

# Herança

Prof. Ricardo P. Mesquita

# Superclasse e Subclasse

- Frequentemente um objeto de uma classe também **é um** objeto de outra classe
  - Por exemplo, um objeto *retângulo* (mais específico) **é um** objeto *quadrilátero* (mais geral)
  - Uma classe retângulo *herda* da classe *quadrilátero*
  - Assim, temos:
    - Classe Retângulo: **subclasse**
    - Classe Quadrilátero: **superclasse**

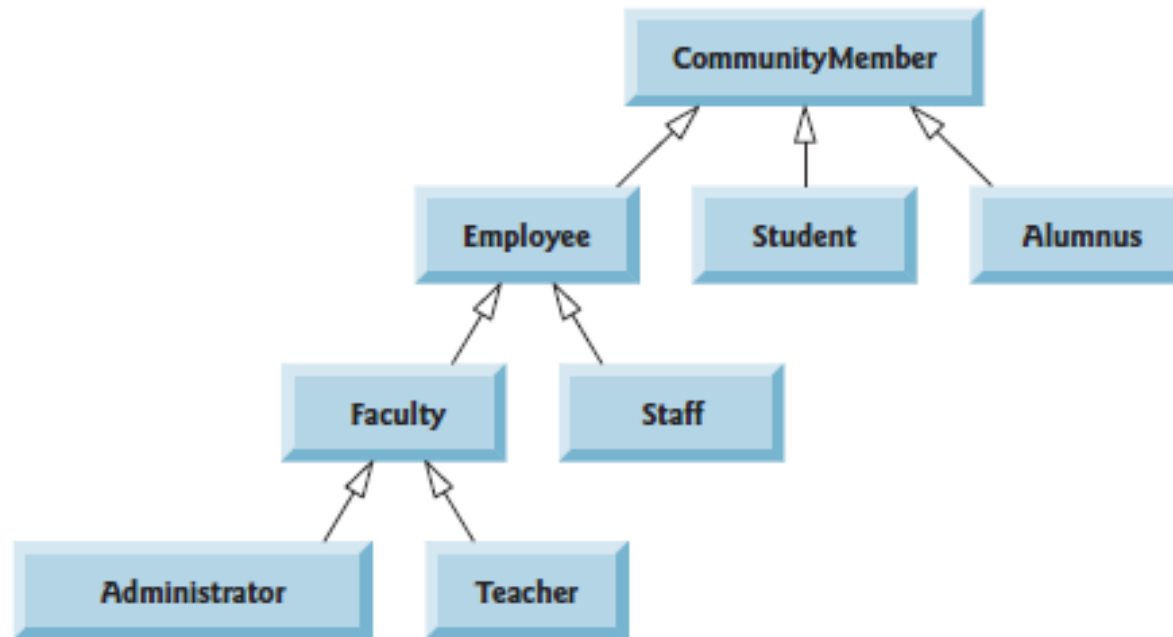
# Superclasse e Subclasse

- Cada objeto de uma subclasse **é um** objeto da superclasse
- Uma superclasse pode ter muitas subclasses

Superclasse	Subclasses
Student	GraduateStudent, UndergraduateStudent
Shape	Circle, Triangle, Rectangle, Sphere, Cube
Loan	CarLoan, HomeImprovementLoan, MortgageLoan
Employee	Faculty, Staff
BankAccount	CheckingAccount, SavingsAccount

# Relações de Herança

- Formam estruturas hierárquicas



# Atenção: Herança é diferente de Composição!

- A herança é definida por um relacionamento **é um**.
  - Por exemplo, um objeto da classe Carro (subclasse) **é um** objeto da classe Meio\_de\_Transporte (superclasse)
- A composição é definida por um relacionamento **tem um**.
  - Um objeto da classe Carro **tem um** objeto da classe Motor.
  - *Note: o motor **não é um** carro.*

# Membros **protected**

- O acesso **protected** oferece um nível de acesso intermediário entre o **public** e o **private**
- Os membros **protected** de uma superclasse podem ser acessados por membros dessa classe, por membros de suas subclasses e por membros de outras classes no mesmo pacote
  - Membros **protected** também têm *acesso de pacote*.

# Membros **protected**

- Utilize o modificador de acesso **protected** quando a superclasse precisar fornecer um método somente para as suas subclasses e outras classes no mesmo pacote, mas não para outros clientes.
  - Declarar tais variáveis **private** permite a implementação de superclasse dessas variáveis de instância sem afetar as implementações de subclasse
  - Quando possível ***não inclua*** variáveis de instância **protected** em uma superclasse.

# Considerações

- Construtores não são herdados!
- Mas... Construtores das superclasses estão disponíveis para as subclasses.
- *A primeira tarefa de qualquer construtor de subclasse é chamar o construtor da superclasse direta, explícita ou implicitamente (se nenhuma chamada ao construtor for especificada).*
- Se o código não incluir uma chamada explícita ao construtor da superclasse, o Java chama implicitamente o construtor padrão (ou sem argumentos) da superclasse.



# Considerações

- A notação **@Override** indica que o método local deve *sobrescrever* um método da superclasse.
- Quando o compilador encontra um método declarado @Override, ele compara as assinaturas do método da superclasse. Se não houver uma correspondência exata, o compilador emite uma mensagem de erro (*method does not override or implement a method from a supertype*)
  - Ou seja, você, acidentalmente, sobrecarregou o método da superclasse.

