

Java GUI

(Graphical User Interface)

Prof. Ricardo P. Mesquita

Exemplo de aplicação Swing

Connect

Cust ID

Name

Street

City, State

Zip Code

Item Description Qty

Item	Description	Price	Quantity
------	-------------	-------	----------

List Customers

List Items

Add Item

Submit

Exit

Status

Construção de uma aplicação Swing

- De uma forma geral, uma aplicação Swing contém os seguintes passos:
 1. Importação dos pacotes Swing
 2. Seleção da aparência (“look & feel”)
 3. Definição do **container** 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

Importando os pacotes Swing

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

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

- A maioria das aplicações Swing também precisam de dois pacotes **AWT**:

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

Selecionando o “look & feel”

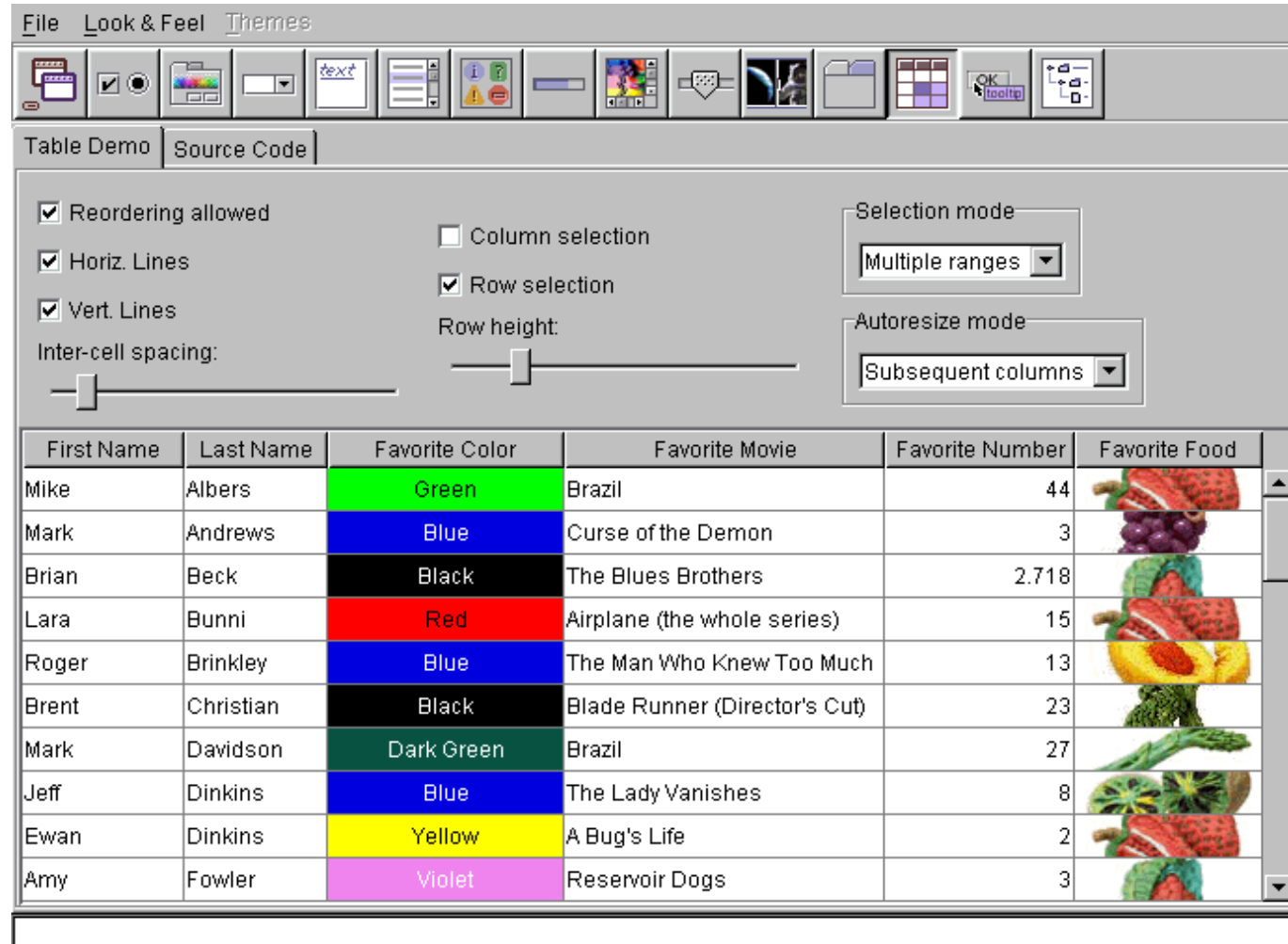
- Swing permite que o programador escolha a aparência (**look & feel**) mais apropriada através do método **setLookAndFeel**. Esse método recebe como argumento um nome inteiramente qualificado de uma subclasse de LookAndFeel.
- Exemplo:

```
try {
    UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());
    .... OU ....
    UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
} catch (Exception e) {}

JFrame.setDefaultLookAndFeelDecorated(true);

# Swing properties swing.defaultlaf=com.sun.java.swing.plaf.windows.WindowsLookAndFeel
```

Windows Look and Feel



Java Look and Feel

File Look & Feel Themes

Table Demo Source Code

☒ Reordering allowed
☒ Horiz. Lines
☒ Vert. Lines
Inter-cell spacing:

☐ Column selection
☒ Row selection
Row height:

