

# Interfaces Gráficas

Componentes Gráficos

Gestores de Posicionamento

Manipuladores de Eventos

*(Livro Big Java, Late Objects – Capítulos 10 e 11)*

- [Introdução](#)
- [Componentes Gráficos](#)
  - [Introdução](#)
  - [Hierarquia de Classes](#)
  - [Interfaces](#)
  - [Categorias](#)
    - [Contentores de Componentes Gráficos](#)
    - [Apresentação de Informação](#)
    - [Controlos Básicos](#)
- [Gestores de Posicionamento](#)
- [Manipuladores de Eventos](#)
- [Bibliografia Geral](#)
- [Índice Remissivo](#)



- [Introdução](#)
- [Componentes Gráficos](#)
  - [Introdução](#)
  - [Hierarquia de Classes](#)
  - [Interfaces](#)
  - [Categorias](#)
    - [Contentores de Componentes Gráficos](#)
    - [Apresentação de Informação](#)
    - [Controlos Básicos](#)
- [Gestores de Posicionamento](#)
- [Manipuladores de Eventos](#)
- [Bibliografia Geral](#)
- [Índice Remissivo](#)

- [Interface do Utilizador](#)
  - [Noção](#)
  - [Tipos](#)
    - [Consola](#)
    - [Gráfica](#) (GUI – *Graphical User Interface*)
- [Interface do Utilizador Gráfica](#)
  - [Constituição](#)
    - [Componentes Gráficos](#)
    - [Gestores de Posicionamento](#)
    - [Manipuladores de Eventos](#)
  - [Toolkits Java para GUIs](#)
    - [AWT](#)
    - [Swing](#)
    - [Java 2D](#)
  - [História AWT e Swing](#)
  - [Construção](#)
    - [Editor GUI](#)
    - [Programação](#)

- **Noção**

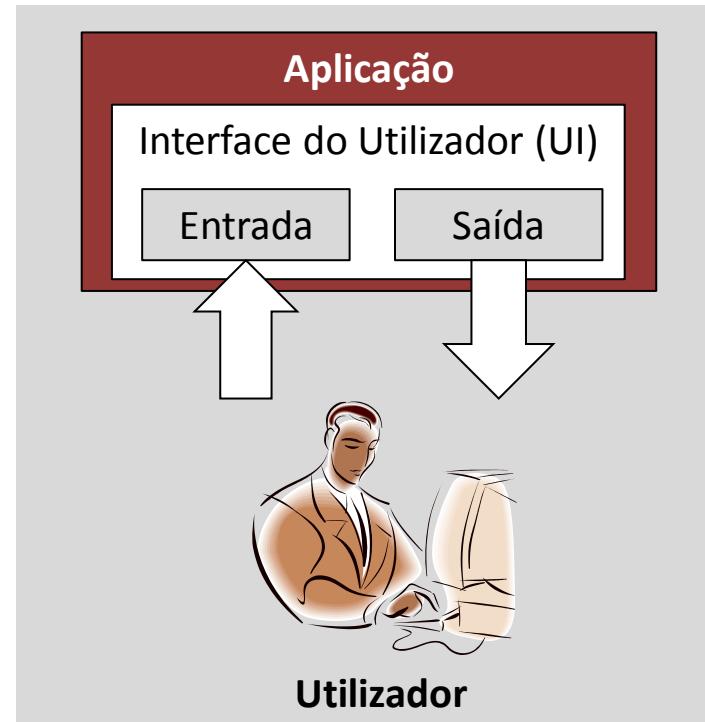
- Segmento de código de uma aplicação que permite a sua interação com o utilizador

- **Objetivo**

- Permitir ao utilizador **controlar** o **funcionamento** do programa

- **Meios Fornecidos**

- Entrada // Para **utilizador** controlar **execução** do programa
  - Saída // Para **programa** indicar **efeitos** do controlo ao **utilizador**



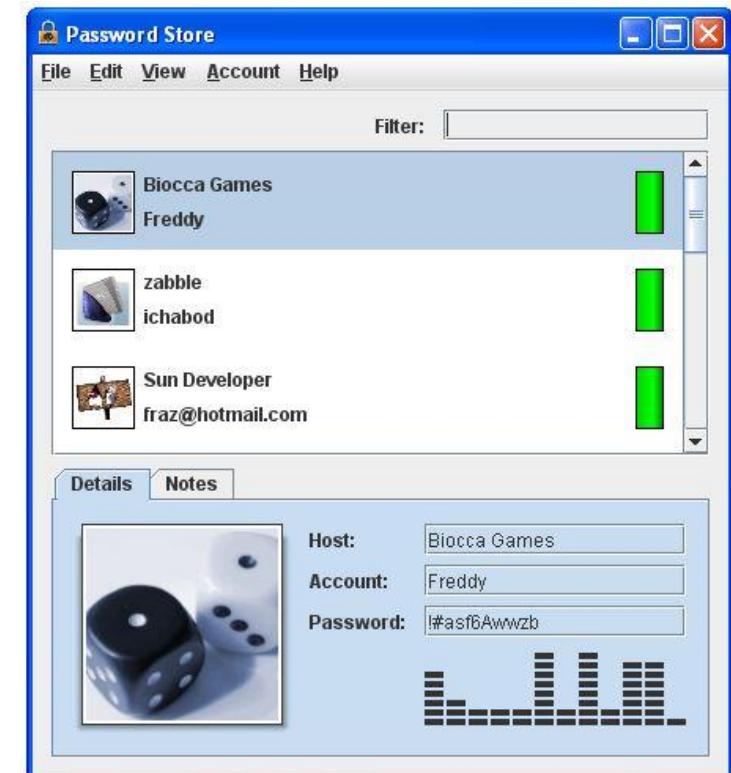
## ▪ Consola

- Interface Textual
- Entrada
  - Suportada por comandos escritos no teclado
- Saída
  - Suportada por texto impresso no monitor



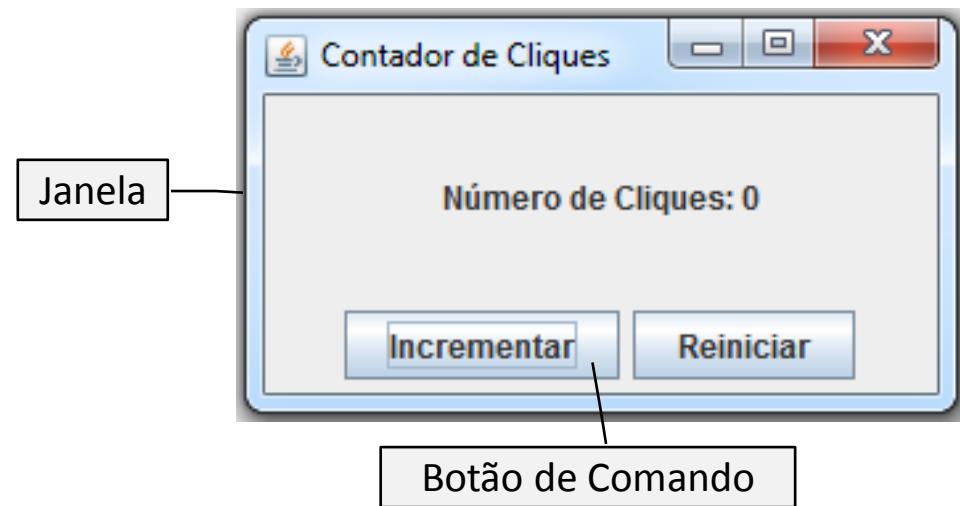
## ▪ Gráfica ( *GUI - Graphical User Interface* )

- Baseada
  - Janelas do Sistema Operativo
  - Constituídas por objetos
- Entrada
  - Suportada por **objetos** de janelas
  - Indicados através de rato e/ou teclado
- Saída
  - Suportada por **objetos** de janelas



## ■ GUI

- Constituído por objetos
- Por exemplo
  - Janela
  - Botões de comando

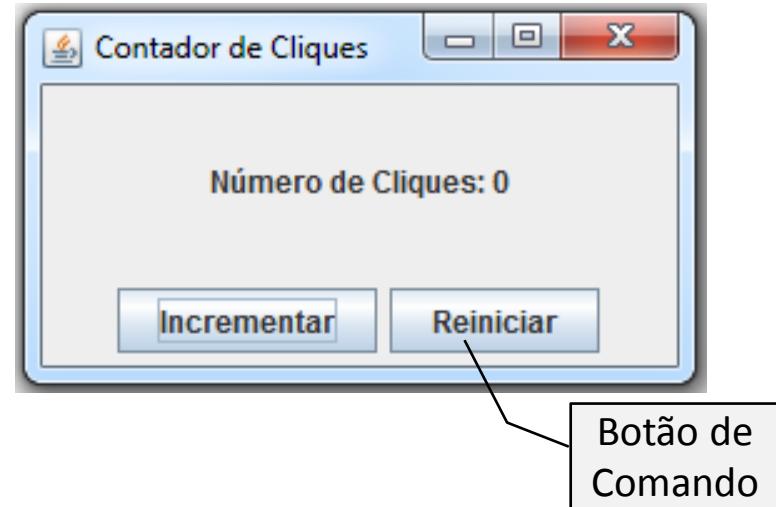


## ■ Tipos de Objetos

- Componentes Gráficos
- Gestores de Posicionamento *// Layout Managers*
- Manipuladores de Eventos *// Event Listeners*

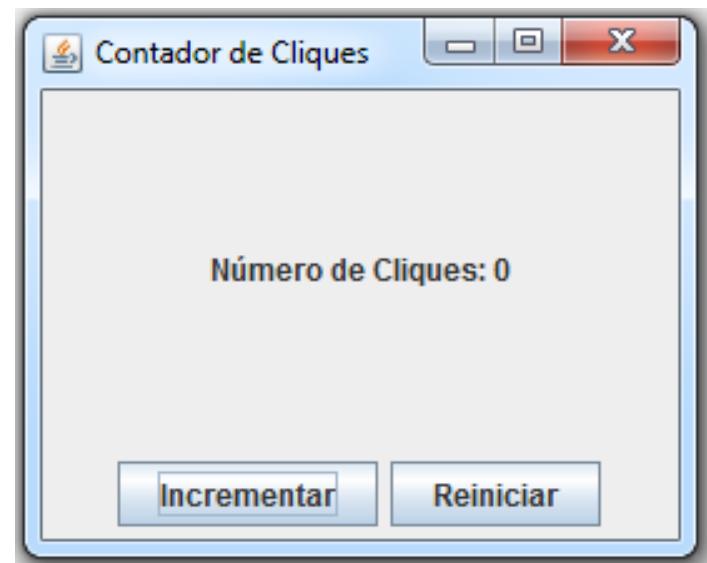
## ■ Componentes Gráficos

- Objetos
  - Com representação gráfica
    - Para mostrar num monitor
  - Podem interagir com o utilizador
- Exemplo
  - Botões de comando



## ■ Gestores de Posicionamento

- Objetos
- Gerem a colocação dos componentes na GUI
  - Automaticamente
- Exemplo
  - Redimensionamento da Janela



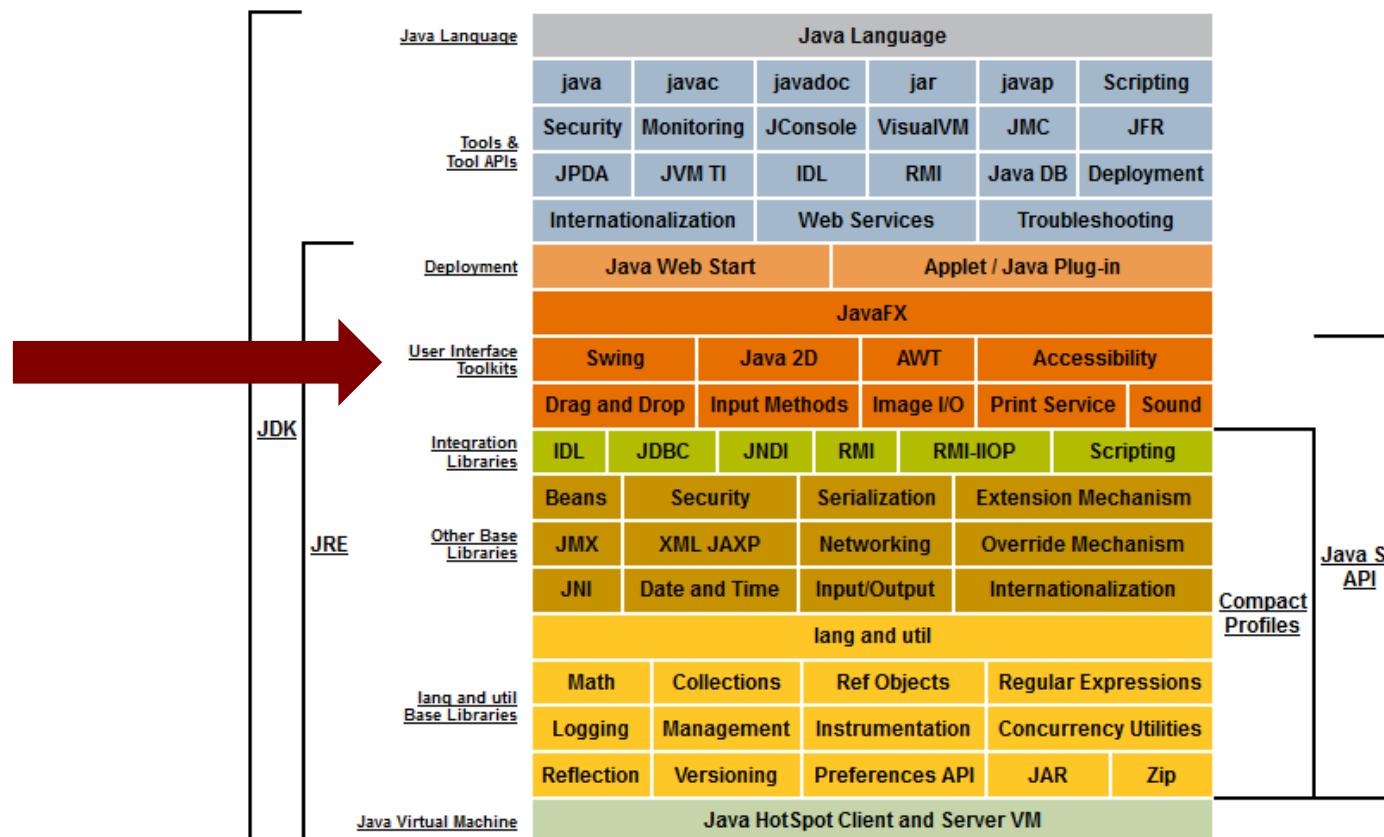
## ■ Manipuladores de Eventos

- Respondem às solicitações do utilizador
- Exemplo
  - Resposta a cliques nos botões de comando

## ■ Principais

- Swing // Componentes Gráficos
- Java 2D // Componentes especiais, para gráficos 2D de alta-qualidade
- AWT // Gestores de Posicionamento
- // Manipuladores de Eventos

## ■ Diagrama de Tecnologias Java Standard Edition 8 [\(http://docs.oracle.com/javase/8/docs/index.html\)](http://docs.oracle.com/javase/8/docs/index.html)

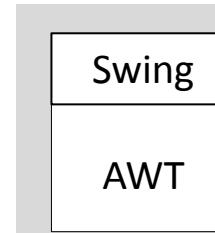


- **AWT (Abstract Window Toolkit)**

- 1º Toolkit Java para construir GUIs
- Introduzido pelo JDK 1.0
- Componentes apresentavam limitações
  - Qualidades Gráfica e Funcional < Plataformas Nativas mais ricas (Windows e Macintosh)

- **Swing**

- 2º Toolkit Java para construir GUIs
- Introduzido pelo JDK 1.1
- Não substitui completamente o AWT
  - Construído por cima do AWT
  - **Manipuladores de Eventos e Gestores de Posicionamento** são AWT
- Estilo dos componentes gráficos
  - Independente do Sistema Operativo
  - Fornecidos vários estilos
- Classes Swing
  - package javax.swing
  - Nome começado por J
    - Em geral



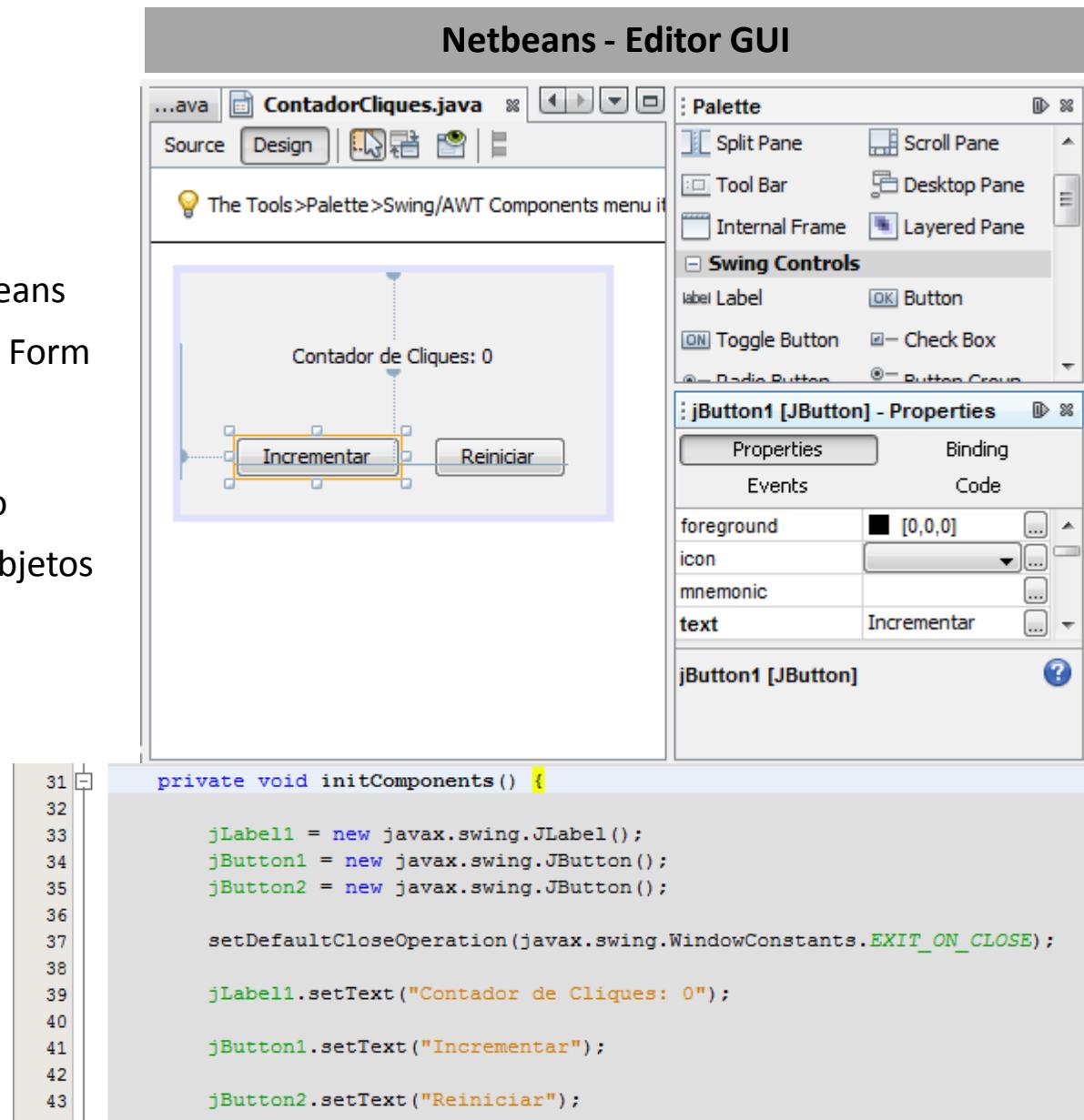
Componentes Gráficos  
Gestores de Posicionamento  
Manipuladores de Eventos

- **Formas**

- Visual
- Programada

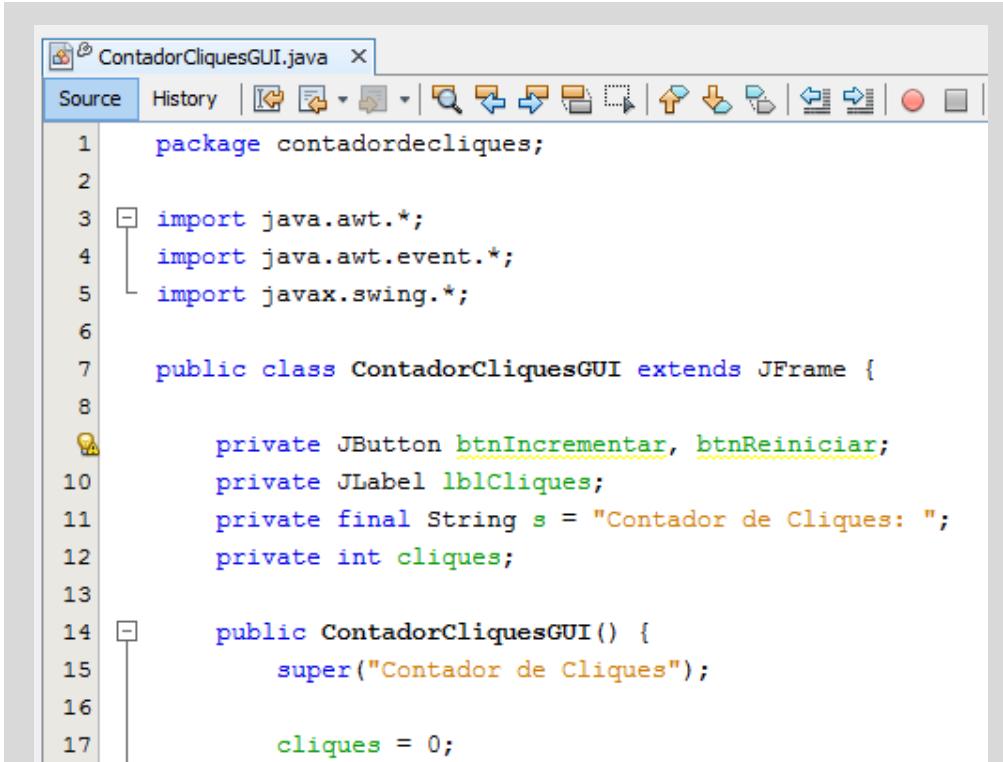
## ■ Visual

- Usando
  - Editor GUI
  - Exemplo
    - Editor GUI do Netbeans
    - Classe JFrame Form
- Programador
  - Desenha interface gráfico
  - Altera propriedades de objetos
- Editor GUI
  - Cria código
    - Automaticamente
- Facilita criação de GUI



## ■ Programada

- Escrita de código GUI
- Programador
  - Precisa de conhecer código GUI
- Código mais simples
  - Do que forma visual
- Estudada
  - Em PPROG



The screenshot shows the Netbeans IDE interface with the file `ContadorCliquesGUI.java` open. The code is written in Java and defines a class `ContadorCliquesGUI` that extends `JFrame`. The code includes imports for `java.awt.*`, `java.awt.event.*`, and `javax.swing.*`. It also defines private fields for a button (`btnIncrementar`, `btnReiniciar`), a label (`lblCliques`), and a string (`s`), along with an integer variable (`cliques`). The constructor initializes the frame with the title "Contador de Cliques" and sets the initial value of `cliques` to 0.

```
1 package contadordecliques;
2
3 import java.awt.*;
4 import java.awt.event.*;
5 import javax.swing.*;
6
7 public class ContadorCliquesGUI extends JFrame {
8
9     private JButton btnIncrementar, btnReiniciar;
10    private JLabel lblCliques;
11    private final String s = "Contador de Cliques: ";
12    private int cliques;
13
14    public ContadorCliquesGUI() {
15        super("Contador de Cliques");
16
17        cliques = 0;
```

Netbeans - Editor de Código Fonte

- [Introdução](#)
- [Componentes Gráficos](#)
  - [Introdução](#)
  - [Hierarquia de Classes](#)
  - [Interfaces](#)
  - [Categorias](#)
    - [Contentores de Componentes Gráficos](#)
    - [Apresentação de Informação](#)
    - [Controlos Básicos](#)
- [Gestores de Posicionamento](#)
- [Manipuladores de Eventos](#)
- [Bibliografia Geral](#)
- [Índice Remissivo](#)



- Introdução

- Categorias de Componentes Gráficos
  - Controlos Básicos
  - Apresentação de Informação
  - Contentores de Componentes Gráficos
- Estilos dos Componentes Swing
  - Exemplos
    - Java
    - Windows
  - Demos

## ■ Controlos Básicos

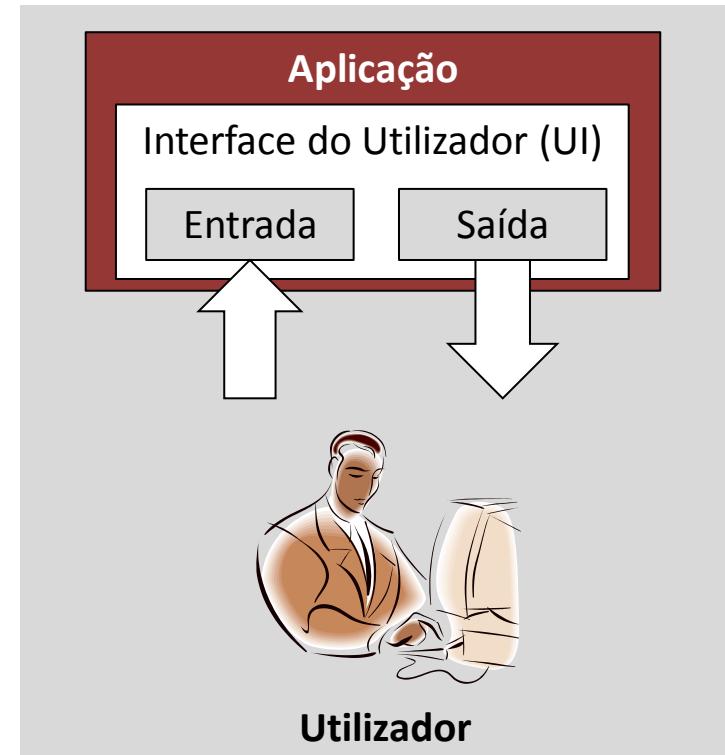
- Usados principalmente na **Entrada da Aplicação**
  - Para obter dados do utilizador

## ■ Apresentação de Informação

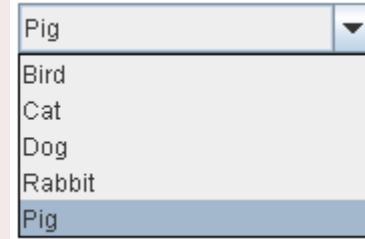
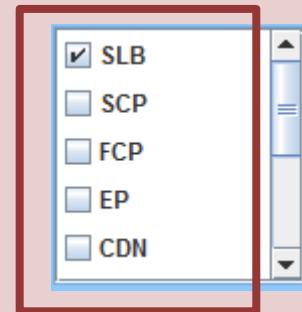
- Usados na **Saída da Aplicação**
  - Para mostrar informação ao utilizador
- Subcategorias
  - Editável
  - Não-Editável

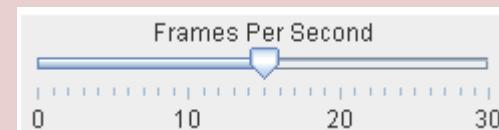
## ■ Contentores de Componentes Gráficos

- Usados na **Estrutura da Interface Gráfica**
  - Para suportar componentes gráficos
- Subcategorias
  - Nível Superior
  - Genéricos
  - Especiais



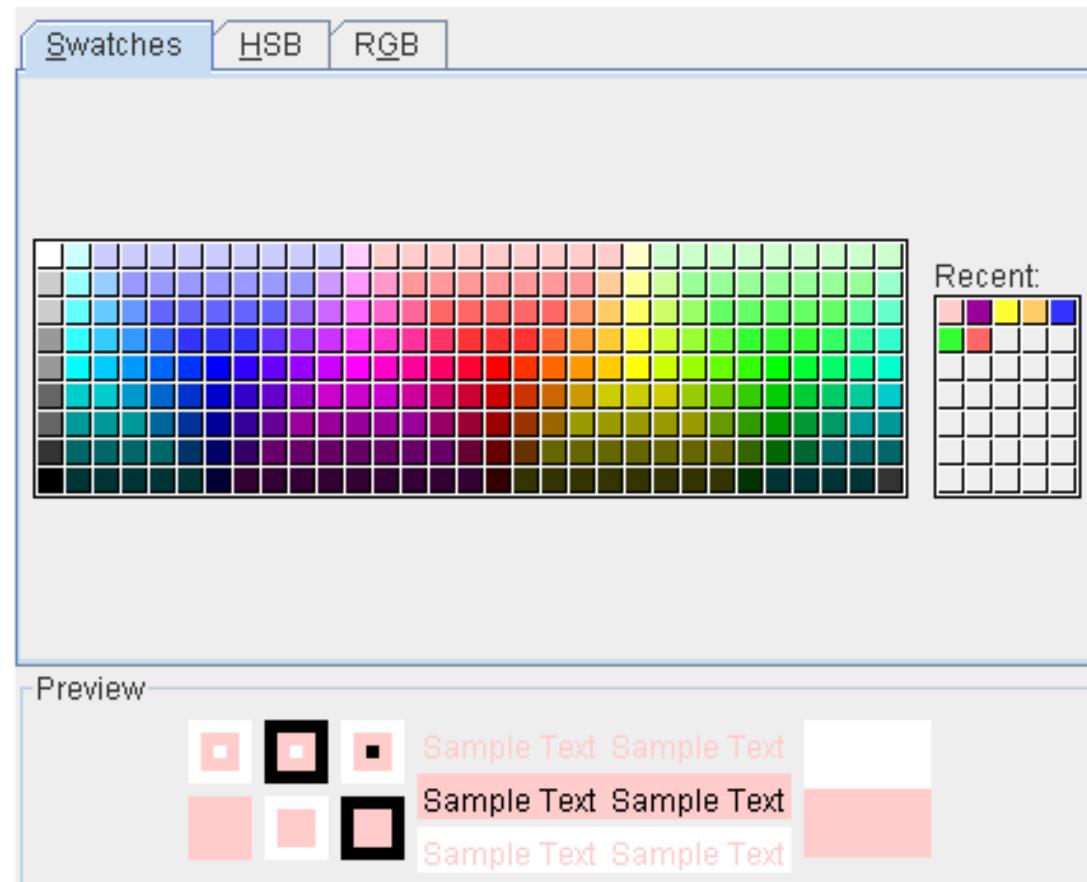
Componente	Classe	Interesse	Representação Gráfica
Botão de Comando	<a href="#">JButton</a>	Ordenar execução de comando.	
Menu	<a href="#">JMenu</a>	<p>Escolher <u>um</u> comando entre múltiplos alternativos;</p> <p>Vantagem:</p> <ul style="list-style-type: none"><li>■ Ocupa espaço reduzido da GUI (área da barra de menus).</li></ul>	
Caixa de Verificação	<a href="#">JCheckBox</a>	Escolher <u>múltiplas</u> opções em simultâneo.	

Componente	Classe	Interesse	Representação Gráfica
Botão de Opção	<a href="#">JRadioButton</a>	Escolher <u>uma</u> opção entre múltiplas alternativas.	
Caixa de Combinação	<a href="#">JComboBox</a>	Escolher <u>um</u> item de uma lista de <u>objetos</u> , sendo possível editar o item escolhido; Cada item visualiza a <i>string</i> retornada pelo método <code>toString()</code> do respetivo objeto.	
Caixa de Listagem	<a href="#">JList</a>	Apresentar uma lista de itens, numa ou mais colunas, para serem escolhidos ( <u>escolha simples</u> ou <u>múltipla</u> ); Os itens correspondem a <u>objetos</u> ; Os componentes gráficos podem ser visualizados.	 <div data-bbox="1397 1264 1820 1321">JList dentro de JScrollPane</div>

Componente	Classe	Interesse	Representação Gráfica
Campo de Texto	<a href="#">JTextField</a>	Introduzir pequena quantidade de texto.	
Campo de Password	<a href="#">JPasswordField</a>	Introduzir pequena quantidade de texto, sem mostrar caracteres digitados.	
Corredíça	JSlider	Introduzir facilmente valor numérico a partir de uma gama de valores;  Vantagem: <ul style="list-style-type: none"><li>▪ Ocupa pouco espaço.</li></ul>	
Spinner	JSpinner	Escolher item de uma gama de itens;  Vantagens: <ul style="list-style-type: none"><li>▪ Ocupa pouco espaço;</li><li>▪ Não mostra lista que pode cobrir outros componentes;</li><li>▪ Utilizador pode editar valor.</li></ul>	

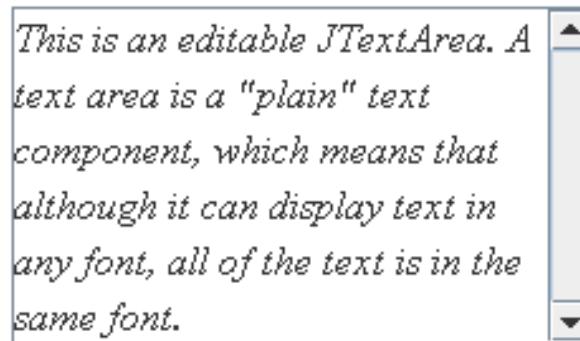
- Informação pode ser modificada por utilizador

Componente	Classe	Interesse
Escolha de Cor	JColorChooser	Escolher cor a partir de paleta de cores.

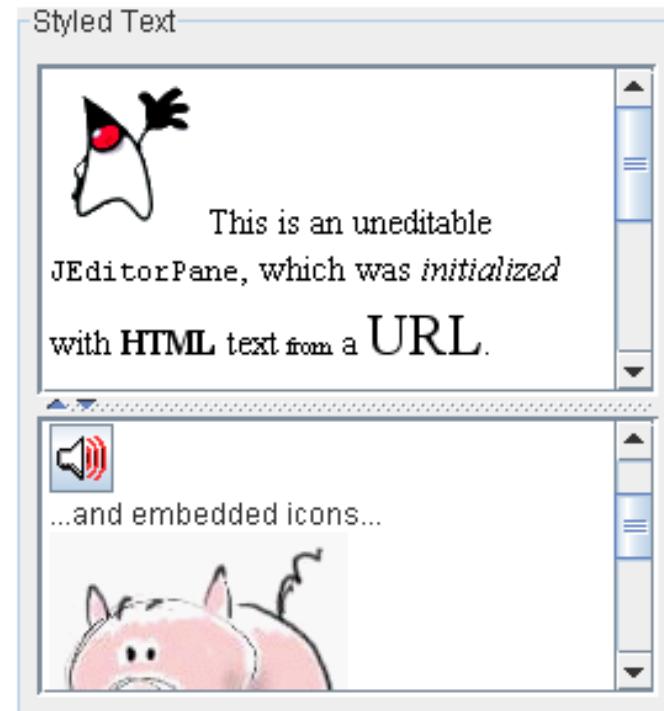


### JColorChooser

Componente	Classe	Interesse
Área de Texto	JTextArea	Mostrar e editar texto sem estilo e em múltiplas linhas.
Painel de Edição	JEditorPane	Mostrar e editar texto com estilo nativo ou definido pelo utilizador.

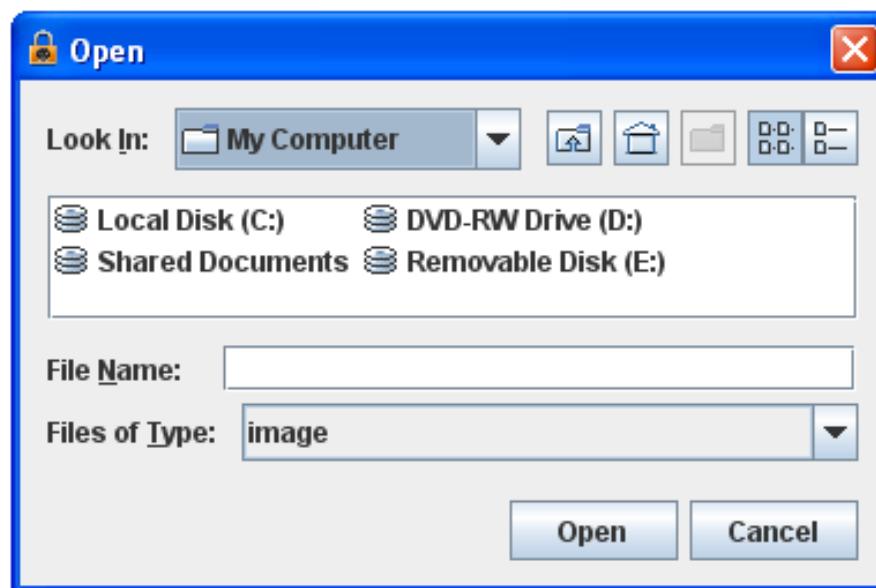


[JTextArea](#)



[JEditorPane](#)

Componente	Classe	Interesse
Escolha de Ficheiro	<a href="#">JFileChooser</a>	Navegar no sistema de ficheiros e depois escolher um ficheiro/diretório da lista apresentada ou introduzir o nome de um ficheiro/diretório.
Tabela	JTable	Mostrar tabela de dados; Opcionalmente, utilizador pode editar os dados.
Árvore	JTree	Mostrar informação organizada de forma hierárquica.

[JFileChooser](#)

Host	User	Password	Last Modified
Biocca Games	Freddy	!#asf6Awwzb	Mar 16, 2006
zabble	ichabod	Tazb!34\$!Z	Mar 6, 2006
Sun Developer	fraz@hotmail.co...	AasW541!fbZ	Feb 22, 2006
Heirloom Seeds	shams@gmail....	bkz ADF78!	Jul 29, 2005
Pacific Zoo Shop	seal@hotmail.c...	vbAf124%z	Feb 22, 2006

[JTable](#)[JTree](#)

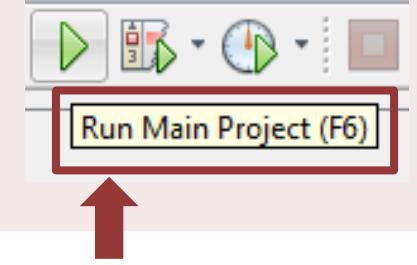
■ **Informação**

- Não pode ser alterada pelo utilizador

■ **Interesse Geral**

- Apenas informar o utilizador

■ **Tipos**

Componente	Classe	Interesse	Representação Gráfica
<i>Label</i>	<a href="#">JLabel</a>	Mostrar imagem e/ou texto não editável.	 Image and Text
Barra de Progresso	JProgressBar	Mostrar graficamente a quantidade de trabalho total completado.	 31%
Separador	<a href="#">JSeparator</a>	Mostrar linha divisória horizontal ou vertical; usado normalmente em menus e barras de ferramentas.	
Dica	JToolTip	Mostrar dica sobre componente GUI.	 Run Main Project (F6)

**■ Interesse Geral**

- Servirem de estrutura da GUI
  - Para suportar componentes gráficos

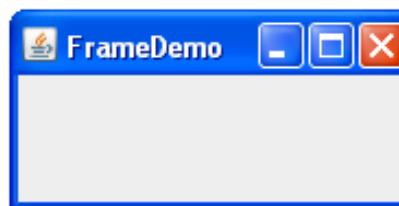
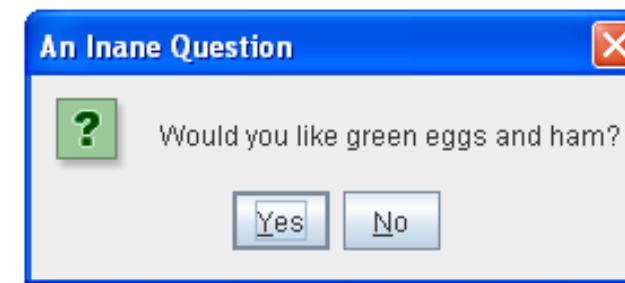
**■ Subcategorias**

- Nível superior
- Genérico
- Especial

## ■ Nível Superior

- Não podem ser incluídos noutras componentes gráficos
- Qualquer aplicação
  - Usa pelo menos um componente deste tipo
- Tipos

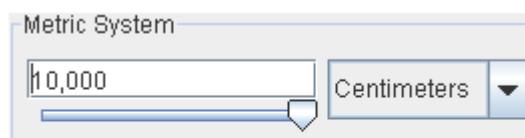
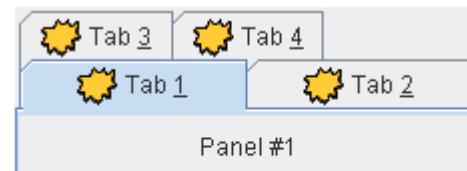
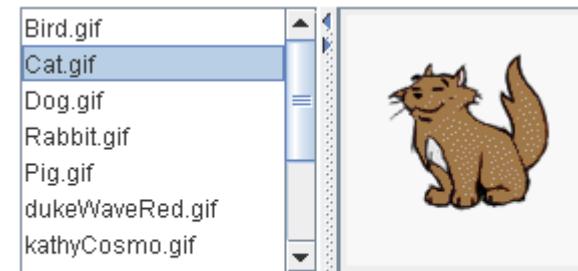
Componente	Classe	Interesse
Janela	<a href="#">JFrame</a>	Criar janelas.
Caixa de Diálogo	<a href="#">JDialog</a>	Criar caixas de diálogo personalizadas.
Applet	JApplet	Criar <i>applets</i> com componentes <i>Swing</i> ; <i>Applet</i> = programa que pode ser incluído numa página HTML.

[JFrame](#)[JDialog](#)[JApplet](#)

## ■ Genéricos

- Para uso geral
- Tipos

Componente	Classe	Interesse
Painel	<a href="#">JPanel</a>	Criar painéis para construção de GUI sofisticados.
Painel de Deslocamento	<a href="#">JScrollPane</a>	Apresentar componente maior que espaço disponível para visualização.
Painel Dividido	<a href="#">JSplitPane</a>	Mostrar 2 componentes, lado-a-lado ou cima-baixo.
Painel de Separadores	<a href="#">JTabbedPane</a>	Permitir espaço partilhado por vários componentes.
Barra de Ferramentas	<a href="#">JToolBar</a>	Criar barras de ferramentas.

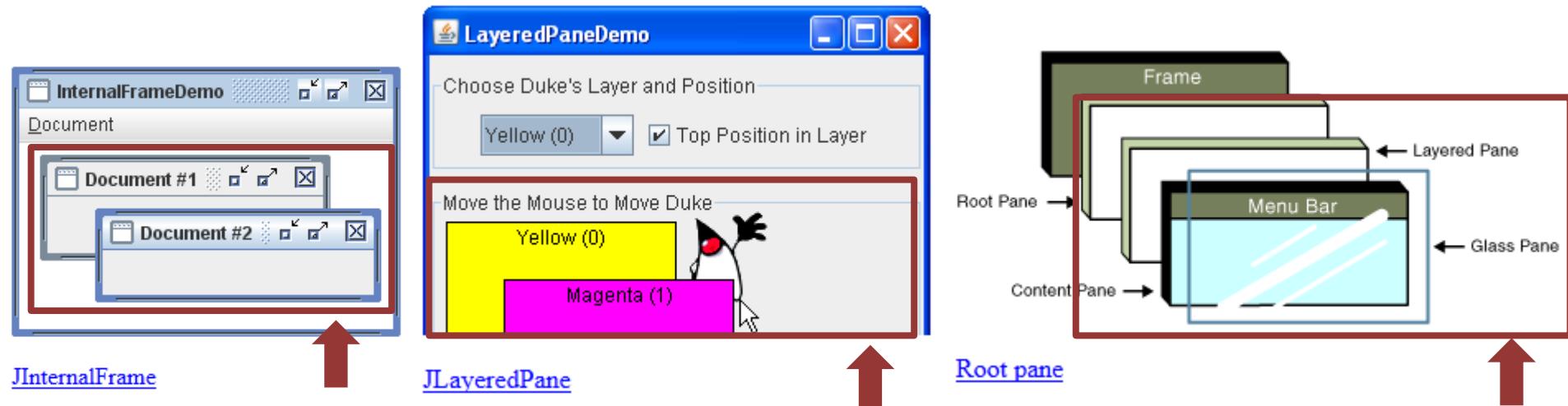
JPanelJTabbedPaneJSplitPaneJToolBarJScrollPane

## Especiais

- Para uso particular
- Tipos

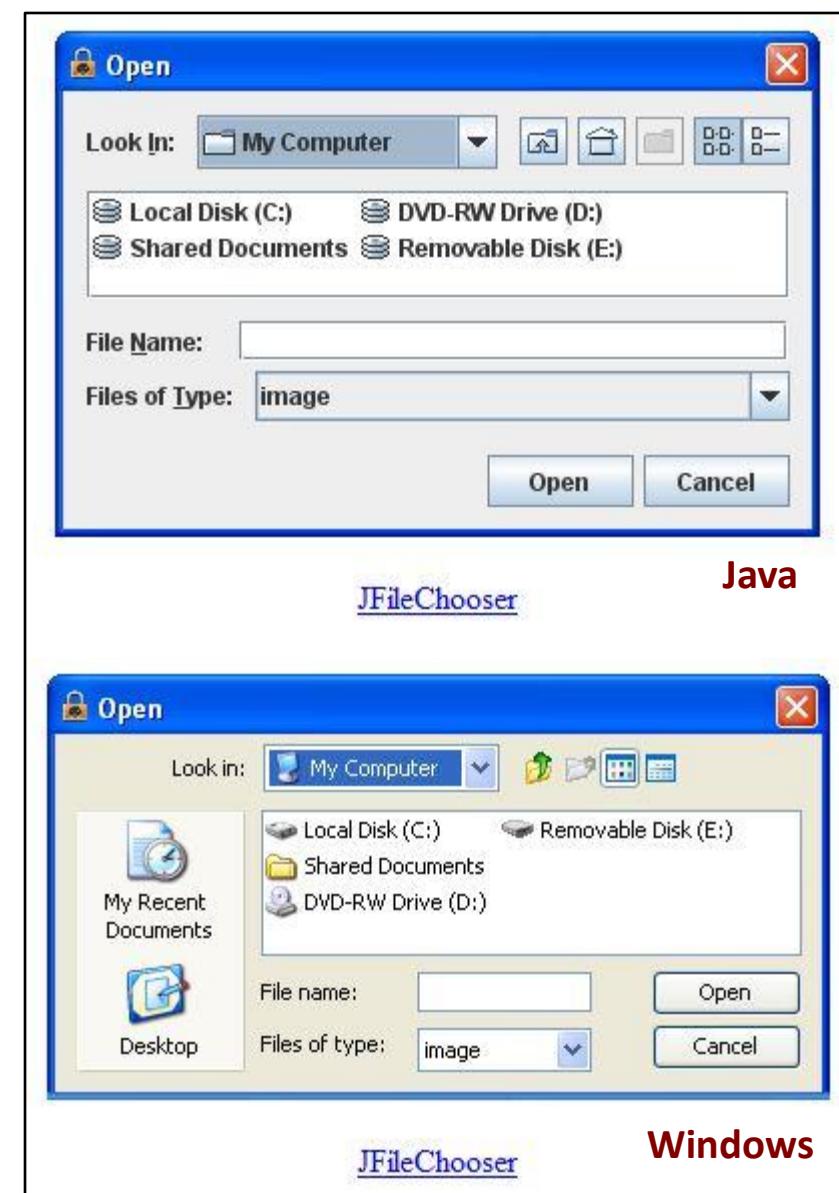
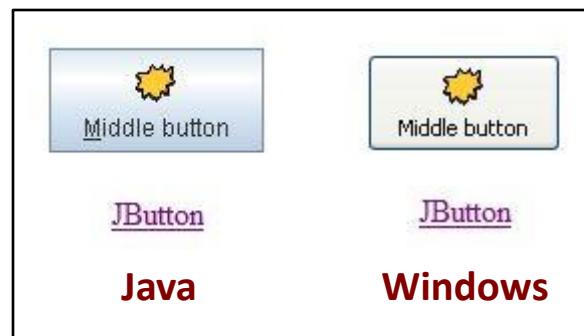
Componente	Classe	Interesse
Janela Interna	<code>JInternalFrame</code>	Criar janelas internas numa janela.
<i>Layered Pane</i>	<a href="#"><code>JLayeredPane</code></a>	Posicionar componentes num sistema de coordenadas tridimensional (em camadas).
<i>Root Pane</i>	<a href="#"><code>JRootPane</code></a>	Presente em contentores de nível superior e em janelas internas.

