

# Interface Gráfica



Prof. Bruno Gomes  
bruno.gomes@ifrn.edu.br

**Programação Orientada a Objetos**

# Agenda

- Componente JOptionPane
- Layout Null
- Tratamento de Eventos
  - Action Listener

# Componente JOptionPane

- *javax.swing.JOptionPane*
- Caixas de dialogo usadas para a entrada e saída de dados

# Componente JOptionPane

Método	Descrição
<b>showConfirmDialog</b>	Pergunta uma confirmação, como sim/não/cancelar
<b>showInputDialog</b>	Solicita um valor de entrada
<b>showMessageDialog</b>	Informa ao usuário alguma coisa que aconteceu

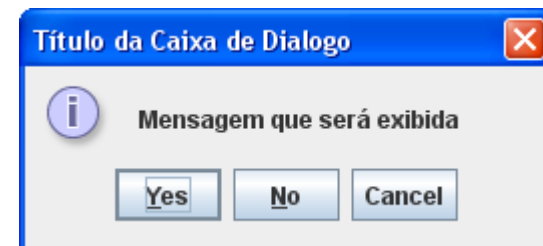
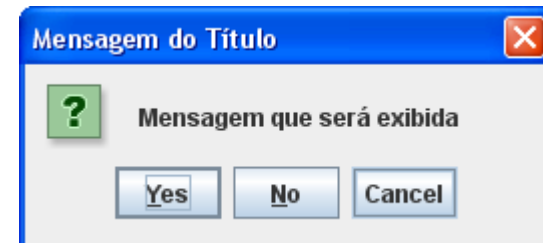
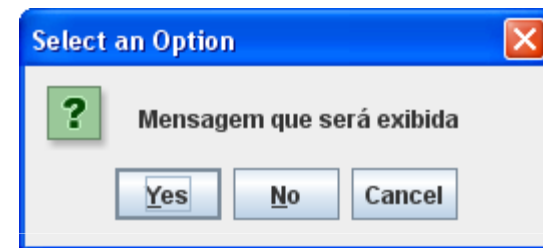
# Componente JOptionPane

- Exemplos - `JOptionPane.showConfirmDialog`

```
JOptionPane.showConfirmDialog(null,  
"Mensagem que será exibida");
```

```
JOptionPane.showConfirmDialog(null, "Mensagem que  
será exibida",  
"Mensagem do Título",  
JOptionPane.YES_NO_CANCEL_OPTION );
```

```
JOptionPane.showConfirmDialog(null, "Mensagem que será  
exibida",  
"Título da Caixa de Dialogo",  
JOptionPane.YES_NO_CANCEL_OPTION,  
JOptionPane.INFORMATION_MESSAGE) ;
```



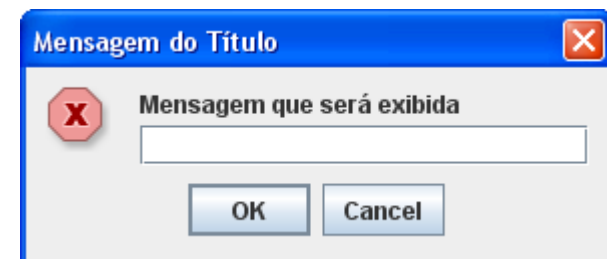
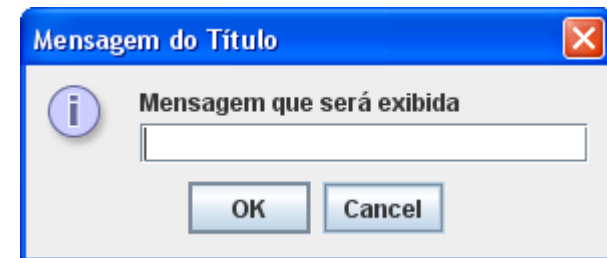
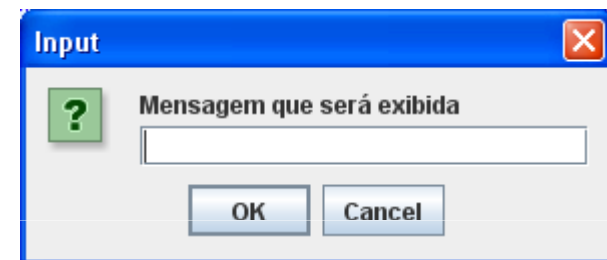
# Componente JOptionPane

