

## Aula 17

# Componentes GUI - Parte I

## Introdução

- Uma **interface gráfica com o usuário** (*Graphical User Interface - GUI*) apresenta um mecanismo amigável ao usuário para interagir com um aplicativo
- Uma GUI dá ao aplicativo uma “aparência” e “comportamento” distintos, sendo mais intuitivos e fáceis de utilizar em comparação com programas baseados em linhas de comando
- Os usuário aprendem mais rapidamente e utilizam mais produtivamente programas baseados em GUI

## Introdução

- As GUIs são construídas a partir de **componentes GUI**
- Um componente GUI é um objeto com o qual o usuário interage via mouse, teclado ou outro formato de entrada (ex: voz)

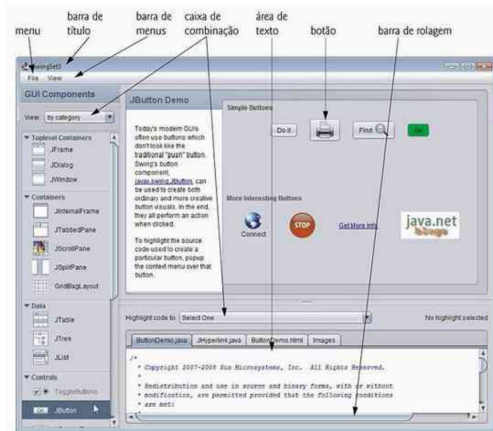
## Introdução

- Muitos IDEs fornecem ferramentas de projeto GUI nas quais é possível especificar o tamanho e a localização exata de um componente de maneira visual utilizando o mouse
- O IDE gera o código GUI automaticamente
- Embora isso simplifique a criação dos GUIs, cada IDE tem diferentes capacidades e geram código distintos
- Além disso, **É IMPORTANTE SABER COMO ESCREVER A MÃO OS CÓDIGOS DOS COMPONENTES GUIs**

## Introdução

Entrada/Saída baseada em GUI simples com JOptionPane  
Visão geral de componentes Swing  
Exibição de Textos e Imagens em uma Janela  
Campos de Texto e Tratamento de Eventos  
Tipos Comuns de Eventos GUI e Interfaces Ouvintes  
Botões  
Material Complementar

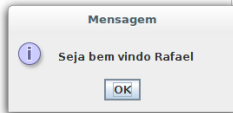
# Demonstração de Componentes Swing GUI do Java



## Entrada/Saída baseada em GUI simples com JOptionPane

- Já utilizamos a classe JOptionPane para exibir ou inserir valores digitados pelo usuário mesmo em programas baseados em linhas de comando via **caixas de diálogo**
- Esses diálogos são exibidos invocando métodos static da classe JOptionPane


```
12 public class Teste {  
13  
14     public static void main(String[] args) {  
15  
16         String nome = JOptionPane.showInputDialog(null, "Digite Seu Nome");  
17  
18         JOptionPane.showMessageDialog(null, "Seja bem vindo " + nome);  
19  
20     }  
21  
22 }  
23
```



## Entrada/Saída baseada em GUI simples com JOptionPane

- O método `showMessageDialog` possui algumas versões sobrecarregadas
- Em uma delas podemos informar um título para uma caixa de diálogo e um “tipo” de diálogo

```
12 public class Teste {  
13  
14     public static void main(String[] args) {  
15  
16         JOptionPane.showMessageDialog(null, "Seja bem vindo ao curso de java");  
17  
18         JOptionPane.showMessageDialog(null, "Seja bem vindo ao curso de java", "POO", JOptionPane.ERROR_MESSAGE);  
19  
20     }  
21  
22 }  
23
```



- O mesmo vale para o método `showInputDialog`

## Constantes de Diálogo

Tipo de diálogo de mensagem	Ícone	Descrição
ERROR_MESSAGE		Indica um erro ao usuário.
INFORMATION_MESSAGE		Indica uma mensagem informativa ao usuário.
WARNING_MESSAGE		Alerta o usuário de um potencial problema.
QUESTION_MESSAGE		Propõe uma questão ao usuário. Normalmente, esse diálogo exige uma resposta, como clicar em um botão Yes ou No.
PLAIN_MESSAGE	Nenhum ícone	Um diálogo que contém uma mensagem, mas nenhum ícone.