# Atenção

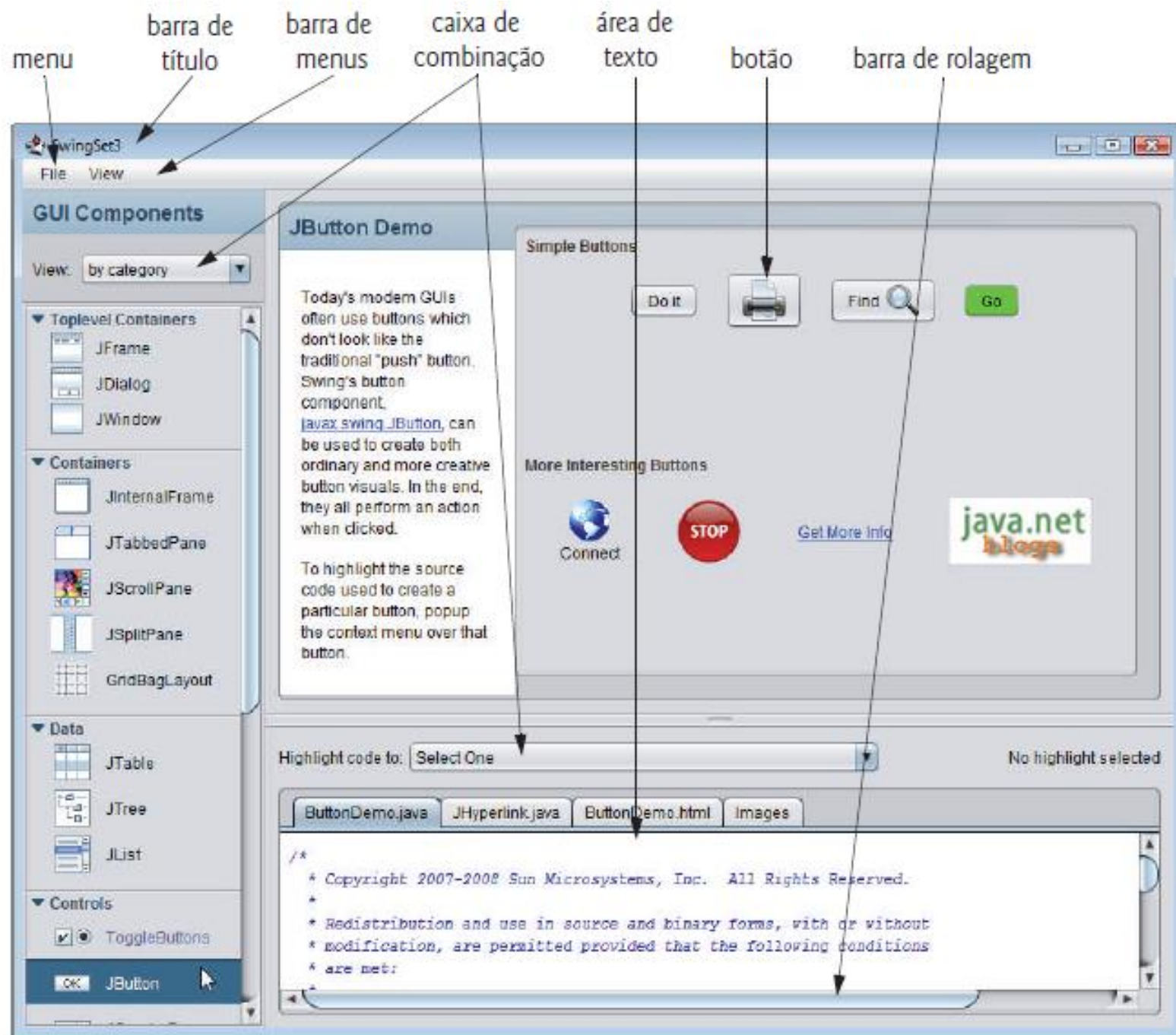
- É um erro de sintaxe sobrescrever um método com um modificador de acesso mais restrito
  - Um método **public** da superclasse não pode tornar-se um método **protected** ou **private** na subclasse.

# Chamando o Construtor da Superclasse

- Sintaxe de chamada de construtor da superclasse:
  - Palavra-chave **super** seguida do conjunto de argumentos do construtor da superclasse (entre parênteses)
  - Quando uma superclasse contiver um construtor sem argumento, você pode utilizar **super()** para chamar esse construtor explicitamente (isso raramente é feito)
  - Um erro de compilação ocorre se um construtor de subclasse chamar um construtor de superclasse com argumentos que não coincidam com o número e os tipos de parâmetros em um dos construtores na superclasse.

# Uso de Componentes Gráficos

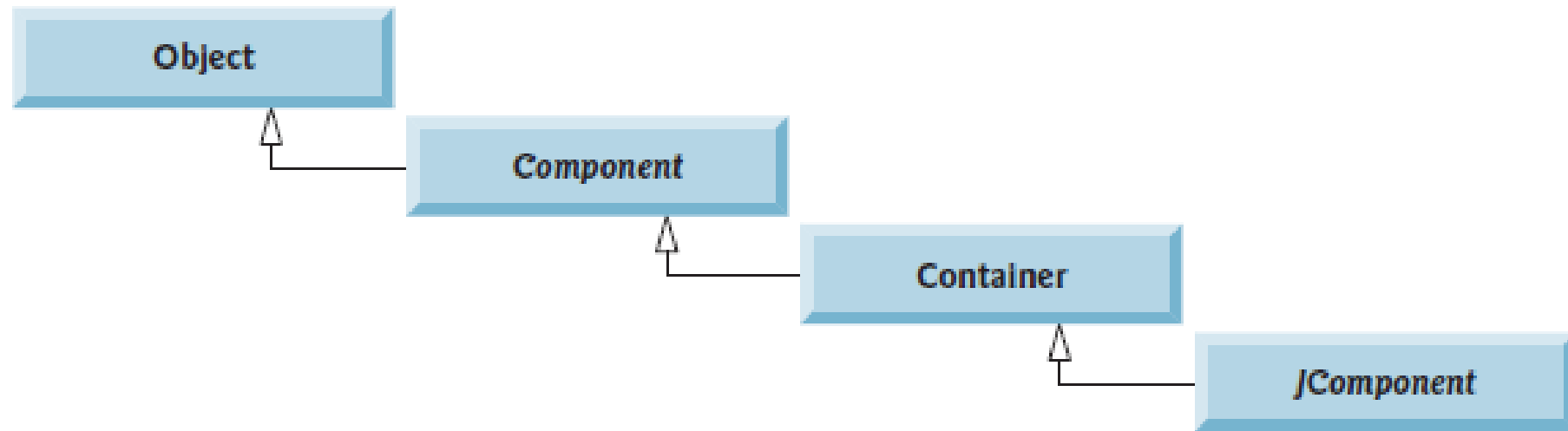
- Para exemplificar a herança, vamos iniciar o estudo de componentes gráficos.
- Componentes GUI (*Graphical User Interface*)



