

UNIP
UNIVERSIDADE PAULISTA

Aplicações de Linguagem de Programação Orientada a Objetos

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

1

UNIP
UNIVERSIDADE PAULISTA

Disciplina: Aplicações de Linguagem de Programação Orientada a Objetos - LPOO

Tema: Componentes Visuais – Swing

Professor: Ricardo Drudi

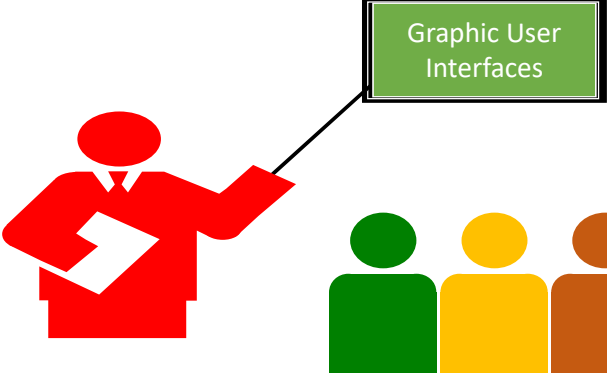
Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

2

POO - Java

UNIP
UNIVERSIDADE PAULISTA



Graphic User Interfaces

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

3

APIs GUI

UNIP
UNIVERSIDADE PAULISTA

- **JDK 1.0**
 - AWT - Abstract Window Toolkit: Write Once Run Anywhere, Applets
 - Componentes de peso pesado (*peers*) que usam recursos do SO
 - Look & feel nativo da plataforma final (incompatibilidades !)
 - Eventos contidos em ondas: *manipulação de eventos limitada*
- **JDK 1.1**
 - AWT com arquitetura *JavaBeans* (contrib. Inprise/Borland), modelo de delegação de eventos, PME – propriedades, métodos e eventos
 - **JFC**: Java Foundation Classes – esforço Sun, Netscape e IBM para criar uma biblioteca gráfica de interface com usuário para o desenvolvimento de aplicações Java interativas (1997 JavaOne)

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

4

JFC

- **Pluggable Look & feel** (configurável)
- **Acessibilidade**: suporte a tecnologias assessoras (Accessibility technologies) tais como leitores de tela e displays em Braille.
- **Suporte a Drag & Drop**: transferência de dados via clipboard, inclusive entre componentes de uma aplicação Java e uma aplicação nativa.
- **Internacionalização**

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

5

APIs GUI

- **Swing GUI Components**:
 - Swing 1.0: Arquitetura MVC, Diagramadores & Listeners do AWT;
 - Swing 1.1: Biblioteca de Componentes: Desktops virtuais (MDI), Objetos Action, Containers aninhados, Bordas compostas, Classes de diálogo padronizadas e customizáveis, Componentes de alto nível como Table, Tree, FileChooser, ColorChooser, Manipulação de texto poderosa, suporte a HTML, operação sem mouse, menus contextuais (popup), Undo: Capacidade genérica de desfazer operações;
- **Java 2D API**: permite ao desenvolvedor incorporar gráficos 2D, texto e imagens em aplicações e em applets. Impressão de alta qualidade, Double-buffering automático.

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

6

Pacote Padrão AWT

- Abstração do sistema nativo
- **Toolkit** gráfico e de interface
- Pacote **awt**
 - elementos de interface
 - diagramadores
 - ferramentas gráficas
- Pacote **awt.event**
 - eventos

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

7

Swing vs. AWT

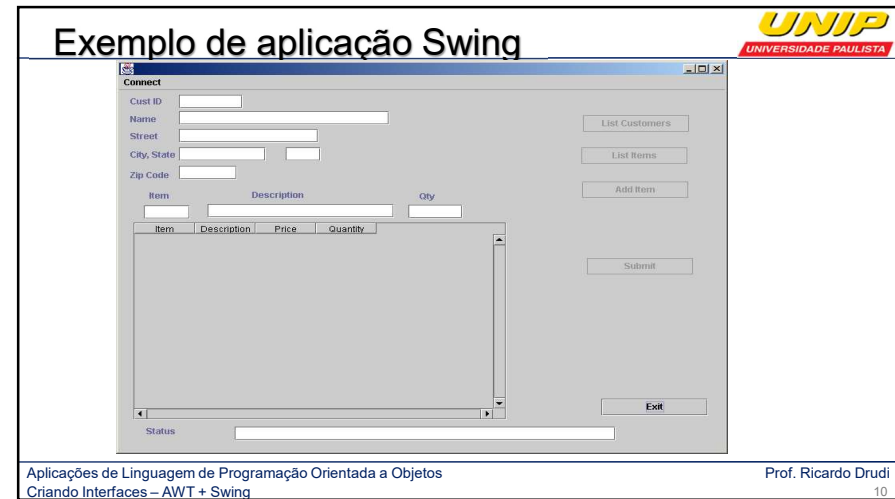
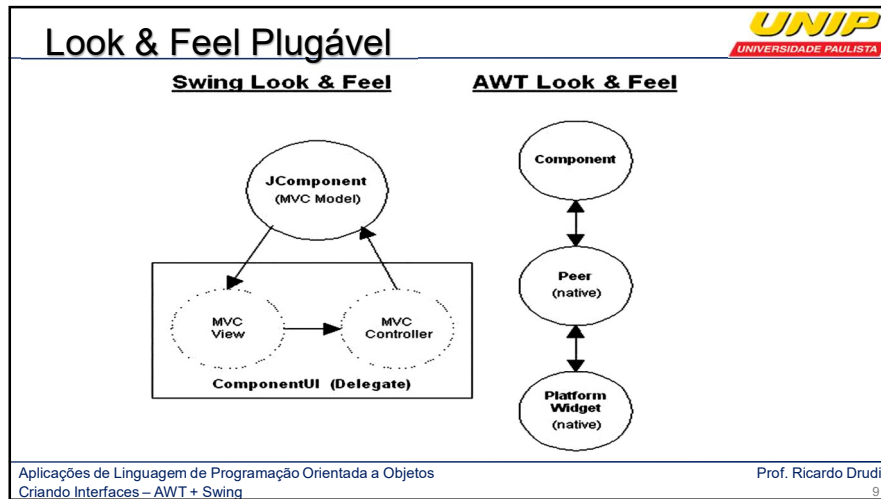
- **Swing** usa componentes leves (lightweight), *non-peer-based GUI toolkit*. Os componentes do **Swing** são implementados sem código nativo
 - maior portabilidade.
 - maior consistência de uso entre plataformas
- Pacotes **javax.swing** e **javax.swing.event** são os mais usados

javax.accessibility	javax.swing.plaf	javax.swing.text.html
javax.swing.text.parser	javax.swing.border	javax.swing.plaf.metal
javax.swing.text.rtf	javax.swing.colorchooser	javax.swing.plaf.multi
javax.swing.tree	javax.swing.event	javax.swing.table
javax.swing.undo	javax.swing.filechooser	javax.swing.text
javax.swing	javax.swing.plaf.basic	