## Visão geral de componentes Swing

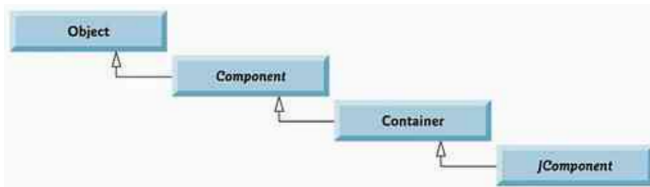
- Embora seja possível realizar entrada e saída utilizando os diálogos JOptionPane, a maioria dos aplicativos GUI exige interfaces com o usuário mais elaboradas e personalizadas
- Para isso, vamos utilizar os **componentes GUI Swing** do pacote javax

# Visão geral de componentes Swing

Componente	Descrição	Exemplo
<b>JLabel</b>	Exibe um texto não editável ou ícones	Nome:
<b>TextField</b>	Permite ao usuário inserir dados do teclado. Também pode ser utilizado para exibir texto editável ou não editável	
<b>Button</b>	Desencadeia um evento quando o usuário clicar nele com o mouse	
<b>CheckBox</b>	Especifica se uma opção está ou não selecionada	
<b>RadioButton</b>	Parecido com o JCheckBox, porém, é usualmente usado para que o usuário possa selecionar uma dentre várias opções	
<b>ComboBox</b>	Fornece uma lista de itens que o usuário pode selecionar. A característica desse componente é que a lista fica contraída e permite a seleção de um único item	
<b>List</b>	Semelhante ao JComboBox, porém, a lista de opções está expandida o tempo todo e permite a seleção de múltiplos itens.	
<b>Panel</b>	Fornecer uma área em que os componentes podem ser colocados organizados. Também pode ser utilizado como uma área de desenho para imagens gráficas.	

## A classe Component

- A classe `Component` é uma subclasse de `Object` que declara muitos dos atributos e comportamentos comuns aos componentes GUI
- A maioria dos componentes GUI estende a classe `Component` direta ou indiretamente



## A classe Container

- A classe Container é uma subclasse de Component
- Components são anexados a Containers, como janelas, de modo que os Components possam ser organizado e exibidos na tela
- Qualquer objeto Container (incluindo objetos que herdam da classe) podem ser usados para organizar outros Components em uma GUI
- Como Container é um Component e pode-se colocar um Container em outros Containers para ajudar a organizar uma GUI

## A classe JComponent

- A classe JComponent é uma subclasse de Container
- JComponent é a superclasse de todos os componentes leves Swing e declara seus atributos e comportamento comuns

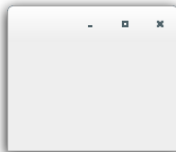
## Exibição de Textos e Imagens em uma Janela

- **A maioria das janelas que vamos criar para acoplar os components Swing GUI são instâncias da classe JFrame ou de uma subclasse de JFrame**
- JFrame fornece atributos e comportamentos básicos de uma janela
  - Barra de títulos
  - Botões para minimizar, maximizar e fechar a janela
  - Redimensionamento da janela
- **Portanto, o básico para se criar uma janela é criar uma subclasse de JFrame**

## Criando uma Janela

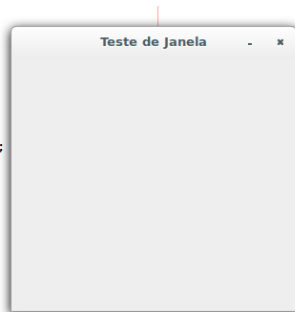
```
11 public class Teste {  
12  
13     public static void main(String[] args){  
14         new Janela();  
15     }  
16  
17 }
```

```
11  
12 public class Janela extends JFrame{  
13  
14     Janela(){  
15         this.setVisible(true);  
16     }  
17  
18 }  
19
```