# Visão Geral de Componentes Swing

Componente	Descrição
JLabel	Exibe <i>texto</i> e/ou ícones <i>não editáveis</i> .
TextField	Normalmente <i>recebe entrada</i> do usuário.
Button	Dispara um evento quando o usuário clicar nele com o mouse.
CheckBox	Especifica uma opção que pode <i>ser</i> ou <i>não selecionada</i> .
ComboBox	Uma <i>lista drop-down dos itens</i> a partir dos quais o <i>usuário</i> pode fazer uma <i>seleção</i> .
List	Uma <i>lista dos itens</i> a partir dos quais o usuário pode fazer uma <i>seleção clicando</i> em <i>qualquer um</i> deles. <i>Múltiplos</i> elementos <i>podem</i> ser selecionados.
Panel	Uma área em que os <i>componentes</i> podem ser <i>colocados e organizados</i> .

# Relacionamento de Componentes Swing



# Exemplo simples

```
import javax.swing.JOptionPane;

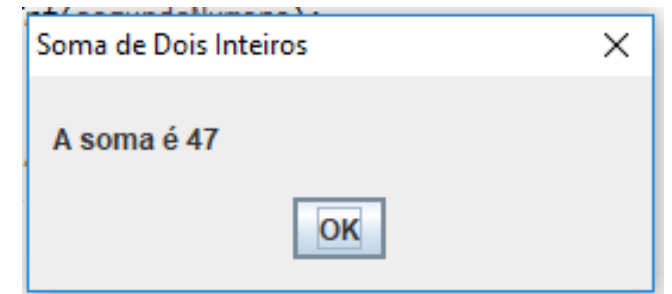
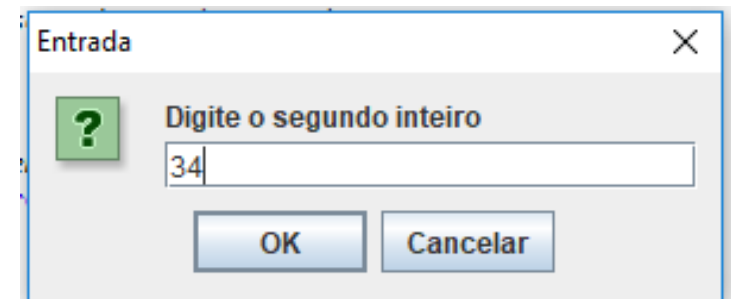
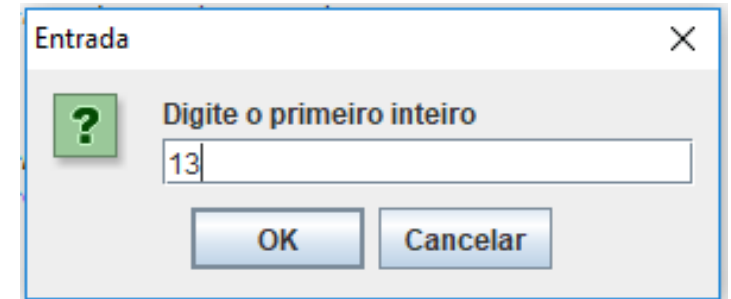
public class Soma {

    public static void main(String[] args)
    {
        String primeiroNumero =
            JOptionPane.showInputDialog("Digite o primeiro inteiro");
        String segundoNumero =
            JOptionPane.showInputDialog("Digite o segundo inteiro");

        int num1 = Integer.parseInt(primeiroNumero);
        int num2 = Integer.parseInt(segundoNumero);





        int soma = num1 + num2;

        JOptionPane.showMessageDialog(null, "A soma é " + soma,
            "Soma de Dois Inteiros", JOptionPane.PLAIN_MESSAGE);
    }
}
```





# Constantes para Diálogo

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

- **Exemplo:**  
exibição de  
textos e imagens

```
import java.awt.FlowLayout;           // layout
import javax.swing.JFrame;            // características da janela
import javax.swing.JLabel;            // exibir texto e imagem
import javax.swing.SwingConstants;    // contantes comuns usadas com Swing
import javax.swing.Icon;              // interface usada para manipular imagens
import javax.swing.ImageIcon;         // carregar uma imagem
```

```
public class RotuloFrame extends JFrame {

    private final JLabel rotulo1;
    private final JLabel rotulo2;
    private final JLabel rotulo3;

    public RotuloFrame()
    {
        super("Testando JLabel");
        setLayout(new FlowLayout());

        rotulo1 = new JLabel("Rótulo com texto");
        rotulo1.setToolTipText("Este é o rótulo 1");
        add(rotulo1);

        Icon bug = new ImageIcon(getClass().getResource("bug1.png"));
        rotulo2 = new JLabel("Rótulo com texto e imagem", bug,
                               SwingConstants.LEFT);
        rotulo2.setToolTipText("Este é o rótulo 2");
    }
}
```

