

riccioni@univali.br

---

---

---

---

---

---

- Componentes de Interface
  - AWT
  - Swing
- Layouts
  - BorderLayout e FlowLayout
  - GridLayout
  - GridBagLayout
- Tratamento de Eventos
  - Listeners e Adapters

---

---

---

---

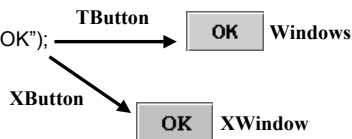
---

---

- Abstract Window Toolkit - desde Java 1.0

```
//...
new Button("OK");
```

TButton → OK Windows



- Componente em Java é criado pela plataforma nativa.
- Desvantagem: cada plataforma pode apresentar o componente de uma maneira diferente.

---

---

---

---

---

---

//...

**Tópicos Especiais em Computação:** Interface Gráfica com o Usuário em Java

---

---

---

---

---

---

### Motif

---

---

---

---

---

---

**Tópicos Especiais em Computação:** Interface Gráfica com o Usuário em Java

---

---

---

---

---

---



Programação em Java:  
Prof. Ricconi  
Teoria e Prática  
UNIVALI São José  
http://www3.univali.br


## Mostrando um Frame

```

public class Sistema {
    public static void main(String args[]) {
        FrExemplo frame = new FrExemplo();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.show();
        // ou frame.setVisible(true);
    }
}

javac *.java

java Sistema
        
```



Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

10

---

---

---

---

---

---

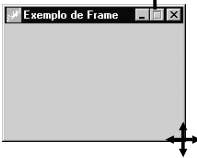
---

---

Programação em Java:  
Prof. Ricconi  
Teoria e Prática  
UNIVALI São José  
http://www3.univali.br

## Principais Métodos de javax.swing.JFrame

- dispose() fecha a janela e libera todos os recursos alocados na sua criação.
- setIconImage(Image imagem) define o ícone do frame. Exemplo:  
 ImageIcon icone = new ImageIcon("icone.gif");  
 frame.setIconImage(icone.getImage());
- setResizable(boolean b)  
 – define se o frame pode ser redimensionado.
- setTitle(String s)



Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

11

---

---

---

---

---

---

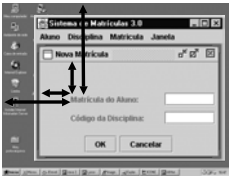
---

---

Programação em Java:  
Prof. Ricconi  
Teoria e Prática  
UNIVALI São José  
http://www3.univali.br

## Principais Métodos de java.awt.Component

- **boolean** isVisible()
- **void** setVisible(**boolean** b)
- **boolean** isEnabled()
- **void** setEnabled(**boolean** b)
- Point getLocation()  
 – coordenadas (x, y) em relação ao container.
- Point getLocationOnScreen()  
 – coordenadas (x, y) da posição na tela.



Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

12

---

---

---

---

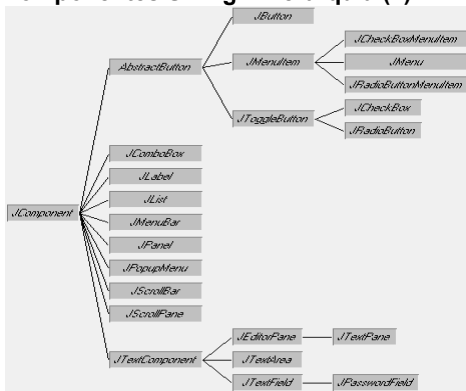
---

---

---

---

## Componentes Swing - Hierarquia (2)



**Tópicos Especiais em Computação:** Interface Gráfica com o Usuário em Java

13

---

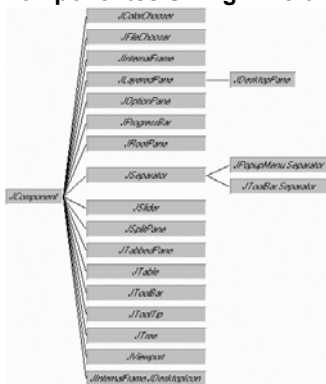
---

---

---

---

---



**Tópicos Especiais em Computação:** Interface Gráfica com o Usuário em Java

14

---

---

---

---

---

---

## JButton

- Construtores:
  - JButton(String rotulo)
  - JButton(Icon icone)
  - JButton(String rotulo, Icon icone)
- Exemplos:

```
JButton jbOK;  
jbOK = new JButton("OK");  
jbOK = new JButton("OK", new ImageIcon("ok.gif"));
```

**Tópicos Especiais em Computação:** Interface Gráfica com o Usuário em Java

15

---

---

---

---

---

---

## Criando os Itens do Menu “Editar”

- ```
public class FrPrincipal extends JFrame {
    //...

    private JMenuItem jmiEditarRecortar =
        new JMenuItem("Recortar");

    private JMenuItem jmiEditarCopiar =
        new JMenuItem("Copiar");

    private JMenuItem jmiEditarCorlar =
        new JMenuItem("Corlar");

    private JMenu jmEditarOpcoes =
        new JMenu("Opções");

    //...
}
```



## Criando o Submenu “Opções”



```
public class FrPrincipal extends JFrame {  
    //  
    private JMenuItem jmiEditarOpcoesSomenteLeitura =  
        new JCheckBoxMenuItem("Somente Leitura");  
    private JMenuItem jmiEditarOpcoesInserir =  
        new JRadioButtonMenuItem("Inserir");  
    private JMenuItem jmiEditarOpcoesSobrescrever =  
        new JRadioButtonMenuItem("Sobrescrever");  
    //  
}
```