## Definindo Algumas Propriedades da Janela

```
10  
11  
12 public class Janela extends JFrame{  
13  
14     Janela(){  
15         this.setSize(300, 300);  
16         this.setTitle("Teste de Janela");  
17         this.setResizable(false);  
18         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
19         this.setVisible(true);  
20     }  
21 }  
22  
23
```

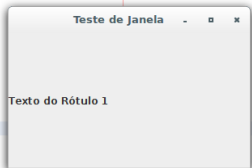




## Adicionando um Componente em uma Janela

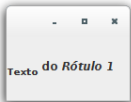
- Para adicionar um componente em uma janela utiliza-se o método `add(Component comp)`
- Portanto, para adicionar um rótulo com um texto basta criar um rótulo, definir o texto do rótulo e adicionar o rótulo do Container ou janela

```
13 public class Janela extends JFrame{
14
15     private JLabel label1;
16
17     Janela(){
18         label1 = new JLabel("Texto do Rótulo 1");
19
20         this.add(label1);
21
22         this.setSize(300, 200);
23         this.setTitle("Teste de Janela");
24         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
25         this.setVisible(true);
26     }
27 }
28
29 }
```



## Usando um código HTML no Lugar de um Texto

```
17 public class Janela extends JFrame{
18
19     private JLabel label1;
20     private JLabel label2;
21     private JLabel label3;
22
23     Janela(){
24         label1 = new JLabel("<html><sub>Texto</sub> do <i>Rótulo<i> 1 </html>");
25
26         this.add(label1);
27
28         this.setVisible(true);
29     }
30
31 }
```



## Adicionando Múltiplos Componentes em uma Janela

- Para adicionar mais de um componente em uma janela, é preciso definir gerenciado do “modo” ou “fluxo” que esses objetos serão adicionar
- Isso é conhecido como **gerenciador de layout**
- A biblioteca padrão do java possui alguns gerenciadores de layout: BorderLayout, BoxLayout, CardLayout, FlowLayout, GridBagLayout, GridLayout, GroupLayout, SpringLayout

# Adicionando Múltiplos Componentes em uma Janela

```
17 public class Janela extends JFrame{
18
19     private JLabel label1;
20     private JLabel label2;
21     private JLabel label3;
22
23     Janela(){
24         this.setLayout(new FlowLayout());
25
26         label1 = new JLabel("Texto do Rótulo 1");
27         label2 = new JLabel();
28         label2.setText("Texto do Rótulo 2");
29
30         Icon bug = new ImageIcon("/home/rafael/Imagens/ufms.png");
31         label3 = new JLabel();
32         label3.setIcon(bug);
33         label3.setText("Rótulo com Figura");
34         label3.setHorizontalTextPosition(SwingConstants.CENTER);
35         label3.setVerticalTextPosition(SwingConstants.BOTTOM);
36         label3.setToolTipText("Imagem UFMS");
37
38         this.add(label1);
39         this.add(label2);
40         this.add(label3);
41
42         this.setSize(300, 200);
43         this.setTitle("Teste de Janela");
44         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
45         this.setVisible(true);
46     }
47 }
48 }
```

## Adicionando Múltiplos Componentes em uma Janela



## Adicionando Múltiplos Componentes em uma Janela

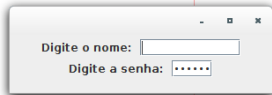
- **OBSERVAÇÃO 1:** para mais informações sobre os gerenciadores de layout consulte <https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html>
- **OBSERVAÇÃO 2:** algumas IDEs fornecem ferramentas de desenho da GUI em que é possível especificar o tamanho e local exato dos componentes visualmente

## Campos de Texto

- Para permitir que o usuário insira informações simples, como nome, CPF ou senhas, podemos utilizar os componente `JTextField` e `JPasswordField`
- Cada um desses componentes é uma área de uma única linha em que o usuário pode inserir textos pelo teclado

