Universidade Federal do ABC
Centro de Matemática, Computação e Cognição

Interface Gráfica com Usuários em JAVA

Monael Pinheiro Ribeiro, D.Sc.

GUI

- GUI → Graphical User Interface
- São componentes para promover uma interface gráfica de interação entre o usuário e a máquina através do tratamento de eventos.
- Eventos podem ser: cliques, focos, pressionar teclas etc.

GUI

- Algumas utilidade da GUI:
  - Familiaridade: Atualmente todos estão acostumados a interagir com o computador usando interfaces GUI.
  - Redução de Tempo: Desenvolvimento, aprendizado (programador e usuário).

AWT e Swing

- A primeira familia de GUI oferecida pelo JAVA foi a Abstract Windowing Toolkit (AWT) do pacote java.awt.
- Foi padrão de componentes GUI em JAVA de 1995 até 1998.
- Os componentes AWT estão associados com recursos da plataforma nativa da JVM, o que fazem os componentes serem diferentes em cada plataforma.

AWT e Swing

- A partir de 1998 (JDK1.2), JAVA ofereceu os componentes Swing do pacote javax.swing, com diversas vantagens sobre o pacote AWT.
- Os compontentes Swing são totalmente nativos do JAVA, o que lhes dão aparência uniforme em todas as plataformas.
- Embora os componentes Swing tornaram-se padrão GUI do JAVA, a partir da versão 1.2, seus componentes ainda utilizam componentes AWT como superclasses.

Parte do pacote java.awt

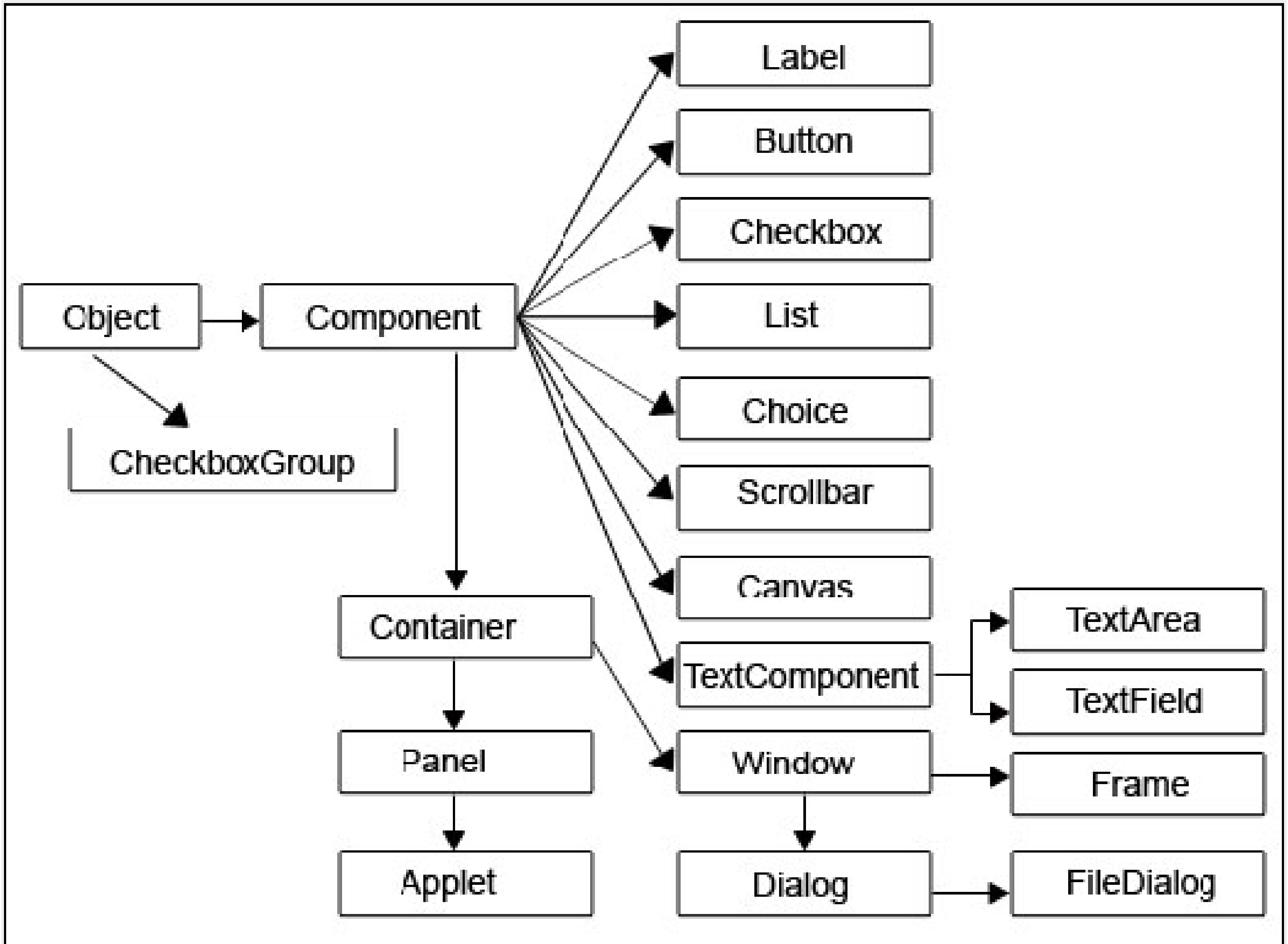
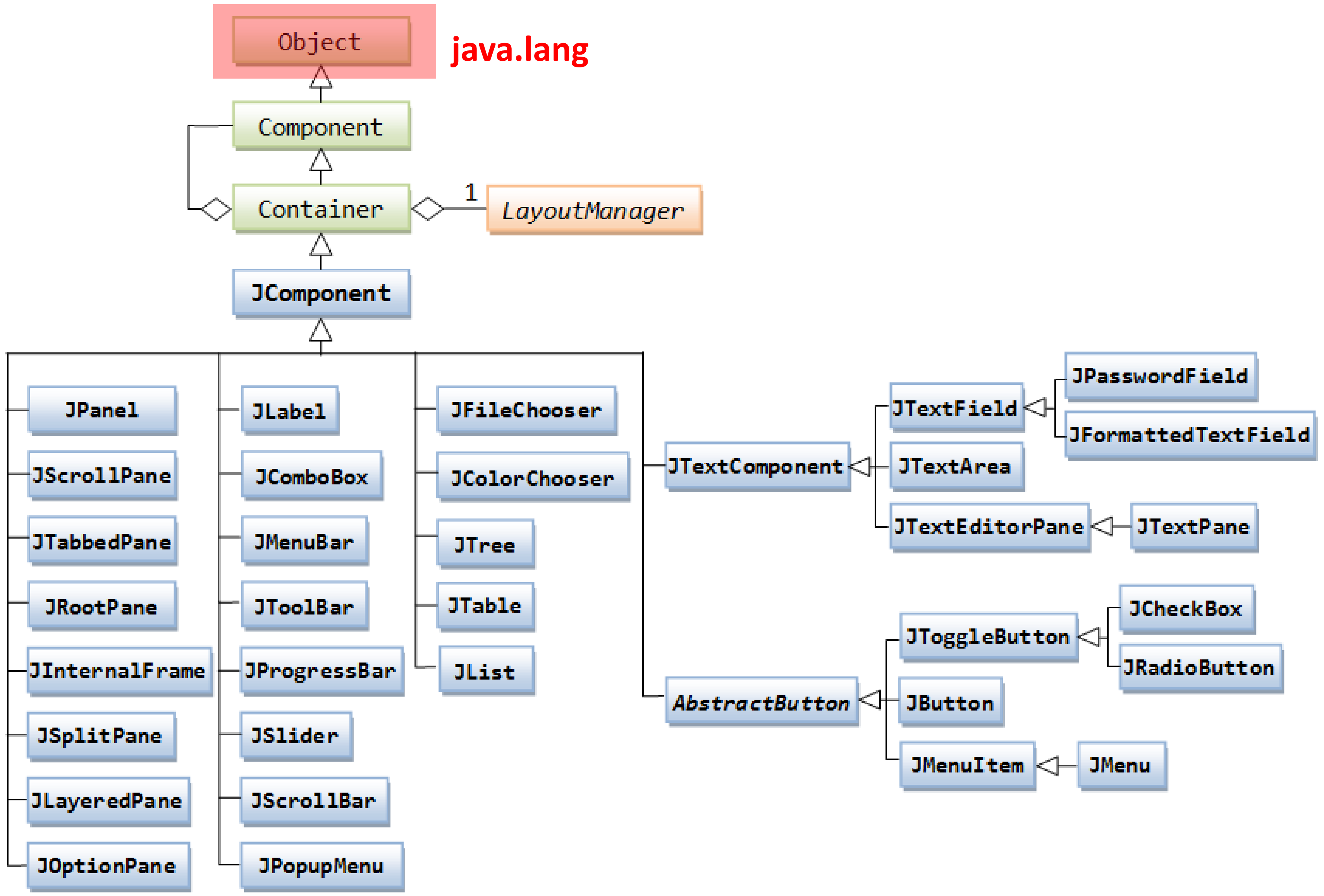
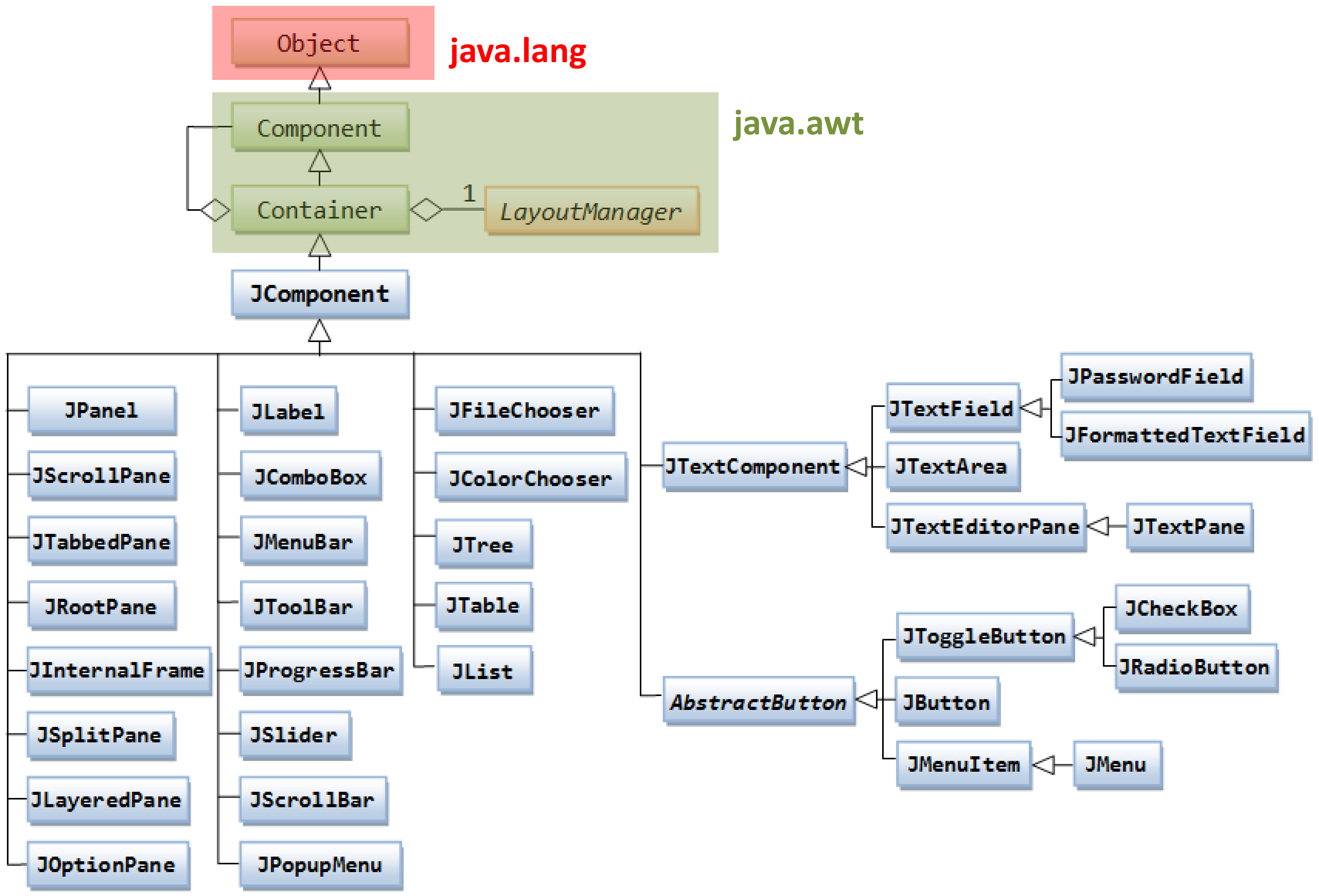


Figure: Component Hierarchy

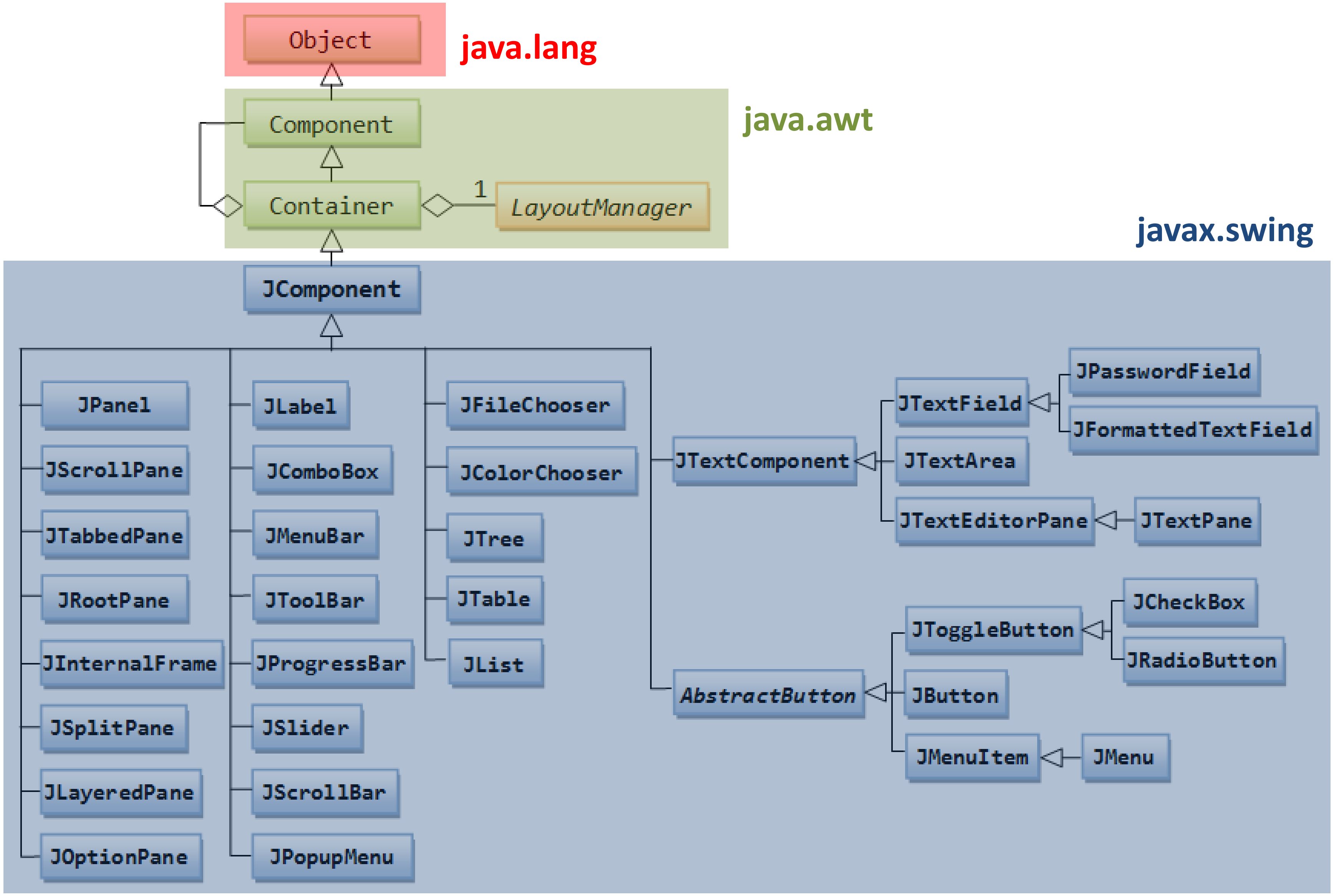
Parte do pacote java.swing



Parte do pacote java.swing



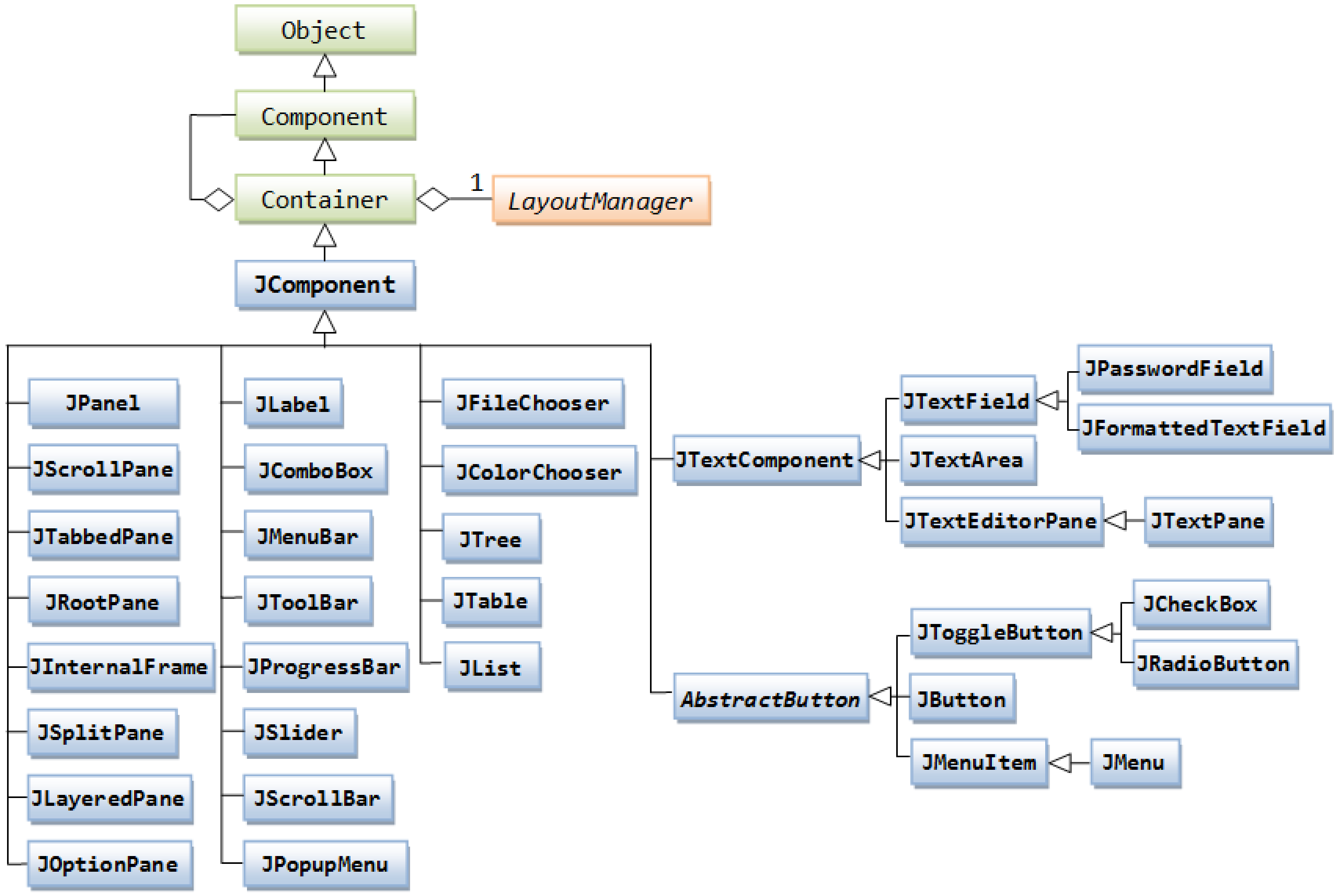
Parte do pacote java.swing



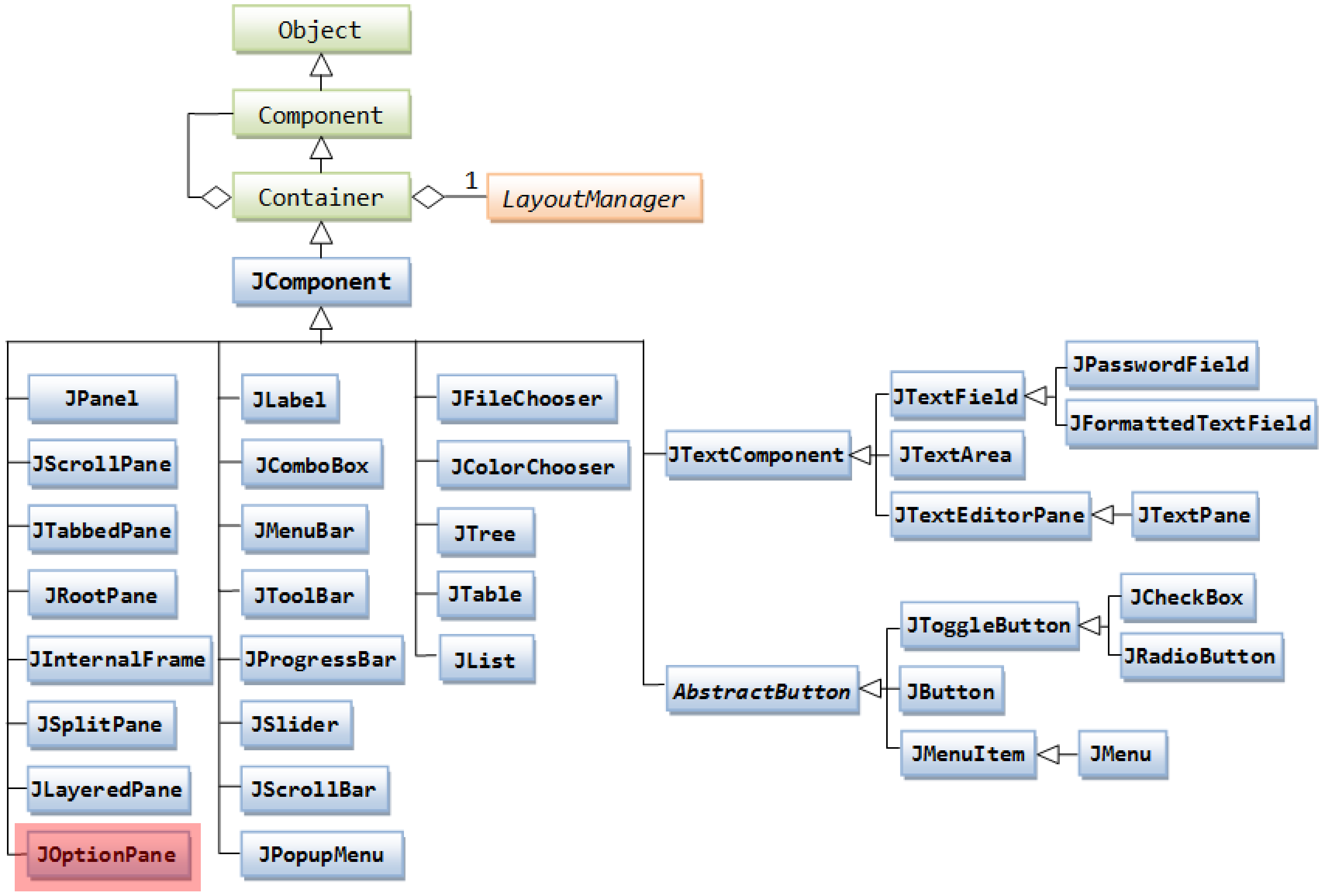
Caixa de Mensagens

- É o compomente mais simples do pacote Swing.
- Está definido na classe JOptionPane.

Parte do pacote java.swing



Parte do pacote java.swing



Caixa de Mensagens

- É o compomente mais simples do pacote Swing.
- Está definido na classe JOptionPane.
- Para exibir há um conjunto de métodos de classe:

Nome do Método	**Descrição**
`showConfirmDialog`	Exibe uma caixa de mensagens de confirmação, tais como: YES, NO, CANCEL.
`showInputDialog`	Exibe uma caixa de mensagens para entrada.
`showMessageDialog`	Exibe uma caixa de mensagens com uma informação.
`showOptionDialog`	Uma união das três anteriores.

Caixa de Mensagens

- É o compomente mais simples do pacote Swing.
- Está definido na classe JOptionPane.
- Para exibir há um conjunto de métodos de classe:

Nome do Método	**Descrição**
`showConfirmDialog`	Exibe uma caixa de mensagens de confirmação, tais como: YES, NO, CANCEL.
`showInputDialog`	Exibe uma caixa de mensagens para entrada.
`showMessageDialog`	Exibe uma caixa de mensagens com uma informação.
`showOptionDialog`	Uma união das três anteriores.

Métodos de Classe showConfirmDialog

Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.

Métodos de Classe showConfirmDialog

Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.

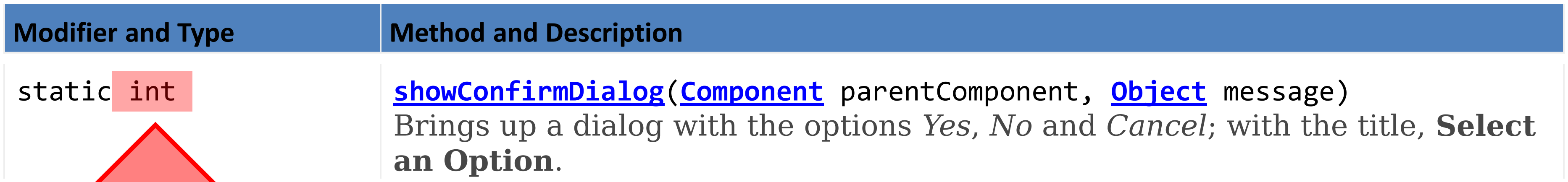
parentComponent - determina a janela onde a caixa de texto é exibida; se null, ou se parentComponent não tem uma janela, uma janela padrão é utilizada.

Métodos de Classe showConfirmDialog

Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.

message - Objeto a ser exibido.

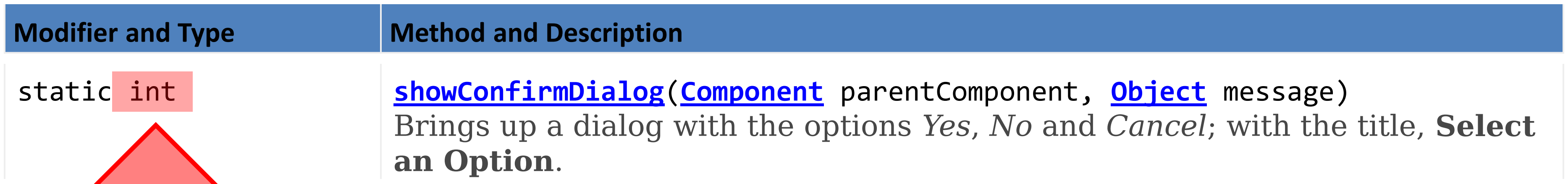
Métodos de Classe showConfirmDialog



Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.

Retorno: Um inteiro indicando a opção selecionado pelo usuário.

Métodos de Classe showConfirmDialog



Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.

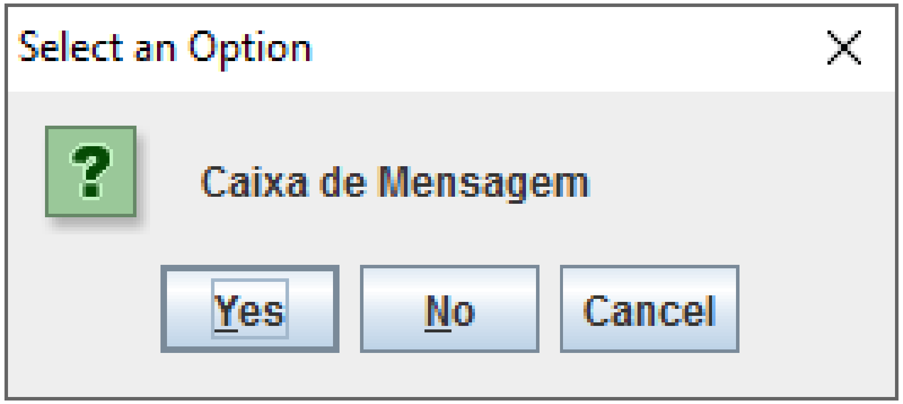
Retorno: Um inteiro indicando a opção selecionado pelo usuário.

Tais inteiros tratam-se de constantes definidas na própria classe JOptionPane e são eles:

- `JOptionPane.YES_OPTION`
- `JOptionPane.NO_OPTION`
- `JOptionPane.OK_OPTION`
- `JOptionPane.CANCEL_OPTION`

Métodos de Classe showConfirmDialog

Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.



```
retorno = JOptionPane.showConfirmDialog(null, "Caixa de Mensagem");
```

Métodos de Classe showConfirmDialog

Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType)` Brings up a dialog where the number of choices is determined by the optionType parameter.

Métodos de Classe showConfirmDialog

Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType)` Brings up a dialog where the number of choices is determined by the optionType parameter.

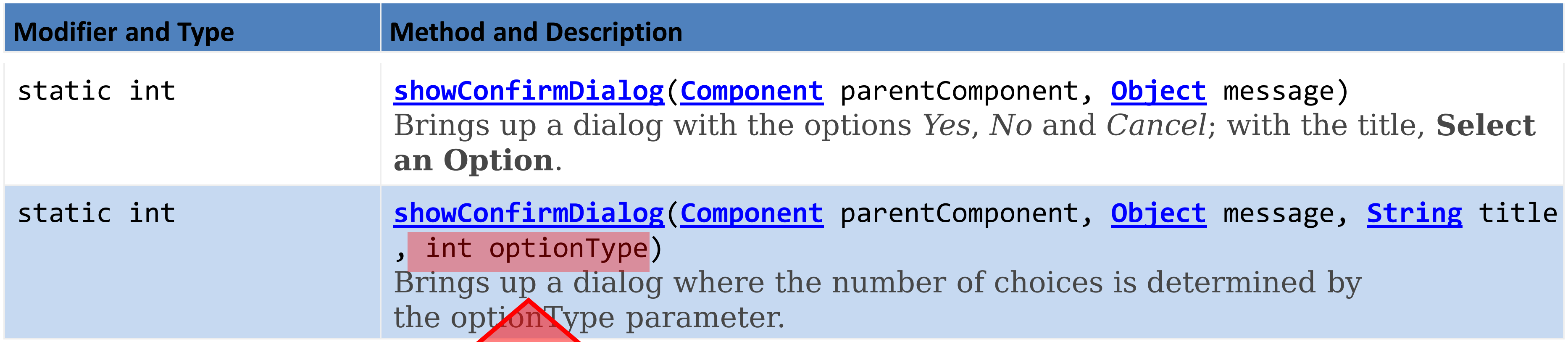
title – Uma string com um titulo para a caixa de mensagem.

Métodos de Classe showConfirmDialog

Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType)` Brings up a dialog where the number of choices is determined by the optionType parameter.

optionType – Um inteiro designando as opções disponiveis para a caixa de texto.

Métodos de Classe showConfirmDialog

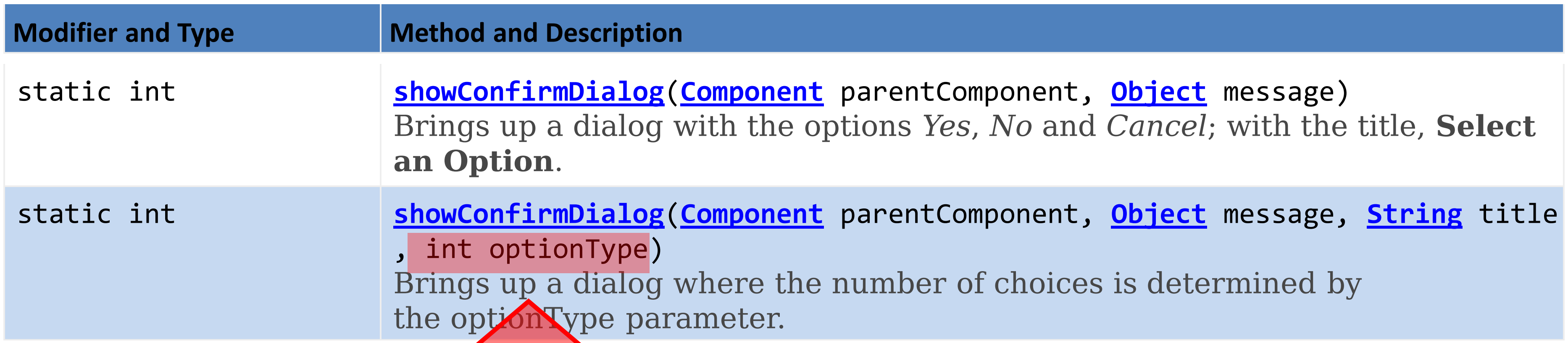


Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType)` Brings up a dialog where the number of choices is determined by the optionType parameter.

optionType – Um inteiro designando as opções disponiveis para a caixa de texto.

Tais inteiros tratam-se de constantes definidas na própria classe JOptionPane.

Métodos de Classe showConfirmDialog



Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType)` Brings up a dialog where the number of choices is determined by the optionType parameter.

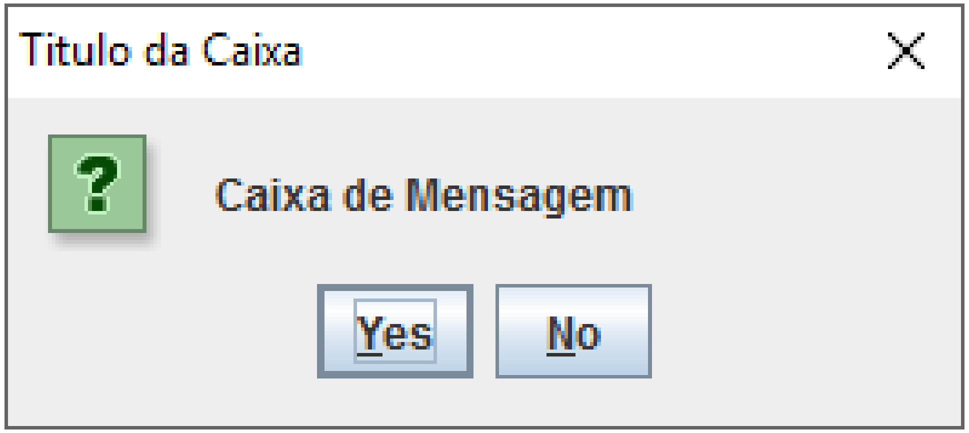
optionType – Um inteiro designando as opções disponiveis para a caixa de texto.

