cb)

Chamado para desenhar um *widget* de *combobox*.

void **drawHorizontalScroll**(Graphics g, Component c, float offsetRatio, float blockSizeRatio)

Desenha uma barra de rolagem horizontal no componente determinado.

abstract void **drawLabel**(Graphics g, Label l)

Chamado para desenhar um *widget* de *label*.

abstract void **drawList**(Graphics g, List l)

Chamado para desenhar um *widget* de lista.

abstract void **drawMonthView**(Graphics g, Calendar cal, Component mv)

Chamado para desenhar um *widget* de visão de mês.

abstract void **drawRadioButton**(Graphics g, RadioButton rb)

Chamado para desenhar um *widget* de *radio button*.

abstract void **drawTabbedPane**(Graphics g, TabbedPane tp)

Chamado para desenhar um *widget* de paninel de guias.

`abstract void` **`drawTabbedPaneContentPane`**`(TabbedPane tp, Graphics g, Rectangle rect, Dimension cellsPreferredSize, int numOfTabs, int selectedTabIndex, Dimension tabsSize, int cellOffsetX, int cellOffsetY)`

Desenha e retorna o *painter* do painel de conteúdo `TabbedPane`.

`abstract void` **`drawTextArea`**`(Graphics g, TextArea ta)`

Chamado para desenhar um *widget* de área de texto.

`abstract void` **`drawTextField`**`(Graphics g, TextField tf)`

Desenha o campo de texto sem seu cursor que é desenhado com um método diferente.

`abstract void` **`drawTextFieldCursor`**`(Graphics g, TextField tf)`

Desenha o cursor do campo de texto, a intermitência é tratada simplesmente evitando-se uma chamada para esse método.

`void` **`drawVerticalScroll`**`(Graphics g, Component c, float offsetRatio, float blockSizeRatio)`

Desenha uma barra de rolagem vertical no componente determinado.

`abstract long` **`findDayAt`**`(int x, int y, Calendar cal, Component mv)`

Retorna o dia do mês no `MonthView` no dado componente relativo dos deslocamentos X/Y.

`abstract Dimension` **`getButtonPreferredSize`**`(Button b)`

Retorna o tamanho preferido para o botão.

`abstract Dimension` **`getCheckBoxPreferredSize`**`(CheckBox cb)`

Retorna o tamanho preferido para a *checkbox*.

`abstract Dimension` **`getComboBoxPreferredSize`**`(ComboBox box)`

Retorna o tamanho preferido para a *combobox*.

`Transition` **`getDefaultDialogTransitionIn`**`()`

Recupera animação-padrão que desenhará a transição para entrar em um diálogo.

`Transition` **`getDefaultDialogTransitionOut`**`()`

Recupera animação-padrão que desenhará a transição para sair de um diálogo.

`Color` **`getDefaultFormTintColor`**`()`

A tonalidade é configurada quando um formulário é parcialmente coberto, seja por um menu ou por um diálogo.

`Transition` **`getDefaultFormTransitionIn`**`()`

Recupera animação-padrão que desenhará a transição para entrar em um formulário.

`Transition` **`getDefaultFormTransitionOut`**`()`

Recupera animação-padrão que desenhará a transição para entrar em um formulário.

`Transition` **`getDefaultMenuTransitionIn`**`()`

Recupera animação-padrão que desenhará a transição para entrar em um menu.

`Transition` **`getDefaultMenuTransitionOut`**`()`

Recupera animação-padrão que desenhará a transição para sair de um menu.

`int` **`getDefaultSmoothScrollingSpeed`**`()`

Indica a velocidade-padrão para rolamento suave.

`Color` **`getDisableColor`**`()`

Essa cor é utilizada para colorir o texto no modo desabilitar.

`int` **`getHorizontalScrollHeight`**`()`

Retorna a altura-padrão de uma barra de rolagem horizontal.

`abstract Dimension` **`getLabelPreferredSize`**`(Label l)`

Retorna o tamanho preferido para o *label*.

`abstract Dimension` **`getListPreferredSize`**`(List l)`

Retorna o tamanho preferido para a lista.

`Image[]` **`getMenuIcons`**`()`

Getter simples para ícones do menu.

`ListCellRenderer` **`getMenuRenderer`**`()`

Retorna o renderizador-padrão do menu.

`abstract Dimension` **`getMonthViewPreferredSize`**`(Component mv)`

Retorna o tamanho preferido para o componente `MonthView`.

`abstract Dimension` **`getRadioButtonPreferredSize`**`(RadioButton rb)`

Retorna o tamanho preferido para o *radio button*.

`abstract Component` **`getTabbedPaneCell`**`(TabbedPane tp, String text, Image icon, boolean isSelected, boolean cellHasFocus, Style cellStyle, Style tabbedPaneStyle, int cellOffsetX, int cellOffsetY, Dimension cellsPreferredSize, Dimension contentPaneSize)`

Desenha e retorna o componente de célula `TabbedPane` (renderizador) de acordo com cada orientação de guia, as bordas estão obtém desenhos.

`abstract Dimension` **`getTextAreaPreferredSize`**`(TextArea ta)`

Retorna o tamanho preferido para a área de texto.

`abstract Dimension` **`getTextFieldPreferredSize`**`(TextField tf)`

Retorna o tamanho preferido para o campo de texto.

`int` **`getVerticalScrollWidth`**`()`

Retorna a largura-padrão de uma barra de rolagem vertical.

`boolean` **`isDefaultSmoothScrolling`**`()`

Indica se as listas e containers devem apresentar rolagem suave como padrão.

`boolean` **`isReverseSoftButtons`**`()`

Indica se os softbuttons devem ter sua orientação-padrão revertida.

`void` **`setDefaultDialogTransitionIn`**`(Transition defaultDialogTransitionIn)`

Permite definir uma animação-padrão que desenhará a transição para entrar em um diálogo.

`void` **`setDefaultDialogTransitionOut`**`(Transition defaultDialogTransitionOut)`

Permite definir uma animação-padrão que desenhará a transição para sair de um diálogo.

`void` **`setDefaultFormTintColor`**`(Color defaultFormTintColor)`

A tonalidade é configurada quando um formulário é parcialmente coberto, seja por um menu ou por um diálogo.

`void` **`setDefaultFormTransitionIn`**`(Transition defaultFormTransitionIn)`

Permite definir uma animação-padrão que desenhará a transição para entrar em um formulário.

`void` **`setDefaultFormTransitionOut`**`(Transition defaultFormTransitionOut)`

Permite definir uma animação-padrão que desenhará a transição para sair de um formulário.

`void` **`setDefaultMenuTransitionIn`**`(Transition defaultMenuTransitionIn)`

Permite definir uma animação-padrão que desenhará a transição para entrar em um menu.

`void` **`setDefaultMenuTransitionOut`**`(Transition defaultMenuTransitionOut)`

Permite definir uma animação-padrão que desenhará a transição para sair de um menu.

`void` **`setDefaultSmoothScrolling`**`(boolean defaultSmoothScrolling)`

Indica se as listas e containers devem apresentar rolagem suave como padrão.

`void` **`setDefaultSmoothScrollingSpeed`**`(int defaultSmoothScrollingSpeed)`

Indica a velocidade-padrão para rolamento suave.

`void` **`setDisableColor`**`(Color disableColor)`

Setter simples para desabilitar a cor.

`void` **`setFG`**`(Graphics g, Component c)`

Configura a cor do primeiro plano e fonte para um componente genérico, reutilizável pela maioria dos códigos de desenho de componente.

`void` **`setMenuIcons`**`(Image select, Image cancel, Image menu)`

Configura globalmente os ícones do menu.

`void` **`setMenuRenderer`**`(ListCellRenderer menuRenderer)`

Retorna o renderizador-padrão do menu.

`void` **`setReverseSoftButtons`**`(boolean reverseSoftButtons)`

Indica se os softbuttons devem ter sua orientação-padrão revertida.

`void` **`uninstall`**`()`

Chamado quando um `LookAndFeel` é removido, pemite que um `LookAndFeel` libere recursos relacionados à amarração de componentes.

31.5.4 Detalhe dos construtores

LookAndFeel

```
public LookAndFeel()
```

31.5.5 Detalhe dos métodos

bind

```
public void bind(Component cmp)
```

Cada componente se amarra ao `LookAndFeel` permitindo que ele customize o componente. A amarração ocorre no final do construtor quando o componente está em estado válido e pronto para ser utilizado. Observar que um componente pode ser amarrado duas vezes ou mais e é responsabilidade do `LookAndFeel` evitar que isso ocorra.

Parâmetros:

`cmp` – instância de componente que pode ser customizada pelo `LookAndFeel`

uninstall

```
public void uninstall()
```

Chamado quando um `LookAndFeel` é removido, pemite que um `LookAndFeel` libere recursos relacionados à amarração de componentes.

Relaciona-se com:

`bind(Component)`

drawButton

```
public abstract void drawButton(Graphics g,
                                Button b)
```

Chamado para desenhar um *widget* de botão.

Parâmetros:

`g` – objetos `Graphics`

`b` – `Button`

drawCheckBox

```
public abstract void drawCheckBox(Graphics g,
                                  CheckBox cb)
```

Chamado para desenhar um *widget* de *checkbox*.

Parâmetros:

`g` – objetos `Graphics`

`cb` - `CheckBox`

drawComboBox

```
public abstract void drawComboBox(Graphics g,
                                  ComboBox cb)
```

Chamado para desenhar um *widget* de *combobox*.

Parâmetros:

`g` – objetos `Graphics`

`cb` - `ComboBox`

drawLabel

```
public abstract void drawLabel(Graphics g,
                               Label l)
```

Chamado para desenhar um *widget* de *label*.

Parâmetros:

`g` – objetos `Graphics`

`l` - `Label`

drawList

```
public abstract void drawList(Graphics g,
                              List l)
```

Chamado para desenhar um *widget* de lista.

Parâmetros:

`g` – objetos `Graphics`

`l` - `List`

drawMonthView

```
public abstract void drawMonthView(Graphics g,
                                   Calendar cal,
                                   Component mv)
```

Chamado para desenhar um *widget* de visão de mês.

Parâmetros:

`g` – objetos `Graphics`

`cal` - calendário

`mv` – componente `MonthView`

findDayAt

```
public abstract long findDayAt(int x,
                               int y,
                               Calendar cal,
                               Component mv)
```

Retorna o dia do mês no `MonthView` no dado componente relativo dos deslocamentos X/Y. Isso é importante para interação em touch screen

Parâmetros:

`x` – coordenada x de deslocamento

`y` - coordenada y de deslocamento

`cal` - calendário

`mv` – componente `MonthView`

Retorna:

dia do mês

drawRadioButton

```
public abstract void drawRadioButton(Graphics g,
                                     RadioButton rb)
```

Chamado para desenhar um *widget* de *radio button*.

Parâmetros:

`g` – objetos `Graphics`

rb – RadioButton

drawTextArea

```
public abstract void drawTextArea(Graphics g,
                                  TextArea ta)
```

Chamado para desenhar um *widget* de área de texto.

Parâmetros:

g – objetos Graphics

ta – TextArea

drawTextField

```
public abstract void drawTextField(Graphics g,
                                   TextField tf)
```

Desenha o campo de texto sem seu cursor que é desenhado com um método diferente. A indicação de modo de input também pode ser desenhada utilizando esse método.

Parâmetros:

g – objetos Graphics

tf – TextField

drawTextFieldCursor

```
public abstract void drawTextFieldCursor(Graphics g,
                                         TextField tf)
```

Desenha o cursor do campo de texto, a intermitência é tratada simplesmente evitando-se uma chamada para esse método.

Parâmetros:

g – objetos Graphics

tf – TextField

drawTabbedPane

```
public abstract void drawTabbedPane(Graphics g,
                                    TabbedPane tp)
```

Chamado para desenhar um *widget* de painel de guias.

Parâmetros:

g – objetos Graphics

tp - TabbedPane

getButtonPreferredSize

```
public abstract Dimension getButtonPreferredSize(Button b)
```

Retorna o tamanho preferido para o botão.

Parâmetros:

b – Button

Retorna:

tamanho preferido para o botão

getCheckBoxPreferredSize

```
public abstract Dimension getCheckBoxPreferredSize(CheckBox cb)
```

Retorna o tamanho preferido para a *checkbox*.

Parâmetros:

`cb` - `CheckBox`

Retorna:

tamanho preferido para a *checkbox*

getLabelPreferredSize

```
public abstract Dimension getLabelPreferredSize(Label l)
```

Retorna o tamanho escolhido para o *label*.

Parâmetros:

`l` - `Label`

Retorna:

tamanho preferido para o *label*

getListPreferredSize

```
public abstract Dimension getListPreferredSize(List l)
```

Retorna o tamanho preferido para a lista.

Parâmetros:

`l` - `List`

Retorna:

tamanho preferido para a lista

getMonthViewPreferredSize

```
public abstract Dimension getMonthViewPreferredSize(Component mv)
```

Retorna o tamanho preferido para o componente `MonthView`.

Parâmetros:

`mv` – componente `MonthView`

Retorna:

Retorna o tamanho preferido para o componente `MonthView`.

getRadioButtonPreferredSize

```
public abstract Dimension getRadioButtonPreferredSize(RadioButton rb)
```

Retorna o tamanho preferido para o *radio button*.

Parâmetros:

`rb` – `RadioButton`

Retorna:

tamanho preferido para o *radio button*

getTextAreaPreferredSize

```
public abstract Dimension getTextAreaPreferredSize(TextArea ta)
```

Retorna o tamanho preferido para a área de texto.

Parâmetros:

`ta` – `TextArea`

Retorna:

tamanho preferido para a área de texto

getTextFieldPreferredSize

```
public abstract Dimension getTextFieldPreferredSize(TextField tf)
```

Retorna o tamanho preferido para o campo de texto.

Parâmetros:

`tf` – `TextField`

Retorna:

tamanho preferido para o campo de texto

getComboBoxPreferredSize

```
public abstract Dimension getComboBoxPreferredSize(ComboBox box)
```

Retorna o tamanho preferido para a *combobox*.

Parâmetros:

`cb` - `ComboBox`

Retorna:

tamanho preferido para a *combobox*

drawVerticalScroll

```
public void drawVerticalScroll(Graphics g,
                               Component c,
                               float offsetRatio,
                               float blockSizeRatio)
```

Desenha uma barra de rolagem vertical no componente determinado.

Parâmetros:

`g` – objetos `Graphics`

`c` - componente

`offsetRatio` – razão de deslocamento

`blockSizeRatio` – razão de tamanho do bloco

drawHorizontalScroll

```
public void drawHorizontalScroll(Graphics g,
                                 Component c,
                                 float offsetRatio,
                                 float blockSizeRatio)
```

Desenha uma barra de rolagem horizontal no componente determinado.

Parâmetros:

`g` – objetos `Graphics`

`c` - componente

`offsetRatio` – razão de deslocamento

`blockSizeRatio` – razão de tamanho do bloco

setFG

```
public void setFG(Graphics g,
                  Component c)
```

Configura a cor do primeiro plano e fonte para um componente genérico, reutilizável pela maioria dos códigos de desenho de componente.

Parâmetros:

`g` – objetos `Graphics`

`c` - componente

getVerticalScrollWidth

```
public int getVerticalScrollWidth()
```

Retorna a largura-padrão de uma barra de rolagem vertical.

Retorna:

largura-padrão de uma barra de rolagem vertical

getHorizontalScrollHeight

```
public int getHorizontalScrollHeight()
```

Retorna a altura-padrão de uma barra de rolagem horizontal.

Retorna:

altura-padrão de uma barra de rolagem horizontal

getTabbedPaneCell

```
public abstract Component getTabbedPaneCell(TabbedPane tp,
                                            String text,
                                            Image icon,
                                            boolean isSelected,
                                            boolean cellHasFocus,
```

```
                                                Style cellStyle,
                                                Style tabbedPaneStyle,
                                                int cellOffsetX,
                                                int cellOffsetY,
                                                Dimension cellsPreferredSize,
                                                Dimension contentPaneSize)
```

Desenha e retorna o componente de célula `TabbedPane` (renderizador) de acordo com cada orientação de guia, as bordas estão obtém desenhos.

Parâmetros:

`tp` - `TabbedPane`

`text` – texto da célula

`icon` – imagem do ícone da célula

`isSelected` – célula que está selecionada

`cellHasFocus` – célula em foco

`cellStyle` – objeto `Style` da célula

`tabbedPaneStyle` – objeto `Style` do `TabbedPane`

`cellOffsetX` - deslocamento quando a célula está ACIMA ou ABAIXO

`cellOffsetY` - deslocamento quando a célula está à ESQUERDA ou DIRETA

`cellsPreferredSize` – `PreferredSize` total da célula

`contentPaneSize` - `contentPaneSize`

Retorna:

componente da célula do `TabbedPane`

drawTabbedPaneContentPane

```
public abstract void drawTabbedPaneContentPane(TabbedPane tp,
                                               Graphics g,
                                               Rectangle rect,
                                               Dimension cellsPreferredSize,
                                               int numOfTabs,
                                               int selectedTabIndex,
                                               Dimension tabsSize,
                                               int cellOffsetX,
                                               int cellOffsetY)
```

Desenha e retorna o painter do painel de conteúdo `TabbedPane`.

Parâmetros:

`tp` - `TabbedPane`

`g` – gráficos do painel de conteúdo

`rect` – área retangular de desenho do painel de conteúdo

`cellsPreferredSize` – `PreferredSize` total da célula

`numOfTabs` – número de guias

`selectedTabIndex` – índice da guia selecionada

`tabsSize` – tamanho das guias

`cellOffsetX` - deslocamento quando a célula está ACIMA ou ABAIXO

`cellOffsetY` - deslocamento quando a célula está à ESQUERDA ou DIRETA

getDefaultFormTransitionIn

`public Transition getDefaultFormTransitionIn()`

Recupera animação-padrão que desenhará a transição para entrar em um formulário.

Retorna:

objeto `Transition`

Relaciona-se com:

`setDefaultFormTransitionIn(com.sun.dtv.lwuit.animations.Transition)`

setDefaultFormTransitionIn

`public void setDefaultFormTransitionIn(Transition defaultFormTransitionIn)`

Permite definir uma animação-padrão que desenhará a transição para entrar em um formulário.

Parâmetros:

`defaultFormTransitionIn` – animação-padrão

Relaciona-se com:

`getDefaultFormTransitionIn()`

getDefaultFormTransitionOut

`public Transition getDefaultFormTransitionOut()`

Recupera animação-padrão que desenhará a transição para entrar em um formulário.

Retorna:

objeto `Transition`

Relaciona-se com:

`setDefaultFormTransitionOut(com.sun.dtv.lwuit.animations.Transition)`

setDefaultFormTransitionOut

`public void setDefaultFormTransitionOut(Transition defaultFormTransitionOut)`

Permite definir uma animação-padrão que desenhará a transição para sair de um formulário.

Parâmetros:

`defaultFormTransitionOut` – animação-padrão

Relaciona-se com:

`getDefaultFormTransitionOut()`

getDefaultMenuTransitionIn

`public Transition getDefaultMenuTransitionIn()`

Recupera animação-padrão que desenhará a transição para entrar em um Menu.

Retorna:

objeto `Transition`

Relaciona-se com:

`setDefaultMenuTransitionIn(com.sun.dtv.lwuit.animations.Transition)`

setDefaultMenuTransitionIn

`public void setDefaultMenuTransitionIn(Transition defaultMenuTransitionIn)`

Permite definir uma animação-padrão que desenhará a transição para entrar em um Menu.

Parâmetros:

`defaultMenuTransitionIn` – animação-padrão

Relaciona-se com:

`getDefaultMenuTransitionIn()`

getDefaultMenuTransitionOut

`public Transition getDefaultMenuTransitionOut()`

Recupera animação-padrão que desenhará a transição para sair de um Menu.

Retorna:

objeto `Transition`

Relaciona-se com:

`setDefaultMenuTransitionOut(com.sun.dtv.lwuit.animations.Transition)`

setDefaultMenuTransitionOut

`public void setDefaultMenuTransitionOut(Transition defaultMenuTransitionOut)`

Permite definir uma animação-padrão que desenhará a transição para sair de um Menu.

Parâmetros:

`defaultMenuTransitionOut` – animação-padrão

Relaciona-se com:

`getDefaultMenuTransitionOut()`

getDefaultDialogTransitionIn

`public Transition getDefaultDialogTransitionIn()`

Recupera animação-padrão que desenhará a transição para entrar em um diálogo.

Retorna:

objeto `Transition`

Relaciona-se com:

`setDefaultDialogTransitionIn(com.sun.dtv.lwuit.animations.Transition)`

setDefaultDialogTransitionIn

`public void setDefaultDialogTransitionIn(Transition defaultDialogTransitionIn)`

Permite definir uma animação-padrão que desenhará a transição para entrar em um diálogo.

Parâmetros:

`defaultDialogTransitionIn` – animação-padrão

Relaciona-se com:

`getDefaultDialogTransitionIn()`

getDefaultDialogTransitionOut

`public Transition getDefaultDialogTransitionOut()`

Recupera animação-padrão que desenhará a transição para sair de um diálogo.

Retorna:

objeto `Transition`

Relaciona-se com:

`setDefaultDialogTransitionOut(com.sun.dtv.lwuit.animations.Transition)`

setDefaultDialogTransitionOut

`public void setDefaultDialogTransitionOut(Transition defaultDialogTransitionOut)`

Permite definir uma animação-padrão que desenhará a transição para sair de um diálogo.

Parâmetros:

`defaultDialogTransitionOut` – animação-padrão

Relaciona-se com:

`getDefaultDialogTransitionOut()`

getDefaultFormTintColor

`public Color getDefaultFormTintColor()`

A tonalidade é configurada quando um formulário é parcialmente coberto, seja por um menu ou por um diálogo. Um `LookAndFeel` pode substituir esse valor-padrão.

Retorna:

a tonalidade de cor

Relaciona-se com:

`setDefaultFormTintColor(java.awt.Color)`

setDefaultFormTintColor

`public void setDefaultFormTintColor(Color defaultFormTintColor)`

A tonalidade é configurada quando um formulário é parcialmente coberto, seja por um menu ou por um diálogo. Um `LookAndFeel` pode substituir esse valor-padrão.

Parâmetros:

`defaultFormTintColor` - a tonalidade de cor

Relaciona-se com:

`getDefaultFormTintColor()`

getDisableColor

`public Color getDisableColor()`

Essa cor é utilizada para colorir o texto no modo desabilitar.

Retorna:

valor da cor

Relaciona-se com:

`setDisableColor(java.awt.Color)`

setDisableColor

`public void setDisableColor(Color disableColor)`

Setter simples para desabilitar a cor.

Parâmetros:

`disableColor` – valor de desabilitar cor

Relaciona-se com:

`getDisableColor()`

isDefaultSmoothScrolling

`public boolean isDefaultSmoothScrolling()`

Indica se as listas e containers devem apresentar rolagem suave como padrão.

Retorna:

true se a rolagem suave estiver ativada

Relaciona-se com:

`setDefaultSmoothScrolling(boolean)`

setDefaultSmoothScrolling

`public void setDefaultSmoothScrolling(boolean defaultSmoothScrolling)`

Indica se as listas e containers devem apresentar rolagem suave como padrão.

Parâmetros:

`defaultSmoothScrolling` - indica se as listas e containers devem apresentar rolagem suave como padrão.

Relaciona-se com:

`isDefaultSmoothScrolling()`

getDefaultSmoothScrollingSpeed

`public int getDefaultSmoothScrollingSpeed()`

Indica a velocidade-padrão para rolamento suave.

Retorna:

velocidade-padrão

Relaciona-se com:

`setDefaultSmoothScrollingSpeed(int)`

setDefaultSmoothScrollingSpeed

`public void setDefaultSmoothScrollingSpeed(int defaultSmoothScrollingSpeed)`

Indica a velocidade-padrão para rolamento suave.

Parâmetros:

`defaultSmoothScrollingSpeed` - velocidade-padrão

Relaciona-se com:

`getDefaultSmoothScrollingSpeed()`

isReverseSoftButtons

`public boolean isReverseSoftButtons()`

Indica se os `softbuttons` devem ter sua orientação-padrão revertida.

Retorna:

true se o `softbutton` reverso estiver ativo

Relaciona-se com:

`setReverseSoftButtons(boolean)`

setReverseSoftButtons

`public void setReverseSoftButtons(boolean reverseSoftButtons)`

Indica se os `softbuttons` devem ter sua orientação-padrão revertida.

Parâmetros:

`reverseSoftButtons` - indica se os softbuttons devem ter sua orientação-padrão revertida.

Relaciona-se com:

`isReverseSoftButtons()`

getMenuRenderer

`public ListCellRenderer getMenuRenderer()`

Retorna o renderizador-padrão do menu.

Retorna:

renderizador-padrão do Menu

Relaciona-se com:

`setMenuRenderer(com.sun.dtv.lwuit.list.ListCellRenderer)`

setMenuRenderer

`public void setMenuRenderer(ListCellRenderer menuRenderer)`

Retorna o renderizador-padrão do menu.

Parâmetros:

`menuRenderer` - renderizador-padrão do menu.

Relaciona-se com:

`getMenuRenderer()`

setMenuIcons

```
public void setMenuIcons(Image select,
                         Image cancel,
                         Image menu)
```

Configura globalmente os ícones do menu.

Parâmetros:

`select` – ícone de seleção

`cancel` – ícone de cancelamento

`menu` – ícone de menu

Relaciona-se com:

`getMenuIcons()`

getMenuIcons

`public Image[] getMenuIcons()`

`Getter` simples para ícones do menu.

Retorna:

um *array* de imagem do tamanho de 3, onde o primeiro é selecionar imagem, o segundo é cancelar imagem e o último é imagem do menu.

Relaciona-se com:

`setMenuIcons(com.sun.dtv.lwuit.Image, com.sun.dtv.lwuit.Image, com.sun.dtv.lwuit.Image)`

31.6 Classe Style

31.6.1 Descrição da classe

`com.sun.dtv.lwuit.plaf`

```
java.lang.Object
  └com.sun.dtv.lwuit.plaf.Style
```

```
public class Style
extends Object
```

Representa a aparência do componente determinado: cores, fontes, transparência, margem e *padding* & imagens.

Cada componente contém um objeto `Style` que permite modificação de estilo no *runtime* utilizando `cmp.getStyle()`. O estilo também é utilizado em temas, quando um tema é alterado, os objetos *style* são atualizados automaticamente.

Ao alterar um tema, os elementos alterados manualmente em um estilo não podem ser atualizados pela alteração do tema como padrão. Há duas maneiras de mudar esse comportamento:

Utilize o método estabelecido que aceita um segundo elemento boolean e o configura para *true*.

Cria um novo objeto de estilo e passa todas as opções no construtor (sem chamar os `setters` manualmente). `Margin` e `Padding` são inspirado pelo *W3 Box Model* (http://www.w3.org/TR/CSS2/box.html).

```
*************************
*         Margin        *
*  ********************  *
*  *      Padding     *  *
*  *    ***********   *  *
*  *    * Content *   *  *
*  *    ***********   *  *
*  *      Padding     *  *
*  ********************  *
*         Margin        *
*************************
```

31.6.2 Índice de campos

`static String` **`BG_COLOR`**

Nome do atributo cor de plano de fundo para a *hashtable* de tema.

`static String` **`BG_IMAGE`**

Nome do atributo imagem de plano de fundo para a *hashtable* de tema.

`static String` **`BG_SELECTION_COLOR`**

Nome do atributo seleção de cor de plano de fundo para a *hashtable* de tema.

`static String` **`BORDER`**

Nome do atributo borda para a *hashtable* de tema.

`static String` **`FG_COLOR`**

Nome do atributo cor de primeiro plano para a *hashtable* de tema.

`static String` **`FG_SELECTION_COLOR`**

Nome do atributo cor de primeiro plano para a *hashtable* de tema.

`static String` **`FONT`**

Nome do atributo fonte para a *hashtable* de tema.

`static String` **`MARGIN`**

Nome do atributo margem para a *hashtable* de tema.

`static String` **`PADDING`**

Nome do atributo Padding para a *hashtable* de tema.

`static String` **`SCALED_IMAGE`**

Nome do atributo imagem redimensionada para a *hashtable* de tema.

`static String` **`TRANSPARENCY`**

Nome do atributo transparência para a *hashtable* de tema.

31.6.3 Índice de construtores

`Style``()`

Cada componente quando se auto-desenha utiliza esse Objeto para determinar em quais cores deve utilizar.

`Style(Style style)`

Cria uma cópia completa de um determinado estilo.

`Style(Color fgColor, Color bgColor, Color fgSelectionColor, Color bgSelectionColor, Font f, byte transparency)`

Cria um novo estilo com os atributos determinados.

`Style(Color fgColor, Color bgColor, Color fgSelectionColor, Color bgSelectionColor, Font f, byte transparency, Image im, boolean scaledImage)`

Cria um novo estilo com os atributos determinados.

31.6.4 Índice de métodos

`void addStyleListener(StyleListener l)`

Adiciona um `Listener` de `Style` ao objeto `Style`,

`Color getBgColor()`

Retorna a cor do plano de fundo para o componente.

`Image getBgImage()`

Retorna a imagem do plano de fundo para o componente.

`Painter getBgPainter()`

Retorna o painter do plano de fundo para esse estilo, normalmente deve ser o painter de imagem/cor interno, mas pode ser determinado pelo usuário.

`Color getBgSelectionColor()`

Retorna a cor de seleção do plano de fundo para o componente.

`byte getBgTransparency()`

Retorna o nível de transparência do componente.

`Border getBorder()`

Retorna a borda para o estilo.

`Color getFgColor()`

Retorna a cor do primeiro plano para o componente.

`Color getFgSelectionColor()`

Retorna a cor de seleção do primeiro plano para o componente.

`Font getFont()`

Retorna a fonte para o componente.

`int getMargin(int orientation)`

Retorna o Margin.

`int getPadding(int orientation)`

Retorna o Padding.

`boolean isModified()`

Retorna *true* se o estilo foi modificado manualmente após ter sido criado pelo `LookAndFeel`.

`boolean isScaleImage()`

Indica se a imagem no plano de fundo foi redimensionada.

`void` **`merge`**`(Style style)`

Combina o novo estilo com o estilo atual sem alterar os elementos que foram modificados.

`void` **`removeStyleListener`**`(StyleListener l)`

Remove um `Listener` de `Style` ao objeto `Style`.

`void` **`setBgColor`**`(Color bgColor)`

Configura a cor do plano de fundo para o componente.

`void` **`setBgColor`**`(Color bgColor, boolean override)`

Configura a cor do plano de fundo para o componente.

`void` **`setBgImage`**`(Image bgImage)`

Configura a imagem do plano de fundo para o componente.

`void` **`setBgImage`**`(Image bgImage, boolean override)`

Configura a imagem do plano de fundo para o componente.

`void` **`setBgPainter`**`(Painter bgPainter)`

Define o painter do plano de fundo para esse estilo, normalmente deve ser o painter de imagem/cor interno, mas pode ser determinado pelo usuário.

`void` **`setBgSelectionColor`**`(Color bgSelectionColor)`

Configura a cor de seleção do plano de fundo para o componente.

`void` **`setBgSelectionColor`**`(Color bgSelectionColor, boolean override)`

Configura a cor de seleção do plano de fundo para o componente.

`void` **`setBgTransparency`**`(byte transparency)`

Configura o nível de transparência do componente.

`void` **`setBgTransparency`**`(int transparency)`

Configura o nível de transparência do componente.

`void` **`setBgTransparency`**`(int transparency, boolean override)`

Configura o nível de transparência do componente.

`void` **`setBorder`**`(Border border)`

Configura a borda para o estilo.

`void` **`setBorder`**`(Border border, boolean override)`

Configura a borda para o estilo.

`void` **`setFgColor`**`(Color fgColor)`

Configura a cor do primeiro plano para o componente.

`void` **`setFgColor`**`(Color fgColor, boolean override)`

Configura a cor do primeiro plano para o componente.

`void` **`setFgSelectionColor`**`(Color fgSelectionColor)`

Configura a cor de seleção do primeiro plano para o componente.

`void` **`setFgSelectionColor`**`(Color fgSelectionColor, boolean override)`

Configura a cor de seleção do primeiro plano para o componente.

`void **setFont**(Font font)`

Configura a fonte para o componente.

`void **setFont**(Font font, boolean override)`

Configura a fonte para o componente.

`void **setMargin**(int orientation, int gap)`

Configura o Style Margin.

`void **setMargin**(int orientation, int gap, boolean override)`

Configura o Style Margin.

`void **setMargin**(int top, int bottom, int left, int right)`

Configura o Style Margin.

`void **setPadding**(int orientation, int gap)`

Configura o Style Padding.

`void **setPadding**(int orientation, int gap, boolean override)`

Configura o Style Padding.

`void **setPadding**(int top, int bottom, int left, int right)`

Configura o Style Padding.

`void **setScaleImage**(boolean scaleImage)`

Configura para *true* se a imagem no plano de fundo estiver em escala, *false* se estiver em mosaico.

`void **setScaleImage**(boolean scaleImage, boolean override)`

Configura para *true* se a imagem no plano de fundo estiver em escala, *false* se estiver em mosaico.

31.6.5 Detalhe dos campos

BG_COLOR

`public static final String BG_COLOR = "bgColor"`

Nome do atributo cor de plano de fundo para a *hashtable* de tema.

FG_COLOR

`public static final String FG_COLOR = "fgColor"`

Nome do atributo cor de primeiro plano para a *hashtable* de tema.

BG_IMAGE

`public static final String BG_IMAGE = "bgImage"`

Nome do atributo imagem de plano de fundo para a *hashtable* de tema.

BG_SELECTION_COLOR

`public static final String BG_SELECTION_COLOR = "bgSelectionColor"`

Nome do atributo seleção de cor de plano de fundo para a *hashtable* de tema.

FG_SELECTION_COLOR

`public static final String FG_SELECTION_COLOR = "fgSelectionColor"`

Nome do atributo cor de primeiro plano para a *hashtable* de tema.

FONT

```
public static final String FONT = "font"
```

Nome do atributo fonte para a *hashtable* de tema.

SCALED_IMAGE

```
public static final String SCALED_IMAGE = "scaledImage"
```

Nome do atributo imagem redimensionada para a *hashtable* de tema.

TRANSPARENCY

```
public static final String TRANSPARENCY = "transparency"
```

Nome do atributo transparência para a *hashtable* de tema.

MARGIN

```
public static final String MARGIN = "margin"
```

Nome do atributo margem para a *hashtable* de tema.

BORDER

```
public static final String BORDER = "border"
```

Nome do atributo borda para a *hashtable* de tema.

PADDING

```
public static final String PADDING = "padding"
```

Nome do atributo Padding para a *hashtable* de tema.

31.6.6 Detalhe dos construtores

Style

```
public Style()
```

Cada componente quando se auto-desenha utiliza esse objeto para determinar em quais cores deve utilizar. Quando um componente é gerando, ele constrói um objeto `Style` padrão. Os valores-padrão para cada componente podem ser alterados utilizando a classe `UIManager`.

Style

```
public Style(Style style)
```

Cria uma cópia completa de um determinado estilo. Observar que se o estilo original for modificado manualmente (chamando o `setter` para ele), ele não pode ser alterado ao mudar um tema ou `LookAndFeel`, no entanto, nesse caso o novo estilo criado será alterado.

Parâmetros:

`style` – estilo a ser copiado

Style

```
public Style(Color fgColor,
             Color bgColor,
             Color fgSelectionColor,
             Color bgSelectionColor,
             Font f,
             byte transparency)
```

Cria um novo estilo com os atributos determinados.

Parâmetros:

`fgColor` – cor do primeiro plano

`bgColor` - cor do plano de fundo

`fgSelectionColor` – cor de seleção do primeiro plano

`fgSelectionColor` – cor de seleção do plano de fundo

`f` - fonte

`transparency` – nível de transparência

Style

```
public Style(Color fgColor,
             Color bgColor,
             Color fgSelectionColor,
             Color bgSelectionColor,
             Font f,
             byte transparency,
             Image im,
             boolean scaledImage)
```

Cria um novo estilo com os atributos determinados.

Parâmetros:

`fgColor` – cor do primeiro plano

`bgColor` - cor do plano de fundo

`fgSelectionColor` – cor de seleção do primeiro plano

`fgSelectionColor` – cor de seleção do plano de fundo

`f` - fonte

`transparency` – nível de transparência

`im` – imagem-base

`scaledImage` – indica se a imagem foi redimensionada

31.6.7 Detalhe dos métodos

merge

```
public void merge(Style style)
```

Combina o novo estilo com o estilo atual sem alterar os elementos que foram modificados.

Parâmetros:

`style` – novos valores de estilos do tema atual

isModified

```
public boolean isModified()
```

Retorna *true* se o estilo foi modificado manualmente após ter sido criado pelo `LookAndFeel`. Se o estilo for modificado manualmente (por um dos métodos configurados), ele deve ser mesclado em vez de substituído.

Retorna:

true se modificado, caso contrário, *false*

getBgColor

```
public Color getBgColor()
```

Retorna a cor do plano de fundo para o componente.

Retorna:

cor do plano de fundo para o componente

Relaciona-se com:

`setBgColor(java.awt.Color)`

getBgImage

```
public Image getBgImage()
```

Retorna a imagem do plano de fundo para o componente.

Retorna:

imagem do plano de fundo para o componente

Relaciona-se com:

`setBgImage(com.sun.dtv.lwuit.Image)`

getFgColor

```
public Color getFgColor()
```

Retorna a cor do primeiro plano para o componente.

Retorna:

cor do primeiro plano para o componente

Relaciona-se com:

`setFgColor(java.awt.Color)`

getFont

```
public Font getFont()
```

Retorna a fonte para o componente.

Retorna:

fonte para o componente

Relaciona-se com:

`setFont(com.sun.dtv.lwuit.Font)`

setBgColor

`public void setBgColor(Color bgColor)`

Configura a cor do plano de fundo para o componente.

Parâmetros:

`bgColor` – cor RRGGBB que ignora o componente alfa

Relaciona-se com:

`getBgColor()`

setBgImage

`public void setBgImage(Image bgImage)`

Configura a imagem do plano de fundo para o componente.

Parâmetros:

`bgImage` - imagem

Relaciona-se com:

`getBgImage()`

setFgColor

`public void setFgColor(Color fgColor)`

Configura a cor do primeiro plano para o componente.

Parâmetros:

`fgColor` – cor

Relaciona-se com:

`getFgColor()`

setFont

`public void setFont(Font font)`

Configura a fonte para o componente.

Parâmetros:

`font` - fonte

Relaciona-se com:

`getFont()`

getBgSelectionColor

`public Color getBgSelectionColor()`

Retorna a cor de seleção do plano de fundo para o componente.

Retorna:

cor de seleção do plano de fundo para o componente

Relaciona-se com:

`setBgSelectionColor(java.awt.Color)`

setBgSelectionColor

`public void setBgSelectionColor(Color bgSelectionColor)`

Configura a cor de seleção do plano de fundo para o componente.

Parâmetros:

`bgSelectionColor` – cor

Relaciona-se com:

`getBgSelectionColor()`

getFgSelectionColor

`public Color getFgSelectionColor()`

Retorna a cor de seleção do primeiro plano para o componente.

Retorna:

cor de seleção do primeiro plano para o componente

Relaciona-se com:

`setFgSelectionColor(java.awt.Color)`

setFgSelectionColor

`public void setFgSelectionColor(Color fgSelectionColor)`

Configura a cor de seleção do primeiro plano para o componente.

Parâmetros:

`fgSelectionColor` – cor

Relaciona-se com:

`getFgSelectionColor()`

isScaleImage

`public boolean isScaleImage()`

Indica se a imagem no plano de fundo foi redimensionada.

Retorna:

true se a imagem no plano de fundo for redimensionada, *false* se estiver em mosaico

Relaciona-se com:

`setScaleImage(boolean)`

setScaleImage

```
public void setScaleImage(boolean scaleImage)
```

Configura para *true* se a imagem no plano de fundo estiver em escala, *false* se estiver em mosaico.

Parâmetros:

`scaleImage` – valor indicando se a imagem no plano de fundo foi redimensionada

Relaciona-se com:

`isScaleImage()`

getBgTransparency

```
public byte getBgTransparency()
```

Retorna o nível de transparência do componente.

Retorna:

nível de transparência do componente

Relaciona-se com:

`setBgTransparency(byte)`

setBgTransparency

```
public void setBgTransparency(byte transparency)
```

Configura o nível de transparência do componente.

Parâmetros:

`transparency` – nível de transparência como byte

Relaciona-se com:

`getBgTransparency()`

setBgTransparency

```
public void setBgTransparency(int transparency)
```

Configura o nível de transparência do componente. Valores válidos devem ser números entre 0-255.

Parâmetros:

`transparency` – valor entre 0-255

Relaciona-se com:

`getBgTransparency()`

setPadding

```
public void setPadding(int top,
                       int bottom,
                       int left,
                       int right)
```

Configura o `Style Padding`.

Parâmetros:

`top` – número de pixels para *pad*

`bottom` – número de pixels para *pad*

`left` – número de pixels para *pad*

`right` – número de pixels para *pad*

Relaciona-se com:

`getPadding(int)`

setPadding

```
public void setPadding(int orientation,
                       int gap)
```

Configura o `Style Padding`.

Parâmetros:

`orientation` – um de: `Component.TOP, Component.BOTTOM, Component.LEFT, Component.RIGHT`

`gap` – número de pixels para *pad*

Relaciona-se com:

`getPadding(int)`

setMargin

```
public void setMargin(int top,
                      int bottom,
                      int left,
                      int right)
```

Configura o Style Margin.

Parâmetros:

`top` – número de pixels da margem

`bottom` – número de pixels da margem

`left` – número de pixels da margem

`right` – número de pixels da margem

Relaciona-se com:

`getMargin(int)`

setMargin

```
public void setMargin(int orientation,
                      int gap)
```

Configura o Style Margin.

Parâmetros:

`orientation` – um de: Component.`TOP, Component.BOTTOM, Component.LEFT, Component.RIGHT`

`gap` – número de pixels da margem

Relaciona-se com:

`getMargin(int)`

getPadding

```
public int getPadding(int orientation)
```

Retorna o Padding.

Parâmetros:

`orientation` – um de: Component.`TOP, Component.BOTTOM, Component.LEFT, Component.RIGHT`

Retorna:

número de pixels do padding em determinada orientação

Relaciona-se com:

`setPadding(int, int)`

getMargin

```
public int getMargin(int orientation)
```

Retorna o Margin.

Parâmetros:

`orientation` – um de: Component.`TOP, Component.BOTTOM, Component.LEFT, Component.RIGHT`

Retorna:

número de pixels da margem em determinada orientação

Relaciona-se com:

`setMargin(int, int)`

setBgColor

```
public void setBgColor(Color bgColor,
                       boolean override)
```

Configura a cor do plano de fundo para o componente.

Parâmetros:

`bgColor` – cor RRGGBB que ignora o componente alfa

`override` – Se configurado para *true*, permite que o `LookAndFeel`/tema substitua o valor nesse atributo ao alterar um tema/`LookAndFeel`

Relaciona-se com:

`getBgColor()`

setBgImage

```
public void setBgImage(Image bgImage,
                       boolean override)
```

Configura a imagem do plano de fundo para o componente.

Parâmetros:

`bgImage` - imagem

`override` – Se configurado para *true*, permite que o `LookAndFeel`/tema substitua o valor nesse atributo ao alterar um tema/`LookAndFeel`

Relaciona-se com:

`getBgImage()`

setFgColor

```
public void setFgColor(Color fgColor,
                       boolean override)
```

Configura a cor do primeiro plano para o componente.

Parâmetros:

`fgColor` – cor

`override` – Se configurado para *true*, permite que o `LookAndFeel`/tema substitua o valor nesse atributo ao alterar um tema/`LookAndFeel`

Relaciona-se com:

`getFgColor()`

setFont

```
public void setFont(Font font,
                    boolean override)
```

Configura a fonte para o componente.

Parâmetros:

`font` - fonte

`override` – Se configurado para *true*, permite que o `LookAndFeel`/tema substitua o valor nesse atributo ao alterar um tema/`LookAndFeel`

Relaciona-se com:

`getFont()`

setBgSelectionColor

```
public void setBgSelectionColor(Color bgSelectionColor,
                                boolean override)
```

Configura a cor de seleção do plano de fundo para o componente.

Parâmetros:

`bgSelectionColor` – cor

`override` – Se configurado para *true*, permite que o `LookAndFeel`/tema substitua o valor nesse atributo ao alterar um tema/`LookAndFeel`

Relaciona-se com:

`getBgSelectionColor()`

setFgSelectionColor

```
public void setFgSelectionColor(Color fgSelectionColor,
                                boolean override)
```

Configura a cor de seleção do primeiro plano para o componente.

Parâmetros:

`fgSelectionColor` – cor

`override` – Se configurado para *true*, permite que o `LookAndFeel`/tema substitua o valor nesse atributo ao alterar um tema/`LookAndFeel`

Relaciona-se com:

`getFgSelectionColor()`

setScaleImage

```
public void setScaleImage(boolean scaleImage,
                          boolean override)
```

Configura para *true* se a imagem no plano de fundo estiver em escala, *false* se estiver em mosaico.

Parâmetros:

`scaleImage` – valor indicando se a imagem no plano de fundo foi redimensionada

`override` – Se configurado para *true*, permite que o `LookAndFeel`/tema substitua o valor nesse atributo ao alterar um tema/`LookAndFeel`

Relaciona-se com:

`isScaleImage()`

setBgTransparency

```
public void setBgTransparency(int transparency,
                              boolean override)
```

Configura o nível de transparência do componente. Valores válidos devem ser números entre 0-255.

Parâmetros:

`transparency` – nível de transparência

`override` – Se configurado para *true*, permite que o `LookAndFeel`/tema substitua o valor nesse atributo ao alterar um tema/`LookAndFeel`

Relaciona-se com:

`getBgTransparency()`

setPadding

```
public void setPadding(int orientation,
                       int gap,
                       boolean override)
```

Configura o `Style Padding`.

Parâmetros:

`orientation` – um de: `Component.TOP, Component.BOTTOM, Component.LEFT, Component.RIGHT`

`gap` – número de pixels para *pad*

`override` – Se configurado para *true*, permite que o `LookAndFeel`/tema substitua o valor nesse atributo ao alterar um tema/`LookAndFeel`

Relaciona-se com:

`getPadding(int)`

setMargin

```
public void setMargin(int orientation,
                      int gap,
                      boolean override)
```

Configura o `Style Margin`.

Parâmetros:

`orientation` – um de: `Component.TOP, Component.BOTTOM, Component.LEFT, Component.RIGHT`

`gap` – número de pixels da margem

`override` – Se configurado para *true*, permite que o `LookAndFeel`/tema substitua o valor nesse atributo ao alterar um tema/`LookAndFeel`

Relaciona-se com:

`getMargin(int)`

addStyleListener

`public void addStyleListener(StyleListener l)`

Adiciona um `StyleListener` ao objeto `Style`,

Parâmetros:

`l` – *listener* de estilo

Relaciona-se com:

`removeStyleListener(com.sun.dtv.lwuit.events.StyleListener)`

removeStyleListener

`public void removeStyleListener(StyleListener l)`

Remove um `StyleListener` ao objeto `Style`.

Parâmetros:

`l` – *listener* de estilo

Relaciona-se com:

`addStyleListener(com.sun.dtv.lwuit.events.StyleListener)`

setBorder

`public void setBorder(Border border)`

Configura a borda para o estilo.

Parâmetros:

`border` – novo objeto de borda para o componente

Relaciona-se com:

`getBorder()`

setBorder

```
public void setBorder(Border border,
                      boolean override)
```

Configura a borda para o estilo.

Parâmetros:

`border` – novo objeto de borda para o componente

`override` – Se configurado para *true*, permite que o `LookAndFeel`/tema substitua o valor nesse atributo ao alterar um tema/`LookAndFeel`

Relaciona-se com:

`getBorder()`

getBorder

```
public Border getBorder()
```

Retorna a borda para o estilo.

Retorna:

borda

Relaciona-se com:

`setBorder(com.sun.dtv.lwuit.plaf.Border)`

getBgPainter

```
public Painter getBgPainter()
```

Retorna o painter do plano de fundo para esse estilo, normalmente deve ser o painter de imagem/cor interno, mas pode ser determinado pelo usuário.

Retorna:

painter

Relaciona-se com:

`setBgPainter(com.sun.dtv.lwuit.Painter)`

setBgPainter

```
public void setBgPainter(Painter bgPainter)
```

Define o painter do plano de fundo para esse estilo, normalmente deve ser o painter de imagem/cor interno, mas pode ser determinado pelo usuário.

Parâmetros:

`bgPainter` – novo painter para instalar no estilo

Relaciona-se com:

```
getBgPainter()
```

31.7 Classe UIManager

31.7.1 Descrição da classe

com.sun.dtv.lwuit.plaf

```
java.lang.Object
  └com.sun.dtv.lwuit.plaf.UIManager
```

```
public class UIManager
extends Object
```

O *singleton* do ponto central gerenciando a aparência do aplicativo. Essa classe permite personalizar os estilos (temas) e a aparência da instância.

31.7.2 Índice de métodos

`Style` **`getComponentStyle`**`(String id)`

Retorna o estilo do componente com um determinado identificador ou uma **nova instância** do estilo-padrão.

`static UIManager` **`getInstance`**`()`

Método de instância de *singleton*.

`LookAndFeel` **`getLookAndFeel`**`()`

Retorna o `LookAndFeel` instalado atualmente.

`Hashtable` **`getResourceBundle`**`()`

O pacote de recurso nos permite localizar implicitamente a UI no fly, uma vez instalado, todos os *string*s de aplicativo interno interrogam o pacote de recursos e extraem seus valores dessa tabela, se aplicável.

`String` **`getThemeName`**`()`

Retorna o nome do tema atual para a UI de troca de tema.

`void` **`loadTheme`**`(InputStream input)`

Carrega um tema a partir de um determinado stream de input.

`String` **`localize`**`(String key, String defaultValue)`

Localiza a *string* determinada a partir do pacote de recursos se tal *string* existir no pacote de recursos.

`void` **`setComponentStyle`**`(String id, Style style)`

Permite que um desenvolvedor instale de maneira programática um estilo no gerenciador da UI.

`void` **`setLookAndFeel`**`(LookAndFeel plaf)`

Configura o `LookAndFeel` instalado atualmente.

`void` **`setResourceBundle`**`(Hashtable resourceBundle)`

O pacote de recurso nos permite localizar implicitamente a UI no fly, uma vez instalado, todos os *string*s de aplicativo interno interrogam o pacote de recursos e extraem seus valores dessa tabela, se aplicável.

`void` **`setThemeProps`**`(Hashtable themeProps)`

Permite o carregamento de tema manual a partir da *hashtable* dos pares tecla/valor.

31.7.3 Detalhe dos métodos

getInstance

```
public static UIManager getInstance()
```

Método de instância de *singleton*.

Retorna:

Instância do gerenciador da UI

getLookAndFeel

```
public LookAndFeel getLookAndFeel()
```

Retorna o `LookAndFeel` instalado atualmente.

Retorna:

`LookAndFeel` instalado atualmente

Relaciona-se com:

```
setLookAndFeel(com.sun.dtv.lwuit.plaf.LookAndFeel)
```

setLookAndFeel

```
public void setLookAndFeel(LookAndFeel plaf)
```

Configura o `LookAndFeel` instalado atualmente.

Parâmetros:

`plaf` – `LookAndFeel`

Relaciona-se com:

```
getLookAndFeel()
```

setComponentStyle

```
public void setComponentStyle(String id,
                              Style style)
```

Permite que um desenvolvedor instale de maneira programática um estilo no gerenciador da UI.

Parâmetros:

`id` – Identificador do componente correspondendo ao estilo determinado

`style` – objeto estilo para instalar

Relaciona-se com:

```
getComponentStyle(java.lang.String)
```

getComponentStyle

```
public Style getComponentStyle(String id)
```

Retorna o estilo do componente com um determinado identificador ou uma **nova instância** do estilo-padrão. Este método sempre retornará uma nova instância de estilo para prevenir a modificação do objeto estilo global.

Parâmetros:

`id` – identificador de componente

Retorna:

estilo apropriado (este método nunca retorna `null`)

Relaciona-se com:

`setComponentStyle(java.lang.String, com.sun.dtv.lwuit.plaf.Style)`

loadTheme

```
public void loadTheme(InputStream input)
               throws IOException
```

Carrega um tema a partir de um determinado *stream* de input. O tema pode ser especificado em um *stream* utilizando a seguinte notação:

```
 ButtonFont=System{FACE_SYSTEM;STYLE_PLAIN;SIZE_SMALL}
 LabelFgColor=333333
```

Parâmetros:

`input` – *stream* de input

Lança:

`IOException` - se o carregamento falhar

getThemeName

```
public String getThemeName()
```

Retorna o nome do tema atual para a UI de troca de tema.

Retorna:

nome do tema atual para a UI de troca de tema.

setThemeProps

```
public void setThemeProps(Hashtable themeProps)
```

Permite o carregamento de tema manual a partir da *hashtable* dos pares tecla/valor.

Parâmetros:

`themeProps` – *hashtable* contendo propostas de tema

getResourceBundle

```
public Hashtable getResourceBundle()
```

O pacote de recurso nos permite localizar implicitamente a UI no fly, uma vez instalado, todos os *string*s de aplicativo interno interrogam o pacote de recursos e extraem seus valores dessa tabela, se aplicável.

Retorna:

pacote de recurso

Relaciona-se com:

`setResourceBundle(java.util.Hashtable)`

setResourceBundle

```
public void setResourceBundle(Hashtable resourceBundle)
```

O pacote de recurso nos permite localizar implicitamente a UI no fly, uma vez instalado, todos os *string*s de aplicativo interno interrogam o pacote de recursos e extraem seus valores dessa tabela, se aplicável.

Parâmetros:

`resourceBundle` - resourceBundle

Relaciona-se com:

`getResourceBundle()`

localize

```
public String localize(String key,
                       String defaultValue)
```

Localiza o determinado *string* a partir do pacote de recursos se tal String existir no pacote de recursos. Se não houver nenhuma tecla no pacote, ou caso o pacote não esteja instalado, o valor-padrão é retornado.

Parâmetros:

`key` – tecla utilizada para consultar no pacote de recurso

`defaultValue` – valor retornado se tal tecla não existir

Retorna:

ou valor-padrão ou valor apropriado

32 Pacote com.sun.dtv.lwuit.util

32.1 Descrição do pacote

Funções de utilitários que são ou muito específicas de um domínio ou não se “encaixam” em nenhum outro pacote.

NOTA Este pacote está especificado desde o Java DTV 1.0.

32.2 Índice de classes

Log

Framework de *logging* plugável que permite que um desenvolvedor inicie saída de *log*.

Resources

Carrega recursos a partir do arquivo de recurso binário gerado durante o processo de construção.

32.3 Classe Log

32.3.1 Descrição da classe

`com.sun.dtv.lwuit.util`

```
java.lang.Object
  └com.sun.dtv.lwuit.util.Log
```

```
public class Log
extends Object
```

Framework de *logging* plugável que permite que um desenvolvedor inicie saída de *log*.

32.3.2 Índice de campos

`static int` **`DEBUG`**

Constante indicando que o nível de *log* `DEBUG` é o padrão e o nível mais baixo seguido por `INFO, WARNING` e `ERROR`.

`static int` **`ERROR`**

Constante indicando que o nível de *log* `ERROR` é o padrão e o nível mais baixo seguido por `INFO, WARNING` e `DEBUG`.

`static int` **`INFO`**

Constante indicando que o nível de *log* `INFO` é o padrão e o nível mais baixo seguido por `WARNING, ERROR e DEBUG`.

`static int` **`WARNING`**

Constante indicando nível de *log* `WARNING` é o padrão e o nível mais baixo seguido por `INFO, ERROR` e `DEBUG`.

32.3.3 Índice de construtores

`Log``()`

32.3.4 Índice de métodos

`static int` **`getLevel`**`()`

Retorna o nível de registro para detalhes de *log* de impressão, quanto mais baixo o valor mais detalhado devem ser os *printouts*.

`static void` **`install`**`(Log newInstance)`

Instala uma subclasse de *log* que pode substituir destino/comportamento de registro.

`static void` **`p`**`(String text)`

O método de `printIn` padrão chama o método de instância de impressão, utiliza nível `DEBUG`.

`static void` **`p`**`(String text, int level)`

O método de `printIn` padrão chama o método de instância de impressão, utiliza nível determinado.

`protected void` **`print`**`(String text, int level)`

Implementação de *log* padrão imprime para o console e conector de arquivo, se aplicável.

`static void` **`setLevel`**`(int level)`

Configura o nível de *log* para detalhes de *log* de impressão, quanto mais baixo o valor mais detalhados devem ser os *printouts*.

`static void` **`showLog`**`()`

Coloca um formulário com o *log* como um `TextArea` na tela, este método pode ser anexado para aparecer em um determinado tempo ou utilizando uma tecla global fixa.

32.3.5 Detalhe dos campos

DEBUG

```
public static final int DEBUG = 1
```

Constante indicando o nível de *log* `DEBUG` é o padrão e o nível mais baixo seguido por `INFO, WARNING` e `ERROR`.

INFO

```
public static final int INFO = 2
```

Constante indicando que o nível de *log* `DEBUG` é o padrão e o nível mais baixo seguido por `INFO, WARNING` e `ERROR`.

WARNING

```
public static final int WARNING = 3
```

Constante indicando que o nível de *log* `DEBUG` é o padrão e o nível mais baixo seguido por `INFO, WARNING` e `ERROR`.

ERROR

```
public static final int ERROR = 4
```

Constante indicando que o nível de *log* `DEBUG` é o padrão e o nível mais baixo seguido por `INFO, WARNING` e `ERROR`.

32.3.6 Detalhe dos construtores

Log

```
public Log()
```

32.3.7 Detalhe dos métodos

install

```
public static void install(Log newInstance)
```

Instala uma subclasse de *log* que pode substituir destino/comportamento de registro.

Parâmetros:

`newInstance` – nova instância para o objeto `Log`

p

```
public static void p(String text)
```

O método de `printIn` padrão chama o método de instância de impressão, utiliza nível `DEBUG`.

Parâmetros:

`text` – texto para imprimir

p

```
public static void p(String text,
                     int level)
```

O método de `printIn` padrão chama o método de instância de impressão, utiliza nível determinado.

Parâmetros:

`text` – texto para imprimir

`level` – um de `DEBUG, INFO, WARNING, ERROR`

setLevel

```
public static void setLevel(int level)
```

Configura o nível de *log* para detalhes de *log* de impressão, quanto mais baixo o valor mais detalhado deve ser os *printouts*.

Parâmetros:

`level` – um de `DEBUG, INFO, WARNING, ERROR`

Relaciona-se com:

`getLevel()`

getLevel

```
public static int getLevel()
```

Retorna o nível de registro para detalhes de *log* de impressão, quanto mais baixo o valor mais detalhado deve ser os *printouts*.

Retorna:

um de `DEBUG, INFO, WARNING, ERROR`

Relaciona-se com:

`setLevel()`

showLog

```
public static void showLog()
```

Coloca um formulário com o *log* como um `TextArea` na tela, este método pode ser anexado para aparecer em um determinado tempo ou utilizando uma tecla global fixa. Utilizar este método pode causar um problema com output de *log* adicional.

print

```
protected void print(String text,
                     int level)
```

Implementação de *log* padrão imprime para o console e conector de arquivo, se aplicável. Também acrescenta ao início informação de *thread* e tempo.

Parâmetros:

`text` – texto para imprimir

`level` – um de `DEBUG, INFO, WARNING, ERROR`

32.4 Classe Resources

32.4.1 Descrição da classe

`com.sun.dtv.lwuit.util`

`java.lang.Object`

└com.sun.dtv.lwuit.util.Resources

```
public class Resources
extends Object
```

Carrega recursos a partir do arquivo de recursos binário gerado durante o processo de construção. Consulte a seção 16 (Especificação de arquivos de recursos) para detalhes. Um recurso pode ser carregado inteiramente na memória, mas também são possíveis outras abordagens. Caso o recurso seja carregado inteiramente na memória, deve-se disponibilizar memória suficiente para acomodar o arquivo de recurso.

32.4.2 Índice de métodos

`StaticAnimation` **`getAnimation`**`(String id)`

Retorna o recurso de animação a partir do arquivo.

`String[]` **`getAnimationResourceNames`**`()`

Retorna os nomes das animações dentro deste pacote.

`InputStream` **`getData`**`(String id)`

Retorna o recurso de dados a partir do arquivo.

`String[]` **`getDataResourceNames`**`()`

Retorna os nomes dos recursos de dados dentro deste pacote.

`Font` **`getFont`**`(String id)`

Retorna o recurso de fonte a partir do arquivo.

`String[]` **`getFontResourceNames`**`()`

Retorna os nomes das fontes dentro deste pacote.

`Image` **`getImage`**`(String id)`

Retorna o recurso de imagem a partir do arquivo.

`String[]` **`getImageResourceNames`**`()`

Retorna os nomes das imagens dentro deste pacote.

`Hashtable` **`getL10N`**`(String id, String locale)`

Retorna um *hashmap* contendo pares tecla/valor de String localizados para determinado nome de Locale.

`String[]` **`getL10NResourceNames`**`()`

Retorna os nomes dos pacotes de localização dentro deste pacote.

`String[]` **`getResourceNames`**`()`

Retorna os nomes dos recursos dentro deste pacote.

`Hashtable` **`getTheme`**`(String id)`

Retorna o recurso de tema a partir do arquivo.

`String[]` **`getThemeResourceNames`**`()`

Retorna os nomes das imagens dentro deste pacote.

`boolean` **`isAnimation`**`(String name)`

Retorna *true* se esse for um recurso de animação.

`boolean` **`isData`**`(String name)`

Retorna *true* se esse for um recurso de dados.

`boolean` **`isFont`**`(String name)`

Retorna *true* se esse for um recurso de fonte.

`boolean` **`isImage`**`(String name)`

Retorna *true* se esse for um recurso de imagem.

`boolean` **`isL10N`**`(String name)`

Retorna *true* se esse for um recurso de locale.

`boolean` **`isTheme`**`(String name)`

Retorna *true* se esse for um recurso de tema.

`static Resources` **`open`**`(InputStream resource)`

Cria um objeto de recurso a partir de um determinado stream de entrada.

`static Resources` **`open`**`(String resource)`

Cria um objeto de recurso a partir do identificador de recurso JAR local.

32.4.3 Detalhe dos métodos

getResourceNames

`public String[] getResourceNames()`

Retorna os nomes dos recursos dentro deste pacote.

Retorna:

array de nomes de todos os recursos neste pacote

getDataResourceNames

`public String[] getDataResourceNames()`

Retorna os nomes dos recursos de dados dentro deste pacote.

Retorna:

array de nomes de recursos de dados neste pacote

getL10NResourceNames

`public String[] getL10NResourceNames()`

Retorna os nomes dos pacotes de localização dentro deste pacote.

Retorna:

array de nomes de recursos de localização neste pacote

getFontResourceNames

`public String[] getFontResourceNames()`

Retorna os nomes das fontes dentro deste pacote.

Retorna:

array de nomes de recursos de fonte neste pacote

getThemeResourceNames

`public String[] getThemeResourceNames()`

Retorna os nomes das imagens dentro deste pacote.

Retorna:

array de nomes de recursos de imagem neste pacote

getImageResourceNames

`public String[] getImageResourceNames()`

Retorna os nomes das imagens dentro deste pacote.

Retorna:

array de nomes de recursos de imagem neste pacote

getAnimationResourceNames

`public String[] getAnimationResourceNames()`

Retorna os nomes das animações dentro deste pacote.

Retorna:

array de nomes de recursos de animação neste pacote

isL10N

`public boolean isL10N(String name)`

Retorna *true* se esse for um recurso de locale.

Parâmetros:

`name` - nome do recurso

Retorna:

true se o recurso for um recurso de locale

Lança:

`NullPointerException` – se o recurso não existir

isTheme

`public boolean isTheme(String name)`

Retorna *true* se esse for um recurso de tema.

Parâmetros:

`name` - nome do recurso

Retorna:

true se o recurso for um tema

Lança:

`NullPointerException` – se o recurso não existir

isFont

`public boolean isFont(String name)`

Retorna *true* se esse for um recurso de fonte.

Parâmetros:

`name` - nome do recurso

Retorna:

true se o recurso for uma fonte

Lança:

`NullPointerException` – se o recurso não existir

isAnimation

`public boolean isAnimation(String name)`

Retorna *true* se esse for um recurso de animação.

Parâmetros:

`name` - nome do recurso

Retorna:

true se o recurso for uma animação

Lança:

`NullPointerException` – se o recurso não existir

isData

`public boolean isData(String name)`

Retorna *true* se esse for um recurso de dados.

Parâmetros:

`name` - nome do recurso

Retorna:

true se o recurso for um recurso de dados

Lança:

`NullPointerException` – se o recurso não existir

isImage

`public boolean isImage(String name)`

Retorna *true* se esse for um recurso de imagem.

Parâmetros:

`name` - nome do recurso

Retorna:

true se o recurso for uma imagem

Lança:

`NullPointerException` – se o recurso não existir

open

```
public static Resources open(String resource)
                      throws IOException
```

Cria um objeto de recurso a partir do identificador de recurso JAR local.

Parâmetros:

`resource` – uma referência local para um recurso utilizando a sintaxe de `Class.getResourceAsStream(String)`

Retorna:

um objeto de recurso

Lança:

`IOException` – se a abertura/leitura de recurso falhar

open

```
public static Resources open(InputStream resource)
                      throws IOException
```

Cria um objeto de recurso a partir de um determinado stream de entrada.

Parâmetros:

`resource` - stream a partir do qual ler o recurso

Retorna:

um objeto de recurso

Lança:

`IOException` – se a abertura/leitura de recurso falhar

getImage

```
public Image getImage(String id)
```

Retorna o recurso de imagem a partir do arquivo.

Parâmetros:

`id` – nome do recurso de imagem

Retorna:

instância de imagem escondida

getAnimation

```
public StaticAnimation getAnimation(String id)
```

Retorna o recurso de animação a partir do arquivo.

Parâmetros:

`id` – nome do recurso de animação

Retorna:

instância de imagem escondida

getData

```
public InputStream getData(String id)
```

Retorna o recurso de dados a partir do arquivo.

Parâmetros:

`id` – nome do recurso de dados

Retorna:

Stream de entrada recém-criado que permite ler os dados do recurso

getL10N

```
public Hashtable getL10N(String id,
                         String locale)
```

Retorna um *hashmap* contendo pares tecla/valor de String localizados para determinado nome de Locale.

Parâmetros:

`id` – nome do recurso de dados

`id` – nome do recurso de locale

Retorna:

Hashtable contendo pares de valor de tecla para dados localizados

getFont

```
public Font getFont(String id)
```

Retorna o recurso de fonte a partir do arquivo.

Parâmetros:

`id` – nome do recurso de fonte

Retorna:

instância de fonte escondida

getTheme

```
public Hashtable getTheme(String id)
```

Retorna o recurso de tema a partir do arquivo.

Parâmetros:

`id` – nome do recurso de tema

Retorna:

instância de tema escondido.

33 Pacote com.sun.dtv.media

33.1 Descrição do pacote

Pacote de base para todas as funcionalidades de mídia relevantes and controles de congelamento e restauração de mídia.

A Figura 11 mostra a estrutura do pacote `Media` do Java DTV.

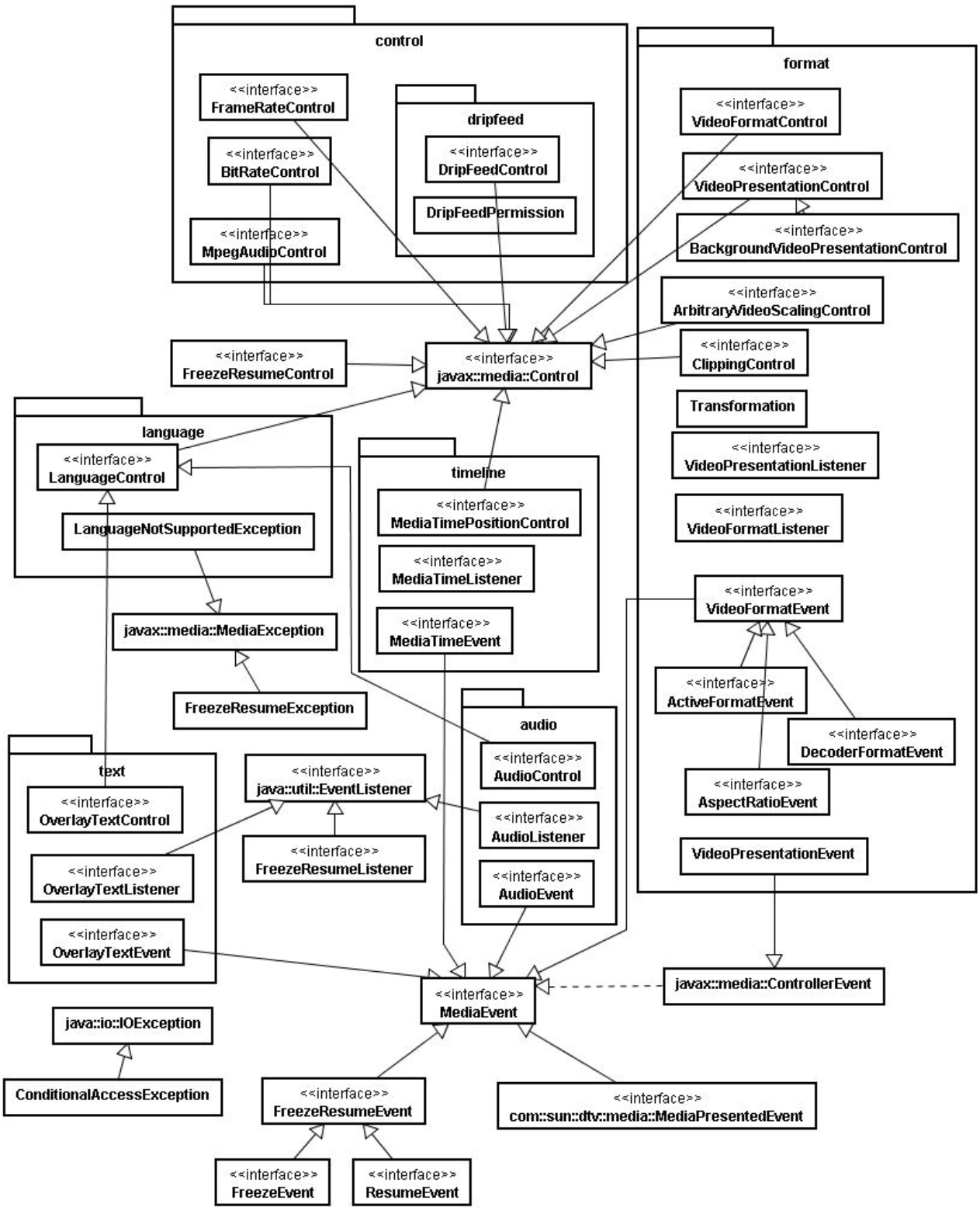


Figura 11 – Pacote `Media`

NOTA Este pacote está especificado desde o Java DTV 1.0.

33.2 Índice de interfaces

FreezeEvent

Indicando que uma ação de congelamento aconteceu seja a partir de um aplicativo ou usuário.

FreezeResumeControl

Essa interface deve ser implementada para permitir que o aplicativo congele o player.

FreezeResumeEvent

Indicando que uma ação de congelamento ou resumo aconteceu seja a partir de um aplicativo ou usuário.

FreezeResumeListener

Interface `Listener` para permitir que o aplicativo receba eventos `Freeze` e `Resume`.

MediaPresentedEvent

Esse Event é gerado após um `javax.media.Player` transferir para o estado Started.

ResumeEvent

Indicando que uma ação de resumo aconteceu seja a partir de um aplicativo ou usuário.

33.3 Índice de exceções

ConditionalAccessException

Indica que a mídia sob controle do `Player` ou `DataSource` está protegida pelo acesso condicional.

FreezeResumeException

Essa excessão indica que tanto o métdo de congelamento quanto o de resumo falharam.

33.4 Classe ConditionalAccessException

33.4.1 Descrição da classe

`com.sun.dtv.media`

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └java.io.IOException
              └com.sun.dtv.media.ConditionalAccessException
```

Todas as interfaces implementadas

`Serializable`

```
public class ConditionalAccessException
extends IOException
```

Indica que a mídia sob controle do `Player` ou `DataSource` está protegida pelo acesso condicional. Esta exceção DEVE ser executada pelo método `Player.start()` ou `DataSource.start()` se o conteúdo estiver protegido.

Relaciona-se com:

`Player, javax.media.protocol.DataSource`

33.4.2 Índice de construtores

`ConditionalAccessException``()`

Construtor sem uma informação razoável.

`ConditionalAccessException``(String reason)`

Construtor com uma informação razoável.

33.4.3 Detalhe dos construtores

ConditionalAccessException

`public ConditionalAccessException()`

Construtor sem uma informação razoável.

ConditionalAccessException

`public ConditionalAccessException(String reason)`

Construtor com uma informação razoável.

Parâmetros:

`reason` – desta exceção de acesso condicional

33.5 Interface FreezeEvent

33.5.1 Descrição da interface

`com.sun.dtv.media`

Todas as super-interfaces

`FreezeResumeEvent, MediaEvent`

`public interface FreezeEvent`

`extends FreezeResumeEvent`

Indicando que uma ação de congelamento aconteceu seja a partir de um aplicativo ou usuário.

Métodos herdados da interface `com.sun.dtv.media.`**`FreezeResumeEvent`**

`getPlayer`

33.6 Interface FreezeResumeControl

33.6.1 Descrição da interface

`com.sun.dtv.media`

Todas as super-interfaces

`Control`

```
public interface FreezeResumeControl
extends Control
```

Essa interface deve ser implementada para permitir que o aplicativo congele o *player*. Durante um congelamento do *player* o último *frame* exibido na tela deve estar ainda presente até que seja executada uma ação de resumo.

33.6.2 Índice de métodos

`void` **`addFreezeResumeListener`**`(FreezeResumeListener freezeResumeListener)`

Adiciona uma interface `Listener` de congelamento e resumo.

`void` **`freeze`**`()`

A chamada desse método congela a decodificação do *stream* de mídia logo que possível e deixa o último *frame* visível para o usuário final.

`void` **`removeFreezeResumeListener`**`(FreezeResumeListener freezeResumeListener)`

Remove uma interface `Listener` de congelamento e resumo.

`void` **`resume`**`()`

A chamada desse método resume a decodificação do *stream* de mídia após uma operação de congelamento.

33.6.3 Detalhe dos métodos

freeze

```
void freeze()
     throws FreezeResumeException
```

A chamada desse método congela a decodificação do *stream* de mídia logo que possível e deixa o último *frame* visível para o usuário final. A decodificação de áudio é interrompida. O tempo base real da mídia-base, no entanto, não é alterado.

Lança:

`FreezeResumeException` – se o congelamento falhar

resume

```
void resume()
     throws FreezeResumeException
```

A chamada desse método resume a decodificação do stream de mídia após uma operação de congelamento. Se o *player* for iniciado e se a decodificação do *stream* de mídia não for previamente congelado, as chamadas para esse método não podem surtir efeito. Se a mídia do *player* é um stream de transmissão, a apresentação iniciará na respectiva `mediaTime` quando o método de resumo for chamado. Isso significa que algumas partes do *stream* não podem ser exibidas.

Lança:

`FreezeResumeException` – se o resumo falhar

addFreezeResumeListener

`void addFreezeResumeListener(FreezeResumeListener freezeResumeListener)`

Adiciona uma interface `Listener` de congelamento e resumo.

Parâmetros:

`freezeResumeListener` – interface que receberá todos os eventos de resumo e congelamento.

Relaciona-se com:

`removeFreezeResumeListener()`

removeFreezeResumeListener

`void removeFreezeResumeListener(FreezeResumeListener freezeResumeListener)`

Remove uma interface `Listener` de congelamento e resumo.

Parâmetros:

`freezeResumeListener` - *listener* para remover.

Relaciona-se com:

`addFreezeResumeListener()`

33.7 Interface FreezeResumeEvent

33.7.1 Descrição da interface

`com.sun.dtv.media`

Todas as super-interfaces

`MediaEvent`

Todas as subinterfaces conhecidas

`FreezeEvent`, `ResumeEvent`

```
public interface FreezeResumeEvent
extends MediaEvent
```

Indicando que uma ação de congelamento ou resumo aconteceu seja a partir de um aplicativo ou usuário.

33.7.2 Índice de métodos

`Player` **`getPlayer`**`()`

Retorna o *player* JMF que for a fonte do evento.

33.7.3 Detalhe dos métodos

getPlayer

`Player getPlayer()`

Retorna o *player* JMF que for a fonte do evento.

Retorna:

Player fonte do evento.

33.8 Classe FreezeResumeException

33.8.1 Descrição da classe

com.sun.dtv.media

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └javax.media.MediaException
              └com.sun.dtv.media.FreezeResumeException
```

Todas as interfaces implementadas

```
Serializable
```

```
public class FreezeResumeException
extends MediaException
```

Essa excessão indica que tanto o métdo de congelamento quanto o de resumo falharam.

33.8.2 Índice de construtores

FreezeResumeException()

Constrói uma exceção sem uma razão.

FreezeResumeException(String reason)

Constrói uma exceção com uma razão.

33.8.3 Detalhe dos construtores

FreezeResumeException

```
public FreezeResumeException()
```

Constrói uma exceção sem uma razão.

FreezeResumeException

```
public FreezeResumeException(String reason)
```

Constrói uma exceção com uma razão.

Parâmetros:

`reason` – razão pela qual a exceção foi executada.

33.9 Interface FreezeResumeListener

33.9.1 Descrição da interface

com.sun.dtv.media

Todas as super-interfaces

`EventListener`

```
public interface FreezeResumeListener
extends EventListener
```

Interface `Listener` para permitir que o aplicativo receba eventos `Freeze` e `Resume`.

33.9.2 Índice de métodos

`void` **`freezeResumeChange`**`(FreezeResumeEvent event)`

Recebe eventos de legenda.

33.9.3 Detalhe dos métodos

freezeResumeChange

```
void freezeResumeChange(FreezeResumeEvent event)
```

Recebe eventos de legenda.

Parâmetros:

`event` – Indica ao aplicativo que uma atividade de `Freeze` e `Resume` aconteceu.

33.10 Interface MediaPresentedEvent

33.10.1 Descrição da interface

com.sun.dtv.media

Todas as super-interfaces

`MediaEvent`

```
public interface MediaPresentedEvent
extends MediaEvent
```

Esse evento é gerado após um `javax.media.Player` transferir para o estado `Started`. Alterações que resultem da utilização `javax.tv.media.MediaSelectControl` não resultarão nesse evento.

Relaciona-se com:

`javax.tv.media.MediaSelectControl, Player`

33.10.2 Índice de métodos

`Player` **`getPlayer`**`()`

Retorna o *player* JMF que for a fonte do evento.

33.10.3 Detalhe dos métodos

getPlayer

```
Player getPlayer()
```

Retorna o *player* JMF que for a fonte do evento.

Retorna:

Player fonte do evento.

33.11 Interface ResumeEvent

33.11.1 Descrição da interface

com.sun.dtv.media

Todas as super-interfaces

```
FreezeResumeEvent, MediaEvent
```

```
public interface ResumeEvent
extends FreezeResumeEvent
```

Indicando que uma ação de resumo aconteceu seja a partir de um aplicativo ou usuário.

Métodos herdados da interface `com.sun.dtv.media.`**`FreezeResumeEvent`**

```
getPlayer
```

34 Pacote com.sun.dtv.media.audio

34.1 Descrição do pacote

Pacote para controle de idioma de áudio. Este pacote fornece funcionalidades para:

- consultar os idiomas selecionáveis associados ao componente do serviço
- selecionar o idioma associado ao componente do serviço
- permitir que um aplicativo registre-se para notificações quando o idioma selecionado para o componente do serviço sofra mudanças

NOTA Este pacote está especificado desde o Java DTV 1.0.

34.2 Índice de interfaces

AudioControl

Fornece controle para o áudio registrar para eventos específicos de áudio.

AudioEvent

Indicando uma mudança com respeito à seleção de idioma de áudio.

AudioListener

`Listener` para alterações no áudio.

34.3 Interface AudioControl

34.3.1 Descrição da interface

`com.sun.dtv.media.audio`

Todas as super-interfaces

`Control, LanguageControl`

`public interface AudioControl`

`extends LanguageControl`

Fornece controle para o áudio registrar para eventos específicos de áudio.

34.3.2 Índice de métodos

`void` **`addAudioListener`**`(AudioListener audioListener)`

Adiciona uma interface audio.

`void` **`removeAudioListener`**`(AudioListener audioListener)`

Remove uma interface `Listener` de áudio.

Métodos herdados da interface `com.sun.dtv.media.language.`**`LanguageControl`**

`getDefaultLanguage, getLanguage, getSupportedLanguages, setLanguage`

34.3.3 Detalhe dos métodos

addAudioListener

`void addAudioListener(AudioListener audioListener)`

Adiciona uma interface audio.

Parâmetros:

`audioListener` - interface que receberá todos os eventos de legenda.

Relaciona-se com:

`removeAudioListener()`

removeAudioListener

`void removeAudioListener(AudioListener audioListener)`

Remove uma interface `Listener` de áudio.

Parâmetros:

`audioListener` – *listener* para remover.

Relaciona-se com:

`addAudioListener()`

34.4 Interface AudioEvent

34.4.1 Descrição da interface

com.sun.dtv.media.audio

Todas as super-interfaces

MediaEvent

```
public interface AudioEvent
extends MediaEvent
```

Indicando uma mudança com respeito à seleção de idioma de áudio.

Relaciona-se com:

LanguageControl

34.4.2 Índice de métodos

Player **getPlayer**()

Retorna o *player* JMF que for a fonte do evento.

34.4.3 Detalhe dos métodos

getPlayer

```
Player getPlayer()
```

Retorna o *player* JMF que for a fonte do evento.

Retorna:

Player fonte do evento.

34.5 Interface AudioListener

34.5.1 Descrição da interface

com.sun.dtv.media.audio

Todas as super-interfaces

EventListener

```
public interface AudioListener
extends EventListener
```

Listener para alterações no áudio.

34.5.2 Índice de métodos

void **audioChange**(AudioEvent audioEvent)

Recebe eventos de Audio.

34.5.3 Detalhe dos métodos

audioChange

```
void audioChange(AudioEvent audioEvent)
```

Recebe eventos de Audio.

Parâmetros:

`audioEvent` – evento que indica que ocorreu uma alteração no áudio.