# Campos de Texto

```
16 public class Janela extends JFrame{
17
18     private JLabel label1;
19     private JLabel label2;
20     private JTextField text1;
21     private JPasswordField password1;
22
23
24     Janela(){
25         this.setLayout(new FlowLayout());
26         this.setSize(300, 100);
27
28         label1 = new JLabel("Digite o nome: ");
29         this.add(label1);
30         text1 = new JTextField(10);
31         this.add(text1);
32
33         label2 = new JLabel("Digite a senha: ");
34         this.add(label2);
35         password1 = new JPasswordField("123456");
36         this.add(password1);
37
38
39         this.setVisible(true);
40     }
41
42 }
```



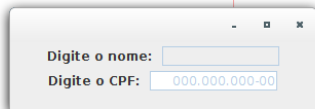


## Método Usuais do Campos de Texto

- `setEditable`: se `true` permite realizar a edição do texto contido no campo
- `setEnabled`: se `true` permite acessar o campo de texto
- `setAlignmentX`: define o alinhamento do texto no eixo horizontal
- `setAlignmentY`: define o alinhamento do texto no eixo vertical
- `getText`: recupera o texto inserido no campo
- `setText`: define o conteúdo do campo de texto

# Método Usuais do Campos de Texto

```
24  Janela(){  
    this.setLayout(new FlowLayout());  
    this.setSize(300, 100);  
  
27  
28    label1 = new JLabel("Digite o nome: ");  
    this.add(label1);  
30    text1 = new JTextField(10);  
    this.add(text1);  
32    text1.setEditable(false);  
33  
34    label2 = new JLabel("Digite o CPF: ");  
    this.add(label2);  
36    text2 = new JTextField("    000.000.000-00");  
37    text2.setEnabled(false);  
38    text2.setAlignmentX(SwingConstants.LEFT);  
    this.add(text2);  
41    this.setVisible(true);  
}
```



# Tratamento de Eventos

- As GUIs são **baseadas em evento**
- Quando o usuário interagir com um componente GUI, a interação (conhecida como evento) poderá fazer com que o programa realizar uma tarefa
- **Exemplos de interações**
  - Clicar em um botão
  - Digitar em um campo texto
  - Selecionar um item de um menu
  - Fechar uma janela
  - Mover um mouse sobre um componente

## Tratamento de Eventos

- O código que realiza uma tarefa em resposta a um evento é chamado de **handler de evento**
- O processo de responder a um evento é chamado de **tratamento de evento**
- Cada tipo de componente GUI pode gerar muitos eventos em resposta a interações de usuário
- Cada evento é representado por uma classe e pode ser processado apenas pelo tipo de *handler* apropriado

## Tratamento de Eventos

- Os eventos suportados por um componente são descritos na documentação da Java API
- Por exemplo, quando um usuário pressionar a tecla *Enter* em um `JTextField` ou `JPasswordField`, ocorre um `ActionEvent` (pacote `java.awt.event`)
- Um evento assim é processado por um objeto que implementa a interface `ActionListener`

## Tratamento de Eventos

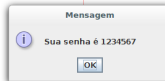
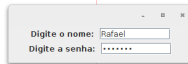
- Ao se implementar a interface `ActionListener`, deve-se implementar o método `ActionPerformed`
- Para se adicionar um `ActionListener` em um componente de campo de texto, basta utilizar o método `addActionListener` e passar um objeto `ActionLister` como argumento

# Tratamento de Eventos

```

19 public class Janela extends JFrame{
20
21     private JLabel label1;
22     private JLabel label2;
23     private JTextField text1;
24     private JPasswordField password1;
25
26
27     Janela(){
28         this.setLayout(new FlowLayout());
29         this.setSize(300, 95);
30
31         label1 = new JLabel("Digite o nome: ");
32         this.add(label1);
33         text1 = new JTextField(10);
34         this.add(text1);
35
36         label2 = new JLabel("Digite a senha: ");
37         this.add(label2);
38         password1 = new JPasswordField(10);
39         this.add(password1);
40
41         TextFieldHandler handler = new TextFieldHandler();
42         text1.addActionListener(handler);
43         password1.addActionListener(handler);
44
45         this.setVisible(true);
46     }
47
48     private class TextFieldHandler implements ActionListener{
49         @Override
50         public void actionPerformed(ActionEvent event) {
51             if(event.getSource() == text1){
52                 JOptionPane.showMessageDialog(null, "Seu nome é " + event.getActionCommand());
53             }else if(event.getSource() == password1){
54                 JOptionPane.showMessageDialog(null, "Sua senha é " + event.getActionCommand());
55             }
56         }
57     }
58 }
59