## Definindo a Barra de Menus

```
public class FrPrincipal extends JFrame {
    //...
    public FrPrincipal() {
        // adiciona os menus à barra de menus.
        barraMenu.add(jmArquivo);
        barraMenu.add(jmEditar);
        barraMenu.add(jmMatricula);
        barraMenu.add(jmJanela);

        // define a barra de menus deste frame.
        this.setJMenuBar(barraMenu);
    }
}
```

## Definindo os Itens do Menu “Editar”




```
public class FrPrincipal extends JFrame {
    //...
    public FrPrincipal() {
        jmEditor.add(jmiEditorRecortar);
        jmEditor.add(jmiEditorCopiar);
        jmEditor.add(jmiEditorColar);

        jmEditor.addSeparator(); // separador (opcional)

        jmEditor.add(jmEditorOpcoes); // "Opções"
    }
}
```

Programação em Java:  
Prof. Ricconi  
Teoria e Prática  
UNIVALI São José  
<http://www.univali.br>

## Definindo os Itens do Submenu “Opções”



```

public class FrPrincipal extends JFrame {
    //...
    public FrPrincipal() {
        jmEditarOpcoes.add(jmiEditarOpcoesSomenteLeitura);
        jmEditarOpcoes.addSeparator();
        jmEditarOpcoes.add(jmiEditarOpcoesInserir);
        jmEditarOpcoes.add(jmiEditarOpcoesSobrescrever);
    }
}
        
```

Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

22

---

---

---

---

---

---

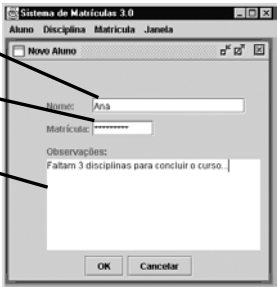
---

---

Programação em Java:  
Prof. Ricconi  
Teoria e Prática  
UNIVALI São José  
<http://www.univali.br>

## Campos e Áreas de Texto

- JTextField
- JPasswordField
- JTextArea  
(texto puro)  
ou  
JTextPane  
(rtf/html)



Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

23

---

---

---

---

---

---

---

---

Programação em Java:  
Prof. Ricconi  
Teoria e Prática  
UNIVALI São José  
<http://www.univali.br>

## JTextField e JPasswordField

- JPasswordField é subclasse de JTextField.
- JTextField:
  - JTextField(int colunas)
  - JTextField(String texto)
  - JTextField(String texto, int colunas)
  - void setColumns(int colunas)
- JPasswordField:
  - JPasswordField(int colunas)
  - JPasswordField(String texto)
  - JPasswordField(String texto, int colunas)

Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

24

---

---

---

---

---

---

---

---

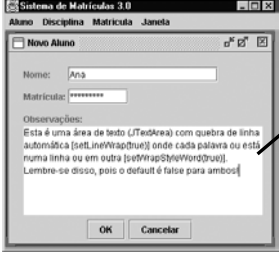


Programação em Java:  
Prof. Ricconi

UNIVALI - São José

## JTextArea

`JTextArea(int linhas, int colunas)`  
`JTextArea(String texto, int linhas, int colunas)`  
`void setLineWrap(boolean wrap) // quebra (true/false)`  
`void setWrapStyleWord(boolean word) // false: quebra por le tra.`



`// true: quebra por palavra.`

Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

25

---

---

---

---

---

---

---

---

---

---

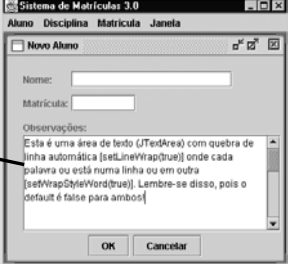
Programação em Java:  
Prof. Ricconi

UNIVALI - São José

## JScrollPane

- Esta classe implementa scroll para qualquer outro componente.
- Exemplo:

**jtaObservacoes** →



`JScrollPane jsp;`  
`jsp = new JScrollPane(jtaObservacoes);`

Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

26

---

---

---

---

---

---

---

---

---

---

Programação em Java:  
Prof. Ricconi

UNIVALI - São José

## JLabel

- Construtores:  
`JLabel(String texto)`  
`JLabel(Icon icone)`  
`JLabel(String texto, int alinhamento)`

|  
**SwingConstants.LEFT,**  
**SwingConstants.CENTER** ou  
**SwingConstants.RIGHT**
- Outros métodos:  
`void setText(String texto)`  
`void setIcon(Icon icone)`

Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

27

---

---

---

---

---

---

---

---

---

---

## JCheckBox

- Construtores:  
JCheckBox(String rotulo)  
JCheckBox(String rotulo, **boolean** estado)  
JCheckBox(String rotulo, Icon icone)
- Outros métodos:  
**boolean** isSelected()  
**void** setSelected(**boolean** estado)

---

---

---

---

---

---

## JRadioButton

```
JRadioButton jrbMatutino = new JRadioButton("Matutino", true);
JRadioButton jrbVespertino = new JRadioButton("Vespertino");
JRadioButton jrbNoturno = new JRadioButton("Noturno");
```

- Estes componentes devem ser agrupados para definir uma escolha exclusiva.
- ButtonGroup implementa grupo de botões, onde apenas um está selecionado.