## Representação Gráfica



## ■ Exemplos

- Java (Metal) // por omissão
- Windows



## ■ Modificação do Estilo por Omissão

## ■ Não-programada

1. Abrir o **ficheiro** jre/lib/swing.properties // subdiretório de instalação do Java
2. Redefinir **propriedade** swing.defaultlaf
  - Usar o nome da classe do LaF pretendida
  - Exemplo: com.sun.java.swing.plaf.windows.**WindowsLookAndFeel**
3. Reiniciar aplicação para modificação ter efeito
  - Esta propriedade só é lida no arranque da aplicação

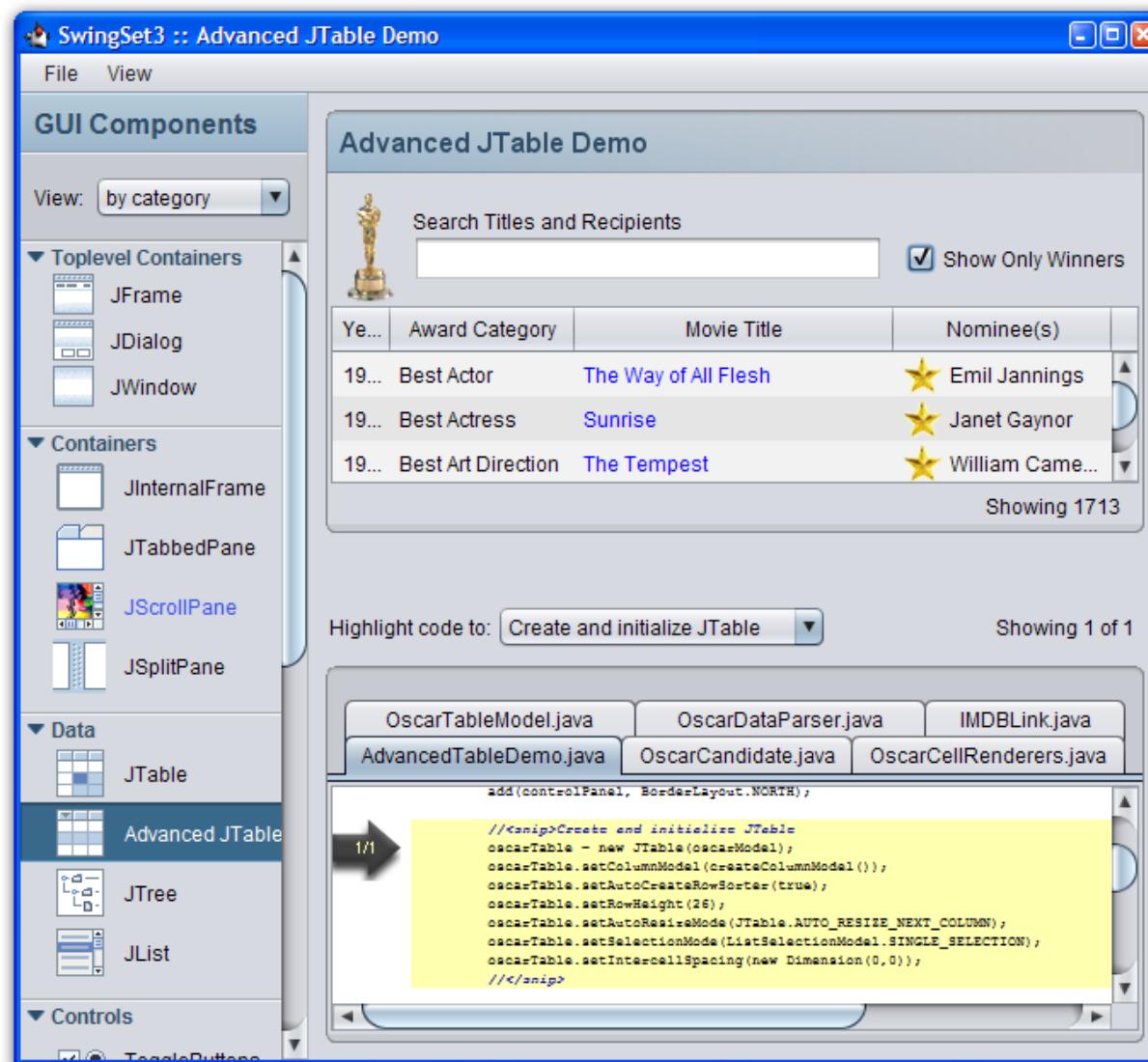
## ■ Programada

## ■ Exemplo

```
public class Main {

    public static void main(String[] args) {
        GUI gui = null;
        try {
            gui = new GUI();
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
            SwingUtilities.updateComponentTreeUI(gui);
        } catch (Exception excecao) {
            JOptionPane.showMessageDialog(gui, excecao.getMessage(), "Erro", JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

- [Java SE Development Kit 8 Demos and Samples Downloads](#)
- **Exemplo:** SwingSet3



- [Introdução](#)
- [Componentes Gráficos](#)
  - [Introdução](#)
  - [Hierarquia de Classes](#)
  - [Interfaces](#)
  - [Categorias](#)
    - [Contentores de Componentes Gráficos](#)
    - [Apresentação de Informação](#)
    - [Controlos Básicos](#)
  - [Gestores de Posicionamento](#)
  - [Manipuladores de Eventos](#)
  - [Bibliografia Geral](#)
  - [Índice Remissivo](#)



- [Hierarquia de Classes](#)

- [Superclasses](#)

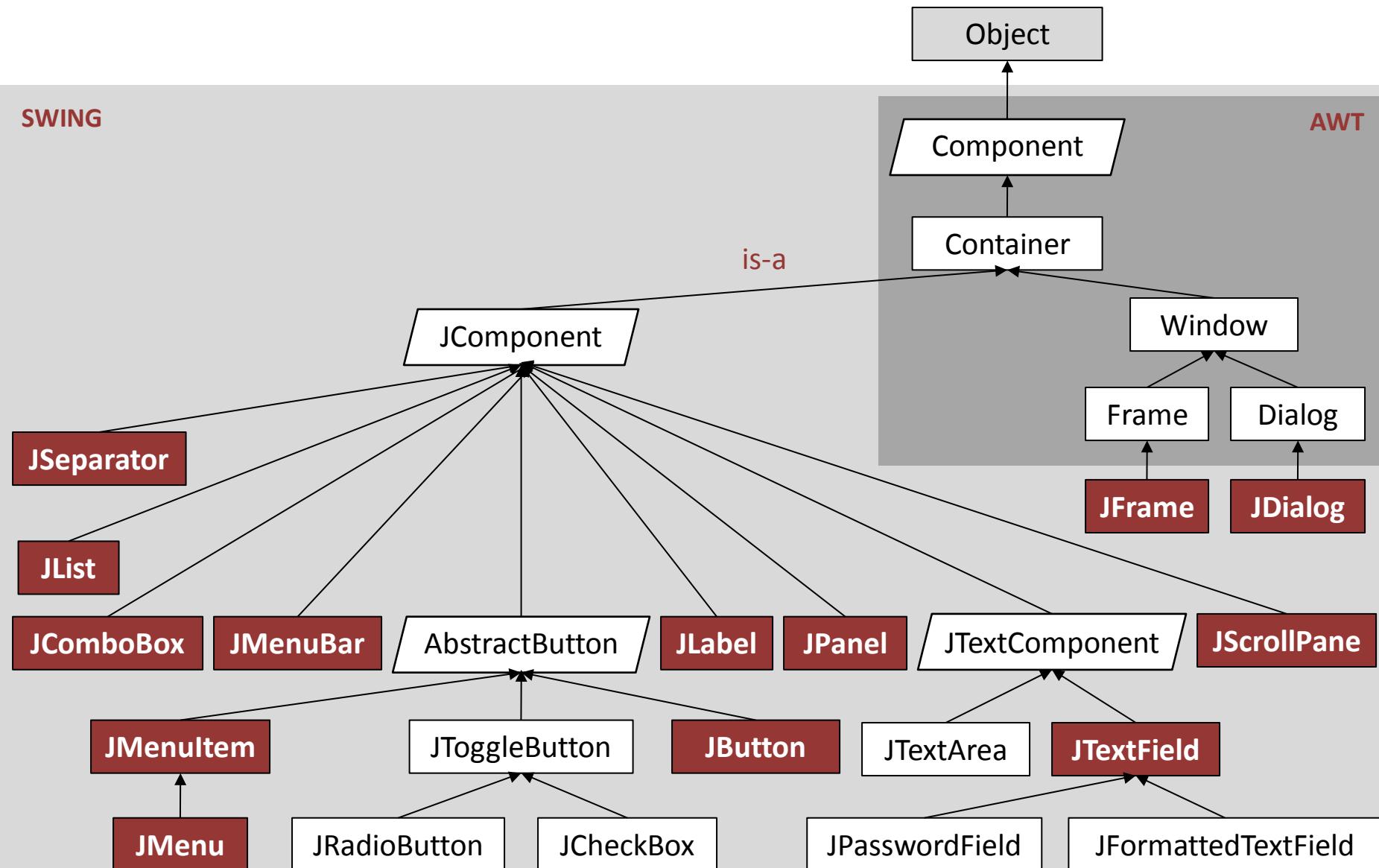
- [Component](#)

- [Container](#)

- [JComponent](#)

SWING

AWT



Legenda: Classe Abstrata

Classe Instanciável

Exercícios PPROG

- **Package**

- java.awt

- **Declaração**

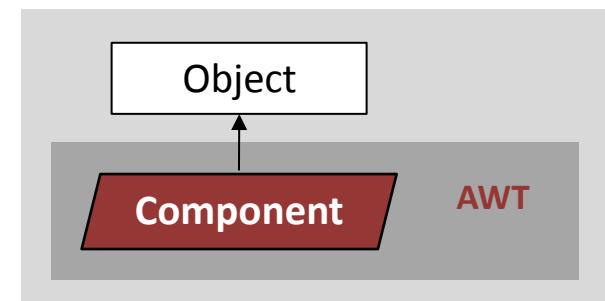
```
public abstract class Component extends Object  
        implements ImageObserver, MenuContainer, Serializable { ... }
```

- **Classe Abstrata**

- Não é instanciável

- **Topo da Hierarquia**

- Superclasse
    - Todas as classes de componentes gráficos
  - Características
    - Comuns a todos os componentes gráficos



- **Objetivo**

- Todos os objetos da hierarquia serem componentes gráficos
    - Tenham representação gráfica num monitor
    - Suportem interação com o utilizador

- Interfaces Implementadas

- ImageObserver // para permitir atualização da imagem do componente quando é alterada
- MenuContainer // relacionada com menus *popup*
- Serializable // permite o armazenamento dos componentes em ficheiro

- Campos

- Para especificar o alinhamento de componentes gráficos
- Mais usados

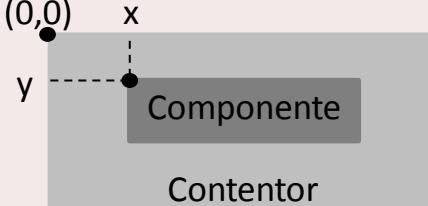
Declaração	Funcionalidade
public static final float <b>TOP_ALIGNMENT</b> = 0.0f;	Constante para especificar alinhamento no topo do componente.
public static final float <b>BOTTOM_ALIGNMENT</b> = 1.0f;	Constante para especificar alinhamento em baixo do componente.
public static final float <b>LEFT_ALIGNMENT</b> = 0.0f;	Constante para especificar alinhamento à esquerda do componente.
public static final float <b>CENTER_ALIGNMENT</b> = 0.5f;	Constante para especificar alinhamento no centro do componente.
public static final float <b>RIGHT_ALIGNMENT</b> = 1.0f;	Constante para especificar o alinhamento à direita do componente.

## Métodos (1/4)

- Relacionados com representação gráfica dos componentes
- Mais usados

Declaração	Funcionalidade
<pre>public void setMinimumSize( Dimension minimumSize )</pre> <ul style="list-style-type: none"><li>▪ Exemplo: <code>obj.setMinimumSize( new Dimension(10,30) );</code></li></ul>	Modifica o tamanho mínimo do componente; A unidade de medida é o pixel.
<pre>public void setMaximumSize( Dimension maximumSize )</pre> <ul style="list-style-type: none"><li>▪ Exemplo: <code>obj.setMaximumSize( new Dimension(20,40) );</code></li></ul>	Modifica o tamanho máximo do componente.
<pre>public void setSize( int width, int height )</pre>	Usado para modificar o tamanho de conteidores gráficos de nível superior, como por exemplo, JFrame e JDialog;  O tamanho é definido através da largura ( <i>width</i> ) e altura ( <i>height</i> ), e é modificado dinamicamente;  Estas dimensões são automaticamente aumentadas se qualquer uma delas for menor que o tamanho mínimo especificado previamente através do método <code>setMinimumSize</code> .

## Métodos (2/4)

Declaração	Funcionalidade
<pre>public void setPreferredSize( Dimension preferredSize )</pre> <ul style="list-style-type: none"> <li>Exemplo:</li> </ul> <pre>obj.setPreferredSize(new Dimension(20,40) );</pre>	<p>Serve para modificar o tamanho de um componente adicionado a um contendor gráfico com gestor de posicionamento. Neste tipo de contendor, o setSize() não modifica o tamanho de um componente.</p> <p>Para visualizar a modificação, usar o método <a href="#">revalidade()</a> do contendor.</p>
<pre>public void setBounds(int x, int y, int width, int height)</pre> 	<p>Move o componente e altera o seu tamanho;</p> <p>A nova localização do canto superior esquerdo é especificada por x e y;</p> <p>A nova dimensão é especificada por width e height.</p>
<pre>public Dimension getSize()</pre>	<p>Retorna tamanho do componente num objeto tipo Dimension.</p>

[JPanel](#)   [JLayeredPane](#)   [JRootPane](#)   [JScrollPane](#)   [JFrame](#)   [JDialog](#)   [JFileChooser](#)   [JButton](#)   [JLabel](#)   [JTextField](#)   [JList](#)   [JComboBox](#)   [JCheckBox](#)

[JMenu](#)   [JMenuItem](#)   [JMenuBar](#)   [BorderLayout](#)   [FlowLayout](#)

## ■ Métodos (3/4)

Declaração	Funcionalidade
public void <b>setLocation</b> ( int x, int y )	<p>Move o componente, colocando o seu canto superior esquerdo no ponto (x,y) do sistema de coordenadas do seu painel contentor;</p> <p>O ponto (0,0) de um componente é o seu canto superior esquerdo.</p>
<p>public void <b>setFont</b>( Font font )</p> <ul style="list-style-type: none"> <li>▪ Exemplo:</li> </ul> <pre>obj.setFont( new Font ("Arial", Font.BOLD + Font.ITALIC, 20 ) );</pre>	Especifica a <i>font</i> do componente.
<p>public void <b>setVisible</b>(boolean b)</p> <ul style="list-style-type: none"> <li>▪ Exemplo:</li> </ul> <pre>obj.setVisible(true) ou obj.setVisible(false)</pre>	Mostra ( <b>setVisible(true)</b> ) ou esconde ( <b>setVisible(false)</b> ) componente.
<p>public void <b>setEnabled</b>( boolean b )</p> <ul style="list-style-type: none"> <li>▪ Exemplo:</li> </ul> <pre>obj.setEnable(true) ou obj.setEnable(false)</pre>	Desinibe ou inibe componente de responder a ações do utilizador (interação).

[JPanel](#)   [JLayeredPane](#)   [JRootPane](#)   [JScrollPane](#)   [JFrame](#)   [JDialog](#)   [JFileChooser](#)   [JButton](#)   [JLabel](#)   [JTextField](#)   [JList](#)   [JComboBox](#)   [JCheckBox](#)

[JMenu](#)   [JMenuItem](#)   [JMenuBar](#)   [BorderLayout](#)   [FlowLayout](#)

## Métodos (4/4)

Declaração	Funcionalidade
<pre>public void setBackground( Color bg )</pre> <ul style="list-style-type: none"> <li>▪ Exemplo:</li> </ul> <pre>obj.setBackground(Color.RED);</pre>	Especifica a cor de fundo do componente.
<pre>public void setForeground( Color fg )</pre> <ul style="list-style-type: none"> <li>▪ Exemplos:</li> </ul> <pre>btnOK.setForeground( Color.RED );</pre> <pre>lblNome.setForeground( Color.red );</pre>	Especifica a cor do conteúdo do componente.
<pre>public void revalidate( )</pre> <ul style="list-style-type: none"> <li>▪ Exemplo: painel adicionado após clique num botão</li> </ul> <pre>...</pre> <pre>add(painel, BorderLayout.NORTH);</pre> <pre>painel.revalidate(); // para visualizar o painel</pre>	<p>Útil para visualizar a modificação dinâmica de um componente gráfico. Por exemplo, para ver um novo painel de componentes adicionado a uma janela, depois de acionado um botão de comando;</p> <p>Revalida a hierarquia de componentes até ao componente raíz validado mais próximo;</p>

[JPanel](#)   [JLayeredPane](#)   [JRootPane](#)   [JScrollPane](#)   [JFrame](#)   [JDialog](#)   [JFileChooser](#)   [JButton](#)   [JLabel](#)   [JTextField](#)   [JList](#)   [JComboBox](#)   [JCheckBox](#)

[JMenu](#)   [JMenuItem](#)   [JMenuBar](#)   [BorderLayout](#)   [FlowLayout](#)

- **Objetivo**

- Tornar componentes em contentores de outros componentes gráficos

- **Package**

- java.awt

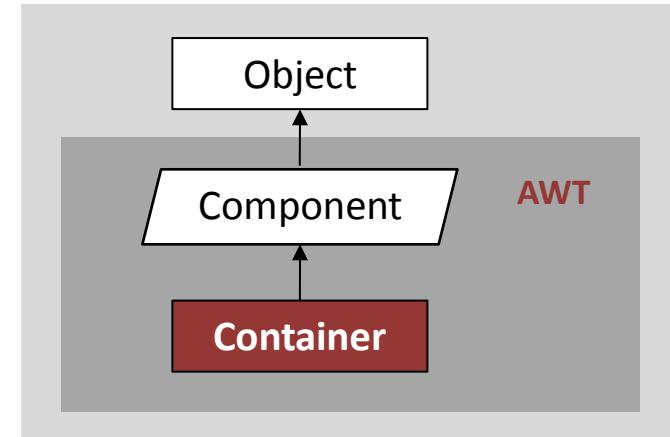
- **Declaração**

```
public class Container extends Component { ... }
```

- **Classe Instanciável**

- **Componentes Adicionados**

- Registados numa lista
    - Ordem na lista define
      - Ordem dos componentes dentro do contentor
        - Frente-para-Trás
  - Sem especificar índice
    - Inseridos no fim da lista
      - Debaixo dos componentes adicionados anteriormente



## Métodos Próprios

- Relacionados com operações típicas de contentores
- Mais usados

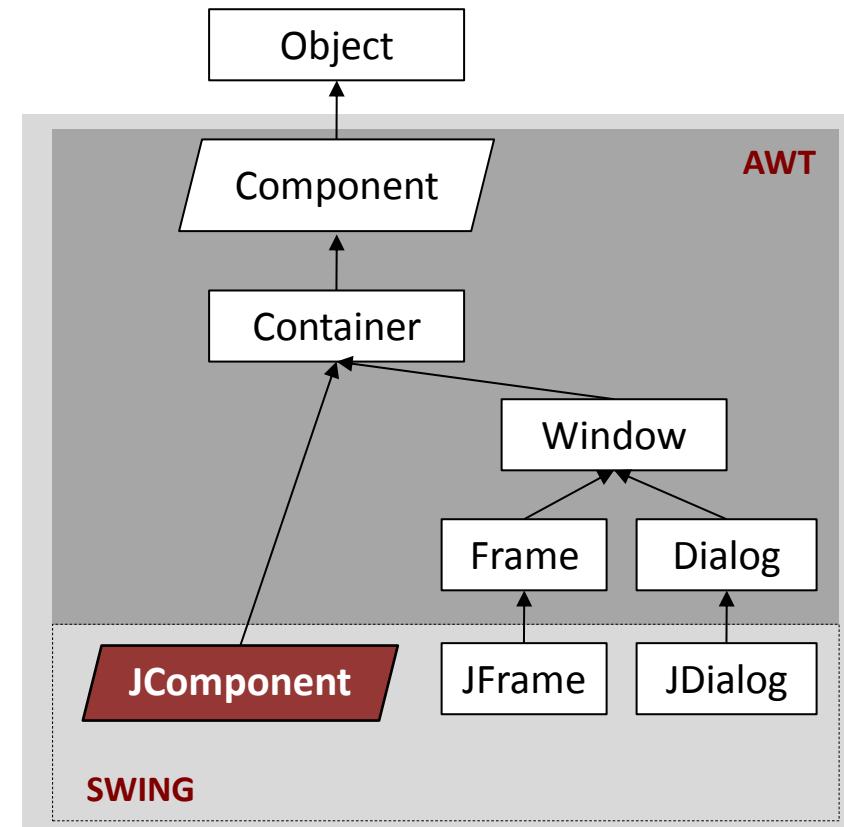
Declaração	Funcionalidade
<pre>public Component add(Component comp)     ▪ Exemplo: obj.add( new JPanel() );</pre>	Adiciona o componente especificado no final do contentor.
<pre>public Component add(Component comp, int index)</pre>	Adiciona o componente especificado ao painel (contentor) na posição indicada.
<pre>public Component getComponent(int n)</pre>	Retorna o componente de índice n do contentor.
<pre>public Component[] getComponents()</pre>	Retorna todos os componentes do contentor.
<pre>public void remove(Component comp)</pre>	Remove do contentor o componente comp especificado.
<pre>public void remove(int index)</pre>	Remove o componente de índice index do contentor.
<pre>public void removeAll()</pre>	Remove todos os componentes adicionados ao contentor.
<pre>public void setLayout( LayoutManager manager )     ▪ Exemplo: obj.setLayout(new FlowLayout());</pre>	Altera o gestor de posicionamento.

- **Objetivo**

- Superclasse de todos os componentes **Swing**
  - Exceções
    - Contentores de nível superior
      - JFrame
      - JDialog

- **Package**

- javax.swing



- **Declaração**

```
public abstract class JComponent extends Container  
    implements Serializable { ... }
```

- **Classe Abstrata**

- Não é instanciável

## Métodos Próprios

- Mais usados

Declaração	Funcionalidade
<pre>public void setBorder(Border border)</pre> <ul style="list-style-type: none"> <li>Exemplos:           <ul style="list-style-type: none"> <li>Bordo Vazio:  <code>painel.setBorder( new EmptyBorder(10,5,5,10) );</code>  <code>(10,5,5,10) = (topo, esquerda, base, direita)</code> </li> <li>Bordo com Título:  <code>painel.setBorder( BorderFactory.CreateTitleBorder("Título") );</code>  </li> </ul> </li> </ul>	Especifica um bordo.
<pre>public void requestFocus()</pre> <ul style="list-style-type: none"> <li>Exemplos:           <ul style="list-style-type: none"> <li><code>txtNome.requestFocus();</code></li> <li><code>btnOK.requestFocus();</code></li> </ul> </li> </ul>	Pede para componente adquirir foco de entrada (teclado).

- [Introdução](#)
- [Componentes Gráficos](#)
  - [Introdução](#)
  - [Hierarquia de Classes](#)
  - [Interfaces](#)
  - [Categorias](#)
    - [Contentores de Componentes Gráficos](#)
    - [Apresentação de Informação](#)
    - [Controlos Básicos](#)
- [Gestores de Posicionamento](#)
- [Manipuladores de Eventos](#)
- [Bibliografia Geral](#)
- [Índice Remissivo](#)



## ▪ Declaração

```
public interface SwingConstants {  
    public static final int CENTER = 0;  
    public static final int TOP = 1;  
    public static final int LEFT = 2;  
    public static final int BOTTOM = 3;  
    public static final int RIGHT = 4;  
    public static final int NORTH = 1;  
    public static final int NORTH_EAST = 2;  
    public static final int EAST = 3;  
    public static final int SOUTH_EAST = 4;  
    public static final int SOUTH = 5;  
    public static final int SOUTH_WEST = 6;  
    public static final int WEST = 7;  
    public static final int NORTH_WEST = 8;  
    public static final int HORIZONTAL = 0;  
    public static final int VERTICAL = 1;  
    public static final int LEADING = 10;  
    public static final int TRAILING = 11;  
    public static final int NEXT = 12;  
    public static final int PREVIOUS = 13;  
}
```

## Implementado nas Classes

- JLabel
- AbstractButton
- JTextField

- Declaração

```
public interface WindowConstants {  
    public static final int DO_NOTHING_ON_CLOSE = 0;  
    public static final int HIDE_ON_CLOSE = 1;  
    public static final int DISPOSE_ON_CLOSE = 2;  
    public static final int EXIT_ON_CLOSE = 3;  
}
```

**Implementado nas classes**

- JPanel
- JFrame

- [Introdução](#)
- [Componentes Gráficos](#)
  - [Introdução](#)
  - [Hierarquia de Classes](#)
  - [Interfaces](#)
  - [Categorias](#)
    - [Contentores de Componentes Gráficos](#)
    - [Apresentação de Informação](#)
    - [Controlos Básicos](#)
- [Gestores de Posicionamento](#)
- [Manipuladores de Eventos](#)
- [Bibliografia Geral](#)
- [Índice Remissivo](#)



- [Contentores de Componentes Gráficos](#)

- [Painéis](#)

- [Noção de Painel](#)
- [Classe JPanel](#)
- [Classe JLayeredPane](#)
- [Classe JRootPane](#)
- [Classe JScrollPane](#)

- [Superclasse Window](#)

- [Janelas](#)

- [Superclasse Frame](#)
- [Classe JFrame](#)

- [Caixas de Diálogo](#)

- [Introdução](#)
- [Classe JOptionPane](#)
- [Classe JFileChooser](#)
- [Próprias](#)
  - [Superclasse Dialog](#)
  - [Classe JDialog](#)

## ■ Contentores de Componentes Gráficos

### ■ Painéis

- [Noção de Painel](#)
- [Classe JPanel](#)
- [Classe JLayeredPane](#)
- [Classe JRootPane](#)
- [Classe JScrollPane](#)

### ■ Superclasse Window

### ■ Janelas

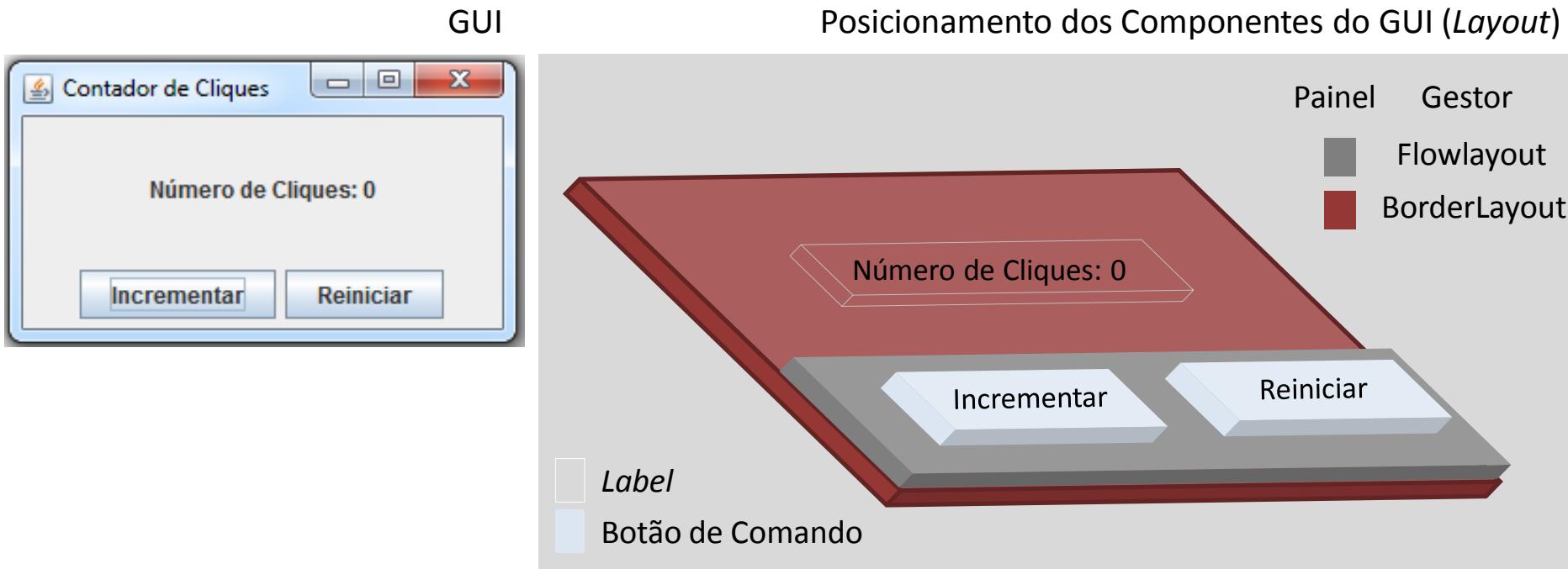
- [Superclasse Frame](#)
- [Classe JFrame](#)

### ■ Caixas de Diálogo

- [Introdução](#)
- [Classe JOptionPane](#)
- [Classe JFileChooser](#)
- Próprias
  - [Superclasse Dialog](#)
  - [Classe JDialog](#)



- Painel
  - Serve de contentor de componentes gráficos // pode armazenar também painéis
- Interesse dos Painéis
  - Permitir posicionamento preciso de componentes gráficos na GUI
    - Posicionamento: feito por objetos **gestores de posicionamento** associados ao painel
    - Precisão: encaixando painéis com diferentes gestores de posicionamento
- Exemplo



## ▪ Contentores de Componentes Gráficos

### ▪ Painéis

- Noção de Painel
- Classe JPanel
- Classe JLayeredPane
- Classe JRootPane
- Classe JScrollPane



### ▪ Superclasse Window

### ▪ Janelas

- Superclasse Frame
- Classe JFrame

### ▪ Caixas de Diálogo

- Introdução
- Classe JOptionPane
- Classe JFileChooser
- Próprias
  - Superclasse Dialog
  - Classe JDialog

■ Package

- javax.swing

## ■ Declaração

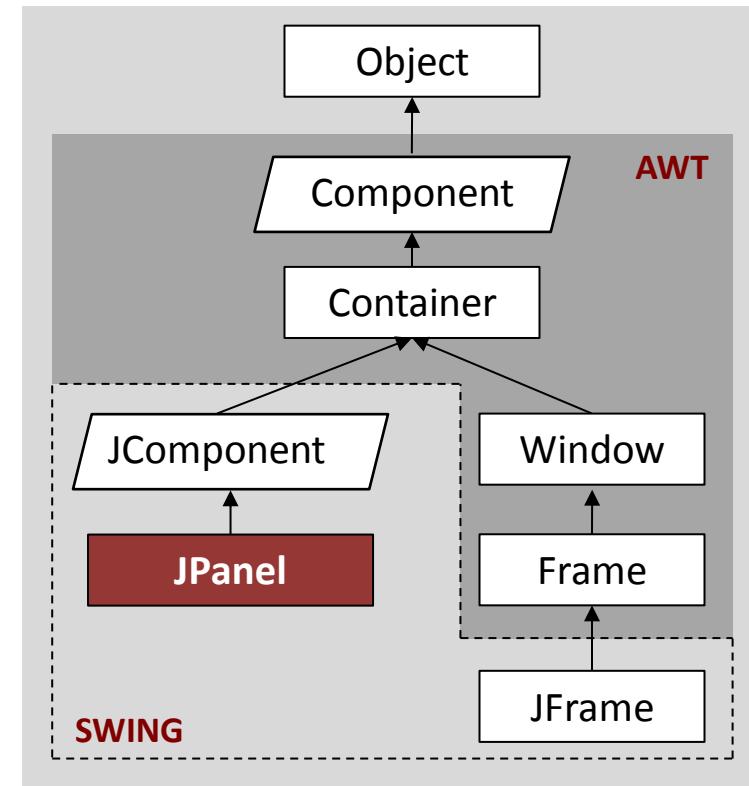
```
public class JPanel extends JComponent  
    implements Accessible { ... }
```

■ Classe Instanciável

- Objetos são painéis

## ■ Gestor de Posicionamento por Omissão

- FlowLayout



Declaração	Funcionalidade
<pre>public JPanel()</pre> <ul style="list-style-type: none"><li data-bbox="130 328 341 361">▪ Exemplo:</li><li data-bbox="168 390 616 423">JPanel p1 = new JPanel();</li></ul>	Cria painel com gestor de posicionamento <i>FlowLayout</i> .
<pre>public JPanel( LayoutManager layout )</pre> <ul style="list-style-type: none"><li data-bbox="130 573 341 606">▪ Exemplo:</li><li data-bbox="168 635 975 668">JPanel p2 = new JPanel( new BorderLayout() );</li></ul>	Cria painel com gestor de posicionamento <i>layout</i> passado por parâmetro.

- **Herdados**

- [Component](#)
- [Container](#)
- [JComponent](#)

- **Próprios**

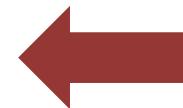
- Pouco usados

- <http://docs.oracle.com/javase/tutorial/uiswing/components/panel.html>

## ▪ Contentores de Componentes Gráficos

### ▪ Painéis

- Noção de Painel
- Classe JPanel
- Classe JLayeredPane
- Classe JRootPane
- Classe JScrollPane



### ▪ Superclasse Window

### ▪ Janelas

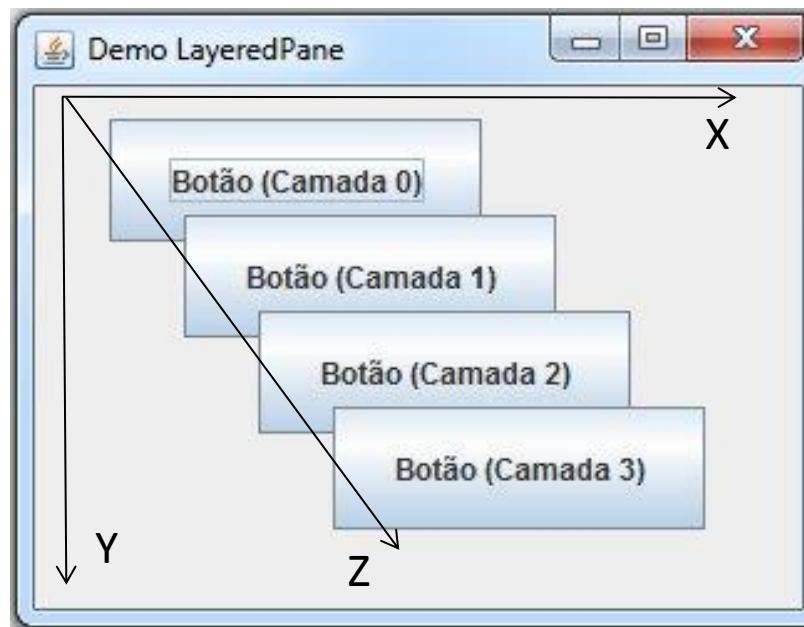
- Superclasse Frame
- Classe JFrame

### ▪ Caixas de Diálogo

- Introdução
- Classe JOptionPane
- Classe JFileChooser
- Próprias
  - Superclasse Dialog
  - Classe JDialog

## ■ Posicionar Componentes em Camadas

- Usa 3<sup>a</sup> dimensão
  - Designações
    - Profundidade
    - Ordem Z
  - Representação
    - Índice
    - Valor Inteiro



## ■ Exemplo

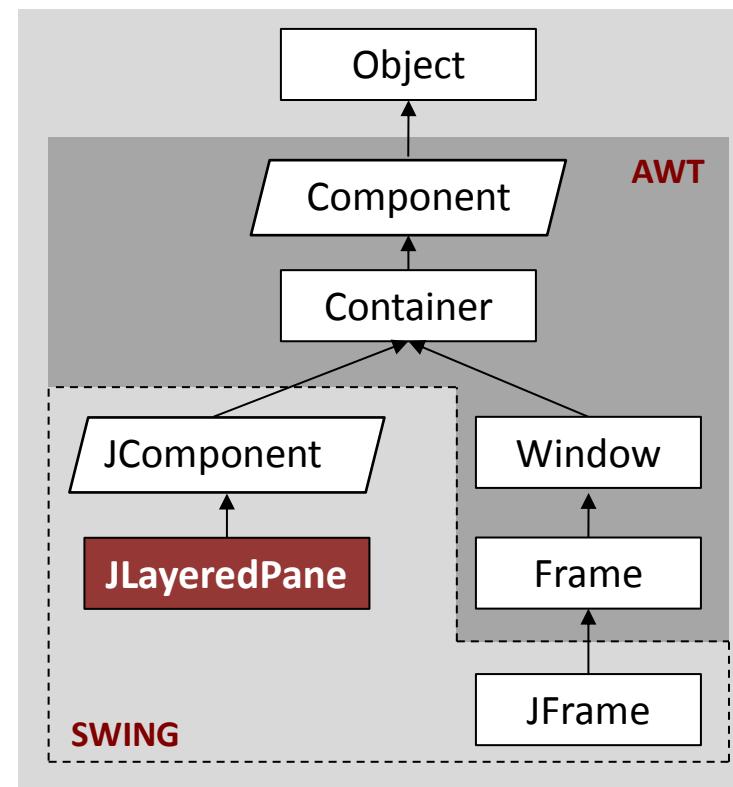


## ■ Package

- javax.swing

## ■ Declaração

```
public class JLayeredPane extends JComponent  
    implements Accessible { ... }
```

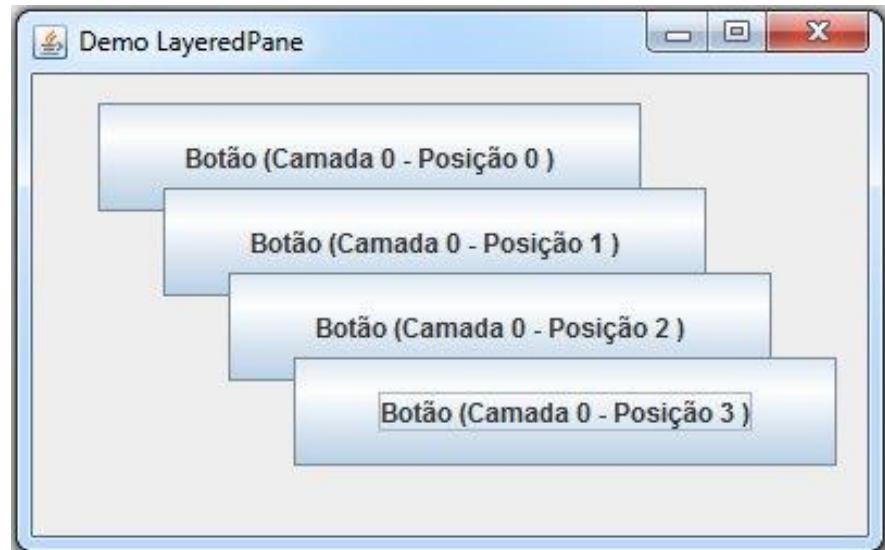


Declaração	Funcionalidade
<pre>public JLayeredPane()</pre> <p>■ Exemplo:</p> <pre>JLayeredPane lp = new JLayeredPane();</pre>	Cria painel JLayeredPane.

## ■ Herdados

- [Component](#)
- [Container](#)
- [JComponent](#)

## ■ Próprios



Declaração	Funcionalidade
<pre>public void moveToBack(Component c)</pre>	Move componente c para baixo de todos os componentes da sua camada corrente (posição -1).
<pre>public void moveToFront(Component c)</pre>	Move componente c para cima de todos os componentes da sua camada corrente (posição 0).
<pre>public void setLayer(Component c, int layer, int position)</pre>	Especifica camada do componente c e a sua posição dentro da camada.

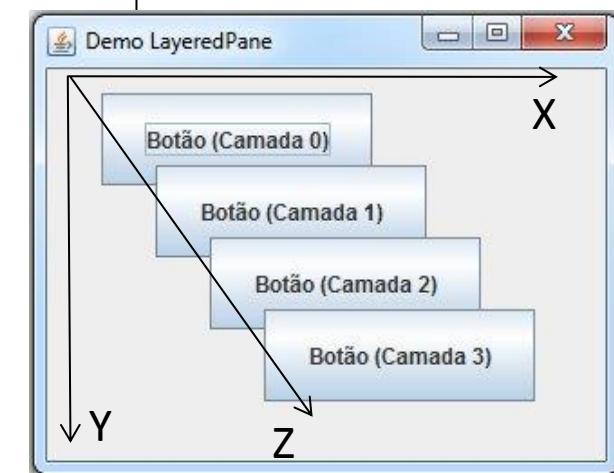
- Requer
  - Especificação da camada // profundidade

- Em Componentes Sobrepostos
  - Componente no Topo = Componente com Maior profundidade

```
import javax.swing.*;
public class DemoLayeredPanel extends JFrame {
    private static final int JANELA_LARGURA = 300, JANELA_ALTURA = 250;
    public DemoLayeredPanel() {
        super("Demo LayeredPane");
        JLAYEREDPANE lp = criarPainel();
        add(lp);
        setSize(JANELA_LARGURA, JANELA_ALTURA);
        setVisible(true);
    }

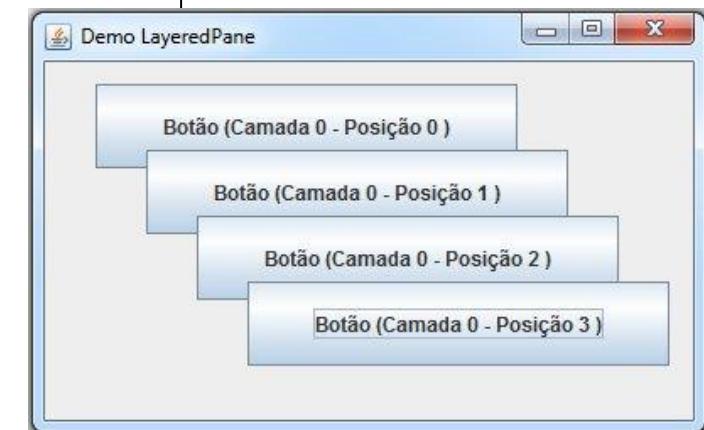
    private JLAYEREDPANE criarPainel() {
        JLAYEREDPANE lp = new JLAYEREDPANE();
        JButton[] botoes = new JButton[4];
        for (int i = 0; i < 4; i++) {
            botoes[i] = new JButton("Botão (Camada " + i + ")");
            botoes[i].setBounds(30 * (i + 1), 13 * ((i * 3) + 1), 150, 50);
            lp.add(botoes[i], i);
        }
        return lp;
    }

    public static void main(String[] args) {
        new DemoLayeredPanel();
    }
}
```



- Pode conter
  - Múltiplos componentes gráficos
- Cada Componente
  - Ocupa uma posição
  - Pode ser movido para baixo ou para cima de todos os outros // da mesma camada

```
import javax.swing.*;  
public class DemoLayeredPane extends JFrame {  
    private static final int JANELA_LARGURA = 400, JANELA_ALTURA = 250;  
    public DemoLayeredPane() {  
        super("Demo LayeredPane");  
        JLAYEREDPANE lp = criarPainel();  
        add(lp);  
        setSize(JANELA_LARGURA, JANELA_ALTURA);  
        setVisible(true);  
    }  
  
    private JLAYEREDPANE criarPainel() {  
        JLAYEREDPANE lp = new JLAYEREDPANE();  
        JButton[] botoes = new JButton[4];  
        for (int i = 0; i < 4; i++) {  
            botoes[i] = new JButton("Botão (Camada 0 Posição " + i + " )");  
            botoes[i].setBounds(30 * (i + 1), 13 * ((i * 3) + 1), 250, 50);  
            lp.add(botoes[i], 0);  
        }  
        return lp;  
    }  
  
    public static void main(String[] args) {  
        new DemoLayeredPane();  
    }  
}
```

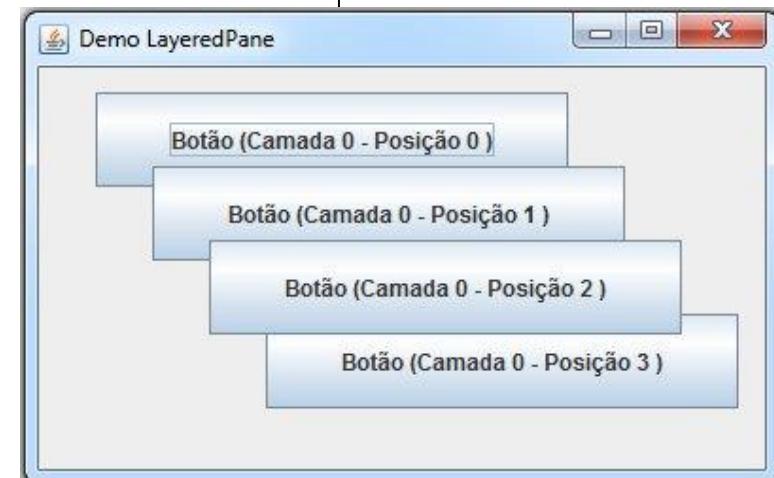


- Componente colocado por cima dos restantes componentes da mesma camada

```
import javax.swing.*;
public class DemoLayeredPane extends JFrame {
    private static final int JANELA_LARGURA = 400, JANELA_ALTURA = 250;
    public DemoLayeredPane() {
        super("Demo LayeredPane");
        JLAYEREDPANE lp = criarPainel();
        add(lp);
        setSize(JANELA_LARGURA, JANELA_ALTURA);
        setVisible(true);
    }

    private JLAYEREDPANE criarPainel() {
        JLAYEREDPANE lp = new JLAYEREDPANE();
        JButton[] botoes = new JButton[4];
        for (int i = 0; i < 4; i++) {
            botoes[i] = new JButton("Botão (Camada 0 - Posição " + i + " )");
            botoes[i].setBounds(30 * (i + 1), 13 * ((i * 3) + 1), 250, 50);
            lp.add(botoes[i], 0);
        }
        lp.moveToFront(botoes[2]); ←
        return lp;
    }

    public static void main(String[] args) {
        DemoLayeredPane gui = new DemoLayeredPane();
    }
}
```



- Em Componentes Gráficos

- [JFrame](#)
- [JDialog](#)

- <http://download.oracle.com/javase/tutorial/uiswing/components/layeredpane.html>

- [Contentores de Componentes Gráficos](#)

- [Painéis](#)

- [Noção de Painel](#)
- [Classe JPanel](#)
- [Classe JLayeredPane](#)
- [Classe JRootPane](#)
- [Classe JScrollPane](#)



- [Superclasse Window](#)

- [Janelas](#)

- [Superclasse Frame](#)
- [Classe JFrame](#)

- [Caixas de Diálogo](#)

- [Introdução](#)
- [Classe JOptionPane](#)
- [Classe JFileChooser](#)
- [Próprias](#)
  - [Superclasse Dialog](#)
  - [Classe JDialog](#)

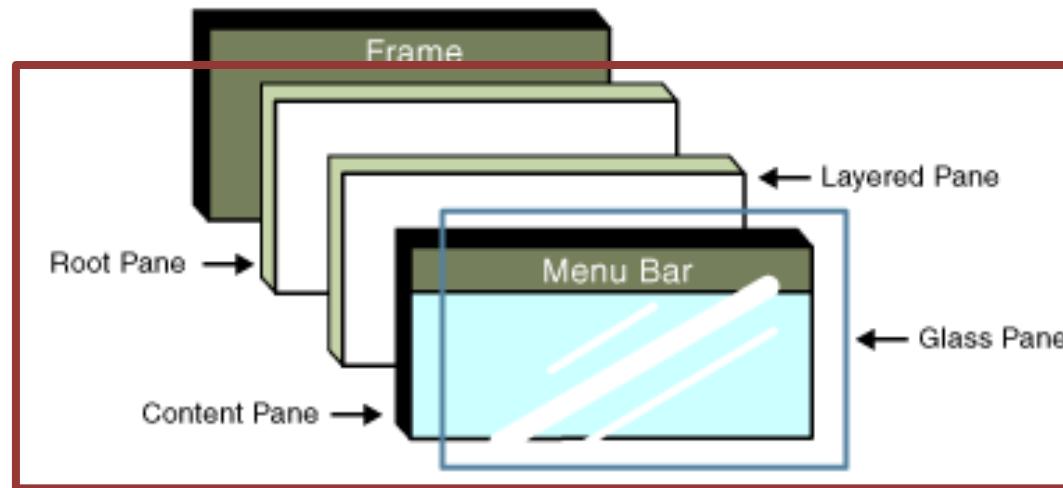
- **Geralmente**

- Não é instanciada

- **Objetos Usados nos Componentes Gráficos**

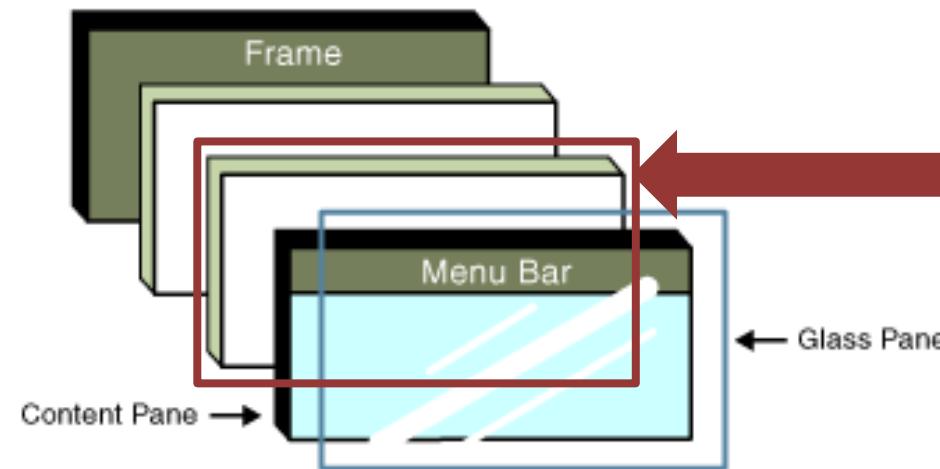
- Janelas Internas
  - Contentores de Nível Superior
    - JFrame
    - JDialog
    - JApplet

- Root Pane contém
  - Painéis
    - Layered Pane
    - Content Pane
    - Glass Pane
  - Barra de Menus // Opcional



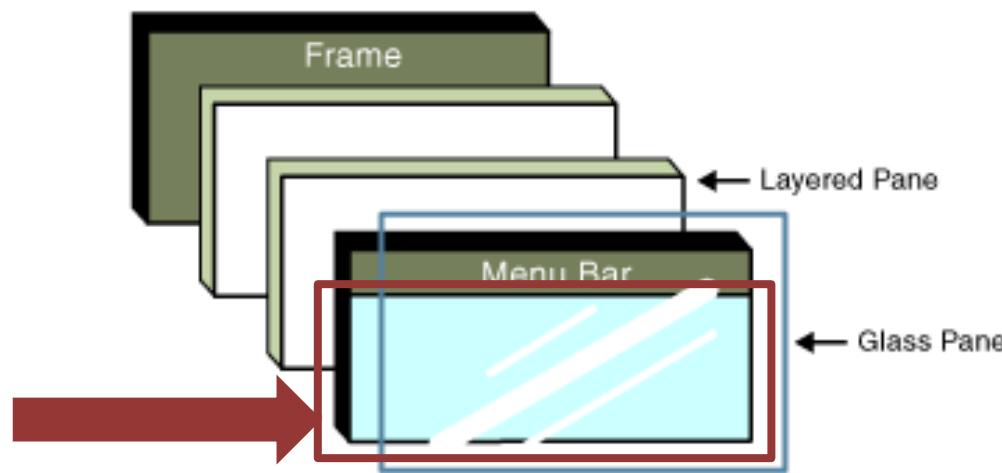
- Interesse

- Posicionar os componentes
  - Content Pane
  - Barra de Menus // opcional
- Adicionalmente
  - Posicionar componentes em camadas



- Interesse

- Contentor dos componentes gráficos visíveis
  - Excluindo Barra de Menus



- Por Omissão

- Invisível

- Quando Visível

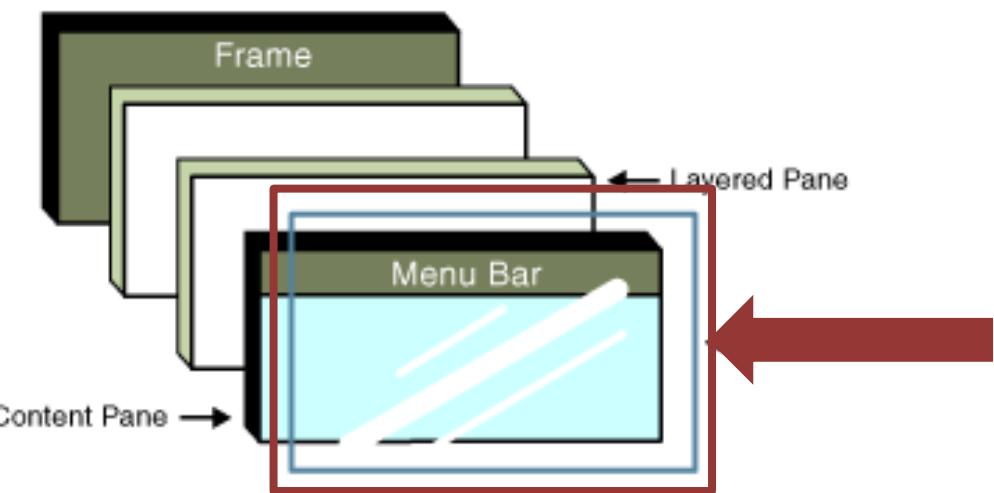
- Está à frente dos outros painéis
  - É Transparente
    - Semelhante ao Vidro

- Capacidade

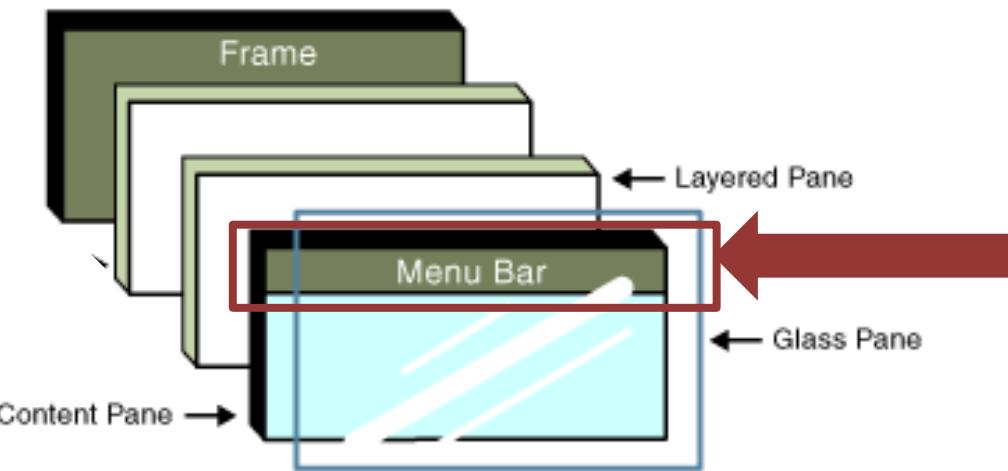
- Capturar Eventos de Entrada

- Interesse

- Capturar eventos do root pane numa **área** que contém **múltiplos componentes gráficos**
    - Exemplo
      - Desativar eventos do rato numa área com vários componentes
  - Pintar **área** que contém **múltiplos componentes gráficos**
    - Exemplo
      - Mostrar imagem sobre vários componentes



- Opcional
- Interesse
  - Contentor de Menus

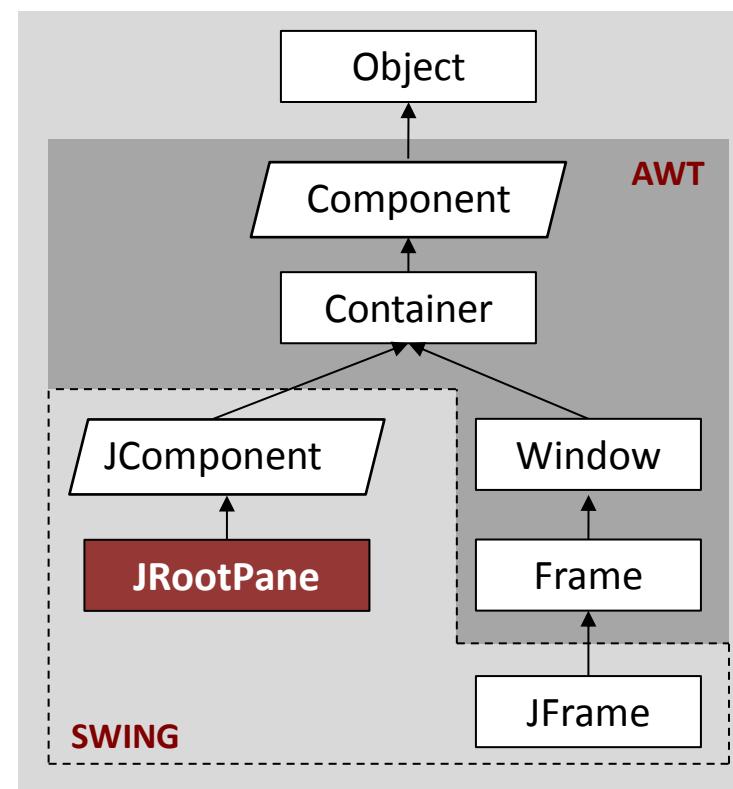


## ■ Package

- javax.swing

## ■ Declaração

```
public class JRootPane extends JComponent  
    implements Accessible { ... }
```



Declaração	Funcionalidade
<pre>public JRootPane()</pre> <ul style="list-style-type: none"><li>▪ Exemplo:<pre>JRootPane rp = new JRootPane();</pre></li></ul>	Cria painel JRootPane.

## ▪ Herdados

- [Component](#)
- [Container](#)
- [JComponent](#)

## ▪ Próprios

Declaração	Funcionalidade
<code>public void setJMenuBar(JMenuBar menubar)</code>	Especifica a barra de menus da janela.
<code>public JMenuBar getJMenuBar()</code>	Retorna barra de menus.
<code>public Container getContentPane()</code>	Retorna Content Pane.
<code>public Component getGlassPane()</code>	Retorna Glass Pane.
<code>public JLayeredPane getLayeredPane()</code>	Retorna Layered Pane.
<code>public void setDefaultButton(JButton defaultButton)</code>	Especifica botão que será acionado quando é premida tecla ENTER, independentemente de ser, ou não, o foco do teclado.

- <http://docs.oracle.com/javase/tutorial/uiswing/components/rootpane.html>

- [Contentores de Componentes Gráficos](#)

- [Painéis](#)

- [Noção de Painel](#)
- [Classe JPanel](#)
- [Classe JLayeredPane](#)
- [Classe JRootPane](#)
- [Classe JScrollPane](#)



- [Superclasse Window](#)

- [Janelas](#)

- [Superclasse Frame](#)
- [Classe JFrame](#)

- [Caixas de Diálogo](#)

- [Introdução](#)
- [Classe JOptionPane](#)
- [Classe JFileChooser](#)
- [Próprias](#)
  - [Superclasse Dialog](#)
  - [Classe JDialog](#)

- Visualizar Componente Gráfico Grande

- Área visualização < Área componente
- Deslocando vista
  - Vertical
  - Horizontal

- Exemplos



Imagen  
dentro de um  
JSScrollPane



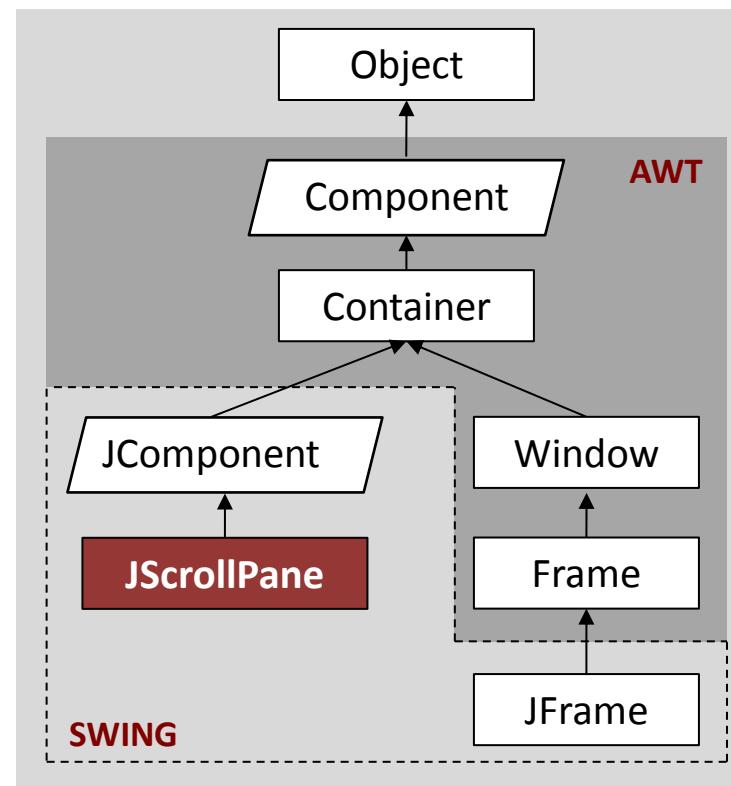
Caixa de Listagem  
dentro de um  
JSScrollPane

## ■ Package

- javax.swing

## ■ Declaração