- Exemplos - `JOptionPane.showInputDialog`

```
JOptionPane.showInputDialog("Mensagem que  
será exibida");
```

```
JOptionPane.showInputDialog(null,  
"Mensagem que será exibida",  
"Mensagem do Título",  
JOptionPane.INFORMATION_MESSAGE);
```

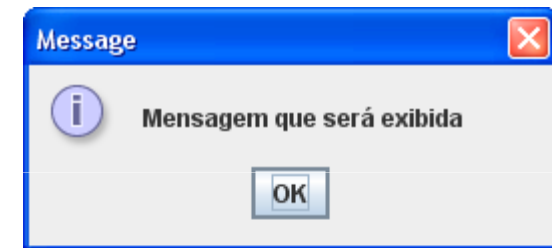
```
JOptionPane.showInputDialog(null, "Mensagem  
que será exibida",  
"Mensagem do Título",  
JOptionPane.ERROR_MESSAGE);
```



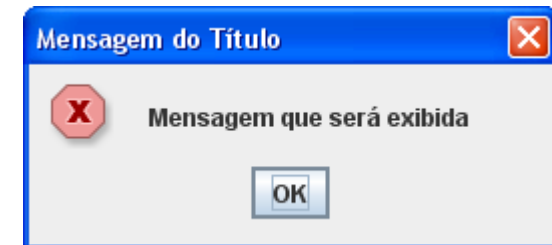
# Componente JOptionPane

- Exemplos - `JOptionPane.showMessageDialog`

```
JOptionPane.showMessageDialog(null, "Mensagem que  
será exibida");
```



```
JOptionPane.showMessageDialog(null, "  
Mensagem que será exibida",  
"Mensagem do Título",  
JOptionPane.ERROR_MESSAGE);
```



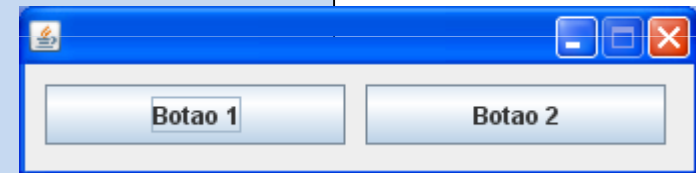
# Layout Null

- Elementos são posicionados manualmente
- Utilizado apenas se a janela não vai ser redimensionada



# Layout Null - Exemplo

```
JFrame frame = new JFrame();  
  
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
frame.setLayout(null);  
frame.setSize(340, 85);  
  
JButton botao1 = new JButton("Botao 1");  
botao1.setLocation(10,10);  
botao1.setSize(150,30);  
  
JButton botao2 = new JButton("Botao 2");  
botao2.setLocation(170,10);  
botao2.setSize(150,30);  
  
frame.add(botao1);  
frame.add(botao2);  
  
frame.setResizable(false);  
frame.setVisible(true);
```



# Tratamento de Eventos

- GUIs são baseadas em eventos
  - Geram eventos quando o usuário interage com a GUI
- Quando ocorre uma interação com usuário, um evento é enviado ao programa
- Informações de eventos GUI são armazenados em um objeto de uma classe que estende **AWTEvent**
- Qualquer objeto pode ser notificado do evento.
  - Tudo que o objeto tem que fazer é implementar a interface apropriada e ser registrado como um tratador de eventos (*event listener*) no evento apropriado.

# Tratamento de Eventos

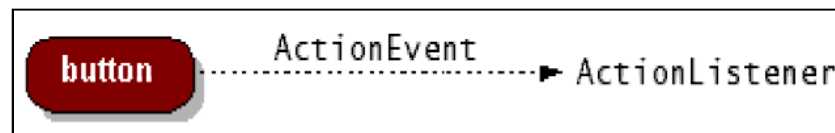
- Mecanismo de tratamento de eventos tem três partes:
  - Origem do evento
    - Classe ou implementa a interface listener ou herda de uma classe que implementa a interface listener
    - **public class MyClass implements ActionListener**
  - Objeto do evento
    - Registra uma instância da classe tratadora do evento como um listener em um ou mais componentes
    - **algunComponent.addActionListener(instanceOf MyClass);**

# Tratamento de Eventos

- “Ouvinte” (listener) do evento
  - A classe tratadora do evento deve conter o código que implementa o método na interface listener.
    - **public void actionPerformed(ActionEvent e) { ...}**

# Tratamento de Eventos

- Em geral, para detectar quando o usuário clica em algum botão (ou tecla um caracter equivalente), um programa deve ter um objeto que implementa a interface ActionListener
- O programa deve registrar seu objeto como um action listener no botão (a origem do evento), usando o método addActionListener
- Quando o usuário clica no botão, o botão dispara um evento (action event). Isto resulta na invocação do método actionPerformed do action listener. O único argumento deste método é o objeto que dá informações sobre o evento e sua origem



# Event Listener

- Qualquer quantidade de objetos “event listener” podem aguardar por todos os tipos de eventos de qualquer número de objetos origem
  - Um programa pode criar um listener por origem de evento
  - Um programa pode ter mais de um listener para um único tipo de evento de uma única origem de eventos.