35 Pacote com.sun.dtv.media.control

35.1 Descrição do pacote

Controles adicionais para obter informações sobre a mídia apresentada. Recupera informações sobre taxas de bit, taxas de *frame* e atributos do áudio MPEG.

NOTA Este pacote está especificado desde o Java DTV 1.0.

35.2 Índice de interfaces

BitRateControl

Esta interface é um controle para recuperar a taxa de bit.

FrameRateControl

Esta interface é um controle para recuperar a taxa de frame.

MpegAudioControl

Esta interface é um controle para recuperar os parâmetros para áudio MPEG.

35.3 Interface BitRateControl

35.3.1 Descrição da interface

`com.sun.dtv.media.control`

Todas as super-interfaces

`Control`

```
public interface BitRateControl
extends Control
```

Esta interface é um controle para recuperar a taxa de bit. As taxas de bits são especificadas em bits por segndo em todos os métodos deste controle. Esse controle pode ser utilizado para exportar as informações de taxa de bit para mídia sob controle de um objeto.

35.3.2 Índice de métodos

`int` **`getBitRate`**`()`

Retorna a taxa de bit atual do objeto proprietário.

`int` **`getMaxSupportedBitRate`**`()`

Retorna a taxa de bit mais alta para a qual esse objeto consegue codificar a mídia, retorna – 1 se a taxa de bit for desconhecida.

`int` **`getMinSupportedBitRate`**`()`

Retorna a taxa de bit mais baixa para a qual esse objeto consegue codificar a mídia, retorna -1 se a taxa de bit for desconhecida.

35.3.3 Detalhe dos métodos

getBitRate

```
int getBitRate()
```

Retorna a taxa de bit atual do objeto proprietário. Se a mídia for de taxa de bit variável, o valor retornado deve ser uma taxa instantânea ou média durante um período de tempo. Se a taxa de bit for desconhecida, o método deve retornar -1.

Retorna:

taxa de bit

getMinSupportedBitRate

```
int getMinSupportedBitRate()
```

Retorna a taxa de bit mais baixa para a qual esse objeto consegue codificar a mídia, retorna -1 se a taxa de bit for desconhecida.

Retorna:

taxa de bit mínima suportada

getMaxSupportedBitRate

```
int getMaxSupportedBitRate()
```

Retorna a taxa de bit mais alta para a qual esse objeto consegue codificar a mídia, retorna – 1 se a taxa de bit for desconhecida.

Retorna:

taxa de bit máxima suportada

35.4 Interface FrameRateControl

35.4.1 Descrição da interface

`com.sun.dtv.media.control`

Todas as super-interfaces

```
Control
```

```
public interface FrameRateControl
```

```
extends Control
```

Esta interface é um controle para recuperar a taxa de frame.

35.4.2 Índice de métodos

`float` **`getFrameRate`**`()`

Retorna a taxa de frame de output atual.

`float` **`getMaxSupportedFrameRate`**`()`

Retorna a taxa máxima de frame de output.

`float` **`getPreferredFrameRate`**`()`

Retorna a taxa de frame de output padrão.

35.4.3 Detalhe dos métodos

getFrameRate

`float getFrameRate()`

Retorna a taxa de frame de output atual. Retorna -1 se for desconhecida.

Retorna:

taxa de saída de frame de output em frames por segundo.

getMaxSupportedFrameRate

`float getMaxSupportedFrameRate()`

Retorna a taxa máxima de frame de output. Retorna -1 se for desconhecida.

Retorna:

taxa máxima de frame de output em frames por segundo.

getPreferredFrameRate

`float getPreferredFrameRate()`

Retorna a taxa de frame de output padrão. Retorna -1 se for desconhecida.

Retorna:

taxa de frame de output padrão em frames por segundo.

35.5 Interface MpegAudioControl

35.5.1 Descrição da interface

`com.sun.dtv.media.control`

Todas as super-interfaces

`Control`

```
public interface MpegAudioControl
extends Control
```

Esta interface é um controle para recuperar os parâmetros para áudio MPEG.

35.5.2 Índice de campos

`int` **`FIVE_CHANNELS_3_0_2_0`**

Indica suporte para layout de cinco canais 3-0 2-0. Esquerda, Centro e Direta do primeiro programa, Esquerda e Direita do segundo programa.

`int` **`FIVE_CHANNELS_3_2`**

Indica suporte para layout de cinco canais 3-2: Surround Esquerdo, Centro, Direito, Esquerdo e surround Direito.

`int` **`FOUR_CHANNELS_2_0_2_0`**

Indica suporte para layout de quatro canais 2-0 2-0. Esquerda e Direta do primeiro programa, Esquerda e Direita do segundo programa.

`int` **`FOUR_CHANNELS_2_2`**

Indica suporte para layout de quatro canais 2-2. Surround Esquerdo, Direito, Esquerdo e surround Direito.

`int` **`FOUR_CHANNELS_3_1`**

Indica suporte para layout de quatro canais 3-1. Surround Esquerdo, Centro, Direito e único.

`int` **`LAYER_1`**

Indica suporte para camada de áudio 1.

`int` **`LAYER_2`**

Indica suporte para camada de áudio 2.

`int` **`LAYER_3`**

Indica suporte para camada de áudio 3.

`int` **`SAMPLING_RATE_16`**

Indica suporte para taxa de amostra de áudio de 16 KHz.

`int` **`SAMPLING_RATE_22_05`**

Indica suporte para taxa de amostra de áudio de 22.05 KHz.

`int` **`SAMPLING_RATE_24`**

Indica suporte para taxa de amostra de áudio de 24 KHz.

`int` **`SAMPLING_RATE_32`**

Indica suporte para taxa de amostra de áudio de 32 KHz.

`int` **`SAMPLING_RATE_44_1`**

Indica suporte para taxa de amostra de áudio de 44.1 KHz.

`int` **`SAMPLING_RATE_48`**

Indica suporte para taxa de amostra de áudio de 48 KHz.

`int` **`SINGLE_CHANNEL`**

Indica suporte para layout de canal único.

`int` **`THREE_CHANNELS_2_1`**

Indica suporte para layout de três canais 2-1. Surround Esquerdo, Direito e único.

`int` **`THREE_CHANNELS_3_0`**

Indica suporte para layout de três canais 3-0. Esquero, Centro e Direito.

`int` **`TWO_CHANNELS_DUAL`**

Indica suporte para duplo layout de dois canais.

`int` **`TWO_CHANNELS_STEREO`**

Indica suporte para layout estéreo de dois canais.

35.5.3 Índice de métodos

`int` **`getAudioLayer`**`()`

Retorna a camada de áudio MPEG atual, ou -1 se a camada de áudio não estiver disponível.

`int` **`getChannelLayout`**`()`

Retorna o layout de canal de áudio MPEG atual, ou -1 se o layout de canal não estiver disponível.

`boolean` **`getLowFrequencyChannel`**`()`

Retorna *true* se o modo de canal de baixa frequência estiver ligado.

`boolean` **`getMultilingualMode`**`()`

Retorna *true* se o modo multilíngue estiver ligado.

`int` **`getSupportedAudioLayers`**`()`

Retorna a capacidade de suporte de camada de áudio.
O valor retornado consiste de um `OR` lógico entre as *flag*s relevantes.

`int` **`getSupportedChannelLayouts`**`()`

Retorna a capacidade de suporte de layout de canal de áudio.
O valor retornado consiste de um ou lógico entre as *flag*s relevantes, ou -1 se a informação não estiver disponível.

`int` **`getSupportedSamplingRates`**`()`

Retorna a capacidade de suporte de taxa de amostra de áudio.
O valor retornado consiste de um `OR` lógico entre as *flag*s relevantes.

`boolean` **`isLowFrequencyChannelSupported`**`()`

Retorna a capacidade de suporte de canal de baixa frequência.

`boolean` **`isMultilingualModeSupported`**`()`

Retorna a capacidade de suporte de modo multilíngue.

35.5.4 Detalhe dos campos

LAYER_1

```
public static final int LAYER_1 = 1
```

Indica suporte para camada de áudio 1.

LAYER_2

```
public static final int LAYER_2 = 2
```

Indica suporte para camada de áudio 2.

LAYER_3

```
public static final int LAYER_3 = 4
```

Indica suporte para camada de áudio 3.

SAMPLING_RATE_16

```
public static final int SAMPLING_RATE_16 = 1
```

Indica suporte para taxa de amostra de áudio de 16 KHz.

SAMPLING_RATE_22_05

```
public static final int SAMPLING_RATE_22_05 = 2
```

Indica suporte para taxa de amostra de áudio de 22.05 KHz.

SAMPLING_RATE_24

```
public static final int SAMPLING_RATE_24 = 4
```

Indica suporte para taxa de amostra de áudio de 24 KHz.

SAMPLING_RATE_32

```
public static final int SAMPLING_RATE_32 = 8
```

Indica suporte para taxa de amostra de áudio de 32 KHz.

SAMPLING_RATE_44_1

```
public static final int SAMPLING_RATE_44_1 = 16
```

Indica suporte para taxa de amostra de áudio de 44.1 KHz.

SAMPLING_RATE_48

```
public static final int SAMPLING_RATE_48 = 32
```

Indica suporte para taxa de amostra de áudio de 48 KHz.

SINGLE_CHANNEL

```
public static final int SINGLE_CHANNEL = 1
```

Indica suporte para layout de canal único.

TWO_CHANNELS_STEREO

```
public static final int TWO_CHANNELS_STEREO = 2
```

Indica suporte para layout estéreo de dois canais.

TWO_CHANNELS_DUAL

```
public static final int TWO_CHANNELS_DUAL = 4
```

Indica suporte para duplo layout de dois canais.

THREE_CHANNELS_2_1

```
public static final int THREE_CHANNELS_2_1 = 8
```

Indica suporte para layout de três canais 2-1. Surround Esquerdo, Direito e único.

THREE_CHANNELS_3_0

```
public static final int THREE_CHANNELS_3_0 = 16
```

Indica suporte para layout de três canais 3-0. Esquero, Centro e Direito.

FOUR_CHANNELS_2_0_2_0

```
public static final int FOUR_CHANNELS_2_0_2_0 = 32
```

Indica suporte para layout de quatro canais 2-0 2-0. Esquerda e Direta do primeiro programa, Esquerda e Direita do segundo programa.

FOUR_CHANNELS_2_2

```
public static final int FOUR_CHANNELS_2_2 = 64
```

Indica suporte para layout de quatro canais 2-2. Surround Esquerdo, Direito, Esquerdo e surround Direito.

FOUR_CHANNELS_3_1

```
public static final int FOUR_CHANNELS_3_1 = 128
```

Indica suporte para layout de quatro canais 3-1. Surround Esquerdo, Centro, Direito e único.

FIVE_CHANNELS_3_0_2_0

```
public static final int FIVE_CHANNELS_3_0_2_0 = 256
```

Indica suporte para layout de cinco canais 3-0 2-0. Esquerda, Centro e Direta do primeiro programa, Esquerda e Direita do segundo programa.

FIVE_CHANNELS_3_2

```
public static final int FIVE_CHANNELS_3_2 = 512
```

Indica suporte para layout de cinco canais 3-2: Surround Esquerdo, Centro, Direito, Esquerdo e surround Direito.

35.5.5 Detalhe dos métodos

getSupportedAudioLayers

```
int getSupportedAudioLayers()
```

Retorna a capacidade de suporte de camada de áudio.
O valor retornado consiste de um OR lógico entre as *flag*s relevantes.

Retorna:

camada de áudio suportada.

Relaciona-se com:

`LAYER_1`, `LAYER_2`, `LAYER_3`

getSupportedSamplingRates

```
int getSupportedSamplingRates()
```

Retorna a capacidade de suporte de taxa de amostra de áudio.
O valor retornado consiste de um OR lógico entre as *flag*s relevantes.

Retorna:

taxas de amostra suportadas.

Relaciona-se com:

`SAMPLING_RATE_16`, `SAMPLING_RATE_22_05`, `SAMPLING_RATE_24`, `SAMPLING_RATE_32`, `SAMPLING_RATE_44_1`, `SAMPLING_RATE_48`

getSupportedChannelLayouts

```
int getSupportedChannelLayouts()
```

Retorna a capacidade de suporte de layout de canal de áudio.
O valor retornado consiste de um `OR` lógico entre as *flag*s relevantes, ou -1 se a informção não estiver disponível.

Retorna:

layout de canal suportado.

Relaciona-se com:

`SINGLE_CHANNEL`, `TWO_CHANNELS_STEREO`, `TWO_CHANNELS_DUAL`, `THREE_CHANNELS_2_1`, `THREE_CHANNELS_3_0`, `FOUR_CHANNELS_2_0_2_0`, `FOUR_CHANNELS_2_2`, `FIVE_CHANNELS_3_0_2_0`, `FIVE_CHANNELS_3_2`

isLowFrequencyChannelSupported

```
boolean isLowFrequencyChannelSupported()
```

Retorna a capacidade de suporte de canal de baixa frequência.

Retorna:

`true` se o canal de baixa frequência for suportado.

isMultilingualModeSupported

```
boolean isMultilingualModeSupported()
```

Retorna a capacidade de suporte de modo multilíngue.

Retorna:

`true` se o modo multilíngue for suportado.

getAudioLayer

```
int getAudioLayer()
```

Retorna a camada de áudio MPEG atual, ou -1 se a camada de áudio não estiver disponível.

Retorna:

camada atual

getChannelLayout

`int getChannelLayout()`

Retorna o layout de canal de áudio MPEG atual, ou -1 se o layout de canal não estiver disponível.

Retorna:

layout de canal atual.

getLowFrequencyChannel

`boolean getLowFrequencyChannel()`

Retorna *true* se o modo de canal de baixa frequência estiver ligado.

Retorna:

true se o modo de canal de baixa frequência estiver ligado.

getMultilingualMode

`boolean getMultilingualMode()`

Retorna *true* se o modo Multilíngue estiver ligado.

Retorna:

true se o modo multilíngue estiver ligado.

36 Pacote com.sun.dtv.media.dripfeed

36.1 Descrição do pacote

Controle para prover dados de imagem fixa para um *player* JMF. Os dados ficam sob controle do aplicativo. As imagens paradas podem ser frames individuais de uma fonte de vídeo.

NOTA Este pacote está especificado desde o Java DTV 1.0.

36.2 Índice de interfaces

DripFeedControl

Este controle permite a alimentação de modo progressivo de partes de um clip em um Player.

36.3 Índice de classes

DripFeedPermission

Esta classe representa uma permissão para acessar o `DripFeedControl`.

36.4 Interface DripFeedControl

36.4.1 Descrição da interface

`com.sun.dtv.media.dripfeed`

Todas as super-interfaces

`Control`

```
public interface DripFeedControl
extends Control
```

Este controle permite a alimentação de modo progressivo de partes de um *clip* em um *player*.

EXEMPLO - os frames que devem ser exibidos pelo *player* são armazenados individualmente em *array*s de byte.

```
...
// Cria um novo MediaLocator e instancia um novo objeto Player
MediaLocator dripLocator = new MediaLocator("dripfeed://");
player = Manager.createPlayer(dripLocator);
// recupera um DripFeedControl do Player
dripControl = (DripFeedControl)player.getControl("com.sun.dtv.media.dripfeed.DripFeedControl")
;

// pega o OutputStream do DripFeedControl
OutputStream writeToPlayer = dripControl.getOutputStream();
BufferedOutputStream bufStream = new BufferedOutputStream(writeToPlayer, maxFrameSize);
bufStream.write(frame);
```

36.4.2 Índice de métodos

`OutputStream` **`getOutputStream`**`()`

Recupera um `OutputStream` do controle que pode ser utilizado para escrever dados de imagens paradas para o Player a partir do qual o controle é recuperado.

36.4.3 Detalhe dos métodos

getOutputStream

```
OutputStream getOutputStream()
                      throws IOException
```

Recupera um `OutputStream` do controle que pode ser utilizado para escrever dados de imagens paradas para o *player* a partir do qual o controle é recuperado.

Retorna:

stream de output que pode ser utilizado pelo aplicativo para escrever para o *player* JMF.

Lança:

`IOException` – caso o *player* não consiga lidar com os dados escritos para o `OutputStream`.

36.5 Classe DripFeedPermission

36.5.1 Descrição da classe

`com.sun.dtv.media.dripfeed`

```
java.lang.Object
  └java.security.Permission
```

└java.security.BasicPermission

└com.sun.dtv.media.dripfeed.DripFeedPermission

Todas as interfaces implementadas

Guard, Serializable

```
public class DripFeedPermission
extends BasicPermission
```

Esta classe representa uma permissão para acessar o DripFeedControl. Se essa permissão não for concedida, o javax.media.Manager executará uma exceção durante a criação de um javax.media.Player se o MediaLocator for criado com a URL "dripfeed://".

36.5.2 Índice de construtores

DripFeedPermission(String name)

Cria uma nova DripFeedPermission com um dado nome.

36.5.3 Índice de métodos

boolean **implies**(Permission p)

Verifica se o DripFeePermisson especificado está "implícito" por esse objeto.

36.5.4 Detalhe dos construtores

DripFeedPermission

```
public DripFeedPermission(String name)
```

Cria uma nova DripFeedPermission com um dado nome.

Parâmetros:

name - nome da permissão.

36.5.5 Detalhe dos métodos

implies

```
public boolean implies(Permission p)
```

Verifica se o DripFeePermisson especificado está "implícito" por esse objeto. Mais especificamente, esse método retorna *true* se:

a classe p é o mesmo que a classe de objeto, e

o nome de p é igual ou (no caso de *wildcards*) está implícito pelo nome desse objeto.

Por exemplo, "a.b.*" implica "a.b.c".

Substituições:

implies na classe BasicPermission.

37 Pacote com.sun.dtv.media.format

37.1 Descrição do pacote

Controle para o formato do vídeo.

NOTA Este pacote está especificado desde o Java DTV 1.0.

37.2 Índice de interfaces

ActiveFormatEvent

Um `VideoFormatEvent` que informa sobre alterações no formato ativo.

ArbitraryVideoScalingControl

Controle para recuperar os intervalos do fator de modificação do tamanho arbitrário do vídeo.

AspectRatioEvent

Um `VideoFormatEvent` que informa sobre alterações na razão de aspecto do *frame* de vídeo transmitido.

BackgroundVideoPresentationControl

Controle de vídeo para os players de plano de fundo.

ClippingControl

Controle para recuperar e configurar o retângulo de clip do vídeo.

DecoderFormatEvent

Evento para informar que a conversão de formato do decoder foi alterada.

VideoFormatControl

Fornece o meio para recuperar informações sobre o formato e a razão de aspecto do vídeo.

VideoFormatEvent

Evento para informar sobre alterações no formato do vídeo.

VideoFormatListener

Interface `Listener` para receber eventos sobre alterações na apresentação de vídeo.

VideoPresentationControl

Fornece meio para consultar e manipular a apresentação real do vídeo.

VideoPresentationListener

Interface `Listener` para receber eventos sobre alterações na apresentação do vídeo assim como todos os tipos de `ContollerEvent`.

37.3 Índice de classes

Transformation

Essa classe representa um container para informações de transformação para transformação de vídeo.

VideoPresentationEvent

Evento para informar sobre alterações na apresentação do vídeo.

37.4 Interface ActiveFormatEvent

37.4.1 Descrição da interface

com.sun.dtv.media.format

Todas as super-interfaces

`MediaEvent, VideoFormatEvent`

```
public interface ActiveFormatEvent
extends VideoFormatEvent
```

Um `VideoFormatEvent` que informa sobre alterações no formato ativo.

37.4.2 Índice de métodos

`int` **`getNewActiveFormat`**`()`

Fornece informações do novo formato ativo, deve ser um dos valores de constante definidos no `VideoFormatControl` para o formato ativo.

37.4.3 Detalhe dos métodos

getNewActiveFormat

```
int getNewActiveFormat()
```

Fornece informações do novo formato ativo, deve ser um dos valores de constante definidos no `VideoFormatControl` para o formato ativo.

Retorna:

novo valor de formato ativo

Relaciona-se com:

`VideoFormatControl`

37.5 Interface ArbitraryVideoScalingControl

37.5.1 Descrição da interface

com.sun.dtv.media.format

Todas as super-interfaces

`Control`

```
public interface ArbitraryVideoScalingControl
extends Control
```

Controle para recuperar os intervalos do fator de modificação do tamanho arbitrário do vídeo. Esse controle está disponível apenas se o redimensionamento artbitrário é suportado para o tipo de mídia pelo sistema.

37.5.2 Índice de métodos

`float[] **getArbitraryHorizontalScalingFactors**()`

Retorna dois valores flutuantes considerando o intervalo de redimensionamento horizontal.

`float[] **getArbitraryVerticalScalingFactors**()`

Retorna dois valores flutuantes considerando o intervalo de redimensionamento vertical.

37.5.3 Detalhe dos métodos

getArbitraryHorizontalScalingFactors

`float[] getArbitraryHorizontalScalingFactors()`

Retorna dois valores flutuantes considerando o intervalo de redimensionamento horizontal.

Retorna:

dois valores flutuantes considerando os fatores de redimensionamento horizontal máximo e mínimo.

getArbitraryVerticalScalingFactors

`float[] getArbitraryVerticalScalingFactors()`

Retorna dois valores flutuantes considerando o intervalo de redimensionamento vertical.

Retorna:

dois valores flutuantes considerando os fatores de redimensionamento vertical máximo e mínimo.

37.6 Interface AspectRatioEvent

37.6.1 Descrição da interface

`com.sun.dtv.media.format`

Todas as super-interfaces

`MediaEvent, VideoFormatEvent`

`public interface AspectRatioEvent`

`extends VideoFormatEvent`

Um `VideoFormatEvent` que informa sobre alterações na razão de aspecto do *frame* de vídeo transmitido.

37.6.2 Índice de métodos

`int **getNewAspectRatio**()`

Fornece a informação da nova razão de aspecto do *frame* de vídeo transmitido, deve ser um dos valores da constante definidos no `VideoFormatControl` para razão de aspecto.

37.6.3 Detalhe dos métodos

getNewAspectRatio

`int getNewAspectRatio()`

Fornece a informação da nova razão de aspecto do *frame* de vídeo transmitido, deve ser um dos valores da

constante definidos no `VideoFormatControl` para razão de aspecto.

Retorna:

novo valor de formato ativo

Relaciona-se com:

`VideoFormatControl`

37.7 Interface BackgroundVideoPresentationControl

37.7.1 Descrição da interface

`com.sun.dtv.media.format`

Todas as super-interfaces

`Control, VideoPresentationControl`

```
public interface BackgroundVideoPresentationControl
extends VideoPresentationControl
```

Controle de vídeo para os *players* de plano de fundo.

37.7.2 Índice de métodos

`Transformation` **`getTransformation`**`()`

Retorna a Transformação atual utilizada para exibir esse vídeo.

`Transformation` **`setTransformation`**`(Transformation t, boolean transform)`

Configura o objeto `Transformations`, o valor de retorno é o valor mais próximo possível para o sistema.

Métodos herdados da interface `com.sun.dtv.media.format.`**`VideoPresentationControl`**

```
addVideoPresentationListener, getActiveVideoOnScreenRectangle,
getActiveVideoRectangle, getActualVideoSize, getDecodedVideoSize,
getHorizontalScalingFactors, getInputSize, getTotalVideoRectangle,
getVerticalScalingFactors, removeVideoPresentationListener
```

37.7.3 Detalhe dos métodos

setTransformation

```
Transformation setTransformation(Transformation t,
                                 boolean transform)
```

Configura o objeto `Transformations`, o valor de retorno é o valor mais próximo possível para o sistema. Se a Transformação não é suportada, deve ser retornado valor `null`.

Parâmetros:

`t` – Objeto de transformação que define as informações de transformação para este vídeo.

`transform` – se verdadeiro a Transformação atual deve ser executada, se *false*, apenas os valores mais próximos devem ser retornados.

Retorna:

um objeto de Transformação representando o valor mais próximo ao objeto de Transformação que pode ser suportado pelo sistema ou `null` se essa Transformação não for suportada.

Relaciona-se com:

`getTransformation()`

getTransformation

`Transformation getTransformation()`

Retorna a Transformação atual utilizada para exibir esse vídeo.

Retorna:

retorna a Transformação atual

Relaciona-se com:

`setTransformation()`

37.8 Interface ClippingControl

37.8.1 Descrição da interface

`com.sun.dtv.media.format`

Todas as super-interfaces

`Control`

```
public interface ClippingControl
extends Control
```

Controle para recuperar e configurar o retângulo de clip do vídeo. Este controle está disponível apenas se o recorte for suportado pelo sistema.

37.8.2 Índice de métodos

`Rectangle` **`getRegion`**`()`

Esse método retorna um retângulo considerando a região de exibição do vídeo.

`Rectangle` **`setRegion`**`(Rectangle clippingRect)`

Configura a região do Retângulo para qual o vídeo deve ser recortado.

37.8.3 Detalhe dos métodos

getRegion

`Rectangle getRegion()`

Esse método retorna um retângulo considerando a região de exibição do vídeo. Se o recorte não for suportado, o retângulo retornado terá o mesmo tamanho do vídeo exibido.

Retorna:

retângulo do vídeo decodificado que deve ser exibido após amostragem como especificado em ETR 154, se aplicável.

Relaciona-se com:

`setRegion()`

setRegion

`Rectangle setRegion(Rectangle clippingRect)`

Configura a região do Retângulo para qual o vídeo deve ser recortado. O `clippingRect` é uma solicitação do aplicativo que o terminal deve configurar o retângulo de recorte para o valor técnico mais próximo possível.

Parâmetros:

`clippingRect` – região de retângulo de corte como um retângulo do vídeo de decodificação após amostragem ETR 154, se aplicável

Retorna:

retângulo de recorte selecionado pelo terminal. Caso o terminal não suporte recorte, deve ser retornado valor `null`.

Relaciona-se com:

`getRegion()`

37.9 Interface DecoderFormatEvent

37.9.1 Descrição da interface

`com.sun.dtv.media.format`

Todas as super-interfaces

`MediaEvent, VideoFormatEvent`

```
public interface DecoderFormatEvent
extends VideoFormatEvent
```

Evento para informar que a conversão de formato de decodificação foi alterada.

Relaciona-se com:

`VideoFormatControl`

37.9.2 Índice de métodos

`int` **`getNewDecoderFormat`**`()`

Retorna o novo formato de decodificação.

37.9.3 Detalhe dos métodos

getNewDecoderFormat

`int getNewDecoderFormat()`

Retorna o novo formato de decodificação.

Retorna:

formato de decodificação deve ser um dos valores de constante definidos `VideoFormatControl`.

37.10 Classe Transformation

37.10.1 Descrição da classe

com.sun.dtv.media.format

```
java.lang.Object
  └com.sun.dtv.media.format.Transformation
```

```
public class Transformation
extends Object
```

Essa classe representa um container para informações de transformação para transformação de vídeo. É utilizada como uma entrada ao `VideoFormatControl`. Recorte, modificação de tamanho e posição de vídeo. Todas as transformações devem ser aplicadas após *up-scaling* como especificado em ETR 154.

37.10.2 Índice de construtores

`Transformation``(Rectangle clippingRectangle, float horizontalScalingFactor, float verticalScalingFactor, Point position)`

Cria uma nova instância de uma `Transformation` para um *stream* de vídeo.

37.10.3 Índice de métodos

`float` **`getHorizontalScalingFactor`**`()`

Retorna o fator de redimensionamento horizontal.

`Point` **`getPosition`**`()`

Retorna a posição do vídeo.

`Rectangle` **`getRegion`**`()`

Esse método retorna um retângulo considerando a região de exibição do vídeo.

`float` **`getVerticalScalingFactor`**`()`

Retorna o fator de redimensionamento vertical.

37.10.4 Detalhe dos construtores

Transformation

```
public Transformation(Rectangle clippingRectangle,
                      float horizontalScalingFactor,
                      float verticalScalingFactor,
                      Point position)
```

Cria uma nova instância de uma Transformação para um stream de vídeo.

Parâmetros:

`clippingRectangle` – define o retângulo de recorte para o vídeo decodificado após amostragem de ETR 154, se aplicável; ou `null` se o recorte não for suportado

`horizontalScalingFactor` – fator de redimensionamento horizontal para este vídeo

`verticalScalingFactor` – fator de redimensionamento vertical para este vídeo

`position` – posição na tela para o vídeo.

37.10.5 Detalhe dos métodos

getRegion

```
public Rectangle getRegion()
```

Esse método retorna um retângulo considerando a região de exibição do vídeo. Se o recorte não for suportado, o valor retornado deve ser `null`.

Retorna:

retângulo do vídeo decodificado que deve ser exibido após amostragem ETR 154, se aplicável, ou `null` se o recorte não for suportado.

getHorizontalScalingFactor

```
public float getHorizontalScalingFactor()
```

Retorna o fator de redimensionamento horizontal.

Retorna:

fator de redimensionamento horizontal

getVerticalScalingFactor

```
public float getVerticalScalingFactor()
```

Retorna o fator de redimensionamento vertical.

Retorna:

fator de redimensionamento vertical

getPosition

```
public Point getPosition()
```

Retorna a posição do vídeo.

Retorna:

localização do vídeo no espaço coordenado da tela.

37.11 Interface VideoFormatControl

37.11.1 Descrição da interface

`com.sun.dtv.media.format`

Todas as super-interfaces

`Control`

```
public interface VideoFormatControl
extends Control
```

Fornece o meio para recuperar informações sobre o formato e a razão de aspecto do vídeo.

37.11.2 Índice de campos

`int` **`AF_149_CENTRAL`**

Constante representando uma descrição de formato ativo de 14:9 (central).

`int` **`AF_169_CENTRAL`**

Constante representando uma descrição de formato ativo de 16:9 (central).

`int` **`AF_169_SHOOT_149_CENTRAL`**

Constante representando uma descrição de formato ativo de 16:9 (com 14:9 central de shoot & protect).

`int` **`AF_169_SHOOT_43_CENTRAL`**

Constante representando uma descrição de formato ativo de 16:9 (com 4:3 central de shoot & protect).

`int` **`AF_43_CENTRAL`**

Constante representando uma descrição de formato ativo de 4:3 (central).

`int` **`AF_43_SHOOT_149_CENTRAL`**

Constante representando uma descrição de formato ativo de 4:3 (com 14:9 central de shoot & protect).

`int` **`AF_AS_THE_CODED_FRAME`**

Constante representando um formato ativo como frame codificado.

`int` **`AF_BOX_149_UPPER`**

Contante representando uma descrição de formato ativo de Box 14:9 (superior).

`int` **`AF_BOX_169_CENTRAL`**

Constante representando uma descrição de formato ativo de Box > 16:9 (central).

`int` **`AF_BOX_169_UPPER`**

Contante representando uma descrição de formato ativo de Box 16:9 (superior).

`int` **`AF_NOT_PRESENT`**

Constante representando uma descrição de formato ativo ausente.

`int` **`ASPECT_RATIO_169`**

Razão de aspecto de 16x9.

`int` **`ASPECT_RATIO_43`**

Razão de aspecto de 4x3.

`int` **`ASPECT_RATIO_NOT_PRESENT`**

Nenhuma razão de aspecto desconhecida.

`int` **`DF_CENTER`**

Constante representando uma descrição de formato de decoder de centro (apenas a parte central do vídeo de 16:9 deve ser exibida em um frame de 4:3).

`int` **`DF_FULL`**

Constante representando uma descrição de formato de decoder de tela cheia.

`int` **`DF_LETTERBOX`**

Constante representando uma descrição de formato de decoder de letterbox (o vídeo total 16:9 deve ser exibido na parte central das barras pretas do frame 4:3) na parte superior e inferior do frame.

`int` **`DF_NOT_PRESENT`**

Constante representando uma descrição de formato de decoder ausente.

`int` **`DF_PANSCAN`**

Constante representando uma descrição de formato de decoder de Pan and Scan (apenas a parte central do vídeo de 16:9 deve ser exibida em um frame de 4:3 utilizando vetores Pan and Scan).

`int` **`DF_SYSTEM`**

Constante representando que a descrição de formato de decoder é configurada pelo sistema.

`int` **`DF_ZOOM`**

Constante representando uma descrição de formato de decoder de zoom (apenas a parte central do vídeo de 4:3 deve ser exibida em um frame de 16:9).

37.11.3 Índice de métodos

`void` **`addVideoFormatListener`**`(VideoFormatListener listener)`

Adiciona um *listener* ao `VideoFormatControl`.

`int` **`getActiveFormatDefinition`**`()`

Dá o active_format da Descrição de formato de arquivo (ver ABNT NBR 15602-1) se presente como um dos valores de constante AFD definido.

`int` **`getAspectRatio`**`()`

Dá a razão de aspecto definida para a mídia sob o controle deste *player* conforme é exibida, ou seja,

`int` **`getDecoderFormatDefinition`**`()`

Dá a definição de formato real que é atualmente utilizada pelo decoder.

`int` **`getOriginalAspectRatio`**`()`

Dá a razão de aspecto original definida para a mídia (conforme foi transmitida).

`Transformation` **`getTransformation`**`(int decoderFormatConversion)`

Retorna a `Transformation` que descreve a determinada conversão de formato de decoder.

`void` **`removeVideoFormatListener`**`(VideoFormatListener listener)`

Remove um objeto `VideoFormatListener`.

37.11.4 Detalhe dos campos

ASPECT_RATIO_43

```
public static final int ASPECT_RATIO_43 = 43
```

Razão de aspecto de 4x3.

ASPECT_RATIO_169

```
public static final int ASPECT_RATIO_169 = 169
```

Razão de aspecto de 16x9.

ASPECT_RATIO_NOT_PRESENT

```
public static final int ASPECT_RATIO_NOT_PRESENT = 0
```

Nenhuma razão de aspecto desconhecida.

AF_NOT_PRESENT

```
public static final int AF_NOT_PRESENT = 0
```

Constante representando uma descrição de formato ativo ausente.

AF_BOX_169_UPPER

```
public static final int AF_BOX_169_UPPER = 16
```

Contante representando uma descrição de formato ativo de Box 16:9 (superior).

AF_BOX_149_UPPER

```
public static final int AF_BOX_149_UPPER = 17
```

Contante representando uma descrição de formato ativo de Box 14:9 (superior).

AF_BOX_169_CENTRAL

```
public static final int AF_BOX_169_CENTRAL = 256
```

Constante representando uma descrição de formato ativo de Box > 16:9 (central).

AF_AS_THE_CODED_FRAME

```
public static final int AF_AS_THE_CODED_FRAME = 4096
```

Constante representando um formato ativo como frame codificado.

AF_43_CENTRAL

```
public static final int AF_43_CENTRAL = 4097
```

Constante representando uma descrição de formato ativo de 4:3 (central).

AF_169_CENTRAL

```
public static final int AF_169_CENTRAL = 4112
```

Constante representando uma descrição de formato ativo de 16:9 (central).

AF_149_CENTRAL

```
public static final int AF_149_CENTRAL = 4113
```

Constante representando uma descrição de formato ativo de 14:9 (central).

AF_43_SHOOT_149_CENTRAL

```
public static final int AF_43_SHOOT_149_CENTRAL = 4353
```

Constante representando uma descrição de formato ativo de 4:3 (com 14:9 central de shoot & protect).

AF_169_SHOOT_149_CENTRAL

```
public static final int AF_169_SHOOT_149_CENTRAL = 4368
```

Constante representando uma descrição de formato ativo de 16:9 (com 14:9 central de shoot & protect).

AF_169_SHOOT_43_CENTRAL

```
public static final int AF_169_SHOOT_43_CENTRAL = 4369
```

Constante representando uma descrição de formato ativo de 16:9 (com 4:3 central de shoot & protect).

DF_NOT_PRESENT

```
public static final int DF_NOT_PRESENT = 0
```

Constante representando uma descrição de formato de decoder ausente.

DF_SYSTEM

```
public static final int DF_SYSTEM = 1
```

Constante representando que a descrição de formato de decoder é configurada pelo sistema.

DF_FULL

```
public static final int DF_FULL = 2
```

Constante representando uma descrição de formato de decoder de tela cheia.

DF_LETTERBOX

```
public static final int DF_LETTERBOX = 3
```

Constante representando uma descrição de formato de decoder de letterbox (o vídeo total 16:9 deve ser exibido na parte central das barras pretas do frame 4:3) na parte superior e inferior do frame.

DF_ZOOM

```
public static final int DF_ZOOM = 4
```

Constante representando uma descrição de formato de decoder de zoom (apenas a parte central do vídeo de 4:3 deve ser exibida em um frame de 16:9).

DF_CENTER

```
public static final int DF_CENTER = 5
```

Constante representando uma descrição de formato de decoder de centro (apenas a parte central do vídeo de 16:9 deve ser exibida em um frame de 4:3).

DF_PANSCAN

```
public static final int DF_PANSCAN = 6
```

Constante representando uma descrição de formato de decoder de Pan and Scan (apenas a parte central do

vídeo de 16:9 deve ser exibida em um frame de 4:3 utilizando vetores Pan and Scan).

37.11.5 Detalhe dos métodos

getAspectRatio

```
int getAspectRatio()
```

Dá a razão de aspecto definida para a mídia sob o controle deste Player conforme é exibida, por exemplo depois que todas as informações tenham sido aplicadas a ela.

Retorna:

Um dos valores de constante `ASPECT_RATIO_*` definida nesta interface que indica a razão de aspecto de mídia conforme é exibida.

getOriginalAspectRatio

```
int getOriginalAspectRatio()
```

Dá a razão de aspecto original definida para a mídia (conforme foi transmitida).

Retorna:

Um dos valores de constante `ASPECT_RATIO_*` definida nesta interface que indica a razão de aspecto de mídia conforme transmitida originalmente.

getActiveFormatDefinition

```
int getActiveFormatDefinition()
```

Dá o active_format da Descrição de formato de arquivo (no ABNT NBR 15602-1:2007) se presente como um dos valores de constante AFD definido.

Retorna:

um dos valores de constante AF_

getDecoderFormatDefinition

```
int getDecoderFormatDefinition()
```

Dá a definição de formato real que é atualmente utilizada pelo decoder.

Retorna:

um dos valores de constante DF_

getTransformation

```
Transformation getTransformation(int decoderFormatConversion)
```

Retorna a Transformação que descreve a determinada conversão de formato de decoder.

Parâmetros:

`decoderFormatConversion` - um dos valores de constante DF_

Retorna:

Transformação para o `deciderFormatConversion` dado (uma das constantes DF) ou `null` se essa conversão

de formato for inválida.

addVideoFormatListener

`void addVideoFormatListener(VideoFormatListener listener)`

Adiciona um *listener* ao `VideoFormatControl`.

Parâmetros:

`listener` – objeto implementando a interface `Listener` a ser adicionada para o controle.

Relaciona-se com:

`removeVideoFormatListener()`

removeVideoFormatListener

`void removeVideoFormatListener(VideoFormatListener listener)`

Remove um objeto `VideoFormatListener`.

Parâmetros:

`listener` – objeto a ser removido do controle.

Relaciona-se com:

`addVideoFormatListener()`

37.12 Interface VideoFormatEvent

37.12.1 Descrição da interface

`com.sun.dtv.media.format`

Todas as super-interfaces

`MediaEvent`

Todas as subinterfaces conhecidas

`ActiveFormatEvent, AspectRatioEvent, DecoderFormatEvent`

```
public interface VideoFormatEvent
extends MediaEvent
```

Evento para informar sobre alterações no formato do vídeo.

37.13 Interface VideoFormatListener

37.13.1 Descrição da interface

`com.sun.dtv.media.format`

```
public interface VideoFormatListener
```

Interface `Listener` para receber eventos sobre alterações na apresentação de vídeo.

37.13.2 Índice de métodos

`void` **`videoFormatChange`**`(VideoFormatEvent event)`

Notificado quando o formato de vídeo muda.

37.13.3 Detalhe dos métodos

videoFormatChange

`void videoFormatChange(VideoFormatEvent event)`

Notificado quando o formato de vídeo muda.

Parâmetros:

`event` – sobre alteração de apresentação de vídeo

37.14 Interface VideoPresentationControl

37.14.1 Descrição da interface

`com.sun.dtv.media.format`

Todas as super-interfaces

Control

Todas as subinterfaces conhecidas

`BackgroundVideoPresentationControl`

`public interface VideoPresentationControl`

`extends Control`

Fornece meio para consultar e manipular a apresentação real do vídeo.

37.14.2 Índice de métodos

`void` **`addVideoPresentationListener`**`(VideoPresentationListener listener)`

Adiciona um *listener* ao `VideoPresentationControl`.

`Rectangle` **`getActiveVideoOnScreenRectangle`**`()`

Retorna o retângulo do vídeo ativo na tela, *letterboxing* ou *pillarboxing* não são parte do retângulo.

`Rectangle` **`getActiveVideoRectangle`**`()`

Retorna o retângulo que cobre a área de vídeo ativo, *letterboxing* ou *pillarboxing* não são parte do retângulo.

`Dimension` **`getActualVideoSize`**`()`

Este método permite que o aplicativo investigue a dimensão real do vídeo apresentado.

`Dimension` **`getDecodedVideoSize`**`()`

Retorna a dimensão de tamanho do vídeo conforme é apresentado ao usuário.

`float[]` **`getHorizontalScalingFactors`**`()`

Retorna os valores dos fatores de redimensionamento horizontal distintos suportados caso não seja suportado nenhum redimensionamento horizontal arbitrário.

`Dimension` **`getInputSize`**`()`

Retorna a dimensão de vídeo antes de qualquer redimensionamento, recorte e decodificação.

`Rectangle` **`getTotalVideoRectangle`**`()`

Este método retorna o retângulo todo usado para o vídeo no stream de transmissão incluindo barras para *letterboxing* e *pillarboxing* (exceto `getActiveVideoRectangle`).

`float[]` **`getVerticalScalingFactors`**`()`

Retorna os valores dos fatores de redimensionamento vertical distintos suportados caso não seja suportado nenhum redimensionamento vertical arbitrário.

`void` **`removeVideoPresentationListener`**`(VideoPresentationListener listener)`

Remove um objeto `VideoFormatListener`.

37.14.3 Detalhe dos métodos

getActiveVideoRectangle

`Rectangle getActiveVideoRectangle()`

Retorna o retângulo que cobre a área de vídeo ativo, letterboxing ou pillarboxing não são parte do Retângulo. O retângulo considera redimensionamento real. O retângulo pode ser menor ou mesmo maior que a tela, e pode haver um deslocamento. Esse deslocamento pode ainda ser negativo, se o canto superior esquerdo do vídeo ativo estiver à esquerda e/ou topo do canto esquerdo superior da tela.

Retorna:

retângulo do vídeo real

getActiveVideoOnScreenRectangle

`Rectangle getActiveVideoOnScreenRectangle()`

Retorna o retângulo do vídeo ativo na tela, *letterboxing* ou *pillarboxing* não são parte do retângulo. O retângulo considera redimensionamento real. Como o retângulo descreve somente a parte do vídeo ativo na tela (exceto `getActiveVideoRectangle`), ele pode ser menor que a tela, e – se houver deslocamento – esse deslocamento é sempre positivo.

Retorna:

retângulo de vídeo na tela

getTotalVideoRectangle

`Rectangle getTotalVideoRectangle()`

Este método retorna o retângulo todo usado para o vídeo no stream de transmissão incluindo barras para *letterboxing* e *pillarboxing* (exceto `getActiveVideoRectangle`). Qualquer redimensionamento é considerado. O retângulo pode ser menor ou mesmo maior que a tela, e pode haver um deslocamento. Esse deslocamento pode ainda ser negativo, se o canto superior esquerdo do vídeo ativo estiver à esquerda e/ou topo do canto esquerdo superior da tela.

Retorna:

retângulo do vídeo na tela com letterbox e pillarbox inclusos.

getInputSize

`Dimension getInputSize()`

Retorna a dimensão de vídeo antes de qualquer redimensionamento, recorte e decodificação.

Retorna:

dimensão de vídeo decodificado

getDecodedVideoSize

```
Dimension getDecodedVideoSize()
```

Retorna a dimensão de tamanho do vídeo conforme é apresentado ao usuário. Exceto para `getInputSize`, o resultado considera a decodificação. Redimensionamento e recorte ainda não são considerados.

Retorna:

dimensão do vídeo da forma que é apresentado para o usuário

getActualVideoSize

```
Dimension getActualVideoSize()
```

Este método permite que o aplicativo investigue a dimensão real do vídeo apresentado. Exceto para `getDecodedVideoSize`, o resultado considera não apenas decodificação, mas também efeitos como recorte e redimensionamento.

Retorna:

dimensão do vídeo apresentado

getHorizontalScalingFactors

```
float[] getHorizontalScalingFactors()
```

Retorna os valores dos fatores de redimensionamento horizontal distintos suportados caso não seja suportado nenhum redimensionamento horizontal arbitrário.

Retorna:

array de valores flutuantes considerando fatores de redimensionamento horizontal, incluindo o fator 1, se nenhum redimensionamento horizontal arbitrário for suportado, caso contrário `null`.

getVerticalScalingFactors

```
float[] getVerticalScalingFactors()
```

Retorna os valores dos fatores de redimensionamento vertical distintos suportados caso não seja suportado nenhum redimensionamento vertical arbitrário.

Retorna:

array de valores flutuantes considerando fatores de redimensionamento vertical, incluindo o fator 1, se nenhum redimensionamento vertical arbitrário for suportado, caso contrário `null`

addVideoPresentationListener

```
void addVideoPresentationListener(VideoPresentationListener listener)
```

Adiciona um *listener* ao `VideoPresentationControl`.

Parâmetros:

`listener` – objeto implementando a interface `Listener` a ser adicionada para o controle.

Relaciona-se com:

`removeVideoPresentationListener()`

removeVideoPresentationListener

`void removeVideoPresentationListener(VideoPresentationListener listener)`

Remove um objeto `VideoFormatListener`.

Parâmetros:

`listener` – objeto a ser removido do controle.

Relaciona-se com:

`addVideoPresentationListener()`

37.15 Classe VideoPresentationEvent

37.15.1 Descrição da classe

`com.sun.dtv.media.format`

```
java.lang.Object
  └java.util.EventObject
      └javax.media.ControllerEvent
          └com.sun.dtv.media.format.VideoPresentationEvent
```

Todas as interfaces implementadas

`MediaEvent, Serializable`

```
public class VideoPresentationEvent
extends ControllerEvent
```

Evento para informar sobre alterações na apresentação do vídeo.

37.15.2 Índice de campos

`static int` **`PRESENTATION_CHANGED`**

Constante indicando que a apresentação foi alterada, mas ainda está disponível.

`static int` **`STREAM_IS_CA_PROTECTED`**

Constante indicando que o stream está protegido.

`static int` **`STREAM_IS_UNAVAILABLE`**

Constante indicando que o stream não está disponível.

37.15.3 Índice de construtores

`VideoPresentationEvent``(Controller controller, int type)`

Construtor para `VideoPresentationEvent`, o `Controller` é a origem do evento.

37.15.4 Índice de métodos

`Controller` **`getController`**`()`

Retorna o `Controller` que for a origem do evento.

`int` **`getEventType`**`()`

Retorna o topo de evento que aconteceu durante a apresentação do stream.

37.15.5 Detalhe dos campos

STREAM_IS_CA_PROTECTED

```
public static final int STREAM_IS_CA_PROTECTED = 1
```

Constante indicando que o stream está protegido.

STREAM_IS_UNAVAILABLE

```
public static final int STREAM_IS_UNAVAILABLE = 2
```

Constante indicando que o stream não está disponível.

PRESENTATION_CHANGED

```
public static final int PRESENTATION_CHANGED = 3
```

Constante indicando que a apresentação foi alterada, mas ainda está disponível.

37.15.6 Detalhe dos construtores

VideoPresentationEvent

```
public VideoPresentationEvent(Controller controller,
                              int type)
```

Construtor para `VideoPresentationEvent`, o `Controller` é a origem do evento.

Parâmetros:

`controller` – origem do evento

`type` – tipo de evento, DEVE ser um dos valores de número inteiro predefinidos nesta classe

37.15.7 Detalhe dos métodos

getController

```
public Controller getController()
```

Retorna o `Controller` que for a origem do evento.

Retorna:

`Controller` que criou o evento

getEventType

```
public int getEventType()
```

Retorna o topo de evento que aconteceu durante a apresentação do stream. DEVE ser um dos valores de número

inteiro predefinidos nesta classe.

Retorna:

tipo de evento

37.16 Interface VideoPresentationListener

37.16.1 Descrição da interface

com.sun.dtv.media.format

Todas as super-interfaces

`ControllerListener, EventListener`

`public interface VideoPresentationListener`

`extends ControllerListener`

Interface `Listener` para receber eventos sobre alterações na apresentação do vídeo assim como todos os tipos de `ContollerEvent`.

37.16.2 Índice de métodos

`void` **`videoPresentationChange`**`(VideoPresentationEvent event)`

Notificado quando a apresentação do vídeo muda.

37.16.3 Detalhe dos métodos

videoPresentationChange

`void videoPresentationChange(VideoPresentationEvent event)`

Notificado quando a apresentação do vídeo muda.

Parâmetros:

`event` – sobre alteração de apresentação de vídeo

38 Pacote com.sun.dtv.media.language

38.1 Descrição do pacote

Provê a funcionalidade básica de informar-se sobre e definir a língua do áudio, das legendas e do *closed caption*.

Fornece a funcionalidade básica para investigar e configurar o idioma para um áudio ou controle de legenda. Todos os idiomas são codificados com três letras de acordo com a ISO 639-2 e ABNT NBR 15603-2 para uso em descritores de serviço.

NOTA Este pacote está especificado desde o Java DTV 1.0.

38.2 Índice de interfaces

LanguageControl

Controle para investigar as linguagens suportadas e configurar um idioma específico para um player.

38.3 Índice de exceções

LanguageNotSupportedException

Exceção executada quando um idioma solicitado não é suportado.

38.4 Interface LanguageControl

38.4.1 Descrição da interface

`com.sun.dtv.media.language`

Todas as super-interfaces

`Control`

Todas as subinterfaces conhecidas

`AudioControl, OverlayTextControl`

```
public interface LanguageControl
extends Control
```

Controle para investigar as linguagens suportadas e configurar um idioma específico para um player. Idiomas devem ser codificados com abreviações de três letras definidas pela ISO 639-2 e especificadas na ABNT NBR 15603-2 para uso em descritores de serviço.

Relaciona-se com:

`Player`

38.4.2 Índice de métodos

`String` **`getDefaultLanguage`**`()`

Retorna o idioma-padrão de sinalização de stream/rede.

`String` **`getLanguage`**`()`

Fornece o idioma selecionado atualmente para o aplicativo.

`String[]` **`getSupportedLanguages`**`()`

Fornece uma lista de todos os idiomas do serviço atual para o aplicativo.

`boolean` **`setLanguage`**`(String language)`

Configura o idioma para o player, substitui todos os idiomas previamente selecionados.

38.4.3 Detalhe dos métodos

getSupportedLanguages

`String[] getSupportedLanguages()`

Fornece uma lista de todos os idiomas do serviço atual para o aplicativo. Se nenhum idioma é suportado, o *array* de idioma possui comprimento zero

Retorna:

Array com todos os idiomas suportados ou `null` se nenhum idioma for suportado

setLanguage

```
boolean setLanguage(String language)
             throws LanguageNotSupportedException,
                    IllegalArgumentException
```

Configura o idioma para o player, substitui todos os idiomas previamente selecionados.

Parâmetros:

`language` – para ser utilizado para áudio ou Legendas codificados em *string* de três letras

Retorna:

`true` se a configuração de idioma for bem-sucedida, *`false`* caso ocorra o contrário

Lança:

`LanguageNotSupportedException` – caso esteja instalado um idioma não seja suportado pelo sistema

`IllegalArgumentException` – se a *string* de idioma não apresentar três caracteres

Relaciona-se com:

`getLanguage()`

getLanguage

```
String getLanguage()
```

Fornece o idioma selecionado atualmente para o aplicativo.

Retorna:

código de idioma selecionado atualmente ou `null` se nenhum idioma for selecionado

Relaciona-se com:

`setLanguage()`

getDefaultLanguage

```
String getDefaultLanguage()
```

Retorna o idioma-padrão de sinalização de stream/rede.

Retorna:

código de idioma padrão do sistema ou `null` se não houver nenhum idioma-padrão

38.5 Classe LanguageNotSupportedException

38.5.1 Descrição da classe

`com.sun.dtv.media.language`

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └javax.media.MediaException
              └com.sun.dtv.media.language.LanguageNotSupportedException
```

Todas as interfaces implementadas

`Serializable`

```
public class LanguageNotSupportedException
extends MediaException
```

Exceção executada quando um idioma solicitado não é suportado.

38.5.2 Índice de construtores

`LanguageNotSupportedException``()`

Construtor vazio para essa exceção.

`LanguageNotSupportedException``(String reason)`

Construtor para exceção de idioma com um código razoável.

38.5.3 Detalhe dos construtores

LanguageNotSupportedException

```
public LanguageNotSupportedException()
```

Construtor vazio para essa exceção.

LanguageNotSupportedException

```
public LanguageNotSupportedException(String reason)
```

Construtor para exceção de idioma com um código razoável.

Parâmetros:

`reason` – indicando a razão para a exceção.

39 Pacote com.sun.dtv.media.text

39.1 Descrição do pacote

Este pacote permite o controle das legendas e do *closed caption*.

Legendas são versões textuais do diálogo para cinema e televisão seja no mesmo idioma ou traduzidos para outros idiomas. Consulte para mais informações ARIB STD-B24 Versão 5.1-E1 Parte 3 “Codificação de legendas e conteúdos sobrepostos”.

Funcionalidades fornecida neste pacote:

- consultar os idiomas selecionáveis associados ao componente do serviço
- selecionar o idioma associado ao componente do serviço
- permitir que um aplicativo registre-se para notificações quando o idioma selecionado para determinado componente do serviço sofra mudanças

NOTA Este pacote está especificado desde o Java DTV 1.0.

39.2 Índice de interfaces

OverlayTextControl

Interface para fornecer controle para “Legendas” e "Closed Caption".

OverlayTextEvent

Eventos que reportam mudanças no `OverlayText, Subtitle` ou `CloseCaption`.

OverlayTextListener

Interface `Listener` para receber eventos específicos de `OverlayText`.

39.3 Interface OverlayTextControl

39.3.1 Descrição da interface

`com.sun.dtv.media.text`

Todas as super-interfaces

`Control, LanguageControl`

```
public interface OverlayTextControl
extends LanguageControl
```

Interface para fornecer controle para “Legendas” e "Closed Caption". Essa interface genérica permite manipular e pedir informações de status sobre dois tipos de overlay ou texto sobreposto. Deve estar disponível quando as informações de `OverlayText` estiverem presentes na mídia.

39.3.2 Índice de métodos

`void` **`addOverlayTextListener`**`(OverlayTextListener overlayTextListener)`

Adiciona uma interface `OverlayTextListener`.

`boolean` **`isTextAvailable`**`()`

Solicita se os textos *overlay* estiverem disponíveis no meio.

`boolean` **`isTextOn`**`()`

Retorna o estado atual da apresentação do `OverlayText`.

`void` **`removeOverlayTextListener`**`(OverlayTextListener overlayTextListener)`

Remove uma interface `OverlayTextListener`.

`void` **`switchTextOff`**`()`

Desliga texto overlay.

`void` **`switchTextOn`**`()`

Liga texto *overlay*.

Métodos herdados da interface `com.sun.dtv.media.language.`**`LanguageControl`**

`getDefaultLanguage, getLanguage, getSupportedLanguages, setLanguage`

39.3.3 Detalhe dos métodos

switchTextOn

`void switchTextOn()`

Liga texto *overlay*.

switchTextOff

`void switchTextOff()`

Desliga texto *overlay*.

isTextAvailable

`boolean isTextAvailable()`

Solicita se os textos *overlay* estiverem disponíveis no meio.

Retorna:

true se as legendas estiverem disponíveis.

isTextOn

`boolean isTextOn()`

Retorna o estado atual da apresentação do `OverlayText`.

Retorna:

`true` se o texto *overlay* estiver presente, *`false`* se não houver texto *overlay*.

addOverlayTextListener

`void addOverlayTextListener(OverlayTextListener overlayTextListener)`

Adiciona uma interface `OverlayTextListener`.

Parâmetros:

`overlayTextListener` - interface que receberá todos os eventos de texto *overlay*.

Relaciona-se com:

`removeOverlayTextListener()`

removeOverlayTextListener

`void removeOverlayTextListener(OverlayTextListener overlayTextListener)`

Remove uma interface `OverlayTextListener`.

Parâmetros:

`overlayTextListener` – `OverlayTextListener` para remover.

Relaciona-se com:

`addOverlayTextListener()`

39.4 Interface OverlayTextEvent

39.4.1 Descrição da interface

`com.sun.dtv.media.text`

Todas as super-interfaces

`MediaEvent`

```
public interface OverlayTextEvent
extends MediaEvent
```

Eventos que reportam mudanças no `OverlayText`, `Subtitle` ou `CloseCaption`.

39.4.2 Índice de campos

`int` **`OVERLAY_TEXT_AVAILABLE`**

Tipo de evento de `Overlay Text` disponível.

`int` **`OVERLAY_TEXT_DESELECTED`**

Tipo de evento de `Overlay Text` não-selecionado.

`int` **`OVERLAY_TEXT_LANGUAGE_CHANGE`**

Tipo de evento de alteração de idioma de `Overlay Text`.

`int` **`OVERLAY_TEXT_NOT_AVAILABLE`**

Tipo de evento de `Overlay Text` indisponível.

`int` **`OVERLAY_TEXT_SELECTED`**

Tipo de evento de `Overlay Text` selecionado.

39.4.3 Índice de métodos

`int` **`getEventType`**`()`

O retorno do tipo de evento deve ser um dos valores de constante `OVERLAY_TEXT__*` definido nesta interdace.

`Player` **`getPlayer`**`()`

Retorna o *player* JMF que for a fonte do evento.

39.4.4 Detalhe dos campos

OVERLAY_TEXT_AVAILABLE

```
public static final int OVERLAY_TEXT_AVAILABLE = 11
```

Tipo de evento de `Overlay Text` disponível.

OVERLAY_TEXT_DESELECTED

```
public static final int OVERLAY_TEXT_DESELECTED = 12
```

Tipo de evento de `Overlay Text` não-selecionado.

OVERLAY_TEXT_SELECTED

```
public static final int OVERLAY_TEXT_SELECTED = 13
```

Tipo de evento de `Overlay Text` selecionado.

OVERLAY_TEXT_LANGUAGE_CHANGE

```
public static final int OVERLAY_TEXT_LANGUAGE_CHANGE = 14
```

Tipo de evento de alteração de idioma de `Overlay Text`.

OVERLAY_TEXT_NOT_AVAILABLE

```
public static final int OVERLAY_TEXT_NOT_AVAILABLE = 15
```

Tipo de evento de `Overlay Text` indisponível.

39.4.5 Detalhe dos métodos

getPlayer

```
Player getPlayer()
```

Retorna o *player* JMF que for a fonte do evento.

Retorna:

Player origem do evento.

getEventType

```
int getEventType()
```

O retorno do tipo de evento deve ser um dos valores de constante `OVERLAY_TEXT__*` definido nesta interdace.

Retorna:

tipo de evento.

39.5 Interface OverlayTextListener

39.5.1 Descrição da interface

`com.sun.dtv.media.text`

Todas as super-interfaces

```
EventListener
```

```
public interface OverlayTextListener
extends EventListener
```

Interface `Listener` para receber eventos específicos de `OverlayText`.

39.5.2 Índice de métodos

`void` **`overlayTextEvent`**`(OverlayTextEvent overlayTextEvent)`

Recebe eventos de *closed caption*.

39.5.3 Detalhe dos métodos

overlayTextEvent

```
void overlayTextEvent(OverlayTextEvent overlayTextEvent)
```

Recebe eventos de *closed caption*.

Parâmetros:

`overlayTextEvent` – evento que indica a ocorrência de uma alteração no `Overlay Text`.

40 Pacote com.sun.dtv.media.timeline

40.1 Descrição do pacote

Este pacote permite o controle da linha de tempo da mídia. Permite configurar e recuperar o tempo de mídia e notificação sobre eventos de tempo na Mídia.

NOTA Este pacote está especificado desde o Java DTV 1.0.

40.2 Índice de interfaces

MediaTimeChangeEvent

Este evento é criado quando o tempo de mídia é alterado por meio do `MediaTimePositionControl`.

MediaTimeChangeListener

Interface `Listener` para receber eventos de alteração de tempo de mídia.

MediaTimeEvent

Evento para informar o aplicativo sobre alterações de tempo de mídia.

MediaTimeListener

Interface `Listener` para receber eventos de tempo de mídia.

MediaTimePositionControl

Controle para configurar e recuperar o tempo de mídia para mídia do stream, a duração máxima de mídia pode ser recuperada a partir da interface se ela é implementada pelo tipo de mídia.

40.3 Interface MediaTimeChangeEvent

40.3.1 Descrição da interface

`com.sun.dtv.media.timeline`

Todas as super-interfaces

`MediaEvent`

```
public interface MediaTimeChangeEvent
extends MediaEvent
```

Este evento é criado quando o tempo de mídia é alterado por meio do `MediaTimePositionControl`.

Relaciona-se com:

`MediaTimePositionControl`

40.3.2 Índice de métodos

`Time` **`getMediaChangeTime`**`()`

O tempo de mídia dado ao objeto `MediaTimeChangeEvent` carrega o tempo em que o evento foi, na verdade, criado pelo Controller.

40.3.3 Detalhe dos métodos

getMediaChangeTime

`Time getMediaChangeTime()`

O tempo de mídia dado ao objeto `MediaTimeChangeEvent` carrega o tempo em que o evento foi, na verdade, criado pelo Controller.

Retorna:

tempo de mídia real, o evento de alteração foi criado pelo *player*

40.4 Interface MediaTimeChangeListener

40.4.1 Descrição da interface

`com.sun.dtv.media.timeline`

Todas as super-interfaces

`ControllerListener`, `EventListener`

```
public interface MediaTimeChangeListener
extends ControllerListener
```

Interface `Listener` para receber eventos de alteração de tempo de mídia.

40.4.2 Índice de métodos

`void` **`controllerUpdate`**`(MediaTimeChangeEvent mediaTimeChangeEvent)`

Notificação de evento de alteração de tempo de mídia.

40.4.3 Detalhe dos métodos

controllerUpdate

`void controllerUpdate(MediaTimeChangeEvent mediaTimeChangeEvent)`

Notificação de evento de alteração de tempo de mídia.

Parâmetros:

`mediaTimeChangeEvent` – evento de alteração de tempo de mídia.

40.5 Interface MediaTimeEvent

40.5.1 Descrição da interface

`com.sun.dtv.media.timeline`

Todas as super-interfaces

`MediaEvent`

```
public interface MediaTimeEvent
```

extends MediaEvent

Evento para informar o aplicativo sobre alterações de tempo de mídia. Um MediaTimeEvent é registrado com um identificador e um tempo de mídia para um Controller. O Controller criará o evento o mais próximo possível do tempo dado durante o registro do evento. Esta interdace deve ser implementada por uma subclasse javax.media.ControllerEvent.

Relaciona-se com:

ControllerEvent

40.5.2 Índice de métodos

int **getEventID**()

Todo MediaTimeEvent é identificado por um identificador que é atribuída ao evento durante o registro da interface Listener com um tempo de mídia específico.

Time **getMediaTime**()

O tempo de mídia dado ao objeto MediaTimeEvent pode ser diferente do tempo em que o evento foi registrado para o verdadeiro objeto Time que carrega o tempo em que o evento foi realmente criado pelo Controller.

40.5.3 Detalhe dos métodos

getEventID

int getEventID()

Todo MediaTimeEvent é identificado por um identificador que é atribuído ao evento durante o registro da interface Listener com um tempo de mídia específico.

Retorna:

Identificador do evento

getMediaTime

Time getMediaTime()

O tempo de mídia dado ao objeto MediaTimeEvent pode ser diferente do tempo em que o evento foi registrado para o verdadeiro objeto Time que carrega o tempo em que o evento foi realmente criado pelo Controller.

Retorna:

tempo de mídia real, o evento foi criado pelo *player*.

40.6 Interface MediaTimeListener

40.6.1 Descrição da interface

com.sun.dtv.media.timeline

Todas as super-interfaces

ControllerListener, EventListener

public interface MediaTimeListener

extends ControllerListener

Interface Listener para receber eventos de tempo de mídia.

40.6.2 Índice de métodos

`void` **`controllerUpdate`**`(MediaTimeEvent mediaTimeEvent)`

Notificação de evento de tempo de mídia.

40.6.3 Detalhe dos métodos

controllerUpdate

`void controllerUpdate(MediaTimeEvent mediaTimeEvent)`

Notificação de evento de tempo de mídia.

Parâmetros:

`mediaTimeEvent` – evento de mídia.

40.7 Interface MediaTimePositionControl

40.7.1 Descrição da interface

`com.sun.dtv.media.timeline`

Todas as super-interfaces

`Control`

```
public interface MediaTimePositionControl
extends Control
```

Controle para configurar e recuperar o tempo de mídia para mídia do stream, a duração máxima de mídia pode ser recuperada a partir da interface se ela é implementada pelo tipo de mídia.

Relaciona-se com:

Interface `Duration`, classe `Time`

40.7.2 Índice de métodos

`void` **`addMediaTimeChangeListener`**`(MediaTimeChangeListener listener)`

Adiciona um `MediaTimeChangeListener` ao controle, o player criará um evento se o tempo de mídia for alterado com o método `setMediaTimePosition.`

`int` **`addMediaTimeListener`**`(MediaTimeListener listener, Time time)`

Adiciona um `MediaTimeListener` ao controle, o player criará um evento se a mídia atingir o dado `Time`, o evento terá o identificador que é devolvida durante o processo de registro para fácil identificação e gerenciamento no nível do aplicativo. O sistema é responsável por criar o evento o mais próximo possível ao dado `Time`.

`Time` **`getMediaTimePosition`**`()`

Retorna o mediaTime da mídia atualmente manipulada pelo Player.

`void` **`removeMediaTimeChangeListener`**`(MediaTimeChangeListener listener)`

Remove um `MediaTimeChangeListener` de um player.

`boolean` **`removeMediaTimeEventID`**`(int ID)`

Remove o `ID` especificado do sistema.

`void` **`removeMediaTimeListener`**`(MediaTimeListener listener)`

Remove um `MediaTimeListener` de um player e todos os dados associados (tempos e identificadores) do sistema.

`Time` **`setMediaTimePosition`**`(Time mediaTime)`

Configura o tempo de mídia o mais próximo possível ao tempo de mídia solicitado.

40.7.3 Detalhe dos métodos

setMediaTimePosition

```
Time setMediaTimePosition(Time mediaTime)
```

Configura o tempo de mídia o mais próximo possível ao tempo de mídia solicitado. Permite retroceder e avançar na mídia. A chamada deste método deve criar um `MediaTimeChangeEvent` quando a ação é executada.

Parâmetros:

`mediaTime` – Tempo de mídia desejado.

Retorna:

posição no tempo em que foi realmente configurado.

Relaciona-se com:

`getMediaTimePosition()`

getMediaTimePosition

```
Time getMediaTimePosition()
```

Retorna o tempo de mídia da mídia atualmente manipulada pelo *player*.

Retorna:

tempo de mídia atual da mídia.

Relaciona-se com:

`setMediaTimePosition()`

addMediaTimeListener

```
int addMediaTimeListener(MediaTimeListener listener,
                         Time time)
```

Adiciona um `MediaTimeListener` ao controle, o player criará um evento se a mídia atingir o dado `Time`, o evento terá o identificador que é devolvido durante o processo de registro para fácil identificação e gerenciamento no nível do aplicativo. O sistema é responsável por criar o evento o mais próximo possível ao dado `Time`.

Parâmetros:

`listener` - *listener* a ser adicionado.

`time` – quando o player deve criar o evento.

Retorna:

identificador de um identificador fornecida pelo controle para identificar o evento.

Relaciona-se com:

`removeMediaTimeListener()`

removeMediaTimeListener

`void removeMediaTimeListener(MediaTimeListener listener)`

Remove um `MediaTimeListener` de um player e todos os dados associados (tempos e identificadores) do sistema.

Parâmetros:

`listener` – a ser removido do sistema.

Relaciona-se com:

`addMediaTimeListener()`

removeMediaTimeEventID

`boolean removeMediaTimeEventID(int ID)`

Remove o `ID` especificado do sistema. Deve ser um identificador dado como valor de retorno do `addMediaTimeListener`

Parâmetros:

`ID` – do evento que deve ser removido do sistema.

Retorna:

true se a remoção for bem-sucedida, *false* se o identificador não estiver no sistema.

addMediaTimeChangeListener

`void addMediaTimeChangeListener(MediaTimeChangeListener listener)`

Adiciona um `MediaTimeChangeListener` ao controle, o player criará um evento se o tempo de mídia for alterado com o método `setMediaTimePosition`.

Parâmetros:

`listener` - *listener* a ser adicionado.

Relaciona-se com:

`setMediaTimePosition()`, `removeMediaTimeListener()`

removeMediaTimeChangeListener

`void removeMediaTimeChangeListener(MediaTimeChangeListener listener)`

Remove um `MediaTimeChangeListener` de um player.

Parâmetros:

`listener` – a ser removido do sistema.

Relaciona-se com:

`addMediaTimeChangeListener()`

41 Pacote com.sun.dtv.net

41.1 Descrição do pacote

Este pacote estende o pacote `java.net` com amplos controles de comunicação de dispositivos.

Aplicativos são ferramentas dadas para controlar os meios de comunicação que estão disponíveis na plataforma. A partir disso, um aplicativo, quando recebe o direito de acesso correspondente, pode descobrir os diferentes dispositivos de rede disponíveis, exemplificar uma conexão à Internet ou um número de serviço. Tal acesso pode exigir parâmetros de conexão específicos (por exemplo, nome de usuário, senha, parametrização de dispositivo e de rede, rede...) que estão disponíveis por meio deste pacote.

`NetworkDevice`: representa uma instância física de um dispositivo de rede na plataforma e é, portanto, manipulada como recurso escasso. Uma vez que o aplicativo tenha sido capaz de reservar tal instância, que então em troca pode controlar quando conectar ou desconectar e qual perfil na preferência do usuário utilizar para a conexão.

`NetworkDeviceStatusListener`: informa o aplicativo sobre o status da conexão e reserva do recurso de rede;

`NetworkDevicePermission`: permite controlar o tipo do acesso para um dado recurso de rede. Atualmente a permissão define filtros de discagem para permitir um controle de custo meticuloso ao discar linhas PSTN.

O dispositivo de rede, sendo um recurso escasso, destaca o sistema de permissão definido por ele (consulte `ScarceResourcePermission`) em que um aplicativo pode receber direitos de `reserve` e `force`.

A Figura 12 mostra a estrutura do pacote `Networking` do Java DTV.

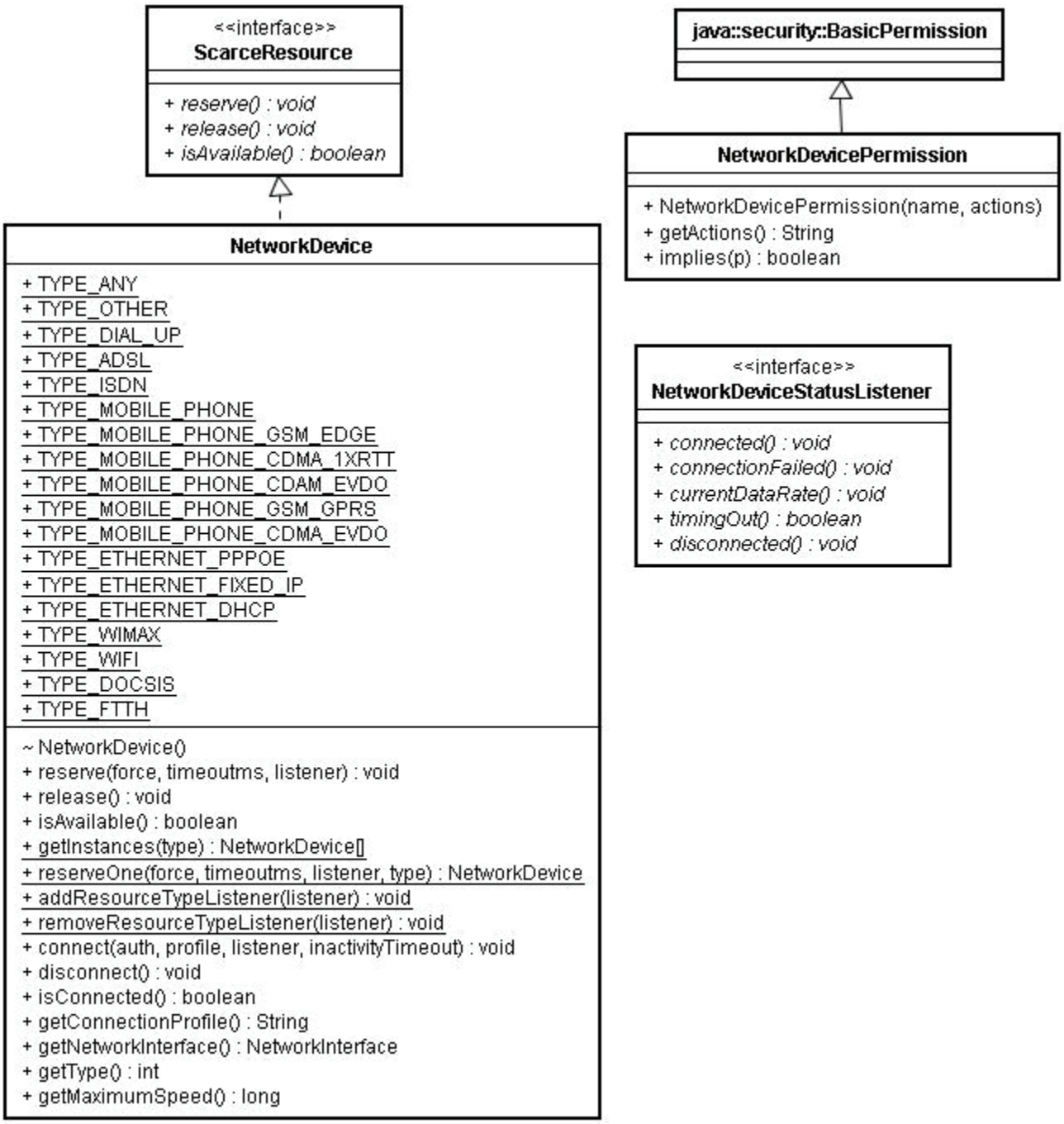


Figura 12 – Pacote `Networking` do JavaDTV

NOTA Este pacote está especificado desde o Java DTV 1.0.

41.2 Índice de interfaces

NetworkDeviceStatusListener

Define a interface *listener* para eventos relacionados aos dispositivos da rede.

41.3 Índice de classes

NetworkDevice

Representa toda instância física de qualquer meio de rede em cujo IP (TCP, UDP) a comunicação pode ser atingida pela plataforma.

NetworkDevicePermission

Esta classe é utilizada por várias permissões relacionadas à falta de recuros de dispositivo de rede.

41.4 Classe NetworkDevice

41.4.1 Descrição da classe

`com.sun.dtv.net`

```
java.lang.Object
  └com.sun.dtv.net.NetworkDevice
```

Todas as interfaces implementadas

```
ScarceResource
```

```
public class NetworkDevice
extends Object
implements ScarceResource
```

Representa toda instância física de qualquer meio de rede em cujo IP (TCP, UDP) a comunicação pode ser atingida pela plataforma. Inclui qualquer tipo de modems e interfaces de rede.

Cada dispositivo é manipulado como um recurso escasso permitindo um uso de controle exclusivo: isso significa que os dispositivos suportam múltiplos formatos, mas permitem que apenas um execute a qualquer momento, a plataforma deve resolver o conflito de forma transparente para o aplicativo.

Relação com o `java.net`

Esta classe `NetworkDevice` estende o `java.net` permitindo qualquer aplicativo DTV, que possua as permissões correspondentes, controlar qualquer dispositivo de rede dado: o controle é definido pela habilidade de configurar/conectar/deconectar o dispositivo:

A exclusividade do controle é, então exigida para toda a plataforma. a implementação não deve permitir que nenhum aplicativo (Java e não Java) que não seja um dado proprietário de um recurso de rede o controle ao mesmo tempo.

Reciprocamente, a implementação deve assegurar que o estado do controle exclusivo seja consistente em todos os ambientes do aplicativo (Java e não Java): ou seja, se um aplicativo nativo assumir um controle exclusivo sobre um dispositivo de rede, o recurso escasso de Java `isAvailable()` correspondente retornará *`false`*.

Da mesma forma a comunicação sobre qualquer dispositivo de rede já conectado por meio de aplicativo Java ou não Java é garantido para qualquer aplicativo de Java utilizando `java.net`, assim como para qualquer aplicativo não-Java.

Quando os aplicativos Java utilizam qualquer instalação de rede fornecida pelo `java.net` sem explicitar anteriormente a utlização deste controle API `NetworkDevice`, a implementação deve iniciar uma conexão utilizando o perfil "`default`" da plataforma que inclui o dispositivo de rede padrão, assim como as configurações-padrão (consulte a próxima seção).

Propriedades

A lista a seguir de propriedades estende a árvore de propriedade definida na classe `User` adicionando vocabulário comum que deve ser utilizado ao definir um dado perfil antes de ser manipulado para o método `connect`. A Tabela 54 lista nomes de propriedade seguidos por uma descrição e lista o tipo de conexão ao qual uma dada propriedade é relevante.

Tabela 54 – Propriedades relevantes

propriedade	descrição	tipos relevantes	direito padrão
<*path*>.number	Número de telefone para discar e estabelecer uma conexão. O número é uma *string* que pode conter qualquer caractere, em especial também "caracteres de separação visual" como " ", "-", "(" ou ")". Ele também pode conter outros caracterees específicos de discagem como aqueles para pausa ou controles DTMF, como "#" ou "*".	`TYPE_DIAL_UP`, `TYPE_ISDN`, `TYPE_MOBILE_PHONES_*`	"read,remove"
<*path*>.dns.1	O endereço de IP de um servidor DNS para resolução de nome.	Qualquer `TYPE_` onde um IP permanente possa ser definido	"read,remove"
<*path*>.dns.2	Um servidor DNS alternativo (pode estar vazio)	Qualquer `TYPE_` onde um IP permanente possa ser definido	"read,remove"

onde `<path>` é equivalente a: "`com.sun.dtv.net.if.profiles.`*`<profile>`*" e onde "*`<profile>`*" é o nome do perfil a ser escolhido ao conectar. Implementações devem suportar um perfil chamado `default`" que fornece todas as configurações exigidas para um meio de comunicação preferido pelo sistema. A plataforma também contém uma propriedade de sistema de *timeout* de inatividade padrão de ampla plataforma em milissegundos no "`com.sun.dtv.net.default.timeout`" (disponível a partir do `java.lang.System.getProperty`).

Todas as propriedades definidas aqui devem ser suportadas pela plataforma. Uma implementação pode estender essa lista, no entanto, o último nome dessas propriedades devem possuir prefixo "`x-`". Por exemplo: "`com.sun.dtv.net.if.profiles.net-1.x-notbefore`" adiciona uma propriedade `notbefore` hipotética no perfil `net-1`.

Como descrito em `User`, o processo de criação e leitura antecede o framework de permissão e o aplicativo deve, portanto, receber o direito de acesso correspondente. As propriedades definidas acima possuem um direito de acesso `read` padrão por todos os aplicativos.

Segurança

O framework de permissão é gerenciado pelo `SecurityManager` através do `ScarceResourcePermission` e `NetworkDevicePermission`, ambos permitem configurar as permissões nos recursos e ações na granularidade do aplicativo:

`ScarceResourcePermission` define duas ações: "`reserve`" e "`force`" para controlar acesso aos métodos de recurso de rede incluindo `connect`.

O `NetworkDevicePermission` define uma ação "`dial`" para controlar números de telefone a serem discados nos casos específicos de dispositivos de rede baseados em dial up.

Em qualquer um dos casos, o aplicativo tem permissão para conectar utilizando o perfil-padrão da plataforma.

Nome da permissão

De acordo com a convenção para nomes para permissões de recursos escassos, o nome da permissão para este recurso escasso é definido pelo "networkdevice.any". O domínio do nome da permissão começando com "networkdevice." é reservado por sua especificação para uso futuro. Isso se aplica atualmente para o ScarceResourcePermission e NetworkDevicePermission.

Exemplo de permissão

```
ScarceResourcePermission("networkdevice.any", "reserve,force");
NetworkDevicePermission("networkdevice.any", "dial=+49*");
```

Exemplo de Código

Uma amostra de código simples seria:

```
import com.sun.dtv.resources.ScarceResource;
import com.sun.dtv.resources.ScarceResourceListener;
import com.sun.dtv.net.NetworkDevice;
import com.sun.dtv.net.NetworkDeviceStatusListener;
import com.sun.dtv.platform.User;
import java.util.Properties;

public class Sample implements ScarceResourceListener,
                               NetworkDeviceStatusListener {

    // Interface : ScarceResourceListener

    void released(ScarceResource resource) {}
    void releasedForced(ScarceResource resource) {}
    boolean releasedRequested(ScarceResource resource) { return true; }

    // Interface : NetworkDeviceStatusListener

    void connected(NetworkDevice device, NetworkInterface netif) {
        // está agora conectado e o 'netif' conhece qual configuração de IP ele tem

        {
            // fazer aqui a sua rede, de preferência em um evento

            // ...
            device.disconnect();
        }
    }
    void connectionFailed(NetworkDevice device, String reason) {
        // não pode conectar por alguma 'razão'

    }
    void currentDataRate(NetworkDevice device, long dataRate) {
        // taxa de dados foi alterada

    }
    void disconnected(NetworkDevice device) {
        // o dispositivo foi desconectado e

    }
    boolean timingOut(NetworkDevice device) {
        return true;
    }

    // Este é o core
```

```
    void coreEngine() {
        //...
        try {
           NetworkDevice device
               = NetworkDevice.reserveOne(false,
                                          1000,
                                          this,
                                          Network.TYPE_DIAL_UP);

           // Ok, tenho um recurso

           Properties props
               = User.get("com.sun.dtv.net.if.profiles.test.*");
           if (props.isEmpty()) {
               // assumindo que a aplicação tem acesso 'write'

               User.setProperty("com.sun.dtv.net.if.profiles.test.number",
                                "+1-555-123-456-789");
               // outras propriedades de usuário aqui ...

