

INF1636 – PROGRAMAÇÃO ORIENTADA A OBJETOS

Departamento de Informática – PUC-Rio

Ivan Mathias Filho

ivan@inf.puc-rio.br

Programa – Capítulo 11

- Outros Painéis Intermediários
- Seleção do *Look & Feel*
- Gerenciadores de Layout
- JComponent
- JLabel
- JButton
- JRadioButton
- JCheckBox
- JList

Programa – Capítulo 11

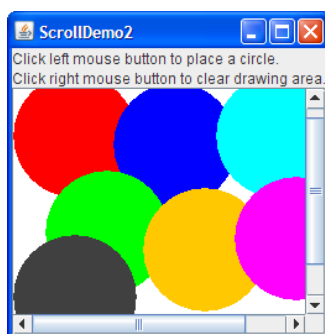


- **Outros Painéis Intermediários**
- Seleção do *Look & Feel*
- Gerenciadores de Layout
- JComponent
- JLabel
- JButton
- JRadioButton
- JCheckBox
- JList

Outros Painéis Intermediários (1)

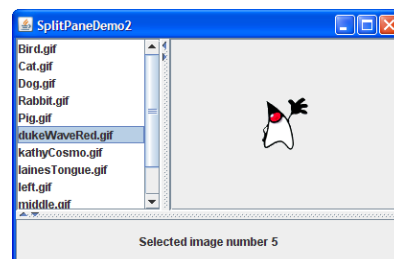


- Além do container `JPanel`, o Swing possui outros *containers* intermediários. Entre eles estão:

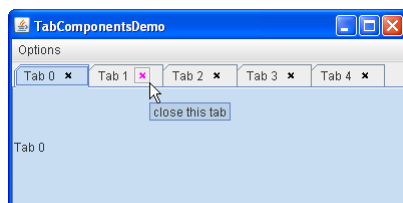


JScrollPane

JSplitPane

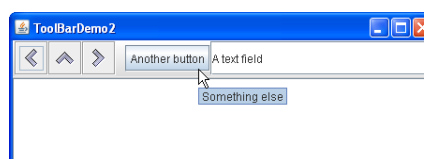


Outros Painéis Intermediários (2)



JTabbedPane

JToolBar



© LES/PUC-Rio

Programa – Capítulo 11



- Outros Painéis Intermediários
- **Seleção do Look & Feel**
- Gerenciadores de Layout
- JComponent
- JLabel
- JButton
- JRadioButton
- JCheckBox
- JList

© LES/PUC-Rio

Seleção do *Look & Feel* (1)



- O Swing fornece várias alternativas de padrão gráfico (*Look & Feel*) de interfaces com o usuário;
- A definição de um padrão é feita por meio da classe `UIManager`;
- O código abaixo especifica o *look & feel* default, chamado de *Java Look & Feel* (independente de plataforma).

```
try
{
    UIManager.setLookAndFeel
        (UIManager.getCrossPlatformLookAndFeelClassName());
}
catch (ClassNotFoundException e)
{ }
catch (UnsupportedLookAndFeelException e)
{ }
catch (Exception e)
{ }
```

© LES/PUC-Rio

Seleção do *Look & Feel* (2)



- Pode-se também optar pelo *look & feel* nativo da plataforma utilizada pelo usuário:

```
try
{
    UIManager.setLookAndFeel
        (UIManager.getSystemLookAndFeelClassName());
}
catch (ClassNotFoundException e)
{ }
catch (UnsupportedLookAndFeelException e)
{ }
catch (Exception e)
{ }
```

© LES/PUC-Rio

Seleção do *Look & Feel* (3)