```



## Tratamento de Eventos

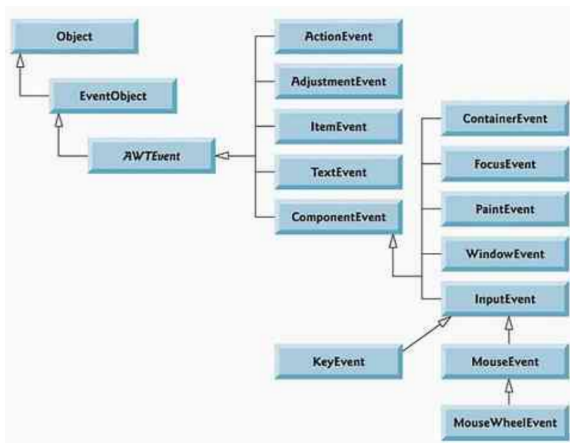
- O componente GUI com o qual o usuário interage é a **origem do evento**
- No exemplo anterior, quando o usuário pressionar *Enter* enquanto um desses componentes GUI tiver o foco, o sistema cria um objeto `ActionEvent` único que contém informações sobre o evento que acabou de ocorrer, como a origem do evento e o texto no campo de texto
- O sistema então passa esse objeto `ActionEvent` em uma chamada de método para o método `actionPerformed` do ouvinte do evento



## Tipos Comuns de Eventos GUI e Interfaces Ouvintes

- Muitos tipos diferentes de eventos podem ocorrer quando o usuário interage com um GUI
- As informações de evento são armazenadas em um objeto de uma classe que estende a classe `AWTEvent` (do pacote `java.awt`)
- Tipos adicionais de eventos que são específicos dos componentes Swing GUI são declarado no pacote `javax.swing.event`

# Algumas classes de eventos do pacote `java.awt.event`



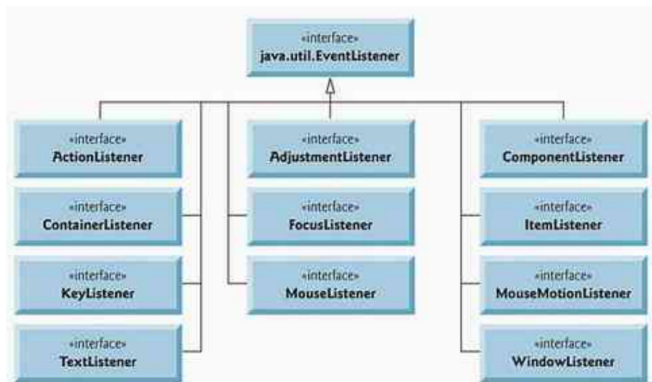
## Tratamento de Eventos

- Basicamente o Tratamento de Eventos pode ser dividido em três partes
  - **A origem do evento:** componente GUI com o qual o usuário interage
  - **O objeto do evento:** encapsula informações sobre o evento que ocorreu (origem, conteúdo, ...)
  - **O ouvinte do evento:** objeto que é notificado pela origem de evento quanto um evento ocorre. Um método do ouvinte de evento recebe um objeto do evento quando o ouvinte de evento é notificado

## Tratamento de Eventos

- Para cada tipo de objeto de evento, há em geral uma interface `listener` de eventos correspondentes
- Um ouvinte de evento para um evento GUI é um objeto de uma classe que implementa uma ou mais das interfaces ouvintes de evento dos pacotes `java.awt.event` e `javax.swing.event`
- Os tipos de ouvinte de eventos adicionais que são específicos dos componentes Swing são declarados no pacote `javax.swing.event`