               User.setProperties(props);
           }
           device.connect(null,
                          "com.sun.dtv.net.if.profiles.test",
                          this,
                          5000);
        }
        catch (TimeoutException e) {
           // sem recurso disponível

        }
    }
}
```

41.4.2 Índice de campos

`static int` **`TYPE_ADSL`**

Tipo específico de dispositivo de rede: `ADSL`.

`static int` **`TYPE_ANY`**

Tipo especial utilizado em `reserve` representando qualquer tipo de dispositivo de rede possível.

`static int` **`TYPE_DIAL_UP`**

Tipo específico de dispositivo de rede: `Dial Up`.

`static int` **`TYPE_DOCSIS`**

Tipo específico de dispositivo de rede: `DOCSIS`.

`static int` **`TYPE_ETHERNET_DHCP`**

Tipo específico de dispositivo de rede: `Ethernet (DHCP)`.

`static int` **`TYPE_ETHERNET_FIXED_IP`**

Tipo específico de dispositivo de rede: `Ethernet (Fixed IP)`.

`static int` **`TYPE_ETHERNET_PPPOE`**

Tipo específico de dispositivo de rede: `Ethernet (PPPoE)`.

`static int` **`TYPE_FTTH`**

Tipo específico de dispositivo de rede: `FTTH`.

`static int` **`TYPE_ISDN`**

Tipo específico de dispositivo de rede: `ISDN`.

`static int` **`TYPE_MOBILE_PHONE`**

Tipo específico de dispositivo de rede: telefone móvel (nesse caso, a tecnologia móvel não é informada).

`static int` **`TYPE_MOBILE_PHONE_CDAM_EVDO`**

Tipo específico de dispositivo de rede: telefone móvel `CDMA/EVDO`.

`static int` **`TYPE_MOBILE_PHONE_CDMA_1XRTT`**

Tipo específico de dispositivo de rede: telefone móvel `CDMA/1xRTT`.

`static int` **`TYPE_MOBILE_PHONE_CDMA_EVDO`**

Tipo específico de dispositivo de rede: telefone móvel `CDMA/EVDO`.

`static int` **`TYPE_MOBILE_PHONE_GSM_EDGE`**

Tipo específico de dispositivo de rede: telefone móvel `GSM/EDGE`.

`static int` **`TYPE_MOBILE_PHONE_GSM_GPRS`**

Tipo específico de dispositivo de rede: telefone móvel `GSM/GPRS`.

`static int` **`TYPE_OTHER`**

Outros tipos específicos de dispositivo de rede.

`static int` **`TYPE_WIFI`**

Tipo específico de dispositivo de rede: `WiFi`.

`static int` **`TYPE_WIMAX`**

Tipo específico de dispositivo de rede: `WiMAX`.

41.4.3 Índice de métodos

`static void` **`addResourceTypeListener`**`(ResourceTypeListener listener)`

Adiciona um `ResourceTypeListener` a implementação.

`void` **`connect`**`(Authenticator auth, String profile, NetworkDeviceStatusListener listener, long inactivityTimeout)`

Inicia a conexão de dado dispositivo de rede para a rede.

`void` **`disconnect`**`()`

Desfaz qualquer conexão que o dado dispositivo de rede esteja estabelendo com a rede.

`String` **`getConnectionProfile`**`()`

Retorna o perfil de conexão de perfil em uso caso o dispositivo seja conectado, ou `null` se o dispositivo não estiver conectado.

`static NetworkDevice[]` **`getInstances`**`(int type)`

Retorna um *array* de handlers cada um representando um recurso `NetworkDevice` disponível na plataforma que atende aos critérios definidos pelo `type`.

`long` **`getMaximumSpeed`**`()`

Retorna a taxa de dados de downstream máximo teórico do recurso.

`NetworkInterface` **`getNetworkInterface`**`()`

Retorna a interface de rede associada se conectado e, caso contrário, `null`.

`int` **`getType`**`()`

Retorna o tipo desse recurso de dispositivo de rede.

`boolean` **`isAvailable`**`()`

Verifica se o dado recurso está correntemente disponível para reserva.

`boolean` **`isConnected`**`()`

Retorna *`true`* se o dispositivo de rede foi conectado utilizando o método `connect()`.

`void` **`release`**`()`

Libera esse recurso.

`static void` **`removeResourceTypeListener`**`(ResourceTypeListener listener)`

Remove um *listener* previamente anexado.

`void` **`reserve`**`(boolean force, long timeoutms, ScarceResourceListener listener)`

Requisita reserva da dada instância de recursos escassos.

`static NetworkDevice` **`reserveOne`**`(boolean force, long timeoutms, ScarceResourceListener listener, int type)`

Retorna uma instância reservada fora do pool de todas as intâncias `NetworkDevice` do dado `type`.

41.4.4 Detalhe dos campos

TYPE_ANY

`public static final int TYPE_ANY = 4095`

Tipo especial utilizado em `reserve` representando qualquer tipo de dispositivo de rede possível.

TYPE_OTHER

`public static final int TYPE_OTHER = 100`

Outros tipos específicos de dispositivo de rede.

TYPE_DIAL_UP

`public static final int TYPE_DIAL_UP = 101`

Tipo específico de dispositivo de rede: `Dial Up`.

TYPE_ADSL

`public static final int TYPE_ADSL = 110`

Tipo específico de dispositivo de rede: `ADSL`.

TYPE_ISDN

`public static final int TYPE_ISDN = 200`

Tipo específico de dispositivo de rede: ISDN.

TYPE_MOBILE_PHONE

```
public static final int TYPE_MOBILE_PHONE = 300
```

Tipo específico de dispositivo de rede: telefone móvel (nesse caso, a tecnologia móvel não é informada).

TYPE_MOBILE_PHONE_GSM_EDGE

```
public static final int TYPE_MOBILE_PHONE_GSM_EDGE = 302
```

Tipo específico de dispositivo de rede: telefone móvel GSM/EDGE.

TYPE_MOBILE_PHONE_CDMA_1XRTT

```
public static final int TYPE_MOBILE_PHONE_CDMA_1XRTT = 310
```

Tipo específico de dispositivo de rede: telefone móvel CDMA/1xRTT.

TYPE_MOBILE_PHONE_CDAM_EVDO

```
public static final int TYPE_MOBILE_PHONE_CDAM_EVDO = 311
```

Tipo específico de dispositivo de rede: telefone móvel CDMA/EVDO.

TYPE_MOBILE_PHONE_GSM_GPRS

```
public static final int TYPE_MOBILE_PHONE_GSM_GPRS = 320
```

Tipo específico de dispositivo de rede: telefone móvel GSM/GPRS.

TYPE_MOBILE_PHONE_CDMA_EVDO

```
public static final int TYPE_MOBILE_PHONE_CDMA_EVDO = 321
```

Tipo específico de dispositivo de rede: telefone móvel CDMA/EVDO.

TYPE_ETHERNET_PPPOE

```
public static final int TYPE_ETHERNET_PPPOE = 401
```

Tipo específico de dispositivo de rede: Ethernet (PPPoE).

TYPE_ETHERNET_FIXED_IP

```
public static final int TYPE_ETHERNET_FIXED_IP = 402
```

Tipo específico de dispositivo de rede: Ethernet (Fixed IP).

TYPE_ETHERNET_DHCP

```
public static final int TYPE_ETHERNET_DHCP = 403
```

Tipo específico de dispositivo de rede: Ethernet (DHCP).

TYPE_WIMAX

```
public static final int TYPE_WIMAX = 601
```

Tipo específico de dispositivo de rede: `WiMAX`.

TYPE_WIFI

```
public static final int TYPE_WIFI = 701
```

Tipo específico de dispositivo de rede: `WiFi`.

TYPE_DOCSIS

```
public static final int TYPE_DOCSIS = 801
```

Tipo específico de dispositivo de rede: `DOCSIS`.

TYPE_FTTH

```
public static final int TYPE_FTTH = 901
```

Tipo específico de dispositivo de rede: `FTTH`.

41.4.5 Detalhe dos métodos

reserve

```
public void reserve(boolean force,
                    long timeoutms,
                    ScarceResourceListener listener)
             throws IllegalArgumentException,
                    TimeoutException,
                    SecurityException
```

Requisita reserva da dada instância de recursos escassos. O método bloquear-se-á até que a instância esteja disponível. O método retorna normalmente apenas se a reserva for bem sucedida. Todos os outros casos são manuseados por exceções.

Primeiro, se houver um gerenciador de segurança, seu método `checkPermission` é chamado com a permissão `ScarceResourcePermission(name, "reserve")`. Se `force` é além disso definido *`true`*, a permissão também verificado no `ScarceResourcePermission(name, "force")`.

Durante o processo de reserva, se a instância do recurso já estiver alocada, a implementação deve notificar o proprietário atual do recurso sobre a requisição de reserva por meio da interface `ScarceResourceListener`:

ou por `ScarceResourceListener.releaseRequested()` se forçar for *false*,

ou por `ScarceResourceListener.releaseForced()` no outro caso.

O *listener* deve ser usado para tais notificações apenas até o recurso ser liberado. Após a liberação, a implementação não deve chamar quaisquer dos métodos da interface do *listener*.

Após esse primeiro evento, a implementação procederá de acordo e liberará (ou não) o recurso requisitado. No caso da implementação liberar o recurso, desencadeará um evento `ScarceResourceListener.released()` ao *listener* proprietário original do recurso informando-o que o recurso não lhe pertence mais.

O aplicativo pode controlar o tempo de espera para que tal recurso esteja disponível ao definir `timeoutms`. No

caso da duração da reserva exceder o tempo expresso em `timeoutms`, um `TimeoutException` é lançado.

Sob operação normal, recursos são liberados usando o método `release`. De qualquer forma, no caso do aplicativo não liberar recursos quando requisitado ou o aplicativo ser terminado, a implementação deve liberar todos os recursos alocados para o aplicativo para permitir que outros aplicativos sejam notificados de mudanças na alocação de recursos e poderem reservá-los. Ver a seção Resource Cleanup do ciclo de vida do aplicativo.

Especificado por:

`reserve` na interface `ScarceResource`

Parâmetros:

`force` - Se *`true`*, esse método retirará o recurso do proprietário atual. Se *`false`*, a implementação bloquear-se-á e esperará até que o recurso esteja disponível (usando `release()`) ou até `timeoutms`.

`timeoutms` - Uma quantidade positiva de milissegundos que esse método esperará para que o recurso seja liberado pelo proprietário atual. O valor `-1` indica que a implementação esperará para sempre.

`listener` - O *listener* a ser anexado para receber notificação sobre o status do recurso.

Lança:

`IllegalArgumentException` - Se o *listener* estiver `null` ou se o valor especificado em `timeoutms` não for válido, isto é, nem um inteiro positivo, nem -1.

`TimeoutException` - Se o tempo especificado em `timeoutms` houver acabado e o recurso não houver podido ser reservado.

`SecurityException` - Se o aplicativo não tiver permissão para ação de reserva para o recurso que está prestes a reservar. Também lançado se forçar for definido *true*, mas o aplicativo não tiver permissão para a ação forçar.

release

`public void release()`

Libera esse recurso.

O recurso deve ser disponibilizado para outro aplicativo através da plataforma. Após chamar esse método, não é mais possível interagir com o dado recurso: chamadas para métodos críticos desses recursos escassos devem ser ignorados e lançar `IllegalStateException`. Essa afirmação é válida e é o comportamento requerido de qualquer classe implementando a interface `ScarceResource`. Para interagir novamente com o dado recurso, o aplicativo deve chamar o método `reserve()` e se tornar o proprietário novamente.

A implementação pode organizar quaisquer recursos do sistema que esse objeto estiver usando. Depois a implementação não deve chamar quaisquer dos métodos do *listener* que estava anexado no momento da reserva.

Se o recurso já estava disponível (isto é, não reservado), esse método não faz nada.

Se o dispositivo de rede estiver conectado quando esse método estiver sendo chamado, a implementação deve chamar disconnect() implicitamente antes de retornar.

Especificado por:

`release` na interface `ScarceResource`

isAvailable

`public boolean isAvailable()`

Verifica se o dado recurso está correntemente disponível para reserva. O valor retornado dá a situação atual e não garante que o recurso ainda estará disponível em um momento posterior, por exemplo, no momento da reserva: outro aplicativo pode ter tomado aquele recurso no tempo decorrido.

Especificado por:

isAvailable na interface ScarceResource

Retorna:

Um boolean definido *true* se o dado recurso estiver atualmente disponível para reserva.

getInstances

```
public static NetworkDevice[] getInstances(int type)
                                    throws IllegalArgumentException
```

Retorna um *array* de handlers cada um representando um recurso NetworkDevice disponível na plataforma que atende aos critérios definidos pelo type. Cada objeto handler é único tanto para aplicativo como para plataforma. Cada objeto handler pode ser diferente para cada um deles em chamadas subsequentes para o método getInstances(). A lista contém todas as instâncias, estejam ou não reservadas.

Se nenhum dispositivo de rede estiver presente, esse método retorna um *array* de comprimento zero.

O type dos critérios permite que o aplicativo foque no subconjunto restrito dos recursos NetworkDevice.

Parâmetros:

type – Qualquer uma das constrantes TYPE_ definidas por essa classe. Em especial o TYPE_ANY permite que a implementação liste todos os dispositivos de rede não importando seu tipo.

Retorna:

Um *array* de instâncias de recursos do tipo NetworkDevice.

Lança:

IllegalArgumentException – se o tipo não corresponde a nenhum dos valores de nenhuma constante TYPE_ definida pela classe.

reserveOne

```
public static NetworkDevice reserveOne(boolean force,
                                       long timeoutms,
                                       ScarceResourceListener listener,
                                       int type)
                                throws SecurityException,
                                       IllegalArgumentException,
                                       TimeoutException
```

Retorna uma instância reservada fora do pool de todas as intâncias NetworkDevice do dado type. Esse método retorna com a instância ou executa uma exceção de acordo com a situação.

O type de parâmetro permite reduzir e focar o pool do qual escolher uma instância NetworkDevice.

Esse método se comporta exatamente como o método reserve().

Parâmetros:

force - Se *true*, esse método tem permissão de retirar qualquer recurso do atual proprietário. Se *false*, a implementação bloqueará e esperará até que um recurso de um dado type esteja disponível (utilizando release()) ou até milissegundos timeoutms.

timeoutms – Uma quantidade positiva de tempo em milissegundos até a qual este método esperará para que qualquer recurso seja liberado por seu proprietário atual. O valor -1 indica que a implementação esperará para sempre.

listener - O *listener* a ser anexado para receber notificação sobre o status do recurso.

`type` – Qualquer uma das constrantes `TYPE_` definidas por essa classe. Em especial o `TYPE_ANY` permite que a implementação escolha qualquer dispositivo de rede não importando seu tipo.

Retorna:

A instância de tipo `NetworkDevice` que foi reservada.

Lança:

`SecurityException` - Se o aplicativo não tiver permissão para ação de reserva para o recurso que está prestes a reservar. Também lançado se forçar for definido *true*, mas o aplicativo não tiver permissão para a ação forçar.

`IllegalArgumentException` - Se o *listener* estiver `null` ou se o valor especificado em `timeoutms` não for válido, isto é, nem um inteiro positivo, nem -1.

`TimeoutException` - Se o tempo especificado em `timeoutms` houver acabado e o recurso não houver podido ser reservado.

Relaciona-se com:

`reserve()`

addResourceTypeListener

```
public static void addResourceTypeListener(ResourceTypeListener listener)
                                    throws NullPointerException
```

Adiciona um `ResourceTypeListener` a implementação. Sempre que um `reserve()` ou um `release()` for chamado para quaisquer recursos do mesmo tipo, a implementação chamará os métodos correspondentes do *listener*.

Parâmetros:

`listener` - O *listener* desencadeado para eventos em recursos do mesmo tipo.

Lança:

`NullPointerException` - Se o *listener* estiver `null`.

Relaciona-se com:

`removeResourceTypeListener()`

removeResourceTypeListener

```
public static void removeResourceTypeListener(ResourceTypeListener listener)
                                       throws NullPointerException
```

Remove um *listener* previamente anexado. Se o *listener* não foi anexado antes, esse método não faz nada.

Parâmetros:

`listener` - O *listener* desencadeado para eventos em recursos do mesmo tipo.

Lança:

`NullPointerException` - Se o *listener* estiver `null`.

Relaciona-se com:

`addResourceTypeListener()`

connect

```
public void connect(Authenticator auth,
                    String profile,
                    NetworkDeviceStatusListener listener,
                    long inactivityTimeout)
```

```
            throws NullPointerException,
                   IllegalArgumentException,
                   SecurityException,
                   IllegalStateException
```

Inicia a conexão de dado dispositivo de rede para a rede. Essa tarefa inclui discagem do número dial-in para dial-ups e configuração da interface de rede correspondente para que a comunicação de IP seja possível (rastreando o IP definido, configuração de DNS...).

Este método espera encontrar os parâmetros necessários para o processo de conexão na árvore de propriedades do usuário apontada pelo `profile`. Deixa-se para a plataforma definir quais parâmetros são necessários para qualquer recurso de rede. por exemplo, um dispositivo de rede específico pode exigir um número dial-up, outro NPA (Nome do ponto de acesso), etc. Também é possível não haver nenhum parâmetro no path de perfil passado, mas o path é necessário.

Espera-se que a chamada para esse método retorne imediatamente e o processo de conexão real possa ser deferido para um processo separado. A implementação utilizará `listener` para informar o aplicativo sobre o status da conexão: `connected` ou `connectionFailed`.

Parâmetros:

`auth` – Objeto manipulando a autenticação. O valor deve ser `null` indicando que não foram fornecidos login e/ou senha. Em casos onde a conexão ainda exija uma autenticação, a implementação utilizará *strings* vazias (ou seja, "").

`profile` – Path de propriedade na árvore de propriedades do usuário onde a configuração associada é armazenada. Esse path deve `null` ou começar com "`com.sun.dtv.net.if.profiles.`" e ser seguido pelo único nome descrevendo o nome do dado perfil. Configurar o valor para `null` é o mesmo que configurá-lo para "`com.sun.dtv.net.if.profiles.default`".

`listener` - `Listener` acionado para alterações de status de conexão.

`inactivityTimeout` – Uma quantidade positiva de tempo em milissegundos de inatividade no canal de rede até que a plataforma acione o método `timingOut()` do *listener* correspondente. O valor `-1` indica que a implementação irá ignorar o *timeout*.

Lança:

`NullPointerException` - Se o *listener* estiver `null`.

`IllegalArgumentException` – Se o path de propriedade dado não foi definido nas propriedades do usuário ou se não começa com "`com.sun.dtv.net.if.profiles.`".

`SecurityException` – se o chamador não tem permissão para se conectar à configuração fornecida como propriedades: isso inclui o caso em que o número a ser discado (para tipo de dial-up de dispositivo) não é disponibilizado pela ação `NetworkDevicePermission dial` correspondente.

`IllegalStateException` – Se o recurso não está sob responsabilidade do aplicativo possuído atualmente (por exemplo, não foi previamente reservado).

Relaciona-se com:

`User`, `NetworkDeviceStatusListener.connected()`, `NetworkDeviceStatusListener.connectionFailed()`

disconnect

```
public void disconnect()
                throws IllegalStateException
```

Desfaz qualquer conexão que o dado dispositivo de rede esteja estabelendo com a rede. Essa tarefa inclui grampeamento da conexão dial-up em caso de discagens. Essa chamada não tem função se o dispositivo de rede

não estiver conectado.

Esse método retornará logo após liberar o *stream* de dados de saída. Ele retornará a um estado que permite uma chamada subsequente à chamada `connect()` correspondente. No momento do retorno, a implementação aciona o método `listener`'s `disconnected()`.

Lança:

`IllegalStateException` – Se o recurso não está sob responsabilidade do aplicativo possuído atualmente (por exemplo, não foi previamente reservado).

Relaciona-se com:

`connect()`, `NetworkDeviceStatusListener.disconnected()`

isConnected

```
public boolean isConnected()
                     throws IllegalStateException
```

Retorna *`true`* se o dispositivo de rede foi conectado utilizando o método `connect()`. Caso contrário esse método retorna *`false`*.

Retorna:

`true` se o dispositivo de rede estiver conectado, caso contrário, *`false`*.

Lança:

`IllegalStateException` – Se o recurso não está sob responsabilidade do aplicativo possuído atualmente (por exemplo, não foi previamente reservado).

getConnectionProfile

```
public String getConnectionProfile()
                              throws IllegalStateException
```

Retorna o perfil de conexão de perfil em uso caso o dispositivo seja conectado, ou `null` se o dispositivo não estiver conectado. O perfil é um path de propriedade na árvore de propriedades do usuário onde a configuração associada é armazenada.

Retorna:

Uma *string* contendo o perfil que foi utilizado ao conectar, ou `null` se o dispositivo estiver desconectado.

Lança:

`IllegalStateException` – Se o recurso não está sob responsabilidade do aplicativo possuído atualmente (por exemplo, não foi previamente reservado).

getNetworkInterface

```
public NetworkInterface getNetworkInterface()
```

Retorna a interface de rede associada se conectado e, caso contrário, `null`. Essa deve ser a mesma instância retornada por meio do `NetworkDeviceStatusListener` no momento da conexão.

Retorna:

interface de rede associada se conectado e, caso contrário, `null`.

getType

```
public int getType()
```

Retorna o tipo desse recurso de dispositivo de rede.

O valor deve corresponder ao das constantes definidas por essa classe. Se nenhum dos valores encaixar-se ao recurso, este método retorna TYPE_OTHER.

Apenas o valor `TYPE_ANY` não pode ser retornado.

Retorna:

O tipo deste recurso.

getMaximumSpeed

```
public long getMaximumSpeed()
```

Retorna a taxa de dados de downstream máximo teórico do recurso. Essa unidade é kbps (kilobits por segundo, onde kilobits = 1000 bits).

Retorna:

Taxa de dados em kbps.

41.5 Classe NetworkDevicePermission

41.5.1 Descrição da classe

`com.sun.dtv.net`

```
java.lang.Object
  └java.security.Permission
      └java.security.BasicPermission
          └com.sun.dtv.net.NetworkDevicePermission
```

Todas as interfaces implementadas

```
Guard, Serializable
```

```
final public class NetworkDevicePermission
extends BasicPermission
```

Esta classe é utilizada por várias permissões relacionadas à falta de recuros de dispositivo de rede. Um `NetworkDevicePermission` contém um nome (também referido como "target name") e uma lista de ações.

Nome

O nome de alvo suportado por essa especificação é "`networkdevice.any`" e permite definir ações em todos os tipos dos dispositivos de rede. Os nomes-alvo definidos aqui estão alinhados àqueles utilizados para `ScarceResourcePermission` para definir ações nos dispositivos de rede representados por `NetworkDevice`. Consequentemente o domínio do nome-alvo `NetworkDevicePermission` começando com "`networkdevice.`" é reservado por essa especificação para uso futuro.

As regras de nomeação também seguem aquelas definidas no `BasicPermission` a partir da qual esta classe herda, em particular, as regras com relação ao curinga "`*`".

Ações

As ações a serem conferidas são passadas ao construtor em uma *string* contendo uma lista de zero ou mais filtros ou palavras-chave separadas por vírgula (por exemplo, campo e pares de valor). O possível filtro é "`dial`" e apresenta o seguinte formato e significado:

`dial=numberFilter`

Pode ser utilizado diversas vezes na string de ação.
Defina um filtro de números permissíveis que podem ser discados pelo `NetworkDevice.connect()`. Essa ação e valor de filtro faz sentido apenas para os dispositivos de rede que utilizam dial-ups.

A *string* de ação é converitdo para caixa-baixa antes do processamento. A representação canônica da lista de ações concedidas é definida para ser a lista de ações presentes em ordem alfabética.

O *`numberFilter`* é definido para ser um *string* representando um número de telefone completo ou parcial que pode incluir qualquer caractere. Além disso, um asterisco pode aparecer sozinho, ou ao final de um número, significando uma correspondência com um curinga. Observar que embora o `DTMF-*` seja um dígito válido para um número dial-up, não pode ser utilizado dentro deste mecanismo de filtro.

Durante o processo de compatibilidade entre o *numberFilter* e o número solicitado para ser discado no tempo de conexão, a implementação deve ignorar qualquer caractere que não seja dígito possivelmente após ter substituído o código de acesso internacional "`+`" com os dígitos correspondentes e comparar os dois valores de *string* resultantes. Números são compatíveis e é concedida discagem se os valores de *string* forem compatíveis.

Exemplos de ações

`dial=*`

Compatibilidade: `*`
Válido; Permite que qualquer número seja discado.

`dial=0123*`

Compatibilidade: `0123*`
Válido; Permite que qualquer número que comece com "`0123`" seja discado.

`dial=+1-234-*`

Compatibilidade: `+1234*`

Válido; Permite que qualquer número com prefixo "`+`" ou dígitos equivalentes (por exemplo "`00`") seguidos por "`1234`" seja discado. Observar o prefixo para código de país internacional assim como o caractere de separação visual.

`dial=001234567890`

Compatibilidade: `001234567890*`
Válido; Permite apenas que o número acima seja discado assim como "`+1234567890`" em países onde "`+`" é equivalente a "`00`".

`dial=+1-234-567-890p12*`

Compatibilidade: `+123456789012*`

Válido; Permite apenas que o número comlpeto mostrado acima seja discado (assim como a variante com o prefixo de discagem internacional anterior) seguido por uma pausa e logo depois `DTMF-1` e `DTMF-2`, então qualquer outro dígito. Isso permite acesso a serviços específicos que estão atrás do AVR e são comandados por menu DTMF.

`dial=123*,dial=456*`

Compatibilidade: `123*` e `456*`
Válido; Permite discagem de qualquer número começando com "`123`" ou "`456`".

`dial=123*4`

Inválido; Asterisco só é permitido no final do dado número.

`dial=12+34`

Inválido; "+" só é permitido como primeiro caractere de um número de telefone.

Representação canônica

A representação canônica da permissão de discagem é definida para ser a representação mínima de *string*.

para essa permissão que segue o formato de descrição acima,

que é totalmente consistente com a tentativa,

A Tabela 55 lista diferentes exemplos e sua representação canônica padrão.

Tabela 55 – Exemplos para representações canônicas de permissões

Permissão	Canônico	Descrição
"dial=123*"	"dial=123*"	Já é mínima.
"dial=123*,dial=456*"	"dial=123*,dial=456*"	Já é mínima.
"dial=12345*,dial=123*"	"dial=123*"	Números em conformidade com o primeiro filtro também passam pelo segundo, portanto, a representação canônica é, na verdade, o segundo filtro.

Regras implícitas

A mesma regra como anteriormente se aplica também à regra implícita. Ou seja, dados os dois *numberFilter*s, se um for naturalmente incluído no segundo, então este implica o primeiro. Por exemplo, "`123*`" implica "`123456789`".

Exemplo de Código

```
NetworkDevicePermission("networkdevice.any", "dial=+1234*");
```

41.5.2 Índice de construtores

`NetworkDevicePermission``(String name, String actions)`

Cria um novo objeto `NetworkDevicePermission` com nome e ações específicos.

41.5.3 Índice de métodos

`String` **`getActions`**`()`

Retorna a "Representação do *string* canônico" das ações.

`boolean` **`implies`**`(Permission p)`

Verifica se o objeto `NetworkDevicePermission` “implica” a permissão específica.

41.5.4 Detalhe dos construtores

NetworkDevicePermission

```
public NetworkDevicePermission(String name,
                               String actions)
```

Cria um novo objeto `NetworkDevicePermission` com nome e ações específicos. Um `NetworkDevicePermission` contém um nome (também é referido como "*target name*") e pode exigir uma lista de ações dependendo do nome-alvo.

Parâmetros:

`name` - nome do `ScarceResourcePermission`.

`actions` – *string* de ações.

41.5.5 Detalhe dos métodos

getActions

```
public String getActions()
```

Retorna a "representação canônica da *string*" das ações. Ou seja, esse método sempre retorna ações presentes em ordem alfabética. `force`, `reserve`. Por exemplo, se este objeto `ScarceResourcePermission` permite ambas ações `reserve`, `force`, uma chamada para `getActions` retornará a *string* "`force,reserve`".

Substituições:

`getActions` na classe `BasicPermission`

Retorna:

A representação canônica da *string* das ações.

implies

```
public boolean implies(Permission p)
```

Verifica se o objeto `NetworkDevicePermission` "implica" a permissão específica.

Mais especificamente, esse método retorna *true* se:

- p é uma instância de `NetworkDevicePermission`,
- as ações de `p` são um subconjunto adequado dessas ações de objeto (referem-se à seção de regras implícitas), e
- o nome de `p` está implícito por esse nome do objeto. Por exemplo, "`network.*`" implica "`network.any`".

Substituições:

`implies` na classe `BasicPermission`

Parâmetros:

`p` – permissão para comparar.

Retorna:

`true` se a permissão específica estiver implícita por esse objeto, *`false`* caso não esteja.

41.6 Interface NetworkDeviceStatusListener

41.6.1 Descrição da interface

com.sun.dtv.net

```
public interface NetworkDeviceStatusListener
```

Define a interface *listener* para eventos relacionados aos dispositivos da rede.

41.6.2 Índice de métodos

`void` **`connected`**`(NetworkDevice device, NetworkInterface netif)`

Reporta quando o `device` foi conectado.

`void` **`connectionFailed`**`(NetworkDevice device, String reason)`

Reporta quando o `device` não pode se conectar por alguma razão.

`void` **`currentDataRate`**`(NetworkDevice device, long dataRate)`

Reporta a taxa de dados atuais de um dado `device` se este foi alterado desde o último relatório.

`void` **`disconnected`**`(NetworkDevice device)`

Reporta quando o `device` foi desconectado.

`boolean` **`timingOut`**`(NetworkDevice device)`

Reporta quando o `device` está prestes a ser desconectado em função de um período de *timeout* de inatividade fornecido no tempo de construção decorrido.

41.6.3 Detalhe dos métodos

connected

```
void connected(NetworkDevice device,
               NetworkInterface netif)
```

Reporta quando o `device` foi conectado.

Junto com tal chamada, a implementação passa um objeto `NetworkInterface` representando a interface que foi atribuída para a conexão. Entre outras facilidades, essa classe permite recuperar os endereços de Internet associados.

Parâmetros:

`device` – Dispositivo de rede que foi conectado.

`netif` – Objeto `NetworkInterface` associado.

connectionFailed

```
void connectionFailed(NetworkDevice device,
                      String reason)
```

Reporta quando o `device` não pode se conectar por alguma razão.

Parâmetros:

`device` – Dispositivo de rede cuja conexão falhou.

`reason` - String curto descrevendo o erro que ocorreu.

currentDataRate

```
void currentDataRate(NetworkDevice device,
                     long dataRate)
```

Reporta a taxa de dados atuais de um dado `device` se este foi alterado desde o último relatório. A taxa de dados é informada apenas quando o dispositivo de rede é conectado anteriormente. Logo após uma conexão, a plataforma deve chamar este método ao menos uma vez.

A implementação não deve chamar excessivamente este método e de forma alguma deve chamar com uma frequência maior do que a cada um segundo.

Parâmetros:

`device` – Dispositivo de rede cuja taxa de dados tenha sido alterada.

`dataRate` - taxa de dados atual em kbps.

Relaciona-se com:

`NetworkDevice.getMaximumSpeed()`

timingOut

```
boolean timingOut(NetworkDevice device)
```

Reporta quando o `device` está prestes a ser desconectado em função de um período de *timeout* de inatividade fornecido no tempo de construção decorrido.

O aplicativo recebe a chance de reagir de acordo, caso não queira ser desconectado: se a implementação desse método retorna *`true`*, a implementação deve desconectar o dispositivo no retorno deste método. Se o método retorna *`false`*, a implementação deve resetar o *timer* para o valor de *timeout* original permitindo que ele acione novamente o aplicativo no próximo *timeout*.

Ao chamar esse método, certas plataformas podem definir um timespan até que o método retorne. Caso não retorne dentro do timeframe, a implementação deve se comportar como se o método retornasse *`true`*. Esse mecanismo permite evitar o bloqueio desnecessário desse método.

Parâmetros:

`device` – Dispositivo de rede cujo *timeout* tenha expirado.

Retorna:

Um boolean para indicar se o aplicativo quer aceitar o *timeout*. O valor *`true`* indica a implementação para desconectar o dado dispositivo.

disconnected

```
void disconnected(NetworkDevice device)
```

Reporta quando o `device` foi desconectado.

Parâmetros:

`device` – Dispositivo de rede que foi desconectado.

42 Pacote com.sun.dtv.platform

42.1 Descrição do pacote

Provê classes que são específicas da plataforma DTV, particularmente classes que são específicas para o consumidor.

Este pacote fornece um meio de controlar informações específicas de consumidor incluindo as preferências do usuário. É composto pelas seguintes classes:

`User`: permite, entre outros, manipulação de preferências de usuário. As preferências do usuário incluem nome, endereço, e-mail e código do país.

`UserPropertyListener`: permite que um aplicativo esteja informado sobre qualquer alteração nas propriedades do usuário.

`UserPropertyPermission`: permite controlar o acesso a dadas propriedades. Um aplicativo pode receber direitos `read`, `write` e `remove`.

A Figura 13 mostra a estrutura do pacote `Platform` do JavaDTV.

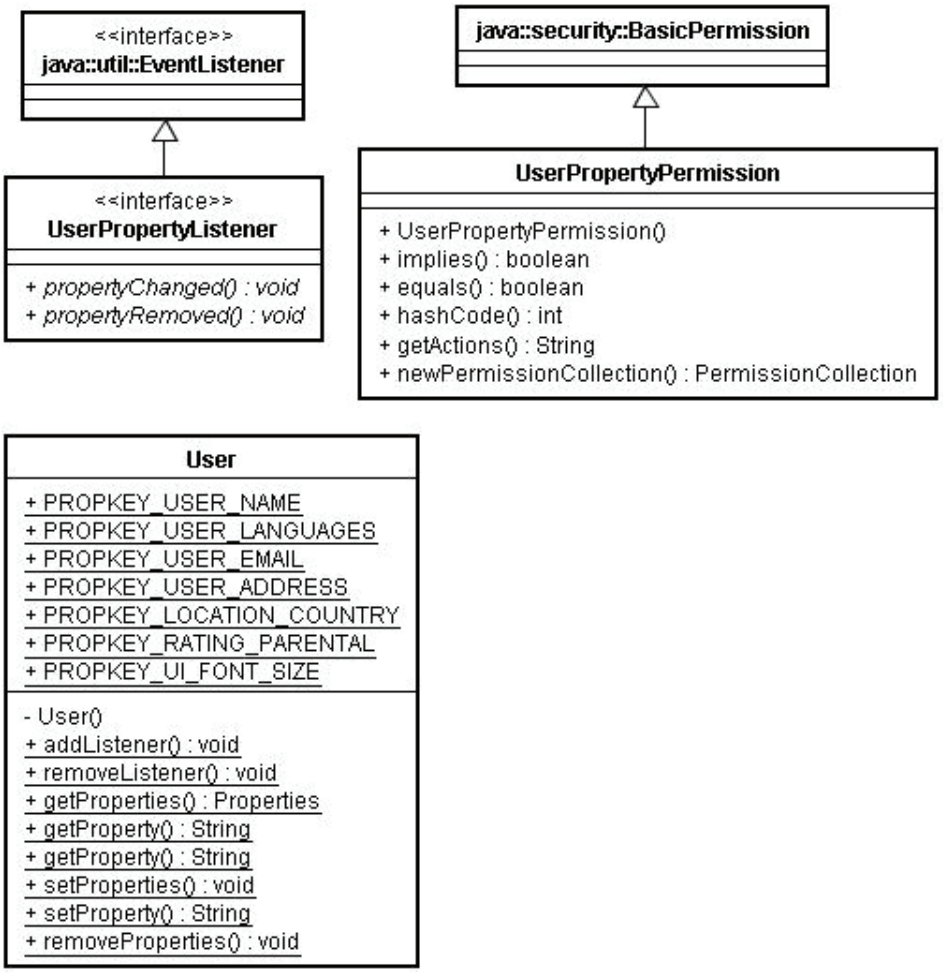


Figura 13 – Pacote `Platform`

NOTA Este pacote está especificado desde o Java DTV 1.0.

42.2 Índice de interfaces

UserPropertyListener

Opcionalmente um aplicativo pode anexar um `UserPropertyListener` ao subsistema de propriedades do usuário para permanecer informado sobre qualquer mudança das propriedades do usuário.

42.3 Índice de classes

User

A classe `User` contém diversos campos de classe e métodos úteis que são específicos ao cliente da plataforma.

UserPropertyPermission

Essa classe descreve a permissão para as propriedades do usuário.

42.4 Classe User

42.4.1 Descrição da classe

`com.sun.dtv.platform`

```
java.lang.Object
  └com.sun.dtv.platform.User
```

```
final public class User
extends Object
```

A classe `User` contém diversos campos de classe e métodos úteis que são específicos ao cliente da plataforma. Não pode ser exemplificado.

Entre as facilidades fornecidas pela classe `User` estão o acesso e a manipulação das propriedades do usuário. As propriedades de usuário são definidas para ser um conjunto de propriedades específicas em runtime a um usuário para uma determinada plataforma, aplicativo ou biblioteca terceirizada.

O subsistema de propriedades de usuário descrito aqui difere das propriedades do sistema definidas em System.getProperties(). A principal diferença entre os dois susbsistemas de gerenciamento de propriedade é que um é para propriedades de sistema e exclusivo para plataforma enquando o outro é dedicado a um determinado usuário. Uma plataforma pode suportar múltiplos usuários, por exemplo identificado por um cartão de token e para o qual a plataforma deve carregar (por exemplo, do cartão de token) o perfil e as propriedades de usuário correspondentes.

Essas propriedades são persistentemente armazenadas e podem ser lidas a qualquer hora por qualquer aplicativo se concedida a permissão `read` correspondente. Essas propriedades podem ser alteradas a qualquer hora por qualquer aplicativo desde que seja concedida a permissão `write` correspondente para a permissão específica.

O framework de permissão é gerenciado pelo `SecurityManager` por meio de `UserPropertyPermission` que permite configurar as permissões na granularidade do aplicativo: qualquer aplicativo pode receber ações de leitura e escrita diferentes para cada uma das propriedades disponíveis. Definir uma permissão para uma propriedade que (ainda) não foi armazenada e manipulada pela plataforma não tem consequências.

Propriedades

a Tabela 56 lista propriedades que devem ser suportadas e estar sempre presentes na plataforma.

Tabela 56 – Propriedades a serem suportadas

Propriedade	Descrição	Direito padrão
`com.sun.dtv.user.name`	Nome do usuário composto por seu primeiro e último nome. O pedido é definido apenas por ser culturalmente apropriado.	"read"
`com.sun.dtv.user.languages`	lista separada por ponto e vírgula composta por códigos de linguagem de 3 letras da ISO-639.	"read"
`com.sun.dtv.user.email`	Endereço de e-mail do usuário como definido pela RFC821.	"read"
`com.sun.dtv.user.address`	Endereço postal do usuário. Pode conter múltiplas linhas separadas pelos retornos de reboque (CR, 13d, 0x0D).	"read"
`com.sun.dtv.location.country`	Código do país da localização da caixa de TV. O formato é um código de país de duas letras como especificado na ISO-3166-1.	"read"
`com.sun.dtv.rating.parental`	Um *string* utilizando a mesma codificação como retornado pelo `ContentRatingAdvisory.getDisplayText()`.	"read"
`com.sun.dtv.ui.font.size`	Um *string* cujo valor representa em pontos o tamanho da fonte preferida.	"read"

Essa árvore de propriedades pode ser estendida a aplicativos e bibliotecas de terceiros. O limite de número de propriedades é específico da plataforma. Novas propriedades devem seguir a mesma convenção para seus nomes e para criação de nome de um novo pacote Java, que é: considerando o nomde de domínio de internet da organização, como `example.com`.Então reverter este nome, componente por componente, anexando o prefixo da árvore ou do nome do pacote: `com.example.voting`. Convenções usadas além do nome de domínio revertido são deixadas a critério de qualquer companhia.

A criação de uma nova propriedade é permitida desde que a permissão `write` correspondente tenha sido concedida para o nome de propriedade correspondente. Consequentemente o aplicativo pode ler essa propriedade apenas se é dado acesso de permissão `read`. Assim qualquer propriedade adicionada não é acessível como padrão.

O nome de domínio `com.sun.dtv` é reservado para propriedades definidas nesta especificação.

Propriedades, exemplos e valores-padrão podem ser vistos na Tabela 57.

Tabela 57 – Exemplos de propriedade

Propriedade	exemplo	valor padrão
com.sun.dtv.user.name	"Jean-Yves Mustermann"	""
com.sun.dtv.user.languages	"deu;fre;eng"	""
com.sun.dtv.user.email	"jy.muster@example.net"	""
com.sun.dtv.user.address	"Heidestr. 17, 81500 Munich, Germany"	""
com.sun.dtv.location.country	"de"	""
com.sun.dtv.rating.parental	"PG16	""
com.sun.dtv.ui.font.size	"18"	"26"

Os valores-padrão definidos na tabela acima devem ser fornecidos dentro de cada objeto `Properties` retornado por essa classe.

Nomeação e filtragem de propriedade

Um `name` é um nome de propriedade ("`com.sun.dtv.user.name`", "`com.sun.dtv.ui.font.size`", etc). A convenção de nomeação segue a convenção de nomeação de propriedade hierárquica.

No caso de um `filter`, um asterisco pode aparecer no final do nome, seguindo um "`.`", ou sozinho, significando uma compatibilidade com um curinga. Por exemplo: "`com.sun.dtv.*`" ou "`*`" são válidos. Por outro lado, "`*user`" ou "`a*b`" não são válidos.

Um filtro pode também ser uma lista separada por ponto-e-vírgula de filtros como descrito anteriormente. Por exemplo "`com.sun.dtv.user.*;com.sun.dtv.ui.*`" também é uma expressão de filtro válida. O filtro "`com.sun.dtv.user.*;;`" também é válido e equivalente ao "`com.sun.dtv.user.*`".

Exemplo de Código

Uma amostra de código simples seria:

```
import com.sun.dtv.platform.User;

// Devem ser armazenados persistentemente imediatamente, assumindo que a aplicação
// tem acesso 'write' access na sua propriedade.
User.setProperty("com.sun.dtv.user.name", "myself");
```

... ou para edição de lote:

```
import com.sun.dtv.platform.User;
import java.util.Properties;

Properties props = User.getProperties("com.sun.dtv.user.*");
props.setProperty("com.sun.dtv.user.name",  "myself");
props.setProperty("com.sun.dtv.user.email", "myself@example.net");
//...

// Agora deve ser armazenado persistentemente
User.setProperties(props);
```

Relaciona-se com:

`Properties`, `SecurityManager`

42.4.2 Índice de campos

`static String` **`PROPKEY_LOCATION_COUNTRY`**

Nome de propriedade padrão para código de país de localização.

`static String` **`PROPKEY_RATING_PARENTAL`**

Nome de propriedade padrão para classificação parental.

`static String` **`PROPKEY_UI_FONT_SIZE`**

Nome de propriedade padrão para tamanho de fonte UI.

`static String` **`PROPKEY_USER_ADDRESS`**

Nome de propriedade padrão para endereço do usuário.

`static String` **`PROPKEY_USER_EMAIL`**

Nome de propriedade padrão para e-mail do usuário.

`static String` **`PROPKEY_USER_LANGUAGES`**

Nome de propriedade padrão para idiomas do usuário.

`static String` **`PROPKEY_USER_NAME`**

Nome de propriedade padrão para nome do usuário.

42.4.3 Índice de métodos

`static void` **`addListener`**`(UserPropertyListener listener, String filter)`

Anexa um *listener* acionado para alterações para qualquer propriedade dentro da faixa definida pelo `filter`.

`static Properties` **`getProperties`**`(String filter)`

Recupera uma cópia de todas as propriedades do usuário compatíveis com o `filter` e para as quais o aplicativo tem direito de acesso `read`.

`static String` **`getProperty`**`(String key)`

Obtém a propriedade do usuário indicada pela tecla especificada.

`static String` **`getProperty`**`(String key, String def)`

Obtém a propriedade do usuário indicada pela tecla especificada.

`static void` **`removeListener`**`(UserPropertyListener listener)`

Desconecta um *listener* de propriedade de usuário que foi previamente anexado utilizando `addListener()`.

`static void` **`removeProperties`**`(String filter)`

Remove essa propriedade e todos os seus descendentes, invalidando qualquer propriedade presente nos nós removidos.

`static void` **`setProperties`**`(Properties properties)`

Adiciona novas propriedades e/ou substitui propriedades existentes.

`static String` **`setProperty`**`(String key, String value)`

Configura a propriedade do usuário indicada pela tecla especificada.

42.4.4 Detalhe dos campos

PROPKEY_USER_NAME

```
public static String PROPKEY_USER_NAME
```

Nome de propriedade padrão para nome do usuário.

PROPKEY_USER_LANGUAGES

```
public static String PROPKEY_USER_LANGUAGES
```

Nome de propriedade padrão para idiomas do usuário.

PROPKEY_USER_EMAIL

```
public static String PROPKEY_USER_EMAIL
```

Nome de propriedade padrão para e-mail do usuário.

PROPKEY_USER_ADDRESS

```
public static String PROPKEY_USER_ADDRESS
```

Nome de propriedade padrão para endereço do usuário.

PROPKEY_LOCATION_COUNTRY

```
public static String PROPKEY_LOCATION_COUNTRY
```

Nome de propriedade padrão para código de país de localização.

PROPKEY_RATING_PARENTAL

```
public static String PROPKEY_RATING_PARENTAL
```

Nome de propriedade padrão para classificação parental.

PROPKEY_UI_FONT_SIZE

```
public static String PROPKEY_UI_FONT_SIZE
```

Nome de propriedade padrão para tamanho de fonte UI.

42.4.5 Detalhe dos métodos

addListener

```
public static void addListener(UserPropertyListener listener,
                               String filter)
                        throws NullPointerException,
                               IllegalArgumentException
```

Anexa um *listener* acionado para alterações para qualquer propriedade dentro da faixa definida pelo `filter`. Anexar o mesmo *listener* a diferentes `filter` é permitido e não executa nenhuma exceção. Anexar o mesmo ou mais *listeners* à faixas que se sobrepõem é permitido e alterações acionarão o(s) *listener*(s) correspondente(s) de acordo com as associações.

Parâmetros:

`listener` - `UserPropertyListener` a ser anexado.

`filter` – Filtro ao qual associar o *listener* de propriedade de usuário.

Lança:

`NullPointerException` – Se nenhum dos argumentos for `null`.

`IllegalArgumentException` – Se o filtro não segue a convenção definida na descrição de classe acima, `null`.

Relaciona-se com:

`removeListener()`

removeListener

```
public static void removeListener(UserPropertyListener listener)
                           throws NullPointerException
```

Desconecta um *listener* de propriedade de usuário que foi previamente anexado utilizando `addListener()`. Se o `listener` for `null` ou se o `listener` não for anexado previamente, nenhuma exceção é executada e nenhuma ação é tomada. Se o `listener` fo anexado a diversas faixas de propriedade, todas as associações devem ser esclarecidas e todas as instâncias de tal *listener* removidas.

Parâmetros:

`listener` - O `serPropertyListener` a ser desconectado. Pode ser `null`.

Lança:

`NullPointerException` - Se o *listener* estiver `null`.

Relaciona-se com:

`addListener()`

getProperties

```
public static Properties getProperties(String filter)
                                 throws SecurityException,
                                        IllegalArgumentException,
                                        NullPointerException
```

Recupera uma cópia de todas as propriedades do usuário compatíveis com o `filter` e para as quais o aplicativo tem direito de acesso `read`. Se não houver compatibilidade, um objeto `Properties` vazio é retornado.

O objeto retornado deve conter em seu atributo `defaults`, uma cópia da lista de todas as propriedades obrigatórias com seus valores-padrão (`Properties.defaults`). No entanto qualquer modificação feita pelo aplicativo a sua coleção de valores e valores-padrão devem ser ignorados pela implementação, por exemplo, não ssubstituirá as propriedades de plataforma padrão. O aplicativo deve chamar o método getProperties() para alterar e armazenar propriedades.

Observação: A implementação não é exigida para validar as teclas e valores disponíveis nas propriedades de usuário da plataforma. As aplicações devem ler os valores retornados com cuidado.

Parâmetros:

`filter` – Filtro para aplicar em todas as propriedades de usuário.

Retorna:

Todas as propriedades de usuário que sejam compatíveis com o `filter` encapsulado em um ojeto `Properties`.

Lança:

`SecurityException` – Se o aplicativo não possuir permissão de leitura para nenhuma das propriedades que estão prestes a serem lidas.

`IllegalArgumentException` – Se o filtro não segue a convenção definida na descrição de classe acima.

`NullPointerException` – se o filtro for `null`.

Relaciona-se com:

`Properties`, `setProperties()`

getProperty

```
public static String getProperty(String key,
                                 String def)
                          throws SecurityException,
                                 IllegalArgumentException,
                                 NullPointerException
```

Obtém a propriedade do usuário indicada pela tecla especificada.

Parâmetros:

`key` – Nome de propriedade de usuário.

`def` – Valor-padrão.

Retorna:

Valor do *string* da propriedade do usuário, ou valor-padrão se não houver nenhuma proriedade de usuário com tal tecla.

Lança:

`SecurityException` – Se o aplicativo não possuir permissão de leitura para a propriedade que está preste a ser lida.

`IllegalArgumentException` – Se a tecla estiver vazia.

`NullPointerException` – Se a tecla for `null`.

Relaciona-se com:

`setProperty()`

getProperty

```
public static String getProperty(String key)
                          throws SecurityException,
                                 IllegalArgumentException,
                                 NullPointerException
```

Obtém a propriedade do usuário indicada pela tecla especificada.

Parâmetros:

`key` – Nome de propriedade de usuário.

Retorna:

Valor da *string* da propriedade do usuário, ou `null` se não houver nenhuma proriedade de usuário com tal tecla.

Lança:

`SecurityException` – Se o aplicativo não possuir permissão de leitura para a propriedade que está preste a ser lida.

`IllegalArgumentException` – Se a tecla estiver vazia.

`NullPointerException` – Se a tecla for `null`.

Relaciona-se com:

`setProperty()`

setProperties

```
public static void setProperties(Properties properties)
                          throws NullPointerException,
                                 SecurityException
```

Adiciona novas propriedades e/ou substitui propriedades existentes. Dependendo se as propriedades presentes em `properties` existirem ou não, a implementação deve substituir ou adicionar respectivamente a propriedade correspondente. A implementação deve armazenar persistentemente as propriedades de usuário de forma sincronizada a este método.

A implementação acionará sincronicamente para esta chamada de método um `UserPropertyListener.propertyChanged()` para cada propriedade que for adicionada ou substituída por essa ação. No entanto se a substuição não alterar o valor (por exemplo, o valor prévio e o novo valor da dada propriedade são iguais), nenhum `propertyChanged()` é chamado.

Os valores dentro dos atributos `Properties.defaults` do argumento `properties` devem ser ignorados pela implementação.

Observação: A implementação não exige validação das chaves e valores passados em `properties`. Os aplicativos devem validar cuidadosamente esses *string*s.

Parâmetros:

`properties` – Um objeto `Properties` que lista todas as propriedades que devem ser alteradas ou, se ainda não existirem, adicionadas às propriedades do usuário.

Lança:

`NullPointerException` – Se a propriedade for `null`.

`SecurityException` – Se o aplicativo não possuir permissão `write` para nenhuma das propriedades que estão prestes a serem lidas. A verificação dessa permissão deve ser realizada antes de adicionar ou substituir propriedades existentes.

Relaciona-se com:

`Properties`, `PropertyPermission`, `getProperties()`

setProperty

```
public static String setProperty(String key,
                                 String value)
                          throws SecurityException,
                                 NullPointerException,
                                 IllegalArgumentException
```

Configura a propriedade do usuário indicada pela tecla especificada. A implementação deve armazenar persistentemente as propriedades de usuário de forma sincronizada a este método.

Quanto a `setProperties`, a implementação acionará sincronicamente a esta chamada um

`UserPropertyListener.propertyChanged()` para a propriedade que foi adicionada ou alterada por essa ação. No entanto se a substuição não alterar o valor (por exemplo, o valor prévio e o novo valor da dada propriedade são iguais), nenhum `propertyChanged()` é chamado.

Parâmetros:

`key` – Nome de propriedade de usuário.

`value` – Valor de propriedade de usuário.

Retorna:

O valor anterior da propriedade do usuário, ou `null` se não possuir um.

Lança:

`SecurityException` – Se o aplicativo não possuir permissão `write` para aquela propriedade que está prestes a ser adicionada ou substituída. A verificação dessa permissão deve ser realizada antes de adicionar ou substituir.

`NullPointerException` – Se a tecla ou valor for `null`.

`IllegalArgumentException` – Se a tecla estiver vazia.

Relaciona-se com:

`getProperty()`

removeProperties

```
public static void removeProperties(String filter)
                            throws SecurityException,
                                   IllegalArgumentException,
                                   NullPointerException
```

Remove essa propriedade e todos os seus descendentes, invalidando qualquer propriedade presente nos nós removidos.

A implementação acionará sincronicamente a esta chamada de método um `UserPropertyListener.propertyRemoved()` para cada propriedade que for removida por essa ação. Este método retorna quando todas as propriedades-alvo (nós e descendentes) forem removidos persistentemente.

Parâmetros:

`filter` – Filtro das propriedades que devem ser removidas.

Lança:

`SecurityException` – Se o aplicativo não possuir permissão `write` para nenhuma das propriedades dadas que estão prestes a serem removidas. A verificação dessa permissão deve ser realizada antes de remover qualquer propriedade compatível com o filtro.

`IllegalArgumentException` – Se o filtro não segue a convenção definida na descrição de classe acima.

`NullPointerException` – Se o filtro for `null`.

42.5 Interface UserPropertyListener

42.5.1 Descrição da interface

`com.sun.dtv.platform`

Todas as super-interfaces

`EventListener`

```
public interface UserPropertyListener
```

`extends EventListener`

Opcionalmente um aplicativo pode anexar um `UserPropertyListener` ao subsistema de propriedades do usuário para permanecer informado sobre qualquer mudança das propriedades do usuário. Isso inclui também propriedades de usuário criadas e gerenciadas por aplicativos e bibliotecas terceirizadas.

Este *listener* deve ser anexado utilizando o método the `User.addListener()` e desconectado utilizando o `User.removeListener()`. O aplicativo pode configurar a sensibilidade do *listener* configurando um filtro que descreve que as propriedades devem ser monitoradas. O *listener* irá então ser acionado pela implementação assim que qualquer uma das propriedades relevantes para aquela máscara for removida ou tenha seu valor alterado.

42.5.2 Índice de métodos

`void` **`propertyChanged`**`(String key, String oldValue, String newValue)`

Reporta que o valor da propriedade identificada pela `key` foi alterado de `oldValue` para `newValue`.

`void` **`propertyRemoved`**`(String key, String oldValue)`

Reporta que a propriedade identificada pela `key` foi removida das propriedades do usuário.

42.5.3 Detalhe dos métodos

propertyChanged

```
void propertyChanged(String key,
                     String oldValue,
                     String newValue)
```

Reporta que o valor da propriedade identificada pela `key` foi alterado de `oldValue` para `newValue`. Este método deve ser acionado apenas quando o `newValue` for diferente do `oldValue`.

Parâmetros:

`key` – Tecla de propriedade cujo valor foi alterado.

`oldValue` – Valor associado a `key` antes da alteração.

`newValue` – Valor associado a `key` após a alteração.

propertyRemoved

```
void propertyRemoved(String key,
                     String oldValue)
```

Reporta que a propriedade identificada pela `key` foi removida das propriedades do usuário. Isso também fornece o valor anterior que foi associado a esta chave.

Observação: especialmente neste caso em que uma subárvore completa de propriedades de usuário é removida, apenas propriedades que realmente possuem um valor devem acionar esta interface.

Parâmetros:

`key` – Tecla de propriedade cujo valor foi alterado.

`oldValue` – Valor associado a `key` antes da remoção da propriedade.

42.6 Classe UserPropertyPermission

42.6.1 Descrição da classe

com.sun.dtv.platform

```
java.lang.Object
  └java.security.Permission
      └java.security.BasicPermission
          └com.sun.dtv.platform.UserPropertyPermission
```

Todas as interfaces implementadas

`Guard, Serializable`

```
final public class UserPropertyPermission
extends BasicPermission
```

Essa classe descreve a permissão para as propriedades do usuário. Apoia-se no mesmo comportamento descrito para `PropertyPermission` mas aplica-se para as propriedades do usuário como definido em `User`.

Nome

O nome é aquele da propriedade do usuário (por exemplo "`com.sun.dtv.user.name`", "`com.sun.dtv.rating.parental`", etc). A convenção de nomeação segue a convenção de nomeação de propriedade hierárquica também descrita em `User`. Além disso, um asterisco pode aparecer no final do nome, seguindo um "`.`", ou sozinho, significando uma compatibilidade com um curinga. Por exemplo "`com.sun.dtv.*`" ou "`*`" é válido, "`*sun`" ou "`com*user`" é inválido.

Ações

As ações a serem conferidas são passadas ao construtor em um *string* contendo uma lista de zero ou mais palavras-chave separadas por vírgula. As palavras-chave possíveis são "`read`" e "`write`". Seu significado é definido como segue:

- ler
  permissão de leitura. Permite que `User.getProperties()` e `User.getProperty()` sejam chamados.
- escrever
  permissão de escrita. Permite que `User.setProperties()`, `User.setProperty()` e `User.removeProperties()` sejam chamados.

Relaciona-se com:

`PropertyPermission`, `BasicPermission`, `Permission`, `User`

42.6.2 Índice de construtores

`UserPropertyPermission``(String name, String actions)`

Cria um novo objeto `UserPropertyPermission` com nome e ações especificados.

42.6.3 Índice de métodos

`boolean` **`equals`**`(Object obj)`

Verifica dois objetos `UserPropertyPermission` para igualdade.

`String` **`getActions`**`()`

Retorna a "representação *string* canônica" de outras ações.

`int` **`hashCode`**`()`

Retorna o valor do código *hash* para esse objeto.

`boolean` **`implies`**`(Permission p)`

Verifica se esse objeto `UserPropertyPermission` "implica" a permissão especificada.

`PermissionCollection` **`newPermissionCollection`**`()`

Retorna um novo objeto `PermissionCollection` para armazenamento de objetos `UserPropertyPermission`.

42.6.4 Detalhe dos construtores

UserPropertyPermission

```
public UserPropertyPermission(String name,
                              String actions)
```

Cria um novo objeto `UserPropertyPermission` com nome e ações especificados. O `name` é o nome da propriedade, e `action` contém uma lista separada por vírgulas de ações desejáveis concedidas para a propriedade. Possíveis ações são `read` e `write`.

Parâmetros:

`name` - Nome da `UserPropertyPermission`.

`actions` – *string* de ações.

42.6.5 Detalhe dos métodos

implies

```
public boolean implies(Permission p)
```

Verifica se esse objeto `UserPropertyPermission` "implica" a permissão especificada.

Mais especificamente, esse método retorna *true* se:

- p é uma instância de `UserPropertyPermission`,
- as ações de `p` são um subconjunto adequado dessas ações de objeto, e
- o nome de `p` está implícito por esse nome do objeto. Por exemplo, "`com.sun.dtv.user.*`" implica "`com.sun.dtv.user.name`".

Substituições:

`implies` na classe `BasicPermission`

Parâmetros:

`p` – permissão para comparar.

Retorna:

`true` se a permissão específica estiver implícita por esse objeto, *`false`* caso não esteja.

equals

```
public boolean equals(Object obj)
```

Verifica dois objetos `UserPropertyPermission` para igualdade. Verifica que o `obj` é uma `UserPropertyPermission`, e possui o mesmo nome e ações que este objeto.

Substituições:

`equals` na classe `BasicPermission`

Parâmetros:

`obj` – Objeto que estamos testando para igualar com este objeto.

Retorna:

`true` se o `obj` for uma `UserPropertyPermission`, e possui o mesmo nome e ações que este objeto `UserPropertyPermission`.

hashCode

```
public int hashCode()
```

Retorna o valor do código *hash* para esse objeto. O código *hash* utilizado é o código deste nome de permissões, ou seja, `getName().hashCode()`, onde `getName` vem da superclasse de `Permission`.

Substituições:

`hashCode` na classe `BasicPermission`

Retorna:

Um valor de código *hash* para este objeto.

getActions

```
public String getActions()
```

Retorna a "representação *string* canônica" de outras ações. Ou seja, esse método sempre retorna ações presentes em ordem alfabética. `read`, `write`. Por exemplo, se este objeto `UserPropertyPermission` permite ambas ações `write` e `read`, uma chamada para `getActions` retornará a *string* "`read,write`".

Substituições:

`getActions` na classe `BasicPermission`

Retorna:

A representação canônica da *string* das ações.

newPermissionCollection

```
public PermissionCollection newPermissionCollection()
```

Retorna um novo objeto `PermissionCollection` para armazenamento de objetos `UserPropertyPermission`.

Substituições:

`newPermissionCollection` na classe `BasicPermission`

Retorna:

Um novo objeto `PermissionCollection` adequado para armazenar `UserPropertyPermissions`.

43 Pacote com.sun.dtv.resources

43.1 Descrição do pacote

Provê um quadro básico de recursos escassos.

O framework genérico é utilizado por classes que representam recursos físicos dentro da plataforma que exige uma manipulação de bloqueio básico. A interface `ScarceResource` deve ser utilizada pela implementação.

Os aplicativos recebem um meio de solicitar para um recurso de uma maneira mais ou menos amigável: pressupondo que seja concedido um direito correspondente, um aplicativo pode forçar uma reserva deixando que o proprietário atual perca seu recurso. Por outro lado, um aplicativo é bem informado sobre qualquer intenção de ter seu recurso solicitado por outros aplicativos.

Este pacote é composto pelas seguintes classes:

`ScarceResource`: é a interface a ser implementada por qualquer classe representando um recurso físico. Permite reservar ou liberar tal recurso;

`ScarceResourceListener`: permite que um aplicativo se informe sempre que um recurso estiver sendo ou estiver prestes a ser liberado;

`ScarceResourcePermission`: permite controlar o acesso para um dado recurso escasso. Um aplicativo pode receber o direito `reserve` ou mesmo `force` para +a reserva de um recurso escasso.

`ResourceEventListener`: permite que um aplicativo siga eventos relacionados à reserva/liberação de qualquer um dos recursos (de um dado tipo). O registro do *listener* é oferecido pelas classes de recurso escasso especializadas.

A Figura 14 mostra a estrutura do pacote `Resources` do Java DTV.

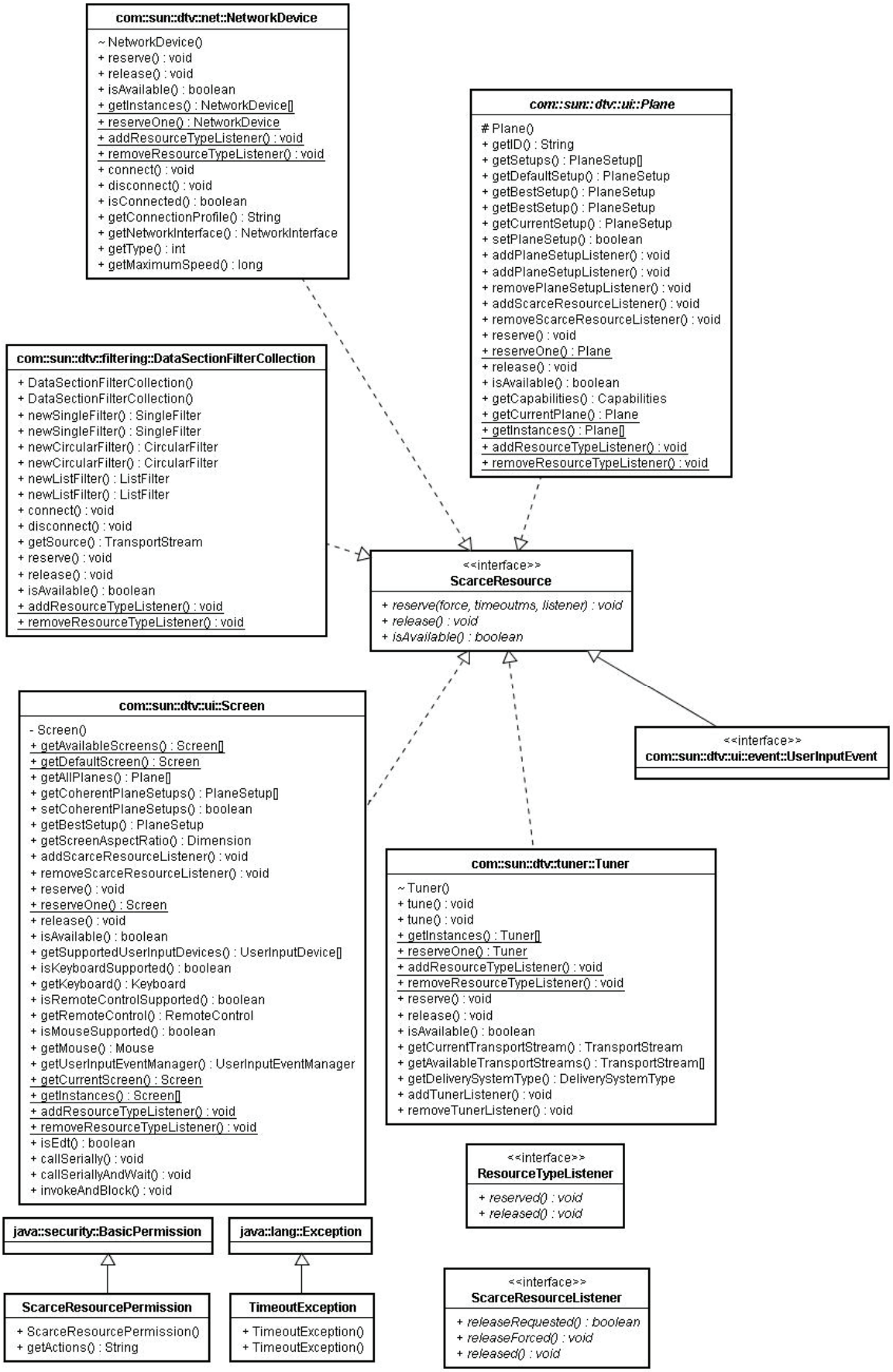


Figura 14 – Pacote `Resources`

NOTA Este pacote está especificado desde o Java DTV 1.0.

43.2 Índice de interfaces

ResourceTypeListener

`Listener` a ser notificado sobre alterações de status sobre recursos do mesmo tipo de objeto ao qual o *listener* foi anexado.

ScarceResource

Representa recursos que exigem manipulação especial para reservar e liberar.

ScarceResourceListener

`Listener` a ser notificado sobre o status de liberação de um dado recurso escasso.

43.3 Índice de classes

ScarceResourcePermission

Esta classe é utilizada por várias permissões relacionadas a recursos escassos.

43.4 Índice de exceções

TimeoutException

Indica a ocorrência de um *timeout*.

43.5 Interface ResourceTypeListener

43.5.1 Descrição da interface

`com.sun.dtv.resources`

```
public interface ResourceTypeListener
```

`Listener` a ser notificado sobre alterações de status sobre recursos do mesmo tipo de objeto ao qual o *listener* foi anexado.

43.5.2 Índice de métodos

`void` **`released`**`(int availableCount)`

Reporta que um recurso foi liberado.

`void` **`reserved`**`(int availableCount)`

Reporta que um recurso foi reservado.

43.5.3 Detalhe dos métodos

reserved

```
void reserved(int availableCount)
```

Reporta que um recurso foi reservado. Este método é acionado apenas para reservas em recursos do mesmo tipo. A implementação passará o número de recursos que estarão disponíveis depois que a reserva for feita.

Parâmetros:

`availableCount` – número de recurso do mesmo tipo ainda disponível.

released

`void released(int availableCount)`

Reporta que um recurso foi liberado. Este método é acionado apenas para liberações em recursos do mesmo tipo. A implementação passará o número de recursos que estarão disponíveis depois que a liberação for realizada.

Parâmetros:

`availableCount` – número de recurso do mesmo tipo ainda disponível.

43.6 Interface ScarceResource

43.6.1 Descrição da interface

`com.sun.dtv.resources`

Todas as subinterfaces conhecidas

`UserInputEvent`

Todas as classes implementadoras conhecidas

`DataSectionFilterCollection, KeyEvent, MouseEvent, NetworkDevice, Plane, RemoteControlEvent, Screen, Tuner`

`public interface ScarceResource`

Representa recursos que exigem manipulação especial para reservar e liberar. Esta interface permite configurar um bloqueio exclusivo em qualquer recurso físico dado possível. O proprietário de tal recurso é definido no momento da reserva, que deve ser a instância do objeto representando um dado recurso escasso. O design dos recursos escassos implica um modelo em que os aplicativos são dados objetos handlers enquanto a implementação está sob controle do recurso físico real.

O aplicativo obtém acesso a recursos físicos especializados selecionando um ou muitos a partir de uma lista de todos os recursos de um dado tipo e que possivelmente atendam a um dado critério de filtro. O método para recuperar todas as instâncias é fornecido por todas as classes de recurso implementando essa interface.

O gerenciamento real dos recursos físicos inluindo a coordenação de bloqueio e desbloqueio compartilham recursos entre a plataforma Java e outros componentes do sistema não é inserido no escopo desta especificação intencionalmente. No entanto, uma implementação em conformidade deve assegurar que o comportamento descrito abaixo seja garantido para aplicativos Java.

Design

Para estar em conformidade com o *design* comandado por esta especificação, as classes instanciáveis representando recursos físicos devem seguir as restrições a seguir:

- devem implementar esta interface,
- devem fornecer um método `getInstances` estático: Este método retorna um *array* de todas as instâncias de um dado tipo de recurso com base em um critério de filtro específico a cada classe de tipo de recurso.
- devem fornecer um método `reserveOne` estático: este método permite reservar qualquer instância de recurso de um mesmo tipo.
- devem fornecer um método `addResourceTypeListener` estático: este método anexa um *listener* para reservar e liberar eventos relacionados a todos os recursos de um mesmo tipo.
- devem fornecer um método `removeResourceTypeListener` estático: este método remove o *listener*.

A descrição para estes métodos adicionalmente requeridos por classes implementadoras desta interface pode ser vista na seção 46.6.3.

Permissão

A permissão para acessar o método `reserve` e seu argumento `force` é governado e controlado pela `ScarceResourcePermission`.

Exemplo de Código

Uma classe de recurso tipicamente seguiria a amostra a seguir de um display LCD *1-liner* embutido imaginário.

```
import com.sun.dtv.resources.*;

// Esta classe representa um componente físico real

// e é interno para implementação

public class LCDDevice {
    static LCD instances[] = { new LCDDevice(80) };
    boolean isLocked = false;
    int length = 0;
    //...
}

// Esta classe representa um handler para o dispositivo LCD anterior

// e é disponibilizada para os aplicativos

public class LCDHandler implements ScarceResource {

    // Interface : ScarceResource

    public void reserve(boolean force,
                        long timeoutms,
                        ScarceResourceListener listener)
        throws IllegalArgumentException,
               TimeoutException { ... }

    public void release() { ... }

    public boolean isAvailable() { ... }

    // Requerido getInstances


    public static LCDHandler[] getInstances() {
        // gera uma nova lista de LCDHandler

    }

    public static LCDHandler reserveOne(boolean force,
                                        long timeoutms,
                                        ScarceResourceListener,
                                        int type)
        throws SecurityException,
               IllegalArgumentException,
               TimeoutException {
        // ...
    }

    public static void addResourceTypeListener(ResourceTypeListener listener)
        throws NullPointerException {
        // ...
```

```
    }
   public static void removeResourceTypeListener(ResourceTypeListener listener)
        throws NullPointerException {
        // ...
    }
    // API específica da classe LCD

    public void display(String text) {
        // Acessa o controlador LCD e atribui o texto
        // ...
    }
    public void clear() {
        display("");
    }
}
```

Consequentemente, um aplicativo pode utilizar o seguinte código:

```
LCDHandler lcd = LCDHandler.getInstances()[0]; // you know what you do!
try {
    lcd.reserve(false, 0, this);
    // o LCD agora é nosso!
    lcd.display("Hello LCD");
}
catch(TimeoutException e) {
    // LCD não pode ser reservado!
}
```

Outro exemplo pode ser encontrado na descrição de classe `NetworkDevice`.

Relaciona-se com:

`ScarceResourcePermission`, `NetworkDevice`

43.6.2 Índice de métodos

`boolean` **`isAvailable`**`()`

Verifica se o dado recurso está correntemente disponível para reserva.

`void` **`release`**`()`

Libera esse recurso.

`void` **`reserve`**`(boolean force, long timeoutms, ScarceResourceListener listener)`

Requisita reserva da dada instância de recursos escassos.

43.6.3 Detalhe dos métodos requeridos nas classes implementadoras

getInstances

`getInstances()`

O método `getInstances` deve estar de acordo com a assinatura a seguir e pode fornecer argumentos opcionais que definem critério de filtro de granulação fina:

`public static` *`ResourceClass`*`[] getInstances(`*`filterCriteria`*`);`

Retorna um *array* de handlers cada um representando um recurso físico do dado tipo disponível na plataforma que atende ao critério definido pelo *filterCriteria*. Cada objeto handler é único tanto para aplicativo como para plataforma. Cada objeto handler pode ser diferente para cada um deles em chamadas subsequentes para o método getInstances(). A lista contém todas as instâncias, estejam ou não reservadas.

O *filterCriteria* dos critérios permite que o aplicativo foque em um subconjunto restrito da *ResourceClass*.

Se nenhuma *ResourceClass* existir, esse método retorna um *array* de comprimento zero.

Retorna:

Um *array* de instâncias de recursos do tipo *ResourceClass*.

reserveOne

reserveOne()

O método reserveOne() deve estar de acordo com a assinatura a seguir e pode fornecer argumentos opcionais específicos à classe de recurso escasso especializado:

```
public static ResourceClass reserveOne(boolean force, long timeoutms, ScarceResourceListener
listener,                                                                          ...)
   throws                                                                  IllegalException,
          TimeoutException,
          SecurityException;
```

Retorna uma instância reservada fora do pool de todas as intâncias do tipo *ResourceClass*. Isso pode estar sujeito aos parâmetros opcionais se algum estiver definido. Observar que este método retorna com a instância que foi reservada ou executa uma exceção de acordo com a situação.

Este método se comporta exatamente como o reserve() descrito na interface ScarceResource.

Parâmetros:

force - Se *true*, esse método tem permissão de retirar qualquer recurso do atual proprietário. Se *false*, a implementação bloqueará e esperará até que um recurso esteja disponível (utilizando release()) ou até timeoutms milissegundos.

timeoutms – Uma quantidade positiva de tempo em milissegundos até a qual este método esperará para que qualquer recurso seja liberado por seu proprietário atual. O valor -1 indica que a implementação esperará para sempre.

listener - O *listener* a ser anexado para receber notificação sobre o status de recurso reservado.

Retorna:

A instância de tipo *ResourceClass* que foi reservada.

Lança:

SecurityException – Se o aplicativo não tem permissão para a ação reserve para o recurso que está prestes a reservar. Também lançado se forçar for definido *true*, mas o aplicativo não tiver permissão para a ação forçar.

IllegalArgumentException – Se o listener for null ou se o valor especificado em timeoutms não é válido, por exemplo, um número inteiro positivo ou -1.

TimeoutException – Se o tempo especificado em timeoutms acabar e o recurso não puder ser reservado.

addResourceTypeListener

`addResourceTypeListener()`

```
public static void addResourceTypeListener(ResourceTypeListener listener)
   throws NullPointerException;
```

Adiciona um `ResourceTypeListener` a implementação. Sempre que um `reserve()` ou um `release()` for chamado para quaisquer recursos do mesmo tipo, a implementação chamará os métodos correspondentes do *listener*.

Parâmetros:

`listener` - O *listener* desencadeado para eventos em recursos do mesmo tipo.

Lança:

`NullPointerException` – Se o *listener* for `null`.

removeResourceTypeListener

`removeResourceTypeListener()`

```
public static void removeResourceTypeListener(ResourceTypeListener listener)
   throws NullPointerException;
```

Remove um *listener* previamente anexado. Se o *listener* não foi anexado antes, esse método não faz nada..

Parâmetros:

`listener` - O *listener* desencadeado para eventos em recursos do mesmo tipo.

Lança:

`NullPointerException` – Se o *listener* for `null`.

43.6.4 Detalhe dos métodos

reserve

```
void reserve(boolean force,
             long timeoutms,
             ScarceResourceListener listener)
      throws IllegalArgumentException,
             TimeoutException,
             SecurityException
```

Requisita reserva da dada instância de recursos escassos. O método bloquear-se-á até que a instância esteja disponível. O método retorna normalmente apenas se a reserva for bem sucedida. Todos os outros casos são manuseados por exceções.

Primeiro, se houver um gerenciador de segurança, seu método `checkPermission` é chamado com a permissão `ScarceResourcePermission(name, "reserve")`. Se `force` é além disso definido *`true`*, a permissão também verificado no `ScarceResourcePermission(name, "force")`.

Durante o processo de reserva, se a instância do recurso já estiver alocada, a implementação deve notificar o proprietário atual do recurso sobre a requisição de reserva por meio da interface `ScarceResourceListener`:

ou por `ScarceResourceListener.releaseRequested()` se forçar for *false*,

ou por `ScarceResourceListener.releaseForced()` no outro caso.

O *listener* deve ser usado para tais notificações apenas até o recurso ser liberado. Após a liberação, a implementação não deve chamar quaisquer dos métodos da interface do *listener*.

Após esse primeiro evento, a implementação procederá de acordo e liberará (ou não) o recurso requisitado. No caso da implementação liberar o recurso, desencadeará um evento `ScarceResourceListener.released()` ao proprietário *listener* original do recurso informando-o que o recurso não lhe pertence mais.

O aplicativo pode controlar o tempo de espera para que tal recurso esteja disponível ao definir `timeoutms`. No caso da duração da reserva exceder o tempo expresso em `timeoutms`, um `TimeoutException` é lançado.

Sob operação normal, recursos são liberados usando o método `release`. De qualquer forma, no caso do aplicativo não liberar recursos quando requisitado ou o aplicativo ser terminado, a implementação deve liberar todos os recursos alocados para o aplicativo para permitir que outros aplicativos sejam notificados de mudanças na alocação de recursos e poderem reservá-los. Ver a seção Resource Cleanup do ciclo de vida do aplicativo.

Parâmetros:

`force` - Se *`true`*, esse método retirará o recurso do proprietário atual. Se *`false`*, a implementação bloquear-se-á e esperará até que o recurso esteja disponível (usando `release()`) ou até `timeoutms`.

`timeoutms` - Uma quantidade positiva de milissegundos que esse método esperará para que o recurso seja liberado pelo proprietário atual. O valor `-1` indica que a implementação esperará para sempre.

`listener` - O *listener* a ser anexado para receber notificação sobre o status do recurso.

Lança:

`IllegalArgumentException` - Se o *listener* estiver `null` ou se o valor especificado em `timeoutms` não for válido, isto é, nem um inteiro positivo, nem -1.

`TimeoutException` - Se o tempo especificado em `timeoutms` houver acabado e o recurso não houver podido ser reservado.

`SecurityException` - Se o aplicativo não tiver permissão para ação de reserva para o recurso que está prestes a reservar. Também lançado se forçar for definido *true*, mas o aplicativo não tiver permissão para a ação forçar.

release

```
void release()
```

Libera esse recurso.

O recurso deve ser disponibilizado para outro aplicativo através da plataforma. Após chamar esse método, não é mais possível interagir com o dado recurso: chamadas para métodos críticos desses recursos escassos devem ser ignorados e lançar `IllegalStateException`. Essa afirmação é válida e é o comportamento requerido de qualquer classe implementando a interface `ScarceResource`. Para interagir novamente com o dado recurso, o aplicativo deve chamar o método `reserve()` e se tornar o proprietário novamente.

A implementação pode organizar quaisquer recursos do sistema que esse objeto estiver usando. Depois a implementação não deve chamar quaisquer dos métodos do *listener* que estava anexado no momento da reserva.

Se o recurso já estava disponível (isto é, não reservado), esse método não faz nada.

isAvailable

```
boolean isAvailable()
```

Verifica se o dado recurso está correntemente disponível para reserva. O valor retornado dá a situação atual e não garante que o recurso ainda estará disponível em um momento posterior, por exemplo, no momento da reserva: outro aplicativo pode ter tomado aquele recurso no tempo decorrido.

Retorna:

Um boolean definido *true* se o dado recurso estiver atualmente disponível para reserva.

43.7 Interface ScarceResourceListener

43.7.1 Descrição da interface

com.sun.dtv.resources

```
public interface ScarceResourceListener
```

Listener a ser notificado sobre o status de liberação de um dado recurso escasso.

Ao chamar qualquer método desta interface, certas plataformas podem definir um timespan até que aqueles métodos retornem. Caso não retornem dentro do timeframe, a implementação deve se comportar como se o método retornasse. Esse mecanismo permite evitar que aplicativos bloqueiem esses métodos desnecessariamente.

43.7.2 Índice de métodos

void **released**(ScarceResource resource)

Reporta que resource foi liberado pela implementação e está disponível para uma nova reserva.

void **releaseForced**(ScarceResource resource)

Reporta que resource está sendo solicitado insistentemente por outro aplicativo e dá ao proprietário atual a chance de fechar sua tarefa.

boolean **releaseRequested**(ScarceResource resource)

Reporta que resource está sendo solicitado por outro aplicativo.

43.7.3 Detalhe dos métodos

releaseRequested

```
boolean releaseRequested(ScarceResource resource)
```

Reporta que resource está sendo solicitado por outro aplicativo. O proprietário atual é livre para decidir se pode e quer liberar o dado recurso a seu competidor.

Se este m;etodo retornar *false*, a implementação entende que o proprietário atual quer manter o recruso e não está, portanto, disponível para o competidor.

Por outro lado, retornar *true* informa a plataforma que o proprietário atual está pronto para liberar seu recurso. Nesse caso a implementação deve liberar o dado recurso. Por sua vez, acionará um evento released nessa interface.

Quando suportado, caso esse método não retorne dentro de um dado timeframe, a implementação deve se comportar como se o método retornasse *true*.

Parâmetros:

resource – Instância do recurso escasso solicitado para ser liberado.

Retorna:

Um boolean indicando se o aplicativo atual e o proprietário não precisam mais do recurso.

releaseForced

```
void releaseForced(ScarceResource resource)
```

Reporta que `resource` está sendo solicitado insistentemente por outro aplicativo e dá ao proprietário atual a chance de fechar sua tarefa. Como resposta a esta chamada, a implementação deve liberar esse recurso. Por sua vez, acionará um evento `released` nessa interface.

Parâmetros:

`resource` – Instância do recurso escasso necessário para ser liberado.

released

```
void released(ScarceResource resource)
```

Reporta que `resource` foi liberado pela implementação e está disponível para uma nova reserva. Este método é chamdado pela implementação subsequente para um `ScarceResource.release()` ou após os eventos `releaseForced()` ou `releaseRequested()`.

Quando a implementação chama este método, o objeto terá sido invalidado, ou seja, o aplicativo que possuía esse recurso não é mais capaz de interagir com ele: as chamadas são ignoradas e os métodos podem executar uma `IllegalStateException`.

Parâmetros:

`resource` – Instância do recurso escasso que foi liberado.

43.8 Classe ScarceResourcePermission

43.8.1 Descrição da classe

`com.sun.dtv.resources`

```
java.lang.Object
  └java.security.Permission
      └java.security.BasicPermission
          └com.sun.dtv.resources.ScarceResourcePermission
```

Todas as interfaces implementadas

```
Guard, Serializable
```

```
final public class ScarceResourcePermission
extends BasicPermission
```

Esta classe é utilizada por várias permissões relacionadas a recursos escassos. Uma `ScarceResourcePermission` contém um nome que corresponde ao nome específico ao dado recurso escasso e uma lista de ações permitidas.

Nome

As regras de nomeação seguem aquelas definidas em `BasicPermission` da qual esta classe herda. Observar que há a possibilidade de endereçar todos os recursos escassos de uma só vez utilizando o asterisco sozinho.

Todas as classes que representam um recurso escasso e implementam a interface `carceResource` devem, onde aplicável, definir o nome da permissão correspondente. A regra-padrão para o nome de permissão é definida para ser correspondente à classe Java em caixa-baixa.

Ações

As ações a serem conferidas são passadas ao construtor em uma *string* contendo uma lista de zero ou mais lista de ações separadas por vírgulas. Essa especificação define as seguintes palavras-chave e significados a serem suportados para cada tipo de recurso escasso:

force

Permite que o dado aplicativo force a reserva do recurso nomeado. Permite que *ResourceClass*.reserve(*true*, ...) seja chamado (ou seja, com o argumento force configurado para *true*).

reserve

Permite que o dado aplicativo reserve o recurso nomeado. Permite que *ResourceClass*.reserve seja chamado.

A *string* de ação é convertida para caixa-baixa antes do processamento. A representação canônica da lista de ações concedidas é definida para ser a lista de ações presentes em ordem alfabética.

ResourceClass é definido como um objeto do tipo representado pelo nome da permissão (por exemplo, se o nome é "networkdevice", então *ResourceClass* é uma classe do tipo NetworkDevice).

Observar que a ação force não implica a ação reserve mesmo se a primeira não fizer sentido sem a segunda.

43.8.2 Índice de construtores

ScarceResourcePermission(String name, String actions)

Cria um novo objeto ScarceResourcePermission com nome e ações específicos.

43.8.3 Índice de métodos

String **getActions**()

Retorna a "Representação do *string* canônico" das ações.

43.8.4 Detalhe dos construtores

ScarceResourcePermission

```
public ScarceResourcePermission(String name,
                                String actions)
```

Cria um novo objeto ScarceResourcePermission com nome e ações específicos. O name é o nome do recurso escasso especializado (por exemplo. "networkdevice" ...) e actions contém uma lista separada por vírgulas das ações desejadas concedidas para tal recurso escasso.

Parâmetros:

name - nome do ScarceResourcePermission.

actions – *string* de ações.

43.8.5 Detalhe dos métodos

getActions

```
public String getActions()
```

Retorna a "Representação do *string* canônico" das ações. Ou seja, esse método sempre retorna ações presentes em ordem alfabética. force, reserve. Por exemplo, se este objeto ScarceResourcePermission permite ambas ações reserve, force, uma chamada para getActions retornará a *string* "force,reserve".

Substituições:

getActions na classe BasicPermission

Retorna:

A representação canônica da *string* das ações.

43.9 Classe TimeoutException

43.9.1 Descrição da classe

com.sun.dtv.resources

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └com.sun.dtv.resources.TimeoutException
```

Todas as interfaces implementadas

`Serializable`

```
public class TimeoutException
extends Exception
```

Indica a ocorrência de um *timeout*.