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

8



Construção de uma aplicação Swing

- Uma aplicação Swing contém os seguintes passos.
 1. Importação dos pacotes Swing
 2. Seleção da aparência (“look & feel”)
 3. Definição do **contêiner** de mais alto nível
 4. Definição dos componentes gráficos
 5. Adição dos componentes a um container
 6. Adição de bordas em componentes
 7. Manipulação de eventos

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

Importando os pacotes Swing

- A linha a seguir importa o pacote principal para aplicações **Swing**:


```
import javax.swing.*;
```
- A maioria das aplicações Swing também precisam de dois pacotes **AWT**:


```
import java.awt.*;
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

Selecionando o "look & feel"



- Swing permite que o programador escolha a aparência (**look & feel**) mas apropriada através do método **setLookAndFeel**. Esse método recebe como argumento um nome inteiramente qualificado de uma subclasse de **LookAndFeel**.
- Os **LookAndFeel** disponíveis nos pacotes do Java 8 atualmente são:
 - Metal
 - Nimbus
 - Motif
 - Windows
 - Windows Classic

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
13

Selecionando o "look & feel"

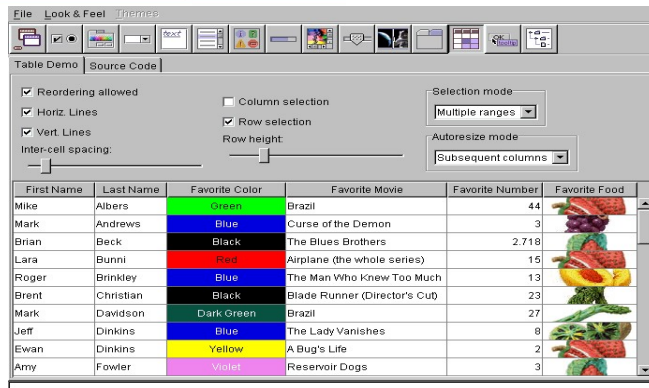


```
private void configuraLookAndFeel() {
    try {
        for (LookAndFeelInfo info : UIManager.getInstalledLookAndFeels()) {
            if ("Windows".equals(info.getName())) {
                UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (UnsupportedLookAndFeelException | ClassNotFoundException |
            InstantiationException | IllegalAccessException e) {
        System.out.println("Erro: " + e.getMessage());
        e.printStackTrace();
    }
}
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

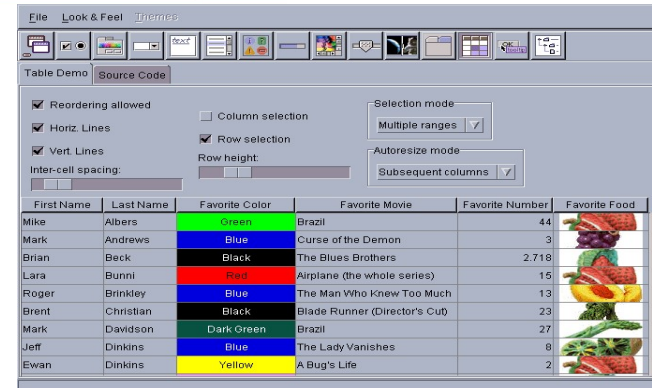
Prof. Ricardo Drudi
14

Windows Look and Feel



Prof. Ricardo Drudi
15

Motif Look and Feel



Prof. Ricardo Drudi
16

Metal Look and Feel

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

Look And Feel Dinâmico

Uma vez ajustado, o LookAndFeel serve toda a aplicação. Entretanto, é possível modificá-lo para um Frame em particular, com a adição das seguintes linhas de código:

```
SwingUtilities.updateComponentTreeUI(frame);
frame.repaint();
```

Caso queira modificar o próprio JFrame que selecionou o LookAndFeel, use a palavra chave `this`.

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

Elementos de Interface AWT

- *Components e Containers*
- **java.awt.Component**
 - base de todos os componentes
 - serializável
 - abstrata
- **java.awt.Container**
 - estende **Component**
 - abstrata

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

Classes do pacote java.awt

- Hierarquia Básica de Classes de java.awt

```

graph LR
    Component[Component] --- Container[Container]
    Component --- Button[Button]
    Container --- Panel[Panel]
    Container --- Window[Window]
    Window --- Dialog[Dialog]
    Window --- Frame[Frame]
    TextComponent[TextComponent] --- TextArea[TextArea]
    TextComponent --- TextField[TextField]
  
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

Pacote java.awt



- Class Component
 - Modela um elemento de interface.
 - Define métodos, a princípio, comuns a qualquer elemento de interface.
- Métodos:


```
public void setForeground(Color c)
public void setEnabled(boolean b)
public Container getParent()
public void addMouseListener(MouseListener l)
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
21