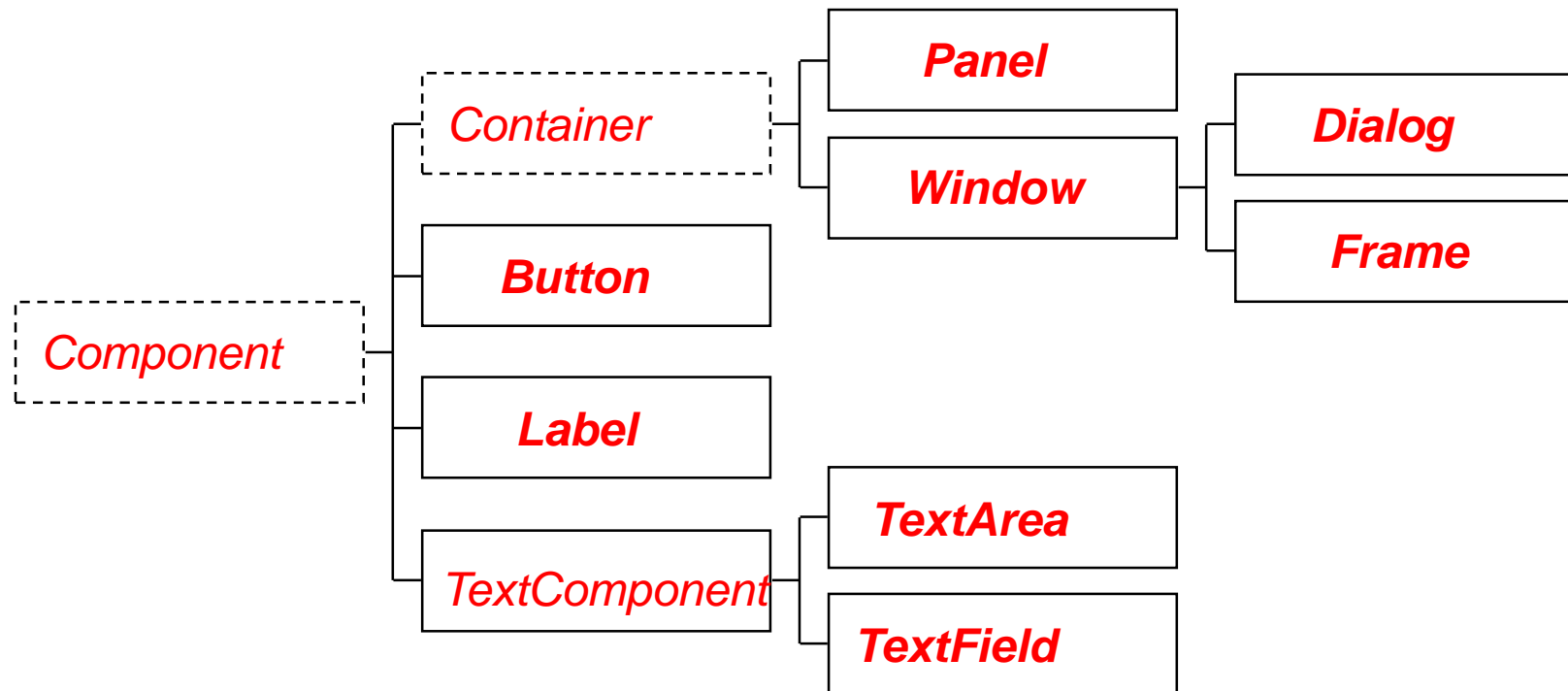
Selection mode
Multiple ranges ▼

Autosize mode
Subsequent columns ▼

First Name	Last Name	Favorite Color	Favorite Movie	Favorite Number	Favorite Food
Mike	Albers	Green	Brazil	44	
Mark	Andrews	Blue	Curse of the Demon	3	
Brian	Beck	Black	The Blues Brothers	2.718	
Lara	Bunni	Red	Airplane (the whole series)	15	
Roger	Brinkley	Blue	The Man Who Knew Too Much	13	
Brent	Christian	Black	Blade Runner (Director's Cut)	23	
Mark	Davidson	Dark Green	Brazil	27	
Jeff	Dinkins	Blue	The Lady Vanishes	8	
Ewan	Dinkins	Yellow	A Bug's Life	2	
Amy	Fowler	Violet	Reservoir Dogs	3	

Classes do pacote java.awt

- Hierarquia Básica de Classes de java.awt



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)
```

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 )
```

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

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);
    }
}
```

Componentes Concretos java.awt

- Botão

- Modela um botão