43.9.2 Índice de construtores

`TimeoutException``()`

Constrói uma `TimeoutException` sem nenhuma mensagem detalhada.

`TimeoutException``(String s)`

Constrói uma `TimeoutException` sem nenhuma mensagem de detalhe especificada.

43.9.3 Detalhe dos construtores

TimeoutException

```
public TimeoutException()
```

Constrói uma `TimeoutException` sem nenhuma mensagem detalhada.

TimeoutException

```
public TimeoutException(String s)
```

Constrói uma `TimeoutException` sem nenhuma mensagem de detalhe especificada.

Parâmetros:

`s` – Mensagem detalhada sobre as razões da exceção.

44 Pacote com.sun.dtv.security

44.1 Descrição do pacote

Pacote que define funcionalidades adicionais de segurança. As funcionalidades básicas são providas pelas classes do pacote `java.security`.

44.2 Índice de interfaces

Callback

Implementações desta interface são passadas para um `CallbackHandler`, permitindo que serviços-base de segurança interajam com um aplicativo para recuperar dados de autenticação específicos, como nome de usuário e senha, ou exibam certas informações, como mensagens de erro e alertas.

CallbackHandler

Um aplicativo implementa um `CallbackHandler` e passa isso para os serviços-base de segurança para que possam interagir com o apllicativo para recuperar dados específicos de autenticação, como nome de usuário e senha, ou exibir certas informações, como mensagens de erro e alertas.

A Figura 15 mostra a estrutura do pacote `Security` do Java DTV.

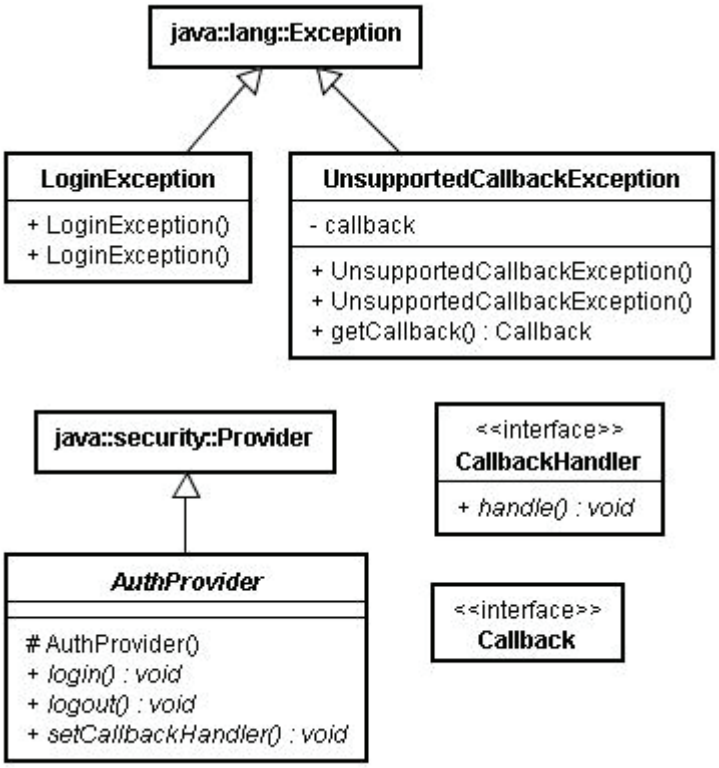


Figura 15 – Pacote `Security`

NOTA Este pacote está especificado desde o Java DTV 1.0.

44.3 Índice de classes

AuthProvider

Esta classe define métodos de login e logout para um fornecedor.

44.4 Índice de exceções

LoginException

Essa é a exceção de login básica.

UnsupportedCallbackException

Executada quando um callback handler é solicitado a lidar com um callback que não pode ser manipulado.

44.5 Classe AuthProvider

44.5.1 Descrição da classe

com.sun.dtv.security

```
java.lang.Object
  └java.util.Dictionary
      └java.util.Hashtable
          └java.util.Properties
              └java.security.Provider
                  └com.sun.dtv.security.AuthProvider
```

Todas as interfaces implementadas

```
Cloneable, Map, Serializable
```

```
abstract public class AuthProvider
extends Provider
```

Esta classe define métodos de login e logout para um fornecedor.

Enquanto os chamadores podem chamar o login diretamente, o fornecedor também pode chamar o login em nome dos chamadores se ele determinar que um login deve ser realizado antes de certas operações.

44.5.2 Índice de construtores

`protected` **`AuthProvider`**`(String name, double version, String info)`

Constrói um fornecedor com o nome, número da versão e informações especificados.

44.5.3 Índice de métodos

`abstract void` **`login`**`(Principal principal, CallbackHandler handler)`

Login para este fornecedor.

`abstract void` **`logout`**`()`

Logout deste fornecedor.

`abstract void` **`setCallbackHandler`**`(CallbackHandler handler)`

Configura um `CallbackHandler`.

44.5.4 Detalhe dos construtores

AuthProvider

```
protected AuthProvider(String name,
```

```
                    double version,
                    String info)
```

Constrói um fornecedor com o nome, número da versão e informações especificados.

Parâmetros:

`name` – nome do fornecedor

`version` – número da versão do fornecedor.

`info` – uma descrição do fornecedor e seus serviços.

44.5.5 Detalhe dos métodos

login

```
public abstract void login(Principal principal,
                           CallbackHandler handler)
                    throws LoginException
```

`Login` para este fornecedor. O fornecedor recorre a um `CallbackHandler` para obter informações de autenticação do chamador (por exemplo, um PIN). Se o chamador passa um handler `null` para este método, o fornecedor utiliza o conjunto do handler no método `setCallbackHandler`.

Parâmetros:

`principal` - o `Principal` que pode conter nomes/identificadores de *login* utilizados para autenticação ou ser preenchido com nomes/*logins* adicionais após autenticação bem-sucedida. Este parâmetro pode ser `null`.

`handler` - `CallbackHandler` utilizado por seu fornecedor para obter autenticação do chamador, que pode ser `null`

Lança:

`LoginException` – se a operação de login falhar

logout

```
public abstract void logout()
                     throws LoginException
```

Logout deste fornecedor.

Lança:

`LoginException` – se a operação de *logout* falhar

setCallbackHandler

```
public abstract void setCallbackHandler(CallbackHandler handler)
```

Configura um `CallbackHandler`. O fornecedor utiliza seu *handler* se um não for passado ao método de *login*. O fornecedor também utiliza seu *handler* se ele chamar o *login* em nome dos chamadores. Se o `CallbackHandler` não estiver configurado, pressupõe-se que o fornecedor tenha meios alternativos para obter as informações de autenticação.

Parâmetros:

`handler` - um `CallbackHandler` para obter informações de autenticação, que pode ser `null`.

44.6 Interface Callback

44.6.1 Descrição da interface

com.sun.dtv.security

```
public interface Callback
```

Implementações desta interface são passadas para um `CallbackHandler`, permitindo que serviços-base de segurança interajam com um aplicativo para recuperar dados de autenticação específicos, como nome de usuário e senha, ou exibam certas informações, como mensagens de erro e alertas. As implementações de `Callback` não recupera ou exibe a informação solicitada por serviços-base de segurança. As implementações `Callback` simplesmente fornecem os meios de passar tais solicitações aos aplicativos, e para aplicativos, se adequado, para retornar informações solicitadas aos serviços-base de segurança.

44.7 Interface CallbackHandler

44.7.1 Descrição da interface

com.sun.dtv.security

```
public interface CallbackHandler
```

Um aplicativo implementa um `CallbackHandler` e passa isso para os serviços-base de segurança para que possam interagir com o apllicativo para recuperar dados específicos de autenticação, como nome de usuário e senha, ou exibir certas informações, como mensagens de erro e alertas.

44.7.2 Índice de métodos

`void` **`handle`**`(Callback[] callbacks)`

Chamado quando um *callback* precisa ser manipulado.

44.7.3 Detalhe dos métodos

handle

```
void handle(Callback[] callbacks)
     throws IOException,
            UnsupportedCallbackException
```

Chamado quando um *callback* precisa ser manipulado.

Parâmetros:

`callbacks` - um *array* de objetos `Callback` fornecido por um serviço-base de segurança que contém a informação a ser recuperada ou exibida.

Lança:

`IOException` – se ocorrer um erro de entrada ou saída ao recuperar a senha

`UnsupportedCallbackException` – se um dos callbacks no *array* não for suportado pelo *handler*

44.8 Classe LoginException

44.8.1 Descrição da classe

com.sun.dtv.security

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └com.sun.dtv.security.LoginException
```

Todas as interfaces implementadas

`Serializable`

```
public class LoginException
extends Exception
```

Essa é a exceção de login básica.

44.8.2 Índice de construtores

`LoginException``()`

Constrói um `LoginException` sem nenhuma mensagem de detalhe.

`LoginException``(String msg)`

Constrói uma `LoginException` sem nenhuma mensagem de detalhe especificada.

44.8.3 Detalhe dos construtores

LoginException

`public LoginException()`

Constrói um `LoginException` sem nenhuma mensagem de detalhe.

LoginException

`public LoginException(String msg)`

Constrói uma `LoginException` sem nenhuma mensagem de detalhe especificada. Uma mensagem de detalhe é uma *string* que descreve essa exceção em particular.

Parâmetros:

`msg` – mensagem de detalhe.

44.9 Classe UnsupportedCallbackException

44.9.1 Descrição da classe

com.sun.dtv.security

```
java.lang.Object
```

```
java.lang.Throwable
    java.lang.Exception
        com.sun.dtv.security.UnsupportedCallbackException
```

Todas as interfaces implementadas

`Serializable`

```
public class UnsupportedCallbackException
extends Exception
```

Executada quando um callback handler é solicitado a lidar com um callback que não pode ser manipulado.

44.9.2 Índice de construtores

`UnsupportedCallbackException``(Callback callback)`

Cria uma nova instância de `UnsupportedCallbackException` sem mensagem de detalhe.

`UnsupportedCallbackException``(Callback callback, String msg)`

Constrói uma `UnsupportedCallbackException` sem nenhuma mensagem de detalhe especificada.

44.9.3 Índice de métodos

`Callback` **`getCallback`**`()`

Obtém o `Callback` irreconhecível.

44.9.4 Detalhe dos construtores

UnsupportedCallbackException

```
public UnsupportedCallbackException(Callback callback)
```

Cria uma nova instância de `UnsupportedCallbackException` sem mensagem de detalhe.

Parâmetros:

`callback` - o `Callback` que não pode ser manipulado

UnsupportedCallbackException

```
public UnsupportedCallbackException(Callback callback,
                                    String msg)
```

Constrói uma `UnsupportedCallbackException` sem nenhuma mensagem de detalhe especificada. Uma mensagem de detalhe é uma *string* que descreve essa exceção em particular.

Parâmetros:

`callback` - o `Callback` irreconhecível.

`msg` – mensagem de detalhe.

44.9.5 Detalhe dos métodos

getCallback

`public Callback getCallback()`

Obtém o `Callback` irreconhecível.

Retorna:

`Callback` irreconhecível.

45 Pacote com.sun.dtv.service

45.1 Descrição do pacote

Esse pacote provê uma interface para acessar o banco de dados SI de Baixo Nível.

Este pacote contém a API de SI para acessar o banco de dados do SI de nível baixo que contém as tabelas de SI. A classe neste pacote fornece um modo de obter uma referência ao Banco de dados de SI de nível baixo, que depende do sistema de transporte. Uma implementação SIDatabase real deve ser derivada do SIDatabase como definido neste pacote e fornecer métodos de acesso e definições de classe para as tabelas SI individuais.

A Figura 16 mostra a estrutura do pacote `Service` do Java DTV.

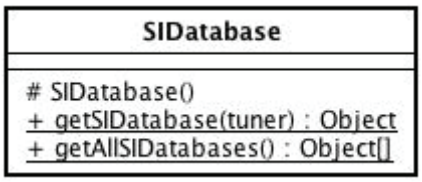


Figura 16 – Pacote `Service`

Este pacote descreve uma abstração comum para fornecer acesso ao SI de nível baixo do sistema de transmissão.

Para ser totalmente flexível o `SIDatabase` fornece uma classe de base comum, da qual a classe de implementação `SIDatabase` real é derivada. Esse mecanismo permite utilizar diferentes formatos de banco de dados SI como fornecido no sistema de transmissão real.

Para esse propósito ele define uma classe de SIDatabase, que deve ser utilizada como uma classe de base para implementações de banco de dados SI real. A partir dessa classe de base, uma implementação de banco de dados SI, um banco de dados SI fornece meios para investigações assíncronas ou tabelas de SI monitor, que são transmitidas no stream de transporte MPEG-2. A especificação MPEG-2 define o formato comum das tabelas SI e ordena um conjunto mínimo de tabelas. Padrões de transmissão reais, como DVB, ARIB ou ATSC definem tabelas adicionais, que correspondem ao SI utilizado nesta rede. Isso depende do padrão de transmissão e localização geográfica e que tipos de tabela são suportados.

NOTA Este pacote está especificado desde o Java DTV 1.0.

45.2 Índice de classes

SIDatabase

45.3 Classe SIDatabase

45.3.1 Descrição da classe

com.sun.dtv.service

```
java.lang.Object
  └com.sun.dtv.service.SIDatabase

public class SIDatabase
extends Object
```

45.3.2 Índice de construtores

`protected` **`SIDatabase`**`()`

Construtor protegido – não pode ser chamado diretamente.

45.3.3 Índice de métodos

`static Object[]` **`getAllSIDatabases`**`()`

Retorna um *array* de todas as instâncias SIDatabase no sistema.

`static Object` **`getSIDatabase`**`(Tuner tuner)`

Retorna a instância SIDatabase que corresponde ao sistema nativo.

45.3.4 Detalhe dos construtores

SIDatabase

```
protected SIDatabase()
```

Construtor protegido – não pode ser chamado diretamente.

45.3.5 Detalhe dos métodos

getSIDatabase

```
public static Object getSIDatabase(Tuner tuner)
                            throws NullPointerException
```

Retorna a instância SIDatabase que corresponde ao sistema nativo.

O `tuner` é utilizado para selecionar a interface.

Parâmetros:

`tuner` - `Tuner` do qual obter o banco de dados do SI.

Retorna:

instância SIDatabase.

Lança:

`NullPointerException` – se o `tuner` for `null`.

getAllSIDatabases

```
public static Object[] getAllSIDatabases()
```

Retorna um *array* de todas as instâncias SIDatabase no sistema.

Retorna:

um *array* de todas as instâncias SIDatabase.

46 Pacote com.sun.dtv.smartcard

46.1 Descrição do pacote

Pacote que provê funcionalidades adicionais de *smartcard*. Os recursos básicos são fornecidos pelo SATSA.

A Figura 17 mostra a estrutura do pacote `Smartcard` do Java DTV.

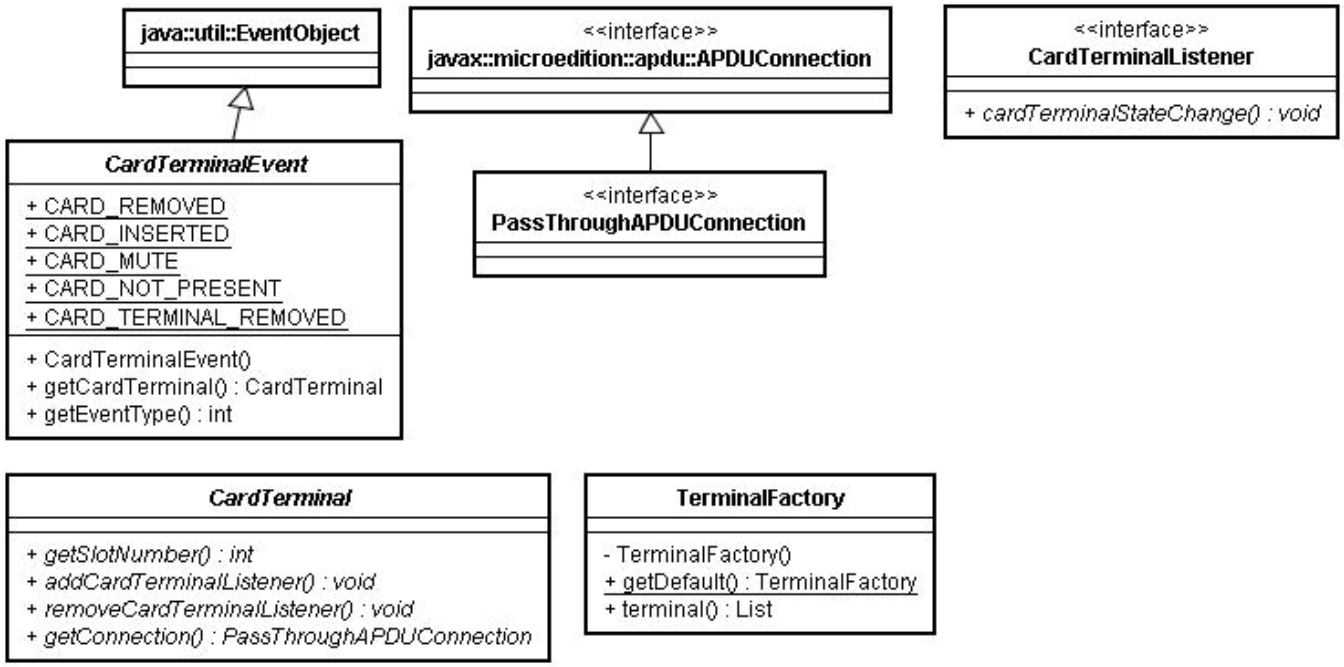


Figura 17 – Pacote `Smartcard`

NOTA Este pacote está especificado desde o Java DTV 1.0.

46.2 Índice de interfaces

CardTerminalListener

Interface `Listener` para receber eventos de terminal de cartão.

PassThroughAPDUConnection

Essa é uma Tagging Interface para identificar um objeto de conexão APDU específico para comunicação direta com o smartcard.

46.3 Índice de classes

CardTerminal

Encapsula a funcionalidade de um terminal de cartão físico.

CardTerminalEvent

Define um objeto de evento que informa os *listeners* sobre alterações de estado que ocorreram em um `CardTerminal`.

TerminalFactory

Fornece acesso ao(s) Terminal(is) de Cartão implementado ou conectado ao dispositivo.

46.4 Classe CardTerminal

46.4.1 Descrição da classe

`com.sun.dtv.smartcard`

```
java.lang.Object
  └com.sun.dtv.smartcard.CardTerminal
```

```
abstract public class CardTerminal
extends Object
```

Encapsula a funcionalidade de um terminal de cartão físico.

46.4.2 Índice de construtores

`CardTerminal``()`

46.4.3 Índice de métodos

`abstract void` **`addCardTerminalListener`**`(CardTerminalListener listener)`

Adiciona um novo `CardTerminalListener` para o `CardTerminal`.

`abstract PassThroughAPDUConnection` **`getConnection`**`(int slotNumber)`

Cria uma `APDUConnection` para o dado `CardTerminal` com o dado número de *slot*.

`abstract int` **`getSlotNumber`**`()`

Recupera o número de slot deste objeto `CardTerminal`, o *framework* é responsável por atribuir um número de *slot* a cada terminal de cartão embutido ou conectado ao dispositivo.

`abstract void` **`removeCardTerminalListener`**`(CardTerminalListener listener)`

Remove um novo `CardTerminalListener` para o `CardTerminal`.

46.4.4 Detalhe dos construtores

CardTerminal

```
public CardTerminal()
```

46.4.5 Detalhe dos métodos

getSlotNumber

```
public abstract int getSlotNumber()
```

Recupera o número de slot deste objeto `CardTerminal`, o framework é responsável por atribuir um número de slot a cada terminal de cartão embutido ou conectado ao dispositivo.

Retorna:

número de *slot* que é atribuído ao `CardTerminal`

addCardTerminalListener

```
public abstract void addCardTerminalListener(CardTerminalListener listener)
```

Adiciona um novo `CardTerminalListener` para o `CardTerminal`.

Parâmetros:

`listener` – receberá eventos deste objeto de `CardTerminal`

Relaciona-se com:

`removeCardTerminalListener()`

removeCardTerminalListener

```
public abstract void removeCardTerminalListener(CardTerminalListener listener)
```

Remove um novo `CardTerminalListener` para o `CardTerminal`.

Parâmetros:

`listener` – ser removido do `CardTerminal`

Relaciona-se com:

`addCardTerminalListener()`

getConnection

```
public abstract PassThroughAPDUConnection getConnection(int slotNumber)
```

Cria uma `APDUConnection` para o dado `CardTerminal` com o dado `slotNumber`. É este `APDUConnection` específico que permite que todos os tipos de comando APDU passem. Nenhuma verificação é permitida no nível de framework como descrito no SATSA para essa conexão.

Parâmetros:

`slotNumber` – Identiticador do *slot*.

Retorna:

Uma conexão para enviar APDUs de comando ao cartão e receber APDUs de resposta do cartão.

46.5 Classe CardTerminalEvent

46.5.1 Descrição da classe

`com.sun.dtv.smartcard`

`java.lang.Object`

```
└java.util.EventObject
    └com.sun.dtv.smartcard.CardTerminalEvent
```

Todas as interfaces implementadas

`Serializable`

```
abstract public class CardTerminalEvent
extends EventObject
```

Define um objeto de evento que informa os *listeners* sobre alterações de estado que ocorreram em um `CardTerminal`.

46.5.2 Índice de campos

`static int` **`CARD_INSERTED`**

Corresponde ao evento quando um cartão é inserido.

`static int` **`CARD_MUTE`**

Corresponde ao evento quando um cartão é mudo.

`static int` **`CARD_NOT_PRESENT`**

Corresponde ao evento quando um cartão está presente.

`static int` **`CARD_REMOVED`**

Corresponde ao evento quando um cartão é removido.

`static int` **`CARD_TERMINAL_REMOVED`**

Corresponde ao evento quando o terminal do cartão é removido.

46.5.3 Índice de construtores

`CardTerminalEvent``(CardTerminal sourceTerminal)`

Constrói um `CardTerminalEvent` a partir da origem.

46.5.4 Índice de métodos

`CardTerminal` **`getCardTerminal`**`()`

Retorna o objeto `CardTerminal` onde este evento ocorreu.

`int` **`getEventType`**`()`

Retorna o topo de evento que aconteceu no terminal de cartão físico.

46.5.5 Detalhe dos campos

CARD_REMOVED

```
public static final int CARD_REMOVED = 1
```

Corresponde ao evento quando um cartão é removido.

CARD_INSERTED

```
public static final int CARD_INSERTED = 2
```

Corresponde ao evento quando um cartão é inserido.

CARD_MUTE

```
public static final int CARD_MUTE = 3
```

Corresponde ao evento quando um cartão é mudo.

CARD_NOT_PRESENT

```
public static final int CARD_NOT_PRESENT = 4
```

Corresponde ao evento quando um cartão está presente.

CARD_TERMINAL_REMOVED

```
public static final int CARD_TERMINAL_REMOVED = 5
```

Corresponde ao evento quando o terminal do cartão é removido.

46.5.6 Detalhe dos construtores

CardTerminalEvent

```
public CardTerminalEvent(CardTerminal sourceTerminal)
```

Constrói um `CardTerminalEvent` a partir da origem.

Parâmetros:

`sourceTerminal` – origem do evento.

46.5.7 Detalhe dos métodos

getCardTerminal

```
public CardTerminal getCardTerminal()
```

Retorna o objeto `CardTerminal` onde este evento ocorreu.

Retorna:

objeto `CardTerminal` onde o evento aconteceu.

getEventType

```
public int getEventType()
```

Retorna o topo de evento que aconteceu no terminal de cartão físico. DEVE ser um dos valores de número inteiro predefinidos nesta classe.

Retorna:

tipo de evento

46.6 Interface CardTerminalListener

46.6.1 Descrição da interface

com.sun.dtv.smartcard

`public interface CardTerminalListener`

Interface `Listener` para receber eventos de terminal de cartão.

46.6.2 Índice de métodos

`void` **`cardTerminalStateChange`**`(CardTerminalEvent cardTerminalEvent)`

Envia um `CardTerminalEvent` ao *listener*.

46.6.3 Detalhe dos métodos

cardTerminalStateChange

`void cardTerminalStateChange(CardTerminalEvent cardTerminalEvent)`

Envia um `CardTerminalEvent` ao *listener*.

Parâmetros:

`cardTerminalEvent` – evento que ocorreu em um terminal de cartão

46.7 Interface PassThroughAPDUConnection

46.7.1 Descrição da interface

com.sun.dtv.smartcard

Todas as super-interfaces

`APDUConnection, Connection`

`public interface PassThroughAPDUConnection`

`extends APDUConnection`

Essa é uma `Tagging Interface` para identificar um objeto de conexão APDU específico para comunicação direta com o *smartcard*. Não há nenhuma restrição no tipo de comando que pode ser enviado ao cartão por meio desta conexão, o framework não irá gerenciar canais ou selecionar um aplicativo de cartão por AID. Todas as palavras de status DEVEM ser transferidas literalmente como no cartão para o aplicativo e devem ser manipuladas ali. Uma chamada ao método `getATR` da interface `APUDConnection` resetará o cartão e transmitirá um novo ATR do cartão.

46.8 Classe TerminalFactory

46.8.1 Descrição da classe

com.sun.dtv.smartcard

java.lang.Object

└com.sun.dtv.smartcard.TerminalFactory

final public class TerminalFactory

extends Object

Fornece acesso ao(s) terminal(is) de cartão implementado ou conectado ao dispositivo. Um CardTerminal é identificado por um valor inteiro de número de *slot*. O set-top-box deve atribuir o número de *slot* zero ao terminal de cartão físico embutido. O número de *slot* pode ser utilizado para recuperar uma javax.microedition.apdu.APDUConnection.

46.8.2 Índice de métodos

static TerminalFactory **getDefault**()

Fornece a implementação-padrão do TerminalFactory disponível no dispositivo.

List **terminal**()

Retorna uma List de objetos CardTerminal que são embutidos ou anexados por meio de uma interface externa (por exemplo, USB) ao dispositivo.

46.8.3 Detalhe dos métodos

getDefault

public static TerminalFactory getDefault()

Fornece a implementação-padrão do TerminalFactory disponível no dispositivo.

Retorna:

a implementação-padrão do TerminalFactory no dispositivo.

terminal

public List terminal()

Retorna uma List de objetos CardTerminal que são embutidos ou anexados por meio de uma interface externa (por exemplo, USB) ao dispositivo.

Retorna:

lista de objetos CardTerminal cada um representando um terminal de cartão físico.

47 Pacote com.sun.dtv.test

47.1 Descrição do pacote

Provê um suporte extensivo de controle de testes.

Esse framework permite o desenvolvimento de um equipamento de teste extensivo com foco em testes de conformidade de aplicativos de suporte. Como encontrado tipicamente em ambiente de teste Java, tal ambiente

envolve um servidor e um cliente para o teste utilizando qualquer meio de comunicação para transferir um dado teste (ou suíte de testes) na plataforma, executá-lo e recuperar (ou receber) os resultados.

As classes a seguir e as interfaces compõem este pacote:

`TestHarness`: este é o ponto de entrada para realizar um teste ou uma suíte de testes e fornece um meio de comunicação com o servidor assim como acesso ao lote de teste local;

`TestCase`: todo teste deve implementar esta interface. Os parâmetros podem ser passados utilizando `args[]` que então devem ser considerados pelo teste.

`Report`: incorpora o resultado de um teste que engloba um código, uma referência e uma razão relacionada.

A Figura 18 mostra a estrutura do pacote `Testing` do Java DTV.

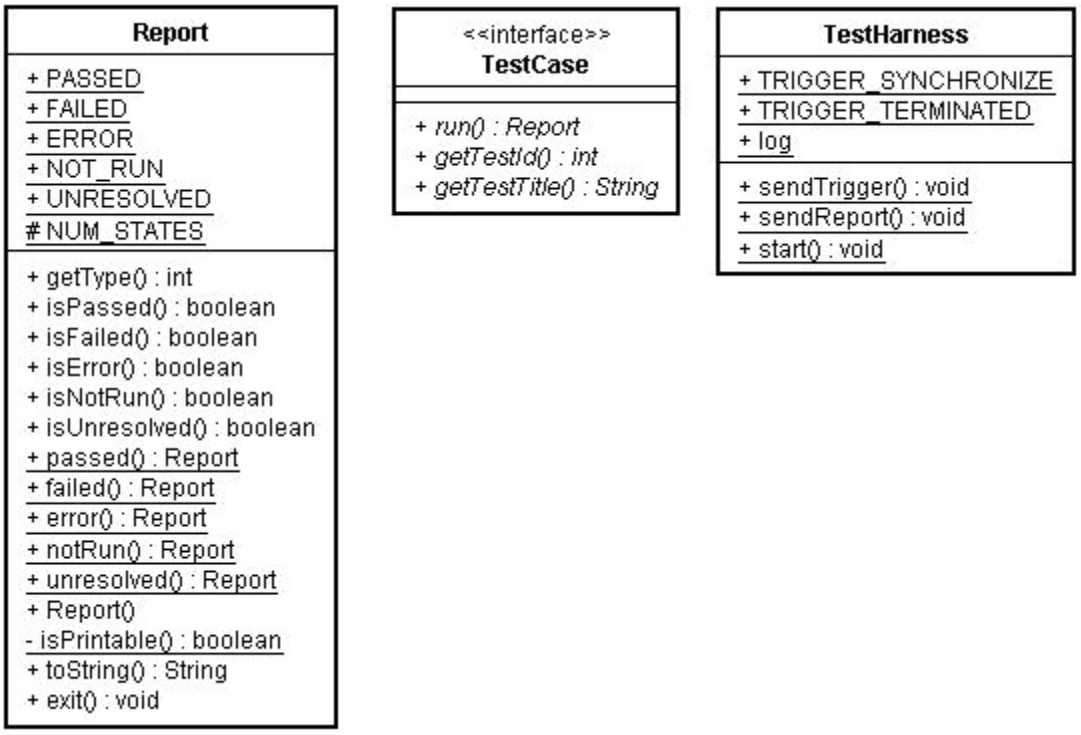


Figura 18 – Pacote `Testing`

NOTA Este pacote está especificado desde o Java DTV 1.0.

47.2 Índice de interfaces

TestCase

Define os métodos necessários que devem ser implementados pelos testes disponíveis para executar no framework de teste fornecido.

47.3 Índice de classes

Report

Incorpora o resultado de um teste: um código, uma referência ao teste e uma razão relacionada.

TestHarness

Fornece um ponto de entrada e um conjunto de ferramentação para casos de teste.

47.4 Classe Report

47.4.1 Descrição da classe

com.sun.dtv.test

```
java.lang.Object
  └com.sun.dtv.test.Report
```

```
public class Report
extends Object
```

Incorpora o resultado de um teste: um código, uma referência ao teste e uma razão relacionada.

47.4.2 Índice de campos

`static int` **`ERROR`**

Um tipo de relatório indicando que o teste não foi executado devido a ocorrência de algum erro antes de ele ser realizado.

`static int` **`FAILED`**

Um tipo de relatório indicando que o teste foi executado, mas o falhou.

`static int` **`NOT_RUN`**

Um tipo de relatório indicando que o teste ainda não foi realizado neste contexto.

`protected static int` **`NUM_STATES`**

Quantidade de tipos de relatório predefinidos como "constantes".

`static int` **`PASSED`**

Um tipo de relatório indicando que o teste foi executado com sucesso.

`static int` **`UNRESOLVED`**

Um tipo de relatório indicando que o teste gerou resultados imprecisos.

47.4.3 Índice de construtores

`Report``(int type, TestCase test, String reason)`

Cria um objeto Report.

47.4.4 Índice de métodos

`static Report` **`error`**`(TestCase test, String reason)`

Cria um relatório para indicar a ocorrência de um erro durante a tentativa de executar um teste: ou seja, o teste não foi completado por alguma razão, então não pode determinar se o elemento sendo testado passou ou falhou.

`void` **`exit`**`()`

Conveniência de função `exit()` para o `main()` de testes para sair de modo que o relatório passe pelos limites de processo sem perder informações, ou seja: códigos de saída não fornecem o texto associado do relatório e os códigos de retorno quando exceções são executadas podem causar resultados não-planejados.

`static Report` **`failed`**`(TestCase test, String reason)`

Cria um relatório para indicar o resultado sem sucesso de um teste: ou seja, ou seja, o teste foi concluído, mas determinou que o elemento sendo testado não passou no teste.

`int` **`getType`**`()`

Obtém o código de tipo indicando o tipo deste objeto `Report`.

`boolean` **`isError`**`()`

Verifica se o código de tipo do relatório é `ERROR`.

`boolean` **`isFailed`**`()`

Verifica se o código de tipo do relatório é `FAILED`.

`boolean` **`isNotRun`**`()`

Verifica se o código de tipo do relatório é `NOT_RUN`.

`boolean` **`isPassed`**`()`

Verifica se o código de tipo do relatório é `PASSED`.

`boolean` **`isUnresolved`**`()`

Verifica se o código de tipo do relatório é `UNRESOLVED`.

`static Report` **`notRun`**`(TestCase test, String reason)`

Cria um relatório para indicar que o teste ainda não foi executado.

`static Report` **`passed`**`(TestCase test, String reason)`

Cria um relatório para indicar o resultado bem-sucedido de um teste.

`String` **`toString`**`()`

Converte um relatório para uma *string*.

`static Report` **`unresolved`**`(TestCase test, String reason)`

Cria um relatório para indicar que o teste gerou resultados duvidosos.

47.4.5 Detalhe dos campos

PASSED

```
public static final int PASSED = 0
```

Um tipo de relatório indicando que o teste foi executado com sucesso.

Relaciona-se com:

`passed(), getType()`

FAILED

```
public static final int FAILED = 1
```

Um tipo de relatório indicando que o teste foi executado, mas o falhou.

Relaciona-se com:

`failed(), getType()`

ERROR

```
public static final int ERROR = 2
```

Um tipo de relatório indicando que o teste não foi executado devido a ocorrência de algum erro antes de ele ser realizado. Esse é geralmente um erro mais sério que `FAILED`.

Relaciona-se com:

`getType()`

NOT_RUN

```
public static final int NOT_RUN = 3
```

Um tipo de relatório indicando que o teste ainda não foi realizado neste contexto. Mais especificamente, nenhum arquivo de status foi registrado para esse teste no diretório de trabalho atual.

Relaciona-se com:

`getType()`

UNRESOLVED

```
public static final int UNRESOLVED = 4
```

Um tipo de relatório indicando que o teste gerou resultados imprecisos. Eles podem ocorrer se a execução do teste for interrompida de forma inesperada, se houver muitos alertas ou se o teste for configurado de forma incorreta. Isso geralmente significa que o operador do teste precisa analisar o *log* do teste assim como provavelmente melhorar o caso.

Relaciona-se com:

`getType()`

NUM_STATES

```
protected static final int NUM_STATES = 5
```

Quantidade de tipos de relatório predefinidos como "constantes".

47.4.6 Detalhe dos construtores

Report

```
public Report(int type,
              TestCase test,
              String reason)
       throws NullPointerException,
              IllegalArgumentException
```

Cria um objeto `Report`. Consulte `passed(TestCase, String)`, `failed(TestCase, String)`, `error(TestCase, String)` etc. para métodos de fábrica mais convenientes para criar objetos `Report`. Toda vez que um novo relatório for criado, deve ser registrado automaticamente no *stream* de saída definido pelo `TestHarness.log`. O formato é definido para ser o valor de retorno de um `toString()` anexo com um caractere "*carriage return*" (CR).

Parâmetros:

`type` – Código de tipo para o objeto `Report`.

`test` – Objeto `TestCase` de teste correspondente.

`reason` – Uma *string* curta para armazenar no relatório. Caracteres que não podem ser impressos (isto é, fora da faixa de 040C a 177C) na *string* são substituídos por um espaço.

Lança:

`NullPointerException` – se o teste ou razão for `null`.

`IllegalArgumentException` – se o tipo especificado for inválido.

47.4.7 Detalhe dos métodos

getType

`public int getType()`

Obtém o código de tipo indicando o tipo deste objeto `Report`.

Retorna:

código de tipo indicando o tipo deste objeto `Report`.

Relaciona-se com:

`PASSED`, `FAILED`, `ERROR`, `NOT_RUN`, `UNRESOLVED`

isPassed

`public boolean isPassed()`

Verifica se o código de tipo do relatório é `PASSED`.

Retorna:

true se o código de tipo for `PASSED`.

Relaciona-se com:

`passed()`, `getType()`, `PASSED`

isFailed

`public boolean isFailed()`

Verifica se o código de tipo do relatório é `FAILED`.

Retorna:

true se o código de tipo for `FAILED`.

Relaciona-se com:

`failed()`, `getType()`, `FAILED`

isError

`public boolean isError()`

Verifica se o código de tipo do relatório é `ERROR`.

Retorna:

true se o código de tipo for `ERROR`.

Relaciona-se com:

`error()`, `getType()`, `ERROR`

isNotRun

`public boolean isNotRun()`

Verifica se o código de tipo do relatório é `NOT_RUN`.

Retorna:

true se o código de tipo for `NOT_RUN`.

Relaciona-se com:

`notRun()`, `getType()`, `NOT_RUN`

isUnresolved

```
public boolean isUnresolved()
```

Verifica se o código de tipo do relatório é `UNRESOLVED`.

Retorna:

true se o código de tipo for `UNRESOLVED`.

Relaciona-se com:

`unresolved()`, `getType()`, `UNRESOLVED`

passed

```
public static Report passed(TestCase test,
                            String reason)
```

Cria um `Report` para indicar o resultado bem-sucedido de um teste.

Parâmetros:

`test` – Objeto de teste `TestCase` que passou.

`reason` - *String* curta descrevendo a razão pela qual o teste passou.

Retorna:

um relatório para indicar o resultado bem-sucedido de um teste.

failed

```
public static Report failed(TestCase test,
                            String reason)
```

Cria um `Report` para indicar o resultado sem sucesso de um teste: ou seja, o teste foi concluído, mas determinou que o elemento sendo testado não passou no teste.

Parâmetros:

`test` – Objeto de teste `TestCase` que falhou.

`reason` - *String* curta descrevendo a razão da falha do teste.

Retorna:

um relatório para indicar a falha de um teste.

error

```
public static Report error(TestCase test,
                           String reason)
```

Cria um `Report` para indicar a ocorrência de um erro durante a tentativa de executar um teste: ou seja, o teste não foi completado por alguma razão, então não pode determinar se o elemento sendo testado passou ou falhou.

Parâmetros:

`test` – Objeto de teste `TestCase` que apresentou um erro.

`reason` - *String* curta descrevendo o erro que ocorreu.

Retorna:

um relatório para indicar o resultado com erro de um teste.

notRun

```
public static Report notRun(TestCase test,
                            String reason)
```

Cria um `Report` para indicar que o teste ainda não foi executado.

Parâmetros:

`test` – Objeto de teste `TestCase` que não foi executado.

`reason` – *String* curta indicando a razão pela qual o teste ainda não foi executado.

Retorna:

um relatório para indicar o resultado de um teste não executado.

unresolved

```
public static Report unresolved(TestCase test,
                                String reason)
```

Cria um `Report` para indicar que o teste gerou resultados duvidosos.

Parâmetros:

`test` – Objeto de teste `TestCase`.

`reason` – *String* curta indicando a razão pela qual o teste gerou resultados duvidosos.

Retorna:

um relatório para indicar o resultado de um teste com resultados duvidosos.

toString

```
public String toString()
```

Converte um `Report` para uma `String`. A convenção é especificar o tipo de relatório, o identificador do teste (como retornado por `TestCase.getTestId()`) e a razão em uma linha separada.

Substituições:

`toString` na classe `Object`

Retorna:

Relatório completo em uma *string*.

exit

```
public void exit()
```

Conveniência de função `exit()` para o `main()` de testes para sair de modo que o relatório passe pelos limites de processo sem perder informações, ou seja: códigos de saída não fornecem o texto associado do relatório e os

códigos de retorno quando exceções são executadas podem causar resultados não-planejados.

Um marcador de identificação é designado para o stream de erro, cujo *script* executando o teste cuida do último resultado antes de retornar, seguido pelo tipo de razão.

O método não retorna. Tipicamente esse método chamaria o `System.exit` com um valor que depende do tipo.

47.5 Interface TestCase

47.5.1 Descrição da interface

`com.sun.dtv.test`

```
public interface TestCase
```

Define os métodos necessários que devem ser implementados pelos testes disponíveis para executar no *framework* de teste fornecido.

Consiste principalmente de um ponto de entrada e alguns métodos *helper*. O *design* do *framework* do teste é independente de plataforma e permite diferentes configurações do equipamento de teste.

O equipamento de teste fornece um meio de avançar os argumentos de linha de comando até um dado caso. É então responsabilidade de cada caso de teste ler os argumentos que são suportados por aquele caso de teste em particular. No entanto, os argumentos que seguem são predefinidos por esse aplicativo e devem ser sempre suportados por todos os casos de teste:

`-testid` *`id`*

utilizado para conferir o identificador do caso de teste. Se este argumento estiver presente em `args`, deve ser o valor retornado pelo `getTestId()`.

`-testtitle` *`title`*

utilizado para conferir o título do caso de teste. Se este argumento estiver presente em `args`, deve ser o valor retornado pelo `getTestTitle()`.

`-select case1,case2,case3...`

este argumento é reservado para indicar o caso de teste que apenas um (sub) caso específico deve ser manipulado. O valor é uma lista de nomes separados por vírgulas que deve corresponder a dados métodos na classe `TestCase`. *Este recurso ainda não é suportado por essa especificação e esse argumento é reservado para uso futuro.*

`-exclude case1,case2,case3...`

semelhante ao argumento `select` anterior, mas não lista os métodos que devem ser excluídos. Se ambos `select` e `exclude` forem utilizados e listarem o mesmo teste, então considera-se excluir aquele teste. *Este recurso ainda não é suportado por essa especificação e esse argumento é reservado para uso futuro.*

Essa lista está sujeita a ser estendida em uma versão futura desta especificação.

Casos de teste individuais podem adicionar suporte para novos argumentos, no entanto, esses argumentos devem ser precedidos por "`-x-`" para indicar que eles não estão padronizados. por exemplo, um dado caso de teste pode querer controlar um *timeout* através do argumento `-x-timeout`.

47.5.2 Índice de métodos

`int` **`getTestId`**`()`

Retorna o identificador do caso de teste.

`String` **`getTestTitle`**`()`

Retorna uma curta descrição do teste.

`Report` **`run`**`(String[] args)`

Executa o teste incluso pela implementação.

47.5.3 Detalhe dos métodos

run

```
Report run(String[] args)
```

Executa o teste incluso pela implementação. Se o formato de qualquer valor de qualquer dado argumento estiver incorreto, o valor deve ser ignorado.

Parâmetros:

`args` – Esses são fornecidos na descrição dessa classe e permitem que um script forneça informações de configuração para um teste, ou reutilize um teste com diferentes valores.

Retorna:

Um objeto `Report` representando o resultado do teste.

getTestId

```
int getTestId()
```

Retorna o identificador do caso de teste. Cada caso de teste tem um identificador cuja semântica, ordem, etc... são definidos dentro do contexto de equipamento de teste.

Deve ser igual ao valor *`id`* passado em `-testid` se aquele argumento foi passado para `run(String[])`.

Retorna:

Um número inteiro correspondente ao identificador do caso de teste.

getTestTitle

```
String getTestTitle()
```

Retorna uma curta descrição do teste. A descrição não deve conter novas linhas (por exemplo, retorno de reboque) e seu comprimento deve ser tipicamente menor que 80 caracteres.

Deve ser igual ao valor *`title`* passado em `-testid` se aquele argumento foi passado para `run(String[])`.

Retorna:

Uma curta descrição de texto do teste.

47.6 Classe TestHarness

47.6.1 Descrição da classe

`com.sun.dtv.test`

```
java.lang.Object
  └com.sun.dtv.test.TestHarness

public class TestHarness
extends Object
```

Fornece um ponto de entrada e um conjunto de ferramentação para casos de teste.

Exemplo de código de teste

Um teste deve também definir `main` como segue:

```
import com.sun.dtv.Report;
import com.sun.dtv.TestHarness;
import com.sun.dtv.TestCase;

public class MyTest implements TestCase {

     private int id = 1234;
     private String title = "Arithmetic Test";

     public MyTest() {
     }

     public static void main(String[] args) {
         // Iniciar o correspondente ao metodo principal TestCase

         TestHarness.start(new MyTest(), args);
     }

     public int getTestId() {
         // Retorna o ID do test (quer um codificado ou que passou através de -testid <id>)

         return id;
     }

     public String getTestTitle() {
         // Retorna o título do teste (quer um codificado ou

         // que passou atraves de -testtitle <test>)

         return title;
     }

     public Report run(String[] args) {
         // 1. avaliar   'args' para   um   parâmetro   particular   específico   para   este
               testcase e para           extrair           o           testid e/ou testtitle
               If -testid <id>, then copy value into id
               If -testtitle <title>, then copy value into title
               ...

         // 2. (dependendo       do       teste) sincronizar       com       o       servidor
         TestHarness.sendTrigger(TestHarness.TRIGGER_SYNCHRONIZE,
                                 getTestId(),
                                 "[SYNC#2881] Switch transport stream encoding NOW!");

         // 3. código    de    teste    que    faz    algo    com    essa    outra    codificação
         if (1 + 1 == 2)
            return Report.passed(this, "OK");
         else
            return Report.failed(this, "1 + 1 did not make 2");
     }
}
```

47.6.2 Índice de campos

`static PrintStream` **`log`**

O stream de saída padrão em uma dada plataforma para armazenar relatórios de teste.

`static int` **`TRIGGER_SYNCHRONIZE`**

Um `triggerType` utilizado para solicitar sincronização em um dado evento entre o cliente e o servidor de teste.

`static int` **`TRIGGER_TERMINATED`**

Um `triggerType` utilizado para sinalizar que o teste identificado pelo `testId` é considerado concluído agora.

47.6.3 Índice de construtores

TestHarness()

47.6.4 Índice de métodos

`static void` **sendReport**`(int testid, Report report)`

Esse método alias permite enviar o relatório de um dado caso de teste de volta ao servidor.

`static void` **sendTrigger**`(int triggerType, int testId, String message)`

Esse método genérico permite enviar um trigger específico para o servidor de teste.

`static void` **start**`(TestCase test, String[] args)`

Esse método é responsável por lançar um dado caso de teste, manipular o relatório de resultado e encerrar o aplicativo de teste atual.

47.6.5 Detalhe dos campos

TRIGGER_SYNCHRONIZE

`public static final int TRIGGER_SYNCHRONIZE = 2`

Um `triggerType` utilizado para solicitar sincronização em um dado evento entre o cliente e o servidor de teste.

Isso significa que o cliente atingiu um estado específico exigido para o teste e que está esperando reconhecimento para continuá-lo. Consequentemente, a implentação de sendTrigger(triggerType, testId, message) esperará até que seja recebida uma confirmação antes de retornar.

Juntamente com o `triggerType` e o `testId`, o aplicativo pode utilizar o argumento `message` para especificar qual milestone de sincronização ele atingiu. Recomenda-se compor a mensagem com uma parte que pode ser analisada (por exemplo, um código de número inteiro) e uma parte que pode ser lida (ou seja, uma curta descrição) para permitir tanto uma implementação automatizada (servidor de teste) quanto uma manual deste recurso.

Relaciona-se com:

`sendTrigger(int,int,String)`

TRIGGER_TERMINATED

`public static final int TRIGGER_TERMINATED = 3`

Um `triggerType` utilizado para sinalizar que o teste identificado pelo `testId` é considerado concluído agora.

Juntamente com o `triggerType` e a `testId` ao chamar o sendTrigger(triggerType, testId, message), o aplicativo deve fornecer o `Report` resultante no argumento `message` chamando o método `Report.toString()` correspondente.

Relaciona-se com:

`sendTrigger(int,int,String)`

log

`public static final PrintStream log`

O stream de saída padrão em uma dada plataforma para armazenar relatórios de teste. Está à esquerda da implementação, onde e como armazenar esse stream de log. As implementações podem escolher armazená-lo como um arquivo no sistema de arquivo ou encaminhar esse stream em uma interface serial ou de rede que esteja conectada ao servidor de teste.

Esse `log` deve ser configurado com `autoFlush=`*`true`*.

Relaciona-se com:

```
java.io.PrintStream(OutputStream, boolean)
```

47.6.6 Detalhe dos construtores

`TestHarness`:

```
public TestHarness()
```

47.6.7 Detalhe dos métodos

sendTrigger

```
public static void sendTrigger(int triggerType,
                               int testId,
                               String message)
                        throws UnsupportedOperationException
```

Esse método genérico permite enviar um *trigger* específico para o servidor de teste. Dependendo do `triggerType`, o método se comportará de forma diferente.

Atualmente os dois *triggers* a seguir devem ser suportados pela implementação. Esta especificação pode definir novos *triggers* em versão subsequente:

`TRIGGER_SYNCHRONIZE`

`TRIGGER_TERMINATED`

Está à esquerda da implementação como fornecer o meio de comunicação entre o cliente e o servidor de teste. Pode-se utilizar uma das seguintes tecnologias: Serial, USB, Networking, ...

No entanto, como último recurso em combinação com qualquer meio de comunicação entre cliente e servidor, a implementação deve exibir um diálogo com a mensagem correspondente ao usuário ou operador do teste na tela da plataforma.

Parâmetros:

`triggerType` – uma de qualquer constante `TRIGGER_` definida nesta classe.

`testId` – um valor de número inteiro identificando o teste. Esse valor pode ser definido pelo próprio caso de teste ou através de argumentos (consulte `TestCase`).

`message` – dependendo do `triggerType`, essa *string* pode apresentar significados diferentes. Este argumento pode ser `null`.

Lança:

`UnsupportedOperationException` – se o `triggerType` não for suportado pela plataforma.

sendReport

```
public static void sendReport(int testid,
                              Report report)
                       throws NullPointerException
```

Esse método alias permite enviar o relatório de um dado caso de teste de volta ao servidor. Se nenhum canal de retorno for definido, de acordo com a descrição do `TestHarness.TRIGGER_TERMINATED`, a *string* de relatório deve ser exibido na tela.

Esse método é equivalenete a chamar:

```
TestHarness.sendTrigger(TestHarness.TRIGGER_TERMINATED,
                        testid,
                        report.toString());
```

Parâmetros:

`testid` – um valor de número inteiro identificando o teste. Esse valor pode ser definido pelo próprio caso de teste ou através de argumentos (consulte `TestCase`).

`report` – relatório a ser enviado.

Lança:

`NullPointerException` – se o relatório for `null`.

start

```
public static void start(TestCase test,
                         String[] args)
```

Esse método é responsável por lançar um dado caso de teste, manipular o relatório de resultado e encerrar o aplicativo de teste atual.

Se a plataforma fornecer um canal de retorno, ela retornará o relatório para o servidor de teste utilizando `sendReport()`, permitindo assim que o servidor inicie o próximo teste.

A implementação deve chamar por fim o método `Report.exit()` que consequentemente faz com que o aplicativo encerre.

Uma implementação tende a se basear em:

```
if (test != null) {
   Report r = test.run(args);
   TestHarness.sendReport(test.getTestId(), r);
   r.exit();
}
```

Parâmetros:

`test` – Instância de caso de teste a ser lançada.

`args` – Lista dos argumentos passados aos casos de teste. Geralmente são aqueles da linha de comando.

48 Pacote com.sun.dtv.transport

48.1 Descrição do pacote

Provê acesso a entidades contidas em um *stream* de transporte.

A Figura 19 mostra a estrutura do pacote `Transport` do Java DTV.

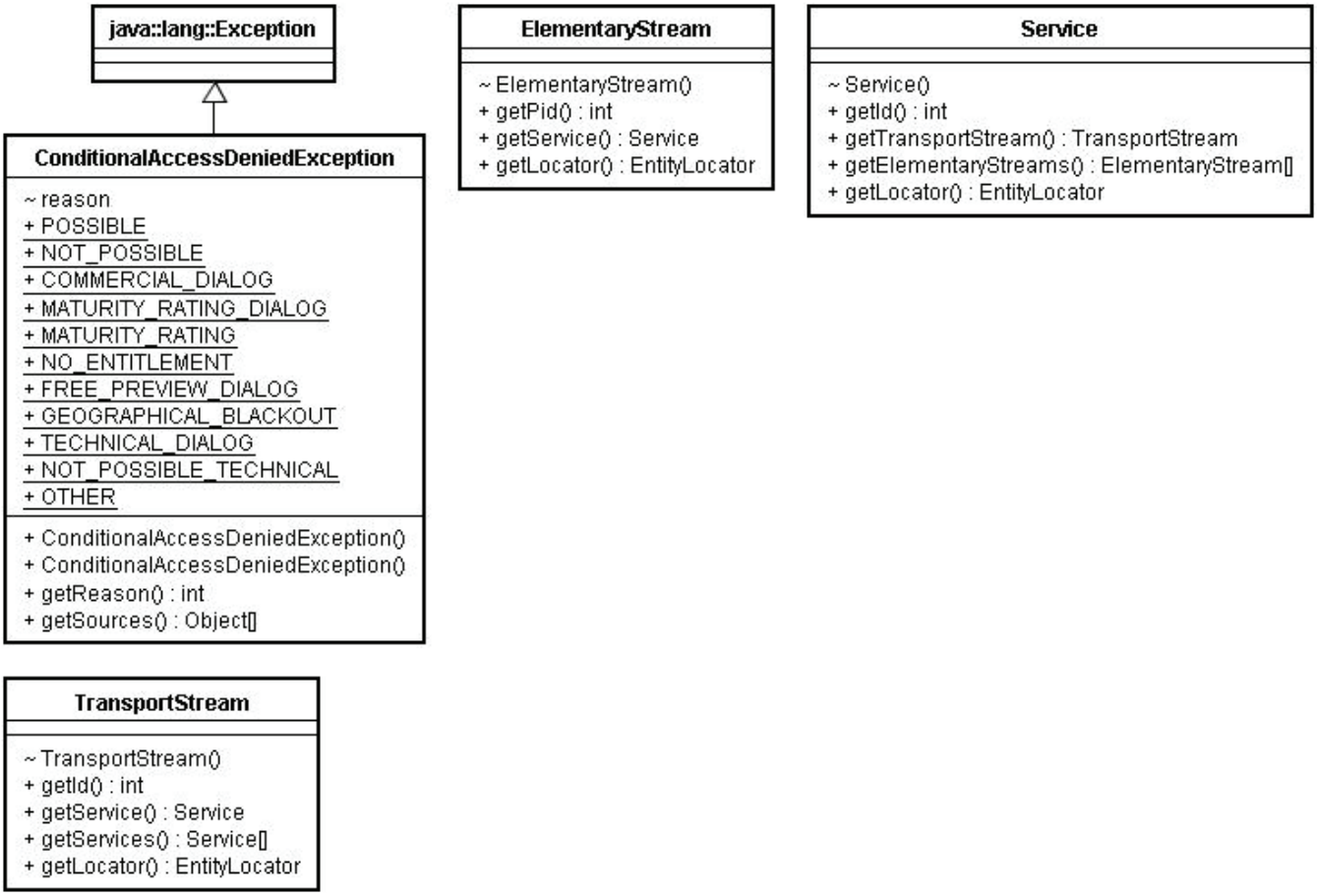


Figura 19 – Pacote `Transport`

Esse pacote fornece acesso de nível baixo às entidades associadas aos streams de transporte, serviços e streams básicos. Um stream de transporte contém uma variedade de serviços que por sua vez são compostos por streams básicos. Um stream básico pode pertencer a múltiplos serviços. Neste caso, para cada serviço há uma instância de `ElementaryStream`.

O acesso às instâncias dessas classes pode ser obtido a partir do `Tuner.getCurrentTransportStream()`.

Observar que as classes neste pacote são utilizadas para obter acesso à entidades físicas relacionadas aos streams de transporte (por exemplo, durante tuning ou o serviço ao qual o aplicativo estiver relacionado). Por outro lado, pode-se conseguir acesso às tabelas de SI e suas atualizações por meio do pacote javax.tv.service e seus subpacotes.

48.2 Índice de classes

ElementaryStream

Representação de um stream básico.

Service

Representação de um serviço presente em um stream de transporte.

TransportStream

Representação de um stream de transporte e seus serviços associados.

48.3 Índice de exceções

ConditionalAccessDeniedException

Essa classe é executada quando é solicitado acesso a informações misturadas e à qual o acesso pelo sistema de segurança não é permitido.

48.4 Classe ConditionalAccessDeniedException

48.4.1 Descrição da classe

com.sun.dtv.transport

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └com.sun.dtv.transport.ConditionalAccessDeniedException
```

Todas as interfaces implementadas

```
Serializable
```

```
public class ConditionalAccessDeniedException
extends Exception
```

Essa classe é executada quando é solicitado acesso a informações misturadas e à qual o acesso pelo sistema de segurança não é permitido.

48.4.2 Índice de campos

`static int` **COMMERCIAL_DIALOG**

necessário diálogo de usuário para pagamento.

`static int` **FREE_PREVIEW_DIALOG**

necessário diálogo de usuário para explicar previsualização grátis.

`static int` **GEOGRAPHICAL_BLACKOUT**

sem permissão por razões geográficas.

`static int` **MATURITY_RATING**

o usuário não possui idade suficiente.

`static int` **MATURITY_RATING_DIALOG**

necessário diálogo de usuário para idade.

`static int` **NO_ENTITLEMENT**

o usuário não possui direitos.

`static int` **NOT_POSSIBLE**

acesso possível mediante algumas condições.

`static int` **`NOT_POSSIBLE_TECHNICAL`**

sem permissão por razões técnicas.

`static int` **`OTHER`**

Alguma outra razão.

`static int` **`POSSIBLE`**

acesso não autorizado.

`static int` **`TECHNICAL_DIALOG`**

necessário diálogo de usuário para fins técnicos.

48.4.3 Índice de construtores

`ConditionalAccessDeniedException``()`

Constrói uma `ConditionalAccessDeniedException` sem nenhuma mensagem de detalhe.

`ConditionalAccessDeniedException``(String s)`

Constrói uma `ConditionalAccessDeniedException` sem nenhuma mensagem de detalhe especificada.

`ConditionalAccessDeniedException``(String s, int reason, Service service)`

Constrói uma `ConditionalAccessDeniedException` com a mensagem de detalhe, razões e serviços especificados.

`ConditionalAccessDeniedException``(String s, int[] reasons, ElementaryStream[] streams)`

Constrói uma `ConditionalAccessDeniedException` com a mensagem de detalhe, razões e streams especificados.

48.4.4 Índice de métodos

`int` **`getReason`**`(int index)`

Retorna a razão pela qual não foi possível desembaralhar.

`Object[]` **`getSources`**`()`

Este método retorna um *array* contendo tanto um objeto `Service` exclusivo ou um conjunto de `ElementaryStreams` que não puderam ser desembaralhados.

48.4.5 Detalhe dos campos

POSSIBLE

`public static final int POSSIBLE = 1`

acesso não autorizado.

NOT_POSSIBLE

`public static final int NOT_POSSIBLE = 2`

acesso possível mediante algumas condições.

COMMERCIAL_DIALOG

`public static final int COMMERCIAL_DIALOG = 16`

necessário diálogo de usuário para pagamento.

MATURITY_RATING_DIALOG

```
public static final int MATURITY_RATING_DIALOG = 32
```

necessário diálogo de usuário para idade.

MATURITY_RATING

```
public static final int MATURITY_RATING = 64
```

o usuário não possui idade suficiente.

NO_ENTITLEMENT

```
public static final int NO_ENTITLEMENT = 128
```

o usuário não possui direitos.

FREE_PREVIEW_DIALOG

```
public static final int FREE_PREVIEW_DIALOG = 256
```

necessário diálogo de usuário para explicar previsualização grátis.

GEOGRAPHICAL_BLACKOUT

```
public static final int GEOGRAPHICAL_BLACKOUT = 512
```

sem permissão por razões geográficas.

TECHNICAL_DIALOG

```
public static final int TECHNICAL_DIALOG = 1024
```

necessário diálogo de usuário para fins técnicos.

NOT_POSSIBLE_TECHNICAL

```
public static final int NOT_POSSIBLE_TECHNICAL = 2048
```

sem permissão por razões técnicas.

OTHER

```
public static final int OTHER = 4096
```

Alguma outra razão.

48.4.6 Detalhe dos construtores

ConditionalAccessDeniedException

```
public ConditionalAccessDeniedException()
```

Constrói uma `ConditionalAccessDeniedException` sem nenhuma mensagem de detalhe.

ConditionalAccessDeniedException

```
public ConditionalAccessDeniedException(String s)
```

Constrói uma `ConditionalAccessDeniedException` sem nenhuma mensagem de detalhe especificada.

Parâmetros:

`s` - a mensagem detalhada.

ConditionalAccessDeniedException

```
public ConditionalAccessDeniedException(String s,
                                        int reason,
                                        Service service)
```

Constrói uma `ConditionalAccessDeniedException` com a mensagem de detalhe, razões e serviços especificados.

Parâmetros:

`s` - a mensagem detalhada.

`reason` - razão.

`service` - serviço.

ConditionalAccessDeniedException

```
public ConditionalAccessDeniedException(String s,
                                        int[] reasons,
                                        ElementaryStream[] streams)
```

Constrói uma `ConditionalAccessDeniedException` com a mensagem de detalhe, razões e *streams* especificados.

Parâmetros:

`s` - a mensagem detalhada.

`reasons` - razões.

`streams` - *streams*.

48.4.7 Detalhe dos métodos

getReason

```
public int getReason(int index)
             throws IndexOutOfBoundsException
```

Retorna a razão pela qual não foi possível desembaralhar.

Parâmetros:

`index` – deve referir-se ao Serviço ou a um stream básico no conjunto retornado por getSources().

Retorna:

razão para exceção não autorizada. A razão é uma lógica binária `OR` de `POSSIBLE` ou `NOT_POSSIBLE` com outro código de razão.

Lança:

`IndexOutOfBoundsException` – Essa exceção deve ser executada onde o índice estiver além do tamaho do *array* retornado por getSources.

Relaciona-se com:

`getSources()`

getSources

`public Object[] getSources()`

Este método retorna um *array* contendo tanto um objeto `Service` exclusivo ou um conjunto de `ElementaryStreams` que não puderam ser desembaralhados.

Retorna:

o conjunto dos objetos `Service` ou `ElementaryStream` que não puderam ser desembaralhados.

48.5 Classe ElementaryStream

48.5.1 Descrição da classe

`com.sun.dtv.transport`

`java.lang.Object`

`└com.sun.dtv.transport.ElementaryStream`

`public class ElementaryStream`

`extends Object`

Representação de um stream básico.

48.5.2 Índice de métodos

`EntityLocator` **`getLocator`**`()`

obtém o localizador desta entidade.

`int` **`getPid`**`()`

obtém o PID deste stream básico.

`Service` **`getService`**`()`

obtém o serviço ao qual esse stream básico pertence.

48.5.3 Detalhe dos métodos

getPid

`public int getPid()`

obtém o PID deste stream básico.

Retorna:

PID

getService

`public Service getService()`

obtém o serviço ao qual esse stream básico pertence.

Retorna:

serviço

getLocator

`public EntityLocator getLocator()`

obtém o localizador desta entidade.

Retorna:

localizador

48.6 Classe Service

48.6.1 Descrição da classe

`com.sun.dtv.transport`

```
java.lang.Object
  └com.sun.dtv.transport.Service
```

`public class Service`

`extends Object`

Representação de um serviço presente em um stream de transporte. Um serviço pode conter uma variedade de streams básicos.

48.6.2 Índice de métodos

`ElementaryStream[] **getElementaryStreams**()`

obtém todos os streams básicos deste serviço.

`int **getId**()`

obtém o identificador de serviço deste serviço.

`EntityLocator **getLocator**()`

obtém o localizador desta entidade.

`TransportStream **getTransportStream**()`

obtém o serviço ao qual esse stream de transporte pertence.

48.6.3 Detalhe dos métodos

getId

`public int getId()`

obtém o identificador de serviço deste serviço.

Retorna:

identificador de serviço

getTransportStream

`public TransportStream getTransportStream()`

obtém o serviço ao qual esse stream de transporte pertence.

Retorna:

stream de transporte

getElementaryStreams

```
public ElementaryStream[] getElementaryStreams()
```

obtém todos os streams básicos deste serviço.

Retorna:

array de streams básicos

getLocator

```
public EntityLocator getLocator()
```

obtém o localizador desta entidade.

Retorna:

localizador

48.7 Classe TransportStream

48.7.1 Descrição da classe

com.sun.dtv.transport

```
java.lang.Object
  └com.sun.dtv.transport.TransportStream
```

```
public class TransportStream
extends Object
```

Representação de um stream de transporte e seus serviços associados.

48.7.2 Índice de métodos

`int` **`getId`**`()`

obtém identificador do stream de transporte.

`EntityLocator` **`getLocator`**`()`

obtém o localizador desta entidade.

`Service` **`getService`**`(int id)`

obtém o objeto de serviço para um identificador de serviço específico.

`Service[]` **`getServices`**`()`

obtém todos os serviços presentes nesse stream de transporte.

48.7.3 Detalhe dos métodos

getId

```
public int getId()
```

obtém identificador do stream de transporte.

Retorna:

identificador do stream de transporte

getService

`public Service getService(int id)`

obtém o objeto de serviço para um identificador de serviço específico.

Parâmetros:

`id` – identificador para o objeto `Service` solicitado

Retorna:

objeto `Service` para o identificador de serviço especificado. Retorna `null` se não existir tal serviço.

getServices

`public Service[] getServices()`

obtém todos os serviços presentes nesse stream de transporte.

Retorna:

array de objetos `Service` presentes

getLocator

`public EntityLocator getLocator()`

obtém o localizador desta entidade.

Retorna:

localizador

49 Pacote com.sun.dtv.tuner

49.1 Descrição do pacote

Provê API para acesso a e controle de uma interface de redes de transmissão (ou “sintonizador”) usado para recepção de *streams* de transporte.

A Figura 20 mostra a estrutura do pacote `Tuner` do Java DTV.

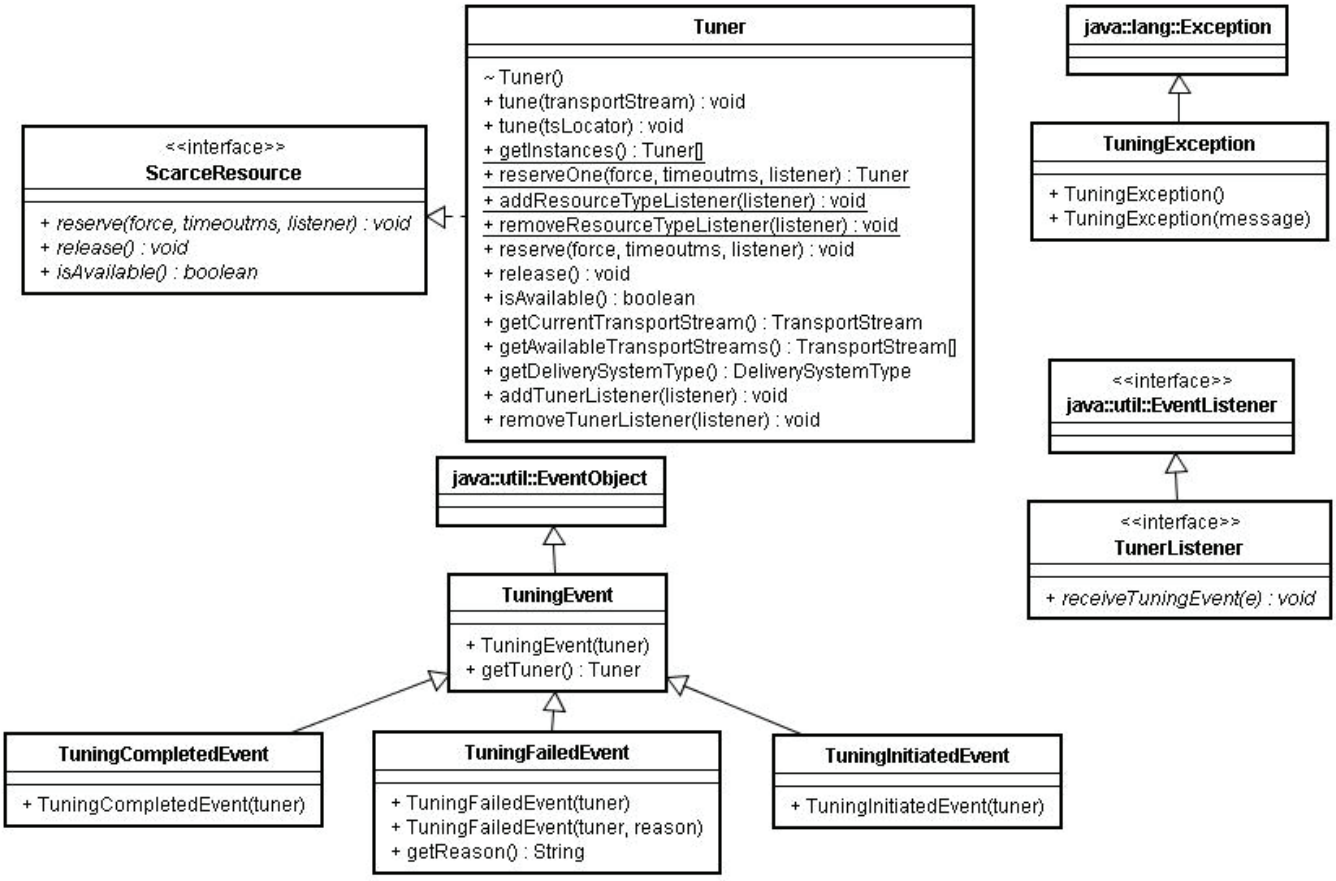


Figura 20 – Pacote `Tuner`

NOTA Este pacote está especificado desde o Java DTV 1.0.

49.2 Índice de interfaces

TunerListener

Interface de *listener* para receber eventos de tuning de um `Tuner`.

49.3 Índice de classes

Tuner

Classe representando uma interface de rede ou "`tuner`" para receber streams de transporte de transmissão.

TuningCompletedEvent

Evento que indica a conclusão bem-sucedida de uma operação de tuning.

TuningEvent

Classe de base para eventos gerados pelas oprações de tuning.

TuningFailedEvent

Evento que indica falha na conclusão de uma operação de tuning.

TuningInitiatedEvent

Evento que indica início de uma operação de tuning.

49.4 Índice de exceções

TuningException

Exceção para geração de relatório sincrônica de falhas de tuning.

49.5 Classe Tuner

49.5.1 Descrição da classe

com.sun.dtv.tuner

```
java.lang.Object
  └com.sun.dtv.tuner.Tuner
```

Todas as interfaces implementadas

```
ScarceResource
```

```
public class Tuner
extends Object
implements ScarceResource
```

Classe representando uma interface de rede ou "`tuner`" para receber streams de transporte de transmissão.

Um `tuner` físico é um recurso de sistema escasso, portanto deve ser reservado através de API de `ScarceResource` antes do uso. O acesso ao método `reserve(boolean, long, ScarceResourceListener)` é guardado por `ScarceResourcePermission("tuner.any", "reserve")`.

Relaciona-se com:

`ScarceResourcePermission`

49.5.2 Índice de métodos

`static void` **`addResourceTypeListener`**`(ResourceTypeListener listener)`

Adiciona um `ResourceTypeListener` a implementação.

`void` **`addTunerListener`**`(TunerListener listener)`

Inscreve um `TunerListener` para receber eventos de sintonização a partir deste `Tuner`.

`TransportStream[]` **`getAvailableTransportStreams`**`()`

Fornece os streams de transporte que podem ser acesados através deste `Tuner`.

`TransportStream` **`getCurrentTransportStream`**`()`

Reporta o stream de transporte ao qual o `Tuner` está atualmente associado.

`DeliverySystemType` **`getDeliverySystemType`**`()`

Fornece o tipo de sistema de entrega do `Tuner`.

`static Tuner[]` **`getInstances`**`()`

Retorna um *array* de handlers representando cada um deles um recurso `Tuner` separado, disponível na plataforma correspondente às interfaces de rede que podem ser utilizadas para sintonização.

`boolean` **`isAvailable`**`()`

Verifica se o dado recurso está correntemente disponível para reserva.

`void` **`release`**`()`

Libera esse recurso.

`static void` **`removeResourceTypeListener`**`(ResourceTypeListener listener)`

Remove um *listener* previamente anexado.

`void` **`removeTunerListener`**`(TunerListener listener)`

Evita que um `TunerListener` receba eventos de sintonização a partir deste `Tuner`.

`void` **`reserve`**`(boolean force, long timeoutms, ScarceResourceListener listener)`

Requisita reserva da dada instância de recursos escassos.

`static Tuner` **`reserveOne`**`(boolean force, long timeoutms, ScarceResourceListener listener)`

Retorna uma instância reservada fora do pool de todas as intâncias físicas do `Tuner`.

`void` **`tune`**`(TransportStream transportStream)`

Sintoniza a interface de rede para o stream de transporte especificado.

`void` **`tune`**`(Locator tsLocator)`

Sintoniza a interface de rede para o stream de transporte referenciado pelo localizador especificado.

49.5.3 Detalhe dos métodos

tune

```
public void tune(TransportStream transportStream)
          throws TuningException
```

Sintoniza a interface de rede para o stream de transporte especificado. Este método completa de forma assíncrona. Se a operação de sintonização for iniciada com sucesso, o `TuningInitiatedEvent` é enviado aos *listeners* deste `Tuner`. Com a conclusão bem-sucedida da operação de sintonização, o `TuningCompletedEvent` é enviado aos *listeners*, e o `transportStream` deve ser o stream de transporte atual do `Tuner`. Se a operação de sintonização falhar, `TuningFailedEvent` é enviado aos *listeners*, e o stream de transporte atual do `Tuner` é indefinido. Se a operação de tuning solicitada não puder ser iniciada, é executada `TuningException`, nenhum evento é postado, e o stream de transporte atual do `Tuner` permanece intacto.

Parâmetros:

`transportStream` – stream de transporte ao qual sintonizar

Lança:

`TuningException` – se a sintonização não puder ser iniciada

`IllegalStateException` – se o chamador não reservou este `Tuner`

`NullPointerException` – se `transportStream` é `null`

Relaciona-se com:

`TunerListener`

tune

```
public void tune(Locator tsLocator)
          throws TuningException,
                 InvalidLocatorException
```

Sintoniza a interface de rede para o *stream* de transporte referenciado pelo localizador especificado. Este método completa de forma assíncrona. Se a operação de sintonização for iniciada com sucesso, o `TuningInitiatedEvent` é enviado aos *listeners* deste `Tuner`. Com a conclusão bem-sucedida da operação de sintonização, o `TuningCompletedEvent` é enviado aos *listeners*, e o `transportStream` deve ser o *stream* de transporte atual do `Tuner`. Se a operação de sintonização falhar, `TuningFailedEvent` é enviado aos *listeners*, e o *stream* de transporte atual do `Tuner` é indefinido. Se a operação de sintonização solicitada não puder ser iniciada, é executada `TuningException`, nenhum evento é postado, e o *stream* de transporte atual do `Tuner` permanece intacto.

Parâmetros:

`tsLocator` – um localizador fazendo referência ao stream de transporte ao qual sintonizar

Lança:

`TuningException` – se a sintonização não puder ser iniciada

`InvalidLocatorException` – se o localizador especificado não faz referência ao *stream* de transporte.

`IllegalStateException` – se o chamador não reservou este `Tuner`

`NullPointerException` – se `transportStream` é `null`

Relaciona-se com:

`TunerListener`

getInstances

```
public static Tuner[] getInstances()
```

Retorna um *array* de handlers representando cada um deles um recurso `Tuner` separado, disponível na plataforma correspondente às interfaces de rede que podem ser utilizadas para sintonização. Cada objeto handler é único tanto para aplicativo como para plataforma. Cada *handler* de objeto pode ser diferente para cada um deles em chamadas subsequentes para o método `getInstances()`. A lista contém todas as instâncias, estejam ou não reservadas.

Se nenhuma interface de rede estiver presente, esse método retorna um *array* de comprimento zero.

Retorna:

um *array* de instâncias de recursos do tipo `Tuner`.

reserveOne

```
public static Tuner reserveOne(boolean force,
                               long timeoutms,
                               ScarceResourceListener listener)
                        throws SecurityException,
                               IllegalArgumentException,
                               TimeoutException
```

Retorna uma instância reservada fora do pool de todas as intâncias físicas do `Tuner`. Esse método retorna com a instância ou executa uma exceção de acordo com a situação.

Esse método se comporta exatamente como o método `reserve()`.

Parâmetros:

`force` - Se *`true`*, esse método tem permissão de retirar qualquer recurso do atual proprietário. Se *`false`*, a implementação bloqueará e esperará até que um recurso de um dado `type` esteja disponível (utilizando `release()`) ou até milissegundos `timeoutms`.

`timeoutms` – Uma quantidade positiva de tempo em milissegundos até a qual este método esperará para que qualquer recurso seja liberado por seu proprietário atual. O valor `-1` indica que a implementação esperará para sempre.

`listener` - O *listener* a ser anexado para receber notificação sobre o status do recurso.

Retorna:

A instância de tipo `Tuner` que foi reservada.

Lança:

`SecurityException` - Se o aplicativo não tiver permissão para ação de reserva para o recurso que está prestes a reservar. Também lançado se forçar for definido *true*, mas o aplicativo não tiver permissão para a ação forçar.

`IllegalArgumentException` - Se o *listener* estiver `null` ou se o valor especificado em `timeoutms` não for válido, isto é, nem um inteiro positivo, nem -1.

`TimeoutException` - Se o tempo especificado em `timeoutms` houver acabado e o recurso não houver podido ser reservado.

Relaciona-se com:

`reserve()`

addResourceTypeListener

```
public static void addResourceTypeListener(ResourceTypeListener listener)
                                    throws NullPointerException
```

Adiciona um `ResourceTypeListener` a implementação. Sempre que um `reserve()` ou um `release()` for chamado para quaisquer recursos do mesmo tipo, a implementação chamará os métodos correspondentes do *listener*.

Parâmetros:

`listener` - O *listener* desencadeado para eventos em recursos do mesmo tipo.

Lança:

`NullPointerException` - Se o *listener* estiver `null`.

