

Desenvolvimento Backend

Prof. Ms. Gustavo Molina

Swing

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*



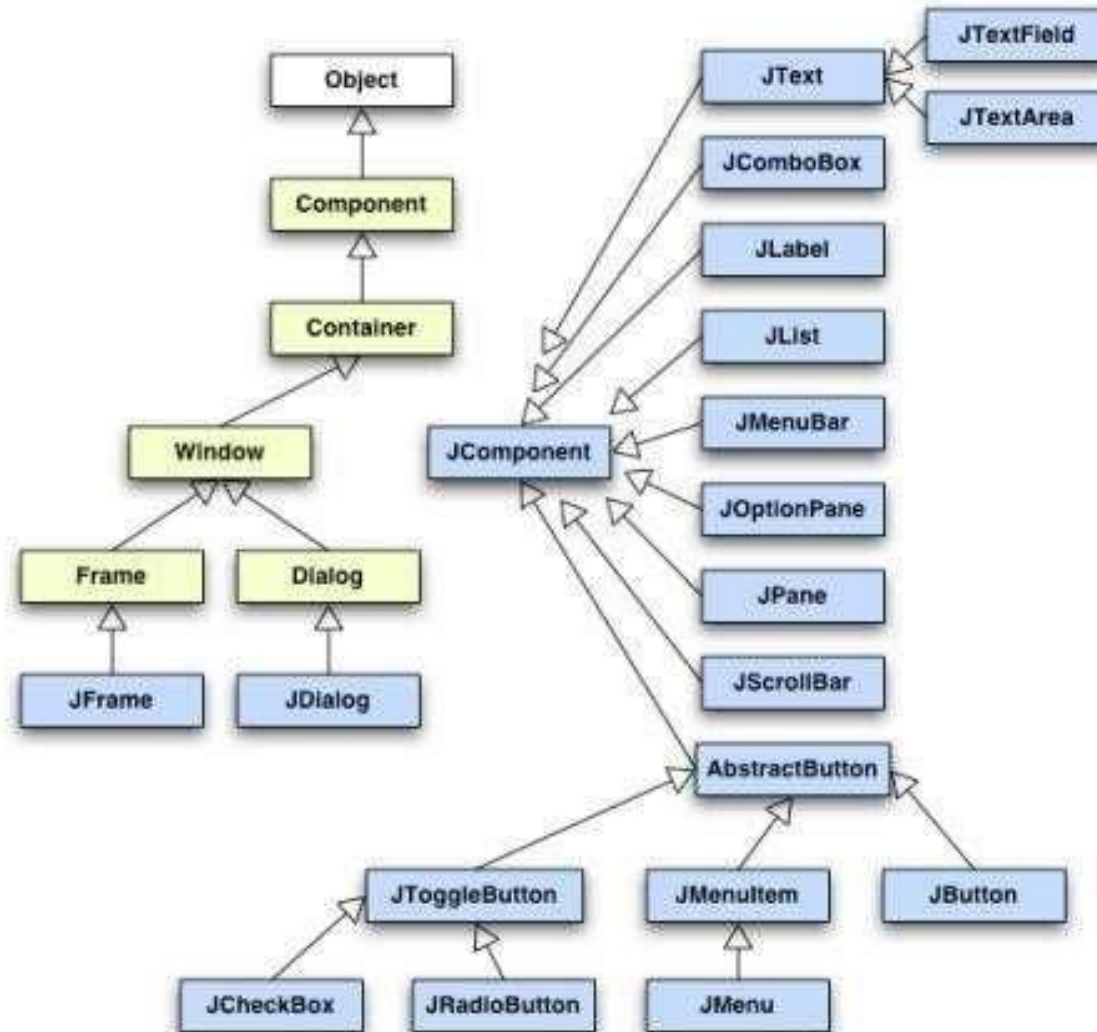
Projeto Swing

- 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*.