Tais inteiros tratam-se de constantes definidas na própria classe JOptionPane e são eles:

- `JOptionPane.YES_NO_OPTION`
- `JOptionPane.YES_NO_CANCEL_OPTION`
- `JOptionPane.OK_CANCEL_OPTION`

Métodos de Classe showConfirmDialog

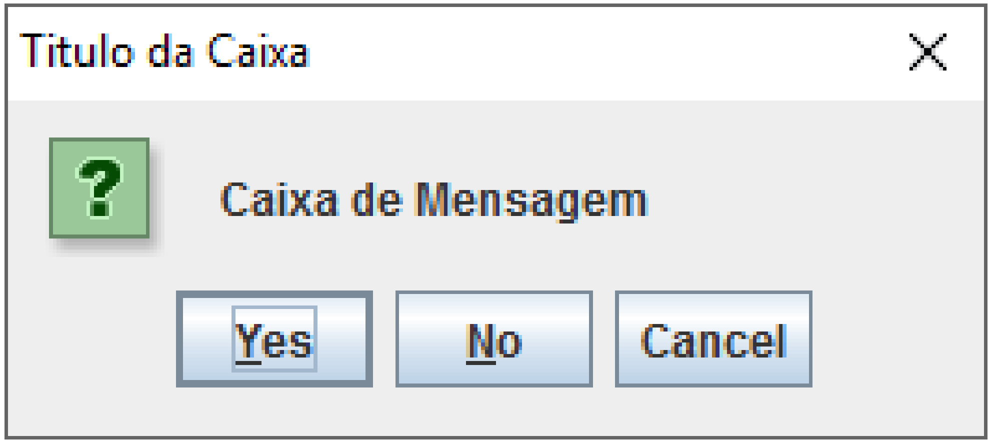
Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType)` Brings up a dialog where the number of choices is determined by the optionType parameter.



```
retorno = JOptionPane.showConfirmDialog(null, "Caixa de Mensagem",
"Titulo da Caixa", JOptionPane.YES_NO_OPTION);
```

Métodos de Classe showConfirmDialog

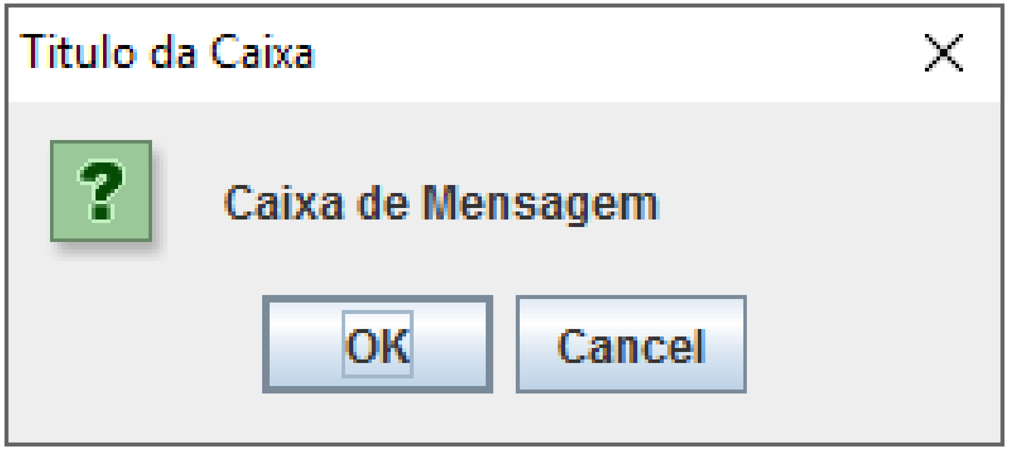
Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType)` Brings up a dialog where the number of choices is determined by the optionType parameter.



```
retorno = JOptionPane.showConfirmDialog(null, "Caixa de Mensagem",
"Titulo da Caixa", JOptionPane.YES_NO_CANCEL_OPTION);
```

Métodos de Classe showConfirmDialog

Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType)` Brings up a dialog where the number of choices is determined by the optionType parameter.



```
retorno = JOptionPane.showConfirmDialog(null, "Caixa de Mensagem",
"Titulo da Caixa", JOptionPane.OK_CANCEL_OPTION);
```

Métodos de Classe showConfirmDialog

Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType)` Brings up a dialog where the number of choices is determined by the optionType parameter.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType, int messageType)` Brings up a dialog where the number of choices is determined by the optionType parameter, where the messageTypeparameter determines the icon to display.

Métodos de Classe showConfirmDialog

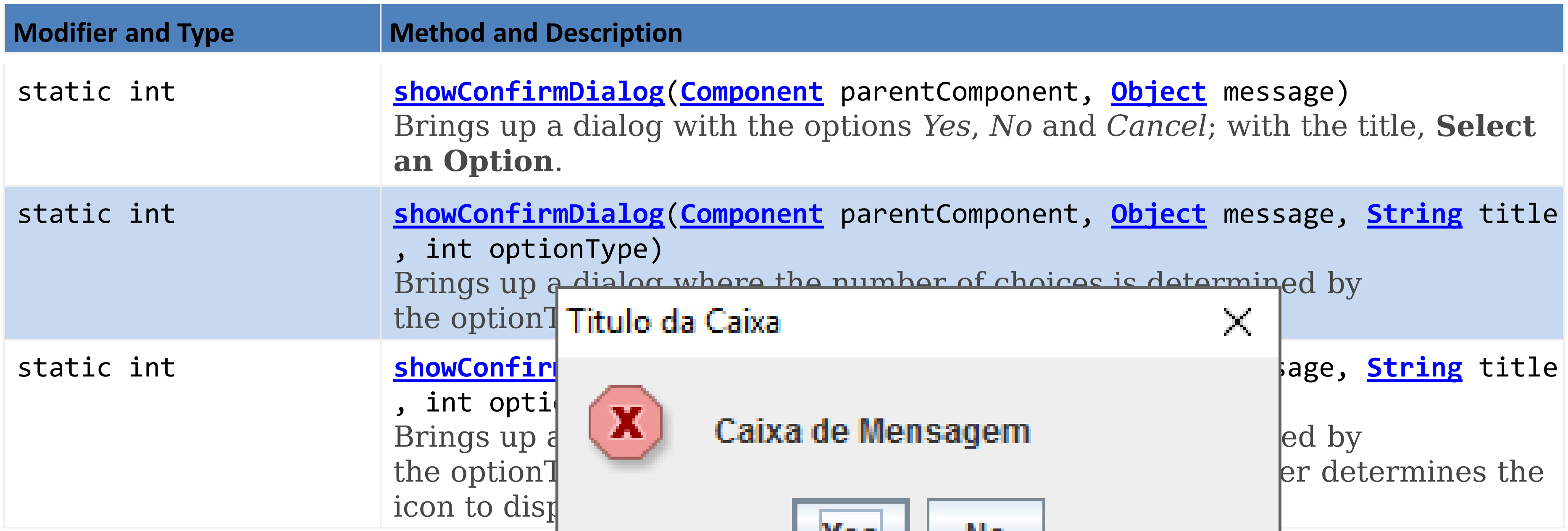
Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType)` Brings up a dialog where the number of choices is determined by the optionType parameter.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType, int messageType)` Brings up a dialog where the number of choices is determined by the optionType parameter, where the messageTypeparameter determines the icon to display.

messageType – O tipo da mensagem a ser exibida.

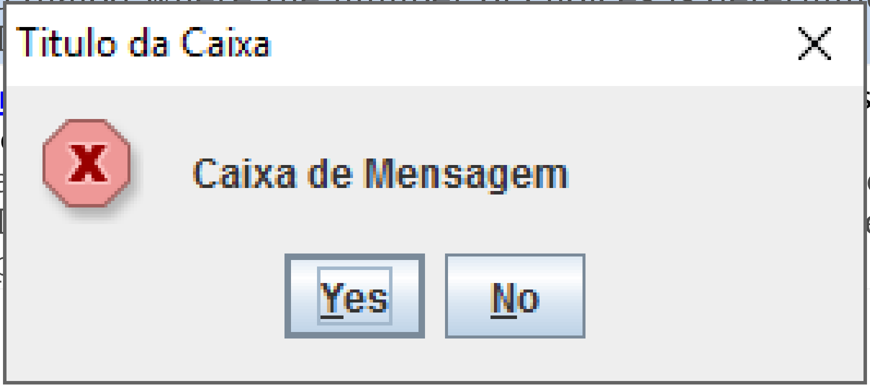
Tais valores tratam-se de constantes definidas na própria classe JOptionPane e são eles:

- `JOptionPane.ERROR_MESSAGE`
- `JOptionPane.INFORMATION_MESSAGE`
- `JOptionPane.WARNING_MESSAGE`
- `JOptionPane.QUESTION_MESSAGE`

Métodos de Classe showConfirmDialog



Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType)` Brings up a dialog where the number of choices is determined by the optionT
`static int`	`showConfir` `sage, String title` `, int optic` Brings up a ed by the optionT er determines the icon to disp

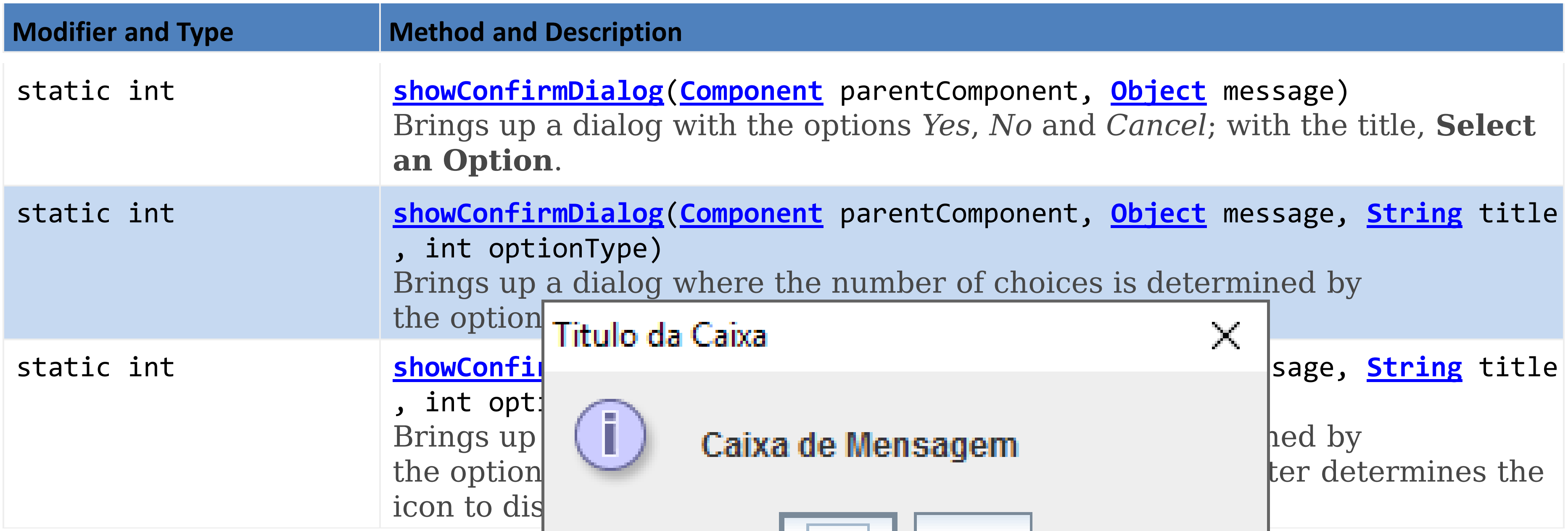


Titulo da Caixa

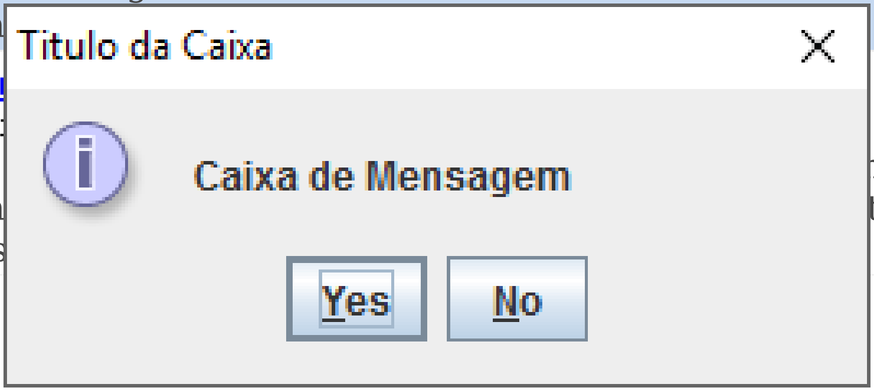
Caixa de Mensagem

```
retorno = JOptionPane.showConfirmDialog(null, "Caixa de Mensagem",
"Titulo da Caixa", JOptionPane.YES_NO_OPTION,
JOptionPane.ERROR_MESSAGE);
```

Métodos de Classe showConfirmDialog



Modifier and Type	Method and Description
static int	showConfirmDialog(Component parentComponent, Object message) Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
static int	showConfirmDialog(Component parentComponent, Object message, String title , int optionType) Brings up a dialog where the number of choices is determined by the option
static int	showConfi sage, String title , int opt Brings up ned by the option ter determines the icon to dis

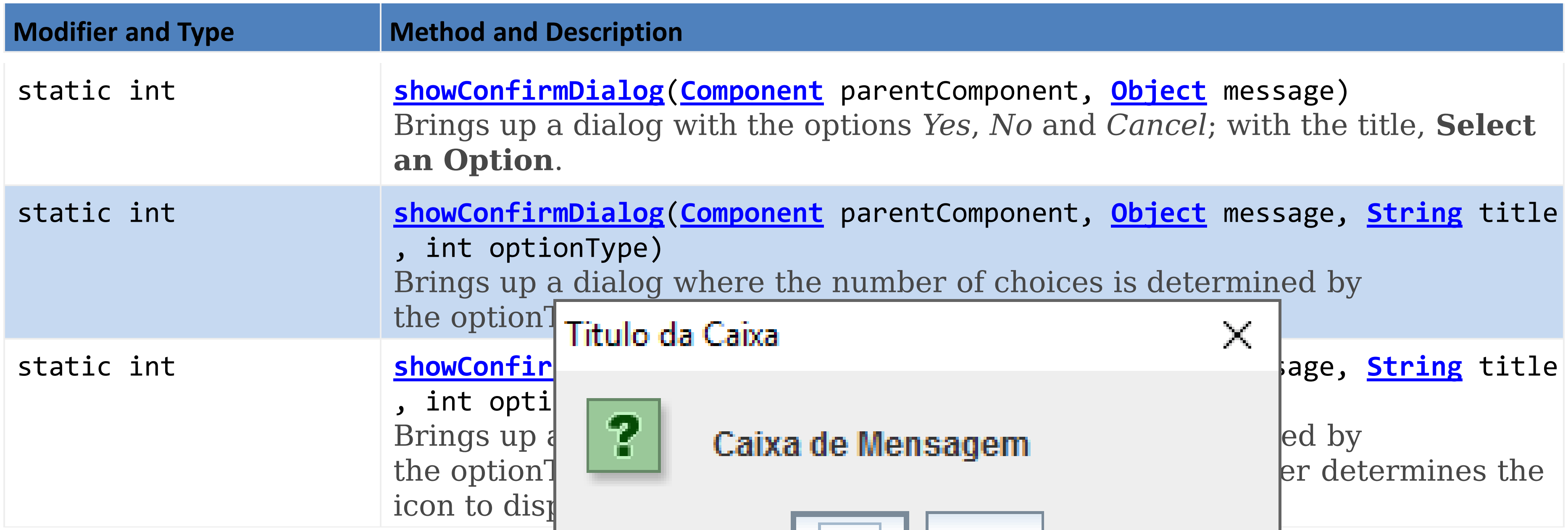


Titulo da Caixa

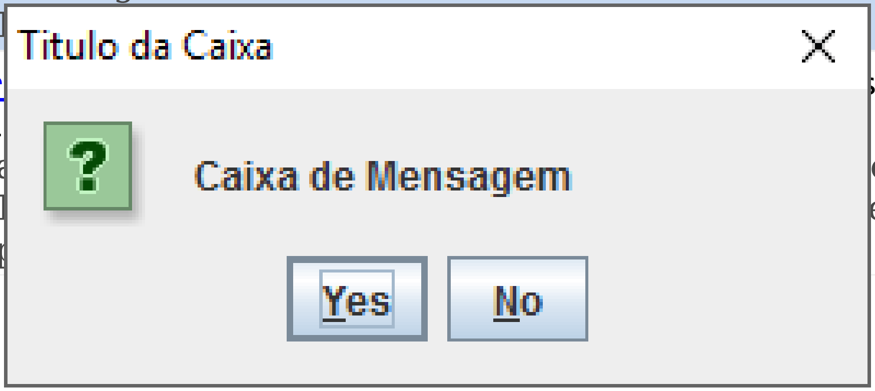
Caixa de Mensagem

```
retorno = JOptionPane.showConfirmDialog(null, "Caixa de Mensagem",
"Titulo da Caixa", JOptionPane.YES_NO_OPTION,
JOptionPane.INFORMATION_MESSAGE);
```

Métodos de Classe showConfirmDialog



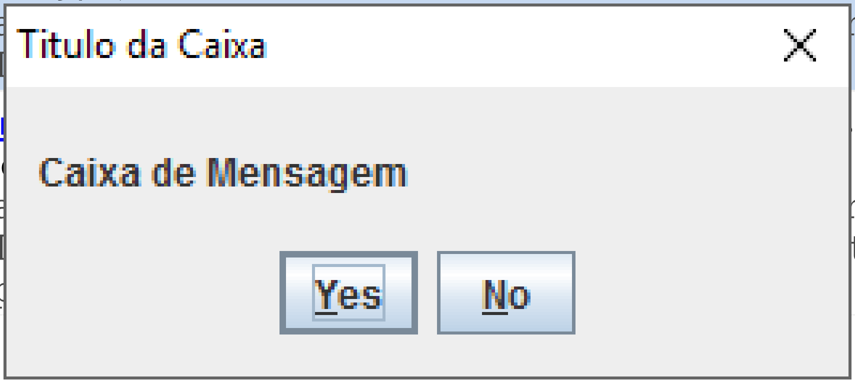
Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType)` Brings up a dialog where the number of choices is determined by the optionT
`static int`	`showConfir` `sage, String title` `, int opti` Brings up a ed by the optionT er determines the icon to disp



```
retorno = JOptionPane.showConfirmDialog(null, "Caixa de Mensagem",
"Titulo da Caixa", JOptionPane.YES_NO_OPTION,
JOptionPane.QUESTION_MESSAGE);
```

Métodos de Classe showConfirmDialog

Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType)` Brings up a … hed by the optionT…
`static int`	`showConfir… sage, String title, int optic…` Brings up a … hed by the optionT… ter determines the icon to disp…


Titulo da Caixa

Caixa de Mensagem

```
retorno = JOptionPane.showConfirmDialog(null, "Caixa de Mensagem",
"Titulo da Caixa", JOptionPane.YES_NO_OPTION,
JOptionPane.PLAIN_MESSAGE);
```

Métodos de Classe showConfirmDialog

Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType)` Brings up a dialog where the number of choices is determined by the optionType parameter.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType, int messageType)` Brings up a dialog where the number of choices is determined by the optionType parameter, where the messageTypeparameter determines the icon to display.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon)` Brings up a dialog with a specified icon, where the number of choices is determined by the optionType parameter.

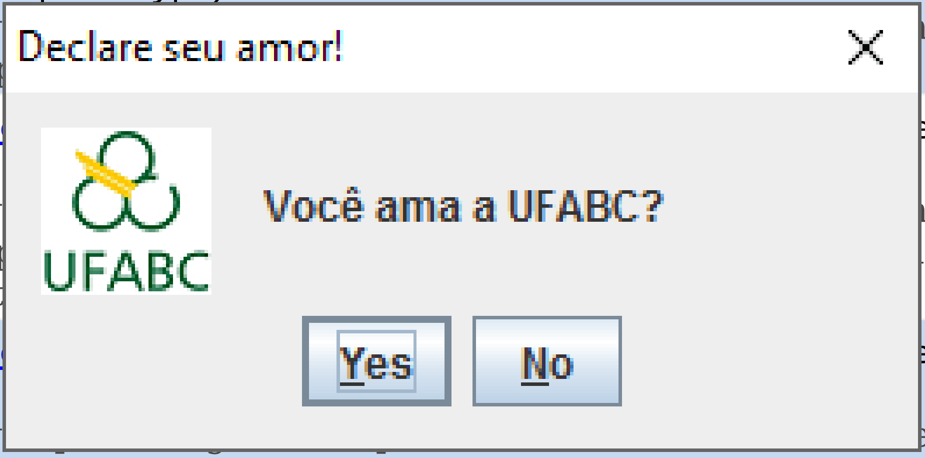
Métodos de Classe showConfirmDialog

Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType)` Brings up a dialog where the number of choices is determined by the optionType parameter.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType, int messageType)` Brings up a dialog where the number of choices is determined by the optionType parameter, where the messageTypeparameter determines the icon to display.
`static int`	`showConfirmDialog(Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon)` Brings up a dialog with a specified icon, where the number of choices is determined by the optionType parameter.

icon – Icone para ser exibido na caixa de mensagem.

Métodos de Classe showConfirmDialog

Modifier and Type	Method and Description
static int	showConfirmDialog(Component parentComponent, Object message) Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.
static int	showConfirmDialog(Component parentComponent, Object message, String title, int optionType) Bring ined by the op
static int	showC essage, String title , int Bring ined by the op eter determines the icon t
static int	showC essage, String title , int Bring er of choices is determined by the optionType parameter.


Declare seu amor!

Você ama a UFABC?

```
ImageIcon icone = new ImageIcon("ufabcicone.png");
retorno = JOptionPane.showConfirmDialog(null, "Você ama a UFABC?",
"Declare seu amor!", JOptionPane.YES_NO_OPTION,
JOptionPane.PLAIN_MESSAGE, icone);
```

Caixa de Mensagens

- É o compomente mais simples do pacote Swing.
- Está definido na classe JOptionPane.
- Para exibir há um conjunto de métodos de classe:

Nome do Método	Descrição
`showConfirmDialog`	Exibe uma caixa de mensagens de confirmação, tais como: YES, NO, CANCEL.
`showInputDialog`	Exibe uma caixa de mensagens para entrada.
`showMessageDialog`	Exibe uma caixa de mensagens com uma informação.
`showOptionDialog`	Uma união das três anteriores.

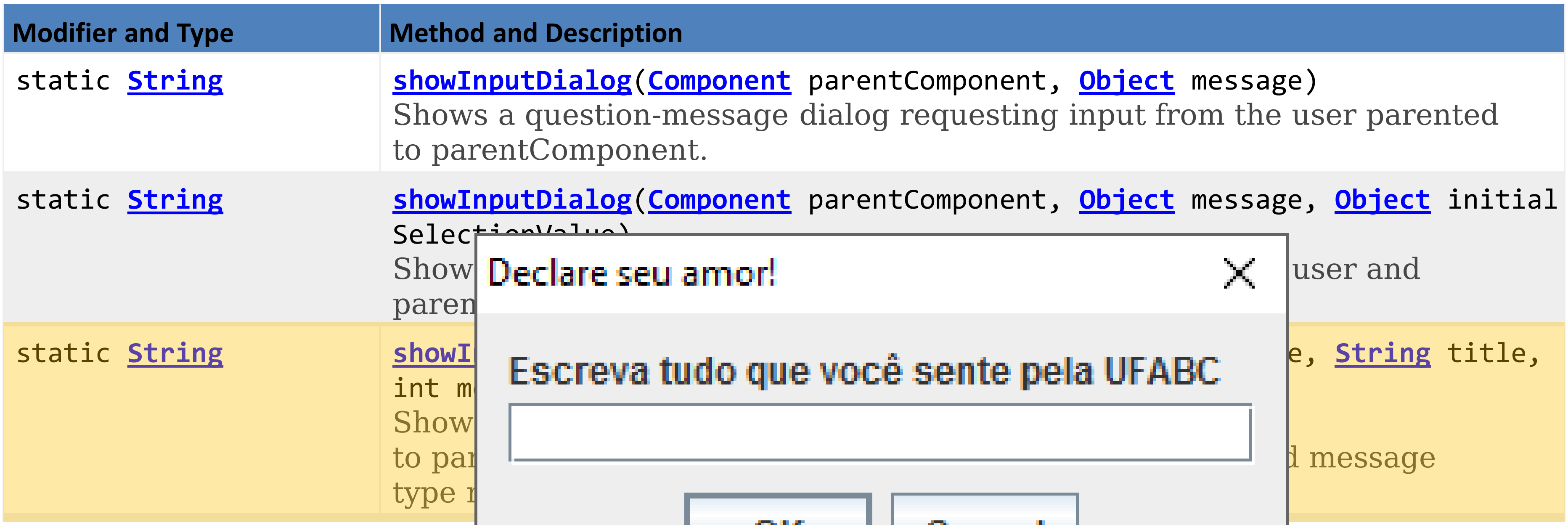
Métodos de Classe showInputDialog

Modifier and Type	Method and Description
static String	showInputDialog(Component parentComponent, Object message) Shows a question-message dialog requesting input from the user parented to parentComponent.
static String	showInputDialog(Component parentComponent, Object message, Object initialSelectionValue) Shows a question-message dialog requesting input from the user and parented to parentComponent.
static String	showInputDialog(Component parentComponent, Object message, String title, int messageType) Shows a dialog requesting input from the user parented to parentComponent with the dialog having the title title and message type messageType.
static Object	showInputDialog(Component parentComponent, Object message, String title, int messageType, Icon icon, Object[] selectionValues, Object initialSelectionValue) Prompts the user for input in a blocking dialog where the initial selection, possible selections, and all other options can be specified.
static String	showInputDialog(Object message) Shows a question-message dialog requesting input from the user.
static String	showInputDialog(Object message, Object initialSelectionValue) Shows a question-message dialog requesting input from the user, with the input value initialized to initialSelectionValue.

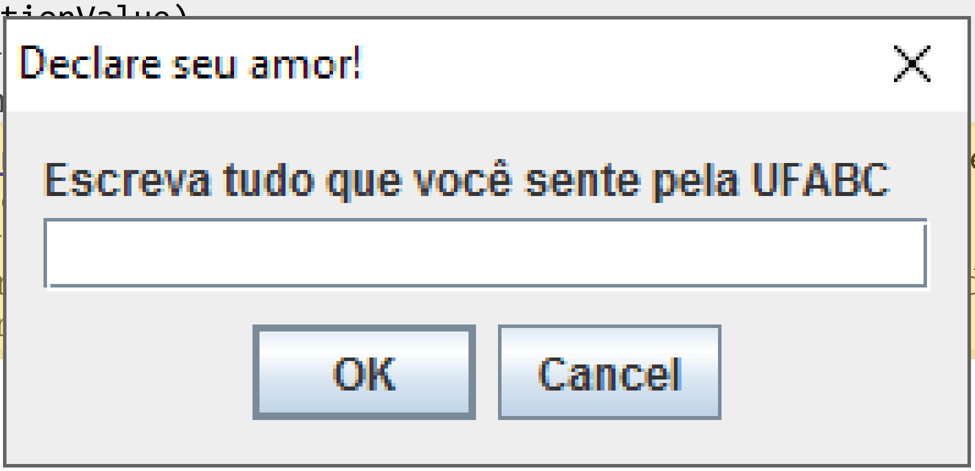
Métodos de Classe showInputDialog

Modifier and Type	Method and Description
static String	showInputDialog(Component parentComponent, Object message) Shows a question-message dialog requesting input from the user parented to parentComponent.
static String	showInputDialog(Component parentComponent, Object message, Object initialSelectionValue) Shows a question-message dialog requesting input from the user and parented to parentComponent.
static String	showInputDialog(Component parentComponent, Object message, String title, int messageType) Shows a dialog requesting input from the user parented to parentComponent with the dialog having the title title and message type messageType.
static Object	showInputDialog(Component parentComponent, Object message, String title, int messageType, Icon icon, Object[] selectionValues, Object initialSelectionValue) Prompts the user for input in a blocking dialog where the initial selection, possible selections, and all other options can be specified.
static String	showInputDialog(Object message) Shows a question-message dialog requesting input from the user.
static String	showInputDialog(Object message, Object initialSelectionValue) Shows a question-message dialog requesting input from the user, with the input value initialized to initialSelectionValue.

Métodos de Classe showInputDialog



Modifier and Type	Method and Description
static String	showInputDialog(Component parentComponent, Object message) Shows a question-message dialog requesting input from the user parented to parentComponent.
static String	showInputDialog(Component parentComponent, Object message, Object initial SelectionValue) Show user and paren
static String	showI e, String title, int m Show to par d message type r

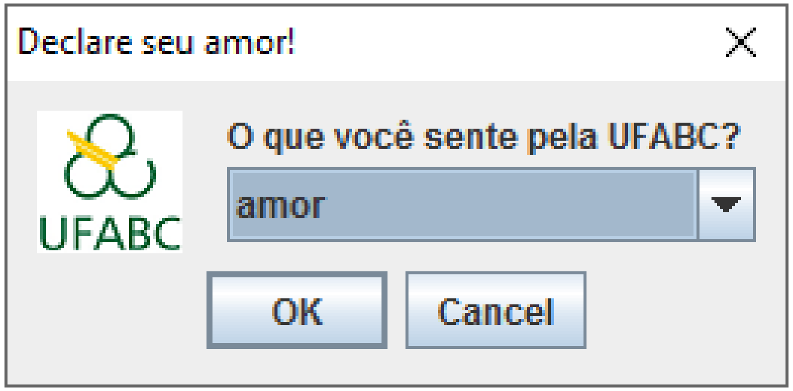


```
retorno = JOptionPane.showInputDialog(null, "Escreva tudo que você
sente pela UFABC", "Declare seu amor!", JOptionPane.PLAIN_MESSAGE);
```

Métodos de Classe showInputDialog

Modifier and Type	Method and Description
static String	showInputDialog(Component parentComponent, Object message) Shows a question-message dialog requesting input from the user parented to parentComponent.
static String	showInputDialog(Component parentComponent, Object message, Object initialSelectionValue) Shows a question-message dialog requesting input from the user and parented to parentComponent.
static String	showInputDialog(Component parentComponent, Object message, String title, int messageType) Shows a dialog requesting input from the user parented to parentComponent with the dialog having the title title and message type messageType.
static Object	showInputDialog(Component parentComponent, Object message, String title, int messageType, Icon icon, Object[] selectionValues, Object initialSelectionValue) Prompts the user for input in a blocking dialog where the initial selection, possible selections, and all other options can be specified.
static String	showInputDialog(Object message) Shows a question-message dialog requesting input from the user.
static String	showInputDialog(Object message, Object initialSelectionValue) Shows a question-message dialog requesting input from the user, with the input value initialized to initialSelectionValue.

Métodos de Classe showInputDialog


Declare seu amor!

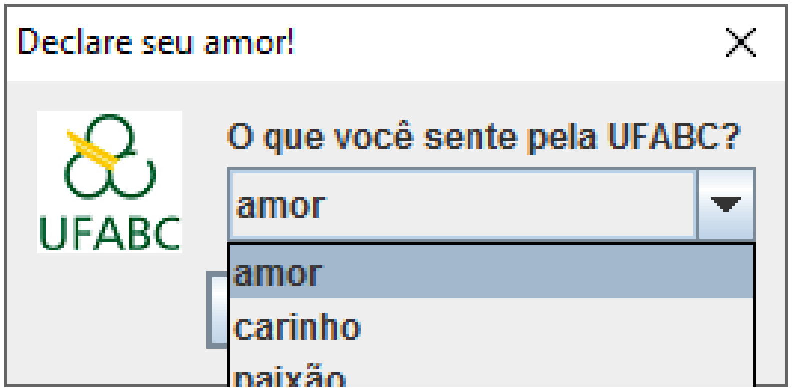
O que você sente pela UFABC?

amor

```
String opcoes[] = {"amor", "carinho", "paixão", "loteria"};
ImageIcon icone = new ImageIcon("ufabcicone.png");
retorno = (String) JOptionPane.showInputDialog(null, "O que você
sente pela UFABC?", "Declare seu amor!",
JOptionPane.QUESTION_MESSAGE, icone, opcoes, opcoes[0]);
```

static Object	showInputDialog(Component parentComponent, Object message, String title, int messageType, Icon icon, Object[] selectionValues, Object initialSelectionValue) Prompts the user for input in a blocking dialog where the initial selection, possible selections, and all other options can be specified.
static String	showInputDialog(Object message) Shows a question-message dialog requesting input from the user.
static String	showInputDialog(Object message, Object initialSelectionValue) Shows a question-message dialog requesting input from the user, with the input value initialized to initialSelectionValue.

Métodos de Classe showInputDialog



```
String opcoes[] = {"amor", "carinho", "paixão", "loteria"};
ImageIcon icone = new ImageIcon("ufabcicone.png");
retorno = (String) JOptionPane.showInputDialog(null, "O que você
sente pela UFABC?", "Declare seu amor!",
JOptionPane.QUESTION_MESSAGE, icone, opcoes, opcoes[0]);
```

`static Object`	`showInputDialog(Component parentComponent, Object message, String title, int messageType, Icon icon, Object[] selectionValues, Object initialSelectionValue)` Prompts the user for input in a blocking dialog where the initial selection, possible selections, and all other options can be specified.
`static String`	`showInputDialog(Object message)` Shows a question-message dialog requesting input from the user.
`static String`	`showInputDialog(Object message, Object initialSelectionValue)` Shows a question-message dialog requesting input from the user, with the input value initialized to initialSelectionValue.

Caixa de Mensagens

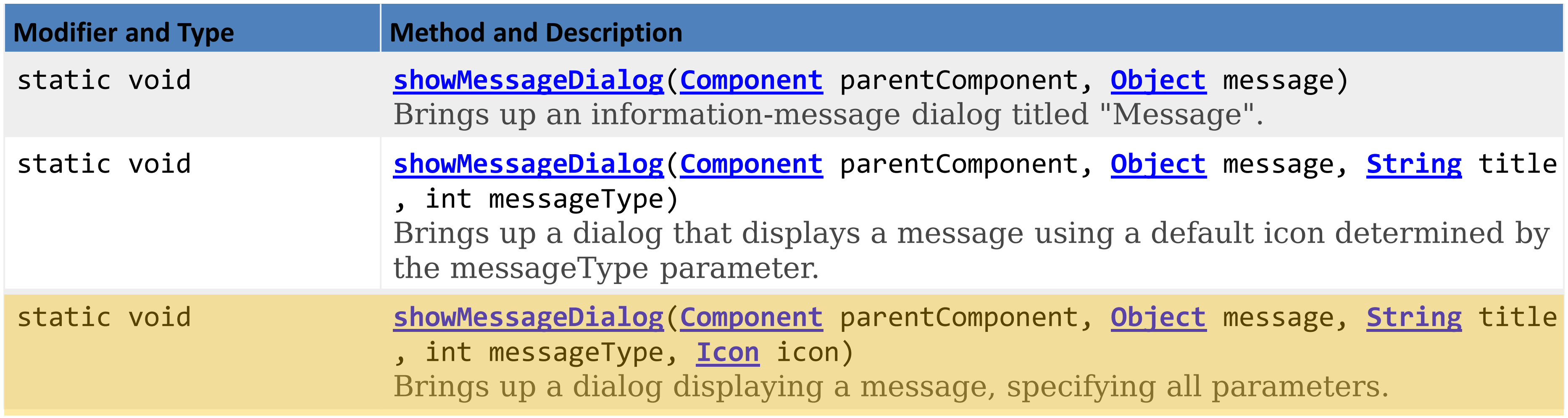
- É o compomente mais simples do pacote Swing.
- Está definido na classe JOptionPane.
- Para exibir há um conjunto de métodos de classe:

Nome do Método	**Descrição**
`showConfirmDialog`	Exibe uma caixa de mensagens de confirmação, tais como: YES, NO, CANCEL.
`showInputDialog`	Exibe uma caixa de mensagens para entrada.
`showMessageDialog`	Exibe uma caixa de mensagens com uma informação.
`showOptionDialog`	Uma união das três anteriores.

Métodos de Classe showMessageDialog

Modifier and Type	Method and Description
`static void`	`showMessageDialog(Component parentComponent, Object message)` Brings up an information-message dialog titled "Message".
`static void`	`showMessageDialog(Component parentComponent, Object message, String title, int messageType)` Brings up a dialog that displays a message using a default icon determined by the messageType parameter.
`static void`	`showMessageDialog(Component parentComponent, Object message, String title, int messageType, Icon icon)` Brings up a dialog displaying a message, specifying all parameters.

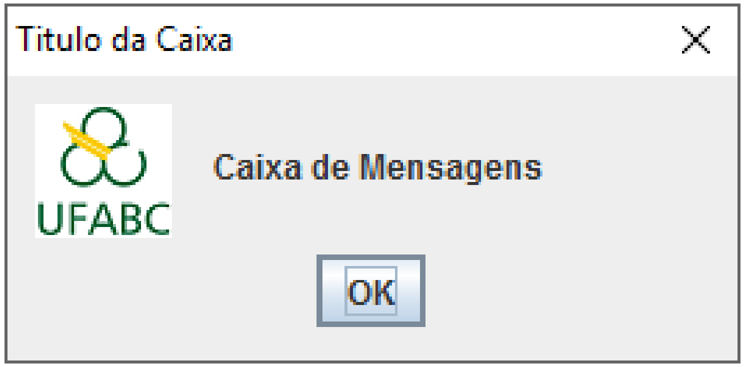
Métodos de Classe showMessageDialog



Modifier and Type	Method and Description
`static void`	`showMessageDialog(Component parentComponent, Object message)` Brings up an information-message dialog titled "Message".
`static void`	`showMessageDialog(Component parentComponent, Object message, String title, int messageType)` Brings up a dialog that displays a message using a default icon determined by the messageType parameter.
`static void`	`showMessageDialog(Component parentComponent, Object message, String title, int messageType, Icon icon)` Brings up a dialog displaying a message, specifying all parameters.