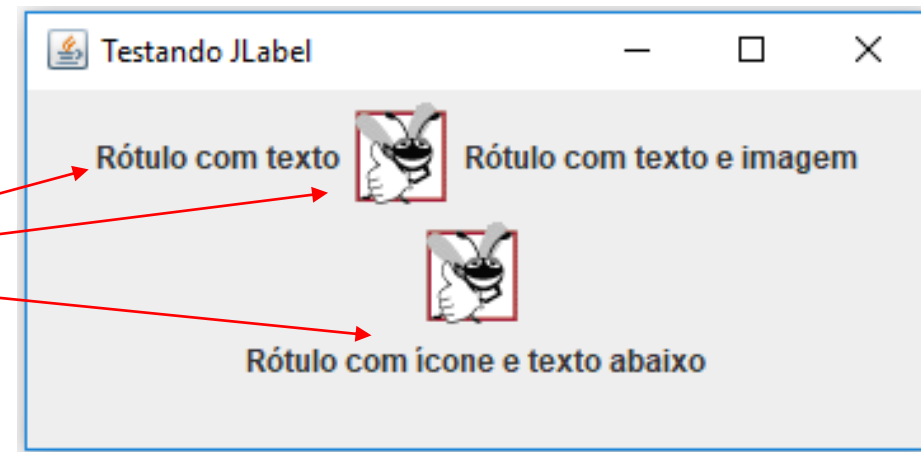
- **Exemplo:**  
exibição de  
textos e imagens

```
import javax.swing.JFrame;

public class RotuloTeste {

    public static void main(String[] args)
    {
        RotuloFrame rotuloFrame = new RotuloFrame();
        rotuloFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        rotuloFrame.setSize(260, 180);
        rotuloFrame.setVisible(true);
    }
}
```

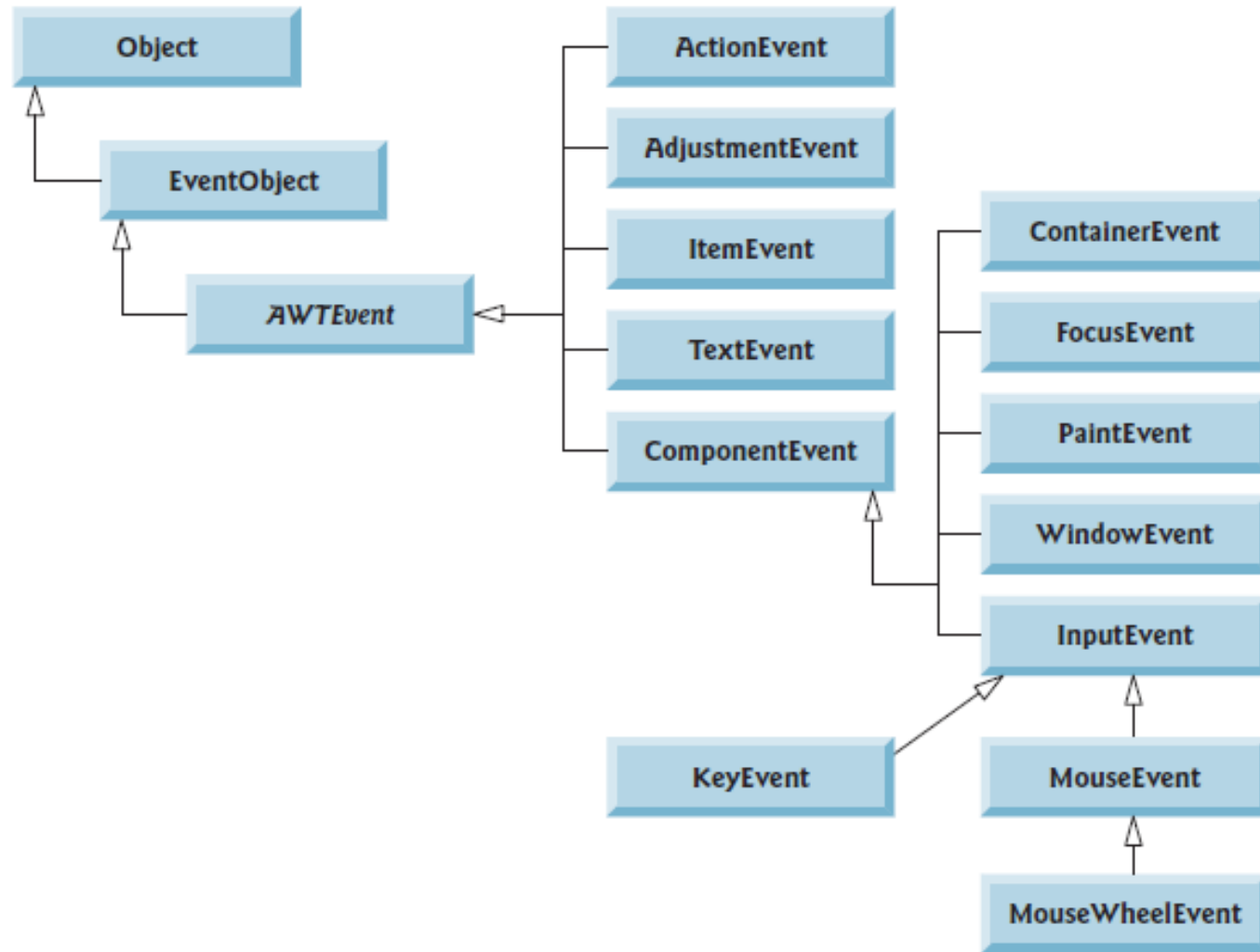
Se você posicionar o mouse sobre o rótulo aparecerá a descrição (rótulo 1, 2 e 3)



