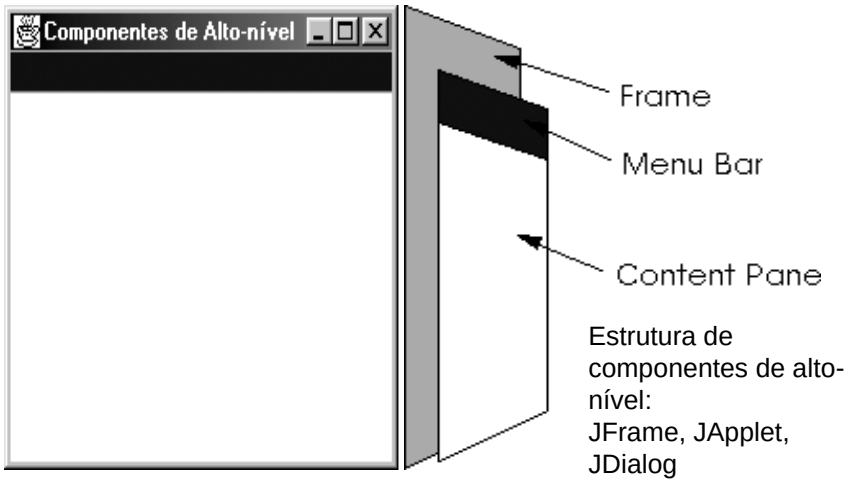
Componentes Swing



Estrutura de componentes de alto-nível:
JFrame, JApplet, JDialog

JPanel

- **Um container genérico e visual. Ela trabalha em cooperação com o controle de layouts.**
- **O construtor padrão cria um objeto JPanel com FlowLayout, porém diferentes layouts podem ser especificados durante a construção ou através do método setLayout().**

JPanel e JFrame

- JPanel jpanel=new JPanel( );
- JButton b=new JButton("Botão");
- panel.add(b);
- JFrame f=new JFrame( );
- f.getContentPane( ).add(jpanel); // ContentPane
- f.pack( ); // para exibir o frame deve-se
- f.setVisible(true); // pack+setVisible

Componente JFrame

JFrame

```
JFrame frame = new JFrame("Teste: Frame");
frame.setSize(800, 800);
ImageIcon image = new ImageIcon("midle.gif");
frame.setIconImage(image.getImage( ));

frame.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e)
 {System.exit(0);}
});

 JLabel label = new JLabel("");
 label.setPreferredSize(new Dimension(400, 100));
 frame.getContentPane().add(label, BorderLayout.NORTH);
 frame.pack( );
 frame.setVisible(true);
```

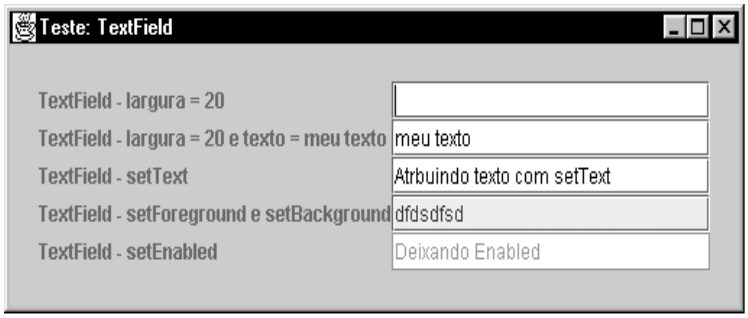

JButton

- JButton (String, Icon)
- JButton (String)
- JButton (Icon)
- JButton ( )
- setMnemonic(char) // tecla que substitui o botão
- setEnabled(boolean)
- setActionCommand(String)
- addActionListener(ActionListener)
- removeActionListener(ActionListener)