Métodos de Classe showMessageDialog

Modifier and Type	Method and Description
`static void`	`showMessageDialog(Component parentComponent, Object message)` Brings up an information-message dialog titled "Message".
`static void`	`showMessageDialog(Component parentComponent, Object message, String title, int messageType)` Brings up a dialog that displays a message using a default icon determined by the messageType parameter.
`static void`	`showMessageDialog(Component parentComponent, Object message, String title, int messageType, Icon icon)` Brings up a dialog displaying a message, specifying all parameters.



```
ImageIcon icone = new ImageIcon("ufabcicone.png");
JOptionPane.showMessageDialog(null, "Caixa de Mensagens", "Titulo da
Caixa", JOptionPane.PLAIN_MESSAGE, icone);
```

Caixa de Mensagens

- É o compomente mais simples do pacote Swing.
- Está definido na classe JOptionPane.
- Para exibir há um conjunto de métodos de classe:

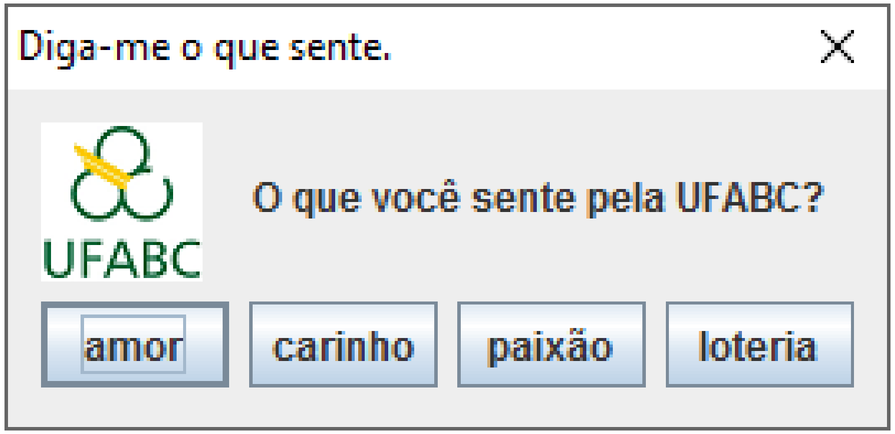
Nome do Método	**Descrição**
`showConfirmDialog`	Exibe uma caixa de mensagens de confirmação, tais como: YES, NO, CANCEL.
`showInputDialog`	Exibe uma caixa de mensagens para entrada.
`showMessageDialog`	Exibe uma caixa de mensagens com uma informação.
`showOptionDialog`	Uma união das três anteriores.

Método de Classe showOptionDialog

Modifier and Type	Method and Description
`static int`	`showOptionDialog(Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon, Object[] options, Object initialValue)` Brings up a dialog with a specified icon, where the initial choice is determined by the initialValue parameter and the number of choices is determined by the optionType parameter.

Método de Classe showOptionDialog

Modifier and Type	Method and Description
`static int`	`showOptionDialog(Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon, Object[] options, Object initialValue)` Brings up a dialog with a specified icon, where the initial choice is determined by the initialValue parameter and the number of choices is determined by the optionType parameter.


Diga-me o que sente.

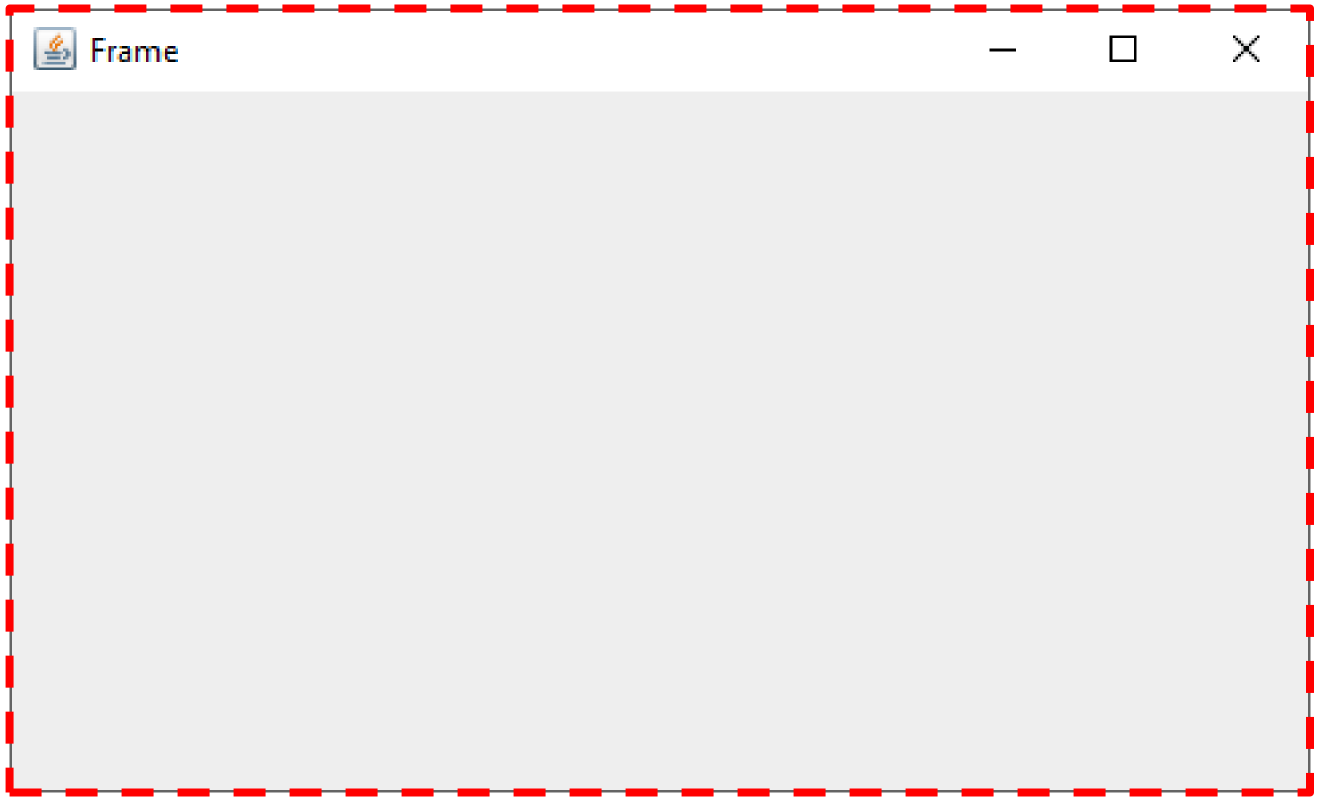
O que você sente pela UFABC?

```
String []opcoes = {"amor", "carinho", "paixão", "loteria"};
ImageIcon icone = new ImageIcon("ufabcicone.png");
retorno = JOptionPane.showOptionDialog(null, "O que você sente pela
UFABC?", "Diga-me o que sente.", JOptionPane.DEFAULT_OPTION,
JOptionPane.PLAIN_MESSAGE, icone, opcoes, opcoes[0]);
```

Componentes e Containers

- Uma interface gráfica baseia-se em dois elementos:
  - **Componentes**: Botões, Labels, Caixas de Texto etc.
  - **Containers**: Recipientes que agrupam componentes.
- Todo programa em JAVA com uma interface gráfica obrigatóriamente possui um Container.
- Uma janela é um **Frame** (awt) ou **JFrame** (swing)
- Um objeto do tipo **JFrame** é um Container, ou seja, ele agrupa vários componentes GUI.

A Classe JFrame



A Classe JFrame

- Parte da Classe JFrame:

Constructor and Description
`JFrame()` Constructs a new frame that is initially invisible.
`JFrame(GraphicsConfiguration gc)` Creates a Frame in the specified GraphicsConfiguration of a screen device and a blank title.
`JFrame(String title)` Creates a new, initially invisible Frame with the specified title.
`JFrame(String title, GraphicsConfiguration gc)` Creates a JFrame with the specified title and the specified GraphicsConfiguration of a screen device.

A Classe JFrame

- Parte da Classe JFrame:

Methods inherited from class java.awt.Window
addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBackground, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getIconImages, getInputContext, getListeners, getLocale, getModalExclusionType, getMostRecentFocusOwner, getOpacity, getOwnedWindows, getOwner, getOwnerlessWindows, getShape, getToolkit, getType, getWarningString, getWindowFocusListeners, getWindowListeners, getWindows, getWindowStateListeners, hide, isActive, isAlwaysOnTop, isAlwaysOnTopSupported, isAutoRequestFocus, isFocusableWindow, isFocusCycleRoot, isFocused, isLocationByPlatform, isOpaque, isShowing, isValidateRoot, pack, paint, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, reshape, setAlwaysOnTop, setAutoRequestFocus, setBounds, setBounds, setCursor, setFocusableWindowState, setFocusCycleRoot, setIconImages, setLocation, setLocation, setLocationByPlatform, setLocationRelativeTo, setMinimumSize, setModalExclusionType, setSize, setSize, setType, setVisible, show, toBack, toFront

A Classe JFrame

- Parte da Classe JFrame:

Methods inherited from class java.awt.Window

addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBackground, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getIconImages, getInputContext, getListeners, getLocale, getModalExclusionType, getMostRecentFocusOwner, getOpacity, getOwnedWindows, getOwner, getOwnerlessWindows, getShape, getToolkit, getType, getWarningString, getWindowFocusListeners, getWindowListeners, getWindows, getWindowStateListeners, hide, isActive, isAlwaysOnTop, isAlwaysOnTopSupported, isAutoRequestFocus, isFocusableWindow, isFocusCycleRoot, isFocused, isLocationByPlatform, isOpaque, isShowing, isValidateRoot, pack, paint, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, reshape, setAlwaysOnTop, setAutoRequestFocus, setBounds, setBounds, setCursor, setFocusableWindowState, setFocusCycleRoot, setIconImages, setLocation, setLocation, setLocationByPlatform, setLocationRelativeTo, setMinimumSize, setModalExclusionType, setSize, setSize, setType, setVisible, show, toBack, toFront

A Classe JFrame

- Parte da Classe JFrame herdado de awt.Window:

Modifier and Type	Method and Description
void	`addWindowListener(WindowListener l)` Adds the specified window listener to receive window events from this window.
void	`setVisible(boolean b)` Shows or hides this Window depending on the value of parameter b.
void	`setSize(int width, int height)` Resizes this component so that it has width width and height height.

- Parte da Classe JFrame:

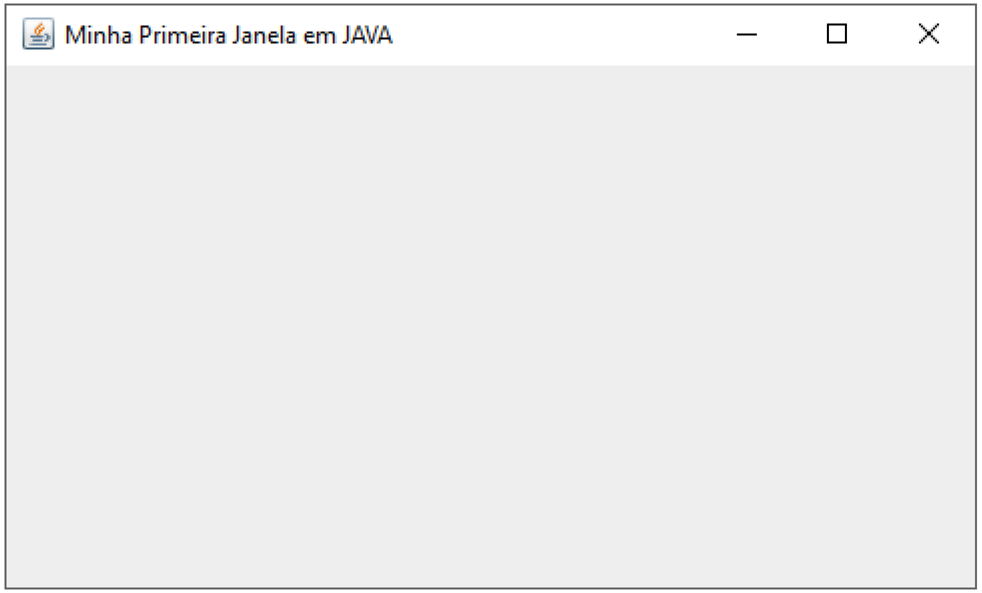
Modifier and Type	Method and Description
void	`setDefaultCloseOperation(int operation)` Sets the operation that will happen by default when the user initiates a "close" on this frame.

JFrame: Primeira Janela em JAVA

```java
import javax.swing.JFrame;

public class PrimeiraJanela
{
        public static void main(String[] args)
        {
                JFrame janela = new JFrame("Minha Primeira Janela em JAVA");
                janela.setSize(500, 300);
                janela.setVisible(true);
        }
}
```

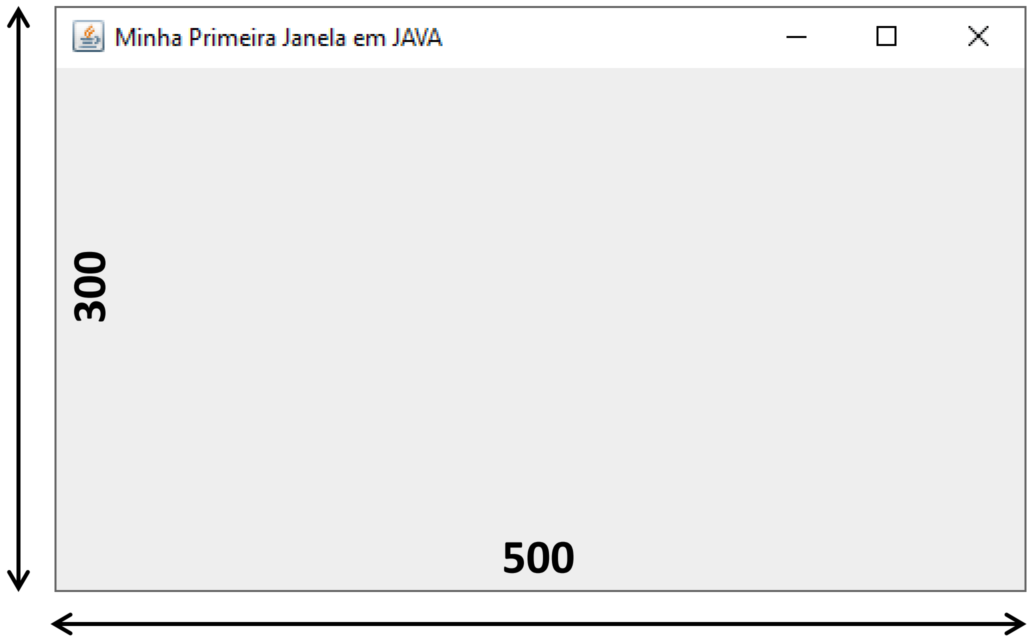
JFrame: Primeira Janela em JAVA

```
import javax.swing.JFrame;

public class PrimeiraJanela
{
	public static void main(String[] args)
	{
		JFrame janela = new JFrame("Minha Primeira Janela em JAVA");
		janela.setSize(500, 300);
		janela.setVisible(true);
	}
}
```



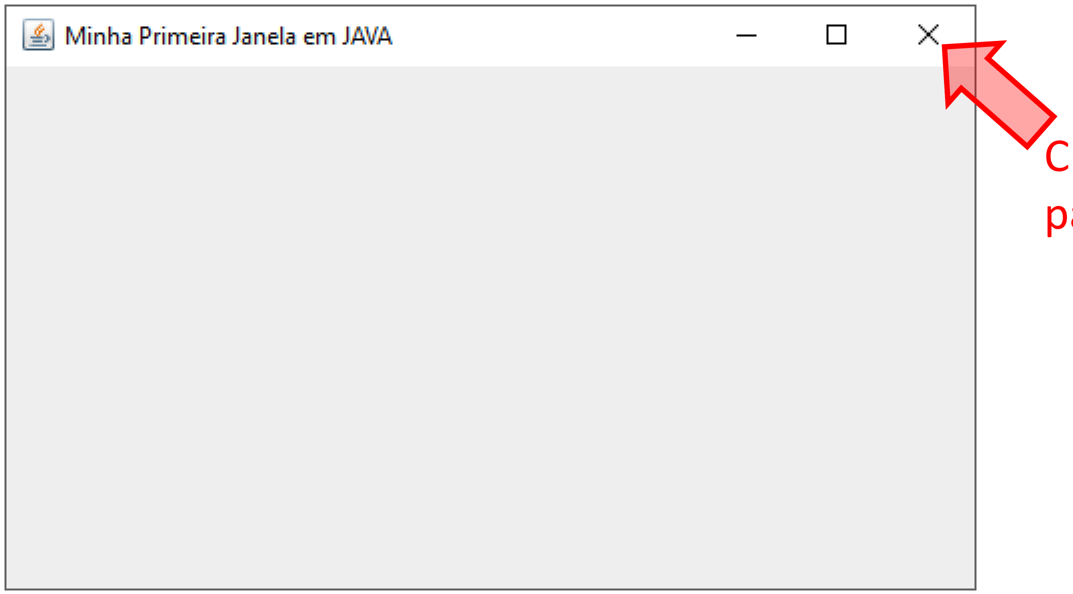
JFrame: Primeira Janela em JAVA

```
import javax.swing.JFrame;

public class PrimeiraJanela
{
        public static void main(String[] args)
        {
                JFrame janela = new JFrame("Minha Primeira Janela em JAVA");
                janela.setSize(500, 300);
                janela.setVisible(true);
        }
}
```



JFrame: Primeira Janela em JAVA

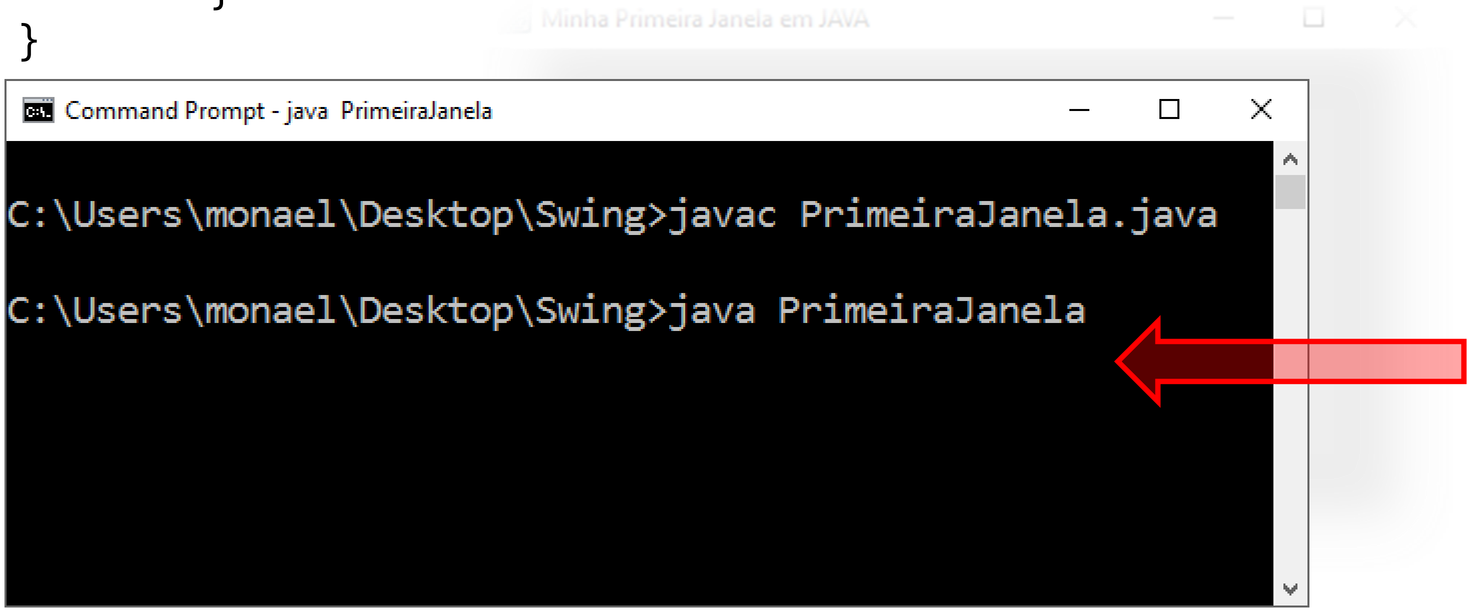
```
import javax.swing.JFrame;

public class PrimeiraJanela
{
        public static void main(String[] args)
        {
                JFrame janela = new JFrame("Minha Primeira Janela em JAVA");
                janela.setSize(500, 300);
                janela.setVisible(true);
        }
}
```



Clicando sobre o X
para fechar

JFrame: Primeira Janela em JAVA

```
import javax.swing.JFrame;

public class PrimeiraJanela
{
        public static void main(String[] args)
        {
                JFrame janela = new JFrame("Minha Primeira Janela em JAVA");
                janela.setSize(500, 300);
                janela.setVisible(true);
        }
}
```


Minha Primeira Janela em JAVA
Command Prompt - java PrimeiraJanela

```
C:\Users\monael\Desktop\Swing>javac PrimeiraJanela.java

C:\Users\monael\Desktop\Swing>java PrimeiraJanela
```

Programa ainda executando no Console.

Fechando a Minha Primeira Janela

- Por padrão, quando clica-se no botão “X” de um objeto **JFrame**, ele é escondido, mas a aplicação permanece executando.
- Para modificar esse comportamento padrão, devemos alterá-lo através do método `setDefaultCloseOperation(int operation)`
- Os possíveis valores para o argumento operation são:

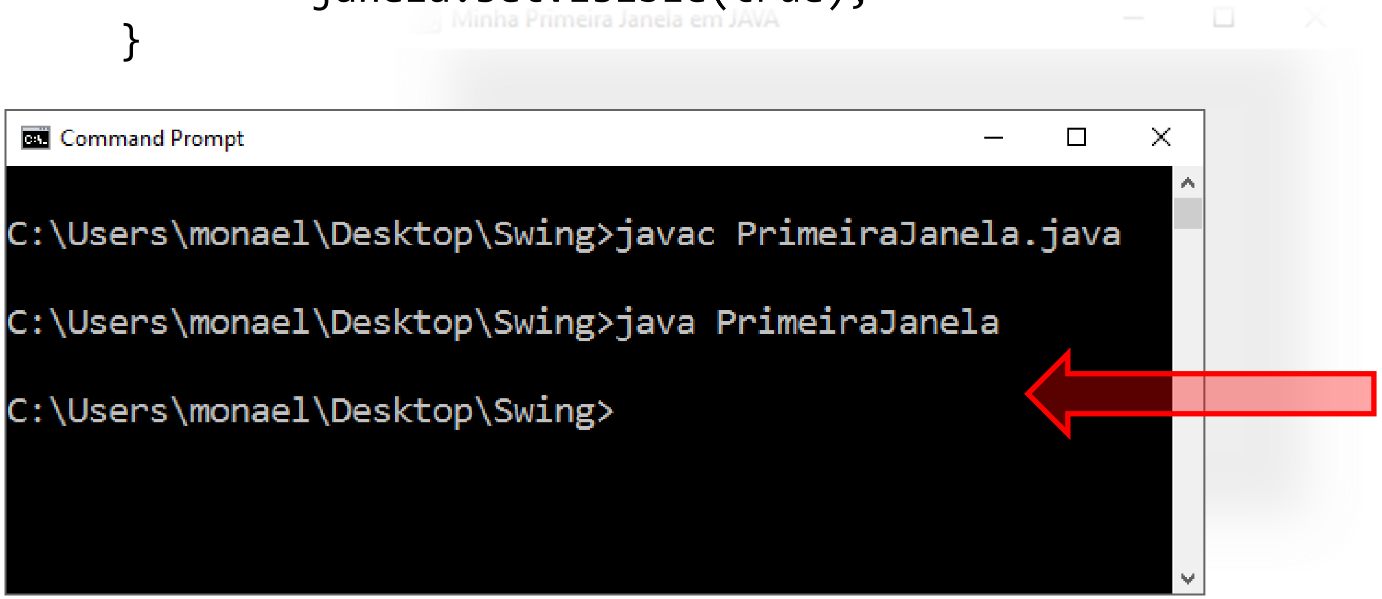
Constantes (JFrame)	**Descrição**
`DO_NOTHING_ON_CLOSE`	Não faz nada; invoca o método windowClosing() se registrado no frame.
`HIDE_ON_CLOSE`	Esconde o frame e mantem seus recursos alocados.
`DISPOSE_ON_CLOSE`	Esconde e elimina o frame e seus recursos alocados.
`EXIT_ON_CLOSE`	Invoca o método exit de System, ou seja, finaliza a aplicação. (Uso exclusivo em Aplicações)

Fechando a Minha Primeira Janela

```
import javax.swing.JFrame;

public class PrimeiraJanela
{
        public static void main(String[] args)
        {
                JFrame janela = new JFrame("Minha Primeira Janela em JAVA");
                janela.setSize(500, 300);
                janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                janela.setVisible(true);
        }
}
```

Fechando a Minha Primeira Janela

```
import javax.swing.JFrame;

public class PrimeiraJanela
{
        public static void main(String[] args)
        {
                JFrame janela = new JFrame("Minha Primeira Janela em JAVA");
                janela.setSize(500, 300);
                janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                janela.setVisible(true);
        }
}
```



```
C:\Users\monael\Desktop\Swing>javac PrimeiraJanela.java

C:\Users\monael\Desktop\Swing>java PrimeiraJanela

C:\Users\monael\Desktop\Swing>
```

Ao clicar em “X”, o programa é encerrado.

Fechando a Minha Primeira Janela

- Caso deseje realizar algo no momento que o usuário clicar em “X”, então deve-se seguir os 3 passos:

1. Primeiramente, deve-se codificar uma classe que herde da Classe **WindowAdapter** ou que implemente **WindowListener**.
2. Depois deve-se sobrescrever o método `windowClosing()` herdado ou implementar todos os descritos na interface.
3. Finalmente, deve-se instanciar um objeto dessa classe e registrá-lo ao objeto **JFrame** através do método `addWindowListener(…)` e alterar o comportamento padrão para DO_NOTHING_ON_CLOSE.

A Classe WindowAdapter

- Passo 1: codificar uma classe que herde da classe **WindowAdapter**.

Modifier and Type	Method and Description
void	**windowActivated**(**WindowEvent** e) Invoked when a window is activated.
void	**windowClosed**(**WindowEvent** e) Invoked when a window has been closed.
void	**windowClosing**(**WindowEvent** e) Invoked when a window is in the process of being closed.
void	**windowDeactivated**(**WindowEvent** e) Invoked when a window is de-activated.
void	**windowDeiconified**(**WindowEvent** e) Invoked when a window is de-iconified.
void	**windowGainedFocus**(**WindowEvent** e) Invoked when the Window is set to be the focused Window, which means that the Window, or one of its subcomponents, will receive keyboard events.
void	**windowIconified**(**WindowEvent** e) Invoked when a window is iconified.
void	**windowLostFocus**(**WindowEvent** e) Invoked when the Window is no longer the focused Window, which means that keyboard events will no longer be delivered to the Window or any of its subcomponents.
void	**windowOpened**(**WindowEvent** e) Invoked when a window has been opened.
void	**windowStateChanged**(**WindowEvent** e) Invoked when a window state is changed.

A Classe WindowAdapter

- Passo 1: codificar uma classe que herde da classe **WindowAdapter**.

Modifier and Type	Method and Description
void	**windowActivated**(**WindowEvent** e) Invoked when a window is activated.
void	**windowClosed**(**WindowEvent** e) Invoked when a window has been closed.
void	**windowClosing**(**WindowEvent** e) Invoked when a window is in the process of being closed.
void	**windowDeactivated**(**WindowEvent** e) Invoked when a window is de-activated.
void	**windowDeiconified**(**WindowEvent** e) Invoked when a window is de-iconified.
void	**windowGainedFocus**(**WindowEvent** e) Invoked when the Window is set to be the focused Window, which means that the Window, or one of its subcomponents, will receive keyboard events.
void	**windowIconified**(**WindowEvent** e) Invoked when a window is iconified.
void	**windowLostFocus**(**WindowEvent** e) Invoked when the Window is no longer the focused Window, which means that keyboard events will no longer be delivered to the Window or any of its subcomponents.
void	**windowOpened**(**WindowEvent** e) Invoked when a window has been opened.
void	**windowStateChanged**(**WindowEvent** e) Invoked when a window state is changed.

A Interface WindowListerner

- Passo 1: codificar uma classe que implemente a interface **WindowListener**.

Modifier and Type	Method and Description
`void`	`windowActivated(WindowEvent e)` Invoked when the Window is set to be the active Window.
`void`	`windowClosed(WindowEvent e)` Invoked when a window has been closed as the result of calling dispose on the window.
`void`	`windowClosing(WindowEvent e)` Invoked when the user attempts to close the window from the window's system menu.
`void`	`windowDeactivated(WindowEvent e)` Invoked when a Window is no longer the active Window.
`void`	`windowDeiconified(WindowEvent e)` Invoked when a window is changed from a minimized to a normal state.
`void`	`windowIconified(WindowEvent e)` Invoked when a window is changed from a normal to a minimized state.
`void`	`windowOpened(WindowEvent e)` Invoked the first time a window is made visible.

A Interface WindowListerner

- Passo 1: codificar uma classe que implemente a interface **WindowListener**.

Modifier and Type	Method and Description
`void`	`windowActivated(WindowEvent e)` Invoked when the Window is set to be the active Window.
`void`	`windowClosed(WindowEvent e)` Invoked when a window has been closed as the result of calling dispose on the window.
`void`	`windowClosing(WindowEvent e)` Invoked when the user attempts to close the window from the window's system menu.
`void`	`windowDeactivated(WindowEvent e)` Invoked when a Window is no longer the active Window.
`void`	`windowDeiconified(WindowEvent e)` Invoked when a window is changed from a minimized to a normal state.
`void`	`windowIconified(WindowEvent e)` Invoked when a window is changed from a normal to a minimized state.
`void`	`windowOpened(WindowEvent e)` Invoked the first time a window is made visible.

Fechando a Minha Primeira Janela

- Passo 1: codificar uma classe que implemente a interface **WindowListener**.
- Passo 2: sobrescrever os métodos da interface **WindowListener**.

```
class GestorJanela implements WindowListener
{
         public void windowActivated(WindowEvent e)
         {        }
         public void windowClosed(WindowEvent e)
         {        }

         public void windowClosing(WindowEvent e)
         {
             System.exit(0);
         }

         public void windowDeactivated(WindowEvent e)
         {        }
         public void windowDeiconified(WindowEvent e)
         {        }
         public void windowIconified(WindowEvent e)
         {        }
         public void windowOpened(WindowEvent e)
         {        }
}
```

Fechando a Minha Primeira Janela

- Passo 1: codificar uma classe que implemente a interface **WindowListener**.
- Passo 2: sobrescrever os métodos da interface **WindowListener**.

```
class GestorJanela implements WindowListener
{
           public void windowActivated(WindowEvent e)
           {          }
           public void windowClosed(WindowEvent e)
           {          }

           public void windowClosing(WindowEvent e)
           {
             if(JOptionPane.showConfirmDialog(null, "Deseja realmente sair?", "Sair",
                                         JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE)
                                       == JOptionPane.YES_OPTION)
             {
                   System.exit(0);
             }
           }

           public void windowDeactivated(WindowEvent e)
           {          }
           public void windowDeiconified(WindowEvent e)
           {          }
           public void windowIconified(WindowEvent e)
           {          }
           public void windowOpened(WindowEvent e)
           {          }
}
```

Acrescentando uma Caixa de Mensagem para confirmação

Fechando a Minha Primeira Janela

- Passo 3: instanciar um objeto dessa classe e registrá-lo ao objeto **JFrame**. E alterar o comportamento padrão para DO_NOTHING_ON_CLOSE.

```
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import java.awt.event.WindowListener;
import java.awt.event.WindowEvent;

public class PrimeiraJanela
{
        public static void main(String[] args)
        {
                JFrame janela = new JFrame("Minha Primeira Janela em JAVA");
                janela.setSize(500, 300);
                janela.addWindowListener(new GestorJanela());
                janela.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
                janela.setVisible(true);
        }
}
```

Fechando a Minha Primeira Janela

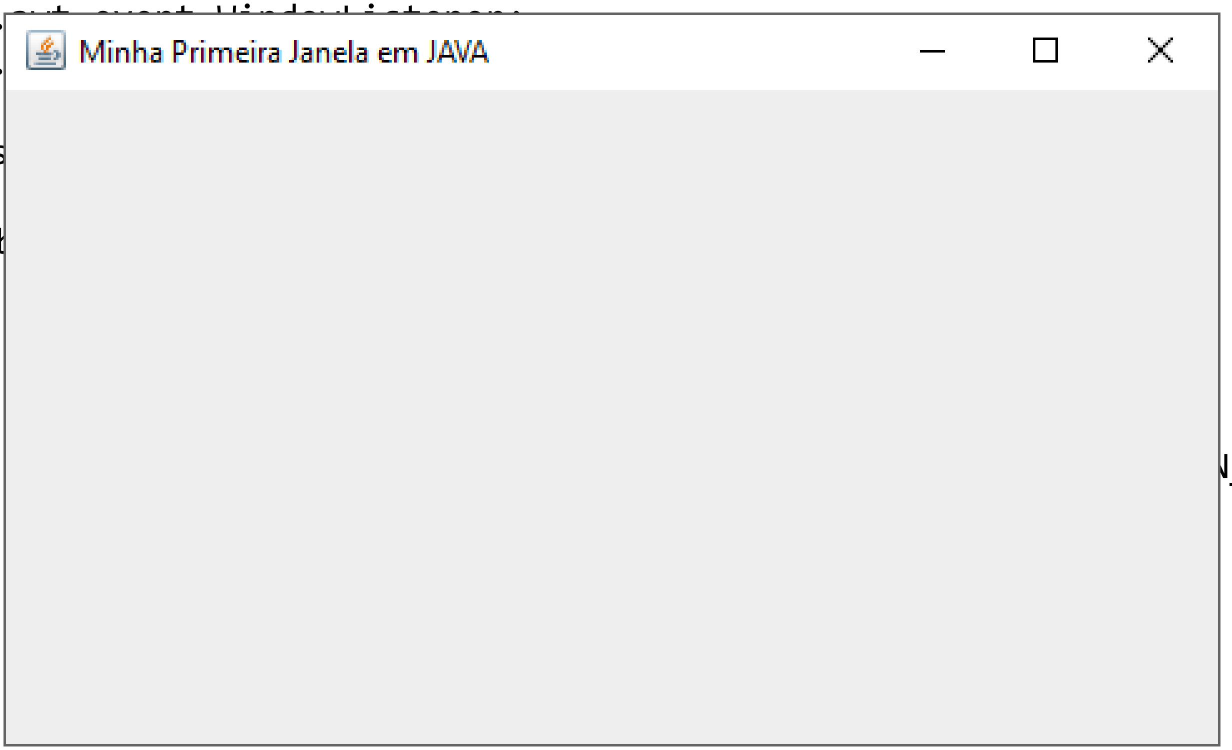
- Passo 3: instanciar um objeto dessa classe e registrá-lo ao objeto **JFrame**. E alterar o comportamento padrão para DO_NOTHING_ON_CLOSE.

```
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import java.awt.event.WindowListener;
import java.awt.event.WindowEvent;

public class PrimeiraJanela
{
	public static void main(String[] args)
	{
		JFrame janela = new JFrame("Minha Primeira Janela em JAVA");
		janela.setSize(500, 300);
		janela.addWindowListener(new GestorJanela());
		janela.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
		janela.setVisible(true);
	}
}
```

Fechando a Minha Primeira Janela

- Passo 3: instanciar um objeto dessa classe e registrá-lo ao objeto **JFrame**. E alterar o comportamento padrão para DO_NOTHING_ON_CLOSE.