# Event Listener

- Cada evento é representado por um objeto que dá informações sobre o evento e identifica a origem do evento
- A origem do evento é, frequentemente, algum componente

# Action Listener

- Action listeners são tratadores de eventos
- Devemos implementar action listeners para responder a indicação do usuário de que alguma ação deve ocorrer
  - Cliques em botões
  - Escolha de itens de menu
  - Digitar Enter em um text field
- Como resposta, uma mensagem actionPerformed é enviada para todos os action listeners que estão registrados no componente associado





# Action Listener

- A API Action Listener:
- Interface ActionListener
  - `actionPerformed(ActionEvent)`


# Exemplo 01

```
public class Evento01 extends JFrame implements ActionListener{  
  
    public Evento01(){  
        iniciarComponentes();  
        this.pack();  
        this.setVisible(true);  
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    }  
  
    public void iniciarComponentes(){  
        botao1 = new JButton("Acao1");  
        botao1.setActionCommand("nomeAcao1");  
        botao1.addActionListener(this);  
        this.add(botao1);  
    }  
}
```



# Exemplo 01

```
public void actionPerformed(ActionEvent e) {  
    if("nomeAcao1".equals(e.getActionCommand())){  
        JOptionPane.showMessageDialog(null, "Acao Disparada");  
    }  
}  
  
public static void main(String[] args) {  
    Evento01 evento01 = new Evento01();  
}  
}
```



# Exemplo 02

```
public class Evento02 extends JFrame implements ActionListener{
protected JButton botao1;


public Evento02(){
    iniciarComponentes();
    this.pack();
    this.setVisible(true);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

public void iniciarComponentes(){
    botao1 = new JButton("Acao1");
    botao1.addActionListener(this);
    this.add(botao1);
}
```



# Exemplo 02

```
public void actionPerformed(ActionEvent e) {  
    if(e.getSource()==botao1 ){  
        JOptionPane.showMessageDialog(null, "Acao Disparada");  
    }  
}  
  
public static void main(String[] args) {  
    Evento02 evento02 = new Evento02();  
}  
}
```



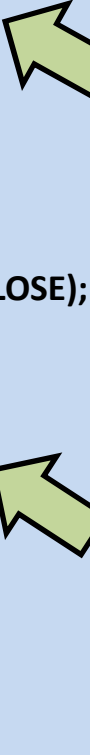
# Exemplo 03 – Classe Separada de Evento

```
public class TratamentoEventos implements ActionListener{

    public void actionPerformed(ActionEvent e) {
        if("nomeAcao1".equals(e.getActionCommand())){
            JOptionPane.showMessageDialog(null, "Acao Disparada");
        }
    }
}
```

# Exemplo 03 – Classe Separada de Evento

```
public class Evento03 extends JFrame {  
    protected JButton botao1;  
    protected TratamentoEventos tratamentoEventos;  
  
    public Evento03(){  
        iniciarComponentes();  
        this.pack();  
        this.setVisible(true);  
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    }  
  
    public void iniciarComponentes(){  
        tratamentoEventos = new TratamentoEventos();  
        botao1 = new JButton("Acao1");  
        botao1.setActionCommand("nomeAcao1");  
        botao1.addActionListener(tratamentoEventos);  
        this.add(botao1);  
    }  
  
    public static void main(String[] args) {  
        Evento03 evento03 = new Evento03();  
    }  
}
```



# Exercício

- Crie um JFrame
- Mude o layout para Null
- Adicionar dois JTextField e três JButton
- Adicione os seguintes eventos para os botões:
  - No primeiro, muda o texto do primeiro JTextField para “O Botão 1 foi clicado”
  - No segundo, muda o texto do segundo JTextField para “O Botão 2 foi clicado”
  - No Terceiro, exibe uma caixa de diálogo (JOptionPane) com o texto: “O Botão 3 foi clicado”