```
public class JScrollPane extends JComponent  
    implements ScrollPaneConstants, Accessible { ... }
```



Declaração	Funcionalidade
<pre>public JScrollPane()</pre> <ul style="list-style-type: none"><li data-bbox="92 303 303 342">▪ Exemplo:</li><pre>JScrollPane sp = new JScrollPane();</pre></ul>	Cria painel JScrollPane vazio, onde as barras de deslocamento vertical e horizontal surgirão apenas quando forem necessárias.
<pre>public JScrollPane( Component view )</pre> <ul style="list-style-type: none"><li data-bbox="92 548 303 587">▪ Exemplo:</li><pre>JList lstNomes = new Jlist(); ... JScrollPane sp = new JScrollPane( lstNomes );</pre></ul>	Cria painel JScrollPane que mostra o componente especificado, onde as barras de deslocamento vertical e horizontal surgirão sempre que o conteúdo do componente for maior do que a vista proporcionada.

## ▪ Herdados

- [Component](#)
- [Container](#)
- [JComponent](#)

## ▪ Próprio

- Exemplo

Declaração	Funcionalidade
<pre>public void setWheelScrollingEnabled( boolean handleWheel )</pre>	Inibe/desinibe o deslocamento controlado pelo movimento da roda do rato.

- <http://docs.oracle.com/javase/tutorial/uiswing/components/scrollpane.html>

- [Contentores de Componentes Gráficos](#)

- [Painéis](#)

- [Noção de Painel](#)
- [Classe JPanel](#)
- [Classe JLayeredPane](#)
- [Classe JRootPane](#)
- [Classe JScrollPane](#)

- [Superclasse Window](#)



- [Janelas](#)

- [Superclasse Frame](#)
- [Classe JFrame](#)

- [Caixas de Diálogo](#)

- [Introdução](#)
- [Classe JOptionPane](#)
- [Classe JFileChooser](#)
- [Próprias](#)
  - [Superclasse Dialog](#)
  - [Classe JDialog](#)

- Criar Janelas de Nível Superior

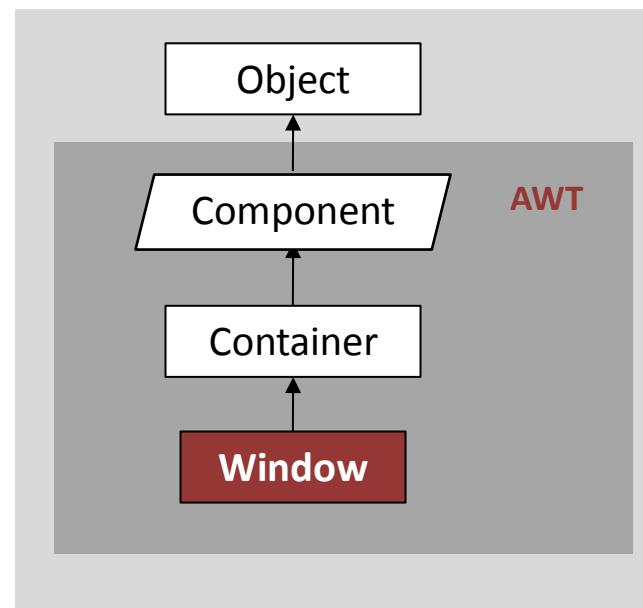
- Sem Moldura
- Sem Barra de Menus

## ■ Package

- java.awt

## ■ Declaração

```
public class Window extends Container  
    implements Accessible { ... }
```



## ■ Herdados

- [Component](#)
- [Container](#)

## ■ Próprios

Declaração	Funcionalidade
<code>public void pack()</code>	Modifica tamanho da Janela para mostrar todos os componentes com o seu tamanho preferido; Deve ser chamado após adição de todos os componentes.
<code>public void dispose()</code>	Liberta todos os recursos nativos do monitor usados pela janela, seus sub-componentes. Fecha a aplicação no caso da janela ser a janela principal da aplicação.
<code>public void setLocationRelativeTo( Component c )</code>	Chamada setLocationRelativeTo(null) coloca a janela no centro do ecrã; Deve ser invocado depois da definição do tamanho da janela ( pack() ou setSize() ).

- [Contentores de Componentes Gráficos](#)

- [Painéis](#)

- [Noção de Painel](#)
- [Classe JPanel](#)
- [Classe JLayeredPane](#)
- [Classe JRootPane](#)
- [Classe JScrollPane](#)

- [Superclasse Window](#)

- [Janelas](#)

- [Superclasse Frame](#)
- [Classe JFrame](#)

- [Caixas de Diálogo](#)

- [Introdução](#)
- [Classe JOptionPane](#)
- [Classe JFileChooser](#)
- [Próprias](#)
  - [Superclasse Dialog](#)
  - [Classe JDialog](#)

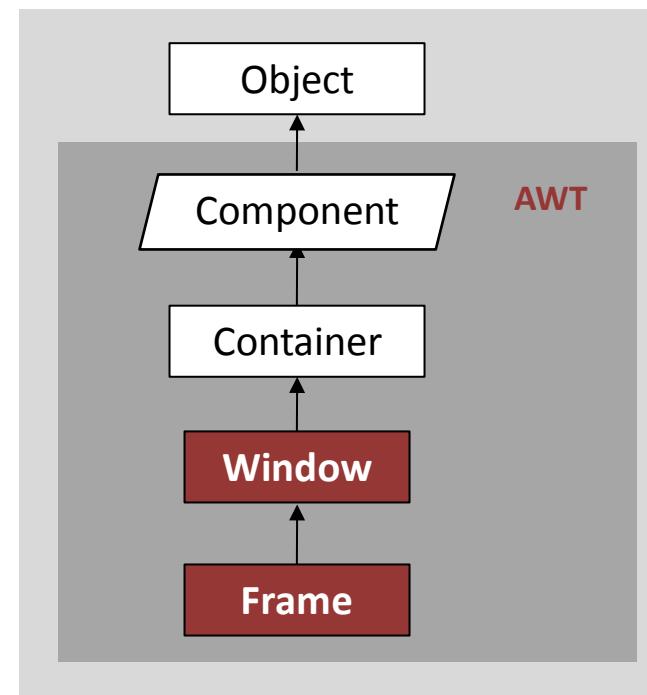


## ■ Package

- java.awt

## ■ Declaração

```
public class Frame extends Window  
    implements MenuContainer { ... }
```



## ▪ Herdados

- [Component](#)
- [Container](#)
- [Window](#)

## ▪ Próprios

- Mais usados

Declaração	Funcionalidade
<pre>public synchronized void setExtendedState( int state )</pre> <ul style="list-style-type: none"><li>▪ Exemplo para maximizar janela: <code>setExtendedState(Frame.MAXIMIZED_BOTH);</code></li></ul>	
<pre>public void setUndecorated( boolean undecorated)</pre> <ul style="list-style-type: none"><li>▪ <a href="#">Exemplo</a></li></ul>	Inibe/desinibe moldura da janela; Só pode ser chamado quando a janela não está visível.

- [Contentores de Componentes Gráficos](#)

- [Painéis](#)

- [Noção de Painel](#)
- [Classe JPanel](#)
- [Classe JLayeredPane](#)
- [Classe JRootPane](#)
- [Classe JScrollPane](#)

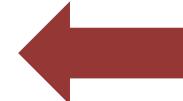
- [Superclasse Window](#)

- [Janelas](#)

- [Superclasse Frame](#)
- [Classe JFrame](#)

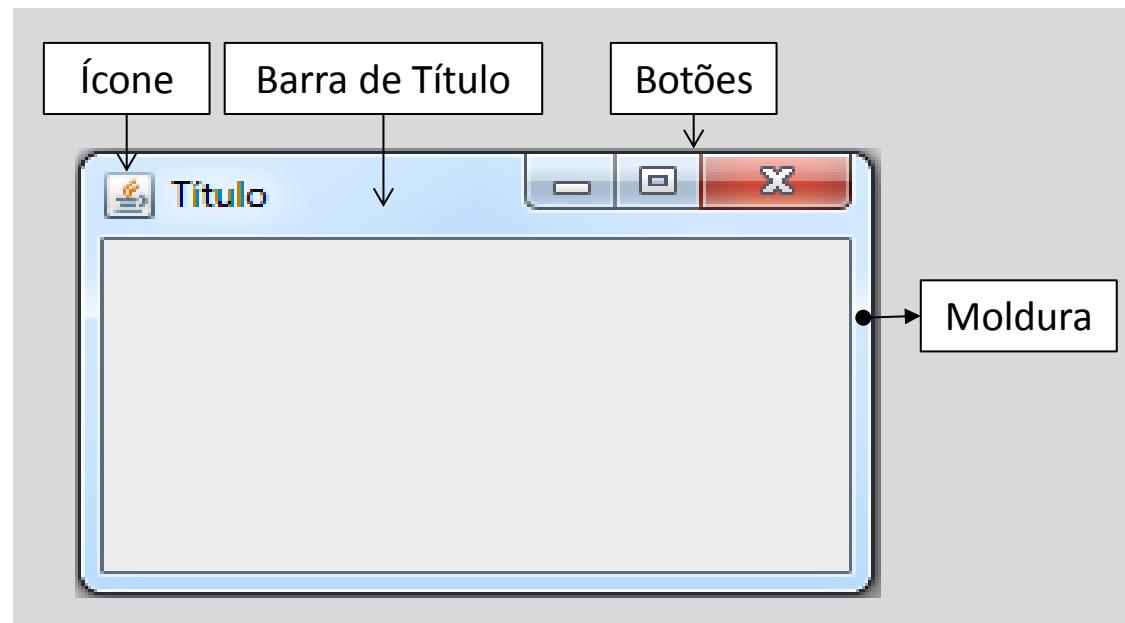
- [Caixas de Diálogo](#)

- [Introdução](#)
- [Classe JOptionPane](#)
- [Classe JFileChooser](#)
- [Próprias](#)
  - [Superclasse Dialog](#)
  - [Classe JDialog](#)



## ■ Criar Janelas

- Contentores de Componentes Gráficos
  - Nível Superior
    - Não pode estar dentro de outro componente
- Exemplo
  - Representação Gráfica

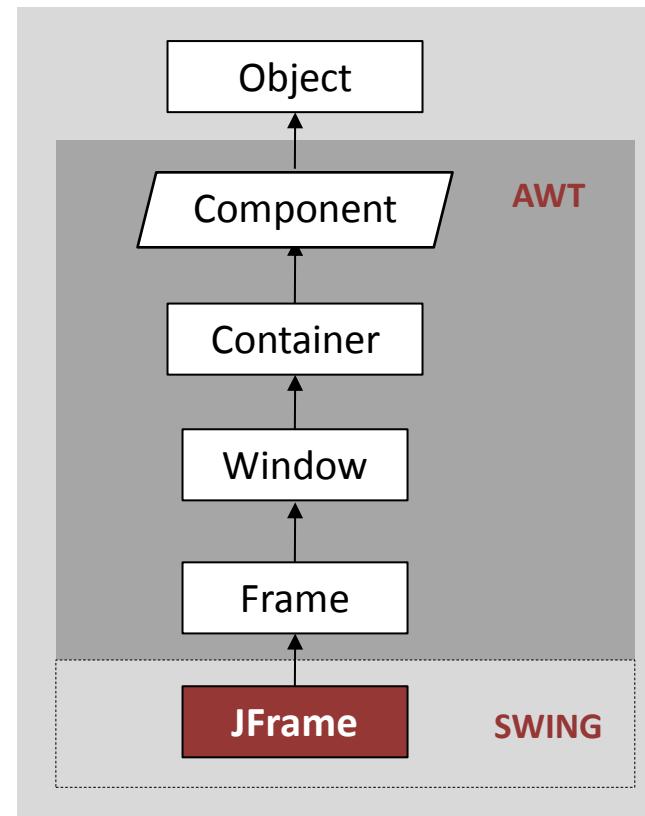


## ■ Package

- javax.swing

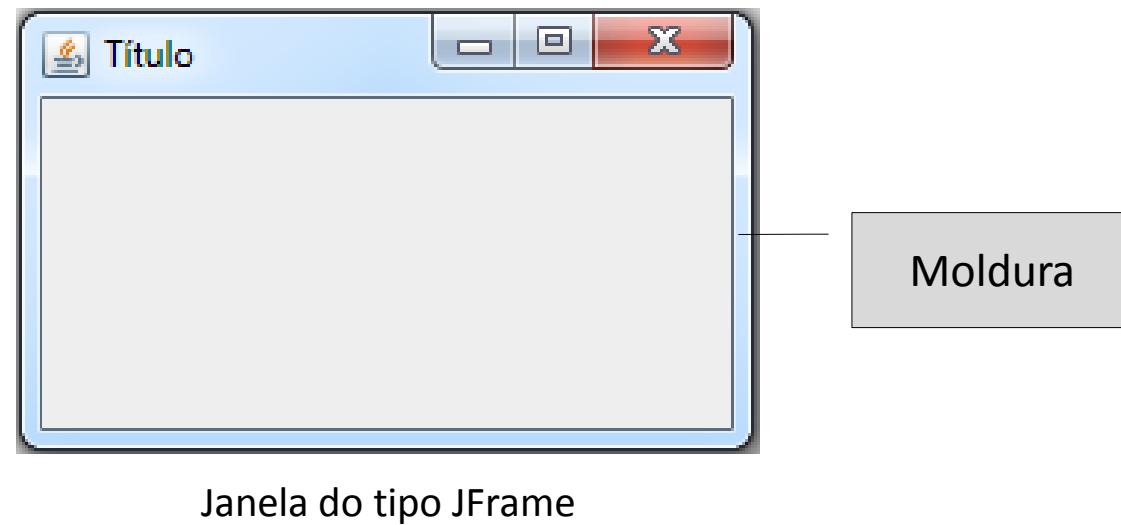
## ■ Declaração

```
public class JFrame extends Frame  
    implements WindowConstants, Accessible, RootPaneContainer { ... }
```

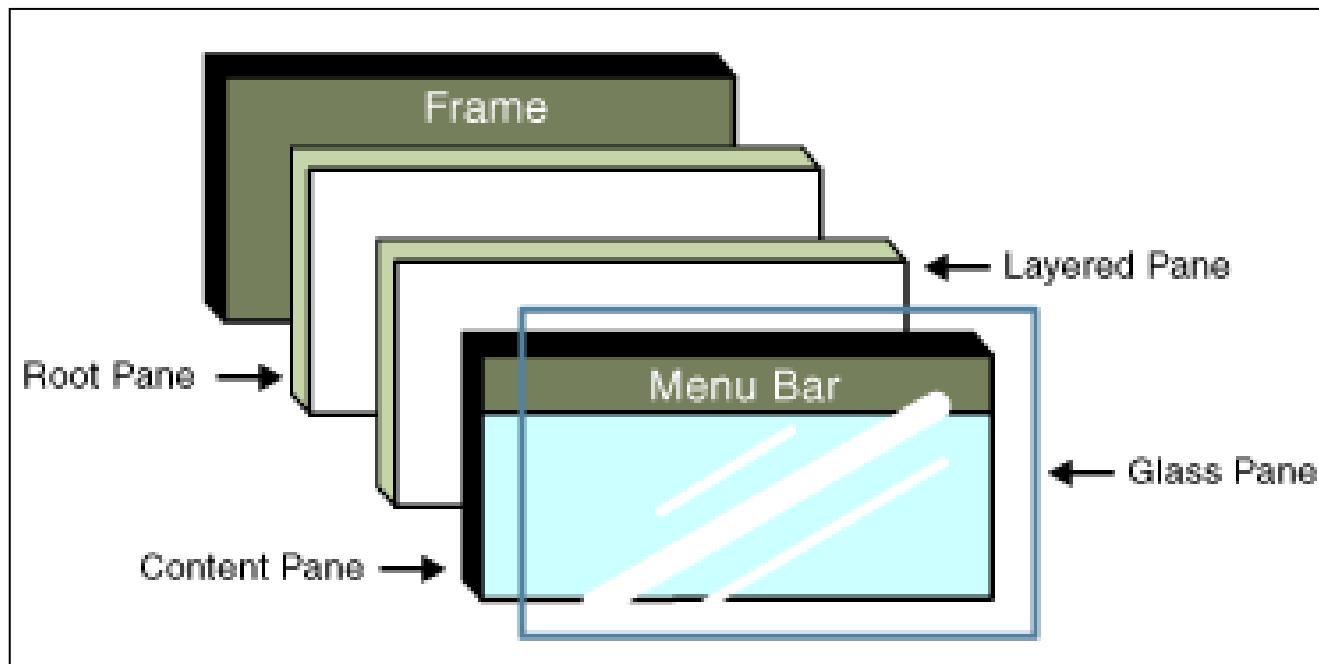


- Janela JFrame é constituída por

- Moldura
- Painéis // contentores de componentes GUI

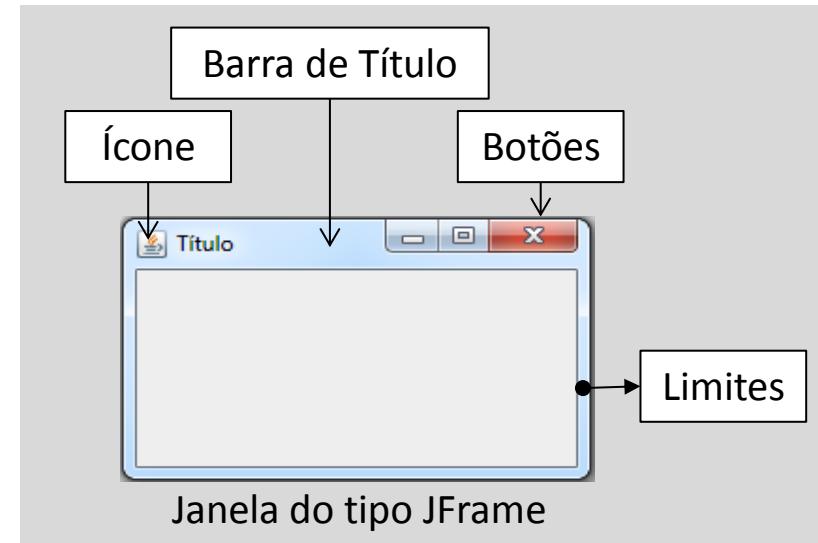


## ■ Janela JFrame contém

■ Root Pane

- Formada pelos Componentes (Decorações)

- Barra de Título
- Botões
  - Minimizar
  - Maximizar
  - Fechar
- Ícone
- Limites

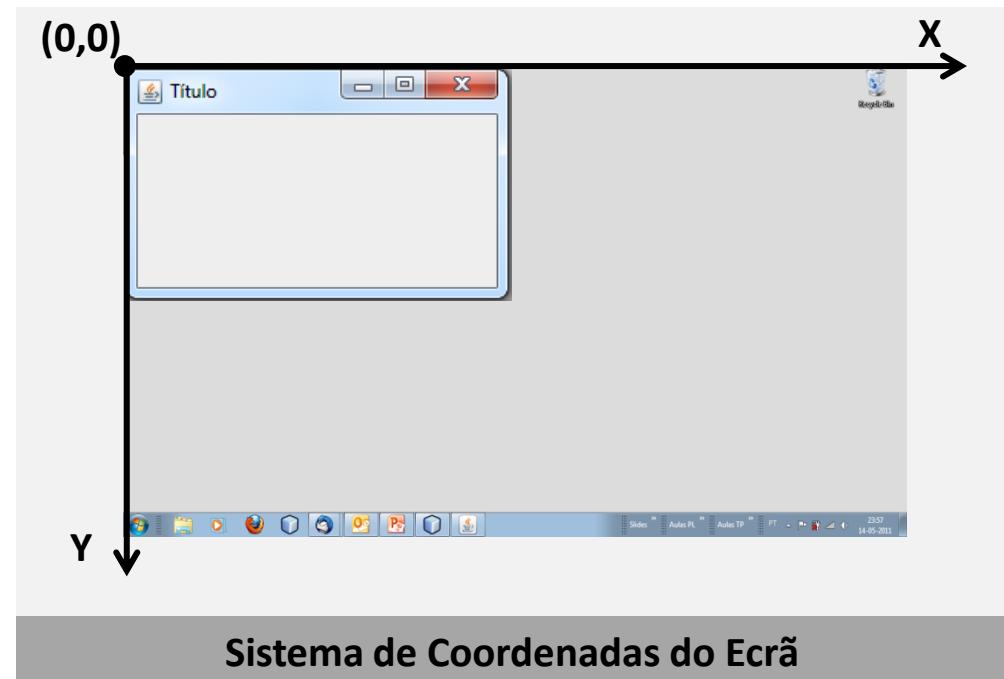


- Desenhada

- Pelo sistema de janelas do sistema operativo
  - Swing
    - Desenha tudo ... **exceto moldura**

## ■ Criam Janelas

- Invisíveis
- Tamanho 0
- Colocadas no canto superior esquerdo do ecrã
  - ponto (0,0) do ecrã



## ■ Declarações

Declaração	Funcionalidade
<code>public JFrame() throws HeadlessException</code>	Cria janela sem título.
<code>public JFrame(String title) throws HeadlessException</code>	Cria janela com título <i>title</i> .

## ▪ Herdados

- [Component](#)
- [Container](#)
- [Window](#)
- [Frame](#)

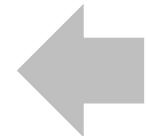
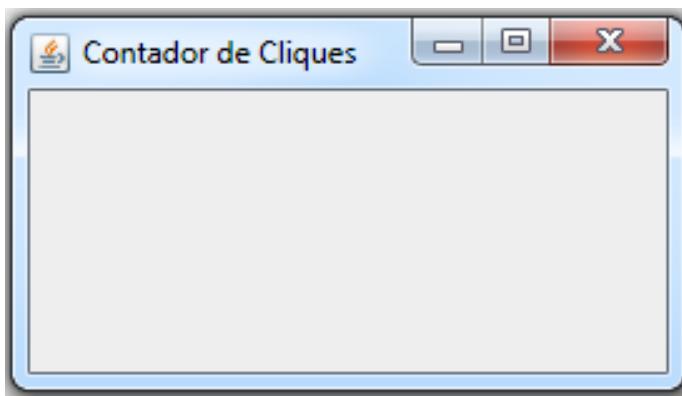
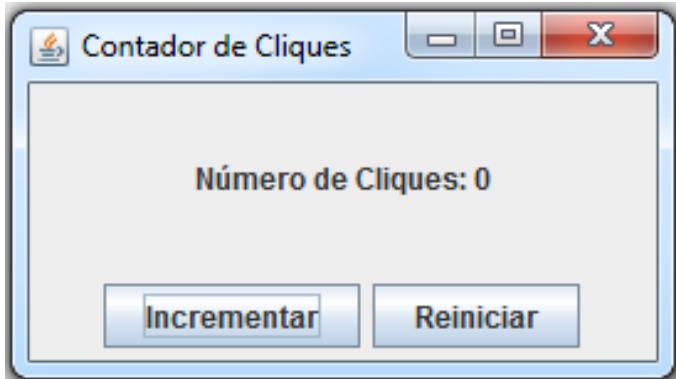
## ▪ Próprios (1/2)

- Mais usados

Declaração	Funcionalidade
public void setResizable( boolean resizable )	setResizable( <b>false</b> ) impede utilizador de redimensionar janela.
public void setTitle( String title )	Modifica título da janela.
public void setJMenuBar(JMenuBar menubar)	Especifica a barra de menus da janela.
public JMenuBar getJMenuBar()	Retorna barra de menus.
public Container getContentPane()	Retorna Content Pane.
public JRootPane getRootPane()	Retorna Root Pane.
public Component getGlassPane()	Retorna Glass Pane.
public JLayeredPane getLayeredPane()	Retorna Layered Pane.

## ■ Próprios (2/2)

Declaração	Funcionalidade								
<pre>public void setDefaultCloseOperation(int operation)</pre> <ul style="list-style-type: none"><li>▪ Exemplo: <code>setDefaultCloseOperation(EXIT_ON_CLOSE)</code></li></ul>	<p>Define operação que ocorrerá, por omissão, quando utilizador inicia fecho da janela no botão close da moldura;</p> <p>Operações à escolha</p> <table><tbody><tr><td><code>DO NOTHING ON CLOSE</code></td><td>// Não faz nada</td></tr><tr><td><code>HIDE ON CLOSE</code></td><td>// Esconde janela</td></tr><tr><td><code>DISPOSE ON CLOSE</code></td><td>// Fecha janela</td></tr><tr><td><code>EXIT ON CLOSE</code></td><td>// Termina aplicação</td></tr></tbody></table> <p>Por omissão, o botão close da janela executa a operação <code>HIDE ON CLOSE</code>.</p> <p>Notas</p> <ul style="list-style-type: none"><li>▪ Janela e lógica do programa são executados em <i>threads</i> diferentes;</li><li>▪ Aplicação gráfica pode ter o programa em execução sem GUI visível.</li></ul>	<code>DO NOTHING ON CLOSE</code>	// Não faz nada	<code>HIDE ON CLOSE</code>	// Esconde janela	<code>DISPOSE ON CLOSE</code>	// Fecha janela	<code>EXIT ON CLOSE</code>	// Termina aplicação
<code>DO NOTHING ON CLOSE</code>	// Não faz nada								
<code>HIDE ON CLOSE</code>	// Esconde janela								
<code>DISPOSE ON CLOSE</code>	// Fecha janela								
<code>EXIT ON CLOSE</code>	// Termina aplicação								

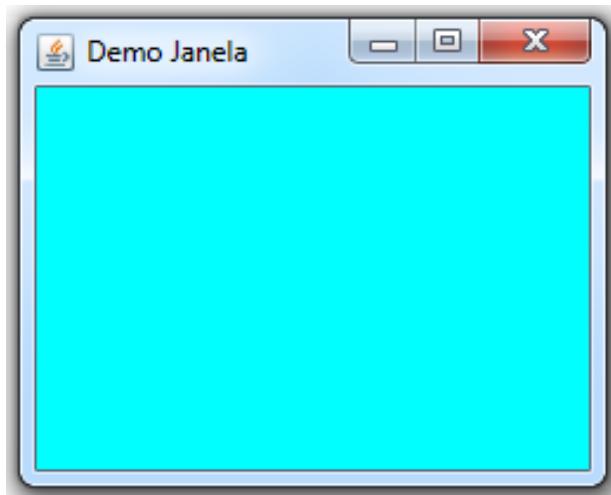


```
public class ContadorGUI extends JFrame {  
  
    private static final int JANELA_LARGURA = 270;  
    private static final int JANELA_ALTURA = 150;  
  
    public ContadorGUI() {  
  
        super("Contador de Cliques");  
  
        setDefaultCloseOperation(EXIT_ON_CLOSE);  
        setSize(JANELA_LARGURA, JANELA_ALTURA);  
        setLocationRelativeTo(null);  
        setVisible(true);  
    }  
}
```

```
public class Main {  
  
    public static void main(String[] args) {  
        new ContadorGUI();  
    }  
}
```

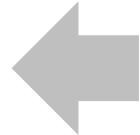
- Cor de fundo da Janela

- É do Content Pane



```
public class DemoJanela extends JFrame {  
  
    private static final int JANELA_LARGURA = 300;  
    private static final int JANELA_ALTURA = 200;  
  
    public DemoJanela() {  
  
        super("Demo Janela");  
  
        getContentPane().setBackground( Color.CYAN );  
  
        setDefaultCloseOperation(EXIT_ON_CLOSE);  
        setSize(JANELA_LARGURA, JANELA_ALTURA);  
        setLocationRelativeTo(null);  
        setVisible(true);  
    }  
}  
  
public class Main {  
  
    public static void main(String[] args) {  
        new DemoJanela();  
    }  
}
```

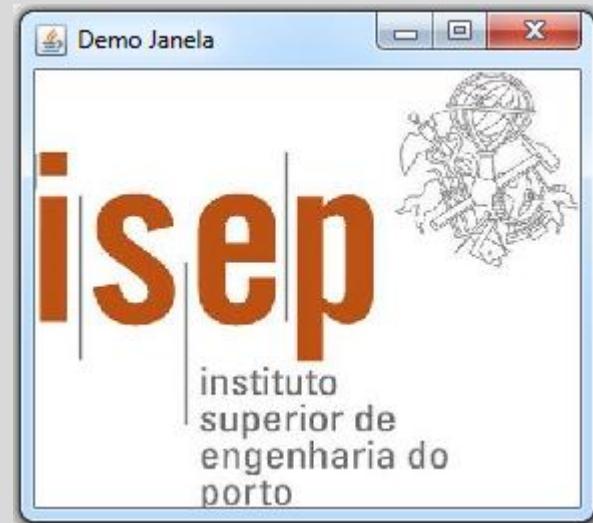
- Janela sem Moldura



```
public class DemoJanelaSemMoldura extends JFrame {  
  
    private static final int JANELA_LARGURA = 270;  
    private static final int JANELA_ALTURA = 150;  
  
    public DemoJanelaSemMoldura () {  
  
        super("Demo Janela Sem Moldura");  
  
        setUndecorated(true);  
  
        setSize(JANELA_LARGURA , JANELA_ALTURA);  
        setLocationRelativeTo(null);  
        setVisible(true);  
    }  
}
```

```
public class Main {  
  
    public static void main(String[] args) {  
  
        new DemoJanelaSemMoldura ();  
    }  
}
```

```
// Exemplo: Imagem de Fundo de uma Janela
public class DemoJanela extends JFrame {
    private static final int JANELA_LARGURA = 300;
    private static final int JANELA_ALTURA = 200;
    public DemoJanela() {
        super("Demo Janela");
        PainelFundo pf = new PainelFundo();
        add(pf);
        setDefaultCloseOperation( EXIT_ON_CLOSE );
        setSize(JANELA_LARGURA, JANELA_ALTURA);
        setResizable(false);
        setVisible(true);
    }
    private class PainelFundo extends JPanel {
        public void paintComponent(Graphics g) { // método reescrito; desenha componentes do painel
            super.paintComponent(g);
            Dimension dimensaoPainel = getSize(); // para redimensionar imagem à medida do painel
            double largura = dimensaoPainel.getWidth();
            double altura = dimensaoPainel.getHeight();
            ImageIcon i1 = new ImageIcon("isep_logo.jpg"); // imagem guardada na pasta do aplicação
            Image i2 = i1.getImage().getScaledInstance( (int) largura, (int) altura, Image.SCALE_SMOOTH );
            Image i3 = new ImageIcon( i2 ).getImage();
            g.drawImage(i3, 0, 0, this);
        }
    }
}
```



Parâmetro especifica algoritmo para redimensionar a imagem.  
Há várias opções, tais como: SCALE\_FAST e SCALE\_AREA\_AVERAGING  
▪ Escolha entre rapidez e qualidade da imagem

## ■ Interesse

- Definir dimensões e posição da janela

```
import java.awt.Toolkit;  
...  
Toolkit tk = Toolkit.getDefaultToolkit();          // getDefaultToolkit é método estático  
Dimension screenSize = tk.getScreenSize();  
int screenWidth = screenSize.width;  
int screenHeight = screenSize.height;
```

Tipos de Evento que Janela pode Gerar (1/2)	Evento
Container	Componente adicionado à janela
	Componente removido da janela
Component	Janela escondida
	Janela mostrado
	Janela movido
	Janela redimensionado
Focus	Janela adquire o foco
	janela perde o foco
Key	Tecla mantida premida
	Tecla libertada
	Tecla premida (toque)
Mouse	Clique no rato
	Rato entrou na janela
	Rato saiu da janela
	Botão do rato premido
	Botão do rato libertado
	Rato movido
	Rato arrastado (premido + movido)
	Roda do rato movida

Tipos de Evento que Janela pode Gerar (2/2)	Evento
	Janela abriu
	Janela fechou
	Janela ficou activa
Window	Janela ficou inactiva
	Janela ficou minimizada
	Janela ficou restaurada (tamanho original)
	Utilizador quer fechar Janela

## ■ Evento Window (1/2)

- Tratamento
  - Realizado
    - Objetos de classes que implementem o **interface** WindowListener
      - Métodos de Evento
        - **windowOpened( WindowEvent e )**
        - **windowClosed( WindowEvent e )**
        - **windowActivated( WindowEvent e )**
        - **windowDeactivated( WindowEvent e )**
        - **windowIconified( WindowEvent e )**
        - **windowDeiconified( WindowEvent e )**
        - **windowClosing( WindowEvent e )**
    - Alguns eventos
      - Classe WindowAdapter
        - implementa interface WindowListener
      - Exemplo

```
private class TrataEvento extends WindowAdapter {  
    public void windowClosing( WindowEvent e) { ... } // método reescrito  
}
```

## ■ Evento Window (2/2)

- Registo de objetos TrataEvento // objetos designados *event listeners*

Declaração	Funcionalidade
<pre>public void addWindowListener( WindowListener l )  ▪ Exemplo:  public class Janela extends JFrame {      public Janela() {          super("Demo");          ...         addWindowListener( new TrataEvento() );      }      private class TrataEvento extends WindowAdapter {          @Override         public void windowClosing( WindowEvent e) {              ...         }     } }</pre>	Regista objeto para tratar evento do tipo Window.

- <http://docs.oracle.com/javase/tutorial/uiswing/components/frame.html>

- [Contentores de Componentes Gráficos](#)

- [Painéis](#)

- [Noção de Painel](#)
- [Classe JPanel](#)
- [Classe JLayeredPane](#)
- [Classe JRootPane](#)
- [Classe JScrollPane](#)

- [Superclasse Window](#)

- [Janelas](#)

- [Superclasse Frame](#)
- [Classe JFrame](#)

- [Caixas de Diálogo](#)

- [Introdução](#)
- [Classe JOptionPane](#)
- [Classe JFileChooser](#)
- [Próprias](#)
  - [Superclasse Dialog](#)
  - [Classe JDialog](#)



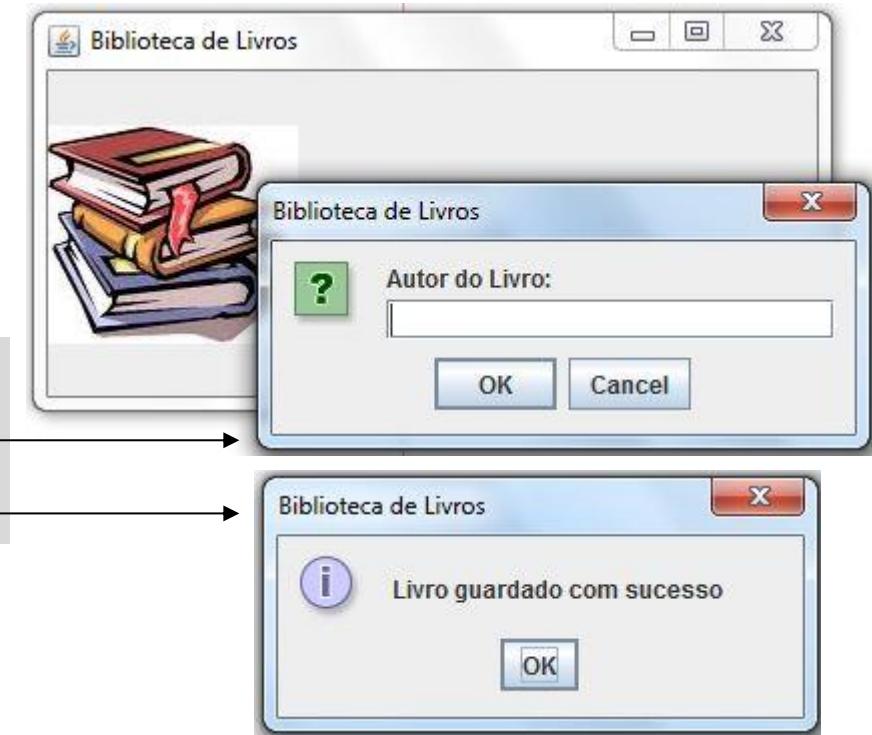
## Noção de Caixa de Diálogo

- Componente GUI
  - Separado da janela da aplicação // não contido na janela
- Interesse
  - Ler Dados
  - Mostrar Informação

Utilizador

### Exemplos de Caixas de Diálogo

- Entrada
- Saída



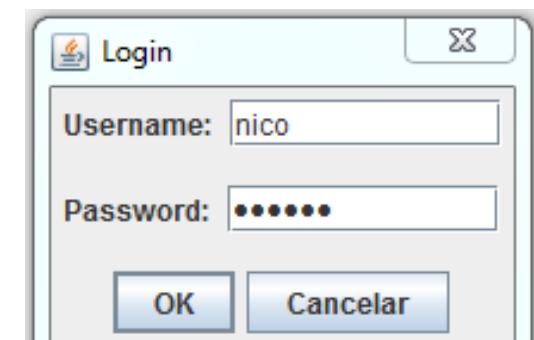
- Dependente de Janelas
  - Destruição de janela ⇒ destruição das caixas de diálogo dependentes
  - Minimização de janela ⇒ esconde caixas de diálogo dependentes
  - Abertura de janela minimizada ⇒ mostra caixas de diálogo dependentes

## ■ Tipos de Funcionamento

- Modal
- Modeless

## ■ Caixa Modal

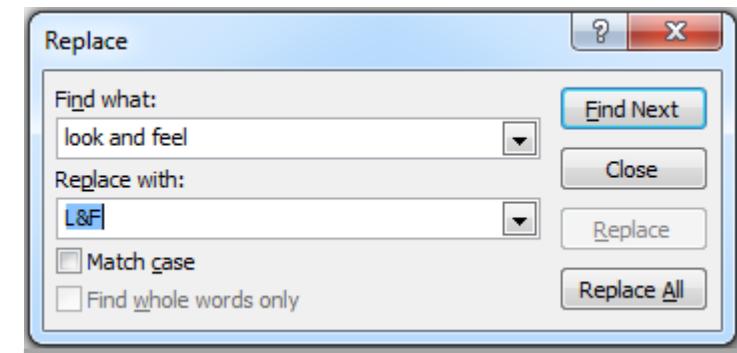
- Bloqueia acesso do utilizador à janela pai (criadora da caixa)
- Interesse
  - Obrigar utilizador a interagir com caixa de diálogo
  - Indispensável para execução da aplicação



Caixa de Diálogo Modal

## ■ Caixa Modeless

- Não bloqueia acesso do utilizador à janela pai
- Interesse
  - Permitir ao utilizador a introdução de dados na janela pai



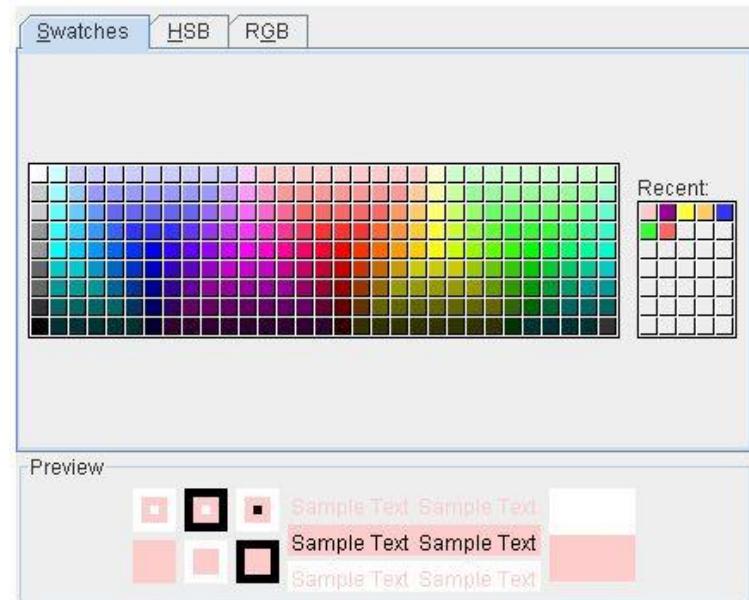
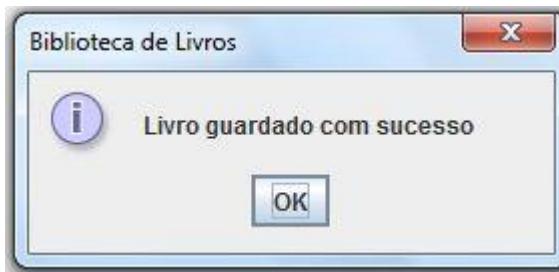
Caixa de Diálogo Modeless

## ■ Tipos de Caixas de Diálogo

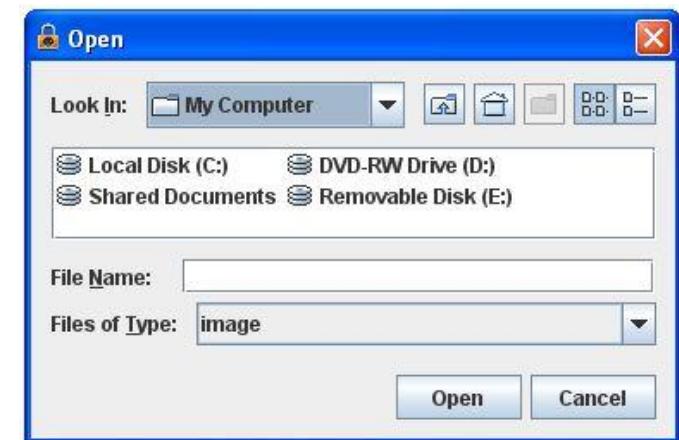
- Nativas
- Próprias

## ■ Nativas

- JColorChooser
- JFileChooser
- JOptionPane
  - Tipo Modal
  - Permite diálogos simples
- Ex:



JColorChooser



JFileChooser

## ■ Próprias

- Para Diálogos
  - Personalizados
  - Mais complexos
- Permite Tipo Modeless
- Classes derivadas da classe **JDialog**

- Bibliografia

- <http://download.oracle.com/javase/tutorial/uiswing/components/dialog.html>

- [Contentores de Componentes Gráficos](#)

- [Painéis](#)

- [Noção de Painel](#)
- [Classe JPanel](#)
- [Classe JLayeredPane](#)
- [Classe JRootPane](#)
- [Classe JScrollPane](#)

- [Superclasse Window](#)

- [Janelas](#)

- [Superclasse Frame](#)
- [Classe JFrame](#)

- [Caixas de Diálogo](#)

- [Introdução](#)
- [Classe JOptionPane](#)
- [Classe JFileChooser](#)
- [Próprias](#)
  - [Superclasse Dialog](#)
  - [Classe JDialog](#)

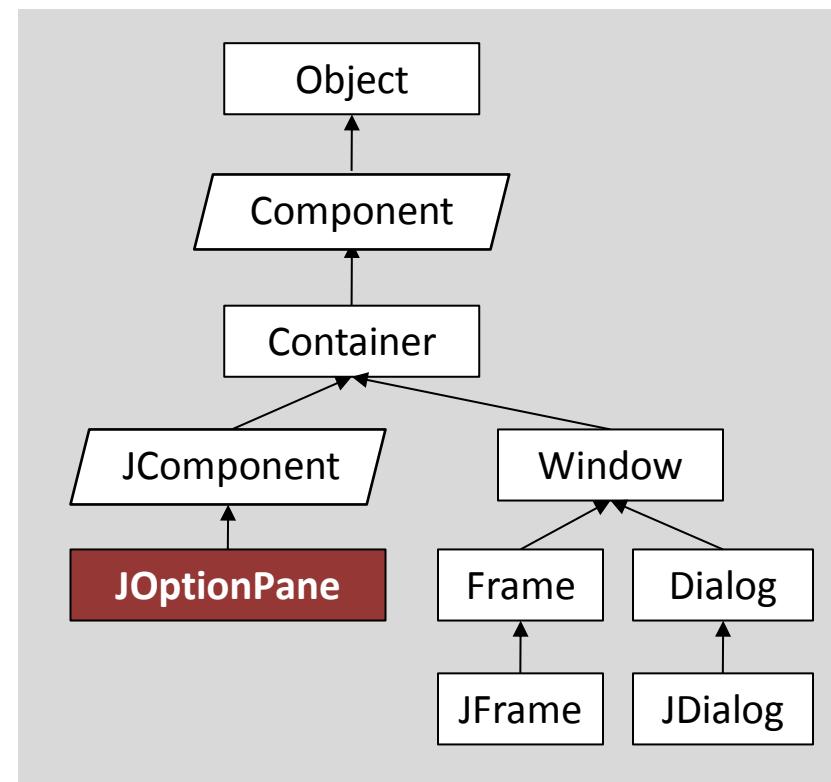


## ■ Package

- javax.swing

## ■ Declaração

```
public class JOptionPane extends JComponent  
    implements Accessible { ... }
```



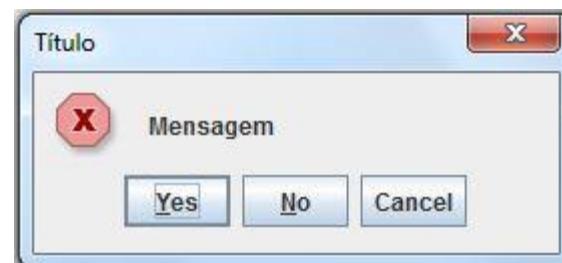
## ■ Criar Caixas de Diálogo

- Para diálogos com utilizador
  - Simples
- Permite
  - Ler
  - Mostrar
- Funcionamento
  - Modal

Caixa de Mensagem  
(Mostrar mensagem)



Caixa de Confirmação  
(Utilizador confirmar ação)

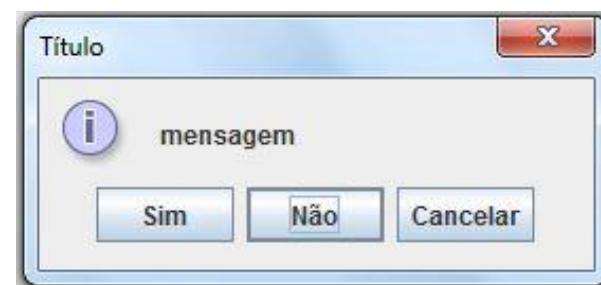


Caixa de Entrada  
(Ler dados do utilizador)



Caixa de Opção  
(Utilizador escolher opção)

- Opções personalizadas
- Podem ser objetos



## ■ Componentes das Caixas

### ▪ Comuns

- Título
- Ícone
  - Identifica tipo mensagem
- Mensagem
- Pode ser contentor JPanel
- Um ou mais botões de opção
  - OK
  - Yes, Cancel, No
  - Sim, Cancelar, Não

### ▪ Específicos

- TextField // Entrada Texto
- ComboBox // Escolha Objetos

- Usando Métodos de Classe (static)

- JOptionPane.showMessageDialog(...)



Caixa de Mensagem

- JOptionPane.showConfirmDialog(...)



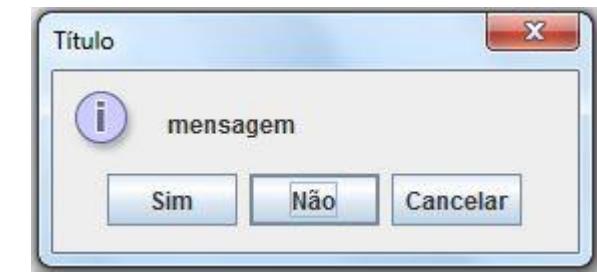
Caixa de Confirmação

- JOptionPane.showInputDialog(...)



Caixa de Entrada

- JOptionPane.showOptionDialog(...)



Caixa de Opção

- Constantes para especificar parâmetros comuns

- Tipos de mensagem

JOptionPane.ERROR\_MESSAGE



JOptionPane.INFORMATION\_MESSAGE



JOptionPane.WARNING\_MESSAGE



JOptionPane.QUESTION\_MESSAGE



JOptionPane.PLAIN\_MESSAGE

sem ícone



Caixa de Mensagem

- Tipos de Opções

JOptionPane.DEFAULT\_OPTION



JOptionPane.OK\_CANCEL\_OPTION

JOptionPane.YES\_NO\_OPTION

JOptionPane.YES\_NO\_CANCEL\_OPTION

Caixa de Confirmação

- Opções para testar retorno (valores inteiros)

JOptionPane.OK\_OPTION



JOptionPane.CANCEL\_OPTION

JOptionPane.YES\_OPTION

JOptionPane.NO\_OPTION

Caixa de Entrada

## ■ Funcionalidade

- Mostra mensagem e espera pelo OK
  - Não retorna nada
  - Mensagem pode ser um componente GUI

## ■ Declarações de Métodos

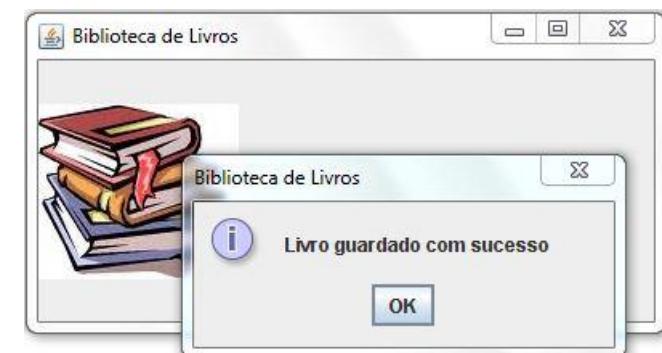
```
public static void showMessageDialog( Component parent,
                                     Object message )
```

```
public static void showMessageDialog( Component parent,
                                     Object message,
                                     String title,
                                     int messageType)
```

```
public static void showMessageDialog( Component parent,
                                     Object message,
                                     String title,
                                     int messageType,
                                     Icon icon )
```



Definição do parâmetro parent garante caixa de diálogo sobre (frente) esse componente



## ■ Exemplo

```
JFrame janela = new JFrame("Biblioteca de Livros");
...
JOptionPane.showMessageDialog( janela, "Livro guardado com sucesso", "Guardar Livro",
                            JOptionPane.INFORMATION_MESSAGE);
```

- Funcionalidade

- Mostra mensagem e retorna confirmação
    - Confirmação = inteiro

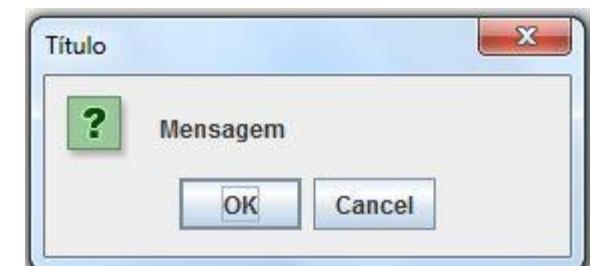
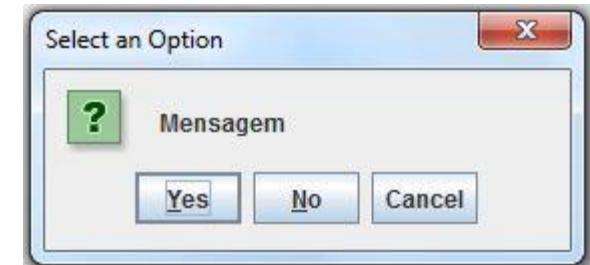
- Declarações Métodos

```
public static int showConfirmDialog( Component parent,  
                                     Object message )
```

```
public static int showConfirmDialog( Component parent,  
                                     Object message,  
                                     String title,  
                                     int optionType)
```

```
public static int showConfirmDialog( Component parent,  
                                     Object message,  
                                     String title,  
                                     int optionType,  
                                     int messageType)
```

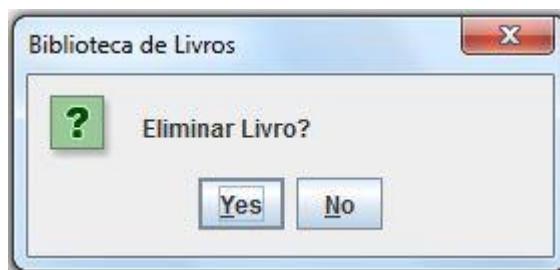
```
public static int showConfirmDialog( Component parent,  
                                     Object message,  
                                     String title,  
                                     int optionType,  
                                     int messageType,  
                                     Icon icon )
```



- Opções para testar o retorno

- JOptionPane.OK\_OPTION
- JOptionPane.CANCEL\_OPTION
- JOptionPane.YES\_OPTION
- JOptionPane.NO\_OPTION

- Exemplo



```
JFrame janela = new JFrame();
...
int resposta = JOptionPane.showConfirmDialog( janela, "Eliminar Livro?", "Biblioteca de Livros",
                                              JOptionPane.YES_NO_OPTION,
                                              JOptionPane.QUESTION_MESSAGE) ;
If( resposta == OptionPane.YES_OPTION )
    livro.eliminar();
```

## ■ Funcionalidade

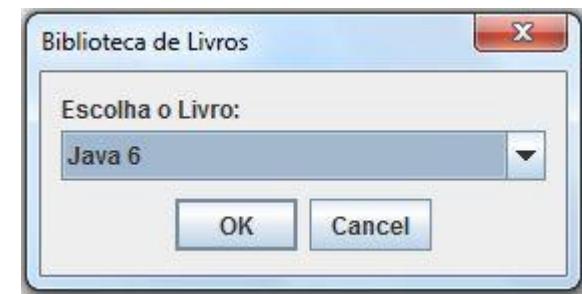
- Mostra **mensagem** e um dos seguintes componentes
  - **TextField**
    - Retorna **texto** inserido pelo utilizador



Caixa de Entrada (**Texto**)

## ■ ComboBox

- Retorna objeto escolhido pelo utilizador
  - Exemplos
    - Livro, Carro, Trabalhador, etc.
  - Objetos são fornecidos num array
    - Tipo de array
      - Pode ser qualquer tipo **referência**
    - Exemplo



Caixa de Entrada (**Objeto**)

```
String[] livros = {"Java 6", "Big Java"};
```

```
Livro[] livros = { new Livro("Java 6"), new Livro("Big Java") }
```

## ▪ Declarações Métodos

```
public static String showInputDialog( Object message )
```

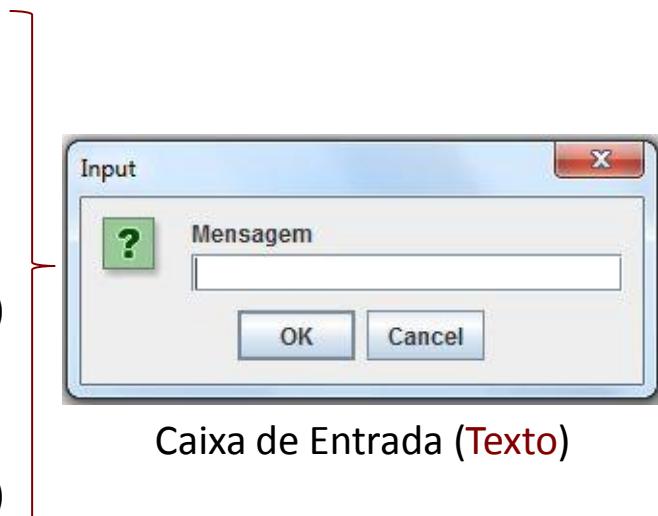
```
public static String showInputDialog( Component parent,  
                                     Object message )
```

```
public static String showInputDialog( Object message,  
                                     Object initialValue )
```

```
public static String showInputDialog( Component parent,  
                                     Object message,  
                                     Object initialValue )
```

```
public static String showInputDialog( Component parent,  
                                     Object message,  
                                     String title,  
                                     int messageType )
```