A classe Container



- Class Container
 - Modela um objeto de interface que pode conter outros objetos, um **agrupador**.
Fornece métodos para adicionar e remover componentes e para trabalhar com layout.
 - Um contêiner pode conter outros contêineres (porque todo contêiner é um componente)
- Métodos:


```
public Component add(Component comp)
public void add(Component comp, Object constraint)
public void remove(Component comp)
public boolean isAncestorOf(Component c)
public Component[] getComponents()
public LayoutManager getLayout()
void setLayout( LayoutManager mgr )
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
22

Containers Concretos de java.awt



- Panel
 - Representa um grupo de elementos
 - Deve ser incluído em outro *container*
 - Usado para estruturar a interface
- Frame
 - Estende **java.awt.Window**
 - Representa uma janela
 - Possui título e borda
 - Pode possuir menu
- Dialog

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

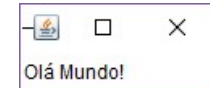
Prof. Ricardo Drudi
23

Exemplo



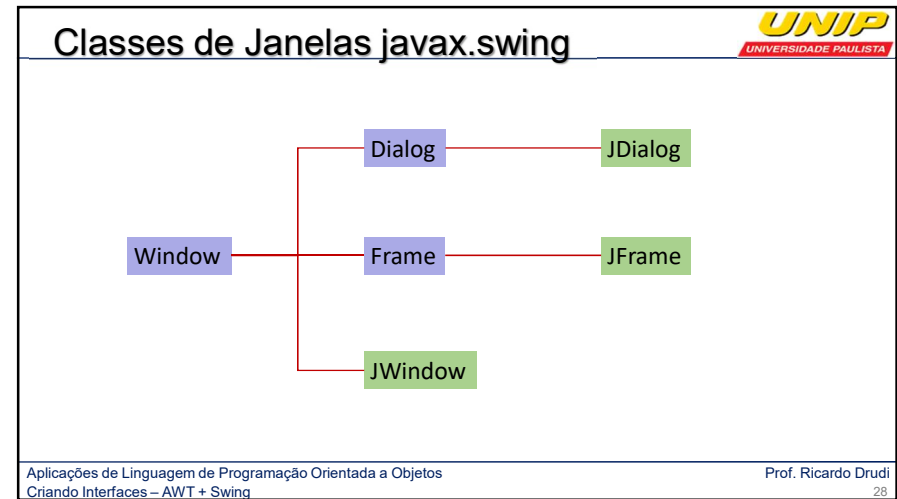
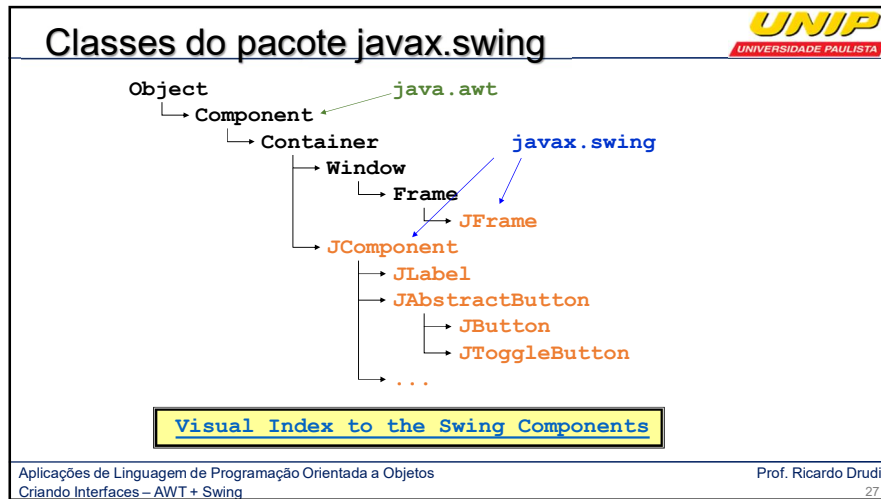
```
import java.awt.*;

public class Mundo {
    public static void main(String[] args) {
        Frame janela = new Frame("Mundo");
        Label mensagem = new Label("Olá Mundo!");
        janela.add(mensagem);
        janela.pack();
        janela.setVisible(true);
    }
}
```



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
24



Classe JComponent

- A classe `JComponent` é a superclasse de todos os componentes Swing.
- Os objetos `JButton`, `JCheckbox`, e `TextField` são todos exemplos de objetos das subclasses de `JComponent`.
- A classe `JComponent` é uma subclasse direta da classe `java.awt.Container` que, por sua vez, é uma subclasse direta de `java.awt.Component`

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

Classe JComponent

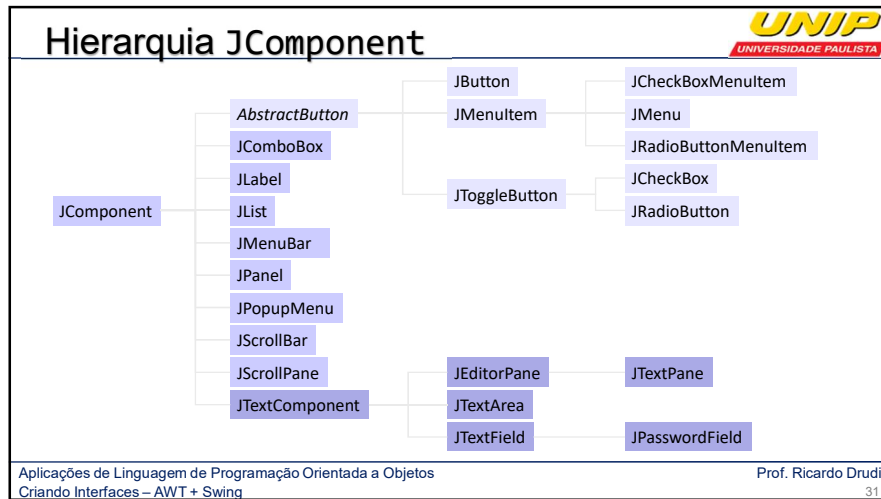
- Superclasse de muitos elementos do Swing, disponibiliza métodos para controlar:
 - Tool tips*
 - Bordas
 - Propriedades
- Métodos de JComponent


```

void setToolTipText(String text)
String getToolTipText()
void setBorder(Border border)
Border getBorder()
final void putClientProperty(Object key, Object val)
final Object getClientProperty(Object key)
      
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi



Pacotes Swing

- javax.*** – extensões padrão (substitui **com.sun.java.***)
- swing.*** – componentes, modelos e interfaces
- border** – estilos de bordas
- colorchooser** – palheta de cores
- event** – eventos e *listeners* específicos do Swing
- filechooser** – seletor de arquivos
- plaf.*** – pacotes de *look & feel* plugável
- table** – componente tabela
- text.*** – *framework* de documentos
- html.*** – suporte para HTML 3.2
- rtf** – suporte para Rich Text Format
- tree** – componente árvore
- undo** – suporte para desfazer operações

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

Container em javax.swing

- Swing possui vários componentes que são contêineres de primeiro nível (top-level container)
 - raiz de uma “*containment hierarchy*”
- Esses contêineres são usados como o “arcabouço” das GUIs: **JApplet**, **JDialog**, **JFrame**, and **JWindow**
 - aplicações tipicamente possuem pelo menos uma hierarquia com um **JFrame** como raiz (janela principal)
 - applets Swing contém uma hierarquia com **JApplet** como raiz