Relaciona-se com:

`removeResourceTypeListener()`

removeResourceTypeListener

```
public static void removeResourceTypeListener(ResourceTypeListener listener)
                                        throws NullPointerException
```

Remove um *listener* previamente anexado. Se o *listener* não foi anexado antes, esse método não faz nada..

Parâmetros:

`listener` - O *listener* desencadeado para eventos em recursos do mesmo tipo.

Lança:

`NullPointerException` - Se o *listener* estiver `null`.

Relaciona-se com:

`addResourceTypeListener()`

reserve

```
public void reserve(boolean force,
                    long timeoutms,
                    ScarceResourceListener listener)
             throws IllegalArgumentException,
                    TimeoutException,
                    SecurityException
```

Requisita reserva da dada instância de recursos escassos. O método bloquear-se-á até que a instância esteja disponível. O método retorna normalmente apenas se a reserva for bem sucedida. Todos os outros casos são manuseados por exceções.

Primeiro, se houver um gerenciador de segurança, seu método `checkPermission` é chamado com a permissão `ScarceResourcePermission(name, "reserve")`. Se `force` é além disso definido *`true`*, a permissão também verificado no `ScarceResourcePermission(name, "force")`.

Durante o processo de reserva, se a instância do recurso já estiver alocada, a implementação deve notificar o proprietário atual do recurso sobre a requisição de reserva por meio da interface `ScarceResourceListener`:

ou por `ScarceResourceListener.releaseRequested()` se forçar for *false*,

ou por `ScarceResourceListener.releaseForced()` no outro caso.

O *listener* deve ser usado para tais notificações apenas até o recurso ser liberado. Após a liberação, a implementação não deve chamar quaisquer dos métodos da interface do *listener*.

Após esse primeiro evento, a implementação procederá de acordo e liberará (ou não) o recurso requisitado. No caso da implementação liberar o recurso, desencadeará um evento `ScarceResourceListener.released()` ao proprietário *listener* original do recurso informando-o que o recurso não lhe pertence mais.

O aplicativo pode controlar o tempo de espera para que tal recurso esteja disponível ao definir `timeoutms`. No caso da duração da reserva exceder o tempo expresso em `timeoutms`, um `TimeoutException` é lançado.

Sob operação normal, recursos são liberados usando o método `release`. De qualquer forma, no caso do aplicativo não liberar recursos quando requisitado ou o aplicativo ser terminado, a implementação deve liberar todos os recursos alocados para o aplicativo para permitir que outros aplicativos sejam notificados de mudanças na alocação de recursos e poderem reservá-los. Ver a seção Resource Cleanup do ciclo de vida do aplicativo.

Especificado por:

`reserve` na interface `ScarceResource`

Parâmetros:

`force` - Se *`true`*, esse método retirará o recurso do proprietário atual. Se *`false`*, a implementação bloquear-se-á e esperará até que o recurso esteja disponível (usando `release()`) ou até `timeoutms`.

`timeoutms` - Uma quantidade positiva de milissegundos que esse método esperará para que o recurso seja liberado pelo proprietário atual. O valor `-1` indica que a implementação esperará para sempre.

`listener` - O *listener* a ser anexado para receber notificação sobre o status do recurso.

Lança:

`IllegalArgumentException` - Se o *listener* estiver `null` ou se o valor especificado em `timeoutms` não for válido, isto é, nem um inteiro positivo, nem -1.

`TimeoutException` - Se o tempo especificado em `timeoutms` houver acabado e o recurso não houver podido ser reservado.

`SecurityException` – Se o chamador não possuir `ScarceResourcePermission("tuner", "reserve")`, ou se a força é verdadeira e o chamador não possui `ScarceResourcePermission("tuner", "force")`.

release

`public void release()`

Libera esse recurso.

O recurso deve ser disponibilizado para outro aplicativo através da plataforma. Após chamar esse método, não é mais possível interagir com o dado recurso: chamadas para métodos críticos desses recursos escassos devem ser ignorados e lançar `IllegalStateException`. Essa afirmação é válida e é o comportamento requerido de qualquer classe implementando a interface `ScarceResource`. Para interagir novamente com o dado recurso, o aplicativo deve chamar o método `reserve()` e se tornar o proprietário novamente.

A implementação pode organizar quaisquer recursos do sistema que esse objeto estiver usando. Depois a implementação não deve chamar quaisquer dos métodos do *listener* que estava anexado no momento da reserva.

Se o recurso já estava disponível (isto é, não reservado), esse método não faz nada.

Especificado por:

`release` na interface `ScarceResource`

isAvailable

`public boolean isAvailable()`

Verifica se o dado recurso está correntemente disponível para reserva. O valor retornado dá a situação atual e não garante que o recurso ainda estará disponível em um momento posterior, por exemplo, no momento da reserva: outro aplicativo pode ter tomado aquele recurso no tempo decorrido.

Especificado por:

`isAvailable` na interface `ScarceResource`

Retorna:

Um boolean definido *`true`* se o dado recurso estiver atualmente disponível para reserva.

getCurrentTransportStream

`public TransportStream getCurrentTransportStream()`

Reporta o stream de transporte ao qual o `Tuner` está atualmente associado. Se o `Tuner` não estiver atualmente sintonizado a um stream de trasnporte, este método retorna `null`.

Retorna:

O stream de transporte ao qual o `Tuner` está atualmente sintonizado

getAvailableTransportStreams

`public TransportStream[] getAvailableTransportStreams()`

Fornece os streams de transporte que podem ser acesados através deste `Tuner`. Se nenhum stream de transporte existir, é retornado um *array* de comprimento zero.

Retorna:

array de streams de transporte acessível através deste `Tuner`

getDeliverySystemType

`public DeliverySystemType getDeliverySystemType()`

Fornece o tipo de sistema de entrega do `Tuner`.

Retorna:

tipo de sistema de entrega

addTunerListener

`public void addTunerListener(TunerListener listener)`

Inscreve um `TunerListener` para receber eventos de sintonização a partir deste `Tuner`.

Parâmetros:

`listener` - `TunerListener` para inscrever

Lança:

`NullPointerException` – se o *listener* for `null`

Relaciona-se com:

`removeTunerListener()`

removeTunerListener

`public void removeTunerListener(TunerListener listener)`

Evita que um `TunerListener` receba eventos de sintonização a partir deste `Tuner`. Se o *listener* especificado não estiver atualmente inscrito, este método não tem função.

Parâmetros:

`listener` - `TunerListener` para retirar inscrição

Lança:

`NullPointerException` – se o *listener* for `null`

Relaciona-se com:

`addTunerListener()`

49.6 Interface TunerListener

49.6.1 Descrição da interface

`com.sun.dtv.tuner`

Todas as super-interfaces

`EventListener`

`public interface TunerListener`

`extends EventListener`

Interface `Listener` para receber eventos de sintonização de um `Tuner`.

49.6.2 Índice de métodos

`void` **`receiveTuningEvent`**`(TuningEvent e)`

Notifica o *listener* sobre um evento de sintonização.

49.6.3 Detalhe dos métodos

receiveTuningEvent

`void receiveTuningEvent(TuningEvent e)`

Notifica o *listener* sobre um evento de sintonização.

Parâmetros:

`e` – evento de sintonização enviado ao *listener*.

49.7 Classe TuningCompletedEvent

49.7.1 Descrição da classe

`com.sun.dtv.tuner`

```
java.lang.Object
  └java.util.EventObject
      └com.sun.dtv.tuner.TuningEvent
          └com.sun.dtv.tuner.TuningCompletedEvent
```

Todas as interfaces implementadas

`Serializable`

```
public class TuningCompletedEvent
extends TuningEvent
```

Evento que indica a conclusão bem-sucedida de uma operação de tuning.

49.7.2 Índice de construtores

`TuningCompletedEvent``(Tuner tuner)`

Constrói um novo `TuningCompletedEvent`.

Métodos herdados da classe `com.sun.dtv.tuner.`**`TuningEvent`**

`getTuner`

49.7.3 Detalhe dos construtores

TuningCompletedEvent

`public TuningCompletedEvent(Tuner tuner)`

Constrói um novo `TuningCompletedEvent`.

Parâmetros:

`tuner` – origem do evento.

49.8 Classe TuningEvent

49.8.1 Descrição da classe

`com.sun.dtv.tuner`

```
java.lang.Object
  └java.util.EventObject
      └com.sun.dtv.tuner.TuningEvent
```

Todas as interfaces implementadas

`Serializable`

Subclasses diretas conhecidas

`TuningCompletedEvent, TuningFailedEvent, TuningInitiatedEvent`

```
public class TuningEvent
extends EventObject
```

Classe de base para eventos gerados pelas oprações de tuning.

49.8.2 Índice de construtores

`TuningEvent``(Tuner tuner)`

Constrói um novo `TuningEvent`.

49.8.3 Índice de métodos

`Tuner` **`getTuner`**`()`

Reporta ao `Tuner` que gerou o evento.

49.8.4 Detalhe dos construtores

TuningEvent

```
public TuningEvent(Tuner tuner)
```

Constrói um novo `TuningEvent`.

Parâmetros:

`tuner` – origem do evento.

49.8.5 Detalhe dos métodos

getTuner

```
public Tuner getTuner()
```

Reporta ao `Tuner` que gerou o evento.

Retorna:

`Tuner` que gerou o evento.

49.9 Classe TuningException

49.9.1 Descrição da classe

com.sun.dtv.tuner

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └com.sun.dtv.tuner.TuningException
```

Todas as interfaces implementadas

`Serializable`

```
public class TuningException
extends Exception
```

Exceção para geração de relatório sincrônica de falhas de tuning.

49.9.2 Índice de construtores

`TuningException``()`

Constrói uma nova `TuningException` com uma mensagem de detalhe `null`.

`TuningException``(String message)`

Constrói uma nova `TuningException` com a mensagem de detalhe especificada.

49.9.3 Detalhe dos construtores

TuningException

```
public TuningException()
```

Constrói uma nova `TuningException` com uma mensagem de detalhe `null`.

49.10 TuningException

```
public TuningException(String message)
```

Constrói uma nova `TuningException` com a mensagem de detalhe especificada.

Parâmetros:

`message` – uma mensagem explicando a causa da exceção.

49.11 Classe TuningFailedEvent

49.11.1 Descrição da classe

com.sun.dtv.tuner

```
java.lang.Object
```

```
└java.util.EventObject
    └com.sun.dtv.tuner.TuningEvent
        └com.sun.dtv.tuner.TuningFailedEvent
```

Todas as interfaces implementadas

`Serializable`

```
public class TuningFailedEvent
extends TuningEvent
```

Evento que indica falha na conclusão de uma operação de sintonização.

49.11.2 Índice de construtores

`TuningFailedEvent``(Tuner tuner)`

Constrói um `TuningFailedEvent`.

`TuningFailedEvent``(Tuner tuner, String reason)`

Constrói um `TuningFailedEvent`.

49.11.3 Índice de métodos

`String` **`getReason`**`()`

Reporta a razão para a falha da operação de sintonização.

Métodos herdados da classe `com.sun.dtv.tuner.`**`TuningEvent`**

`getTuner`

49.11.4 Detalhe dos construtores

TuningFailedEvent

```
public TuningFailedEvent(Tuner tuner)
```

Constrói um `TuningFailedEvent`.

Parâmetros:

`tuner` – origem do evento.

TuningFailedEvent

```
public TuningFailedEvent(Tuner tuner,
                         String reason)
```

Constrói um `TuningFailedEvent`.

Parâmetros:

`tuner` – origem do evento.

`reason` – uma descrição da causa da falha.

49.11.5 Detalhe dos métodos

getReason

`public String getReason()`

Reporta a razão para a falha da operação de sintonização.

Retorna:

cauda da falha.

49.12 Classe TuningInitiatedEvent

49.12.1 Descrição da classe

`com.sun.dtv.tuner`

```
java.lang.Object
  └java.util.EventObject
      └com.sun.dtv.tuner.TuningEvent
          └com.sun.dtv.tuner.TuningInitiatedEvent
```

Todas as interfaces implementadas

`Serializable`

`public class TuningInitiatedEvent`

`extends TuningEvent`

Evento que indica início de uma operação de sintonização.

49.12.2 Índice de construtores

`TuningInitiatedEvent``(Tuner tuner)`

Constrói o `TuningInitiatedEvent`.

Métodos herdados da classe `com.sun.dtv.tuner.`**`TuningEvent`**

`getTuner`

49.12.3 Detalhe dos construtores

TuningInitiatedEvent

`public TuningInitiatedEvent(Tuner tuner)`

Constrói o `TuningInitiatedEvent`.

Parâmetros:

`tuner` – origem do evento.

50 Pacote com.sun.dtv.ui

50.1 Descrição do pacote

Funcionalidades de UI específicas para TV.

A Figura 21 mostra a estrutura do pacote `UI` do Java DTV.

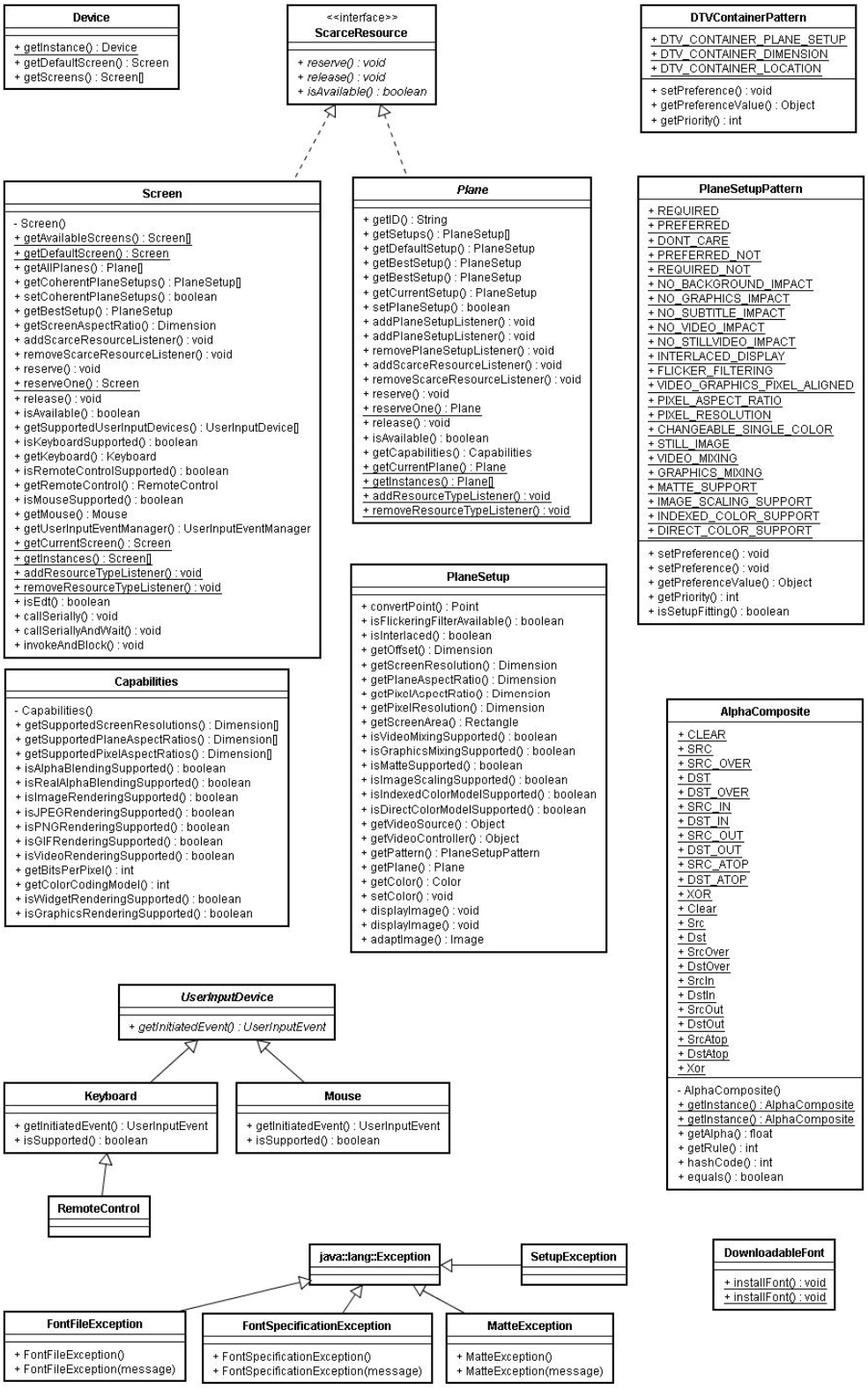


Figura 21 – Pacote `User Interface`

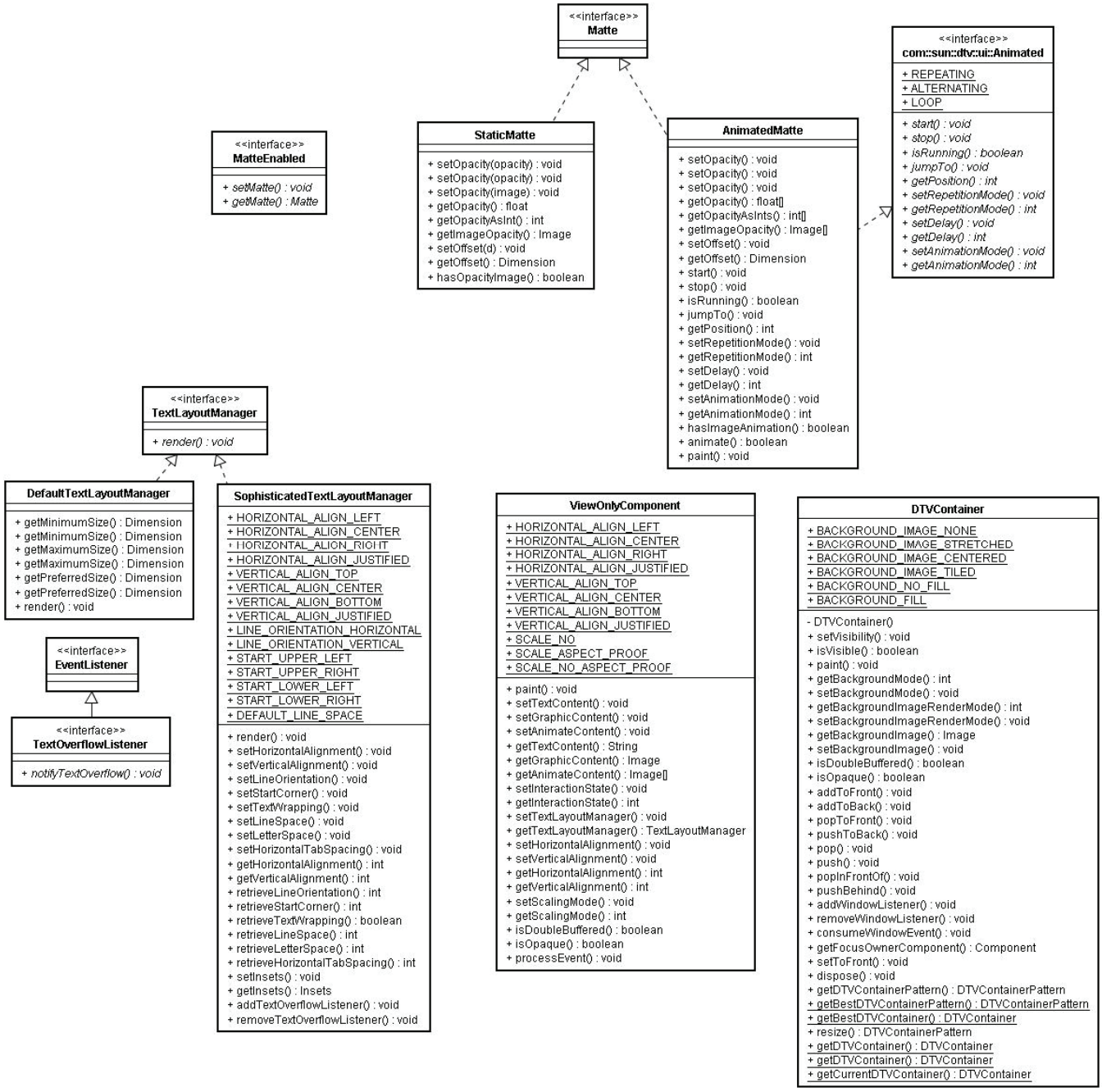


Figura 21 (continuação)

NOTA Este pacote está especificado desde o Java DTV 1.0.

50.2 Índice de interfaces

Animated

Esta interface fornece métodos para configurar e recuperar parâmetros de um efeito de animação.

Matte

Interface básica para todas as classes Matte.

MatteEnabled

O objetivo desta interface é habilitar componentes para combinação de Matte.

TextLayoutManager

O objetivo desta interface é definir funcionalidade para o layout de *string*s e sua renderização na tela.

TextOverflowListener

O objetivo de um `TextOverflowListener` é ser notificado quando um *string* de texto não se encaixa em um componente durante uma tentativa de rederizá-lo.

ViewOnlyComponent

Esta interface representa qualquer tipo de componente não-interativo no sistema e os fornece um `LookAndFeel` plugável.

50.3 Índice de classes

AlphaComposite

Essa classe `AlphaComposite` implementa todas as regras de composição para combinar pixels de fonte e destino para alcançar efeitos de mesclagem e transparência em gráficos e imagens.

AnimatedMatte

Essa classe representa um matte animado com uma máscara de imagem dinâmica, em que os valores de pixel determinam a transparência do matte para todos os pixels para um certo momento.

Capabilities

Descreve as capacidades de um plano.

DefaultTextLayoutManager

Essa classe fornece um mecanismo-padrão para renderização de textos fornecidos e é ao mesmo tempo a implementação mais simples da interface `TextLayoutManager`.

Device

Essa classe é uma representação de um aparelho de TV.

DownloadableFont

Essa classe introduz a possibilidade de baixar fontes.

DTVContainer

Um `DTVContainer` é o container de nível mais alto na hierarquia de componente da API Java DTV, representando o *widget* contendo qualquer outra coisa visível em um plano em particular.

DTVContainerPattern

O `DTVContainerPattern` é um meio de descrever as características de um `DTVContainer` válido.

Keyboard

Essa classe representa um teclado que pode ser utilizado para controlar uma `Screen` em particular como um `UserInputDevice`.

Mouse

Essa classe representa um mouse que pode ser utilizado para controlar uma `Screen` em particular como um `UserInputDevice`.

Plane

Uma instância de classe `Screen` representa um sinal de saída de vídeo de um aparelho de TV.

PlaneSetup

A classe `PlaneSetup` é capaz de descrever as características de um plano.

PlaneSetupPattern

Essa classe é um meio de descrever a configuração de um plano de tela especificando várias propriedades e sua importância para o aplicativo.

RemoteControl

Essa classe representa um controle remoto que pode ser utilizado para controlar uma `Screen` em particular como um `UserInputDevice`.

Screen

Essa classe é uma representação de uma tela de TV.

SophisticatedTextLayoutManager

Essa classe fornece um `TextLayoutManager` com mais recursos que o `DefaultTextLayoutManager` para habilitar possibilidades de design de layout mais sofisticadas para o teste ser mostrado em componentes de TV.

StaticMatte

Essa classe representa um *matte* não-animado.

UserInputDevice

Essa classe abstrata é a base para todos os dispositivos de entrada que podem ser utilizados para controlar uma tela.

50.4 Índice de exceções

FontFileException

Essa exceção deve ser executará se for realizada uma tentaiva de ler um arquivo de fonte, e esse arquivo não possuir o formato apropriado.

FontSpecificationException

Essa exceção deve ser executada se for realizada uma tentativa de especificar características de uma fonte a partir de um arquivo de fonte, e essas características não estiverem disponíveis para essa fonte em particular.

MatteException

Uma `MatteException` é executada sempre que um aplicativo não conseguir, por alguma razão, realizar uma associação de matte a um elemento gráfico.

SetupException

Exceção a ser executada em diversas situações onde tenta-se realizar uma alteração de configuração ilegal de um ou mais planos de uma `Screen`.

50.5 Classe AlphaComposite

50.5.1 Descrição da classe

`com.sun.dtv.ui`

```
java.lang.Object
  └com.sun.dtv.ui.AlphaComposite

final public class AlphaComposite
extends Object
```

Essa classe `AlphaComposite` implementa todas as regras de composição para combinar pixels de fonte e destino para alcançar efeitos de mesclagem e transparência em gráficos e imagens. As regras implementadas por essa classe são o conjunto de regras Porter-Duff descritas em [T. Porter & T. Duff].

Essa classe `AlphaComposite` é semelhante à classe `java.awt.AlphaComposite` mas difere pelo fato de que todas as regras alfa de composição são suportadas aqui enquanto a variante PBP de `java.awt.AlphaComposite` suporta apenas as três primeiras regras descritas abaixo.

As regras a seguir são suportadas:

- `Clear`
- `A (Src)`
- `A sobre B (SrcOver)`
- `B (Dst)`
- `B sobre A (DstOver)`
- `A em B (SrcIn)`
- `B em A (DstIn)`
- `A suportado por B (SrcOut)`
- `B suportado por A (DstOut)`
- `A em cima de B (SrcAtop)`
- `B em cima de A (DstAtop)`
- `A xor B (Xor)`

Codificação de regra

Se qualquer entrada não possuir um canal alfa, um valor alfa de 1,0f, que é completamente opaco, é atribuído para todos os pixels. Um valor alfa de constante pode também ser especificado para ser multiplicado com o valor alfa dos pixels da origem.

As abreviações a seguir são utilizadas na descrição das regras:

- `Cs` = um dos componentes de cor do pixel de origem.
- `Cd` = um dos componentes de cor do pixel de destino.
- `As` = componente alfa do pixel de origem.
- `Ad` = componente alfa do pixel de destino.
- `Fs` = fração do pixel de origem que contribui para a saída.
- `Fd` = fração do pixel de destino de entrada que contribui para a saída.

A cor e os componentes alfa produzidos por uma opração de composição são calculados como segue:

```
Cd = Cs*Fs + Cd*Fd
Ad = As*Fs + Ad*Fd
```

onde `Fs` e `Fd` são especificados por cada regra. As equações acima presumem que tanto os pixels de origem quanto de destino têm os componentes de cor premultiplicados pelo componente alfa. Da mesma forma, as equações expressadas nas definições de regras de composição abaixo presumem alfa premultiplicado.

O alfa resultante da operação de composição é armazenado no destino se este possuir um canal alfa. Do contrário, a cor resultante é dividida pelo alfa resultante antes de ser armazenada no destino e o alfa é descartado. Se o valor alfa for 0,0f, os valores de cor são configurados para 0,0f.

Se os pixels de origem e destino possuírem componentes de cor que não tenham sido premultiplicados pelo componente alfa, são realizadas conversões apropriadas antes e depois da operação de composição.

Observação: não se aplicam restrições à implementação dessas regras.

Relaciona-se com:

`AlphaComposite`

50.5.2 Índice de campos

`static int` **`CLEAR`**

Regra Porter-Duff `Clear`.

`static AlphaComposite` **`Clear`**

Objeto `AlphaComposite` que implementa a regra `CLEAR` opaco com um alfa de 1,0f.

`static int` **`DST`**

Regra Porter-Duff Destination.

`static AlphaComposite` **`Dst`**

objeto `AlphaComposite` que implementa a regra `DST` opaco com um alfa de 1,0f.

`static int` **`DST_ATOP`**

Regra Porter-Duff `Destination Atop Source`.

`static int` **`DST_IN`**

Regra Porter-Duff `Destination In Source`.

`static int` **`DST_OUT`**

Regra Porter-Duff `Destination Held Out By Source`.

`static int` **`DST_OVER`**

Regra Porter-Duff `Destination Over Source`.

`static AlphaComposite` **`DstAtop`**

Objeto `AlphaComposite` que implementa a regra `DST_ATOP` opaco com um alfa de 1,0f.

`static AlphaComposite` **`DstIn`**

Objeto `AlphaComposite` que implementa a regra `DST_IN` opaco com um alfa de 1,0f.

`static AlphaComposite` **`DstOut`**

objeto `AlphaComposite` que implementa a regra opaco `DST_OUT` com um alfa de 1,0f.

`static AlphaComposite` **`DstOver`**

Objeto `AlphaComposite` que implementa a regra `DST_OVER` opaco com um alfa de 1,0f.

`static int` **`SRC`**

Regra Porter-Duff `Source`.

`static AlphaComposite` **`Src`**

Objeto `AlphaComposite` que implementa a regra `SRC` opaco com um alfa de 1,0f.

`static int` **`SRC_ATOP`**

Regra Porter-Duff `Source Atop Destination`.

`static int` **`SRC_IN`**

Regra Porter-Duff `Source In Destination`.

`static int` **`SRC_OUT`**

Regra Porter-Duff `Source Held Out By Destination`.

`static int` **`SRC_OVER`**

Regra Porter-Duff `Source Over Destination`.

`static AlphaComposite` **`SrcAtop`**

Objeto `AlphaComposite` que implementa a regra `SRC_ATOP` opaco com um alfa de 1,0f.

`static AlphaComposite` **`SrcIn`**

Objeto `AlphaComposite` que implementa a regra `SRC_IN` opaco com um alfa de 1,0f.

`static AlphaComposite` **`SrcOut`**

Objeto `AlphaComposite` que implementa a regra `SRC_OUT` opaco com um alfa de 1,0f.

`static AlphaComposite` **`SrcOver`**

Objeto `AlphaComposite` que implementa a regra `SRC_OVER` opaco com um alfa de 1,0f.

`static int` **`XOR`**

Regra Porter-Duff `Source Xor Destination`.

`static AlphaComposite` **`Xor`**

objeto `AlphaComposite` que implementa a regra `XOR` opaco com um alfa de 1,0f.

50.5.3 Índice de métodos

`boolean` **`equals`**`(Object obj)`

Determina se o objeto especificado é igual ao esse `AlphaComposite`.

`float` **`getAlpha`**`()`

Retorna o valor alfa desse `AlphaComposite`.

`static AlphaComposite` **`getInstance`**`(int rule)`

Cria um objeto `AlphaComposite` com a regra especificada.

`static AlphaComposite` **`getInstance`**`(int rule, float alpha)`

Cria um objeto `AlphaComposite` com a regra especificada e constante alfa para multiplicar com o alfa de origem.

`int` **`getRule`**`()`

Retorna a regra de composição desse `AlphaComposite`.

`int` **`hashCode`**`()`

Retorna o *hashcode* para esta combinação.

50.5.4 Detalhe dos campos

CLEAR

```
public static final int CLEAR = 1
```

Regra Porter-Duff `Clear`. Tanto a cor e o alfa de destino são limpos. Nem a fonte nem o destino são utilizados como entrada.

`Fs = 0 e Fd = 0`, então:

```
  Cd = 0
  Ad = 0
```

SRC

```
public static final int SRC = 2
```

Regra Porter-Duff `Source`. A origem é copiada para o destino. O destino não utilizado como entrada.

`Fs = 1 e Fd = 0`, então:

```
Cd = Cs
Ad = As
```

SRC_OVER

```
public static final int SRC_OVER = 3
```

Regra Porter-Duff `Source Over Destination`. A origem é combinada com o destino.

`Fs = 1 e Fd = (1-As)`, então:

```
Cd = Cs + Cd*(1-As)
Ad = As + Ad*(1-As)
```

DST

```
public static final int DST = 4
```

Regra Porter-Duff `Destination`. O destino permanece intacto.

`Fs = 0 e Fd = 1`, então:

```
Cd = Cd
Ad = Ad
```

DST_OVER

```
public static final int DST_OVER = 5
```

Regra Porter-Duff `Destination Over Source`. O destino é combinado com a origem e o resultado substitui o destino.

`Fs = (1-Ad) e Fd = 1`, então:

```
Cd = Cs*(1-Ad) + Cd
Ad = As*(1-Ad) + Ad
```

SRC_IN

```
public static final int SRC_IN = 6
```

Regra Porter-Duff `Source In Destination`. A parte da origem dentro do destino substitui o destino.

`Fs = Ad e Fd = 0`, então:

```
Cd = Cs*Ad
Ad = As*Ad
```

DST_IN

```
public static final int DST_IN = 7
```

Regra Porter-Duff `Destination In Source`. A parte do destino dentro da origem substitui o destino.

`Fs = 0 e Fd = As`, então:

```
Cd = Cd*As
```

```
  Ad = Ad*As
```

SRC_OUT

```
public static final int SRC_OUT = 8
```

Regra Porter-Duff `Source Held Out By Destination`. A parte da origem fora do destino substitui o destino.

`Fs = (1-Ad) e Fd = 0`, então:

```
  Cd = Cs*(1-Ad)
  Ad = As*(1-Ad)
```

DST_OUT

```
public static final int DST_OUT = 9
```

Regra Porter-Duff `Destination Held Out By Source`. A parte do destino fora da origem substitui o destino.

`Fs = 0 e Fd = (1-As)`, então:

```
  Cd = Cd*(1-As)
  Ad = Ad*(1-As)
```

SRC_ATOP

```
public static final int SRC_ATOP = 10
```

Regra Porter-Duff `Source Atop Destination`. A parte da origem dentro do destino é combinada com o destino.

`Fs = Ad e Fd = (1-As)`, então:

```
  Cd = Cs*Ad + Cd*(1-As)
  Ad = As*Ad + Ad*(1-As) = Ad
```

DST_ATOP

```
public static final int DST_ATOP = 11
```

Regra Porter-Duff `Destination Atop Source`. A parte do destino dentro da origem é combinada com a origem e substitui o destino.

`Fs = (1-Ad) e Fd = As`, então:

```
  Cd = Cs*(1-Ad) + Cd*As
  Ad = As*(1-Ad) + Ad*As = As
```

XOR

```
public static final int XOR = 12
```

Regra Porter-Duff `Source Xor Destination`. A parte da origem fora do destino é combinada com a parte do destino fora da origem.

`Fs = (1-Ad) e Fd = (1-As)`, então:

```
  Cd = Cs*(1-Ad) + Cd*(1-As)
  Ad = As*(1-Ad) + Ad*(1-As)
```

Clear

```
public static final AlphaComposite Clear
```

Objeto `AlphaComposite` que implementa a regra `CLEAR` opaco com um alfa de 1,0f.

Relaciona-se com:

`CLEAR`

Src

```
public static final AlphaComposite Src
```

Objeto `AlphaComposite` que implementa a regra `SRC` opaco com um alfa de 1,0f.

Relaciona-se com:

`SRC`

Dst

```
public static final AlphaComposite Dst
```

objeto `AlphaComposite` que implementa a regra `DST` opaco com um alfa de 1,0f.

Relaciona-se com:

`DST`

SrcOver

```
public static final AlphaComposite SrcOver
```

Objeto `AlphaComposite` que implementa a regra `SRC_OVER` opaco com um alfa de 1,0f.

Relaciona-se com:

`SRC_OVER`

DstOver

```
public static final AlphaComposite DstOver
```

Objeto `AlphaComposite` que implementa a regra `DST_OVER` opaco com um alfa de 1,0f.

Relaciona-se com:

`DST_OVER`

SrcIn

```
public static final AlphaComposite SrcIn
```

Objeto `AlphaComposite` que implementa a regra `SRC_IN` opaco com um alfa de 1,0f.

Relaciona-se com:

`SRC_IN`

DstIn

```
public static final AlphaComposite DstIn
```

Objeto `AlphaComposite` que implementa a regra `DST_IN` opaco com um alfa de 1,0f.

Relaciona-se com:

DST_IN

SrcOut

```
public static final AlphaComposite SrcOut
```

Objeto AlphaComposite que implementa a regra SRC_OUT opaco com um alfa de 1,0f.

Relaciona-se com:

SRC_OUT

DstOut

```
public static final AlphaComposite DstOut
```

objeto AlphaComposite que implementa a regra opaco DST_OUT com um alfa de 1,0f.

Relaciona-se com:

DST_OUT

SrcAtop

```
public static final AlphaComposite SrcAtop
```

Objeto AlphaComposite que implementa a regra SRC_ATOP opaco com um alfa de 1,0f.

Relaciona-se com:

SRC_ATOP

DstAtop

```
public static final AlphaComposite DstAtop
```

Objeto AlphaComposite que implementa a regra DST_ATOP opaco com um alfa de 1,0f.

Relaciona-se com:

DST_ATOP

Xor

```
public static final AlphaComposite Xor
```

objeto AlphaComposite que implementa a regra XOR opaco com um alfa de 1,0f.

Relaciona-se com:

XOR

50.5.5 Detalhe dos métodos

getInstance

```
public static AlphaComposite getInstance(int rule)
```

Cria um objeto AlphaComposite com a regra especificada.

Parâmetros:

rule – regra de combinação.

Retorna:

um objeto AlphaComposite.

Lança:

IllegalArgumentException – se a regra não for uma das seguintes: CLEAR, SRC, SRC_OVER, DST, DST_OVER, SRC_IN, DST_IN, SRC_OUT, DST_OUT, SRC_ATOP, DST_ATOP ou XOR

getInstance

```
public static AlphaComposite getInstance(int rule,
                                         float alpha)
```

Cria um objeto AlphaComposite com a regra especificada e constante alfa para multiplicar com o alfa de origem. A fonte é multiplicada com o alfa especificado antes de ser combinada com o destino.

Parâmetros:

rule – regra de combinação.

alpha – O alfa constante a ser multiplicado com o alfa da origem. o alpha deve ser um número de ponto flutuante na faixa inclusa [0,0f, 1,0f].

Retorna:

um objeto AlphaComposite.

Lança:

IllegalArgumentException - se alpha for menor que 0,0f ou maior que 1,0f, ou se a regra não for uma das seguintes: CLEAR, SRC, SRC_OVER, DST, DST_OVER, SRC_IN, DST_IN, SRC_OUT, DST_OUT, SRC_ATOP, DST_ATOP ou XOR

getAlpha

```
public float getAlpha()
```

Retorna o valor alfa desse AlphaComposite. Se este AlphaComposite não possuir um valor alfa, é retornado 1,0f.

Retorna:

valor alfa desse AlphaComposite.

getRule

```
public int getRule()
```

Retorna a regra de composição desse AlphaComposite.

Retorna:

regra de combinação desse AlphaComposite.

hashCode

```
public int hashCode()
```

Retorna o *hashcode* para esta combinação.

Substituições:

hashCode na classe Object

Retorna:

Um código *hash* para essa combinação.

equals

`public boolean equals(Object obj)`

Determina se o objeto especificado é igual ao esse `AlphaComposite`.

O resultado é *`true`* se e apenas se o argumento não for `null` e for um objeto `AlphaComposite` que possua a mesma regra de combinação e valor alfa que este objeto.

Substituições:

`equals` na classe `Object`

Parâmetros:

`obj` - `Object` para teste de igualdade

Retorna:

`true` se `obj` for igual a este `AlphaComposite`; caso contrário, *`false`*.

50.6 Interface Animated

50.6.1 Descrição da interface

`com.sun.dtv.ui`

Todas as classes implementadoras conhecidas

`AnimatedMatte, AWTComponent, Button, Calendar, CheckBox, ComboBox, CommonTransitions, Component, Container, DefaultListCellRenderer, Dialog, DTVContainer, Form, Label, List, MediaComponent, RadioButton, StaticAnimation, TabbedPane, TextArea, TextField, Transition`

`public interface Animated`

Esta interface fornece métodos para configurar e recuperar parâmetros de um efeito de animação. Uma animação é sempre interrompida por padrão e deve ser iniciada utilizando o método start. Antes de alterar o conteúdo de uma animação, ela deve ser interrompida se estiver executando no momento. O valor-padrão para a posição da animação é sempre a primeira imagem, a que apresenta índice 0. O modo de animação padrão é `REPEATING`, e o modo de repetição padrão é `LOOP`.

50.6.2 Índice de campos

`int ALTERNATING`

Valor para modo de animação.

`int LOOP`

Valor para modo de repetição.

`int REPEATING`

Valor para modo de animação.

50.6.3 Índice de métodos

`int getAnimationMode()`

Obtém o modo de animação para essa animação.

`int getDelay()`

Obtém o atraso após cada imagem dessa animação ao rodar.

`int getPosition()`

Obtém a posição atual que a animação ocupa no momento da chamada do método.

`int getRepetitionMode()`

Retorna o modo de repetição dessa animação na forma de um número.

`boolean isRunning()`

Obtém o modo rodando de uma animação.

`void jumpTo(int position)`

Força uma animação a saltar para a posição indicada pelo parâmetro.

`void setAnimationMode(int mode)`

Define o modo de animação para essa animação.

`void setDelay(int n)`

Determina o atraso após cada imagem dessa animação ao rodar.

`void setRepetitionMode(int n)`

Determina quão freqüentemente a animação é repetida, uma vez que for iniciada.

`void start()`

Inicia uma animação.

`void stop()`

Pára uma animação.

50.6.4 Detalhe dos campos

REPEATING

`public static final int REPEATING = 1`

Valor para modo de animação. `REPEATING` significa que a animação é repetida adiante. O número de repetiçoes depende do modo de repetição.

ALTERNATING

`public static final int ALTERNATING = 2`

Valor para modo de animação. `ALTERNATING` significa que a animação é repetida alternando adiante e para trás. O número de repetiçoes depende do modo de repetição.

LOOP

`public static final int LOOP = 0`

Valor para modo de repetição. `LOOP` significa que a animação executa em um *loop* sem fim. Como exatamente as animações executam neste *loop*, depende do modo de animação.

50.6.5 Detalhe dos métodos

start

`void start()`

Inicia uma animação. Considere que uma animação fica em modo parado automaticamente, portanto deve ser iniciada explicitamente uma vez que é criada. No caso da animação já estar rodando, uma chamada para `start()` faz com que ela reinicie.

stop

`void stop()`

Pára uma animação. No caso da animação já ter parado, uma chamada para `stop()` não tem efeito.

isRunning

`boolean isRunning()`

Obtém o modo rodando de uma animação. Retorna *true* se a animação já estiver rodando; caso contrário *false*.

Retorna:

true se a animação já estiver rodando; caso contrário *false*.

jumpTo

`void jumpTo(int position)`

Força uma animação a saltar para a posição indicada pelo parâmetro. Uma animação pode ser considerada como uma fila de imagens, o parâmetro desse método provê o índice dentro desse fila. Se a animação for parada, uma chamada para esse método faz com que a animação mostre a imagem na posição indicada, mas não que inicie. Se a animação estiver rodando, uma chamada para esse método faz com que a animação salte para a imagem na posição indicada e continue a rodar imediatamente.

Parâmetros:

`position` - o índice na fila de imagens fazendo a animação para o qual a animação é forçada a saltar

getPosition

`int getPosition()`

Obtém a posição atual que a animação ocupa no momento da chamada do método. Uma animação pode ser considerada como uma fila de imagens, esse método provê o índice dentro desse fila.

Retorna:

a posição atual da imagem dentro da fila de imagens fazendo a animação

setRepetitionMode

`void setRepetitionMode(int n)`

Determina quão freqüentemente a animação é repetida, uma vez que for iniciada.

Parâmetros:

`n` - número de repetições a ser determinado. Pode ser um número maior que zero ou `LOOP` alternativamente. `LOOP` significa que a animação roda em um *loop* infinito até que o método `stop()` seja chamado.

Relaciona-se com:

`getRepetitionMode()`

getRepetitionMode

`int getRepetitionMode()`

Retorna o modo de repetição dessa animação na forma de um número. O número lê quantas repetições foram determinadas para essa animação.

Retorna:

número de repetições determinado para essa animação ou `LOOP`, o que significa que a animação roda em um *loop* infinito até que o método `stop()` seja chamado.

Relaciona-se com:

`setRepetitionMode(int)`

setDelay

`void setDelay(int n)`

Determina o atraso após cada imagem dessa animação ao rodar.

Parâmetros:

`n` - atraso em milissegundos

Relaciona-se com:

`getDelay()`

getDelay

`int getDelay()`

Obtém o atraso após cada imagem dessa animação ao rodar.

Retorna:

o atraso em milissegundos

Relaciona-se com:

`setDelay(int)`

setAnimationMode

`void setAnimationMode(int mode)`

Define o modo de animação para essa animação. O modo de animação determina como a animação está rodando: sempre para frente, ou para frente e para trás alternando.

Parâmetros:

`mode` - o modo da animação. Valores possíveis são: `REPEATING` and `ALTERNATING`.

Relaciona-se com:

`getAnimationMode()`

getAnimationMode

`int getAnimationMode()`

Obtém o modo de animação para essa animação. O modo de animação determina como a animação está rodando: sempre para frente, ou para frente e para trás alternando.

Retorna:

O modo da animação. Valores possíveis são: `REPEATING` and `ALTERNATING`.

Relaciona-se com:

`setAnimationMode()`

50.7 Classe AnimatedMatte

50.7.1 Descrição da classe

`com.sun.dtv.ui`

`java.lang.Object`

`└com.sun.dtv.ui.AnimatedMatte`

Todas as interfaces implementadas

`Animated, Animation, Matte`

```
public class AnimatedMatte
extends Object
implements Matte, Animation, Animated
```

Essa classe representa um matte animado com uma máscara de imagem dinâmica, em que os valores de pixel determinam a transparência do matte para todos os pixels para um certo momento.

Relaciona-se com:

`Matte`

Campos herdados da interface `com.sun.dtv.ui.`**`Animated`**

`ALTERNATING, LOOP, REPEATING`

50.7.2 Índice de construtores

`AnimatedMatte()`

50.7.3 Índice de métodos

`boolean animate()`

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*.

`int getAnimationMode()`

Obtém o modo de animação para a animação de matte.

`int getDelay()`

Obtém o atraso depois de cada imagem para a animação de matte quando estiver executando.

```
Image[] getImageOpacity()
```

Obtém as imagens de valor de opacidade para este matte se uma animação de imagem tiver sido previamente atribuída a este matte, caso contrário, `null`.

```
Dimension getOffset(int index)
```

Obtém o deslocamento que o matte de imagem possui relativamente no canto esquerdo superior da área do componente para uma certa posição dentro da animação.

```
float[] getOpacity()
```

Obtém os valores de opacidade para a animação deste matte se uma sequência de valores de opacidade unitários for atribuída a ele no momento, caso contrário, `null`.

```
int[] getOpacityAsInts()
```

Obtém os valores de opacidade para a animação deste matte se uma sequência de valores de opacidade unitários for atribuída a ele no momento, caso contrário, `null`.

```
int getPosition()
```

Obtém a posição atual que a animação de matte ocupa no momento da chamada do método.

```
int getRepetitionMode()
```

Retorna o modo de repetição da animação de matte em forma de um número.

```
boolean hasImageAnimation()
```

Obtém se a animação atribuída atualmente a este matte é baseada em uma sequência de imagens ou não.

```
boolean isRunning()
```

Obtém o modo de execução para a animação de matte.

```
void jumpTo(int index)
```

Força a animação de matte a saltar para a posição indicada pelo parâmetro.

```
void paint(Graphics g)
```

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

```
void setAnimationMode(int mode)
```

Configura o modo de animação para a animação de matte.

```
void setDelay(int delay)
```

Determina o atraso depois de cada imagem ao executar a animação de matte.

```
void setOffset(int index, Dimension dimension)
```

Determina o deslocamento para a imagem de opacidade a ser atribuída com este matte para uma certa posição na animação.

```
void setOpacity(Image[] images)
```

Configura a opacidade do matte para uma sequência e imagens causando uma animação de alteração de imagens de opacidade.

```
void setOpacity(float[] opacities)
```

Configura valores para este matte, fazendo com que uma animação altere a opacidade, enquanto a opacidade em um certo momento é unitária para todos os pixels.

`void setOpacity(int[] opacities)`

Configura valores para este matte, fazendo com que uma animação altere a opacidade, enquanto a opacidade em um certo momento é unitária para todos os pixels.

`void setRepetitionMode(int repetitions)`

Determina a frequência com que a animação de matte é repetida uma vez, assim que for iniciada.

`void start()`

Inicia a animação de matte.

`void stop()`

Interrompe a animação de matte.

50.7.4 Detalhe dos construtores

AnimatedMatte

```
public AnimatedMatte()
```

50.7.5 Detalhe dos métodos

setOpacity

```
public void setOpacity(int[] opacities)
```

Configura valores para este matte, fazendo com que uma animação altere a opacidade, enquanto a opacidade em um certo momento é unitária para todos os pixels. Uma chamada a este método substitui uma animação de imagens ou valores unitários de opacidade previamente configurada. Este método tem exatamente o mesmo objetivo que `setOpacity(float[])`, onde um valor de número inteiro `i` no *array* passado para este método possui o mesmo efeito de um valor flutuante `(float)(i/255)` na mesma posição no *array* passado a `setOpacity(float[])`.

Parâmetros:

`opacities` – valores para alterar a opacidade durante a animação deste matte. Os elementos do *array* devem ser valores de número inteiro entre 0 e 255, onde 0 representa total transparência e 255 total opacidade. Valores abaixo de 0 devem ser interpretados como 0 (totalmente transparente), valores acima de 255 como 255 (totalmente opaco). Se o parâmetro é `null`, valores de opacidade anteriormente são removidos.

Lança:

`IllegalArgumentException` – se o comprimento do *array* de opacidade for 0.

Relaciona-se com:

`getOpacityAsInts()`

setOpacity

```
public void setOpacity(float[] opacities)
```

Configura valores para este matte, fazendo com que uma animação altere a opacidade, enquanto a opacidade em um certo momento é unitária para todos os pixels. Uma chamada a este método substitui uma animação de imagens ou valores unitários de opacidade previamente configurada. Este método tem exatamente o mesmo objetivo que `setOpacity(int[])`, onde um valor de ponto flutuante `f` no *array* passado para este método possui o mesmo efeito de um valor flutuante `(int)(f * 255)` na mesma posição no *array* passado a `setOpacity(int[])`.

Parâmetros:

`opacities` – valores para alterar a opacidade durante a animação deste matte. Os elementos de *array* devem ser valores de ponto flutuante entre 0,0f e 1,0f, onde 0,0f representa total transparência e 1,0f total opacidade.

Valores abaixo de 0,0f devem ser interpretados como 0,0f (totalmente transparente), e valores acima de 1,0f como 1 (totalmente opaco). Se o parâmetro é `null`, valores de opacidade anteriormente são removidos.

Lança:

`IllegalArgumentException` – se o comprimento do *array* de opacidade for 0.

Relaciona-se com:

`getOpacity()`

setOpacity

```
public void setOpacity(Image[] images)
```

Configura a opacidade do matte para uma sequência e imagens causando uma animação de alteração de imagens de opacidade. As imagens fornecidas podem ser menores que o componente afetado, neste caso a área restante é mostrada como seria coberta pelo `StaticMatte` com opacidade 1,0. Como padrão o matte de imagens é alinhado com o canto esquero superior da área dos componentes. Isso pode ser alterado (para qualquer imagem separadamente) utilizando o método `setOffset()`. Uma chamada a este método substitui uma animação de imagens ou valores unitários de opacidade previamente configurada.

Parâmetros:

`images` – valores para opacidade, que devem ser um *array* de imagens, contendo valores de pixel entre 0,0 e 1,0. 0,0 indica total transparência, e 1,0 total opacidade. Valores abaixo de 0,0 devem ser interpretados como 0,0 (totalmente transparente), valores acima de 1,0 como 1,0 (totalmente opaco). Observar que se uma imagem não estiver disponível enquanto a animação estiver executando (porque o elemento do *array* é `null` ou porque a imagem ainda está carregando) essa imagem deve ser pulada.

Relaciona-se com:

`getOpacity()`

getOpacity

```
public float[] getOpacity()
```

Obtém os valores de opacidade para a animação deste `matte` se uma sequência de valores de opacidade unitários for atribuída a ele no momento, caso contrário, `null`.

Retorna:

valores de opacidade para animação deste `matte`. Os elementos de *array* são sempre valores de ponto flutuante em uma faixa de 0,0 e 1,0, onde 0,0 indica total transparência, e 1,0 total opacidade. Se for retornado `null`, nenhum valor unitário de opacidade foi atribuído a este matte no momento, mas sim uma sequência de imagens de opacidade.

Relaciona-se com:

`setOpacity(float[])`

getOpacityAsInts

```
public int[] getOpacityAsInts()
```

Obtém os valores de opacidade para a animação deste matte se uma sequência de valores de opacidade unitários for atribuída a ele no momento, caso contrário, `null`.

Retorna:

valores de opacidade para animação deste `matte`. Os elementos de *array* são sempre valores de número inteiro

em uma faixa de 0 e 255, onde 0 indica total transparência, e 255 total opacidade. Se for retornado `null`, nenhum valor unitário de opacidade foi atribuído a este matte no momento, mas sim uma sequência de imagens de opacidade.

Relaciona-se com:

`setOpacity(int[])`

getImageOpacity

```
public Image[] getImageOpacity()
```

Obtém as imagens de valor de opacidade para este `matte` se uma animação de imagem tiver sido previamente atribuída a este `matte`, caso contrário, `null`.

Retorna:

os valores de opacidade para este `matte` em forma de um *array* de imagem, onde cada imagem contém valores de pixel entre 0,0 e 1,0. 0,0 indica total transparência, e 1,0 total opacidade. Se for retornado `null`, não há nenhuma animação de imagem de opacidade atribuída a este matte no momento.

Relaciona-se com:

`setOpacity(com.sun.dtv.lwuit.Image[])`

setOffset

```
public void setOffset(int index,
                      Dimension dimension)
```

Determina o deslocamento para a imagem de opacidade a ser atribuída com este `matte` para uma certa posição na animação. Apesar de, como padrão, o `matte` de imagens ser alinhado no canto esquerdo superior da área de componentes, isso pode alterado utilizando este método – separamente para qualquer posição de animação. Se atualmente uma animação baseada em valores unitários de opacidade é atribuída a este `matte`, uma chamada a este método não tem efeito.

Parâmetros:

`index` – índice do *array* de imagem de conteúdo para o qual o deslocamento deve ser configurado.

`dimension` – um objeto `Dimension` determinando o deslocamento em direção ao canto esquerdo superior da área de componente. Se o parâmetro for `null`, nenhum deslocamento deve ser determinado, e um deslocamento previamente atribuído deve ser removido.

Relaciona-se com:

`getOffset(int)`

getOffset

```
public Dimension getOffset(int index)
```

Obtém o deslocamento que o matte de imagem possui relativamente no canto esquerdo superior da área do componente para uma certa posição dentro da animação. Se atualmente uma animação baseada em valores unitários de opacidade é atribuída a este matte, retorna-se `null`.

Parâmetros:

`index` – o índice de posição de animação do valor de deslocamento deve ser obtido para

Retorna:

o deslocamento para esta posição de animação em forma de um objeto `Dimension` ou `null` (neste caso não há nenhum deslocamento para esta posição de animação, ou a animação válida atualmente é baseada em valores de opacidade unitários)

Relaciona-se com:

`setOffset(int, com.sun.dtv.lwuit.geom.Dimension)`

start

`public void start()`

Inicia a animação de matte. Considere que uma animação fica em modo parado automaticamente, portanto deve ser iniciada explicitamente uma vez que é criada. No caso da animação já estar rodando, uma chamada para `start()` faz com que ela reinicie.

Especificado por:

`start` na interface `Animated`

stop

`public void stop()`

Interrompe a animação de matte. No caso da animação já ter parado, uma chamada para `stop()` não tem efeito.

Especificado por:

`stop` na interface `Animated`

isRunning

`public boolean isRunning()`

Obtém o modo de execução para a animação de matte. Retorna *true* se a animação já estiver rodando; caso contrário *false*.

Especificado por:

`isRunning` na interface `Animated`

Retorna:

true se a animação já estiver rodando; caso contrário *false*.

jumpTo

`public void jumpTo(int index)`

Força a animação de matte a saltar para a posição indicada pelo parâmetro. Uma animação pode ser considerada como uma fila de imagens, o parâmetro desse método provê o índice dentro desse fila. Se a animação for parada, uma chamada para esse método faz com que a animação mostre a imagem na posição indicada, mas não que inicie. Se a animação estiver rodando, uma chamada para esse método faz com que a animação salte para a imagem na posição indicada e continue a rodar imediatamente.

Especificado por:

`jumpTo` na interface `Animated`

Parâmetros:

`index` – índice de sequência de imagens, construindo a animação, que esta é forçada a pular

getPosition

`public int getPosition()`

Obtém a posição atual que a animação de matte ocupa no momento da chamada do método. Uma animação pode ser considerada como uma fila de imagens, esse método provê o índice dentro desse fila.

Especificado por:

`getPosition` na interface `Animated`

Retorna:

a posição atual da imagem dentro da fila de imagens fazendo a animação

setRepetitionMode

`public void setRepetitionMode(int repetitions)`

Determina a frequência com que a animação de matte é repetida uma vez, assim que for iniciada.

Especificado por:

`setRepetitionMode` na interface `Animated`

Parâmetros:

`repetitions` – número de repetições a ser determinado. Pode ser um número maior que zero ou `LOOP` alternativamente. `LOOP` significa que a animação roda em um *loop* infinito até que o método `stop()` seja chamado.

Relaciona-se com:

`getRepetitionMode()`

getRepetitionMode

`public int getRepetitionMode()`

Retorna o modo de repetição da animação de matte em forma de um número. O número lê quantas repetições foram determinadas para essa animação.

Especificado por:

`getRepetitionMode` na interface `Animated`

Retorna:

número de repetições determinado para essa animação ou `LOOP`, o que significa que a animação roda em um *loop* infinito até que o método `stop()` seja chamado.

Relaciona-se com:

`setRepetitionMode(int)`

setDelay

`public void setDelay(int delay)`

Determina o atraso depois de cada imagem ao executar a animação de matte.

Especificado por:

`setDelay` na interface `Animated`

Parâmetros:

`delay` – atraso em milissegundos

Relaciona-se com:

`getDelay()`

getDelay

`public int getDelay()`

Obtém o atraso depois de cada imagem para a animação de matte quando estiver executando.

Especificado por:

`getDelay` na interface `Animated`

Retorna:

o atraso em milissegundos

Relaciona-se com:

`setDelay(int)`

setAnimationMode

`public void setAnimationMode(int mode)`

Configura o modo de animação para a animação de matte. O modo de animação determina como a animação está rodando: sempre para frente, ou para frente e para trás alternando.

Especificado por:

`setAnimationMode` na interface `Animated`

Parâmetros:

`mode` - o modo da animação. Valores possíveis são: `REPEATING` and `ALTERNATING`.

Relaciona-se com:

`getAnimationMode()`

getAnimationMode

`public int getAnimationMode()`

Obtém o modo de animação para a animação de `matte`. O modo de animação determina como a animação está rodando: sempre para frente, ou para frente e para trás alternando.

Especificado por:

`getAnimationMode` na interface `Animated`

Retorna:

O modo da animação. Valores possíveis são: `REPEATING` and `ALTERNATING`.

Relaciona-se com:

`setAnimationMode(int)`

hasImageAnimation

`public boolean hasImageAnimation()`

Obtém se a animação atribuída atualmente a este matte é baseada em uma sequência de imagens ou não.

Retorna:

true se a animação atual estiver baseada em uma sequência de imagens de opacidade, *false* se estiver baseada em uma sequência de valores de opacidade unitários.

animate

```
public boolean animate()
```

Descrição copiada da interface: **Animation**

Permite a animação a reduzir chamadas para "repintura", quando retorna *false*. É chamado uma vez para cada quadro.

Especificado por:

`animate` na interface `Animation`

Retorna:

true se uma repintura for desejada ou *false* se a repintura não for necessária

paint

```
public void paint(Graphics g)
```

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

Especificado por:

`paint` na interface `Animation`

50.8 Classe Capabilities

50.8.1 Descrição da classe

com.sun.dtv.ui

```
java.lang.Object
  └com.sun.dtv.ui.Capabilities
```

```
public class Capabilities
extends Object
```

Descreve as capacidades de um plano. Cada objeto plano (ou seja, uma instância de uma classe subclassificando a classe `Plane`) possui uma instância associada a essa classe. Essa instância pode ser recuperada com o método `Plane.getCapabilities()`.

50.8.2 Índice de métodos

```
int getBitsPerPixel()
```

Indica quantos bits são utilizados para codificar os pixels se as imagens forem renderizadas em um plano associado a esta instância de `Capabilities`.

```
int getColorCodingModel()
```

Indica o modelo de codificação de cor utilizado pelo plano associado a esta instância de `Capabilities`.

```
Dimension[] getSupportedPixelAspectRatios()
```

Fornece todas as razões de aspecto de pixel suportadas pelo plano associado a esta instância de `Capabilities`.

`Dimension[] getSupportedPlaneAspectRatios()`

Fornece todas as razões de aspecto de plano suportadas pelo plano associado a esta instância de `Capabilities`.

`Dimension[] getSupportedScreenResolutions()`

Fornece todos os tamanhos de tela suportados pelo plano associado a esta instância de `Capabilities`.

`boolean isAlphaBlendingSupported()`

Indica se uma combinação alfa é suportada pelo plano associado a esta instância de `Capabilities`.

`boolean isGIFRenderingSupported()`

Indica se uma renderização de imagens GIF é suportada para o plano associado a esta instância de `Capabilities`.

`boolean isGraphicsRenderingSupported()`

Indica se uma renderização de gráficos é suportada pelo plano associado a esta instância de `Capabilities`.

`boolean isImageRenderingSupported()`

Indica se uma renderização de imagens é suportada para o plano associado a esta instância de `Capabilities`.

`boolean isJPEGRenderingSupported()`

Indica se a renderização de imagens JPEG é suportada para o plano associado a esta instância de `Capabilities`.

`boolean isPNGRenderingSupported()`

Indica se a renderização de imagens PNG é suportada para o plano associado a esta instância de `Capabilities`.

`boolean isRealAlphaBlendingSupported()`

Indica se a combinação alfa real é suportada pelo plano associado a esta instância de `Capabilities`.

`boolean isVideoRenderingSupported()`

Indica se a renderização de vídeos é suportada para o plano associado a esta instância de `Capabilities`.

`boolean isWidgetRenderingSupported()`

Indica se a renderização de *widget* é suportada pelo plano associado a esta instância de `Capabilities`.

50.8.3 Detalhe dos métodos

getSupportedScreenResolutions

`public Dimension[] getSupportedScreenResolutions()`

Fornece todos os tamanhos de tela suportados pelo plano associado a esta instância de `Capabilities`.

Retorna:

um *array* de todos os tamanhos de tela suportados

getSupportedPlaneAspectRatios

`public Dimension[] getSupportedPlaneAspectRatios()`

Fornece todas as razões de aspecto de plano suportadas pelo plano associado a esta instância de `Capabilities`.

Retorna:

um *array* de todas as razões de aspecto de tela suportadas

getSupportedPixelAspectRatios

```
public Dimension[] getSupportedPixelAspectRatios()
```

Fornece todas as razões de aspecto de pixel suportadas pelo plano associado a esta instância de `Capabilities`.

Retorna:

um *array* de todas as razões de aspecto de pixel suportadas

isAlphaBlendingSupported

```
public boolean isAlphaBlendingSupported()
```

Indica se uma combinação alfa é suportada pelo plano associado a esta instância de `Capabilities`. O valor de retorno também é *true* se a cofiguração não suporta combinação alfa real (suportada por hardware), mas utiliza um suposto plano de comutação. Esse plano de comutação, oferecido por algumas implementações para poupar a combinação alfa suportada por hardware para planos de vídeo, é um plano adicional com um bit por pixel, apenas indicando para cada pixel se este plano ou o plano abaixo deve estar visível.

Para indicar se uma combinação alfa real (suportada por hardware) está disponível, o método `isRealAlphaBlendingSupported()` deve ser chamado.

Retorna:

true se a combinação alfa (realizada pelo suporte do hardware ou pelo uso de um plano de comutação) é suportada, caso contrário, *false*.

Relaciona-se com:

`isRealAlphaBlendingSupported()`

isRealAlphaBlendingSupported

```
public boolean isRealAlphaBlendingSupported()
```

Indica se a combinação alfa real é suportada pelo plano associado a esta instância de `Capabilities`. O valor de retorno é *true* apenas se o plano suporta combinação alfa real (suportada por hardware). Se a combinação alfa é apenas simulada pelo uso de um suposto plano de comutação, o valor de retorno é *false*.

Retorna:

true se a combinação alfa (descoberta pelo suporte do hardware, não pelo uso de um plano de comutação) é suportada, caso contrário, *false*.

Relaciona-se com:

`isAlphaBlendingSupported()`

isImageRenderingSupported

```
public boolean isImageRenderingSupported()
```

Indica se uma renderização de imagens é suportada para o plano associado a esta instância de `Capabilities`. O valor de retorno pode ser *`true`* para configurações para gráficos ou planos parados de vídeo, mas *`false`* para uma configuração adequada para um plano de legenda.

Retorna:

`true` se a renderização de imagem for suportada, caso contrário, *`false`*

isJPEGRenderingSupported

```
public boolean isJPEGRenderingSupported()
```

Indica se a renderização de imagens JPEG é suportada para o plano associado a esta instância de `Capabilities`. O valor de retorno só pode ser *`true`* se o valor de retorno de `isImageRenderingSupported()`, chamado para o mesmo objeto, também for *`true`*.

Retorna:

`true` se a renderização de imagem JPEG for suportada, caso contrário, *`false`*

isPNGRenderingSupported

```
public boolean isPNGRenderingSupported()
```

Indica se a renderização de imagens PNG é suportada para o plano associado a esta instância de `Capabilities`. O valor de retorno só pode ser *`true`* se o valor de retorno de `isImageRenderingSupported()`, chamado para o mesmo objeto, também for *`true`*.

Retorna:

`true` se a renderização de imagem PNG for suportada, caso contrário, *`false`*

isGIFRenderingSupported

```
public boolean isGIFRenderingSupported()
```

Indica se uma renderização de imagens GIF é suportada para o plano associado a esta instância de `Capabilities`. O valor de retorno só pode ser *`true`* se o valor de retorno de `isImageRenderingSupported()`, chamado para o mesmo objeto, também for *`true`*.

Retorna:

`true` se a renderização de imagem GIF for suportada, caso contrário, *`false`*

isVideoRenderingSupported

```
public boolean isVideoRenderingSupported()
```

Indica se a renderização de vídeos é suportada para o plano associado a esta instância de `Capabilities`. O valor de retorno poderia ser *`true`* para configurações para planos parados de vídeo, mas *`false`* para uma configuração adequada para um plano de legenda.

Retorna:

`true` se a renderização de vídeo for suportada, caso contrário, *`false`*

getBitsPerPixel

```
public int getBitsPerPixel()
                    throws SetupException
```

Indica quantos bits são utilizados para codificar os pixels se as imagens forem renderizadas em um plano associado a esta instância de `Capabilities`. A chamada para esse método faz sentido apenas se a renderização de imagem for suportada por este tipo de plano, ou seja, se o valor de retorno de `isImageRenderingSupported()` for *`true`* para este objeto `Capabilities`, caso contrário, uma exceção é

lançada.

Retorna:

número de bits utilizado para codificação de um único *pixel*

Lança:

`SetupException` – se esse método é chamado para um objeto `Capabilities` para o qual `isImageRenderingSupported()` retorna *false*

getColorCodingModel

```
public int getColorCodingModel()
                        throws SetupException
```

Indica o modelo de codificação de cor utilizado pelo plano associado a esta instância de `Capabilities`. O valor de retorno depende da implementação. As implementações são informadas para definir uma enumeração contendo valores para todos os modelos possíveis de codificação de cor suportados (por exemplo, YUV444, YUV422, YUV420, ARGB888 etc.). A chamada para esse método só faz sentido se no mínimo um dos seguintes métodos, chamadao para o mesmo objeto `Capabilities`, retorna *`true`*:

`isImageRenderingSupported()`

`isWidgetRenderingSupported()`

`isGraphicsRenderingSupported()`

caso contrário é executada uma exceção.

Retorna:

um valor dependente da implementação indicando que modelo de codificação de cor é utilizado

Lança:

`SetupException` – se esse método é chamado para um objeto `Capabilities` para o qual todos os métodos mencionados acima retornam *`false`*

isWidgetRenderingSupported

```
public boolean isWidgetRenderingSupported()
```

Indica se a renderização de *widget* é suportada pelo plano associado a esta instância de `Capabilities`. Um plano, para o qual um `DTVContainer` deve ser solicitado ao chamar um dos métodos `DTVContainer.getDTVContainer()`, deve estar associado ao objeto `Capabilities` para o qual este ou o método `isGraphicsRenderingSupported()` retorna *`true`*.

Retorna:

`true` se a renderização de *widget* for suportada, caso contrário, *`false`*

isGraphicsRenderingSupported

```
public boolean isGraphicsRenderingSupported()
```

Indica se uma renderização de gráficos é suportada pelo plano associado a esta instância de `Capabilities`. Um plano, para o qual um `DTVContainer` deve ser solicitado ao chamar um dos métodos `DTVContainer.getDTVContainer()`, deve estar associado ao objeto `Capabilities` para o qual este ou o método `isWidgetRenderingSupported()` retorna *`true`*.

Retorna:

`true` se a renderização de gráficos for suportada, caso contrário, *`false`*

50.9 Classe DefaultTextLayoutManager

50.9.1 Descrição da classe

com.sun.dtv.ui

java.lang.Object

└com.sun.dtv.ui.DefaultTextLayoutManager

Todas as interfaces implementadas

TextLayoutManager

```
public class DefaultTextLayoutManager
extends Object
implements TextLayoutManager
```

Essa classe fornece um mecanismo-padrão para renderização de textos fornecidos e é ao mesmo tempo a implementação mais simples da interface `TextLayoutManager`.

O `DefaultTextLayoutManager` é capaz de manipular alinhamento e justificação do texto contido no *string* fornecido tanto na horizontal quanto na vertical. A direção depende do modo de alinhamento atual válido do `ViewOnlyComponent`. No entanto, o redimensionamento não é suportado, e mesmo o modo de redimensionamento do `ViewOnlyComponent` é ignorado.

Textos de multilinhas são suportados no parâmetro do *string* fornecido para o método de renderização. As informações de cor e fonte necessárias para a renderização devem ser obtidas a patir do `ViewOnlyComponent` fornecido.

Relaciona-se com:

`TextLayoutManager`, `SophisticatedTextLayoutManager`

50.9.2 Índice de construtores

`DefaultTextLayoutManager()`

50.9.3 Índice de métodos

`Dimension getMaximumSize(ViewOnlyComponent component, String text)`

Fornece o tamanho máximo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

`Dimension getMaximumSize(ViewOnlyComponent component, String text, Insets insets)`

Fornece o tamanho máximo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

`Dimension getMinimumSize(ViewOnlyComponent component, String text)`

Fornece o tamanho mínimo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

`Dimension getMinimumSize(ViewOnlyComponent component, String text, Insets insets)`

Fornece o tamanho mínimo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

`Dimension getPreferredSize(ViewOnlyComponent component, String text)`

Fornece o tamanho preferido exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

`Dimension getPreferredSize(ViewOnlyComponent component, String text, Insets insets)`

Fornece o tamanho preferido exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

`void render(String text, Graphics g, ViewOnlyComponent component, Insets insets)`

Renderizar um *string.*

50.9.4 Detalhe dos construtores

DefaultTextLayoutManager

```
public DefaultTextLayoutManager()
```

50.9.5 Detalhe dos métodos

getMinimumSize

```
public Dimension getMinimumSize(ViewOnlyComponent component,
                                String text)
```

Descrição copiada da interface: **`TextLayoutManager`**

Fornece o tamanho mínimo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado. Nesta versão do método, presume-se que o parâmetro `ViewOnlyComponent` define as bordas pelos seus limites.

Especificado por:

`getMinimumSize` na interface `TextLayoutManager`

Parâmetros:

`component` - `ViewOnlyComponent` para o qual olhar

`text` – texto fornecido para ser renderizado

Retorna:

tamanho mínimo como objeto `com.sun.dtv.lwuit.geom.Dimension.`

getMinimumSize

```
public Dimension getMinimumSize(ViewOnlyComponent component,
                                String text,
                                Insets insets)
```

Descrição copiada da interface: **`TextLayoutManager`**

Fornece o tamanho mínimo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

Especificado por:

`getMinimumSize` na interface `TextLayoutManager`

Parâmetros:

component - ViewOnlyComponent para o qual olhar

text – texto fornecido para ser renderizado

insets – suplementos para definir a área na qual o texto deve ser colocado

Retorna:

tamanho mínimo como objeto com.sun.dtv.lwuit.geom.Dimension.

getMaximumSize

```
public Dimension getMaximumSize(ViewOnlyComponent component,
                                String text)
```

Descrição copiada da interface: **TextLayoutManager**

Fornece o tamanho máximo exigido para renderizar o conteúdo de texto fornecido no ViewOnlyComponent especificado. Nesta versão do método, presume-se que o parâmetro ViewOnlyComponent define as bordas pelos seus limites.

Especificado por:

getMaximumSize na interface TextLayoutManager

Parâmetros:

component - ViewOnlyComponent para o qual olhar

text – texto fornecido para ser renderizado

Retorna:

tamanho máximo como objeto com.sun.dtv.lwuit.geom.Dimension.

getMaximumSize

```
public Dimension getMaximumSize(ViewOnlyComponent component,
                                String text,
                                Insets insets)
```

Descrição copiada da interface: **TextLayoutManager**

Fornece o tamanho máximo exigido para renderizar o conteúdo de texto fornecido no ViewOnlyComponent especificado.

Especificado por:

getMaximumSize na interface TextLayoutManager

Parâmetros:

component - ViewOnlyComponent para o qual olhar

text – texto fornecido para ser renderizado

insets – suplementos para definir a área na qual o texto deve ser colocado

Retorna:

tamanho máximo como objeto com.sun.dtv.lwuit.geom.Dimension.

getPreferredSize

```
public Dimension getPreferredSize(ViewOnlyComponent component,
                                  String text)
```

Descrição copiada da interface: **`TextLayoutManager`**

Fornece o tamanho preferido exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado. Nesta versão do método, presume-se que o parâmetro `ViewOnlyComponent` define as bordas pelos seus limites.

Especificado por:

`getPreferredSize` na interface `TextLayoutManager`

Parâmetros:

`component` - `ViewOnlyComponent` para o qual olhar

`text` – texto fornecido para ser renderizado

Retorna:

tamanho preferido como objeto `com.sun.dtv.lwuit.Dimension`.

getPreferredSize

```
public Dimension getPreferredSize(ViewOnlyComponent component,
                                  String text,
                                  Insets insets)
```

Descrição copiada da interface: **`TextLayoutManager`**

Fornece o tamanho preferido exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

Especificado por:

`getPreferredSize` na interface `TextLayoutManager`

Parâmetros:

`component` - `ViewOnlyComponent` para o qual olhar

`text` – texto fornecido para ser renderizado

`insets` – suplementos para definir a área na qual o texto deve ser colocado

Retorna:

tamanho preferido como objeto `com.sun.dtv.lwuit.Dimension`.

render

```
public void render(String text,
                   Graphics g,
                   ViewOnlyComponent component,
                   Insets insets)
```

Renderiza uma *string*. O `ViewOnlyComponent` passado pode ser utilizado para determinar qualquer informação adicional necessária para renderizar a *string* de forma adequada (por ex., Fonte, Cor, etc.), se a classe `TextLayoutManager` implementando esta interface não fornecer as informações adicionais para isso.

O `ViewOnlyComponent` também define a área de layout por seus limites, enquanto os suplementos fornecidos também devem ser considerados se não forem `null`. Porém, o retângulo de recorte do objeto `Graphics` não

deve estar sujeito a mudanças pelo `TextLayoutManager`.

Especificado por:

`render` na interface `TextLayoutManager`

Parâmetros:

`text` – *string* a ser renderizado A *string* pode ter multilinhas, onde cada linha é separada por um "\n "(0x0A). Se a *string* não couber no espaço disponível, deve ser truncado e deve ser adicionada uma elipse ("...") para indicar o truncamento.

`g` – contexto dos gráficos. Isso inclui também um retângulo de recorte, que deve ser respeitado como bordas dentro das quais a renderização é permtida. Um parâmetro de suplementos diferente de `null` deve ser também levado em consideração.

`component` - `ViewOnlyComponent` no qual renderizar. As informações de cor e fonte a serem obtidas a partir daqui. Se a fonte especificada não estiver disponível, deve ser substituída pela fonte existente mais próxima. Cada caractere que estiver faltando é substituído por um caractere "*".

`insets` – suplementos para definir a área na qual o texto deve ser colocado. Este parâmetro também pode ser `null`: nesse caso, o parâmetro `ViewOnlyComponent` define as bordas pelos seus limites.

50.10 Classe Device

50.10.1 Descrição da classe

`com.sun.dtv.ui`

```
java.lang.Object
  └com.sun.dtv.ui.Device
```

```
public class Device
extends Object
```

Essa classe é uma representação de um aparelho de TV. Tal dispositivo possui no mínimo uma `Screen`, devem existir também diversas telas. `UserInputDevices` estão sempre ligados a elas. Mesmo se existir um dispositivo de input de usuário fisicamente (por exemplo, um `RemoteControl`) que pode ser utilizado para controlar mais que uma tela (por exemplo, como possui uma chave habilitando o usuário a trocar entre telas a serem controladas por tal dispositivo), ele deve ser logicamente dividido em um `UserInputDevice` por `Screen`.

A seguir está um recorte de código, demonstrando como trabalhar com um dispositivo e as telas associadas.

```
// Em primeiro lugar, pegar Screen. Iremos usar apenas a tela padrão uma vez que
// na maioria dos casos, teremos, provavelmente, só uma.
device = Device.getInstance();
screen = device.getDefaultScreen();
// Agora que temos a tela, seremos capazes de configurar vários
// planos sobre ela. Como este é um UIDemo, vamos nos concentrar no
// plano gráfico.
Plane[] planes = screen.getAllPlanes();
// busca de um plano com recursos gráficos
for(int i=0; i<planes.length; i++) {
     Capabilities cap = planes[i].getCapabilities();
     if (cap.isGraphicsRenderingSupported()) {
             plane = planes[i];
             break;
```

```
      // neste caso, tomamos o primeiro plano com capacidade
      // gráfica para utilização posterior como plano de gráficos.
      // claro que também podemos buscar por vários
      // planos com capacidades gráficas,
      // e escolher um, verificando recursos adicionais
     }
}
// Cria um novo template para os gráficos
// instalação e começar a definir preferências
PlaneSetupPattern pattern;
pattern = new PlaneSetupPattern();
// Preferimos uma instalação que suporte dimensionamento de imagem
pattern.setPreference(PlaneSetupPattern.IMAGE_SCALING_SUPPORT,
                      PlaneSetupPattern.PREFERRED);
// Também é ncessário uma instalação que não afete os aplicativos que estão em execução
pattern.setPreference(PlaneSetupPattern.NO_GRAPHICS_IMPACT,
                       PlaneSetupPattern.REQUIRED);
// Agora pegue um configuração do plano que corresponda a nossas preferências
PlaneSetup setup;
setup = plane.getBestSetup(pattern);
// Finalmente, somos capazes de definir a configuração. Antes de fazer isso,
// precisamos verificar se nossa configuração não é null (para se certificar de
// que nossas preferências poderiam realmente ser atingidas).
if (setup != null) {
     try {
      // temos de reservar o primeiro plano como um recurso escasso
      // como planejamos modificá-lo definindo uma nova configuração
      // não podemos forçar isto, ou seja, esperar até que plano seja liberado
      // para nós, não podemos especificar um tempo limite, e o
      // ScarceResourceListener é o nosso aplicativo (tem que
      // implementar a interface ScarceResourceListener para isto
              plane.reserve(false,-1,this);
              plane.setPlaneSetup(setup);
              // depois de definir a configuração, nos podemos liberar o plano
              plane.release();
     } catch (Exception e) {
              // faça algo se a definição da configuração não funcionar
     }
}
// Agora, com o GraphicsPlane configurado, nós podemos recorrer a ele
// Primeiro, precisaremos de um DTVContainer, o componente de nível mais alto
DTVContainer container;
container = DTVContainer.getDTVContainer(plane);
// Alternativamente, poderíamos neste caso tomar
// o container DTV padrão, como temos apenas uma tela com
// um plano de gráfico:
// container = DTVContainer.getDefaultDTVContainer();
// Agora nós podemos trabalhar e desenhar widgets e gráficos com o DTVContainer.
```

```
// ...
// Desta vez, nós precisamos saber se nossa tela suporta o controle remoto da TV
// e - if yes (se sim) – pegue-a:
if(screen.isRemoteControlSupported()) {
  RemoteControl rc = screen.getRemoteControl();
  // ...
}
// ...
```

50.10.2 Índice de construtores

`protected Device()`

Não há intenção em habilitar os usuários a criar um dispositivo.

50.10.3 Índice de métodos

`Screen getDefaultScreen()`

Retorna a tela padrão para este dispositivo.

`static Device getInstance()`

Recuperação da instância `Device` que representa o aparelho de TV utilizado a partir da implementação.

`Screen[] getScreens()`

Retorna uma lista de todas as telas associadas com o `Device`.

50.10.4 Detalhe dos construtores

Device

`protected Device()`

Não há intenção em habilitar os usuários a criar um dispositivo. A instância do dispositivo deve ser recuperada a partir da implementação das API Java DTV em um dispositivo em particular chamando `getInstance()`.

50.10.5 Detalhe dos métodos

getInstance

`public static Device getInstance()`

Recuperação da instância `Device` que representa o aparelho de TV utilizado a partir da implementação.

Retorna:

instância de dispositivo representando o aparelho de TV utilizado atualmente.

getDefaultScreen

`public Screen getDefaultScreen()`

Retorna a tela padrão para este dispositivo.

Retorna:

Tela padrão para este dispositivo.

getScreens

```
public Screen[] getScreens()
```

Retorna uma lista de todas as telas associadas com o dispositivo.

Retorna:

lista de todas as telas associadas ao dispositivo.

50.11 Classe DownloadableFont

50.11.1 Descrição da classe

com.sun.dtv.ui

```
java.lang.Object
  └com.sun.dtv.ui.DownloadableFont
```

```
public class DownloadableFont
extends Object
```

Essa classe introduz a possibilidade de baixar fontes. O método oferecido baixa uma fonte (família) a partir de uma URL especificada, e a instala na plataforma. Em seguida, a fonte instalada (ou fontes se o arquivo trouxer informações para uma família de fonte) pode ser utilizada como qualquer outra fonte preinstalada, utilizando a classe Font.

As implementações devem garantir que qualquer sintaxe URI seja suportada, inclusive qualquer tipo de localizadores.

O arquivo de fonte localizado no lugar especificado pela URL deve estar presente em formato de fonte aberto.

50.11.2 Índice de construtores

Down**loadableFont**()

50.11.3 Índice de métodos

```
static void installFont(URL url)
```

Baixa uma fonte a partir de uma URL especificada e instala a fonte (ou fontes se as informações para uma família de fonte forem encontradas) no sistema.

