

# ALPOO – Aplicações de Linguagem de Programação Orientada a Objetos

Prof. Ms. Gustavo Molina

[msc.gustavo.unip@gmail.com](mailto:msc.gustavo.unip@gmail.com)

Aula 02 – Swing Parte 1

# Interface Gráfica

- Os elementos básicos necessários para criar um GUI (*Graphical User Interface* - Interface Gráfica do Usuário) residem em dois pacotes:
  - `java.awt.*`: *Abstract Window Toolkit*
  - `javax.swing.*`: *Swing*



shutterstock.com · 622734824

# AWT (*Abstract Window Toolkit*)

- Abstração do sistema nativo.
- *Toolkit* gráfico e de interface.
- Pacote mais básico para se trabalhar com interfaces gráficas em Java . Este pacote possui classes para os principais componentes e containers de uma interface, tais como botões e janelas , além de permitir um tratamento bem simplificado de eventos.

# Componentes AWT

✓ O pacote AWT disponibiliza 8 componentes básicos de interface:

❖ Button.

❖ Label.

❖ TextField.

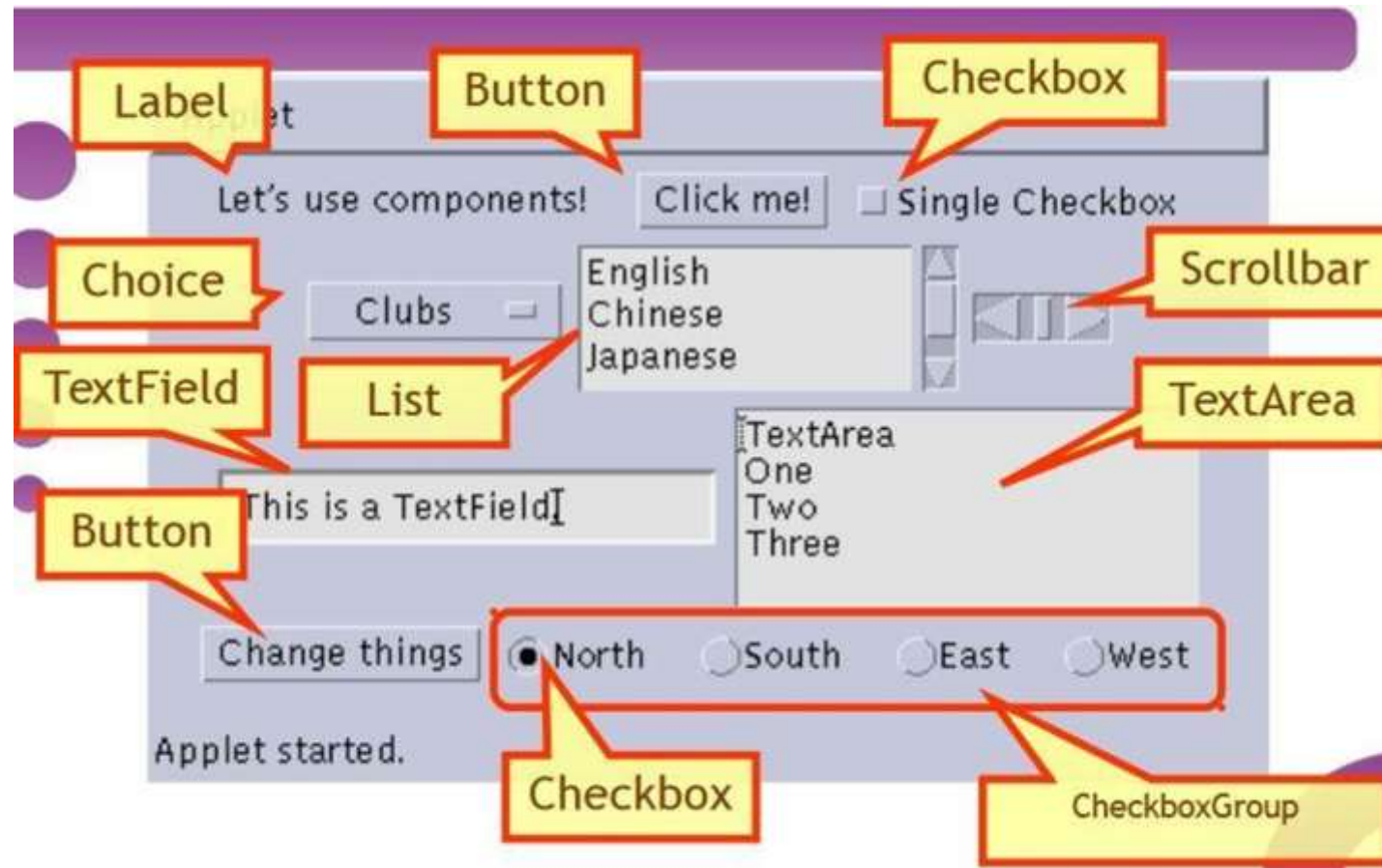
❖ TextArea.

❖ Checkbox.

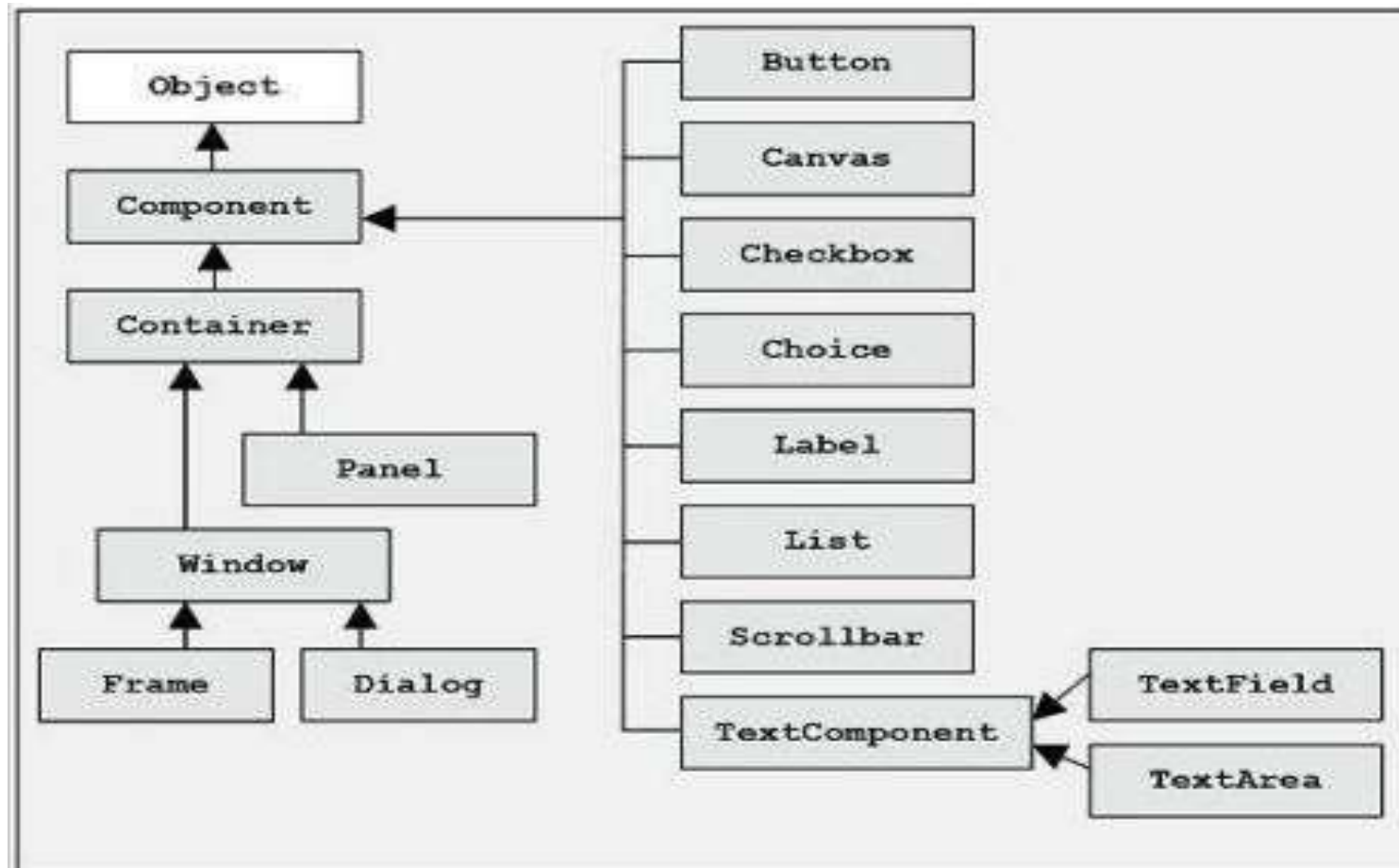
❖ Choice.

❖ List.

❖ Scrollbar.



# Hierarquia de Classes



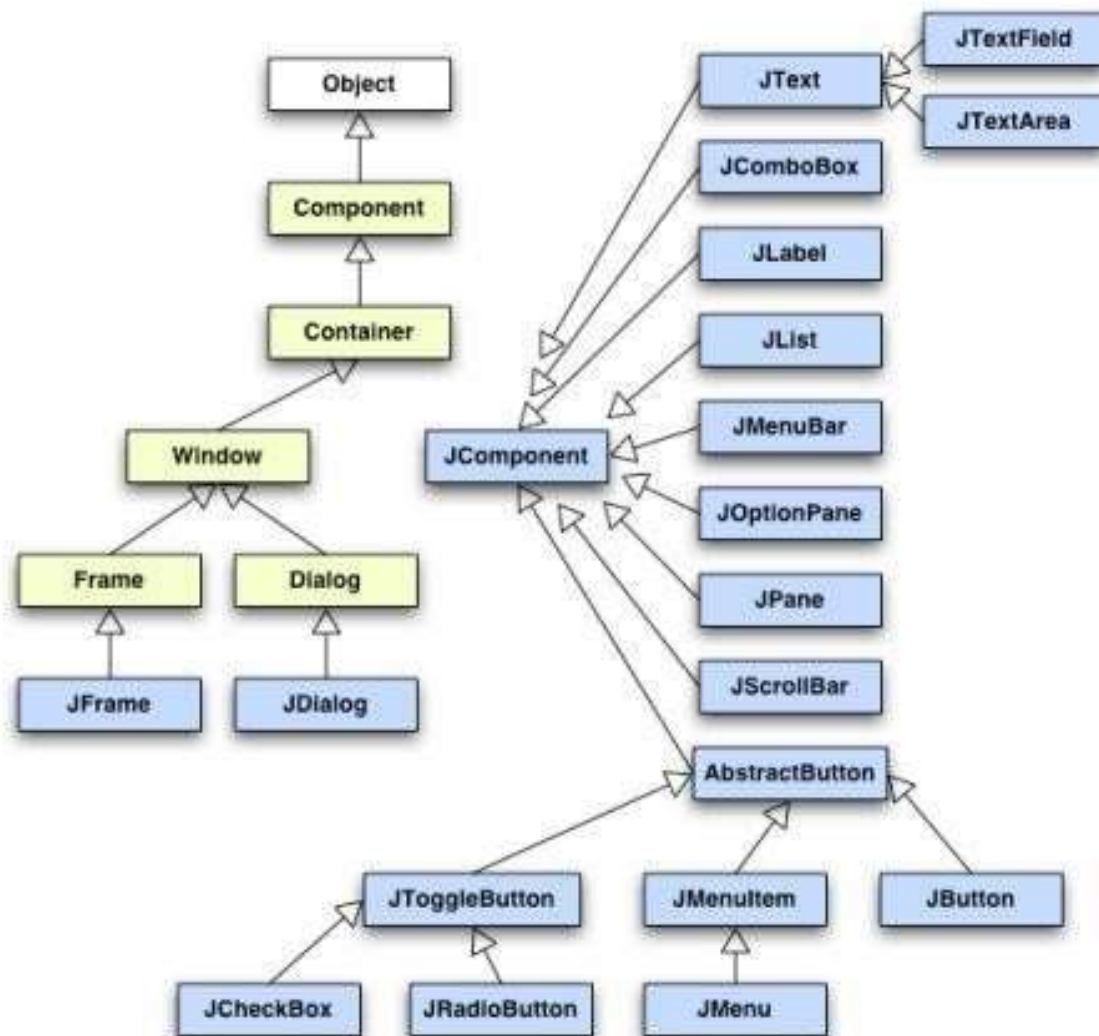
# Projeto Swing

- Projeto Swing é parte do JFC, que implementa um novo conjunto de elementos de interface com o usuário com mecanismo look- and-feel embutido.
- É baseado no JDK 1.1 *Lightweight UI Framework*, um ambiente que tornou as interfaces menos pesadas e mais adaptáveis.
- Os componentes do Swing são implementados sem código nativo, logo temos **maior portabilidade e maior consistência de uso entre plataformas.**

# Swing - Conceitos

- O pacote Swing **não** é um substituto do pacote *AWT*.
- O *Swing* é visto como uma camada disposta sobre o *AWT* e que utiliza internamente os componentes da *AWT*. Diferentemente da *AWT*, onde alguns componentes gráficos utilizavam a capacidade de renderização da interfaces gráficas em que o aplicativo estava sendo executado, os componentes do Swing são todos escritos em Java puro.
- Um componente do pacote Swing é reconhecido pela letra *J* antecedendo o nome do mesmo componente no pacote *AWT*.