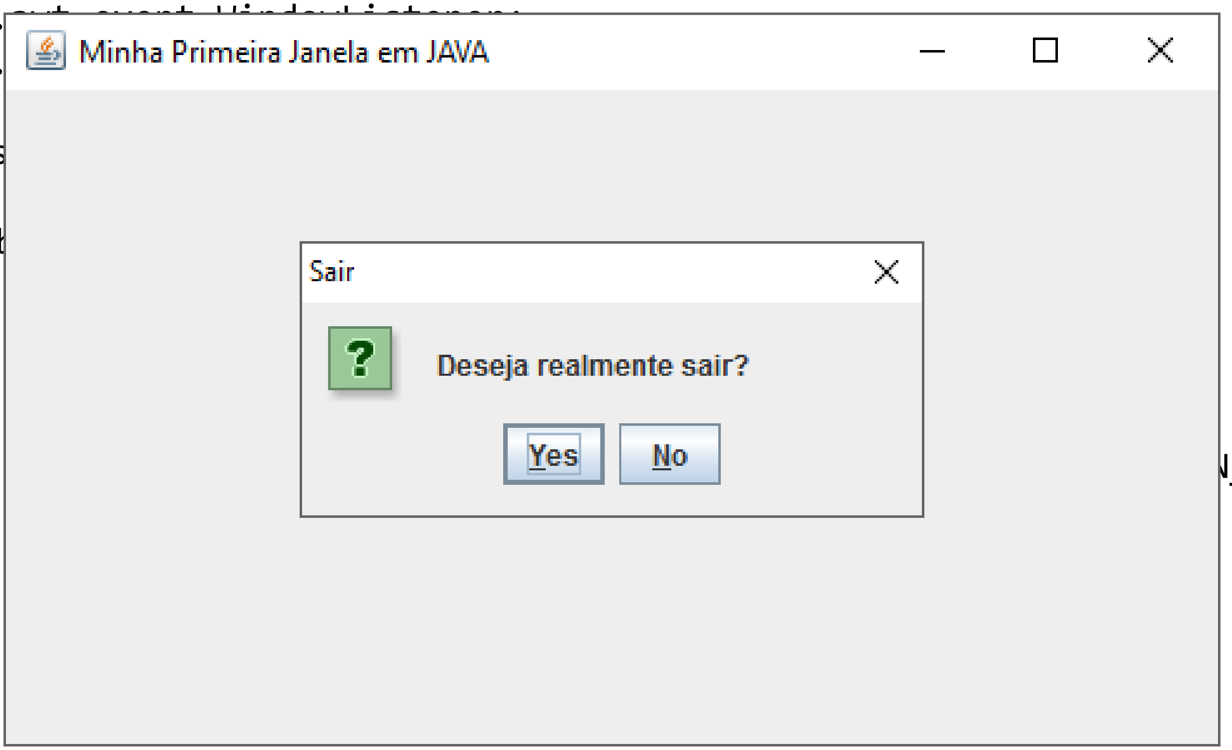
```
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import java.awt.event.WindowListener;
import java.

public class
{
          pub
          {
                                                                   JAVA");



                                                                  N_CLOSE);


          }
}
```



Fechando a Minha Primeira Janela

- Passo 3: instanciar um objeto dessa classe e registrá-lo ao objeto **JFrame**. E alterar o comportamento padrão para DO_NOTHING_ON_CLOSE.

```
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import java.
import java.

public class
{
        pub
        {
                                                                JAVA");


                                                               _CLOSE);


        }
}
```

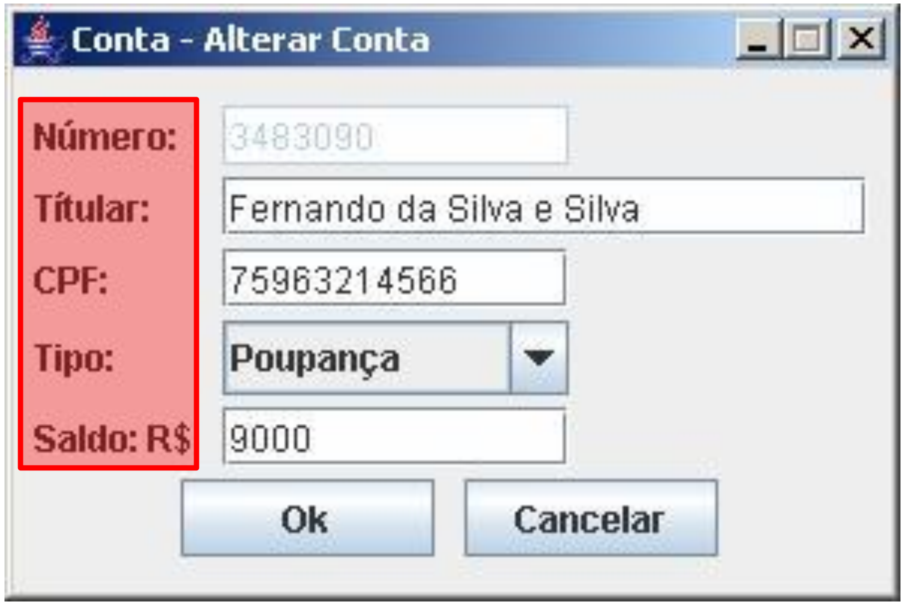

Minha Primeira Janela em JAVA

Sair

Deseja realmente sair?

A Classe JLabel

- Objetos da classe JLabel fornecem uma área para textos, imagens ou ambos.
- Objetos JLabel não reagem a eventos de entrada, por isso não obtem foco para edição.



Número: 3483090
Títular: Fernando da Silva e Silva
CPF: 75963214566
Tipo: Poupança
Saldo: R$ 9000

A Classe JLabel

- Construtores da Classe JLabel:

Constructor and Description
`JLabel()` Creates a JLabel instance with no image and with an empty string for the title.
`JLabel(Icon image)` Creates a JLabel instance with the specified image.
`JLabel(Icon image, int horizontalAlignment)` Creates a JLabel instance with the specified image and horizontal alignment.
`JLabel(String text)` Creates a JLabel instance with the specified text.
`JLabel(String text, Icon icon, int horizontalAlignment)` Creates a JLabel instance with the specified text, image, and horizontal alignment.
`JLabel(String text, int horizontalAlignment)` Creates a JLabel instance with the specified text and horizontal alignment.

A Classe JLabel

- Alguns Métodos da Classe JLabel:

Modifier and Type	Method and Description
String	getText() Returns the text string that the label displays.
void	setText(String text) Defines the single line of text this component will display.
void	setLabelFor(Component c) Set the component this is labelling.

- Método herdado de JComponet:

Modifier and Type	Method and Description
void	setFont(Font font) Sets the font for this component.

Hello World em javax.swing

- Entratégia:
  1. Criar uma Classe JanelaHello que será a janela da aplicação, portanto herdará de JFrame.
  2. A Classe JanelaHello terá um atributo privado do tipo JLabel.
  3. No método construtor de JanelaHello,
     1. Definir o titulo da janela através da chamada ao construtor da classe base.
     2. O objeto JLabel será instanciado com a frase e alinhamento central.
     3. A fonte do JLabel será mudada para Comic Sans MS, Negrito e tamanho 36, através do método setFont().
     4. Dimensionar o objeto Jframe
     5. Adicionar o objeto JLabel à janela através do método add()
     6. Atribuir um comportamento para o fechamento da janela.
     7. Exibir o objeto JFrame.
  4. No método main() da Classe Hello somente instancia-se um objeto da classe JanelaHello.

Hello World em javax.swing

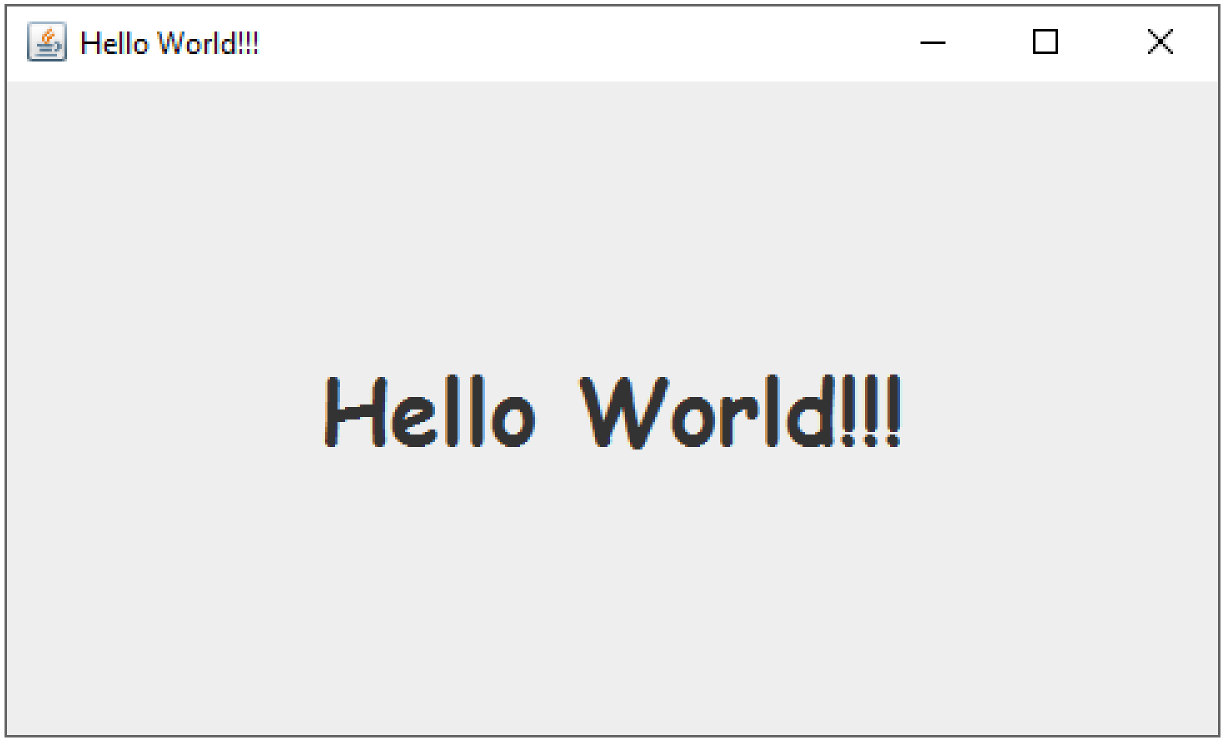
```
import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.Font;

class JanelaHello extends JFrame
{
    private JLabel lblMensagem;

    public JanelaHello(String s)
    {
      super(s);
      lblMensagem = new JLabel(s, JLabel.CENTER);
      lblMensagem.setFont(new Font("Comic Sans MS", Font.BOLD, 36));
      this.setSize(500, 300);
      this.add(lblMensagem);
      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      this.setVisible(true);
   }
}
```

Hello World em javax.swing

```
public class Hello
{
    public static void main(String[] args)
    {
        JanelaHello janela = new JanelaHello("Hello World!!!");
    }
}
```

Hello World em javax.swing

```
public class Hello
{
    public static void main(String[] args)
    {
        JanelaHello janela = new JanelaHello("Hello World!!!");
    }
}
```

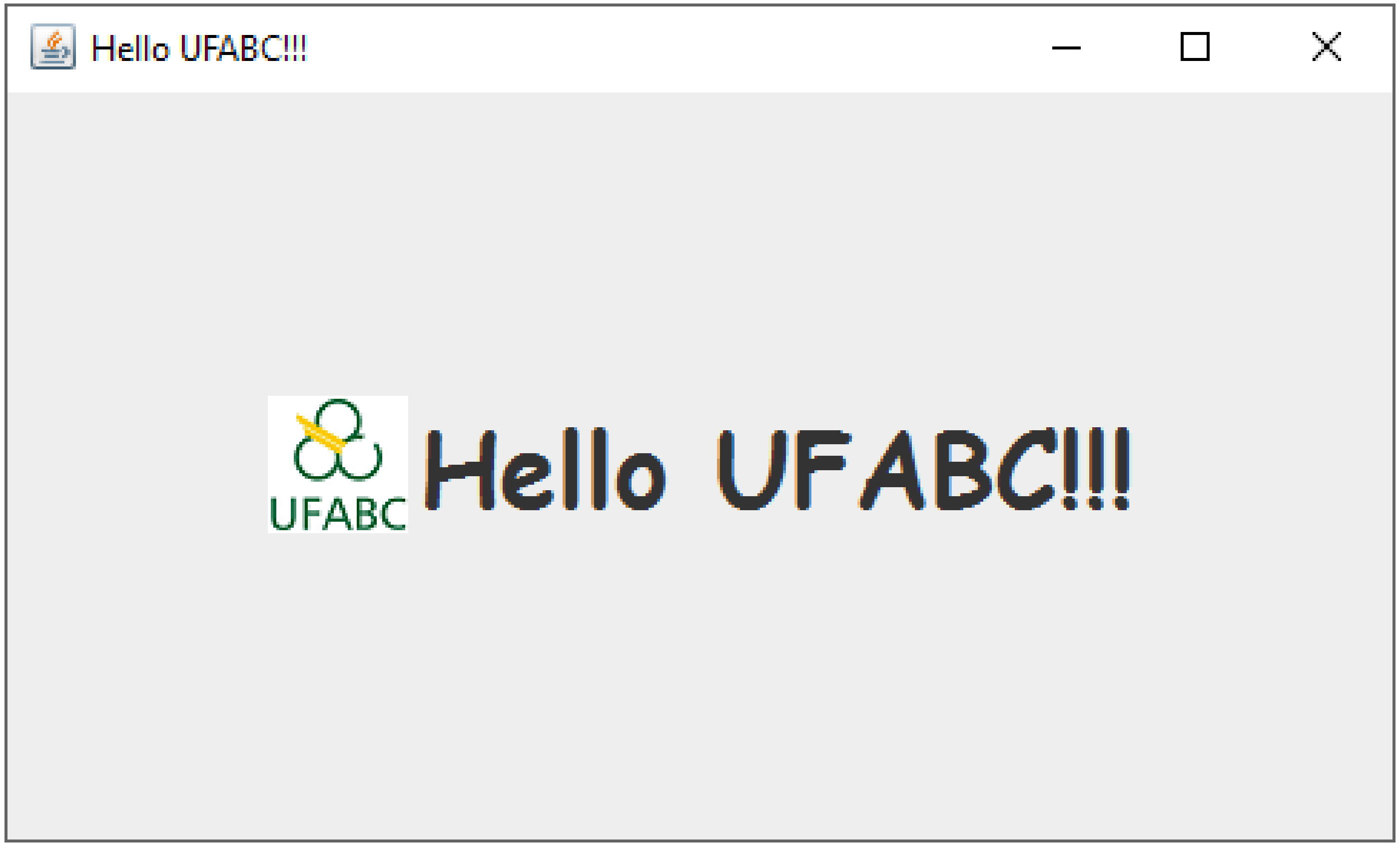


Hello World!!!

Hello World!!!

Hello UFABC

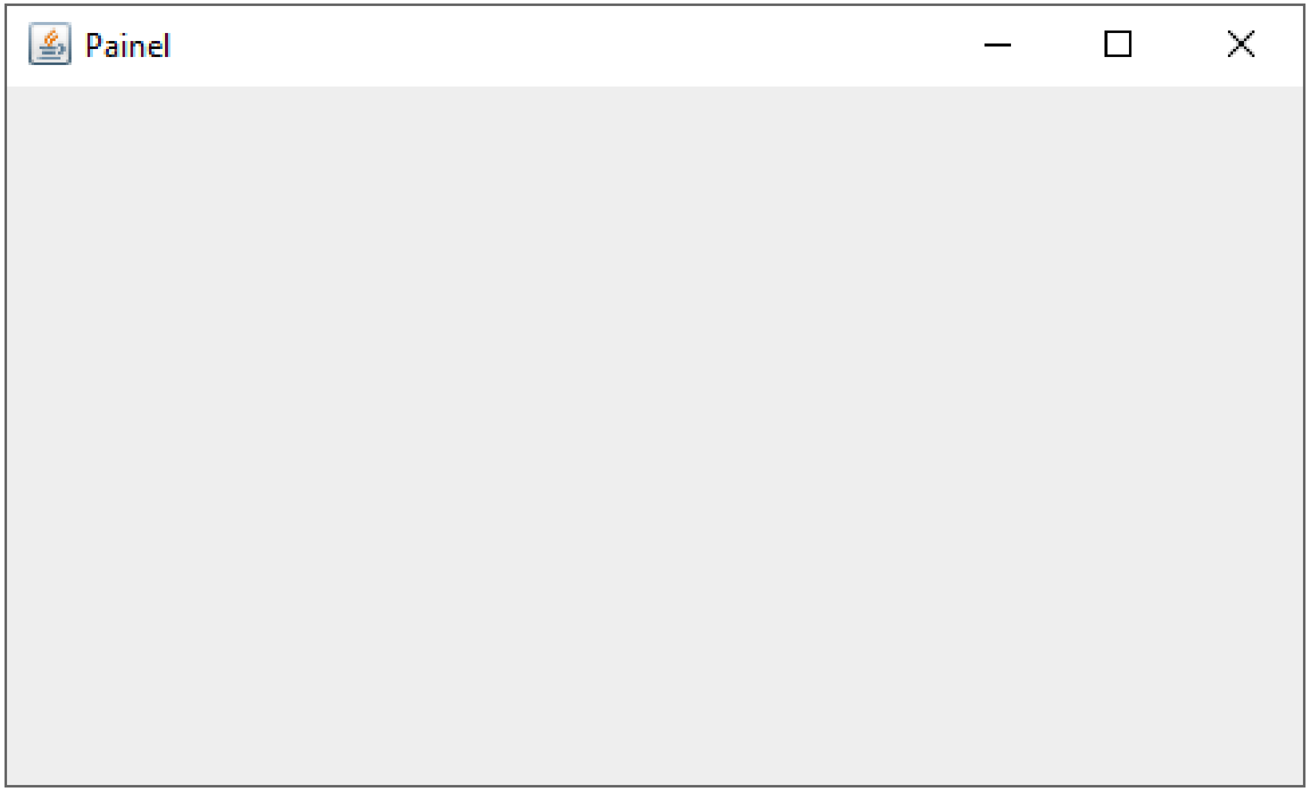
- Usando o Construtor:
  - JLabel(String text, Icon icon, int horizontalAlignment)



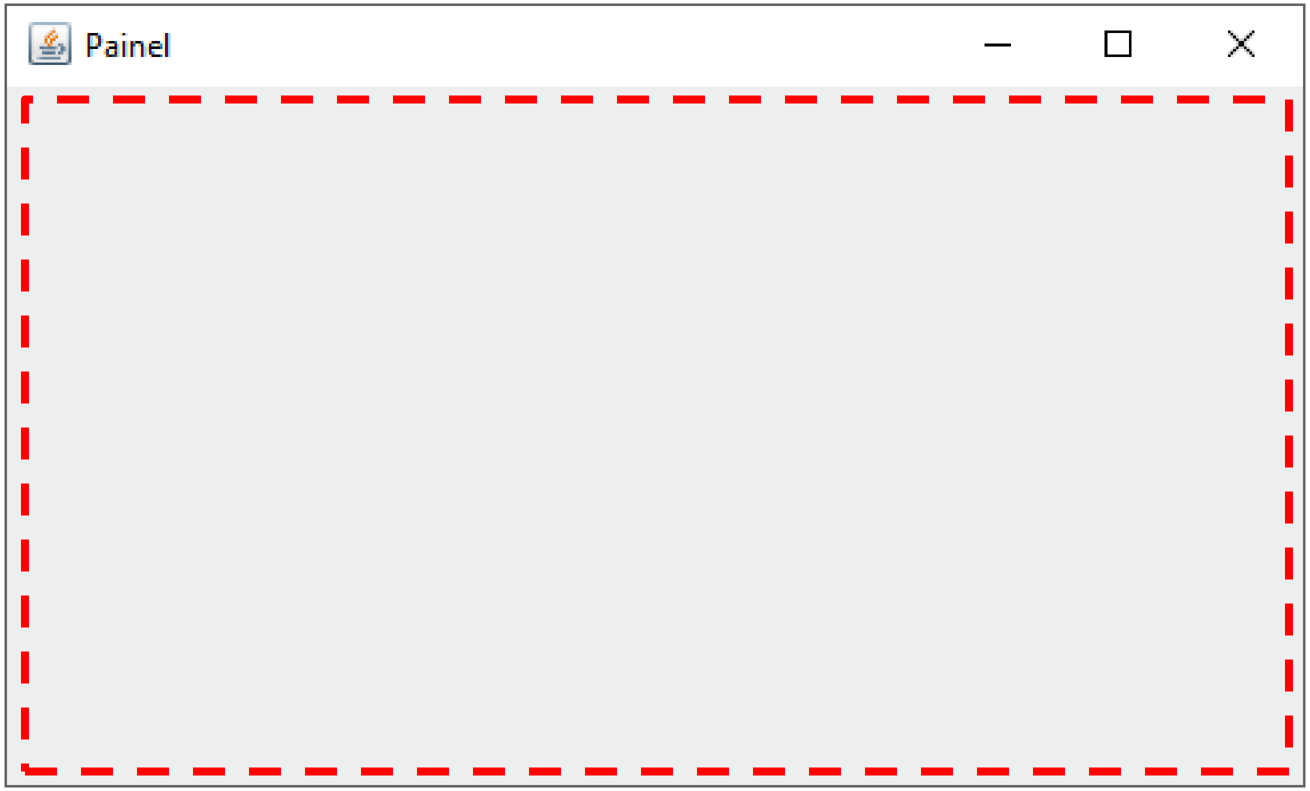
A Classe JPanel

- Um objeto da Classe Panel trata-se de um container para se adicionar outros componentes.
- Sua principal tarefa é organizar os componentes.
- É sempre interessante organizar os componentes dentro de um panel antes e então adicionar o panel ao frame.
- Uma aplicação GUI só tem um Frame, mas pode ter vários Panel.

A Classe JPanel



A Classe JPanel



A Classe JPanel

- Métodos Construtores:

Constructor and Description
`JPanel()` Creates a new JPanel with a double buffer and a flow layout.
`JPanel(boolean isDoubleBuffered)` Creates a new JPanel with FlowLayout and the specified buffering strategy.
`JPanel(LayoutManager layout)` Create a new buffered JPanel with the specified layout manager
`JPanel(LayoutManager layout, boolean isDoubleBuffered)` Creates a new JPanel with the specified layout manager and buffering strategy.

- Método herdado de JComponent:

Modifier and Type	Method and Description
`void`	`setBackground(Color bg)` Sets the background color of this component.

- Método herdado de JContainer:

Modifier and Type	Method and Description
`LayoutManager`	`getLayout()` Gets the layout manager for this container.

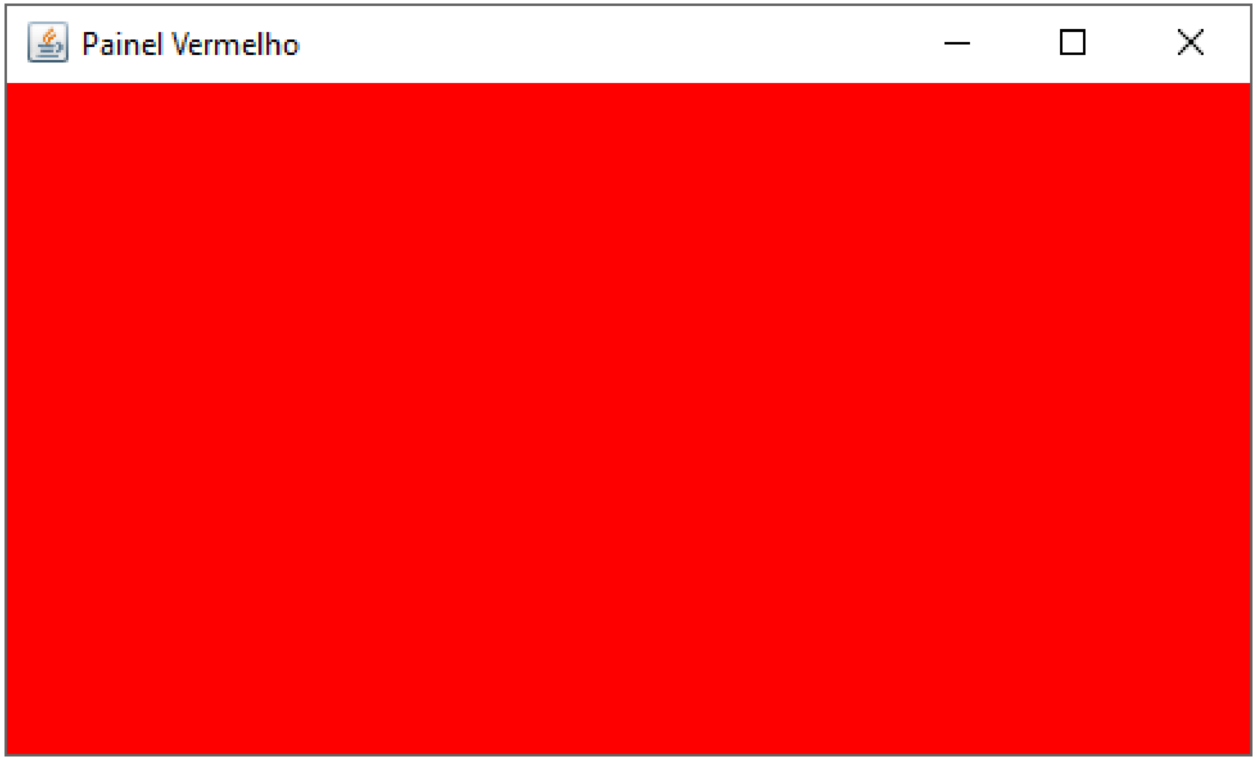
Minha Janela Vermelha

- Entratégia:
  1. Instanciar um objeto do tipo JFrame e um objeto do tipo JPanel.
  2. Alterar a propriedade background do Panel usando o método `setBackground()`, passando um objeto Color com a cor vermelha.
  3. Adicionar o objeto JPanel à janela.
  4. Definir tamanho, comportamento da janela ao fechar e a visibilidade.

Minha Janela Vermelha

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.Color;
public class Painel
{
       public static void main(String[] args)
       {
              JFrame janela = new JFrame("Painel");
              JPanel paneVermelho = new JPanel();

              paneVermelho.setBackground(new Color(255,0,0));

              janela.setSize(500,300);
              janela.add(paneVermelho);
              janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

              janela.setVisible(true);
       }
}
```

Minha Janela Vermelha



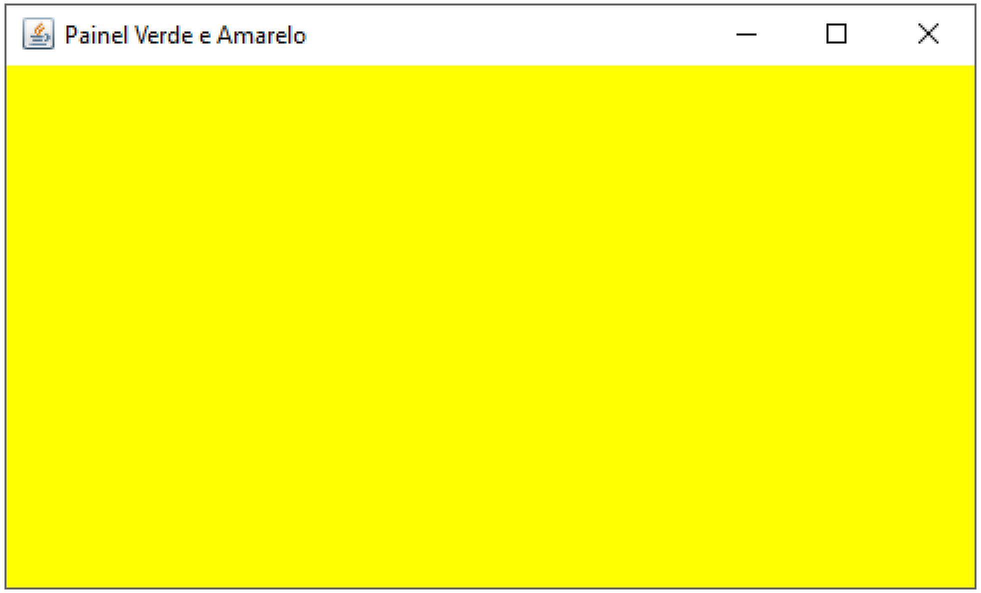
Minha Janela Verde e Amarela

- Entratégia:
  1. Instanciar um objeto do tipo JFrame e dois objetos do tipo JPanel.
  2. Alterar a propriedade background dos Panels usando o método `setBackground()`, passando um objeto Color com a cor vermelha e outro com a cor amarela.
  3. Adicionar os objetos JPanels à janela.
  4. Definir tamanho, comportamento da janela ao fechar e a visibilidade.

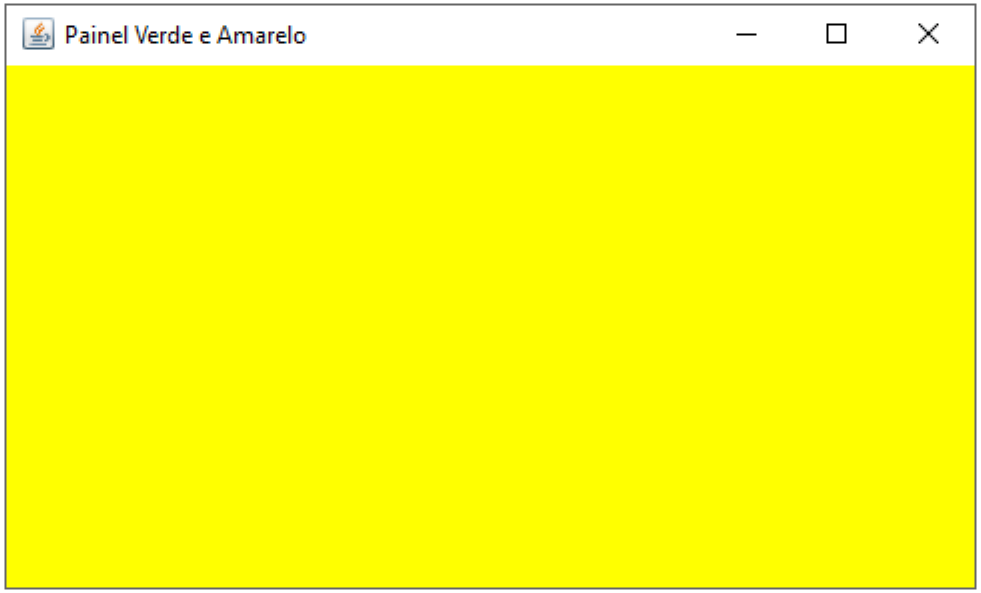
Minha Janela Verde e Amarela

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.Color;
public class Painel {
	public static void main(String[] args) {
		JFrame janela = new JFrame("Painel Verde e Amarelo");
		JPanel paneVerde = new JPanel();
		JPanel paneAmarelo = new JPanel();

		paneVerde.setBackground(new Color(0,255,0));
		paneAmarelo.setBackground(new Color(255,255,0));

		janela.setSize(500,300);
		janela.add(paneVerde);
		janela.add(paneAmarelo);
		janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		janela.setVisible(true);
	}
}
```

Minha Janela Verde e Amarela

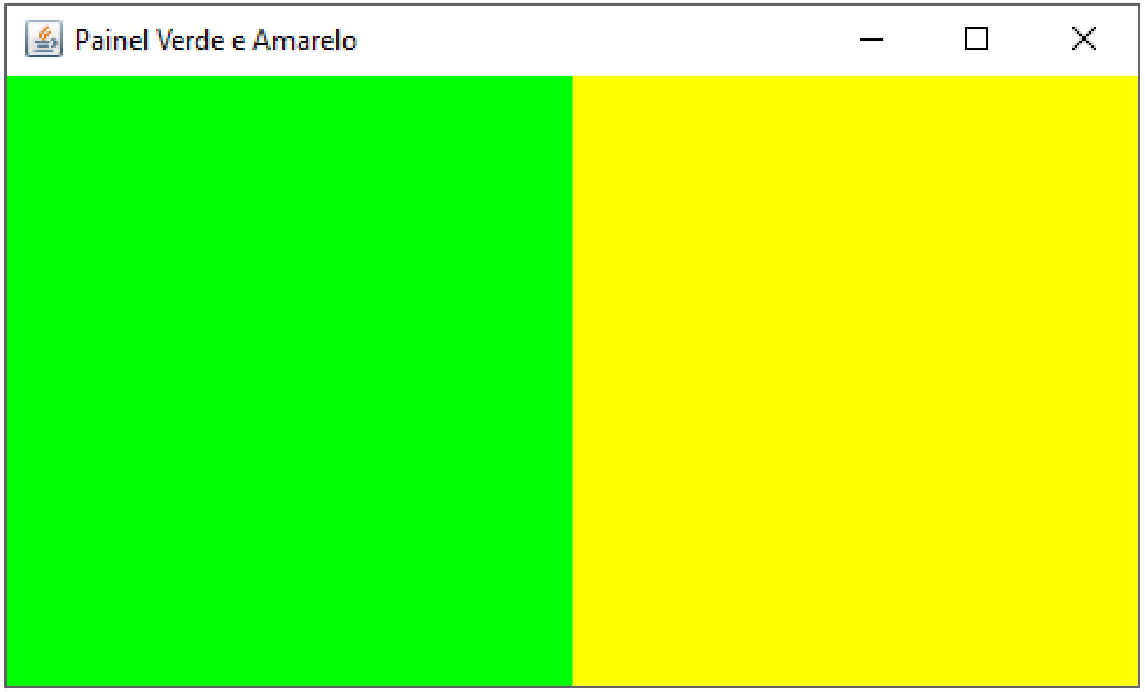


Minha Janela Verde e Amarela



- Perceba que o Panel amarelo foi sobreposto ao verde, pois foi o último componente a ser adicionado à janela.

Minha Janela Verde e Amarela



- Perceba que o Panel amarelo foi sobreposto ao verde, pois foi o último componente a ser adicionado à janela.
- Para se adicionar vários componentes à um Frame precisa-se organizá-los usando **Gerenciadores de Layout** do Frame.

Gerenciadores de Layout

- A aparencia e disposição dos componentes na janela é importante.
- Até este momento estamos adicionando componentes nos containers em organizá-los.
- Nas aplicações GUI em JAVA os componentes são dispostos nos containers (JFrame e JPanel) usando gerenciadores de Layout.

Gerenciadores de Layout

- Os Layout Managers são classes que implementam interfaces que determinam como os componentes de um container são arranjados.
- Ou seja, indica como os componentes devem se organizar, se distribuir e se posicionar.
- Dentre diversas classes de Layout, as principais são:
  - BorderLayout
  - FlowLayout
  - GridLayout
  - CardLayout
  - GridBagLayout

Layout de Borda

- Métodos Construtores:

Constructor and Description
`BorderLayout()` Constructs a new border layout with no gaps between components.
`BorderLayout(int hgap, int vgap)` Constructs a border layout with the specified gaps between components.

- Principais Métodos:

Modifier and Type	Method and Description
`void`	`addLayoutComponent(Component comp, Object constraints)` Adds the specified component to the layout, using the specified constraint object.
`Object`	`getConstraints(Component comp)` Gets the constraints for the specified component
`Component`	`getLayoutComponent(Container target, Object constraints)` Returns the component that corresponds to the given constraint location based on the target Container's component orientation.
`Component`	`getLayoutComponent(Object constraints)` Gets the component that was added using the given constraint

Layout de Borda

- Principais Atributos:

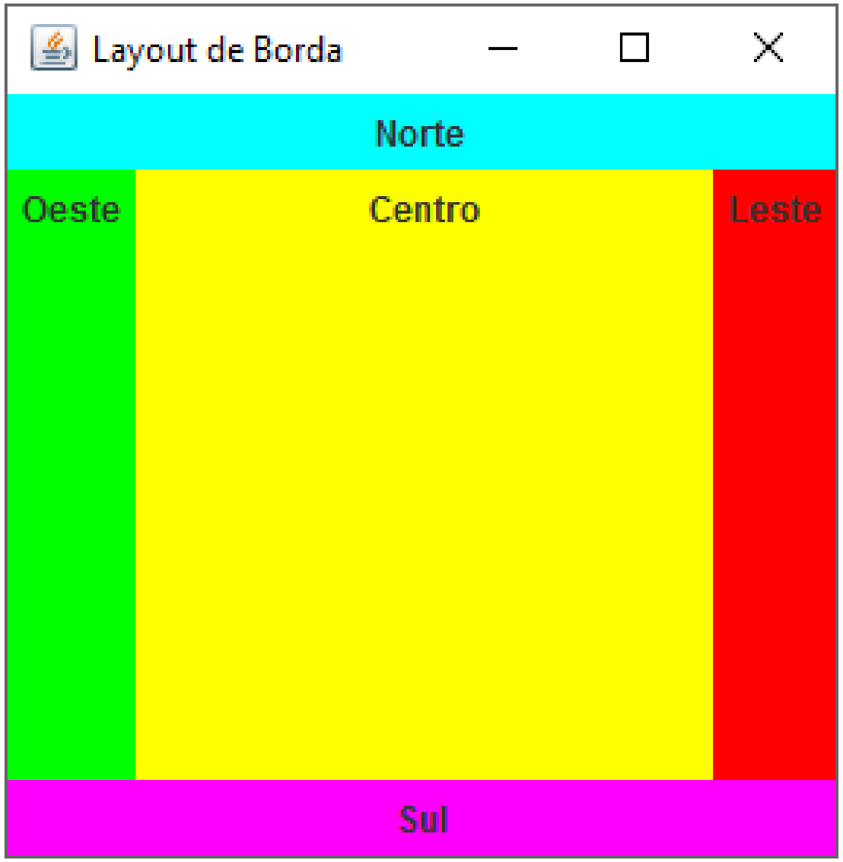
Modifier and Type	Field and Description
static String	CENTER The center layout constraint (middle of container).
static String	EAST The east layout constraint (right side of container).
static String	NORTH The north layout constraint (top of container).
static String	SOUTH The south layout constraint (bottom of container).
static String	WEST The west layout constraint (left side of container).