```
static void installFont(URL url, int timeout)
```

Baixa uma fonte a partir de uma URL especificada e instala a fonte (ou fontes se as informações para uma família de fonte forem encontradas) no sistema.

50.11.4 Detalhe dos construtores

DownloadableFont

```
public DownloadableFont()
```

50.11.5 Detalhe dos métodos

installFont

```
public static void installFont(URL url)
                        throws IOException,
                               IllegalArgumentException,
```

```
                                SecurityException,
                                FontFileException
```

Baixa uma fonte a partir de uma URL especificada e instala a fonte (ou fontes se as informações para uma família de fonte forem encontradas) no sistema. A fonte ou fontes instaladas podem ser, em seguida, utilizadas como fontes preinstaladas, utilizando a classe `Font`.

Atente ao fato de que esse método trabalha de forma sincrônica, ou seja, retorna logo depois que as informações de fonte foram baixadas com sucesso.

O arquivo encontrado nos locais especificados pela URL devem conter as informações de fonte em formato aberto.

Parâmetros:

`url` - a URL especificando o arquivo de fonte a ser baixado; qualquer tipo de URL é válido, inclusive qualquer forma de localizador

Lança:

`IOException` – se um erro ocorre durante a tentativa de acessar os dados aos quais a URL faz referência

`IllegalArgumentException` – se não for fornecida uma URL válida

`SecurityException` – se a política de segurança atual não permite acesso à URL especificada

`FontFileException` – se o arquivo ao qual a URL faz referência não pode ser reconhecido como um arquivo de fonte válido em um formato aberto

installFont

```
public static void installFont(URL url,
                               int timeout)
                        throws IOException,
                               IllegalArgumentException,
                               SecurityException,
                               FontFileException
```

Baixa uma fonte a partir de uma URL especificada e instala a fonte (ou fontes se as informações para uma família de fonte forem encontradas) no sistema. A fonte ou fontes instaladas podem ser, em seguida, utilizadas como fontes preinstaladas, utilizando a classe `Font`.

Observar que este método trabalha de forma sincrônica, ou seja, retorna logo depois que as informações de fonte forem baixadas com sucesso, ou o tempo especificado no parâmetro de *timeout* acabou sem um download bem-sucedido das informações de fonte.

O arquivo encontrado nos locais especificados pela URL devem conter as informações de fonte em formato aberto.

Parâmetros:

`url` - a URL especificando o arquivo de fonte a ser baixado; qualquer tipo de URL é válido, inclusive qualquer forma de localizador

`timeout` - *timeout* em milissegundos

Lança:

`IOException` – se um erro ocorre durante a tentativa de acessar os dados que a URL faz referência, inclusive *timeout* como especificado em um parâmetro de *timeout*

`IllegalArgumentException` – se não for fornecida uma URL válida

`SecurityException` – se a política de segurança atual não permite acesso à URL especificada

`FontFileException` – se o arquivo ao qual a URL faz referência não pode ser reconhecido como um arquivo de

fonte válido em um formato aberto.

50.12 Classe DTVContainer

50.12.1 Descrição da classe

com.sun.dtv.ui

```
java.lang.Object
  └com.sun.dtv.lwuit.Component
      └com.sun.dtv.lwuit.Container
          └com.sun.dtv.ui.DTVContainer
```

Todas as interfaces implementadas

```
Animated, Animation, MatteEnabled, StyleListener
```

```
public class DTVContainer
extends Container
```

Um `DTVContainer` é o container de nível mais alto na hierarquia de componente da API Java DTV, representando o *widget* contendo qualquer outra coisa visível em um plano em particular. Sempre há apenas um `DTVContainer` disponível por plano.

O `DTVContainer` nunca está sujeito a atividades de pintura, sua única tarefa é o recorte dos componentes contidos. Com respeito a isso, ele difere dos *containers* "normais", embora esteja subclassificando essa classe.

O `DTVContainer` também adiciona capacidades *Z-ordering* para habilitar sobreposição de *widget*s dentro de um `DTVContainer` a ser organizado por um aplicativo no *Z-ordering* desejado.

Sendo um `Component`, um `DTVContainer` é capaz de receber eventos de input e manipulá-los com o uso de handlers de evento anexos. A visibilidade de um `DTVContainer` não é garantia de que também receberá eventos. Por isso, o aplicativo associado deve garantir que o `DTVContainer` receba foco de input. O `DTVContainer` então passará o foco de input para um dos componentes contidos de uma maneira dependente da implementação. Se o aplicativo inteiro ganhar ou perder o foco de input (e ao mesmo tempo também o `DTVContainer` associado), isso deve ser indicado pelo sistema enviando um evento `WINDOW_ACTIVATED` ou `WINDOW_DEACTIVATED` ao `DTVContainer`, respectivamente.

Um `DTVContainer` não pode ser criado ao chamar explicitamente um construtor. A razão é que ao solicitar um `DTVContainer` deve-se considerar uma variedade de condições, determinadas pelas capacidades do aparelho de TV, a tela, e o plano aos quais o `DTVContainer` é solicitado. Diversos métodos de fábrica desta classe fornecem a capacidade de considerar tudo e fornecer o `DTVContainer` ideal para um aplicativo. Além disso, um aplicativo pode especificar as características de um `DTVContainer` desejado e as prioridades para as características desejadas, utilizando a classe `DTVContainerPattern`. O método `getBestDTVContainer` aceita um argumento do tipo `DTVContainerPattern` e fornece um `DTVContainer` considerando as preferências, se possível.

50.12.2 Índice de campos

`static int BACKGROUND_FILL`

Constante a ser utilizada no método `setBackgroundMode`.

`static int BACKGROUND_IMAGE_CENTERED`

Constante a ser utilizada no método `setBackgroundImageRenderMode`.

`static int BACKGROUND_IMAGE_NONE`

Constante a ser utilizada no método `setBackgroundImageRenderMode`.

`static int BACKGROUND_IMAGE_STRETCHED`

Constante a ser utilizada no método `setBackgroundImageRenderMode`.

`static int BACKGROUND_IMAGE_TILED`

Constante a ser utilizada no método `setBackgroundImageRenderMode`.

`static int BACKGROUND_NO_FILL`

Constante a ser utilizada no método `setBackgroundMode`.

Campos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

`BOTTOM, BRB_CENTER_OFFSET, BRB_CONSTANT_ASCENT, BRB_CONSTANT_DESCENT, BRB_OTHER, CENTER, LEFT, RIGHT, TOP`

Campos herdados da interface `com.sun.dtv.ui.`**`Animated`**

`ALTERNATING, LOOP, REPEATING`

50.12.3 Índice de métodos

`void addToBack(Component component)`

Adiciona um componente a este `DTVContainer` atrás de todos os componentes adicionados anteriormente.

`void addToFront(Component component)`

Adiciona um componente a este `DTVContainer` diretamente na frente de todos os componentes adicionados anteriormente.

`void addWindowListener(WindowListener listener)`

Adiciona um *listener* para receber qualquer `java.awt.event.WindowEvents` enviado ao `DTVContainer`.

`void consumeWindowEvent(WindowEvent event)`

Consome um `java.awt.event.WindowEvents` para este `DTVContainer`.

`void dispose()`

Descarte deste `DTVContainer`.

`Image getBackgroundImage()`

Obtém a imagem de plano de fundo atribuída a este `DTVContainer`, se aplicável.

`int getBackgroundImageRenderMode()`

Obtém o modo de renderização para uma imagem de plano de fundo como especificado utilizando o método `setBackgroundImageRenderMode`.

`int getBackgroundMode()`

Obtém o modo de plano de fundo como especificado utilizando o método `setBackgroundMode`.

`static DTVContainer getBestDTVContainer(DTVContainerPattern pattern)`

Retorna um `DTVContainer` que melhor corresponda ao input `DTVContainerPattern`, ou `null` se tal `DTVContainer` não puder ser encontrado.

`static DTVContainerPattern getBestDTVContainerPattern(DTVContainerPattern pattern)`

Retorna um `DTVContainerPattern` mais próximo ao input `DTVContainerPattern`, ao mesmo tempo especificando um `DTVContainer` que pôde ser criado com sucesso na plataforma ao mesmo tempo que este método é chamado.

```
static DTVContainer getCurrentDTVContainer()
```

Retorna o `DTVContainer` em que o chamador é exibido dentro.

```
static DTVContainer getDTVContainer(Plane plane)
```

Cria um `DTVContainer` para o plano especificado.

```
static DTVContainer getDTVContainer(Plane plane, Container container)
```

Cria um `DTVContainer` para o plano especificado, que pode ser utilizado como um container Xlet.

```
DTVContainerPattern getDTVContainerPattern()
```

Retorna um `DTVContainerPattern` descrevendo este `DTVContainer`.

```
Component getFocusOwnerComponent()
```

Obtém o componente pertencente a este `DTVContainer` que atualmente possui foco de input.

```
boolean isDoubleBuffered()
```

Obtém se todas as operações de desenho executadas para este `DTVContainer` são automaticamente armazenados duas vezes.

```
boolean isOpaque()
```

Obtém se a área total do `DTVContainer` como dada pelo método `Component.getBounds()` é totalmente opaca.

```
void paint(Graphics g)
```

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

```
void pop(Component component)
```

Move o component especificado um nível acima no `Z-order`.

```
void popInFrontOf(Component move, Component behind)
```

Move o Component especificado diretamente em frente da parte de trás do Component.

```
void popToFront(Component component)
```

Traz o Component especificado para frente do `Z-order` neste `DTVContainer`.

```
void push(Component component)
```

Move o Component especificado um nível abaixo no `Z-order`.

```
void pushBehind(Component move, Component front)
```

Move o Component especificado diretamente atrás da parte da frente do componente.

```
void pushToBack(Component component)
```

Traz o Component especificado para trás do `Z-order` neste `DTVContainer`.

```
void removeWindowListener(WindowListener listener)
```

Remove um *listener* do `DTVContainer` de forma que ele não receba mais qualquer `java.awt.event.WindowEvents`.

```
DTVContainerPattern resize(DTVContainerPattern pattern)
```

Redimensiona este `DTVContainer` de forma que melhor corresponda ao `DTVContainerPattern` de entrada, ou não faz nada se não for necessário.

```
void setBackgroundImage(Image image)
```

Configura uma imagem de plano de fundo a ser pintada no plano de fundo do `DTVContainer`.

```
void setBackgroundImageRenderMode(int mode)
```

Configura o modo de renderização de uma imagem de plano de fundo para este `DTVContainer`.

```
void setBackgroundMode(int mode)
```

Configura o modo de plano de fundo deste `DTVContainer`.

```
void setToFront()
```

Traz o `DTVContainer` para frente.

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Container`**

```
addComponent, addComponent, addComponent, contains, doLayout, getComponentAt,
getComponentAt, getComponentCount, getComponentIndex, getLayout, getLayoutHeight,
getLayoutWidth, getMatte, invalidate, isScrollableX, isScrollableY, layoutContainer,
pointerPressed, refreshTheme, removeAll, removeComponent, replace, revalidate,
scrollComponentToVisible, setCellRenderer, setLayout, setMatte, setScrollable,
setScrollableX, setScrollableY
```

Métodos herdados da classe `com.sun.dtv.lwuit.`**`Component`**

```
addFocusListener, animate, calcPreferredSize, contains, deinitialize, getAbsoluteX,
getAbsoluteY, getAnimationMode, getBaseline, getBaselineResizeBehavior, getBottomGap,
getBounds, getClientProperty, getComponentForm, getDelay, getHeight,
getNextFocusDown, getNextFocusLeft, getNextFocusRight, getNextFocusUp, getParent,
getPosition, getPreferredSize, getRepetitionMode, getScrollAnimationSpeed,
getScrollX, getScrollY, getSideGap, getStyle, getUIID, getWidth, getX, getY,
handlesInput, hasFocus, initialize, isEnabled, isFocusable, isFocusPainted,
isInitialized, isRunning, isScrollVisible, isSmoothScrolling, isVisible, jumpTo,
keyPressed, keyReleased, keyRepeated, longKeyPress, paintBackgrounds, paintComponent,
paintComponent, pointerDragged, pointerReleased, putClientProperty,
removeFocusListener, repaint, repaint, requestFocus, scrollRectToVisible,
setAnimationMode, setDelay, setEnabled, setFocus, setFocusable, setFocusPainted,
setHandlesInput, setHeight, setInitialized, setIsScrollVisible, setNextFocusDown,
setNextFocusLeft, setNextFocusRight, setNextFocusUp, setPreferredSize,
setRepetitionMode, setScrollAnimationSpeed, setScrollX, setScrollY,
setShouldCalcPreferredSize, setSize, setSmoothScrolling, setStyle, setUIID,
setVisible, setWidth, setX, setY, start, stop, styleChanged, toString
```

50.12.4 Detalhe dos campos

BACKGROUND_IMAGE_NONE

```
public static final int BACKGROUND_IMAGE_NONE = 0
```

Constante a ser utilizada no método `setBackgroundImageRenderMode`. Não causa renderização na imagem de plano de fundo neste `DTVContainer`.

BACKGROUND_IMAGE_STRETCHED

```
public static final int BACKGROUND_IMAGE_STRETCHED = 1
```

Constante a ser utilizada no método `setBackgroundImageRenderMode`. Faz com que uma imagem de plano de

fundo renderizada seja esticada para preencher o DTVContainer.

BACKGROUND_IMAGE_CENTERED

```
public static final int BACKGROUND_IMAGE_CENTERED = 2
```

Constante a ser utilizada no método setBackgroundImageRenderMode. Faz com que uma imagem de plano de fundo renderizada seja centralizada dentro das bordas do DTVContainer.

BACKGROUND_IMAGE_TILED

```
public static final int BACKGROUND_IMAGE_TILED = 3
```

Constante a ser utilizada no método setBackgroundImageRenderMode. Faz com que uma imagem de plano de fundo renderizada seja mosaicada dentro das bordas do DTVContainer.

BACKGROUND_NO_FILL

```
public static final int BACKGROUND_NO_FILL = 0
```

Constante a ser utilizada no método setBackgroundMode. Faz com que o DTVContainer não seja preenchido (com cores) antes que uma imagem ou componentes de plano de fundo sejam pintados.

BACKGROUND_FILL

```
public static final int BACKGROUND_FILL = 1
```

Constante a ser utilizada no método setBackgroundMode. Faz com que o DTVContainer não seja preenchido (com a cor atual do plano de fundo) antes que uma imagem ou componentes de plano de fundo sejam pintados.

50.12.5 Detalhe dos métodos

paint

```
public void paint(Graphics g)
```

Desenha a animação, dentro um componente o método de pintura padrão seria invocado, uma vez que ele carrega a mesma assinatura exata.

Se o modo de plano de fundo atual for BACKGROUND_FILL, a área toda do DTVContainer é primeiramente preenchida com a cor do plano de fundo atual. Se a imagem de plano de fundo tiver sido configurada utilizando o método setBackgroundImage, a imagem especificada é pintada a menos que o modo de renderização de imagem de plano de fundo atual seja BACKGROUND_IMAGE_NONE. Em todos os outros casos, a imagem deve ser redimensionada ou mosaicada como especificado pela chamada do método setBackgroundImageRenderMode(). Então, todos os componentes presentes são pintados levando em consideração o Z-order especificado.

Especificado por:

paint na interface Animation

Substituições:

paint na classe Container

Parâmetros:

g - os gráficos do componente

getBackgroundMode

```
public int getBackgroundMode()
```

Obtém o modo de plano de fundo como especificado utilizando o método `setBackgroundMode`.

Retorna:

um do `BACKGROUND_NO_FILL` ou `BACKGROUND_FILL`.

Relaciona-se com:

`setBackgroundMode(int)`

setBackgroundMode

```
public void setBackgroundMode(int mode)
                       throws IllegalArgumentException
```

Configura o modo de plano de fundo deste `DTVContainer`.

Se o modo de plano de fundo for configurado para `BACKGROUND_FILL`, automaticamente fará com que o método isOpaque retorne *`true`*.

Parâmetros:

`mode` - um do `BACKGROUND_NO_FILL` ou `BACKGROUND_FILL`.

Lança:

`IllegalArgumentException` – se o parâmetro do modo apresentar um valor inválido

Relaciona-se com:

`getBackgroundMode()`

getBackgroundImageRenderMode

```
public int getBackgroundImageRenderMode()
```

Obtém o modo de renderização para uma imagem de plano de fundo como especificado utilizando o método `setBackgroundImageRenderMode`.

Retorna:

um de `BACKGROUND_IMAGE_NONE, BACKGROUND_IMAGE_STRETCHED, BACKGROUND_IMAGE_CENTERED` ou `BACKGROUND_IMAGE_TILED`.

Relaciona-se com:

`setBackgroundImageRenderMode(int)`

setBackgroundImageRenderMode

```
public void setBackgroundImageRenderMode(int mode)
```

Configura o modo de renderização de uma imagem de plano de fundo para este `DTVContainer`.

Parâmetros:

`mode` - um de `BACKGROUND_IMAGE_NONE, BACKGROUND_IMAGE_STRETCHED,`

BACKGROUND_IMAGE_CENTERED ou BACKGROUND_IMAGE_TILED.

Relaciona-se com:

getBackgroundImageRenderMode()

getBackgroundImage

public Image getBackgroundImage()

Obtém a imagem de plano de fundo atribuída a este DTVContainer, se aplicável. Considere que mesmo que uma imagem seja atribuída, ela possivelmente não pode ser renderizada, dependendo do modo de renderização da imagem de plano de fundo.

Retorna:

imagem de plano de fundo atribuída a este DTVContainer, ou null se nenhuma imagem for atribuída.

Relaciona-se com:

setBackgroundImage(com.sun.dtv.lwuit.Image), setBackgroundImageRenderMode()

setBackgroundImage

public void setBackgroundImage(Image image)

Configura uma imagem de plano de fundo a ser pintada no plano de fundo do DTVContainer. O modo como a imagem deve ser renderizada depende do modo de renderização atualmente válido como configurado utilizando o método setBackgroundImageRenderMode. Considere que um plano de fundo atribuído pode não ser renderizado, dependendo do modo de renderização da imagem de plano de fundo atualmente válido.

Parâmetros:

image – imagem de plano de fundo a ser atribuída a este DTVContainer. Um valor null faz com que uma imagem atualmente atribuída seja removida.

Relaciona-se com:

getBackgroundImage(), setBackgroundImageRenderMode()

isDoubleBuffered

public boolean isDoubleBuffered()

Obtém se todas as operações de desenho executadas para este DTVContainer são automaticamente armazenados duas vezes.

Retorna:

true se todas as operações de desenho forem armazenadas duas vezes, caso contrário, *false*.

isOpaque

public boolean isOpaque()

Obtém se a área total do DTVContainer como dada pelo método Component.getBounds() é totalmente opaca.

Retorna:

true se todos os pixels dentro da área dada pelo método Component.getBounds() forem pintados com uma cor opaca, caso conttrário *false*.

addToFront

public void addToFront(Component component)

Adiciona um Component a este DTVContainer diretamente na frente de todos os componentes adicionados anteriormente.

Parâmetros:

component - Component a ser adicionado a este DTVContainer

addToBack

```
public void addToBack(Component component)
```

Adiciona um Component a este DTVContainer atrás de todos os componentes adicionados anteriormente.

Parâmetros:

component - Component a ser adicionado a este DTVContainer

popToFront

```
public void popToFront(Component component)
                throws IllegalArgumentException
```

Traz o componente especificado para frente do Z-order neste DTVContainer. Se o componente já estiver na frente do Z-order, ou for o único componente neste DTVContainer, a chamada do método não tem nehum efeito.

Parâmetros:

component - componente a ser trazido para frente do Z-order deste DTVContainer.

Lança:

IllegalArgumentException – se o componente especificado não estiver presente neste DTVContainer

pushToBack

```
public void pushToBack(Component component)
                throws IllegalArgumentException
```

Traz o componente especificado para trás do Z-order neste DTVContainer. Se o componente já estiver atrás do Z-order, ou for o único componente neste DTVContainer, a chamada do método não tem nehum efeito.

Parâmetros:

component - componente a ser trazido para trás do Z-order deste DTVContainer.

Lança:

IllegalArgumentException – se o componente especificado não estiver presente neste DTVContainer

pop

```
public void pop(Component component)
         throws IllegalArgumentException
```

Move o componente especificado um nível acima no Z-order. Isso faz com que o componente diretamente em frente ao componente especificado seja movido um nível abaixo. Se o componente for o único componente neste DTVContainer, a chamada de método não tem efeito. Se o componente já estiver na frente do Z-order, a ordem permanece inalterada e a chamada do método não tem nehum efeito.

Parâmetros:

component - componente a ser movido.

Lança:

IllegalArgumentException – se o componente especificado não estiver presente neste DTVContainer

push

```
public void push(Component component)
          throws IllegalArgumentException
```

Move o componente especificado um nível abaixo no Z-order. Isso faz com que o componente diretamente atrás do componente especificado seja movido um nível acima. Se o componente for o único componente neste DTVContainer, a chamada de método não tem efeito. Se o componente já estiver atrás do Z-order, a ordem permanece inalterada e a chamada do método não tem nehum efeito.

Parâmetros:

component - componente a ser movido.

Lança:

IllegalArgumentException – se o componente especificado não estiver presente neste DTVContainer

popInFrontOf

```
public void popInFrontOf(Component move,
                         Component behind)
                  throws IllegalArgumentException
```

Move o primeiro componente especificado (move) diretamente à frente do segundo componente especificado (behind). Isso faz com que o component behind seja movido um nível abaixo no Z-order.

Parâmetros:

move - componente a ser movido.

behind – o componente que deve ficar posicionado atrás do primeiro componente especificado.

Lança:

IllegalArgumentException – se os componentes especificados não estiverem presentes neste DTVContainer

pushBehind

```
public void pushBehind(Component move,
                       Component front)
                throws IllegalArgumentException
```

Move o primeiro componente especificado (move) diretamente atrás do segundo componente especificado (front).

Parâmetros:

move - componente a ser movido.

front – o componente que deve ficar posicionado à frente do primeiro componente especificado.

Lança:

IllegalArgumentException – se os componentes especificados não estiverem presentes neste DTVContainer

addWindowListener

```
public void addWindowListener(WindowListener listener)
```

Adiciona um *listener* para receber qualquer `java.awt.event.WindowEvents` enviado ao `DTVContainer`. Se o *listener* especificado já tiver sido adicionado, outras chamadas adicionarão mais referências ao *listener*, que então receberá múltiplas cópias do mesmo evento.

Parâmetros:

`listener` - `java.awt.event.WindowListener` a ser notificado sobre qualquer `java.awt.event.WindowEvent`.

Relaciona-se com:

`removeWindowListener(java.awt.event.WindowListener)`

removeWindowListener

```
public void removeWindowListener(WindowListener listener)
```

Remove um *listener* deste `DTVContainer` de forma que não receba mais qualquer `java.awt.event.WindowEvents`. Se o *listener* especificado não for previamente registrado, a chamada para este método não tem efeito. Se múltiplas referências a um único *listener* forem registradas antes, a chamada a este método causará a remoção de exatamente apenas uma referência.

Parâmetros:

`listener` - `java.awt.event.WindowListener` a ser removido

Relaciona-se com:

`addWindowListener(java.awt.event.WindowListener)`

consumeWindowEvent

```
public void consumeWindowEvent(WindowEvent event)
```

Consome um `java.awt.event.WindowEvents` para este `DTVContainer`.

Parâmetros:

`event` - `java.awt.event.WindowEvents` a ser consumido.

getFocusOwnerComponent

```
public Component getFocusOwnerComponent()
```

Obtém o componente pertencente a este `DTVContainer` que atualmente possui foco de entrada.

Retorna:

Componente possuindo o foco, ou `null` se o `DTVContainer` não tiver o foco de entrada.

setToFront

```
public void setToFront()
```

Traz o `DTVContainer` para frente.

Se este DTVContainer não for visível, a chamada a este método não tem efeito.

dispose

```
public void dispose()
```

Descarte deste DTVContainer. Este método deve ser chamado por um aplicativo antes de sair, a fim de liberar recursos utilizados para o DTVContainer.

getDTVContainerPattern

```
public DTVContainerPattern getDTVContainerPattern()
```

Retorna um DTVContainerPattern descrevendo este DTVContainer.

Retorna:

um DTVContainerPattern descrevendo este DTVContainer.

getBestDTVContainerPattern

```
public static DTVContainerPattern getBestDTVContainerPattern(DTVContainerPattern patt
ern)
```

Retorna um DTVContainerPattern mais próximo ao input DTVContainerPattern, ao mesmo tempo especificando um DTVContainer que pôde ser criado com sucesso na plataforma ao mesmo tempo que este método é chamado.

Em seguida, chamadas a este método com o mesmo argumento podem resultar em valores de retorno diferentes. Observar que podem ocorrer conflitos entre propriedades no DTVContainerPattern especificado e um PlaneSetupPattern descrevendo um PlaneSetup atualmente ativo, associados a um plano de gráficos habilitados. Caso esses conflitos não possam ser solucionados, este método falhará e retornará null.

Parâmetros:

pattern - DTVContainerPattern fornecendo as propriedades que o DTVContainer deve satisfazer.

Retorna:

um DTVContainerPattern que melhor corresponde ao argumento de entrada, ou null se ocorre um conflito não-solucionado com um PlaneSetup atualmente ativo

getBestDTVContainer

```
public static DTVContainer getBestDTVContainer(DTVContainerPattern pattern)
```

Retorna um DTVContainer que melhor corresponda ao input DTVContainerPattern, ou null se tal DTVContainer não puder ser encontrado.

Observar que podem ocorrer conflitos entre propriedades no DTVContainerPattern especificado e um PlaneSetupPattern descrevendo um PlaneSetup atualmente ativo, associados a um plano de gráficos habilitados. Caso esses conflitos não possam ser solucionados, este método falhará e retornará null.

Parâmetros:

pattern - DTVContainerPattern especificando as propriedades a serem consideradas

Retorna:

DTVContainer que combina com as propriedades como melhor especificado, ou null se elas não puderem ser satisfeitas, ou se nenhum outro DTVContainer estiver disponível para este aplicativo.

resize

```
public DTVContainerPattern resize(DTVContainerPattern pattern)
                           throws IllegalStateException
```

Redimensiona este DTVContainer de forma que melhor corresponda ao DTVContainerPattern da entrada, ou não faz nada se não for necessário.

Parâmetros:

pattern - DTVContainerPattern que denota o novo tamanho e localização. Claramente apenas opções de tamanho e localização no DTVContainerPattern devem ser consideradas.

Retorna:

um DTVContainerPattern indicando as propriedades DTVContainer (depois de possível redimensionamento).

Lança:

IllegalStateException – se o DTVContainer tiver sido descartado anteriormente.

getDTVContainer

```
public static DTVContainer getDTVContainer(Plane plane)
                                    throws IllegalArgumentException
```

Cria um DTVContainer para o plano especificado. Um plano apropriado para receber um DTVContainer deve ser associado a um objeto Capabilities para o qual uma chamada a pelo menos um dos seguintes métodos retornarão *true*:

Capabilities.isWidgetRenderingSupported()

Capabilities.isGraphicsRenderingSupported()

.

Parâmetros:

plane - plano para o qual o DTVContainer deve ser retornado.

Retorna:

DTVContainer para o plano especificado.

Lança:

IllegalArgumentException – se o plano especificado não for um adequado para receber um DTVContainer (com.sun.dtv.ui.Capabilities)

getDTVContainer

```
public static DTVContainer getDTVContainer(Plane plane,
                                           Container container)
                                    throws IllegalArgumentException
```

Cria um DTVContainer para o plano especificado, que pode ser utilizado como um container Xlet. Para isso, o java.awt.Container, fornecido pelo contexto Xlet, deve ser passado como um segundo parâmetro. Um plano apropriado para receber um DTVContainer deve ser associado a um objeto Capabilities para o qual uma chamada a pelo menos um dos seguintes métodos retornarão *true*:

Capabilities.isWidgetRenderingSupported()

Capabilities.isGraphicsRenderingSupported()

Parâmetros:

plane - plano para o qual o DTVContainer deve ser retornado.

container – o java.awt.Container fornecido chamando o método getContainer() do javax.microedition.xlet.XletContext.

Retorna:

DTVContainer para o plano especificado.

Lança:

IllegalArgumentException – se o plano especificado não for um adequado para receber um DTVContainer (com.sun.dtv.ui.Capabilities)

getCurrentDTVContainer

public static DTVContainer getCurrentDTVContainer()

Retorna o DTVContainer em que o chamador é exibido dentro.

Retorna:

tela atual.

50.13 Classe DTVContainerPattern

50.13.1 Descrição da classe

com.sun.dtv.ui

java.lang.Object

  └com.sun.dtv.ui.DTVContainerPattern

public class DTVContainerPattern

extends Object

O DTVContainerPattern é um meio de descrever as características de um DTVContainer válido. Após exemplificação desta classe um aplicativo pode configurar todos os atributos como desejado, enquanto configurar DTV_CONTAINER_LOCATION e DTV_CONTAINER_DIMENSION é obrigatório. O objeto DTVContainerPattern criado pode então ser utilizado como um argumento para o método DTVContainer.getBestDTVContainer(com.sun.dtv.ui.DTVContainerPattern) a fim de criar um objeto DTVContainer que corresponda ou supere o que foi solicitado no DTVContainerPattern.

50.13.2 Índice de campos

static int DTV_CONTAINER_DIMENSION

Um valor de preferência para uso nos métodos setPreference, getPreferenceValue e getPriority no DTVContainerPattern.

static int DTV_CONTAINER_LOCATION

Um valor de preferência para uso nos métodos setPreference, getPreferenceValue e getPriority no DTVContainerPattern.

static int DTV_CONTAINER_PLANE_SETUP

Um valor de preferência para uso nos métodos setPreference, getPreferenceValue e getPriority no DTVContainerPattern.

50.13.3 Índice de construtores

`DTVContainerPattern()`

50.13.4 Índice de métodos

`Object getPreferenceValue(int preference)`

Retorna o valor de preferência para a preferência especificada.

`int getPriority(int preference)`

Retorna a prioridade de preferência para a preferência especificada.

`void setPreference(int preference, Object value, int priority)`

Configura a preferência indicada para ter o valor especificado com a prioridade especificada.

50.13.5 Detalhe dos campos

DTV_CONTAINER_PLANE_SETUP

`public static final int DTV_CONTAINER_PLANE_SETUP = 16`

Um valor de preferência para uso nos métodos `setPreference`, `getPreferenceValue` e `getPriority` no `DTVContainerPattern`. Isso indica que o `DTVContainer` deve ser criado utilizando o `PlaneSetup` especificado.

DTV_CONTAINER_DIMENSION

`public static final int DTV_CONTAINER_DIMENSION = 17`

Um valor de preferência para uso nos métodos `setPreference`, `getPreferenceValue` e `getPriority` no `DTVContainerPattern`. Isso indica que o `DTVContainer` tem a dimensão preferida em pixels como especificado no objeto `Dimension` fornecido. Configurar este campo de preferência em um `DTVContainerPattern` existente é obrigatório.

DTV_CONTAINER_LOCATION

`public static final int DTV_CONTAINER_LOCATION = 18`

Um valor de preferência para uso nos métodos `setPreference`, `getPreferenceValue` e `getPriority` no `DTVContainerPattern`. Isso indica que o `DTVContainer` tem a localização em pixels preferida como especificado no objeto Point fornecido. Configurar este campo de preferência em um `DTVContainerPattern` existente é obrigatório.

50.13.6 Detalhe dos construtores

DTVContainerPattern

`public DTVContainerPattern()`

50.13.7 Detalhe dos métodos

setPreference

```
public void setPreference(int preference,
                          Object value,
                          int priority)
                   throws IllegalArgumentException
```

Configura a preferência indicada para ter o valor especificado com a prioridade especificada. Uma chamada a este método pode também ser utilizada para substituir valores e prioridades de preferências configuradas anteriormente.

Parâmetros:

`preference` – preferência a ser indicada. Os valores válidos são `DTV_CONTAINER_GRAPHICS_SETUP`, `DTV_CONTAINER_DIMENSION`, e `DTV_CONTAINER_LOCATION`.

`value` – valor a ser configurado para a preferência indicada

`priority` – prioridade da preferência indicada. Os valores válidos são: `REQUIRED`, `PREFERRED`, e `DONT_CARE`

Lança:

`IllegalArgumentException` – se qualquer um dos parâmetros apresentar valor inadequado

getPreferenceValue

```
public Object getPreferenceValue(int preference)
                          throws IllegalArgumentException
```

Retorna o valor de preferência para a preferência especificada.

Parâmetros:

`preference` – preferência para a qual o valor deve ser recuperado. Os valores válidos são `DTV_CONTAINER_PLANE_SETUP`, `DTV_CONTAINER_DIMENSION`, e `DTV_CONTAINER_LOCATION`.

Retorna:

valor válido atualmente para a preferência especificada. Esse deve ser um objeto `PlaneSetup` para `DTV_CONTAINER_PLANE_SETUP`, e um `Dimension` para `DTV_CONTAINER_DIMENSION` ou `DTV_CONTAINER_LOCATION`

Lança:

`IllegalArgumentException` – se o parâmetro apresentar um valor inadequado

getPriority

```
public int getPriority(int preference)
                throws IllegalArgumentException
```

Retorna a prioridade de preferência para a preferência especificada.

Parâmetros:

`preference` – preferência para a qual o valor deve ser recuperado. Os valores válidos são `DTV_CONTAINER_GRAPHICS_SETUP`, `DTV_CONTAINER_DIMENSION`, e `DTV_CONTAINER_LOCATION`.

Retorna:

prioridade válida atualmente para a preferência especificada. Os possíveis valores de retorno são: `REQUIRED`, `PREFERRED`, e `DONT_CARE`

Lança:

`IllegalArgumentException` – se o parâmetro apresentar um valor inadequado

50.14 Classe FontFileException

50.14.1 Descrição da classe

com.sun.dtv.ui

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └com.sun.dtv.ui.FontFileException
```

Todas as interfaces implementadas

`Serializable`

```
public class FontFileException
extends Exception
```

Essa exceção deve ser executará se for realizada uma tentaiva de ler um arquivo de fonte, e esse arquivo não possuir o formato apropriado.

50.14.2 Índice de construtores

`FontFileException()`

Constrói um objeto FontFileException.

`FontFileException(String message)`

Constrói um objeto `FontFileException`.

50.14.3 Detalhe dos construtores

FontFileException

```
public FontFileException()
```

Constrói um objeto `FontFileException`.

FontFileException

```
public FontFileException(String message)
```

Constrói um objeto `FontFileException`.

Parâmetros:

`message` – uma mensagem fornecida à exceção

50.15 Classe FontSpecificationException

50.15.1 Descrição da classe

com.sun.dtv.ui

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └com.sun.dtv.ui.FontSpecificationException
```

Todas as interfaces implementadas

`Serializable`

```
public class FontSpecificationException
extends Exception
```

Essa exceção deve ser executada se for realizada uma tentativa de especificar características de uma fonte a partir de um arquivo de fonte, e essas características não estiverem disponíveis para essa fonte em particular.

50.15.2 Índice de construtores

`FontSpecificationException()`

Constrói um objeto `FontSpecificationException`.

`FontSpecificationException(String message)`

Constrói um objeto `FontSpecificationException`.

50.15.3 Detalhe dos construtores

FontSpecificationException

```
public FontSpecificationException()
```

Constrói um objeto `FontSpecificationException`.

FontSpecificationException

```
public FontSpecificationException(String message)
```

Constrói um objeto `FontSpecificationException`.

Parâmetros:

`message` – uma mensagem fornecida à exceção

50.16 Classe Keyboard

50.16.1 Descrição da classe

com.sun.dtv.ui

```
java.lang.Object
```

```
└com.sun.dtv.ui.UserInputDevice
    └com.sun.dtv.ui.Keyboard
```

Subclasses diretas conhecidas

`RemoteControl`

```
public class Keyboard
extends UserInputDevice
```

Essa classe representa um teclado que pode ser utilizado para controlar uma `Screen` em particular como um `UserInputDevice`.

50.16.2 Índice de construtores

`protected Keyboard()`

Não convém ser possível criar um `Keyboard` para todos.

50.16.3 Índice de métodos

`UserInputEvent getInitiatedEvent(int code)`

Implementa método abstrato `UserInputDevice.getInitiatedEvent()` definido em `UserInputDevice`.

`boolean isSupported(int keyCode)`

Indica se o código de chave especificado é suportado pelo dispositivo de entrada em particular.

50.16.4 Detalhe dos construtores

Keyboard

`protected Keyboard()`

Não convém ser possível criar um `Keyboard` para todos. Como qualquer `UserInputDevice`, ele está ligado a uma tela.

50.16.5 Detalhe dos métodos

getInitiatedEvent

```
public UserInputEvent getInitiatedEvent(int code)
                                 throws IllegalArgumentException
```

Implementa método abstrato `UserInputDevice.getInitiatedEvent()` definido em `UserInputDevice`.

Substituições:

`getInitiatedEvent` na classe `UserInputDevice`

Parâmetros:

`code` – código ao qual o evento de input é iniciado, se inserido

Retorna:

evento de entrada iniciado

Lança:

`IllegalArgumentException` – se o código especificado não for suportado por este `UserInputDevice`

isSupported

`public boolean isSupported(int keyCode)`

Indica se o código de chave especificado é suportado pelo dispositivo de entrada em particular.

Parâmetros:

`keyCode` – código de tecla a ser verificado

Retorna:

true se o código de tecla for suportado pelo dispositivo de entrada particular, caso contrário, *false*

50.17 Interface Matte

50.17.1 Descrição da interface

`com.sun.dtv.ui`

Todas as classes implementadoras conhecidas

`AnimatedMatte, StaticMatte`

`public interface Matte`

Interface básica para todas as classes `Matte`. Caso os pixels do componente já estiverem alocados com um valor alfa, para cada pixel, o valor alfa orignal e aquele adicionado pelo `matte` são multiplicados a fim de obter um valor de resultado determinando o que está finalmente sendo exibido.

Relaciona-se com:

`MatteEnabled`

50.18 Interface MatteEnabled

50.18.1 Descrição da interface

`com.sun.dtv.ui`

Todas as subinterfaces conhecidas

`ViewOnlyComponent`

Todas as classes implementadoras conhecidas

`Button, Calendar, CheckBox, ComboBox, Container, DefaultListCellRenderer, Dialog, DTVContainer, Form, Label, List, RadioButton, TabbedPane`

`public interface MatteEnabled`

O objetivo desta interface é habilitar componentes para combinação de `matte`. Essa interface adiciona a funcionalidade necessária para incorporar a funcionalidade do `matte` em um componente, o próprio `matte` é implementado na classe `Matte` e classes herdando desta.

Relaciona-se com:

`Matte`

50.18.2 Índice de métodos

`Matte getMatte()`

Retorna o `Matte` atualmente associado ao componente implementando essa interface.

`void setMatte(Matte matte)`

Adiciona um `matte` ao componente implementando essa interface para possibilitar composição de `matte`.

50.18.3 Detalhe dos métodos

setMatte

```
void setMatte(Matte matte)
       throws MatteException
```

Adiciona um `Matte` ao componente implementando essa interface para possibilitar composição de `matte`. Se já houver um `Matte` designado para esse componente e esse `Matte` for animado, ele tem de ser parado antes de qualquer chamada para esse método.

Parâmetros:

`matte` - o `Matte` a ser designado ao componente. Todo o tempo, só pode haver um `matte` associado ao componente, devido a isso qualquer `matte` associado anteriormente deve ser sobreposto por uma chamada para esse método. O parâmetro Matte `pode` também ser `null`, nesse caso não há `matte` associado ao componente após a chamada, mesmo se houvesse um antes.

Lança:

`MatteException` - se o `Matte` tem um tipo não-suportado, a plataforma não suporta mattes em nenhuma circunstância ou um `matte` animado está associado ao componente e ainda está rodando

Relaciona-se com:

`getMatte()`

getMatte

`Matte getMatte()`

Retorna o `Matte` atualmente associado ao componente implementando essa interface.

Retorna:

o `Matte` atualmente associado ao componente ou `null` se não houver nenhum

Relaciona-se com:

`setMatte(com.sun.dtv.ui.Matte)`

50.19 Classe MatteException

50.19.1 Descrição da classe

`com.sun.dtv.ui`

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
```

└com.sun.dtv.ui.MatteException

Todas as interfaces implementadas

Serializable

```
public class MatteException
extends Exception
```

Uma MatteException é executada sempre que um aplicativo não conseguir, por alguma razão, realizar uma associação de matte a um elemento gráfico.

Relaciona-se com:

MatteEnabled

50.19.2 Índice de construtores

MatteException()

Constrói um objeto MatteException.

MatteException(String message)

Constrói um objeto MatteException.

50.19.3 Detalhe dos construtores

MatteException

```
public MatteException()
```

Constrói um objeto MatteException.

MatteException

```
public MatteException(String message)
```

Constrói um objeto MatteException.

Parâmetros:

message – uma mensagem fornecida à exceção

50.20 Classe Mouse

50.20.1 Descrição da classe

com.sun.dtv.ui

java.lang.Object

└com.sun.dtv.ui.UserInputDevice

└com.sun.dtv.ui.Mouse

```
public class Mouse
extends UserInputDevice
```

Essa classe representa um mouse que pode ser utilizado para controlar uma Screen em particular como um UserInputDevice.

50.20.2 Índice de construtores

`protected Mouse()`

Não convém ser possível criar um `Mouse` para todos.

50.20.3 Índice de métodos

`UserInputEvent getInitiatedEvent(int code)`

Implementa método abstrato `UserInputDevice.getInitiatedEvent()` definido em `UserInputDevice`.

`boolean isSupported(int mouseCode)`

Indica se o código de mouse especificado é suportado pelo dispositivo de entrada em particular.

50.20.4 Detalhe dos construtores

Mouse

`protected Mouse()`

Não convém ser possível criar um `Mouse` para todos. Como qualquer `UserInputDevice`, ele está ligado a uma tela.

50.20.5 Detalhe dos métodos

getInitiatedEvent

```
public UserInputEvent getInitiatedEvent(int code)
                                 throws IllegalArgumentException
```

Implementa método abstrato `UserInputDevice.getInitiatedEvent()` definido em `UserInputDevice`.

Substituições:

`getInitiatedEvent` na classe `UserInputDevice`

Parâmetros:

`code` – código que o evento de input para o qual é iniciado, se inserido. Tem que ser um dos seguintes: `MouseEvent.MOUSE_LEFT_CLICK`, `MouseEvent.MOUSE_RIGHT_CLICK`, `MouseEvent.MOUSE_LEFT_PRESS`, `MouseEvent.MOUSE_RIGHT_PRESS`, `MouseEvent.MOUSE_LEFT_RELEASE`, `MouseEvent.MOUSE_RIGHT_RELEASE`, `MouseEvent.MOUSE_MOVE`, `MouseEvent.MOUSE_DRAG`, `MouseEvent.MOUSE_ENTER` or `MouseEvent.MOUSE_EXIT`.

Retorna:

evento de entrada iniciado

Lança:

`IllegalArgumentException` – se o código especificado não for suportado por este dispositivo de entrada; sempre executado se o código não for um dos códigos mencionados na descrição do parâmetro `code`.

isSupported

`public boolean isSupported(int mouseCode)`

Indica se o código de mouse especificado é suportado pelo dispositivo de entrada em particular.

Parâmetros:

`mouseCode` – código do mouse a ser verificado. Qualquer um dos códigos a seguir podem ser suportados ou não: `MouseEvent.MOUSE_LEFT_CLICK`, `MouseEvent.MOUSE_RIGHT_CLICK`, `MouseEvent.MOUSE_LEFT_PRESS`, `MouseEvent.MOUSE_RIGHT_PRESS`, `MouseEvent.MOUSE_LEFT_RELEASE`, `MouseEvent.MOUSE_RIGHT_RELEASE`, `MouseEvent.MOUSE_MOVE`, `MouseEvent.MOUSE_DRAG`, `MouseEvent.MOUSE_ENTER` or `MouseEvent.MOUSE_EXIT`. Qualquer outro código não pode ser suportado e sempre gerará *`false`*.

Retorna:

`true` se o código de *mouse* for suportado pelo dispositivo de entrada particular, caso contrário, *`false`*.

50.21 Classe Plane

50.21.1 Descrição da classe

`com.sun.dtv.ui`

```
java.lang.Object
  └com.sun.dtv.ui.Plane
```

Todas as interfaces implementadas

`ScarceResource`

```
abstract public class Plane
extends Object
implements ScarceResource
```

Uma instância de classe `Screen` (tela) representa um sinal de saída de vídeo de um aparelho de TV. Se um aparelho de TV possui mais que um sinal de saída de vídeo independente, deve ser necessária uma instância desta classe para cada sinal. Um único sinal de saída de vídeo é composto de conteúdos de diversos planos suportados pela tela. Qualquer plano contribuinte é uma instância de uma classe derivada da classe abstrata `Plane`.

Uma dada tela pode suportar qualquer número desses objetos até onde diz respeito à API, no entanto, alguma forma de definição do perfil pode limitar isso (por exemplo, há geralmente apenas um plano de fundo).

Cada plano pode ter diversas variantes de configurações em forma de instâncias de `PlaneSetup`. Apenas uma dessas configurações pode ser ativada a qualquer momento. A configuração atualmente válida pode ser determinada para qualquer o utilizando o método `getCurrentSetup`. Presumindo que um aplicativo tenha direitos suficientes para isso, ele também pode modificar a configuração atualmente válida utilizando o método `setCurrentSetup`.

É possível criar uma instância de `PlaneSetupPattern` para preencher critérios para a escolha da melhor configuração definindo uma variedade de preferências cada uma com uma prioridade. A implementação então corresponde com esse padrão de acordo com a variedade de possíveis configurações, tentando encontrar uma que corresponda com ao menos todas as preferências obrigatórias (prioridades `REQUIRED` e `REQUIRED_NOT`). Se isso não for possível, a chamada do método falhará e não retornará nenhuma configuração. Outras prioridades `PREFERRED` e `PREFERRED_NOT` não são obrigatórias, mas devem devem ser respeitadas ao máximo possível.

Relaciona-se com:

`ScarceResource.reserve()`, `ScarceResource.release()`

50.21.2 Índice de construtores

`protected Plane()`

Para uso das classes de plano específicas subclassificando o plano.

50.21.3 Índice de métodos

`void addPlaneSetupListener(PlaneSetupListener listener)`

Adiciona um `PlaneSetupListener` para este plano.

`void addPlaneSetupListener(PlaneSetupListener listener, PlaneSetupPattern pattern)`

Adiciona um `PlaneSetupListener` para este plano.

`static void addResourceTypeListener(ResourceTypeListener listener)`

Adiciona um `ResourceTypeListener` a implementação.

`void addScarceResourceListener(ScarceResourceListener listener)`

Registra um *listener* para eventos sobre mudanças no estado de propriedade deste plano.

`PlaneSetup getBestSetup(PlaneSetupPattern pattern)`

Retorna a melhor configuração possível considerando os critérios definidos neste `PlaneSetupPattern` ou `null`.

`PlaneSetup getBestSetup(PlaneSetupPattern[] patterns)`

Retorna a melhor configuração possível considerando os critérios definidos em um dos objetos `PlaneSetupPattern` dentro do *array* de argumentos, ou `null`.

`Capabilities getCapabilities()`

Retorna o objeto `Capabilities` associado a este plano.

`static Plane getCurrentPlane()`

Retorna o plano em que o chamador é exibido.

`PlaneSetup getCurrentSetup()`

Retorna o `PlaneSetup` atual para este plano.

`PlaneSetup getDefaultSetup()`

Retorna o padrão `PlaneSetup` associado a este plano.

`String getID()`

Retorna a *string* de identificação do plano.

`static Plane[] getInstances()`

Retorna a lista de todas as intâncias de plano existentes, estejam reservadas ou não.

`PlaneSetup[] getSetups()`

Retorna todos os objetos `PlaneSetup` disponíveis associados a este plano.

`boolean isAvailable()`

Verifica se o dado recurso está correntemente disponível para reserva.

`void release()`

Libera esse recurso.

`void removePlaneSetupListener(PlaneSetupListener listener)`

Remove um `PlaneSetupListener` deste plano.

`static void removeResourceTypeListener(ResourceTypeListener listener)`

Remove um *listener* previamente anexado.

`void removeScarceResourceListener(ScarceResourceListener listener)`

Remove um *listener* para eventos sobre mudanças no estado de propriedade deste plano.

`void reserve(boolean force, long timeout, ScarceResourceListener listener)`

Requisita reserva da dada instância de recursos escassos.

`static Plane reserveOne(boolean force, long timeoutms, ScarceResourceListener listener)`

Retorna uma instância reservada fora do *pool* de todas as intâncias `Plane`.

`boolean setPlaneSetup(PlaneSetup setup)`

Estabelece a configuração para este plano.

50.21.4 Detalhe dos construtores

Plane

`protected Plane()`

Para uso das classes de plano específicas subclassificando o plano.

50.21.5 Detalhe dos métodos

getID

`public String getID()`

Retorna a *string* de identificação do plano.

Retorna:

uma *string* de identificação

getSetups

`public PlaneSetup[] getSetups()`

Retorna todos os objetos `PlaneSetup` disponíveis associados a este plano. Alguns desses podem ser válidos para este plano apenas em um certo momento ou para modos em particular do plano.

Retorna:

um *array* dos objetos `PlaneSetup`

Relaciona-se com:

`PlaneSetup`

getDefaultSetup

`public PlaneSetup getDefaultSetup()`

Retorna o padrão `PlaneSetup` associado a este plano. Há apenas uma configuração-padrão, poderia ser, por exemplo, uma configuração mínima.

Retorna:

`PlaneSetup` padrão do plano

Relaciona-se com:

`PlaneSetup`

getBestSetup

`public PlaneSetup getBestSetup(PlaneSetupPattern pattern)`

Retorna a melhor configuração possível considerando os critérios definidos neste `PlaneSetupPattern` ou `null`.

Para detalhes do algorítimo para seleção de uma configuração, Ver `getBestSetup(PlaneSetupPattern[])`

Parâmetros:

`pattern` - - um objeto `PlaneSetupPattern` utilizado para entregar critérios para um `PlaneSetup` válido. Se este parâmetro for `null`, é escolhida a configuração-padrão.

Retorna:

um objeto `PlaneSetup` preenchendo os critérios como especificado no `PlaneSetupPattern`, ou `null` se não houver nenhum.

getBestSetup

`public PlaneSetup getBestSetup(PlaneSetupPattern[] patterns)`

Retorna a melhor configuração possível considerando os critérios definidos em um dos objetos `PlaneSetupPattern` dentro do *array* de argumentos ou `null`. Os objetos `PlaneSetupPattern` são considerados na ordem que aparecem no *array* de parâmetros.

Para detalhes do algorítimo para seleção de uma configuração, Consulte `getBestSetup(PlaneSetupPattern[])`.

Parâmetros:

`patterns` - - *array* `PlaneSetupPattern` utilizado para entregar critérios para um `PlaneSetup` válido.

Retorna:

um objeto `PlaneSetup` que preenche os critérios como especificado em um dos objetos `PlaneSetupPattern` dentro do *array* de parâmetros

getCurrentSetup

`public PlaneSetup getCurrentSetup()`

Retorna o `PlaneSetup` atual para este plano.

Retorna:

`PlaneSetup` atual para este plano.

Relaciona-se com:

`PlaneSetup`

setPlaneSetup

`public boolean setPlaneSetup(PlaneSetup setup)`

```
        throws SecurityException,
               SetupException
```

Estabelece a configuração para este plano.

Como o plano de uma tela é um `ScarceResource`, esse método pode apenas ser chamado por um aplicativo que tenha reservado esse plano anteriormente utilizando os meios apropriados fornecidos pelo mecanismo do `ScarceResource`. A chamada deste método também está sujeita à política de segurança da plataforma, e as regras a seguir decidem sobre o sucesso da chamada:

- Escolher uma configuração inválida para o plano naquele momento ou devido a um modo em particular, o plano é assim uma `SetupException` a ser executada.
- Escolher uma configuração que não seja conflitante com nenhuma configuração de um dos outros planos na mesma tela faz com que ela seja selecionada.
- Escolher uma configuração conflitante com qualquer configuração de um dos outros planos nesta tela, que não está sob controle do aplicativo de chamada em função da política de segurança da plataforma faz com que uma `SecurityException` seja executada.
- Escolher uma configuração conflitante com qualquer configuração de um dos outros planos nesta tela, não estando sob controle do aplicativo de chamada em função de outro aplicativo possuir o direito de controle do outro plano e a plataforma não conceder tal direito a este aplicativo faz com que uma `SecurityException` seja executada.
- Escolher uma configuração conflitante com qualquer configuração de um dos outros planos nesta tela estando sob controle do aplicativo de chamada em função de este ou nenhum aplicativo não ter reservado o plano, a configuração do outro plano muda automaticamente (sem qualquer efeito para a propriedade do recurso do outro plano).

Se a configuração de outro plano tiver sido alterada como uma consequência da chamada deste método, a mudança deve ser refletida chamando um método apropriado da API do plano em questão.

Os aplicativos podem impedir ou limitar mudanças às configurações de outros planos, não programados, utilizando as constantes `PlaneSetupPattern.NO_GRAPHICS_IMPACT`, `PlaneSetupPattern.NO_SUBTITLE_IMPACT`, `PlaneSetupPattern.NO_VIDEO_IMPACT`, `PlaneSetupPattern.NO_STILLVIDEO_IMPACT` e `PlaneSetupPattern.NO_BACKGROUND_IMPACT` em seus padrões de configuração.

Se a alteração da configuração especificada foi bem-sucedida, o plano deve disparar pelo menos um `PlaneSetupEvents` para todos os *listeners* registrados atualmente.

Parâmetros:

`setup` - `PlaneSetup` ao qual esse plano deve ser configurado.

Retorna:

Um valor booleano indicando se a configuração foi alterada com sucesso. Observar que caso o valor de retorno seja *`false`*, não há garantia de que a configuração do plano seja a mesma de antes da chamada.

Lança:

`SecurityException` – se o aplicativo atualmente não possui direitos suficientes para mudar a configuração para este plano.

`SetupException` – se a configuração não é válida para este plano ou é conflitante com as configurações dos outros planos, e o conflito não pode ser solucionado.

addPlaneSetupListener

```
public void addPlaneSetupListener(PlaneSetupListener listener)
```

Adiciona um `PlaneSetupListener` para este plano. Este *listener* deve ser notificado sempre que a configuração do plano for modificada. Se um *listener* já tiver sido adicionado antes, as chamadas a este método adicionarão mais referências ao mesmo *listener*. Consequentemente essas referências receberão então múltiplas cópias de um único evento.

Parâmetros:

`listener` - `PlaneSetupListener` a ser adicionado a este plano.

Relaciona-se com:

`removePlaneSetupListener(com.sun.dtv.ui.event.PlaneSetupListener)`

addPlaneSetupListener

```
public void addPlaneSetupListener(PlaneSetupListener listener,
                                  PlaneSetupPattern pattern)
```

Adiciona um `PlaneSetupListener` para este plano. Este *listener* deve ser notificado sempre que uma configuração do plano for modificada de forma que não seja mais compatível com o `PlaneSetupPattern` especificado. Se um *listener* já tiver sido adicionado antes, as chamadas a este método adicionarão mais referências ao mesmo *listener*. Consequentemente essas referências receberão então múltiplas cópias de um único evento.

Observar que se a configuração do plano não corresponde ao padrão especificado, o *listener* deve ser adicionado e um `PlaneSetupEvent` imediatamente gerado para o `PlaneSetupListener` especificado.

Parâmetros:

`listener` - `PlaneSetupListener` a ser adicionado a este plano.

`pattern` - `PlaneSetupPattern` a ser utilizado para determinar a compatibilidade com a configuração do plano.

Relaciona-se com:

`removePlaneSetupListener(com.sun.dtv.ui.event.PlaneSetupListener)`

removePlaneSetupListener

```
public void removePlaneSetupListener(PlaneSetupListener listener)
```

Remove um `PlaneSetupListener` deste plano. Se o *listener* especificado não for registrado antes, o método não tem efeito. Se múltiplas referências a um único *listener* forem registradas, qualquer chamada a este método removerá apenas uma referência.

Parâmetros:

`listener` - `PlaneSetupListener` a ser removido deste plano.

Relaciona-se com:

`addPlaneSetupListener(com.sun.dtv.ui.event.PlaneSetupListener),`
`addPlaneSetupListener(com.sun.dtv.ui.event.PlaneSetupListener,`
`com.sun.dtv.ui.PlaneSetupPattern)`

addScarceResourceListener

```
public void addScarceResourceListener(ScarceResourceListener listener)
```

Registra um *listener* para eventos sobre mudanças no estado de propriedade deste plano. Se este *listener* for adicionado antes, as chamadas adicionarão mais referências ao *listener*. Consequentemente essas referências receberão múltiplas cópias de um único evento.

Parâmetros:

`listener` – objeto a ser informado sobre mudanças do estado

Relaciona-se com:

`removeScarceResourceListener(com.sun.dtv.resources.ScarceResourceListener)`

removeScarceResourceListener

```
public void removeScarceResourceListener(ScarceResourceListener listener)
```

Remove um *listener* para eventos sobre mudanças no estado de propriedade deste plano. Este método não tem efeito se o *listener* especificado não tiver sido registrado antes.

Parâmetros:

`listener` – objeto que não está mais interessado

Relaciona-se com:

```
addScarceResourceListener(com.sun.dtv.resources.ScarceResourceListener)
```

reserve

```
public void reserve(boolean force,
                    long timeout,
                    ScarceResourceListener listener)
             throws IllegalArgumentException,
                    TimeoutException
```

Requisita reserva da dada instância de recursos escassos. O método bloquear-se-á até que a instância esteja disponível. O método retorna normalmente apenas se a reserva for bem sucedida. Todos os outros casos são manuseados por exceções.

Primeiro, se houver um gerenciador de segurança, seu método `checkPermission` é chamado com a permissão `ScarceResourcePermission(name, "reserve")`. Se `force` é além disso definido *`true`*, a permissão também verificado no `ScarceResourcePermission(name, "force")`.

Durante o processo de reserva, se a instância do recurso já estiver alocada, a implementação deve notificar o proprietário atual do recurso sobre a requisição de reserva por meio da interface `ScarceResourceListener`:

- ou por `ScarceResourceListener.releaseRequested()` se forçar for *`false`*,
- ou por `ScarceResourceListener.releaseForced()` no outro caso.

O *listener* deve ser usado para tais notificações apenas até o recurso ser liberado. Após a liberação, a implementação não deve chamar quaisquer dos métodos da interface do *listener*.

Após esse primeiro evento, a implementação procederá de acordo e liberará (ou não) o recurso requisitado. No caso da implementação liberar o recurso, desencadeará um evento `ScarceResourceListener.released()` ao proprietário *listener* original do recurso informando-o que o recurso não lhe pertence mais.

O aplicativo pode controlar o tempo de espera para que tal recurso esteja disponível ao definir `timeoutms`. No caso da duração da reserva exceder o tempo expresso em `timeoutms`, um `TimeoutException` é lançado.

Sob operação normal, recursos são liberados usando o método `release`. De qualquer forma, no caso do aplicativo não liberar recursos quando requisitado ou o aplicativo ser terminado, a implementação deve liberar todos os recursos alocados para o aplicativo para permitir que outros aplicativos sejam notificados de mudanças na alocação de recursos e poderem reservá-los. Ver a seção Resource Cleanup do ciclo de vida do aplicativo.

Especificado por:

`reserve` na interface `ScarceResource`

Parâmetros:

`force` - Se *`true`*, esse método retirará o recurso do proprietário atual. Se *`false`*, a implementação bloquear-se-á e esperará até que o recurso esteja disponível (usando `release()`) ou até `timeoutms`.

`listener` - O *listener* a ser anexado para receber notificação sobre o estado do recurso.

Lança:

`IllegalArgumentException` - Se o *listener* estiver `null` ou se o valor especificado em `timeoutms` não for válido isto é, nem um inteiro positivo, nem -1.

`TimeoutException` - Se o tempo especificado em `timeoutms` houver acabado e o recurso não houver podido ser reservado.

reserveOne

```
public static Plane reserveOne(boolean force,
                               long timeoutms,
                               ScarceResourceListener listener)
                        throws IllegalArgumentException,
                               TimeoutException
```

Retorna uma instância reservada fora do pool de todas as intâncias `Plane`. Esse método retorna com a instância ou executa uma exceção de acordo com a situação.

Esse método se comporta exatamente como o método `reserve()`.

Parâmetros:

`force` - Se *`true`*, esse método tem permissão de retirar qualquer recurso do atual proprietário. Se *`false`*, a implementação bloqueará e esperará até que um recurso de um dado `type` esteja disponível (utilizando `release()`) ou até milissegundos `timeoutms`.

`timeoutms` – Uma quantidade positiva de tempo em milissegundos até a qual este método esperará para que qualquer recurso seja liberado por seu proprietário atual. O valor `-1` indica que a implementação esperará para sempre.

`listener` - O *listener* a ser anexado para receber notificação sobre o status do recurso.

Retorna:

A instância de tipo `Plane` que foi reservada.

Lança:

`IllegalArgumentException` - Se o *listener* estiver `null` ou se o valor especificado em `timeoutms` não for válido, isto é, nem um inteiro positivo, nem -1.

`TimeoutException` - Se o tempo especificado em `timeoutms` houver acabado e o recurso não houver podido ser reservado.

Relaciona-se com:

`reserve()`

release

```
public void release()
```

Libera esse recurso.

O recurso deve ser disponibilizado para outro aplicativo através da plataforma. Após chamar esse método, não é mais possível interagir com o dado recurso: chamadas para métodos críticos desses recursos escassos devem ser ignorados e lançar `IllegalStateException`. Essa afirmação é válida e é o comportamento requerido de qualquer classe implementando a interface `ScarceResource`. Para interagir novamente com o dado recurso, o aplicativo deve chamar o método `reserve()` e se tornar o proprietário novamente.

A implementação pode organizar quaisquer recursos do sistema que esse objeto estiver usando. Depois a implementação não deve chamar quaisquer dos métodos do *listener* que estava anexado no momento da reserva.

Se o recurso já estava disponível (isto é, não reservado), esse método não faz nada.

Especificado por:

`release` na interface `ScarceResource`

isAvailable

```
public boolean isAvailable()
```

Verifica se o dado recurso está correntemente disponível para reserva. O valor retornado dá a situação atual e não garante que o recurso ainda estará disponível em um momento posterior, por exemplo, no momento da reserva: outro aplicativo pode ter tomado aquele recurso no tempo decorrido.

Especificado por:

`isAvailable` na interface `ScarceResource`

Retorna:

Um boolean definido *`true`* se o dado recurso estiver atualmente disponível para reserva.

getCapabilities

```
public Capabilities getCapabilities()
```

Retorna o objeto `Capabilities` associado a este plano.

Retorna:

objeto `Capabilities` associado a este plano.

getCurrentPlane

```
public static Plane getCurrentPlane()
```

Retorna o plano em que o chamador é exibido.

Retorna:

plano atual

getInstances

```
public static Plane[] getInstances()
```

Retorna a lista de todas as intâncias de plano existentes, estejam reservadas ou não.

Retorna:

lista de todas as instâncias de plano

addResourceTypeListener

```
public static void addResourceTypeListener(ResourceTypeListener listener)
                                    throws NullPointerException
```

Adiciona um `ResourceTypeListener` a implementação. Sempre que um `reserve()` ou um `release()` for chamado para quaisquer recursos do mesmo tipo, a implementação chamará os métodos correspondentes do *listener*.

Parâmetros:

`listener` - O *listener* desencadeado para eventos em recursos do mesmo tipo.

Lança:

`NullPointerException` - Se o *listener* estiver `null`.

Relaciona-se com:

`removeResourceTypeListener()`

removeResourceTypeListener

```
public static void removeResourceTypeListener(ResourceTypeListener listener)
                                        throws NullPointerException
```

Remove um *listener* previamente anexado. Se o *listener* não foi anexado antes, esse método não faz nada..

Parâmetros:

`listener` - O *listener* desencadeado para eventos em recursos do mesmo tipo.

Lança:

`NullPointerException` - Se o *listener* estiver `null`.

Relaciona-se com:

`addResourceTypeListener()`

50.22 Classe PlaneSetup

50.22.1 Descrição da classe

`com.sun.dtv.ui`

```
java.lang.Object
  └com.sun.dtv.ui.PlaneSetup
```

```
public class PlaneSetup
extends Object
```

A classe `PlaneSetup` é capaz de descrever as características de um plano. Um plano pode ter muitos objetos `PlaneSetup` associados a ele, mas sempre há apenas um válido para um certo momento.

Relaciona-se com:

`Plane`

50.22.2 Índice de construtores

`PlaneSetup``()`

50.22.3 Índice de métodos

`Image adaptImage(Image input, boolean dither)`

Adapta uma dada imagem, se necessário, para ganhar compatibilidade com o `PlaneSetup` válido atualmente.

`Point convertPoint(PlaneSetup destination, Point source)`

Este método foi criado para evitar erros redundantes durante a conversão das posições de pixel de um sistema de coordenadas para outro ao passar por coordenadas normalizadas.

`void displayImage(Image image)`

Mostrar uma imagem (plano de fundo) no plano associado.

`void displayImage(Image image, Rectangle rectangle)`

Mostra uma imagem (plano de fundo) para cobrir uma área em particular do plano associado.

`Color getColor()`

Fornece a cor do plano de fundo válida atualmente deste plano, se houver.

`Dimension getOffset(PlaneSetup setup)`

Retorna o deslocamento entre a origem do espaço de coordenada do pixel do `PlaneSetup` especificado, e a origem espaço de coordenada do pixel atual deste `PlaneSetup`.

`PlaneSetupPattern getPattern()`

Retorna o objeto `PlaneSetupPattern` que descreve e identifica este `PlaneSetup`.

`Dimension getPixelAspectRatio()`

Retorna a razão de aspecto de pixel desta configuração.

`Dimension getPixelResolution()`

Retorna a resolução desta configuração em pixels.

`Plane getPlane()`

Retorna o plano associado a este `PlaneSetup`.

`Dimension getPlaneAspectRatio()`

Retorna a razão de aspecto desta configuração desde que conhecida.

`Rectangle getScreenArea()`

Retorna a posição e tamanho do plano associado a esta configuração na tela em coordenadas de tela.

`Dimension getScreenResolution()`

Retorna a resolução atual da tela desta configuração

`Object getVideoController()`

Fornece o objeto de controle de apresentação para o vídeo exibido atualmente pelo plano associado a esta configuração.

`Object getVideoSource()`

Fornece a fonte do vídeo sendo apresentado pelo plano associado a este `PlaneSetup` no momento.

`boolean isDirectColorModelSupported()`

Indica se o modelo *Direct Color* é suportado por esta configuração.

`boolean isFlickeringFilterAvailable()`

Retorna se esta configuração inclui filtragem para redução de flicker integrado.

`boolean isGraphicsMixingSupported()`

Indica se a configuração do plano suporta a exibição de gráficos além de streams de vídeo.

`boolean isImageScalingSupported()`

Indica se o redomensionamento de imagem rápido é suportado por esta configuração.

`boolean isIndexedColorModelSupported()`

Indica se uma *Color Lookup Table* é suportada por esta configuração.

`boolean isInterlaced()`

Retorna se esta configuração esta entrelaçada.

`boolean isMatteSupported()`

Indica se o recurso de mattes é suportado por esta configuração.

`boolean isVideoMixingSupported()`

Indica se a configuração suporta transparência no sistema de gráficos de forma que o output de um decodificador de vídeo fundamental esteja visível.

```
void setColor(Color color)
```

Configura a cor válida para o plano de fundo do plano associado a esta configuração para um novo valor.

50.22.4 Detalhe dos construtores

PlaneSetup

```
public PlaneSetup()
```

50.22.5 Detalhe dos métodos

convertPoint

```
public Point convertPoint(PlaneSetup destination,
                          Point source)
```

Este método foi criado para evitar erros redundantes durante a conversão das posições de pixel de um sistema de coordenadas para outro ao passar por coordenadas normalizadas. Como uma conversão direta não é sempre possível, o valor de retorno pode ser `null`. As razões para uma conversão falha podem ser:

- pelo menos uma das duas `PlaneSetups` não é baseada em pixel
- um sistema de coordenadas é transformado em outro utilizando uma função não-linear
- a função de transformação por alguma razão é desconhecida
- a função de transformação não é constante com o tempo
- A posição da origem é interpretada no sistema de coordenadas do objeto `PlaneSetup` na qual este método é chamado.

Parâmetros:

`destination` – `PlaneSetup` de destino.

`source` – posição de pixel neste `PlaneSetup`.

Retorna:

posição da posição de pixel especificada medida no sistema de coordenadas de destino, ou `null` se a conversão falha.

isFlickeringFilterAvailable

```
public boolean isFlickeringFilterAvailable()
```

Retorna se esta configuração inclui filtragem para redução de flicker integrado.

Retorna:

true se a filtragem estiver inclusa, caso contrário, *false*.

isInterlaced

```
public boolean isInterlaced()
```

Retorna se esta configuração esta entrelaçada.

Retorna:

true se esta configuração estiver entrelaçada, caso contrário, *false*.

getOffset

`public Dimension getOffset(PlaneSetup setup)`

Retorna o deslocamento entre a origem do espaço de coordenada do pixel do `PlaneSetup` especificado, e a origem espaço de coordenada do pixel atual deste `PlaneSetup`. O deslocamento é retornado no espaço de coordenadas de pixel do `PlaneSetup` para o qual este método é chamado.

Parâmetros:

`setup` - `PlaneSetup` alvo

Retorna:

o deslocamento entre o espaço de coordenadas de pixel do `PlaneSetup` especificado e o espaço de coordenadas de pixel atual do `PlaneSetup` de origem. Se o deslocamento não pode ser determinado, é retornado `null`.

getScreenResolution

`public Dimension getScreenResolution()`

Retorna a resolução de tela atual desta configuração, ou seja 960 x 540, 1920 x 1080, etc.

Retorna:

um objeto `Dimension` epecificando a resolução da tela desta configuração

getPlaneAspectRatio

`public Dimension getPlaneAspectRatio()`

Retorna a razão de aspecto da configuração desde que conhecida, ou seja, 4:3, 16:9, etc.

Esta dimensão pode ser utilizada para determinar a razão de aspecto de pixel para determinados `PlaneSetups`.

Retorna:

um objeto `Dimension` especificando a resolução da tela da configuração

getPixelAspectRatio

`public Dimension getPixelAspectRatio()`

Retorna a razão de aspecto de pixel desta configuração. Some examples are {16:15}, {64:45}, {1:1}.

Retorna:

razão de aspecto dos pixels nesta configuração.

getPixelResolution

`public Dimension getPixelResolution()`

Retorna a resolução desta configuração em pixels. O sistema de coordenadas de pixels utilizado é aquele do plano associado a esta configuração.

Retorna:

a resolução desta configuração em pixels.

getScreenArea

`public Rectangle getScreenArea()`

Retorna a posição e tamanho do plano associado a esta configuração na tela em coordenadas de tela.

Retorna:

a área na tela desta configuração nas coordenadas de tela.

isVideoMixingSupported

```
public boolean isVideoMixingSupported()
```

Indica se a configuração suporta transparência no sistema de gráficos de forma que o output de um decodificador de vídeo fundamental esteja visível. (Isso inclui também imagens paradas de vídeos.) O valor de retorno só pode ser *true* se o valor de retorno do `Capabilities.isImageRenderingSupported()`, chamado para a instância de `Capabilities` associada ao plano que é associado atualmente a esta configuração, também for *true*. As configurações a seguir preenchem este requisito:

Configura com uma transformação definida entre pixels de vídeo / vídeo parado e pixels de gráficos (por exemplo, pixels são do mesmo tamanho).

Configura onde um aplicativo exibe gráficos de vídeo / vídeo parado, enquanto o vídeo / vídeo parado é utilizado como um plano de fundo. Portanto, não há transformação exigida entre os dois conjuntos de pixels em tais tipos de configurações.

Os aplicativos podem especificar um vídeo / vídeo parado em particular com o qual a combinação deve ser suportada. Nesse caso, a configuração de vídeo / vídeo parado é especificada como uma configuração de vídeo ou vídeo parado, respectivamente. Nesses casos em que nenhuma configuração de vídeo específica é exigida, não é necessário especificar tal configuração e em vez disso pode-se utilizar `null`.

Retorna:

true se a combinação de vídeo for suportada, caso contrário, *false*

isGraphicsMixingSupported

```
public boolean isGraphicsMixingSupported()
```

Indica se a configuração do plano suporta a exibição de gráficos além de streams de vídeo. Isso é verdadeiro quando a tela inclui planos associados a ambos os tipos de configurações em que os pixels de vídeo e de gráfico estão totalmente alinhados (mesmo tamanho), assim como configurações em que são exibidas juntas, mas onde há uma relação mais complexa entre os dois espaços de coordenadas de pixel. O valor de retorno só pode ser *true* se o valor de retorno do `Capabilities.isVideoRenderingSupported()`, chamado para a instância de `Capabilities` associada ao plano que é associado atualmente a esta configuração, também for *true*.

Retorna:

true se a combinação de gráficos for suportada, caso contrário, *false*

isMatteSupported

```
public boolean isMatteSupported()
```

Indica se o recurso de mattes é suportado por esta configuração. O valor de retorno só pode ser *true* se o valor de retorno do `Capabilities.isImageRenderingSupported()`, chamado para a instância de `Capabilities` associada ao plano que é associado atualmente a esta configuração, também for *true*.

Retorna:

true se mattes forem suportados, caso contrário, *false*

isImageScalingSupported

```
public boolean isImageScalingSupported()
```

Indica se o redomensionamento de imagem rápido é suportado por esta configuração. O valor de retorno só pode ser *true* se o valor de retorno do `Capabilities.isImageRenderingSupported()`, chamado para a instância de `Capabilities` associada ao plano que é associado atualmente a esta configuração, também for *true*.

Retorna:

true se o redimensionamento de imagem rápido for suportado, caso contrário, *false*

isIndexedColorModelSupported

```
public boolean isIndexedColorModelSupported()
```

Indica se uma *Color Lookup Table* é suportada por esta configuração. O valor de retorno só pode ser *true* se o valor de retorno de no mínimo um dos seguintes métodos, chamados para a instância de `Capabilities` associada ao plano que é associado atualmente a esta configuração, também for *true*.

`Capabilities.isImageRenderingSupported()`

`Capabilities.isWidgetRenderingSupported()`

`Capabilities.isGraphicsRenderingSupported()`

.

Retorna:

true se o modelo de cor indicado for suportado, caso contrário, *false*

isDirectColorModelSupported

```
public boolean isDirectColorModelSupported()
```

Indica se o modelo Direct Color é suportado por esta configuração. O valor de retorno só pode ser *true* se o valor de retorno de no mínimo um dos seguintes métodos, chamados para a instância de `Capabilities` associada ao plano que é associado atualmente a esta configuração, também for *true*.

`Capabilities.isImageRenderingSupported()`

`Capabilities.isWidgetRenderingSupported()`

`Capabilities.isGraphicsRenderingSupported()`

.

Retorna:

true se o modelo de cor direta for suportado, caso contrário, *false*

getVideoSource

```
public Object getVideoSource()
                     throws SecurityException,
                            SetupException
```

Fornece a fonte do vídeo sendo apresentado pelo plano associado a este `PlaneSetup` no momento. A especificação de classe a ser retornada não está incluída no escopo deste API. Se nenhum vídeo estiver sendo apresentado, é retornado `null`. A chamada a este método só faz sentido se a rendereização de vídeo for suportada pelo plano associado a esta configuração, ou seja, se o valor de retorno de

`Capabilities.isVideoRenderingSupported()` for *`true`*. Caso contrário é executada uma exceção.

Retorna:

uma referência à origem do vídeo

Lança:

`SecurityException` – se o aplicativo atualmente não possui direitos suficientes para obter o objeto `VideoSource`

`SetupException` – se o plano associado não suporta renderização de vídeo

getVideoController

```
public Object getVideoController()
                          throws SecurityException,
                                 SetupException
```

Fornece o objeto de controle de apresentação para o vídeo exibido atualmente pelo plano associado a esta configuração. Se nenhum vídeo estiver sendo apresentado, é retornado `null`. Um possível objeto de retorno é, por exemplo, uma referência em uma instância a `javax.media.Player` que possui o recurso. A chamada a este método só faz sentido se a rendereização de vídeo for suportada pelo plano associado a esta configuração, ou seja, se o valor de retorno de `Capabilities.isVideoRenderingSupported()` for *`true`*. Caso contrário é executada uma exceção.

Retorna:

objeto que controla a apresentação de vídeo

Lança:

`SecurityException` – se o aplicativo não possui direitos suficientes para obter o objeto `VideoPlayer`.

`SetupException` – se o plano associado não suporta renderização de vídeo

getPattern

```
public PlaneSetupPattern getPattern()
```

Retorna o objeto `PlaneSetupPattern` que descreve e identifica este `PlaneSetup`. Consequentemente, por programação

```
Plane.getBestSetup(PlaneSetup.getPattern())
```

deve obter o `PlaneSetup` original.

Os recursos implementados no `PlaneSetup` devem ser então representados no padrão resultante com prioridade `PlaneSetupPattern.REQUIRED`. Os recursos não-implementados representados com prioridade `PlaneSetupPattern.REQUIRED_NOT`. As preferências não configuradas pela plataforma aparecerão com prioridade `PlaneSetupPattern.DONT_CARE`.

Retorna:

objeto `PlaneSetupPattern` que descreve e identifica este `PlaneSetup`.

getPlane

```
public Plane getPlane()
```

Retorna o plano associado a este `PlaneSetup`.

Retorna:

objeto `Plane` associado a este `PlaneSetup`.

getColor

```
public Color getColor()
```

Fornece a cor do plano de fundo válida atualmente deste plano, se houver. Para chamar este método não é necessário reservar o plano anteriomente. Não se garante que o valor retornado seja o valor configurado pela última chamada ao `setColor` já que as plataformas podem oferecer um espaço de cor diferente para planos de fundo e o valor realmente utilizado deve ser retornado. Se nenhuma cor de plano de fundo for configurada para esta configuração deve ser retornado `null`.

Retorna:

Cor atual desta configuração

Relaciona-se com:

`setColor(java.awt.Color)`

setColor

```
public void setColor(Color color)
              throws SecurityException,
                     SetupException
```

Configura a cor válida para o plano de fundo do plano associado a esta configuração para um novo valor. Passar uma cor não-opaca é ilegal. Se a resolução de cor disponível para planos de fundo não for suficiente para o dado valor, a plataforma pode aproximá-la para o valor mais próximo disponível. O método `getColor` retornará o valor realmente em uso.

Parâmetros:

`color` – cor a ser utilizada para o plano de fundo

Lança:

`SecurityException` – se o aplicativo não tiver atualmente reservado o plano associado a este `PlaneSetup` ou se este não for o `PlaneSetup` atualmente válido do plano reservado.

`SetupException` – se a cor especificada for ilegal para esta plataforma.

Relaciona-se com:

`getColor()`

displayImage

```
public void displayImage(Image image)
                  throws IOException,
                         IllegalArgumentException,
                         SecurityException,
                         NullPointerException,
                         SetupException
```

Mostrar uma imagem (plano de fundo) no plano associado. Este método bloqueará enquanto os dados de imagem são carregados, se ainda não estiver concluído. Dependendo da plataforma, a imagem

- é redimensionada para caber,
- recortada (quando for muito grande) ou
- repetida (quando for muito pequena).

A posição da imagem também é dependente da plataforma. Se a imagem não for redimensionada para caber, a cor anterior configurada utilizando `setColor` pode ficar visível na área restante do plano.

A imagem deve ser removida se o método `setColor` for chamado em seguida.

Se uma chamada para o `Capabilities.isImageRenderingSupported()` retorna *`false`* para o plano associado a esta configuração, uma chamada a este método fará com que uma `SetupException` seja executada.

Parâmetros:

`image` – imagem a ser exibida.

Lança:

`IOException` – se os dados para a `com.sun.dtv.lwuit.Image` não puderem ser carregados.

`IllegalArgumentException` – se a `com.sun.dtv.lwuit.Image` não contém uma imagem em um formato de codificação de imagem suportado

`SecurityException` – se o plano associado a este `PlaneSetup` não for reservado pelo aplicativo de chamada

`NullPointerException` – se a imagem do argumento for `null`

`SetupException` - se o `PlaneSetup` não estiver configurado para seu plano, ou se não for adequado para imagens de renderização.

displayImage

```
public void displayImage(Image image,
                         Rectangle rectangle)
                  throws IOException,
                         IllegalArgumentException,
                         SecurityException,
                         NullPointerException,
                         SetupException
```

Mostra uma imagem (plano de fundo) para cobrir uma área em particular do plano associado. Este método bloqueará enquanto os dados de imagem são carregados, se ainda não estiver concluído. Dependendo da plataforma, a imagem

- é redimensionada para caber,
- recortada (quando for muito grande) ou
- repetida (quando for muito pequena).

A posição da imagem dentro do retângulo também depende da plataforma. Se a imagem não for redimensionada para caber no retângulo, ou se o retângulo não cobrir toda a área do plano, a cor configurada anteriormente utilizando o `setColor` pode ficar visível na área restante do plano.

A imagem deve ser removida se o método `setColor` for chamado em seguida.

Se uma chamada para o `Capabilities.isImageRenderingSupported()` retorna *`false`* para o plano associado a esta configuração, uma chamada a este método fará com que uma `SetupException` seja executada.

Parâmetros:

`image` – imagem a ser exibida

`rectangle` – área do plano em que a imagem deve ser exibida

Lança:

`IOException` – se os dados para a `com.sun.dtv.lwuit.Image` não puderem ser carregados.

`IllegalArgumentException` – se a `com.sun.dtv.lwuit.Image` não contém uma imagem em um formato de codificação de imagem suportado

`SecurityException` – se o plano associado a este `PlaneSetup` não for reservado pelo aplicativo de chamada

`NullPointerException` – se nenhum ou ambos argumentos forem `null`

`SetupException` - se o `PlaneSetup` não estiver configurado para seu plano, ou se não for adequado para imagens de renderização.

adaptImage

```
public Image adaptImage(Image input,
                        boolean dither)
                 throws SetupException
```

Adapta uma dada imagem, se necessário, para ganhar compatibilidade com o `PlaneSetup` válido atualmente. Se o tipo de imagem de input permite, isso pode envolver a mistura de cores para uma paleta de cores restrita. Se não for necessária nenhuma adaptação, a imagem original retornará se uma chamada para o `Capabilities.isImageRenderingSupported()` retorna *`false`* para o plano associado a esta configuração, uma chamada a este método fará com que uma `SetupException` seja executada.