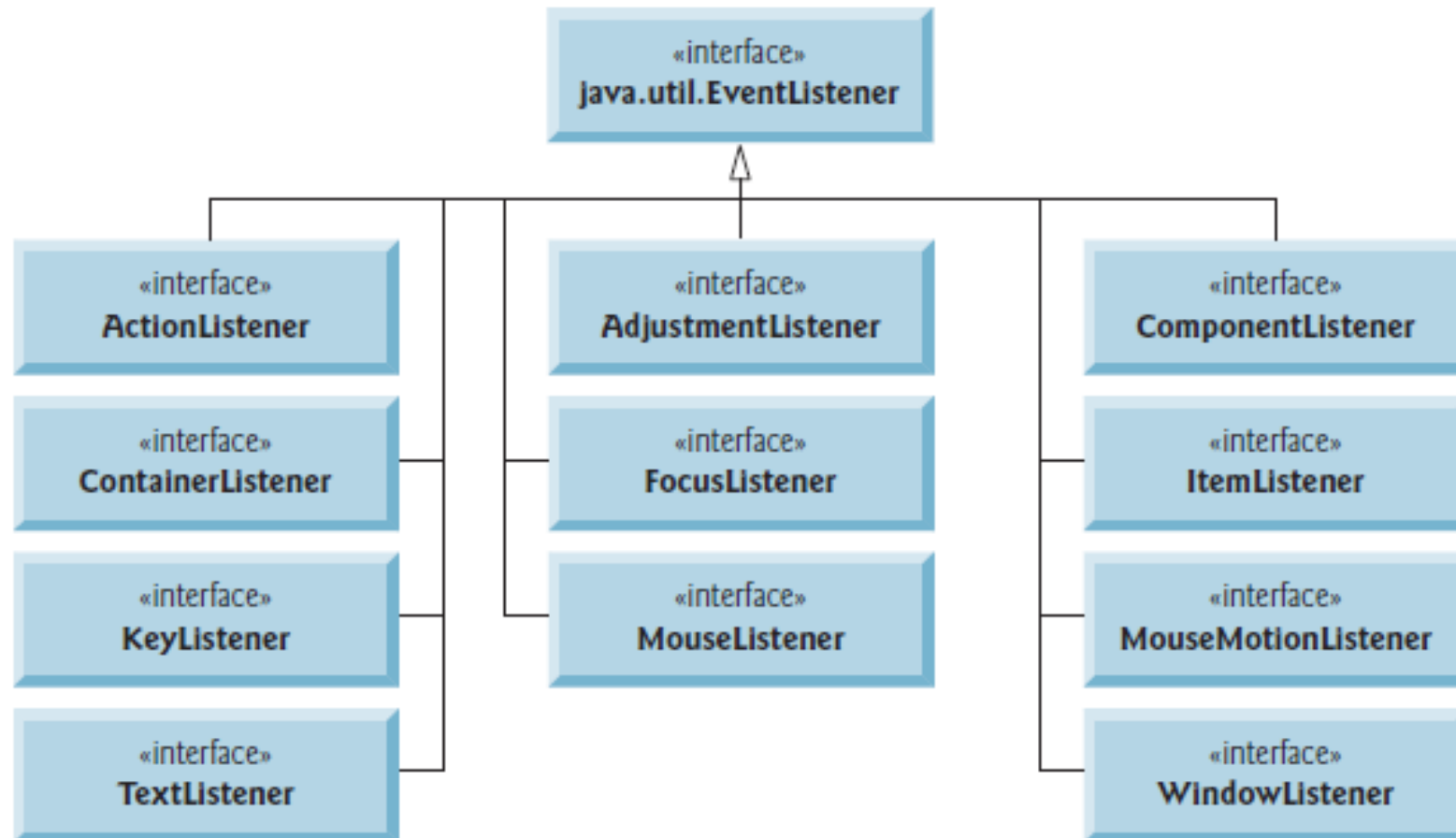
# Constante de Posicionamento

Constante	Descrição	Constante	Descrição
<i>Constantes de posição horizontal</i>		<i>Constantes de posição vertical</i>	
LEFT	Coloca o texto à esquerda	TOP	Coloca o texto na parte superior
CENTER	Coloca o texto no centro	CENTER	Coloca o texto no centro
RIGHT	Coloca o texto à direita	BOTTOM	Coloca o texto na parte inferior