# AWT e Swing

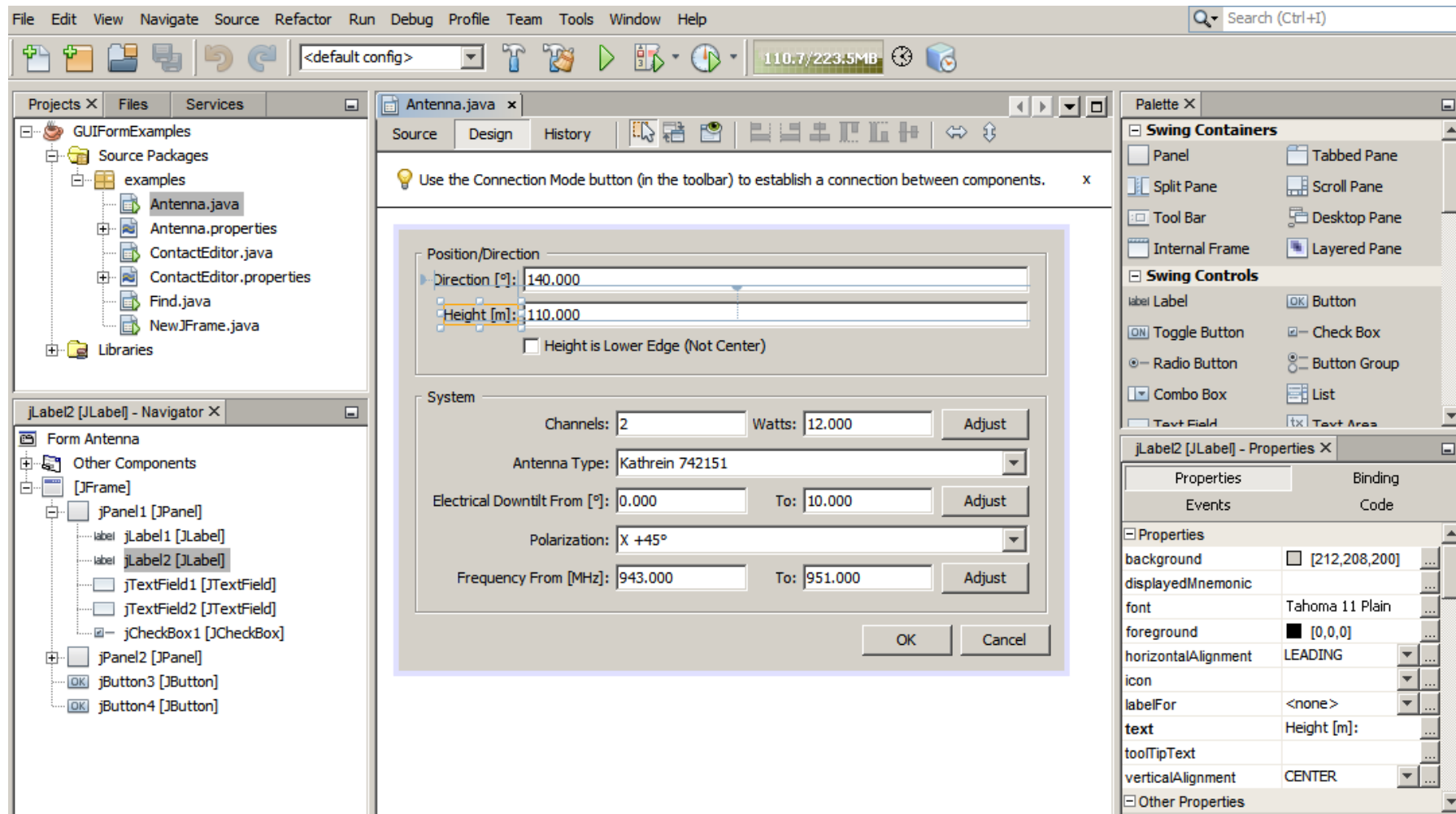




# Containers e Componentes

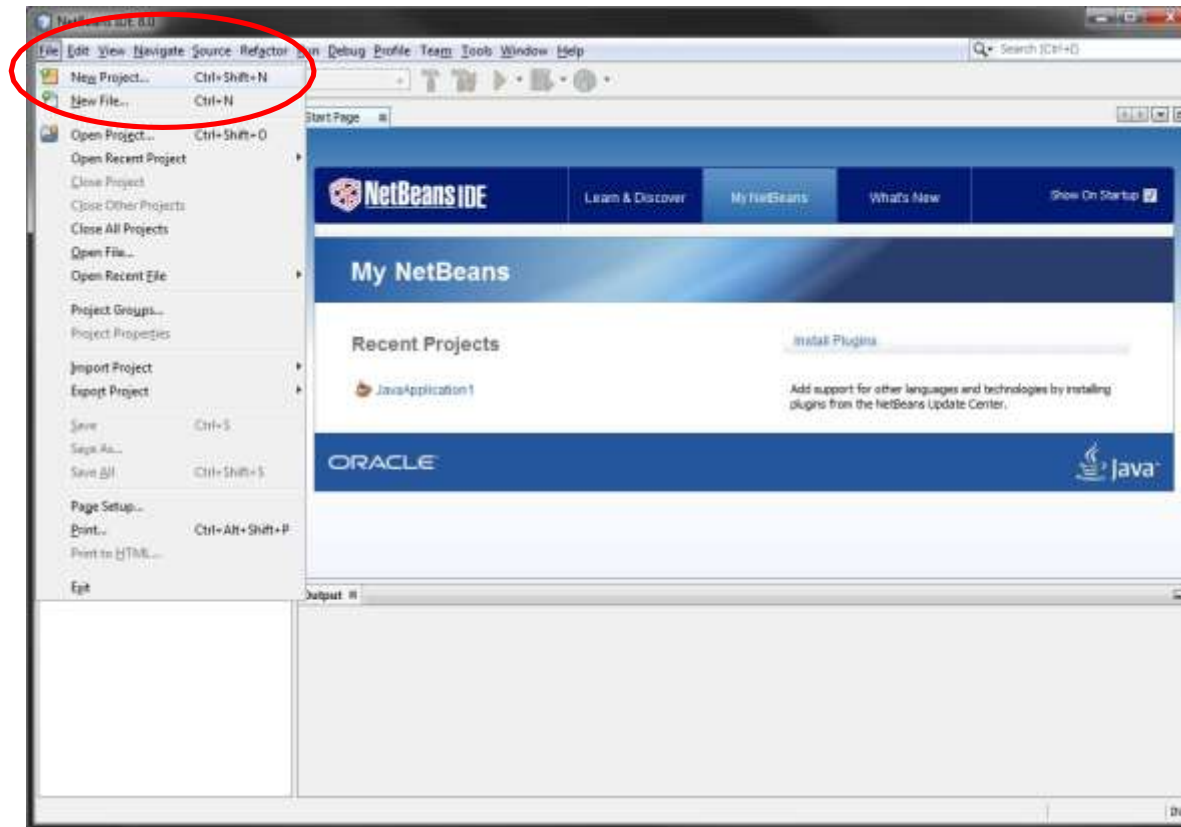
- Uma interface gráfica em Java é baseada em dois elementos:
  - **Containers**: servem para agrupar e exibir outros componentes.
  - **Componentes**: botões, labels, scrollbars, etc.
- Todo programa Java que ofereça uma interface possui pelo menos um container.
- Uma janela de nível mais alto (que não fica contida dentro de outra janela) é um Frame ou, na versão Swing, um **JFrame**;
- O JFrame é um container. Isso significa que ele pode conter outros componentes de interface com o usuário.

# Netbeans GUI Builder



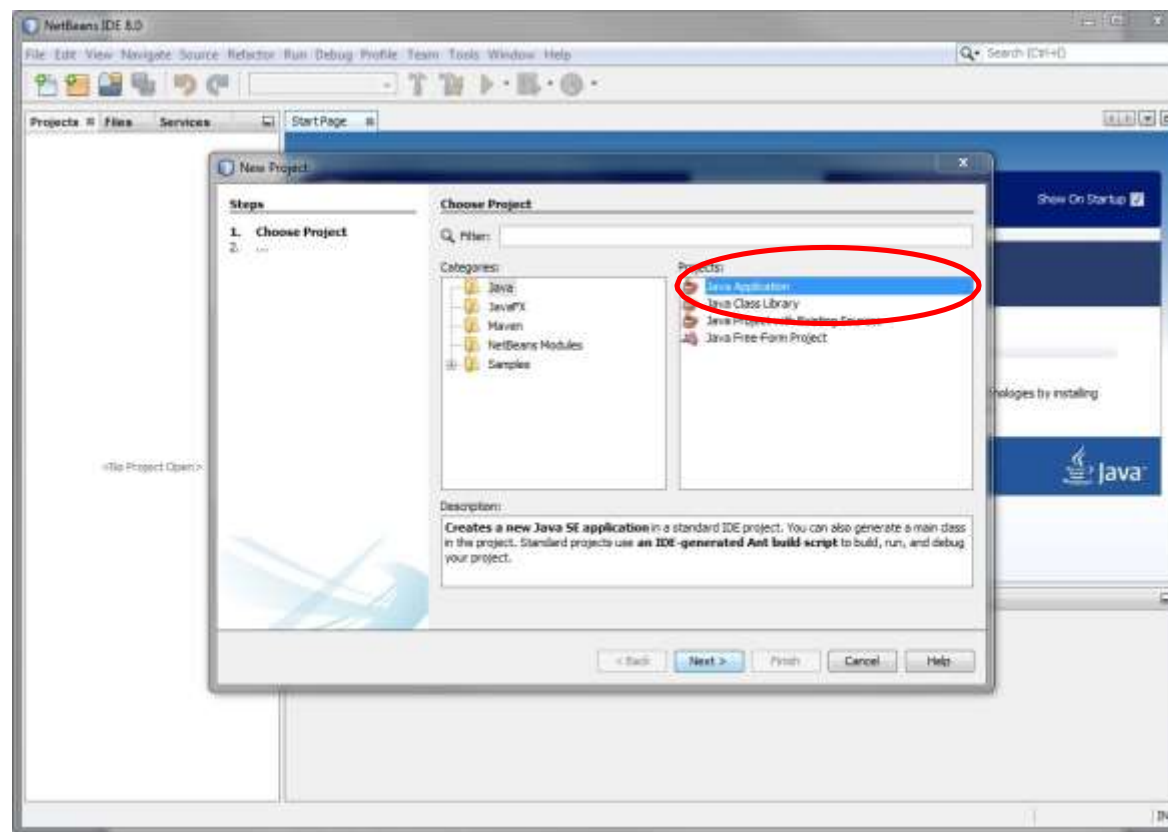
# Netbeans GUI Builder – Criando Projeto

1) Acesse o menu File -> New Project



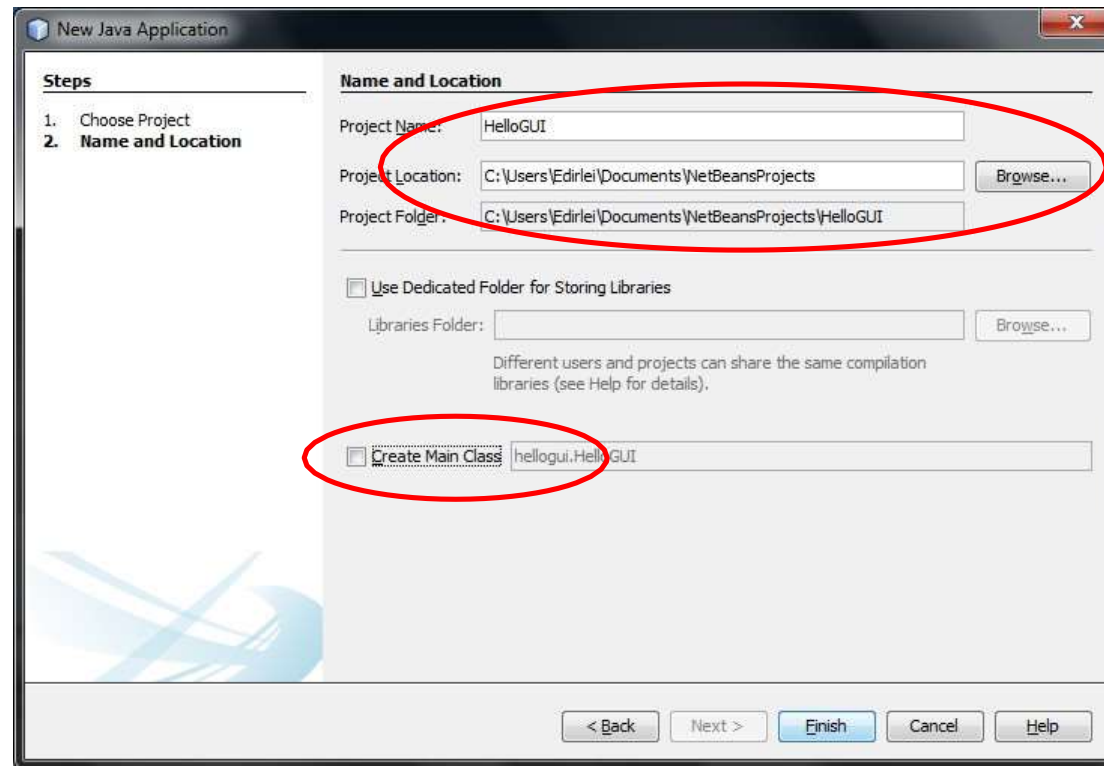
# Netbeans GUI Builder – Criando Projeto

- 2) Selecione o tipo de projeto “Java Application” e em seguida clique em “Next”:



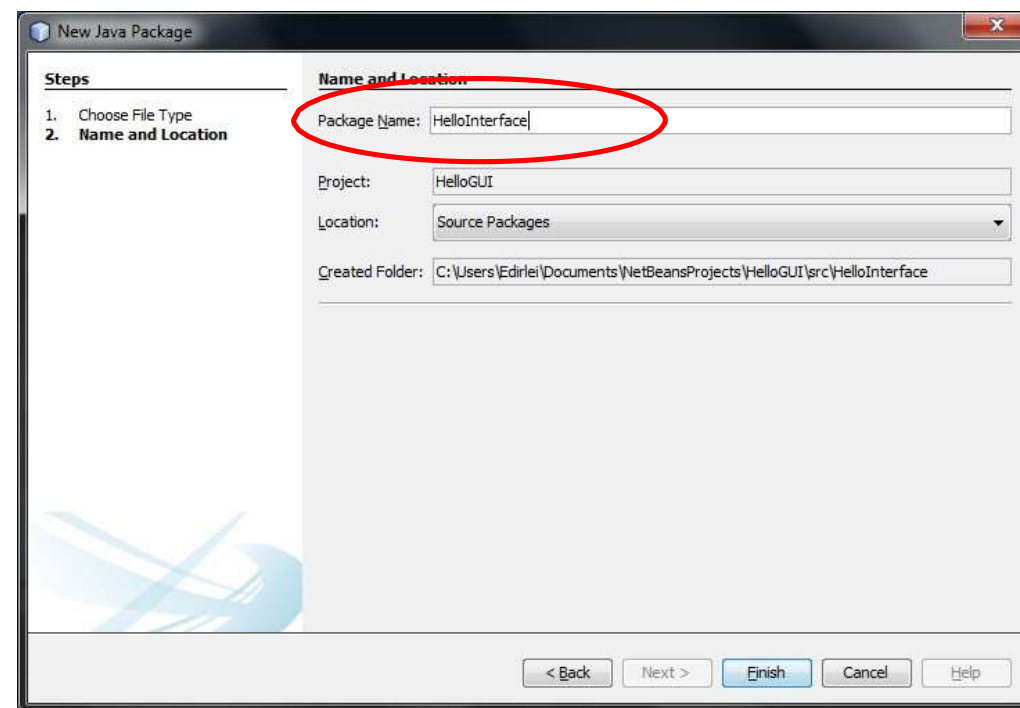
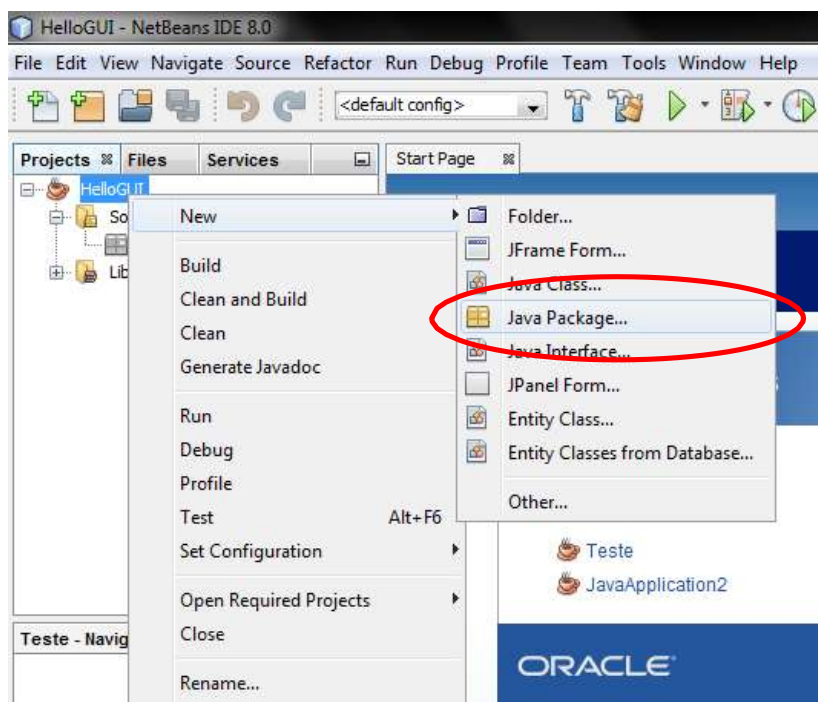
# Netbeans GUI Builder – Criando Projeto

- 3) De um nome para o projeto, selecione o local onde ele será salvo e desmarque a opção “Create Main Class”. Em seguida clique em “Finish”:



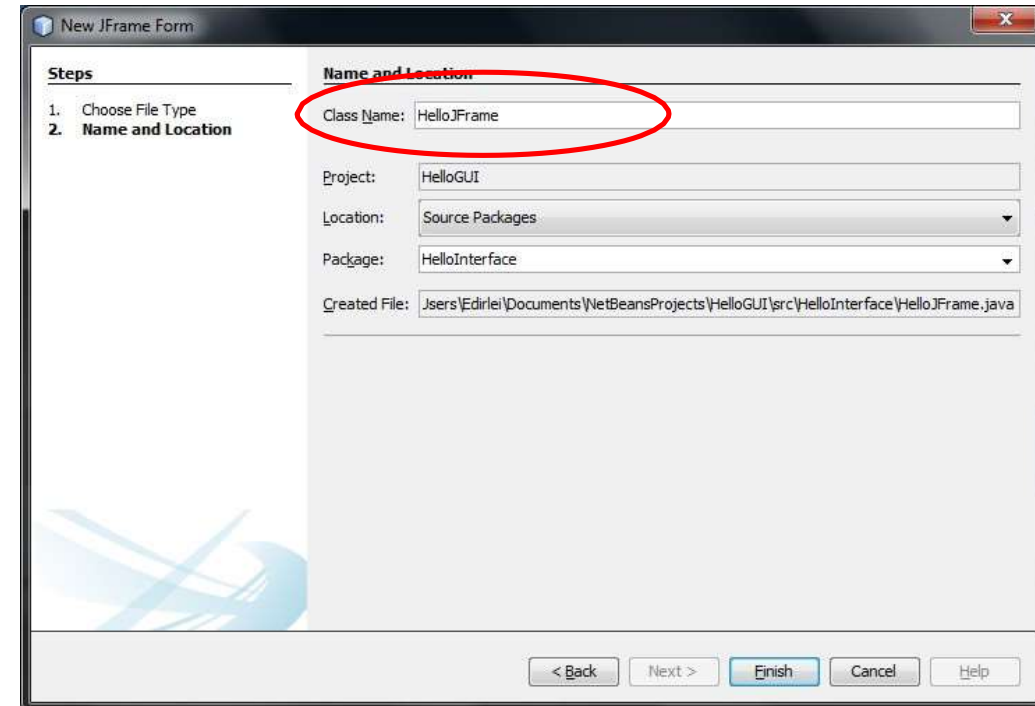
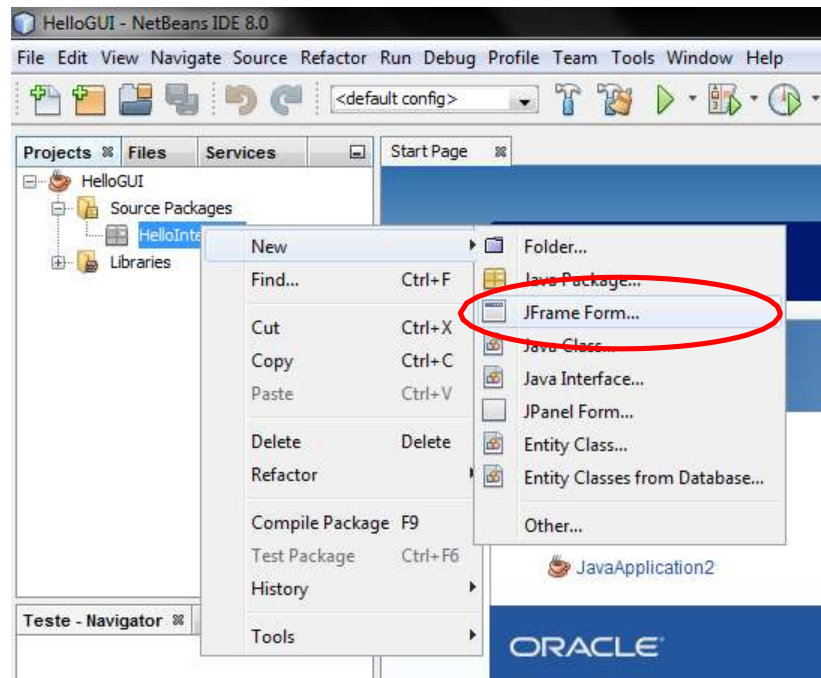
# Netbeans GUI Builder – Criando Projeto

4) Crie um novo “Java Package” no projeto:

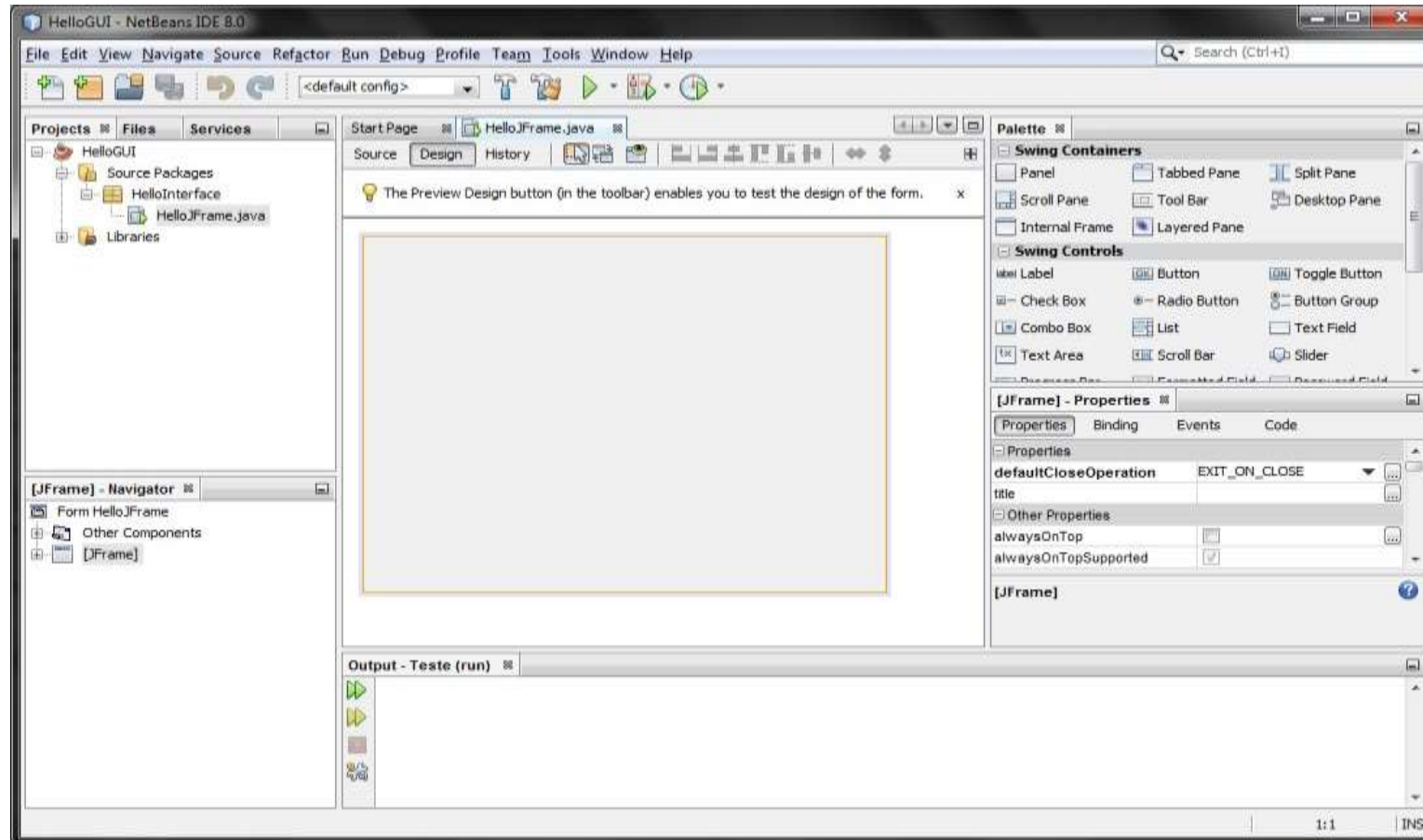


# Netbeans GUI Builder – Criando Projeto

5) Crie um novo “JFrame Form”:



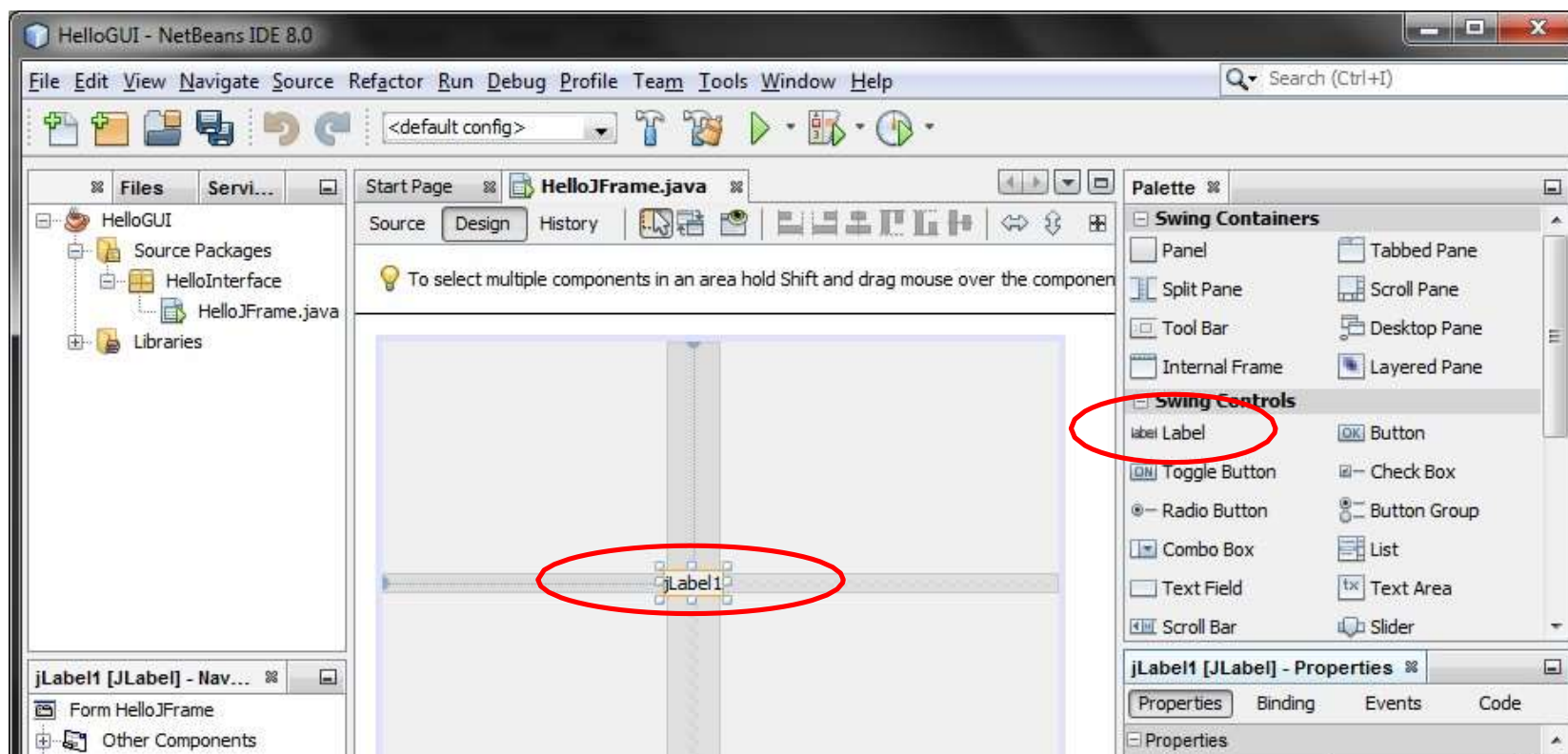
# Netbeans GUI Builder – Criando Projeto