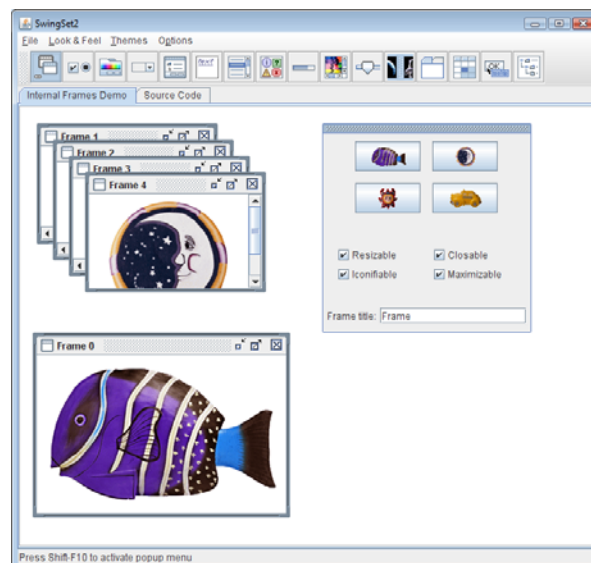


- Por último, pode-se passar como parâmetro o nome da classe que implementa um determinado *look & feel*;
- No Windows estão disponíveis os seguintes *look & feel*:
 - `javax.swing.plaf.metal.MetalLookAndFeel`
 - `com.sun.java.swing.plaf.motif.MotifLookAndFeel`
 - `com.sun.java.swing.plaf.windows.WindowsLookAndFeel`

```
try
{
    UIManager.setLookAndFeel
        ("com.sun.java.swing.plaf.motif.MotifLookAndFeel");
}
```

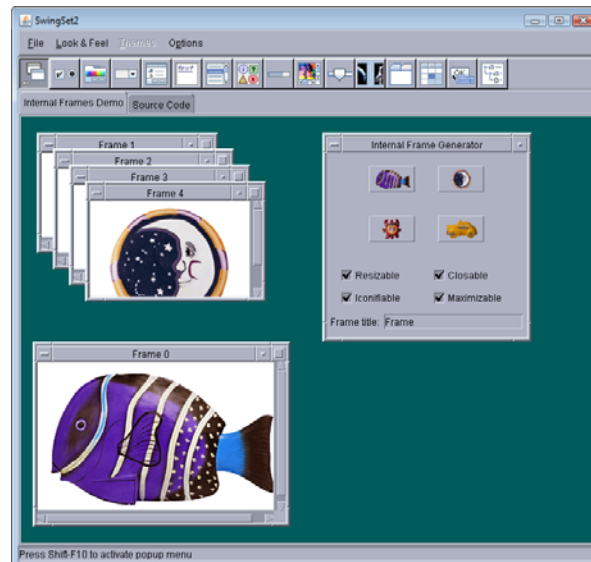
© LES/PUC-Rio

Java *Look & Feel*



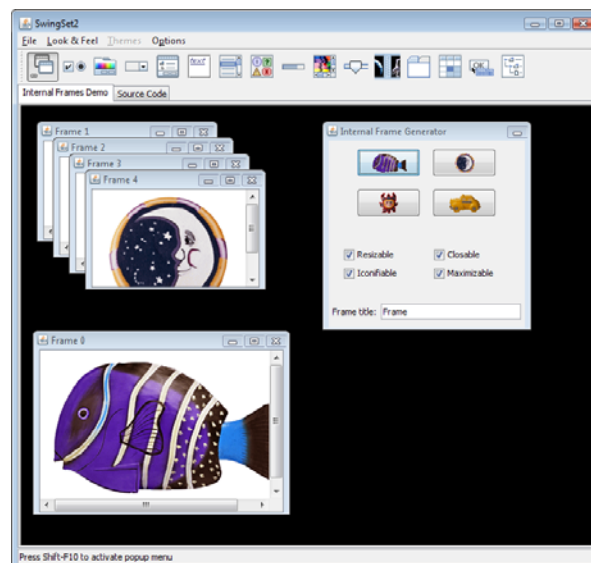
© LES/PUC-Rio

Motif *Look & Feel*



© LES/PUC-Rio

Windows *Look & Feel*



© LES/PUC-Rio

Programa – Capítulo 11



- Outros Painéis Intermediários
- Seleção do *Look & Feel*
- **Gerenciadores de Layout**
- JComponent
- JLabel
- JButton
- JRadioButton
- JCheckBox
- JList

Gerenciadores de layout



- Ferramentas como o Visual Basic, C# e Delphi usam coordenadas (x,y) para definir a localização de componentes na interface gráfica;
- O Swing usa Gerenciadores de Layout (*Layout Managers*) para controlar o posicionamento dos componentes;
- Sem um gerenciador de layout, os componentes podem ser movidos para posições inesperadas quando a tela é redimensionada.