```
public static Object showInputDialog(Component parent,  
                                     Object message,  
                                     String title,  
                                     int messageType,  
                                     Icon icon,  
                                     Object[] selectionValues,  
                                     Object initialValue )
```



Caixa de Entrada (Texto)



Caixa de Entrada (Objeto)

## ■ Exemplos

```
JFrame janela = new JFrame();
```

```
...
```

```
String autor = JOptionPane.showInputDialog( janela,"Autor do Livro:", "Biblioteca de Livros",
JOptionPane.QUESTION_MESSAGE);
```

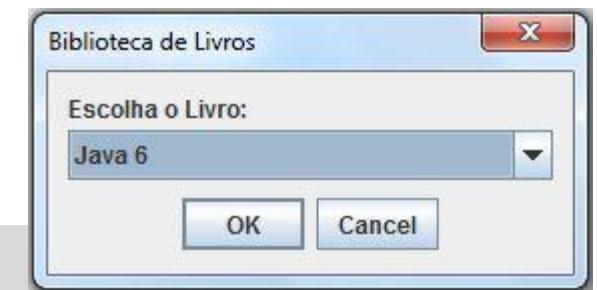


```
JFrame janela = new JFrame();
```

```
...
```

```
String[ ] opcoes = {"Java 6", "Big Java", "Core Java"};
```

```
String livro = JOptionPane.showInputDialog( janela, "Escolha o Livro:", "Biblioteca de Livros",
JOptionPane.PLAIN_MESSAGE, null,
opcoes, opcoes[0]);
```



- Funcionalidade

- Mostra uma mensagem e retorna a opção escolhida

- Botões de Opção

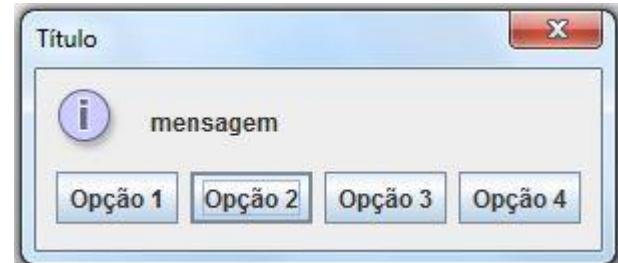
- Podem ser personalizados
  - Definidos através de array de qualquer tipo referência

- Retorna

- Índice de array da opção escolhida

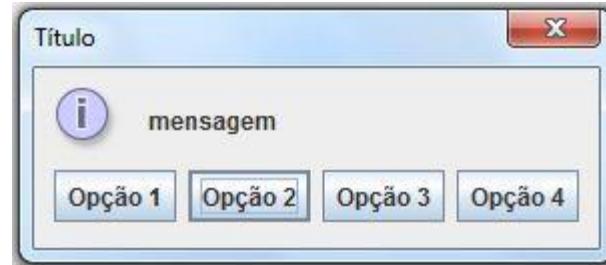
- Declaração de Método

```
public static int showOptionDialog( Component parent,  
                                  Object message,  
                                  String title,  
                                  int optionType,          // irrelevante para options ≠ null  
                                  int messageType,  
                                  Icon icon,              // personaliza ícone  
                                  Object[] options,  
                                  Object initialValue )   // botão que adquire o foco (omissão)
```



Caixa de Opção

## ■ Exemplo



Caixa de Opção

```
JFrame janela = new JFrame();
...
String[] opcoes = {"Opção 1", "Opção 2", "Opção 3", "Opção 4"};
int resposta = JOptionPane.showOptionDialog( janela, "mensagem", "Título", 0,
                                              JOptionPane.INFORMATION_MESSAGE,
                                              null,
                                              opcoes,
                                              opcoes[1] );
switch (resposta){
    case 0: ... ; break;
    case 1: ... ; break;
    case 2: ... ; break;
    case 3: ... ;
}
```

- <http://download.oracle.com/javase/tutorial/uiswing/components/layeredpane.html>

- [Contentores de Componentes Gráficos](#)

- [Painéis](#)

- [Noção de Painel](#)
- [Classe JPanel](#)
- [Classe JLayeredPane](#)
- [Classe JRootPane](#)
- [Classe JScrollPane](#)

- [Superclasse Window](#)

- [Janelas](#)

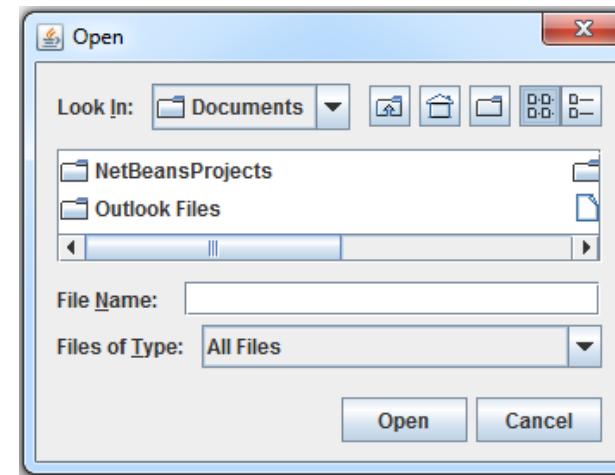
- [Superclasse Frame](#)
- [Classe JFrame](#)

- [Caixas de Diálogo](#)

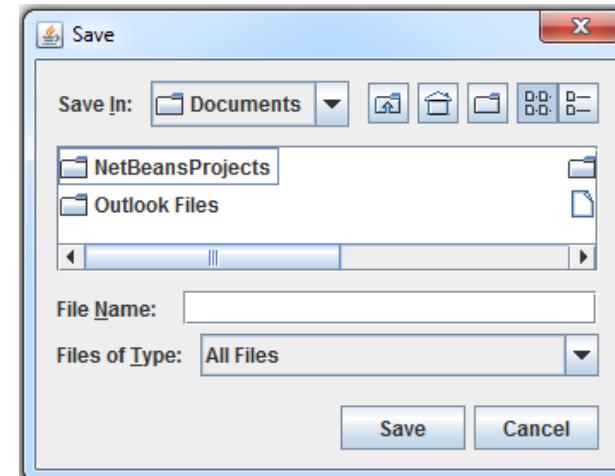
- [Introdução](#)
- [Classe JOptionPane](#)
- [Classe JFileChooser](#)
- [Próprias](#)
  - [Superclasse Dialog](#)
  - [Classe JDialog](#)



- Navegação no Sistema de Ficheiros
  - P. ex., para suportar operações de
    - Abertura de ficheiro/pasta



- Gravação de ficheiros



Caixa open permite escolher

- Pasta
- Ficheiro para abrir

Processamento do ficheiro ou da pasta **escolhida**

- Responsabilidade da aplicação

Caixa save permite escolher

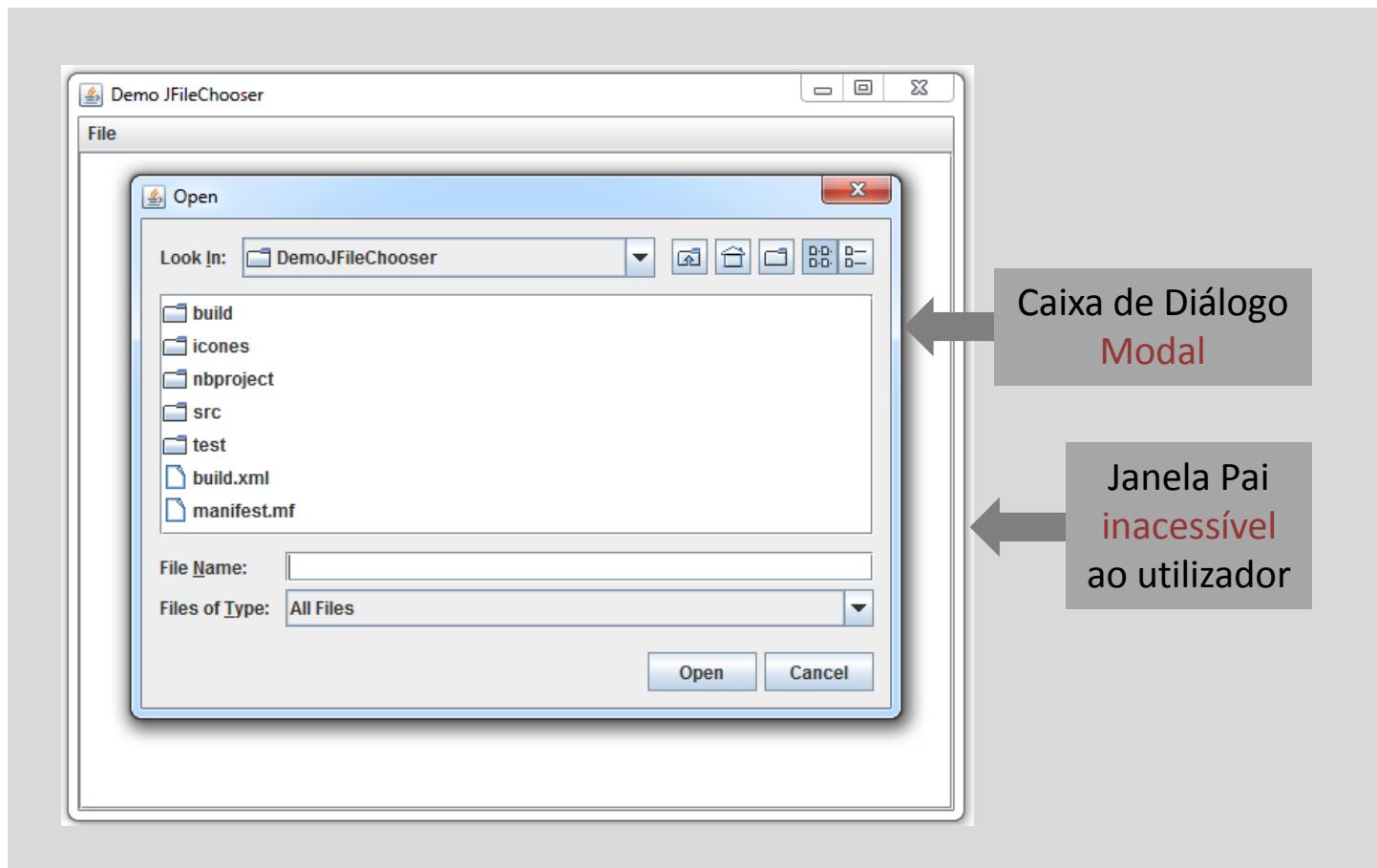
- Pasta para gravar ficheiros

Gravação de ficheiros

- Responsabilidade da aplicação

## ▪ Modal

- Bloqueia acesso do utilizador à janela pai // janela que criou caixa
- Obriga utilizador a interagir com caixa
  - Para regressar à janela pai da caixa



- Há 3 Tipos

- Abertura de Ficheiro/Pasta
- Gravação de Ficheiro
- Personalizada
  - // para outras tarefas (não abrir/gravar )
  - // Ex: anexar ficheiro num e-mail ; executar aplicação

- Caixas de Diálogo JFileChooser

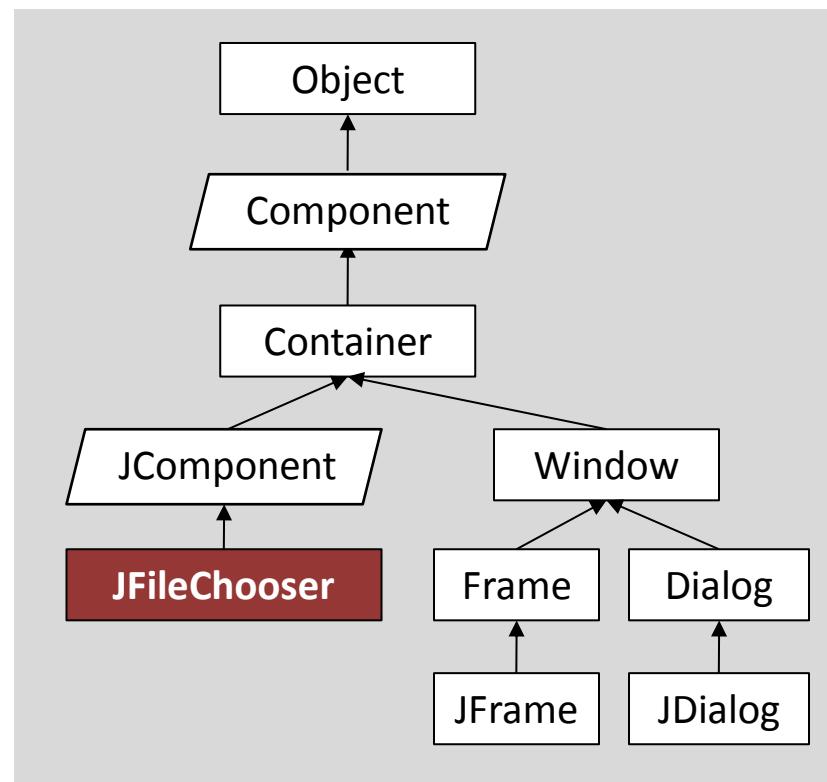
- Objetos da classe JFileChooser

## ■ Package

- javax.swing

## ■ Declaração

```
public class JFileChooser extends JComponent  
    implements Accessible { ... }
```



- Mais usados

Declaração	Funcionalidade
<pre>public JFileChooser()</pre> <ul style="list-style-type: none"> <li>▪ Exemplo:</li> </ul> <pre>JFileChooser fc = new JFileChooser();</pre>	<p>Constrói <b>objeto JFileChooser</b> usando a <b>pasta do utilizador</b>, por omissão;</p> <p>Esta pasta <b>depende</b> do sistema operativo:</p> <ul style="list-style-type: none"> <li>▪ Windows: "Meus Documentos"</li> <li>▪ Unix: "Home"</li> </ul>
<pre>public JFileChooser(String currentDirectoryPath)</pre> <ul style="list-style-type: none"> <li>▪ Exemplo:</li> </ul> <pre>JFileChooser fc = new JFileChooser("c:/MeuProjeto");</pre>	<p>Constrói <b>objeto JFileChooser</b> usando a <b>pasta especificada</b> através de uma <b>string</b>;</p> <p>Parâmetro <b>null</b> especifica pasta do utilizador por omissão.</p>
<pre>public JFileChooser(File currentDirectory)</pre> <ul style="list-style-type: none"> <li>▪ Exemplo:</li> </ul> <pre>File f = new File("c:/MeuProjeto"); JFileChooser fc = new JFileChooser( f );</pre>	<p>Constrói <b>objeto JFileChooser</b> usando a <b>pasta especificada</b> através de <b>objeto File</b>;</p> <p>Parâmetro <b>null</b> especifica pasta do utilizador por omissão.</p>

## ■ Herdados

- [Component](#)
- [Container](#)
- [JComponent](#)

## ■ Próprios (1/2)

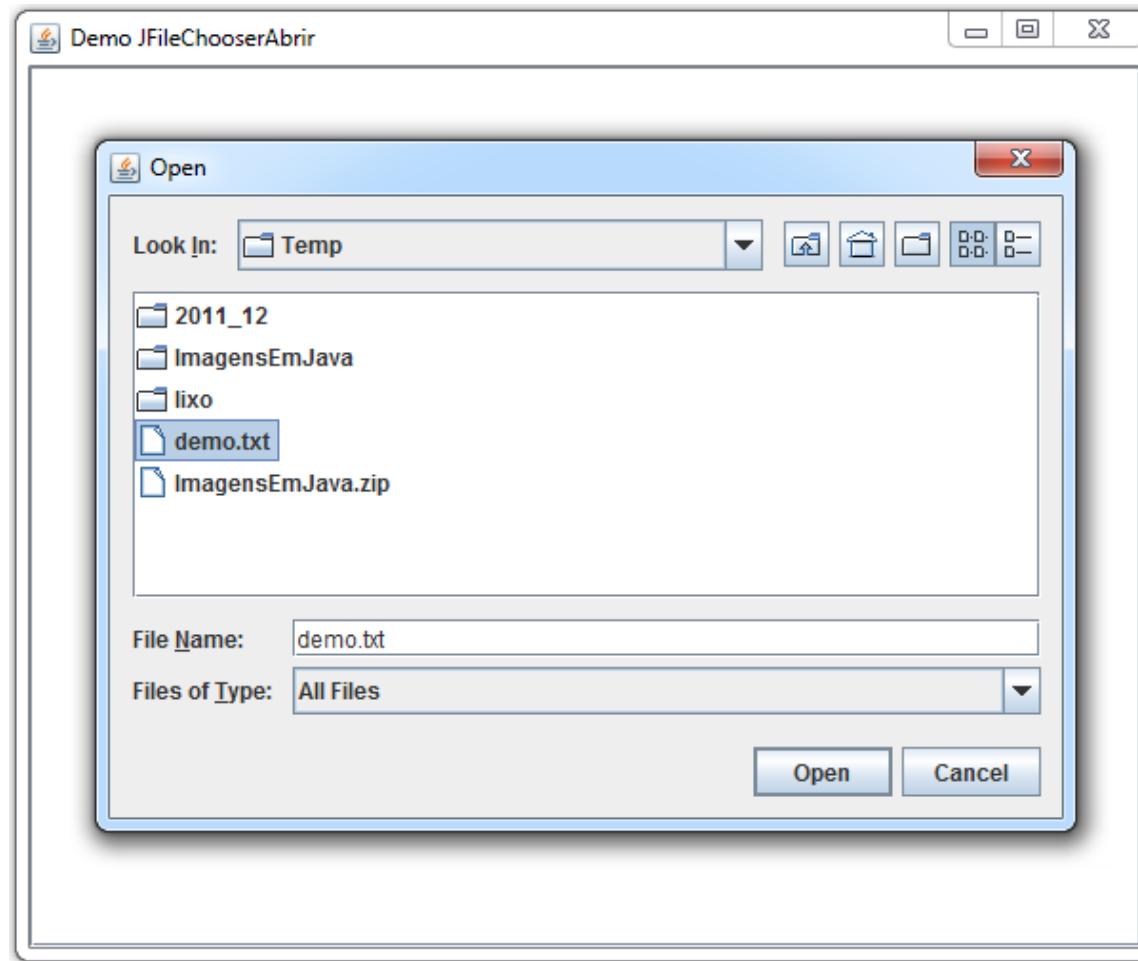
Declaração	Funcionalidade
<pre>public int showOpenDialog(Component parent)</pre> <ul style="list-style-type: none"><li>▪ Exemplo:<pre>JFileChooser fc = new JFileChooser(); int resp = fc.showOpenDialog(this); // this=janela pai if( resp == JFileChooser.APPROVE_OPTION) ....</pre></li></ul>	<p>Mostra caixa de diálogo JFileChooser para <b>abrir</b> um ficheiro;</p> <p>Tipos de retorno (tipo <b>int</b>):</p> <ul style="list-style-type: none"><li>▪ JFileChooser.CANCEL_OPTION</li><li>▪ JFileChooser.APPROVE_OPTION</li><li>▪ JFileChooser.ERROR_OPTION</li></ul>
<pre>public int showSaveDialog(Component parent)</pre> <ul style="list-style-type: none"><li>▪ Exemplo:<pre>JFileChooser fc = new JFileChooser(); int resp = fc.showSaveDialog(this); // this=janela pai if( resp == JFileChooser.APPROVE_OPTION) ....</pre></li></ul>	<p>Mostra caixa de diálogo JFileChooser para <b>gravar</b> ficheiro;</p> <p>Tipos de retorno (tipo <b>int</b>):</p> <ul style="list-style-type: none"><li>▪ JFileChooser.CANCEL_OPTION</li><li>▪ JFileChooser.APPROVE_OPTION</li><li>▪ JFileChooser.ERROR_OPTION</li></ul>

## ■ Próprios (2/2)

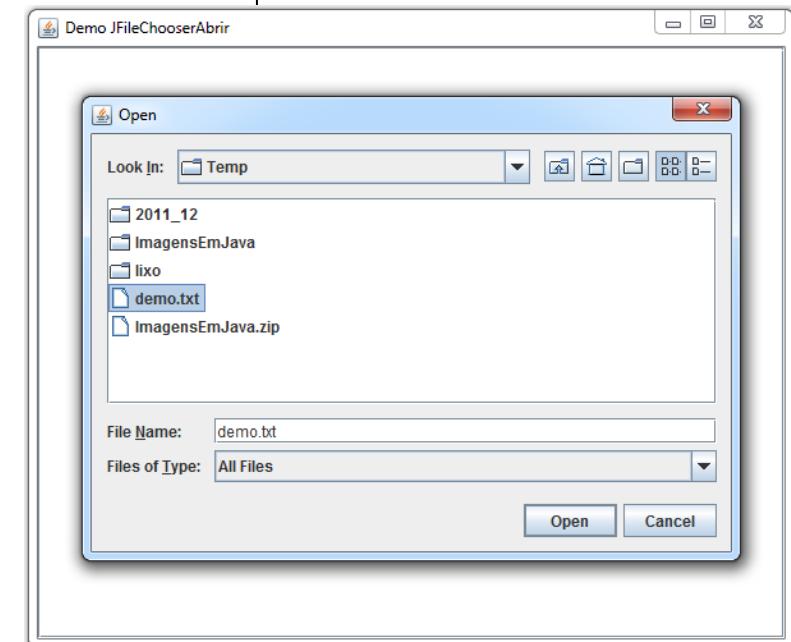
Declaração	Funcionalidade
<pre>public int showDialog(Component parent, String approveButtonText)</pre> <ul style="list-style-type: none"><li>▪ Exemplo:<pre>JFileChooser fc = new JFileChooser(); int resp = fc.showDialog(this, "Anexar"); if( resp == JFileChooser.APPROVE_OPTION) ....</pre></li></ul>	<p>Mostra caixa de diálogo JFileChooser com <b>título</b> e <b>botão approve</b> personalizados;</p> <p>Tipos de retorno (tipo <b>int</b>):</p> <ul style="list-style-type: none"><li>▪ JFileChooser.CANCEL_OPTION</li><li>▪ JFileChooser.APPROVE_OPTION</li><li>▪ JFileChooser.ERROR_OPTION</li></ul>
<pre>public File getSelectedFile()</pre> <ul style="list-style-type: none"><li>▪ Exemplo:<pre>JFileChooser fc = new JFileChooser(); int resp = fc.showOpenDialog(this); // this=janela pai if( resp == JFileChooser.APPROVE_OPTION) File f = fc.getSelectedFile(); ...</pre></li></ul>	Retorna ficheiro selecionado num <b>objeto File</b> .

- Ficheiro de Texto

- Lido para área de texto // JTextArea



```
public class DemoJFileChooserAbrir extends JFrame {  
  
    private static JTextArea txtArea;  
    private static final int JANELA_LARGURA=600, JANELA_ALTURA=500;  
  
    public DemoJFileChooserAbrir() {  
        super("Demo JFileChooserAbrir");  
  
        txtArea = new JTextArea();  
        JScrollPane scrTxtArea = new JScrollPane(txtArea);  
        add(scrTxtArea);  
  
        setDefaultCloseOperation(EXIT_ON_CLOSE);  
        setSize(JANELA_LARGURA, JANELA_ALTURA);  
        setLocationRelativeTo(null);  
        setVisible(true);  
    }  
  
    public static void main(String[] args) {  
        DemoJFileChooserAbrir gui = new DemoJFileChooserAbrir();  
        try {  
            JFileChooser fc = new JFileChooser("d:/temp");  
            int resp = fc.showOpenDialog(gui);  
            if (resp == JFileChooser.APPROVE_OPTION) {  
                File file = fc.getSelectedFile();  
                txtArea.read(new FileReader(file), null);  
            } catch (IOException ex) {  
                JOptionPane.showMessageDialog(gui, "Ficheiro não lido!!",  
                    "Ficheiro", JOptionPane.ERROR_MESSAGE);  
            }  
        }  
    }  
}
```



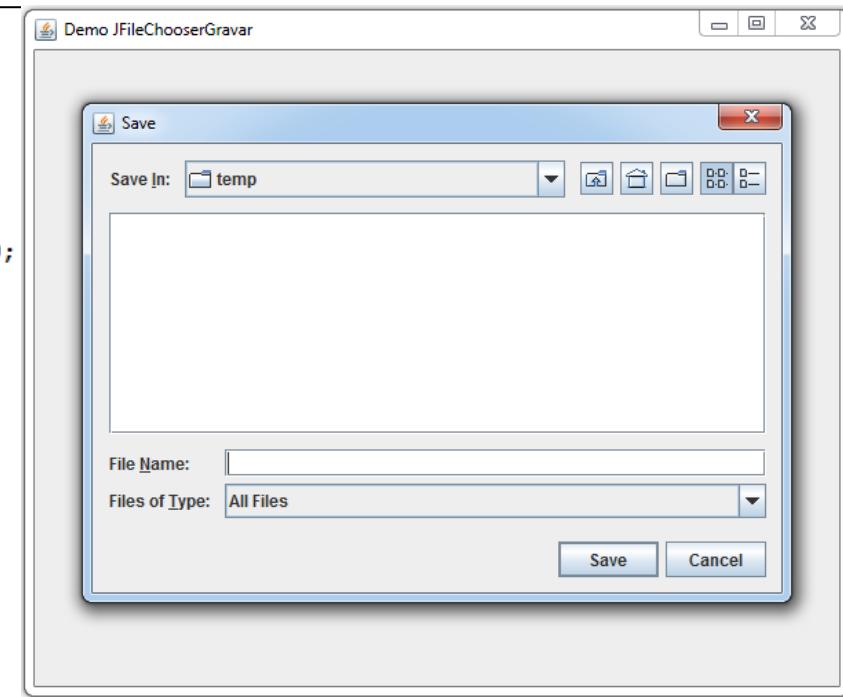
## Exemplo: Gravação de Ficheiro

```
import java.io.*;
import java.util.*;
import javax.swing.*;

public class DemoJFileChooserGravar extends JFrame {

    private static final int JANELA_LARGURA=600, JANELA_ALTURA=500;

    public DemoJFileChooserGravar() {
        super("Demo JFileChooserGravar");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(JANELA_LARGURA, JANELA_ALTURA);
        setLocationRelativeTo(null);
        setVisible(true);
    }
    public static void main(String[] args) {
        DemoJFileChooserGravar gui = new DemoJFileChooserGravar();
        try {
            JFileChooser fc = new JFileChooser("d:/temp");
            int resp = fc.showSaveDialog(gui);
            if (resp == JFileChooser.APPROVE_OPTION) {
                File f = fc.getSelectedFile();
                FileOutputStream fos = new FileOutputStream(f);
                ObjectOutputStream out = new ObjectOutputStream(fos);
                out.writeObject(getLista());
                out.close();
            }
        } catch (IOException ex) {
            JOptionPane.showMessageDialog( gui, "Ficheiro não Gravado!!",
                                         "Ficheiro", JOptionPane.ERROR_MESSAGE);
        }
    }
    private static ArrayList<String> getLista() {...3 lines...}
}
```



- Exemplos:

```
JFileChooser fc = new JFileChooser();

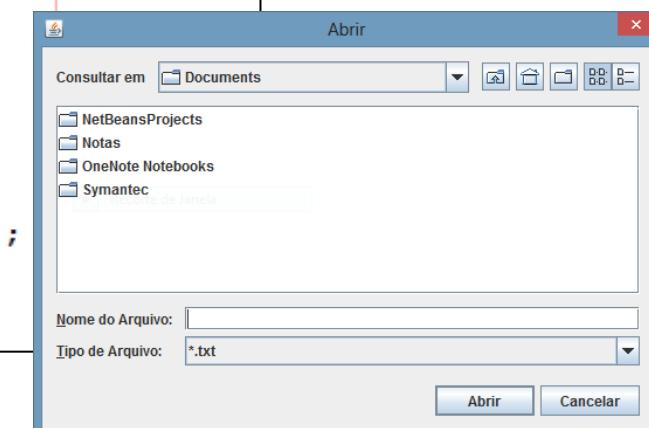
UIManager.put("FileChooser.openDialogTitleText", "Abrir");
UIManager.put("FileChooser.lookInLabelText", "Consultar em:");
UIManager.put("FileChooser.openButtonText", "Abrir");
UIManager.put("FileChooser.openButtonToolTipText", "Abrir o Ficheiro Selecionado");

UIManager.put("FileChooser.cancelButtonText", "Cancelar");
UIManager.put("FileChooser.cancelButtonToolTipText", "Cancelar");
UIManager.put("FileChooser.fileNameLabelText", "Nome do Ficheiro:");
UIManager.put("FileChooser.filesOfTypeLabelText", "Tipo de Ficheiro:");
UIManager.put("FileChooser.acceptAllFileFilterText", "Todos os Ficheiros");

UIManager.put("FileChooser.saveDialogTitleText", "Guardar");
UIManager.put("FileChooser.saveButtonText", "Guardar");
UIManager.put("FileChooser.saveInLabelText", "Guardar em:");

UIManager.put("FileChooser.deleteFileButtonText", "Eliminar");
UIManager.put("FileChooser.detailsViewButtonToolTipText", "Detalhes");

SwingUtilities.updateComponentTreeUI(fc);
```



- Lista de opções completa:

- [http://www.java2s.com/Tutorial/Java/0240\\_Swing/CustomizingaJFileChooserLookandFeel.htm](http://www.java2s.com/Tutorial/Java/0240_Swing/CustomizingaJFileChooserLookandFeel.htm)

- Exemplo – Filtrar Diretórios e Ficheiros \*.txt (usando objeto FileFilter)

```

private void definirFiltro(JFileChooser fc) {
    // objeto de subclasse de FileFilter - classe anónima
    fc.setFileFilter(new javax.swing.filechooser.FileFilter() {

        // método obrigatório - aceita todos os diretórios e ficheiros .txt
        public boolean accept(File f) {
            if (f.isDirectory()) { return true; }

            String extensao = extensao(f);
            if (extensao != null) { return extensao.equals("txt"); }
            return false;
        }

        // método obrigatório
        public String getDescription() { return "*.txt"; }

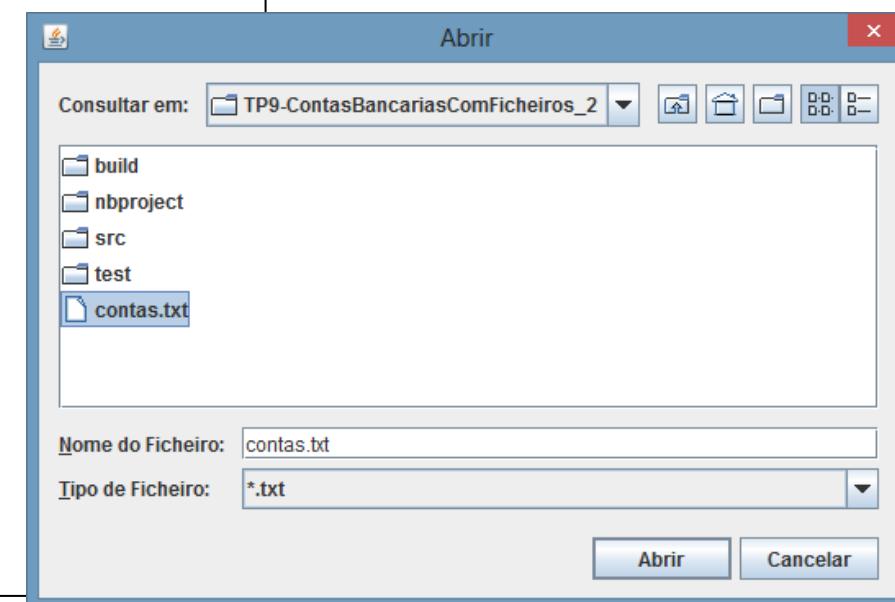
        // método auxiliar
        public String extensao(File f) {
            String ext = null;
            String s = f.getName();
            int i = s.lastIndexOf('.');

            if (i!=-1) {
                ext = s.substring(i + 1).toLowerCase();
            }
            return ext;
        }
    });
}

```

## Por omissão

- Um *file chooser* lista todos os diretórios e ficheiros não escondidos



- <http://download.oracle.com/javase/tutorial/uiswing/components/filechooser.html>

- [Contentores de Componentes Gráficos](#)

- [Painéis](#)

- [Noção de Painel](#)
- [Classe JPanel](#)
- [Classe JLayeredPane](#)
- [Classe JRootPane](#)
- [Classe JScrollPane](#)

- [Superclasse Window](#)

- [Janelas](#)

- [Superclasse Frame](#)
- [Classe JFrame](#)

- [Caixas de Diálogo](#)

- [Introdução](#)
- [Classe JOptionPane](#)
- [Classe JFileChooser](#)
- [Próprias](#)
  - [Superclasse Dialog](#)
  - [Classe JDialog](#)

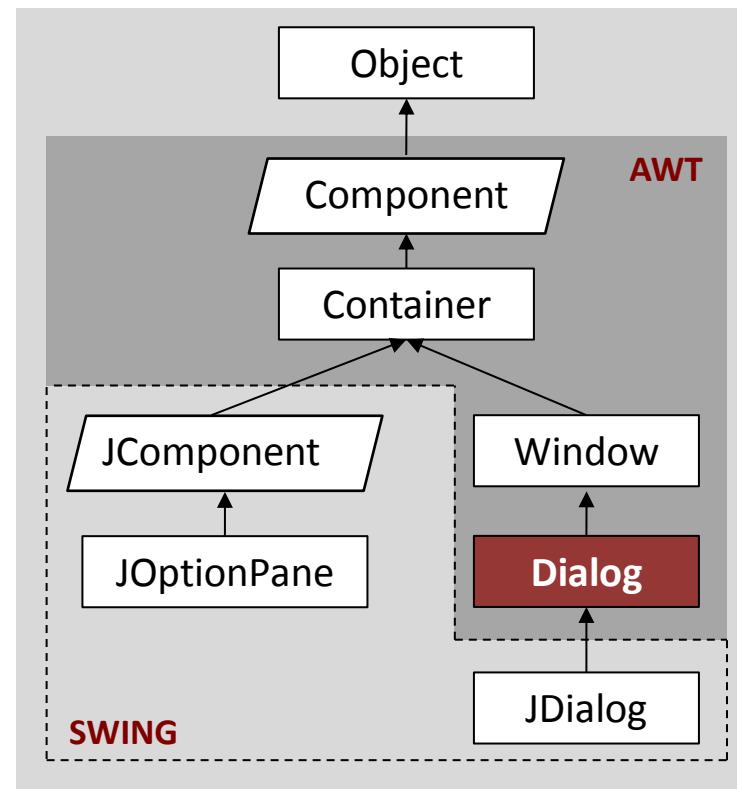


## ■ Package

- java.awt

## ■ Declaração

```
public class Dialog extends Window { ... }
```



Declaração	Funcionalidade
public <b>Dialog</b> ( Frame owner, String title, boolean modal )	Cria caixa de diálogo <b>invisível</b> , com a janela pai, título e tipo de funcionamento (true=modal ou false=modeless) especificados.
public <b>Dialog</b> ( Frame owner, String title )	Cria caixa de diálogo <b>invisível</b> e <b>modeless</b> , com a janela pai e título especificados.
public <b>Dialog</b> ( Frame owner, boolean modal )	Cria caixa de diálogo <b>invisível</b> e <b>sem título</b> , com janela pai e tipo de funcionamento (true=modal ou false=modeless) especificados.

## ■ Herdados

- [Component](#)
- [Container](#)
- [Window](#)

## ■ Próprios

Declaração	Funcionalidade
<pre>public void setBackground(Color bgColor)</pre> <ul style="list-style-type: none"><li>■ Exemplo: <code>getContentPane().setBackground(Color.RED);</code></li></ul>	Especifica cor de fundo da caixa de diálogo.
<pre>public void setResizable(boolean resizable)</pre>	<code>setResizable(false)</code> impede utilizador de redimensionar janela.
<pre>public void setTitle(String title)</pre>	Modifica título da caixa de diálogo.
<pre>public void setVisible(boolean b)</pre>	Mostra ( <code>setVisible(true)</code> ) ou esconde ( <code>setVisible(false)</code> ) caixa de diálogo.
<pre>public void setUndecorated(boolean undecorated)</pre>	Inibe/desinibe moldura da caixa de diálogo; Só pode ser chamado quando a caixa de diálogo não está visível.

- [Contentores de Componentes Gráficos](#)

- [Painéis](#)

- [Noção de Painel](#)
- [Classe JPanel](#)
- [Classe JLayeredPane](#)
- [Classe JRootPane](#)
- [Classe JScrollPane](#)

- [Superclasse Window](#)

- [Janelas](#)

- [Superclasse Frame](#)
- [Classe JFrame](#)

- [Caixas de Diálogo](#)

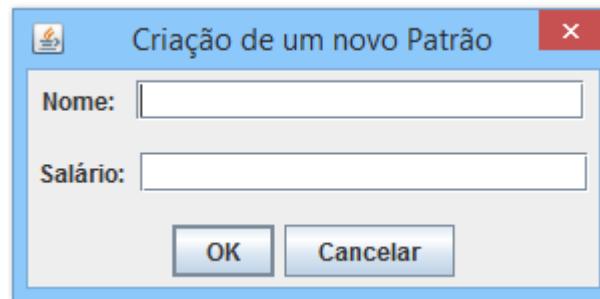
- [Introdução](#)
- [Classe JOptionPane](#)
- [Classe JFileChooser](#)
- [Próprias](#)
  - [Superclasse Dialog](#)
  - [Classe JDialog](#)



## ■ Criar Caixas de Diálogo Próprias

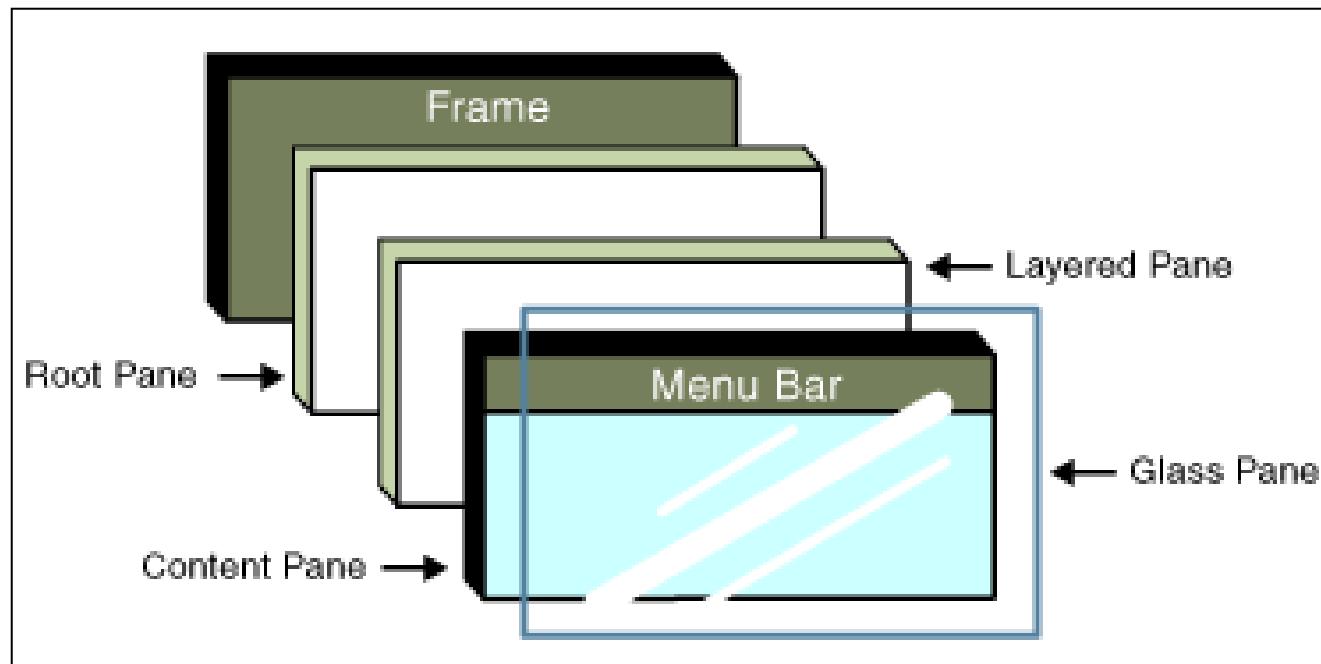
- Para diálogos
  - Personalizados
  - Mais complexos
- Tipo
  - Modal
  - Modeless

## ■ Exemplo



- Semelhante à da JFrame

- Contém
  - [Root Pane](#)

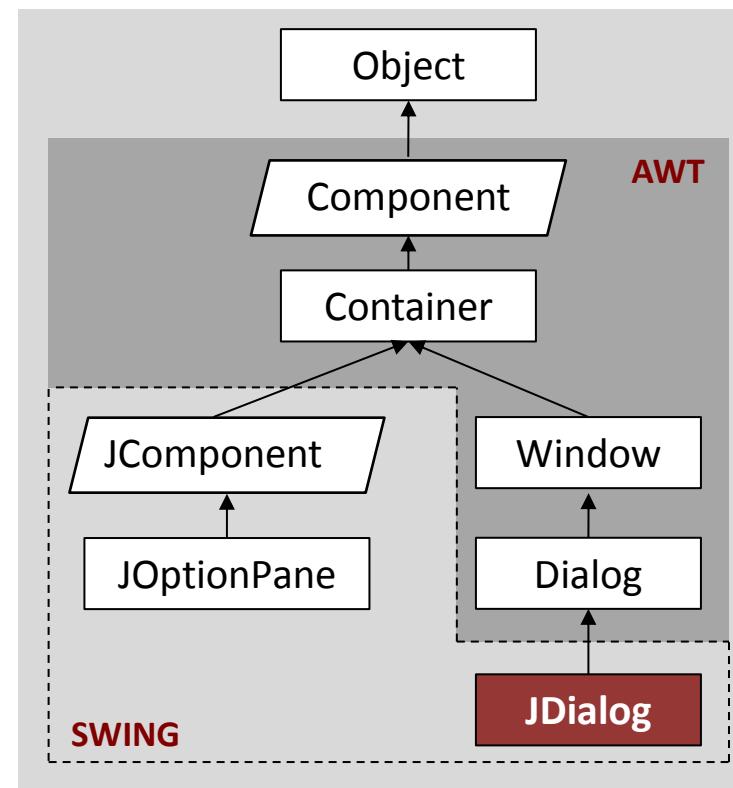


## ■ Package

- javax.swing

## ■ Declaração

```
public class JDialog extends Dialog  
    implements WindowConstants, Accessible, RootPaneContainer,  
    TransferHandler.HasGetTransferHandler { ... }
```



## ▪ Mais Usados

Declaração	Funcionalidade
public <b>JDialog</b> ( Frame owner, String title, boolean modal)	Cria caixa de diálogo <b>invisível</b> , com a janela pai, título e tipo de funcionamento (true=modal ou false=modeless) especificados.
public <b>JDialog</b> ( Frame owner, String title )	Cria caixa de diálogo <b>invisível</b> e modeless, com a janela pai e título especificados.
public <b>JDialog</b> ( Frame owner, boolean modal)	Cria caixa de diálogo <b>invisível</b> e <b>sem título</b> , com janela pai e tipo de funcionamento (true=modal ou false=modeless) especificados.

## ■ Herdados

- [Component](#)
- [Container](#)
- [Window](#)
- [Dialog](#)

## ■ Próprios

Declaração	Funcionalidade
public void <b>setLayout</b> (LayoutManager manager)	Especifica o gestor de posicionamento do Content Pane.
public void <b>setJMenuBar</b> (JMenuBar menuBar)	Especifica a barra de menus da caixa de diálogo.
public JRootPane <b>getRootPane</b> ()	Retorna Root Pane.
public JLAYEREDPANE <b>getLayeredPane</b> ()	Retorna Layered Pane.
public Component <b>getGlassPane</b> ()	Retorna Glass Pane.
public Container <b>getContentPane</b> ()	Retorna Content Pane.

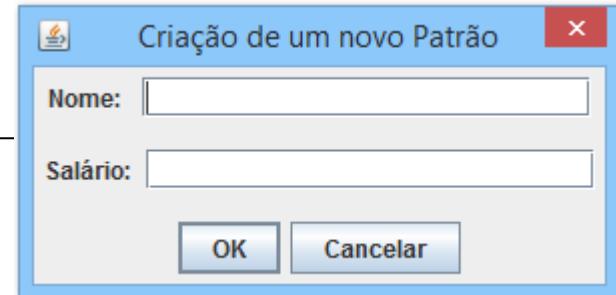
## ■ Próprios (2/2)

Declaração	Funcionalidade
<pre>public void setDefaultCloseOperation(int operation)</pre> <ul style="list-style-type: none"><li>▪ Exemplo: <code>setDefaultCloseOperation(DO NOTHING ON CLOSE)</code></li></ul>	<p>Especifica operação que ocorrerá, por omissão, quando utilizador inicia fecho da caixa de diálogo (botão close da moldura);</p> <p>Operações à escolha:</p> <ul style="list-style-type: none"><li><code>DO NOTHING ON CLOSE</code> // Não faz nada</li><li><code>HIDE ON CLOSE</code> // Esconde caixa</li><li><code>DISPOSE ON CLOSE</code> // Fecha caixa</li></ul> <p>Por omissão, botão close executa a operação <code>HIDE ON CLOSE</code>.</p>

**■ Procedimento**

1. Criar nova classe derivada da classe JDialog

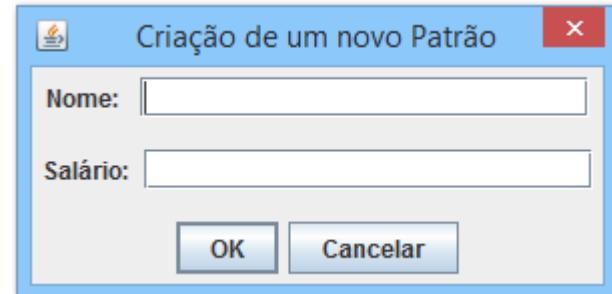
```
import javax.swing.*;  
  
public class DialogCriarPatrao extends JDialog {  
  
}
```



## ■ Procedimento

### 2. Criar construtor da nova classe ... Com parâmetro tipo JFrame

- Parâmetro tipo JFrame para
  - Receber janela pai da caixa de diálogo
  - Permitir caixa mostrada sempre sobre essa janela



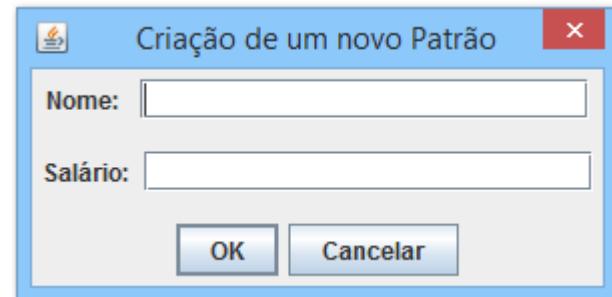
```
import javax.swing.*;  
  
public class DialogCriarPatrao extends JDialog {  
  
    public DialogCriarPatrao(JFrame pai) {  
        }  
    }  
}
```



## ■ Procedimento

3. No construtor, **invocar o construtor da superclasse JDialog**

- **Argumentos** definem propriedades da caixa:
  - Janela **pai**
  - **Título**
  - **Tipo de funcionamento**
    - modal // true
    - modeless // false



```
import javax.swing.*;

public class DialogCriarPatrao extends JDialog {

    public DialogCriarPatrao(JFrame pai) {
        super(pai, "Criação de um novo Patrão", true); //ex: modal ←
    }

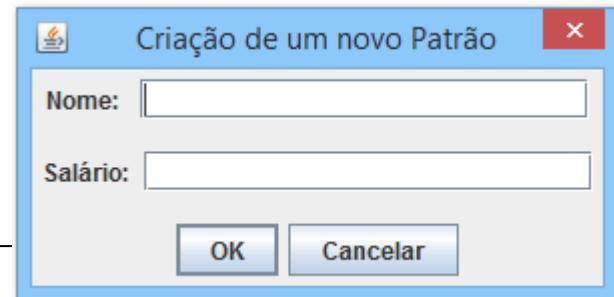
}
```

## ■ Procedimento

## 4. Definir a localização da caixa de diálogo

- Relativa à janela pai

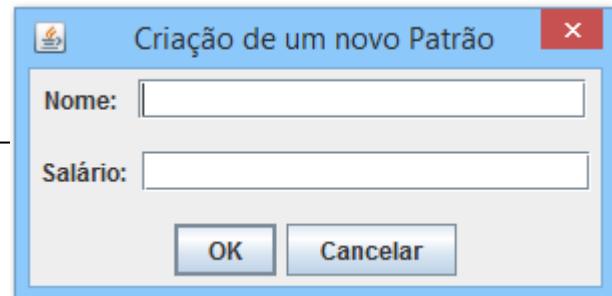
```
import javax.swing.*;  
  
public class DialogCriarPatrao extends JDialog {  
  
    private static final int DESVIO_X = 100, DESVIO_Y = 100;  
  
    public DialogCriarPatrao(JFrame pai) {  
        super(pai, "Criação de um novo Patrão", true); //ex: modal  
  
        setLocation(pai.getX() + DESVIO_X, pai.getY() + DESVIO_Y);  
    }  
}
```



## ■ Procedimento

### 5. Tornar a caixa de diálogo visível

```
import javax.swing.*;  
  
public class DialogCriarPatrao extends JDialog {  
  
    private static final int DESVIO_X = 100, DESVIO_Y = 100;  
  
    public DialogCriarPatrao(JFrame pai) {  
        super(pai, "Criação de um novo Patrão", true); //ex: modal  
  
        setLocation(pai.getX() + DESVIO_X, pai.getY() + DESVIO_Y);  
        setVisible(true);  
    }  
  
}
```



## ■ Procedimento

### 6. Adicionar componentes gráficos à caixa de diálogo (1/3)

```
import java.awt.*;
import javax.swing.*;

public class DialogCriarPatrao extends JDialog {

    private JTextField txtNome, txtSalario;

    private static final int DESVIO_X = 100, DESVIO_Y = 100;

    public DialogCriarPatrao(JFrame pai) {
        super(pai, "Criação de um novo Patrão", true); // ex: modal

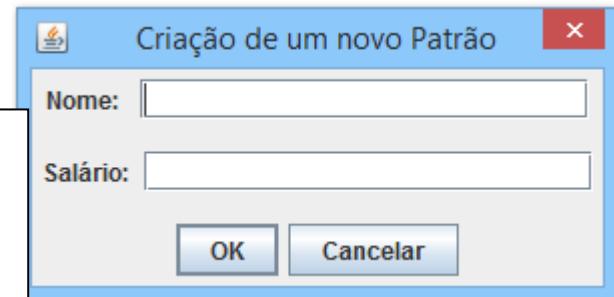
        setLayout(new GridLayout(3, 1));

        JPanel p1 = criarPainelNome();
        JPanel p2 = criarPainelSalario();
        JPanel p3 = criarPainelBotoes();

        add(p1);
        add(p2);
        add(p3);

        pack();
        setLocation(pai.getX() + DESVIO_X, pai.getY() + DESVIO_Y);
        setVisible(true);
    }

    private JPanel criarPainelNome() {...10 lines}
    private JPanel criarPainelSalario() {...9 lines}
    private JPanel criarPainelBotoes() {...12 lines}
    private JButton criarBotaoOK() {...4 lines}
    private JButton criarBotaoCancelar() {...4 lines}
}
```



## ■ Procedimento

### 6. Adicionar componentes GUI à caixa de diálogo (2/3)

```
import java.awt.*;
import javax.swing.*;

public class DialogCriarPatrao extends JDialog {

    private JTextField txtNome, txtSalario;

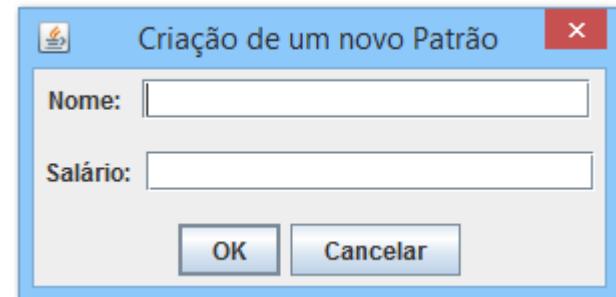
    private static final int DESVIO_X = 100, DESVIO_Y = 100;

    public DialogCriarPatrao(JFrame pai) { ...17 lines }

    private JPanel criarPainelNome() {
        JLabel label;
        label = new JLabel("Nome: ");
        final int CAMPO_LARGURA = 20;
        txtNome = new JTextField(CAMPO_LARGURA);
        JPanel p = new JPanel();
        p.add(label);
        p.add(txtNome);
        return p;
    }

    private JPanel criarPainelSalario() {
        JLabel lbl = new JLabel("Salário: ");
        final int CAMPO_LARGURA = 20;
        txtSalario = new JTextField(CAMPO_LARGURA);
        JPanel p = new JPanel();
        p.add(lbl);
        p.add(txtSalario);
        return p;
    }

    private JPanel criarPainelBotoes(){...12 lines}
    private JButton criarBotaoOK(){...4 lines}
    private JButton criarBotaoCancelar(){...4 lines}
}
```



## ■ Procedimento

### 6. Adicionar componentes GUI à caixa de diálogo (3/3)

```
import java.awt.*;
import javax.swing.*;

public class DialogCriarPatrao extends JDialog {

    private JTextField txtNome, txtSalario;

    private static final int DESVIO_X = 100, DESVIO_Y = 100;

    public DialogCriarPatrao(JFrame pai) {...17 lines}
    private JPanel criarPainelNome() {...10 lines}
    private JPanel criarPainelSalario() {...9 lines}

    private JPanel criarPainelBotoes(){
        JButton btnOK = criarBotaoOK();
        JButton btnCancel = criarBotaoCancelar();

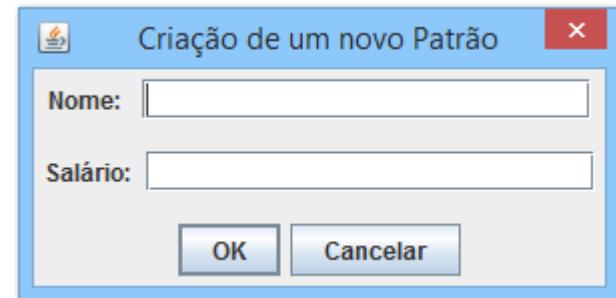
        getRootPane().setDefaultButton(btnOK);

        JPanel p = new JPanel();
        p.add(btnOK);
        p.add(btnCancel);

        return p;
    }

    private JButton criarBotaoOK(){
        JButton btn = new JButton("OK");
        return btn;
    }