Layout de Borda

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import java.awt.BorderLayout;
import java.awt.Color;

public class Borda {
    public static void main(String[] args) {
        JFrame janela;
        JPanel leste, oeste, norte, sul, centro;

        janela = new JFrame("Layout de Borda");
        leste = new JPanel();
        oeste = new JPanel();
        norte = new JPanel();
        sul = new JPanel();
        centro = new JPanel();

        leste.setBackground(Color.RED);
        leste.add(new JLabel("Leste"));
        oeste.setBackground(Color.GREEN);
        oeste.add(new JLabel("Oeste"));
        norte.setBackground(Color.CYAN);
        norte.add(new JLabel("Norte"));
        sul.setBackground(Color.MAGENTA);
        sul.add(new JLabel("Sul"));
        centro.setBackground(Color.YELLOW);
        centro.add(new JLabel("Centro"));

        janela.setLayout(new BorderLayout());
        janela.add(leste, BorderLayout.EAST);
        janela.add(oeste, BorderLayout.WEST);
        janela.add(norte, BorderLayout.NORTH);
        janela.add(sul, BorderLayout.SOUTH);
        janela.add(centro, BorderLayout.CENTER);
        janela.setSize(300, 300);
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        janela.setVisible(true);
    }
}
```

Layout de Borda

```java
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import java.awt.BorderLayout;
import java.awt.Color;

public class Borda {
    public static void main(String[] args) {
        JFrame janela;
        JPanel leste, oeste, norte, sul, centro;

        janela = new JFrame("Layout de Borda");
        leste = new JPanel();
        oeste = new JPanel();
        norte = new JPanel();
        sul = new JPanel();
        centro = new JPanel();

        leste.setBackground(Color.RED);
        leste.add(new JLabel("Leste"));
        oeste.setBackground(Color.GREEN);
        oeste.add(new JLabel("Oeste"));
        norte.setBackground(Color.CYAN);
        norte.add(new JLabel("Norte"));
        sul.setBackground(Color.MAGENTA);
        sul.add(new JLabel("Sul"));
        centro.setBackground(Color.YELLOW);
        centro.add(new JLabel("Centro"));

        janela.setLayout(new BorderLayout());
        janela.add(leste, BorderLayout.EAST);
        janela.add(oeste, BorderLayout.WEST);
        janela.add(norte, BorderLayout.NORTH);
```



```java
        janela.add(sul, BorderLayout.SOUTH);
        janela.add(centro, BorderLayout.CENTER);
        janela.setSize(300, 300);
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        janela.setVisible(true);
    }
}
```

Layout de Fluxo

- Métodos Construtores:

Constructor and Description
FlowLayout() Constructs a new FlowLayout with a centered alignment and a default 5-unit horizontal and vertical gap.
FlowLayout(int align) Constructs a new FlowLayout with the specified alignment and a default 5-unit horizontal and vertical gap.
FlowLayout(int align, int hgap, int vgap) Creates a new flow layout manager with the indicated alignment and the indicated horizontal and vertical gaps.

- Principais Métodos:

Modifier and Type	Method and Description
void	addLayoutComponent(String name, Component comp) Adds the specified component to the layout.
void	layoutContainer(Container target) Lays out the container.

Layout de Fluxo

- Principais Atributos:

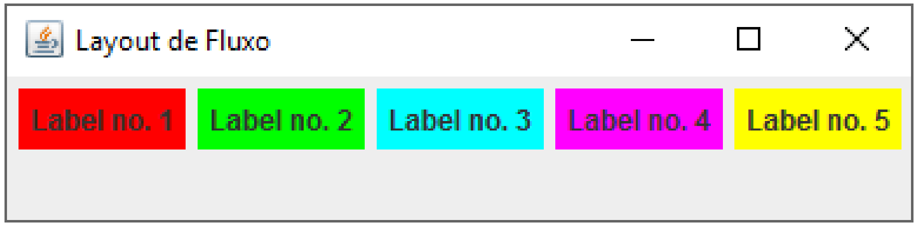
Modifier and Type	Field and Description
static int	**CENTER** This value indicates that each row of components should be centered.
static int	**LEADING** This value indicates that each row of components should be justified to the leading edge of the container's orientation, for example, to the left in left-to-right orientations.
static int	**LEFT** This value indicates that each row of components should be left-justified.
static int	**RIGHT** This value indicates that each row of components should be right-justified.
static int	**TRAILING** This value indicates that each row of components should be justified to the trailing edge of the container's orientation, for example, to the right in left-to-right orientations.

Layout de Fluxo

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import java.awt.FlowLayout;
import java.awt.Color;

public class Fluxo {
    public static void main(String[] args) {
         JFrame janela = new JFrame("Layout de Fluxo");
         JPanel pane1, pane2, pane3, pane4, pane5;

         pane1 = new JPanel();
         pane1.setBackground(Color.RED);
         pane1.add(new JLabel("Label no. 1"));
         pane2 = new JPanel();
         pane2.setBackground(Color.GREEN);
         pane2.add(new JLabel("Label no. 2"));
         pane3 = new JPanel();
         pane3.setBackground(Color.CYAN);
         pane3.add(new JLabel("Label no. 3"));
         pane4 = new JPanel();
         pane4.setBackground(Color.MAGENTA);
         pane4.add(new JLabel("Label no. 4"));
         pane5 = new JPanel();
         pane5.setBackground(Color.YELLOW);
         pane5.add(new JLabel("Label no. 5"));

         janela.setLayout(new FlowLayout());
         janela.add(pane1);
         janela.add(pane2);
         janela.add(pane3);
         janela.add(pane4);
         janela.add(pane5);
         janela.setSize(400,100);
         janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         janela.setVisible(true);
    }
}
```

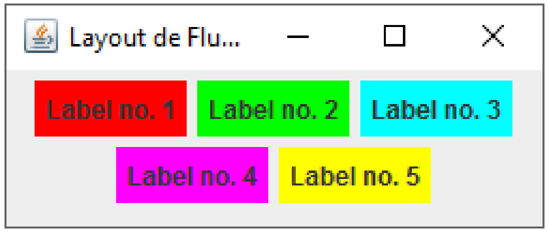
Layout de Fluxo

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import java.awt.FlowLayout;
import java.awt.Color;

public class Fluxo {
    public static void main(String[] args) {
         JFrame janela = new JFrame("Layout de Fluxo");
         JPanel pane1, pane2, pane3, pane4, pane5;

         pane1 = new JPanel();
         pane1.setBackground(Color.RED);
         pane1.add(new JLabel("Label no. 1"));
         pane2 = new JPanel();
         pane2.setBackground(Color.GREEN);
         pane2.add(new JLabel("Label no. 2"));
         pane3 = new JPanel();
         pane3.setBackground(Color.CYAN);
         pane3.add(new JLabel("Label no. 3"));
         pane4 = new JPanel();
         pane4.setBackground(Color.MAGENTA);
         pane4.add(new JLabel("Label no. 4"));
         pane5 = new JPanel();
         pane5.setBackground(Color.YELLOW);
         pane5.add(new JLabel("Label no. 5"));

         janela.setLayout(new FlowLayout());
         janela.add(pane1);
         janela.add(pane2);
         janela.add(pane3);
         janela.add(pane4);
         janela.add(pane5);
         janela.setSize(400,100);
         janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         janela.setVisible(true);
    }
}
```



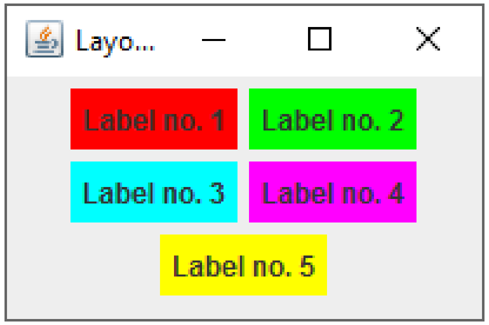
Layout de Fluxo

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import java.awt.FlowLayout;
import java.awt.Color;

public class Fluxo {
    public static void main(String[] args) {
         JFrame janela = new JFrame("Layout de Fluxo");
         JPanel pane1, pane2, pane3, pane4, pane5;

         pane1 = new JPanel();
         pane1.setBackground(Color.RED);
         pane1.add(new JLabel("Label no. 1"));
         pane2 = new JPanel();
         pane2.setBackground(Color.GREEN);
         pane2.add(new JLabel("Label no. 2"));
         pane3 = new JPanel();
         pane3.setBackground(Color.CYAN);
         pane3.add(new JLabel("Label no. 3"));
         pane4 = new JPanel();
         pane4.setBackground(Color.MAGENTA);
         pane4.add(new JLabel("Label no. 4"));
         pane5 = new JPanel();
         pane5.setBackground(Color.YELLOW);
         pane5.add(new JLabel("Label no. 5"));


         janela.setLayout(new FlowLayout());
         janela.add(pane1);
         janela.add(pane2);
         janela.add(pane3);
         janela.add(pane4);
         janela.add(pane5);
         janela.setSize(400,100);
         janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         janela.setVisible(true);
    }
}
```



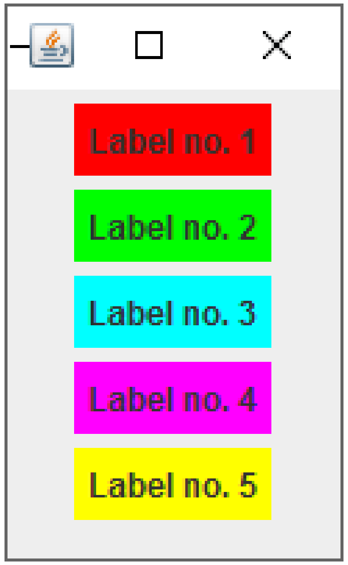
Layout de Fluxo

```java
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import java.awt.FlowLayout;
import java.awt.Color;

public class Fluxo {
    public static void main(String[] args) {
         JFrame janela = new JFrame("Layout de Fluxo");
         JPanel pane1, pane2, pane3, pane4, pane5;

         pane1 = new JPanel();
         pane1.setBackground(Color.RED);
         pane1.add(new JLabel("Label no. 1"));
         pane2 = new JPanel();
         pane2.setBackground(Color.GREEN);
         pane2.add(new JLabel("Label no. 2"));
         pane3 = new JPanel();
         pane3.setBackground(Color.CYAN);
         pane3.add(new JLabel("Label no. 3"));
         pane4 = new JPanel();
         pane4.setBackground(Color.MAGENTA);
         pane4.add(new JLabel("Label no. 4"));
         pane5 = new JPanel();
         pane5.setBackground(Color.YELLOW);
         pane5.add(new JLabel("Label no. 5"));


         janela.setLayout(new FlowLayout());
         janela.add(pane1);
         janela.add(pane2);
         janela.add(pane3);
         janela.add(pane4);
         janela.add(pane5);
         janela.setSize(400,100);
         janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         janela.setVisible(true);
    }
}
```



Layout de Fluxo

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import java.awt.FlowLayout;
import java.awt.Color;

public class Fluxo {
    public static void main(String[] args) {
        JFrame janela = new JFrame("Layout de Fluxo");
        JPanel pane1, pane2, pane3, pane4, pane5;

        pane1 = new JPanel();
        pane1.setBackground(Color.RED);
        pane1.add(new JLabel("Label no. 1"));
        pane2 = new JPanel();
        pane2.setBackground(Color.GREEN);
        pane2.add(new JLabel("Label no. 2"));
        pane3 = new JPanel();
        pane3.setBackground(Color.CYAN);
        pane3.add(new JLabel("Label no. 3"));
        pane4 = new JPanel();
        pane4.setBackground(Color.MAGENTA);
        pane4.add(new JLabel("Label no. 4"));
        pane5 = new JPanel();
        pane5.setBackground(Color.YELLOW);
        pane5.add(new JLabel("Label no. 5"));


        janela.setLayout(new FlowLayout());
        janela.add(pane1);
        janela.add(pane2);
        janela.add(pane3);
        janela.add(pane4);
        janela.add(pane5);
        janela.setSize(400,100);
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        janela.setVisible(true);
    }
}
```



Layout de Grade

- Métodos Construtores:

Constructor and Description
GridLayout() Creates a grid layout with a default of one column per component, in a single row.
GridLayout(int rows, int cols) Creates a grid layout with the specified number of rows and columns.
GridLayout(int rows, int cols, int hgap, int vgap) Creates a grid layout with the specified number of rows and columns.

- Principais Métodos:

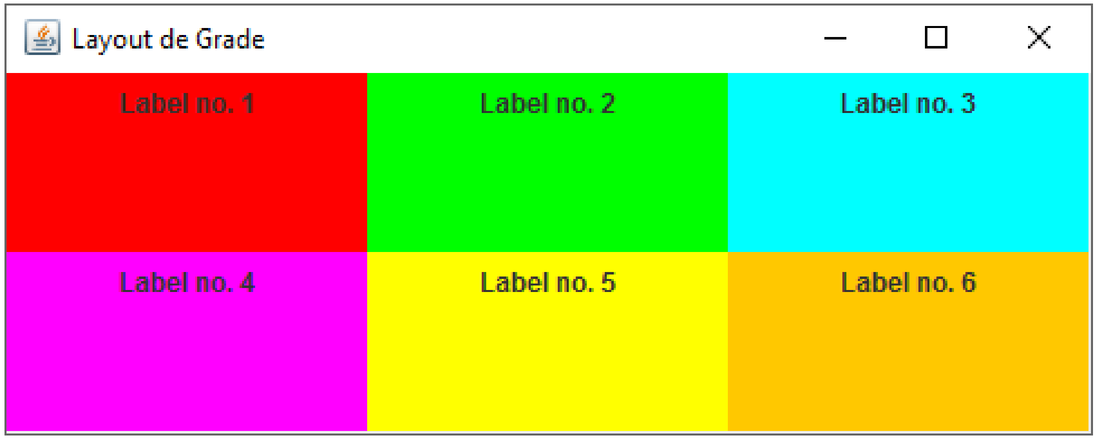
Modifier and Type	Method and Description
void	addLayoutComponent(String name, Component comp) Adds the specified component with the specified name to the layout.
int	getColumns() Gets the number of columns in this layout.
int	getRows() Gets the number of rows in this layout.

Layout de Grade

```java
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import java.awt.GridLayout;
import java.awt.Color;

public class Grade {
   public static void main(String[] args) {
        JFrame janela = new JFrame("Layout de Grade");
        JPanel pane1, pane2, pane3, pane4, pane5, pane6;

        pane1 = new JPanel();
        pane1.setBackground(Color.RED);
        pane1.add(new JLabel("Label no. 1"));
        pane2 = new JPanel();
        pane2.setBackground(Color.GREEN);
        pane2.add(new JLabel("Label no. 2"));
        pane3 = new JPanel();
        pane3.setBackground(Color.CYAN);
        pane3.add(new JLabel("Label no. 3"));
        pane4 = new JPanel();
        pane4.setBackground(Color.MAGENTA);
        pane4.add(new JLabel("Label no. 4"));
        pane5 = new JPanel();
        pane5.setBackground(Color.YELLOW);
        pane5.add(new JLabel("Label no. 5"));
        pane6 = new JPanel();
        pane6.setBackground(Color.ORANGE);
        pane6.add(new JLabel("Label no. 6"));

        janela.setLayout(new GridLayout(2, 3));
        janela.add(pane1);
        janela.add(pane2);
        janela.add(pane3);
        janela.add(pane4);
        janela.add(pane5);
        janela.add(pane6);
        janela.setSize(500, 200);
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        janela.setVisible(true);
   }
}
```

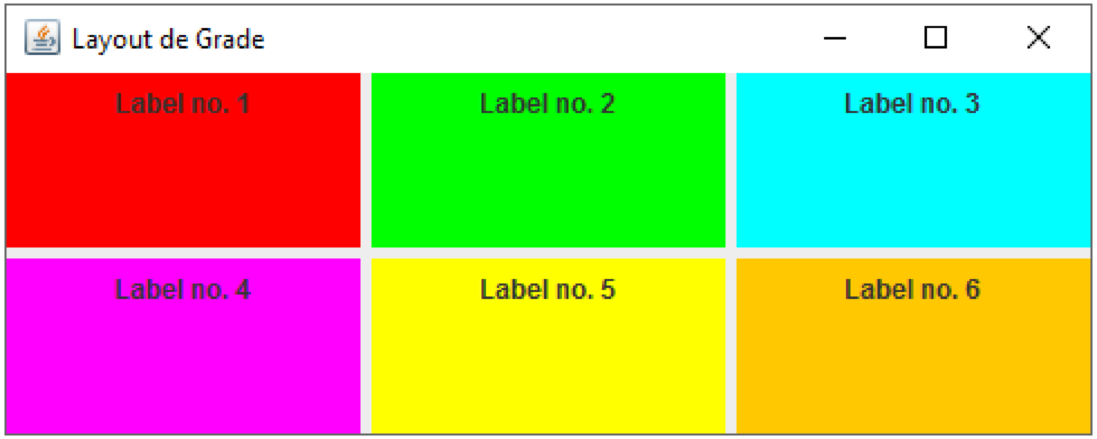
Layout de Grade

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import java.awt.GridLayout;
import java.awt.Color;

public class Grade {
   public static void main(String[] args) {
        JFrame janela = new JFrame("Layout de Grade");
        JPanel pane1, pane2, pane3, pane4, pane5, pane6;

        pane1 = new JPanel();
        pane1.setBackground(Color.RED);
        pane1.add(new JLabel("Label no. 1"));
        pane2 = new JPanel();
        pane2.setBackground(Color.GREEN);
        pane2.add(new JLabel("Label no. 2"));
        pane3 = new JPanel();
        pane3.setBackground(Color.CYAN);
        pane3.add(new JLabel("Label no. 3"));
        pane4 = new JPanel();
        pane4.setBackground(Color.MAGENTA);
        pane4.add(new JLabel("Label no. 4"));
        pane5 = new JPanel();
        pane5.setBackground(Color.YELLOW);
        pane5.add(new JLabel("Label no. 5"));
        pane6 = new JPanel();
        pane6.setBackground(Color.ORANGE);
        pane6.add(new JLabel("Label no. 6"));

        janela.setLayout(new GridLayout(2, 3));
        janela.add(pane1);
        janela.add(pane2);
        janela.add(pane3);
        janela.add(pane4);
        janela.add(pane5);
        janela.add(pane6);
        janela.setSize(500, 200);
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        janela.setVisible(true);
   }
}
```



Layout de Grade

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import java.awt.GridLayout;
import java.awt.Color;

public class Grade {
   public static void main(String[] args) {
        JFrame janela = new JFrame("Layout de Grade");
        JPanel pane1, pane2, pane3, pane4, pane5, pane6;

        pane1 = new JPanel();
        pane1.setBackground(Color.RED);
        pane1.add(new JLabel("Label no. 1"));
        pane2 = new JPanel();
        pane2.setBackground(Color.GREEN);
        pane2.add(new JLabel("Label no. 2"));
        pane3 = new JPanel();
        pane3.setBackground(Color.CYAN);
        pane3.add(new JLabel("Label no. 3"));
        pane4 = new JPanel();
        pane4.setBackground(Color.MAGENTA);
        pane4.add(new JLabel("Label no. 4"));
        pane5 = new JPanel();
        pane5.setBackground(Color.YELLOW);
        pane5.add(new JLabel("Label no. 5"));
        pane6 = new JPanel();
        pane6.setBackground(Color.ORANGE);
        pane6.add(new JLabel("Label no. 6"));

        janela.setLayout(new GridLayout(2, 3, 5, 5));
        janela.add(pane1);
        janela.add(pane2);
        janela.add(pane3);
        janela.add(pane4);
        janela.add(pane5);
        janela.add(pane6);
        janela.setSize(500, 200);
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        janela.setVisible(true);
   }
}
```

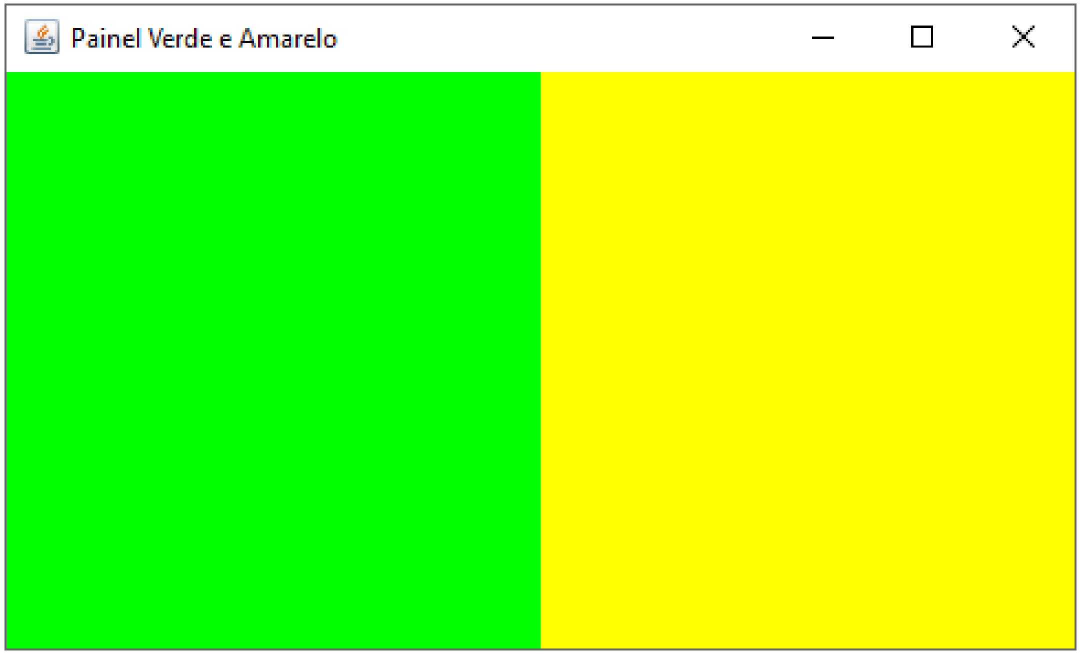


Layout de Grade

- Minha Janela
  Verde e Amarela

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.Color;
import java.awt.GridLayout.

public class Painel
{
     public static void main(String[] args)
     {
          JFrame janela = new JFrame("Painel Verde e Amarelo");
          JPanel paneVerde = new JPanel();
          JPanel paneAmarelo = new JPanel();

          paneVerde.setBackground(new Color(0,255,0));
          paneAmarelo.setBackground(new Color(255,255,0));

          janela.setLayout(new GridLayout(1,2));
          janela.setSize(500,300);
          janela.add(paneVerde);
          janela.add(paneAmarelo);
          janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          janela.setVisible(true);
      }
}
```

Layout de Grade



- Minha Janela Verde e Amarela

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.Color;
import java.awt.GridLayout.

public class Painel
{
     public static void main(String[] args)
     {
          JFrame janela = new JFrame("Painel Verde e Amarelo");
          JPanel paneVerde = new JPanel();
          JPanel paneAmarelo = new JPanel();

          paneVerde.setBackground(new Color(0,255,0));
          paneAmarelo.setBackground(new Color(255,255,0));

          janela.setLayout(new GridLayout(1,2));
          janela.setSize(500,300);
          janela.add(paneVerde);
          janela.add(paneAmarelo);
          janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          janela.setVisible(true);
      }
}
```

Layout de Carta

- Métodos Construtores:

Constructor and Description
`CardLayout()` Creates a new card layout with gaps of size zero.
`CardLayout(int hgap, int vgap)` Creates a new card layout with the specified horizontal and vertical gaps.

- Principais Métodos:

Modifier and Type	Method and Description
`void`	`addLayoutComponent(Component comp, Object constraints)` Adds the specified component to this card layout's internal table of names.
`void`	`show(Container parent, String name)` Flips to the component that was added to this layout with the specified name, using addLayoutComponent.

Layout de Carta

- Estratégia:
  1. Instanciar um objeto JFrame e 7 objetos JPanel.
  2. Alterar a cor de fundo de 6 painéis.
  3. Adicionar os 6 painéis no painel principal com rótulos.
  4. Determinar tamanho do JFrame e exibi-lo.
  5. Exibir um Option Dialog com cada uma das cores e uma opção sair.
  6. Entrar em um laço enquanto a opção sair não for escolhida.
  7. A cada cor escolhida, obter o layout do painel principal e exibir o painel rotulado de acordo com a cor escolhida.
  8. Exibir o Option Dialog do passo 5 e voltar ao passo 6.

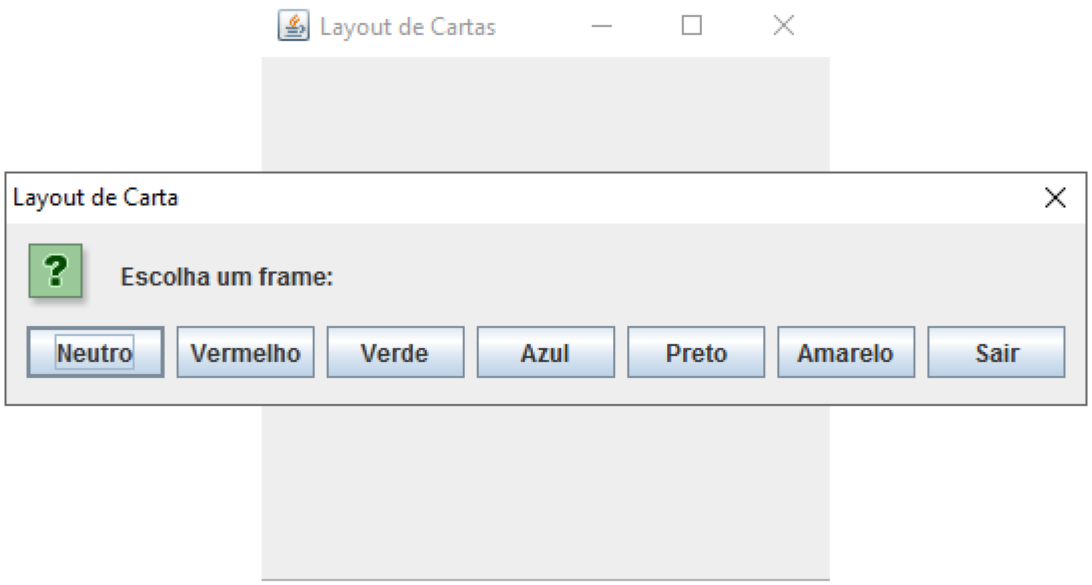
Layout de Carta

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JOptionPane;
import java.awt.CardLayout;
import java.awt.Color;

public class Carta2  {
   public static void main(String[] args)  {
       JFrame janela = new JFrame("Layout de Cartas");
       JPanel paneCores, paneRed, paneGreen, paneBlue, paneBlack, paneYellow, paneNeutro;
       String opcoes[] = {"Neutro", "Vermelho", "Verde", "Azul", "Preto", "Amarelo", "Sair"};
       int escolha;

       paneCores = new JPanel();
       paneNeutro = new JPanel();
       paneRed = new JPanel();
       paneRed.setBackground(Color.RED);
       paneGreen = new JPanel();
       paneGreen.setBackground(Color.GREEN);
       paneBlue = new JPanel();
       paneBlue.setBackground(Color.BLUE);
       paneBlack = new JPanel();
       paneBlack.setBackground(Color.BLACK);
       paneYellow = new JPanel();
       paneYellow.setBackground(Color.YELLOW);

       paneCores.setLayout(new CardLayout());
       paneCores.add(paneNeutro, "paneNeutro");
       paneCores.add(paneRed, "paneRed");
       paneCores.add(paneGreen, "paneGreen");
       paneCores.add(paneBlue, "paneBlue");
       paneCores.add(paneBlack, "paneBlack");
       paneCores.add(paneYellow, "paneYellow");
       janela.add(paneCores);
```

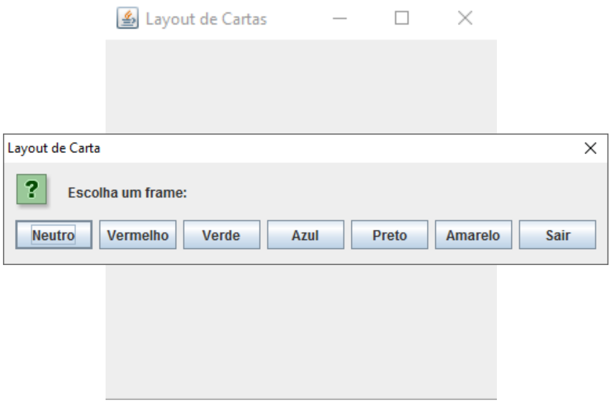
Layout de Carta

```
        janela.add(paneCores);
        janela.setSize(300,300);
        janela.setVisible(true);

        escolha = JOptionPane.showOptionDialog(janela, "Escolha um frame:", "Layout de Carta",
                                JOptionPane.DEFAULT_OPTION, JOptionPane.QUESTION_MESSAGE, null, opcoes, opcoes[0]);

        while(escolha!=6)
        {
            switch(escolha)
            {
                case 0: ((CardLayout)paneCores.getLayout()).show(paneCores, "paneNeutro"); break;
                case 1: ((CardLayout)paneCores.getLayout()).show(paneCores, "paneRed"); break;
                case 2: ((CardLayout)paneCores.getLayout()).show(paneCores, "paneGreen"); break;
                case 3: ((CardLayout)paneCores.getLayout()).show(paneCores, "paneBlue"); break;
                case 4: ((CardLayout)paneCores.getLayout()).show(paneCores, "paneBlack"); break;
                case 5: ((CardLayout)paneCores.getLayout()).show(paneCores, "paneYellow"); break;
            }
            escolha = JOptionPane.showOptionDialog(janela, "Escolha um frame:", "Layout de Carta",
                                JOptionPane.DEFAULT_OPTION, JOptionPane.QUESTION_MESSAGE, null, opcoes, opcoes[0]);
        }
        System.exit(0);
    }
}
```

Layout de Carta



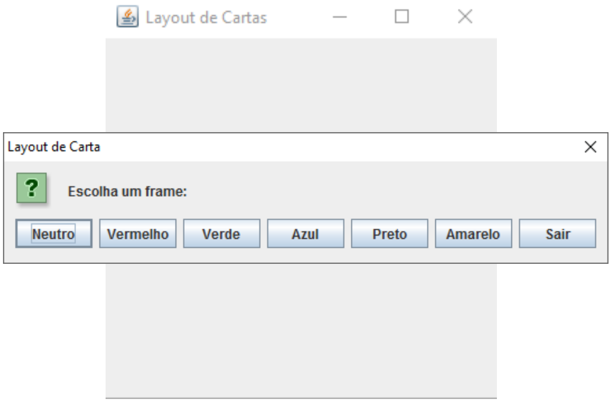
Layout de Carta

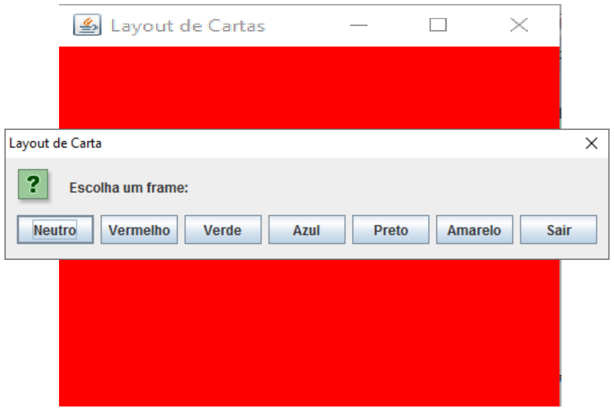

Layout de Cartas

Layout de Carta

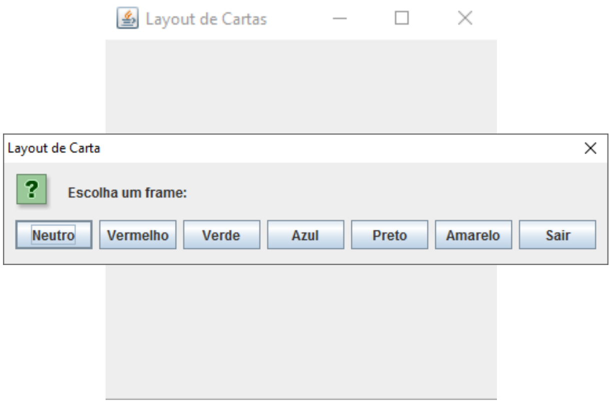
Escolha um frame:

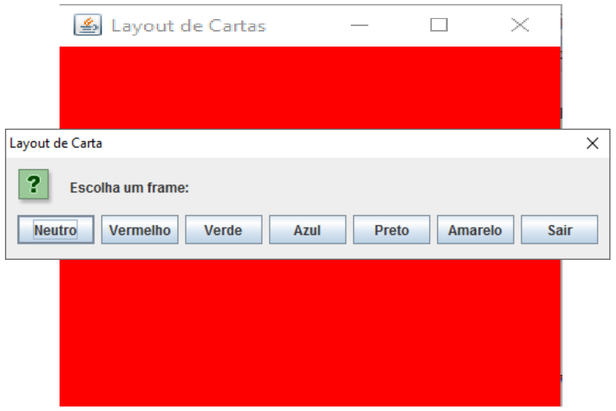
Layout de Carta

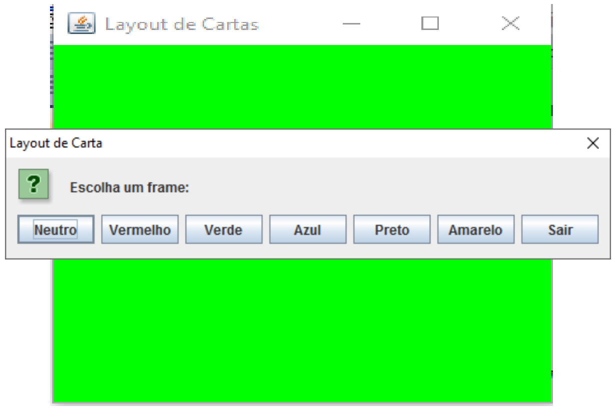




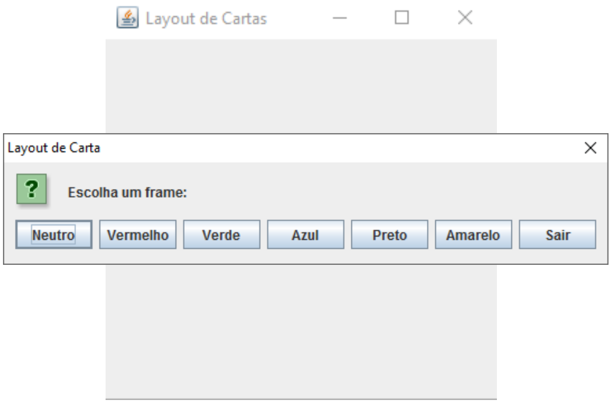
Layout de Carta

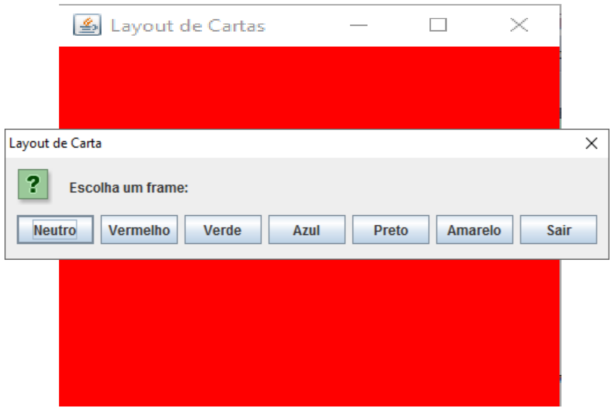


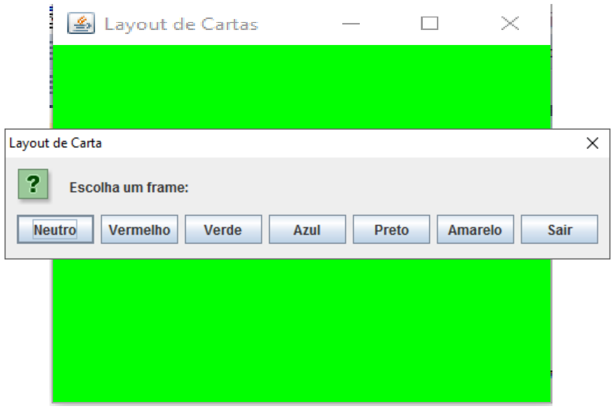


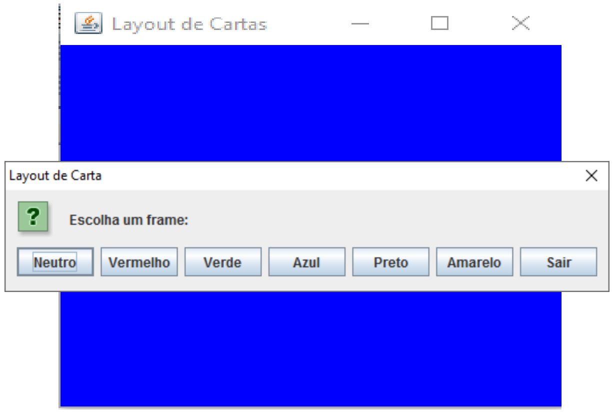


Layout de Carta

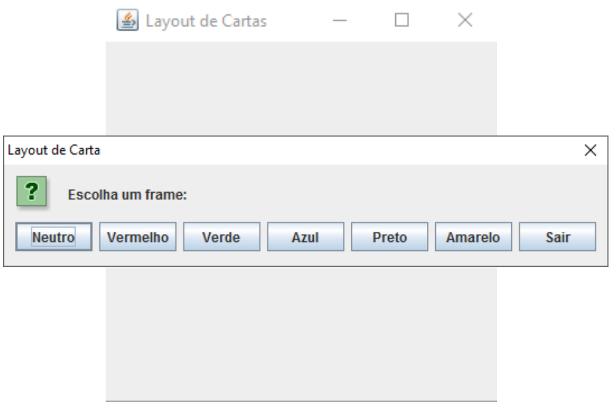




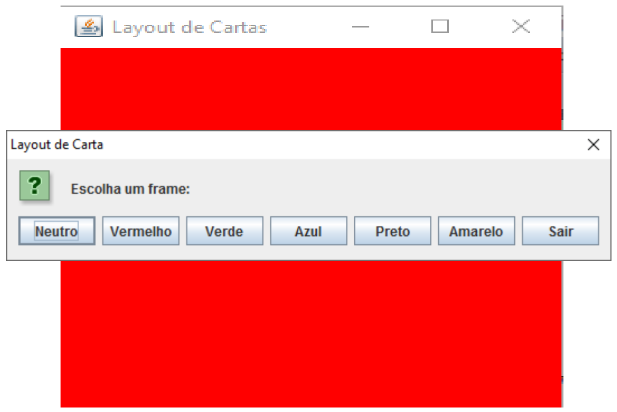




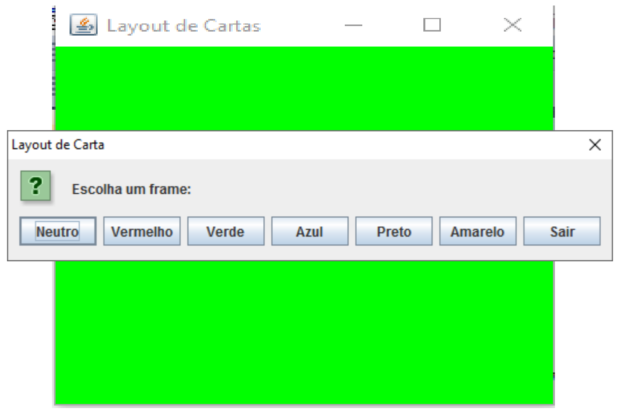
Layout de Carta



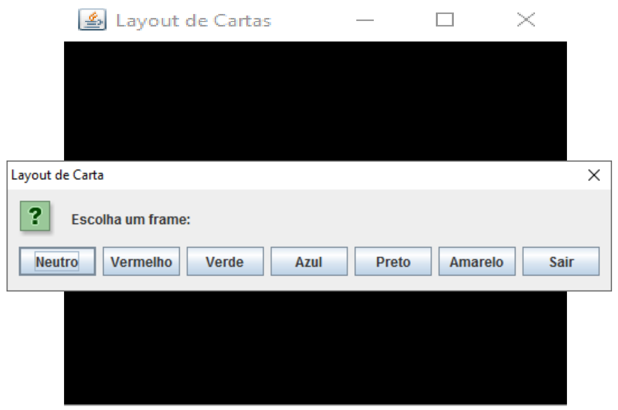
Escolha um frame:



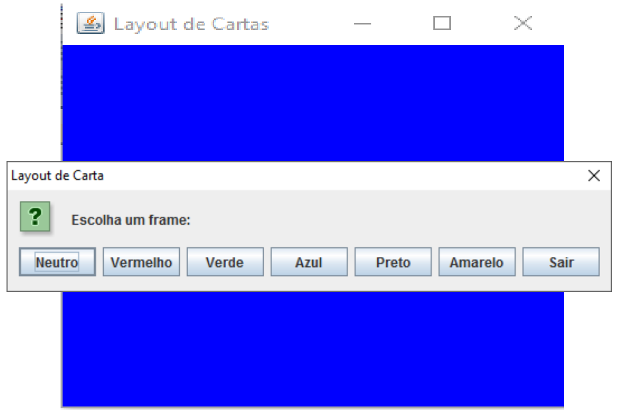
Escolha um frame:



Escolha um frame:

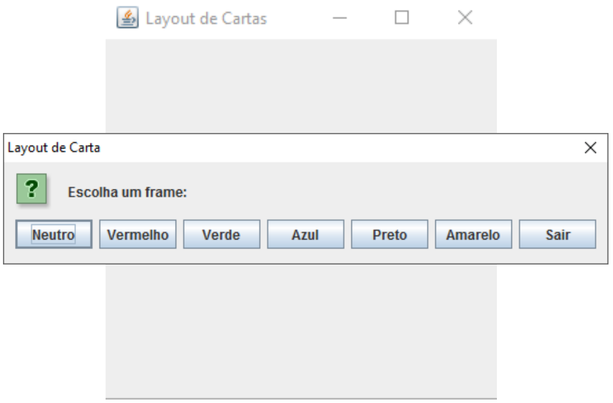


Escolha um frame:

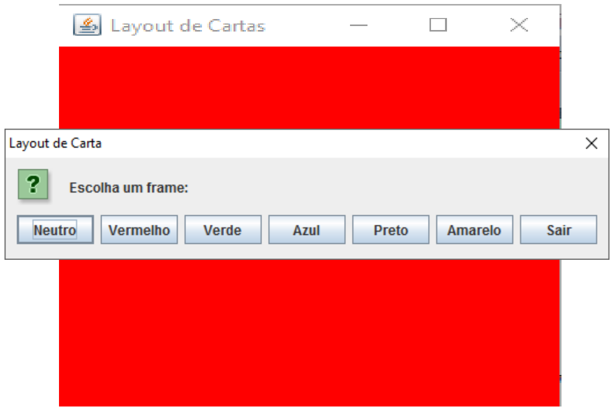


Escolha um frame:

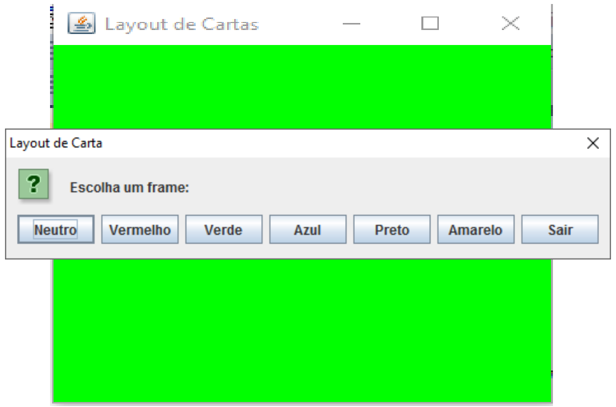
Layout de Carta



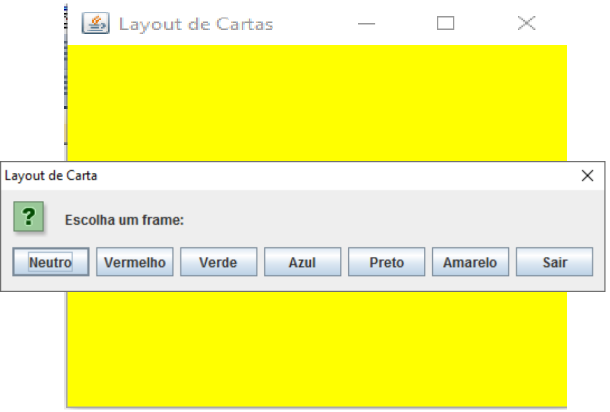
Escolha um frame:



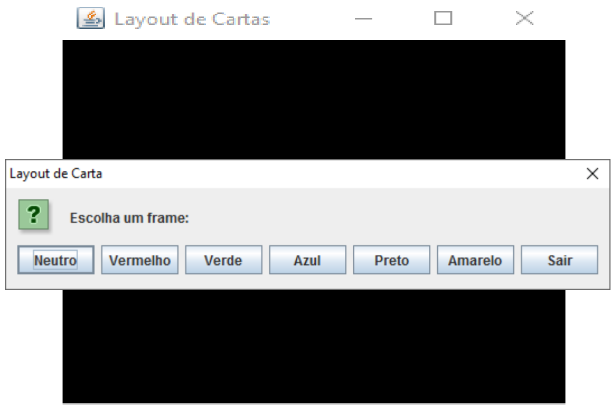
Escolha um frame:



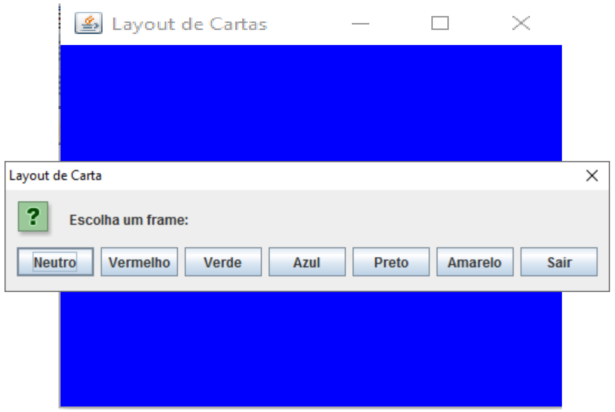
Escolha um frame:



Escolha um frame:



Escolha um frame:



Escolha um frame:

Layout GridBag

- Métodos Construtores:

Constructor and Description
GridBagLayout() Creates a grid bag layout manager.

- Principais Métodos:

Modifier and Type	Method and Description
void	addLayoutComponent(Component comp, Object constraints) Adds the specified component to the layout, using the specified constraints object.
void	setConstraints(Component comp, GridBagConstraints constraints) Sets the constraints for the specified component in this layout.

Layout GridBag

- É um dos mais poderosos layouts disponíveis no AWT do JAVA;
- É semelhante ao Layout de Grade, entretanto os componentes podem ocupar mais de uma célula da grade.
- Para cada componente adicionado ao container deve-se configurar um objeto do tipo GridBagConstraints e associá-lo ao componente e ao container usando o método setConstraints() do objeto GridBagLayout.

A Class GridBagConstraints

- Métodos Construtores:

Constructor and Description
`GridBagConstraints()` Creates a GridBagConstraint object with all of its fields set to their default value.
`GridBagConstraints(int gridx, int gridy, int gridwidth, int gridheight, double weightx, double weighty, int anchor, int fill,Insets insets, int ipadx, int ipady)` Creates a GridBagConstraints object with all of its fields set to the passed-in arguments.

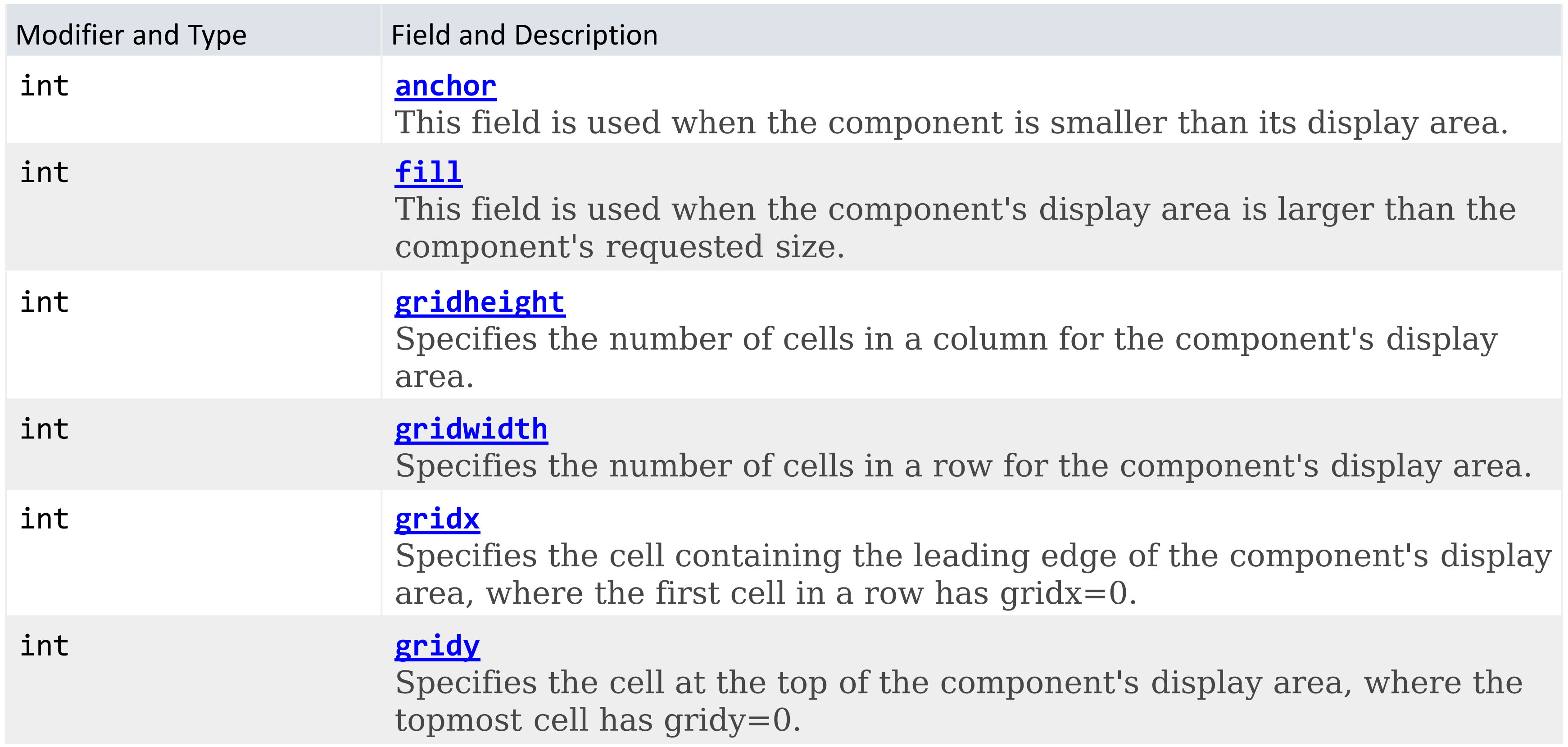
- Principais Métodos:

Modifier and Type	Method and Description
`Object`	`clone()` Creates a copy of this grid bag constraint.

  – Só possui os métodos herdados de Object:
    - equals(), finalize(), getClass(), hashCode(), notify(), notifyAll(), toString(), wait()

A Class GridBagConstraints

- Principais Atributos



Modifier and Type	Field and Description
`int`	**anchor** This field is used when the component is smaller than its display area.
`int`	**fill** This field is used when the component's display area is larger than the component's requested size.
`int`	**gridheight** Specifies the number of cells in a column for the component's display area.
`int`	**gridwidth** Specifies the number of cells in a row for the component's display area.
`int`	**gridx** Specifies the cell containing the leading edge of the component's display area, where the first cell in a row has gridx=0.
`int`	**gridy** Specifies the cell at the top of the component's display area, where the topmost cell has gridy=0.

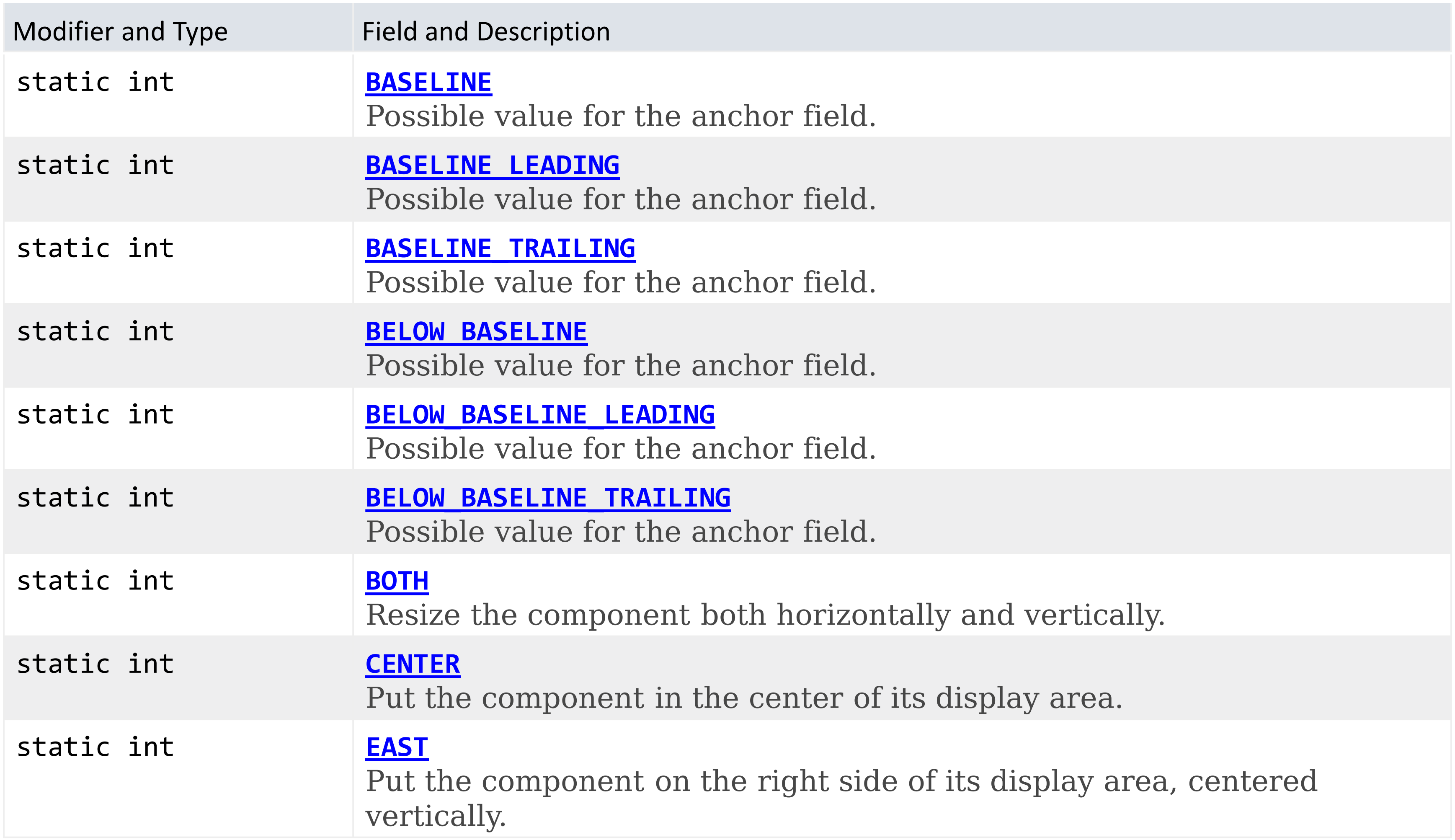
A Class GridBagConstraints

- Principais Atributos

Modifier and Type	Field and Description
double	**weightx** Specifies how to distribute extra horizontal space.
double	**weighty** Specifies how to distribute extra vertical space.
Insets	**insets** This field specifies the external padding of the component, the minimum amount of space between the component and the edges of its display area.
int	**ipadx** This field specifies the internal padding of the component, how much space to add to the minimum width of the component.
int	**ipady** This field specifies the internal padding, that is, how much space to add to the minimum height of the component.

A Class GridBagConstraints

- Principais Atributos de Classe

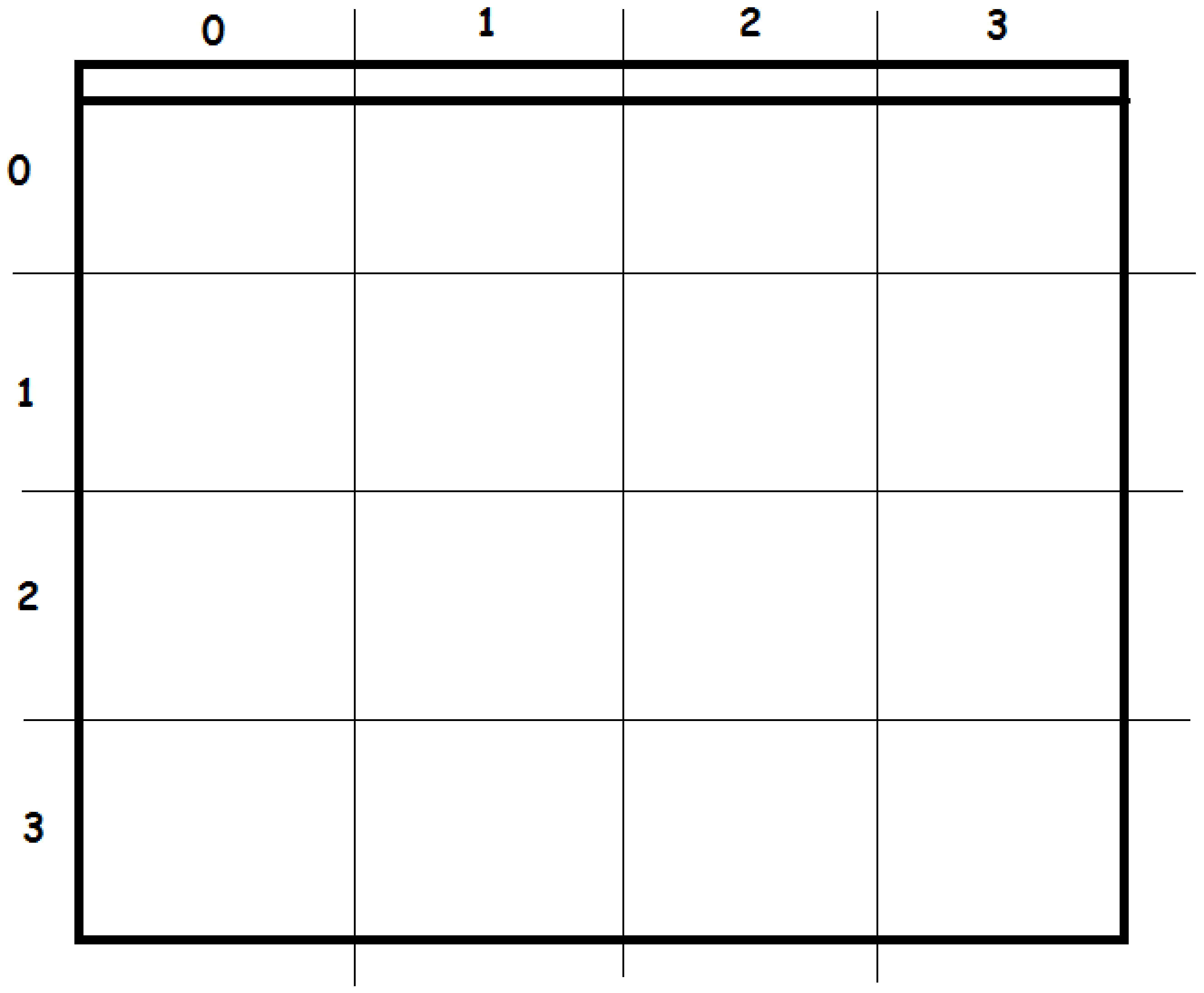


Modifier and Type	Field and Description
static int	BASELINE Possible value for the anchor field.
static int	BASELINE_LEADING Possible value for the anchor field.
static int	BASELINE_TRAILING Possible value for the anchor field.
static int	BELOW_BASELINE Possible value for the anchor field.
static int	BELOW_BASELINE_LEADING Possible value for the anchor field.
static int	BELOW_BASELINE_TRAILING Possible value for the anchor field.
static int	BOTH Resize the component both horizontally and vertically.
static int	CENTER Put the component in the center of its display area.
static int	EAST Put the component on the right side of its display area, centered vertically.

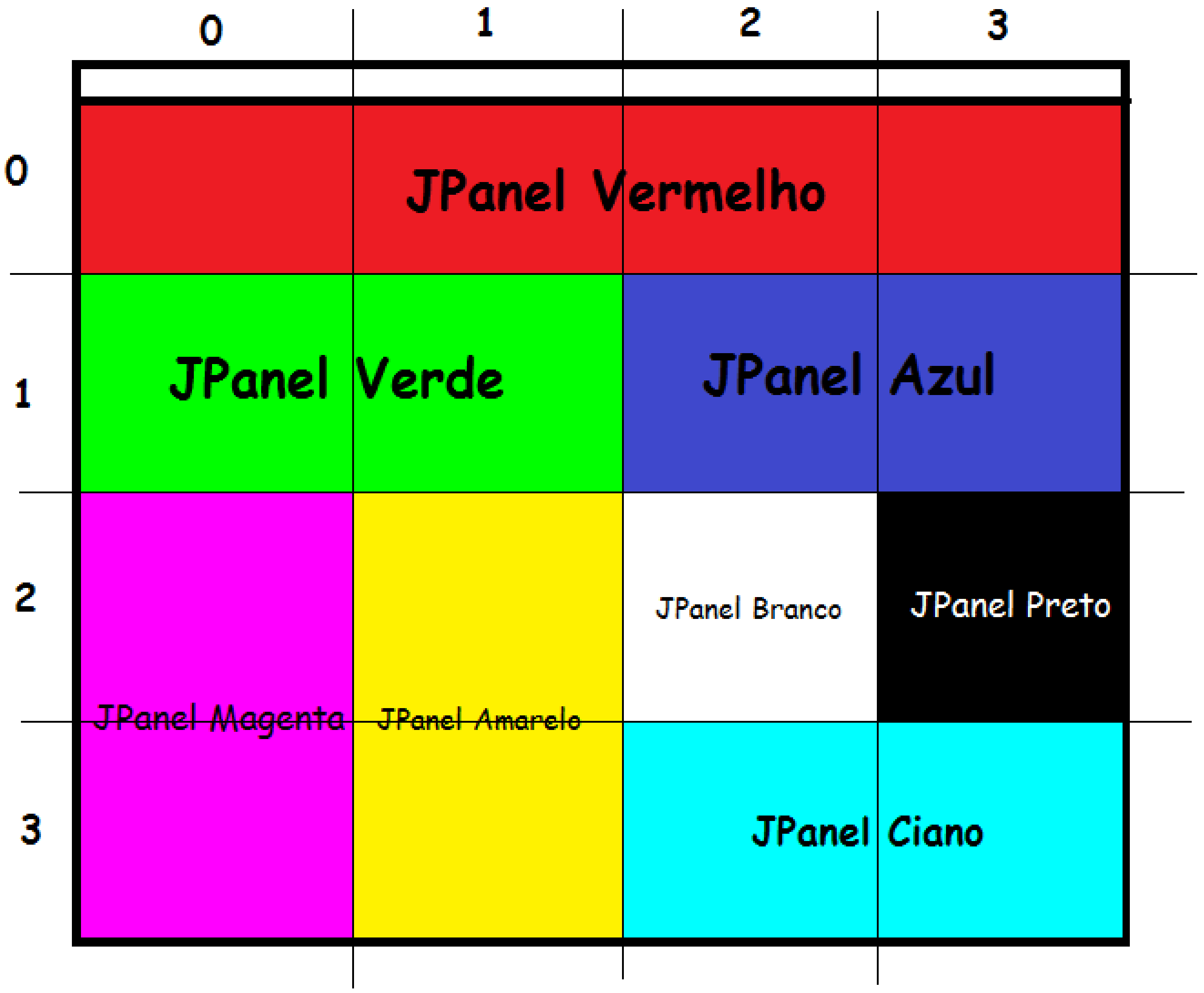
Layout GridBag

- Estratégia:
  1. Antes de começar a programação desenhe em um papel o layout que você deseja. Lembre-se que cada célula comporta apenas um componente, entretanto a recíproca não é verdadeira.

Layout GridBag



Layout GridBag



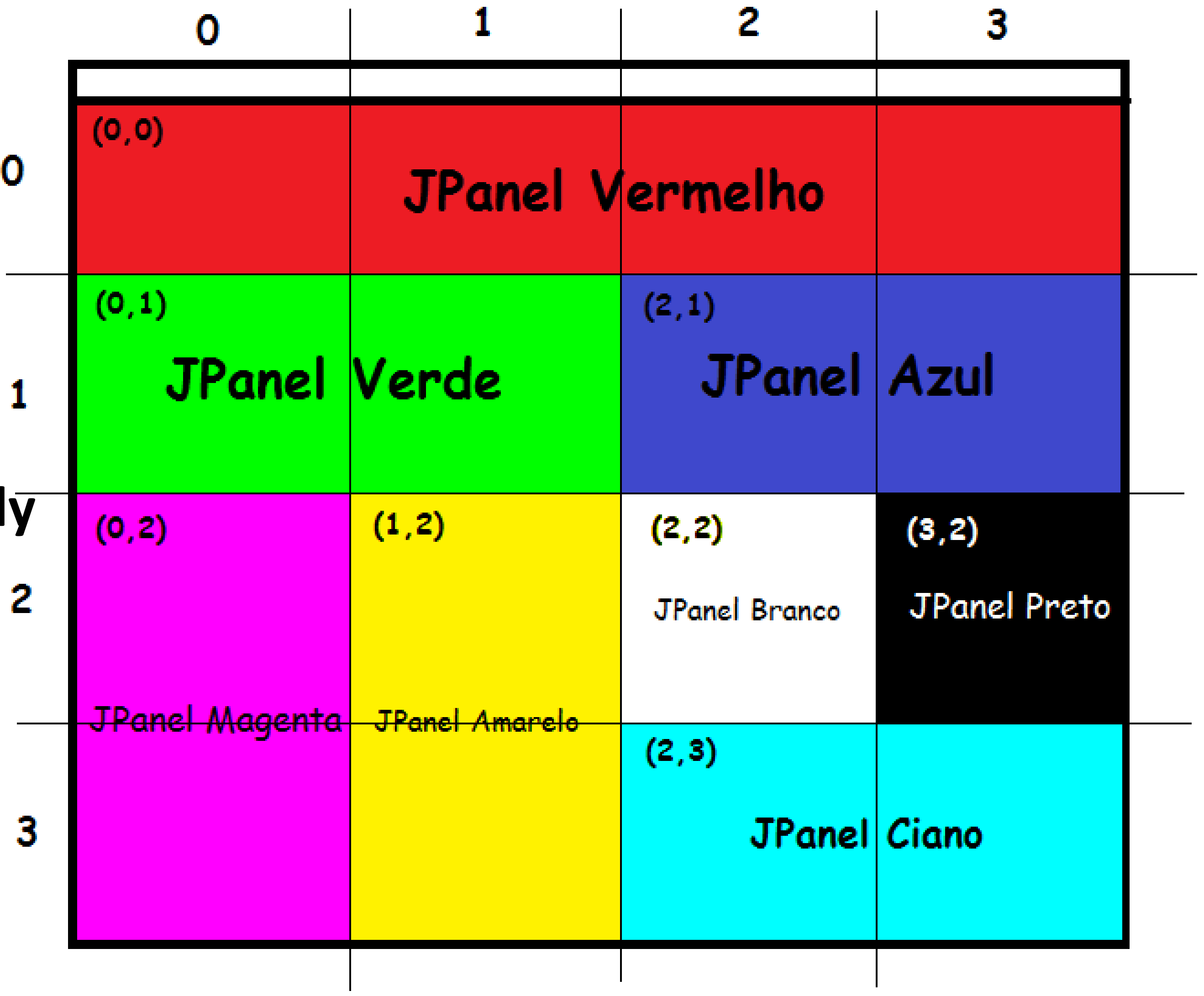
Layout GridBag

- Estratégia:
  1. Cria-se e configura-se todos os componentes a serem usados.
  2. Cria-se um objeto do tipo GridBagLayout.
  3. Cria-se um objeto do tipo GridBagConstraints.
  4. Vincula-se o layout ao container usando o método setLayout() do container.
  5. Configura-se os atributos gerais no GridBagConstraints.
  6. Para cada componente que se deseja adicionar ao container:
     1. Configura-se os parametros desejados no objeto GridBagLayout;
     2. Vincula-se ao layout usando o método setConstraints()
     3. Adiciona-se ao container.

Layout GridBag

- No exemplo:
  - Atributos de GridBagConstraints iguais para todos os componentes adicionados:
    - Atributo fill: GridBagConstraints.BOTH, para o componente extender-se por toda area a ele destinada.
    - Atributos weightx e weighty iguais a 1, para determinar que proporção entre as células é 100% em todas as direções.
  - Demais atributos são particulares para cada componente.

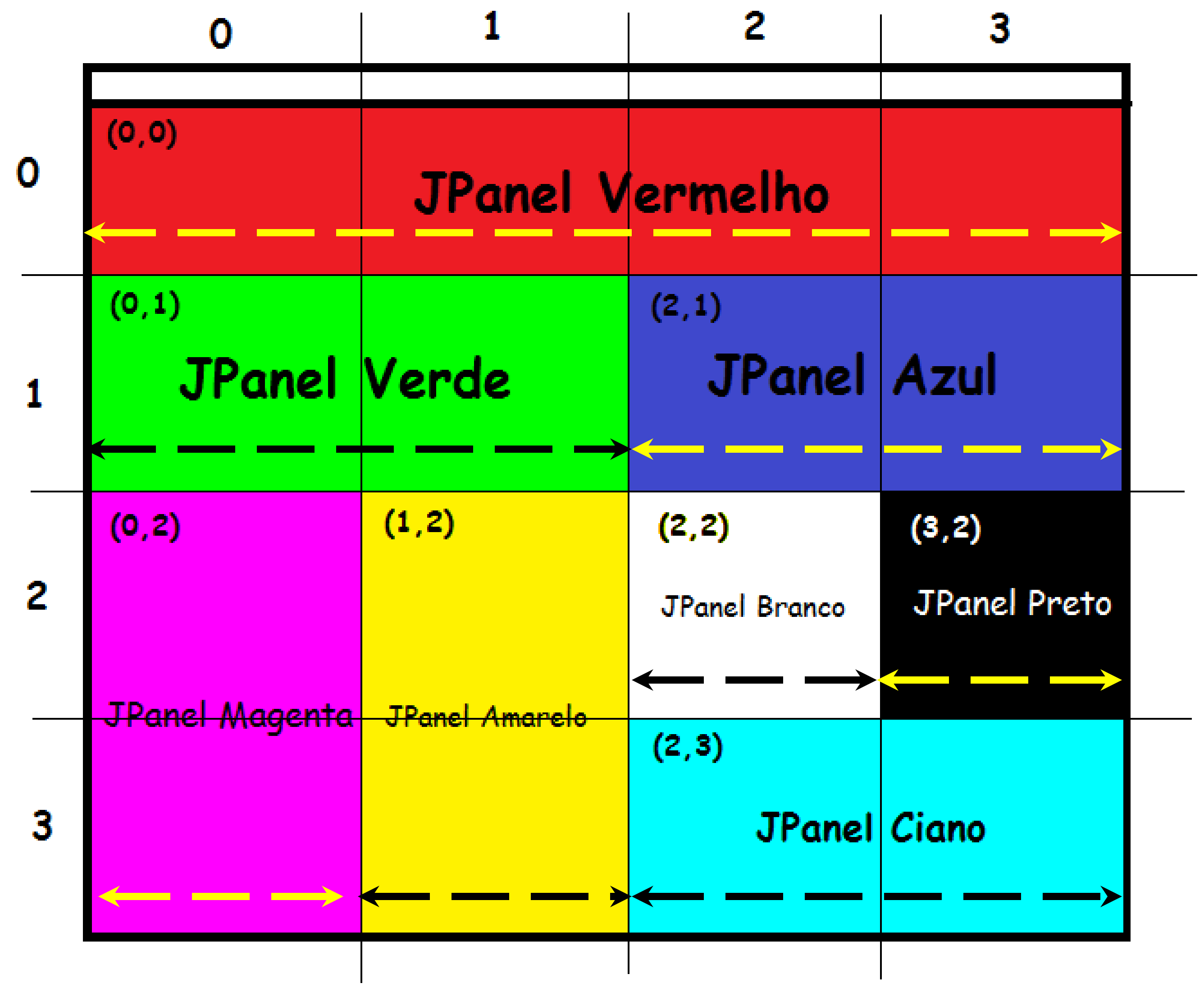
Layout GridBag



Atributos gridx e gridy

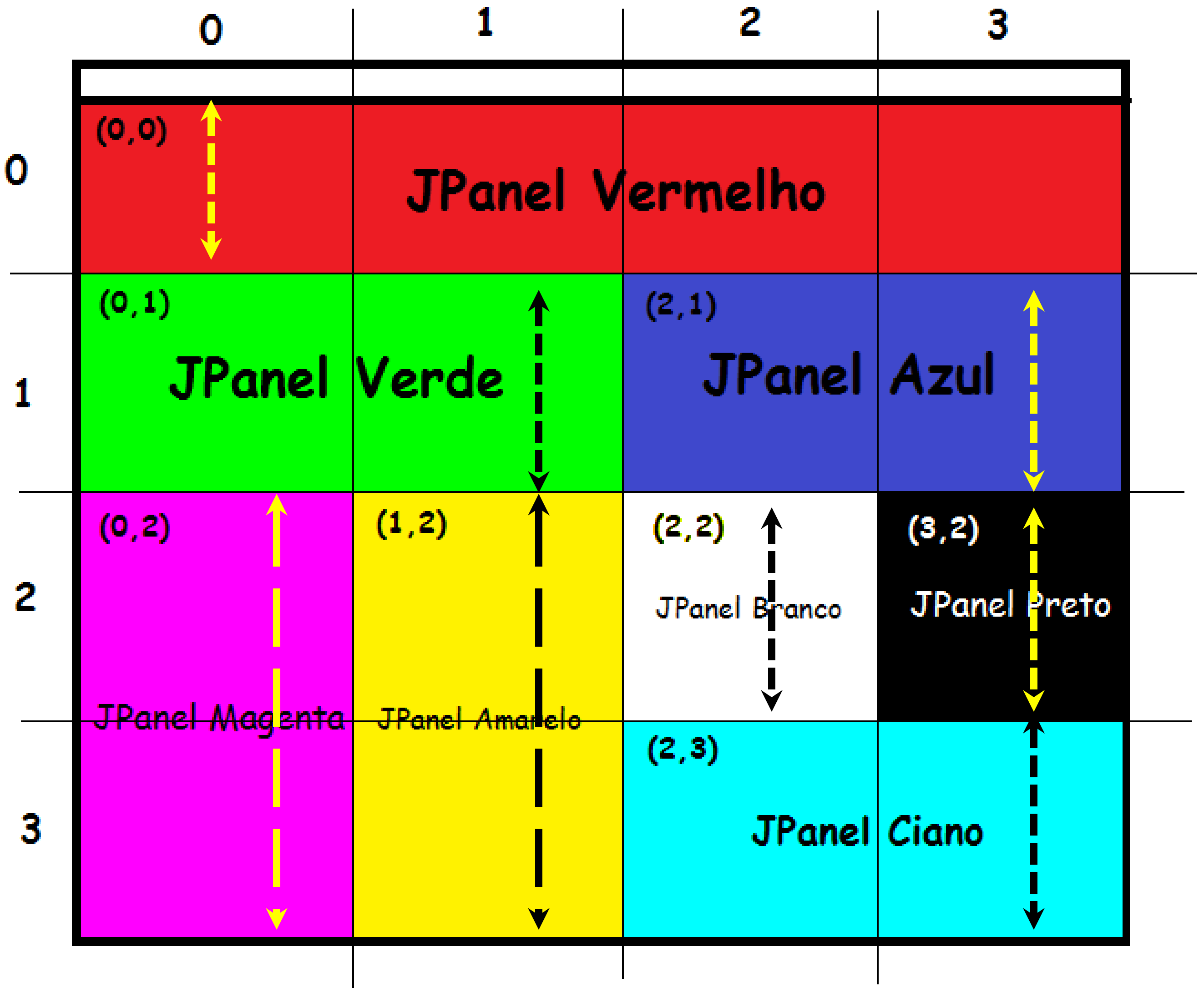
Layout GridBag

Atributos gridwidth



Layout GridBag

Atributos gridheight



Layout GridBag

- Estratégia:
  1. Cria-se e configura-se todos os componentes a serem usados.
  2. Cria-se um objeto do tipo GridBagLayout.
  3. Cria-se um objeto do tipo GridBagConstraints.
  4. Vincula-se o layout ao container usando o método setLayout() do container.
  5. Configura-se os atributos gerais no GridBagConstraints.
  6. Para cada componente que se deseja adicionar ao container:
     1. Configura-se os parametros desejados no objeto GridBagLayout;
     2. Vincula-se ao layout usando o método setConstraints()
     3. Adiciona-se ao container.