```
public void setLabel(String label)
```

```
public void setActionCommand(String command)
```

```
public void addActionListener(ActionListener l)
```

```
public void removeActionListener(ActionListener l)
```

- *Label*

- Modela um texto não editável

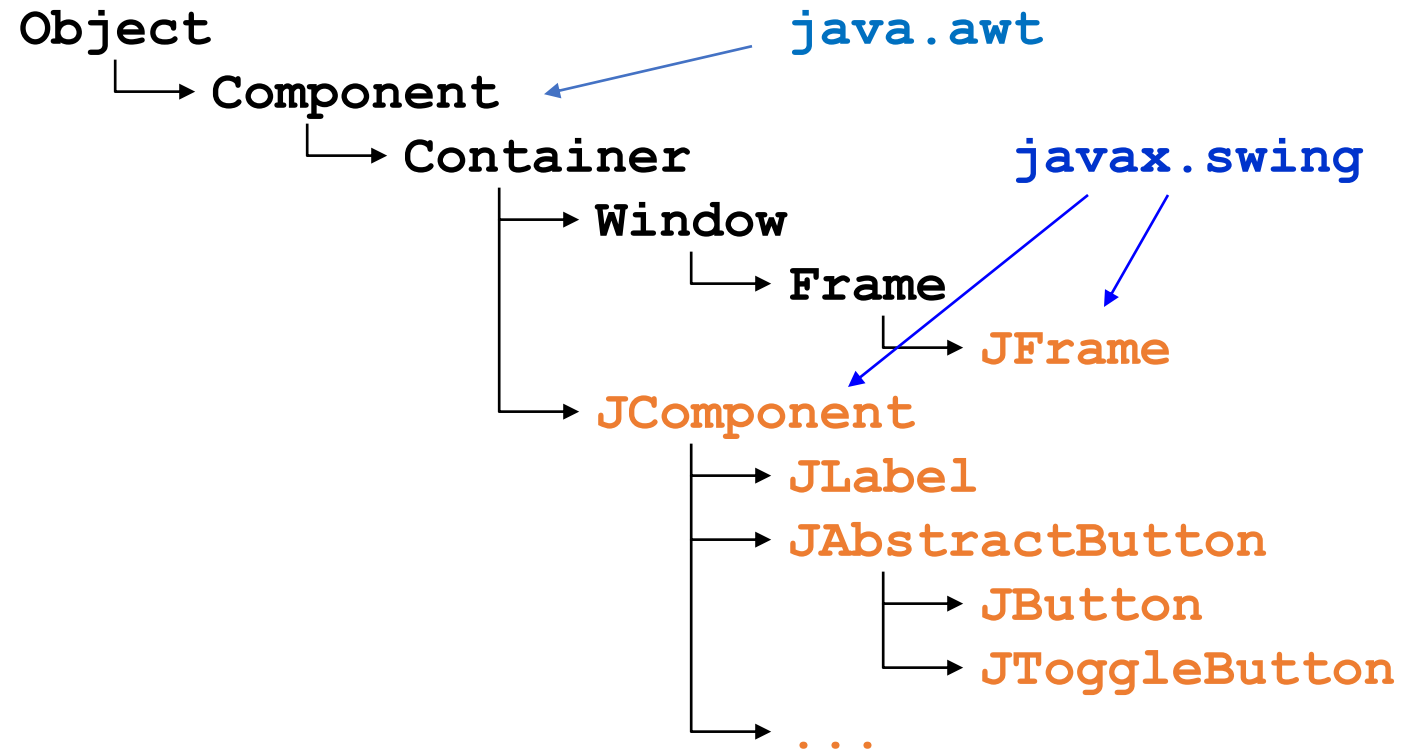
```
public void setText(String text)
```

```
public void setAlignment(int align)
```

Componentes Concretos java.awt

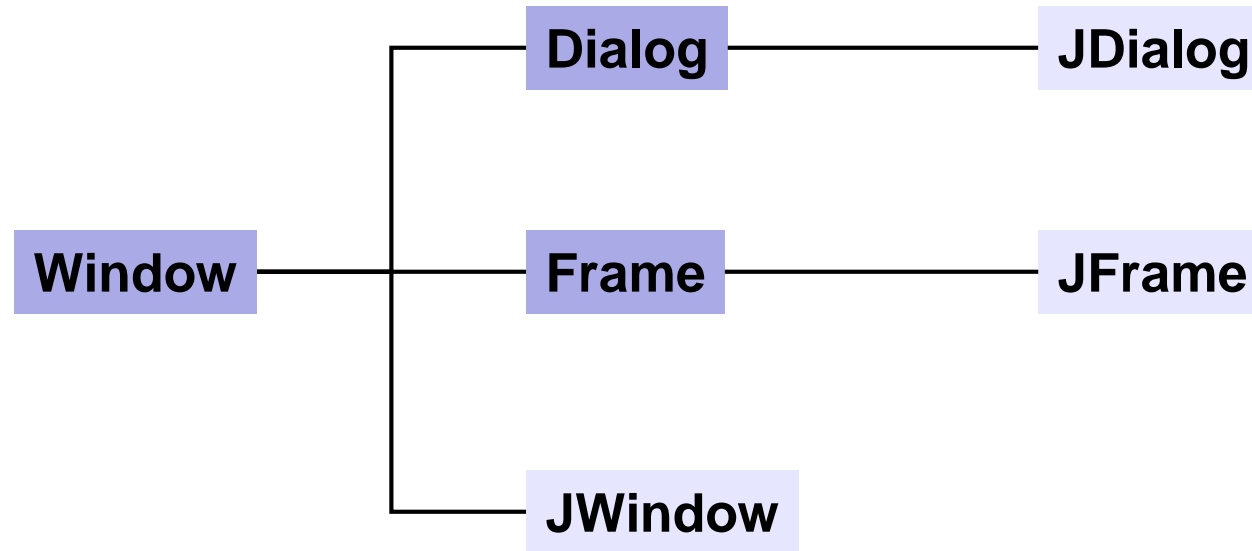
- Texto
- *CheckBox*
- Lista
- Lista *dropdown*
- *Canvas*

Classes do pacote javax.swing



Visual Index to the Swing Components

Classes de Janelas javax.swing



A classe JComponent

- A classe **JComponent** é a superclasse de todos os componentes Swing.
- Os objetos **JButton**, **JCheckbox**, e **JTextField** 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**

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()
```