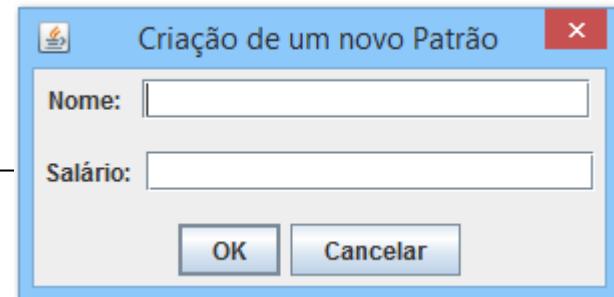
    private JButton criarBotaoCancelar(){
        JButton btn = new JButton("Cancelar");
        return btn;
    }
}
```



## ■ Procedimento

7. Criar e registrar objetos para tratarem os eventos gerados pelos botões de comando da caixa de diálogo (1/2)

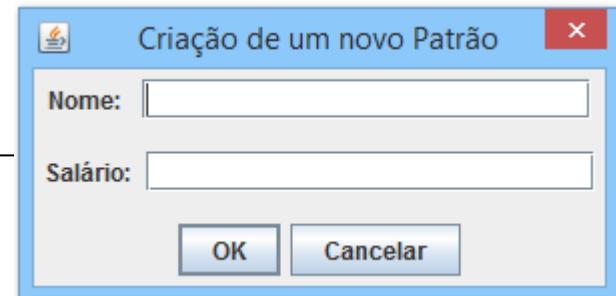
```
private JButton criarBotaoCancelar() {
    JButton btn = new JButton("Cancelar");
    btn.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            dispose();
        }
    });
    return btn;
}
```



## ■ Procedimento

7. Criar e registrar objetos para tratarem os eventos gerados pelos botões de comando da caixa de diálogo (2/2)

```
private JButton criarBotaoOK() {
    JButton btn = new JButton("OK");
    btn.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            try {
                String nome = txtNome.getText();
                float salario = Float.parseFloat(txtSalario.getText());
                Dados.adicionarPatrao(new Patrao(nome, salario));
                dispose();
            } catch (NumberFormatException | SalarioInvalidoException excecao) {
                JOptionPane.showMessageDialog(DialogCriarPatrao.this,
                    "Tem de introduzir um valor positivo ou zero no campo 'Salario'.",
                    "Criação de um novo Patrão",
                    JOptionPane.ERROR_MESSAGE);
                txtSalario.requestFocus();
            } catch (NomeInvalidoException excecao) {
                JOptionPane.showMessageDialog(DialogCriarPatrao.this,
                    "Tem de introduzir um nome no campo 'Nome'.",
                    "Criação de um novo Patrão",
                    JOptionPane.ERROR_MESSAGE);
                txtNome.requestFocus();
            }
        }
    });
    return btn;
}
```



### Transferência de Dados

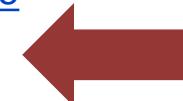
Para a classe **Dados**, responsável pelo armazenamento de dados da aplicação.

- [Introdução](#)
- [Componentes Gráficos](#)
  - [Introdução](#)
  - [Hierarquia de Classes](#)
  - [Interfaces](#)
  - [Categorias](#)
    - [Contentores de Componentes Gráficos](#)
    - [Apresentação de Informação](#)
    - [Controlos Básicos](#)
- [Gestores de Posicionamento](#)
- [Manipuladores de Eventos](#)
- [Bibliografia](#)
- [Índice Remissivo](#)



- Apresentação de Informação

- Classe JLabel
- Classe JSeparator



## ■ Apresentação Não-Editável

### ■ Texto

- Simples

Águia

// Linha de Texto

- Formatado (HTML)

Aves de Rapina

As aves de rapina são aves carnívoras que compartilham características semelhantes, bicos recurvados e pontiagudos, garras fortes e visão de longo alcance.  
Exemplos:

- Águia
- Falcão
- Abutre
- Milhafre

### ■ Imagem



### ■ Texto e Imagem



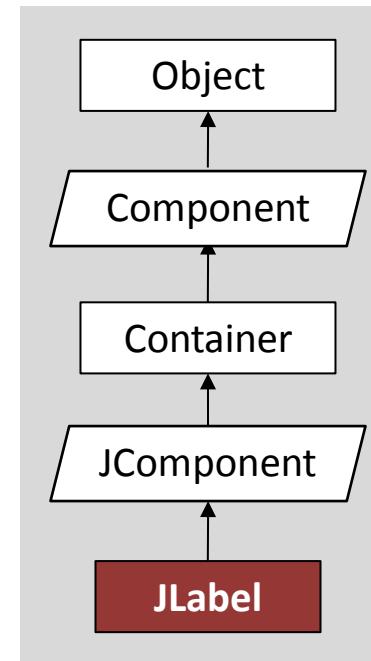
Águia

## ■ Package

- javax.swing

## ■ Declaração

```
public class JLabel extends JComponent  
    implements SwingConstants, Accessible { ... }
```



Declaração	Funcionalidade
<pre>public JLabel()</pre> <ul style="list-style-type: none"><li>▪ Exemplo</li></ul> <pre>JLabel lbl = new JLabel();</pre>	Cria objeto JLabel sem imagem e sem texto.
<pre>public JLabel(String text)</pre> <ul style="list-style-type: none"><li>▪ Exemplo <b>Texto Simples</b></li><li>▪ Exemplo <b>Texto Formatado</b></li></ul> <pre>JLabel lblAguia = new JLabel("Águia");</pre> <pre>String texto = "&lt;html&gt;As aves de rapina são aves carnívoras que compartilham "+     " &lt;P&gt; características semelhantes, bicos recurvados e " +     " &lt;P&gt; pontiagudos, garras fortes e visão de longo alcance." +     " &lt;P&gt; Exemplos:" +     "&lt;UL&gt;" +     " &lt;LI&gt;Águia" +     " &lt;LI&gt;Falcão" +     " &lt;LI&gt;Abutre" +     " &lt;LI&gt;Milhafre" +     "&lt;/UL&gt;&lt;/html&gt;";</pre> <pre>JLabel lblAvesRapina = new JLabel( texto );</pre> <pre>lblAvesRapina.setBorder( BorderFactory.createTitledBorder( "Aves de Rapina" ) );</pre>	Cria objeto JLabel com texto especificado, alinhado à esquerda e centrado na vertical.

Declaração	Funcionalidade
<pre>public JLabel( String text, int horizontalAlignment )</pre> <ul style="list-style-type: none"><li>▪ Exemplos: JLabel lblAguia = new JLabel( "Águia", SwingConstants.RIGHT ); JLabel lblAguia = new JLabel("Águia", JLabel.RIGHT );</li></ul>	Cria objeto JLabel com texto especificado, centrado na vertical e com alinhamento horizontal especificado.
<pre>public JLabel( Icon image )</pre> <ul style="list-style-type: none"><li>▪ Exemplo: JLabel lblAguia = new JLabel( new ImageIcon("aguia.jpg") );</li></ul>	Cria objeto JLabel com imagem especificada, centrada na vertical e horizontal.
<pre>public JLabel( Icon image, int horizontalAlignment )</pre> <ul style="list-style-type: none"><li>▪ Exemplo: JLabel lblAguia = new JLabel( new ImageIcon("aguia.jpg"), JLabel.LEFT );</li></ul>	Cria objeto JLabel com imagem especificada, centrada na vertical e com alinhamento horizontal especificado.
<pre>public JLabel( String text, Icon icon, int horizontalAlignment )</pre>	Cria objeto JLabel com texto, imagem e alinhamento horizontal especificados, centrados na vertical.

## ■ Herdados

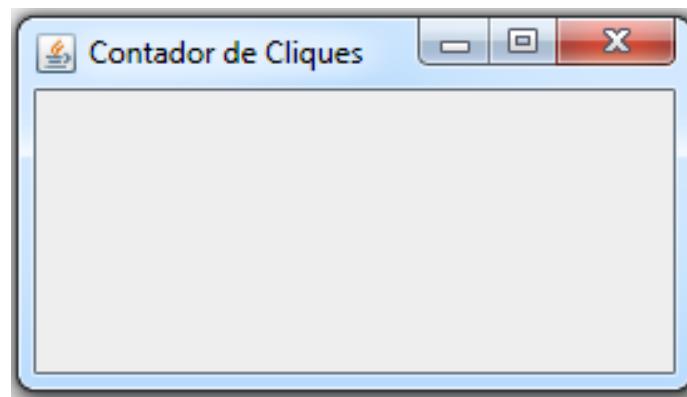
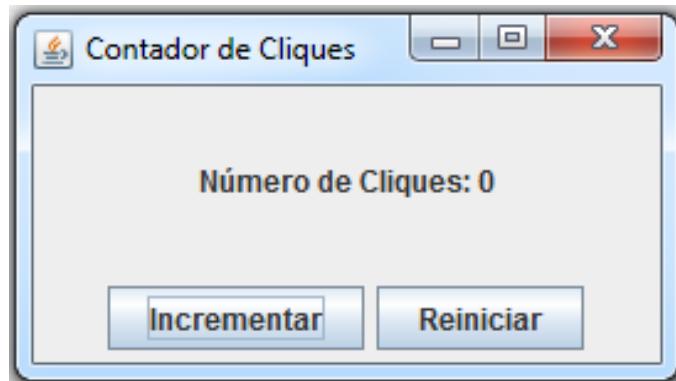
- [Component](#)
- [Container](#)
- [JComponent](#)

## ■ Próprios (1/2)

Declaração	Funcionalidade
<pre>public void setText(String text)</pre> <ul style="list-style-type: none"><li>▪ Ex: JLabel lblAguia = new JLabel(); lbl.setText("Águia");</li></ul>	Especifica texto.
<pre>public String getText()</pre> <ul style="list-style-type: none"><li>▪ Ex: String texto = lbl.getText();</li></ul>	Retorna texto.
<pre>public void setIcon( Icon defaultIcon )</pre> <ul style="list-style-type: none"><li>▪ Ex: lbl.setIcon(new ImageIcon("aguia.jpg"));</li></ul>	Especifica imagem.
<pre>public void setIconTextGap(int iconTextGap)</pre> <ul style="list-style-type: none"><li>▪ Ex: lbl.setIconTextGap( 10 );</li></ul>	Especifica intervalo entre texto e imagem.

## ■ Próprios (2/2)

Declaração	Funcionalidade
<pre>public void setVerticalAlignment( int alignment )</pre> <ul style="list-style-type: none"><li>▪ Exemplo 1: JLabel lblAguia = new JLabel("Águia"); lblAguia.setVerticalAlignment( SwingConstants.TOP );</li><li>▪ Exemplo 2: JLabel lblAguia = new JLabel("Águia"); lblAguia.setVerticalAlignment( JLabel.TOP );</li></ul>	Especifica alinhamento vertical do texto e imagem; Opções de alinhamento: <ul style="list-style-type: none"><li>▪ SwingConstants.CENTER</li><li>▪ SwingConstants.TOP</li><li>▪ SwingConstants.BOTTOM</li><li>▪ JLabel.CENTER</li><li>▪ JLabel.TOP</li><li>▪ JLabel.BOTTOM</li></ul>
<pre>public void setHorizontalAlignment( int alignment )</pre> <ul style="list-style-type: none"><li>▪ Exemplo 1: JLabel lblAguia = new JLabel("Águia"); lblAguia.setHorizontalAlignment( SwingConstants.LEFT );</li><li>▪ Exemplo 2: JLabel lblAguia = new JLabel("Águia"); lblAguia.setHorizontalAlignment( JLabel.LEFT );</li></ul>	Especifica alinhamento horizontal do texto e imagem; Opções de alinhamento: <ul style="list-style-type: none"><li>▪ SwingConstants.CENTER</li><li>▪ SwingConstants.RIGHT</li><li>▪ SwingConstants.LEFT</li><li>▪ JLabel.CENTER</li><li>▪ JLabel.RIGHT</li><li>▪ JLabel.LEFT</li></ul>



- Código Incompleto

```
public class ContadorGUI extends JFrame {  
    private JLabel lblNumero;  
    private static String s = "Número de Cliques: ";  
    private static final int JANELA_LARGURA = 270;  
    private static final int JANELA_ALTURA = 150;  
  
    public ContadorGUI() {  
        super("Contador de Cliques");  
  
        lblNumero = new JLabel(s + "0");  
  
        setDefaultCloseOperation(EXIT_ON_CLOSE);  
        setSize(JANELA_LARGURA, JANELA_ALTURA);  
        setLocationRelativeTo(null);  
        setVisible(true);  
    }  
}
```

- Código mais Completo

Tipos de Evento que <i>label</i> pode Gerar	Evento
Container	Componente adicionado à <i>label</i>
	Componente removido da <i>label</i>
Component	<i>Label</i> escondida
	<i>Label</i> mostrada
	<i>Label</i> movida
	<i>Label</i> redimensionada
Focus	<i>Label</i> adquire o foco de entrada
	<i>Label</i> perde o foco de entrada
Key	Tecla mantida premida
	Tecla libertada
	Tecla premida (toque)
Mouse	Clique no rato
	Rato entrou na <i>label</i>
	Rato saiu da <i>label</i>
	Botão do rato premido
	Botão do rato libertado
	Rato movido
	Rato arrastado (premido + movido)
	Roda do rato movida

## ■ Evento Mouse

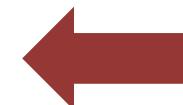
- Iniciado por
  - Clique no rato // exemplo: [duplo-clique](#)
  - Rato movido
  - Rato entrou/saiu da *label*
  - Rato com botão premido / libertado
  - Rato arrastado
  - Roda do rato movida
- Tratamento
  - Realizado por objetos de classes que:
    - **Implementem** o interface MouseListener // para implementar todos os métodos
    - **Derivadas** da classe MouseAdapter // para implementar alguns métodos
  - Exemplo

Declaração	Funcionalidade
<pre>public synchronized void addMouseListener( ActionListener l )</pre> <ul style="list-style-type: none"><li>■ Exemplo:<pre>JLabel lbl = new JLabel("Águia"); // Classe TrataEvento implementa interface MouseListener TrataEvento t = new TrataEvento(); lbl.addMouseListener( t );</pre></li></ul>	Regista objeto para tratar evento do tipo Mouse.

- <http://docs.oracle.com/javase/tutorial/uiswing/components/label.html>

- Apresentação de Informação

- Classe JLabel
- Classe JSeparator

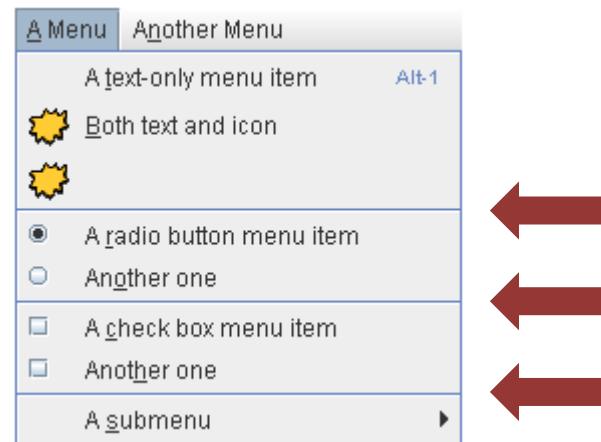


## ■ Separar Componentes Gráficos

- Qualquer parte do GUI
- Separador = Linha

## ■ Exemplos

- Menus
  - Separador de grupos lógicos de itens de menus // separador horizontal



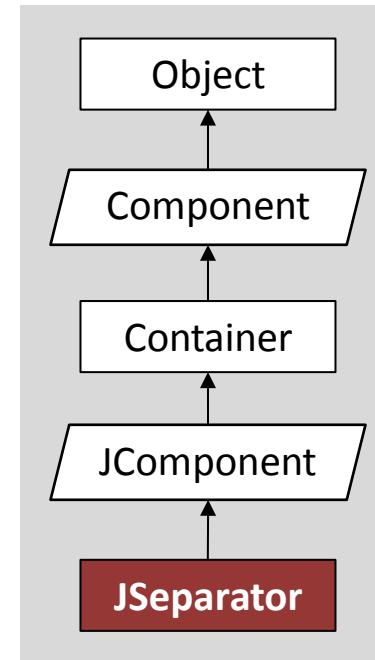
- Barra de Ferramentas
  - Separador de Ferramentas // separador vertical

## ■ Package

- javax.swing

## ■ Declaração

```
public class JSeparator extends JComponent  
    implements SwingConstants, Accessible { ... }
```



Declaração	Funcionalidade
<pre>public JSeparator()</pre> <ul style="list-style-type: none"><li>▪ Exemplo: JSeparator sp1 = new JSeparator();</li></ul>	Cria separador horizontal.
<pre>public JSeparator( int orientation )</pre> <ul style="list-style-type: none"><li>▪ Exemplos: JSeparator sp1 = new JSeparator (JSeparator.VERTICAL); Ex: JSeparator sp2 = new JSeparator (JSeparator.HORIZONTAL);</li></ul>	Cria separador horizontal ou vertical.

**▪ Herdados**

- [Component](#)
- [Container](#)
- [JComponent](#)

**▪ Próprio**

- Mais usado

Declaração	Funcionalidade
<pre>public void setOrientation( int orientation )  ▪ Exemplo   JSeparator sp1 = new JSeparator();   sp1.setOrientation( JSeparator.VERTICAL );</pre>	Especifica orientação.

- <http://docs.oracle.com/javase/tutorial/uiswing/components/separator.html>

- [Introdução](#)
- [Componentes Gráficos](#)
  - [Introdução](#)
  - [Hierarquia de Classes](#)
  - [Interfaces](#)
  - [Categorias](#)
    - [Contentores de Componentes Gráficos](#)
    - [Apresentação de Informação](#)
    - [Controlos Básicos](#)
- [Gestores de Posicionamento](#)
- [Manipuladores de Eventos](#)
- [Bibliografia Geral](#)
- [Índice Remissivo](#)

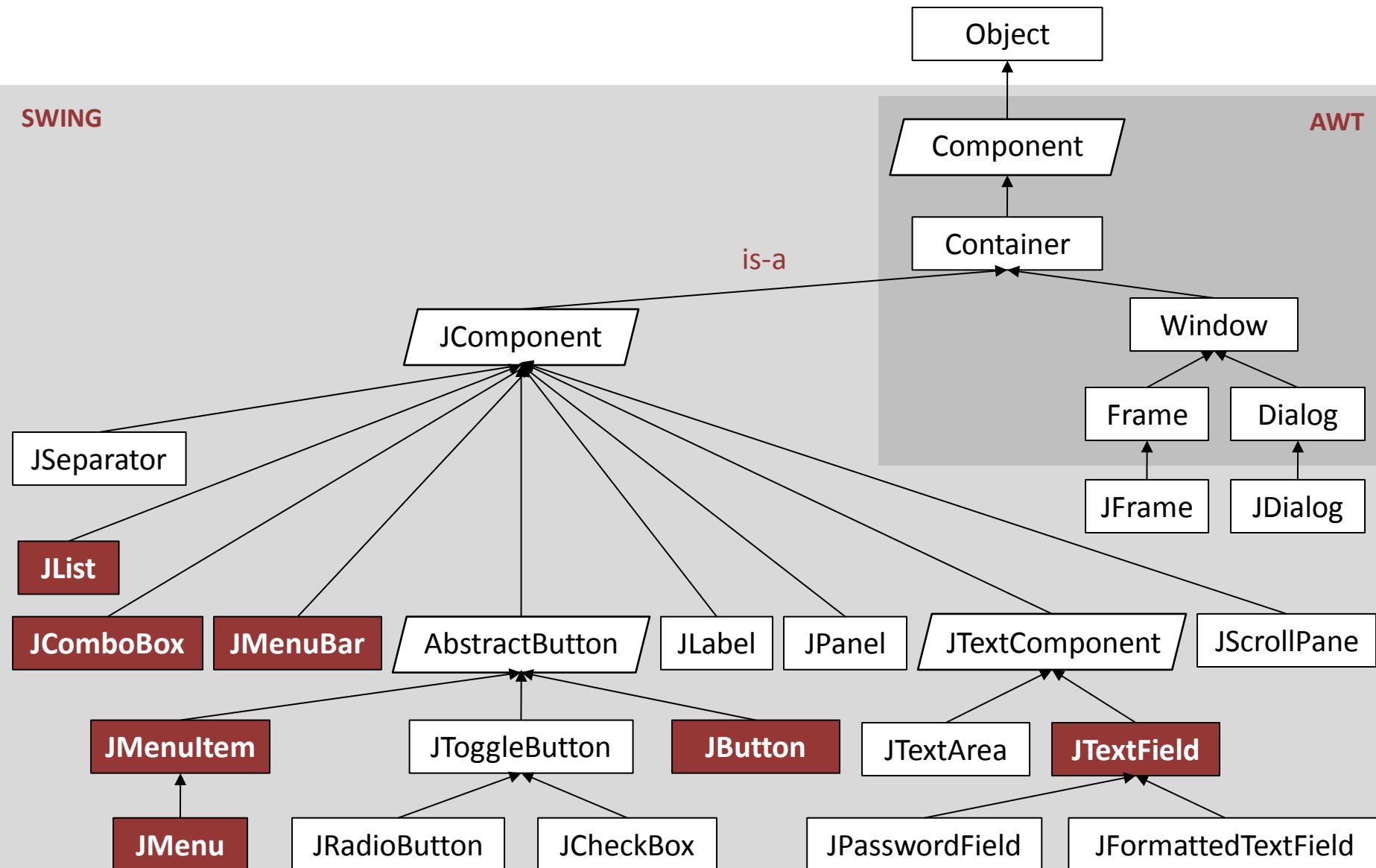


- [Controlos Básicos](#)
  - [Hierarquia de Classes](#)
  - [Superclasses](#)
    - [AbstractButton](#)
    - [JToggleButton](#)
  - [Botão de Comando](#)
  - [Entrada de Texto](#)
  - [Menu](#)
  - [Caixa de Listagem](#)
  - [Caixa de Combinação](#)
  - [Caixa de Verificação](#)
  - [Botão de Opção](#)

# Hierarquia de Classes de Controlos Básicos

SWING

AWT



Legenda: Classe Abstrata

Classe Instanciável

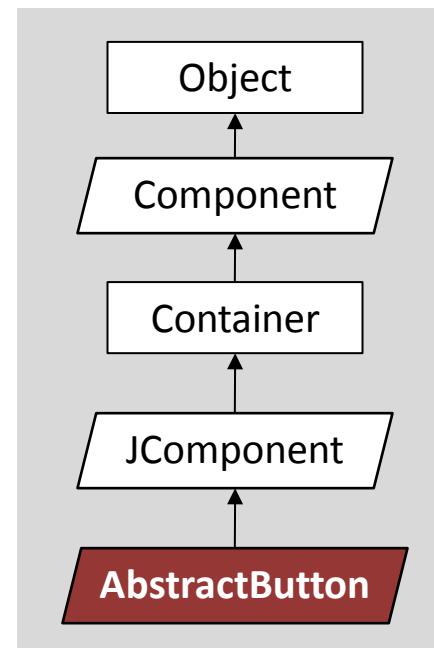
Controlos Básicos em PPROG

## ■ Package

- javax.swing

## ■ Declaração

```
public abstract class AbstractButton extends JComponent  
    implements ItemSelectable, SwingConstants { ... }
```



▪ Métodos Próprios (1/3)

Declaração	Funcionalidade
<p>public void <b>setMnemonic</b>(char mnemonic)</p> <ul style="list-style-type: none"> <li>▪ Exemplo:</li> </ul> <pre data-bbox="124 396 1104 439">btnCancelar.setMnemonic(KeyEvent.VK_C); // ALT + c</pre>	Especifica tecla de atalho
<p>public void <b>setDisplayedMnemonicIndex</b>(int index)</p> <ul style="list-style-type: none"> <li>▪ Exemplo:</li> </ul> <pre data-bbox="124 597 1104 640">btnCancelar.setDisplayedMnemonicIndex(0); // ALT + c</pre>	Especifica tecla de atalho
<p>public void <b>setVerticalAlignment</b>( int alignment )</p>	<p>Especifica alinhamento vertical do texto e ícone;</p> <p>Opções de alinhamento:</p> <ul style="list-style-type: none"> <li>▪ SwingConstants.CENTER (por omissão)</li> <li>▪ SwingConstants.TOP</li> <li>▪ SwingConstants.BOTTOM</li> </ul>
<p>public void <b>setHorizontalAlignment</b>( int alignment )</p> <ul style="list-style-type: none"> <li>▪ Exemplos:</li> </ul> 	<p>Especifica alinhamento horizontal do texto e ícone;</p> <p>Opções de alinhamento:</p> <ul style="list-style-type: none"> <li>▪ SwingConstants.CENTER (por omissão)</li> <li>▪ SwingConstants.RIGHT</li> <li>▪ SwingConstants.LEFT</li> </ul>

## ▪ Métodos Próprios (2/3)

Declaração	Funcionalidade
<pre>public void setVerticalTextPosition( int textPosition )</pre> <ul style="list-style-type: none"><li>▪ Exemplos:</li></ul> 	Especifica a posição <b>vertical</b> do texto relativamente ao ícone; Opções: <ul style="list-style-type: none"><li>▪ SwingConstants.CENTER (por omissão)</li><li>▪ SwingConstants.TOP</li><li>▪ SwingConstants.BOTTOM</li></ul>
<pre>public void setHorizontalTextPosition( int textPosition )</pre> <ul style="list-style-type: none"><li>▪ Exemplos:</li></ul> 	Especifica a posição <b>horizontal</b> do texto relativamente ao ícone; Opções: <ul style="list-style-type: none"><li>▪ SwingConstants.RIGHT (por omissão)</li><li>▪ SwingConstants.LEFT</li><li>▪ SwingConstants.CENTER</li></ul>
<pre>public void setText( String text )</pre>	Especifica texto: simples ou formatado por HTML.
<pre>public String getText()</pre>	Retorna o texto.

## ■ Métodos Próprios (3/3)

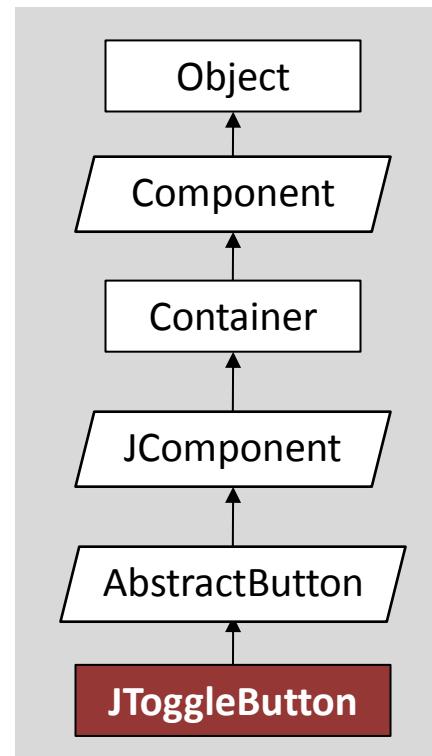
Declaração	Funcionalidade
public void <b>setIcon</b> ( Icon defaultIcon )	Especifica o ícone.
public void <b>setDisabledIcon</b> ( Icon disabledIcon )  ▪ Exemplo:  btn.setDisabledIcon ( new ImageIcon("c:\\x.gif") );	Especifica ícone para botão inibido.
public void <b>setRolloverIcon</b> ( Icon rolloverIcon )  ▪ Exemplo:  btn.setRolloverIcon( new ImageIcon("c:\\tips.gif") );	Especifica novo ícone à passagem do apontador do rato.
public void <b>setIconTextGap</b> ( int iconTextGap )  ▪ Exemplo:  	Especifica intervalo entre texto e ícone.

## ■ Package

- javax.swing

## ■ Declaração

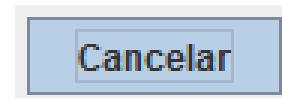
```
public abstract class JTogglleButton extends AbstractButton  
    implements Accessible { ... }
```



- Interesse

- Implementação de botões de dois estados

- Selecionado



- Desselecionado

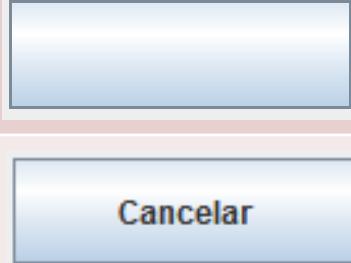


- Herdada pelas classes

- JCheckBox

- JRadioButton

## ■ Construtores

Declaração	Funcionalidade
<pre>public JToggleButton()</pre> <ul style="list-style-type: none"><li>▪ Exemplo: <code>JToggleButton tbt1 = new JToggleButton();</code></li></ul>	Cria botão visível e vazio.
<pre>public JToggleButton( String text )</pre> <ul style="list-style-type: none"><li>▪ Exemplo: <code>JToggleButton tbtCancelar = new JToggleButton("Cancelar");</code></li></ul>	 Cria botão visível com texto.
<pre>public JToggleButton( Icon icon )</pre> <ul style="list-style-type: none"><li>▪ Exemplo: <code>JToggleButton tbtCancelar = new JToggleButton( new ImageIcon("x.gif"));</code></li></ul>	 Cria botão visível com ícone.
<pre>public JButton( String text, Icon icon )</pre> <ul style="list-style-type: none"><li>▪ Exemplo: <code>Icon icon = new ImageIcon("x.gif"); JToggleButton tbtCancelar = new JToggleButton("Cancelar", icon );</code></li></ul>	Cria botão visível com ícone e texto.  

- **Bibliografia**

- <http://docs.oracle.com/javase/tutorial/uiswing/components/button.html>

- [Controlos Básicos](#)
  - [Hierarquia de Classes](#)
  - [Superclasses](#)
    - [AbstractButton](#)
    - [JToggleButton](#)
  - [Botão de Comando](#)
  - [Entrada de Texto](#)
  - [Menu](#)
  - [Caixa de Listagem](#)
  - [Caixa de Combinação](#)
  - [Caixa de Verificação](#)
  - [Botão de Opção](#)



- Permitir ao Utilizador

- Ordenar execução de comando (i.e., **ação**)

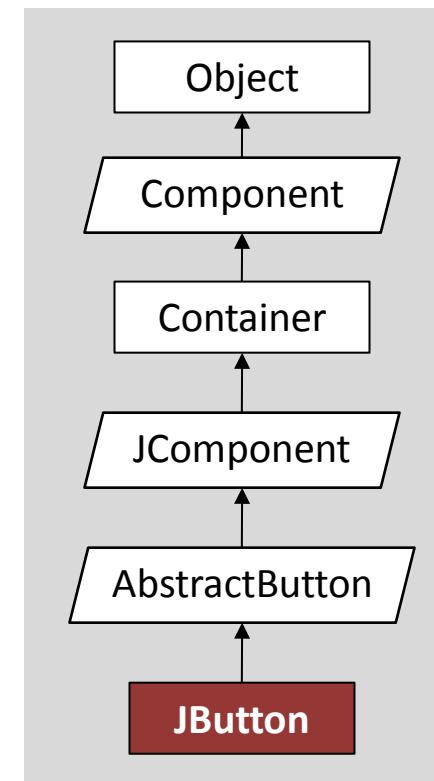


## ■ Package

- javax.swing

## ■ Declaração

```
public class JButton extends AbstractButton  
    implements Accessible { ... }
```



## ■ Construtores

Declaração	Funcionalidade
<pre>public JButton()     ▪ Exemplo:         JButton btnCancelar = new JButton();</pre>	Cria objeto JButton visível vazio.
<pre>public JButton( String text )     ▪ Exemplo:         JButton btnCancelar = new JButton("Cancelar");</pre>	Cria objeto JButton visível com texto.
<pre>public JButton( Icon icon )     ▪ Exemplo:         JButton btnCancelar = new JButton(new ImageIcon("x.gif"));</pre>	Cria objeto JButton visível com ícone.
<pre>public JButton( String text, Icon icon )     ▪ Exemplo:         JButton btnCancelar = new JButton("Cancelar", new ImageIcon("x.gif"));</pre>	Cria objeto JButton visível com ícone e texto.

## ■ Por omissão

- Botão é visível

## ■ Métodos

## ■ Herdados

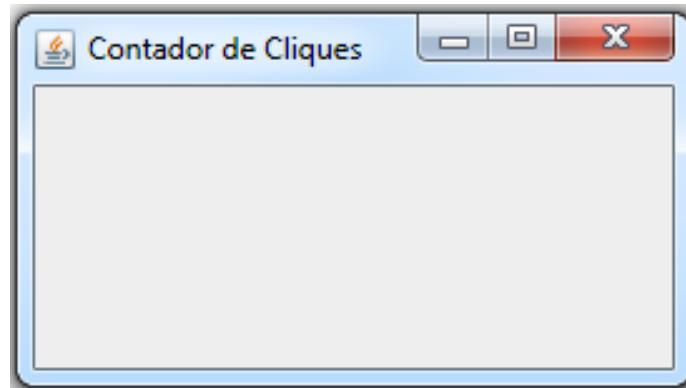
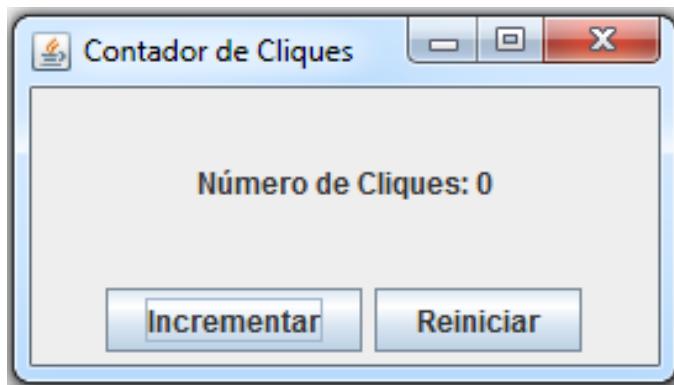
- [Component](#)
- [Container](#)
- [JComponent](#)
- [AbstractButton](#)

## ■ Próprios

Declaração	Funcionalidade
<code>public void setText(String text)</code>	Especifica o texto (simples ou HTML).
<code>public String getText()</code>	Retorna o texto.
<code>public void setIcon( Icon defaultIcon )</code>	Especifica o ícone.
<code>public void setIconTextGap(int iconTextGap)</code>	Especifica o intervalo entre texto e ícone.

# Classe JButton

## ■ Exemplo



## ■ Código Incompleto

```
public class ContadorGUI extends JFrame {  
    private JLabel lblNumero;  
    private JButton btnIncrementar, btnReiniciar;  
    private static String s = "Número de Cliques: ";  
    private static final int JANELA_LARGURA = 270;  
    private static final int JANELA_ALTURA = 150;  
  
    public ContadorGUI() {  
        super("Contador de Cliques");  
        criarComponentes();  
        setDefaultCloseOperation(EXIT_ON_CLOSE);  
        setSize(JANELA_LARGURA, JANELA_ALTURA);  
        setLocationRelativeTo(null);  
        setVisible(true);  
    }  
  
    private void criarComponentes(){  
        lblNumero = new JLabel(s + "0");  
        btnIncrementar = new JButton("Incrementar");  
        btnReiniciar = new JButton("Reiniciar");  
    }  
}
```

## ■ Código Completo

Tipos de Evento que Botão pode Gerar	Evento
Action	Clique no botão
Container	Componente adicionado ao botão
	Componente removido do botão
Component	Botão escondido
	Botão mostrado
	Botão movido
	Botão redimensionado
Focus	Botão adquire o foco
	Botão perde o foco
Key	Tecla mantida premida
	Tecla libertada
	Tecla premida (toque)
Mouse	Clique no rato
	Rato entrou no botão
	Rato saiu do botão
	Botão do rato premido
	Botão do rato libertado
	Rato movido
	Rato arrastado (premido + movido)
	Roda do rato movida

## ■ Evento Action

- Iniciado
  - Clique no botão de comando
- Tratamento
  - Realizado
    - Por objetos de classes que implementem o interface ActionListener
- Exemplo

Declaração	Funcionalidade
<pre>public void <b>addActionListener</b>( ActionListener l )</pre> <ul style="list-style-type: none"><li>■ Exemplo:<pre>JButton btn = new JButton("Cancelar");  // Classe TrataEvento implementa interface ActionListener TrataEvento t = new TrataEvento();  btn.addActionListener( t );</pre></li></ul>	Regista objeto para tratar evento do tipo Action.

- <http://docs.oracle.com/javase/tutorial/uiswing/components/button.html>

- [Controlos Básicos](#)
  - [Hierarquia de Classes](#)
  - [Superclasses](#)
    - [AbstractButton](#)
    - [JToggleButton](#)
  - [Botão de Comando](#)
  - [Entrada de Texto](#)
  - [Menu](#)
  - [Caixa de Listagem](#)
  - [Caixa de Combinação](#)
  - [Caixa de Verificação](#)
  - [Botão de Opção](#)



- Permite ao utilizador da aplicação
  - Introdução de texto
  - Edição de texto

- Tipos de Componentes Swing
  - JTextField

- Aceita uma só linha de texto

- JTextArea

- Pode aceitar múltiplas linhas de texto

- JPasswordField

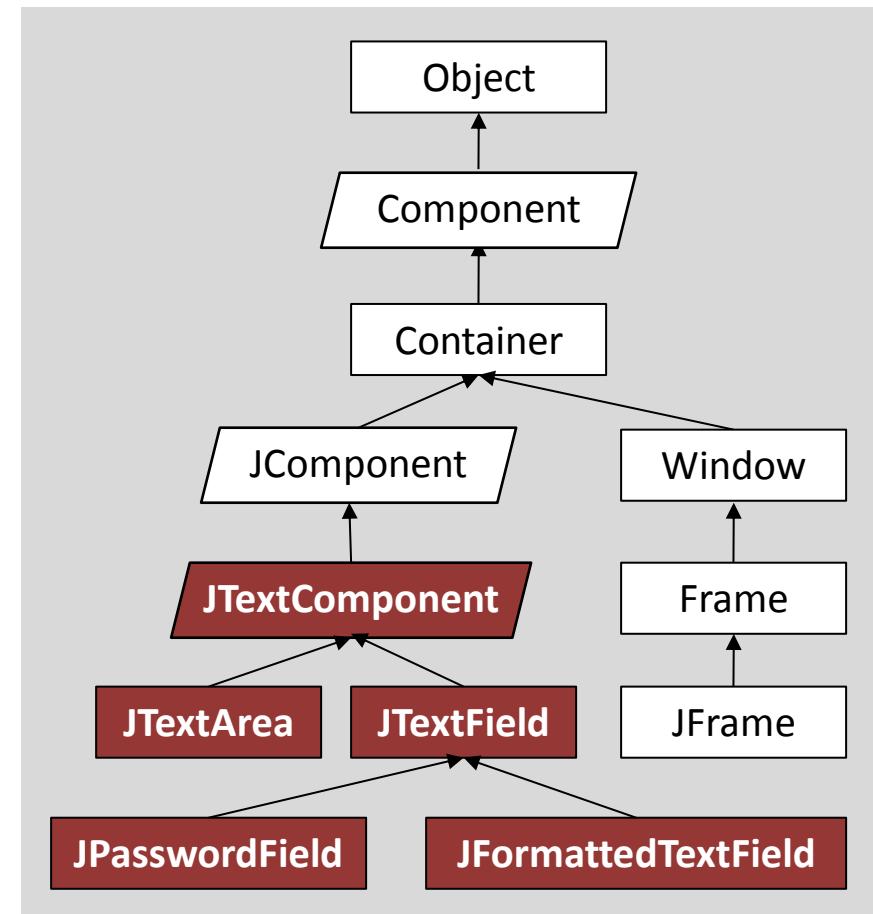
- Aceita só uma linha de texto
  - Esconde texto introduzido

- JFormattedTextField

- Para entradas específicas
  - Exemplos
    - Datas e Endereços IP

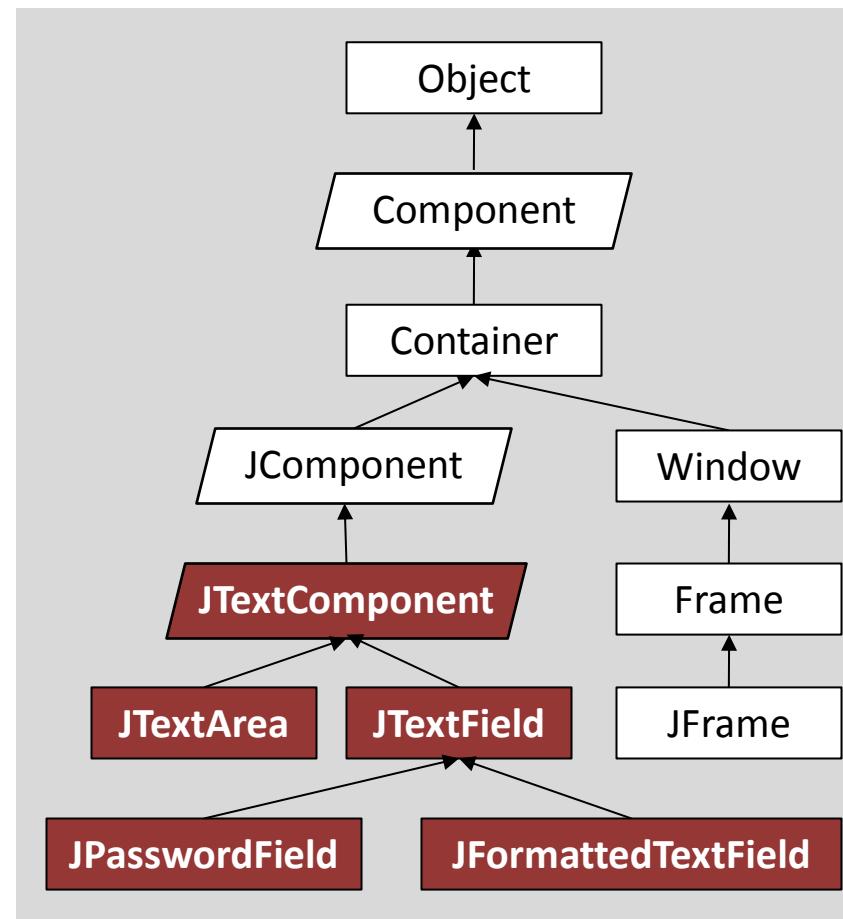
- Herdam Classe JTextComponent

- Classe abstrata
    - Não instanciável



- Classe base de componentes swing de texto

- Características comuns



- Package

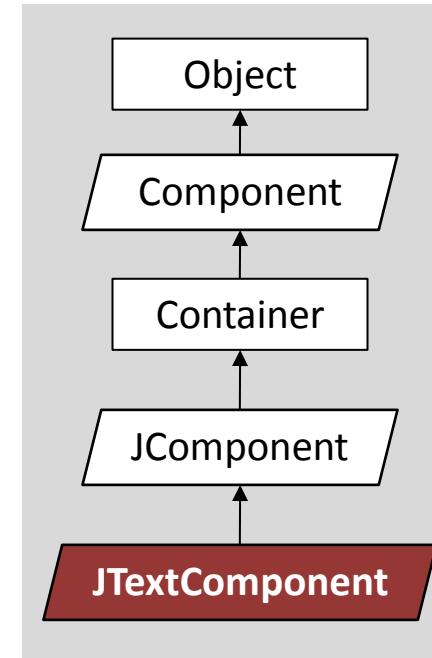
- javax.swing.text

- Declaração

```
public abstract class JTextComponent extends JComponent  
    implements Scrollable, Accessible { ... }
```

- Classe Abstrata

- Não Instanciável



## ▪ Herdados

- [Component](#)
- [Container](#)
- [JComponent](#)

## ▪ Próprios

- Mais usados

Declaração	Funcionalidade
<code>public void setText( String t )</code>	Especifica texto t para o componente; <code>setText(null)</code> ou <code>setText("")</code> elimina texto do componente.
<code>public String getText( )</code>	Retorna texto contido no componente.
<code>public void setEditable(boolean b)</code>	Especifica se o texto pode ou não ser editado.

- Entrada de Texto Simples

- Linha de texto

- Exemplo

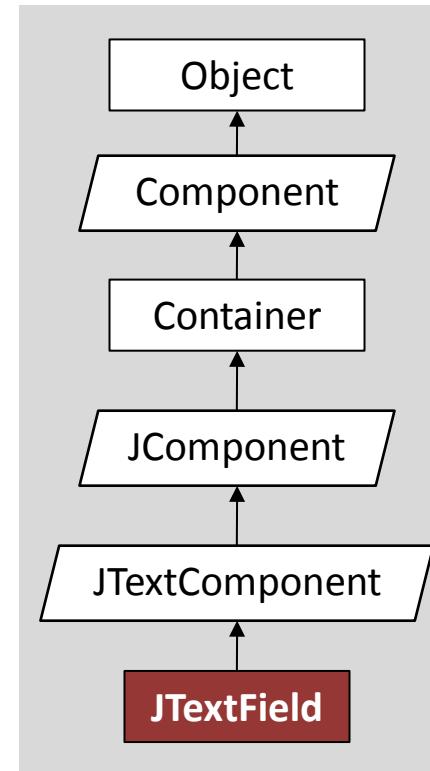


- Package

- javax.swing

- Declaração

```
public class JTextField extends JTextComponent  
    implements SwingConstants { ... }
```



Declaração	Funcionalidade
<pre>public JTextField( )</pre> <ul style="list-style-type: none"><li>▪ Exemplo:</li></ul> <pre>JTextField txtNome = new JTextField();</pre>	Cria campo de texto vazio e com nº de colunas 0 (tamanho nulo).
<pre>public JTextField( String text )</pre>	Cria campo de texto inicializado com o texto especificado.
<pre>public JTextField( int columns )</pre>	Cria campo de texto vazio e com nº de colunas especificado (tamanho).
<pre>public JTextField( String text , int columns )</pre>	Cria campo de texto inicializado com texto e nº de colunas especificados.

## ▪ Herdados

- [Component](#)
- [Container](#)
- [JComponent](#)
- [JTextComponent](#)

## ▪ Próprios

- Mais usados

Declaração	Funcionalidade
<code>public void setFont( Font f )</code>	Especifica tipo, estilo e tamanho das letras.
<code>public void setHorizontalAlignment( int alignment )</code>	Especifica alinhamento horizontal do texto; Opções: <ul style="list-style-type: none"><li>▪ JTextField.LEFT ou JTextField.LEADING</li><li>▪ JTextField.CENTER</li><li>▪ JTextField.RIGHT ou JTextField.TRAILING</li></ul>

Tipos de Evento que Campo de Texto pode Gerar	Evento
Action	Terminada edição (tecla ENTER premida)
Container	Componente adicionado ao campo de texto Componente removido do campo de texto
Component	Campo de texto escondido Campo de texto mostrado Campo de texto movido Campo de texto redimensionado
Focus	Campo de texto adquire o foco de entrada Campo de texto perde o foco de entrada
Key	Tecla mantida premida Tecla libertada Tecla premida (toque)
Mouse	Clique no rato Rato entrou no campo de texto Rato saiu do campo de texto Botão do rato premido Botão do rato libertado Rato movido Rato arrastado (premido + movido) Roda do rato movida
Text	Texto alterado

## ■ Evento Action

- Iniciado premindo tecla ENTER dentro do campo de texto
- Tratamento
  - Realizado
    - Por objetos de classes que implementem o interface ActionListener
  - Exemplo

Declaração	Funcionalidade
<pre>public void <b>addActionListener</b>( ActionListener l )      ■ Exemplo:      JTextField txt = new JTextField( 10 );      // Classe TrataEvento implementa interface ActionListener     TrataEvento t = new TrataEvento();      txt.addActionListener( t );</pre>	Regista objeto para tratar evento do tipo Action.

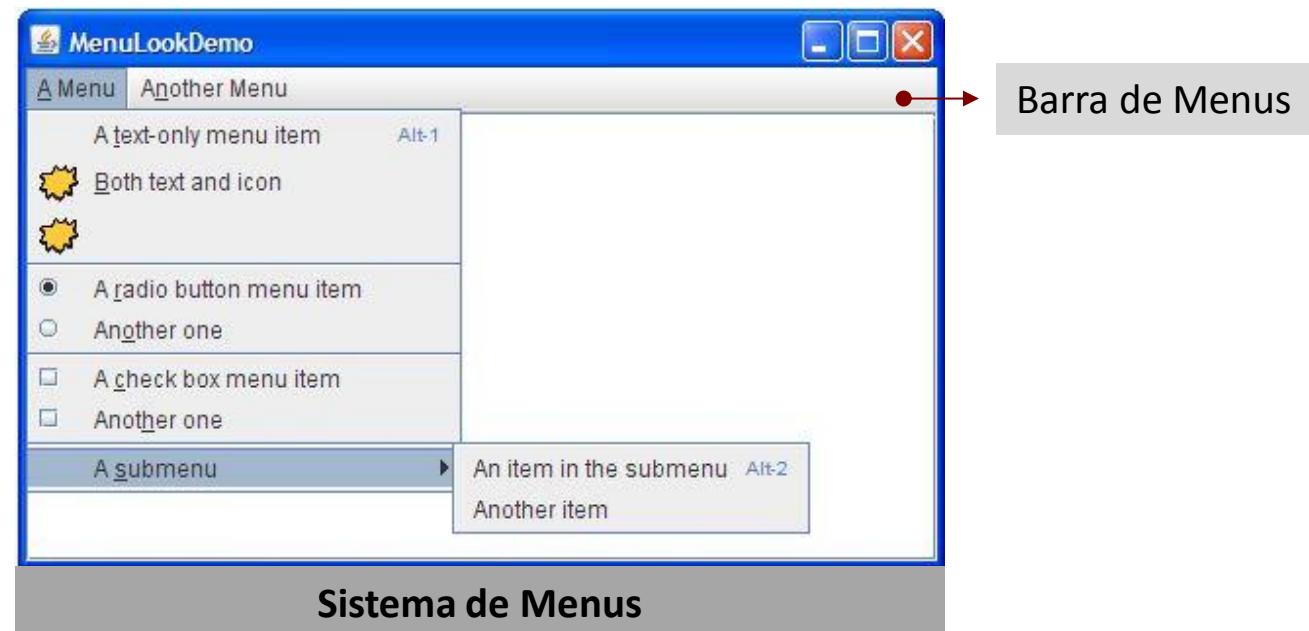
- <http://docs.oracle.com/javase/tutorial/uiswing/components/textfield.html>

- [Controlos Básicos](#)
  - [Hierarquia de Classes](#)
  - [Superclasses](#)
    - [AbstractButton](#)
    - [JToggleButton](#)
  - [Botão de Comando](#)
  - [Entrada de Texto](#)
  - [Menu](#)
  - [Caixa de Listagem](#)
  - [Caixa de Combinação](#)
  - [Caixa de Verificação](#)
  - [Botão de Opção](#)



## Sistema de Menus

- Permitir ao utilizador escolher
  - Uma de várias opções
- Salvar espaço do GUI
  - Só ocupa espaço ... da Barra de Menus



- Sistema de Menus

- Estrutura Hierárquica
  - Menus organizados
  - Vários níveis

- Constituído
  - Barra de Menus

- Contém Menus
  - Contêm
    - Itens de Menu
    - Separadores
    - Submenus



- **Colocada**

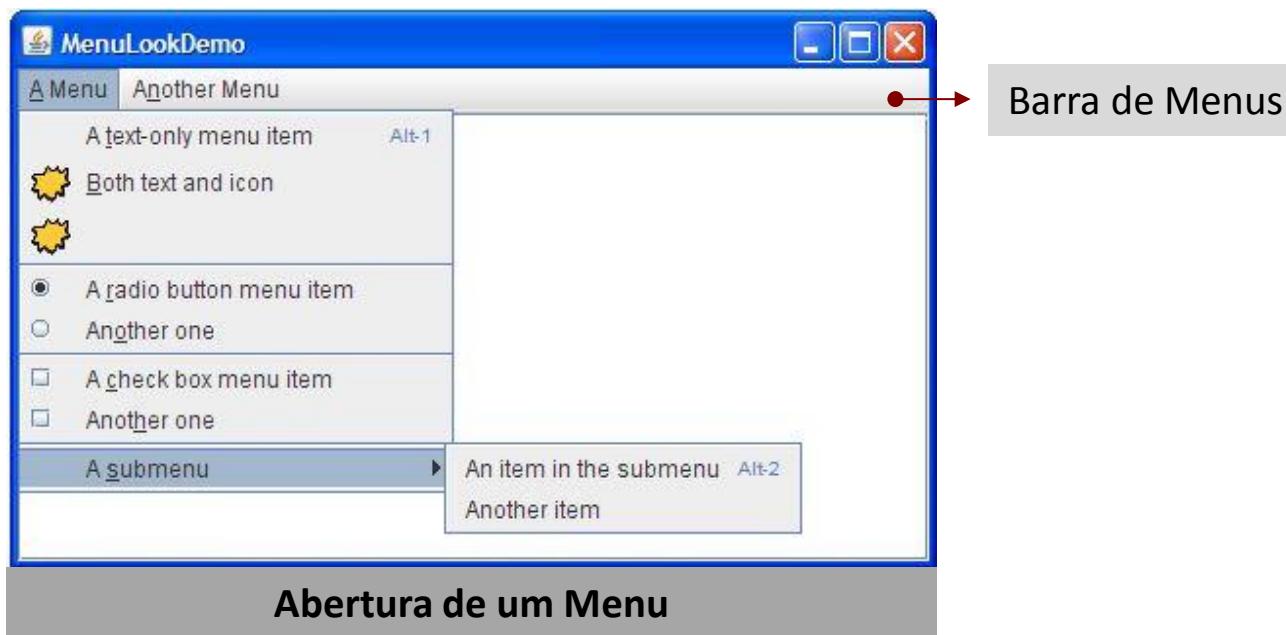
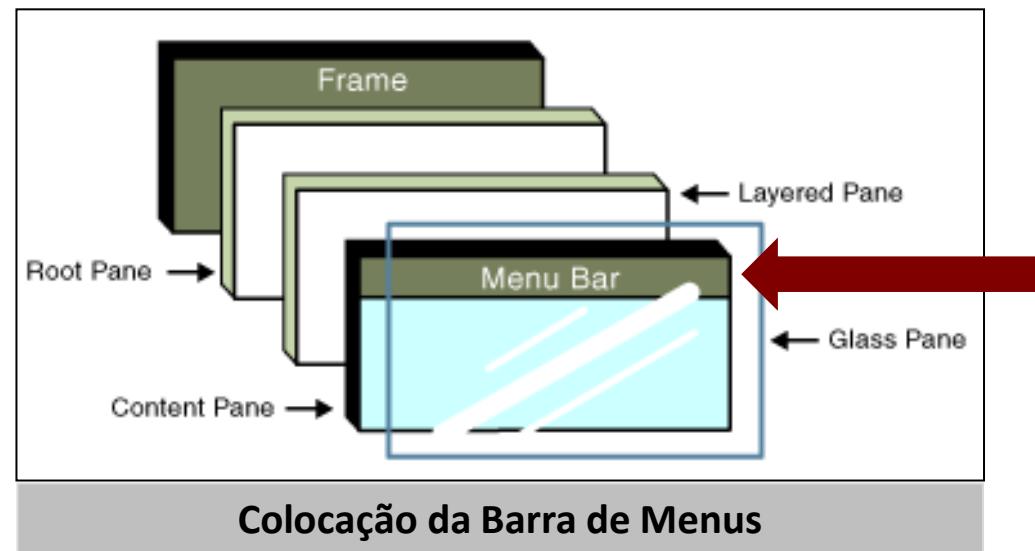
- Topo da janela
- Acima do *Content Pane*

- **Constituída**

- Menus de nível superior

- **Clique no nome de menu**

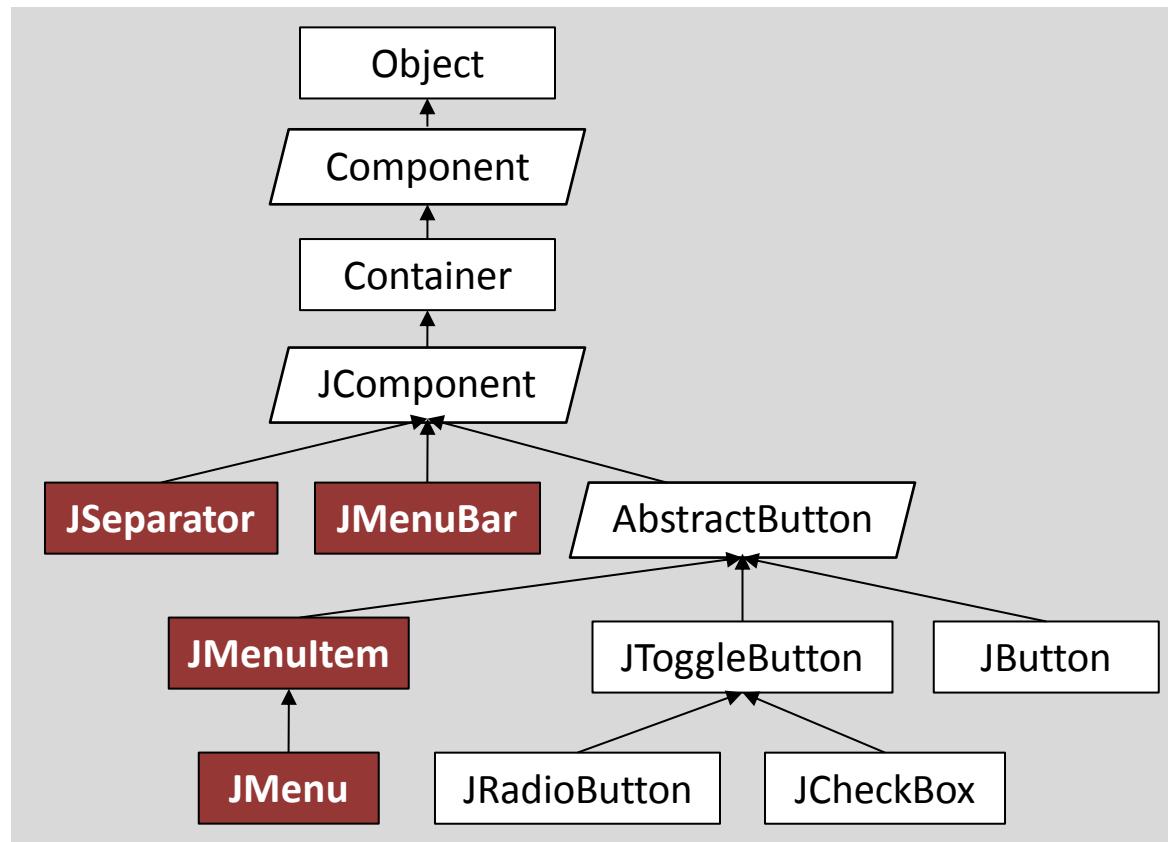
- **Abre menu**
- Torna **visível**



- Classes de Componentes

- JMenuBar // Barra de Menus
- JMenu // Menus e Submenus
- JMenuItem // Itens de Menus
- JSeparator // Separador de itens de menu // ou método `addSeparator()` de JMenu

- Hierarquia



## ■ Package

- javax.swing

## ■ Declaração