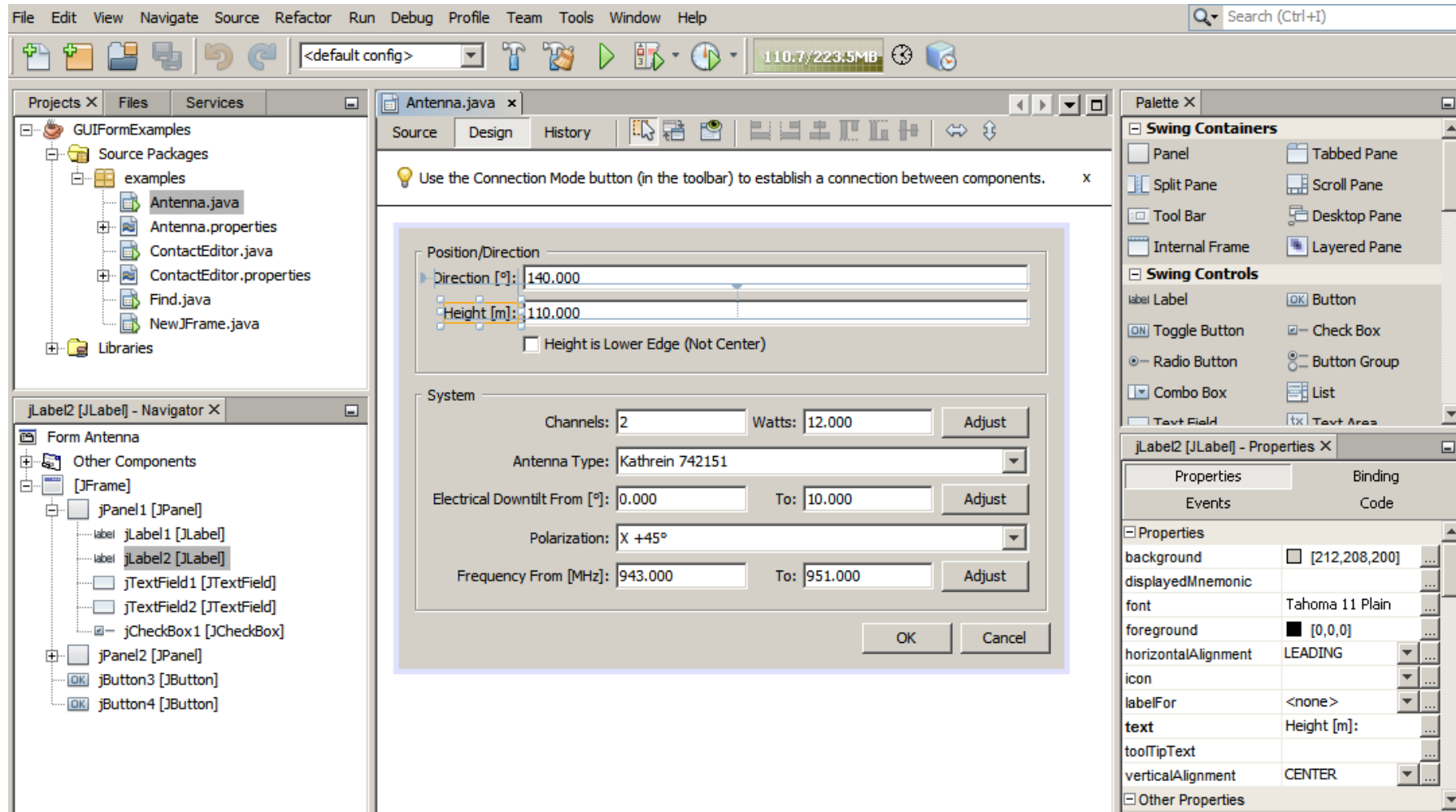
AWT e Swing



Containers e Componentes

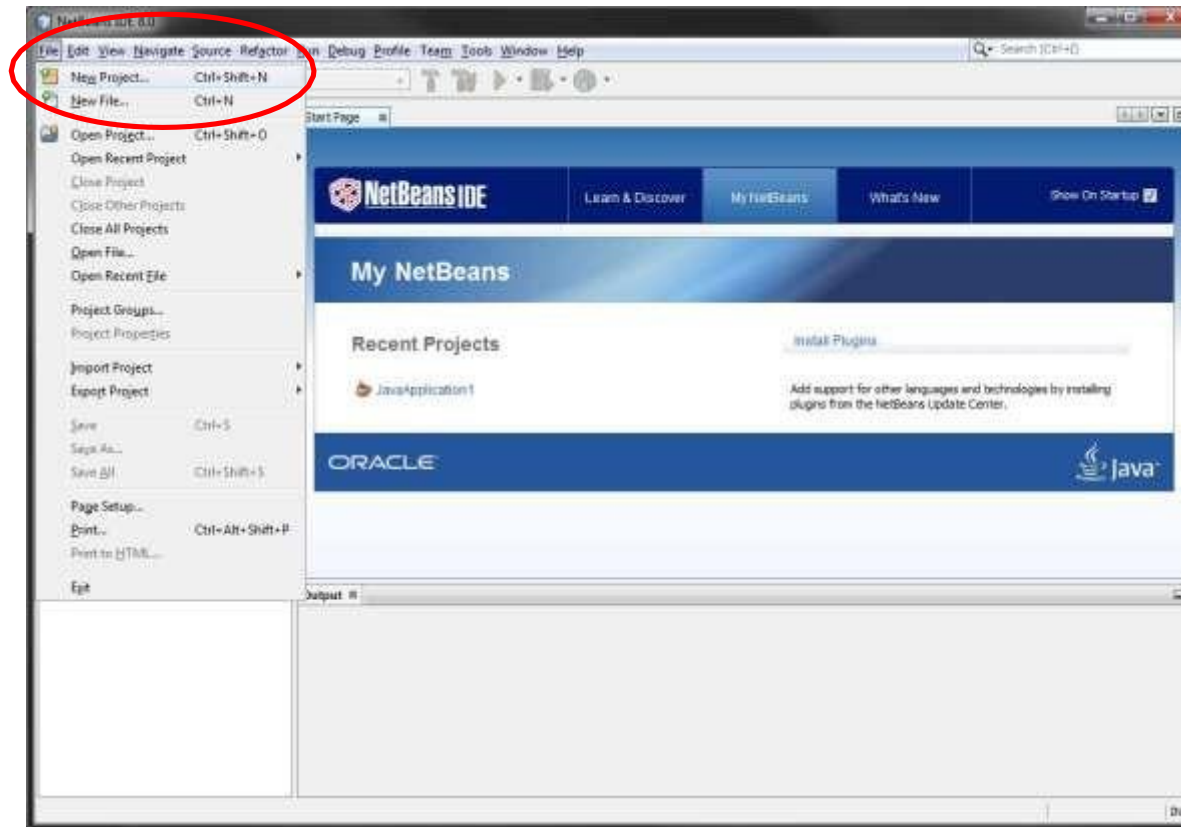
- Uma interface gráfica em Java é baseada em dois elementos:
 - **Containers:** servem para agrupar e exibir outros componentes.
 - **Componentes:** botões, labels, scrollbars, etc.
- Todo programa Java que ofereça uma interface possui pelo menos um container.
- Uma janela de nível mais alto (que não 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