# Tratamento de Eventos



## Botões

- Um **botão** é um componente em que o usuário clica para acionar uma ação específica
- Um aplicativo Java pode usar vários tipos de botões
  - Botões de comando
  - Caixas de seleção
  - Botões de alternância
  - Botões de opção

## JButton

- Um botão de comando gera um `ActionEvent` quando o usuário clica nele
- Os botões de comando são criados com a classe `JButton`
- O texto na face de um `JButton` é chamado **rótulo de botão**
- Botões podem inclusive conter ícones

**JButton**

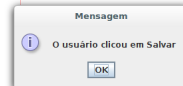
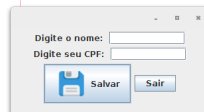
Botões que Mantêm o Estado  
 JRadioButton

# JButton

```

30  Janela(){
    this.setLayout(new FlowLayout());
    this.setSize(300, 160);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

34
35    JLabel l1 = new JLabel("Digite o nome: ");
    this.add(l1);
37    JTextField text1 = new JTextField(10);
    this.add(text1);
39
40    JLabel l2 = new JLabel("Digite seu CPF: ");
    this.add(l2);
42    JTextField text2 = new JTextField(10);
    this.add(text2);
44
45
46    JButton button1 = new JButton();
47    button1.setText("Salvar");
48    Icon icon = new ImageIcon("/home/rafael/Imagens/save_icon.png");
49    button1.setIcon(icon);
    this.add(button1);
51
52    JButton button2 = new JButton("Sair");
    this.add(button2);
54
55
56    ButtonHandler handler = new ButtonHandler();
57    button1.addActionListener(handler);
58    button2.addActionListener(handler);
59
60    this.setVisible(true);
61  }
62
63  private class ButtonHandler implements ActionListener{
64      @Override
65      public void actionPerformed(ActionEvent event) {
66          JOptionPane.showMessageDialog(null, "O usuário clicou em " + event.getActionCommand());
67      }
68
69  }
  
```





## Botões que Mantêm o Estado

- Botões que mantêm o estado são botões possuem valores de ativado/desativado ou verdadeiro/falso
- A classe Swing GUI contém três tipos de **botões de estado**
  - JToggleButton
  - JCheckBox
  - JRadioButton

## JCheckBox

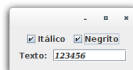
- Um JCheckBox ou caixa de seleção permite selecionar (ou marcar) opções
- Podem haver várias caixa de seleção em um Container e elas podem ser selecionadas independentes umas das outras
- Principais da classe JCheckBox
  - isSelected: retorna verdadeiro se um objeto está selecionado e falso caso contrário
  - setSelected: define se um botão está ou não selecionado

## JCheckBox

- Quando o usuário clicar em um JCheckBox, um `ItemEvent` ocorre
- Esse objeto pode ser tratado por um objeto `ItemListener` que deve implementar o método `itemStateChanged`
- Ao criar o handler, basta adicioná-lo utilizando o método `addItemListener`

## JCheckBox

```
26 Janela(){  
27     this.setLayout(new FlowLayout());  
28     this.setSize(200, 100);  
29     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
30  
31     check1 = new JCheckBox();  
32     check1.setText("Itálico");  
33     this.add(check1);  
34  
35     check2 = new JCheckBox("Negrito");  
36     this.add(check2);  
37  
38     label1 = new JLabel("Texto: ");  
39     this.add(label1);  
40     text1 = new JTextField(10);  
41     this.add(text1);  
42  
43     CheckHandler handler = new CheckHandler();  
44     check1.addItemListener(handler);  
45     check2.addItemListener(handler);  
46  
47     this.setVisible(true);  
48 }  
49  
50 private class CheckHandler implements ItemListener{  
51  
52     public void itemStateChanged(ItemEvent e) {  
53         Font font = null;  
54         if(check1.isSelected() && check2.isSelected()){  
55             font = new Font("Serif", Font.BOLD + Font.ITALIC, 12);  
56         }else if(check1.isSelected()){  
57             font = new Font("Serif", Font.ITALIC, 12);  
58         }else if(check2.isSelected()){  
59             font = new Font("Serif", Font.BOLD, 12);  
60         }else{  
61             font = new Font("Serif", Font.PLAIN, 12);  
62         }  
63         text1.setFont(font);  
64     }  
65 }  
66 }
```