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

Classe JFrame

- Modela uma janela do sistema nativo
- Equivalente, no Swing, à classe **Frame** do AWT
- Métodos de JFrame


```

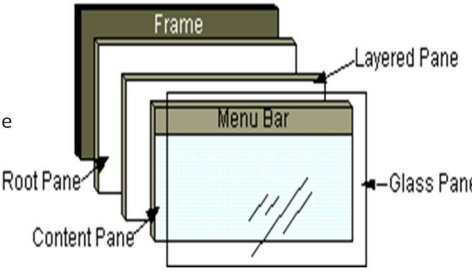
JRootPane getRootPane()
Container getContentPane()
void setJMenuBar(JMenuBar menubar)
JMenuBar getJMenuBar()
void setDefaultCloseOperation(int op)
      
```

 - EXIT_ON_CLOSE
 - HIDE_ON_CLOSE
 - DISPOSE_ON_CLOSE
 - DO_NOTHING_ON_CLOSE

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

Camadas do JFrame



- **RootPane**
 - gerencia as demais camadas
 - botão "default"
- **LayeredPane**
 - Contém a *menu bar* e o *ContentPane*
 - Pode conter subcamadas (Z order)
- **ContentPane**
 - contém os componentes visíveis
- **GlassPane**
 - invisível por *default*
 - interceptação de eventos/pintura sobre uma região

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

37

Exemplo de JFrame

```
import javax.swing.*;


public class HelloWorldSwing {
    /**
     * Create the GUI and show it. For thread safety, this
     * should be invoked from the event-dispatching thread.
     */
    private static void createAndShowGUI() {
        //Make sure we have nice window decorations.
        JFrame.setDefaultLookAndFeelDecorated(true);
        //Create and set up the window.
        JFrame frame = new JFrame("HelloWorldSwing");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        //Add the ubiquitous "Hello World" label
        JLabel label = new JLabel("Hello World");
    }
}
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

38

Exemplo de JFrame



```
frame.getContentPane().add(label);
//Display the window.
frame.pack(); frame.setVisible(true);
}

public static void main(String[] args) {
    //Schedule a job for the event-dispatching thread
    //creating and showing this application's GUI.
    SwingUtilities.invokeLater(new Runnable(){
        public void run() {
            createAndShowGUI();
        }
    });
}
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

39

Classe JPanel

- Modela um *container* sem decoração, normalmente utilizado para estruturar a interface. Equivalente, no Swing, à classe **Panel** do AWT
- Métodos de JPanel


```
JPanel()
JPanel(LayoutManager mgr)
void setLayout(LayoutManager mgr)
Component add(Component comp)
void add(Component c, Object constraints)
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

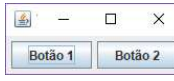
Prof. Ricardo Drudi

41

Exemplo de JPanel1

```
import javax.swing.*;