---

---

---

---

---

---

## ButtonGroup

- Ao contrário de algumas interfaces, um objeto da classe `ButtonGroup` *não* é um componente de interface, ou seja, não tem aparência e não é adicionado à interface gráfica.
- `ButtonGroup` é apenas uma classe que implementa a lógica de grupo de botões.
- Exemplo:

```
ButtonGroup grupo = new ButtonGroup();
grupo.add(jrbMatutino);
grupo.add(jrbVespertino);
grupo.add(jrbNoturno);
```

---

---

---

---

---

---

## JComboBox

- Construtores:  
JComboBox()  
JComboBox(Object[] itens)  
JComboBox(Vector itens)
- Outros métodos:  
**void** setEditable(**boolean** b)  
**void** addItem(Object item)  
**void** insertItemAt(Object item, **int** indice)  
**void** removeItem(Object item)  
**void** removeItemAt(**int** indice)  
**void** removeAllItems()  
Object getSelectedItem()

---

---

---

---

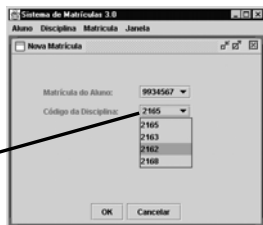
---

---

## Utilizando JComboBox

- Exemplo: selecionando aluno e disciplina no sistema de matrículas:

jcbCodDisciplina



- Para as disciplinas:
- ```
Hashtable cadDisciplinas = aplicacao.retorneCadDisciplinas();
Set codigos = cadDisciplinas.keySet();
jcbCodDisciplina = new JComboBox(new Vector(codigos));
```

---

---

---

---

---

---

## JOptionPane.showMessageDialog

```
static void showMessageDialog(Component pai,
                                Object mensagem,
                                String titulo,
                                opcionais — int tipo,
                                           Icon icone)
```

- Exemplo (em FrNovaDisciplina):  
`JOptionPane.showMessageDialog(this, "Código inválido!", "Erro", JOptionPane.ERROR_MESSAGE);`



---

---

---

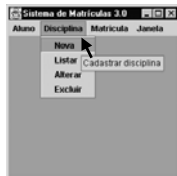
---

---

---

## Tooltips

- A classe `JComponent` define um método:  
`void setToolTipText(String texto)`
- Assim, para qualquer componente de classes derivadas, podemos definir uma dica para o usuário quando o mouse passa sobre o componente:



## Layouts

- Como definir uma interface gráfica?
- Apenas criar componentes não é suficiente.
- Precisamos definir a posição (lay out) dos componentes no interior dos containers.
- Mesmo utilizando ferramentas de design, algumas vezes é preciso entender como funcionam para obter a interface desejada.

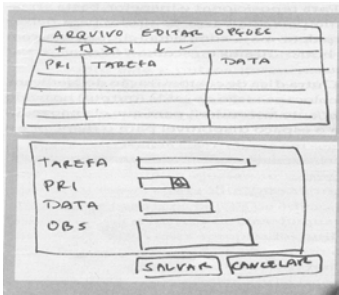


## Layouts

- Antes de iniciar a construção de uma interface visual, é sempre bom desenhar um esboço contendo os principais componentes e sua disposição nas janelas. Papel e caneta ou um programa de desenho são geralmente melhores para isso do que o seu IDE favorito, pois permitem que as idéias fluam sem serem "viciadas" pela estrutura e componentes padrões do IDE.

## Layouts

Exemplo:



Programação em Java:  
Teoria e Prática

Prof. Ricconi

UNIVALI São José

http://www.univali.br

## Layouts

### Arquitetura da aplicação

- É um padrão em desenvolvimento orientado a objetos utilizar a arquitetura MVC como base de uma aplicação interativa. Dessa forma, o código da nossa aplicação será organizado em classes de modelo, visão e controlador, utilizando para tal uma estrutura de pacotes.
- Primeira etapa do desenvolvimento da Aplicação, que é a prototipação da interface com o usuário, utilizando os recursos de desenho de interfaces Swing. Vamos limitar o código ao mínimo que possibilite a navegação e exibição de informações; assim poderemos validar a usabilidade e a adequação da interface às necessidades da aplicação.

Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

40

---

---

---

---

---

---

---

---

Programação em Java:  
Teoria e Prática

Prof. Ricconi

UNIVALI São José

http://www.univali.br

## Layouts

### Alguns conceitos de Swing

Para os que estão começando com a programação visual em Java, são apresentados aqui alguns conceitos do Swing, que são usados ou citados ao longo do texto.

- Componentes, em um sentido amplo, são objetos visuais (ex.: JCheckBox, JButton, JSeparator), ou objeto não visuais (como GridBagLayout) que podem interagir com objetos visuais por meio dos padrões JavaBeans.
- O termo *"formulário"* é utilizado de forma genérica para referendar janelas, diálogos e painéis

Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

41

---

---

---

---

---

---

---

---

Programação em Java:  
Teoria e Prática

Prof. Ricconi

UNIVALI São José

http://www.univali.br

## Layouts

### Alguns conceitos de Swing

- Um container é qualquer objeto que possa conter outros objetos visuais.
- Todo container tem um gerenciador de layout que organiza o posicionamento e dimensionamento dos componentes dentro do container. Exemplos: JPanel e JDialog.
- Componentes definem um tamanho mínimo, que é a menor dimensão (altura, largura) capaz de exibir todo o seu conteúdo (texto, ícone ou ambos). Em alguns casos, este tamanho é derivado de outras propriedades do componente; por exemplo, em um JTextArea podem ser especificadas colunas e linhas da sua área de texto visível.

Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

42

---

---

---

---

---

---

---

---

## Gerenciadores de layout

O motivo é que nestes ambientes se costuma posicionar os componentes de modo fixo (em pixels) nos formulários, enquanto que no Swing o posicionamento é determinado pelos **gerenciadores de layout**.

Por isso foi preparado este quadro, que relaciona os usos mais comuns dos principais gerenciadores de layout do J2SE, além das facilidades oferecidas pelo IDE para a customização visual de componentes com esses gerenciadores.

---

---

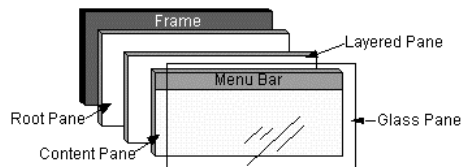
---

---

---

---

- Root e Layered pane: servem para organizar o content pane e a barra de menus.
- Content pane: onde adicionamos componentes



```
Container content = getContentPane();
content.add(...);
```

---

---

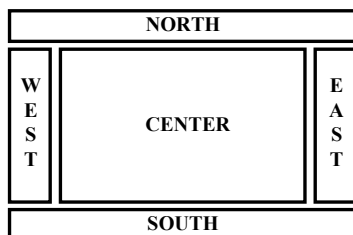
---

---

---

---

- Por default, o content pane de todo JFrame é gerenciado por um *BorderLayout*.
- Neste layout, há apenas 5 posições onde podemos inserir componentes:



---

---

---

---

---

---

Programação em Java:  
Prof. Ricconi

Teoria e Prática

UNIVALI São José

http://www.univali.br

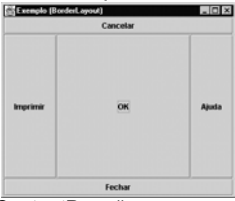
## Exemplo (BorderLayout)

```

public class FrExemplo extends JFrame {
    // declara os botões
    private JButton jbOK;
    private JButton jbCancelar;
    //...continua.

    public FrExemplo() {
        super("Exemplo ...");
        Container content = this.getContentPane();
        jbOK = new JButton("OK");
        content.add(jbOK, BorderLayout.CENTER);
        jbCancelar = new JButton("Cancelar");
        content.add(jbCancelar, BorderLayout.NORTH);
        //...continua.
    }

```



Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

46

---

---

---

---

---

---

---

---

---

---

Programação em Java:  
Prof. Ricconi

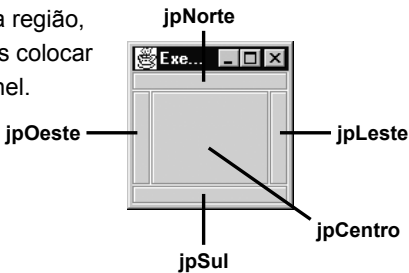
Teoria e Prática

UNIVALI São José

http://www.univali.br

## Containers Aninhados

- Como podemos criar uma interface com apenas 5 posições possíveis?
- O segredo está em aninhar containers.
- Em cada região, podemos colocar um JPanel.



Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

47

---

---

---

---

---

---

---

---

---

---

Programação em Java:  
Prof. Ricconi


Teoria e Prática

UNIVALI São José

http://www.univali.br

## FlowLayout

- O layout default de todo JPanel é FlowLayout.
- Neste layout, os componentes são dispostos lado a lado na sequência em que são adicionados.
- A posição dos componentes ajusta-se quando o frame é redimensionado:



Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

48

---

---

---

---

---

---

---


---

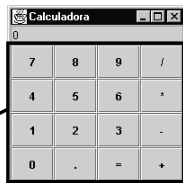
---


---



## GridLayout

- Em alguns casos, é conveniente definir o layout utilizando-se uma tabela.
  - Todas as células da tabela têm as mesmas largura e altura.
- 



- **Construtor:**  
`GridLayout(lin, col)`
  - **Exemplo: GridLayout(4, 4)**
    - Obs: o número de linhas tem mais prioridade do que o número de colunas. Defina zero linhas caso o número de colunas seja mais importante.
- 
- |   |   |   |   |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 4 | 5 | 6 | * |
| 1 | 2 | 3 | . |
| 0 | . | = | + |

---

---

---

---

---

---

## GridBagLayout

- Quando a interface possui um layout mais complexo, precisamos utilizar GridBagLayout.
- Definido por células e parâmetros para cada componente.



região

---

---

---

---

---

---

## GridBagLayout

## Entenda o GridBagConstraints

- As propriedades que determinam a posição e dimensões de um componente dentro de um `GridBagLayout` são reunidas em um objeto chamado `GridBagConstraints`.
- Cada componente adicionado a um container cujo gerenciador de layout seja um `GridBagLayout` possui seu próprio objeto `GridBagConstraints`.

---

---

---

---

---

---

## GridBagLayout

## Entenda o GridBagConstraints

- Note que identificamos as propriedades da classe `GridBagConstraints` do modo como aparecem no customizador do `GridBagLayout`.
- Estes nomes não são os mesmos que serão encontrados na documentação javadoc da classe, onde são resumidos e seguem a sintaxe de Java.

Por exemplo, a propriedade "Grid Width" do customizador é na verdade `gridWidth`; e "Internal Padding X" é `ipadx`.