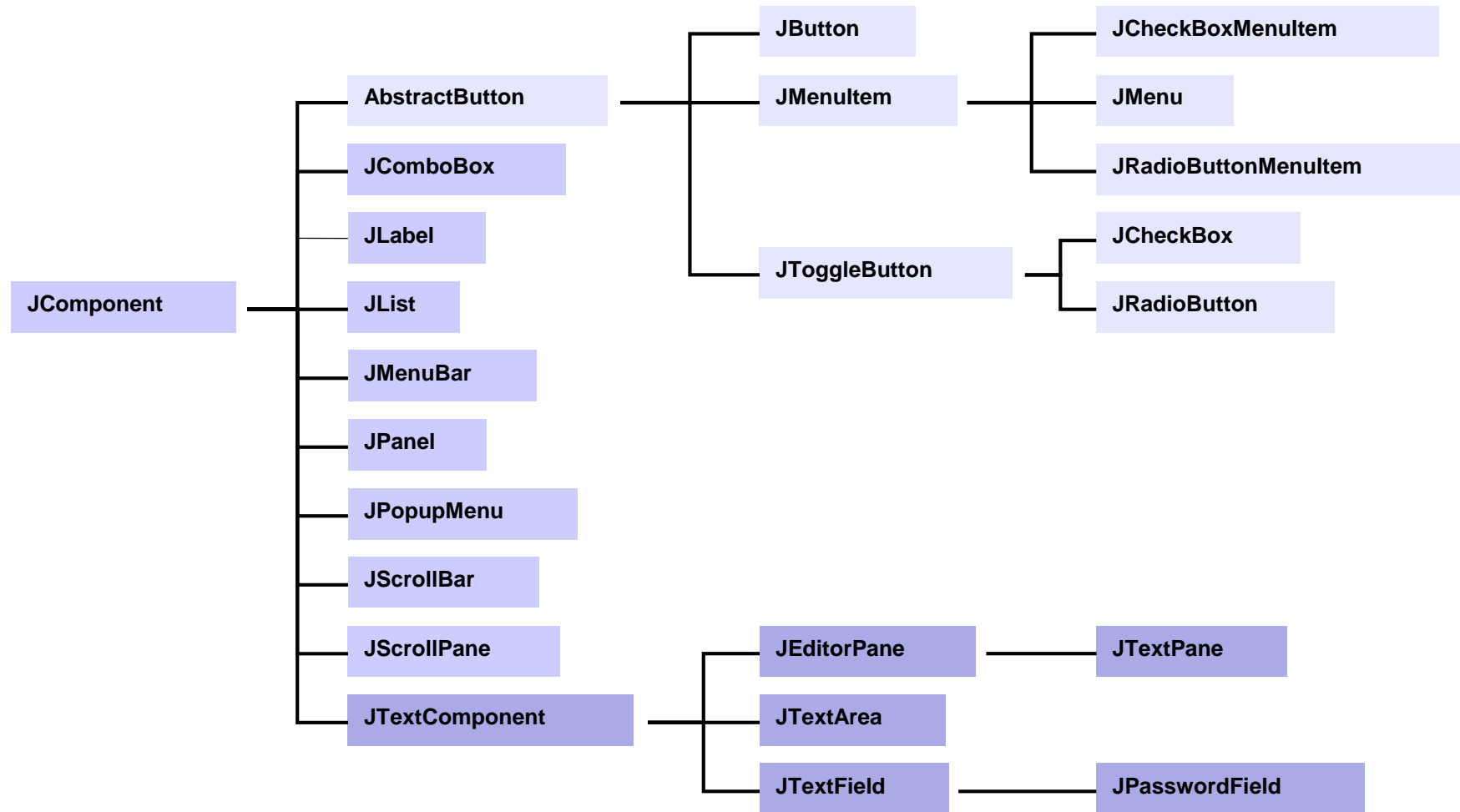
```
void setBorder(Border border)
```

```
Border getBorder()
```

```
final void putClientProperty(Object key, Object val)
```

```
final Object getClientProperty(Object key)
```

Hierarquia JComponent



A classe JComponent (cont.)

- Classes definidas no pacote javax.swing e subclasses (diretas ou indiretas) de **JComponent**:
 - JButton
 - JLabel
 - JMenu
 - JMenuItem
 - JTextField
 - JTable
 - JSlider - Simula um slider
 - JProgressBar – Exibe o progresso de uma operação.
- Comportamento comum de um JComponent
 - Definir dimensões
 - Modificar cor
 - Definir fontes
 - Atrelar ajuda de contexto (tool tip)

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

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

Container em javax.swing

- Contêineres de primeiro nível possuem um contêiner separado chamado de **"content pane"** que contém os elementos visíveis. Os demais componentes são adicionados no **"content pane"**.
- Para adicionar um componente a um **JApplet**, **JDialog**, **JFrame**, ou **JWindow**, você deve primeiro invocar o método **getContentPane** e adicionar o componente ao objeto retornado.