public class ExemploJPanel {
    public static void main(String[] args) {
        JFrame f = new JFrame("Teste");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JButton b1 = new JButton("Botão 1");
        JButton b2 = new JButton("Botão 2");
        JPanel p = new JPanel();
        p.add(b1);
        p.add(b2);
        f.getContentPane().add(p);
        f.pack();
        f.setVisible(true);
    }
}
```

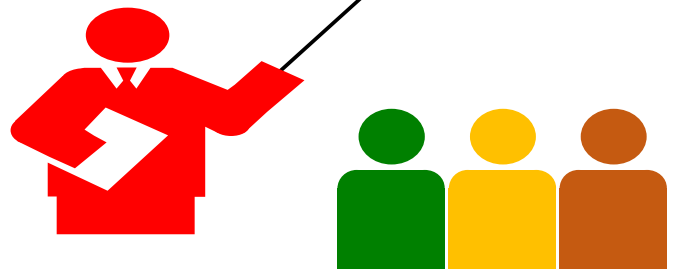


Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

42

POO - Java



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

43

Gerenciadores de Layout

- Ambientes como o Visual Basic e Delphi usam coordenadas (x,y) para definir a localização de componentes na interface gráfica.
- Swing usa **Gerenciadores de Layout** (*Layout Managers*) para controlar os componentes serão posicionados.
- Sem um **gerenciador de layout**, os componentes podem ser movidos para posições inesperadas quando a tela é redimensionada.
- Existem diferentes estilos de arrumação
 - como fluxo de texto
 - orientada pelas bordas
 - em forma de grade, e outros...

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

44

Gerenciadores de Layout

O Java conta com vários gerenciadores de layout de tela. Embora seja mais tentador utilizar o **AbsoluteLayout**, por permitir que o desenvolvedor posicione os componentes exatamente onde deseja, deve ser considerada a perda de interoperabilidade entre diferentes ambientes de execução. Computadores e dispositivos têm sistemas operacionais, telas, resoluções e interfaces distintos, e um ótimo layout no ambiente de desenvolvimento pode se tornar inutilizável no ambiente de produção.

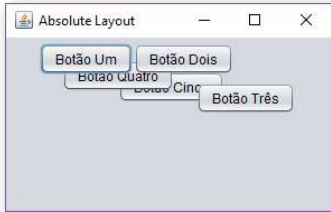
Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

45

Absolute Layout

Posiciona os objetos livremente na tela, em coordenadas determinadas pelo desenvolvedor. Permite que os objetos sejam sobrepostos, embora ainda seja possível algum controle sobre os componentes.



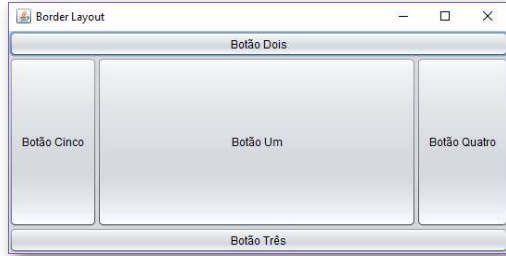
Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

46

Border Layout

Composto de 5 áreas conforme abaixo, sendo que a **área central** toma o maior espaço possível enquanto as demais áreas se expandem o necessário para preencher os espaços restantes.




Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

47

Box Layout

Empilha seus componentes em linha ou em coluna de acordo com a opção selecionada.




Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

48

Card Layout

o **CardLayout** permite que diferentes grupos de componentes sejam exibidos no mesmo contêiner. É útil quando se quer modificar a tela de interação com o usuário dependendo de configurações selecionadas anteriormente.



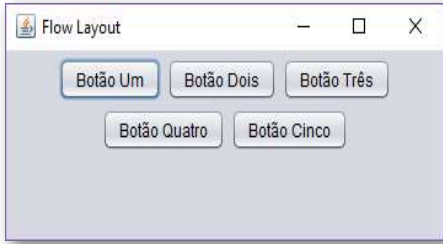
Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

49

Flow Layout

Dispõe os componentes da esquerda para a direita e de cima para baixo conforme cabem no contêiner. Segue a ideia do padrão de texto corrido.



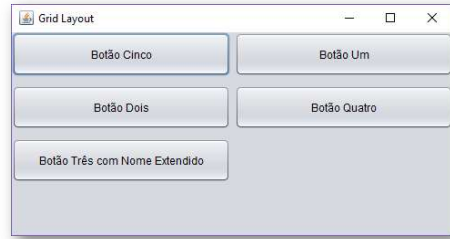
Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

50

Grid Layout

Divide o contêiner em linhas e colunas de tamanho fixo. O preenchimento das células obedecem ao padrão do **FlowLayout**.



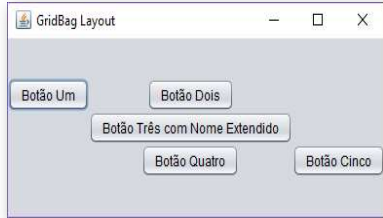
Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

51

GridBag Layout

Permite que o contêiner seja dividido em um padrão de tabela, com linhas e colunas de tamanho fixo. Nem todas as células da tabela têm que estar ocupadas.




Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

52

Group Layout

Separa integralmente a composição dos elementos visuais nos eixos horizontal e vertical. Os espaços entre os componentes também são construídos com componentes de espaçamento (gaps).



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

53

Outros Gerenciadores

Além dos gerenciadores de layouts padrões no Java, existem outros gerenciadores que não fazem parte do pacote de classes padrão. Alguns desses gerenciadores alternativos de layout são:

- JGoodies Form layout
- TableLayout
- MigLayout
- PageLayout
- RiverLayout
- SGLayout
- GroupLayout
- MultisplitPane
- UIHierarchy

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

POO - Java

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

Exemplos Swing

Uma lista completa de exemplos de uso dos componentes Swing pode ser encontrada no site da Oracle, no link abaixo.

<https://docs.oracle.com/javase/tutorial/uiswing/examples/components/index.html#TextSamplerDemo>

Example	Zip File (contains all files necessary for the example plus NetBeans IDE project metadata)
BorderDemo [Launch]	Border Demo Project
ButtonDemo [Launch]	Button Demo Project
ButtonHtmlDemo [Launch]	Button Html Demo Project
CheckBoxDemo [Launch]	Check Box Demo Project
ColorChooserDemo [Launch]	Color Chooser Demo Project
ColorChooserDemo2 [Launch]	Color Chooser 2 Demo Project
ComboBoxDemo [Launch]	Combo Box Demo Project

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

Classe JLabel

- Essa classe modela um texto e/ou imagem não editável, isto é, sem interação com o usuário. É o equivalente, no Swing, ao **Label** do AWT, só que com mais recursos
- Pode-se controlar tanto o alinhamento horizontal como o vertical, e o **JLabel** pode passar o foco para outro elemento
- Pode também manipular conteúdo HTML
 - Se o texto possuir "<html>...</html>", o conteúdo é apresentado como HTML.
 - As fontes são ignoradas se HTML é usado. Nesse caso, todo o controle de fontes deve ser realizado através de tags HTML.
 - Deve ser usado <P>, e não
, para forçar uma quebra de linha. O suporte a novas tags HTML é progressivo.

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

Métodos de JLabel

```
void setText(String text)
void setIcon(Icon icon)
void setIconTextGap(int gap)
void setHorizontalAlignment(int a)
void setVerticalAlignment(int a)
void setLabelFor(Component c)
void setDisplayedMnemonic(int key)
void setDisplayedMnemonic(char aChar)
```

Exemplo de JLabel

```
JFrame f = new JFrame("Teste");
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
JLabel l = new JLabel("Isso é um JLabel");
l.setIcon(new ImageIcon("javaLogo.gif"));
Container cp = f.getContentPane();
cp.add(l);
f.pack();
f.show();
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

59

JLabel com HTML

```
String labelText = "<html><FONT COLOR=BLACK>Texto em</FONT> " +
    "<html><FONT COLOR=WHITE>Branco</FONT> e " +
    "<FONT COLOR=GRAY>Cinza</FONT></html>";

coloredLabel = new JLabel(labelText, JLabel.CENTER);

labelText = "<html><B>Bold</B> e <I>Itálico</I> Text</html>";

JLabel boldLabel = new JLabel(labelText, JLabel.CENTER);

labelText = "<html><h2><span style='color: #ffffff; background-color: #333399;'><em><strong>Java&nbsp;</strong></em></span>&nbsp;<em>: a melhor linguagem porque:</h2> <ul> <li>&eacute; orientada a objetos de <span style='color: #ff0000;'><em>verdade</em></span>;</li> " + ...;

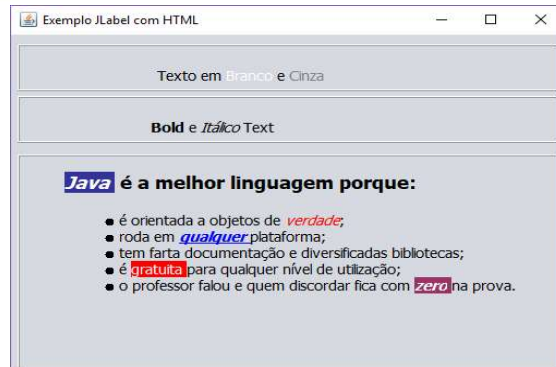
JLabel javaRocksLabel = new JLabel(labelText, JLabel.CENTER);
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

60 |

JLabel



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

61

Classe JButton

- Modela um *push-button*
 - Sendo uma sub-classe de **AbstractButton**, herda os métodos **getLabel** e **setLabel** que permitem consultar e alterar seu texto. Permite que o botão seja cadastrado como *default button* do **RootPane**

- Métodos de JButton