JLabel

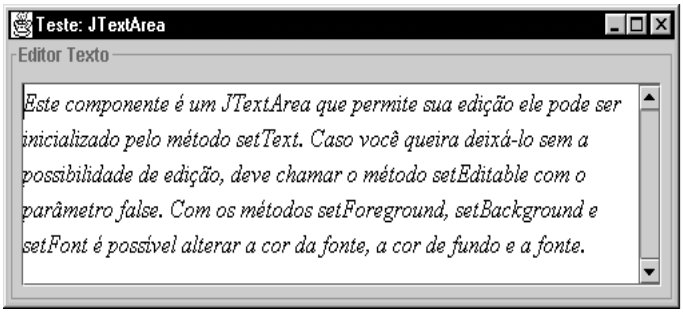
- Apresenta textos e/ou imagens não selecionáveis
- JLabel (String), JLabel (Icon)
- JLabel(String, Icon, int)
  - alinhamento: LEFT, CENTER, RIGHT, LEADING ou TRAILING
- setText(String), getText( ), setIcon(Icon), getIcon( )
  - seta ou capta o texto/imagem do label

JTextField



- Usado para entrada de texto de uma linha
  - JTextField( )
  - JTextField(int tamanho)
  - JTextField(String frase)
  - JTextField(String frase, int tamanho)

JTextArea



- **Usada para entrada de textos extensos**
  - **JTextArea(), JTextArea(int linhas, int colunas)**
  - **JTextArea(String texto), JTextArea(String texto, int linhas, int cols)**

JCheckBox

- **Botão de seleção, utilizado para entrada de informações de escolha tipo sim ou não**
  - **JCheckBox (String)**
  - **JCheckBox (String, boolean)**
  - **JCheckBox (Icon)**
  - **JCheckBox (Icon, boolean)**
  - **JCheckBox (String, Icon)**
  - **JCheckBox (String, Icon, boolean)**
  - **JCheckBox ( )**

JCheckBox

```
public JanelaCheckBox( ) extends JFrame implements ActionListener
{
 public JanelaCheckBox( )
 {
     JPanel p = new JPanel( );
     negrito = opcaoCheckBox (p, “Negrito”);
     italico = opcaoCheckBox (p, “Italico”);
     add (p, “South”);
 }

private CheckBox negrito;
private CheckBox italico;
// o método abaixo devolve uma CheckBox criada e
// e acrescentada no painel p.
// o tratamento de eventos é feito no próprio Frame JanelaCheckBox

JCheckBox opcaoCheckBox (JPanel p, String nome)
{
     JCheckBOx c = new CheckBox (nome);
     c.addActionListener (this);
     p.add(c);
     return c;
}
```


JCheckBox

- **Para indicar como escolhida uma opção**
  - **negrito.setSelected(true);**
- **Quando uma opção não está escolhida**
  - **O clique causa um *action event* que é tratado pelo método *actionPerformed***
- **O estado da seleção pode ser testado**
  - **negrito.isSelected( )**
  - **italico.isSelected( )**

JCheckBox

```
// Verificando a caixa marcada e realização a ação.
// No caso, inicializando a variável m com fonte bold e/ou
  itálico

public void actionPerformed (ActionEvent evt)
{
  Font m = (negrito.isSelected( ) ? Font.BOLD : 0) +
           (italico.isSelected( )? Font.ITALIC : 0 );
  panel.setFont (m);
}
```


JRadioButton

- Botão de seleção, utilizado para permitir a seleção de apenas uma opção
  - JRadioButton(String)
  - JRadioButton(String, boolean)
  - JRadioButton(Icon)
  - JRadioButton(Icon, boolean)
  - JRadioButton(String, Icon)
  - JRadioButton(String, Icon, boolean)
  - JRadioButton( )
- addItemListener(ItemListener)
- removeItemListener( )

JRadioButton

```
class RadioButtonFrame extends JFrame implements actionListener
{
 public RadioButtonFrame( )
  {
   setTitle (“RadioButtonTest”);
   ...
   // smallButton, etc. são private
   smallButton = new JRadioButton (“Small”, false);
   mediumButton = new JRadioButton (“Medium”, true);
   ...
   ButtonGroup group = new ButtonGroup( );
   group.add (smallButton);
   group.add (mediumButton);
   ...
  }
```

(continua...)