```
public class JMenuItem extends AbstractButton  
    implements Accessible, MenuElement { ... }
```

## ■ Construtores

Declaração	Funcionalidade
<code>public JMenuItem( String text )</code> <ul style="list-style-type: none"><li>▪ Ex: JMenuItem mitAcerca = new JMenuItem("Acerca");</li></ul>	Constrói item de menu com texto especificado.
<code>public JMenuItem( String text, int mnemonic )</code> <ul style="list-style-type: none"><li>▪ Ex: JMenuItem mitAcerca = new JMenuItem("Acerca", 'A');</li></ul>	Constrói item de menu com texto e mnemónica do teclado especificados.
<code>public JMenuItem(String text, Icon icon )</code> <ul style="list-style-type: none"><li>▪ Ex: Icon ic1 = new ImageIcon("acerca.jpg");</li><li>JMenuItem mitAcerca = new JMenuItem("Acerca", ic1);</li></ul>	Constrói item de menu com texto e ícone especificados.
<code>public JMenuItem()</code> <ul style="list-style-type: none"><li>▪ Ex: JMenuItem mitSair = new JMenuItem();</li></ul>	Constrói item de menu sem texto e sem ícone.

## ■ Métodos

- Herdados
  - [Component](#)
  - [Container](#)
  - [JComponent](#)
  - [AbstractButton](#)
- Próprios (mais usados)

Declaração	Funcionalidade
<pre>public void setAccelerator(KeyStroke keyStroke)</pre> <ul style="list-style-type: none"><li>■ Exemplos:<pre>JMenuItem mitAcerca = new JMenuItem("Acerca", 'A'); mitAcerca.setAccelerator( KeyStroke.getKeyStroke("ctrl A") ); ... mitAcerca.setAccelerator( KeyStroke.getKeyStroke("alt A") ); ... mitAcerca.setAccelerator( KeyStroke.getKeyStroke('A') );</pre></li></ul>	Especifica o acelerador do item de menu; Acelerador é uma tecla ou combinação de teclas que aciona o item de menu sem necessidade de navegar pelo sistema de menus.
<pre>public void setEnabled(boolean b)</pre> <ul style="list-style-type: none"><li>■ Ex: mitAcerca.setEnabled(false);</li></ul>	Inibe/desinibe item de menu.

- Package

- javax.swing

- Declaração

```
public class JMenu extends JMenuItem  
    implements Accessible, MenuElement { ... }
```

- Construtores

Declaração	Funcionalidade
<pre>public JMenu( String text )      ■ Exemplo:      JMenu mnuAjuda = new JMenu("Ajuda");</pre>	Constrói menu com texto especificado
<pre>public JMenu( )      ■ Exemplo:      JMenu mitSair = new JMenu();</pre>	Constrói menu sem texto

## Métodos

### Herdados

- [Component](#)
- [Container](#)
- [JComponent](#)
- [AbstractButton](#)
- [JMenuItem](#)

### Próprios

Declaração	Funcionalidade
<pre>public JMenuItem add(JMenuItem menulitem )</pre> <ul style="list-style-type: none"><li>▪ Exemplo: <pre> JMenuItem mitAcerca = new JMenuItem("Acerca"); JMenu mnuAjuda = new JMenu("Ajuda"); mnuAjuda.add( mitAcerca );</pre></li></ul>	<p>Adiciona item de menu no fim do menu;</p> <p>Retorna o item adicionado.</p>
<pre>public void addSeparator()</pre> <ul style="list-style-type: none"><li>▪ Exemplo: <pre>mnuAjuda.addSeparator();</pre></li></ul>	Adiciona separador no fim do menu.

## ■ Package

- javax.swing

## ■ Declaração

```
public class JMenuBar extends JComponent  
    implements Accessible, MenuElement { ... }
```

## ■ Construtor

Declaração	Funcionalidade
<pre>public JMenuBar()</pre> <ul style="list-style-type: none"><li>■ Exemplo: <code>JMenuBar mb = new JMenuBar();</code></li></ul>	Constrói barra de menus.

## Métodos

- Herdados
  - [Component](#)
  - [Container](#)
  - [JComponent](#)
- Próprio
  - Mais usado

Declaração	Funcionalidade
<pre>public JMenu add(JMenu c)</pre> <ul style="list-style-type: none"><li>▪ Exemplo<pre>JMenuBar mb = new JMenuBar(); JMenu mnuAjuda = new JMenu("Ajuda"); mb.add( mnuAjuda );</pre></li></ul>	Insere menu especificado no <b>fim</b> da barra de menus.

## ■ Exemplo de Implementação (1/3)

```

import java.awt.event.*;
import javax.swing.*;
public class DemoMenus extends JFrame {
    private static final int JANELA_LARGURA = 250, JANELA_ALTURA = 200;
    public DemoMenus() {
        super("Demo Menus");

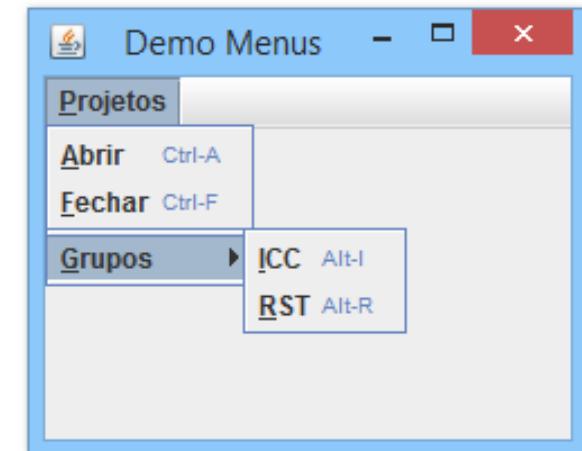
        JMenuBar menuBar = new JMenuBar();
        setJMenuBar(menuBar);
        menuBar.add(criarMenuProjetos());

        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(JANELA_LARGURA, JANELA_ALTURA);
        setVisible(true);
    }

    private JMenu criarMenuProjetos(){
        JMenu menu = new JMenu("Projetos");
        menu.setMnemonic(KeyEvent.VK_P);
        menu.add(criarItemAbrir());
        menu.add(criarItemFehar());
        menu.addSeparator();
        menu.add(criarSubMenuGrupos());
        return menu;
    }

    private JMenuItem criarItemAbrir() {...11 lines}
    private JMenuItem criarItemFehar() {...11 lines}
    private JMenu criarSubMenuGrupos() {...7 lines}
    private JMenuItem criarItemICC() {...11 lines}
    private JMenuItem criarItemRST() {...11 lines}
    public static void main(String[] args) {...3 lines}
}

```



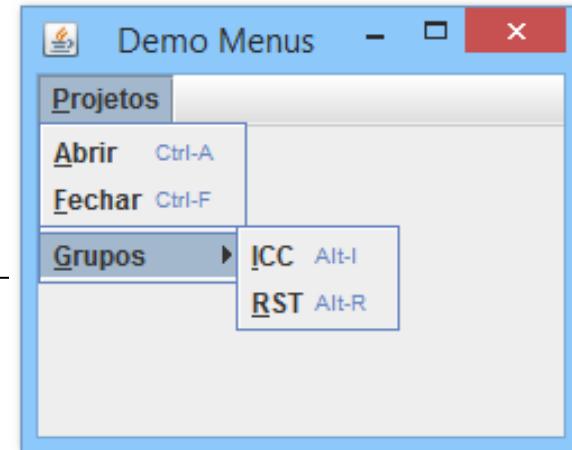
Criado item **Abrir** e adicionado ao Menu **Projetos**, abaixo

Separador de itens

Criado submenu **Grupos** e adicionado no fim do Menu **Projetos** ⇒ Submenu

## ■ Exemplo de Implementação (2/3)

```
private JMenuItem criarItemAbrir() {  
    JMenuItem item = new JMenuItem("Abrir",'A');  
    item.setAccelerator(KeyStroke.getKeyStroke("ctrl A"));  
    item.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            JOptionPane.showMessageDialog(DemoMenus.this,"Selecionado Item Abrir");  
        }  
    });  
    return item;  
}  
  
private JMenuItem criarItemFechar() {  
    JMenuItem item = new JMenuItem("Fechar",'F');  
    item.setAccelerator(KeyStroke.getKeyStroke("ctrl F"));  
    item.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            JOptionPane.showMessageDialog(DemoMenus.this,"Selecionado Item Fechar");  
        }  
    });  
    return item;  
}
```



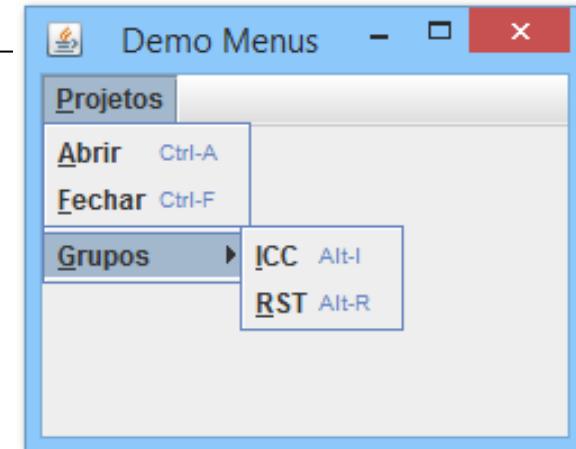
Método de  
Evento Action  
do item **Abrir**

Método de  
Evento Action  
do item **Fechar**

## ■ Exemplo de Implementação (3/3)

```
private JMenu criarSubMenuGrupos() {  
    JMenu menu = new JMenu("Grupos");  
    menu.setMnemonic(KeyEvent.VK_G);  
    menu.add(criarItemICC());  
    menu.add(criarItemRST());  
    return menu;  
}  
  
private JMenuItem criarItemICC() {  
    JMenuItem item = new JMenuItem("ICC", 'I');  
    item.setAccelerator(KeyStroke.getKeyStroke("alt I"));  
    item.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            JOptionPane.showMessageDialog(DemoMenus.this,"Selecionado Item ICC");  
        }  
    });  
    return item;  
}  
  
private JMenuItem criarItemRST() {  
    JMenuItem item = new JMenuItem("RST", 'R');  
    item.setAccelerator(KeyStroke.getKeyStroke("alt R"));  
    item.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            JOptionPane.showMessageDialog(DemoMenus.this,"Selecionado Item RST");  
        }  
    });  
    return item;  
}
```

Submenu:  
Menu adicionado  
a outro menu

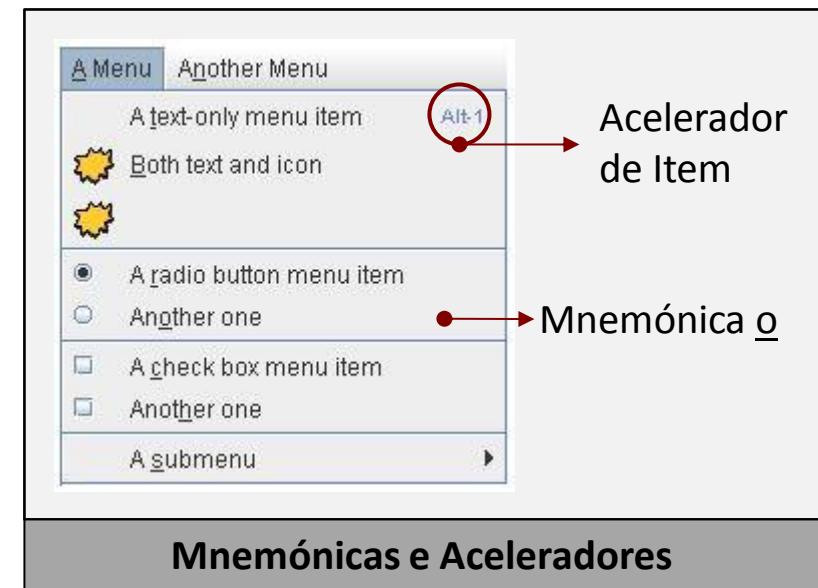


Item de Submenu

- **Menus suportam Acesso pelo Teclado**
  - Aumenta acessibilidade das aplicações
- **Tipos de Acesso pelo Teclado**
  - Mnemónicas
  - Aceleradores

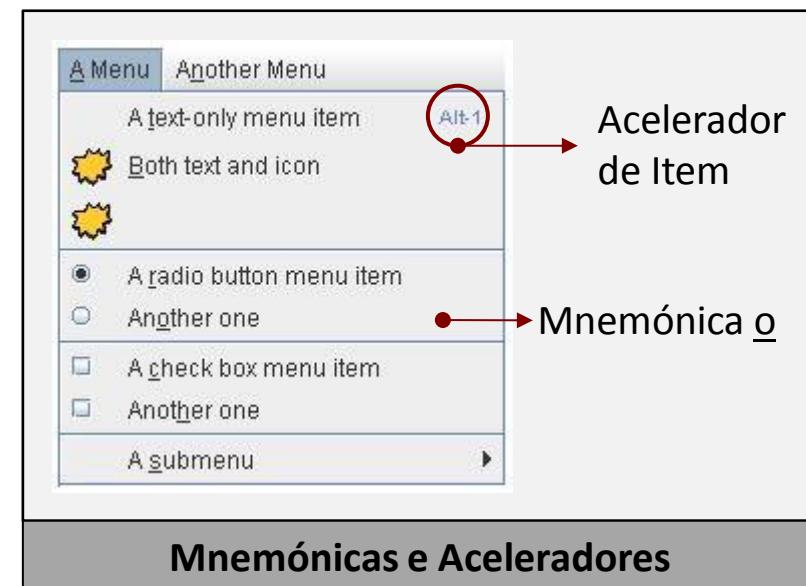
## Mnemónicas

- Teclas
  - Permitem navegar pela hierarquia de menus
- Identificadas
  - Por carácter sublinhado no nome
    - Menu
    - Item de menu
- Usadas
  - Simultaneamente com tecla ALT
    - ALT + Mnemónica
  - Exemplos (figura)
    - ALT + A // abre menu
    - ALT + s // abre submenu
  - Itens de menu
    - Têm de estar visíveis



## Aceleradores

- Teclas de atalho
  - Para **itens** de menu
  - Menus podem estar **fechados**
- Acesso + rápido
  - Evitam **navegação** pela hierarquia de menus
- Tipos
  - Tecla simples
    - Exemplo: x
  - Combinação de teclas
    - Exemplos
      - Alt+1 // figura
      - CTRL+X



## ■ Implementação de Mnemónica

- Tem de ser carater do nome de Menu, Item ou Submenu
- Formas

- **JMenu**

```
public void setMnemonic( char mnemonic )           // sublinha 1º carater  
public void setDisplayedMnemonicIndex(int index)    // índice a partir de 0
```

- Exemplo

```
JMenu menuEditar = new JMenu("Editar");  
menuEditar.setMnemonic('E');                      // pelicas = char; ALT+E  
menuEditar.setDisplayedMnemonicIndex(0);            // ALT + E
```

- **JMenuItem**

- Métodos semelhantes aos anteriores e um construtor:

```
public JMenuItem(String text, int mnemonic)          // construtor
```

- Exemplo **Item** de Menu

```
JMenuItem menutem = new JMenuItem("Cut", 'C');      // C ou ALT+C  
ou
```

```
JMenuItem menutem = new JMenuItem("Cut");  
menutem.setMnemonic('C');
```

## ■ Implementação de Acelerador

- Usar método `setAccelerator`

```
public void setAccelerator( KeyStroke keyStroke )
```

- Exemplos

```
JMenuItem itemCortar = new JMenuItem("Cortar");  
  
itemCortar.setAccelerator( KeyStroke.getKeyStroke("ctrl X") );           // combinação CTRL+X  
  
itemCortar.setAccelerator( KeyStroke.getKeyStroke("alt X") );           // combinação ALT+X  
  
itemCortar.setAccelerator( KeyStroke.getKeyStroke('x') );                // x
```

## ■ Classe `Keystroke`

- Cada instância representa uma tecla
- Método

```
static Keystroke getKeyStroke( int keyCode )
```

- Cria instância KeyStroke
  - Encapsula tecla premida correspondente a evento KEY\_PRESSED
- Keycode
  - Representa código virtual duma tecla

Tipos de Evento que Menu/Item podem Gerar	Evento
Action	Clique em menu/item
Container	Componente adicionado a menu/item Componente removido de menu/item
Component	Menu/item escondido Menu/item mostrado Menu/item movido Menu/item redimensionado
Focus	Menu/item adquire o foco Menu/item perde o foco
Item	Selecionado/desselecionado item
Key	Tecla mantida premida Tecla libertada Tecla premida (toque)
Mouse	Clique no rato Rato entrou no menu/item Rato saiu do menu/item Botão do rato premido Botão do rato libertado Rato movido Rato arrastado (premido + movido) Roda do rato movida

## ■ Evento Action

- Iniciado por seleção de **Menu** ou **Item** de Menu através de
  - Clique
  - Teclado
- Tratamento
  - Realizado
    - Por objetos de classes que implementem o interface ActionListener
  - Exemplo

```
JMenuItem itemSair = new JMenuItem("Sair");
itemSair.addActionListener(new ActionListener() {           // classe (interna) anónima
    public void actionPerformed( ActionEvent e ) {
        System.exit(0);                                // ou dispose()
    }
});
```

- <http://download.oracle.com/javase/tutorial/uiswing/components/menu.html>
  
- <http://www.faqs.org/docs/javap/c7/s5.html>

- [Controlos Básicos](#)
  - [Hierarquia de Classes](#)
  - [Superclasses](#)
    - [AbstractButton](#)
    - [JToggleButton](#)
  - [Botão de Comando](#)
  - [Entrada de Texto](#)
  - [Menu](#)
  - [Caixa de Listagem](#)
  - [Caixa de Combinação](#)
  - [Caixa de Verificação](#)
  - [Botão de Opção](#)



## ■ Implementar Listas de Itens

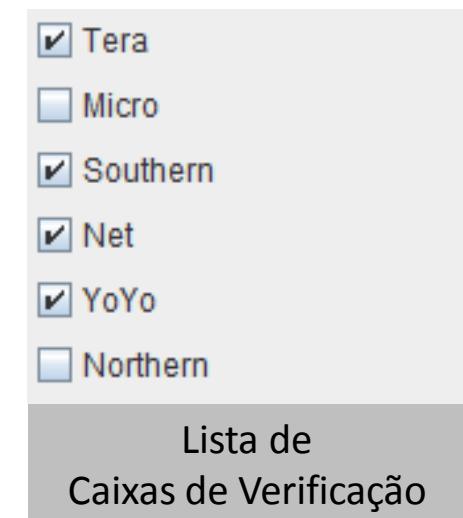
- Para utilizador escolher
  - Um item
  - Múltiplos itens
- Apresentados
  - Uma coluna
  - Múltiplas colunas

## ■ Itens de Lista

- Colocados numa simples caixa
- Objetos arbitrários
- Exemplos
  - Strings
  - Caixas de verificação (JCheckBox)

## ■ Alternativa à Caixa de Combinação

- Mais complexa
  - Tem mais capacidades



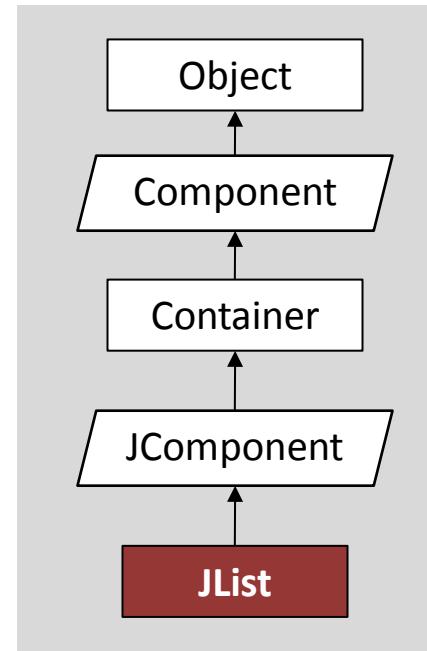
JComboBox

- Package

- javax.swing

- Declaração

```
public class JList<E> extends JComponent  
    implements Scrollable, Accessible { ... }
```



- Tipo E
  - Especifica tipo de item da caixa de listagem

## ■ Classe JList

- Usa a arquitetura Model-View-Controller
  - Para separar
    - Vista dos dados // View
    - Dados subjacentes // Model
      - Coleções de objetos
- Responsável apenas pela vista dos dados
  - Sabe muito pouco sobre a forma como dados estão armazenados
  - Não fornece métodos para inserir e remover itens da lista
- Dados
  - Guardados em objetos de classes que implementam interface ListModel
  - Exemplo
    - Classe DefaultListModel
      - Permite construir um objeto modelo da lista
      - Guarda dados num objeto da classe Vector



```
DefaultListModel lstModel = new DefaultListModel();  
  
lstModel.addElement("Ana"); // adiciona elemento no final da lista  
  
JList lstNomes = new JList( lstModel );  
  
lstModel.removeElement("Ana");
```

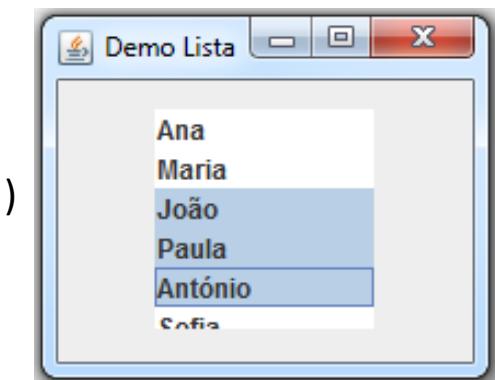
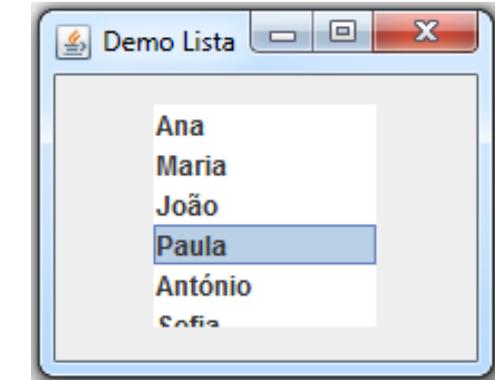
Declaração	Funcionalidade
<pre>public JList()</pre> <ul style="list-style-type: none"><li>▪ Ex: JList lstNomes = new JList();</li></ul>	Constrói caixa de listagem com modelo de dados vazio e que apenas pode ser lido
<pre>public JList(final E[] listData)</pre>	Constrói caixa de listagem que mostra elementos do <u>array</u> especificado;  Cria um modelo de dados carregado com os elementos do array e que apenas pode ser lido;  O modelo referencia o array especificado.
<pre>public JList(ListModel&lt;E&gt; dataModel)</pre>	Constrói caixa de listagem que mostra os elementos do modelo de dados especificado.

- Por omissão

- Mostra 8 itens

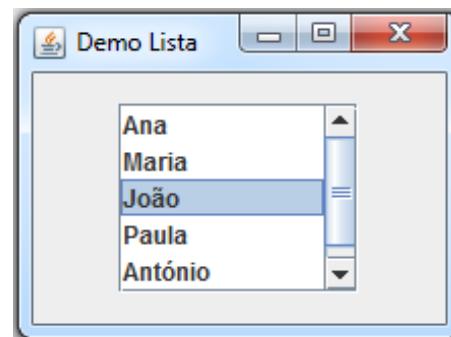
- Itens

- Podem ser **selecionados** por cliques
  - Seleção pode ser
    - Múltipla // por omissão
    - Simples // através do método
      - `setSelectionMode( ListSelectionModel.SINGLE_SELECTION )`



- Visualização de listas grandes

- Requer painel [JScrollPane](#)



JList + JScrollPane

- **Herdados**

- [Component](#)
- [Container](#)
- [JComponent](#)

- **Próprios (1/2)**

- Mais usados

Declaração	Funcionalidade
<pre>public void setSelectionMode(int selectionMode)</pre> <ul style="list-style-type: none"><li>▪ Exemplo</li></ul> <pre>String[ ] itens = {"Ana", "Maria", "João"}; JList lst = new JList(itens); lst.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);</pre>	Especifica modo de seleção de itens <ul style="list-style-type: none"><li>▪ SINGLE_SELECTION</li><li>▪ SINGLE_INTERVAL_SELECTION</li><li>▪ MULTIPLE_INTERVAL_SELECTION</li></ul>

## ■ Próprios (2/2)

Declaração	Funcionalidade
<code>public int getSelectedIndex()</code>	Retorna índice do item selecionado / menor índice no modo multi-seleção; Retorna -1, sem item selecionado.
<code>public int[] getSelectedIndices()</code>	Retorna array com todos os índices selecionados, ordenados de modo ascendente; Retorna array vazio, sem item selecionado.
<code>public E getSelectedValue()</code>	Retorna item selecionado / com menor índice no modo multi-seleção; Retorna null, sem item selecionado.
<code>public List&lt;E&gt; getSelectedValuesList()</code>	Retorna lista de itens selecionados, ordenados de modo ascendente dos seus índices; Retorna lista vazia, sem item selecionado.
<code>public void setVisibleRowCount(int visibleRowCount)</code>	Especifica o nº de linhas apresentada sem recorrer ao <i>scrolling</i> .

## ■ Exemplo

- Lista de strings de tamanho fixo com painel de deslocamento (ScrollPane)

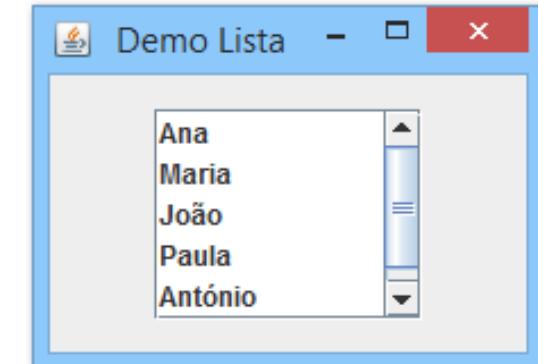
```
import java.awt.*;
import javax.swing.border.*;
import javax.swing.*;

public class DemoLista extends JFrame {
    private static final int JANELA_LARGURA = 250, JANELA_ALTURA = 200;
    public DemoLista() {
        super("Demo Lista");
        JPanel p = criarPainelLista();
        add(p);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(250, 200);
        setVisible(true);
    }

    private JPanel criarPainelLista(){
        String[] itens = {"Ana", "Maria", "João", "Paula", "António", "Sofia"};
        JList lstNomes = new JList(itens);
        lstNomes.setVisibleRowCount(5); // nº de itens visiveis
        final int LISTA_LARGURA = 100, LISTA_ALTURA=100;
        lstNomes.setPreferredSize(new Dimension(LISTA_LARGURA,LISTA_ALTURA));
        JScrollPane scrLstNomes = new JScrollPane(lstNomes);

        JPanel p = new JPanel();
        final int MARGEM_SUPERIOR = 10, MARGEM_INFERIOR = 10;
        final int MARGEM_ESQUERDA = 10, MARGEM_DIREITA = 10;
        p.setBorder(new EmptyBorder(MARGEM_SUPERIOR, MARGEM_ESQUERDA,
            MARGEM_INFERIOR, MARGEM_DIREITA));
        p.add(scrLstNomes);
        return p;
    }

    public static void main(String[] args) {...3 lines}
}
```



**ScrollPane**  
Para visualizar toda a lista

- <http://download.oracle.com/javase/tutorial/uiswing/components/list.html>

- [Controlos Básicos](#)
  - [Hierarquia de Classes](#)
  - [Superclasses](#)
    - [AbstractButton](#)
    - [JToggleButton](#)
  - [Botão de Comando](#)
  - [Entrada de Texto](#)
  - [Menu](#)
  - [Caixa de Listagem](#)
  - [Caixa de Combinação](#)
  - [Caixa de Verificação](#)
  - [Botão de Opção](#)



## ■ Caixa de Combinação

- É um componente gráfico
- Permite ao utilizador
  - Selecionar uma opção
- Formas
  - Não-Editável // por omissão
  - Editável

The diagram shows two parts of a dropdown menu. Part (1) shows a small button labeled "Audi" with a dropdown arrow. Part (2) shows a larger window listing car brands: Audi, BMW, Ferrari, Fiat, Ford, Mazda, Nissan, and Opel. A red bracket on the right side of part (2) groups the "Botão" (button) and the "Lista" (list) together.

(1) (2)

**Caixa de Combinação Não-Editável**  
(1 - Lista Escondida ; 2 - Lista Visível)

## ■ Caixa de Combinação Não-Editável

- Combina os elementos
  - Botão
  - Lista

The diagram shows two parts of an editable dropdown menu. On the left, a text input field contains "Toyota" with a dropdown arrow. On the right, a larger window lists car brands: Audi, BMW, Ferrari, Fiat, Ford, Mazda, Nissan, and Opel. A red bracket on the right side of the window groups the "Campo de Texto com Botão" (text field with button) and the "Lista" (list) together.

Campo de Texto com Botão

Lista

**Caixa de Combinação Editável**

## ■ Caixa de Combinação Editável

- Combina os elementos
  - Campo de Texto com Botão
  - Lista

## ■ Botão (Seta)

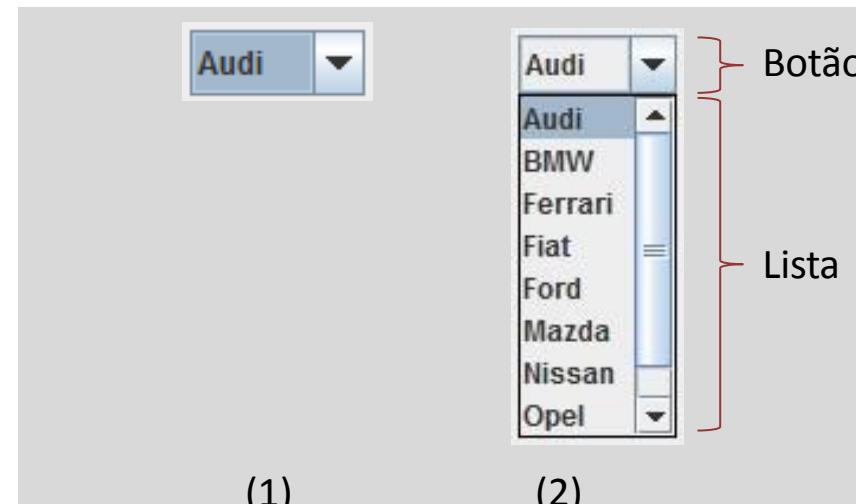
- Permite ao utilizador
  - Visualizar a Lista

## ■ Lista

- Permite ao utilizador
  - Selecionar uma opção

## ■ Campo de Texto

- Permite ao utilizador
  - Editar opção selecionada
- Interesse
  - Em listas grandes
  - Para seleção mais rápida
    - Opção escondida



## Caixa de Combinação Não-Editável

(1 - Lista Escondida ; 2 - Lista Visível)



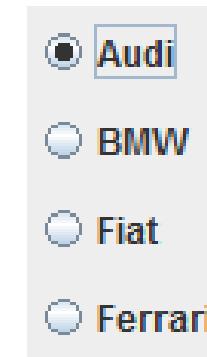
## Caixa de Combinação Editável

- Permitir ao Utilizador

- Escolher uma opção ... entre **nº considerável** de alternativas

- Vantagem

- Minimização do espaço GUI ocupado
- Alternativa
  - Com Botões de Opção
    - Ocupa muito mais espaço GUI

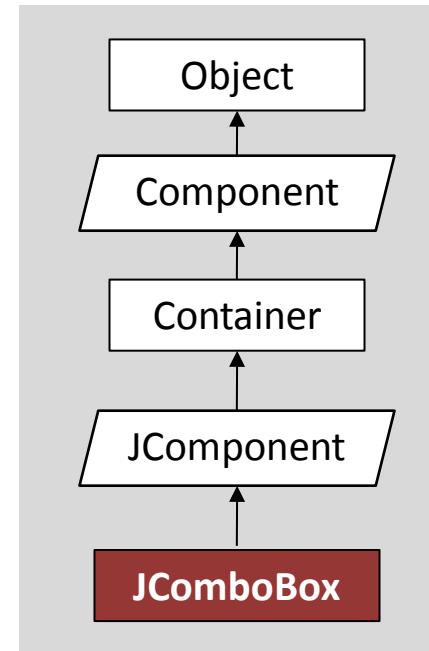


## ■ Package

- javax.swing

## ■ Declaração

```
public class JComboBox<E> extends JComponent  
    implements ItemSelectable, ListDataListener, ActionListener, Accessible { ... }
```



## ■ Tipo E

- Especifica tipo de opção da caixa de combinação

- Mais usados

Declaração	Funcionalidade
<pre data-bbox="88 267 456 308">public JComboBox()</pre> <ul style="list-style-type: none"> <li>■ Exemplos</li> </ul> <pre data-bbox="178 414 956 455">JComboBox cbMarcas = new JComboBox();</pre> <pre data-bbox="178 483 1100 524">JComboBox&lt;String&gt; cbMarcas = new JComboBox();</pre>	 <p>Constrói caixa de combinação <b>não-editável</b> com uma lista de opções vazia.</p>
<pre data-bbox="88 584 622 624">public JComboBox( E[ ] items)</pre> <ul style="list-style-type: none"> <li>■ Exemplo 1</li> </ul> <pre data-bbox="178 731 956 771">String[ ] marcas = {"Audi", "BMW", "Fiat"};</pre> <pre data-bbox="178 800 1100 840">JComboBox cbMarcas = new JComboBox( marcas );</pre> <ul style="list-style-type: none"> <li>■ Exemplo 2</li> </ul> <pre data-bbox="178 976 1100 1016">String[ ] nums= {"1","2","3","4","5","6","7","8","9"};</pre> <pre data-bbox="178 1045 1052 1085">JComboBox cbDemo = new JComboBox( nums );</pre>	 <p>Constrói caixa de combinação <b>não-editável</b> que mostra os elementos do <i>array</i> especificado;</p> <p>Por omissão:</p> <ul style="list-style-type: none"> <li>■ Selecionada 1º opção da lista;</li> <li>■ Comprimento máximo da lista = 8.</li> </ul> 

## ■ Herdados

- [Component](#)
- [Container](#)
- [JComponent](#)

## ■ Próprios (1/3)

- Mais usados

Declaração	Funcionalidade
<pre>public void setEditable(boolean aFlag)</pre> <ul style="list-style-type: none"><li>■ Exemplo – caixa de combinação editável<pre>JComboBox cb1 = new JComboBox(); cb1.setEditable(true);</pre></li><li>■ Exemplo – caixa de combinação não-editável<pre>JComboBox cb2 = new JComboBox(); cb2.setEditable(false);</pre></li></ul>	Especifica forma da caixa de combinação: <ul style="list-style-type: none"><li>■ Editável;</li><li>■ Não-Editável.</li></ul>
<pre>public void setMaximumRowCount(int count)</pre> <ul style="list-style-type: none"><li>■ Exemplo<pre>cb1.setMaximumRowCount(5);</pre></li></ul>	Especifica nº máximo de opções visíveis (comprimento máximo visível da lista); Se a quantidade total de opções exceder o valor especificado, a lista mostra uma <i>scrollbar</i> .

## ■ Próprios (2/3)

Declaração	Funcionalidade
<pre>public void addItem(E item)</pre> <ul style="list-style-type: none"><li>▪ Exemplo</li></ul> <pre>JComboBox cbMarcas = new JComboBox( ); cbMarcas.addItem("Audi");</pre>	Adiciona opção à lista.
<pre>public void insertItemAt(E item, int index)</pre> <ul style="list-style-type: none"><li>▪ Exemplo</li></ul> <pre>cbMarcas.insertItemAt("Alfa Romeo", 0);</pre>	Adiciona opção à lista, na posição index; índices a partir de 0.
<pre>public void removeItem(Object item)</pre> <ul style="list-style-type: none"><li>▪ Exemplo</li></ul> <pre>cbMarcas.removeItem("Audi");</pre>	Remove opção especificada.
<pre>public void removeItemAt(int index)</pre>	Remove opção na posição especificada.
<pre>public void removeAllItems()</pre>	Remove todas as opções da lista.

## ■ Próprios (3/3)

Declaração	Funcionalidade
<pre>public void setSelectedIndex(int index)</pre> <ul style="list-style-type: none"><li>▪ Exemplo</li></ul> <pre>cbMarcas.setSelectedIndex(2);</pre>	Seleciona opção com índice especificado; índice 0 seleciona 1ª opção da lista; índice -1 não seleciona opção. 
<pre>public Object getSelectedItem()</pre> <ul style="list-style-type: none"><li>▪ Exemplo</li></ul> <pre>Object obj = cbMarcas.getSelectedItem();</pre>	Retorna opção selecionada; Numa caixa de combinação editável pode retornar opção que não foi adicionada à lista. 

## ■ Exemplo

```
import javax.swing.*;
import java.awt.event.*;
import javax.swing.border.*;

public class DemoCaixaCombinada extends JFrame {
    private JComboBox<String> cbMarcas;
    private static final int JANELA_LARGURA = 300, JANELA_ALTURA = 200;

    public DemoCaixaCombinada() {
        super("Demo Caixa Combinada");
        JPanel p = criarPainelCaixaCombinacao();
        add(p);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(JANELA_LARGURA, JANELA_ALTURA);
        setVisible(true);
    }

    private JPanel criarPainelCaixaCombinacao() {
        String[] marcas = {"Audi", "BMW", "Fiat", "Ford", "Mazda", "Porche"};
        cbMarcas = new JComboBox(marcas);
        cbMarcas.setSelectedIndex(-1);
        cbMarcas.setMaximumRowCount(5);
        cbMarcas.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(DemoCaixaCombinada.this,
                    "Item selecionando: " + cbMarcas.getSelectedItem());
            }
        });
        JPanel p = new JPanel();
        final int MARGEM_SUPERIOR = 10, MARGEM_INFERIOR = 10;
        final int MARGEM_ESQUERDA = 0, MARGEM_DIREITA = 0;
        p.setBorder(new EmptyBorder(MARGEM_SUPERIOR, MARGEM_ESQUERDA,
            MARGEM_INFERIOR, MARGEM_DIREITA));
        p.add(cbMarcas);
        return p;
    }

    public static void main(String[] args) {...3 lines}
}
```



Tipos de Evento que Caixa de Combinação pode Gerar	Evento
Action	Selecionado item
Container	Componente adicionado à caixa de combinação
	Componente removido da caixa de combinação
Component	Caixa de combinação escondida
	Caixa de combinação mostrada
	Caixa de combinação movida
	Caixa de combinação redimensionada
Focus	Caixa de combinação adquire o foco de entrada
	Caixa de combinação perde o foco de entrada
Item	Selecionado/desselecionado item
Key	Tecla mantida premida
	Tecla libertada
	Tecla premida (toque)
Mouse	Clique no rato
	Rato entrou na caixa de combinação
	Rato saiu da caixa de combinação
	Botão do rato premido
	Botão do rato libertado
	Rato movido
	Rato arrastado (premido + movido)
	Roda do rato movida

## ■ Evento Action

- Iniciado por seleção de item
- Tratamento
  - Realizado
    - Objetos de classes que implementem o interface ActionListener
  - Exemplo

```
String[ ] marcas = {"Audi", "BMW", "Fiat"};  
JComboBox cbMarcas = new JComboBox( marcas );  
cbMarcas.addActionListener(new ActionListener() { // classe (interna) anónima  
    public void actionPerformed( ActionEvent e ) {  
        // Método de evento executado após seleção de item  
        ...  
    }  
});
```

- <http://docs.oracle.com/javase/tutorial/uiswing/components/combobox.html>

- [Controlos Básicos](#)
  - [Hierarquia de Classes](#)
  - [Superclasses](#)
    - [AbstractButton](#)
    - [JToggleButton](#)
  - [Botão de Comando](#)
  - [Entrada de Texto](#)
  - [Menu](#)
  - [Caixa de Listagem](#)
  - [Caixa de Combinação](#)
  - [Caixa de Verificação](#)
  - [Botão de Opção](#)



- **Bibliografia**

- <http://docs.oracle.com/javase/tutorial/uiswing/components/button.html#checkbox>

- [Controlos Básicos](#)
  - [Hierarquia de Classes](#)
  - [Superclasses](#)
    - [AbstractButton](#)
    - [JToggleButton](#)
  - [Botão de Comando](#)
  - [Entrada de Texto](#)
  - [Menu](#)
  - [Caixa de Listagem](#)
  - [Caixa de Combinação](#)
  - [Caixa de Verificação](#)
  - [Botão de Opção](#)



- **Bibliografia**

- <http://docs.oracle.com/javase/tutorial/uiswing/components/button.html#radiobutton>

- [Introdução](#)
- [Componentes Gráficos](#)
  - [Introdução](#)
  - [Hierarquia de Classes](#)
  - [Interfaces](#)
  - [Categorias](#)
    - [Contentores de Componentes Gráficos](#)
    - [Apresentação de Informação](#)
    - [Controlos Básicos](#)
  - [Gestores de Posicionamento](#)
  - [Manipuladores de Eventos](#)
  - [Bibliografia](#)
  - [Índice Remissivo](#)



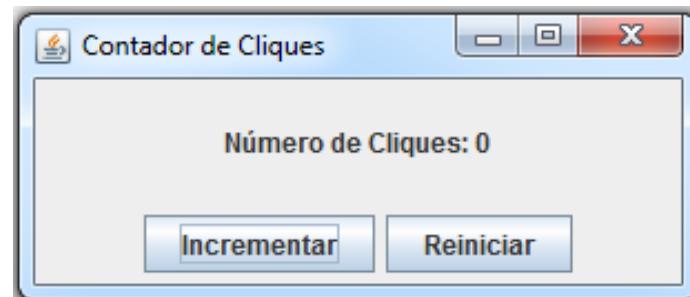
- [Noção](#)
- [Tipos](#)
- [Hierarquia de Classes](#)
- [Gestores de Posicionamento](#)
  - [BorderLayout](#)
  - [FlowLayout](#)
  - [GridLayout](#)
  - [CardLayout](#)
- [Combinação de Diferentes Tipos num GUI](#)

# Noção de Gestor de Posicionamento

- **Gestores de Posicionamento (*Layout Managers*)**

- Objetos que gerem dinamicamente a posição de componentes gráficos dentro dos contentores

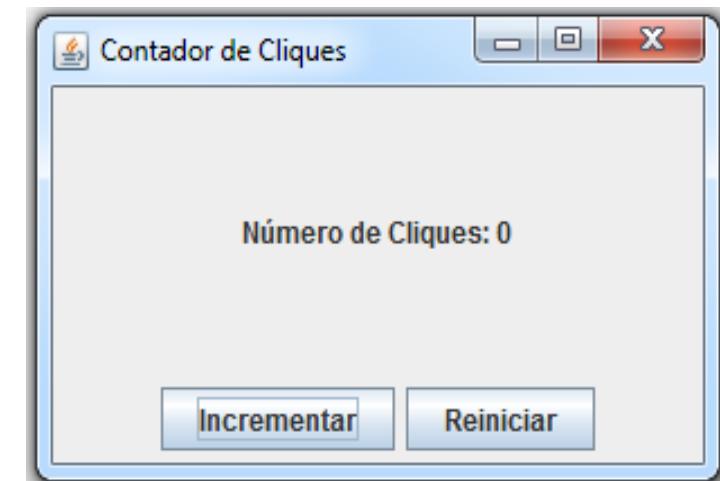
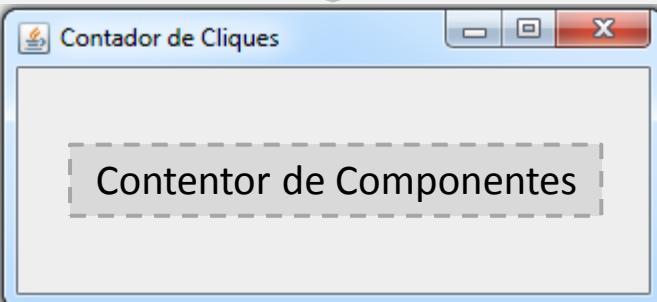
- **Exemplo**



Gestores de Posicionamento  
repositionam componentes durante  
redimensionamento da janela

Reiniciar Número de Cliques: 0 Incrementar

Gestores de Posicionamento  
colocam componentes no  
contentor



- Tipos de Gestores de Posicionamento

- Nativos
- Próprios // Definidos pelo Programador

- Nativos

- BorderLayout
- FlowLayout
- GridLayout
- CardLayout
- BoxLayout
- GridBagLayout
- GroupLayout
- SpringLayout



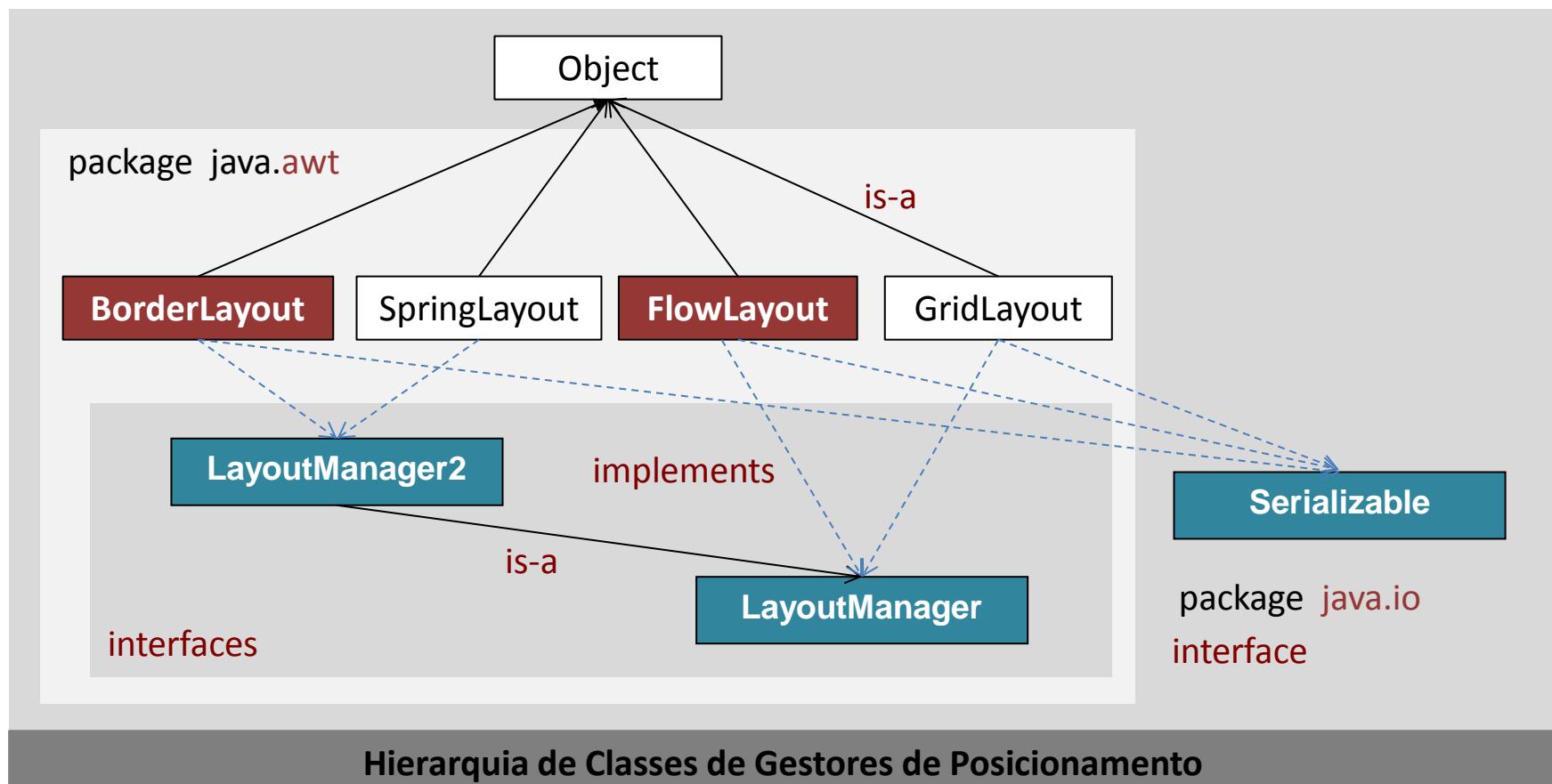
Abordados nestes slides

# Hierarquia de Classes de Gestores de Posicionamento

- Java Fornece uma Classe

- Para cada tipo de gestor

- Hierarquia de Classes



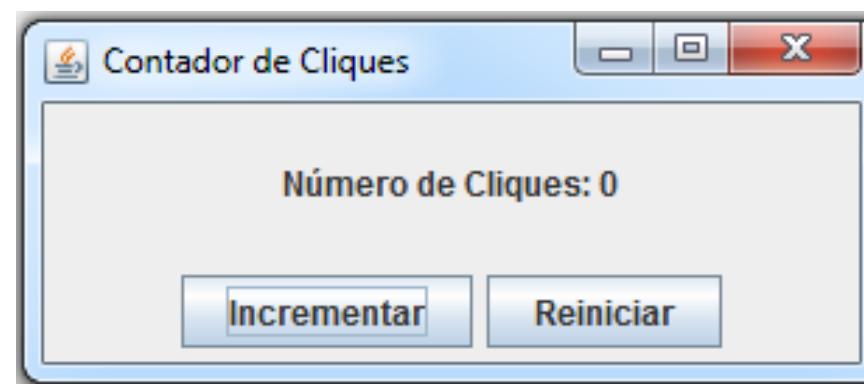
- Usados (por omissão)

- Content Pane de componentes JFrame

- Interesse

- Permitir colocação **precisa** de componentes gráficos num contentor
    - Numa posição específica

- Exemplo



- Colocar a Meio: Número de Cliques
  - Colocar no Fundo: Botões de Comando

## ■ Funcionamento (1/3)

- Dividem um contentor em 5 regiões
  - Norte
  - Oeste
  - Centro
  - Este
  - Sul
- Colocam automaticamente um componente gráfico na região
  - Indicada pelo programa
  - Centro // por omissão
- Múltiplos componentes gráficos adicionados na mesma região
  - Sobrepostos
  - Visível
    - Último adicionado



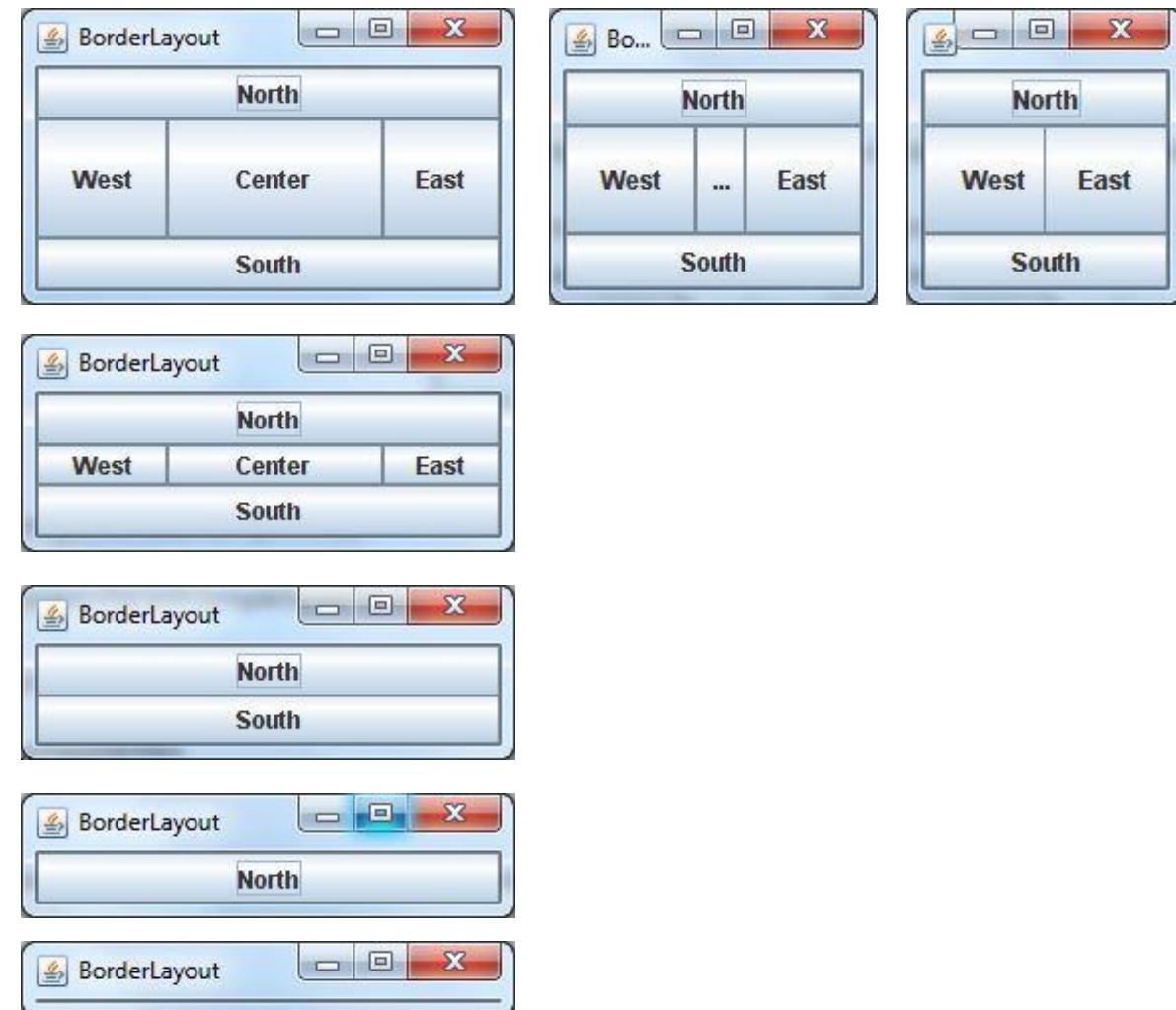
## ■ Funcionamento (2/3)

- Dimensões do componente adicionado
  - Norte e Sul
    - Altura
      - Preferida do componente
      - Modificável via método [setPreferredSize\(\)](#)
      - Para garantir visualização da alteração
        - Usar o método [revalidate\(\)](#)
    - Largura
      - Contentor
  - Oeste e Este
    - Largura
      - Preferida do componente
    - Altura
      - Altura Contentor – Altura Norte – Altura Sul
  - Centro
    - Preenche toda a região



## ■ Funcionamento (3/3)

- Redimensão do contentor
  - Em altura
    - Altera altura
      - Este
      - Centro
      - Oeste
  - Em largura
    - Altera largura
      - Norte
      - Centro
      - Sul

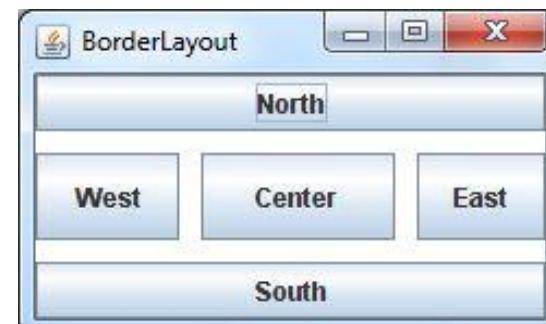


## ■ Configurações de Funcionamento (1/2)

- Espaço entre regiões contíguas
  - Por omissão
    - Nulo
    - Exemplo

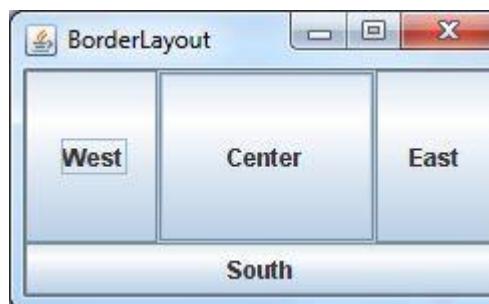
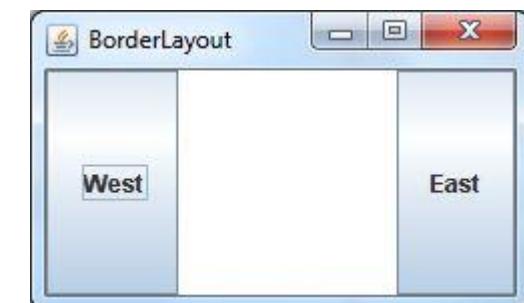
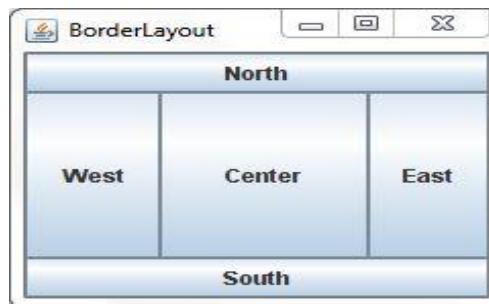


- Modificável pelo programa
  - Exemplo



## ■ Configurações de Funcionamento (2/2)

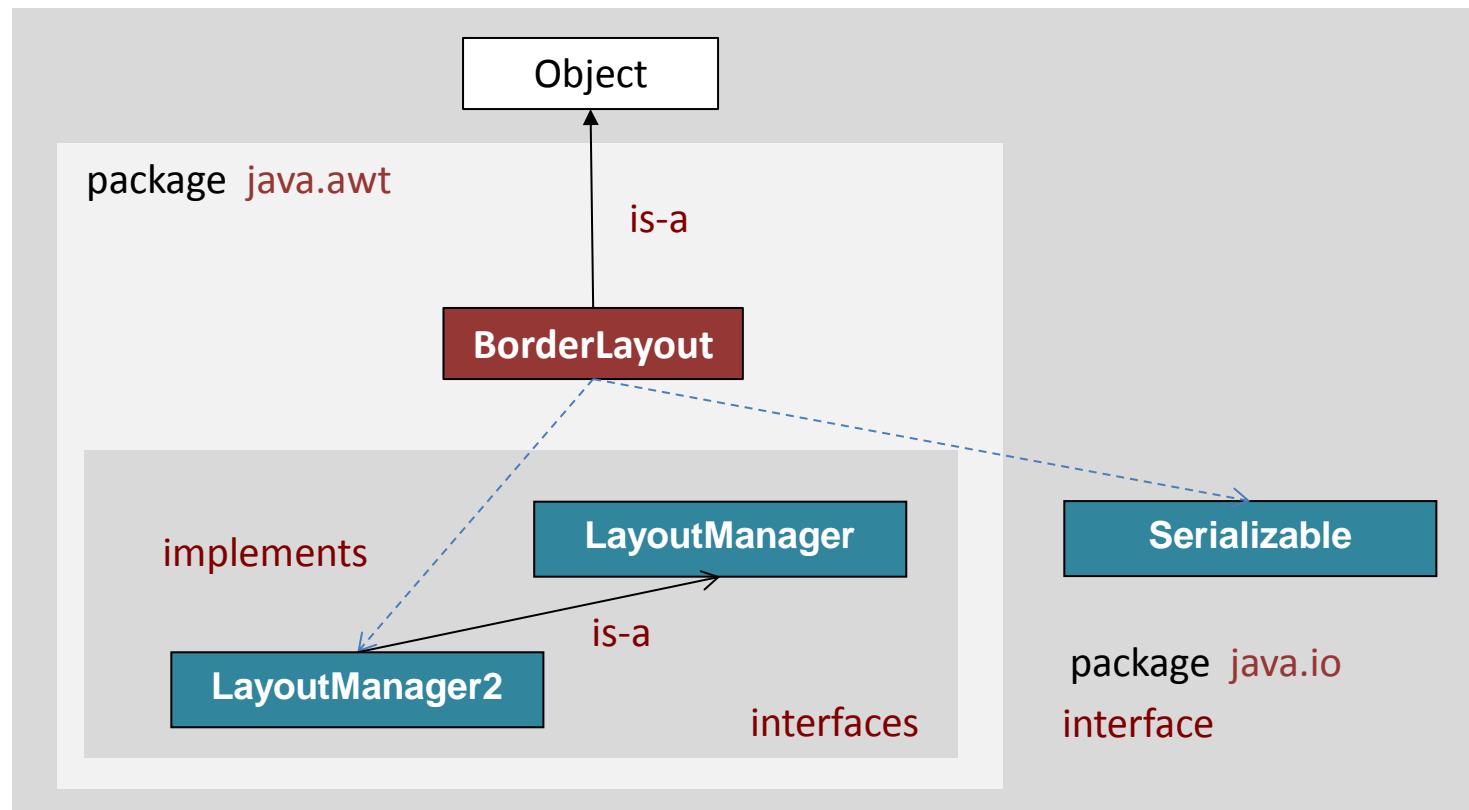
- Uso das regiões é opcional



## ■ Classe (1/4)

## ▪ Declaração