```

JButton(String text)
JButton(String text, Icon icon)

void setLabel(String label)
String getLabel()

```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

62

Métodos de JButton



- Instanciação (exemplos)
 - JButton btnOk = new javax.swing.JButton();
 - JButton proximo = new JButton(new ImageIcon("direita.gif"));
 - JButton proximo = new JButton("Próximo", rightArrow);
 - JButton button = new JButton("Eu sou um botão!");
 - button.setMnemonic(KeyEvent.VK_I); // tecla "I" com sendo a tecla de atalho
- Principais métodos
 - void setText(String)
 - void setToolTipText(String)
 - void addActionListener(ActionListener)
 - void setBounds(int x, int y, int width, int height)
 - void setVisible(boolean)
 - void setEnabled(boolean)

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
63

Exemplo de JButton



```
JFrame f = new JFrame("Teste");
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
JButton b1 = new JButton("Botão 1");
JButton b2 = new JButton("Botão 2");
Container cp = f.getContentPane();
cp.setLayout(new GridLayout(1,0));
cp.add(b1);
cp.add(b2);
f.pack();
f.show();
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
64

Classe JRadioButton



- Modela um botão de escolha que pode ser marcado e desmarcado
- Objetos do tipo **JRadioButton** são organizados em grupos
- Apenas um único botão de um grupo pode estar marcado em um dado momento
- Métodos de JRadioButton
 - JRadioButton(String label)
 - JRadioButton(String label, boolean state)
 - boolean isSelected()
 - void setSelected(boolean state)

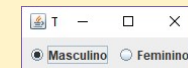
Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
65

Exemplo de JRadioButton



```
JFrame f = new JFrame("Teste");
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
JRadioButton bm = new JRadioButton("Masculino",true);
JRadioButton bf = new JRadioButton("Feminino");
ButtonGroup bg = new ButtonGroup();
bg.add(bm);
bg.add(bf);
Container cp = f.getContentPane();
cp.setLayout(new FlowLayout());
cp.add(bm);
cp.add(bf);
f.pack();
f.setVisible(true);
```



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

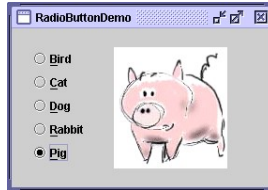
Prof. Ricardo Drudi
66

Classe ButtonGroup

- Cria um “escopo” de exclusão para um grupo de botões
- Basta criar um **ButtonGroup** e adicionar a ele os **JRadioButtons** que compõem o grupo

- Métodos de ButtonGroup

```
void add(AbstractButton b)
ButtonModel getSelection()
boolean isSelected(ButtonModel m)
void setSelected(ButtonModel m, boolean state)
```



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
67

Classe JCheckBox

- Modela um botão de escolha que pode ser marcado e desmarcado
- Métodos **JCheckBox**

```
public JCheckBox(String label)
public JCheckBox(String label, boolean state)
public boolean isSelected()
public void setSelected(boolean state)
```



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
68

Classe JComboBox

- Um componente que combina um botão, um campo (editável ou não) e uma lista *drop-down*.

- Principais métodos

```
int getSelectedIndex()
void setSelectedIndex(int)
void removeAllItems()
void addItem(String)
String getSelectedItem()
void setSelectedItem(String)
void addActionListener (ActionListener)
```



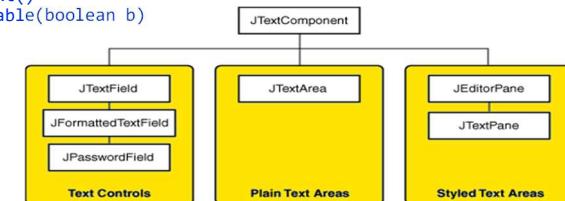
Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
69

Classe JTextComponent

- Classe abstrata que modela o que há de comum entre diferentes elementos de edição de texto
- Métodos **JTextComponent**

```
public void setText(String t)
public String getText()
public void setEditable(boolean b)
```



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
70

Classe JTextField

- Cria campo de edição de texto de uma linha


```
JTextField()
JTextField(String text)
JTextField(int columns)
JTextField(String text, int columns)
```
- Principais métodos


```
setBounds(int x, int y, int width, int height)
setEnabled(boolean)
setText(String)
String getText()
requestFocus()
```

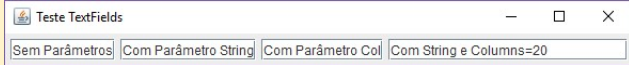
Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

71

Exemplo de JTextField

```
JFrame f = new JFrame("Teste TextFields");
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
JTextField tf1 = new JTextField();
tf1.setText("Sem Parâmetros");
JTextField tf2 = new JTextField("Com Parâmetro String");
JTextField tf3 = new JTextField(10);
tf3.setText("Com Parâmetro Columns=10");
JTextField tf4 = new JTextField("Com String e Columns=20", 20);
Container cp = f.getContentPane();
cp.setLayout(new FlowLayout());
cp.add(tf1);
cp.add(tf2);
cp.add(tf3);
cp.add(tf4);
f.pack();
f.setVisible(true);
```



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

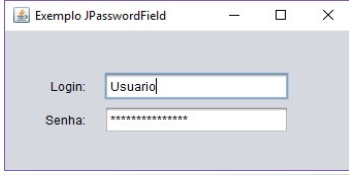
Prof. Ricardo Drudi

72

Classe JPasswordField

- Estende **JTextField**
- Caracteres digitados não são exibidos
- Principais métodos


```
JPasswordField()
JPasswordField(int columns)
JPasswordField(String text, int columns)
char[] getPassword()
void setEchoChar(char c)
```



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

73

JFormattedTextField - Máscara de Entrada

- Estende **JTextField**
- Permite ao desenvolvedor especificar o conjunto de caracteres aceitos como entrada do campo.
- A classe **MaskFormatter** é usada para formatar e editar *Strings*. O comportamento da classe **MaskFormatter** é controlado por um tipo de máscara de *String* que especifica os caracteres válidos que podem ser digitados naquele campo.