Tipos de gerenciadores de layout (1)



- **BorderLayout**
 - Divide o contêiner em 5 seções: *North, South, East, West* e *Center*.
- **BoxLayout**
 - Coloca os componentes em uma única linha ou coluna.
- **FlowLayout**
 - Componentes posicionados da esquerda para a direita e de cima para baixo;
 - Muda para uma nova linha quando necessário.

Tipos de gerenciadores de layout (2)



- **GridLayout**
 - Posiciona componentes em uma grade de linhas e colunas;
 - Força os componentes a terem o mesmo tamanho.
- **NullLayout**
 - O programador é responsável pelo posicionamento de cada componente.

BorderLayout



- Composto de cinco áreas;
- A área central toma a maior parte do espaço disponível;
- As demais áreas se expandem o necessário para preencher os espaços restantes.

© LES/PUC-Rio

BorderLayout – Exemplo (1)

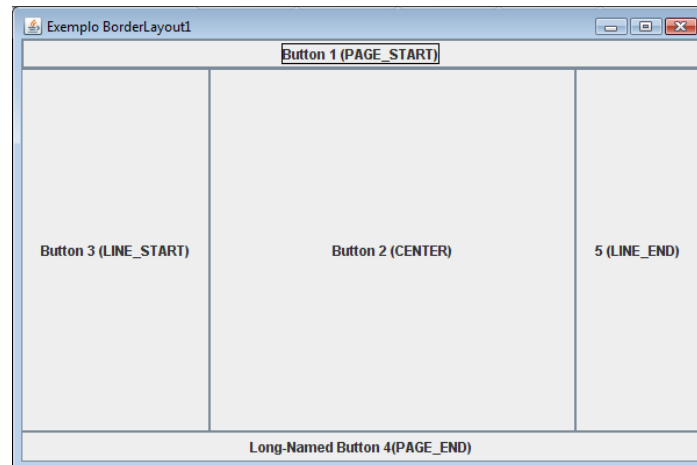


```
import javax.swing.*;
import java.awt.*;

public class ExemploBorderLayout extends JFrame {
    public ExemploBorderLayout(String s) {
        super(s);
        JButton b;
        JPanel p = new JPanel();
        getContentPane().add(p);
        p.setLayout(new BorderLayout());
        b=new JButton("Button 1 (PAGE_START)");
        p.add(b, BorderLayout.PAGE_START);
        b=new JButton("Button 2 (CENTER)");
        p.add(b, BorderLayout.CENTER);
        b=new JButton("Button 3 (LINE_START)");
        p.add(b, BorderLayout.LINE_START);
        b=new JButton("Long-Named Button 4 (PAGE_END)");
        p.add(b, BorderLayout.PAGE_END);
        b=new JButton("5 (LINE_END)");
        p.add(b, BorderLayout.LINE_END);
        setSize(600,400);
        setVisible(true);
    }
}
```

© LES/PUC-Rio

BorderLayout – Exemplo (2)



© LES/PUC-Rio

BoxLayout – Exemplo (1)

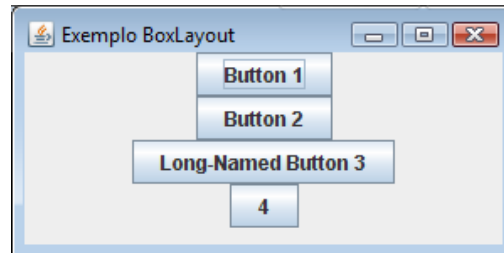


```
import javax.swing.*;
import java.awt.*;

public class ExemploBoxLayout extends JFrame {
    public ExemploBoxLayout(String s) {
        super(s);
        JButton b;
        JPanel p = new JPanel();
        getContentPane().add(p);
        p.setLayout(new BoxLayout(p, BoxLayout.Y_AXIS));
        b = new JButton("Button 1");
        b.setAlignmentX(Component.CENTER_ALIGNMENT);
        p.add(b);
        b = new JButton("Button 2");
        b.setAlignmentX(Component.CENTER_ALIGNMENT);
        p.add(b);
        b = new JButton("Long-Named Button 3");
        b.setAlignmentX(Component.CENTER_ALIGNMENT);
        p.add(b);
        b = new JButton("4");
        b.setAlignmentX(Component.CENTER_ALIGNMENT);
        p.add(b);
        setSize(300, 150);
        setVisible(true);
    }
}
```