Layout GridBag

```
import java.awt.GridBagLayout;
import java.awt.GridBagConstraints;
import java.awt.Insets;
import java.awt.Color;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class GridBag  {
       public static void main(String[] args)  {
              JFrame janela = new JFrame("Layout de GridBag");
              GridBagLayout lay = new GridBagLayout();
              GridBagConstraints cons = new GridBagConstraints();
              JPanel paneRed = new JPanel();
              JPanel paneGreen = new JPanel();
              JPanel paneBlue = new JPanel();
              JPanel paneYellow = new JPanel();
              JPanel paneWhite = new JPanel();
              JPanel paneMagenta = new JPanel();
              JPanel paneCyan = new JPanel();
              JPanel paneBlack = new JPanel();

              paneRed.setBackground(Color.RED);
              paneGreen.setBackground(Color.GREEN);
              paneBlue.setBackground(Color.BLUE);
              paneYellow.setBackground(Color.YELLOW);
              paneWhite.setBackground(Color.WHITE);
              paneMagenta.setBackground(Color.MAGENTA);
              paneCyan.setBackground(Color.CYAN);
              paneBlack.setBackground(Color.BLACK);
```

Layout GridBag

- Estratégia:
  1. Cria-se e configura-se todos os componentes a serem usados.
  2. Cria-se um objeto do tipo GridBagLayout.
  3. Cria-se um objeto do tipo GridBagConstraints.
  4. Vincula-se o layout ao container usando o método setLayout() do container.
  5. Configura-se os atributos gerais no GridBagConstraints.
  6. Para cada componente que se deseja adicionar ao container:
     1. Configura-se os parametros desejados no objeto GridBagLayout;
     2. Vincula-se ao layout usando o método setConstraints()
     3. Adiciona-se ao container.

Layout GridBag

```
janela.setLayout(lay);

cons.fill = GridBagConstraints.BOTH;
cons.weightx = 0.5;
cons.weighty = 0.5;
```

Layout GridBag

- Estratégia:
  1. Cria-se e configura-se todos os componentes a serem usados.
  2. Cria-se um objeto do tipo GridBagLayout.
  3. Cria-se um objeto do tipo GridBagConstraints.
  4. Vincula-se o layout ao container usando o método setLayout() do container.
  5. Configura-se os atributos gerais no GridBagConstraints.
  6. Para cada componente que se deseja adicionar ao container:
     1. Configura-se os parametros desejados no objeto GridBagLayout;
     2. Vincula-se ao layout usando o método setConstraints()
     3. Adiciona-se ao container.

Layout GridBag

```
cons.gridx = 0;
cons.gridy = 1;
cons.gridwidth = 2;
cons.gridheight = 1;
lay.setConstraints(paneGreen, cons);
janela.add(paneGreen);

cons.gridx = 2;
cons.gridy = 1;
cons.gridwidth = 2;
cons.gridheight = 1;
lay.setConstraints(paneBlue, cons);
janela.add(paneBlue);

cons.gridx = 0;
cons.gridy = 2;
cons.gridwidth = 1;
cons.gridheight = 2;
lay.setConstraints(paneMagenta, cons);
janela.add(paneMagenta);

cons.gridx = 1;
cons.gridy = 2;
cons.gridwidth = 1;
cons.gridheight = 2;
lay.setConstraints(paneYellow, cons);
janela.add(paneYellow);
```

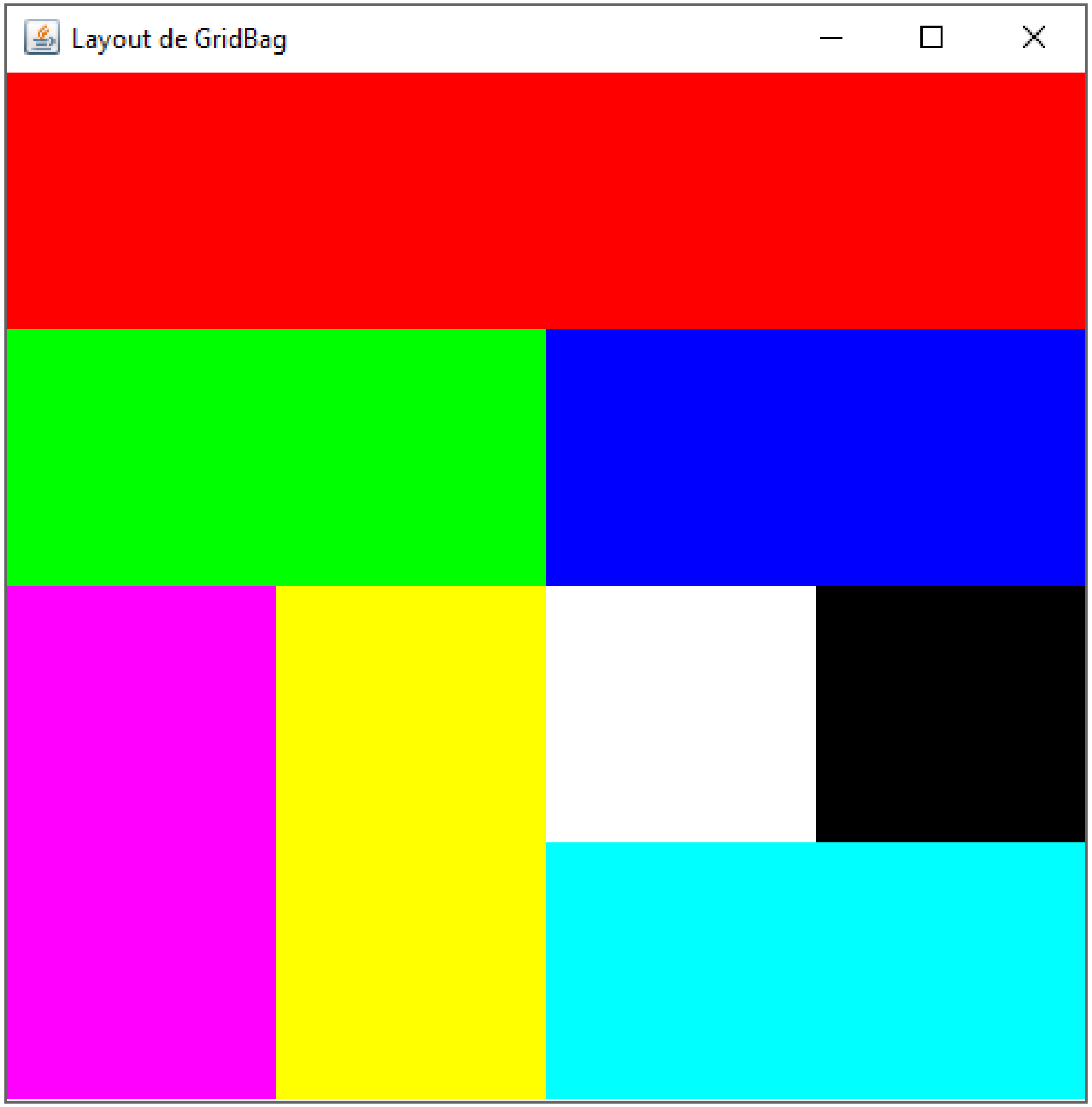
```
cons.gridx = 2;
cons.gridy = 2;
cons.gridwidth = 1;
cons.gridheight = 1;
lay.setConstraints(paneWhite, cons);
janela.add(paneWhite);

cons.gridx = 3;
cons.gridy = 2;
cons.gridwidth = 1;
cons.gridheight = 1;
lay.setConstraints(paneBlack, cons);
janela.add(paneBlack);

cons.gridx = 2;
cons.gridy = 3;
cons.gridwidth = 2;
cons.gridheight = 1;
lay.setConstraints(paneCyan, cons);
janela.add(paneCyan);
```

Layout GridBag

- Configurando detalhes do container:

```
        janela.setSize(500, 500);
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        janela.setVisible(true);
    }
}
```

Layout GridBag



Layout GridBag

- É recomendado que se implemente um método axiliar para configurar o objeto GridBagConstraint e reduzir as linhas de código.

```
public static GridBagConstraints addConstraints( int gx, int gy, int gw, int gh, double wx, double wy, int a,
                                                 int f, int top, int left, int bottom, int right, int ix, int iy)
{
   return new GridBagConstraints(gx, gy, gw, gh, wx, wy, a, f, new Insets(top, left, bottom, right), ix, iy);
}
```

Layout GridBag

- É recomendado que se implemente um método axiliar para configurar o objeto GridBagConstraint e reduzir as linhas de código.

```
public static GridBagConstraints addConstraints( int gx, int gy, int gw, int gh, double wx, double wy, int a,
                                                 int f, int top, int left, int bottom, int right, int ix, int iy)
{
   return new GridBagConstraints(gx, gy, gw, gh, wx, wy, a, f, new Insets(top, left, bottom, right), ix, iy);
}

lay.setConstraints(paneRed, addConstraints(0, 0, 4, 1, 0.5, 0.5, GridBagConstraints.BASELINE,
                                           GridBagConstraints.BOTH, 0, 0, 0, 0, 0, 0));
janela.add(paneRed);

lay.setConstraints(paneGreen, addConstraints(0, 1, 2, 1, 0.5, 0.5, GridBagConstraints.BASELINE,
                                             GridBagConstraints.BOTH, 0, 0, 0, 0, 0, 0));
janela.add(paneGreen);

lay.setConstraints(paneBlue, addConstraints(2, 1, 2, 1, 0.5, 0.5, GridBagConstraints.BASELINE,
                                            GridBagConstraints.BOTH, 0, 0, 0, 0, 0, 0));
janela.add(paneBlue);

lay.setConstraints(paneMagenta, addConstraints(0, 2, 1, 2, 0.5, 0.5, GridBagConstraints.BASELINE,
                                               GridBagConstraints.BOTH, 0, 0, 0, 0, 0, 0));
janela.add(paneMagenta);
```

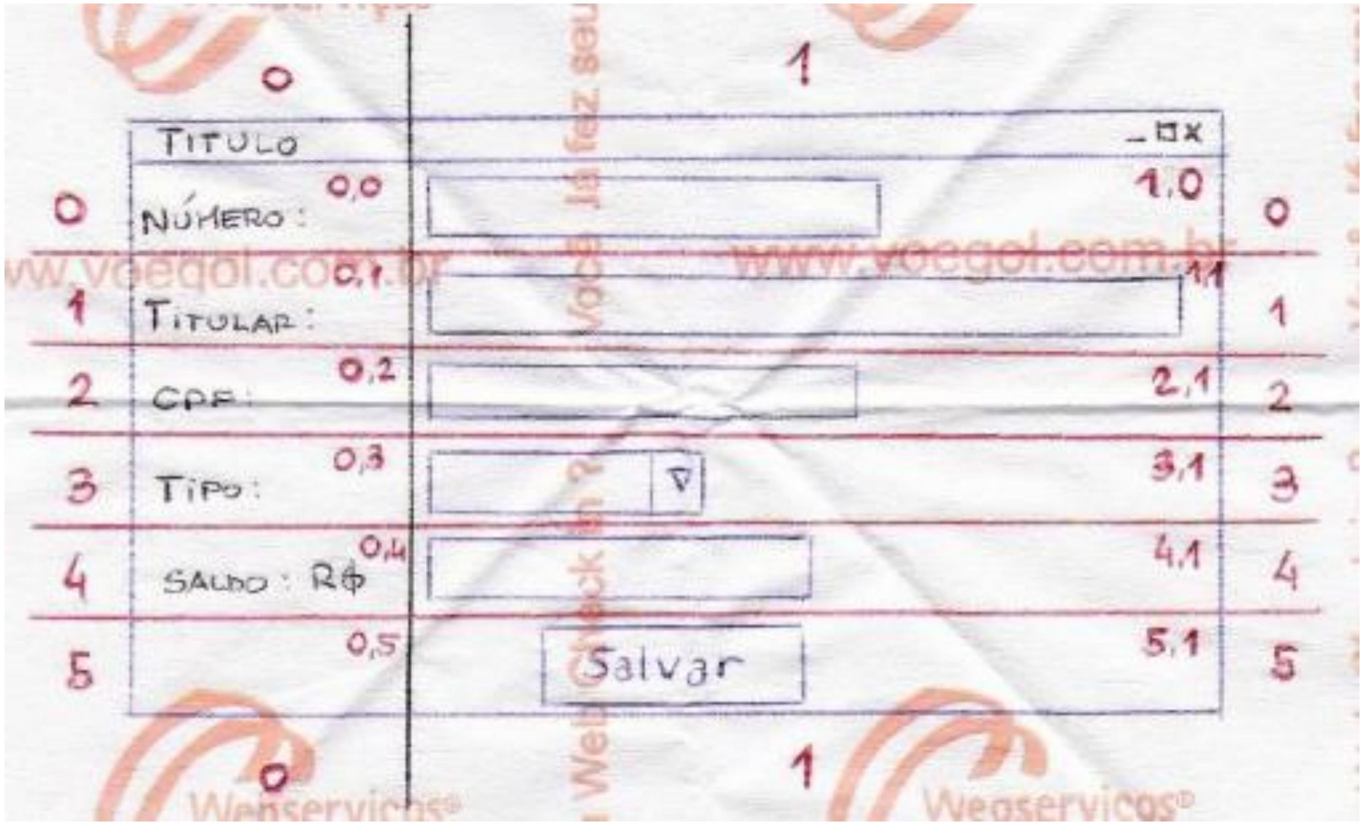
Layout GridBag

- É recomendado que se implemente um método axiliar para configurar o objeto GridBagConstraint e reduzir as linhas de código.

```
public static GridBagConstraints addConstraints( int gx, int gy, int gw, int gh, double wx, double wy, int a,
                                                 int f, int top, int left, int bottom, int right, int ix, int iy)
{
   return new GridBagConstraints(gx, gy, gw, gh, wx, wy, a, f, new Insets(top, left, bottom, right), ix, iy);
}

lay.setConstraints(paneYellow, addConstraints(1, 2, 1, 2, 0.5, 0.5, GridBagConstraints.BASELINE,
                                              GridBagConstraints.BOTH, 0, 0, 0, 0, 0, 0));
janela.add(paneYellow);

lay.setConstraints(paneWhite, addConstraints(2, 2, 1, 1, 0.5, 0.5, GridBagConstraints.BASELINE,
                                             GridBagConstraints.BOTH, 0, 0, 0, 0, 0, 0));
janela.add(paneWhite);

lay.setConstraints(paneBlack, addConstraints(3, 2, 1, 1, 0.5, 0.5, GridBagConstraints.BASELINE,
                                             GridBagConstraints.BOTH, 0, 0, 0, 0, 0, 0));
janela.add(paneBlack);

lay.setConstraints(paneCyan, addConstraints(2, 3, 2, 1, 0.5, 0.5, GridBagConstraints.BASELINE,
                                            GridBagConstraints.BOTH, 0, 0, 0, 0, 0, 0));
janela.add(paneCyan);
```

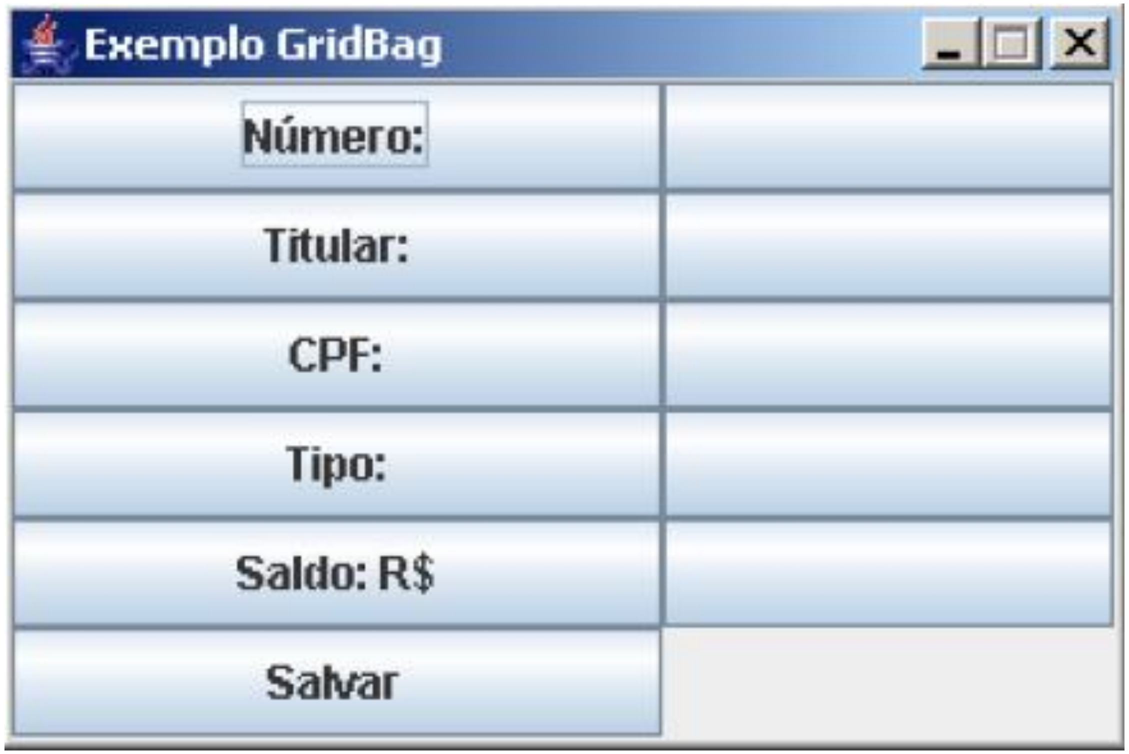
Layout GridBag

- Outro exemplo de uso do Layout GridBag



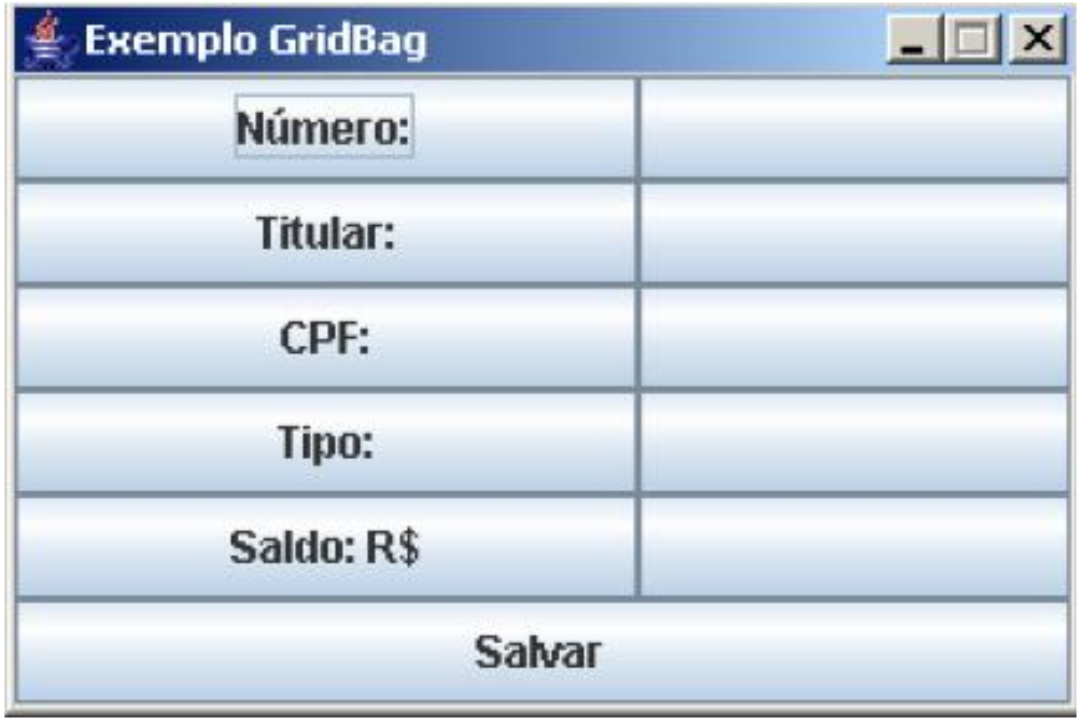
Layout GridBag

- Outro exemplo de uso do Layout GridBag



Layout GridBag

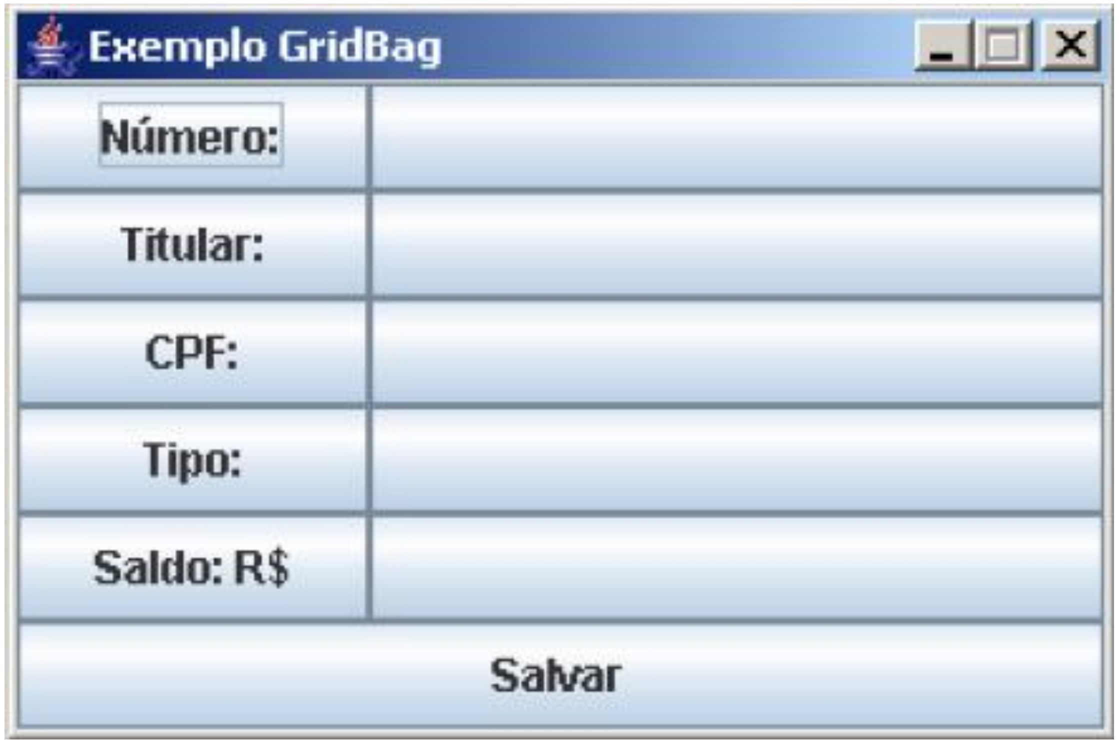
- Outro exemplo de uso do Layout GridBag



- Atributo `gridwidth` e `gridheight`

Layout GridBag

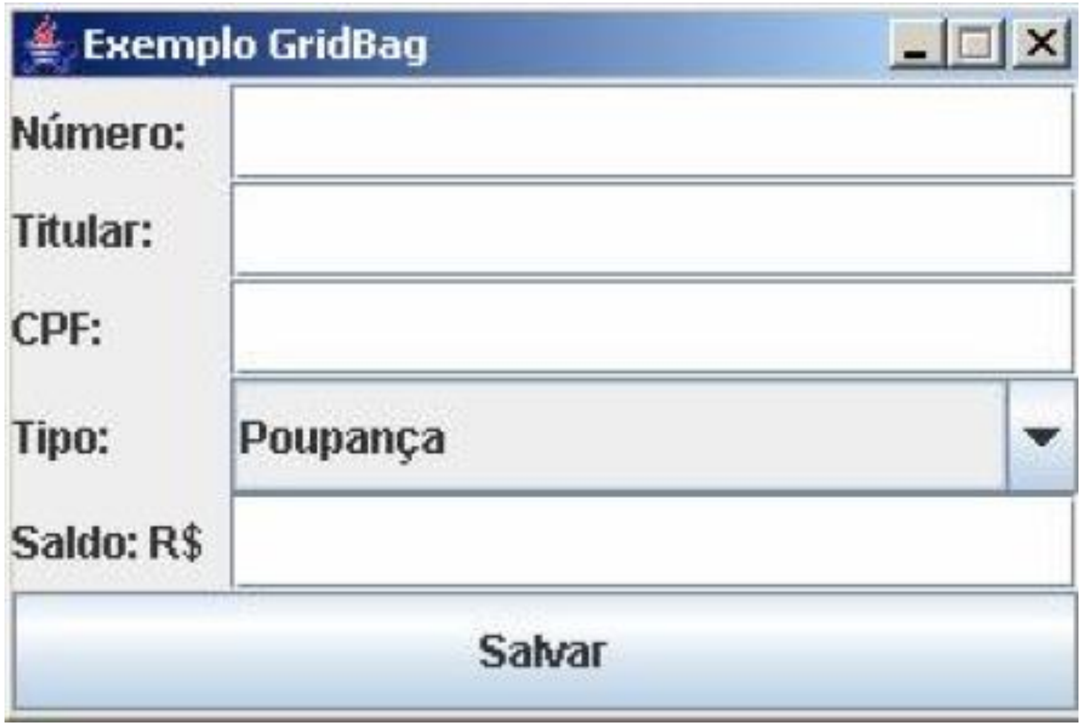
- Outro exemplo de uso do Layout GridBag



- Atributo `weightx` e `weighty`

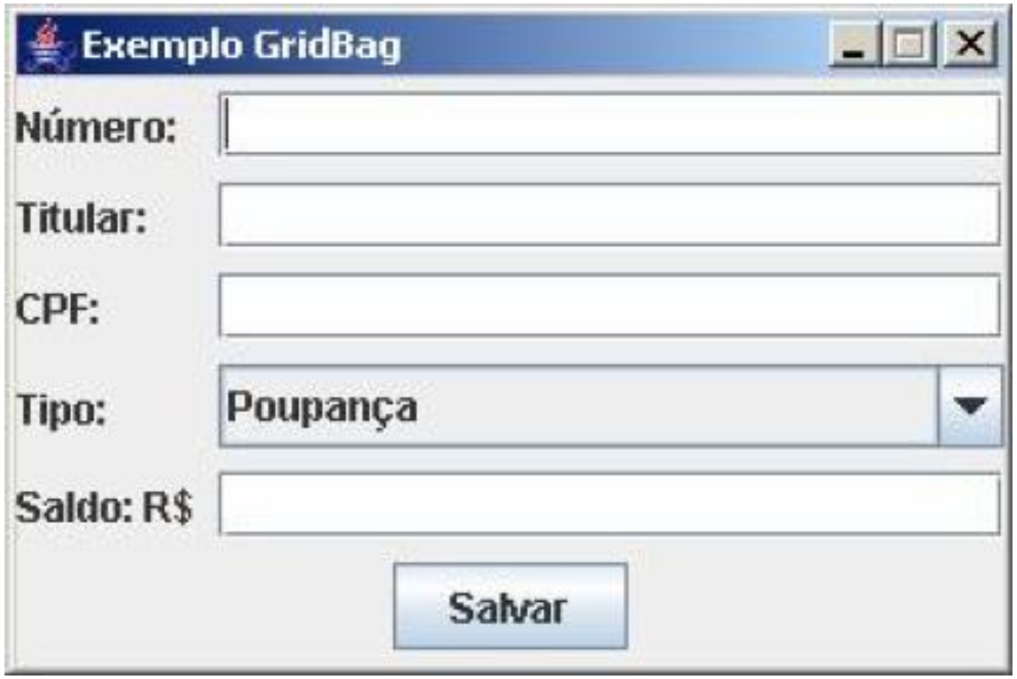
Layout GridBag

- Outro exemplo de uso do Layout GridBag



Layout GridBag

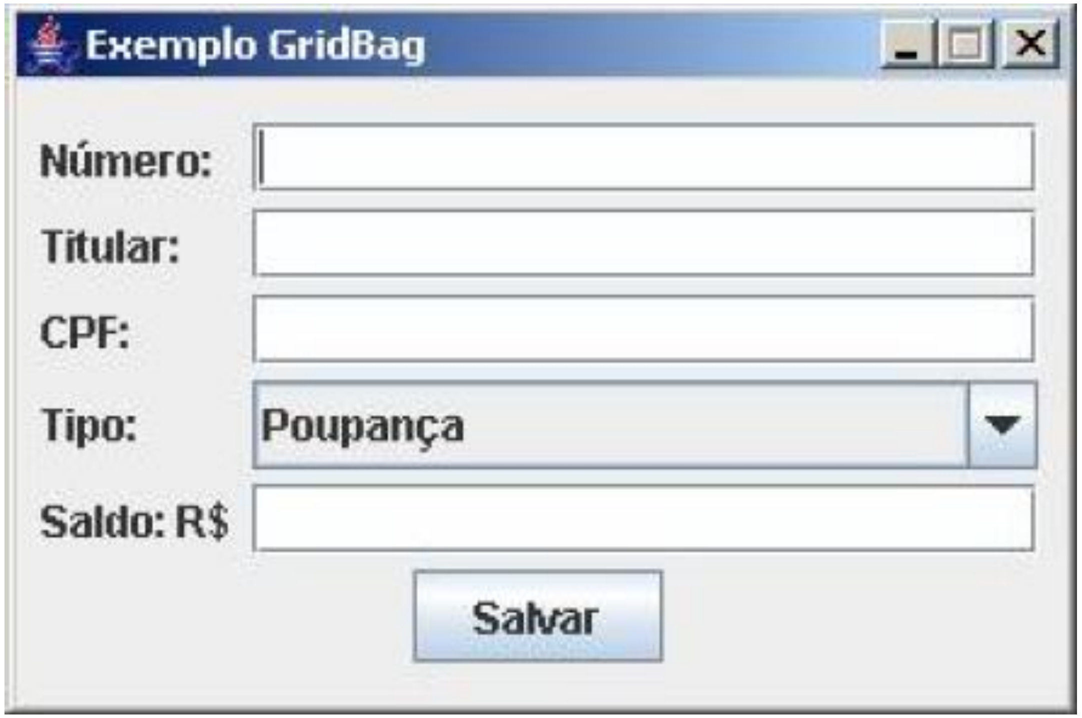
- Outro exemplo de uso do Layout GridBag



- Atributos `anchor` e `fill`

Layout GridBag

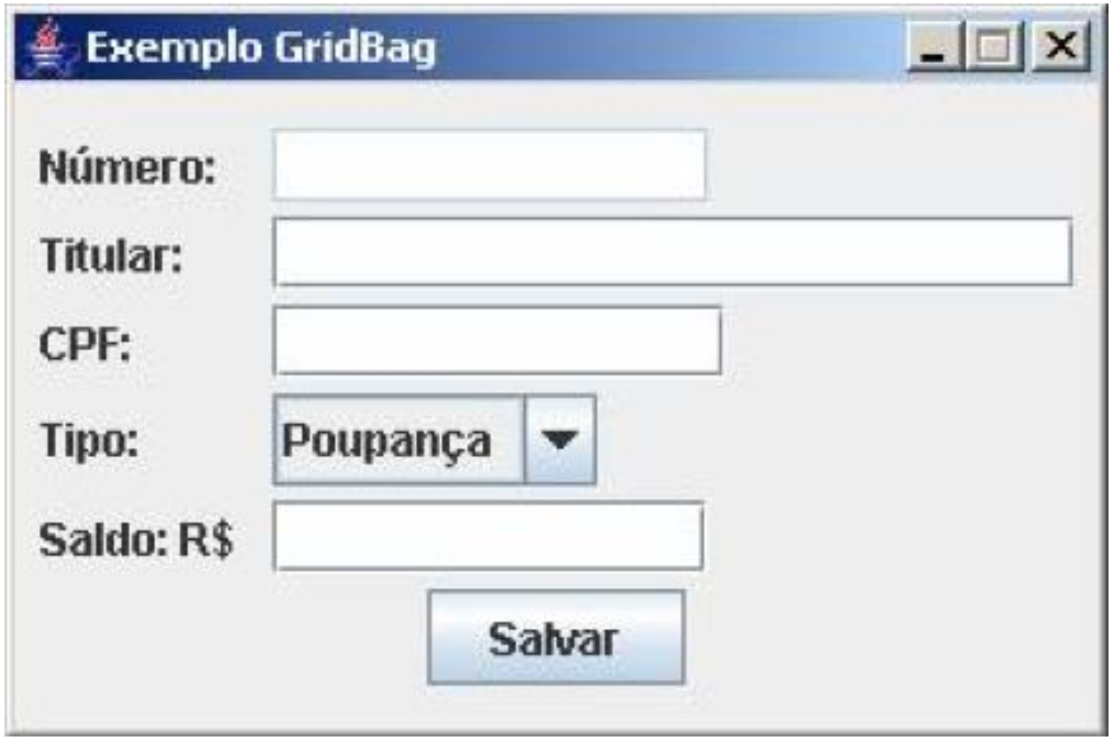
- Outro exemplo de uso do Layout GridBag



- Sobreescrever o Método `getInsets()` instanciando um objeto `Insets`.

Layout GridBag

- Outro exemplo de uso do Layout GridBag



Número:

Titular:

CPF:

Tipo: Poupança

Saldo: R$

- Sobreescrever o Método `getInsets()` instanciando um objeto `Insets`.

A Classe JButton

- Métodos Construtores:

Constructor and Description
`JButton()` Creates a button with no set text or icon.
`JButton(Action a)` Creates a button where properties are taken from the Action supplied.
`JButton(Icon icon)` Creates a button with an icon.
`JButton(String text)` Creates a button with text.
`JButton(String text, Icon icon)` Creates a button with initial text and an icon.

- Principal Método herdado de AbstractButton

Modifier and Type	Method and Description
void	`addActionListener(ActionListener l)` Adds an ActionListener to the button.