Classe JFrame

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

`JRootPane getRootPane()`

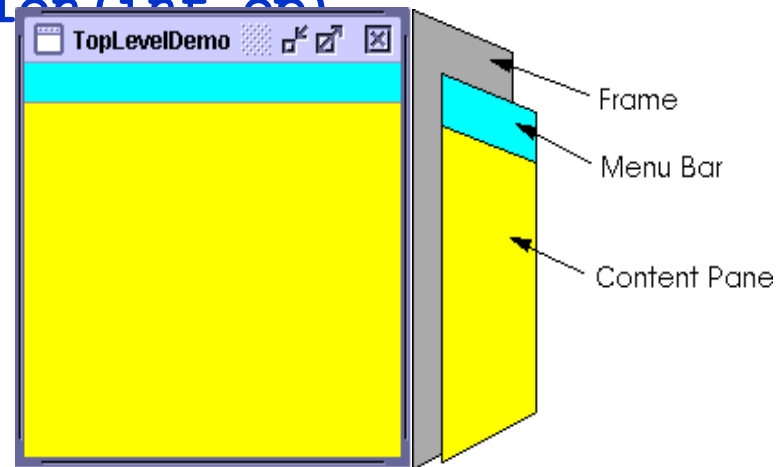
`Container getContentPane()`

`void setJMenuBar(JMenuBar menubar)`

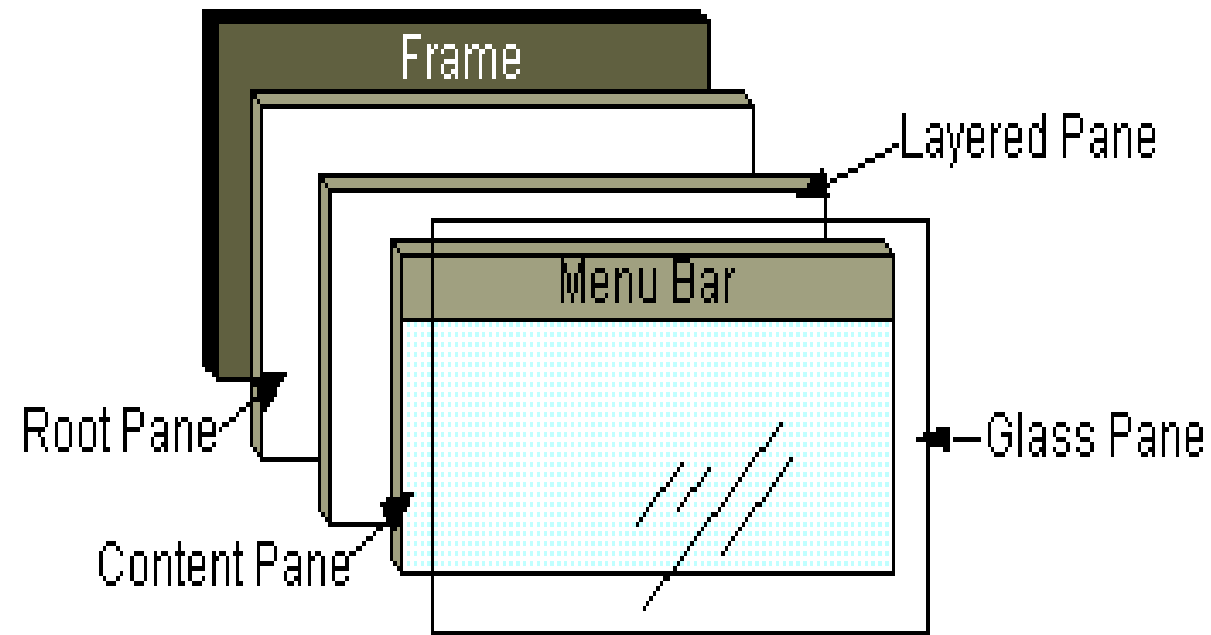
`JMenuBar getJMenuBar()`

`void setDefaultCloseOperation(int on)`

- **EXIT_ON_CLOSE**
- **HIDE_ON_CLOSE**
- **DISPOSE_ON_CLOSE**
- **DO_NOTHING_ON_CLOSE**



Estrutura de um JFrame



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

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");
```

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();  
            }  
        });  
    }  
}
```



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)
```

Exemplo de JPanel

```
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);
```

Gerenciadores de Layout

- 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...

Gerenciadores de Layout (cont.)

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

GridLayout

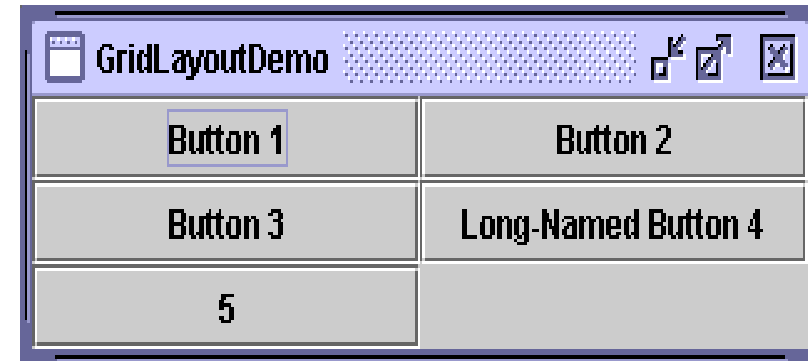
- Posiciona componentes em uma grade de linhas e colunas, com todas as células são de tamanho igual.
- Principais Construtores
 - `p.setLayout(new GridLayout()`; // One row. Columns expand.
 - `p.setLayout(new GridLayout(rows, cols))`;
 - `p.setLayout(new GridLayout(rows, cols, hgap, vgap))`;
- Não se pode escolher aleatoriamente em que célula posicionar um componente.
 - Adicione-os na ordem “de cima para baixo” e da “esquerda para a direita”.
 - Conseqüentemente, não é possível deixar uma célula “indefinida”.

Exemplo: GridLayout

- Exemplo

```
panel.setLayout(new GridLayout(0,2));  
panel.add(new JButton("Button 1"));  
panel.add(new JButton("Button 2"));  
panel.add(new JButton("Button 3"));  
panel.add(new JButton("Long-Named Button 4"));  
panel.add(new JButton("5"));
```

[GridLayoutDemo](#)



BorderLayout

- 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.