© LES/PUC-Rio

BoxLayout – Exemplo (2)



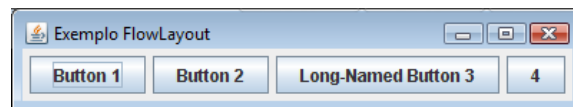
FlowLayout – Exemplo (1)



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

public class ExemploFlowLayout extends JFrame {
    public ExemploFlowLayout(String s) {
        super(s);
        JButton b;
        JPanel p = new JPanel();
        getContentPane().add(p);
        p.setLayout(new FlowLayout());
        b=new JButton("Button 1");
        p.add(b);
        b=new JButton("Button 2");
        p.add(b);
        b=new JButton("Long-Named Button 3");
        p.add(b);
        b=new JButton("4");
        p.add(b);
        pack();
        setVisible(true);
    }
}
```

FlowLayout – Exemplo (2)



GridLayout



- Posiciona componentes em uma grade de linhas e colunas;
- Todas as células são do mesmo tamanho;
- Principais Construtores:
 - `GridLayout()` // uma linha - colunas se expandem
 - `GridLayout(rows,cols)`
 - `GridLayout(rows,cols,hgap,vgap)`
- Adiciona os componentes na ordem “de cima para baixo” e da “esquerda para a direita”.

GridLayout – Exemplo (1)

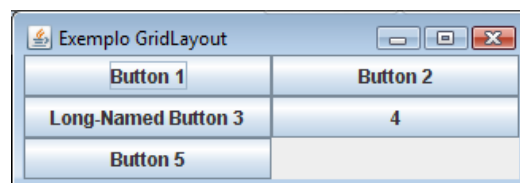


```
import javax.swing.*;
import java.awt.*;

public class ExemploGridLayout extends JFrame {
    public ExemploGridLayout(String s) {
        super(s);
        JButton b;
        JPanel p = new JPanel();
        getContentPane().add(p);
        p.setLayout(new GridLayout(0,2));
        b=new JButton("Button 1");
        p.add(b);
        b=new JButton("Button 2");
        p.add(b);
        b=new JButton("Long-Named Button 3");
        p.add(b);
        b=new JButton("4");
        p.add(b);
        b=new JButton("Button 5");
        p.add(b);
        pack();
        setVisible(true);
    }
}
```

© LES/PUC-Rio

GridLayout – Exemplo (2)



© LES/PUC-Rio

NullLayout – Exemplo (1)



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

public class ExemploNullLayout extends JFrame {
    public ExemploNullLayout(String s) {
        super(s);
        Dimension size;
        JButton b1,b2,b3;
        JPanel p = new JPanel();
        getContentPane().add(p);
        p.setLayout(null);
        b1=new JButton("Button 1");
        p.add(b1);
        b2=new JButton("Button 2");
        p.add(b2);
        b3=new JButton("Button 3");
        p.add(b3);
        Insets in=p.getInsets();
        size=b1.getPreferredSize();
        b1.setBounds(25+in.left, 5+in.top,size.width,size.height);
        size=b2.getPreferredSize();
        b2.setBounds(55+in.left,40+in.top,size.width,size.height);
        size=b3.getPreferredSize();
        b3.setBounds(150+in.left,15+in.top,size.width+50,size.height+20);
    }
}
```

© LES/PUC-Rio

NullLayout – Exemplo (2)



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

public class Main {

    public static void main(String[] args) {
        ExemploNullLayout f=new ExemploNullLayout("Exemplo NullLayout");
        Insets ins=f.getInsets();