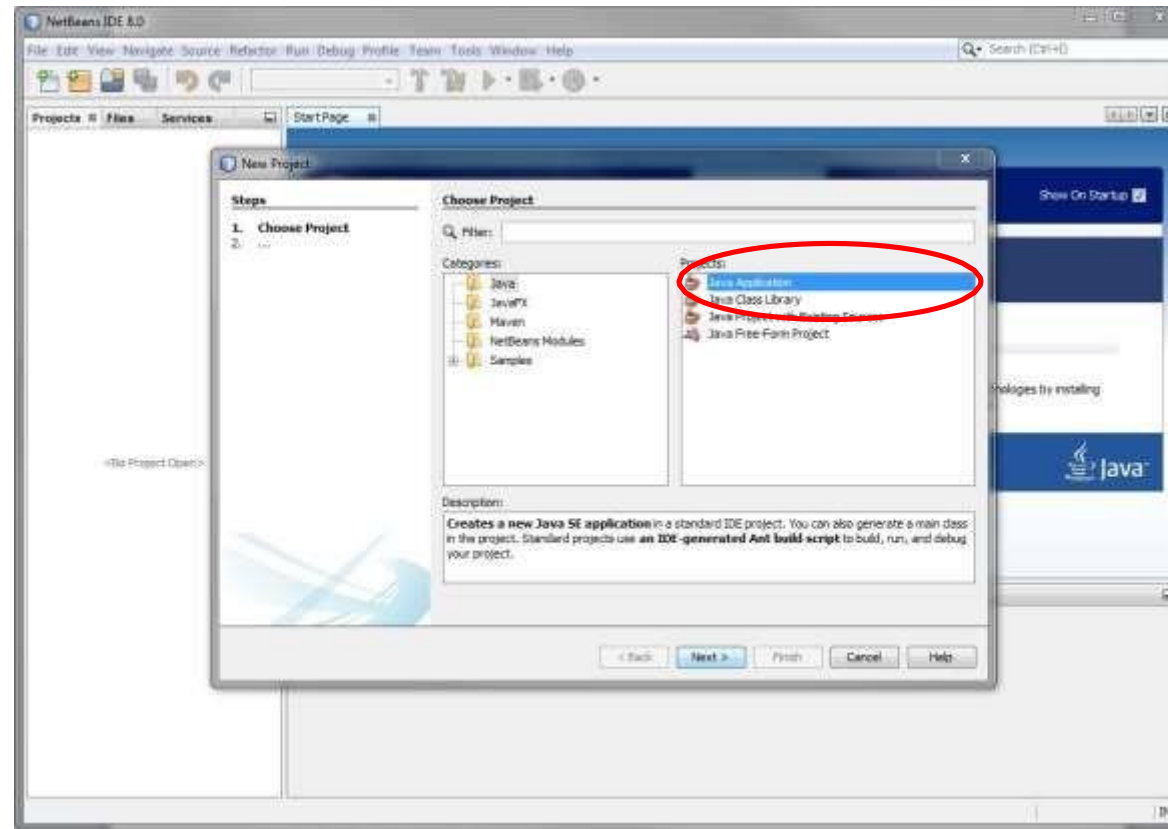
Netbeans GUI Builder – Criando Projeto

1) Acesse o menu File -> New Project



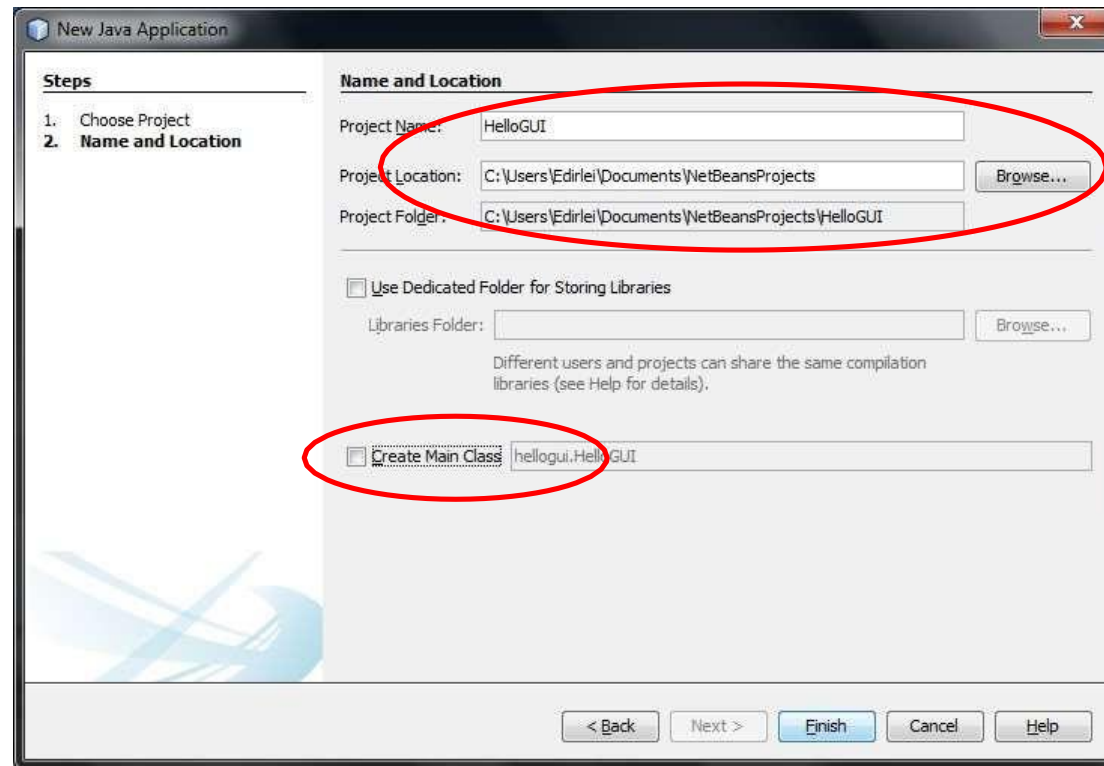
Netbeans GUI Builder – Criando Projeto