```
...//Container panel = aFrame.getContentPane()...
```

```
panel.setLayout(new BorderLayout());
```

```
JButton b = new JButton("Button 1 (PAGE_START)");
```

```
panel.add(b, BorderLayout.PAGE_START);
```

```
b = new JButton("Button 2 (CENTER)");
```

```
b.setPreferredSize(new Dimension(200, 100));
```

```
panel.add(b, BorderLayout.CENTER);
```

```
b = new JButton("Button 3 (LINE_START)");
```

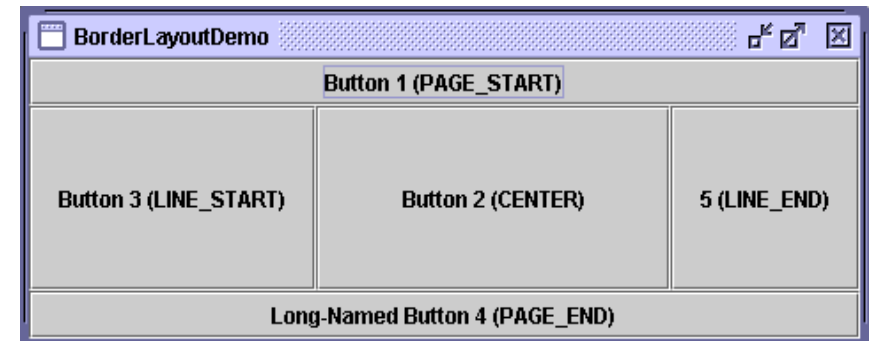
```
panel.add(b, BorderLayout.LINE_START);
```

```
b = new JButton("Long-Named Button 4 (PAGE_END)");
```

```
panel.add(b, BorderLayout.PAGE_END);
```

```
b = new JButton("5 (LINE_END)");
```

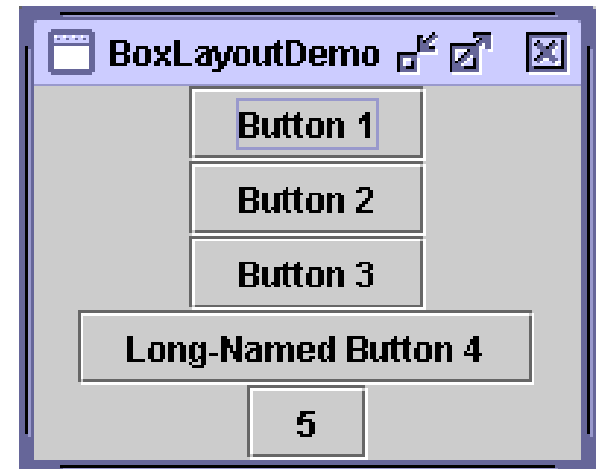
```
panel.add(b, BorderLayout.LINE_END);
```



BoxLayout

- Empilha seus componentes em linha ou em coluna de acordo com opção selecionado.

```
...//Container panel = aFrame.getContentPane()...  
panel.setLayout(new BoxLayout(pane, BoxLayout.Y_AXIS));  
JButton b = new JButton("Button 1");  
b.setAlignmentX(Component.CENTER_ALIGNMENT); panel.add(b);  
b = new JButton("Button 2");  
b.setAlignmentX(Component.CENTER_ALIGNMENT); panel.add(b);  
b = new JButton("Button 3");  
b.setAlignmentX(Component.CENTER_ALIGNMENT);  
panel.add(b);  
b = new JButton("Long-Named Button 4");  
b.setAlignmentX(Component.CENTER_ALIGNMENT);  
panel.add(b);  
b = new JButton("5");  
b.setAlignmentX(Component.CENTER_ALIGNMENT);  
panel.add(b);
```



FlowLayout

- Empilha seus componentes em linha ou em coluna de acordo com opção selecionado.

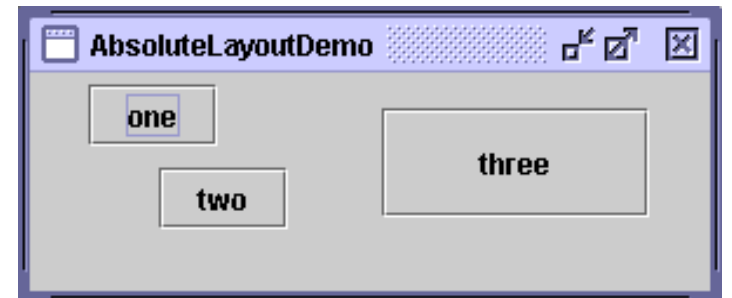
```
...//Container panel = aFrame.getContentPane() ...  
panel.setLayout(new FlowLayout());  
JButton b = new JButton("Button 1"); panel.add(b);  
b = new JButton("Button 2"); panel.add(b);  
b = new JButton("Button 3"); panel.add(b);  
b = new JButton("Long-Named Button 4"); panel.add(b);  
b = new JButton("5"); panel.add(b);
```



NullLayout

- Posiciona seus componentes de acordo com a posicao absoluta

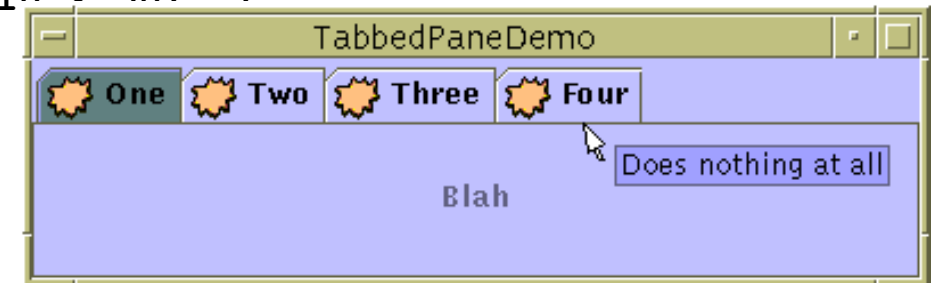
```
...//Container panel = aFrame.getContentPane()...  
panel.setLayout(null);      JButton b1,b2,b3;    Dimension size;  
b1 = new JButton("one"); panel.add(b1);  
b2 = new JButton("two"); panel.add(b2);  
b3 = new JButton("three"); panel.add(b3);  
Insets insets = panel.getInsets();  
size = b1.getPreferredSize();  
b1.setBounds(25+insets.left, 5+insets.top, size.width, size.height);  
size = b2.getPreferredSize();  
b2.setBounds(55+insets.left, 40+insets.top, size.width, size.height);  
size = b3.getPreferredSize();  
b3.setBounds(150+insets.left, 15+insets.top, size.width+50, size.height+20);  
  