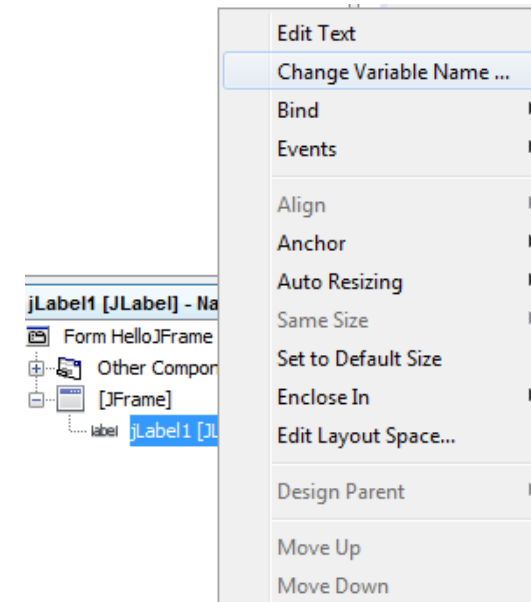
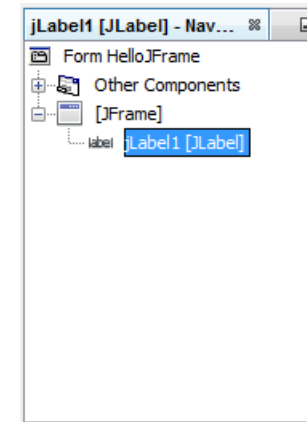
# Componentes Básicos - Label

- Componente para exibição de texto não-editável ou ícones.



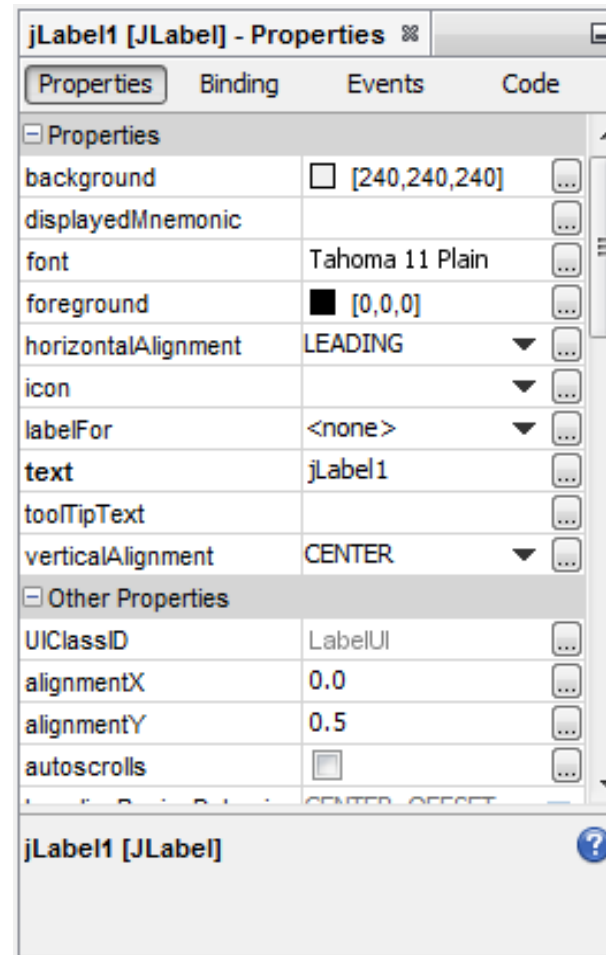
# Componentes Básicos - Label

- Containers e componentes e estrutura da interface gráfica;
- Todos os elementos que fazem parte da interface gráfica são **objetos**;
- Todos os objetos possuem um nome (variable name) que pode (e deve!) ser alterado.



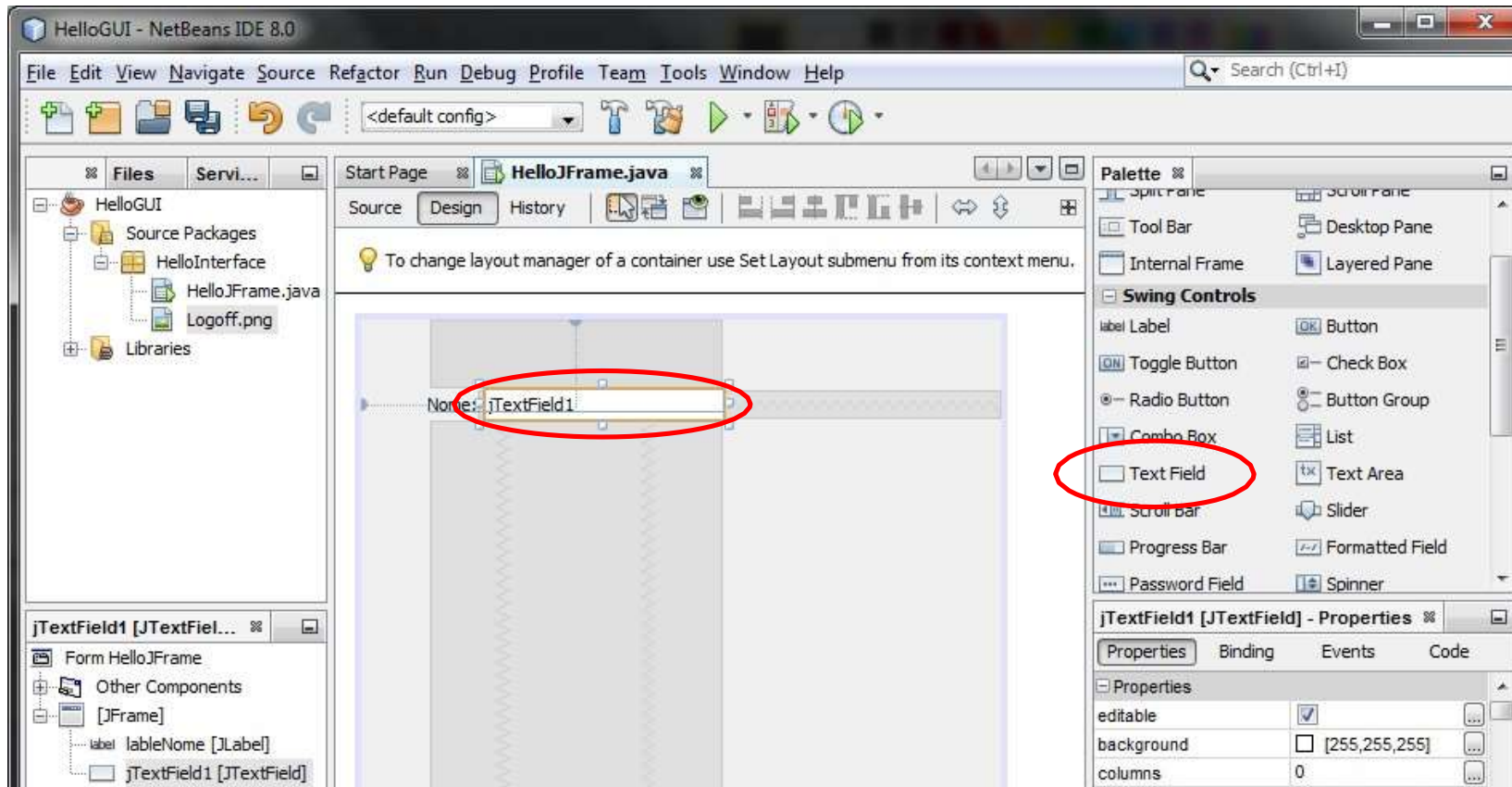
# Componentes Básicos - Label

- Principais Propriedades (JLabel):
  - text;
  - foreground;
  - background;
  - font;
  - icon;
  - toolTipText;
  - border;



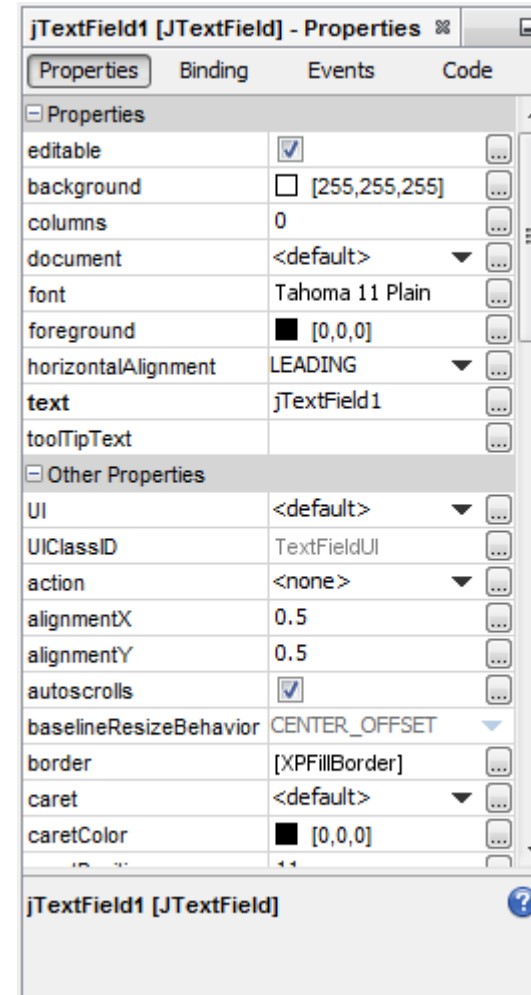
# Componentes Básicos – TextField

- Componente para entrada, edição e exibição de texto.



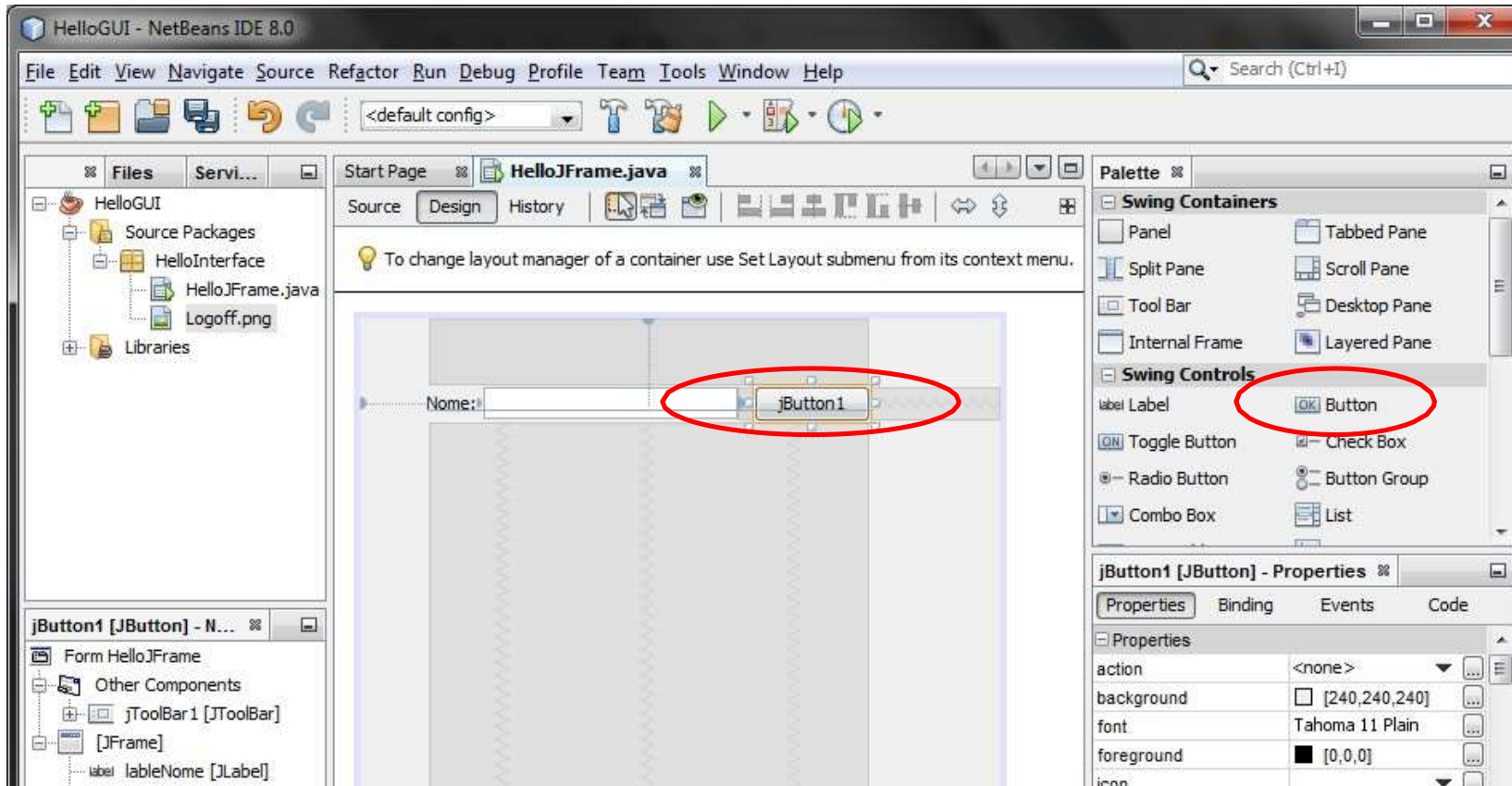
# Componentes Básicos – TextField

- Principais Propriedades (JTextField):
  - text;
  - editable;
  - foreground;
  - background;
  - font;
  - toolTipText;
  - border;
  - enabled;