# Classes de Evento

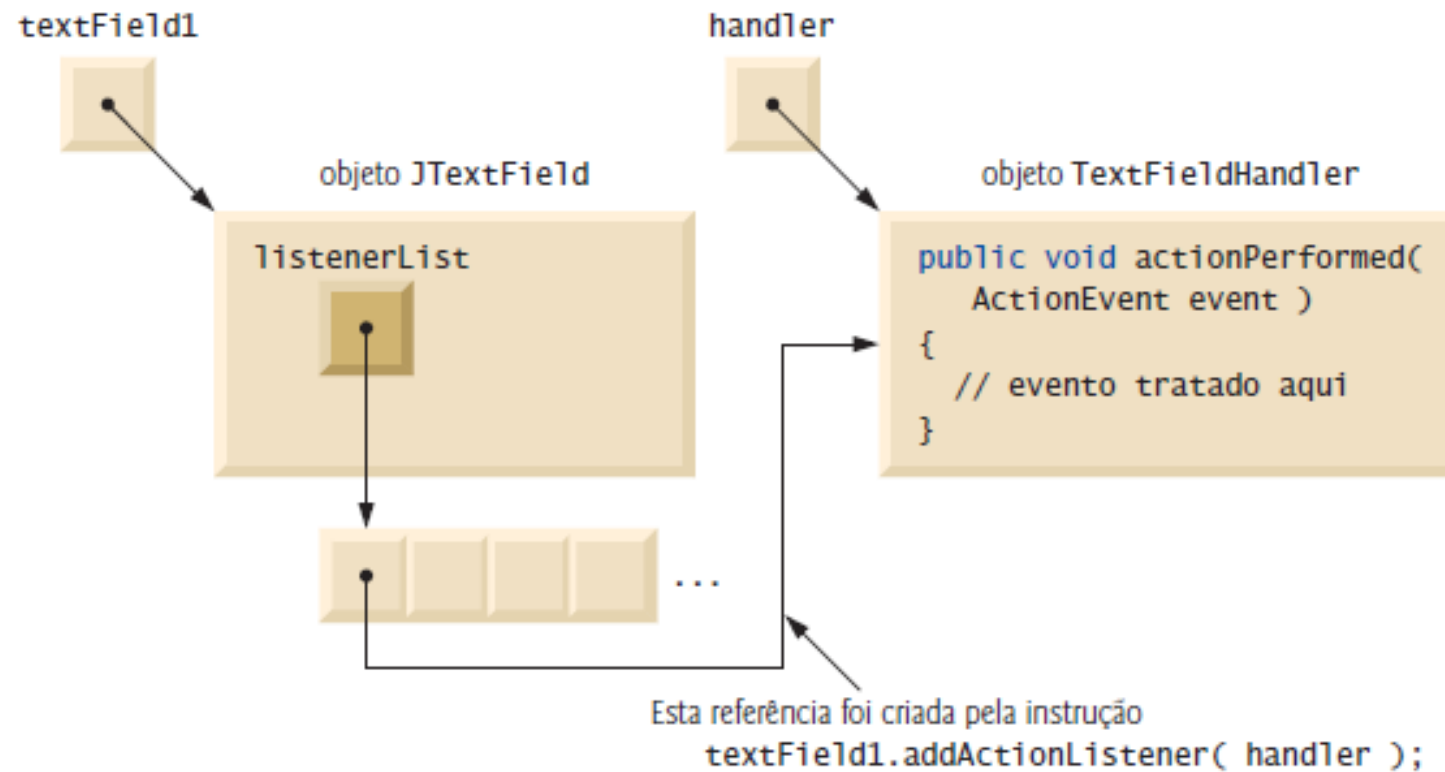


# Interfaces Listener de Eventos Comuns

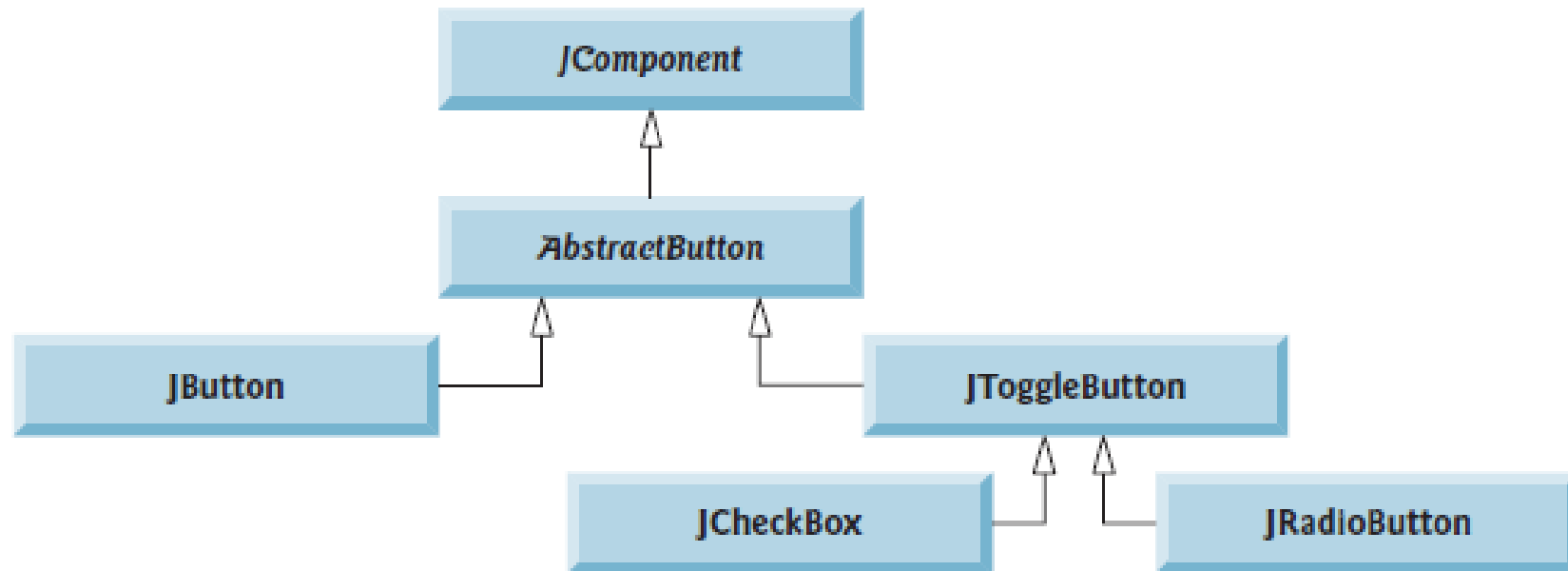


# Tratamento de Eventos

- Registro de evento para JTextField textField1.