---

---

---

---

---

---

## GridBagLayout

## Entenda o GridBagConstraints

- Grid X e Grid Y - indicam a posição do componente dentro da tabela ou grade utilizada pelo GridBagLayout para posicionar os componentes.
- Grid Width e Grid Height - determinam a quantidade de células da tabela que serão ocupadas pelo componente, respectivamente na largura e altura.
- Fill - indica se o componente irá ocupar seu tamanho mínimo, deixando vazio o restante das células ocupadas; ou se ele será expandido para ocupar todo o espaço alocado para a célula. A expansão pode ser apenas na vertical (valor Vertical), apenas na horizontal (Horizontal), ou em ambos os sentidos (Both).

---

---

---

---

---

---

## GridBagLayout

## Entenda o GridBagConstraints

- Internal Padding X e Internal Padding Y - indicam espaço acrescentado ao próprio componente, aumentando o seu tamanho mínimo, em vez de acrescentado à célula que o contém.
- Anchor - especifica o alinhamento do componente em relação à suas células, caso ele não preencha toda a área alocada a elas. Seus valores possíveis são baseados nos pontos cardeais, como North ou SouthEast.

---

---

---

---

---

---

Programação em Java:  
Teoria e Prática

Prof. Riccioni

UNIVALI São José

http://www3.univali.br

## GridBagLayout

### Entenda o GridBagConstraints

- Weight X e Weight Y - valores maiores do que zero indicam que a célula será expandida para além do seu tamanho mínimo, ocupando o espaço na largura ou altura que sobrar no container. Estas propriedades costumam ser utilizadas apenas quando o container pode ser redimensionado pelo usuário.
- Insets - indicam espaçamentos a serem inseridos em volta da célula, afastando o componente de suas bordas. Dois componentes em células adjacentes e com espaçamentos zerados serão exibidos "grudados" um no outro.

Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

55

---

---

---

---

---

---

---

---

Programação em Java:  
Teoria e Prática

Prof. Riccioni

UNIVALI São José

http://www3.univali.br

## GridBagLayout (Parâmetros)

- gridx e gridy: coluna/linha (definem a célula).
- gridwidth e gridheight: quantas colunas/linhas.
- fill: NONE, HORIZONTAL, VERTICAL, BOTH.
- anchor: NORTH, WEST, CENTER, ...
- ipadx, ipady: padding interno do componente
- insets (top, left, bottom, right): padding externo (entre o componente e a fronteira da célula).
- weightx e weighty: em que proporção o componente será redimensionado com o frame. Default igual a zero (o tamanho não muda).

Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

56

---

---

---

---

---

---

---

---

Programação em Java:  
Teoria e Prática

Prof. Riccioni

UNIVALI São José

http://www3.univali.br

## Receita para GridBagLayout

1. Faça um rascunho da interface no papel.
2. Considere o menor componente para definir as dimensões de cada célula.
3. Numere cada coluna e cada linha 0, 1, 2, 3, ... (permite definir gridx, gridy, gridwidth e gridheight para cada componente)
4. Verifique se cada componente deve ocupar toda a região (fill - HORIZONTAL, VERTICAL ou BOTH) e qual parte da célula (anchor - NORTH, WEST, ...)
5. Geralmente, utilize weightx = 0 e weighty = 0, ou seja os componentes não são redimensionados com o frame. Defina weightx = 100 e weighty = 100, caso deseje redimensionamento automático.

Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

57

---

---

---

---

---

---

---

---

```
public class FrNovoAluno
```

**Tópicos Especiais em Computação:** Interface Gráfica com o Usuário em Java

---

---

---

---

---

---

**Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java**



---

---

---

---

---

---

**Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java**



---

---

---

---

---

---

## Sem Gerenciador de Layout

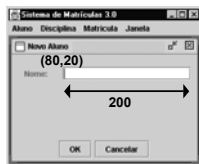
- Quando não se dispõe de uma ferramenta de design é difícil utilizar GridBagLayout.
- Para criar um protótipo da interface gráfica rapidamente, é possível defini-la sem gerenciador de layout:

```
jpPrincipal.setLayout(null);
```

```

JTextField jtfNome = new JTextField();
jpPrincipal.add(jtfNome);
jtfNome.setBounds(80, 20, 200, 20);

```



- **Obs:** quando não se utiliza gerenciador de layout, não se pode garantir portabilidade da interface.

---

---

---

---

---

---

## Borders

- São úteis para visualizar grupos de componentes.
- Estilos mais usados:



---

---

---

---

---

---

### Borders (Exemplos)

```
Border etched = BorderFactory.createEtchedBorder();
Border titled = BorderFactory.createTitledBorder(etched,
```

```
jpTurno.setBorder(titled);
```



```
Border borda = BorderFactory.createEtchedBorder();
jpBotoes.setBorder(borda);
```

---

---

---

---

---

---

- Em outras linguagens, os eventos são tratados por procedimentos determinados:

- Em Java, objetos são criados para representar:
  - a fonte do evento (botão, menu, tecla...)
  - o evento (ação, clique, ...)
  - quem trata o evento (listener ou adapter)

---

---

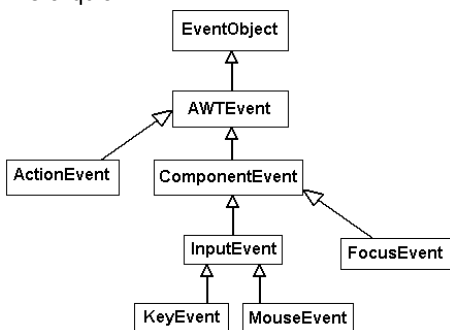
---

---

---

---

- Hierarquia



---

---

---

---

---

---

1. Identificar a qual classe pertence este evento.  
 ActionEvent, MouseEvent, KeyEvent, FocusEvent, ...

2. Criar um objeto de uma classe que implemente a interface correspondente. ActionListener, MouseListener, KeyListener, ...