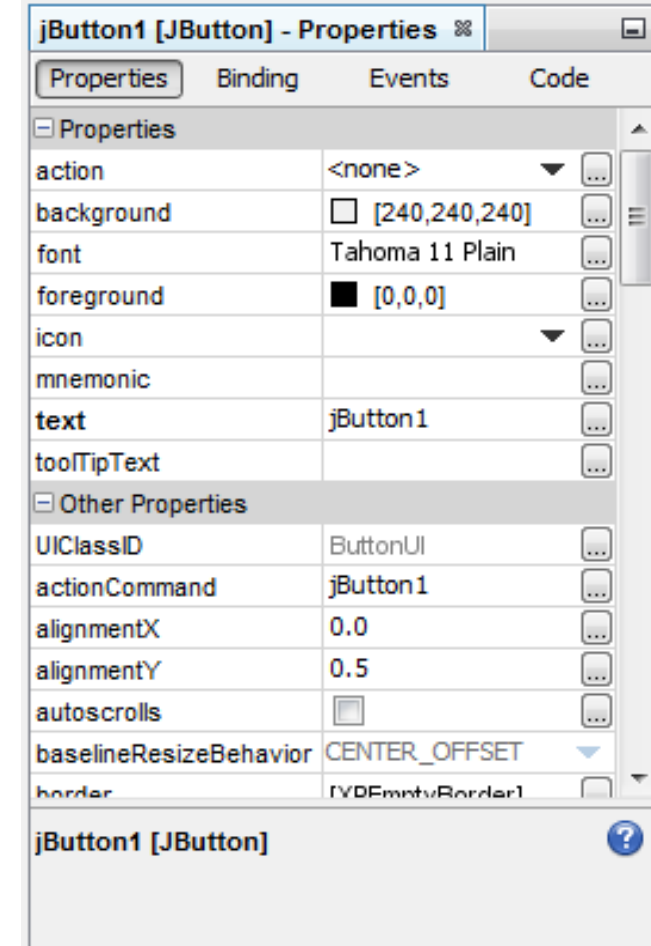
# Componentes Básicos – Button

- Componente que representa um botão.



# Componentes Básicos – Button

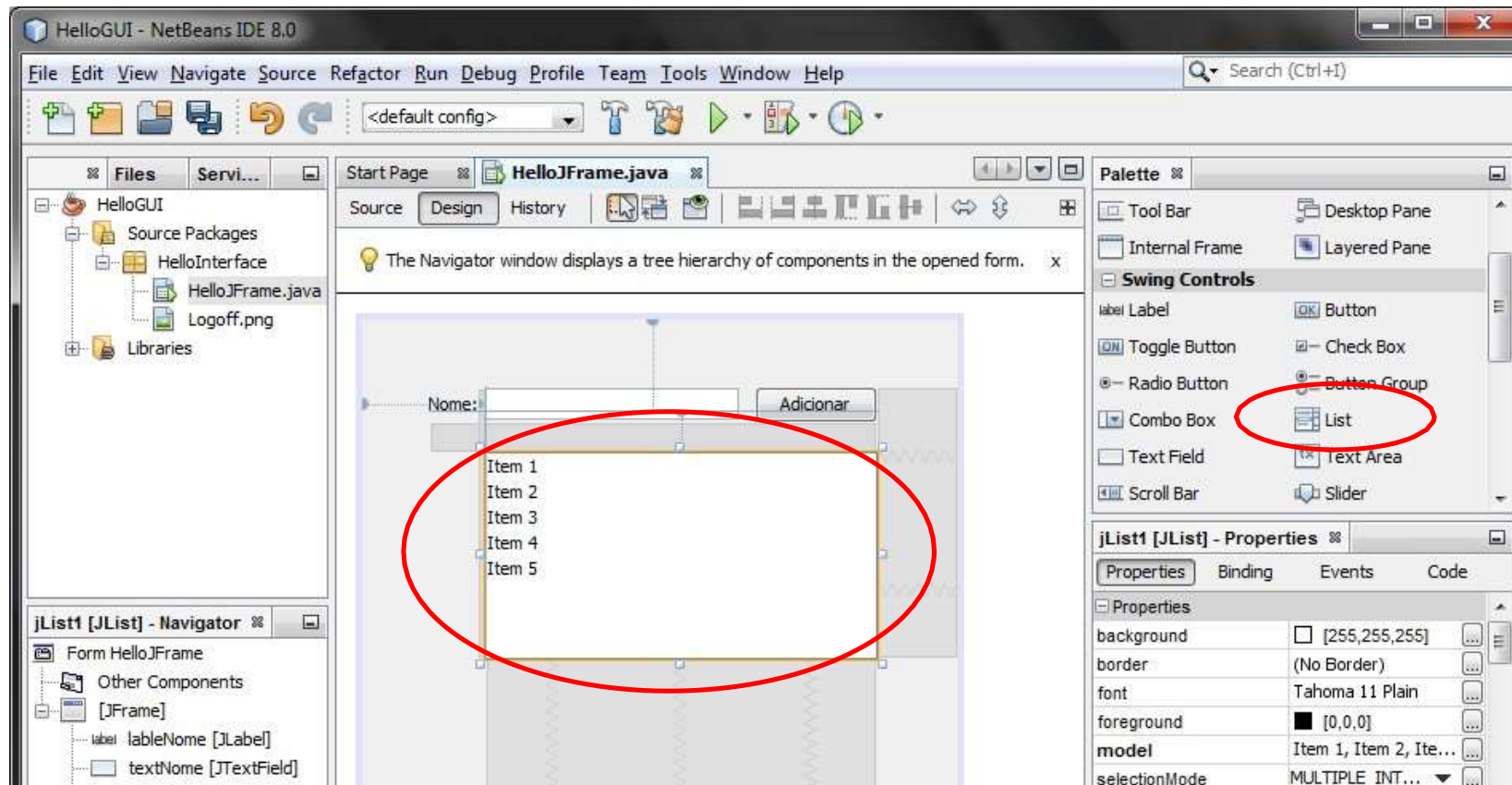
- Principais Propriedades (JButton):
  - text;
  - foreground;
  - background;
  - font;
  - icon;
  - tooltipText;
  - border;
  - enabled;





# Componentes Básicos – List

- Componente que exibe uma lista de itens e permite que o usuário possa seleciona-los.

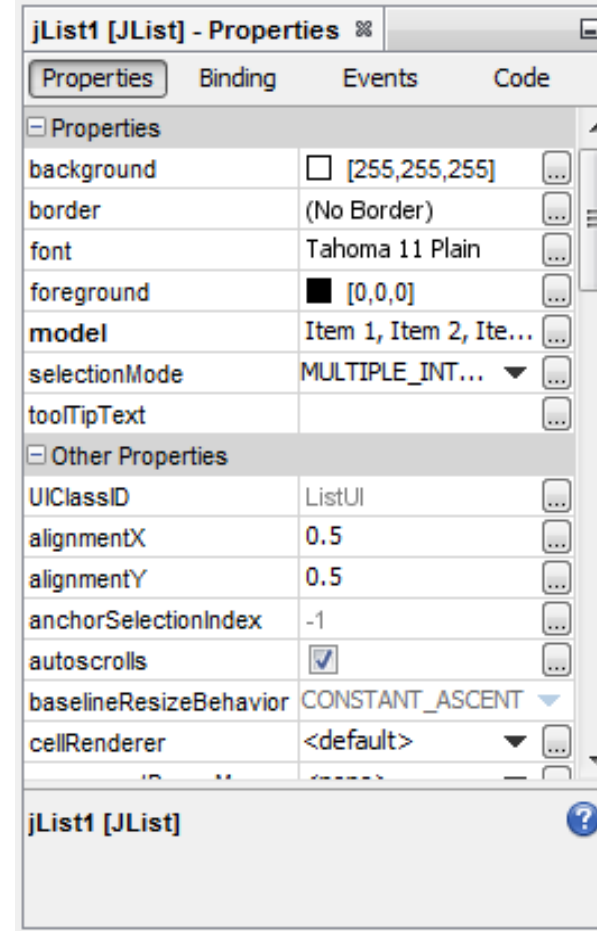




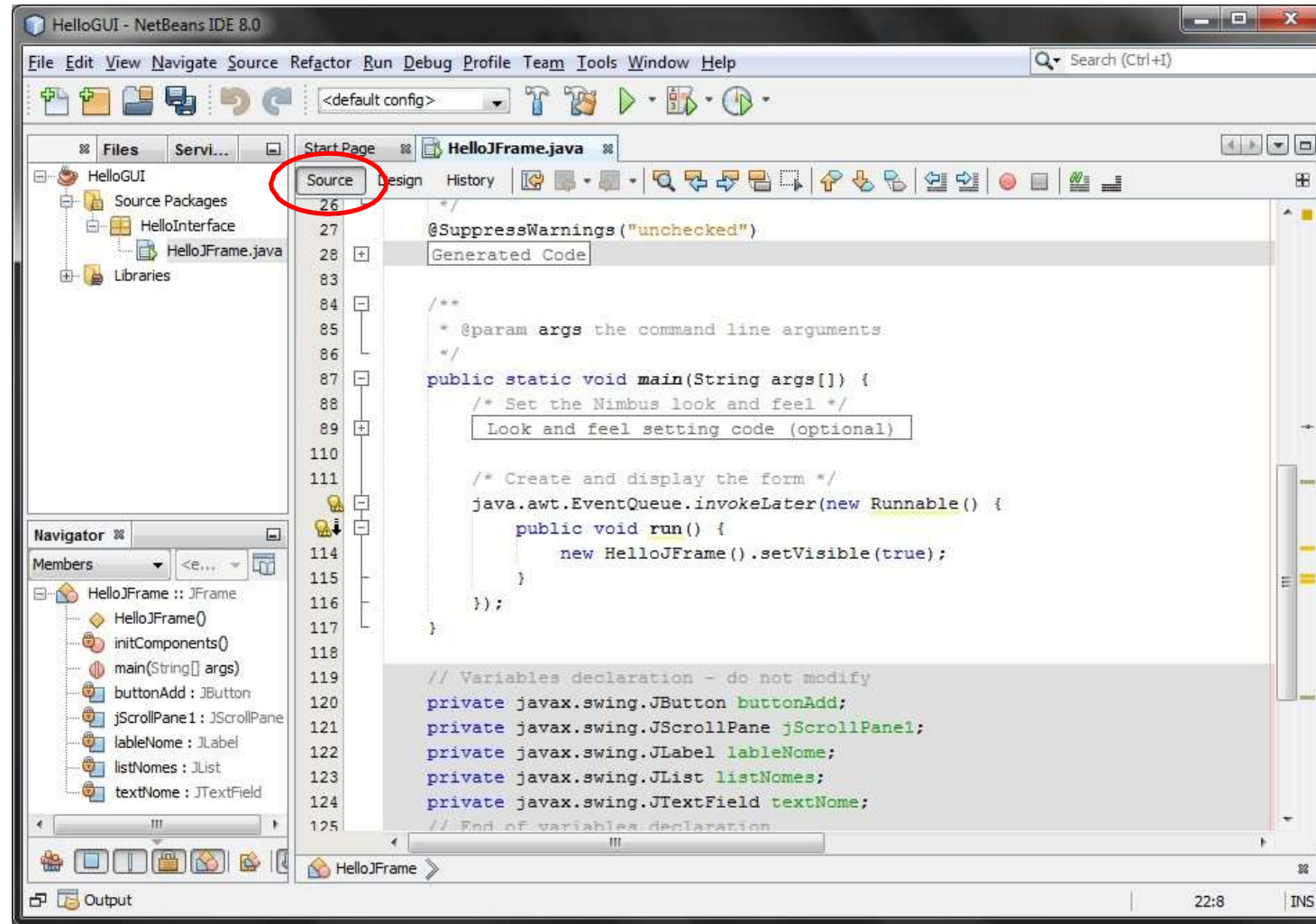
# Componentes Básicos – List

- Principais Propriedades (JList):

- model;
- selectionMode;
- selectedIndex;
- visibleRowCount;
- foreground;
- background;
- font;
- toolTipText;
- border;
- enabled;