# Hierarquia do Botão Swing





- Exemplo

```
import java.awt.FlowLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;

public class BotaoFrame extends JFrame {

    private final JButton textoJButton; // botão com texto
    private final JButton imagemJButton; // botão com um ícone

    public BotaoFrame()
    {
        super("Testando Botões");
        setLayout(new FlowLayout());

        textoJButton = new JButton("Botão texto");
        add(textoJButton);

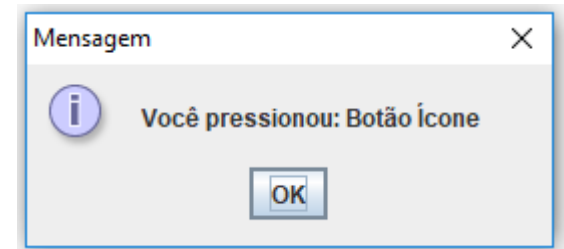
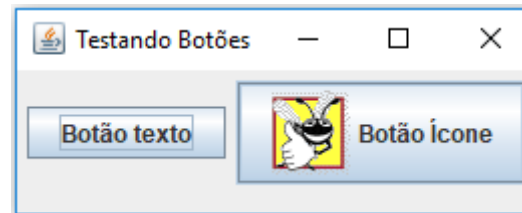
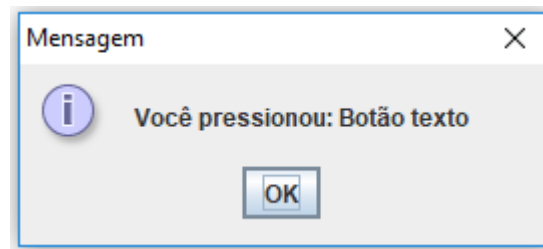
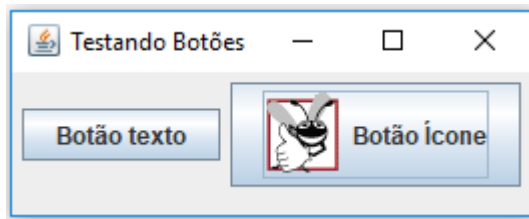
        Icon bug1 = new ImageIcon(getClass().getResource("bug1.gif"));
        Icon bug2 = new ImageIcon(getClass().getResource("bug2.gif"));
        imagemJButton = new JButton("Botão Ícone", bug1);
        imagemJButton.setRolloverIcon(bug2);
        add(imagemJButton);
    }
}
```

- Exemplo

```
import javax.swing.JFrame;

public class BotaoTeste {

    public static void main(String[] args)
    {
        BotaoFrame botaoFrame = new BotaoFrame();
        botaoFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        botaoFrame.setSize(275, 110);
        botaoFrame.setVisible(true);
    }
}
```



# CheckBox

- Exemplo:

```
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.event.ItemListener;
import java.awt.event.ItemEvent;
import javax.swing.JFrame;
import javax.swing.JTextField;
import javax.swing.JCheckBox;

public class CheckBoxFrame extends JFrame {

    private final JTextField textField;
    private final JCheckBox boldJCheckBox;
    private final JCheckBox italicJCheckBox;

    public CheckBoxFrame()
    {
        super("Teste do JCheckBox");
        setLayout(new FlowLayout());

        textField = new JTextField("Veja as alterações na fonte", 20);
        textField.setFont(new Font("Serif", Font.PLAIN, 14));
        add(textField);

        boldJCheckBox = new JCheckBox("Bold");
        italicJCheckBox = new JCheckBox("Italic");
        add(boldJCheckBox);
        add(italicJCheckBox);
    }
}
```

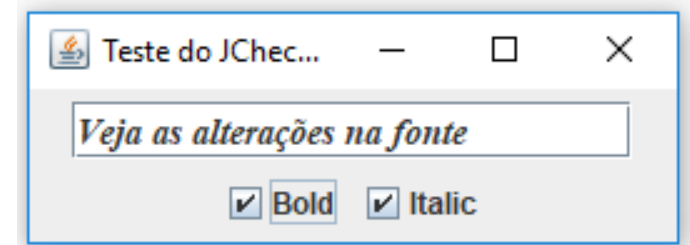
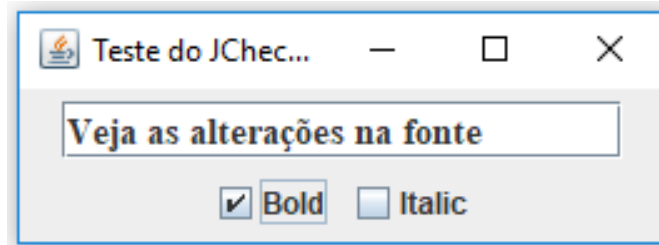
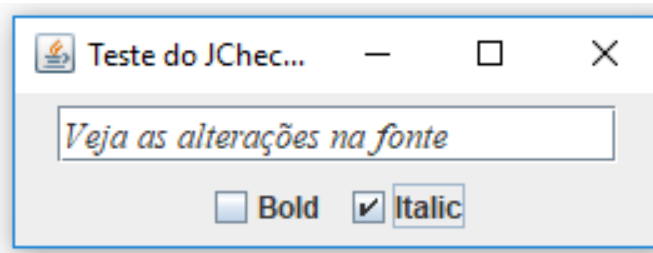
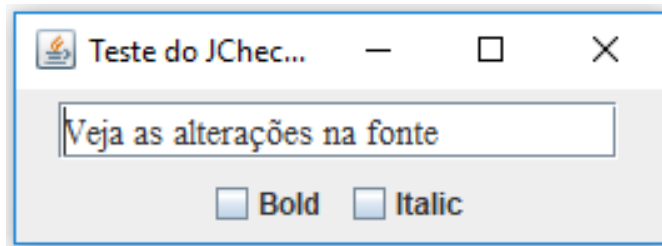
# CheckBox

- Exemplo

```
import javax.swing.JFrame;

public class CheckBoxTeste {

    public static void main(String[] args)
    {
        CheckBoxFrame checkBoxFrame = new CheckBoxFrame();
        checkBoxFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        checkBoxFrame.setSize(275, 100);
        checkBoxFrame.setVisible(true);
    }
}
```