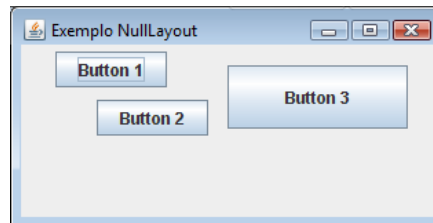
        f.setSize(300+ins.left+ins.right,125+ins.top+ins.bottom);

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

Um objeto Insets é uma representação das bordas de um contêiner. Ele especifica o espaço que um contêiner deve deixar em cada uma das suas bordas. O espaço pode ser uma borda, espaços em branco ou um título.

© LES/PUC-Rio

NullLayout – Exemplo (3)




Programa – Capítulo 11



- Outros Painéis Intermediários
- Seleção do *Look & Feel*
- Gerenciadores de Layout
- **JComponent**
- JLabel
- JButton
- JRadioButton
- JCheckBox
- JList

Laboratório de Engenharia de Software

JComponent




- Superclasse da maioria dos componentes Swing;
- Principais características:
 - *Look & feel* adaptável;
 - Tratamento de eventos;
 - *Tooltips*;
 - Tecnologias de assistência, tais como Braille;
 - Teclas de atalho.

© LES/PUC-Rio


Laboratório de Engenharia de Software


Subclasses concretas de JComponent



- Algumas classes descendentes de JComponent, definidas no pacote javax.swing:
 - JButton;
 - JLabel;
 - JMenu;
 - JMenuItem;
 - JTextField.
- Operações mais comumente aplicadas a um componente:
 - Definir dimensões;
 - Modificar cor;
 - Definir fontes;
 - Atrelar ajuda de contexto (*tool tip*).

© LES/PUC-Rio

Laboratório de Engenharia de Software	<h2>Programa – Capítulo 11</h2>	
	<ul style="list-style-type: none"> • Outros Painéis Intermediários • Seleção do <i>Look & Feel</i> • Gerenciadores de Layout • JComponent • JLabel • JButton • JRadioButton • JCheckBox • JList 	
<p>© LES/PUC-Rio</p>		

Laboratório de Engenharia de Software	<h2>Classe JLabel</h2>	
	<ul style="list-style-type: none"> • Usada para a exibição de texto e/ou imagem não editável, isto é, sem interação com o usuário; • Pode-se controlar tanto o seu alinhamento horizontal como o vertical; • Permite exibir conteúdo baseado em HTML: <ul style="list-style-type: none"> – Se o texto possuir "<html>...</html>", o conteúdo é apresentado como HTML; – O tipo de fonte é ignorado se HTML for usado. Nesse caso, o controle do tipo de fonte deve ser realizado por meio de <i>tags</i> HTML; – O suporte a HTML tem limitações. 	
<p>© LES/PUC-Rio</p>		

Classe JLabel – Principais métodos



- JLabel()
- JLabel(String text)
- JLabel(String text, Icon icon, int horizontalAlignment)
- void setText(String text)
- void setIcon(Icon icon)
- void setIconTextGap(int iconTextGap)
- void setHorizontalAlignment(int alignment)
- void setVerticalAlignment(int alignment)