```
public class BorderLayout extends Object  
    implements LayoutManager2, Serializable { ... }
```



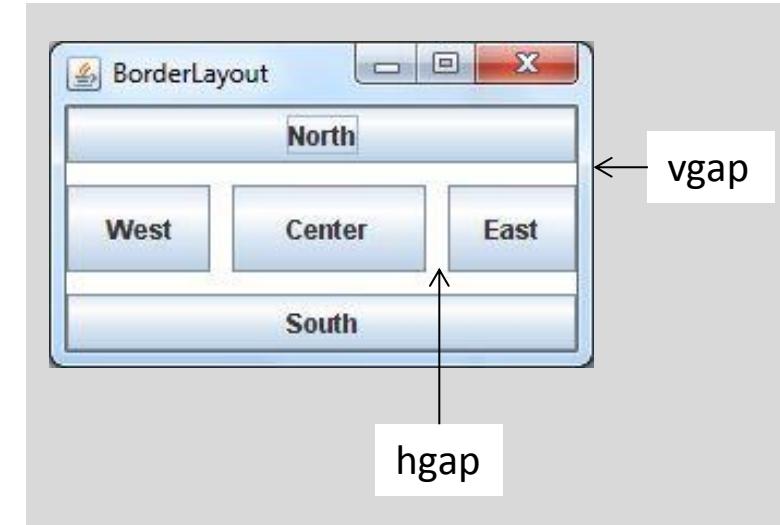
## ■ Classe (2/4)

- **Construtores** // constroem novos objetos gestores BorderLayout

Declaração	Funcionalidade
public BorderLayout()	Gestor posiciona componentes sem intervalos entre eles.
public BorderLayout(int hgap, int vgap)	Gestor posiciona componentes com intervalos entre eles. Intervalo horizontal = hgap pixels Intervalo vertical = vgap pixels

## ■ Exemplos de Uso

```
BorderLayout b1 = new BorderLayout();  
  
BorderLayout b2 = new BorderLayout( 20, 30 );
```



## ■ Classe (3/4)

## ■ Métodos de Instância // mais usados

Declaração	Funcionalidade
public void setHgap(int hgap)	Indica o intervalo horizontal entre componentes. Unidade de medida é o pixel.
public void setVgap(int vgap)	Indica o intervalo vertical entre componentes. Unidade de medida é o pixel.

## ■ Exemplos de Uso

```
BorderLayout b1 = new BorderLayout();  
  
b1.setVgap(10);  
  
b1.setHgap(20);
```

## ■ Classe (4/4)

## ■ Campos de Classe (mais usados)

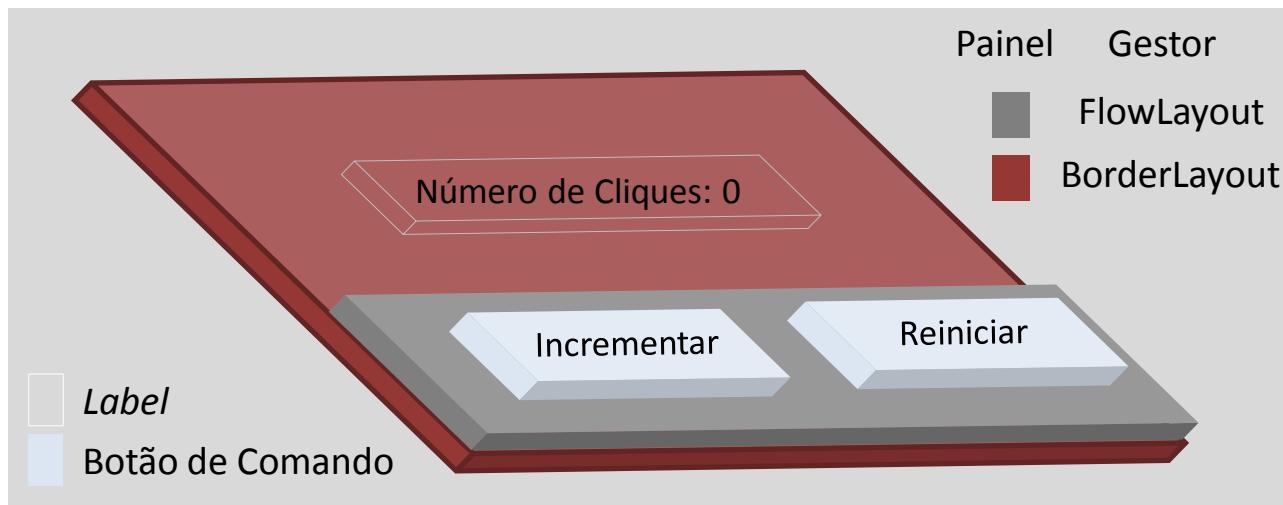
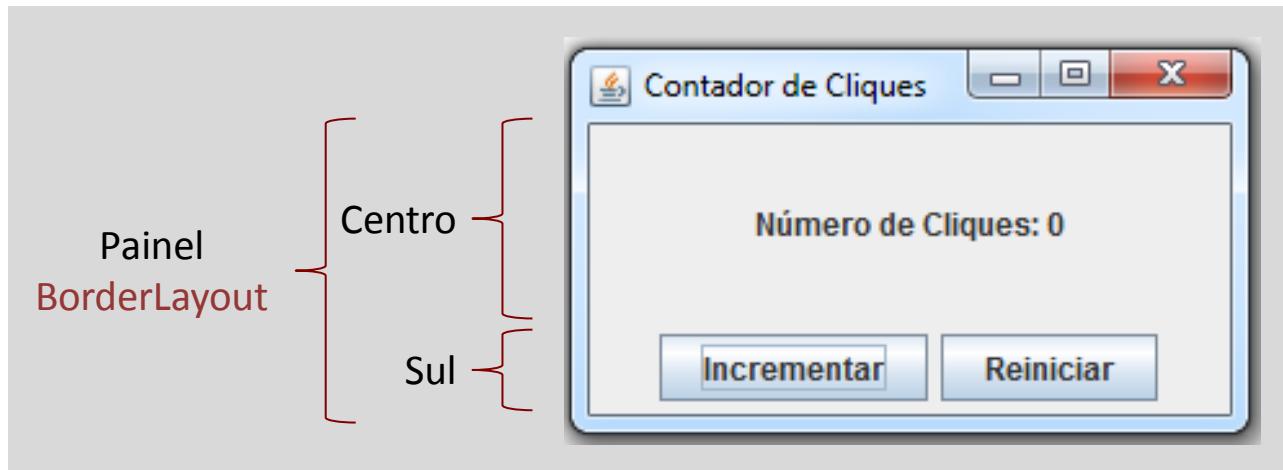
Declaração	Funcionalidade
public static final String NORTH = "North";	Valor indica região Norte
public static final String SOUTH = "South";	Valor indica região Sul
public static final String EAST = "East";	Valor indica região Este
public static final String WEST = "West";	Valor indica região Oeste
public static final String CENTER = "Center";	Valor indica região Centro

## ■ Exemplos de Uso

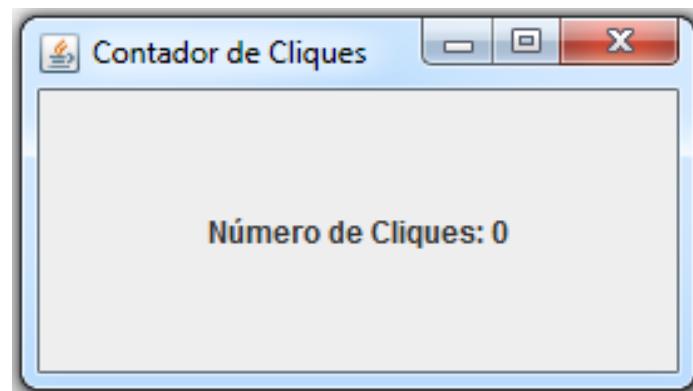
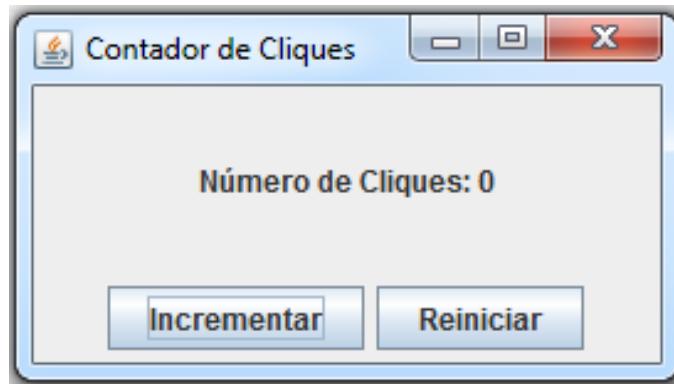
```
JPanel p = new JPanel( new BorderLayout() );  
  
JLabel lbl1 = new JLabel("Norte");  
  
JLabel lbl2 = new JLabel("Este");  
  
p.add( lbl1, BorderLayout.NORTH );  
  
p.add( lbl2, BorderLayout.EAST );
```

## ■ Exemplo – Contador de Cliques

- Posicionamento dos Componentes do GUI (*Layout*)



## ■ Exemplo



```
public class ContadorGUI extends JFrame {  
  
    private JLabel lblNumero;  
    private static String s = "Número de Cliques: ";  
    private static final int JANELA_LARGURA = 270;  
    private static final int JANELA_ALTURA = 150;  
  
    public ContadorGUI() {  
  
        super("Contador de Cliques");  
  
        lblNumero = new JLabel(s + "0");  
  
        add( lblNumero, BorderLayout.CENTER );  
  
        setDefaultCloseOperation(EXIT_ON_CLOSE);  
        setSize(JANELA_LARGURA , JANELA_ALTURA );  
        setLocationRelativeTo(null);  
        setVisible(true);  
    }  
}
```

JLabel

- Usados (por omissão)

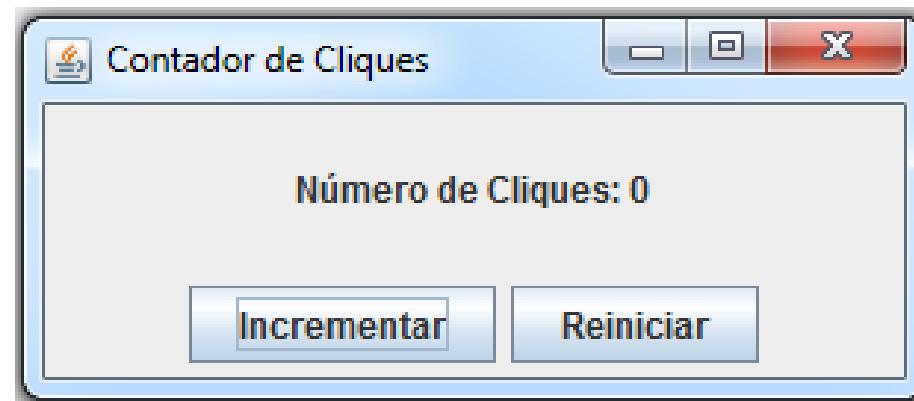
- Paineis (Componentes JPanel)

- Interesse

- Posicionar componentes gráficos num contentor
    - Em linha

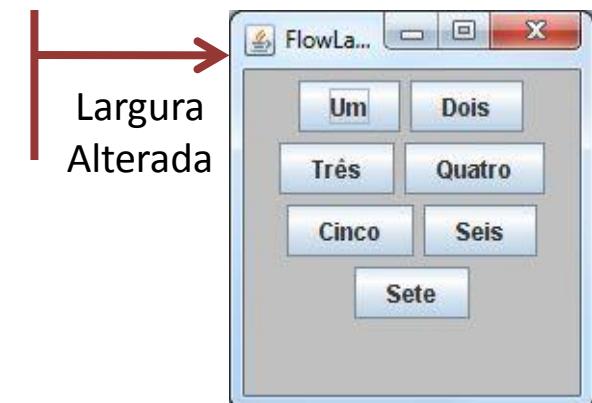
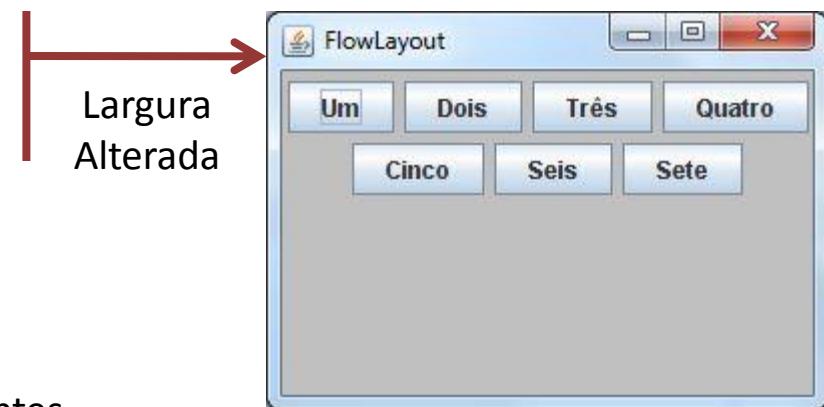
- Exemplo

- Alinhar botões de comando na horizontal



## ■ Funcionamento

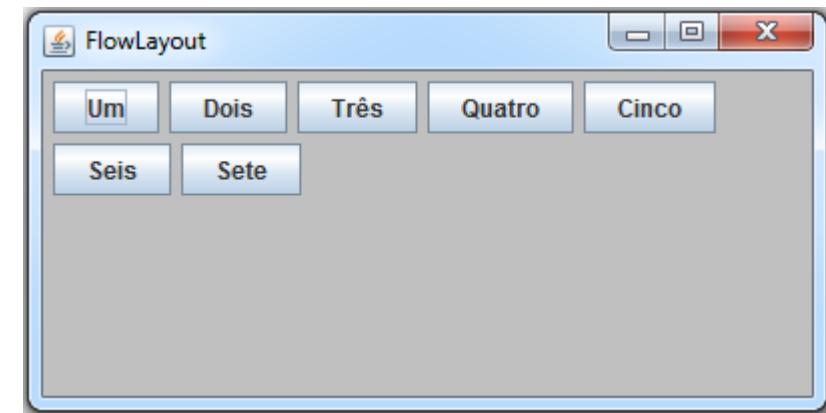
- Usam tamanhos **preferidos** dos componentes
  - Modificáveis via método [setPreferredSize\(\)](#)
  - Método [setSize\(...\)](#) não altera tamanho
  - Visualização garantida por método [revalidate\(\)](#)
- Colocam componentes em linhas consecutivas
  - Automaticamente
  - Por omissão
    - Linhas alinhadas ao centro
    - Intervalos entre componentes/bordo
      - 5 pixels
- Quando espaço é insuficiente numa linha
  - Tentam colocar os restantes componentes na linha seguinte
- Quando o contentor é **redimensionado**, os componentes
  - São **reposicionados** automaticamente de modo a preencherem o espaço disponível em linhas consecutivas
  - **Não são redimensionados**
- Controlam completamente a **posição** de cada componente
  - Programador não consegue definir
    - Posição **precisa** de um componente



## ■ Configurações de Funcionamento

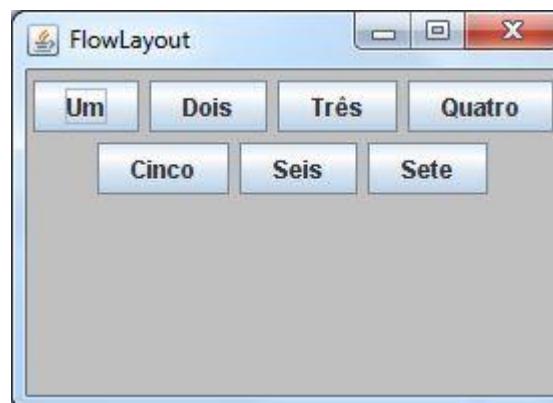
### ■ Alinhamento das linhas de componentes

- Esquerda
- Direita
- Centro



### ■ Intervalos entre componentes e entre componentes e bordo do contentor

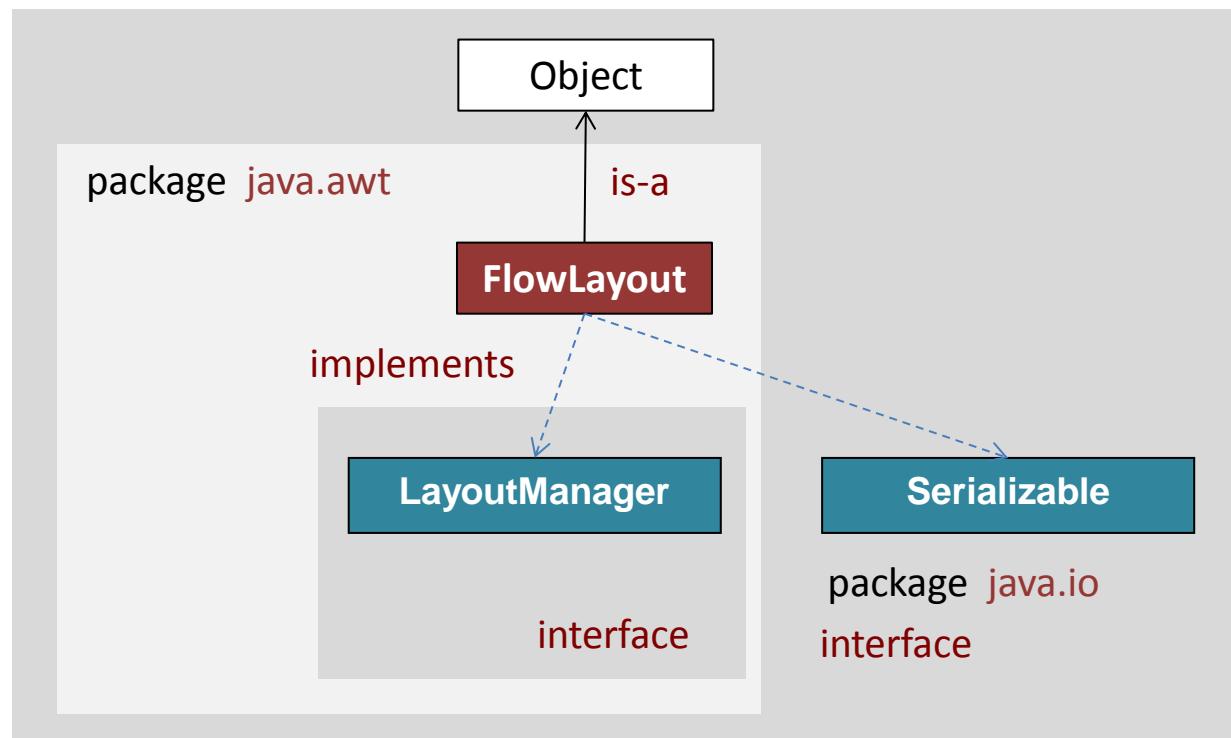
- Vertical
- Horizontal



## ▪ Classe (1/3)

## ▪ Declaração

```
public class FlowLayout extends Object  
    implements LayoutManager, Serializable { ... }
```



## ▪ Classe (1/3)

## ▪ Campos de Classe

Declaração	Funcionalidade
public static final int <b>LEFT</b> = 0;	Valor indica que cada linha de componentes deverá ser alinhada à esquerda.
public static final int <b>CENTER</b> = 1;	Valor indica que cada linha de componentes deverá ser alinhada ao centro.
public static final int <b>RIGHT</b> = 2;	Valor indica que cada linha de componentes deverá ser alinhada à direita.

## ▪ Exemplos de Uso

FlowLayout.LEFT

FlowLayout.CENTER

FlowLayout.RIGHT

## ▪ Classe (2/3)

- **Construtores** // constroem novos objetos gestores FlowLayout

Declaração	Funcionalidade
public FlowLayout()	Alinhamento de linhas centrado. Intervalos horizontais e verticais entre componentes e entre componentes e o bordo de 5 píxeis.
public FlowLayout(int align)	Alinhamento das linhas <i>align</i> . Intervalos horizontais e verticais de 5 píxeis.
public FlowLayout(int align, int hgap, int vgap)	Alinhamento das linhas <i>align</i> . Intervalos horizontais e verticais de <i>hgap</i> e <i>vgap</i> píxeis, respetivamente.

## ▪ Exemplos de Uso

```
FlowLayout f1 = new FlowLayout();  
  
FlowLayout f2 = new FlowLayout( FlowLayout.LEFT );  
  
FlowLayout f3 = new FlowLayout( FlowLayout.RIGHT );  
  
FlowLayout f4 = new FlowLayout( FlowLayout.RIGHT, 20, 30 );
```

## ■ Classe (3/3)

## ■ Métodos de Instância // mais usados

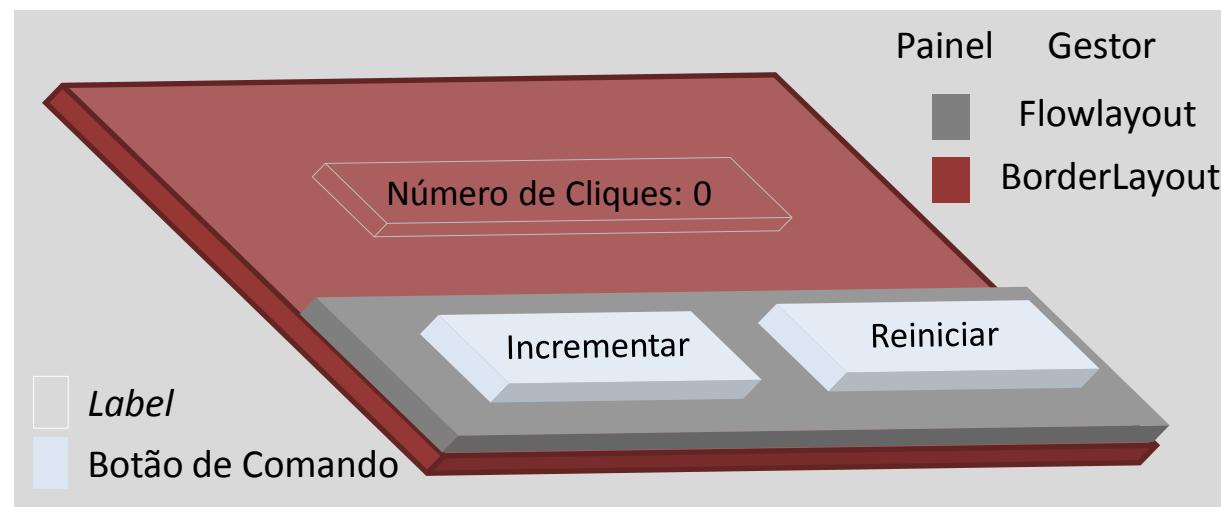
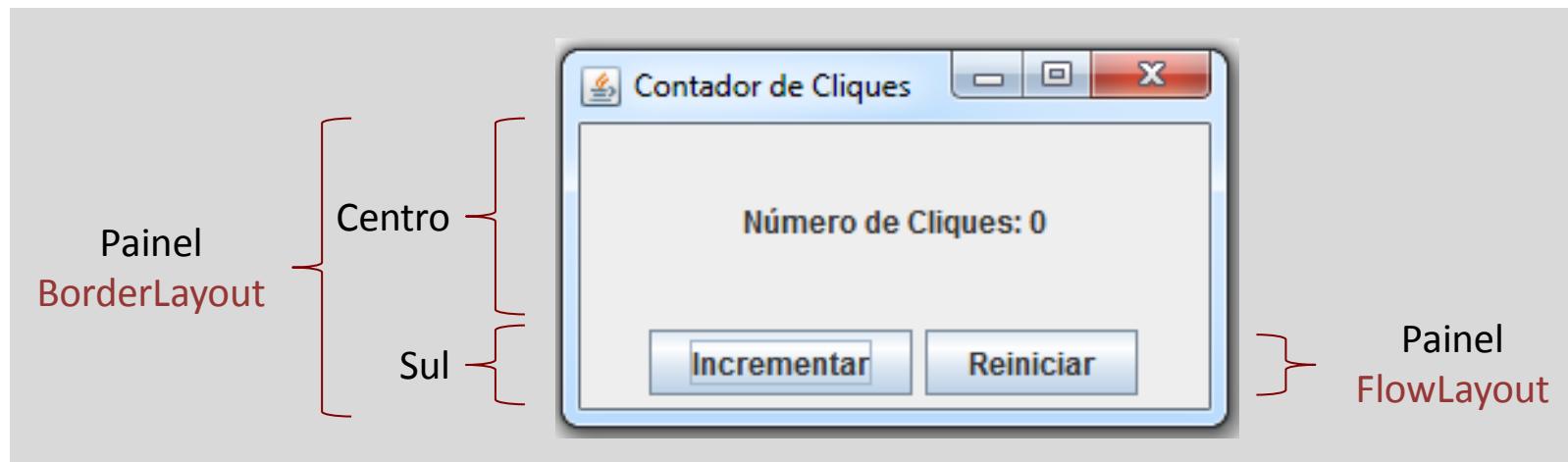
Declaração	Funcionalidade
public void <b>setAlignment</b> (int align)	Indica alinhamento align do gestor FlowLayout
public void <b>setHgap</b> (int hgap)	Indica intervalo horizontal entre componentes e entre componentes e bordo do contentor. Unidade de medida é o pixel.
public void <b>setVgap</b> (int vgap)	Indica intervalo vertical entre componentes e entre componentes e bordo do contentor. Unidade de medida é o pixel.

## ■ Exemplos de Uso

```
FlowLayout f1 = new FlowLayout();
f1.setAlignment( FlowLayout.LEFT );      // alinhamento especificado por campo de classe LEFT
f1.setVgap( 10 );
f1.setHgap( 20 );
```

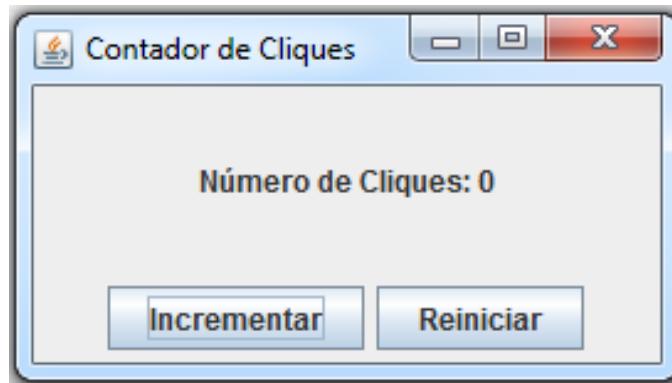
## ■ Exemplo

- Posicionamento dos Componentes do GUI (*Layout*)



# Gestores FlowLayout

## ■ Exemplo



```
public class ContadorGUI extends JFrame {  
    private JLabel lblNumero;  
    private JButton btnIncrementar, btnReiniclar;  
  
    private static String s = "Número de Cliques: ";  
    private static final int JANELA_LARGURA = 270;  
    private static final int JANELA_ALTURA = 150;  
  
    public ContadorGUI() {  
        super("Contador de Cliques");  
  
        lblNumero = new JLabel(s + "0");  
        JPanel p = criarPainelBotoes();  
  
        add(lblNumero, BorderLayout.CENTER);  
        add(p, BorderLayout.SOUTH);  
  
        setDefaultCloseOperation(EXIT_ON_CLOSE);  
        setSize(JANELA_LARGURA, JANELA_ALTURA);  
        setLocationRelativeTo(null);  
        setVisible(true);  
    }  
  
    private JPanel criarPainelBotoes(){  
        btnIncrementar = new JButton("Incrementar");  
        btnReiniclar = new JButton("Reiniclar");  
        JPanel p = new JPanel();  
        p.add(btnIncrementar);  
        p.add(btnReiniclar);  
        return p;  
    }  
}
```

JButton

## ■ Divide contentor em

- Linhas e colunas
  - Programador define nº de linhas e nº de colunas
- Todas as células têm **sempre** o tamanho igual
  - Componentes gráficos são redimensionados na **mesma proporção** do redimensionamento do contentor

// semelhante a folha de cálculo



- Construtor permite especificar

- Número de linhas e de colunas

- Exemplo

```
GridLayout g = new GridLayout(3,3);
```

- Opcionalmente, o **intervalo** entre linhas (intH) e colunas ( intV)

- Sintaxe

- ```
GridLayout( nº linhas, nº colunas, intH, intV)
```

- Exemplo

- ```
GridLayout g = new GridLayout(3,3,4,5);
```

- Unidade de medida = Pixel

- Posicionamento de componentes

- Colocados linha-a-linha (1<sup>a</sup>, depois 2<sup>a</sup>, etc.)

- Exemplo

```
JPanel pane = new JPanel( g );
```

```
pane.add( new JButton("Um") ); // ocupará posição (1,1)
```

```
pane.add( new JButton("Dois") ); // ocupará posição (1,2)
```

```
...
```



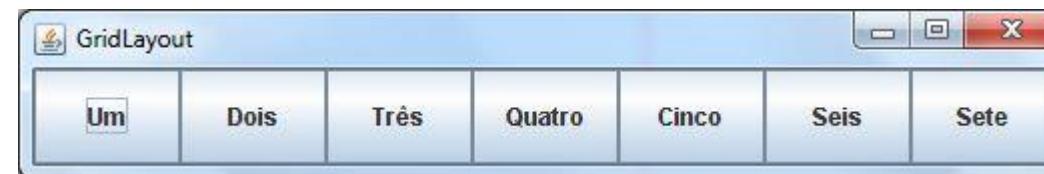
Intervalo entre Componentes

## ■ Definição da Grid (1/3)

- Nº de Linhas = Nº de Colunas = 0 // Ex: GridLayout g = new GridLayout(0,0);
  - Erro de execução
- Nº de Linhas > 0
  - Nº de colunas é **irrelevante** // pode ser qualquer nº
    - Gestor **maximiza** uso de linhas
  - Exemplos
    - Nº de linhas = 1: todos os componentes colocados na mesma linha

```
GridLayout g = new GridLayout(1,0);
```

```
GridLayout g = new GridLayout(1,5);
```



(1,0)  
(1,5)

- Nº de linhas = 5

```
GridLayout g = new GridLayout(5,0);
```

```
GridLayout g = new GridLayout(5,3);
```



(5,0)  
(5,3)

- Definição da Grid (2/3)

- Nº de Linhas > 0

- Exemplos

```
GridLayout g = new GridLayout(9,0);
```

```
GridLayout g = new GridLayout(9,3);
```

```
GridLayout g = new GridLayout(9,5);
```



(9,0)

(9,3)

(9,5)

## ■ Definição da Grid (3/3)

- Nº de Linhas = 0
  - Nº de colunas especificado é relevante
  - Gestor **maximiza** uso de colunas
  - Exemplos
    - Nº de Colunas = 1

```
GridLayout g = new GridLayout(0,1);
```

- Todos componentes colocados na mesma coluna

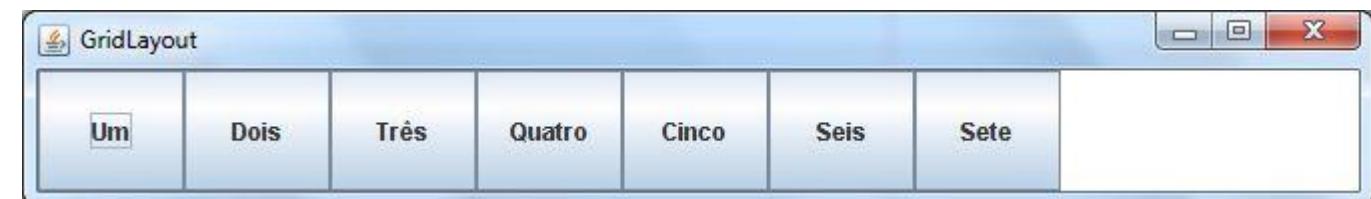
- Nº de Colunas = 3

```
GridLayout g = new GridLayout(0,3);
```

- Componentes distribuídos por 3 colunas

- Nº de Colunas = 9

```
GridLayout g = new GridLayout(0,9);
```



## ■ Exemplo (completo)

```
import javax.swing.*;  
  
public class GridLayoutGUIDemo extends JFrame {  
  
    private static final int JANELA_LARGURA = 400;  
    private static final int JANELA_ALTURA = 200;  
  
    public GridLayoutGUIDemo() {  
        super("GridLayout");  
        setLayout( new GridLayout(3,3) );  
  
        add( new JButton("Um") );  
        add( new JButton("Dois") );  
        add( new JButton("Três") );  
        add( new JButton("Quatro" ) );  
        add(new JButton("Cinco" ) );  
        add(new JButton(" Seis") );  
        add(new JButton("Sete") );  
  
        setDefaultCloseOperation(EXIT_ON_CLOSE);  
        setSize(JANELA_LARGURA, JANELA_ALTURA);  
        setLocationRelativeTo(null);  
        setVisible(true);  
    }  
}
```



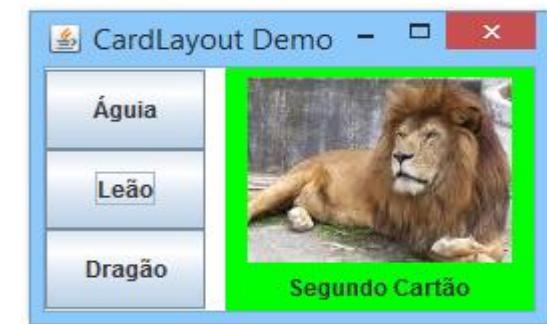
## ■ Exemplo (direita)

- Aplicação permite trocar o painel visível entre 3 painéis que partilham a mesma área do GUI

## ■ Interesse

- Gerir a **troca** do componente **visível**, entre vários **escondidos**, que **partilham** a mesma área do GUI

- i.e., mostrar um componente de cada vez, entre vários
- Em geral, esses componentes são contentores JPanel
- Análogo a uma pilha de cartões (ex: bancários)
  - Só o cartão de cima é visível
  - Cartão é um contentor de elementos
  - Daí, chamar **cartões (cards)** aos componentes geridos



Contentor CardLayout

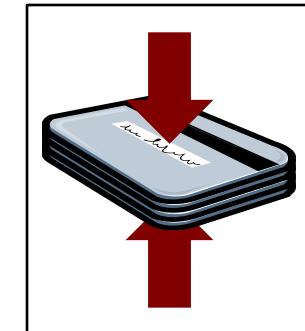
## ■ Troca do cartão visível

- Feita pela aplicação
- Pode ser originada pelo utilizador através de outros componentes
  - Exemplo
    - Botões de comando // Águia, Dragão e Leão

- Formas de trocar cartão visível

- Pedindo o **primeiro** ( first ) ou **último** ( last ) cartão
  - Segundo a ordem de adição dos cartões ao contentor
  - Métodos de instância CardLayout

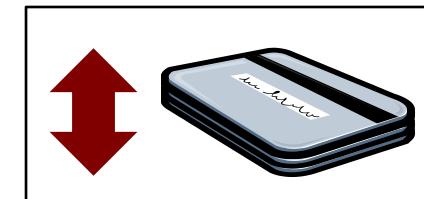
```
public void first( ContainerCardLayout parent )
public void last(ContainerCardLayout parent )
```



- Atravessando a pilha para o cartão **seguinte** ( next ) ou cartão **anterior** ( previous )

- Segundo a ordem de adição dos cartões ao contentor
- Métodos de instância CardLayout

```
public void next ( ContainerCardLayout parent )
    ■ último → primeiro
```

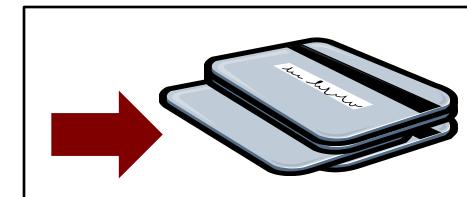


```
public void previous ( ContainerCardLayout parent )
    ■ primeiro → último
```

- Pedindo o cartão com o **nome especificado** na adição

- Método de instância CardLayout

```
public void show (ContainerCardLayout parent , String name)
```



## ■ Procedimento para usar (1/2)

1. Criar um contentor de componentes gerido por um CardLayout

```
private final CardLayout cl;  
private final JPanel pCardLayout; // pilha de cartões  
  
public CardLayoutDemo() {  
    super("CardLayout Demo");  
    cl = new CardLayout();  
    pCardLayout = criarPainelCardLayout();
```



2. Criar um contentor para cada cartão

```
private JPanel criarPainelAguia() {  
    JPanel p = new JPanel();  
    p.setBackground(Color.RED);  
    p.add(new JLabel(new ImageIcon("aguia.jpg")));  
    p.add(new JLabel("Primeiro Cartão"));  
    return p;  
}
```



3. Adicionar cada cartão criado ao contentor CardLayout (pilha)

```
private JPanel criarPainelCardLayout() {  
    JPanel p = new JPanel(cl);  
    p.add(criarPainelAguia(), AGUIA); // cartão 1 designado AGUIA  
    p.add(criarPainelLeao(), LEOA); // cartão 2 designado LEOA  
    p.add(criarPainelDragao(), DRAGAO); // cartão 3 designado DRAGAO  
    add(p, BorderLayout.WEST);  
    return p;  
}
```



```
private static final String AGUIA = "águia", LEOA = "leão", DRAGAO = "dragão";
```

## ■ Procedimento para usar (2/2)

## 4. Trocar o cartão visível

```
private JButton criarBotaoAguia() {
    JButton btn = new JButton("Águia");
    btn.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            cl.first(pCardLayout); // ou: cl.show(pCardLayout, AGUIA);
        }
    });
    return btn;
}

private JButton criarBotaoLeao() {
    JButton btn = new JButton("Leão");
    btn.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            cl.show(pCardLayout, LEAO);
            // Alternativa: cl.first(pCardLayout); cl.next(pCardLayout);
        }
    });
    return btn;
}

private JButton criarBotaoDragao() {
    JButton btn = new JButton("Dragão");
    btn.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            cl.last(pCardLayout); // ou: cl.show(pCardLayout, DRAGAO);
        }
    });
    return btn;
}
```



## ■ Exemplo Completo (1/3)

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class CardLayoutDemo extends JFrame {
    private final CardLayout cl;
    private final JPanel pCardLayout;
    private static final String AGUIA = "águia", LEAO = "leão", DRAGAO = "dragão";
    private static final int JANELA_LARGURA = 260, JANELA_ALTURA = 170;

    public CardLayoutDemo() {
        super("CardLayout Demo");
        getContentPane().setBackground(Color.white);
        final int INTERVALO_HORIZONTAL=10, INTERVALO_VERTICAL=0; // barra banca
        setLayout(new BorderLayout(INTERVALO_HORIZONTAL,INTERVALO_VERTICAL));
        JPanel pBotoes = criarPainelBotoes();
        cl = new CardLayout();
        pCardLayout = criarPainelCardLayout();
        add(pBotoes, BorderLayout.WEST);
        add(pCardLayout, BorderLayout.CENTER);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(JANELA_LARGURA, JANELA_ALTURA);
        setVisible(true);
    }
    private JPanel criarPainelCardLayout() { ...8 lines }
    private JPanel criarPainelAguia() { ...8 lines }
    private JPanel criarPainelLeao() { ...8 lines }
    private JPanel criarPainelDragao() { ...8 lines }
    private JPanel criarPainelBotoes() { ...11 lines }
    private JButton criarBotaoAguia() { ...10 lines }
    private JButton criarBotaoLeao() { ...10 lines }
    private JButton criarBotaoDragao() { ...10 lines }
    public static void main(String[] args) { ...3 lines }
}
```



## ■ Exemplo Completo (2/3)

```
private JPanel criarPainelCardLayout() {  
    JPanel p = new JPanel(c1);  
    p.add(criarPainelAguia(), AGUIA); // cartão 1 designado AGUIA  
    p.add(criarPainelLeao(), LEAO); // cartão 2 designado LEAO  
    p.add(criarPainelDragao(), DRAGAO); // cartão 3 designado DRAGAO  
    add(p, BorderLayout.WEST);  
    return p;  
}  
  
private JPanel criarPainelAguia() {  
    JPanel p = new JPanel();  
    p.setBackground(Color.RED);  
    JLabel lblAguia = new JLabel(new ImageIcon("aguia.jpg"));  
    p.add(lblAguia);  
    p.add(new JLabel("Primeiro Cartão"));  
    return p;  
}  
  
private JPanel criarPainelLeao() {  
    JPanel p = new JPanel();  
    p.setBackground(Color.GREEN);  
    JLabel lblLeao = new JLabel(new ImageIcon("leao.jpg"));  
    p.add(lblLeao);  
    p.add(new JLabel("Segundo Cartão"));  
    return p;  
}  
  
private JPanel criarPainelDragao() {  
    JPanel p = new JPanel();  
    p.setBackground(Color.BLUE);  
    JLabel lblDragao = new JLabel(new ImageIcon("dragao.jpg"));  
    p.add(lblDragao);  
    p.add(new JLabel("Terceiro Cartão"));  
    return p;  
}
```



## ■ Exemplo Completo (3/3)

```
private JPanel criarPainelBotoes() {
    JPanel p = new JPanel(new GridLayout(3, 1));
    final int PAINEL_LARGURA = 80, PAINEL_ALTURA = 40;
    p.setPreferredSize(new Dimension(PAINEL_LARGURA, PAINEL_ALTURA));
    p.add(criarBotaoAguia());
    p.add(criarBotaoLeao());
    p.add(criarBotaoDragao());
    return p;
}

private JButton criarBotaoAguia() {
    JButton btn = new JButton("Águia");
    btn.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            cl.first(pCardLayout); // ou: cl.show(pCardLayout, AGUIA);
        }
    });
    return btn;
}

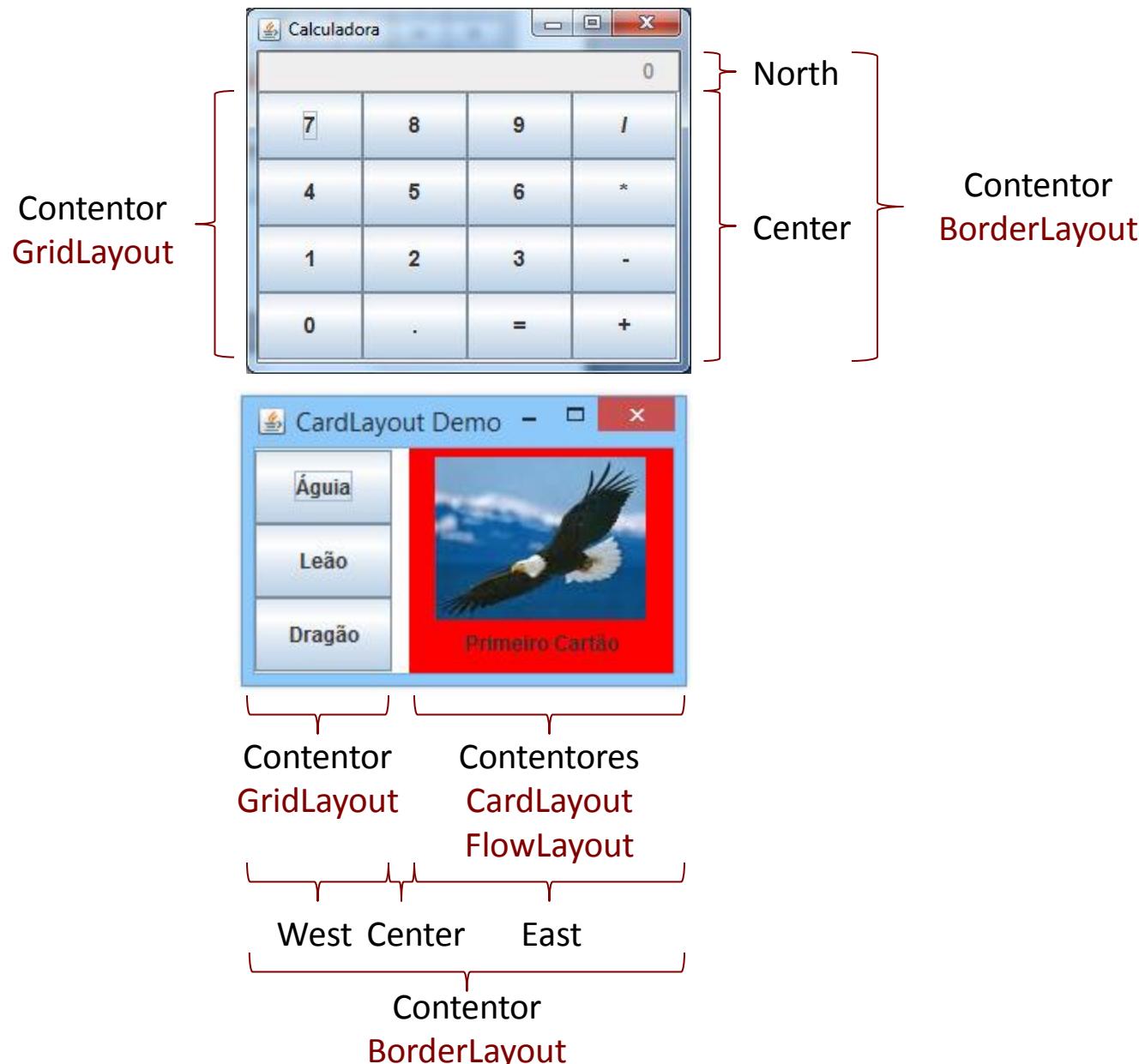
private JButton criarBotaoLeao() {
    JButton btn = new JButton("Leão");
    btn.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            cl.show(pCardLayout, LEO);
            // Alternativa: cl.first(pCardLayout); cl.next(pCardLayout);
        }
    });
    return btn;
}

private JButton criarBotaoDragao() {
    JButton btn = new JButton("Dragão");
    btn.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            cl.last(pCardLayout); // ou: cl.show(pCardLayout, DRAGAO);
        }
    });
    return btn;
}

public static void main(String[] args) {...3 lines...}
}
```



# Combinação de Vários Tipos de Gestores



- [Introdução](#)
- [Componentes Gráficos](#)
  - [Introdução](#)
  - [Hierarquia de Classes](#)
  - [Interfaces](#)
  - [Categorias](#)
    - [Contentores de Componentes Gráficos](#)
    - [Apresentação de Informação](#)
    - [Controlos Básicos](#)
- [Gestores de Posicionamento](#)
- [Manipuladores de Eventos](#)
- [Bibliografia Geral](#)
- [Índice Remissivo](#)



- [Conceitos Básicos](#)
- [Manipulação de Eventos](#)
- [Demos](#)

- [Conceitos Básicos](#)