# RadioButton

- Exemplo

```
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.event.ItemListener;
import java.awt.event.ItemEvent;
import javax.swing.JFrame;
import javax.swing.JTextField;
import javax.swing.JRadioButton;
import javax.swing.ButtonGroup;

public class RadioButtonFrame extends JFrame {

    private JTextField textField;
    private Font plainFont;
    private Font boldFont;
    private Font italicFont;
    private Font boldItalicFont;
    private JRadioButton plainJRadioButton;
    private JRadioButton boldJRadioButton;
    private JRadioButton italicJRadioButton;
    private JRadioButton boldItalicJRadioButton;
    private ButtonGroup radioGroup;
```

# RadioButton

- Exemplo

```
public RadioButtonFrame()  
{  
    super("RadioButton Test");  
    setLayout(new FlowLayout());  
  
    textField = new JTextField("Veja as alterações da fonte", 25);  
    add(textField);  
  
    plainJRadioButton = new JRadioButton("Normal", true);  
    boldJRadioButton = new JRadioButton("Negrito", false);  
    italicJRadioButton = new JRadioButton("Itálico", false);  
    boldItalicJRadioButton = new JRadioButton("Negrito/Itálico", false);  
    add(plainJRadioButton);  
    add(boldJRadioButton);  
    add(italicJRadioButton);  
    add(boldItalicJRadioButton);  
  
    radioGroup = new ButtonGroup();  
    radioGroup.add(plainJRadioButton);  
    radioGroup.add(boldJRadioButton);  
    radioGroup.add(italicJRadioButton);  
    radioGroup.add(boldItalicJRadioButton);  
  
    plainFont = new Font("Serif", Font.PLAIN, 14);  
    boldFont = new Font("Serif", Font.BOLD, 14);  
    italicFont = new Font("Serif", Font.ITALIC, 14);  
    boldItalicFont = new Font("Serif", Font.BOLD + Font.ITALIC, 14);  
}
```

# RadioButton

- Exemplo

```
private class RadioButtonHandler implements ItemListener
{
    private Font font;

    public RadioButtonHandler(Font f)
    {
        font = f;
    }

    @Override
    public void itemStateChanged(ItemEvent event)
    {
        textField.setFont(font);
    }
}
```

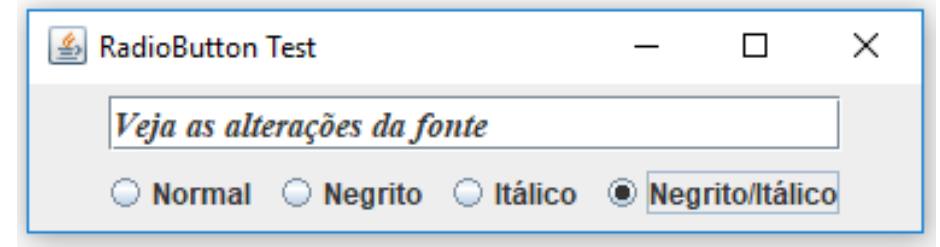
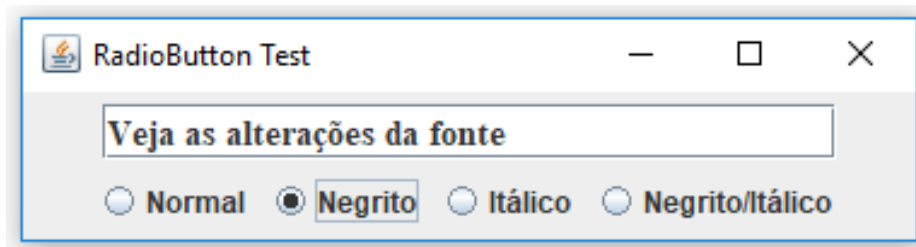
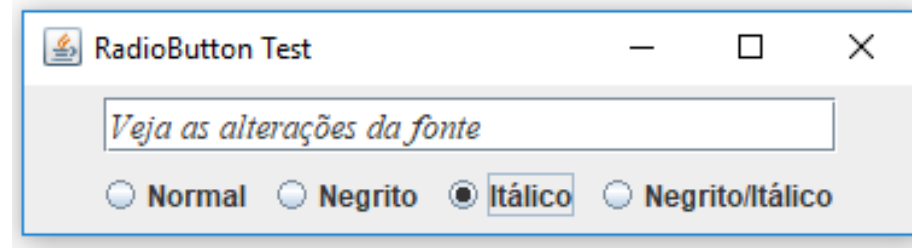
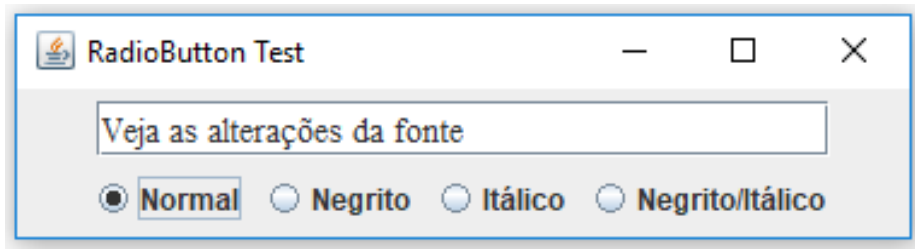
# RadioButton

- Exemplo

```
import javax.swing.JFrame;

public class RadioButtonTest {

    public static void main(String[] args)
    {
        RadioButtonFrame radioButtonFrame = new RadioButtonFrame();
        radioButtonFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        radioButtonFrame.setSize(300, 100);
        radioButtonFrame.setVisible(true);
    }
}
```





# Dúvidas?