JRadioButton

```
(...continua)

// verificando o botão marcado e realizando a ação

public void actionPerformed (ActionEvent evt)
{
  Object botao = evt.getSource( );
  if (source == smallButton)
    panel.setSize (8);
  else
     if (source == mediumButton)
       panel.setSize(12);
    else
         if (source == largeButton)
           panel.setSize(14);
        else ...
}
```

JComboBox

- Quando um JComboBox é clicado são disponibilizadas opções para o usuário
- Funciona como uma lista que é apresentada quando o campo é clicado

JComboBox

- JComboBox( ),
- JComboBox(ComboBoxModel),
- JComboBox(Object[ ]),
- JComboBox(Vector)
- Métodos
  - void addItem(Object)
    - adiciona um item no JComboBox
  - void insertItemAt(Object, int)
    - adiciona um item no JComboBox na posição especificada
  - Object getItemAt(int)
    - retorna o item da posição
  - Object getSelectedItem()
    - retorna o item selecionado

JComboBox

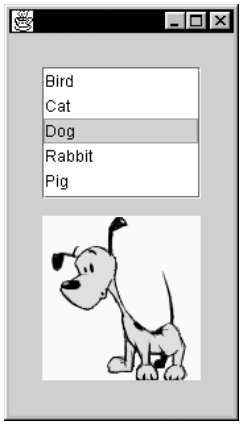
```
class ComboBoxFrame extends JFrame implements ActionListener
{
 public ComboBoxFrame( )
  {  setTitle("ComboBoxTest");
     style = new JComboBox( );
     style.setEditable(true);
     style.addItem("Serif");
     style.addItem("SansSerif");
     style.addItem("Monospaced");
     style.addItem("Dialog");
     style.addItem("DialogInput");
     style.addActionListener(this);
     JPanel p = new JPanel( );
     p.add(style);
     getContentPane( ).add(p, "South");
     panel = new ComboBoxTestPanel( );
     getContentPane( ).add(panel, "Center");

 public void actionPerformed(ActionEvent evt)
  {  JComboBox source = (JComboBox)evt.getSource( );
     String item = (String)source.getSelectedItem( );
     panel.setStyle(item);
  }}
```


JList

- Apresenta um grupo de itens colocados em uma coluna, possibilitando ao usuário a seleção de um ou mais destes itens de diferentes tipos:
  - SINGLE_SELECTION: seleção simples
  - SINGLE_INTERVAL_SELECTION: seleção de um único intervalo
  - MULTIPLE_INTERVAL_SELECTION: seleção de múltiplos intervalos

JList



- **Construtores**
  - **JList(ListModel)**
  - **JList(Object[ ])**
  - **JList(Vector)**
- **Métodos**
  - **void setModel(ListModel) fixa uma ListModel para a JList**
  - **ListModel getModel() retorna a ListModel da Jlist**
  - **void addListSelectionListener (ListSelectionListener) acrescenta um ListSelectionListener para a JList**

JList

- **Alguns métodos**
  - **void setSelectedIndex(int) marca como selecionado o elemento do índice passado com parâmetro**
  - **void setSelectedValue(Object, boolean) marca como selecionado ou não selecionado o Object da Jlist**
  - **int getSelectedIndex( ) retorna o índice do elemento selecionado**
  - **int getMinSelectionIndex( ) retorna o menor índice dos elementos selecionados**
  - **int getMaxSelectionIndex( ) retorna o maior índice dos elementos selecionados**
  - **int[ ] getSelectedIndices( ) retorna os índices dos elementos selecionados**