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

74

JFormattedTextField

- Exemplo: CEP


```
MaskFormatter mask = null;
try {
    mask = new MaskFormatter("#####-####");
    mask.setPlaceholderCharacter('_');
} catch (ParseException ex) {
    .....
}
JFormattedTextField tfCep = new JFormattedTextField(mask); // na criação
tfCep.setFormatterFactory(new DefaultFormatterFactory(mask,mask)); // depois de criado
```
- Telefone


```
mask = new MaskFormatter("(##)####-####");
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

75

JFormattedTextField

Caractere	Descrição
#	Dígitos de 0 a 9. Usa Character.isDigit().
,	Caractere de escape, usado para incluir comandos escape e/ou caracteres especiais de escape.
U	Letras de A a Z. Usa Character.isLetter(). Converte as letras em maiúsculas.
L	Letras de a a z. Usa Character.isLetter(). Converte as letras em minúsculas.
A	Letra ou dígito. Usa Character.isDigit() Character.isLetter().
?	Qualquer caractere gerado pelo teclado (caracteres imprimíveis).
*	Qualquer código gerado (incluindo <enter>, <ctrl>, etc.)
H	Notação hexadecimal (0-9, A-F ou a-f).

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

76

Classe JTextArea

- Cria um campo de edição de texto com múltiplas linhas


```
JTextArea(int rows, int columns)
JTextArea(String text, int rows, int columns)
```
- Principais métodos


```
void append(String t)
void insert(String t, int pos)
void setLineWrap(boolean wrap)
void setWrapStyleWord(boolean word)
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

77

Elementos de Edição de Texto

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

79

Bordas

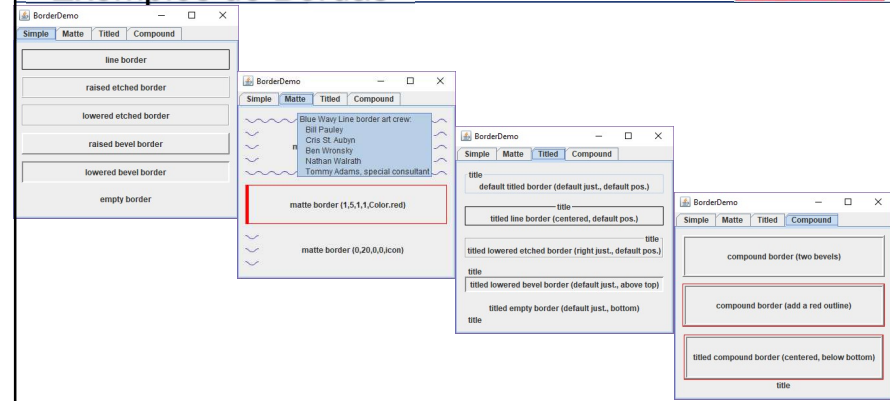


- O Swing permite a criação de bordas (molduras) envolvendo componentes
- O método `setBorder` de `JComponent` permite “emoldurar” um componente com uma borda
- Bordas são especialmente úteis para distinguir diferentes áreas de uma tela de entrada ou exibição de dados.

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
80

Exemplos de Bordas



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
81

Painéis, Bordas e Botões



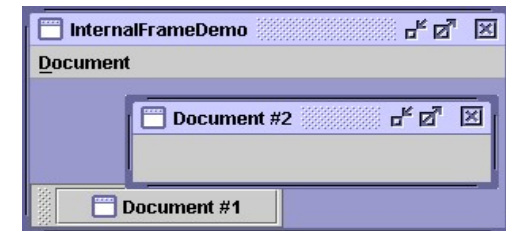
Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
83

JDesktopPane



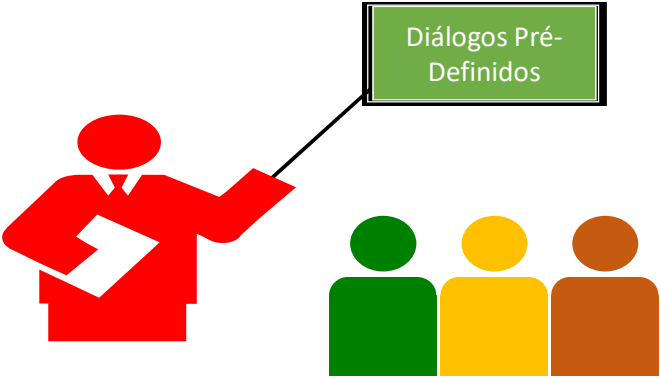
- Um objeto `JDesktopPane` é um contêiner para construir uma aplicação de múltiplos documento (MDI). Serve como a janela mãe (*desktop*) das demais janelas.
- Um objeto `JDesktopPane` pode conter diversos objetos `JInternalFrame`.
- Principais métodos
 - `add(JInternalFrame)`
 - `cascadeFrames()`



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
95

POO - Java



Diálogos Pré-Definidos

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

110

Diálogos Pré-definidos

- O Swing oferece um conjunto de diálogos simples pré-definidos para uso em interações breves com o usuário
 - mensagens de erro, de alerta
 - obtenção de uma confirmação
 - entrada de um único campo de texto
- Esses diálogos são *modais*
- A classe que disponibiliza as telas de diálogos é a `JOptionPane`

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

111

JOptionPane

- Classe que permite exibir diferentes tipos de caixas de diálogo.
- Cinco métodos estáticos principais
 - `JOptionPane.showMessageDialog`
 - Icon, message, OK button
 - `JOptionPane.showConfirmDialog`
 - Icon, message, and buttons: OK, OK/Cancel, Yes/No, or Yes/No/Cancel
 - `JOptionPane.showInputDialog`
 - Icon, message, textfield ou combobox, buttons
 - `JOptionPane.showOptionDialog`
 - Icon, message, array de buttons ou de outros componentes

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

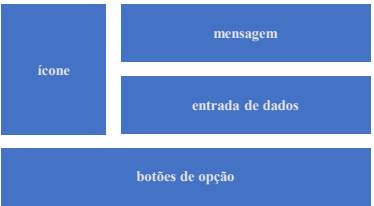
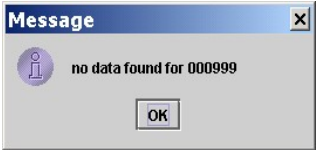
Prof. Ricardo Drudi

112

Classe JOptionPane

- Métodos estáticos para a criação desses diálogos simples
- Estrutura básica:

Exemplo:


Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing


Prof. Ricardo Drudi


113


MessageDialog


- Exibe uma mensagem e aguarda OK do usuário


INFORMATION 

QUESTION 

WARNING 

ERROR 

PLAIN 





Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing


Prof. Ricardo Drudi


115

JOptionPane: Message Dialogs









Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

116

Método showMessageDialog

```

void showMessageDialog(Component parentComponent
    Object message);

void showMessageDialog(Component parentComponent,
    Object message,
    String title,
    int messageType);

void showMessageDialog(Component parentComponent,
    Object message,
    String title,
    int messageType,
    Icon icon);
  
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

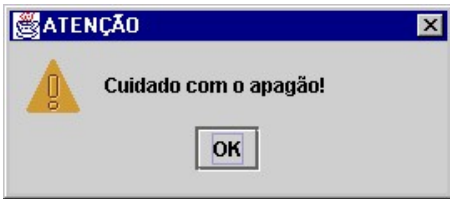
Prof. Ricardo Drudi

117

Exemplo de MessageDialog

```

JOptionPane.showMessageDialog(null,
    "Cuidado com o apagão!",
    "ATENÇÃO",
    JOptionPane.WARNING_MESSAGE);
  
```



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

118

ConfirmDialog

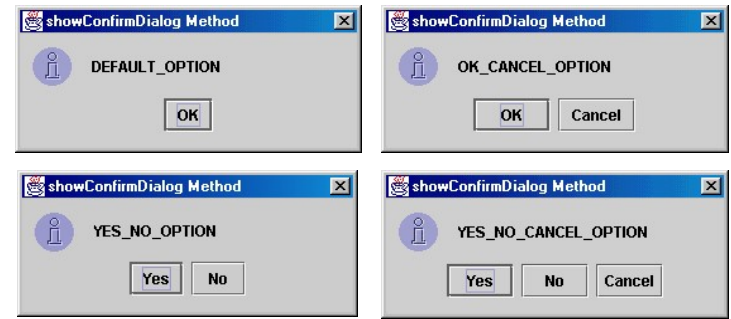
- Exibe uma mensagem e obtém uma confirmação (YES/NO,OK/CANCEL)
- Conjuntos de botões de opção (optionType):
 - JOptionPane.DEFAULT_OPTION
 - JOptionPane.YES_NO_OPTION
 - JOptionPane.YES_NO_CANCEL_OPTION
 - JOptionPane.OK_CANCEL_OPTION

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

119

JOptionPane: Confirmation Dialogs



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

120

Método showConfirmDialog

```
int showConfirmDialog(Component parentComponent,
                     Object message);

int showConfirmDialog(Component parentComponent,
                     Object message,
                     String title,
                     int optionType);

int showConfirmDialog(Component parentComponent,
                     Object message,
                     String title,
                     int optionType,
                     int messageType,
                     Icon icon);
```

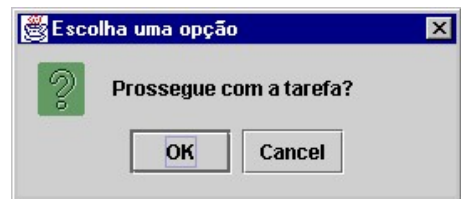
Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi

121

Exemplo de ConfirmDialog

```
int resp = JOptionPane.showConfirmDialog(janela,
                                         "Prossegue com a tarefa?",
                                         "Escolha uma opção",
                                         JOptionPane.OK_CANCEL_OPTION);
```



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

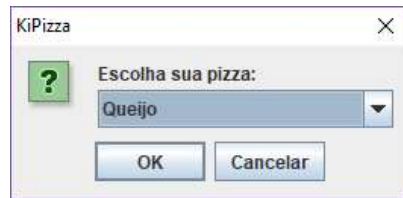
Prof. Ricardo Drudi

122

InputDialog



- Exibe uma mensagem e obtém um valor de entrada do usuário:
 - campo de texto editável
 - combobox



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
123

Método showInputDialog



```
String showInputDialog(Component parentComponent,
    Object message);

String showInputDialog(Component parentComponent,
    Object message,
    String title,
    int messageType);

Object showInputDialog(Component parentComponent,
    Object message,
    String title,
    int messageType,
    Icon icon,
    Object[] selectionValues,
    Object defaultSelection);
```

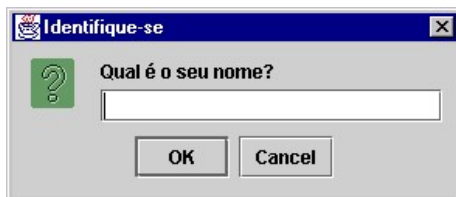
Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
124

Exemplo de InputDialog



```
String nome = JOptionPane.showInputDialog(janela,
    "Qual é o seu nome?",
    "Identifique-se",
    JOptionPane.QUESTION_MESSAGE);
```



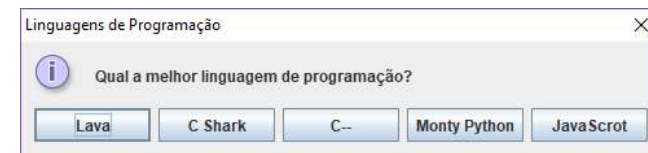
Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
125

OptionDialog



- Exibe uma mensagem (ou objeto) e obtém uma opção escolhida pelo usuário
- O número de botões e seus textos são configuráveis
- A opção *default* é configurável



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
126

Método showDialog


```
int showDialog(Component parentComponent,  
               Object message,  
               String title,  
               int optionType  
               int messageType,  
               Icon icon,  
               Object[] options,  
               Object initialValue);
```

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
127

Exemplo de OptionDialog

```
Object[] opções = {"Sim", "Não", "Mais Tarde", "Amanhã", "Sei lá!"};  
int resp = JOptionPane.showMessageDialog(janela,  
    "Prossegue com a tarefa?",  
    "Escolha uma opção",  
    JOptionPane.DEFAULT_OPTION,  
    JOptionPane.QUESTION_MESSAGE,  
    null,  
    opções,  
    opções[0]);
```



Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
128

Muito obrigado a todos.

Copyright © 2018 Prof. Ricardo Drudi

Todos direitos reservados. A reprodução ou divulgação total ou parcial deste documento é expressamente proibida sem o consentimento formal, por escrito, do professor Ricardo Drudi.

Aplicações de Linguagem de Programação Orientada a Objetos
Criando Interfaces – AWT + Swing

Prof. Ricardo Drudi
137