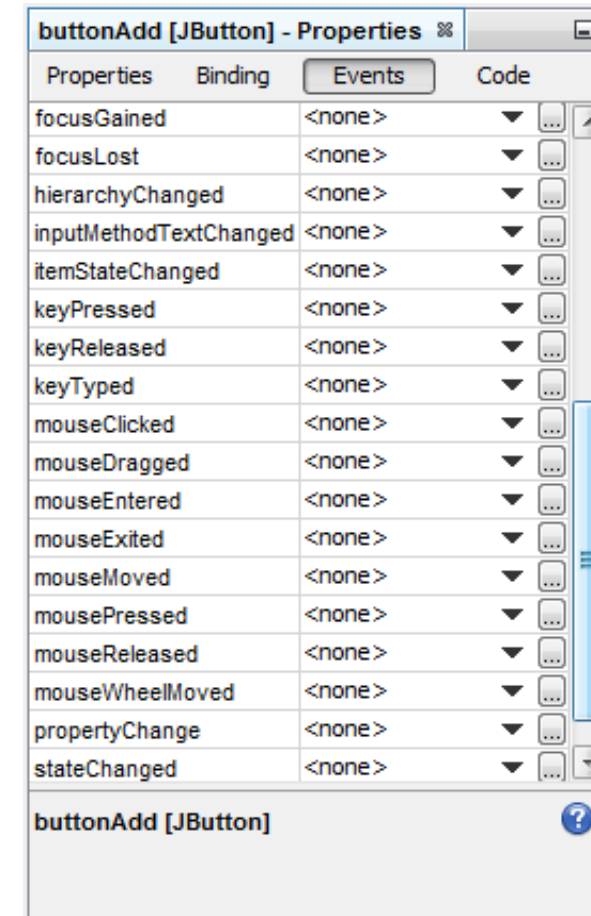


# Netbeans GUI Builder – Código Gerado



# Eventos – Button

- Principais Eventos (JButton):
  - actionPerformed;
  - mouseClicked;
  - mousePressed;
  - mouseReleased;
  - mouseMoved;
  - mouseEntered
  - mouseExited;
  - focusGained;
  - focusLost;



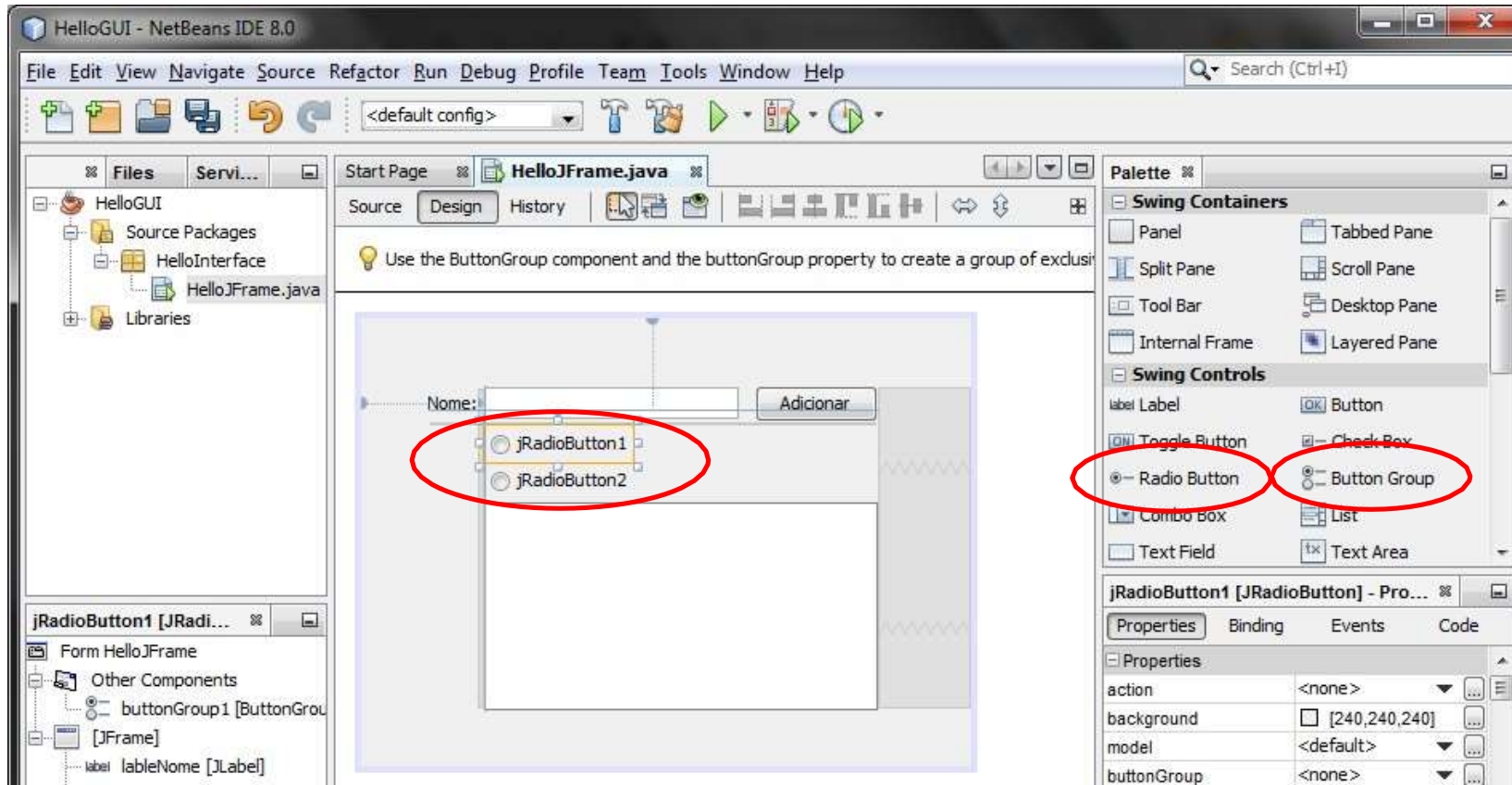
# Evento actionPerformed – Button

- **Exemplo** – Mostrar mensagem com o conteúdo do TextField:

```
private void buttonAddActionPerformed(java.awt.event.ActionEvent evt)
{
    JOptionPane.showMessageDialog(this, "Hello " + textNome.getText());
}
```

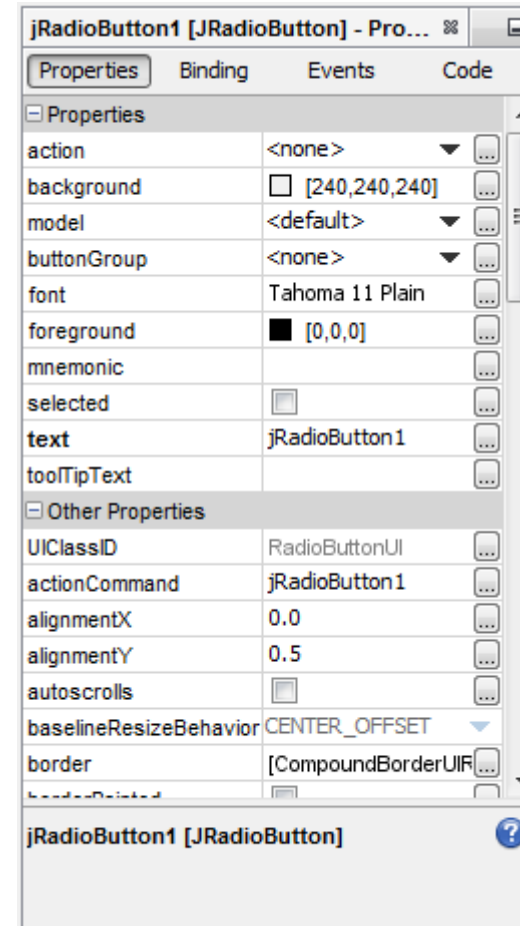
# Componentes Básicos – RadioButton e ButtonGroup

- Componentes que permitem a seleção de opções.



# Componentes Básicos – RadioButton

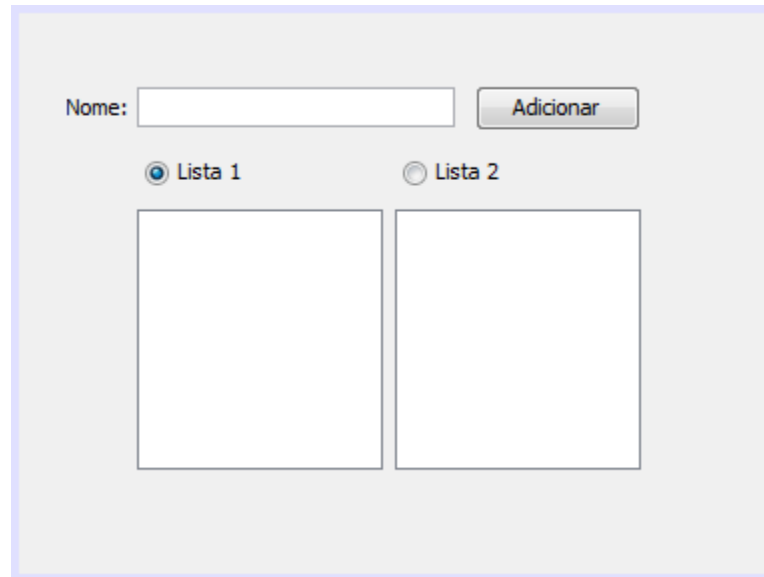
- Principais Propriedades (JRadioButton):
  - text;
  - buttonGroup;
  - Selected;
  - foreground;
  - background;
  - font;
  - icon;
  - toolTipText;
  - border;
  - enabled;



# Usando o RadioButton

- **Exemplo** – Adicionar o conteúdo do TextField em duas List de acordo com a opção selecionada no RadioButton.

– **Interface:**



The interface is a light gray rectangular box. At the top, it contains a label 'Nome:' followed by a white text input field. To the right of the input field is a gray button with the text 'Adicionar'. Below the input field, there are two radio buttons. The first is selected (indicated by a blue dot) and is labeled 'Lista 1'. The second is unselected (indicated by a gray dot) and is labeled 'Lista 2'. Below each radio button is a large, empty white rectangular box, representing a list or container for the added content.

# Usando o RadioButton

- **Exemplo** – Adicionar o conteúdo do TextField nas Lists de acordo com a opção selecionada no RadioButton.

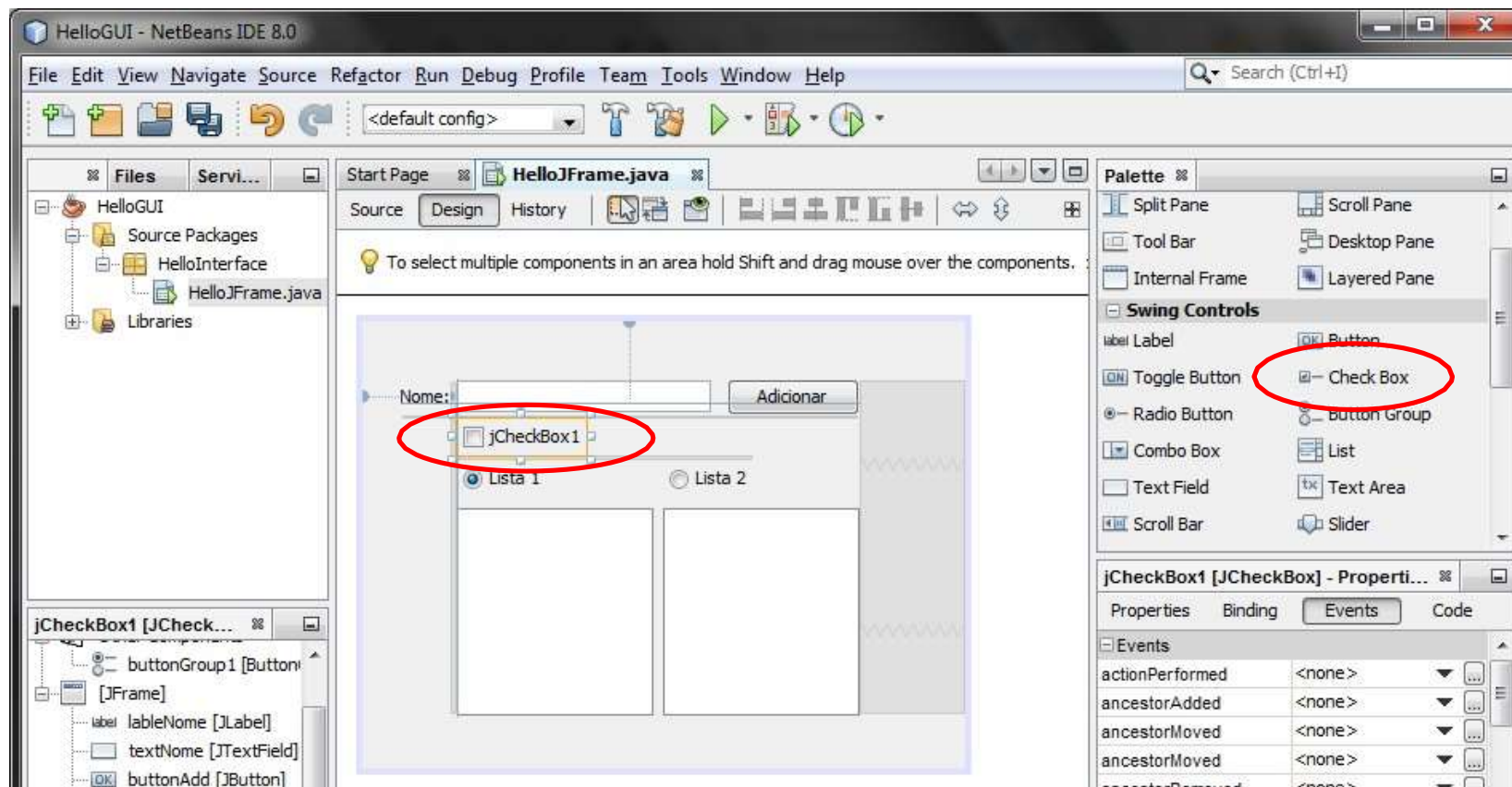
```
...  
  
private void buttonAddActionPerformed(java.awt.event.ActionEvent evt)  
{  
    if (radioBt1.isSelected())  
        listModel1.addElement(textNome.getText());  
    else if (radioBt2.isSelected())  
        listModel2.addElement(textNome.getText());  
  
    textNome.setText("");  
}  
  
...
```





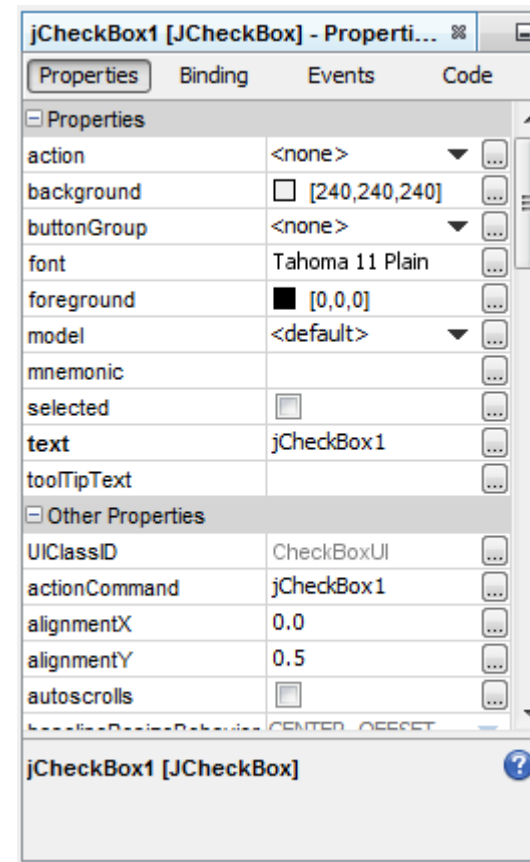
# Componentes Básicos – CheckBox

- Componente que permite a seleção de opções (marcado ou não marcado).