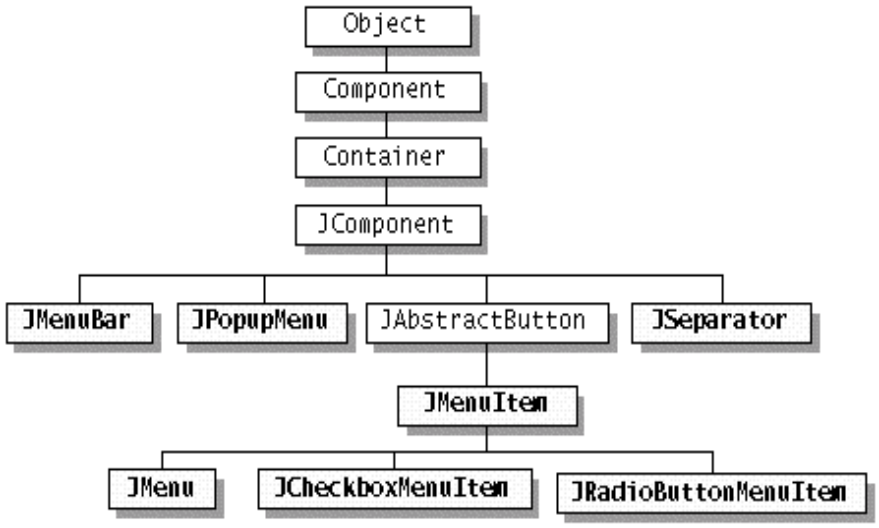
JList

- Alguns métodos
  - Object getSelectedValue( ) retorna o Objeto selecionado
  - Object[ ] getSelectedValues( ) retorna os Objetos selecionados
  - void setSelectionMode(int) seleciona o tipo de seleção utilizada na JList
  - int getSelectionMode( ) retorna o tipo de seleção utilizada na JList

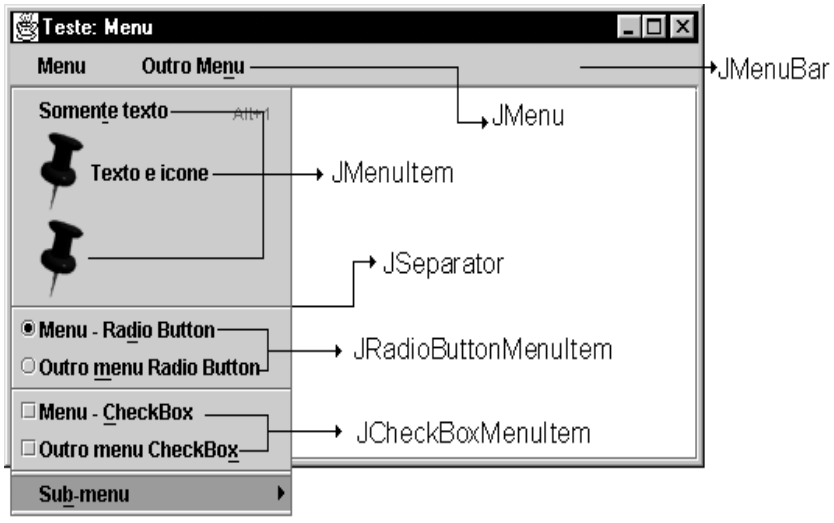
Menus

- Barras de menus podem ser criadas no topo de janelas
- Sendo componentes, as barras podem ser inseridas em qualquer container
- Elementos na barra de menu são nomes de menus pull-down, que aparecem quando o nome é selecionado na barra

Menus


Object
Component
Container
JComponent
JMenuBar
JPopupMenu
JAbstractButton
JSeparator
JMenuItem
JMenu
JCheckboxMenuItem
JRadioButtonMenuItem

Menus


Teste: Menu
Menu
Outro Menu
JMenuBar
Somente texto
JMenu
Texto e icone
JMenuItem
JSeparator
Menu - Radio Button
Outro menu Radio Button
JRadioButtonMenuItem
Menu - CheckBox
Outro menu CheckBox
JCheckBoxMenuItem
Sub-menu

Menus

- **Criando a barra superior numa subclasse de JFrame**

```
JMenuBar menuBar = new JMenuBar( );
setJMenuBar(menuBar);//método de Frame
// especifica a cor
menuBar.setForeground(Color.red);
```

- **Se não é diretamente subclasse de JFrame:**

```
frame.setJMenuBar(menuBar);
```


Menus

- **Criando submenus e adicionando na barra de menus**

```
JMenu menu = new JMenu("Menu");
menu.setMnemonic('e');
menuBar.add(menu); // adiciona o menu
                   // “Menu” na barra de menus
```

Menus

- **Criando e adicionando itens no sub-menu**

```
menuItem = new JMenuItem(”Somente texto”);
menuItem.addActionListener(this);
menu.add(menuItem);
// Texto e icone
JMenuItem mi = new JMenuItem("Texto e icone",icon);
// icon é um ImageIcon
menu.add(mi);
mi.addActionListener(this);
```