A Classe JButton

- Adicionando um Botão usando o Layout GridBag.

```
import javax.swing.JFrame;
import javax.swing.JButton;
import java.awt.GridBagLayout;
import java.awt.GridBagConstraints;

class Janela extends JFrame {
    private JButton btnHello;
    public Janela() {
         super("Tratando Evento");
         this.btnHello = new JButton("Aperte-me");
         GridBagLayout gbl = new GridBagLayout();
         GridBagConstraints gbc = new GridBagConstraints();

         gbc.gridx = 0;
         gbc.gridy = 0;
         gbc.fill = GridBagConstraints.NONE;
         gbl.setConstraints(btnHello, gbc);
         this.setLayout(gbl);
         this.add(btnHello);
         this.setSize(500,300);
         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         this.setVisible(true);
    }
}
```

A Classe JButton

- Adicionando um Botão usando o Layout GridBag.

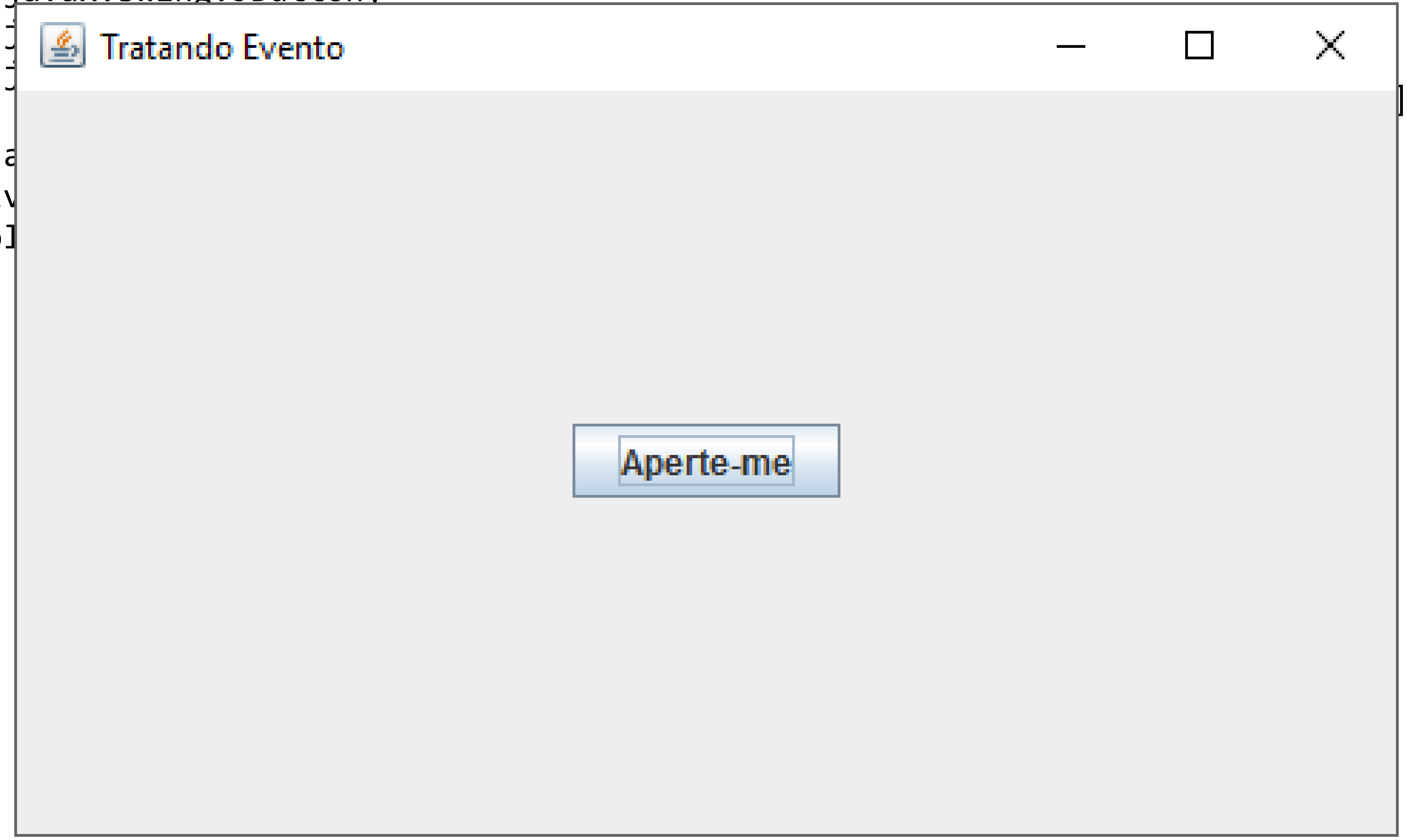
```
import javax.swing.JFrame;
import javax.swing.JButton;
import java.awt.GridBagLayout;
import java.awt.GridBagConstraints;

class Janela extends JFrame {
    private JButton btnHello;
    public Janela() {
         super("Tratando Evento");
         this.btnHello = new JButton("Aperte-me");
         GridBagLayout gbl = new GridBagLayout();
         GridBagConstraints gbc = new GridBagConstraints();

         gbc.gridx = 0;
         gbc.gridy = 0;
         gbc.fill = GridBagConstraints.NONE;
         gbl.setConstraints(btnHello, gbc);
         this.setLayout(gbl);
         this.add(btnHello);
         this.setSize(500,300);
         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         this.setVisible(true);
    }
}
```

```
public class Aplicacao {
    public static void main(String[] args) {
         Janela jan = new Janela();
    }
}
```

A Classe JButton

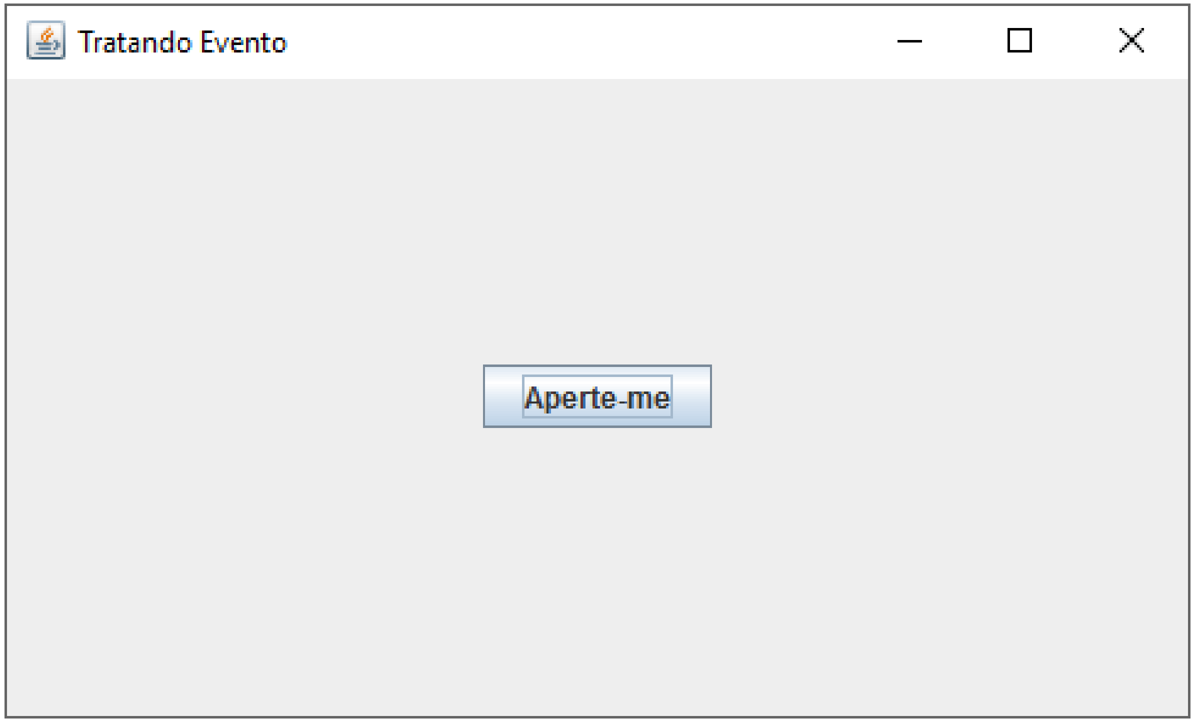
- Adicionando um Botão usando o Layout GridBag.

```
import javax.swing.JFrame;
import javax.swing.JButton;
import j
import j
                                                                     ] args) {
class Ja
    priv
    publ







                                              
    }
}
```



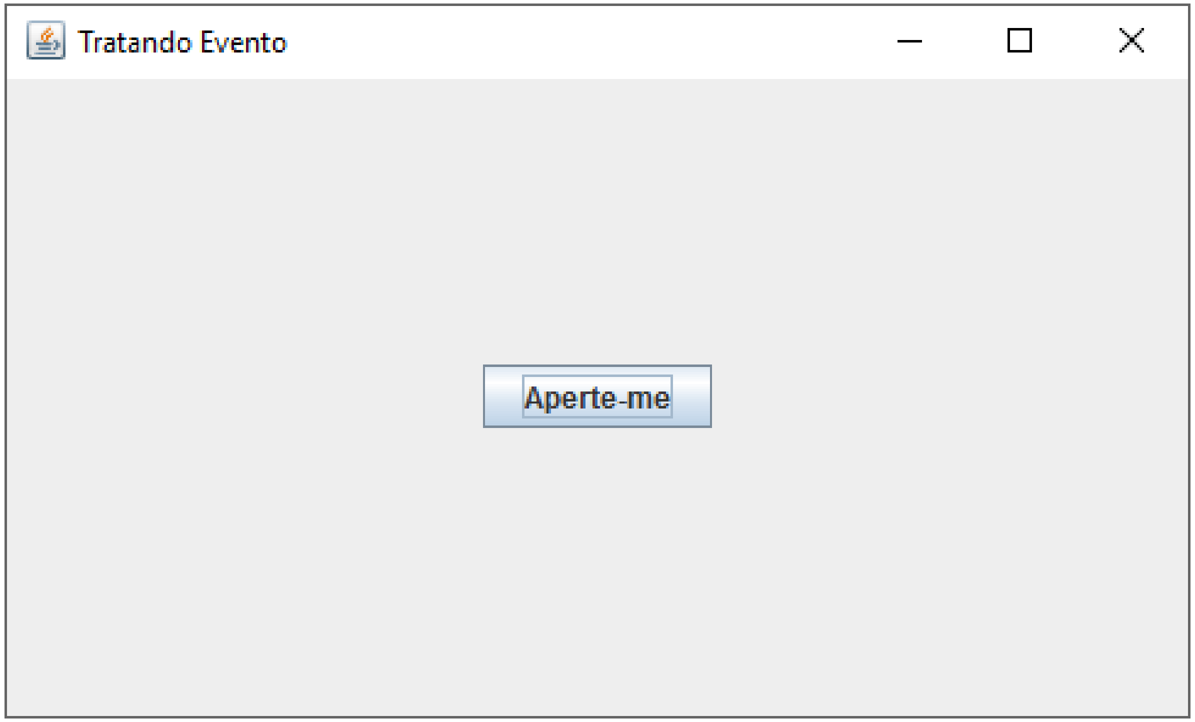
A Classe JButton

- Entretanto ao clicar sobre o botão nada acontece.



A Classe JButton

- Entretanto ao clicar sobre o botão nada acontece.
- Para isso, deve-se capturar o evento e tratá-lo.



Tratando Eventos no JButton

- Entretanto ao clicar sobre o botão nada acontece.
- Para isso, deve-se capturar o evento e tratá-lo.
- Isso é possível implementando a interface `ActionListener`; que por sua vez, exige a implementação do método público `actionPerformed()`.
- Métodos da Interface ActionListener:

Modifier and Type	Method and Description
void	actionPerformed(ActionEvent e) Invoked when an action occurs.

Tratando Eventos no JButton

- Quando o componente `JButton` sofre um evento do tipo clique, tal evento pode ser capturado e tratado em um método chamado `actionPerformed()`.
- Para isso o botão deve ser vinculado à um objeto que implemente a classe `ActionListener` através do método `addActionListener()`; e por sua vez, esta classe deve implementar o método `actionPerformed()`.
- No método `actionPerformed()` é possível saber de qual componente partiu o evento através do método `getSource()`, que retorna um `Objetc`.

Tratando Eventos no JButton

- Importanto as Classes e Interfaces:

```
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.JOptionPane;
import java.awt.GridBagLayout;
import java.awt.GridBagConstraints;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
```

Tratando Eventos no JButton

- Adicionando um manipulador de eventos ao JButton

```
class Janela extends JFrame implements ActionListener{
    private JButton btnHello;
    private int apertos;
    public Janela() {
         super("Tratando Evento");
         this.btnHello = new JButton("Aperte-me");
         this.apertos = 0;
         GridBagLayout gbl = new GridBagLayout();
         GridBagConstraints gbc = new GridBagConstraints();

         gbc.gridx = 0;
         gbc.gridy = 0;
         gbc.fill = GridBagConstraints.NONE;
         gbl.setConstraints(btnHello, gbc);

         btnHello.addActionListener(this);

         this.setLayout(gbl);
         this.add(btnHello);
         this.setSize(500,300);
         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         this.setVisible(true);
    }
}
```

Tratando Eventos no JButton

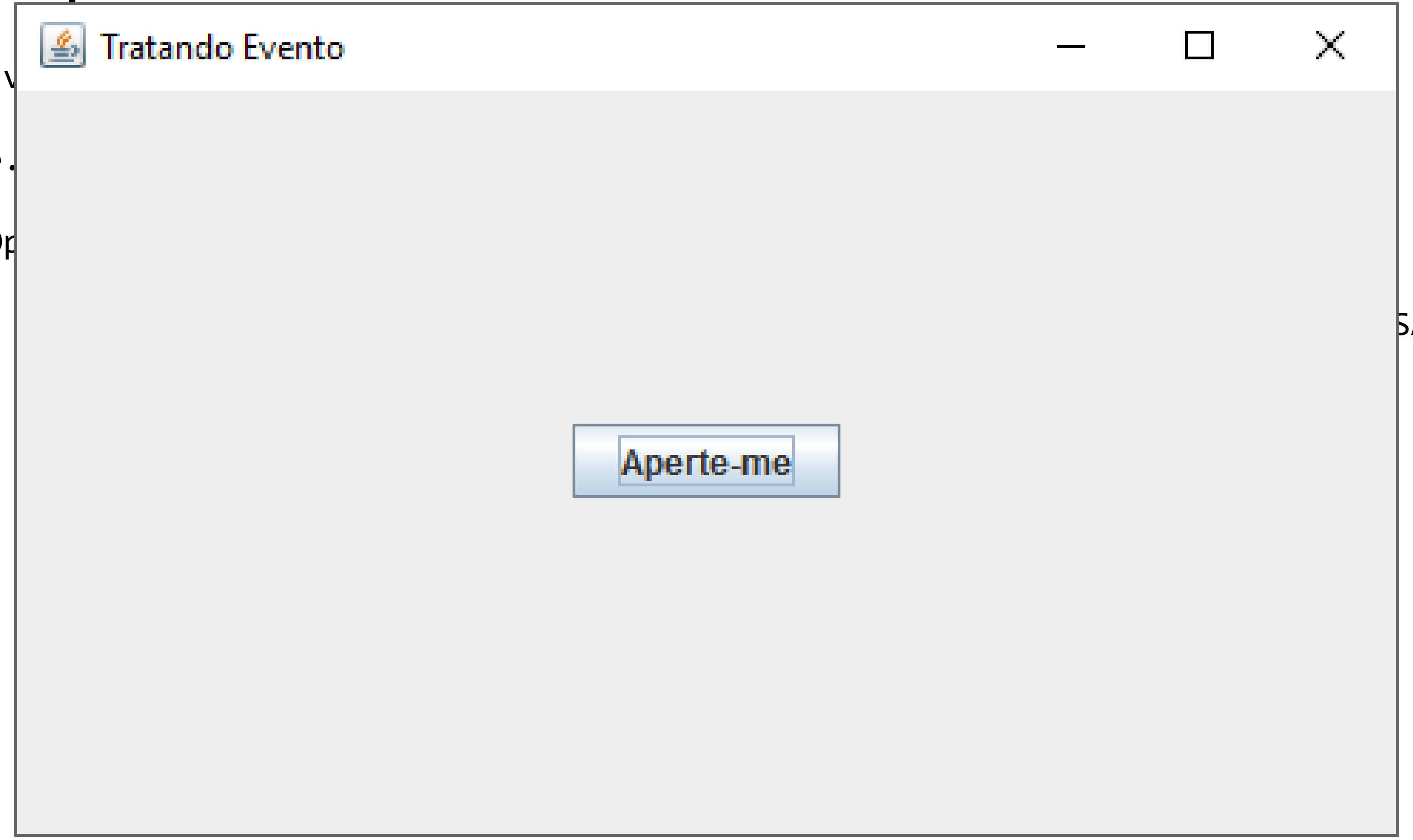
- Implementando o método da interface

```
public void actionPerformed(ActionEvent e)
{
   if(e.getSource() == btnHello)
   {
     JOptionPane.showMessageDialog(null, "Botao foi apertado " +
                                       (++this.apertos) + " vezes.",
                                    "Apertou o botão", JOptionPane.INFORMATION_MESSAGE);
   }
}
```

Tratando Eventos no JButton

- Implementando o método da interface

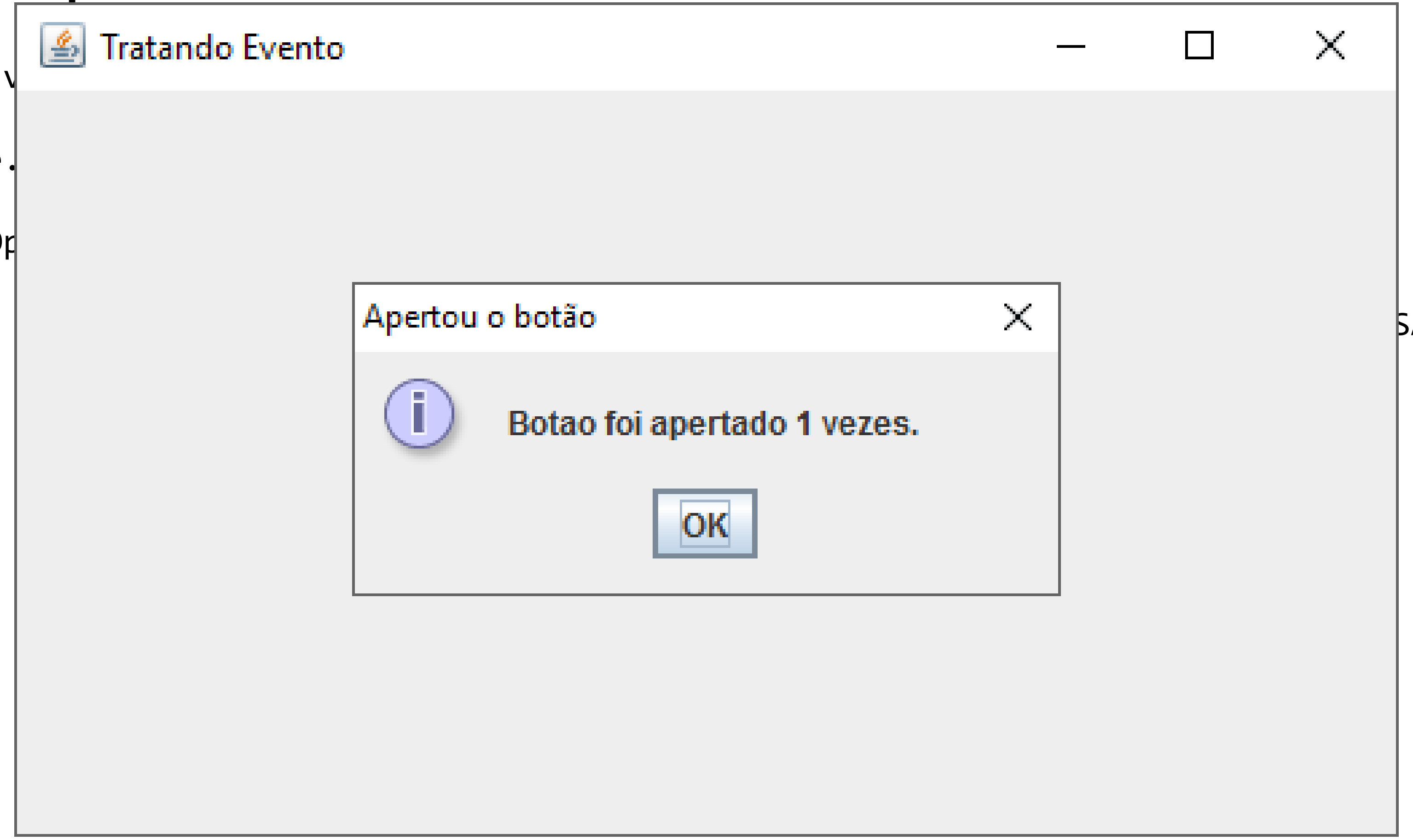
```
public v
{
   if(e.
   {
     JOp

                                                                   SAGE);
   }
}
```



Tratando Eventos no JButton

- Implementando o método da interface

```
public v
{
   if(e.
   {
     JOp
                                                                    SAGE);
   }
}
```


Tratando Evento

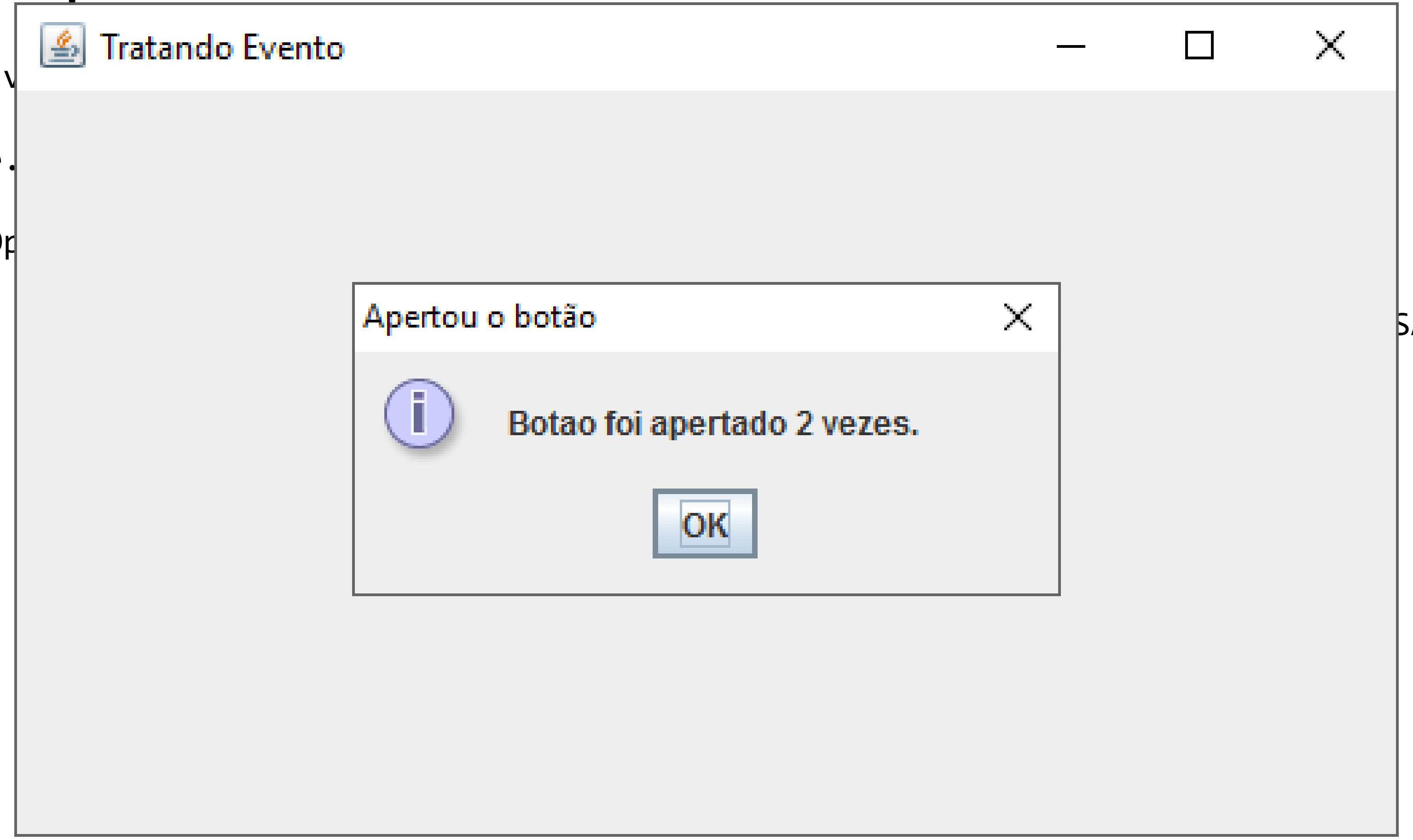
Apertou o botão

Botao foi apertado 1 vezes.

Tratando Eventos no JButton

- Implementando o método da interface

```
public v
{
   if(e.
   {
     JOp
                                                                    SAGE);
   }
}
```



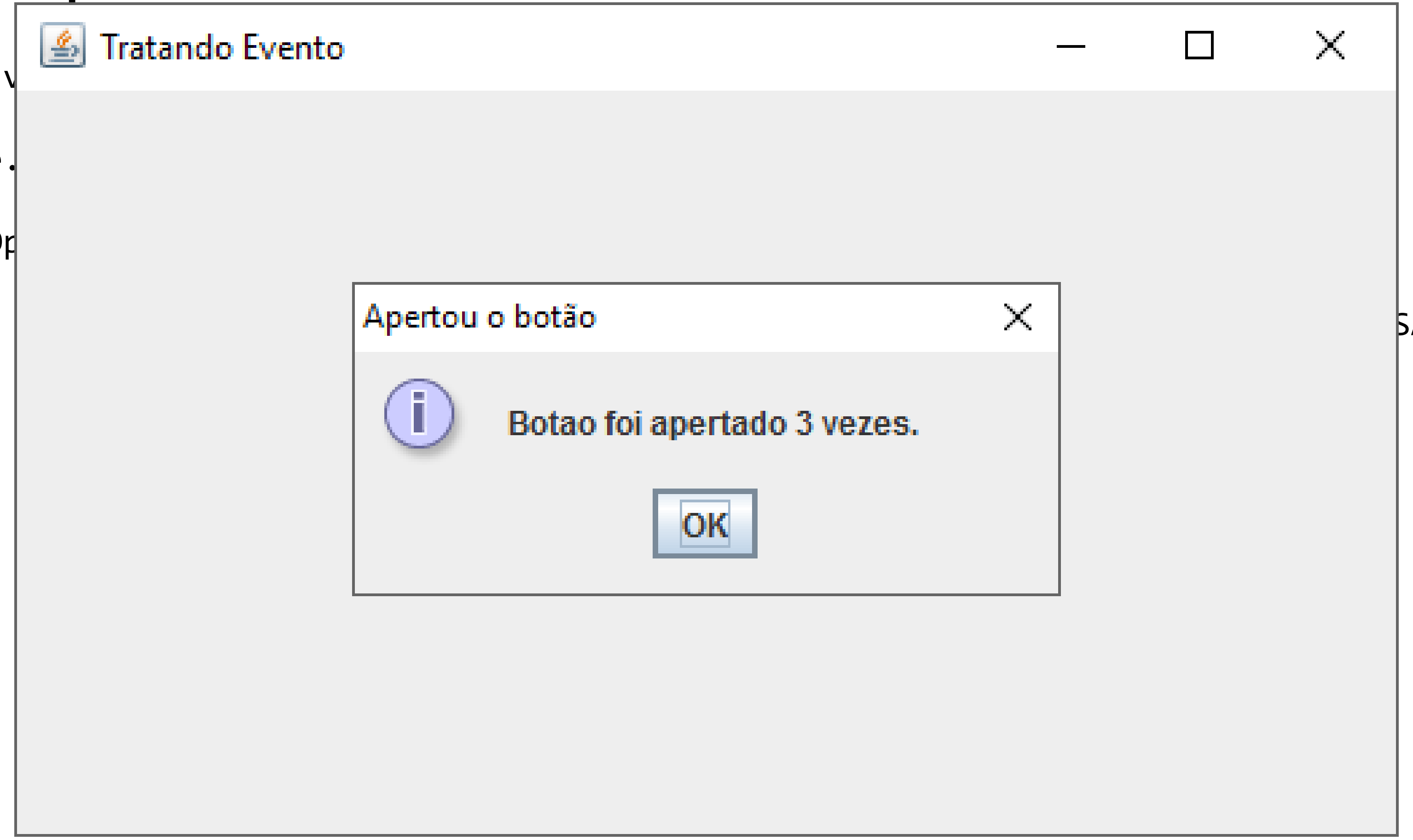
Apertou o botão

Botao foi apertado 2 vezes.

Tratando Eventos no JButton

- Implementando o método da interface

```
public v
{
   if(e.
   {
      JOp
                                                                         SAGE);
   }
}
```



Tratando Evento

Apertou o botão

Botao foi apertado 3 vezes.

OK

Tratando Evento de Teclas

- Da mesma forma que a implementação da interface `ActionListener` é necessaria para capturar e tratar um evento de clique em um componente `JButton`. A interface `KeyListener` é responsavel por manupular eventos de pressão de teclas.
- Métodos da Interface KeyListener:

Modifier and Type	Method and Description
`void`	`keyPressed(KeyEvent e)` Invoked when a key has been pressed.
`void`	`keyReleased(KeyEvent e)` Invoked when a key has been released.
`void`	`keyTyped(KeyEvent e)` Invoked when a key has been typed.

A Classe KeyEvent

- Principais Métodos

Modifier and Type	Method and Description
`int`	`getKeyCode()` Returns the integer keyCode associated with the key in this event.
`static String`	`getKeyText(int keyCode)` Returns a String describing the keyCode, such as "HOME", "F1" or "A".

- Principais Atributos

Modifier and Type	Field and Description
`static int`	`VK_A` VK_A thru VK_Z are the same as ASCII 'A' thru 'Z' (0x41 - 0x5A)
`static int`	`VK_ENTER`
`static int`	`VK_0` VK_0 thru VK_9 are the same as ASCII '0' thru '9' (0x30 - 0x39)
`static int`	`VK_ESCAPE`

Tratando Evento de Teclas

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import java.awt.BorderLayout;
import java.awt.CardLayout;
import java.awt.Color;
import java.awt.event.KeyListener;
import java.awt.event.KeyEvent;

class TrocaTecla extends JFrame
implements KeyListener
{
       private JLabel lblAviso;
       private JPanel panelPrin;
       private JPanel panels[];
```

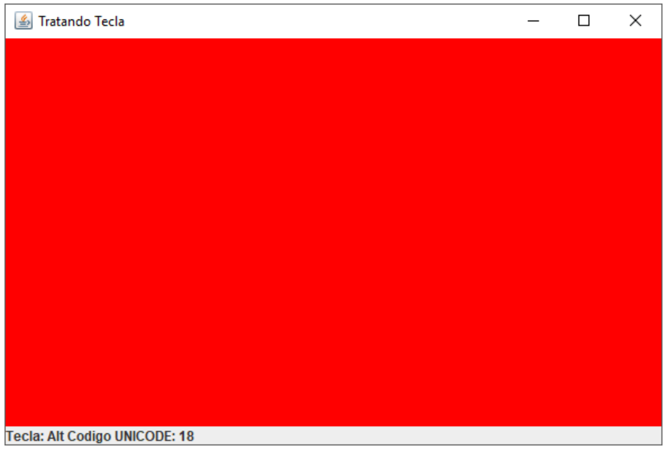
Tratando Evento de Teclas

```
public TrocaTecla()  {
       super("Tratando Tecla");
       this.lblAviso = new JLabel("Aperte qualquer tecla para ver os dados ou ENTER para
                                                             mudar de cor um ESC para Sair");
       this.panelPrin = new JPanel();
       this.panels = new JPanel[5];
       this.panelPrin.setLayout(new CardLayout());
       this.panels[0] = new JPanel();
       this.panels[0].setBackground(Color.RED);
       this.panelPrin.add(this.panels[0], "paneRed");
       this.panels[1] = new JPanel();
       this.panels[1].setBackground(Color.GREEN);
       this.panelPrin.add(this.panels[1], "paneGreen");
       this.panels[2] = new JPanel();
       this.panels[2].setBackground(Color.BLUE);
       this.panelPrin.add(this.panels[2], "paneBlue");
       this.panels[3] = new JPanel();
       this.panels[3].setBackground(Color.YELLOW);
       this.panelPrin.add(this.panels[3], "paneYellow");
       this.panels[4] = new JPanel();
       this.panels[4].setBackground(Color.MAGENTA);
       this.panelPrin.add(this.panels[4], "paneMagenta");
       this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       this.setSize(600,400);
       this.setVisible(true);
}
```

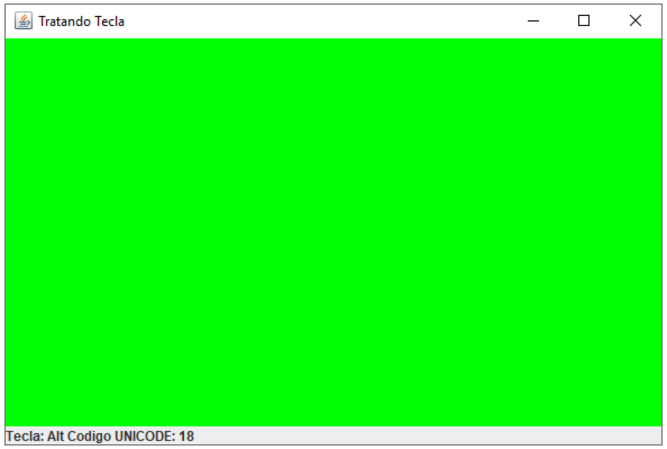
Tratando Evento de Teclas

```
       public void keyPressed(KeyEvent e) {
              this.lblAviso.setText("Tecla: " + KeyEvent.getKeyText(e.getKeyCode()) + " " +
                                                  "Codigo UNICODE: " + e.getKeyCode() + "\n");
              if(e.getKeyCode() == KeyEvent.VK_ENTER) {
                       ((CardLayout)this.panelPrin.getLayout()).next(panelPrin);
              }
              else if(e.getKeyCode() == KeyEvent.VK_ESCAPE) {
                       System.exit(0);
              }
        }

        public void keyReleased(KeyEvent e)
        {   }

        public void keyTyped(KeyEvent e)
        {   }
}

public class JanelaTecla
{
         public static void main(String[] args)
         {
                  TrocaTecla jan = new TrocaTecla();
         }
}
```

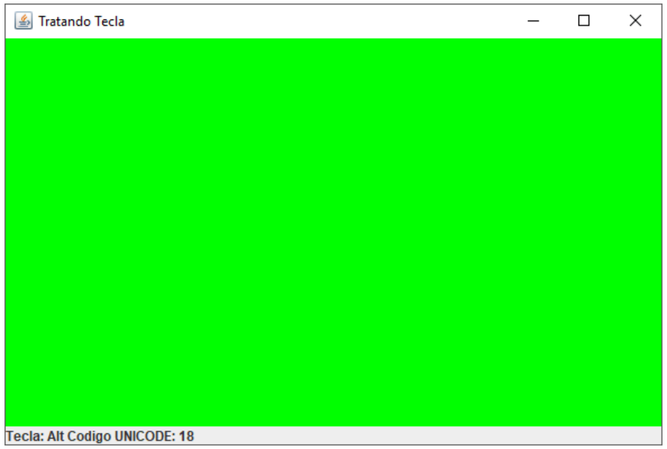
Tratando Evento de Teclas



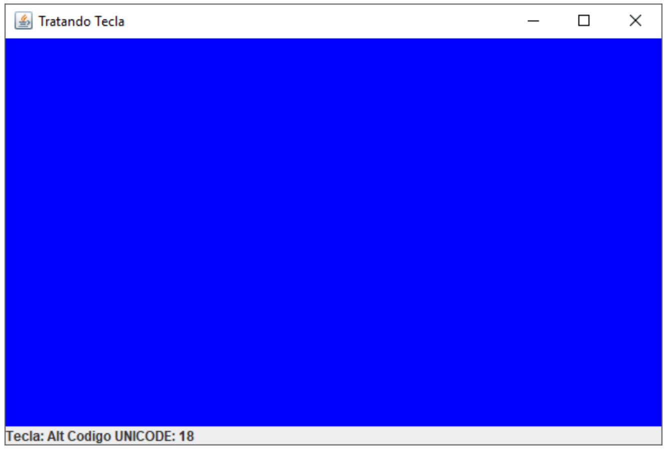
Tecla: Alt Codigo UNICODE: 18



Tecla: Alt Codigo UNICODE: 18



Tecla: Alt Codigo UNICODE: 18



Tecla: Alt Codigo UNICODE: 18