...//In the main method:  
Insets insets = frame.getInsets();  
frame.setSize(300+insets.left+insets.right,  
              125+insets.top+insets.bottom);
```



JTabbedPane

- JTabbedPane permiti que diversos componentes, em geral JPanel, compartilhem um mesmo espaço.

```
JTabbedPane tP = new JTabbedPane(); JComponent p1, p2, p3, p4;  
ImageIcon icon = createImageIcon("images/middle.gif");  
p1 = makeTextPanel("Panel #1");  
tP.addTab("One", icon, p1, "Does nothing");  
tP.setMnemonicAt(0, KeyEvent.VK_1);  
p2 = makeTextPanel("Panel #2");  
tP.addTab("Two", icon, p2, "Does twice as much nothing");  
tP.setMnemonicAt(1, KeyEvent.VK_2);  
p3 = makeTextPanel("Panel #3");  
tP.addTab("Three", icon, p3, "nothing"); tP.setMnemonicAt(2, KeyEvent.VK_3);  
p4 = makeTextPanel("Panel #4 (has a preferred size of 410 x 50)");  
p4.setPreferredSize(new Dimension(410, 50));  
tP.addTab("Four", icon, p4, "Does nothing at all");  
tP.setMnemonicAt(3, KeyEvent.VK_4);
```



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.
 - JLabel fontes são ignoradas se HTML é usado. Nesse caso, todo o de fontes deve ser realizado através de tags HTML.
 - Deve ser usado **<P>**, e não **
**, para forçar uma quebra de linha. O suporte HTML ainda é incipiente.

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();
```

JLabel

```
String labelText =
    "<html><FONT COLOR=WHITE>WHITE</FONT> and " +
    "<FONT COLOR=GRAY>GRAY</FONT> Text</html>";
JLabel coloredLabel =
    new JLabel(labelText, JLabel.CENTER);
...
labelText =
    "<html><B>Bold</B> and <I>Italic</I> Text</html>";
JLabel boldLabel =
    new JLabel(labelText, JLabel.CENTER);
labelText =
    "<html>The Applied Physics Laboratory is..." +
    "of the Johns Hopkins University." +
    "<P>" + ... "...</html>";
```

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()
```

Métodos de JButton

- Instanciação (exemplos)

- `JButton btnOk = new javax.swing.JButton();`
- `JButton proximo = new JButton(new ImageIcon("right.gif"));`
- `JButton proximo = new JButton("Next", rightArrow);`
- `JButton button = new JButton("I'm a button!");`
- `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)`

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();
```

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)
```

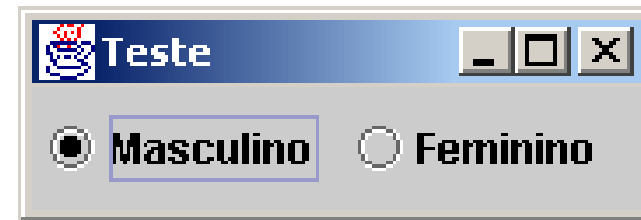
```
JRadioButton(String label, boolean state)
```

```
boolean isSelected()
```

```
void setSelected(boolean state)
```

Exemplo de JRadioButton

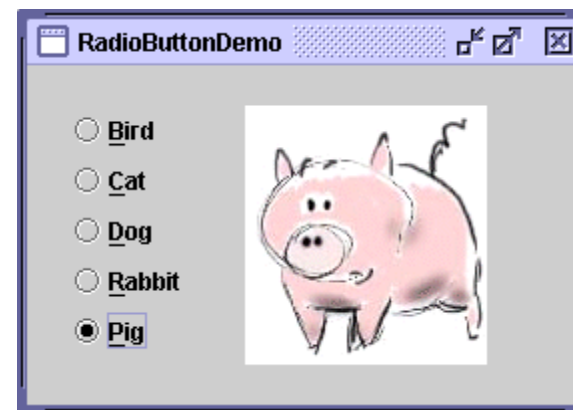
```
JFrame f = new JFrame("Teste");  
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
JRadioButton bm = new JRadioButton("Masculino", true);  
JRadioButton bf = new JRadioButton("Feminino");  
ButtonGroup bg = new ButtonGroup();  
bg.add(bm);  
bg.add(bf);  
Container cp = f.getContentPane();  
cp.setLayout(new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