Menus

- **Criando e adicionando itens no sub-menu com teclas de atalho**

```
// tecla T aciona o menu
JMenuItem mi = new JMenuItem("Somente
texto",KeyEvent.VK_T);
// ALT + 1 aciona o menu
mi.setAccelerator(KeyStroke.getKeyStroke(
                KeyEvent.VK_1, ActionEvent.ALT_MASK));
```

Menus

- **Itens apenas com ícone e uso de separadores**

```
// criando menu somente imagem
JMenuItem = new JMenuItem(icon);
menuItem.addActionListener(this);
menu.add(menuItem);
menu.addSeparator( );
```


Menus

- **Criando RadioButtons como itens de um menu**

```
ButtonGroup group = new ButtonGroup( );
rbMenuItem = new JRadioButtonMenuItem("Menu – Radio Button");
rbMenuItem.setSelected(true);
rbMenuItem.setMnemonic('d');
group.add(rbMenuItem);
rbMenuItem.addItemListener(this);
menu.add(rbMenuItem);
rbMenuItem = new JRadioButtonMenuItem("Outro menu Radio Button");
rbMenuItem.setMnemonic('m');
group.add(rbMenuItem);
rbMenuItem.addItemListener(this);
menu.add(rbMenuItem);
```

Menus

- **Adicionando CheckBox como itens de menu**

```
cbMenuItem = new JCheckBoxMenuItem("Menu -
 CheckBox");
cbMenuItem.setMnemonic('c');
cbMenuItem.addItemListener(this);
menu.add(cbMenuItem);
```


Menus

- **Tratando a escolha de uma opção do menu**

```
public void actionPerformed(ActionEvent e)
{
  JMenuItem source = (JMenuItem)(e.getSource( ));
   String s = source.getText();
   if(s.equals(”Soemente texto")) // faz alguma ação
   else
     if(s.equals(”Texto e icone")) //faz outra ação
     else
       if(s.equals("")) // faz ação da 3a. opção
       else ....
```

Menus

- Tratando a mudança de estado dos RadioButtons e CheckBoxes

```
public void itemStateChanged(ItemEvent e) {
   JMenuItem source = (JMenuItem)(e.getSource( ));
   String s = source.getText( );
   if (e.getStateChange( ) == ItemEvent.SELECTED){
    if(s.equals("Menu-RadioButton")) // faz algo
    else
        if(s.equals("Outro radio button"))  //
        else  ...
```


Menus

- Se não houver necessidade de realização de uma ação no exato momento da mudança de estado, basta testar o estado do item:

```
JCheckBoxMenuItem negrito = new JCheckBoxMenuItem
  ("negrito");
menu.add(negrito);
JCheckBoxMenuItem italico = new JCheckBoxMenuItem
  ("italico");
menu.add(negrito);
if (negrito.isSelected) // faz algo
```

Outros recursos: JSplitPane

- Dispõe dois componentes, lado a lado ou um sobre o outro
- Pode-se definir o espaço de cada componente arrastando o divisor
- Podem ser aninhados, arranjando vários componentes
- Componentes podem ser definidos dinamicamente

JSplitPane: alguns métodos

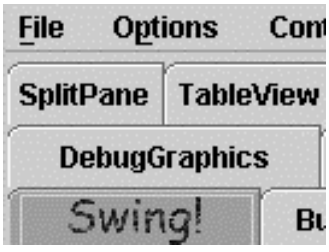
- JSplitPane([int], [boolean], [Component, Component])
  - orientação: HORIZONTAL_SPLIT (default) ou VERTICAL_SPLIT
  - boolean: componentes redesenhados continuamente
- setOrientation(int), getOrientation();

JSplitPane: alguns métodos

- setBottomComponent(Component), getBottomComponent()
- setTopComponent(Component), getTopComponent()
- setLeftComponent(Component), getLeftComponent()
- setRightComponent(Component), getRightComponent()

Outros recursos Swing: JTabbedPane

- Vários componentes dividindo o mesmo espaço
- Um tab correspondente a cada componente
- Tabs podem aparecer em cima, embaixo, à esquerda ou à direita
- Não é necessário tratamento de eventos
  - JTabbedPane trata automaticamente eventos de mouse