3. Registrar este objeto como tratador de eventos.

```
componenteFonte.addXXXXXXListener(( ));
```

---

---

---

---

---

---

Programação em Java:  
Teoria e Prática  
UNIVALI - São José  
Prof. Ricloni  
http://www.sj.univali.br

Eventos e Listeners

- Botão ou Menu
  - ActionEvent
  - **interface** ActionListener
- Teclado
  - KeyEvent
  - **interface** KeyListener
- Mouse
  - MouseEvent
  - **interface** MouseListener

Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java67

---

---

---

---

---

---

---

---

Programação em Java:  
Teoria e Prática  
UNIVALI - São José  
Prof. Ricloni  
http://www.sj.univali.br

Interfaces mais Utilizadas

- ActionListener (botões ou menus)

```
public interface ActionListener extends EventListener {  
    public void actionPerformed(ActionEvent evento);  
}
```

Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java68

---

---

---

---

---

---

---

---

Programação em Java:  
Teoria e Prática  
UNIVALI - São José  
Prof. Ricloni  
http://www.sj.univali.br

Interfaces mais Utilizadas

- KeyListener (teclado)

```
public interface KeyListener extends EventListener {  
    public void keyTyped(KeyEvent evento);  
    public void keyPressed(KeyEvent evento);  
    public void keyReleased(KeyEvent evento);  
}
```

Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java69

---

---

---

---

---

---

---

---

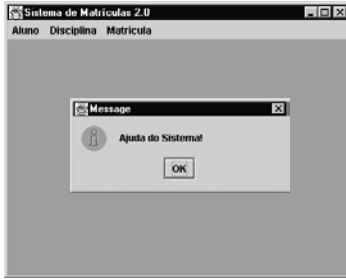






## Outro Exemplo

- Tratamento do evento de pressionar a tecla "F1":



frPrincipal

---

---

---

---

---

---

## KeyListener

```
public interface KeyListener extends EventListener {  
    public void keyTyped(KeyEvent evento);  
    public void keyPressed(KeyEvent evento);  
    public void keyReleased(KeyEvent evento);  
}
```

- Apenas o método `keyPressed` será utilizado para mostrar a mensagem de ajuda do sistema.
- Mas sendo uma interface, devemos implementar todos os métodos declarados, caso contrário nossa classe será abstrata.

---

---

---

---

---

---

## Classe Abstrata

```
public class FrPrincipal extends JFrame {
    public FrPrincipal() {
        //...
        this.addKeyListener(new KeyListener() {
            public void keyPressed(KeyEvent e) {
                if (e.getKeyCode() == KeyEvent.VK_F1) {
                    ...showMessageDialog(..., "Ajuda do Sistema!");
                }
            }
        });
    }
}
```

- **Error: class ... should be declared abstract; it does not define method keyReleased(...) in interface java.awt.event.KeyListener.**

---

---

---

---

---

---

### Solução 1 - KeyListener

```
public class FrPrincipal extends JFrame {  
    public FrPrincipal() {  
        this.addKeyListener(new KeyListener() {  
            public void keyPressed(KeyEvent e) {  
                if(e.getKeyCode() == KeyEvent.VK_F1) {  
                    ...showMessageDialog(..., "Ajuda do Sistema!");  
                }  
            }  
            public void keyReleased(KeyEvent e) {  
                // nada a fazer! (mas obrigatório)  
            }  
            public void keyTyped(KeyEvent e) {  
                // nada a fazer! (mas obrigatório)  
            }  
        });  
    }  
}
```

---

---

---

---

---

---

## Adapters x Listeners

- Um Adapter é uma classe, enquanto um Listener é uma interface.
- Uma classe Adapter já implementa todos os métodos da interface Listener correspondente.
- A implementação é simplesmente vazia (nenhuma linha de código).
- Assim, não precisamos implementar métodos que não vamos usar. Em vez disso, utilizamos o Adapter correspondente.
- A API do Java define os adapters convenientes para substituir o uso de listeners.

---

---

---

---

---

---

## Exemplo: KeyAdapter x KeyListener

```
abstract class KeyAdapter implements KeyListener {  
    public void keyPressed(KeyEvent e) {  
        // vazio.  
    }  
    public void keyReleased(KeyEvent e) {  
        // vazio.  
    }  
    public void keyTyped(KeyEvent e) {  
        // vazio.  
    }  
}
```

- Esta classe existe por conveniência para criar objetos listeners.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



## Método fechaFrame()

```
public class FrNovoAluno extends JInternalFrame {  
    //...
```

```
public void fecharFrame() {  
    // Remove referência para o listenerFechar.  
    // Importante para evitar vazamento de memória.  
    frPrincipal.removeListenerFechar(listenerFechar);  
}
```

```
// Fecha este frame.  
this.dispose();
```

- Outros frames podem implementar ainda mais: fechar arquivos, desconectar, ...