© LES/PUC-Rio

Classe JLabel – Exemplo



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

public class MeuFrame extends JFrame {
    public MeuFrame(String nome) {
        super(nome);
        setLayout(new FlowLayout());
        JLabel l=new JLabel("Isso é um B-777");
        l.setIcon(new ImageIcon("B-777.jpg"));
        getContentPane().add(l);
        pack();
        setVisible(true);
    }
}
```

© LES/PUC-Rio

Classe JLabel – Exemplo



© LES/PUC-Rio


Programa – Capítulo 11



- Outros Painéis Intermediários
- Seleção do *Look & Feel*
- Gerenciadores de Layout
- JComponent
- JLabel
- **JButton**
- JRadioButton
- JCheckBox
- JList

© LES/PUC-Rio

Classe JButton




Laboratório de Engenharia de Software

- Permite criar *push-buttons*;
- Sendo uma subclasse de `AbstractButton`, herda os métodos `getLabel` e `setLabel`, que permitem consultar e alterar o seu texto;
- Permite que o botão seja cadastrado como *default button* do **RootPane**.

© LES/PUC-Rio

Classe JButton– Principais métodos



Laboratório de Engenharia de Software

- `JButton()`
- `JButton(Icon icon)`
- `JButton(String text)`
- `JButton(String text, Icon icon)`
- `void addActionListener(ActionListener l)`
- `void setBounds(int x,int y,int width,int height)`
- `void setText(String text)`
- `void setToolTipText(String text)`
- `void setEnabled(boolean b)`
- `void setVisible(boolean aFlag)`

© LES/PUC-Rio

Classe JButton – Exemplo



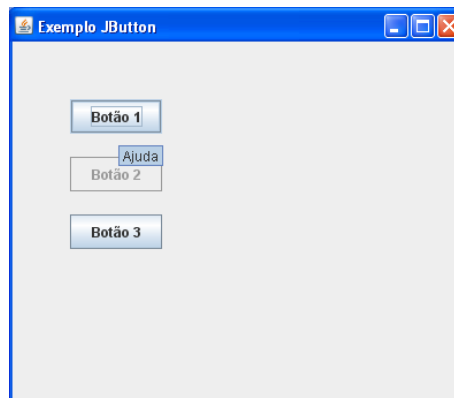
```
import javax.swing.*;
import java.awt.*;

public class MeuFrame extends JFrame {
    public MeuFrame(String nome) {
        super(nome);
        setLayout(null);
        Container c=getContentPane();
        JButton b1 = new JButton("Botão 1");
        JButton b2 = new JButton("Botão 2");
        JButton b3 = new JButton("Botão 3");
        b1.setBounds(50,50,80,30);
        b2.setBounds(50,100,80,30);
        b3.setBounds(50,150,80,30);


        b1.setToolTipText("Ajuda");
        b2.setEnabled(false);
        c.add(b1);
        c.add(b2);
        c.add(b3);
        setSize(400,350);
        setVisible(true);
    }
}
```


© LES/PUC-Rio

Classe JButton – Exemplo



© LES/PUC-Rio

Laboratório de Engenharia de Software	<h2>Programa – Capítulo 11</h2>	
	<ul style="list-style-type: none">• Outros Painéis Intermediários• Seleção do <i>Look & Feel</i>• Gerenciadores de Layout• JComponent• JLabel• JButton• JRadioButton• JCheckBox• JList	
© LES/PUC-Rio		

Laboratório de Engenharia de Software	<h2>Classe JRadioButton</h2>	
	<ul style="list-style-type: none">• Permite criar botões de escolha, que podem ser marcados e desmarcados;• Objetos do tipo JRadioButton são organizados em grupos;• Apenas um único botão de um grupo pode estar marcado em um dado momento.	
© LES/PUC-Rio		

Classe JRadioButton – Principais métodos



- JRadioButton()
- JRadioButton(Icon icon)
- JRadioButton(Icon icon, boolean selected)
- JRadioButton(String text)
- JRadioButton(String text, boolean selected)
- JRadioButton(String text, Icon icon)
- JRadioButton(String text, Icon icon, boolean selected)
- void setSelected(boolean b)
- boolean isSelected()

© LES/PUC-Rio

Classe ButtonGroup



- Cria um escopo de exclusão para um grupo de botões;
- Quando um botão de um determinado grupo é selecionado todos os demais botões do mesmo grupo são desmarcados;
- Deve-se criar um **ButtonGroup** e adicionar a ele os **JRadioButtons** que compõem o grupo.

© LES/PUC-Rio

Classe ButtonGroup – Principais métodos



- ButtonGroup()
- void add(AbstractButton b)
- ButtonModel getSelection()
- boolean isSelected(ButtonModel m)
- void setSelected(ButtonModel m, boolean b)

Classe JRadioButton – Exemplo



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

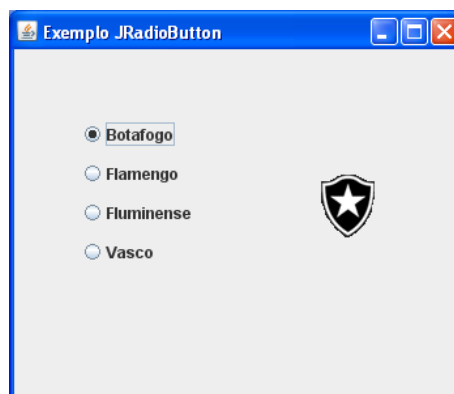
public class MeuFrame extends JFrame {
    public MeuFrame(String nome) {
        super(nome);
        setLayout(null);
        JLabel l=new JLabel();
        l.setIcon(new ImageIcon("escudo.gif"));
        Container c=getContentPane();
        JRadioButton b1=new JRadioButton("Botafogo",true);
        JRadioButton b2=new JRadioButton("Flamengo");
        JRadioButton b3=new JRadioButton("Fluminense");
        JRadioButton b4=new JRadioButton("Vasco");
        ButtonGroup bg = new ButtonGroup();
```