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)
```

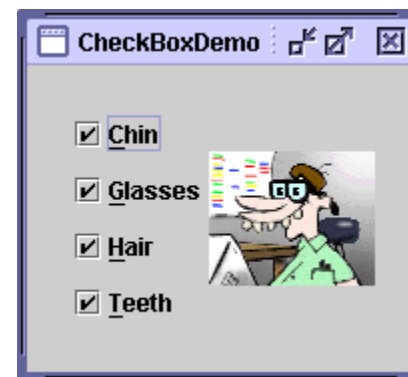


Classe JCheckBox

- 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)
```



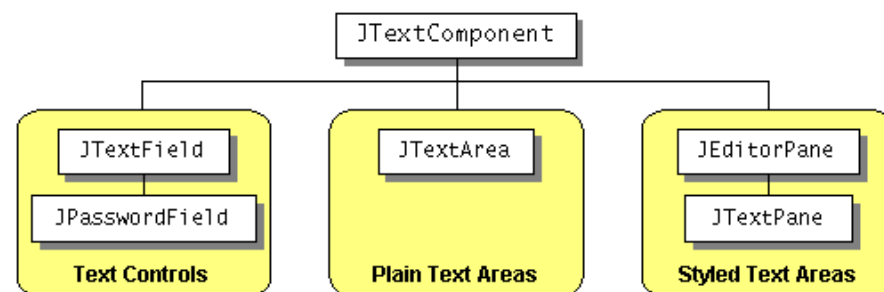
Classe JTextComponent

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

```
public void setText(String t)
```

```
public String getText()
```

```
public void setEditable(boolean b)
```



Classe JTextField

- Cria campo de edição de texto de uma linha

```
JTextField()  
JTextField(String text)  
JTextField(int columns)  
JTextField(String text, int columns)  
void setColumns(int columns)
```

- Principais métodos

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

Classe JPasswordField

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

```
JPasswordField()
```

```
JPasswordField(int columns)
```

```
JPasswordField(String text, int columns)
```

```
char[] getPassword()
```

```
void setEchoChar(char c)
```

JFormattedTextField - Máscara de Entrada

- Estende **JTextField** (a partir de 1.4)
- 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.

JFormattedTextField

- Cep

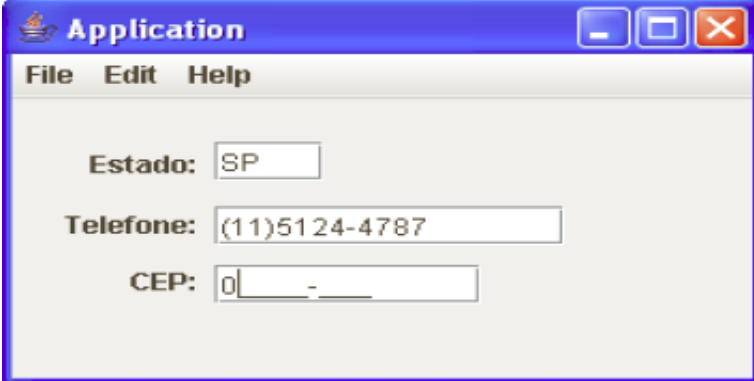
```
MaskFormatter mask = null;

try {
    mask = new MaskFormatter("#####-###");
    mask.setPlaceholderCharacter('_');
} catch (java.text.ParseException exc) {
    .....
}

JFormattedTextField cep =
    new JFormattedTextField(mask);
```

- Telefone

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



The screenshot shows a Java application window titled "Application" with a menu bar containing "File", "Edit", and "Help". The main content area contains three text input fields with labels: "Estado:" followed by a field containing "SP"; "Telefone:" followed by a field containing "(11)5124-4787"; and "CEP:" followed by a field containing "0" and two underscores as placeholders.

Caractere	Descrição
#	Qualquer número válido, usa Character.isDigit.
,	Caractere de escape, usado para o escape de qualquer caractere de formato especial.
U	Qualquer caractere(Character.isLetter). Todas as letras minúsculas são transformadas em maiúsculas.
L	Qualquer caractere(Character.isLetter). Todas as letras são transformadas para minúsculas.
A	Qualquer caractere ou número (Character.isLetter ou Character.isDigit).
?	Qualquer caractere.
*	Qualquer coisa.
H	Qualquer caractere hexadecimal (0-9, a-f ou A-F).

Classe JTextArea

- Cria um campo de edição de texto com múltiplas linhas

```
JTextArea(int rows, int columns)
```

```
JTextArea(String text, int rows, int columns)
```

```
void append(String t)
```

```
void insert(String t, int pos)
```

```
void setLineWrap(boolean wrap)
```

```
void setWrapStyleWord(boolean word)
```

Classe JTextPane

- Componente de texto que suporta atributos representados graficamente (*styled text*)
- Permite o uso de diversas “fontes” no texto
- Permite a inclusão de imagens e de outros componentes