Parâmetros:

`input` – imagem a ser modificada, se necessário

`dither` - um *boolean* que indica se a imagem de entrada permite ou não a mistura de cores

Retorna:

uma imagem adaptada de forma a permitir apresentação ideal no plano associado a este `PlaneSetup`, ou a imagem original se não houver necessidade de adaptação.

Lança:

`SetupException` – se o `PlaneSetup` não for adequado para imagens de renderização.

50.23 Classe PlaneSetupPattern

50.23.1 Descrição da classe

`com.sun.dtv.ui`

```
java.lang.Object
  └com.sun.dtv.ui.PlaneSetupPattern
```

```
public class PlaneSetupPattern
extends Object
```

Essa classe é um meio de descrever a configuração de um plano de tela especificando várias propriedades e sua importância para o aplicativo. É utilizada para solicitar uma instância válida de acordo com a configuração para a descrição fornecida. Subclasses desta classe definem propriedades adicionais. Essas classes podem ser subclassificadas a fim de adicionar mais propriedades, se necessário.

Em caso de preferências conflitantes entre `PlaneSetupPatterns`, aplicam-se as seguintes regras:

Caso todas as preferências conflitantes apresentem prioridade `PREFERRED`, `PREFERRED_NOT` ou `DONT_CARE`, as preferências conflitantes devem ser descartadas do padrão.

Caso todas as preferências conflitantes apresentem prioridade `REQUIRED` ou `REQUIRED_NOT`, a chamada do método falhará.

Em todos os outros casos, devem ser descartadas quantas preferências com prioridade `PREFERRED`, `PREFERRED_NOT` or `DONT_CARE` forem necessárias. Se ainda houver conflito depois disso, o método falhará.

Diversas preferências nesta classe não podem ser preenchidas nos padrões gerados pela plataforma.

50.23.2 Índice de campos

`static int CHANGEABLE_SINGLE_COLOR`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int DIRECT_COLOR_SUPPORT`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int DONT_CARE`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int FLICKER_FILTERING`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int GRAPHICS_MIXING`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int IMAGE_SCALING_SUPPORT`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int INDEXED_COLOR_SUPPORT`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int INTERLACED_DISPLAY`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int MATTE_SUPPORT`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int NO_BACKGROUND_IMPACT`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int NO_GRAPHICS_IMPACT`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int NO_STILLVIDEO_IMPACT`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int NO_SUBTITLE_IMPACT`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int NO_VIDEO_IMPACT`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int PIXEL_ASPECT_RATIO`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int PIXEL_RESOLUTION`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int PREFERRED`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int PREFERRED_NOT`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int REQUIRED`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int REQUIRED_NOT`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int STILL_IMAGE`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority`() no `PlaneSetupPattern`.

`static int VIDEO_GRAPHICS_PIXEL_ALIGNED`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

`static int VIDEO_MIXING`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`.

50.23.3 Índice de construtores

`PlaneSetupPattern()`

50.23.4 Índice de métodos

`Object getPreferenceValue(int preference)`

Retorna o valor de preferência para a preferência especificada.

`int getPriority(int preference)`

Retorna a prioridade para a preferência especificada.

`boolean isSetupFitting(PlaneSetup setup)`

Informa se o `PlaneSetup` especificado pode ser utilizado ou não para criar um plano que suporta os recursos configurados neste padrão.

`void setPreference(int preference, int priority)`

Configura a preferência indicada para ter a prioridade especificada.

`void setPreference(int preference, Object value, int priority)`

Configura a preferência indicada para ter o valor especificado com a prioridade especificada.

50.23.5 Detalhe dos campos

REQUIRED

`public static final int REQUIRED = 1`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que este recurso é necessário no `PlaneSetup`. Se este recurso não estiver disponível, o objeto `PlaneSetup` não pode ser selecionado.

PREFERRED

`public static final int PREFERRED = 2`

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que este recurso é desejado no `PlaneSetup`. Prefere-se uma seleção com esse recurso do que uma seleção que não o inclua, embora ambas sejam válidas.

DONT_CARE

```
public static final int DONT_CARE = 3
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que a presença ou ausência deste recurso no `PlaneSetup` não tem importância. Preferências com esta prioridade não influenciam o resultado da seleção.

PREFERRED_NOT

```
public static final int PREFERRED_NOT = 4
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que este recurso não é desejado estar presente no `PlaneSetup`. Prefere-se uma seleção sem esse recurso do que uma seleção que o inclua, embora ambas sejam válidas.

REQUIRED_NOT

```
public static final int REQUIRED_NOT = 5
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que este recurso é não necessário estar presente no `PlaneSetup`. Se este recurso estiver disponível, o objeto `PlaneSetup` não pode ser selecionado.

NO_BACKGROUND_IMPACT

```
public static final int NO_BACKGROUND_IMPACT = 6
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que a configuração do plano não deve causar impacto em planos de fundo já exibidos.

Se utilizado com a prioridade `REQUIRED`, significa que não podem ocorrer alterações. Se utilizado com a prioridade `PREFERRED`, significa que podem ser realizadas alterações, porém minimizadas

As prioridades `PREFERRED_NOT` e `REQUIRED_NOT` podem ser ignoradas na seleção de um `PlaneSetup` para esse tipo de preferência.

Essa preferência é utilizada pela plataforma como uma restrição na seleção de configurações. Padrões gerados pela plataforma e então retornados para os aplicativos (por exemplo, de um método `getSetupPattern`) não podem ter suas preferências preenchidas pela plataforma.

NO_GRAPHICS_IMPACT

```
public static final int NO_GRAPHICS_IMPACT = 7
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que a configuração do plano não deve causar impacto em aplicativos gráficos já em execução.

Se utilizado com a prioridade `REQUIRED`, significa que não podem ocorrer alterações. Se utilizado com a prioridade `PREFERRED`, significa que podem ser realizadas alterações, porém minimizadas

As prioridades `PREFERRED_NOT` e `REQUIRED_NOT` podem ser ignoradas na seleção de um `PlaneSetup` para esse tipo de preferência.

Essa preferência é utilizada pela plataforma como uma restrição na seleção de configurações. Padrões gerados pela plataforma e então retornados para os aplicativos (por exemplo, de um método `getSetupPattern`) não

podem ter suas preferências preenchidas pela plataforma.

NO_SUBTITLE_IMPACT

```
public static final int NO_SUBTITLE_IMPACT = 8
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que a configuração do plano não deve causar impacto em aplicativos de legenda já em execução.

Se utilizado com a prioridade `REQUIRED`, significa que não podem ocorrer alterações. Se utilizado com a prioridade `PREFERRED`, significa que podem ser realizadas alterações, porém minimizadas

As prioridades `PREFERRED_NOT` e `REQUIRED_NOT` podem ser ignoradas na seleção de um `PlaneSetup` para esse tipo de preferência.

Essa preferência é utilizada pela plataforma como uma restrição na seleção de configurações. Padrões gerados pela plataforma e então retornados para os aplicativos (por exemplo, de um método `getSetupPattern`) não podem ter suas preferências preenchidas pela plataforma.

NO_VIDEO_IMPACT

```
public static final int NO_VIDEO_IMPACT = 9
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que a configuração do plano não deve causar impacto em aplicativos de streams de vídeo já em execução. Isso inclui também imagens paradas de vídeo.

Se utilizado com a prioridade `REQUIRED`, significa que não podem ocorrer alterações. Se utilizado com a prioridade `PREFERRED`, significa que podem ser realizadas alterações, porém minimizadas

As prioridades `PREFERRED_NOT` e `REQUIRED_NOT` podem ser ignoradas na seleção de um `PlaneSetup` para esse tipo de preferência.

Essa preferência é utilizada pela plataforma como uma restrição na seleção de configurações. Padrões gerados pela plataforma e então retornados para os aplicativos (por exemplo, de um método `getSetupPattern`) não podem ter suas preferências preenchidas pela plataforma.

NO_STILLVIDEO_IMPACT

```
public static final int NO_STILLVIDEO_IMPACT = 10
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que a configuração do plano não deve apresentar impacto em imagens paradas de vídeo já exibidas. Isso não inclui streams de vídeo em geral.

Se utilizado com a prioridade `REQUIRED`, significa que não podem ocorrer alterações. Se utilizado com a prioridade `PREFERRED`, significa que podem ser realizadas alterações, porém minimizadas.

As prioridades `PREFERRED_NOT` e `REQUIRED_NOT` podem ser ignoradas na seleção de um `PlaneSetup` para esse tipo de preferência.

Essa preferência é utilizada pela plataforma como uma restrição na seleção de configurações. Padrões gerados pela plataforma e então retornados para os aplicativos (por exemplo, de um método `getSetupPattern`) não podem ter suas preferências preenchidas pela plataforma.

INTERLACED_DISPLAY

```
public static final int INTERLACED_DISPLAY = 11
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que a configuração do plano suporta um display entrelaçado.

FLICKER_FILTERING

```
public static final int FLICKER_FILTERING = 12
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que a configuração do plano suporta filtragem de flicker (se suporta uma tela entrelaçada).

VIDEO_GRAPHICS_PIXEL_ALIGNED

```
public static final int VIDEO_GRAPHICS_PIXEL_ALIGNED = 13
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que a configuração do plano suporta a exibição de streams de vídeo e gráficos com pixels alinhados do mesmo tamanho. Isso não inclui alinhamento de origens de dois espaços de coordenadas de pixels. Onde um plano de vídeo está movendo o vídeo relativo à tela em tempo real (por exemplo, implementando "*Pan e Scan"*), as configurações de gráficos devem apenas suportar esse recurso onde a implementação de `GraphicsPlane` possa rastrear automaticamente as mudanças de posição no `VideoPlane`.

Essa preferência é utilizada pela plataforma como uma restrição na seleção de configurações. Padrões gerados pela plataforma e então retornados para os aplicativos (por exemplo, de um método `getSetupPattern`) não podem ter suas preferências preenchidas pela plataforma.

PIXEL_ASPECT_RATIO

```
public static final int PIXEL_ASPECT_RATIO = 14
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que a configuração do plano suporta a razão de aspecto, como especificado em um objeto `com.sun.dtv.lwuit.geom.Dimension` que indica a razão de aspecto de pixel x, y (relativo).

Instâncias de `PlaneSetupPattern` geradas pela plataforma e retornadas para os aplicativos (por exemplo a partir de um método `getPattern`) deve ter essa preferência configurada para um valor específico da plataforma com a prioridade `REQUIRED`.

PIXEL_RESOLUTION

```
public static final int PIXEL_RESOLUTION = 15
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que a configuração do plano suporta a resolução de pixels, como especificado em um objeto `com.sun.dtv.lwuit.geom.Dimension` que indica a resolução de `GraphicsPlane`.

Instâncias de `PlaneSetupPattern` geradas pela plataforma e retornadas para os aplicativos (por exemplo a partir de um método `getPattern`) deve ter essa preferência configurada para um valor específico da plataforma com a prioridade `REQUIRED`.

CHANGEABLE_SINGLE_COLOR

```
public static final int CHANGEABLE_SINGLE_COLOR = 16
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority`() no `PlaneSetupPattern`. Indica que um plano de fundo de uma só cor é exigido, onde a única cor pode ser alterada pelos aplicativos.

STILL_IMAGE

```
public static final int STILL_IMAGE = 17
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority`() no `PlaneSetupPattern`. Indica que um plano de fundo é necessário, e que pode suportar imagens paradas.

VIDEO_MIXING

```
public static final int VIDEO_MIXING = 18
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que a configuração deve suportar transparência no sistema de gráficos de forma que o output de um decodificador de vídeo esteja visível. (Isso inclui também imagens paradas de vídeos.) As configurações a seguir preenchem este requisito:

Configura com uma transformação definida entre pixels de vídeo / vídeo parado e pixels de gráficos (por exemplo, pixels são do mesmo tamanho).

Configura onde um aplicativo exibe gráficos de vídeo / vídeo parado, enquanto o vídeo / vídeo parado é utilizado como um plano de fundo. Portanto, não há transformação exigida entre os dois conjuntos de pixels em tais tipos de configurações.

Os aplicativos podem especificar um vídeo/vídeo parado em particular com o qual a combinação deve ser suportada. Nesses casos em que nenhuma configuração de vídeo específica é exigida, não é necessário especificar tal configuração e em vez disso pode-se utilizar `null`.

Essa preferência é utilizada pela plataforma como uma restrição na seleção de configurações. Padrões gerados pela plataforma e então retornados para os aplicativos (por exemplo, de um método `getSetupPattern`) não podem ter suas preferências preenchidas pela plataforma.

GRAPHICS_MIXING

```
public static final int GRAPHICS_MIXING = 19
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que a configuração do plano suporta a exibição de gráficos além de streams de vídeo. A tela inclui planos associados a ambos os tipos de configurações em que os pixels de vídeo e de gráfico estão totalmente alinhados (mesmo tamanho), assim como configurações em que são exibidos juntos, mas onde há uma relação mais complexa entre os dois espaços de coordenadas de pixels.

Essa preferência é utilizada pela plataforma como uma restrição na seleção de configurações. Padrões gerados pela plataforma e então retornados para os aplicativos (por exemplo, de um método `getSetupPattern`) não podem ter suas preferências preenchidas pela plataforma.

MATTE_SUPPORT

```
public static final int MATTE_SUPPORT = 20
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que a configuração deve suportar o recurso de mattes.

IMAGE_SCALING_SUPPORT

```
public static final int IMAGE_SCALING_SUPPORT = 21
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que a configuração deve suportar o redimensionamento de imagem rápido.

INDEXED_COLOR_SUPPORT

```
public static final int INDEXED_COLOR_SUPPORT = 22
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que a configuração de gráficos deve suportar uma `Color Lookup Table`.

DIRECT_COLOR_SUPPORT

```
public static final int DIRECT_COLOR_SUPPORT = 23
```

Um valor de preferência para uso nos métodos `setPreference` e `getPriority` no `PlaneSetupPattern`. Indica que a configuração de gráficos deve suportar `Direct Color`.

50.23.6 Detalhe dos construtores

PlaneSetupPattern

```
public PlaneSetupPattern()
```

50.23.7 Detalhe dos métodos

setPreference

```
public void setPreference(int preference,
                          int priority)
                   throws IllegalArgumentException
```

Configura a preferência indicada para ter a prioridade especificada. Se a preferência foi configurada previamente, a prioridade anterior para esta preferência deve ser substituída.

Atributos que não são preenchidos em um padrão (através de `setPreference`()), devem ter prioridade `DONT_CARE`. Toda configuração sempre satisfaz esses atributos.

Parâmetros:

`preference` – preferência a ser indicada. Os valores válidos para um `PlaneSetupPattern` são: `NO_GRAPHICS_IMPACT`, `NO_SUBTITLE_IMPACT`, `NO_BACKGROUND_IMPACT`, `NO_VIDEO_IMPACT`, `NO_STILLVIDEO_IMPACT`, `INTERLACED_DISPLAY`, `FLICKER_FILTERING`, `VIDEO_GRAPHICS_PIXEL_ALIGNED`, `CHANGEABLE_SINGLE_COLOR`, `STILL_IMAGE`, `VIDEO_MIXING`, `GRAPHICS_MIXING`, `MATTE_SUPPORT`, `IMAGE_SCALING_SUPPORT`, `INDEXED_COLOR_SUPPORT`, and `DIRECT_COLOR_SUPPORT`.

As subclasses podem adicionar valores válidos. Uma `IllegalArgumentException` deve ser executada se a preferência não for um valor válido para esta instância do `PlaneSetupPattern`

`priority` – prioridade da preferência. Os valores válidos são: `REQUIRED`, `PREFERRED`, `DONT_CARE`, `PREFERRED_NOT` e `REQUIRED_NOT`.

Se a `priority` não for uma prioridade válida como definida aqui, uma `IllegalArgumentException` deve ser executada.

Lança:

`IllegalArgumentException` – se a preferência ou parâmetro de prioridade não possui um valor válido

setPreference

```
public void setPreference(int preference,
                          Object value,
                          int priority)
                   throws IllegalArgumentException
```

Configura a preferência indicada para ter o valor especificado com a prioridade especificada. Uma chamada a este método pode também ser utilizada para substituir valores e prioridades de preferências configuradas anteriormente.

Parâmetros:

`preference` – preferência a ser indicada. Valores válidos são `PIXEL_ASPECT_RATIO` e `PIXEL_RESOLUTION`.

`value` – valor a ser configurado para a preferência indicada

`priority` – prioridade da preferência indicada. Os valores válidos são: `REQUIRED`, `PREFERRED`, `DONT_CARE`, `PREFERRED_NOT` e `REQUIRED_NOT`.

Lança:

`IllegalArgumentException` – se qualquer um dos parâmetros apresentar valor inadequado

getPreferenceValue

```
public Object getPreferenceValue(int preference)
                          throws IllegalArgumentException
```

Retorna o valor de preferência para a preferência especificada.

Parâmetros:

`preference` – preferência para a qual o valor deve ser recuperado. Valores válidos são `PIXEL_ASPECT_RATIO` e `PIXEL_RESOLUTION`.

Retorna:

valor válido atualmente para a preferência especificada. Deve ser um objeto `java.awt.Dimension`

Lança:

`IllegalArgumentException` – se o parâmetro apresentar um valor inadequado

getPriority

```
public int getPriority(int preference)
                throws IllegalArgumentException
```

Retorna a prioridade para a preferência especificada.

Propriedades individuais no `PlaneSetupPattern` podem ser examinadas utilizando o método `getPriority`. Os recursos implementados retornarão `REQUIRED`, os não implementados retornarão `REQUIRED_NOT`.

Parâmetros:

`preference` – preferência a ser indicada. Os valores válidos para um `PlaneSetupPattern` são: `NO_GRAPHICS_IMPACT`, `NO_SUBTITLE_IMPACT`, `NO_BACKGROUND_IMPACT`, `NO_VIDEO_IMPACT`, `NO_STILLVIDEO_IMPACT`, `INTERLACED_DISPLAY`, `FLICKER_FILTERING`, `VIDEO_GRAPHICS_PIXEL_ALIGNED`, `CHANGEABLE_SINGLE_COLOR`, `STILL_IMAGE`, `VIDEO_MIXING`, `GRAPHICS_MIXING`, `MATTE_SUPPORT`, `IMAGE_SCALING_SUPPORT`, `INDEXED_COLOR_SUPPORT`, `DIRECT_COLOR_SUPPORT`, `PIXEL_ASPECT_RATIO`, and `PIXEL_RESOLUTION`.

As subclasses podem adicionar valores válidos. Uma `IllegalArgumentException` deve ser executada se a

preferência não for um valor válido para esta instância do `PlaneSetupPattern`

Retorna:

prioridade para a preferência especificada.

Lança:

`IllegalArgumentException` – se o parâmetro da preferência apresentar um valor inválido

isSetupFitting

```
public boolean isSetupFitting(PlaneSetup setup)
```

Informa se o `PlaneSetup` especificado pode ser utilizado ou não para criar um plano que suporta os recursos configurados neste padrão.

Parâmetros:

`setup` - - objeto `PlaneSetup` a ser testado contra este padrão.

Retorna:

`true` se este objeto `PlaneSetup` puder ser utilizado para criar um plano que suporta o recurso configurado neste padrão, caso contrário, *`false`*,

50.24 Classe RemoteControl

50.24.1 Descrição da classe

`com.sun.dtv.ui`

```
java.lang.Object
  └com.sun.dtv.ui.UserInputDevice
      └com.sun.dtv.ui.Keyboard
          └com.sun.dtv.ui.RemoteControl
```

```
public class RemoteControl
extends Keyboard
```

Essa classe representa um controle remoto que pode ser utilizado para controlar uma tela em particular como um `UserInputDevice`. Um controle remoto é um caso especial de um `Keyboard`.

50.24.2 Índice de construtores

```
protected RemoteControl()
```

Não convém ser possível criar um `RemoteControl` para todos.

Métodos herdados da classe `com.sun.dtv.ui.`**`Keyboard`**

```
getInitiatedEvent, isSupported
```

50.24.3 Detalhe dos construtores

RemoteControl

`protected RemoteControl()`

Não convém ser possível criar um `RemoteControl` para todos. Como qualquer `UserInputDevice`, ele está ligado a uma tela.

50.25 Classe Screen

50.25.1 Descrição da classe

`com.sun.dtv.ui`

`java.lang.Object`

`└com.sun.dtv.ui.Screen`

Todas as interfaces implementadas

`ScarceResource`

`public class Screen`

`extends Object`

`implements ScarceResource`

Essa classe é uma representação de uma tela de TV. A tela consiste de uma variedade de planos dedicados a diversos proprósitos como vídeo, gráficos, legendas e plano de fundo. Tudo isso deve ser combinado antes de ser exibido. Caso uma plataforma suporte mais que um display, ela deve manipular uma instância desta classe para cada um. A saída de áudio também é suportada onde necessário. Além disso, há uma variedade de `UserInputDevices` dedicados a esta tela. Esses dispositivos de entrada de usuário podem ser utilizados para controlar esta tela em particular.

Embora cada plano de dispositivo de tela possua sua própria configuração, todos devem ter certas propriedades em comum. Configurações conflitantes de diversos planos para a mesma tela não são suportadas.

Essa classe também manipula a *thread* principal para as ferramentas citadas aqui como a EDT (*thread* de despacho de evento) semelhante à EDT do Swing. Essa *thread* encapsula a entrega do evento específico da plataforma e semântica de pintura e habilita os recursos de *threading* como animações, etc...

O EDT não pode ser bloqueado já que as operações e eventos também seriam bloqueados na maior parte do mesmo jeito que seriam em outras plataformas. Para serializar chamadas de volta para a EDT utilize os métodos `callSerially()` e `callSeriallyAndWait()`.

Observar que todas as chamadas LWUIT ocorrem na EDT (eventos, pintura, animações, etc...), LWUIT devem normalmente ser manipuladas também na EDT (portanto os métodos `callSerially()` e `callSeriallyAndWait()`). Teoricamente deve ser possível manipular alguns recursos LWUIT de outras *threads*, mas não se pode garantir que funcionará para todos os casos de uso.

50.25.2 Índice de métodos

`static void addResourceTypeListener(ResourceTypeListener listener)`

Adiciona um `ResourceTypeListener` a implementação.

`void addScarceResourceListener(ScarceResourceListener listener)`

Registra um *listener* para eventos sobre mudanças no estado de propriedade desta tela.

`void callSerially(Runnable r)`

Faz com que o executável seja chamado na EDT.

`void callSeriallyAndWait(Runnable r)`

Idêntico ao callSerially com o benefício agregado de esperar a conclusão do método `Runnable`.

`Plane[] getAllPlanes()`

Retorna uma lista de todos os planos para esta tela.

`static Screen[] getAvailableScreens()`

Retorna todas telas disponíveis neste sistema.

`PlaneSetup getBestSetup(PlaneSetupPattern[] patterns)`

Retorna um `PlaneSetup` de um plano que está presente nesta tela que melhor corresponde a no mínimo um dos `PlaneSetupPatterns` especificados.

`PlaneSetup[] getCoherentPlaneSetups(PlaneSetupPattern[] patterns)`

Retorna uma configuração coerente de `PlaneSetups` correspondente ao conjunto de padrões.

`static Screen getCurrentScreen()`

Retorna a tela em que o chamador é exibido.

`static Screen getDefaultScreen()`

Retorna a tela padrão para este aplicativo.

`static Screen[] getInstances()`

Retorna a lista de todas as intâncias de tela existentes, estejam reservadas ou não.

`Keyboard getKeyboard()`

Retorna o teclado que pode ser utilizado para controlar esta tela, se houver.

`Mouse getMouse()`

Retorna o mouse que pode ser utilizado para controlar esta tela, se houver.

`RemoteControl getRemoteControl()`

Retorna o controle remoto que pode ser utilizado para controlar esta tela, se houver.

`Dimension getScreenAspectRatio()`

Retorna a razão de aspecto desta tela.

`UserInputDevice[] getSupportedUserInputDevices()`

Retorna uma lista de `UserInputDevices` que estão associados a esta tela.

`UserInputEventManager getUserInputEventManager()`

Retorna o `UserInputEventManager` desta tela.

`void invokeAndBlock(Runnable r)`

Chama executáveis e bloqueia a *thread* atual, se a *thread* atual for a EDT ele ainda deve ser bloqueado, no entanto, uma *thread* diferente seria lançada para executar as tarefas da EDT enquanto está bloqueado.

`boolean isAvailable()`

Verifica se o dado recurso está correntemente disponível para reserva.

`boolean isEdt()`

Retorna *true* se estivermos na EDT.

`boolean isKeyboardSupported()`

Indica se esta tela pode ser controlada utilizando um teclado.

`boolean isMouseSupported()`

Indica se esta tela pode ser controlada utilizando um mouse.

`boolean isRemoteControlSupported()`

Indica se esta tela pode ser controlada utilizando um controle remoto.

`void release()`

Libera esse recurso.

`static void removeResourceTypeListener(ResourceTypeListener listener)`

Remove um *listener* previamente anexado.

`void removeScarceResourceListener(ScarceResourceListener listener)`

Remove um *listener* para eventos sobre mudanças no estado de propriedade desta tela.

`void reserve(boolean force, long timeout, ScarceResourceListener listener)`

Requisita reserva da dada instância de recursos escassos.

`static Screen reserveOne(boolean force, long timeoutms, ScarceResourceListener listener)`

Retorna uma instância reservada fora do pool de todas as intâncias `Screen`.

`boolean setCoherentPlaneSetups(PlaneSetup[] setups)`

Modifica as configurações para um conjunto de planos, baseado em seus `PlaneSetups` fornecidos.

50.25.3 Detalhe dos métodos

getAvailableScreens

`public static Screen[] getAvailableScreens()`

Retorna todas telas disponíveis neste sistema.

Retorna:

um *array* de telas representando todas as telas disponíveis neste sistema.

getDefaultScreen

`public static Screen getDefaultScreen()`

Retorna a tela padrão para este aplicativo.

Retorna:

tela padrão para este aplicativo.

getAllPlanes

`public Plane[] getAllPlanes()`

Retorna uma lista de todos os planos para esta tela.

Retorna:

um *array* de objetos `Plane` ou `null` se não existir nenhum.

getCoherentPlaneSetups

```
public PlaneSetup[] getCoherentPlaneSetups(PlaneSetupPattern[] patterns)
                                    throws IllegalArgumentException
```

Retorna uma configuração coerente de `PlaneSetups` correspondente ao conjunto de padrões. Um `PlaneSetup` retornará para cada `PlaneSetupPattern` fornecido como entrada.

Coerente significa que todas as propriedades são respeitadas em todos os padrões fornecidos e que uma configuração pode ser retornada para cada padrão fornecido.

Conflitos entre padrões são resolvidos como visto na descrição de `PlaneSetupPattern`.

Parâmetros:

`patterns` – um *array* de objetos descrevendo configurações desejadas.

Retorna:

um *array* de objetos não-nulos descrevendo um conjunto coerente de configurações de plano de tela, ou `null` se nenhum conjunto coerente for possível.

Lança:

`IllegalArgumentException` – se um *array* de comprimento zero for passado como um argumento

Relaciona-se com:

`setCoherentPlaneSetups(com.sun.dtv.ui.PlaneSetup[])`

setCoherentPlaneSetups

```
public boolean setCoherentPlaneSetups(PlaneSetup[] setups)
                               throws SecurityException,
                                      IllegalArgumentException,
                                      SetupException
```

Modifica as configurações para um conjunto de planos, baseado em seus `PlaneSetups` fornecidos.

Parâmetros:

`setups` – o *array* de configurações que devem ser aplicados (onde possível).

Retorna:

Um boolean indicando se todos os `PlaneSetups` puderam ser aplicados com sucesso. Mesmo que nenhum dos `PlaneSetups` possa ser aplicado com sucesso, a configuração depois deste método pode não corresponder à configuração de planos antes de este método ser chamado. não há garantia de que a implementação não tenha alterado configurações de tela para melhor atender a requisitos (mesmo que nem todos os requisitos possam ser preenchidos). Os aplicativos devem, portanto, investigar se uma mudança parcial das configurações foi realizada para cada plano mesmo neste caso.

Lança:

`SecurityException` – se o aplicativo não tem direitos suficientes para configurar o `PlaneSetup` para qualquer plano.

`IllegalArgumentException` – se o *array* especificado estiver vazio.

`SetupException` – se um elemento do *array* de `PlaneSetup` especificado não for válido para nenhum dos planos.

Relaciona-se com:

getCoherentPlaneSetups(com.sun.dtv.ui.PlaneSetupPattern[])

getBestSetup

public PlaneSetup getBestSetup(PlaneSetupPattern[] patterns)

Retorna um PlaneSetup de um plano que está presente nesta tela que melhor corresponde a no mínimo um dos PlaneSetupPatterns especificados. Se isso não for possível, é retornado null.

Melhor no sentido deste método significa que as configurações satisfazem a maior quantidade possível de preferências com prioridades PREFERRED e PREFERRED_NOT, assim como com REQUIRED e REQUIRED_NOT. Configurações são escolhidas aplicando-se o seguinte algorítimo, com base nas prioridades como fornecido para setPreference. Configuração de escolha:

DEVE satisfazer todas as preferências cujas prioridades foram REQUIRED

NÃO PODE satisfazer nenhuma das preferências cujas prioridades foram REQUIRED_NOT.

CONVÉM satisfazer a maior quantidade possível de preferências cujas prioridades foram PREFERRED.

CONVÉM satisfazer a menor quantidade possível de preferências cujas prioridades foram PREFERRED_NOT.

As preferências cujas prioridades foram DONT_CARE são ignoradas.

Este método retorna null se nenhuma configuração que estiver presente satisfaz todas as prioridades REQUIRED e REQUIRED_NOT.

Parâmetros:

patterns - *array* de objetos PlaneSetupPattern para escolher.

Retorna:

um objeto PlaneSetup que seja a melhor configuração correspondente possível, ou null se nenhum for encontrado.

getScreenAspectRatio

public Dimension getScreenAspectRatio()

Retorna a razão de aspecto desta tela.

Retorna:

razão de aspecto desta tela

addScarceResourceListener

public void addScarceResourceListener(ScarceResourceListener listener)

Registra um *listener* para eventos sobre mudanças no estado de propriedade desta tela. Se este *listener* for adicionado antes, as chamadas adicionarão mais referências ao *listener*. Consequentemente essas referências receberão múltiplas cópias de um único evento.

Parâmetros:

listener – objeto a ser informado sobre mudanças do estado.

Relaciona-se com:

removeScarceResourceListener(com.sun.dtv.resources.ScarceResourceListener)

removeScarceResourceListener

public void removeScarceResourceListener(ScarceResourceListener listener)

Remove um *listener* para eventos sobre mudanças no estado de propriedade desta tela. Este método não tem efeito se o *listener* especificado não tiver sido registrado antes.

Parâmetros:

`listener` – objeto que não está mais interessado.

Relaciona-se com:

`addScarceResourceListener(com.sun.dtv.resources.ScarceResourceListener)`

reserve

```
public void reserve(boolean force,
                    long timeout,
                    ScarceResourceListener listener)
             throws IllegalArgumentException,
                    TimeoutException
```

Requisita reserva da dada instância de recursos escassos. O método bloquear-se-á até que a instância esteja disponível. O método retorna normalmente apenas se a reserva for bem sucedida. Todos os outros casos são manuseados por exceções.

Primeiro, se houver um gerenciador de segurança, seu método `checkPermission` é chamado com a permissão `ScarceResourcePermission(name, "reserve")`. Se `force` é além disso definido *`true`*, a permissão também verificado no `ScarceResourcePermission(name, "force")`.

Durante o processo de reserva, se a instância do recurso já estiver alocada, a implementação deve notificar o proprietário atual do recurso sobre a requisição de reserva por meio da interface `ScarceResourceListener`:

- ou por `ScarceResourceListener.releaseRequested()` se forçar for *false*,
- ou por `ScarceResourceListener.releaseForced()` no outro caso.

O *listener* deve ser usado para tais notificações apenas até o recurso ser liberado. Após a liberação, a implementação não deve chamar quaisquer dos métodos da interface do *listener*.

Após esse primeiro evento, a implementação procederá de acordo e liberará (ou não) o recurso requisitado. No caso da implementação liberar o recurso, desencadeará um evento `ScarceResourceListener.released()` ao *listener* proprietário original do recurso informando-o que o recurso não lhe pertence mais.

O aplicativo pode controlar o tempo de espera para que tal recurso esteja disponível ao definir `timeoutms`. No caso da duração da reserva exceder o tempo expresso em `timeoutms`, um `TimeoutException` é lançado.

Sob operação normal, recursos são liberados usando o método `release`. De qualquer forma, no caso do aplicativo não liberar recursos quando requisitado ou o aplicativo ser terminado, a implementação deve liberar todos os recursos alocados para o aplicativo para permitir que outros aplicativos sejam notificados de mudanças na alocação de recursos e poderem reservá-los. Ver a seção Resource Cleanup do ciclo de vida do aplicativo.

Especificado por:

`reserve` na interface `ScarceResource`

Parâmetros:

`force` - Se *`true`*, esse método retirará o recurso do proprietário atual. Se *`false`*, a implementação bloquear-se-á e esperará até que o recurso esteja disponível (usando `release()`) ou até `timeoutms`.

`listener` - O *listener* a ser anexado para receber notificação sobre o status do recurso.

Lança:

`IllegalArgumentException` - Se o *listener* estiver `null` ou se o valor especificado em `timeoutms` não for válido, isto é, nem um inteiro positivo, nem -1.

`TimeoutException` - Se o tempo especificado em `timeoutms` houver acabado e o recurso não houver podido ser reservado.

reserveOne

```
public static Screen reserveOne(boolean force,
                                long timeoutms,
                                ScarceResourceListener listener)
                         throws IllegalArgumentException,
                                TimeoutException
```

Retorna uma instância reservada fora do pool de todas as intâncias `Screen`. Esse método retorna com a instância ou executa uma exceção de acordo com a situação.

Esse método se comporta exatamente como o método `reserve()`.

Parâmetros:

`force` - Se *`true`*, esse método tem permissão de retirar qualquer recurso do atual proprietário. Se *`false`*, a implementação bloqueará e esperará até que um recurso de um dado `type` esteja disponível (utilizando `release()`) ou até milissegundos `timeoutms`.

`timeoutms` – Uma quantidade positiva de tempo em milissegundos até a qual este método esperará para que qualquer recurso seja liberado por seu proprietário atual. O valor `-1` indica que a implementação esperará para sempre.

`listener` - O *listener* a ser anexado para receber notificação sobre o status do recurso.

Retorna:

A instância de tipo `Screen` que foi reservada.

Lança:

`IllegalArgumentException` - Se o *listener* estiver `null` ou se o valor especificado em `timeoutms` não for válido, isto é, nem um inteiro positivo, nem -1.

`TimeoutException` - Se o tempo especificado em `timeoutms` houver acabado e o recurso não houver podido ser reservado.

Relaciona-se com:

`reserve()`

release

```
public void release()
```

Libera esse recurso.

O recurso deve ser disponibilizado para outro aplicativo através da plataforma. Após chamar esse método, não é mais possível interagir com o dado recurso: chamadas para métodos críticos desses recursos escassos devem ser ignorados e lançar `IllegalStateException`. Essa afirmação é válida e é o comportamento requerido de qualquer classe implementando a interface `ScarceResource`. Para interagir novamente com o dado recurso, o aplicativo deve chamar o método `reserve()` e se tornar o proprietário novamente.

A implementação pode organizar quaisquer recursos do sistema que esse objeto estiver usando. Depois a implementação não deve chamar quaisquer dos métodos do *listener* que estava anexado no momento da reserva.

Se o recurso já estava disponível (isto é, não reservado), esse método não faz nada.

Especificado por:

`release` na interface `ScarceResource`

isAvailable

`public boolean isAvailable()`

Verifica se o dado recurso está correntemente disponível para reserva. O valor retornado dá a situação atual e não garante que o recurso ainda estará disponível em um momento posterior, por exemplo, no momento da reserva: outro aplicativo pode ter tomado aquele recurso no tempo decorrido.

Especificado por:

`isAvailable` na interface `ScarceResource`

Retorna:

Um boolean definido *`true`* se o dado recurso estiver atualmente disponível para reserva.

getSupportedUserInputDevices

`public UserInputDevice[] getSupportedUserInputDevices()`

Retorna uma lista de `UserInputDevices` que estão associados a esta tela.

Retorna:

uma lista de `UserInputDevices` associados a esta tela

isKeyboardSupported

`public boolean isKeyboardSupported()`

Indica se esta tela pode ser controlada utilizando um teclado.

Retorna:

true se o teclado for suportado, caso contrário, *false*

getKeyboard

`public Keyboard getKeyboard()`

Retorna o teclado que pode ser utilizado para controlar esta tela, se houver.

Retorna:

o objeto Keyboard representando o teclado que pode ser utilizado para controlar esta tela, `null` se não houver nenhum.

isRemoteControlSupported

`public boolean isRemoteControlSupported()`

Indica se esta tela pode ser controlada utilizando um controle remoto.

Retorna:

true se o controle remoto for suportado, caso contrário, *false*.

getRemoteControl

`public RemoteControl getRemoteControl()`

Retorna o controle remoto que pode ser utilizado para controlar esta tela, se houver.

Retorna:

o objeto `RemoteControl` representando o controle remoto que pode ser utilizado para controlar esta tela, `null` se não houver nenhum.

isMouseSupported

`public boolean isMouseSupported()`

Indica se esta tela pode ser controlada utilizando um mouse.

Retorna:

true se o mouse for suportado, caso contrário, *false*.

getMouse

`public Mouse getMouse()`

Retorna o mouse que pode ser utilizado para controlar esta tela, se houver.

Retorna:

o objeto Mouse representando o mouse que pode ser utilizado para controlar esta tela, `null` se não houver nenhum.

getUserInputEventManager

`public UserInputEventManager getUserInputEventManager()`

Retorna o `UserInputEventManager` desta tela. Qualquer tela possui uma instância fixa de `UserInputEventManager` para ser capaz de manipular eventos de input de usuário vindos de diversos aplicativos. O `UserInputEventManager` é criado junto com a tela, ele não pode ser configurado ou alterado. Os aplicativos devem utilizar este método a fim de ganhar acesso ao gerenciador para aplicar para acesso exclusivo ou não-exclusivo ao `UserInputEvents`.

Retorna:

`UserInputEventManager` desta tela.

getCurrentScreen

`public static Screen getCurrentScreen()`

Retorna a tela em que o chamador é exibido.

Retorna:

tela atual.

getInstances

`public static Screen[] getInstances()`

Retorna a lista de todas as intâncias de tela existentes, estejam reservadas ou não.

Retorna:

lista de todas as instâncias existentes do tela.

addResourceTypeListener

```
public static void addResourceTypeListener(ResourceTypeListener listener)
                                    throws NullPointerException
```

Adiciona um `ResourceTypeListener` a implementação. Sempre que um `reserve()` ou um `release()` for chamado para quaisquer recursos do mesmo tipo, a implementação chamará os métodos correspondentes do *listener*.

Parâmetros:

`listener` - O *listener* desencadeado para eventos em recursos do mesmo tipo.

Lança:

`NullPointerException` - Se o *listener* estiver `null`.

Relaciona-se com:

`removeResourceTypeListener()`

removeResourceTypeListener

```
public static void removeResourceTypeListener(ResourceTypeListener listener)
                                       throws NullPointerException
```

Remove um *listener* previamente anexado. Se o *listener* não foi anexado antes, esse método não faz nada..

Parâmetros:

`listener` - O *listener* desencadeado para eventos em recursos do mesmo tipo.

Lança:

`NullPointerException` - Se o *listener* estiver `null`.

Relaciona-se com:

`addResourceTypeListener()`

isEdt

```
public boolean isEdt()
```

Retorna *true* se estivermos na EDT. Isso é útil para um código genérico que pode ser utilizado tanto com a EDT como fora dela.

Retorna:

tue se estivermos na EDT; caso contrário, *false.*

callSerially

```
public void callSerially(Runnable r)
```

Faz com que o executável seja chamado na EDT. Este método retorna imediatamente e não esperará que a chamada serial aconteça.

Parâmetros:

`r` - executável (não é uma *thread*) que deve ser chamado na EDT serial para a pintura e eventos de manipulação de chave.

Lança:

`IllegalStateException` – se este método for chamado na EDT (por exemplo, durante pintura ou manipulação de evento).

callSeriallyAndWait

```
public void callSeriallyAndWait(Runnable r)
```

Idêntico ao `callSerially` com o benefício agregado de esperar a conclusão do método `Runnable`.

Parâmetros:

`r` - executável (não é uma *thread*) que deve ser chamado na EDT serial para a pintura e eventos de manipulação de chave.

Lança:

`IllegalStateException` – se este método for chamado na *thread* de despacho de evento (por exemplo, durante pintura ou manipulação de evento).

invokeAndBlock

```
public void invokeAndBlock(Runnable r)
```

Chama executáveis e bloqueia a *thread* atual, se a *thread* atual for a EDT ela ainda deve ser bloqueada, no entanto, uma *thread* diferente seria lançada para executar as tarefas da EDT enquanto está bloqueada. Com o bloqueio concluído, a EDT é restaurada na sua posição original. Isso é muito semelhante à caixa de ferramentas "*foxtrot*" do Swing e permite codificar lógica "mais simples" que exige código de bloqueio no meio das áreas sensíveis a evento.

Parâmetros:

`r` - executável (não é uma *thread*) que deve ser chamado sincronicamente por este método.

50.26 Classe SetupException

50.26.1 Descrição da classe

`com.sun.dtv.ui`

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └com.sun.dtv.ui.SetupException
```

Todas as interfaces implementadas

```
Serializable
```

```
public class SetupException
extends Exception
```

Exceção a ser executada em diversas situações onde tenta-se realizar uma alteração de configuração ilegal de um ou mais planos de uma tela.

50.26.2 Índice de construtores

```
SetupException()
```

50.26.3 Detalhe dos construtores

SetupException

```
public SetupException()
```

50.27 Classe SophisticatedTextLayoutManager

50.27.1 Descrição da classe

com.sun.dtv.ui

```
java.lang.Object
  └com.sun.dtv.ui.SophisticatedTextLayoutManager
```

Todas as interfaces implementadas

`TextLayoutManager`

```
public class SophisticatedTextLayoutManager
extends Object
implements TextLayoutManager
```

Essa classe fornece um `TextLayoutManager` com mais recursos que o `DefaultTextLayoutManager` para habilitar possibilidades de design de layout mais sofisticadas para o teste ser mostrado em componentes de TV. É recomendado que os algorítimos para rederização de texto sigam a nomenclatura e algorítimo padrão para layout de texto bidirecional como definido em Unicode 3.0. Consulte http://unicode.org para detalhes.

Relaciona-se com:

`DefaultTextLayoutManager`, `TextLayoutManager`

50.27.2 Índice de campos

`static int DEFAULT_LINE_SPACE`

Constante a ser utilizada com o método `setLineSpace` de `SophisticatedTextLayoutManager`.

`static int HORIZONTAL_ALIGN_CENTER`

Constante a ser utilizada com o método `setHorizontalAlignment` de `SophisticatedTextLayoutManager`.

`static int HORIZONTAL_ALIGN_JUSTIFIED`

Constante a ser utilizada com o método `setHorizontalAlignment` de `SophisticatedTextLayoutManager`.

`static int HORIZONTAL_ALIGN_LEFT`

Constante a ser utilizada com o método `setHorizontalAlignment` de `SophisticatedTextLayoutManager`.

`static int HORIZONTAL_ALIGN_RIGHT`

Constante a ser utilizada com o método `setHorizontalAlignment` de

SophisticatedTextLayoutManager.

static int LINE_ORIENTATION_HORIZONTAL

Constante a ser utilizada com o método setLineOrientation de SophisticatedTextLayoutManager.

static int LINE_ORIENTATION_VERTICAL

Constante a ser utilizada com o método setLineOrientation de SophisticatedTextLayoutManager.

static int START_LOWER_LEFT

Constante a ser utilizada com o método setStartCorner de SophisticatedTextLayoutManager.

static int START_LOWER_RIGHT

Constante a ser utilizada com o método setStartCorner de SophisticatedTextLayoutManager.

static int START_UPPER_LEFT

Constante a ser utilizada com o método setStartCorner de SophisticatedTextLayoutManager.

static int START_UPPER_RIGHT

Constante a ser utilizada com o método setStartCorner de SophisticatedTextLayoutManager.

static int VERTICAL_ALIGN_BOTTOM

Constante a ser utilizada com o método setVerticalAlignment de SophisticatedTextLayoutManager.

static int VERTICAL_ALIGN_CENTER

Constante a ser utilizada com o método setVerticalAlignment de SophisticatedTextLayoutManager.

static int VERTICAL_ALIGN_JUSTIFIED

Constante a ser utilizada com o método setVerticalAlignment de SophisticatedTextLayoutManager.

static int VERTICAL_ALIGN_TOP

Constante a ser utilizada com o método setVerticalAlignment de SophisticatedTextLayoutManager.

50.27.3 Índice de construtores

SophisticatedTextLayoutManager()

50.27.4 Índice de métodos

void addTextOverflowListener(TextOverflowListener listener)

Registra um TextOverflowListener.

int getHorizontalAlignment()

Recupera alinhamento horizontal.

Insets getInsets()

Retorna os adendos como configurado pelo método setInsets.

Dimension getMaximumSize(ViewOnlyComponent component, String text)

Fornece o tamanho máximo exigido para renderizar o conteúdo de texto fornecido no ViewOnlyComponent especificado.

Dimension getMaximumSize(ViewOnlyComponent component, String text, Insets insets)

Fornece o tamanho máximo exigido para renderizar o conteúdo de texto fornecido no ViewOnlyComponent especificado.

Dimension getMinimumSize(ViewOnlyComponent component, String text)

Fornece o tamanho mínimo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

```
Dimension getMinimumSize(ViewOnlyComponent component, String text, Insets insets)
```

Fornece o tamanho mínimo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

```
Dimension getPreferredSize(ViewOnlyComponent component, String text)
```

Fornece o tamanho preferido exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

```
Dimension getPreferredSize(ViewOnlyComponent component, String text, Insets insets)
```

Fornece o tamanho preferido exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

```
int getVerticalAlignment()
```

Recupera alinhamento vertical.

```
void removeTextOverflowListener(TextOverflowListener listener)
```

Remove um `TextOverflowListener` previamente registrado deste `TextLayoutManager`.

```
void render(String text, Graphics g, ViewOnlyComponent component, Insets insets)
```

Renderizar um *string*.

```
int retrieveHorizontalTabSpacing()
```

Recupera espaçamento de tabulador horizontal.

```
int retrieveLetterSpace()
```

Recupera espaço de letra.

```
int retrieveLineOrientation()
```

Recupera orientação de linha.

```
int retrieveLineSpace()
```

Recupera espaçamento de linha.

```
int retrieveStartCorner()
```

Recupera ponto de início.

```
boolean retrieveTextWrapping()
```

Recupera “text wrapping” (contorno de texto).

```
void setHorizontalAlignment(int alignment)
```

Determina o alinhamento horizontal.

```
void setHorizontalTabSpacing(int spacing)
```

Determina o espaçamento de tabulação horizontal.

```
void setInsets(Insets insets)
```

Determina os adendos a serem utilizados pelo `SophisticatedTextLayoutManager` para fornecer uma margem virtual.

```
void setLetterSpace(int spacing)
```

Determina o espaço entre letras.

`void setLineOrientation(int orientation)`

Determina a orientação da linha.

`void setLineSpace(int spacing)`

Determina o espaço entre linhas.

`void setStartCorner(int corner)`

Determina o ponto de início para o texto.

`void setTextWrapping(boolean wrap)`

Determina se o texto é contornado.

`void setVerticalAlignment(int alignment)`

Determina o alinhamento vertical.

50.27.5 Detalhe dos campos

HORIZONTAL_ALIGN_LEFT

`public static final int HORIZONTAL_ALIGN_LEFT = 0`

Constante a ser utilizada com o método `setHorizontalAlignment` de `SophisticatedTextLayoutManager`. Indica que todo o conteúdo deve ser alinhado à esquerda. Alinhado à esquerda significa que o texto é alinhado à esquerda se a orientação for horizontal, o texto inicia no canto superior esquerdo, e é lido da esquerda para a direita, de cima para baixo.

HORIZONTAL_ALIGN_CENTER

`public static final int HORIZONTAL_ALIGN_CENTER = 1`

Constante a ser utilizada com o método `setHorizontalAlignment` de `SophisticatedTextLayoutManager`. Indica que todo o conteúdo deve ser centralizado horizontalmente. Centralizado horizontalmente significa que o texto é centralizado horizontalmente se a linha de orientação for horizontal, o texto inicia no canto superior esquerdo, e é lido da esquerda para a direita, de cima para baixo.

HORIZONTAL_ALIGN_RIGHT

`public static final int HORIZONTAL_ALIGN_RIGHT = 2`

Constante a ser utilizada com o método `setHorizontalAlignment` de `SophisticatedTextLayoutManager`. Indica que todo o conteúdo deve ser alinhado à direita. Alinhado à direita significa que o texto é alinhado à direita se a orientação for horizontal, o texto inicia no canto superior esquerdo, e é lido da esquerda para a direita, de cima para baixo.

HORIZONTAL_ALIGN_JUSTIFIED

`public static final int HORIZONTAL_ALIGN_JUSTIFIED = 3`

Constante a ser utilizada com o método `setHorizontalAlignment` de `SophisticatedTextLayoutManager`. Indica que todo o conteúdo deve ser justificado horizontalmente. Justificado horizontalmente significa que o texto é escrito como um bloco horizontal se a linha de orientação for horizontal, o texto inicia no canto superior esquerdo, e é lido da esquerda para a direita, de cima para baixo.

VERTICAL_ALIGN_TOP

`public static final int VERTICAL_ALIGN_TOP = 4`

Constante a ser utilizada com o método `setVerticalAlignment` de `SophisticatedTextLayoutManager`.

Indica que todo o conteúdo deve ser alinhado acima.

VERTICAL_ALIGN_CENTER

```
public static final int VERTICAL_ALIGN_CENTER = 5
```

Constante a ser utilizada com o método `setVerticalAlignment` de `SophisticatedTextLayoutManager`. Indica que todo o conteúdo deve ser centralizado verticalmente.

VERTICAL_ALIGN_BOTTOM

```
public static final int VERTICAL_ALIGN_BOTTOM = 6
```

Constante a ser utilizada com o método `setVerticalAlignment` de `SophisticatedTextLayoutManager`. Indica que todo o conteúdo deve ser alinhado abaixo.

VERTICAL_ALIGN_JUSTIFIED

```
public static final int VERTICAL_ALIGN_JUSTIFIED = 7
```

Constante a ser utilizada com o método `setVerticalAlignment` de `SophisticatedTextLayoutManager`. Indica que todo o conteúdo deve ser justificado verticalmente.

LINE_ORIENTATION_HORIZONTAL

```
public static final int LINE_ORIENTATION_HORIZONTAL = 8
```

Constante a ser utilizada com o método `setLineOrientation` de `SophisticatedTextLayoutManager`. Indica que a orientação da linha é horizontal.

LINE_ORIENTATION_VERTICAL

```
public static final int LINE_ORIENTATION_VERTICAL = 9
```

Constante a ser utilizada com o método `setLineOrientation` de `SophisticatedTextLayoutManager`. Indica que a orientação da linha é vertical.

START_UPPER_LEFT

```
public static final int START_UPPER_LEFT = 10
```

Constante a ser utilizada com o método `setStartCorner` de `SophisticatedTextLayoutManager`. Indica que o texto inicia no canto superior esquerdo.

START_UPPER_RIGHT

```
public static final int START_UPPER_RIGHT = 11
```

Constante a ser utilizada com o método `setStartCorner` de `SophisticatedTextLayoutManager`. Indica que o texto inicia no canto superior direito.

START_LOWER_LEFT

```
public static final int START_LOWER_LEFT = 12
```

Constante a ser utilizada com o método `setStartCorner` de `SophisticatedTextLayoutManager`. Indica que o texto inicia no canto inferior esquerdo.

START_LOWER_RIGHT

```
public static final int START_LOWER_RIGHT = 13
```

Constante a ser utilizada com o método `setStartCorner` de `SophisticatedTextLayoutManager`. Indica que o texto inicia no canto inferior direito.

DEFAULT_LINE_SPACE

```
public static final int DEFAULT_LINE_SPACE = -1
```

Constante a ser utilizada com o método `setLineSpace` de `SophisticatedTextLayoutManager`. Indica que o espaço da linha deve ser configurado com o valor-padrão, determinado pela fonte-padrão.

50.27.6 Detalhe dos construtores

SophisticatedTextLayoutManager

```
public SophisticatedTextLayoutManager()
```

50.27.7 Detalhe dos métodos

getMinimumSize

```
public Dimension getMinimumSize(ViewOnlyComponent component,
                                String text)
```

Descrição copiada da interface: **`TextLayoutManager`**

Fornece o tamanho mínimo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado. Nesta versão do método, presume-se que o parâmetro `ViewOnlyComponent` define as bordas pelos seus limites.

Especificado por:

`getMinimumSize` na interface `TextLayoutManager`

Parâmetros:

`component` - `ViewOnlyComponent` para o qual olhar

`text` – texto fornecido para ser renderizado

Retorna:

tamanho mínimo como objeto `com.sun.dtv.lwuit.geom.Dimension`.

getMinimumSize

```
public Dimension getMinimumSize(ViewOnlyComponent component,
                                String text,
                                Insets insets)
```

Descrição copiada da interface: **`TextLayoutManager`**

Fornece o tamanho mínimo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

Especificado por:

`getMinimumSize` na interface `TextLayoutManager`

Parâmetros:

`component` - `ViewOnlyComponent` para o qual olhar

`text` – texto fornecido para ser renderizado

`insets` – suplementos para definir a área na qual o texto deve ser colocado

Retorna:

tamanho mínimo como objeto `com.sun.dtv.lwuit.geom.Dimension`.

getMaximumSize

```
public Dimension getMaximumSize(ViewOnlyComponent component,
                                String text)
```

Descrição copiada da interface: **`TextLayoutManager`**

Fornece o tamanho máximo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado. Nesta versão do método, presume-se que o parâmetro `ViewOnlyComponent` define as bordas pelos seus limites.

Especificado por:

`getMaximumSize` na interface `TextLayoutManager`

Parâmetros:

`component` - `ViewOnlyComponent` para o qual olhar

`text` – texto fornecido para ser renderizado

Retorna:

tamanho máximo como objeto `com.sun.dtv.lwuit.geom.Dimension`.

getMaximumSize

```
public Dimension getMaximumSize(ViewOnlyComponent component,
                                String text,
                                Insets insets)
```

Descrição copiada da interface: **`TextLayoutManager`**

Fornece o tamanho máximo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

Especificado por:

`getMaximumSize` na interface `TextLayoutManager`

Parâmetros:

`component` - `ViewOnlyComponent` para o qual olhar

`text` – texto fornecido para ser renderizado

`insets` – suplementos para definir a área na qual o texto deve ser colocado

Retorna:

tamanho máximo como objeto `com.sun.dtv.lwuit.geom.Dimension`.

getPreferredSize

```
public Dimension getPreferredSize(ViewOnlyComponent component,
                                  String text)
```

Descrição copiada da interface: **`TextLayoutManager`**

Fornece o tamanho preferido exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado. Nesta versão do método, presume-se que o parâmetro `ViewOnlyComponent` define as bordas pelos seus limites.

Especificado por:

`getPreferredSize` na interface `TextLayoutManager`

Parâmetros:

`component` - `ViewOnlyComponent` para o qual olhar

`text` – texto fornecido para ser renderizado

Retorna:

tamanho preferido como objeto `com.sun.dtv.lwuit.Dimension`.

getPreferredSize

```
public Dimension getPreferredSize(ViewOnlyComponent component,
                                  String text,
                                  Insets insets)
```

Descrição copiada da interface: **`TextLayoutManager`**

Fornece o tamanho preferido exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

Especificado por:

`getPreferredSize` na interface `TextLayoutManager`

Parâmetros:

`component` - `ViewOnlyComponent` para o qual olhar

`text` – texto fornecido para ser renderizado

`insets` – suplementos para definir a área na qual o texto deve ser colocado

Retorna:

tamanho preferido como objeto `com.sun.dtv.lwuit.Dimension`.

render

```
public void render(String text,
                   Graphics g,
                   ViewOnlyComponent component,
                   Insets insets)
```

Renderizar um *string*. O `ViewOnlyComponent` passado pode ser utilizado para determinar qualquer informação adicional necessária para renderizar a *string* de forma adequada (por ex., fonte, cor, etc.), se a classe `TextLayoutManager` implementando esta interface não fornecer as informações adicionais para isso.

O `ViewOnlyComponent` também define a área de layout por seus limites, enquanto os suplementos fornecidos também devem ser considerados se não forem `null`. Porém, o retângulo de recorte do objeto `Graphics` não deve estar sujeito a mudanças pelo `TextLayoutManager`.

Especificado por:

`render` na interface `TextLayoutManager`

Parâmetros:

`text` – *string* a ser renderizada

`g` – contexto dos gráficos. Isso inclui também um retângulo de recorte, que deve ser respeitado como bordas dentro das quais a renderização é permtida. Um parâmetro de suplementos diferente de `null` deve ser também levado em consideração.

`component` - `ViewOnlyComponent` no qual renderizar.

`insets` – suplementos para definir a área na qual o texto deve ser colocado. Este parâmetro também pode ser `null`: nesse caso, o parâmetro `ViewOnlyComponent` define as bordas pelos seus limites.

setHorizontalAlignment

```
public void setHorizontalAlignment(int alignment)
```

Determina o alinhamento horizontal.

Parâmetros:

`alignment` – valor para alinhamento horizontal Valores possíveis são: `HORIZONTAL_ALIGN_LEFT`, `HORIZONTAL_ALIGN_RIGHT`, `HORIZONTAL_ALIGN_CENTER` e `HORIZONTAL_ALIGN_JUSTIFIED`.

Relaciona-se com:

`getHorizontalAlignment()`

setVerticalAlignment

```
public void setVerticalAlignment(int alignment)
```

Determina o alinhamento vertical.

Parâmetros:

`alignment` – valor para alinhamento vertical. Valores possíveis são: `VERTICAL_ALIGN_TOP`, `VERTICAL_ALIGN_BOTTOM`, `VERTICAL_ALIGN_CENTER` e `VERTICAL_ALIGN_JUSTIFIED`.

Relaciona-se com:

`getVerticalAlignment()`

setLineOrientation

```
public void setLineOrientation(int orientation)
```

Determina a orientação da linha.

Parâmetros:

`orientation` – valor de orientação da linha. Valores possíveis são: `LINE_ORIENTATION_HORIZONTAL` e `LINE_ORIENTATION_VERTICAL`.

setStartCorner

```
public void setStartCorner(int corner)
```

Determina o ponto de início para o texto.

Parâmetros:

`corner` – valor para o ponto de início. Valores possíveis são: `START_UPPER_LEFT`, `START_UPPER_RIGHT`, `START_LOWER_LEFT` e `START_LOWER_RIGHT`.

setTextWrapping

`public void setTextWrapping(boolean wrap)`

Determina se o texto é contornado.

Parâmetros:

`wrap` - *`true`* se o texto deve ser contornado, caso contrário, *`false`*

setLineSpace

`public void setLineSpace(int spacing)`

Determina o espaço entre linhas.

Parâmetros:

`spacing` – valor para espaçamento de linha; um número inteiro ou `DEFAULT_LINE_SPACE`

setLetterSpace

`public void setLetterSpace(int spacing)`

Determina o espaço entre letras.

Parâmetros:

`spacing` – configuração de espaço da letra em unidades de 1/256 pontos

setHorizontalTabSpacing

`public void setHorizontalTabSpacing(int spacing)`

Determina o espaçamento de tabulação horizontal.

Parâmetros:

`spacing` – espaçamento de tabulador horizontal em pontos

getHorizontalAlignment

`public int getHorizontalAlignment()`

Recupera alinhamento horizontal.

Retorna:

alinhamento horizontal

Relaciona-se com:

`setHorizontalAlignment(int)`

getVerticalAlignment

`public int getVerticalAlignment()`

Recupera alinhamento vertical.

Retorna:

alinhamento vertical

Relaciona-se com:

`setVerticalAlignment(int)`

retrieveLineOrientation

`public int retrieveLineOrientation()`

Recupera orientação de linha.

Retorna:

orientação da linha

retrieveStartCorner

`public int retrieveStartCorner()`

Recupera ponto de início.

Retorna:

ponto de início

retrieveTextWrapping

`public boolean retrieveTextWrapping()`

Recupera “text wrapping” (contorno de texto).

Retorna:

contorno de texto

retrieveLineSpace

`public int retrieveLineSpace()`

Recupera espaçamento de linha.

Retorna:

espaçamento da linha: um número ou `DEFAULT_LINE_SPACE`, se o espaçamento de linha padrão for determinado

retrieveLetterSpace

`public int retrieveLetterSpace()`

Recupera espaço de letra.

Retorna:

espaçamento de letra em 1/256 pontos

retrieveHorizontalTabSpacing

`public int retrieveHorizontalTabSpacing()`

Recupera espaçamento de tabulador horizontal.

Retorna:

espaçamento de tabulador horizontal

setInsets

```
public void setInsets(Insets insets)
```

Determina os adendos a serem utilizados pelo `SophisticatedTextLayoutManager` para fornecer uma margem virtual. Os adendos determinados por este método deve ser adicionado aos adendos passados ao método `render`. Se este método não for chamado, os adendos-padrão são 0 em cada extremidade.

Parâmetros:

`insets` – adendos determinados

Relaciona-se com:

`getInsets()`

getInsets

```
public Insets getInsets()
```

Retorna os adendos como configurado pelo método `setInsets`. Quando não configurado previamente, nenhum adendo é retornado.

Retorna:

adendos como configurado pelo método `setInsets`

Relaciona-se com:

`setInsets(java.awt.Insets)`

addTextOverflowListener

```
public void addTextOverflowListener(TextOverflowListener listener)
```

Registra um `TextOverflowListener`. Este *listener* deve ser notificado se uma *string* de texto não couber no componente durante uma tentativa de renderizá-lo.

Parâmetros:

`listener` - *listener* a ser registrado

Relaciona-se com:

`removeTextOverflowListener(com.sun.dtv.ui.TextOverflowListener)`

removeTextOverflowListener

```
public void removeTextOverflowListener(TextOverflowListener listener)
```

Remove um `TextOverflowListener` previamente registrado deste `TextLayoutManager`.

Parâmetros:

`listener` - *listener* a ser removido

Relaciona-se com:

`addTextOverflowListener(com.sun.dtv.ui.TextOverflowListener)`

50.28 Classe StaticMatte

50.28.1 Descrição da classe

com.sun.dtv.ui

```
java.lang.Object
  └com.sun.dtv.ui.StaticMatte
```

Todas as interfaces implementadas

Matte

```
public class StaticMatte
extends Object
implements Matte
```

Esta classe representa um matte inanimado, ou seja, um matte imutável com o tempo. Uma máscara estática pode ter um valor de opacidade único, sendo constante para todo o matte, ou alternativamente uma máscara de imagem, onde os valores de pixel determina a transparência do matte para cada pixel.

Relaciona-se com:

`Matte`

50.28.2 Índice de construtores

`StaticMatte()`

50.28.3 Índice de métodos

`Image getImageOpacity()`

Obtém a imagem do valor de opacidade para este matte.

`Dimension getOffset()`

Obtém o deslocamento que o matte de imagem apresenta relativamente com canto superior esquerdo da área do componente.

`float getOpacity()`

Obtém o valor de opacidade para este matte.

`int getOpacityAsInt()`

Obtém o valor de opacidade para este matte como número inteiro.

`boolean hasOpacityImage()`

Obtém se uma imagem de opacidade é atribuída a esse matte.

`void setOffset(Dimension d)`

Determina o deslocamento para a imagem de opacidade a ser atribuída com esse matte.

`void setOpacity(Image image)`

Configura a opacidade do matte.

`void setOpacity(float opacity)`

Configura a opacidade do matte para um valor único, válido para todos os pixels.

`void setOpacity(int opacity)`

Configura a opacidade do matte para um valor único, válido para todos os pixels.

50.28.4 Detalhe dos construtores

StaticMatte

`public StaticMatte()`

50.28.5 Detalhe dos métodos

setOpacity

`public void setOpacity(int opacity)`

Configura a opacidade do matte para um valor único, válido para todos os pixels. Um valor de opacidade ou imagem de opacidade atribuído anteriormente deve ser substituído pela opacidade unitária causada por uma chamada a este método. Este método tem exatamente o mesmo objetivo que o `setOpacity(float)`, em que uma chamada a `setOpacity(i)` tem o mesmo efeito que uma chamada a `setOpacity((float)(i/255))`.

Parâmetros:

`opacity` – valor para a opacidade, que deve ser um valor de número inteiro entre 0 e 255, onde 0 indica total tranparência e and 255, total opacidade. Valores abaixo de 0 devem ser interpretados como 0 (totalmente transparente), valores acima de 255 como 255 (totalmente opaco).

Relaciona-se com:

`getOpacityAsInt()`, `setOpacity(float)`

setOpacity

`public void setOpacity(float opacity)`

Configura a opacidade do matte para um valor único, válido para todos os pixels. Um valor de opacidade ou imagem de opacidade atribuído anteriormente deve ser substituído pela opacidade unitária causada por uma chamada a este método. Este método tem exatamente o mesmo objetivo que o `setOpacity(int)`, em que uma chamada a `setOpacity(f)` tem o mesmo efeito que uma chamada a `setOpacity((int)(f * 255))`.

Parâmetros:

`opacity` – valor para a opacidade, que deve ser um valor de ponto flutuante entre 0,0 e 1,0, onde 0,0 indica total tranparência e and 1,0, total opacidade. Valores abaixo de 0,0 devem ser interpretados como 0,0 (totalmente transparente), valores acima de 1,0 como 1,0 (totalmente opaco).

Relaciona-se com:

`getOpacity()`, `setOpacity(int)`

setOpacity

`public void setOpacity(Image image)`

Configura a opacidade do matte. A imagem fornecida pode ser menor que o componente afetado, nesse caso a área restante é mostrada da forma que seria se coberta por um matte com opacidade unitária de 1,0 (mesmo que essa imagem substitua uma opacidade unitária configurada anteriormente com um valor diferente). Como padrão o matte de imagens é alinhado com o canto esquero superior da área dos componentes. Isso pode ser alteado com o método `setOffset()`. Um valor de opacidade ou imagem de opacidade atribuído anteriormente pela

imagem de opacidade fornecido por uma chamada a este método.

Parâmetros:

`image` – valor para opacidade, que deve ser uma imagem, contendo valores de pixel entre 0,0 e 1,0. 0,0 indica total transparência, e 1,0 total opacidade. Valores abaixo de 0,0 devem ser interpretados como 0,0 (totalmente transparente), valores acima de 1,0 como 1,0 (totalmente opaco).

Relaciona-se com:

`getImageOpacity(),getOpacity()`

getOpacity

```
public float getOpacity()
```

Obtém o valor de opacidade para este matte. Se o matte tem uma imagem de opacidade atribuída, sempre é retornado 1,0. Para descobrir se isso realmente significa uma opacidade unitária de 1,0, ou se uma opacidade de imagem `hasOpacityImage()` deve ser chamada.

Retorna:

valor de opacidade para este matte. Esse é sempre um valor de ponto flutuante na faixa de 0,0 e 1,0, onde 0,0 indica total transparência, e 1,0 total opacidade. 1,0 pode também significar que há uma imagem de opacidade atribuída a este matte.

Relaciona-se com:

`setOpacity(float)`

getOpacityAsInt

```
public int getOpacityAsInt()
```

Obtém o valor de opacidade para este matte como número inteiro. Se o matte tem uma imagem de opacidade atribuída, sempre é retornado 255. Para descobrir se isso realmente significa uma opacidade unitária de 255, ou se uma opacidade de imagem `hasOpacityImage()` deve ser chamada.

Retorna:

valor de opacidade para este matte. Esse é sempre um valor de de número inteiro na faixa entre 0 e 255, onde 0 indica total transparência, e 255 total opacidade. 255 pode também significar que há uma imagem de opacidade atribuída a este matte.

Relaciona-se com:

`setOpacity(int),getOpacity()`

getImageOpacity

```
public Image getImageOpacity()
```

Obtém a imagem do valor de opacidade para este matte.

Retorna:

o valor de opacidade para este matte na forma de uma imagem, contendo valores de pixel entre 0,0 e 1,0. 0,0 indica total transparência, e 1,0 total opacidade. Se for retornado `null`, há um valor de opacidade unitária atribuída a este matte no momento.

Relaciona-se com:

`setOpacity(com.sun.dtv.lwuit.Image)`

setOffset

`public void setOffset(Dimension d)`

Determina o deslocamento para a imagem de opacidade a ser atribuída com esse matte. Enquanto por padrão, o matte de imagens é alinhado no canto esquerdo superior da área de componentes, isso pode alterado utilizando este método. Se não houver uma imagem de opacidade atribuída a este matte, uma chamada a este método não tem efeito.

Parâmetros:

`d` – um objeto `Dimension` determinando o deslocamento em direção ao canto superior esquerdo da área do componente

se o parâmetro for `null`, nenhuma deslocamento é determinado, e um deslocamento previamente atribuído deve ser removido

Relaciona-se com:

`getOffset()`

getOffset

`public Dimension getOffset()`

Obtém o deslocamento que o matte de imagem apresenta relativamente com canto superior esquerdo da área do componente. Caso não haja imagem de opacidade atribuída a este matte, é retornado `null`.

Retorna:

o deslocamento em forma de um objeto `Dimension` ou `null` (nesse caso ou não há deslocamento ou é um valor de opacidade unitária atribuído ao matte em vez de uma imagem)

Relaciona-se com:

`setOffset(com.sun.dtv.lwuit.geom.Dimension)`

hasOpacityImage

`public boolean hasOpacityImage()`

Obtém se uma imagem de opacidade é atribuída a esse matte. Retorna *true* se esse for o caso, caso contrário, *false*.

Retorna:

true se uma imagem de opacidade for atribuída a este matte, *false* se for um valor de opacidade unitário.

50.29 Interface TextLayoutManager

50.29.1 Descrição da interface

`com.sun.dtv.ui`

Todas as classes implementadoras conhecidas

`DefaultTextLayoutManager`, `SophisticatedTextLayoutManager`

`public interface TextLayoutManager`

O objetivo desta interface é definir funcionalidade para o layout de *string*s e sua renderização na tela. O *string* a ser renderizado pode conter, além de seu conteúdo real, metainformações, por exemplo, fonte, cor, estilo e formatação de texto. Apesar disso, está fora do escopo dessa interface para forçar funcionalidade apropriada para

isso. Um gerenciador de layout de texto de implementação padrão pode implementar apenas o método de renderização como solicitado pela interface, enquanto gerenciadores de layout mais sofisticados podem adicionar muitos métodos para processar metainformações, por exemplo

alteração de cores, fontes e estilos

forçar quebras de linha

alinhamento do texto

contorno de texto

orientação do texto

suporte de linguagem específica

Relaciona-se com:

`DefaultTextLayoutManager`, `SophisticatedTextLayoutManager`

50.29.2 Índice de métodos

`Dimension getMaximumSize(ViewOnlyComponent component, String text)`

Fornece o tamanho máximo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

`Dimension getMaximumSize(ViewOnlyComponent component, String text, Insets insets)`

Fornece o tamanho máximo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

`Dimension getMinimumSize(ViewOnlyComponent component, String text)`

Fornece o tamanho mínimo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

`Dimension getMinimumSize(ViewOnlyComponent component, String text, Insets insets)`

Fornece o tamanho mínimo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

`Dimension getPreferredSize(ViewOnlyComponent component, String text)`

Fornece o tamanho preferido exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

`Dimension getPreferredSize(ViewOnlyComponent component, String text, Insets insets)`

Fornece o tamanho preferido exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

`void render(String text, Graphics g, ViewOnlyComponent component, Insets insets)`

Renderizar um *string*.

50.29.3 Detalhe dos métodos

getMinimumSize

```
Dimension getMinimumSize(ViewOnlyComponent component,
                         String text)
```

Fornece o tamanho mínimo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado. Nesta versão do método, presume-se que o parâmetro `ViewOnlyComponent` define as bordas

pelos seus limites.

Parâmetros:

`component` - `ViewOnlyComponent` para o qual olhar

`text` – texto fornecido para ser renderizado

Retorna:

tamanho mínimo como objeto `com.sun.dtv.lwuit.geom.Dimension`.

getMinimumSize

```
Dimension getMinimumSize(ViewOnlyComponent component,
                         String text,
                         Insets insets)
```

Fornece o tamanho mínimo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

Parâmetros:

`component` - `ViewOnlyComponent` para o qual olhar

`text` – texto fornecido para ser renderizado

`insets` – suplementos para definir a área na qual o texto deve ser colocado

Retorna:

tamanho mínimo como objeto `com.sun.dtv.lwuit.geom.Dimension`.

getMaximumSize

```
Dimension getMaximumSize(ViewOnlyComponent component,
                         String text)
```

Fornece o tamanho máximo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado. Nesta versão do método, presume-se que o parâmetro `ViewOnlyComponent` define as bordas pelos seus limites.

Parâmetros:

`component` - `ViewOnlyComponent` para o qual olhar

`text` – texto fornecido para ser renderizado

Retorna:

tamanho máximo como objeto `com.sun.dtv.lwuit.geom.Dimension`.

getMaximumSize

```
Dimension getMaximumSize(ViewOnlyComponent component,
                         String text,
                         Insets insets)
```

Fornece o tamanho máximo exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

Parâmetros:

`component` - `ViewOnlyComponent` para o qual olhar

`text` – texto fornecido para ser renderizado

`insets` – suplementos para definir a área na qual o texto deve ser colocado

Retorna:

tamanho máximo como objeto `com.sun.dtv.lwuit.geom.Dimension`.

getPreferredSize

```
Dimension getPreferredSize(ViewOnlyComponent component,
                           String text)
```

Fornece o tamanho preferido exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado. Nesta versão do método, presume-se que o parâmetro `ViewOnlyComponent` define as bordas pelos seus limites.

Parâmetros:

`component` - `ViewOnlyComponent` para o qual olhar

`text` – texto fornecido para ser renderizado

Retorna:

tamanho preferido como objeto `com.sun.dtv.lwuit.Dimension`.

getPreferredSize

```
Dimension getPreferredSize(ViewOnlyComponent component,
                           String text,
                           Insets insets)
```

Fornece o tamanho preferido exigido para renderizar o conteúdo de texto fornecido no `ViewOnlyComponent` especificado.

Parâmetros:

`component` - `ViewOnlyComponent` para o qual olhar

`text` – texto fornecido para ser renderizado

`insets` – suplementos para definir a área na qual o texto deve ser colocado

Retorna:

tamanho preferido como objeto `com.sun.dtv.lwuit.Dimension`.

render

```
void render(String text,
            Graphics g,
            ViewOnlyComponent component,
            Insets insets)
```

Renderizar um *string*. O `ViewOnlyComponent` passado pode ser utilizado para determinar qualquer informação adicional necessária para renderizar o *string* de forma adequada (por ex., Fonte, Cor, etc.), se a classe `TextLayoutManager` implementando esta interface não fornecer as informações adicionais para isso.

O `ViewOnlyComponent` também define a área de layout por seus limites, enquanto os suplementos fornecidos também devem ser considerados se não forem `null`. Porém, o retângulo de recorte do objeto `Graphics` não deve estar sujeito a mudanças pelo `TextLayoutManager`.

Parâmetros:

`text` – *string* a ser renderizado

`g` – contexto dos gráficos. Isso inclui também um retângulo de recorte, que deve ser respeitado como bordas dentro das quais a renderização é permtida. Um parâmetro de suplementos diferente de `null` deve ser também levado em consideração.

`component` - `ViewOnlyComponent` no qual renderizar.

`insets` – suplementos para definir a área na qual o texto deve ser colocado. Este parâmetro também pode ser `null`: nesse caso, o parâmetro `ViewOnlyComponent` define as bordas pelos seus limites.

50.30 Interface TextOverflowListener

50.30.1 Descrição da interface

`com.sun.dtv.ui`

Todas as super-interfaces

`EventListener`

```
public interface TextOverflowListener
```

```
extends EventListener
```

O objetivo de um `TextOverflowListener` é ser notificado quando uma *string* de texto não se encaixa em um componente durante uma tentativa de rederizá-lo.

Relaciona-se com:

`SophisticatedTextLayoutManager`

50.30.2 Índice de métodos

`void notifyTextOverflow(String text, ViewOnlyComponent component, boolean overflowHorizontal, boolean overflowVertical)`

Este método é chamado pelo `SophisticatedTextLayoutManager` se um *string* de texto não couber no componente durante uma tentativa de rederizá-lo.

50.30.3 Detalhe dos métodos

notifyTextOverflow

```
void notifyTextOverflow(String text,
                        ViewOnlyComponent component,
                        boolean overflowHorizontal,
                        boolean overflowVertical)
```

Este método é chamado pelo `SophisticatedTextLayoutManager` se um *string* de texto não couber no componente durante uma tentativa de rederizá-lo.

Parâmetros:

`text` – *string* a ser renderizado

`component` - `ViewOnlyComponent` no qual a tentativa de renderização ocorreu

`overflowHorizontal` - *`true`* se o texto ultrapassar horizontalmente; caso contrário, *`false`*

`overflowVertical` - *`true`* se o texto ultrapassar verticalmente; caso contrário, *`false`*

50.31 Classe UserInputDevice

50.31.1 Descrição da classe

com.sun.dtv.ui

```
java.lang.Object
  └com.sun.dtv.ui.UserInputDevice
```

Subclasses diretas conhecidas

`Keyboard, Mouse`

```
abstract public class UserInputDevice
extends Object
```

Essa classe abstrata é a base para todos os dispositivos de entrada que podem ser utilizados para controlar uma tela.

50.31.2 Índice de construtores

`protected UserInputDevice()`

Não pode ser possível criar instâncias de subclasses de `UserInputDevice` por todos, elas estão ligadas a uma tela.

50.31.3 Índice de métodos

`abstract UserInputEvent getInitiatedEvent(int code)`

Retorna o evento de input que deve ser produzido por uma classe de implementação para o código especificado.

50.31.4 Detalhe dos construtores

UserInputDevice

`protected UserInputDevice()`

Não pode ser possível criar instâncias de subclasses de `UserInputDevice` por todos, elas estão ligadas a uma tela.

50.31.5 Detalhe dos métodos

getInitiatedEvent

```
public abstract UserInputEvent getInitiatedEvent(int code)
                                          throws IllegalArgumentException
```

Retorna o evento de input que deve ser produzido por uma classe de implementação para o código especificado.

Parâmetros:

`code` – código ao qual o evento de input é iniciado, se inserido

Retorna:

evento de input iniciado

Lança:

`IllegalArgumentException` – se o código especificado não for suportado por este `UserInputDevice`

50.32 Interface ViewOnlyComponent

50.32.1 Descrição da interface

`com.sun.dtv.ui`

Todas as super-interfaces

`MatteEnabled`

Todas as classes implementadoras conhecidas

`Button, CheckBox, ComboBox, DefaultListCellRenderer, Label, List, RadioButton`

```
public interface ViewOnlyComponent
extends MatteEnabled
```

Esta interface representa qualquer tipo de componente não-interativo no sistema e os fornece um `LookAndFeel` plugável.

50.32.2 Índice de campos

`int HORIZONTAL_ALIGN_CENTER`

Constante a ser utilizada com o método `setHorizontalAlignment` do `ViewOnlyComponent`.

`int HORIZONTAL_ALIGN_JUSTIFIED`

Constante a ser utilizada com o método `setHorizontalAlignment` do `ViewOnlyComponent`.

`int HORIZONTAL_ALIGN_LEFT`

Constante a ser utilizada com o método `setHorizontalAlignment` do `ViewOnlyComponent`.

`int HORIZONTAL_ALIGN_RIGHT`

Constante a ser utilizada com o método `setHorizontalAlignment` do `ViewOnlyComponent`.

`int SCALE_ASPECT_PROOF`

Constante a ser utilizada com o método `setScalingMode` do `ViewOnlyComponent`.

`int SCALE_NO`

Constante a ser utilizada com o método `setScalingMode` do `ViewOnlyComponent`.

`int SCALE_NO_ASPECT_PROOF`

Constante a ser utilizada com o método `setScalingMode` do `ViewOnlyComponent`.

`int STATE_DISABLED`

Constante a ser utilizada com diversos métodos de `ViewOnlyComponent` exigindo um parâmetro de estado.

`int STATE_ENABLED`

Constante a ser utilizada com diversos métodos de `ViewOnlyComponent` exigindo um parâmetro de estado.

`int VERTICAL_ALIGN_BOTTOM`

Constante a ser utilizada com o método `setVerticalAlignment` do `ViewOnlyComponent`.

`int VERTICAL_ALIGN_CENTER`

Constante a ser utilizada com o método `setVerticalAlignment` do `ViewOnlyComponent`.

`int VERTICAL_ALIGN_JUSTIFIED`

Constante a ser utilizada com o método `setVerticalAlignment` do `ViewOnlyComponent`.

`int VERTICAL_ALIGN_TOP`

Constante a ser utilizada com o método `setVerticalAlignment` do `ViewOnlyComponent`.

50.32.3 Índice de métodos

`Image[] getAnimateContent(int state)`

Retorna o conteúdo animado desse componente, dependendo do estado atual.

`Image getGraphicContent(int state)`

Retorna conteúdo gráfico desse componente, dependendo do estado atual.

`int getHorizontalAlignment()`

Restaura o alinhamento horizontal de qualquer conteúdo baseado em estado nesse componente.

`int getInteractionState()`

Retorna o estado de interação atual do componente.

`int getScalingMode()`

Restaura o modo de redimensionamento desse componente.

`String getTextContent(int state)`

Retorna conteúdo de texto desse componente, dependendo do estado atual.

`TextLayoutManager getTextLayoutManager()`

Obtém o gerenciador de layout de texto que está em uso para o layout de texto nesse componente.

`int getVerticalAlignment()`

Restaura o alinhamento vertical de qualquer conteúdo com base em estado nesse componente.

`boolean isDoubleBuffered()`

Retorna *true* se o buffering duplo estiver disponível e ativado.

`boolean isOpaque()`

Retorna *true* se toda a área do componente (conforme retornado pelo método `com.sun.dtv.lwuit.Component#getBounds`, for opaco.

`void paint(Graphics g)`

Método para pintar o componente.

`void processEvent(AWTEvent event)`

Lida com o `AWTEvent` determinado.

`void setAnimateContent(Image[] images, int state)`

Designa um *array* de conteúdo gráfico para ser usado em animação desse componente, que é dependente do estado.

`void setGraphicContent(Image image, int state)`

Designa o conteúdo gráfico a esse componente que é dependente do estado.

`void setHorizontalAlignment(int alignment)`

Define o alinhamento horizontal para qualquer conteúdo baseado no estado desse componente.

`void setInteractionState(int state)`

Define o estado atual do componente com relação a interação.

`void setScalingMode(int scaling)`

Define o modo de redimensionamento para esse componente.

`void setTextContent(String text, int state)`

Designa o conteúdo do texto a esse componente que é dependente do estado.

`void setTextLayoutManager(TextLayoutManager manager)`

Define o gerenciador de *layout* de texto que deve ser usado para o *layout* de texto nesse componente.

`void setVerticalAlignment(int alignment)`

Define o alinhamento vertical de qualquer conteúdo com base em estado desse componente.

Métodos herdados da interface `com.sun.dtv.ui.`**`MatteEnabled`**

`getMatte, setMatte`

50.32.4 Detalhe dos campos

HORIZONTAL_ALIGN_LEFT

`public static final int HORIZONTAL_ALIGN_LEFT = 0`

Constante a ser utilizada com o método `setHorizontalAlignment` do `ViewOnlyComponent`. Indica que todo o conteúdo deve ser alinhado à esquerda.

HORIZONTAL_ALIGN_CENTER

`public static final int HORIZONTAL_ALIGN_CENTER = 1`

Constante a ser utilizada com o método `setHorizontalAlignment` do `ViewOnlyComponent`. Indica que todo o conteúdo deve ser centralizado horizontalmente.

HORIZONTAL_ALIGN_RIGHT

`public static final int HORIZONTAL_ALIGN_RIGHT = 2`

Constante a ser utilizada com o método `setHorizontalAlignment` do `ViewOnlyComponent`. Indica que todo o conteúdo deve ser alinhado à direita.

HORIZONTAL_ALIGN_JUSTIFIED

`public static final int HORIZONTAL_ALIGN_JUSTIFIED = 3`

Constante a ser utilizada com o método `setHorizontalAlignment` do `ViewOnlyComponent`. Indica que todo o conteúdo deve ser justificado horizontalmente.

VERTICAL_ALIGN_TOP