- [Noção de Evento](#)
- [Tipos de Evento](#)
- [Classes de Evento](#)
  - [Noção](#)
  - [Hierarquia](#)
- [Interfaces Trata Evento](#)
  - [Noção](#)
  - [Hierarquia](#)
  - [Métodos de Evento](#)
- [Tipos de Eventos Gerados por Componentes](#)
  - [Principais](#)

- **Funcionamento de Aplicações GUI**

- É orientado por eventos
- Em geral, só executam ações após a ocorrência de acontecimentos, tais como:
  - Cliques no botão do rato
  - Movimentos do rato
  - Teclas premidas

- **Sistema operativo que suporta aplicações GUIs**

- Monitoriza constantemente eventos
- Reporta esses eventos às aplicações em execução
- Cada aplicação decide a forma de responder a esses eventos
  - Faz a **captura** e o **tratamento** dos eventos

- **Quais são os eventos que aplicação GUI pode manipular?**



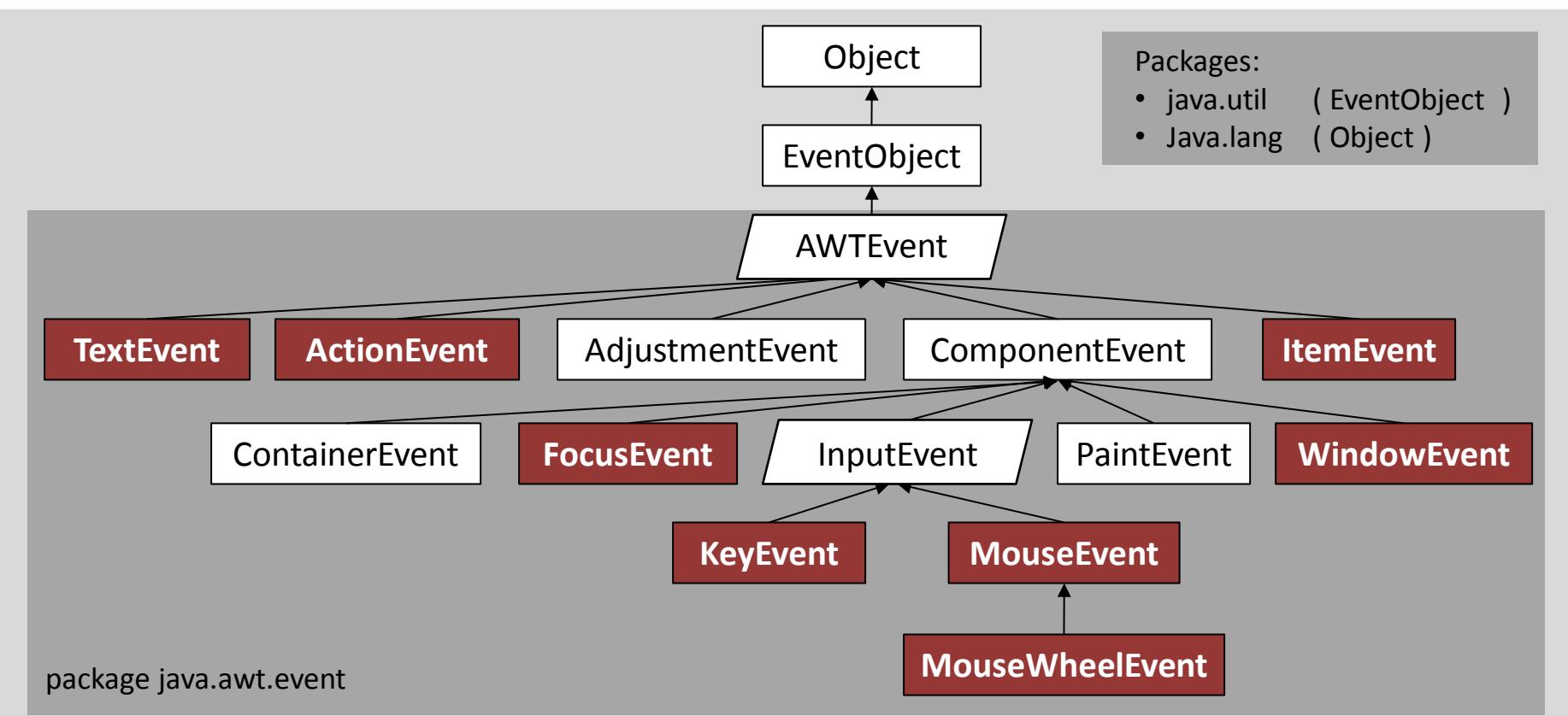
Tipo de Evento	Evento
Action	Clique num botão de comando
	Clique num item de menu
	Terminada edição de campo de texto
	Selecionado item de caixa de combinação
Component	Componente escondido
	Componente mostrado
	Componente movido
	Componente redimensionado
Key	Tecla mantida premida
	Tecla libertada
	Tecla premida (toque)
Mouse	Clique no rato
	Rato entrou no componente
	Rato saiu do componente
	Botão do rato premido
	Botão do rato libertado
	Rato movido
	Rato arrastado (premido + movido)
	Roda do rato movida

Tipo de Evento	Evento
Container	Componente adicionado
	Componente removido
Item	Item selecionado
	Item desselecionado
Adjustment	Scrollbar movido
Focus	Componente ganha foco
	Componente perde foco
Text	Texto alterado
Window	Janela abriu
	Janela fechou
	Janela ficou activa
	Janela ficou inactiva
	Janela ficou minimizada
	Janela ficou restaurada (tamanho original)
	Utilizador quer fechar Janela

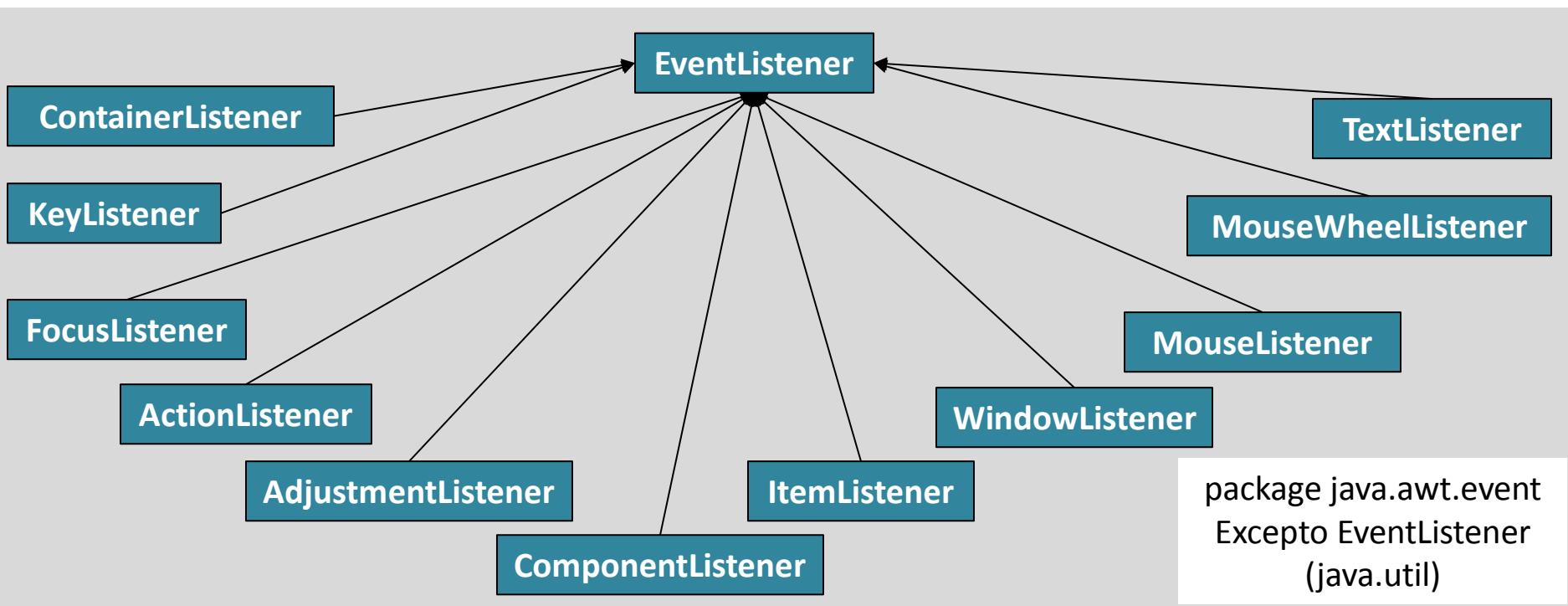
#### ■ Eventos iniciados pelo utilizador

- Eventos do rato
  - Movimentos // tipo Mouse
  - Cliques // tipo Action, Mouse
- Eventos do teclado
  - Teclas pressionadas // tipo Key

- Cada Tipo de Evento
  - Representado por **classe** de evento // Ex: tipo Action representado por classe ActionEvent
- Classes de Evento
  - Pertencem à hierarquia de classes ilustrada
  - São subclasses de **EventObject**
    - Topo da hierarquia (superclasse)



- Cada Tipo de Evento
  - Tem um interface associado
- Exemplos: Tipo Action      ↔ Interface ActionListener // tipo de Evento + Listener
- Exemplos: Tipo Window      ↔ Interface WindowListener
- Pertencem à hierarquia de interfaces ilustrada
  - São subinterfaces de EventListener // Topo da Hierarquia (superinterface)



- Especificam
  - Métodos de evento para tratar todos os eventos do tipo correspondente // slides seguintes

Evento	Classe de Evento	Interface Trata Evento	Método de Evento
Clique num botão de comando	ActionEvent	ActionListener	actionPerformed(ActionEvent e)
Clique num item de menu			
Termina edição de campo de texto			
Selecionado item caixa combinação			
Item seleccionado/des seleccionado	ItemEvent	ItemListener	itemStateChanged(ItemEvent e)
Componente escondido	ComponentEvent	ComponentListener	componentHidden(ComponentEvent e)
Componente mostrado			componentShown(ComponentEvent e)
Componente movido			componentMoved(ComponentEvent e)
Componente redimensionado			componentResized(ComponentEvent e)
Componente adquire foco	FocusEvent	FocusListener	focusGained(FocusEvent e)
Componente perde foco			focusLost(FocusEvent e)
Tecla mantida premida	KeyEvent	KeyListener	keyPressed(KeyEvent e)
Tecla libertada			keyReleased(KeyEvent e)
Tecla premida			keyTyped(KeyEvent e)
Clique no rato	MouseEvent	MouseListener	mouseClicked(MouseEvent e)
Rato entrou no componente			mouseEntered(MouseEvent e)
Rato saiu do componente			mouseExited(MouseEvent e)
Botão do rato premido			mousePressed(MouseEvent e)
Botão do rato libertado			mouseReleased(MouseEvent e)
Rato movido		MouseMotionListener	mouseMoved(MouseEvent e)
Rato arrastado (movido + premido)		mouseDragged(MouseEvent e)	
Roda do rato movida		MouseWheelListener	mouseWheelMoved(MouseWheelEvent e)

Evento	Classe de Evento	Interface Trata Evento	Método de Evento
Componente adicionado	ContainerEvent	ContainerListener	componentAdded(ContainerEvent e)
Componente removido			componentRemoved(ContainerEvent e)
Scrollbar movido	AdjustmentEvent	AdjustmentListener	adjutmentValue(AdjustmentEvent e)
Texto alterado	TextEvent	TextListener	textValueChanged(TextEvent e)
Janela abriu	WindowEvent	WindowListener	windowOpened(WindowEvent e)
Janela fechou			windowClosed(WindowEvent e)
Janela ficou activa			windowActivated(WindowEvent e)
Janela ficou inactiva			windowDeactivated(WindowEvent e)
Janela ficou minimizada			windowIconified(WindowEvent e)
Janela restaurada (tamanho original)			windowDeiconified(WindowEvent e)
Utilizador quer fechar Janela			windowClosing(WindowEvent e)

# Tipos de Eventos Gerados por Componentes GUI

Componente	Action	Adjustment	Component	Container	Focus	Item	Key	Mouse	Text	Window
JButton										
JCheckBox										
JComboBox										
JComponent										
Container										
JDialog										
JFileDialog										
JFrame										
JLabel										
JList										
JMenu										
JMenuItem										
JPopupMenu										
JPanel										
JScrollBar										
JScrollPane										
JTextArea										
JTextComponent										
JTextField										
Window										

Componente	Evento	Tipo de Evento Gerado
JButton	Clique no botão	Action
JComboBox	Selecionado item	
JList	Duplo clique	
JMenuItem	Selecionado Item de menu	
JTextField	Terminada edição de texto com ENTER	
JCheckBox	Selecionado ou desselecionado	Item
JCheckboxMenuItem	Selecionado ou desselecionado	
JList	Selecionado ou desselecionado item	
JTextComponent	Texto alterado	Text
JScrollBar	Scrollbar movido	Adjustment
JComponent	Componente movido, redimensionado, escondido ou mostrado	Component
	Componente adquire ou perde foco	Focus
	Tecla pressionada ou libertada	Key
	Rato com botão pressionado ou libertado	Mouse
Window	Utilizador quer fechar janela	Window

- Manipulação de Eventos

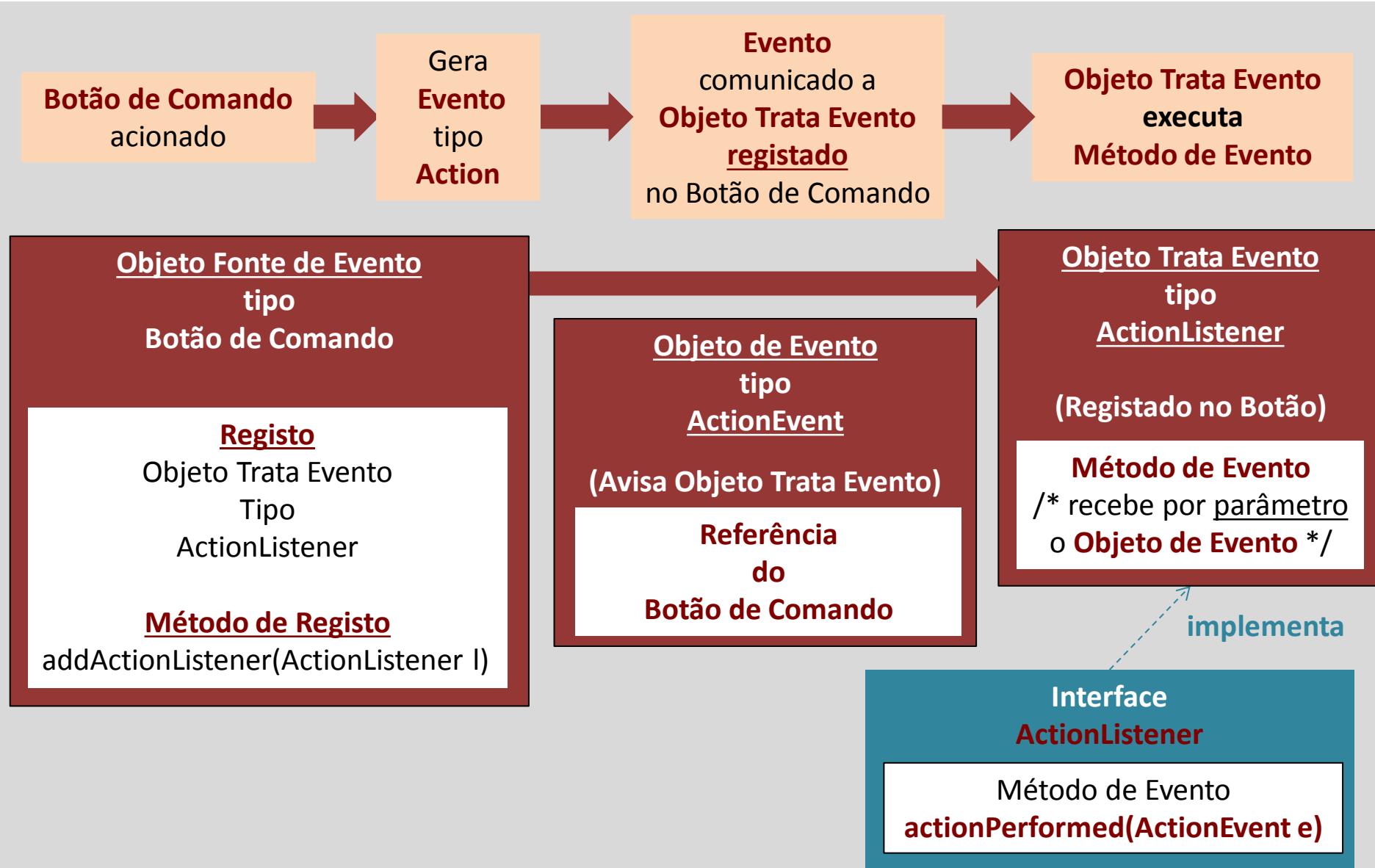
- Modelo de Eventos AWT
  - Trata Evento
  - Objeto Evento
  - Fonte de Evento
- Tratamento de Eventos
- Classes de Objetos Trata Evento
  - Externas
  - Internas
  - Anónimas
- Classes Adapter

- Aplicações GUI precisam de fazer:
  - Captura de eventos
  - Tratamento de eventos
- Programador precisa de conhecer
  - Modelo de Eventos (AWT)



# Modelo de Eventos (AWT)

- Exemplo de um Evento Action Gerado por um Botão de Comando (1/3)



## ■ Exemplo de um Evento Action Gerado por um Botão de Comando (2/3)

### ▪ Classe Interna para Manipular o Evento

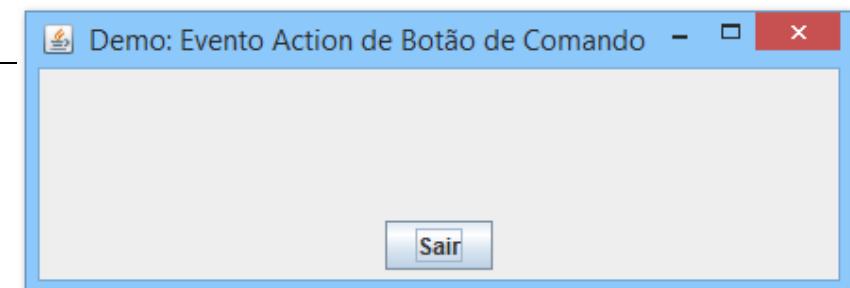
```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class Janela extends JFrame {
    private static final int JANELA_LARGURA = 330;
    private static final int JANELA_ALTURA = 150;
    public Janela() {
        super("Demo: Evento Action de Botão de Comando");

        add(criarPainelBotao(), BorderLayout.SOUTH);

        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(JANELA_LARGURA, JANELA_ALTURA);
        setVisible(true);
    }
    private JPanel criarPainelBotao() {
        JPanel p = new JPanel();
        p.add(criarBotaoSair());
        return p;
    }
    private JButton criarBotaoSair() {
        JButton btn = new JButton("Sair");
        TrataEvento t = new TrataEvento();
        btn.addActionListener(t);
        return btn;
    }
    private class TrataEvento implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            dispose();
        }
    }
    public static void main(String[] args) {
        new Janela();
    }
}

```



Objeto Fonte do Evento

Criação do Objeto Trata Evento

Registo do Objeto Trata Evento

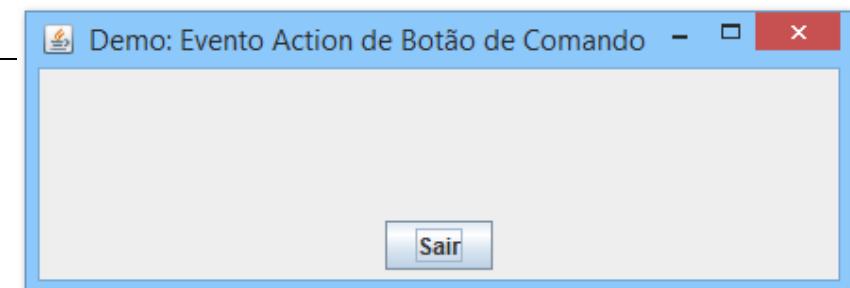
Método de Evento  
do  
Objeto Trata Evento

/\* recebe, por parâmetro, o  
**Objeto de Evento** enviado pelo  
botão btnSair \*/

## ■ Exemplo de um Evento Action Gerado por um Botão de Comando (3/3)

## ■ Classe Anónima para Manipular o Evento

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class Janela extends JFrame {
    private static final int JANELA_LARGURA = 330;
    private static final int JANELA_ALTURA = 150;
    public Janela() {
        super("Demo: Evento Action de Botão de Comando");
        add(criarPainelBotao(), BorderLayout.SOUTH);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(JANELA_LARGURA, JANELA_ALTURA);
        setVisible(true);
    }
    private JPanel criarPainelBotao() {
        JPanel p = new JPanel();
        p.add(criarBotaoSair());
        return p;
    }
    private JButton criarBotaoSair() {
        JButton btn = new JButton("Sair");
        btn.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                dispose();
            }
        });
        return btn;
    }
    public static void main(String[] args) {...3 lines...}
}
```



Objeto Fonte do Evento

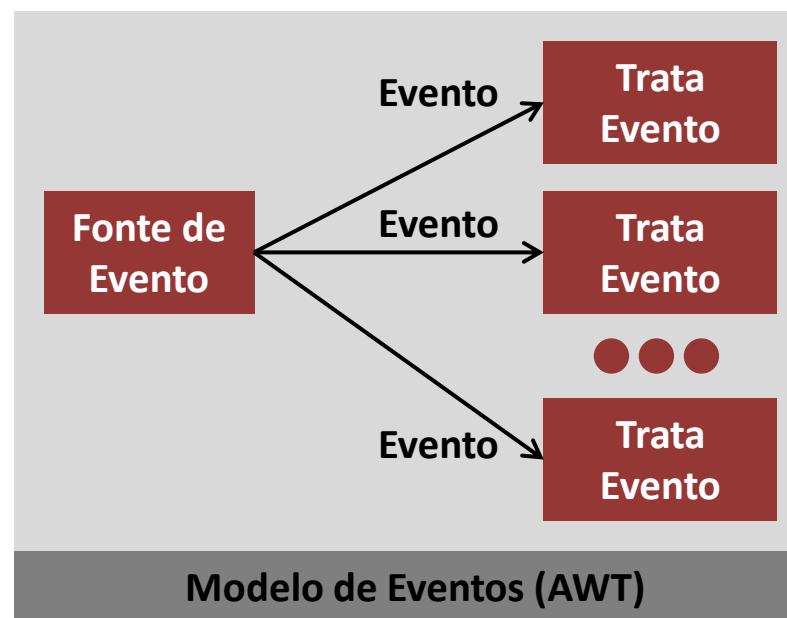
Registo do Objeto Trata Evento  
(instância de classe anónima)

Método de Evento  
do  
Objeto Trata Evento

/\* Recebe, por parâmetro, o  
**Objeto de Evento** enviado pelo  
botão btnSair \*/

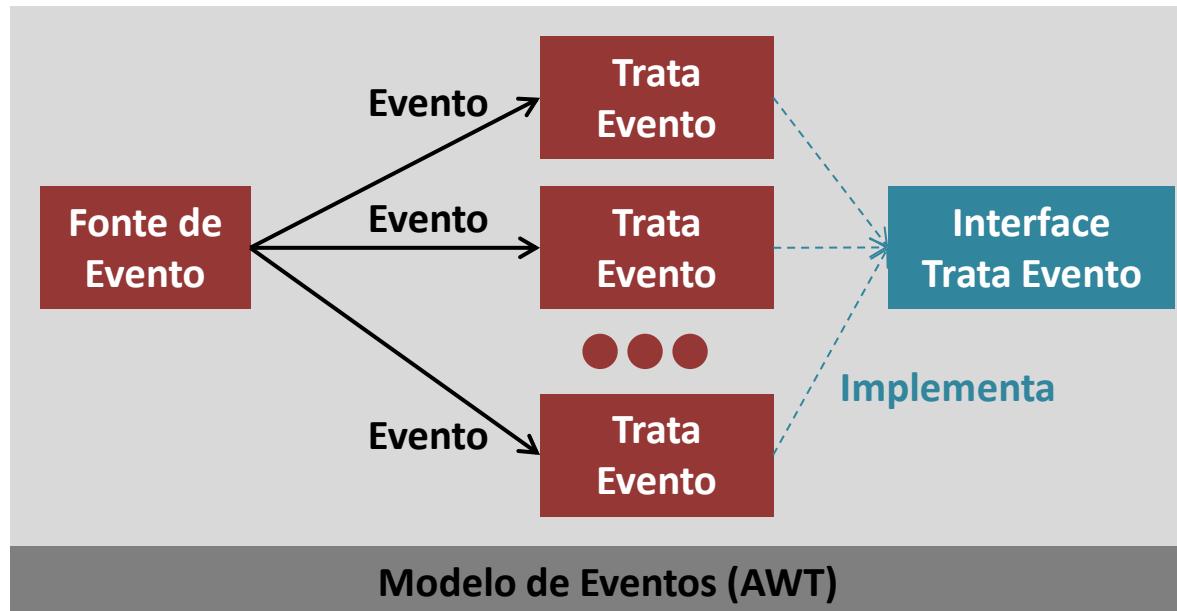
- Tipos de Objetos Envolvidos num Evento

- Fonte de Evento
  - /\* Gera um tipo de evento e comunica-o, através de objetos Evento, a objetos Trata Evento registados nessa fonte e nesse tipo de evento \*/
  - // Num evento, existe apenas uma fonte
- Evento
  - // Comunica o evento a um objeto Trata Evento
  - // Gerado pela Fonte de Evento (um para cada objeto Trata Evento)
- Trata Evento
  - // Trata o evento após notificação recebida num objeto Evento
  - // Podem existir múltiplos



- Responsabilidade

- Tratar um evento após notificação recebida num objeto Evento



- Têm de ser objetos de classes que implementem Interfaces Trata Evento
  - Designadas Classes Trata Evento
  - Interface Trata Evento deve corresponder ao tipo de evento a tratar
    - Exemplos
      - Slide seguinte

## ■ Exemplos de classes Trata Evento

- Para tratar eventos do tipo Action
  - Implementar interface ActionListener ⇒ implementar método de evento actionPerformed especificado pela interface
- Para tratar eventos do tipo Mouse
  - Implementar interface MouseListener e/ou MouseMotionListener ⇒ têm de implementar todos os métodos de evento correspondentes descritos na tabela abaixo
  - Alternativa, herdar a classe MouseAdapter (Classes Adapter apresentadas mais adiante):
    - Permitem implementar apenas alguns dos métodos especificados pela Interface

Evento	Classe de Evento	Interface Trata Evento	Método de Evento
Clique num botão de comando	ActionEvent	ActionListener	actionPerformed(ActionEvent e)
Clique num item de menu			
Termina edição de campo de texto			
Selecionado item caixa de combinação			
Clique no rato	MouseEvent	MouseListener	mouseClicked(MouseEvent e)
Rato entrou no componente			mouseEntered(MouseEvent e)
Rato saiu do componente			mouseExited(MouseEvent e)
Botão do rato premido			mousePressed(MouseEvent e)
Botão do rato libertado		MouseMotionListener	mouseReleased(MouseEvent e)
Rato movido			mouseMoved(MouseEvent e)
Rato arrastado (movido + premido)			mouseDragged(MouseEvent e)

- Responsabilidade

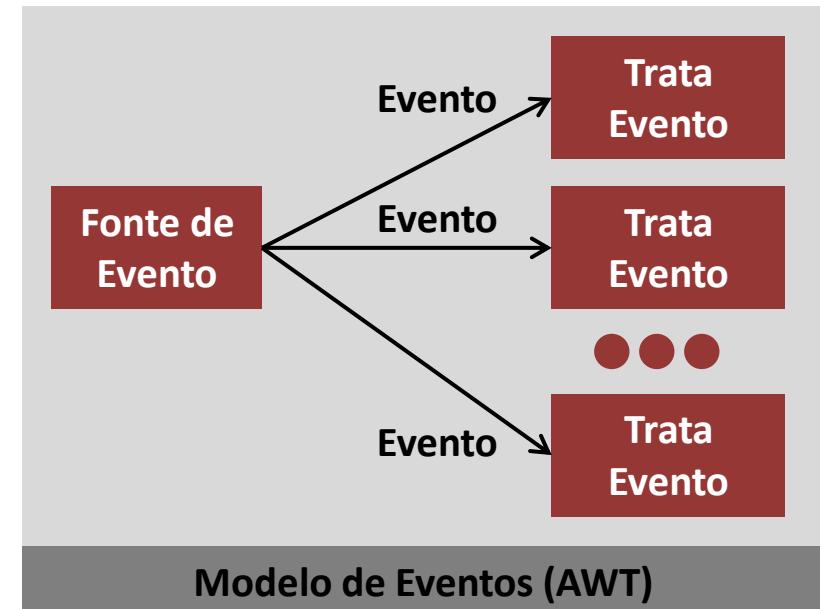
- Gerar um tipo de evento e comunicá-lo aos objetos Trata Evento registrados nesse tipo de evento
  - A comunicação do evento é feita através de objetos de Evento

- É componente GUI

- Exemplos
  - Botão de comando // objeto JButton
  - Label // objeto JLabel
  - Campo de texto // objeto JTextField

- Cada Componente GUI fornece

- Métodos para registrar objetos Trata Evento dos tipos de evento que pode gerar
  - Um método para cada tipo de evento que pode gerar
    - Nome e parâmetro do género
      - `addEvent Listener( EventListener I )` // Event varia: Action, Mouse, etc.
- Exemplos
  - `addAction Listener( ActionListener I )` // regista objetos Trata Evento de eventos tipo Action
  - `addMouse Listener( MouseListener I )` // regista objetos Trata Evento de eventos tipo Mouse
  - `addKey Listener( KeyListener I )` // regista objetos Trata Evento de eventos tipo Key



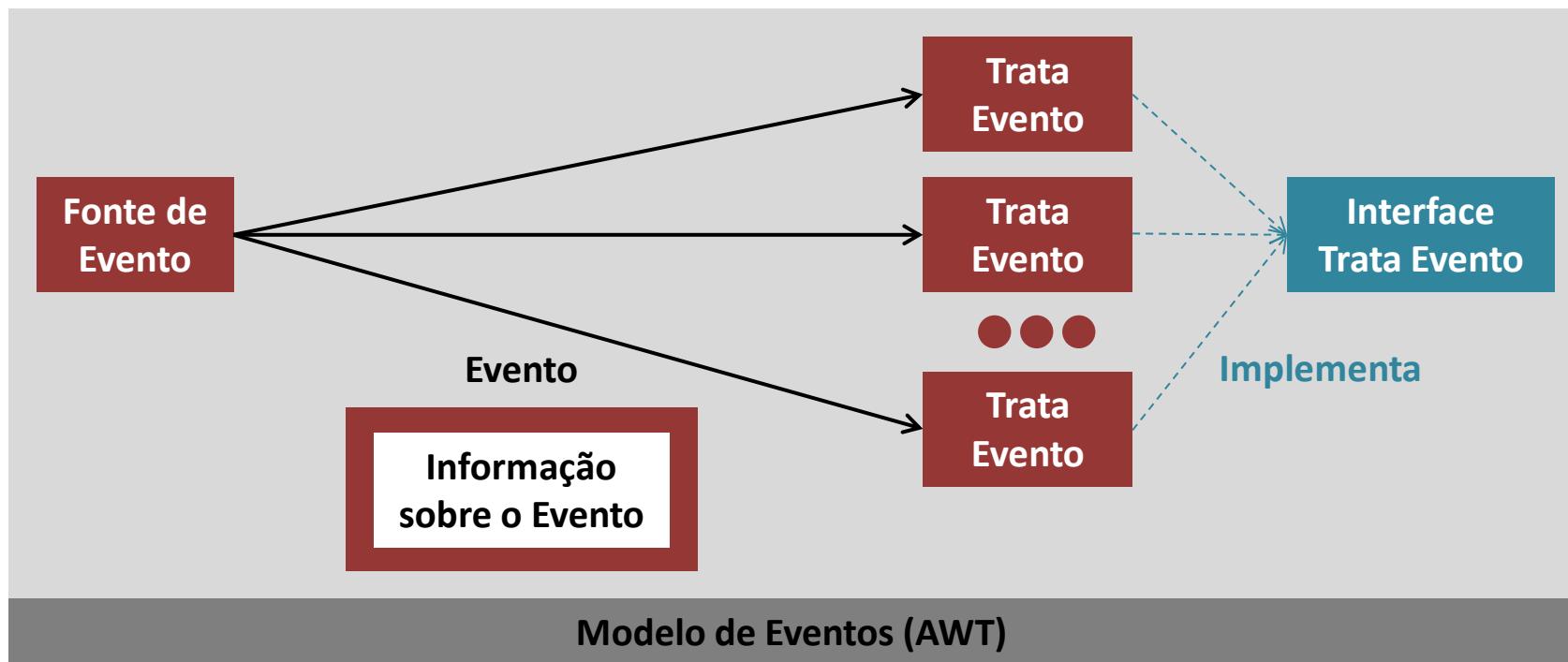
# Fonte de Evento

## ■ Exemplo do Botão de Comando (tipo JButton)

- Fornece os seguintes métodos para **registar** objetos Trata Evento
  - `public void addActionListener( ActionListener l )`
  - `public synchronized void addComponentListener( ComponentListener l )`
  - `public synchronized void addFocusListener( FocusListener l )`
  - `public synchronized void addKeyListener( KeyListener l )`
  - `public synchronized void addMouseListener( MouseListener l )`
  - `public synchronized void addMouseMotionListener( MouseMotionListener l )`
  - `public synchronized void addMouseWheelListener(MouseWheelListener l)`

Componente	Action	Adjustment	Component	Container	Focus	Item	Key	Mouse	Text	Windows
JButton										
JDialog										
JFrame										
JLabel										
JList										
JMenu										
JMenuItem										
JPanel										
JTextField										
Window										

- **Gerado**
    - Pela Fonte de Evento
  - **Enviado**
    - Aos objetos Trata Evento desse evento, registados na Fonte de Evento
  - **Responsabilidade**
    - Encapsular **informação** sobre o evento ocorrido
      - Exemplo
        - Referência do componente Fonte de Evento



## ■ É instância de Classe de Evento

- Existe uma dessas classes para cada tipo de evento
- Exemplos

Evento	Tipo de Evento	Classe de Evento
Clique num JButton	Action	ActionEvent
Clique num JMenuItem		
Termina a edição de um TextField		
Selecionado item caixa de combinação	Item	ItemEvent
Item selecionado/desselectedo		
Componente escondido	Component	ComponentEvent
Componente mostrado		
Componente movido		
Componente redimensionado		
Componente adquire foco	Focus	FocusEvent
Componente perde foco		
Clique no rato	Mouse	MouseEvent
Rato entrou no componente		
Rato saiu do componente		
Rato com botão premido		
Rato com botão libertado		MouseMotionEvent
Rato movido		
Rato arrastado (movido + premido)		

## ■ Passado por Parâmetro

- Aos métodos de evento dos objetos Trata Evento
- Exemplo

```
private class TrataEvento implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        dispose();
    }
}
```



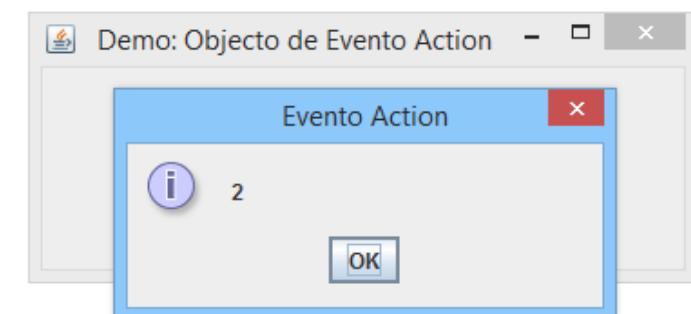
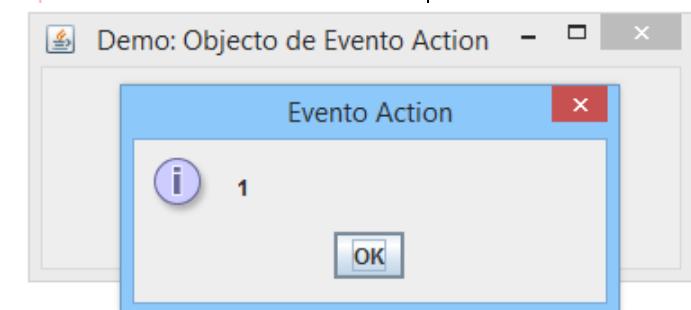
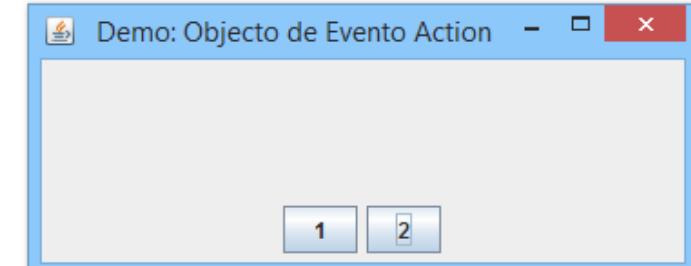
### Parâmetro e

- Passa para dentro do método o **objeto Evento** gerado pela Fonte de Evento

## ■ Métodos (mais usados)

Método	Obs
public Object getSource()	<ul style="list-style-type: none"> <li>• Retorna o objeto Fonte do Evento</li> <li>• Permite a um <b>método de evento</b> processar diferentes fontes           <ul style="list-style-type: none"> <li>• Exemplo: e.getSource() == btnSair</li> </ul> </li> </ul>
public String getActionCommand()	<ul style="list-style-type: none"> <li>• Retorna o título do comando associado a esta ação</li> <li>• Permite identificar a ação pretendida</li> <li>• Só disponível em eventos do tipo Action           <ul style="list-style-type: none"> <li>• Exemplo: retorna o título do botão Sair</li> </ul> </li> </ul>

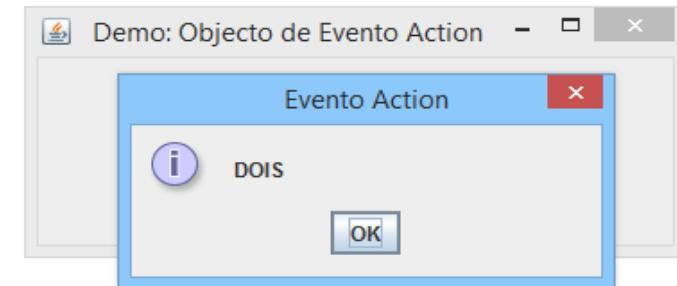
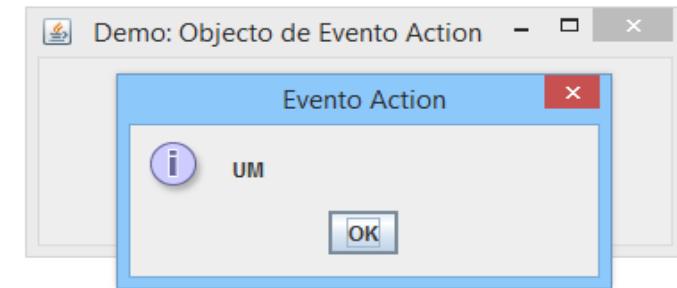
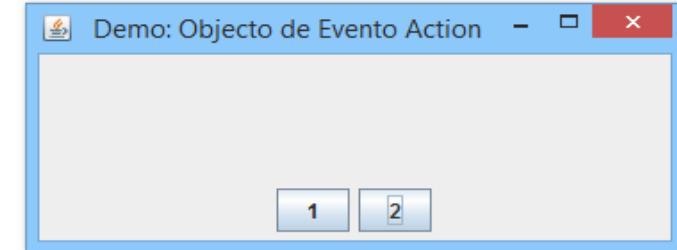
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class DemoObjectoEvento1 extends JFrame {
    private static final int JANELA_LARGURA = 370;
    private static final int JANELA_ALTURA = 150;
    public DemoObjectoEvento1() {
        super("Demo: Objeto de Evento Action");
        add(criarPainelBotoes(), BorderLayout.SOUTH);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(JANELA_LARGURA, JANELA_ALTURA);
        setVisible(true);
    }
    private JPanel criarPainelBotoes(){
        TrataEvento t = new TrataEvento();
        JPanel p = new JPanel();
        p.add(criarBotao("1", t));
        p.add(criarBotao("2", t));
        return p;
    }
    private JButton criarBotao(String titulo, TrataEvento t){
        JButton btn = new JButton(titulo);
        btn.addActionListener(t);
        return btn;
    }
    private class TrataEvento implements ActionListener{
        public void actionPerformed(ActionEvent e) {
            JOptionPane.showMessageDialog(DemoObjectoEvento1.this,e.getActionCommand(),"Evento Action",1);
        }
    }
    public static void main(String[] args) {
        new DemoObjectoEvento1();
    }
}
```



```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class DemoObjectoEvento2 extends JFrame {
    private JButton btn1, btn2;
    private static final int JANELA_LARGURA = 370;
    private static final int JANELA_ALTURA = 150;
    public DemoObjectoEvento2() {
        super("Demo: Objecto de Evento Action");
        add(criarPainelBotoes(), BorderLayout.SOUTH);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(JANELA_LARGURA, JANELA_ALTURA);
        setVisible(true);
    }
    private JPanel criarPainelBotoes() {
        TrataEvento t = new TrataEvento();
        JPanel p = new JPanel();
        btn1 = criarBotao("1", t);
        btn2 = criarBotao("2", t);
        p.add(btn1);
        p.add(btn2);
        return p;
    }
    private JButton criarBotao(String titulo, TrataEvento t) {
        JButton btn = new JButton(titulo);
        btn.addActionListener(t);
        return btn;
    }
    private class TrataEvento implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            Object obj = e.getSource();
            if (obj == btn1) {
                JOptionPane.showMessageDialog(DemoObjectoEvento2.this, "UM", "Evento Action", 1);
            } else if (obj == btn2) {
                JOptionPane.showMessageDialog(DemoObjectoEvento2.this, "DOIS", "Evento Action", 1);
            }
        }
    }
    public static void main(String[] args) {...3 lines}
}

```



Método de Evento  
trata eventos do tipo  
Action gerados por  
duas Fontes de  
Evento diferentes

- Procedimento

1. Criar **classe** de objetos Trata Evento
2. Criar objeto **Trata Evento**
3. Registar objeto Trata Evento na Fonte de Evento

## Procedimento – Passo 1: Criar Classe Trata Evento

- Pode haver duas formas:
  - Através da Implementação de um Interface Trata Evento
    - Interface Trata Evento do tipo de evento a tratar
    - Exemplo: clique num botão de comando (tipo JButton)
      - JButton gera evento do tipo Action
  - ↓  
Classe Trata Evento deve implementar interface ActionListener
- Através da Herança de uma Classe Adapter
  - Usada em casos particulares // analisaremos mais adiante
- Relativamente à Localização, uma Classe Trata Evento pode ser:
  - Classe Externa // permite o seu uso em diferentes classes
  - Classe Interna // permite acesso a membros inacessíveis de uma classe
    - Identificada // permite criar múltiplos Objetos de Evento
    - Anónima // permite tratar múltiplas Fontes de Evento
  - Anónima // permite tratar apenas um tipo de evento de uma Fonte
- Exemplos
  - Próximos Slides

- **Interesse:** ser reutilizada em múltiplas classes
- **Possível:** se não precisar de membros inacessíveis de outras classes

```
public class TrataEventoAction implements ActionListener {           // Classe Trata Evento Externa
    public void actionPerformed(ActionEvent e) {
        // código de resposta a eventos do tipo Action; p.ex., gerado pelo botão
    }
}
public class DemoTrataEventoAction1 extends JFrame {
    private static final int JANELA_LARGURA = 200;
    private static final int JANELA_ALTURA = 200;
    public DemoTrataEventoAction1() {
        super("Demo Trata Evento Action");
        add( criarBotao() );
        setDefaultCloseOperation( EXIT_ON_CLOSE );
        setSize(JANELA_LARGURA, JANELA_ALTURA);
        setVisible(true);
    }
    private JButton criarBotao(){
        JButton btn = new JButton("Calcular");
        TrataEventoAction t = new TrataEventoAction();      // cria objeto t para tratar eventos tipo Action
        btn.addActionListener( t );                         /* regista objeto t para tratar o evento Action do botao btn */
        return btn;
    }
}
public class Main {
    public static void main(String[] args) { new DemoTrataEventoAction1(); }
}
```

## ■ Interesse

- Classe interna: acesso a membros de classe inacessíveis do exterior
- Identificação: criar vários objetos Trata Evento e tratar eventos de múltiplas Fontes de Evento

```
public class DemoTrataEventoAction2 extends JFrame {  
  
    private static final int JANELA_LARGURA = 200;  
    private static final int JANELA_ALTURA = 200;  
  
    public DemoTrataEventoAction2() {  
        super("Demo Trata Evento Action");  
        add( criarBotao() );  
        setDefaultCloseOperation( EXIT_ON_CLOSE );  
        setSize(JANELA_LARGURA, JANELA_ALTURA);  
        setVisible(true);  
    }  
    private JButton criarBotao(){  
        JButton btn = new JButton("Calcular");  
        TrataEventoAction t = new TrataEventoAction();      // cria objeto t para tratar eventos tipo Action  
        btn.addActionListener( t );                         /* regista objeto t para tratar o evento Action do botao btn */  
        return btn;  
    }  
    private class TrataEventoAction implements ActionListener { // Classe Trata Evento Interna  
        public void actionPerformed(ActionEvent e) {  
            // código de resposta ao evento tipo Action; p.ex., gerado pelo botão  
        }  
    }  
    public class Main { public static void main(String[] args) { new DemoTrataEventoAction2(); } }  
}
```

- **Interesse:** tratar apenas um tipo de evento de uma Fonte de Evento

```
public class DemoTrataEventoAction3 extends JFrame {  
  
    private static final int JANELA_LARGURA = 200;  
    private static final int JANELA_ALTURA = 200;  
  
    public DemoTrataEventoAction3() {  
        super("Demo Trata Evento Action");  
        add( criarBotao() );  
        setDefaultCloseOperation( EXIT_ON_CLOSE );  
        setSize(JANELA_LARGURA, JANELA_ALTURA);  
        setVisible(true);  
    }  
  
    private JButton criarBotao(){  
        JButton btn = new JButton("Calcular");  
  
        btn.addActionListener( new ActionListener() {  
  
            public void actionPerformed(ActionEvent e) {  
                // código de resposta ao evento tipo Action gerado apenas pelo botão  
            }  
        });  
  
        return btn;  
    }  
  
    public class Main { public static void main(String[] args) { new DemoTrataEventoAction3(); } }  
  
    // Classe Trata Evento Anónima  
    // Objeto Trata Evento é instanciado e registado
```

## ■ Interesse

- Simplificar **criação** de Classes Trata Evento que precisam de implementar apenas **alguns** métodos de Interfaces Trata Evento que especificam múltiplos métodos
- Exemplo
  - Interface **WindowListener** especifica vários métodos e obriga a sua implementação em classes instanciáveis
    - `windowOpened(WindowEvent e)`
    - `windowClosed(WindowEvent e)`
    - `windowActivated(WindowEvent e)`
    - `windowDeactivated(WindowEvent e)`
    - `windowIconified(WindowEvent e)`
    - `windowDeiconified(WindowEvent e)`
    - `windowClosing(WindowEvent e)`
  - Em muitas situações
    - Apenas precisamos de tratar alguns destes métodos
- Java fornece Classe Adapter para cada Interface Trata Evento com **múltiplos** métodos especificados
  - Exemplos
    - `WindowAdapter`
    - `MouseListener`

- Usadas para definir Classes Trata Evento

- Têm de ser **herdadas** da classe Adapter
- Reescrevem apenas métodos de evento necessários
- Exemplos

```
// Classe Trata Evento Externa ou Interna
```

```
public class TrataEventoTerminator extends WindowAdapter {  
    public void windowClosing(WindowEvent e) {      // implementado apenas o método pretendido  
        System.exit(0);  
    }  
}
```

```
// Classe Trata Evento Anónima
```

```
JFrame janela = new JFrame();  
janela.addWindowListener(new WindowAdapter() {      // Classe anónima  
    public void windowClosing(WindowEvent e) {  
        System.exit(0);  
    };
```

- Demos

- Duplo-Clique numa JLabel
- Confirmação no Botão Fechar da Janela

## ■ Duplo-Clique numa JLabel

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class DemoDuploClique extends JFrame {

    private static final int JANELA_LARGURA = 330;
    private static final int JANELA_ALTURA = 150;

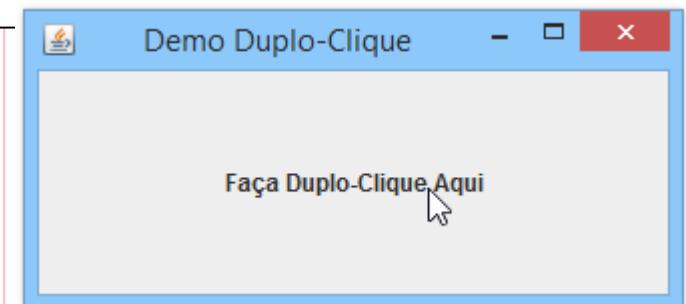
    public DemoDuploClique() {
        super("Demo Duplo-Clique");

        add(criarLegenda(), BorderLayout.CENTER);

        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(JANELA_LARGURA, JANELA_ALTURA);
        setVisible(true);
    }

    private JLabel criarLegenda(){
        JLabel lbl = new JLabel("Faça Duplo-Clique Aqui",JLabel.CENTER);
        lbl.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                if (e.getClickCount() == 2) {
                    JOptionPane.showMessageDialog(DemoDuploClique.this,"Duplo-Clique");
                }
            }
        });
        return lbl;
    }

    public static void main(String[] args) {
        new DemoDuploClique();
    }
}
```



JLabel

## ■ Confirmação no Botão Fechar da Janela

```
import java.awt.event.*;
import javax.swing.*;
public class DemoBotaoFehcarJanela extends JFrame {
    private static final int JANELA_LARGURA = 330;
    private static final int JANELA_ALTURA = 150;

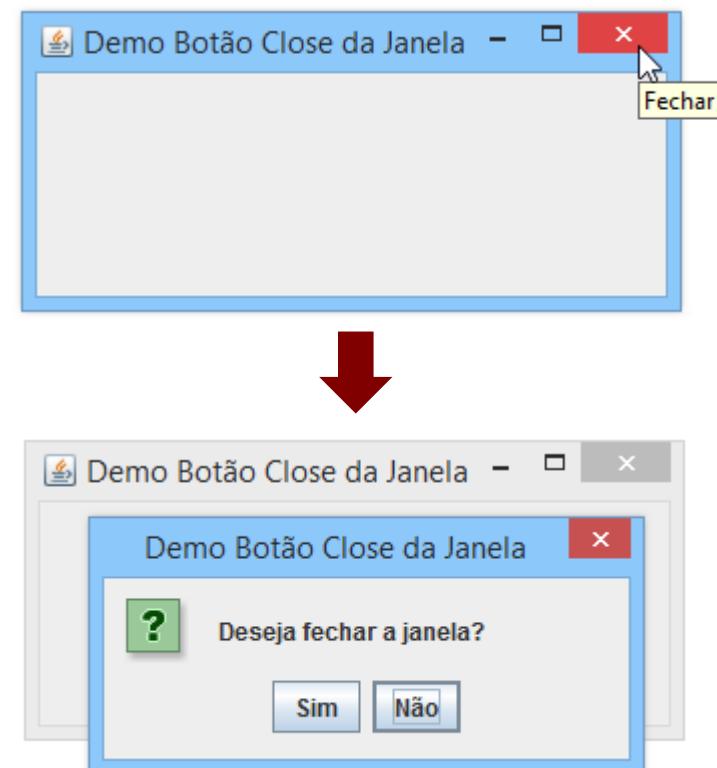
    public DemoBotaoFehcarJanela() {
        super("Demo Botão Close da Janela");

        addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                fechar();
            }
        });
    }

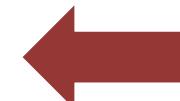
    setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
    setSize(JANELA_LARGURA, JANELA_ALTURA);
    setVisible(true);
}

private void fechar() {
    String[] opSimNao = {"Sim", "Não"};
    int resposta = JOptionPane.showOptionDialog(this,
        "Deseja fechar a janela?",
        "Demo Botão Close da Janela",
        0,
        JOptionPane.QUESTION_MESSAGE,
        null,
        opSimNao,
        opSimNao[1]);
    if (resposta == 0) {
        dispose();
    }
}

public static void main(String[] args) {...3 lines}
}
```



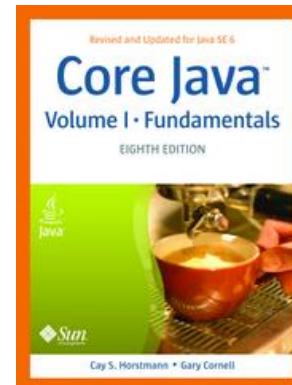
- [Introdução](#)
- [Componentes Gráficos](#)
  - [Introdução](#)
  - [Hierarquia de Classes](#)
  - [Interfaces](#)
  - [Categorias](#)
    - [Contentores de Componentes Gráficos](#)
    - [Apresentação de Informação](#)
    - [Controlos Básicos](#)
- [Gestores de Posicionamento](#)
- [Manipuladores de Eventos](#)
- [Bibliografia Geral](#)
- [Índice Remissivo](#)



# Bibliografia Geral

## ■ Livro

- Core Java, Volume I – Fundamentals  
Cay S. Horstmann and Gary Cornell  
Eighth Edition, 2007  
Prentice Hall



## ■ Tutoriais

- Componentes
  - <http://docs.oracle.com/javase/tutorial/uiswing/components/index.html>
- Gestores de Posicionamento
  - <http://download.oracle.com/javase/tutorial/uiswing/layout/index.html>
- Eventos
  - <http://download.oracle.com/javase/tutorial/uiswing/events/index.html>

## ■ API (*Application Programming Interface*)

- <http://docs.oracle.com/javase/8/docs/api/index.html>

# Índice Remissivo

- **Componentes Gráficos**
  - Controlos Básicos
    - Botão de Comando
      - [JButton](#)
    - Entrada de Texto
      - [JTextField](#)
    - Escolha de Opções
      - [JCheckBox](#)
      - [JComboBox](#)
      - [JList](#)
      - [JRadioButton](#)
    - Menus
      - [JMenu](#)
      - [JMenuBar](#)
      - [JMenuItem](#)
  - Apresentação de Informação
    - [JLabel](#)
    - [JSeparator](#)
- **Componentes Gráficos (continuação)**
  - Contentores de Componentes Gráficos
    - Painéis
      - [JPanel](#)
      - [JLayeredPane](#)
      - [JRootPane](#)
      - [JScrollPane](#)
    - Janelas
      - [JFrame](#)
    - Caixas de Diálogo
      - [JFileChooser](#)
      - [JOptionPane](#)
      - [JDialog](#)
  - Gestores de Posicionamento
    - [FlowLayout](#)
    - [BorderLayout](#)
    - [GridLayout](#)
    - [CardLayout](#)
  - Manipuladores de Eventos
    - [Evento Action](#)