Classe JRadioButton – Exemplo




```
bg.add(b1);  
bg.add(b2);  
bg.add(b3);  
bg.add(b4);  
l.setBounds(230,70,100,100);  
b1.setBounds(50,50,100,30);  
b2.setBounds(50,80,100,30);  
b3.setBounds(50,110,100,30);  
b4.setBounds(50,140,100,30);  
c.add(l);  
c.add(b1);  
c.add(b2);  
c.add(b3);  
c.add(b4);  
setSize(350,300);  
setVisible(true);  
}
```

Classe JRadioButton – Exemplo



Programa – Capítulo 11




Laboratório de Engenharia de Software

- Outros Painéis Intermediários
- Seleção do *Look & Feel*
- Gerenciadores de Layout
- JComponent
- JLabel
- JButton
- JRadioButton
- **JCheckBox**
- JList

© LES/PUC-Rio

Classe JCheckBox



Laboratório de Engenharia de Software

- Permite criar caixas de escolha, que podem ser marcadas ou desmarcadas;
- Por convenção, qualquer número de *check boxes* em um grupo pode ser selecionado.

© LES/PUC-Rio

Classe JCheckBox – Principais métodos



- JCheckBox()
- JCheckBox(String label);
- JCheckBox(String label,boolean state)
- boolean isSelected()
- void setSelected(boolean state)

© LES/PUC-Rio

Classe JCheckBox – Exemplo



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

public class Main extends JFrame {
    public Main(String nome) {
        super(nome);
        setLayout(null);
        JLabel l=new JLabel();
        l.setIcon(new ImageIcon("bandeira.jpg"));
        Container c=getContentPane();
        JCheckBox cb1=new JCheckBox("Verde",true);
        JCheckBox cb2=new JCheckBox("Vermelho",false);
        JCheckBox cb3=new JCheckBox("Preto",false);
        JCheckBox cb4=new JCheckBox("Amarelo",true);
```

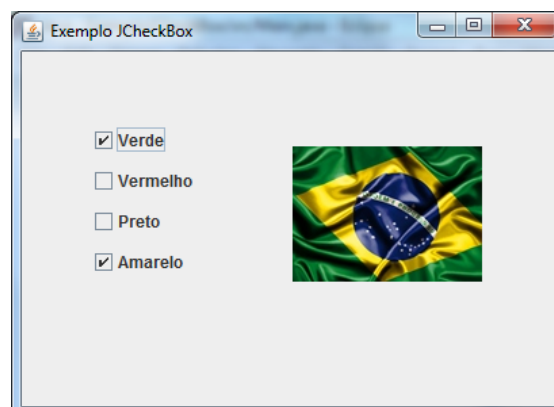
© LES/PUC-Rio

Classe JCheckBox – Exemplo