## JRadioButton

- Os JRadioButtons ou botões de opção também permitem habilitar/desabilitar opções
- Entretanto, a ideia é que as opções apareçam em um grupo de opções e apenas uma opção do grupo possa ser selecionada (opções mutuamente exclusivas)
- Esse relacionamento entre os botões é mantido por um objeto ButtonGroup (pacote javax.swing)
- Os JRadioButtons são adicionados a um objeto ButtonGroup utilizando o método add

## JRadioButton

- Um objeto ButtonGroup não precisa ser adicionado a um componente
- Assim como no JCheckBox, um dos principais métodos do JRadioButton é o método isSelected
- Assim como o JCheckBox, o JRadioButton também geram ItemEvents quando se clicam neles
- Vários métodos do JRadioButton são parecidos com os do JCheckBox

# JRadioButton

```

29  Janela(){
30      this.setLayout(new FlowLayout());
31      this.setSize(400, 100);
32      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
33
34      buttonPlano = new JRadioButton("Plano", true);
35      this.add(buttonPlano);
36      buttonNegrito = new JRadioButton("Negrito");
37      this.add(buttonNegrito);
38      buttonItalico = new JRadioButton("Italico");
39      this.add(buttonItalico);
40      buttonNegIt = new JRadioButton("Negrito/Italico");
41      this.add(buttonNegIt);
42
43      ButtonGroup buttonGroup = new ButtonGroup();
44      buttonGroup.add(buttonNegrito);
45      buttonGroup.add(buttonNegIt);
46      buttonGroup.add(buttonItalico);
47      buttonGroup.add(buttonPlano);
48
49      JLabel label1 = new JLabel("Texto: ");
50      this.add(label1);
51      text1 = new JTextField(15);
52      this.add(text1);
53
54      OptionHandler handler = new OptionHandler();
55      buttonNegrito.addItemListener(handler);
56      buttonItalico.addItemListener(handler);
57      buttonNegIt.addItemListener(handler);
58      buttonPlano.addItemListener(handler);
59
60      this.setVisible(true);
61  }
62
63  private class OptionHandler implements ItemListener{
64      public void itemStateChanged(ItemEvent e) {
65          Font font = null;
66          if(buttonNegrito.isSelected()){
67              font = new Font("Serif", Font.BOLD, 12);
68          }else if(buttonItalico.isSelected()){
69              font = new Font("Serif", Font.ITALIC, 12);
70          }else if(buttonNegIt.isSelected()){
71              font = new Font("Serif", Font.BOLD + Font.ITALIC, 12);
72          }else{
73              font = new Font("Serif", Font.PLAIN, 12);
74          }
75          text1.setFont(font);
76      }
77  }
78  }
79  }

```

## Material Complementar

- Projetando uma GUI Swing no NetBeans IDE

[https://netbeans.org/kb/docs/java/quickstart-gui\\_pt\\_BR.html](https://netbeans.org/kb/docs/java/quickstart-gui_pt_BR.html)

- Interfaces gráficas com Swing

[http://www.caelum.com.br/apostila-java-testes-xml-design-patterns/  
interfaces-graficas-com-swing/#5-1-interfaces-graficas-em-java](http://www.caelum.com.br/apostila-java-testes-xml-design-patterns/interfaces-graficas-com-swing/#5-1-interfaces-graficas-em-java)



## Material Complementar

Na programação,  
3% é talento.  
**Os outros 97% é não  
se distrair na internet.**

**dynx**

# Programação Orientada a Objetos

<http://lives.ufms.br/moodle/>

Rafael Geraldeli Rossi  
rafael.g.rossi@ufms.br

Slides baseados em [Deitel and Deitel, 2010]

## Referências Bibliográficas I



Deitel, P. and Deitel, H. (2010).

*Java: How to Program.*

How to program series. Pearson Prentice Hall, 8th edition.