---

---

---

---

---

---

## Validação

- Podemos criar um objeto que verifique se a entrada do usuário é válida ou não.
- `javax.swing.JComponent` possui o método:  
**`void`** `setInputVerifier(InputVerifier verificador)`
- A classe abstrata `InputVerifier` define um método que é chamado quando um componente perde o *focus*, o que permite a validação do texto digitado pelo usuário.  
**`public abstract class`** `InputVerifier` {  
    **`public abstract boolean`** `verify`(`JComponent` input);  
}

---

---

---

---

---

---

### Exemplo de Validação

- No sistema de matrículas, o código de toda disciplina deve ser um número inteiro.
- Toda vez que o usuário digitar algum texto e o campo perder o focus, o conteúdo será verificado. Caso seja válido (inteiro), a mudança de focus será permitida. Caso contrário, o focus permanecerá no componente de texto.



---

---

---

---

---

---



## Criando um Frame Interno Genericamente

```
public InternalFrame mostreFrame(String nomeClasse, Object
    param[]) {
    InternalFrame frame = null;
    try {
        Class classe = Class.forName(nomeClasse);
        Class tipos[] = new Class[param.length];
        for(int i=0; i<tipos.length; i++) {
            tipos[i] = param[i].getClass();
        }
        Constructor construtor = classe.getDeclaredConstructor(tipos);
        frame = (InternalFrame)construtor.newInstance(param);
        this.mostreFrame(frame); // método sobrecarregado
    } catch (Exception excecao) {
        excecao.printStackTrace();
    }
    return frame;
}
```

**Não funciona para tipos primitivos**

```
public void mostreFrame(JInternalFrame frame) {
    try {
        // centraliza o frame
        Dimension thisdim = this.size();
        Dimension frdim = frame.size();
        int x = (int)((thisdim.getWidth() - frdim.getWidth()) / 2);
        int y = (int)((thisdim.getHeight() - frdim.getHeight()) / 2);
        frame.setBounds(x, y, frdim.width, frdim.height);
        // mostra o frame
        frame.setVisible(true);
        desktop.add(frame);
        frame.setSelected(true);
    } catch (Exception excecao) {
        excecao.printStackTrace();
    }
}
```



Programação em Java:  
Prof. Ricdoni

UNIVALI São José

http://www.univali.br

## Definindo o Focus Inicial (Ex. FrAluno)

```

...
this.addInternalFrameListener(new
    InternalFrameAdapter() {
        public void
        internalFrameActivated(InternalFrameEvent ev) {
            this._internalFrameActivated(ev);
        }
    }
);
...
public void
this._internalFrameActivated(InternalFrameEvent e) {
    jtMatricula.requestFocus();
}
...

```

Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

97

---

---

---

---

---

---

---

---

Programação em Java:  
Prof. Ricdoni

UNIVALI São José

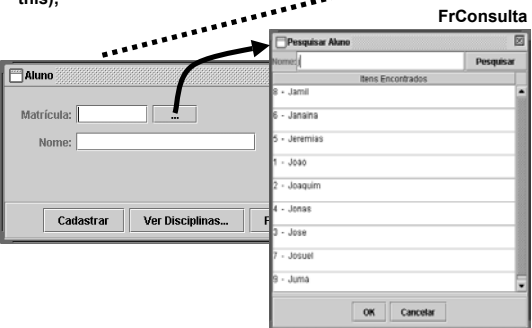
http://www.univali.br

## Componente de Pesquisa Genérico (FrConsulta)

```

new FrConsulta("Aluno", "matricula", "nome",
this);

```



**FrConsulta**

Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

98

---

---

---

---

---

---

---

---

Programação em Java:  
Prof. Ricdoni

UNIVALI São José

http://www.univali.br

## Construtor de FrConsulta

```

FrConsulta(String entidade,
            String chave, String descricao,
            Atualizavel atualizavel)

```

- Para manter a generalidade, utilizamos uma interface:

```

interface Atualizavel {
    public void atualize(String valorChave,
                        String valorDescricao);
}

```

Tópicos Especiais em Computação: Interface Gráfica com o Usuário em Java

99

---

---

---

---

---

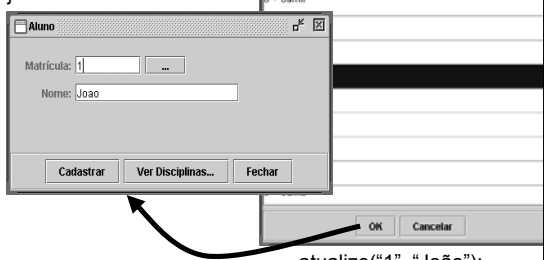
---

---

---

## Atualização

```
public void atualize(String chave, String descricao) {
    jtfMatricula.setText(chave);
    jtfNome.setText(descricao);
}
```



```
atualize("1", "João");
```

## Tratamento de Exceções

- Quando ocorre um erro na execução de um programa, podemos tratá-lo.
- O tratamento permite continuar a execução.
- Estrutura geral:

```
try {
    // ... trecho de código onde pode ocorrer erro...
} catch (Exception excecao) {
    // tratamento da exceção
    excecao.printStackTrace(); // mostra a pilha! (opcional)
} finally {
    // o que é feito em ambos os casos:
    // com ou sem exceção.
}
```

## Exemplo de Tratamento de Exceções

- Divisão por Zero:

```
try {
    int a, b, c;
    a = 10;
    b = 0;
    c = a / b;
} catch (Exception excecao) {
    excecao.printStackTrace();
}
System.out.println("continua a execucao...");
```

```
java.lang.ArithmeticException: / by zero
    at FrPrincipal.<init>(FrPrincipal.java:39)
    at SisMat3.<init>(SisMat3.java:9)
    at SisMat3.main(SisMat3.java:21)
continua a execução...
```