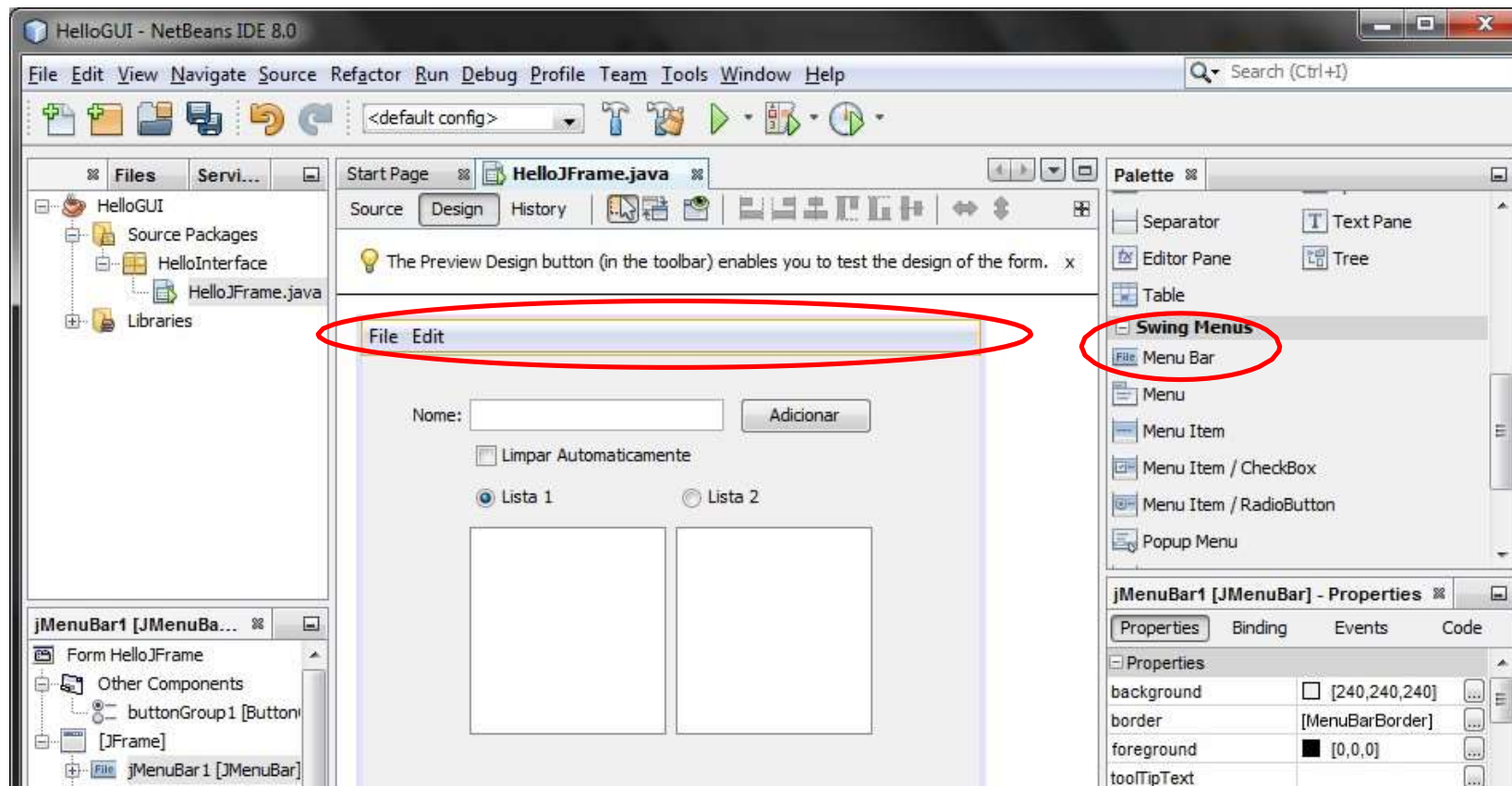
# Componentes Básicos – CheckBox

- Principais Propriedades (JCheckBox):
  - text;
  - buttonGroup;
  - Selected;
  - foreground;
  - background;
  - font;
  - icon;
  - toolTipText;
  - border;
  - enabled;



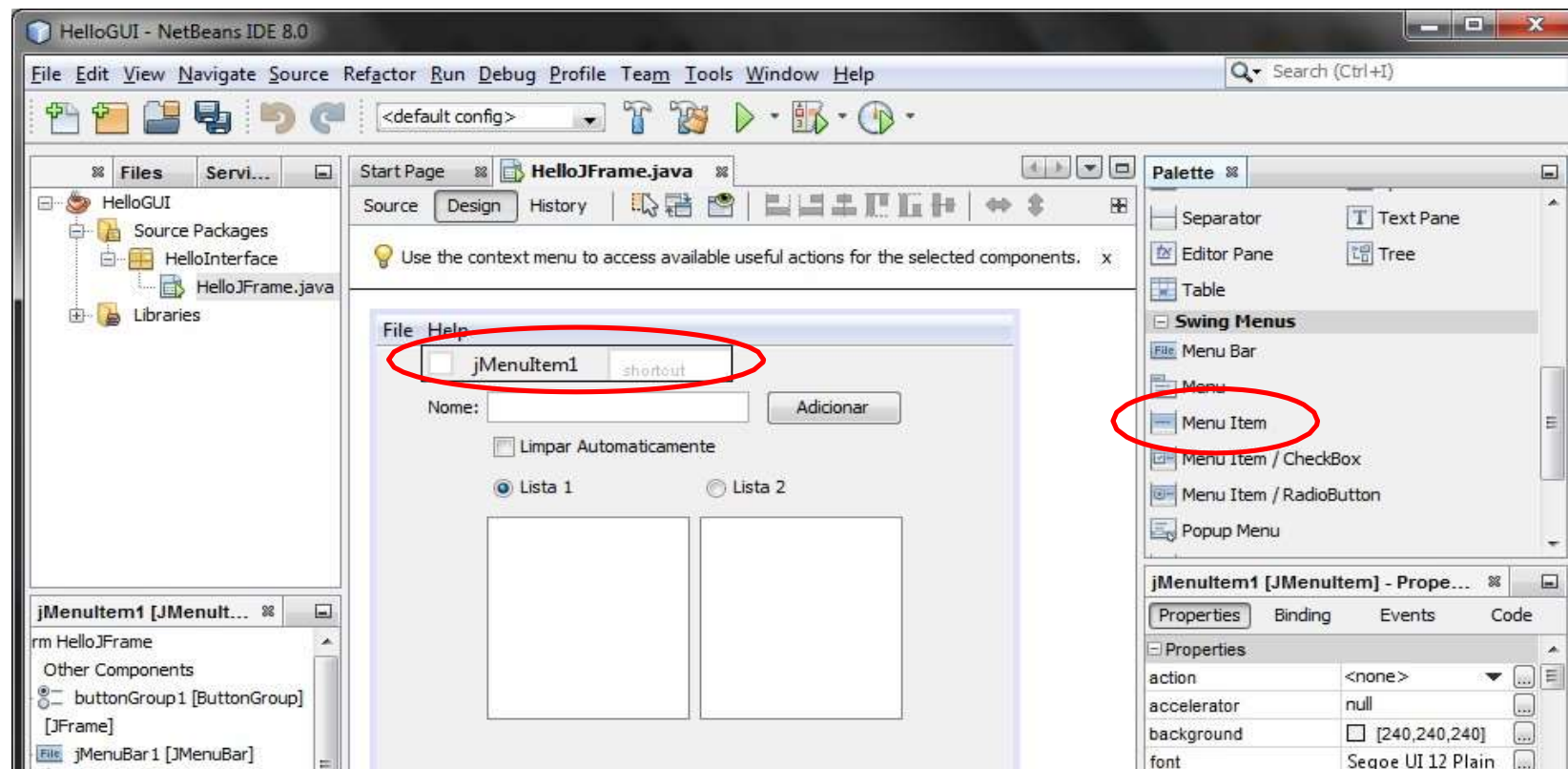
# Componentes Básicos – MenuBar

- Componente que permite a criação de uma barra de menu.



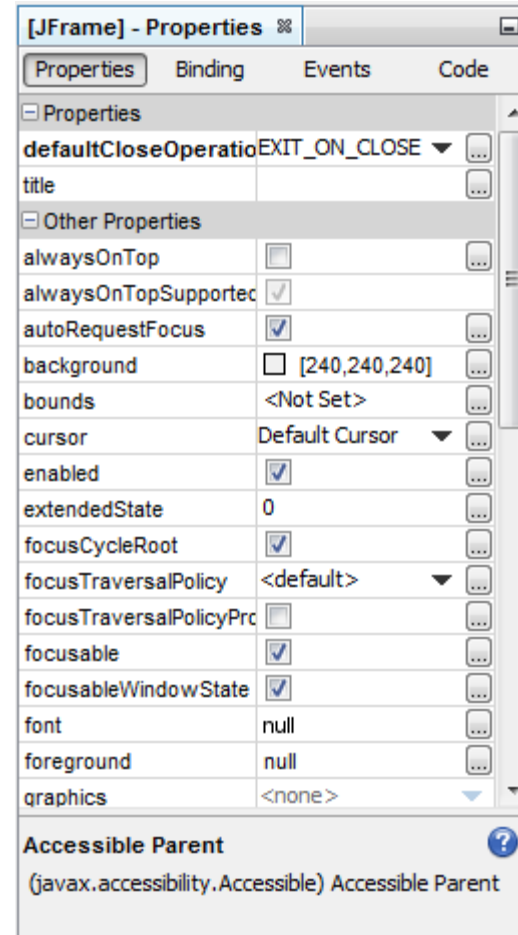
# Componentes Básicos – MenuItem

- Componente que permite a criação de itens para a barra de menu.



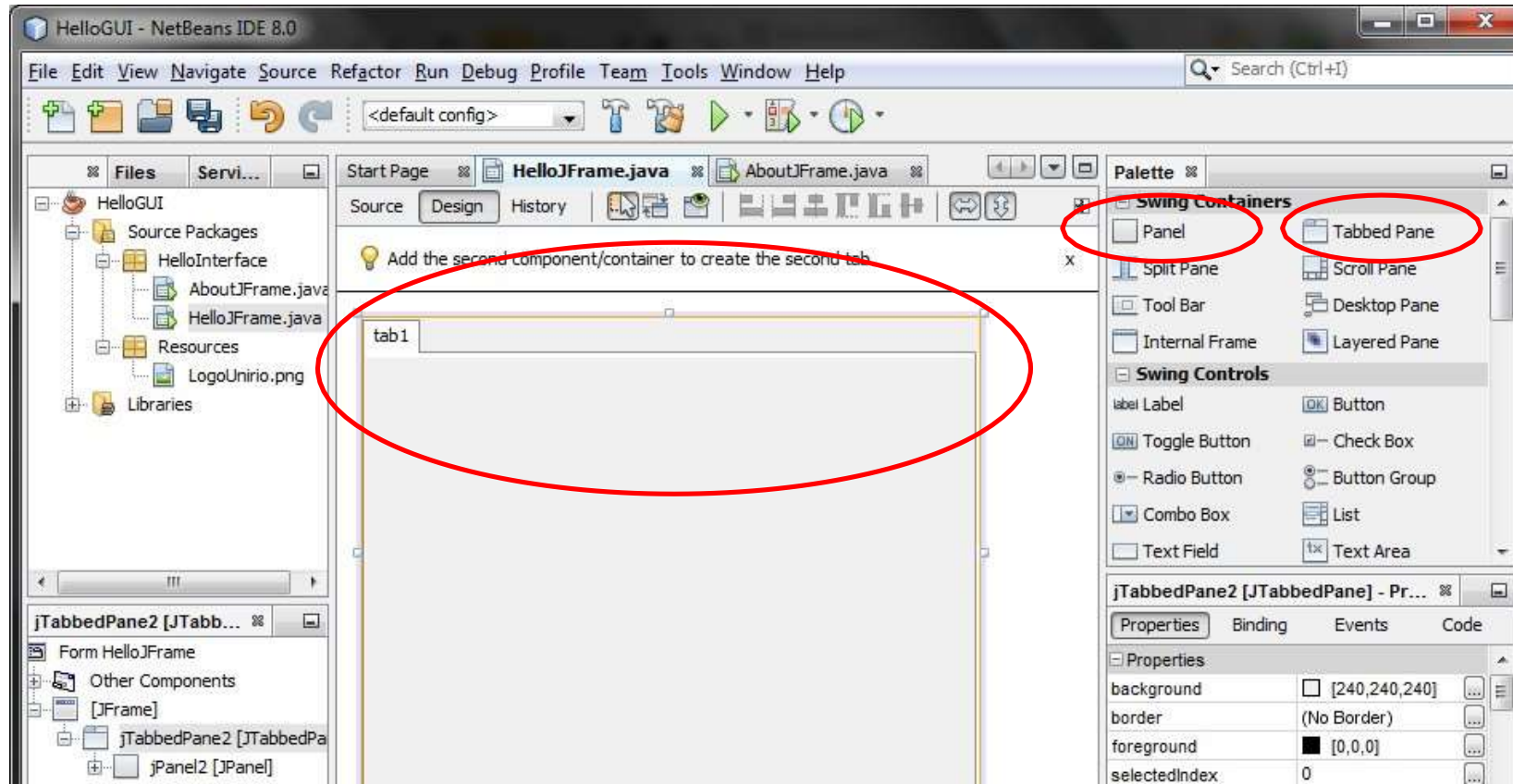
# Componentes Básicos – Frame

- Principais Propriedades (JFrame):
  - defaultCloseOperation;
  - title;
  - background;
  - alwaysOnTop;
  - iconImage;
  - resizable;
  - undecorated;
  - type;