```
public static final int VERTICAL_ALIGN_TOP = 4
```

Constante a ser utilizada com o método `setVerticalAlignment` do `ViewOnlyComponent`. Indica que todo o conteúdo deve ser alinhado acima.

VERTICAL_ALIGN_CENTER

```
public static final int VERTICAL_ALIGN_CENTER = 5
```

Constante a ser utilizada com o método `setVerticalAlignment` do `ViewOnlyComponent`. Indica que todo o conteúdo deve ser centralizado verticalmente.

VERTICAL_ALIGN_BOTTOM

```
public static final int VERTICAL_ALIGN_BOTTOM = 6
```

Constante a ser utilizada com o método `setVerticalAlignment` do `ViewOnlyComponent`. Indica que todo o conteúdo deve ser alinhado abaixo.

VERTICAL_ALIGN_JUSTIFIED

```
public static final int VERTICAL_ALIGN_JUSTIFIED = 7
```

Constante a ser utilizada com o método `setVerticalAlignment` do `ViewOnlyComponent`. Indica que todo o conteúdo deve ser justificado verticalmente.

SCALE_NO

```
public static final int SCALE_NO = 8
```

Constante a ser utilizada com o método `setScalingMode` do `ViewOnlyComponent`. Indica que nada do conteúdo deve ser redimensionado para receber o componente.

SCALE_ASPECT_PROOF

```
public static final int SCALE_ASPECT_PROOF = 9
```

Constante a ser utilizada com o método `setScalingMode` do `ViewOnlyComponent`. Indica que todos os conteúdos devem ser redimensionados de modo a preservar a razão de aspecto.

SCALE_NO_ASPECT_PROOF

```
public static final int SCALE_NO_ASPECT_PROOF = 10
```

Constante a ser utilizada com o método `setScalingMode` do `ViewOnlyComponent`. Indica que todos os conteúdos devem ser redimensionados de modo a não necessariamente preservar a razão de aspecto.

STATE_ENABLED

```
public static final int STATE_ENABLED = 1
```

Constante a ser utilizada com diversos métodos de `ViewOnlyComponent` exigindo um parâmetro de estado. Indica que uma operação executada no método relacionado é realizada de acordo com o estado habilitado do componente.

STATE_DISABLED

```
public static final int STATE_DISABLED = 0
```

Constante a ser utilizada com diversos métodos de `ViewOnlyComponent` exigindo um parâmetro de estado. Indica que uma operação executada no método relacionado é realizada de acordo com o estado desabilitado do componente.

50.32.5 Detalhe dos métodos

paint

```
void paint(Graphics g)
```

Método para pintar o componente.

Parâmetros:

`g` - o contexto gráfico a ser usado para pintar.

setTextContent

```
void setTextContent(String text,
                    int state)
             throws IllegalArgumentException
```

Designa o conteúdo do texto a esse componente que é dependente do estado.

Parâmetros:

`text` – o conteúdo do texto, ou `null` (remove o conteúdo do texto que foi designado antes)

`state` - o estado do componente para o qual esse conteúdo deve ser exibido; `STATE_ENABLED` ou `STATE_DISABLED`

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

Relaciona-se com:

`getTextContent(int)`

setGraphicContent

```
void setGraphicContent(Image image,
                       int state)
                throws IllegalArgumentException
```

Designa o conteúdo gráfico a esse componente que é dependente do estado.

Parâmetros:

`image` - o conteúdo gráfico, ou `null` (remove o conteúdo gráfico que foi designado antes)

`state` - o estado do componente para o qual esse conteúdo deve ser exibido; `STATE_ENABLED` ou `STATE_DISABLED`

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

Relaciona-se com:

`getGraphicContent(int)`

setAnimateContent

```
void setAnimateContent(Image[] images,
                       int state)
                throws IllegalArgumentException
```

Designa um *array* de conteúdo gráfico para ser usado em animação desse componente, que é dependente do estado.

Parâmetros:

`images` - o conteúdo gráfico de um *array*, ou `null` (remove qualquer conteúdo gráfico que foi designado antes)

`state` - o estado do componente para o qual esse conteúdo deve ser exibido; `STATE_ENABLED` ou `STATE_DISABLED`

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

Relaciona-se com:

`getAnimateContent(int)`

getTextContent

```
String getTextContent(int state)
               throws IllegalArgumentException
```

Retorna conteúdo de texto desse componente, dependendo do estado atual.

Parâmetros:

`state` - o estado do componente para o qual esse conteúdo deve ser restaurado; `STATE_ENABLED` ou `STATE_DISABLED`

Retorna:

O conteúdo do texto associado ao estado determinado, ou `null` se não houver nenhum.

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

Relaciona-se com:

`setTextContent(java.lang.String, int)`

getGraphicContent

```
Image getGraphicContent(int state)
                throws IllegalArgumentException
```

Retorna conteúdo gráfico desse componente, dependendo do estado atual.

Parâmetros:

`state` - o estado do componente para o qual esse conteúdo deve ser restaurado; `STATE_ENABLED` ou `STATE_DISABLED`

Retorna:

O conteúdo gráfico associado ao estado determinado, ou `null` se não houver nenhum.

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

Relaciona-se com:

`setGraphicContent(com.sun.dtv.lwuit.Image, int)`

getAnimateContent

```
Image[] getAnimateContent(int state)
                   throws IllegalArgumentException
```

Retorna o conteúdo animado desse componente, dependendo do estado atual.

Parâmetros:

`state` - o estado do componente para o qual esse conteúdo deve ser restaurado; `STATE_ENABLED` ou `STATE_DISABLED`

Retorna:

O conteúdo gráfico associado ao estado determinado, ou `null` se não houver nenhum.

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

Relaciona-se com:

`setAnimateContent(com.sun.dtv.lwuit.Image[], int)`

setInteractionState

```
void setInteractionState(int state)
                  throws IllegalArgumentException
```

Define o estado atual do componente com relação a interação.

Parâmetros:

`state` – o estado de interação para esse conteúdo; `STATE_ENABLED` ou `STATE_DISABLED`

Lança:

`IllegalArgumentException` – se o parâmetro do estado não representar um estado

Relaciona-se com:

`getInteractionState()`

getInteractionState

```
int getInteractionState()
```

Retorna o estado de interação atual do componente.

Retorna:

o estado de interação atual do componente.

Relaciona-se com:

`setInteractionState(int)`

setTextLayoutManager

`void setTextLayoutManager(TextLayoutManager manager)`

Define o gerenciador de layout de texto que deve ser usado para o layout de texto nesse componente.

Parâmetros:

`manager` – o `TextLayoutManager`

Relaciona-se com:

`getTextLayoutManager()`

getTextLayoutManager

`TextLayoutManager getTextLayoutManager()`

Obtém o gerenciador de layout de texto que está em uso para o layout de texto nesse componente.

Retorna:

o `TextLayoutManager`

Relaciona-se com:

`setTextLayoutManager(com.sun.dtv.ui.TextLayoutManager)`

setHorizontalAlignment

`void setHorizontalAlignment(int alignment)`

Define o alinhamento horizontal para qualquer conteúdo baseado no estado desse componente.

Parâmetros:

`alignment` – modo de alinhamento horizontal, um de `HORIZONTAL_ALIGN_LEFT`, `HORIZONTAL_ALIGN_CENTER`, `HORIZONTAL_ALIGN_RIGHT` ou `HORIZONTAL_ALIGN_JUSTIFIED`.

Relaciona-se com:

`getHorizontalAlignment()`

setVerticalAlignment

`void setVerticalAlignment(int alignment)`

Define o alinhamento vertical de qualquer conteúdo com base em estado desse componente.

Parâmetros:

`alignment` – modo de alinhamento vertical, um de `VERTICAL_ALIGN_TOP`, `VERTICAL_ALIGN_CENTER`, `VERTICAL_ALIGN_BOTTOM` ou `VERTICAL_ALIGN_JUSTIFIED`.

Relaciona-se com:

`getVerticalAlignment()`

getHorizontalAlignment

`int getHorizontalAlignment()`

Restaura o alinhamento horizontal de qualquer conteúdo baseado em estado nesse componente.

Retorna:

modo de alinhamento horizontal atual, um de `HORIZONTAL_ALIGN_LEFT`, `HORIZONTAL_ALIGN_CENTER`, `HORIZONTAL_ALIGN_RIGHT` ou `HORIZONTAL_ALIGN_JUSTIFIED`.

Relaciona-se com:

`setHorizontalAlignment(int)`

getVerticalAlignment

`int getVerticalAlignment()`

Restaura o alinhamento vertical de qualquer conteúdo com base em estado nesse componente.

Retorna:

modo de alinhamento horizontal atual, um de `VERTICAL_ALIGN_TOP`, `VERTICAL_ALIGN_CENTER`, `VERTICAL_ALIGN_BOTTOM` ou `VERTICAL_ALIGN_JUSTIFIED`.

Relaciona-se com:

`setVerticalAlignment(int)`

setScalingMode

`void setScalingMode(int scaling)`

Define o modo de redimensionamento para esse componente.

Parâmetros:

`scaling` – modo de redimensionamento, um de `SCALE_NO`, `SCALE_NO_ASPECT_PROOF` ou `SCALE_ASPECT_PROOF`.

Relaciona-se com:

`getScalingMode()`

getScalingMode

`int getScalingMode()`

Restaura o modo de redimensionamento desse componente.

Retorna:

modo de redimensionamento atual, um de `SCALE_NO`, `SCALE_NO_ASPECT_PROOF` ou `SCALE_ASPECT_PROOF`.

Relaciona-se com:

`setScalingMode(int)`

isDoubleBuffered

`boolean isDoubleBuffered()`

Retorna *true* se o buffering duplo estiver disponível e ativado.

Retorna:

true se o buffering duplo estiver disponível e ativado, caso contrário, *false*

isOpaque

```
boolean isOpaque()
```

Retorna *true* se toda a área do componente (conforme retornado pelo método `com.sun.dtv.lwuit.Component#getBounds`, for opaco.

Retorna:

true se todos os pixels dentro da área determinada pelo método `com.sun.dtv.lwuit.Component#getBounds` forem opacos, ou seja, todos os pixels estiverem pintados com uma cor opaca, caso contrário, *false*.

processEvent

```
void processEvent(AWTEvent event)
```

Lida com o `AWTEvent` determinado.

Parâmetros:

`event` – um `java.awt.AWTEvent` para lidar.

51 Pacote com.sun.dtv.ui.event

51.1 Descrição do pacote

Subpacote de eventos de funcionalidades de UI específicas para TV.

A Figura 22 mostra a estrutura do pacote `UI Event` do Java DTV.

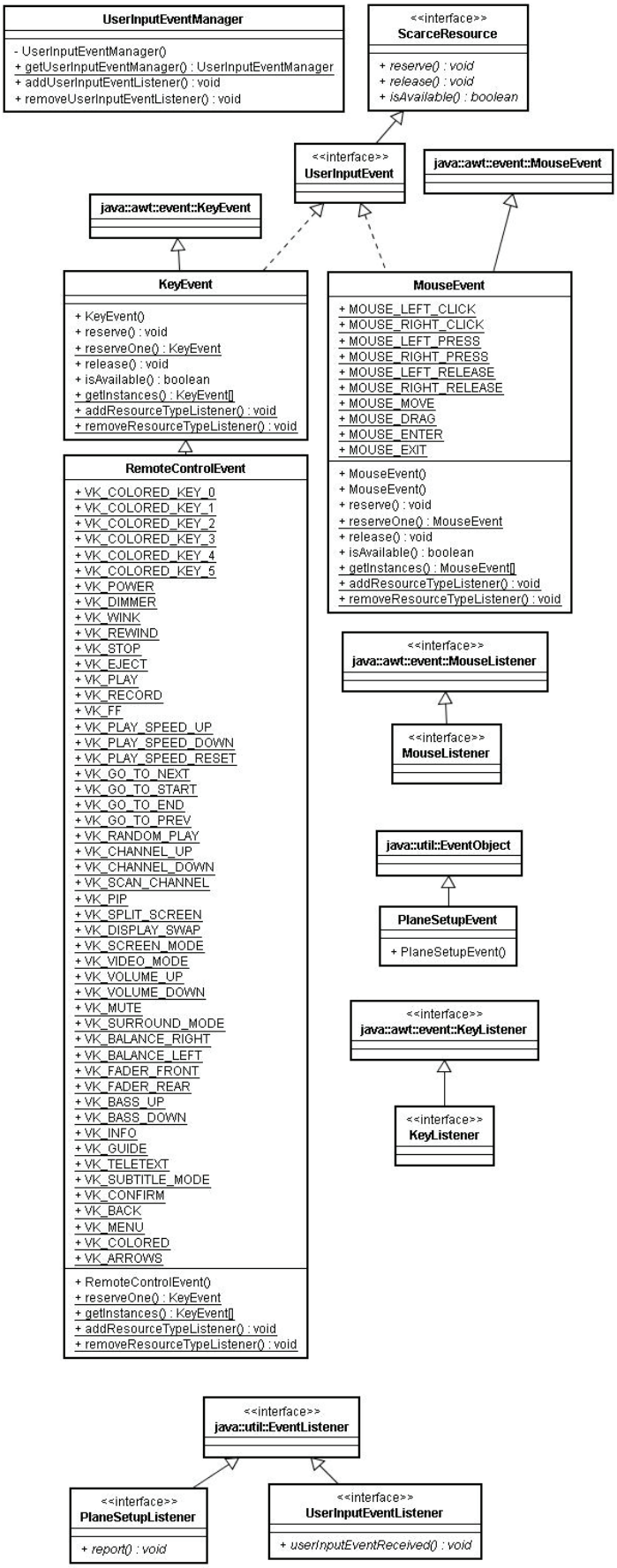


Figura 22 – Pacote `User Interface Event`

NOTA Este pacote está especificado desde o Java DTV 1.0.

51.2 Índice de interfaces

KeyListener

Este *listener* é apenas um *wrapper* para introduzir `java.awt.event.KeyListener` na API, como o LWUIT não suporta objetos de evento de tecla e, portanto, fornece um `ActionListener`.

MouseListener

Este *listener* é apenas um *wrapper* para introduzir `java.awt.event.MouseListener` na API, como o LWUIT não suporta objetos de evento de mouse e, portanto, fornece um `ActionListener`.

PlaneSetupListener

Este *listener* é utilizado para monitorar quando a configuração de um plano é modificada.

UserInputEvent

Esta interface é uma abstração para eventos de input de usuário enviados a todos os consumidores registrados quando ocorre um input utilizando um `UserInputDevice`.

UserInputEventListener

Esta interface deve ser implementada pelas classes a fim de receber `UserInputEvents`.

51.3 Índice de classes

KeyEvent

Esta classe estende a classe `KeyEvent` por meio da implementação da interface de `UserInputEvent`.

MouseEvent

Esta classe estende a classe `MouseEvent` por meio da implementação da interface de `UserInputEvent`.

PlaneSetupEvent

Este evento é enviado para todos os `PlaneSetupListeners` registrados quando um plano modifica sua configuração.

RemoteControlEvent

Esta classe estende a classe `KeyEvent` adicionando códigos de tecla específicos do controle remoto.

UserInputEventManager

As instâncias desta classe são os gerenciadores de evento que manipulam os eventos de input de usuário a serem manipulados por um *widget*.

51.4 Classe KeyEvent

51.4.1 Descrição da classe

`com.sun.dtv.ui.event`

```
java.lang.Object
  └java.util.EventObject
      └java.awt.AWTEvent
          └java.awt.event.ComponentEvent
              └java.awt.event.InputEvent
```

```
└java.awt.event.KeyEvent
    └com.sun.dtv.ui.event.KeyEvent
```

Todas as interfaces implementadas

`ScarceResource, Serializable, UserInputEvent`

Subclasses diretas conhecidas

`RemoteControlEvent`

```
public class KeyEvent
extends KeyEvent
implements UserInputEvent
```

Esta classe estende a classe `KeyEvent` por meio da implementação da interface de `UserInputEvent`. Embora seja apenas um marcador de interface, o `UserInputEvent` é derivado da interface `ScarceResource`, portanto, seus métodos devem ser implementados aqui.

Relaciona-se com:

`ScarceResource.reserve()`, `ScarceResource.release()`

51.4.2 Índice de construtores

`KeyEvent(Component source, int id, long when, int modifiers, int keyCode, char keyChar)`

Constrói um objeto `KeyEvent`.

51.4.3 Índice de métodos

`static void addResourceTypeListener(ResourceTypeListener listener)`

Adiciona um `ResourceTypeListener` a implementação.

`static KeyEvent[] getInstances()`

Retorna a lista de todas as intâncias de `KeyEvent` existentes, estejam reservadas ou não.

`boolean isAvailable()`

Verifica se o dado recurso está correntemente disponível para reserva.

`void release()`

Libera esse recurso.

`static void removeResourceTypeListener(ResourceTypeListener listener)`

Remove um *listener* previamente anexado.

`void reserve(boolean force, long timeout, ScarceResourceListener listener)`

Requisita reserva da dada instância de recursos escassos.

`static KeyEvent reserveOne(int keyCode, boolean force, long timeoutms, ScarceResourceListener listener)`

Retorna uma instância reservada fora do *pool* de todas as intâncias `KeyEvent` que correspondem ao código de tecla especificado.

51.4.4 Detalhe dos construtores

KeyEvent

```
public KeyEvent(Component source,
                int id,
                long when,
                int modifiers,
                int keyCode,
                char keyChar)
```

Constrói um objeto `KeyEvent`.

Parâmetros:

`source` – componente que originou o evento

`id` – um número inteiro identificando o tipo de evento

`when` – um número inteiro longo que especifica o tempo em que o evento ocorreu

`modifiers` – teclas modificadoras durante o evento (shift, ctrl, alt, meta). Tanto os modificadores estendedidos `_DOWN_MASK` quanto o `old _MASK` devem ser utilizados, mas nenhum dos modelos deve ser misturado em um evento. Utilizar os modificadores estendidos é preferido.

`keyCode` – código de número inteiro para uma tecla real, ou `VK_UNDEFINED` (para um evento *key-typed*) (como definido em `java.awt.event.KeyEvent`)

`keyChar` – caractere Unicode gerado por este evento, ou `CHAR_UNDEFINED` (para eventos *key-pressed* e *key-released* que não mapeiam para um caractere *Unicode* válido) (como definido em `java.awt.event.KeyEvent`)

Relaciona-se com:

`KeyEvent`

51.4.5 Detalhe dos métodos

reserve

```
public void reserve(boolean force,
                    long timeout,
                    ScarceResourceListener listener)
             throws IllegalArgumentException,
                    TimeoutException
```

Requisita reserva da dada instância de recursos escassos. O método bloquear-se-á até que a instância esteja disponível. O método retorna normalmente apenas se a reserva for bem sucedida. Todos os outros casos são manuseados por exceções.

Primeiro, se houver um gerenciador de segurança, seu método `checkPermission` é chamado com a permissão `ScarceResourcePermission(name, "reserve")`. Se `force` é além disso definido *`true`*, a permissão também verificado no `ScarceResourcePermission(name, "force")`.

Durante o processo de reserva, se a instância do recurso já estiver alocada, a implementação deve notificar o proprietário atual do recurso sobre a requisição de reserva por meio da interface `ScarceResourceListener`:

- ou por `ScarceResourceListener.releaseRequested()` se forçar for *false*,
- ou por `ScarceResourceListener.releaseForced()` no outro caso.

O *listener* deve ser usado para tais notificações apenas até o recurso ser liberado. Após a liberação, a implementação não deve chamar quaisquer dos métodos da interface do *listener*.

Após esse primeiro evento, a implementação procederá de acordo e liberará (ou não) o recurso requisitado. No

caso da implementação liberar o recurso, desencadeará um evento `ScarceResourceListener.released()` ao *listener* proprietário original do recurso informando-o que o recurso não lhe pertence mais.

O aplicativo pode controlar o tempo de espera para que tal recurso esteja disponível ao definir `timeoutms`. No caso da duração da reserva exceder o tempo expresso em `timeoutms`, um `TimeoutException` é lançado.

Sob operação normal, recursos são liberados usando o método `release`. De qualquer forma, no caso do aplicativo não liberar recursos quando requisitado ou o aplicativo ser terminado, a implementação deve liberar todos os recursos alocados para o aplicativo para permitir que outros aplicativos sejam notificados de mudanças na alocação de recursos e poderem reservá-los. Ver a seção Resource Cleanup do ciclo de vida do aplicativo.

Especificado por:

`reserve` na interface `ScarceResource`

Parâmetros:

`force` - Se *`true`*, esse método retirará o recurso do proprietário atual. Se *`false`*, a implementação bloquear-se-á e esperará até que o recurso esteja disponível (usando `release()`) ou até `timeoutms`.

`listener` - O *listener* a ser anexado para receber notificação sobre o status do recurso.

Lança:

`IllegalArgumentException` - Se o *listener* estiver `null` ou se o valor especificado em `timeoutms` não for válido, isto é, nem um inteiro positivo, nem -1.

`TimeoutException` - Se o tempo especificado em `timeoutms` houver acabado e o recurso não houver podido ser reservado.

reserveOne

```
public static KeyEvent reserveOne(int keyCode,
                                  boolean force,
                                  long timeoutms,
                                  ScarceResourceListener listener)
                           throws IllegalArgumentException,
                                  TimeoutException
```

Retorna uma instância reservada fora do pool de todas as intâncias `KeyEvent` que correspondem ao código de tecla especificado. Esse método retorna com a instância ou executa uma exceção de acordo com a situação.

Esse método se comporta exatamente como o método `reserve()`.

Parâmetros:

`keyCode` – código de número inteiro para uma tecla real, ou `VK_UNDEFINED` (para um evento key-typed) (como definido em `java.awt.event.KeyEvent`)

`force` - Se *`true`*, esse método tem permissão de retirar qualquer recurso do atual proprietário. Se *`false`*, a implementação bloqueará e esperará até que um recurso de um dado `type` esteja disponível (utilizando `release()`) ou até milissegundos `timeoutms`

`timeoutms` – Uma quantidade positiva de tempo em milissegundos até a qual este método esperará para que qualquer recurso seja liberado por seu proprietário atual. O valor `-1` indica que a implementação esperará para sempre

`listener` - O *listener* a ser anexado para receber notificação sobre o status do recurso.

Retorna:

A instância de tipo `KeyEvent` que foi reservada.

Lança:

`IllegalArgumentException` - Se o *listener* estiver `null` ou se o valor especificado em `timeoutms` não for válido, isto é, nem um inteiro positivo, nem -1.

`TimeoutException` - Se o tempo especificado em `timeoutms` houver acabado e o recurso não houver podido ser reservado.

Relaciona-se com:

`reserve()`

release

```
public void release()
```

Libera esse recurso.

O recurso deve ser disponibilizado para outro aplicativo através da plataforma. Após chamar esse método, não é mais possível interagir com o dado recurso: chamadas para métodos críticos desses recursos escassos devem ser ignorados e lançar `IllegalStateException`. Essa afirmação é válida e é o comportamento requerido de qualquer classe implementando a interface `ScarceResource`. Para interagir novamente com o dado recurso, o aplicativo deve chamar o método `reserve()` e se tornar o proprietário novamente.

A implementação pode organizar quaisquer recursos do sistema que esse objeto estiver usando. Depois a implementação não deve chamar quaisquer dos métodos do *listener* que estava anexado no momento da reserva.

Se o recurso já estava disponível (isto é, não reservado), esse método não faz nada.

Especificado por:

`release` na interface `ScarceResource`

isAvailable

```
public boolean isAvailable()
```

Verifica se o dado recurso está correntemente disponível para reserva. O valor retornado dá a situação atual e não garante que o recurso ainda estará disponível em um momento posterior, por exemplo, no momento da reserva: outro aplicativo pode ter tomado aquele recurso no tempo decorrido.

Especificado por:

`isAvailable` na interface `ScarceResource`

Retorna:

Um boolean definido *`true`* se o dado recurso estiver atualmente disponível para reserva.

getInstances

```
public static KeyEvent[] getInstances()
```

Retorna a lista de todas as intâncias de `KeyEvent` existentes, estejam reservadas ou não.

Retorna:

lista de todas as instâncias de plano existentes.

addResourceTypeListener

```
public static void addResourceTypeListener(ResourceTypeListener listener)
                                    throws NullPointerException
```

Adiciona um `ResourceTypeListener` a implementação. Sempre que um `reserve()` ou um `release()` for chamado para quaisquer recursos do mesmo tipo, a implementação chamará os métodos correspondentes do *listener*.

Parâmetros:

`listener` - O *listener* desencadeado para eventos em recursos do mesmo tipo.

Lança:

`NullPointerException` - Se o *listener* estiver `null`.

Relaciona-se com:

`removeResourceTypeListener()`

removeResourceTypeListener

```
public static void removeResourceTypeListener(ResourceTypeListener listener)
                                        throws NullPointerException
```

Remove um *listener* previamente anexado. Se o *listener* não foi anexado antes, esse método não faz nada..

Parâmetros:

`listener` - O *listener* desencadeado para eventos em recursos do mesmo tipo.

Lança:

`NullPointerException` - Se o *listener* estiver `null`.

Relaciona-se com:

`addResourceTypeListener()`

51.5 Interface KeyListener

51.5.1 Descrição da interface

`com.sun.dtv.ui.event`

Todas as super-interfaces

`EventListener`, `KeyListener`

```
public interface KeyListener
extends KeyListener
```

Este *listener* é apenas um *wrapper* para introduzir `java.awt.event.KeyListener` na API, como o LWUIT não suporta objetos de evento de tecla e, portanto, fornece um `ActionListener`.

Relaciona-se com:

`KeyEvent`

51.6 Classe MouseEvent

51.6.1 Descrição da classe

`com.sun.dtv.ui.event`

```
java.lang.Object
  └java.util.EventObject
      └java.awt.AWTEvent
          └java.awt.event.ComponentEvent
```

```
java.awt.event.InputEvent
    java.awt.event.MouseEvent
        com.sun.dtv.ui.event.MouseEvent
```

Todas as interfaces implementadas

`ScarceResource, Serializable, UserInputEvent`

```
public class MouseEvent
extends MouseEvent
implements UserInputEvent
```

Esta classe estende a classe `MouseEvent` por meio da implementação da interface de `UserInputEvent`. Embora seja apenas um marcador de interface, o `UserInputEvent` é derivado da interface `ScarceResource`, portanto, seus métodos devem ser implementados aqui.

Relaciona-se com:

`ScarceResource.reserve(), ScarceResource.release()`

51.6.2 Índice de campos

`static int MOUSE_DRAG`

Constante utilizada para especificação de um código de mouse.

`static int MOUSE_ENTER`

Constante utilizada para especificação de um código de mouse.

`static int MOUSE_EXIT`

Constante utilizada para especificação de um código de mouse.

`static int MOUSE_LEFT_CLICK`

Constante utilizada para especificação de um código de mouse.

`static int MOUSE_LEFT_PRESS`

Constante utilizada para especificação de um código de mouse.

`static int MOUSE_LEFT_RELEASE`

Constante utilizada para especificação de um código de mouse.

`static int MOUSE_MOVE`

Constante utilizada para especificação de um código de mouse.

`static int MOUSE_RIGHT_CLICK`

Constante utilizada para especificação de um código de mouse.

`static int MOUSE_RIGHT_PRESS`

Constante utilizada para especificação de um código de mouse.

`static int MOUSE_RIGHT_RELEASE`

Constante utilizada para especificação de um código de mouse.

51.6.3 Índice de construtores

`MouseEvent(Component source, int id, long when, int modifiers, int x, int y, int clickCount, boolean popupTrigger)`

Constrói um objeto `MouseEvent`.

`MouseEvent(Component source, int id, long when, int modifiers, int x, int y, int clickCount, boolean popupTrigger, int button)`

Constrói um objeto `MouseEvent`.

51.6.4 Índice de métodos

`static void addResourceTypeListener(ResourceTypeListener listener)`

Adiciona um `ResourceTypeListener` a implementação.

`static MouseEvent[] getInstances()`

Retorna a lista de todas as intâncias de `MouseEvent` existentes, estejam reservadas ou não.

`boolean isAvailable()`

Verifica se o dado recurso está correntemente disponível para reserva.

`void release()`

Libera esse recurso.

`static void removeResourceTypeListener(ResourceTypeListener listener)`

Remove um *listener* previamente anexado.

`void reserve(boolean force, long timeout, ScarceResourceListener listener)`

Requisita reserva da dada instância de recursos escassos.

`static MouseEvent reserveOne(boolean force, long timeoutms, ScarceResourceListener listener, int id)`

Retorna uma instância reservada fora do pool de todas as intâncias `MouseEvent`.

51.6.5 Detalhe dos campos

MOUSE_LEFT_CLICK

`public static int MOUSE_LEFT_CLICK`

Constante utilizada para especificação de um código de mouse. Indica que o botão esquerdo do mouse é clicado. Valor 521.

MOUSE_RIGHT_CLICK

`public static int MOUSE_RIGHT_CLICK`

Constante utilizada para especificação de um código de mouse. Indica que o botão direito do mouse é clicado. Valor 522.

MOUSE_LEFT_PRESS

`public static int MOUSE_LEFT_PRESS`

Constante utilizada para especificação de um código de mouse. Indica que o botão esquerdo do mouse está pressionado. Valor 523.

MOUSE_RIGHT_PRESS

```
public static int MOUSE_RIGHT_PRESS
```

Constante utilizada para especificação de um código de mouse. Indica que o botão direito do mouse está pressionado. Valor 524.

MOUSE_LEFT_RELEASE

```
public static int MOUSE_LEFT_RELEASE
```

Constante utilizada para especificação de um código de mouse. Indica que o botão esquerdo do mouse é liberado. Valor 525.

MOUSE_RIGHT_RELEASE

```
public static int MOUSE_RIGHT_RELEASE
```

Constante utilizada para especificação de um código de mouse. Indica que o botão direito do mouse é liberado. Valor 526.

MOUSE_MOVE

```
public static int MOUSE_MOVE
```

Constante utilizada para especificação de um código de mouse. Indica que o mouse é movimentado. Valor 527.

MOUSE_DRAG

```
public static int MOUSE_DRAG
```

Constante utilizada para especificação de um código de mouse. Indica que o mouse é arrastado. Valor 528.

MOUSE_ENTER

```
public static int MOUSE_ENTER
```

Constante utilizada para especificação de um código de mouse. Indica que o cursor do mouse entra na área visível da tela controlada por este mouse. Valor 529.

MOUSE_EXIT

```
public static int MOUSE_EXIT
```

Constante utilizada para especificação de um código de mouse. Indica que o cursor do mouse sai da área visível da tela controlada por este mouse. Valor 530.

51.6.6 Detalhe dos construtores

MouseEvent

```
public MouseEvent(Component source,
                  int id,
                  long when,
                  int modifiers,
                  int x,
```

```
                  int y,
                  int clickCount,
                  boolean popupTrigger)
```

Constrói um objeto `MouseEvent`.

Parâmetros:

`source` – componente que originou o evento

`id` – número inteiro que identifica o evento

`when` – um número inteiro longo que dá o tempo em que o evento ocorreu

`modifiers` – teclas modificadoras durante o evento (por exemplo shift, ctrl, alt, meta). Tanto os modificadores `_DOWN_MASK` estendido quanto o antigo `_MASK` devem ser utilizados, mas nenhum dos modelos deve ser misturado em um evento. Utilizar os modificadores estendidos é preferido.

`x` – coordenada x horizontal para a localização do mouse

`y` – coordenada y vertical para a localização do mouse

`clickCount` – número de cliques do mouse associado ao evento

`popupTrigger` - um boolean, *true* se este evento for um acionador para um menu popup

Relaciona-se com:

`MouseEvent`

MouseEvent

```
public MouseEvent(Component source,
                  int id,
                  long when,
                  int modifiers,
                  int x,
                  int y,
                  int clickCount,
                  boolean popupTrigger,
                  int button)
```

Constrói um objeto `MouseEvent`.

Parâmetros:

`source` – componente que originou o evento

`id` – número inteiro que identifica o evento

`when` – um número inteiro longo que dá o tempo em que o evento ocorreu

`modifiers` – teclas modificadoras durante o evento (por exemplo shift, ctrl, alt, meta). Tanto os modificadores `_DOWN_MASK` estendidos quanto o antigo `_MASK` devem ser utilizados, mas nenhum dos modelos deve ser misturado em um evento. Utilizar os modificadores estendidos é preferido.

`x` – coordenada x horizontal para a localização do mouse

`y` – coordenada y vertical para a localização do mouse

`clickCount` – número de cliques do mouse associado ao evento

`popupTrigger` - um boolean, *true* se este evento for um acionador para um menu popup

`button` – botões do mouse que apresentam estado alterado. `NOBUTTON`, `BUTTON1`, `BUTTON2` ou `BUTTON3` (como definido em `java.awt.event.MouseEvent`).

Relaciona-se com:

`MouseEvent`

51.6.7 Detalhe dos métodos

reserve

```
public void reserve(boolean force,
                    long timeout,
                    ScarceResourceListener listener)
             throws IllegalArgumentException,
                    TimeoutException
```

Requisita reserva da dada instância de recursos escassos. O método bloquear-se-á até que a instância esteja disponível. O método retorna normalmente apenas se a reserva for bem sucedida. Todos os outros casos são manuseados por exceções.

Primeiro, se houver um gerenciador de segurança, seu método `checkPermission` é chamado com a permissão `ScarceResourcePermission(name, "reserve")`. Se `force` é além disso definido *`true`*, a permissão também verificado no `ScarceResourcePermission(name, "force")`.

Durante o processo de reserva, se a instância do recurso já estiver alocada, a implementação deve notificar o proprietário atual do recurso sobre a requisição de reserva por meio da interface `ScarceResourceListener`:

- ou por `ScarceResourceListener.releaseRequested()` se forçar for *false*,
- ou por `ScarceResourceListener.releaseForced()` no outro caso.

O *listener* deve ser usado para tais notificações apenas até o recurso ser liberado. Após a liberação, a implementação não deve chamar quaisquer dos métodos da interface do *listener*.

Após esse primeiro evento, a implementação procederá de acordo e liberará (ou não) o recurso requisitado. No caso da implementação liberar o recurso, desencadeará um evento `ScarceResourceListener.released()` ao *listener* proprietário original do recurso informando-o que o recurso não lhe pertence mais.

O aplicativo pode controlar o tempo de espera para que tal recurso esteja disponível ao definir `timeoutms`. No caso da duração da reserva exceder o tempo expresso em `timeoutms`, um `TimeoutException` é lançado.

Sob operação normal, recursos são liberados usando o método `release`. De qualquer forma, no caso do aplicativo não liberar recursos quando requisitado ou o aplicativo ser terminado, a implementação deve liberar todos os recursos alocados para o aplicativo para permitir que outros aplicativos sejam notificados de mudanças na alocação de recursos e poderem reservá-los. Ver a seção Resource Cleanup do ciclo de vida do aplicativo.

Especificado por:

`reserve` na interface `ScarceResource`

Parâmetros:

`force` - Se *`true`*, esse método retirará o recurso do proprietário atual. Se *`false`*, a implementação bloquear-se-á e esperará até que o recurso esteja disponível (usando `release()`) ou até `timeoutms`.

`listener` - O *listener* a ser anexado para receber notificação sobre o status do recurso.

Lança:

`IllegalArgumentException` - Se o *listener* estiver `null` ou se o valor especificado em `timeoutms` não for válido, isto é, nem um inteiro positivo, nem -1.

`TimeoutException` - Se o tempo especificado em `timeoutms` houver acabado e o recurso não houver podido ser reservado.

reserveOne

```
public static MouseEvent reserveOne(boolean force,
                                    long timeoutms,
                                    ScarceResourceListener listener,
                                    int id)
                             throws IllegalArgumentException,
                                    TimeoutException
```

Retorna uma instância reservada fora do pool de todas as intâncias `MouseEvent`. Esse método retorna com a instância ou executa uma exceção de acordo com a situação.

Esse método se comporta exatamente como o método `reserve()`.

Parâmetros:

`force` - Se *`true`*, esse método tem permissão de retirar qualquer recurso do atual proprietário. Se *`false`*, a implementação bloqueará e esperará até que um recurso de um dado `type` esteja disponível (utilizando `release()`) ou até milissegundos `timeoutms`.

`timeoutms` – Uma quantidade positiva de tempo em milissegundos até a qual este método esperará para que qualquer recurso seja liberado por seu proprietário atual. O valor `-1` indica que a implementação esperará para sempre.

`listener` - O *listener* a ser anexado para receber notificação sobre o status do recurso.

`id` – número inteiro que identifica o evento

Retorna:

A instância de tipo `MouseEvent` com o identificador especificado que foi reservado.

Lança:

`IllegalArgumentException` - Se o *listener* estiver `null` ou se o valor especificado em `timeoutms` não for válido, isto é, nem um inteiro positivo, nem -1.

`TimeoutException` - Se o tempo especificado em `timeoutms` houver acabado e o recurso não houver podido ser reservado.

Relaciona-se com:

`reserve()`

release

```
public void release()
```

Libera esse recurso.

O recurso deve ser disponibilizado para outro aplicativo através da plataforma. Após chamar esse método, não é mais possível interagir com o dado recurso: chamadas para métodos críticos desses recursos escassos devem ser ignorados e lançar `IllegalStateException`. Essa afirmação é válida e é o comportamento requerido de qualquer classe implementando a interface `ScarceResource`. Para interagir novamente com o dado recurso, o aplicativo deve chamar o método `reserve()` e se tornar o proprietário novamente.

A implementação pode organizar quaisquer recursos do sistema que esse objeto estiver usando. Depois a implementação não deve chamar quaisquer dos métodos do *listener* que estava anexado no momento da reserva.

Se o recurso já estava disponível (isto é, não reservado), esse método não faz nada.

Especificado por:

`release` na interface `ScarceResource`

isAvailable

```
public boolean isAvailable()
```

Verifica se o dado recurso está correntemente disponível para reserva. O valor retornado dá a situação atual e não garante que o recurso ainda estará disponível em um momento posterior, por exemplo, no momento da reserva: outro aplicativo pode ter tomado aquele recurso no tempo decorrido.

Especificado por:

`isAvailable` na interface `ScarceResource`

Retorna:

Um boolean definido *`true`* se o dado recurso estiver atualmente disponível para reserva.

getInstances

```
public static MouseEvent[] getInstances()
```

Retorna a lista de todas as intâncias de `MouseEvent` existentes, estejam reservadas ou não.

Retorna:

lista de todas as instâncias do `MouseEvent` existentes.

addResourceTypeListener

```
public static void addResourceTypeListener(ResourceTypeListener listener)
                                    throws NullPointerException
```

Adiciona um `ResourceTypeListener` a implementação. Sempre que um `reserve()` ou um `release()` for chamado para quaisquer recursos do mesmo tipo, a implementação chamará os métodos correspondentes do *listener*.

Parâmetros:

`listener` - O *listener* desencadeado para eventos em recursos do mesmo tipo.

Lança:

`NullPointerException` - Se o *listener* estiver `null`.

Relaciona-se com:

`removeResourceTypeListener()`

removeResourceTypeListener

```
public static void removeResourceTypeListener(ResourceTypeListener listener)
                                       throws NullPointerException
```

Remove um *listener* previamente anexado. Se o *listener* não foi anexado antes, esse método não faz nada..

Parâmetros:

`listener` - O *listener* desencadeado para eventos em recursos do mesmo tipo.

Lança:

`NullPointerException` - Se o *listener* estiver `null`.

Relaciona-se com:

addResourceTypeListener()

51.7 Interface MouseListener

51.7.1 Descrição da interface

com.sun.dtv.ui.event

Todas as super-interfaces

EventListener, MouseListener

```
public interface MouseListener
extends MouseListener
```

Este *listener* é apenas um *wrapper* para introduzir java.awt.event.MouseListener na API, como o LWUIT não suporta objetos de evento de mouse e, portanto, fornece um ActionListener.

Relaciona-se com:

MouseEvent

51.8 Classe PlaneSetupEvent

51.8.1 Descrição da classe

com.sun.dtv.ui.event

```
java.lang.Object
  └java.util.EventObject
      └com.sun.dtv.ui.event.PlaneSetupEvent
```

Todas as interfaces implementadas

Serializable

```
public class PlaneSetupEvent
extends EventObject
```

Este evento é enviado para todos os PlaneSetupListeners registrados quando um plano modifica sua configuração.

51.8.2 Índice de construtores

PlaneSetupEvent(Object source)

Constrói um PlaneSetupEvent.

51.8.3 Detalhe dos construtores

PlaneSetupEvent

```
public PlaneSetupEvent(Object source)
```

Constrói um PlaneSetupEvent.

Parâmetros:

`source` - plano cuja configuração foi alterada

51.9 Interface PlaneSetupListener

51.9.1 Descrição da interface

`com.sun.dtv.ui.event`

Todas as super-interfaces

`EventListener`

```
public interface PlaneSetupListener
extends EventListener
```

Este *listener* é utilizado para monitorar quando a configuração de um plano é modificada.

Relaciona-se com:

`PlaneSetupEvent`

51.9.2 Índice de métodos

`void report(PlaneSetupEvent event)`

Este método é chamado quando a configuração de um plano é modificada.

51.9.3 Detalhe dos métodos

report

`void report(PlaneSetupEvent event)`

Este método é chamado quando a configuração de um plano é modificada.

Parâmetros:

`event` – evento notificando o *listener* da modificação.

51.10 Classe RemoteControlEvent

51.10.1 Descrição da classe

`com.sun.dtv.ui.event`

```
java.lang.Object
  └java.util.EventObject
      └java.awt.AWTEvent
          └java.awt.event.ComponentEvent
              └java.awt.event.InputEvent
                  └java.awt.event.KeyEvent
                      └com.sun.dtv.ui.event.KeyEvent
```

└com.sun.dtv.ui.event.RemoteControlEvent

Todas as interfaces implementadas

ScarceResource, Serializable, UserInputEvent

```
public class RemoteControlEvent
extends KeyEvent
```

Esta classe estende a classe KeyEvent adicionando códigos de tecla específicos do controle remoto.

51.10.2 Índice de campos

static int VK_ARROWS

Qualquer tecla de seta.

static int VK_BACK

Código de tecla de retorno.

static int VK_BALANCE_LEFT

Código de tecla esquerda de balanço.

static int VK_BALANCE_RIGHT

Código de tecla direita de balanço.

static int VK_BASS_DOWN

Código de tecla de diminuição do grave.

static int VK_BASS_UP

Código de tecla de aumento do grave.

static int VK_CHANNEL_DOWN

Código de tecla de redução de canal.

static int VK_CHANNEL_UP

Código de tecla de aumento de canal.

static int VK_COLORED

Qualquer tecla colorida.

static int VK_COLORED_KEY_0

Código de tecla de tecla 0 colorida.

static int VK_COLORED_KEY_1

Código de tecla de tecla 1 colorida.

static int VK_COLORED_KEY_2

Código de tecla de tecla 2 colorida.

static int VK_COLORED_KEY_3

Código de tecla de tecla 3 colorida.

static int VK_COLORED_KEY_4

Código de tecla de tecla 4 colorida.

static int VK_COLORED_KEY_5

Código de tecla de tecla 5 colorida.

`static int VK_CONFIRM`

Código de tecla de confirmação.

`static int VK_DIMMER`

Código de tecla de dispositivo dimmer.

`static int VK_DISPLAY_SWAP`

Código de tecla de troca de display.

`static int VK_EJECT`

Código de tecla de botão de ejetar.

`static int VK_FADER_FRONT`

Código de tecla fader frontal.

`static int VK_FADER_REAR`

Código de tecla fader traseira.

`static int VK_FF`

Código de tecla de botão FF (avançar rápido)

`static int VK_GO_TO_END`

Código de tecla ir para posição final.

`static int VK_GO_TO_NEXT`

Código de tecla ir para próxima posição.

`static int VK_GO_TO_PREV`

Código de tecla ir para posição anterior.

`static int VK_GO_TO_START`

Código de tecla ir para posição inicial.

`static int VK_GUIDE`

Código de tecla de guia de programa

`static int VK_INFO`

Código de tecla info.

`static int VK_MENU`

Código de tecla de menu.

`static int VK_MUTE`

Código de tecla mute.

`static int VK_PIP`

Código de tecla PIP (picture in picture).

`static int VK_PLAY`

Código de tecla de botão play (PLAY).

`static int VK_PLAY_SPEED_DOWN`

Código de tecla reduzir velocidade de mídia.

`static int VK_PLAY_SPEED_RESET`

Código de tecla redefinir velocidade de mídia.

`static int VK_PLAY_SPEED_UP`

Código de tecla aumentar velocidade de mídia.

`static int VK_POWER`

Código de tecla ligar/desligar dispositivo.

`static int VK_RANDOM_PLAY`

código de tecla de tocar em modo aleatório

`static int VK_RECORD`

Código de tecla de botão gravar (REC).

`static int VK_REWIND`

Código de tecla de botão voltar (REW).

`static int VK_SCAN_CHANNEL`

Código de tecla buscar canal.

`static int VK_SCREEN_MODE`

Código de tecla mudar modo de tela.

`static int VK_SPLIT_SCREEN`

Código de tecla dividir tela.

`static int VK_STOP`

Código de tecla de botão parar (STOP).

`static int VK_SUBTITLE_MODE`

Código de tecla mudar modo legenda.

`static int VK_SURROUND_MODE`

Código de tecla mudar modo surround.

`static int VK_TELETEXT`

Código de tecla teletext.

`static int VK_VIDEO_MODE`

Código de tecla mudar modo de vídeo.

`static int VK_VOLUME_DOWN`

Código de tecla de diminuição do volume.

`static int VK_VOLUME_UP`

Código de tecla de aumento do volume.

`static int VK_WINK`

Código de tecla wink de dispositivo.

51.10.3 Índice de construtores

`RemoteControlEvent(Component source, int id, long when, int modifiers, int keyCode, char keyChar)`

Constrói um objeto `RemoteControlEvent`.

51.10.4 Índice de métodos

`static void addResourceTypeListener(ResourceTypeListener listener)`

Adiciona um `ResourceTypeListener` a implementação.

`static KeyEvent[] getInstances()`

Retorna a lista de todas as intâncias de `RemoteControlEvent` existentes, estejam reservadas ou não.

`static void removeResourceTypeListener(ResourceTypeListener listener)`

Remove um *listener* previamente anexado.

```
static KeyEvent reserveOne(int keyCode, boolean force, long timeoutms,
ScarceResourceListener listener)
```

Retorna uma instância reservada fora do pool de todas as intâncias `RemoteControlEvent`.

Métodos herdados da classe `com.sun.dtv.ui.event.KeyEvent`

```
addResourceTypeListener, getInstances, isAvailable, release,
removeResourceTypeListener, reserve, reserveOne
```

51.10.5 Detalhe dos campos

VK_COLORED_KEY_0

`public static final int VK_COLORED_KEY_0 = 33`

Código de tecla de tecla 0 colorida. Um controle remoto pode ter até seis teclas coloridas.

VK_COLORED_KEY_1

`public static final int VK_COLORED_KEY_1 = 34`

Código de tecla de tecla 1 colorida. Um controle remoto pode ter até seis teclas coloridas.

VK_COLORED_KEY_2

`public static final int VK_COLORED_KEY_2 = 35`

Código de tecla de tecla 2 colorida. Um controle remoto pode ter até seis teclas coloridas.

VK_COLORED_KEY_3

`public static final int VK_COLORED_KEY_3 = 36`

Código de tecla de tecla 3 colorida. Um controle remoto pode ter até seis teclas coloridas.

VK_COLORED_KEY_4

`public static final int VK_COLORED_KEY_4 = 37`

Código de tecla de tecla 4 colorida. Um controle remoto pode ter até seis teclas coloridas.

VK_COLORED_KEY_5

`public static final int VK_COLORED_KEY_5 = 38`

Código de tecla de tecla 5 colorida. Um controle remoto pode ter até seis teclas coloridas.

VK_POWER

```
public static final int VK_POWER = 39
```

Código de tecla ligar/desligar dispositivo.

VK_DIMMER

```
public static final int VK_DIMMER = 40
```

Código de tecla de dispositivo dimmer.

VK_WINK

```
public static final int VK_WINK = 41
```

Código de tecla wink de dispositivo.

VK_REWIND

```
public static final int VK_REWIND = 42
```

Código de tecla de botão voltar (REW).

VK_STOP

```
public static final int VK_STOP = 43
```

Código de tecla de botão parar (STOP).

VK_EJECT

```
public static final int VK_EJECT = 44
```

Código de tecla de botão de ejetar.

VK_PLAY

```
public static final int VK_PLAY = 45
```

Código de tecla de botão play (PLAY).

VK_RECORD

```
public static final int VK_RECORD = 46
```

Código de tecla de botão gravar (REC).

VK_FF

```
public static final int VK_FF = 47
```

Código de tecla de botão FF (avançar rápido)

VK_PLAY_SPEED_UP

```
public static final int VK_PLAY_SPEED_UP = 48
```

Código de tecla aumentar velocidade de mídia.

VK_PLAY_SPEED_DOWN

```
public static final int VK_PLAY_SPEED_DOWN = 49
```

Código de tecla reduzir velocidade de mídia.

VK_PLAY_SPEED_RESET

```
public static final int VK_PLAY_SPEED_RESET = 50
```

Código de tecla redefinir velocidade de mídia.

VK_GO_TO_NEXT

```
public static final int VK_GO_TO_NEXT = 51
```

Código de tecla ir para próxima posição.

VK_GO_TO_START

```
public static final int VK_GO_TO_START = 52
```

Código de tecla ir para posição inicial.

VK_GO_TO_END

```
public static final int VK_GO_TO_END = 53
```

Código de tecla ir para posição final.

VK_GO_TO_PREV

```
public static final int VK_GO_TO_PREV = 54
```

Código de tecla ir para posição anterior.

VK_RANDOM_PLAY

```
public static final int VK_RANDOM_PLAY = 55
```

código de tecla de tocar em modo aleatório

VK_CHANNEL_UP

```
public static final int VK_CHANNEL_UP = 56
```

Código de tecla de aumento de canal.

VK_CHANNEL_DOWN

```
public static final int VK_CHANNEL_DOWN = 57
```

Código de tecla de redução de canal.

VK_SCAN_CHANNEL

```
public static final int VK_SCAN_CHANNEL = 58
```

Código de tecla buscar canal.

VK_PIP

```
public static final int VK_PIP = 59
```

Código de tecla PIP (picture in picture).

VK_SPLIT_SCREEN

```
public static final int VK_SPLIT_SCREEN = 60
```

Código de tecla dividir tela.

VK_DISPLAY_SWAP

```
public static final int VK_DISPLAY_SWAP = 61
```

Código de tecla de troca de display.

VK_SCREEN_MODE

```
public static final int VK_SCREEN_MODE = 62
```

Código de tecla mudar modo de tela.

VK_VIDEO_MODE

```
public static final int VK_VIDEO_MODE = 63
```

Código de tecla mudar modo de vídeo.

VK_VOLUME_UP

```
public static final int VK_VOLUME_UP = 64
```

Código de tecla de aumento do volume.

VK_VOLUME_DOWN

```
public static final int VK_VOLUME_DOWN = 65
```

Código de tecla de diminuição do volume.

VK_MUTE

```
public static final int VK_MUTE = 66
```

Código de tecla mute.

VK_SURROUND_MODE

```
public static final int VK_SURROUND_MODE = 67
```

Código de tecla mudar modo surround.

VK_BALANCE_RIGHT

```
public static final int VK_BALANCE_RIGHT = 68
```

Código de tecla direita de balanço.

VK_BALANCE_LEFT

```
public static final int VK_BALANCE_LEFT = 69
```

Código de tecla esquerda de balanço.

VK_FADER_FRONT

```
public static final int VK_FADER_FRONT = 70
```

Código de tecla fader frontal.

VK_FADER_REAR

```
public static final int VK_FADER_REAR = 71
```

Código de tecla fader traseira.

VK_BASS_UP

```
public static final int VK_BASS_UP = 72
```

Código de tecla de aumento do grave.

VK_BASS_DOWN

```
public static final int VK_BASS_DOWN = 73
```

Código de tecla de diminuição do grave.

VK_INFO

```
public static final int VK_INFO = 74
```

Código de tecla info.

VK_GUIDE

```
public static final int VK_GUIDE = 75
```

Código de tecla de guia de programa

VK_TELETEXT

```
public static final int VK_TELETEXT = 76
```

Código de tecla teletext.

VK_SUBTITLE_MODE

```
public static final int VK_SUBTITLE_MODE = 77
```

Código de tecla mudar modo legenda.

VK_CONFIRM

```
public static final int VK_CONFIRM = 80
```

Código de tecla de confirmação.

VK_BACK

```
public static final int VK_BACK = 81
```

Código de tecla de retorno.

VK_MENU

```
public static final int VK_MENU = 82
```

Código de tecla de menu.

VK_COLORED

```
public static final int VK_COLORED = 96
```

Qualquer tecla colorida.

VK_ARROWS

```
public static final int VK_ARROWS = 97
```

Qualquer tecla de seta.

51.10.6 Detalhe dos construtores

RemoteControlEvent

```
public RemoteControlEvent(Component source,
                          int id,
                          long when,
                          int modifiers,
                          int keyCode,
                          char keyChar)
```

Constrói um objeto `RemoteControlEvent`.

Parâmetros:

`source` – componente que originou o evento

`id` – um número inteiro identificando o tipo de evento

`when` – um número inteiro longo que especifica o tempo em que o evento ocorreu

`modifiers` – teclas modificadoras durante o evento (shift, ctrl, alt, meta). Tanto os modificadores `_DOWN_MASK` estendidos quanto o antigo `_MASK` devem ser utilizados, mas nenhum dos modelos deve ser misturado em um evento. Utilizar os modificadores estendidos é preferido.

keyCode – código de número inteiro para uma tecla real, ou VK_UNDEFINED (para um evento *key-typed*) (como definido em java.awt.event.KeyEvent)

keyChar – caractere Unicode gerado por este evento, ou CHAR_UNDEFINED (para eventos de *key-pressed* e *key-released* que não mapeiam para um caractere Unicode válido) (como definido em java.awt.event.KeyEvent)

Relaciona-se com:

KeyEvent

51.10.7 Detalhe dos métodos

reserveOne

```
public static KeyEvent reserveOne(int keyCode,
                                  boolean force,
                                  long timeoutms,
                                  ScarceResourceListener listener)
                           throws IllegalArgumentException,
                                  TimeoutException
```

Retorna uma instância reservada fora do pool de todas as intâncias RemoteControlEvent. Esse método retorna com a instância ou executa uma exceção de acordo com a situação.

Este método se comporta exatamente como o método reserve(). Esse método substitui aquele definido em KeyEvent.

Parâmetros:

keyCode – código de número inteiro para uma tecla real, ou VK_UNDEFINED (para um evento *key-typed*) (como definido em java.awt.event.KeyEvent)

force - Se *true*, esse método tem permissão de retirar qualquer recurso do atual proprietário. Se *false*, a implementação bloqueará e esperará até que um recurso de um dado type esteja disponível (utilizando release()) ou até milissegundos timeoutms.

timeoutms – Uma quantidade positiva de tempo em milissegundos até a qual este método esperará para que qualquer recurso seja liberado por seu proprietário atual. O valor -1 indica que a implementação esperará para sempre.

listener - O *listener* a ser anexado para receber notificação sobre o status do recurso.

Retorna:

A instância de tipo RemoteControlEvent que foi reservada.

Lança:

IllegalArgumentException - Se o *listener* estiver null ou se o valor especificado em timeoutms não for válido, isto é, nem um inteiro positivo, nem -1.

TimeoutException - Se o tempo especificado em timeoutms houver acabado e o recurso não houver podido ser reservado.

Relaciona-se com:

KeyEvent.reserve()

getInstances

```
public static KeyEvent[] getInstances()
```

Retorna a lista de todas as intâncias de RemoteControlEvent existentes, estejam reservadas ou não. Esse método substitui aquele definido em KeyEvent.

Retorna:

lista de todas as instâncias do `RemoteControlEvent` existentes

addResourceTypeListener

```
public static void addResourceTypeListener(ResourceTypeListener listener)
                                    throws NullPointerException
```

Adiciona um `ResourceTypeListener` a implementação. Sempre que um `reserve()` ou um `release()` for chamado para quaisquer recursos do mesmo tipo, a implementação chamará os métodos correspondentes do *listener*.

Parâmetros:

`listener` - O *listener* desencadeado para eventos em recursos do mesmo tipo.

Lança:

`NullPointerException` - Se o *listener* estiver `null`.

Relaciona-se com:

`removeResourceTypeListener()`

removeResourceTypeListener

```
public static void removeResourceTypeListener(ResourceTypeListener listener)
                                       throws NullPointerException
```

Remove um *listener* previamente anexado. Se o *listener* não foi anexado antes, esse método não faz nada..

Parâmetros:

`listener` - O *listener* desencadeado para eventos em recursos do mesmo tipo.

Lança:

`NullPointerException` - Se o *listener* estiver `null`.

Relaciona-se com:

`addResourceTypeListener()`

51.11 Interface UserInputEvent

51.11.1 Descrição da interface

`com.sun.dtv.ui.event`

Todas as super-interfaces

`ScarceResource`

Todas as classes implementadoras conhecidas

`KeyEvent, MouseEvent, RemoteControlEvent`

```
public interface UserInputEvent
extends ScarceResource
```

Esta interface é uma abstração para eventos de input de usuário enviados a todos os consumidores registrados quando ocorre um input utilizando um `UserInputDevice`.

Métodos herdados da interface `com.sun.dtv.resources.`**`ScarceResource`**

`isAvailable, release, reserve`

51.12 Interface UserInputEventListener

51.12.1 Descrição da interface

`com.sun.dtv.ui.event`

Todas as super-interfaces

`EventListener`

```
public interface UserInputEventListener
extends EventListener
```

Esta interface deve ser implementada pelas classes a fim de receber `UserInpitEvents`. A instância da classe de implementação deve ser registrada no `UserInputEventManager` atribuído à tela, utilizando o método `UserInputEventManager.addUserInputEventListener(com.sun.dtv.ui.event.UserInputEventListener, com.sun.dtv.ui.event.UserInputEvent)` para fazer com que seja notificado sobre o `UserInputEvent` especificado.

51.12.2 Índice de métodos

`void userInputEventReceived(UserInputEvent event)`

Chamado pela plataforma sempre que um `UserInputEvent` for recebido.

51.12.3 Detalhe dos métodos

userInputEventReceived

`void userInputEventReceived(UserInputEvent event)`

Chamado pela plataforma sempre que um `UserInputEvent` for recebido. A notificação não ocorrerá se outro aplicativo reservou exclusivamente o `UserInputEvent` recebido

Parâmetros:

`event` – evento recebido

51.13 Classe UserInputEventManager

51.13.1 Descrição da classe

`com.sun.dtv.ui.event`

```
java.lang.Object
  └com.sun.dtv.ui.event.UserInputEventManager
```

```
final public class UserInputEventManager
extends Object
```

As instâncias desta classe são os gerenciadores de evento que manipulam os eventos de input de usuário a serem manipulados por um *widget*. O objetivo desses gerenciadores é principalmente gerenciar eventos reservados exclusicamente e instalar os *listeners* para manipulá-los.

A própria reserva exclusiva deve ser realizada utilizando o mecanismo de recurso escasso. Os eventos de input de usuário a serem manipulados dessa forma implementam a interface `ScarceResource` para poder serem manipulados de tal forma. A seguir está um recorte de código, demonstrando como registrar exclusivamente para um evento de input de usuário.

```
// MyClass implementa UserInputEventListener, desta forma pode ser usada como
 // sua própria UserInputEventListener
 import com.sun.dtv.ui.event.*;

 public class MyClass implements com.sun.dtv.ui.event.UserInputEventListener {
 ...
    // implementando a interface necessária para implementar o
    // método userInputEventReceived:
    public void userInputEventReceived(UserInputEvent event) {
       ...
    }
    ...
    public void myMethod(...) {
       ...
       // neste método, nós registramos exclusivamente para os eventos de entrada do usuário
       // primeiro, nós temos que recuperar o UserInputEventManager para nossa
       // tela atual
       UserInputEventManager manager =
            UserInputEventManager.getUserInputEventManager(currentScreen);
       // agora nós podemos registrar exclusivamente para vários tipos de eventos de entrada
do usuário
       // primeiro, construímos um padrão de evento do mouse, cobrindo todos os eventos do
mouse
       // queremos ser notificados sobre:
       MouseEvent anyMouseLeftClick =
          new MouseEvent(null, // não importa qual o componente é afetado
                         MouseEvent.MOUSE_LEFT_CLICK,
                         0, // não se preocupa com o tempo
                         0, // sem modificadores
                         0, 0, // não se preocupa com a posição
                         1, // um clique associado com o evento
                         false); // sem disparador de popup
      // agora registramos exclusivamente para este tipo de evento do mouse
      anyMouseLeftClick.reserve(true, -1, aScarceResourceListener);
      // Forçamos a reserva, sem tempo limite. Para detalhes sobre o
      // ScarceResourceListener veja a documentação sobre os recursos escassos.
      // Agora faça o tipo de evento exclusivamente reservado para a entrada do usuário
tratados:
      manager.addUserInputEventListener(this, anyMouseLeftClick);
      // Agora seremos notificados exclusivamente sobre qualquer clique esquerdo do mouse
      // recebidos em nossa tela, por exemplo, ninguem mais - incluindo o mecanismos
      // tratatado do evento awt – nós seremos notificados sobre qualquer tipo deste
      // evento. Isto é verdade até que liberemos  o tipo de evento ou perdemos
      // o controle exclusivo sobre ele por uma reserva feita por alguém
      // (neste caso o ScarceResourceListener especificado deve ser notificado fora)
      // Vamos ainda reservar alguns cliques de tecla para teclas coloridas
      RemoteControlEvent anyColoredKeyTyped =
         new RemoteControlEvent(null, // no matter which component is affected
               java.awt.event.KeyEvent.KEY_TYPED,
               0, // no matter when
```

```
                0, // no modifiers
                RemoteControlEvent.VK_COLORED, // any colored key
                CHAR_UNDEFINED); // no
        anyColoredKeyTyped.reserve(true, -1, aScarceResourceListener),
        // veja exemplo de clique de mouse acima para detalhes
        manager.addUserInputEventListener(this, anyColoreKeyTyped);
        ...
    }
    ...
}
```

Relaciona-se com:

`ScarceResource`, `UserInputEvent`, `KeyEvent`, `MouseEvent`, `RemoteControlEvent`

51.13.2 Índice de métodos

`void addUserInputEventListener(UserInputEventListener listener, UserInputEvent event)`

Adicionar o *listener* especificado que deveria ser notificado sobre o `UserInputEvent` especificado.

`static UserInputEventManager getUserInputEventManager(Screen screen)`

Retorna o único `UserInputEventManager` para a tela especificada.

`void removeUserInputEventListener(UserInputEventListener listener)`

Remove o *listener* especificado do `UserInputEventManager` para esta tela.

51.13.3 Detalhe dos métodos

getUserInputEventManager

```
public static UserInputEventManager getUserInputEventManager(Screen screen)
```

Retorna o único `UserInputEventManager` para a tela especificada. Consequentemente, este método criará o `UserInputEventManager` se chamado primeiramente para a tela especificada (geralmente isso acontecerá fora do construtor da tela), e retornará a instância idêntica para qualquer chamada subsequente.

Parâmetros:

`screen` – tela para a qual deseja-se obter o `UserInputEventManager`

Retorna:

a instância `UserInputEventManager` para a tela especificada

addUserInputEventListener

```
public void addUserInputEventListener(UserInputEventListener listener,
                                      UserInputEvent event)
```

Adicionar o *listener* especificado que deveria ser notificado sobre o `UserInputEvent` especificado. Observar que o *listener* não pode ser notificado sobre `UserInputEvents` recebidos caso outro aplicativo tenha reservado exclusivamente esse evento utilizando um mecanismo de recurso escasso anteriormente e a reserva ainda seja válida. Se este método for chamado mais de uma vez para o mesmo *listener*, registrando para uma notificação sobre o mesmo evento de input de usuário, depois da primeira, as chamadas subsequentes não têm efeito, ou seja, um *listener* pode se registrar apenas uma vez para um evento de input de usuário. Se diversos *listeners* registrarem para o mesmo evento de input de usuário, mas um tiver reservado esse evento exclusivamente, os outros *listeners* não podem ser notificados, mesmo que ainda estejam registrados. Se mais de um *listener* tentar

reservar o evento exclusivamente, deve ser gerenciado seguindo as regras do gerenciamento de recurso escasso.

NOTA A implementação deve assegurar que notificações sobre todos os eventos são processados sequencialmente. Deve ser assegurado que não mais que uma notificação seja manipulada ao mesmo tempo. Qualquer *listener* único deve ser noficado apenas sobre uma instância de cada vez. Se um novo evento for gerado antes que o método de *listener* notificado retornar do processamento do evento anterior, a implementação CONVÉM esperar pelo retorno antes de processar a próxima chamada de notificação.

Parâmetros:

`listener` - `UserInputEventListener` a ser adicionado.

`event` - `UserInputEvent` que o *listener* deve obedecer

Relaciona-se com:

```
KeyEvent.reserve(boolean,     long,     com.sun.dtv.resources.ScarceResourceListener),
MouseEvent.reserve(boolean,    long,    com.sun.dtv.resources.ScarceResourceListener),
KeyEvent.reserve(boolean,     long,     com.sun.dtv.resources.ScarceResourceListener),
removeUserInputEventListener(com.sun.dtv.ui.event.UserInputEventListener)
```

removeUserInputEventListener

```
public void removeUserInputEventListener(UserInputEventListener listener)
```

Remove o *listener* especificado do `UserInputEventManager` para esta tela. Se o `UserInputEventListener` especificado não for adicionado antes a este gerenciados, a chamada a este método não tem efeito.

Parâmetros:

`listener` - `UserInputEventListener` a ser removido

Relaciona-se com:

```
addUserInputEventListener(com.sun.dtv.ui.event.UserInputEventListener,
com.sun.dtv.ui.event.UserInputEvent)
```

Anexo A
(informativo)

Termos da licença da Oracle

A.1 Considerações gerais

O *copyright* 2008-2009 pertence a Oracle Corporation (“Oracle”) situada em 500 Oracle Parkway, Redwood Shores, CA 94065, U.S.A.

Direitos do Governo - Software Comercial. Usuários do governo estão sujeitos ao contrato de licença padrão da Oracle e provisões aplicáveis do FAR e seus suplementos. A Oracle, o logo da Oracle e do Java são marcas ou marcas registradas da Oracle nos EUA e outros países. Produtos cobertos por e informações contidas nesta Norma são controladas pelas leis de controle de exportação dos EUA e podem estar sujeitas às leis de exportação e importação em outros países. Fins de uso ou usuários finais para armas nucleares, mísseis, químicas, biológicas ou marítimo nuclear, direta ou indiretamente, são estritamente proibidos. Exportação ou reexportação para países sujeitos a embargo dos EUA ou para entidades identificadas em listas de exclusão de exportação nos EUA, incluindo, mas não se limitando as pessoas banidas e listas nacionais especialmente designadas é estritamente proibido.

Os trechos em caixa alta desse Anexo referem-se ao destaque solicitado pela Oracle Microsystem para manter a fidelidade de conteúdo e forma ao texto jurídico formal do licenciamento.

A DOCUMENTAÇÃO É CONCEDIDA SEM GARANTIAS E TODAS AS CONDIÇÕES, REPRESENTAÇÕES E GARANTIAS EXPLÍCITAS OU IMPLÍCITAS, INCLUINDO QUALQUER GARANTIA IMPLÍCITA DE MERCANTIBILIDADE, ADEQUAÇÃO A UM FIM ESPECÍFICO OU NÃO-INFRINGIMENTO SÃO EXONERADOS, EXCETO NA MEDIDA EM QUE ESSAS EXONERAÇÕES SE TORNEM LEGALMENTE INVÁLIDAS.

Todos os direitos reservados.

A ORACLE LICENCIARÁ ESSA ESPECIFICAÇÃO JAVA^{TM} DTV ("ESPECIFICAÇÃO") APENAS SOB AS CONDIÇÕES DE QUE TODOS OS TERMOS CONTIDOS NESTE CONTRATO SEJAM ACEITOS. LER OS TERMOS E CONDIÇÕES DESTE CONTRATO CUIDADOSAMENTE. AO BAIXAR ESTA ESPECIFICAÇÃO, OS TERMOS E CONDIÇÕES DO CONTRATO SÃO ACEITOS CASO NÃO DESEJE SE VINCULAR A ELE, PRESSIONAR O BOTÃO "RECUSAR" NA PARTE INFERIOR DA PÁGINA OU, SE RECEBEU A ESPECIFICAÇÃO COMO UM ARQUIVO ".zip", DELETE O ARQUIVO E NÃO FAÇA USO DA ESPECIFICAÇÃO

A.2 Concessões de licença limitada

A.2.1 Licença para fins de avaliação e teste

A Oracle por meio deste concede uma licença limitada (sem o direito de sublicenciar) paga, não-exclusiva, não-transferível, mundial, sob os direitos de propriedade intelectual aplicáveis da Oracle para visualizar, baixar, usar e reproduzir a Especificação para fim de avaliação interna e teste. Isso inclui: (a) desenvolvimento de aplicativos designados para rodar em uma implementação da Especificação, estipulado que estes mesmos aplicativos não implementem nenhuma porção(ões) da Especificação; e (b) discussão da Especificação com qualquer terceiro; e (c) extrair porções breves da Especificação em comunicações orais ou escritas que discutam a Especificação estipulado que tais excertos não constituem agregadamente uma porção significativa da Especificação.

A.2.2 Licença para distribuição de implementações conformes

A Oracle por meio deste concede uma licença limitada (sem o direito de sublicenciar) perpétua, não-exclusiva, não-transferível, mundial, totalmente paga, livre de royalties, sob todos os direitos de cópia aplicáveis ou, sujeito às provisões da Seção A.2.4 abaixo, direitos de patente que a Oracle pode ter cobrindo a Especificação para criar e/ou distribuir implementações da Especificação, estipulado que tal implementação: (a) implemente completamente a Especificação, incluindo todas as suas interfaces e funcionalidades requeridas; e (b) não modifique, sub-agrupe, super-agrupe ou de qualquer outra maneira estenda o Domínio da Oracle, ou inclua

quaisquer pacotes públicos ou protegidos, classes, interfaces de Java, campos ou métodos nos limites do Domínio da Oracle exceto aqueles requeridos/autorizados pela Especificação (“Implementação Conforme”). Em acréscimo, a licença precedente é expressamente condicionada na não-ação fora do escopo da licença. Nenhuma licença é concedida subseqüentemente para qualquer outro fim (incluindo, por exemplo, modificar a Especificação, exclusive dentro dos limites dos seus direitos de uso justos, ou distribuir a Especificação para terceiros). Igualmente, nenhum direito, título ou interesse em ou para quaisquer marcas, marcas de serviços, ou nomes comerciais da Oracle ou dos licenciadores da Oracle deve ser concedido subseqüentemente. Java e logos relacionados à Java, marcas e nomes são marcas ou marcas registradas da Oracle. nos EUA e outros países. Para os fins desta Seção A.2.2, “Domínio da Oracle” refere-se às classes públicas ou declarações de interfaces cujos nomes comecem com “java”, “javax”, “com.sun” ou seus equivalentes em qualquer convenção de nomenclatura subseqüente adotada pela Oracle por meio do Java Community Process, ou quaisquer sucessores reconhecidos ou substituições desta.

A.2.3. Condições para transferência

Não é necessário incluir as limitações (a)-(b) da Seção A.2.2 acima ou quaisquer outros requerimentos específicos de transferência em qualquer licença que for concedida concernente ao uso de implementação da Especificação ou produtos derivados de tal implementação. De qualquer forma, exceto em respeito a Implementações não-Java DTV, (e produtos derivados delas) que satisfaçam as limitações (a)-(b) da Seção A.2.2 acima, não é permitido: (a) conceder ou de qualquer outra maneira transferir para os seus licenciados quaisquer licenças sob direitos de propriedade intelectual da Oracle aplicáveis; nem (b) autorizar seus licenciados a fazer quaisquer reclamações concernentes à conformidade de suas implementações com a Especificação em questão. Para os fins desta Seção A.2.3, “*Implementações não-Java DTV*” refere-se a uma implementação da Especificação Java DTV desenvolvida pelo SBTVD, mas apenas se tais implementações não-Java DTV estiverem fora do Domínio da Oracle.

A.2.4. Reciprocidade concernente à licença de patentes

(i) Com respeito a quaisquer reclamações de patente cobertas pela licença concedida sob a Seção A.2.2 acima que seriam infringidas por todas as implementações tecnicamente possíveis da Especificação, tal licença é condicionada na sua oferta de uma licença perpétua, não-exclusiva, não-transferível, mundial, livre de *royalties*, de não compromisso mútuo, justa, razoável e com termos não-discriminatórios para quaisquer partes interessadas, sob os seus direitos de patente que são ou seriam infringidos por todas as implementações tecnicamente possíveis da Especificação para desenvolver, distribuir e usar uma implementação que preencha os requerimentos estabelecidos de (a) a (b) da Seção A.2.2 acima.

(ii) Com respeito a quaisquer reclamações de patentes de propriedade da Oracle e cobertas pela licença concedida sob a Seção A.2.2, se as infrações podem ou não ser evitadas de uma maneira tecnicamente possível ao implementar a Especificação, tal licença deve terminar em respeito a tais reclamações se, ao iniciar uma reclamação contra a Oracle de que ela, no curso do cumprimento de suas responsabilidades da Especificação, induziu qualquer outra entidade a infringir direitos de patente da reclamante.

(iii) Igualmente em respeito a quaisquer reclamações de patentes de propriedade da Oracle e cobertas pela licença concedida sob a Seção A.2.2 acima, onde as infrações de tais reclamações podem ser evitadas de uma maneira tecnicamente possível ao implementar a Especificação, tal licença deve terminar, em respeito a tais reclamações, se iniciar uma reclamação contra a Oracle de que a sua fabricação, uso, oferta de venda, venda ou importação de uma Implementação Conforme infringe os direitos de patente da reclamante.

A.3 Renúncia de Garantias

A ESPECIFICAÇÃO É CONCEDIDA SEM GARANTIAS (“AS IS”) A ORACLE NÃO FAZ REPRESENTAÇÕES OU GARANTIAS, EXPLÍCITAS OU IMPLÍCITAS, INCLUINDO, MAS NÃO LIMITANDO-SE A GARANTIA DE MERCANTIBILIDADE, ADEQUAÇÃO A UM FIM ESPECÍFICO, NÃO-VIOLAÇÃO (INCLUINDO COMO CONSEQUÊNCIA DE QUALQUER PRÁTICA OU IMPLEMENTAÇÃO DA ESPECIFICAÇÃO), OU QUE O CONTEÚDO DA ESPECIFICAÇÃO É APROPRIADO PARA QUALQUER FIM. Esse documento não representa nenhum comprometimento em liberar ou implementar qualquer porção da Especificação em qualquer produto. Em acréscimo, a Especificação pode incluir imprecisões técnicas ou erros tipográficos.

A.4 Limitação de Responsabilidades Legais

AO LIMITE NÃO PROIBIDO POR LEI, EM NENHUM EVENTO A ORACLE OU SEUS LICENCIADOS SERÃO RESPONSÁVEIS POR QUAISQUER DANOS, INCLUINDO PERDA DE RENDIMENTO, LUCROS OU DADOS SEM LIMITAÇÃO OU POR DANOS ESPECIAIS, INDIRETOS, CONSEQUENTES, INCIDENTAIS OU PUNITIVOS

CAUSADOS POR QUALQUER RAZÃO E INDEPENDENTE DA TEORIA DA RESPONSABILIDADE CIVIL, SURGINDO DE OU EM RELAÇÃO A SUA POSSE, IMPLEMENTAÇÃO OU USO DE QUALQUER OUTRA FORMA DA ESPECIFICAÇÃO, MESMO SE A ORACLE E/OU SEUS LICENCIADORES TENHAM SIDO ALERTADOS QUANTO A POSSIBILIDADE DE TAIS DANOS.

Você irá indenizar, inocentar e defender a Oracle e seus licenciadores de quaisquer reclamações surgindo ou resultando de: (i) uso não-autorizado da Especificação; (ii) o uso ou distribuição da sua aplicação, *applet* e/ou implementação Java; e/ou (iii) quaisquer reclamações de que versões ou lançamentos posteriores a qualquer Especificação provida a você são incompatíveis com a Especificação provida sob esta licença.

A.5 Inscrição de direitos restritos

Governo EUA: Se esta Especificação estiver sendo adquirida por ou em benefício do governo dos EUA ou por um contratado principal ou subcontratado do governo (em qualquer nível), os direitos do governo do *software* e da documentação acompanhando devem ser como estabelecidos nesta licença; de acordo com a 48 C. F. R. 227,7201. 7201 a 227. 227,7202-4 (para aquisições do Departamento de Defesa (DoD)) e com a C. F. R. 2,101. 12,212 e 12. 212 (para aquisições não-DoD).

A.6 Relatório

Se você prover a Oracle quaisquer comentários ou sugestões concernindo a Especificação (“Feeback”), você por meio desse: (i) concorda que tal Feedback é provido em uma base não-proprietária e não-confidencial, e (ii) concede a Oracle licença irrevogável, não-exclusiva, mundial, totalmente paga, com o direito de sublicenciar através de múltiplos níveis de sublicenciados para incorporar, expor e usar sem limitações o Feedback para qualquer fim.

A.7 Termos gerais

Qualquer ação relacionada a esse Contrato é regida pela lei da Califórnia e pela lei federal regente nos EUA. A Convenção da ONU para Venda Internacional de Mercadorias e a escolha de leis de qualquer jurisdição não se aplicarão.

A Especificação está sujeita às leis de controle de exportação dos EUA e pode estar sujeita a regulamentos de exportação ou importação em outros países. Os licenciados concordam em obedecer estritamente todas essas leis e regulamentos e reconhecem que têm a responsabilidade de obter licenças para exportar, reexportar ou importar como pode ser requerido após envio para o licenciado.

Esse Contrato é o acordo completo de todas as partes ao que se relaciona ao seu tema. Ele suplanta todas as comunicações orais ou escritas anteriores ou contemporâneas, propostas, condições, representações e garantias e prevalece sobre quaisquer termos conflituosos ou adicionais de qualquer declaração, ordem, atestado, ou outra comunicação entre as partes relacionadas ao seu tema durante o termo deste Contrato. Nenhuma modificação desse Contrato é obrigatória, a menos que escrita e assinada por um representante oficial de cada uma das partes.

Este Contrato encerrará imediatamente sem notificação da Oracle se você violar o Acordo ou agir fora do escopo das licenças concedidas acima.

Bibliografia

[1] [J200] ITU. “ITU-T Recommendation J.200: Worldwide common core – Application environment for digital interactive television services”, 2001.

[2] [J201] ITU. “ITU-T Recommendation J.201: Harmonization of declarative content format for interactive television applications”, 2004.

[3] [J202] ITU. “ITU-T Recommendation J.202: Harmonization of declarative content format for interactive television applications”, 2003.

[4] [ARIB-B23] ARIB. “ARIB STD-B23 Versão 1.1: Plataforma para Mecanismo de Execução de Aplicativo para Transmissão Digital (Tradução portuguesa)” Padrão ARIB, 2004.

[5] [ARIB-B10] ARIB. “ARIB STD-B10 “Serviços de Informação para Sistema de Transmissão digital”. Padrão ARIB, 2004.

[6] [RFC 2119], Key words for use in RFCs to Indicate Requirement Levels - http://www.ietf.org/rfc/rfc2119.txt.

[7] [RFC 1951], DEFLATE Especificação de Compressão de Dados Versão 1.3 - http://www.ietf.org/rfc/rfc1951.txt.

[8] [RFC 1950], ZLIB Compressed Data Format Specification version 3.3- http://www.ietf.org/rfc/rfc1950.txt.

[9] [X.509], ITU-T Recommendation X.509: Information technology – Open System Interconnection – The Directory: Public-key and attribute certificate frameworks.

[10] [Application Note on the .ZIP file format], http://www.pkware.com/support/zip-application-note#_blank.

[11] [T. Porter & T. Duff] - "Compositing Digital Images", Computer Graphics Volume 18, Number 3 July 1984 pp 253-259.

[12] [Ant Tool] - http://ant.apache.org/

[13] [sRGB] - A Standard Default Color Space for the Internet, http://www.w3.org/Graphics/Color/sRGB.html.