## Menus Flutuantes

**Um menu flutuante (pop-up) é um menu que não está associado a uma barra de menu mas que pode aparecer em qualquer lugar.**



Ele é criado da mesma forma como se cria um menu comum, exceto que um menu flutuante não tem título.

```
JPopupMenu popup = new JPopupMenu();
```

**Os itens de menu são adicionados de maneira usual:**

```
JMenuItem item = new JMenuItem("Cut");
```

```
item.addActionListener(this);
```

```
popup.add(item);
```

---

---

---

---

---

---

## Menus Flutuantes

Diferente da barra de menu que é sempre exibida no topo do *frame*, pode-se exibir um menu flutuante explicitamente usando o método *show*, com a seguinte sintaxe:

```
popup.show(painel, x, y);
```

Onde,  $x$  e  $y$  são as coordenadas (no espaço de coordenadas de painel) do canto superior esquerdo do menu flutuante.

E painel é o componente sobre o qual o menu flutuante vai aparecer.

---

---

---

---

---

---

## Menus Flutuantes

Normalmente para abrir um menu flutuante é quando o usuário clica em um botão particular do mouse(costuma-se ser o direito) acionando-se o gatilho de menu flutuante:

**1.Instale um ouvinte de mouse;**

**2. Adicione código como este a seguir ao manipulador de eventos do mouse:**

```
public void mouseReleased(MouseEvent evt)
```

```
{ if (evt.isPopupTrigger())
```

```
popup.show(evt.getComponent(), evt.getX(), evt.getY());
```

}

**Esse código vai exibir o menu flutuante na posição em que o mouse estiver quando for clicado pelo usuário.**

---

---

---

---

---

---

## Mnemônicos e Teclas de Atalho de Teclado

**Bastante conveniente para usuário experiente selecionar item de menu através de mnemônico de teclado. Através do construtor do item de menu:**



```
JMenuItem indexItem = new JMenuItem("Index", 'I');
```

*On*

```
JMenuItem indexItem = new JMenuItem("About", 'A');
```

**Quando o menu é exibido basta o usuário pressionar a tecla sublinhada.**

---

---

---

---

---

---

## Mnemônicos e Teclas de Atalho de Teclado

**Infelizmente, somente pode-se fornecer uma letra mnemônica ao construtor de um item de menu, e não ao construtor do menu.**



Em vez disso, para associar um mnemônico a um menu, é necessário usar o método *setMnemonic*:

```
JMenu helpMenu = new JMenu("Help");
```

```
helpMenu.setMnemonic('H');
```

Para selecionar um menu de mais alto nível da barra de menus, usa-se a tecla **ALT**+ a letra mnemônica.

**Ex:**  $ALT + H$

---

---

---

---

---

---

## Mnemônicos e Teclas de Atalho de Teclado

**Os Mnemônicos de teclado permite selecionar um submenu ou item de menu no menu atualmente aberto.**



Teclas de atalhos são atalhos via teclado CTRL + O e CTRL + S, para Open (Abrir) e Save (Salvar) itens no menu File (Arquivo). Usa-se o método *setAccelerator* para associar uma tecla aceleradora a um item de menu. Esse método recebe um objeto do tipo *Keystroke*. EX: CTRL + O

```
JMenuItem openItem = new JMenuItem("Open");
openItem.setAccelerator (KeyStroke.getKeyStroke
(KeyEvent.VK_O,InputEvent.CTRL_MASK));
```

---

---

---

---

---

---

## Mnemônicos e Teclas de Atalho de Teclado

Quando o usuário pressiona a combinação da tecla de atalho, essa ação seleciona automaticamente a opção do menu e dispara um evento de ação, como se o usuário tivesse selecionado a opção do menu manualmente



**As teclas aceleradoras somente podem ser associadas a itens de menu e não a menus.**

**As teclas de atalho agilizam a seleção e o disparo dos eventos de ação. Menus não tem nenhum evento de ação a eles associados.**

---

---

---

---

---

---

## Mnemônicos e Teclas de Atalho de Teclado

**Inclusão de ícones nos itens de menu. EX:**



```
JMenuItem cutItem = new JMenuItem("Cut", new ImageIcon(
"C:\\MenuTest\\cut.gif"));
```

```
JMenuItem pasteItem = new JMenuItem("Paste", new  
ImageIcon("C:\\MenuTest\\paste.gif"));
```

---

---

---

---

---

---

## Mnemônicos e Teclas de Atalho de Teclado

**Como ativar e desativar um item de menu, usa-se o método set Enabled:**

```
saveItem.setEnabled(false);
```



**Existem 2 estratégias para se ativar ou não itens de menu:**

1. Cada vez que as circunstâncias mudam, pode-se usar `setEnabled` nos itens de menu relevantes.
2. É não se preocupar com o estado dos itens de menu no restante do programa e especificá-los exatamente antes de exibir o menu. Para fazer isso, é necessário registrar um `Listener` (ouvinte) no evento “menu selecionado” (`menuSelected`).

---

---

---

---

---

---

## Mnemônicos e Teclas de Atalho de Teclado

O pacote *javax.swing.event* define uma interface **MenuListener** com três métodos:

*void menuSelected(MenuEvent evt)*

*void menuDeselected(MenuEvent evt)*

*void menuCanceled(MenuEvent evt)*

---

---

---

---

---

---

---