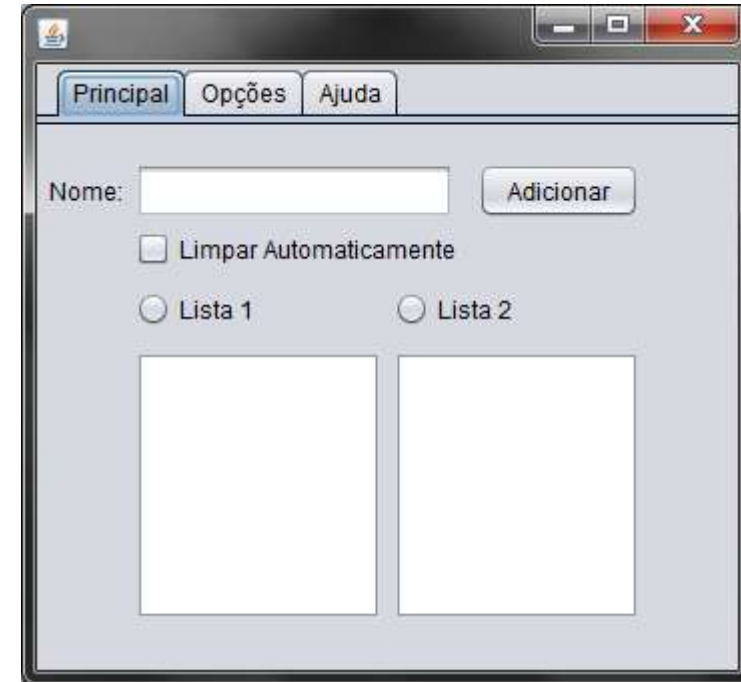
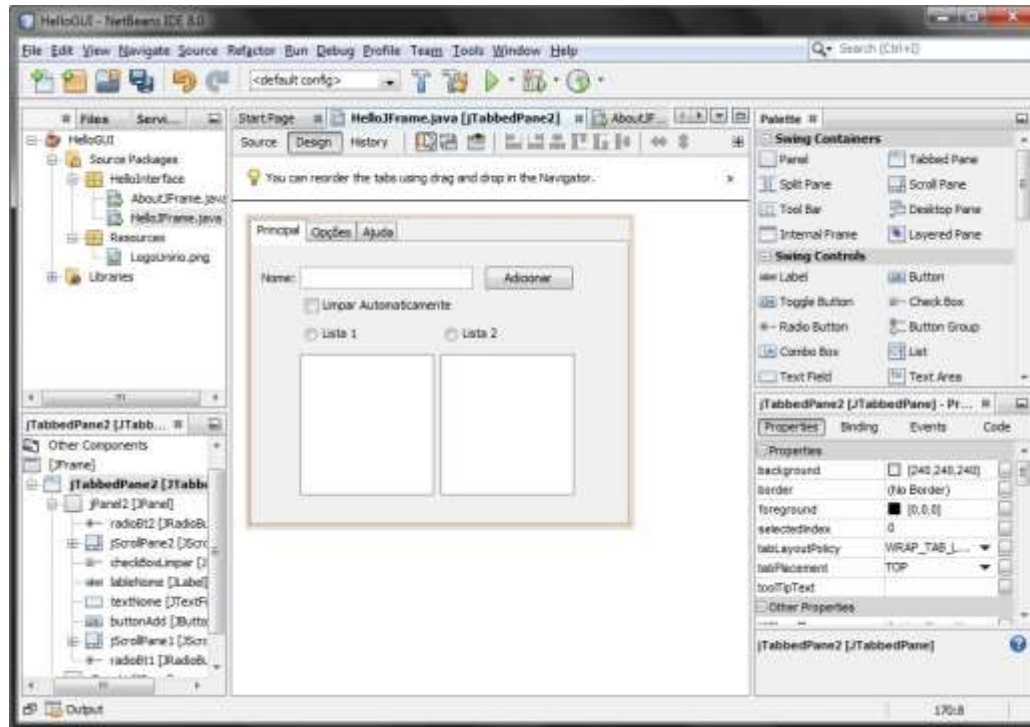
# Componentes Básicos – Tabbed Pane

- Componente que permite a criação de telas com abas.





# Componentes Básicos – Tabbed Pane



# Gerenciadores de Layout

- Java é portátil, o que dispensa aos desenvolvedores de preocupações com aspectos de hardware. Interfaces gráficas, entretanto, possuem dependência dos dispositivos nos quais serão exibidas – a resolução, cores e suporte a eventos são exemplos de aspectos relevantes em um projeto que envolve interface gráfica com o usuário (GUI).
- Na maioria das linguagens, o programador define previamente a aparência da GUI, incluindo o tamanho e posicionamento dos componentes, e este aspecto é fixo e imutável a menos que haja uma mudança no código.



# Gerenciadores de Layout

- Imagine um programa codificado para rodar em um monitor com resolução de 800x600 sendo executado em apenas 640x400. Provavelmente isto acarretará problemas de posicionamento dos componentes ou eventualmente a perda de visibilidade destes.
- Linguagens compiladas como C++ ou Delphi exigem que o programador saiba de antemão as características de hardware para os quais ele está programando, ou então adotar estratégias de verificação destas características no momento da abertura ou instalação dos programas – o que agrega complexidade ao algoritmo e reduz a portabilidade dos programas.

# Gerenciadores de Layout

Em Java não é tratado o posicionamento e dimensionamento dos componentes gráficos rigidamente, mas por meio de processos independentes chamados de gerenciadores de layout.

Vantagens:

Portabilidade: código gerado em SO Windows, em alta resolução, é executado sem perda de forma ou função em SO's, como Linux ou Macintosh – ou mesmo em dispositivos especiais, como Palms ou telefones celulares.

# Gerenciadores de Layout

- Todos os gerenciadores de layout implementam a interface **LayoutManager** que faz parte do pacote **java.awt**.
- O método **setLayout** da classe **Container** aceita um objeto que implementa a interface **LayoutManager** como um argumento.
- As três maneiras básicas de organizar componentes em uma GUI:
  - Posicionamento absoluto
  - Gerenciadores de layout
  - Programação visual em uma IDE
- Cada **Container** individual pode ter apenas um gerenciador de layout, mas vários **Containers** no mesmo aplicativo podem utilizar cada um gerenciador de layout.

# Containers

Os principais gerenciadores de layout para containers AWT:

- FlowLayout
- BorderLayout
- GridLayout
- CardLayout
- GridBagLayout



# FlowLayout

- Os componentes são colocados em um Container da esquerda para a direita na ordem em que são adicionados no Container
- Quando a borda do Container é alcançada, os componentes continuarão a ser exibidos na próxima linha
- A classe FlowLayout permite aos componentes GUI ser alinhados à esquerda, centralizados (padrão) e alinhados à direita.

# Exemplo FlowLayout



# Exemplo FlowLayout

```
import java.awt.*;

public class FlowLayoutTest extends Frame {

    FlowLayoutTest() {
        setSize(400, 350);
        setLayout(new FlowLayout());
        add(new Button("Um") );
        add(new Button("Dois"));
        add(new Button("Três"));
    }

    public static void main(String[] args) {
        FlowLayoutTest flowLayoutTest = new FlowLayoutTest();
        flowLayoutTest.setVisible(true);
    }
}
```

# BorderLayout

Considerar a interface como uma moldura dividida em cinco partes:

NORTH - borda superior

SOUTH - borda inferior

EAST - borda esquerda

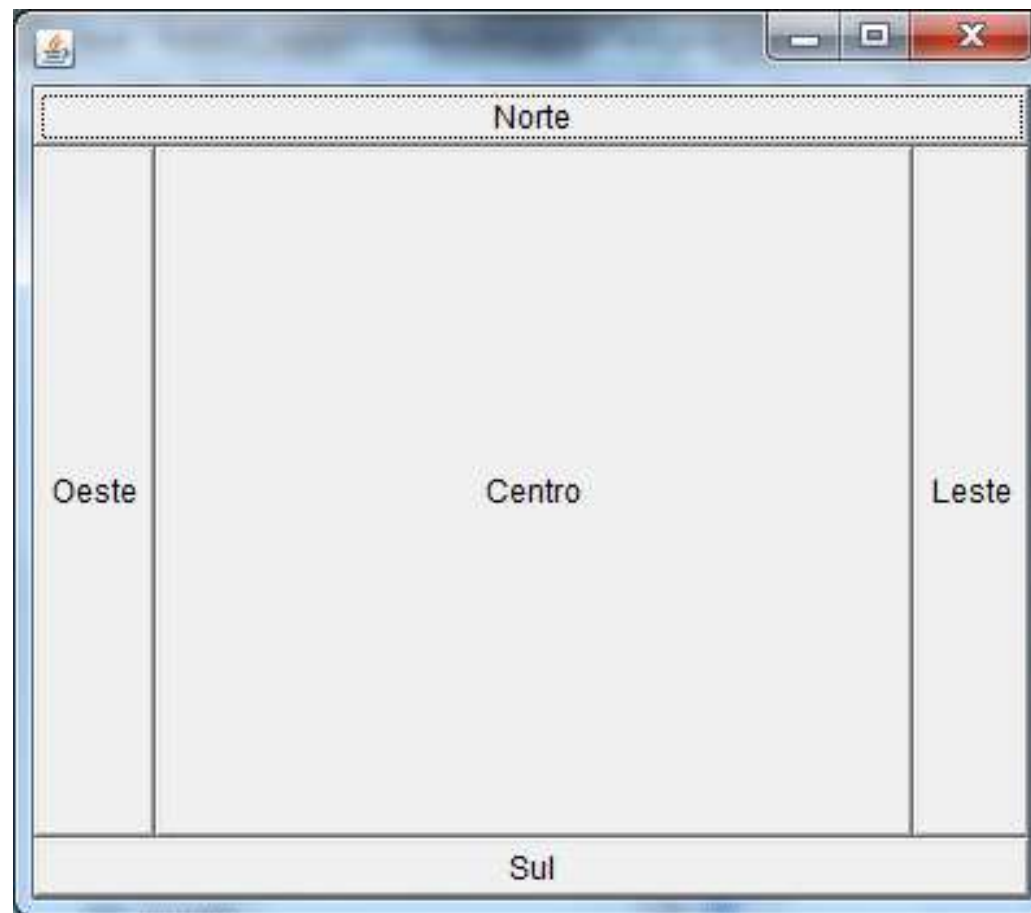
WEST - borda direita

CENTER - área central

- A área central prevalece sobre as demais quando esta "moldura" for redimensionada.
- O componente no centro da interface é redimensionado em igual proporção ao redimensionamento do container enquanto os demais componentes apenas preenchem os espaços que forem adicionados em suas respectivas bordas.



# Exemplo BorderLayout



# Exemplo BorderLayout

```
import java.awt.*;

public class BorderLayoutTest extends Frame {
    BorderLayoutTest() {
        setSize(400, 350);
        setLayout(new BorderLayout());
        add( BorderLayout.NORTH,new Button("Norte"));
        add( BorderLayout.EAST, new Button("Leste"));
        add( BorderLayout.SOUTH, new Button("Sul"));
        add( BorderLayout.WEST, new Button("Oeste"));
        add( BorderLayout.CENTER, new Button("Centro"));
    }

    public static void main(String[] args) {
        BorderLayoutTest borderLayoutTest = new BorderLayoutTest();
        borderLayoutTest.setVisible(true);
    }
}
```

# GridLayout

- É um gerenciador de layout que divide o Container em uma grade de modo que os componentes podem ser colocados nas linhas e colunas.
- A classe **GridLayout** estende a classe **Object** e implementa a interface **LayoutManager**.
- Cada componente no **GridLayout** tem os mesmos tamanhos, onde podem ser inserida uma célula na parte superior esquerda da grade que prossegue da esquerda para a direita até preencher todas as células

# Exemplo GridLayout

```
import java.awt.*;

public class GridLayoutTest extends Frame {
    GridLayoutTest() {
        setSize(100, 200);
        setLayout(new GridLayout(4,3));
        add(new Button("7"));
        add(new Button("8"));
        add(new Button("9"));
        add(new Button("4"));
        add(new Button("5"));
        add(new Button("6"));
        add(new Button("1"));
        add(new Button("2"));
        add(new Button("3"));
        add(new Button("0"));
    }
    public static void main(String[] args) {
        GridLayoutTest gridLayoutTest = new GridLayoutTest();
        gridLayoutTest.setVisible(true);
    }
}
```

# Exemplo GridLayout



# Dúvidas?