JTabbedPane: alguns métodos

- JTabbedPane([int])
  - o argumento opcional indica a posição: TOP (padrão), BOTTOM, RIGHT, LEFT;
- addTab(String, [Icon], Component, [String])
  - o primeiro argumento especifica o texto do tab, o último especifica o tool tip;
- insertTab(String, Icon, Component, String, int)
  - insere o tab em um índice determinado

JTabbedPane: alguns métodos

- indexOfTab(String | Icon)
  - retorna o índice do tab especificado
- setSelectedIndex(int)
- setSelectedComponent(Component)
  - seleciona o tab especificado e seu componente
- getSelectedIndex(),
- getSelectedComponent()
  - retorna o índice ou componente do tab selecionado;

JToolBar

- Agrupa vários componentes em linha ou coluna
  - geralmente botões com ícones
- Funcionalidade de menus
- Possibilidade de arrastar a barra para qualquer borda ou para fora da janela
  - necessário uso de BorderLayout

JToolBar - métodos

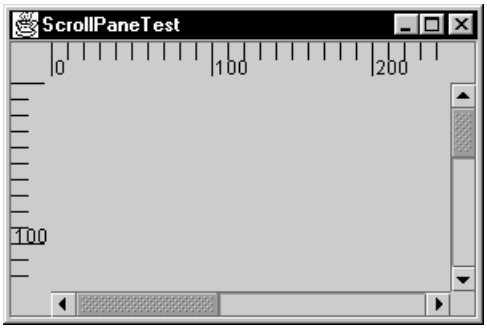
- JToolBar() - cria o objeto;
- add(Action | Component), addSeparator()
  - adiciona um componente
  - argumento Action cria um JButton automaticamente
  - separador é adicionado ao final da barra
- setFloatable(boolean), isFloatable()
  - indica se a barra pode ser arrastada para fora da janela

Outros recursos Swing: JScrollPane

- Barras de scroll exibidas no componente
- Código pode ser mínimo
  - Scroll pane cria as barras de rolagem (scrollBar)
  - redesenha o objeto cliente sempre que as barras são movimentadas
  - barras podem ser exibidas sempre ou quando necessário

Exemplos: ScrollPane

- Eventos do tipo AdjustmentEvent



JScrollPane

- Comportamento das barras
  - uma “ScrollBar policy” para cada barra
  - VERTICAL_SCROLLBAR_AS_NEEDED, HORIZONTAL_SCROLLBAR_AS_NEEDED;
  - VERTICAL_SCROLLBAR_ALWAYS, HORIZONTAL_SCROLLBAR_ALWAYS;
  - VERTICAL_SCROLLBAR_NEVER, HORIZONTAL_SCROLLBAR_NEVER;

JScrollPane: alguns métodos

- JScrollPane([Component], [int, int])
  - parâmetros int: policy vertical e horizontal;
- setViewportView(Component);
- setVerticalScrollBarPolicy(int), getVerticalScrollBarPolicy(), setHorizontalScrollBarPolicy(int), getHorizontalScrollBarPolicy();
- setViewportBorder(Border), getViewportBorder();

Aparência geral da interface

- Look&Feel = Aparência e Comportamento
  - aparência = como componentes são desenhados na tela
  - comportamento = como eles reagem aos eventos.
- Look&Feel:
  - Windows 95 ou NT ou Motif X-Windows
  - Metal multiplataforma do swing
  - padrão Java

Aparência geral da interface

- Para determinar a aparência usa-se:
  - getCrossPlataformLookAndFeelClassName()
    - plataforma Metal
  - getSystemLookAndFeelClassName()
    - plataforma do sistema atual

Aparência geral da interface

- Determinando nova aparência para as janelas:
  - "javax.swing.plaf.metal.MetalLookAndFeel"
  - "com.sun.java.swing.plaf.Windows.WindowsLookAndFeel"
  - "com.sun.java.swing.plaf.motif.MotifookAndFeel"
  - "javax.swing.plaf.mac.MacLookAndFeel"
- UIManager.setLookAndFeel("com.sun.java.swing.plaf.motif.MotifLookAndFeel");

Exemplo: Look & Feel