```
l.setBounds(200,70,170,100);
cb1.setBounds(50,50,100,30);
cb2.setBounds(50,80,100,30);
cb3.setBounds(50,110,100,30);
cb4.setBounds(50,140,100,30);
c.add(l);
c.add(cb1);
c.add(cb2);
c.add(cb3);
c.add(cb4);
setSize(410,300);
try {
    UIManager.setLookAndFeel
        ("com.sun.java.swing.plaf.motif.MotifLookAndFeel");
}
catch (Exception e) { }
setVisible(true);
}
```

Classe JCheckBox – Exemplo



Laboratório de Engenharia de Software

Programa – Capítulo 11




- Outros Painéis Intermediários
- Seleção do *Look & Feel*
- Gerenciadores de Layout
- JComponent
- JLabel
- JButton
- JRadioButton
- JCheckBox
- **JList**

© LES/PUC-Rio

Laboratório de Engenharia de Software

Classe JList



- Usada para criar componentes gráficos que apresentam listas de opções ao usuário;
- Permite seleção simples (um único elemento) ou múltipla (vários elementos).

© LES/PUC-Rio

Classe JList – Principais métodos



- JList()
- JList(Object[] listData)
- JList(Vector<?> listData)
- void setSelectionModel(ListSelectionModel selectionModel)
- int getSelectedIndex()
- int[] getSelectedIndices()
- Object getSelectedValue()
- Object[] getSelectedValues()

© LES/PUC-Rio

Classe JList – Modos de seleção



- Os valores dos modos de seleção foram definidos como constantes na interface **ListSelectionModel**:
 - MULTIPLE_INTERVAL_SELECTION
 - Não há restrições sobre a seleção ("default").
 - SINGLE_INTERVAL_SELECTION
 - Um intervalo contíguo de elementos pode ser selecionado
 - SINGLE_SELECTION
 - Apenas um elemento pode ser selecionado

© LES/PUC-Rio

Classe JList – Exemplo



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

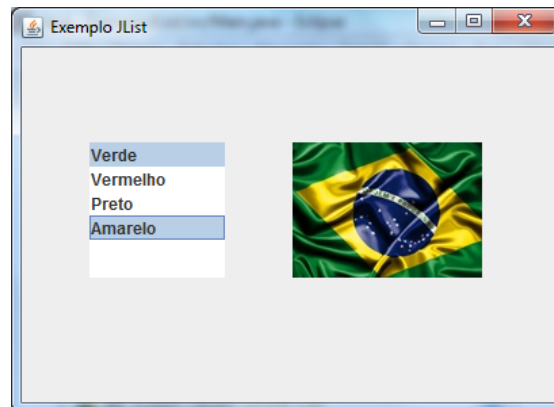
public class Main extends JFrame {
    public Main(String nome) {
        super(nome);
        setLayout(null);
        String[] lc={"Verde","Vermelho","Preto","Amarelo"};
        JLabel l = new JLabel();
        l.setIcon(new ImageIcon("bandeira.jpg"));
        Container c=getContentPane();
        JList jl=new JList(lc);
        jl.setSelectionMode
            (ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
    }
}
```

Classe JList – Exemplo



```
jl.addListSelectionListener(new MeuListListener());
l.setBounds(200,70,170,100);
jl.setBounds(50,70,100,100);
c.add(l);
c.add(jl);
setSize(410,300);
try {
    UIManager.setLookAndFeel
        ("javax.swing.plaf.metal.MetalLookAndFeel");
}
catch (Exception e) { }
setVisible(true);
}
```

Classe JList – Exemplo



© LES/PUC-Rio

Classe JList – Eventos de seleção



- Eventos de seleção são gerados sempre que a seleção de uma lista é alterada;
- Esses eventos podem ser tratados por meio da adição de um **ListSelectionListener**;
- A interface **ListSelectionListener** pertence ao pacote **javax.swing.event** e define apenas uma operação:
 - `void valueChanged(ListSelectionEvent e)`

© LES/PUC-Rio

ListSelectionListener – Exemplo



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

public class MeuListListener implements ListSelectionListener {
    public void valueChanged(ListSelectionEvent e) {
        if(e.getValueIsAdjusting())
            return;
        JList lista=(JList)e.getSource();
        if(lista.isSelectionEmpty())
        { }
        else {
            int index=lista.getSelectedIndex();
            String val=(String)lista.getSelectedValue();
            System.out.println(val);
        }
    }
}
```