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



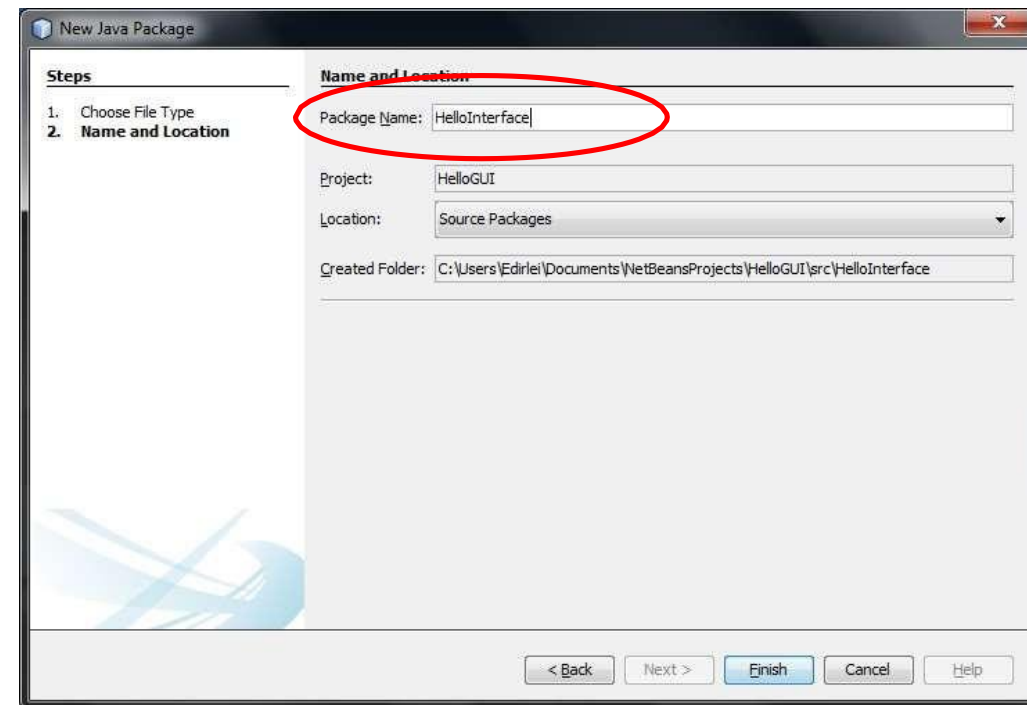
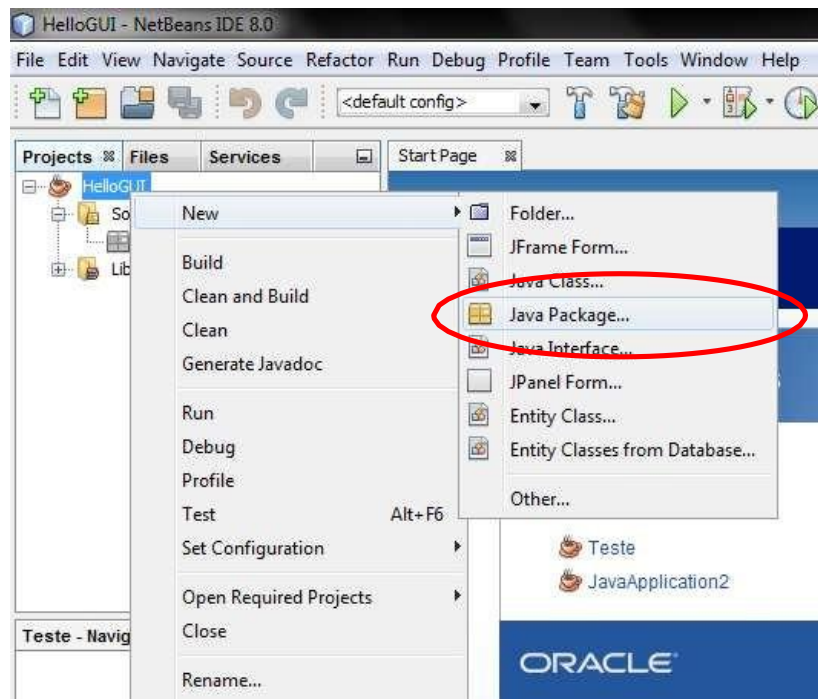
Netbeans GUI Builder – Criando Projeto

- 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”:



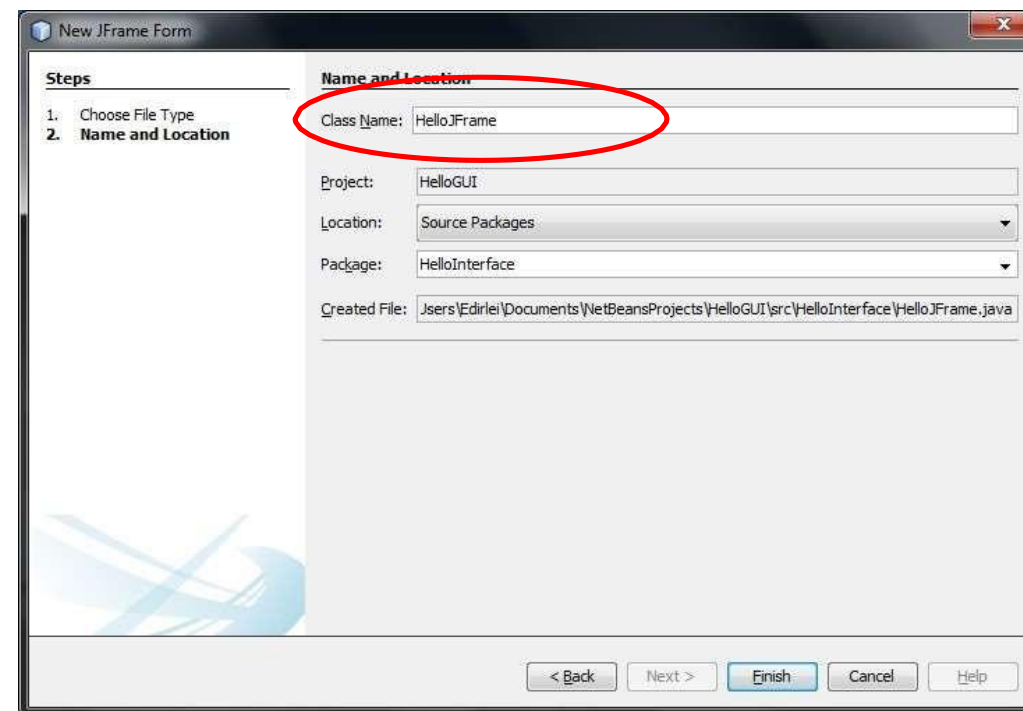
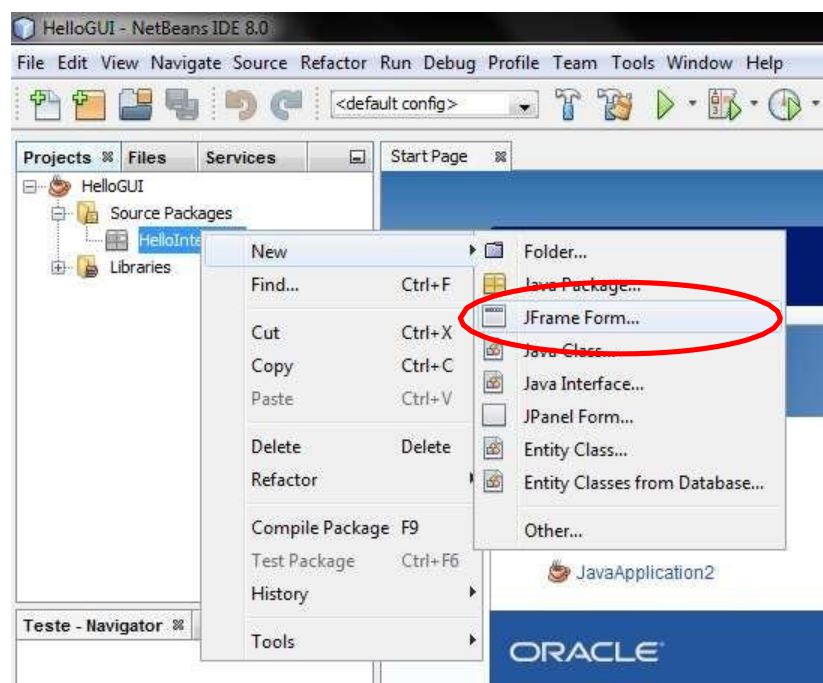
Netbeans GUI Builder – Criando Projeto

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

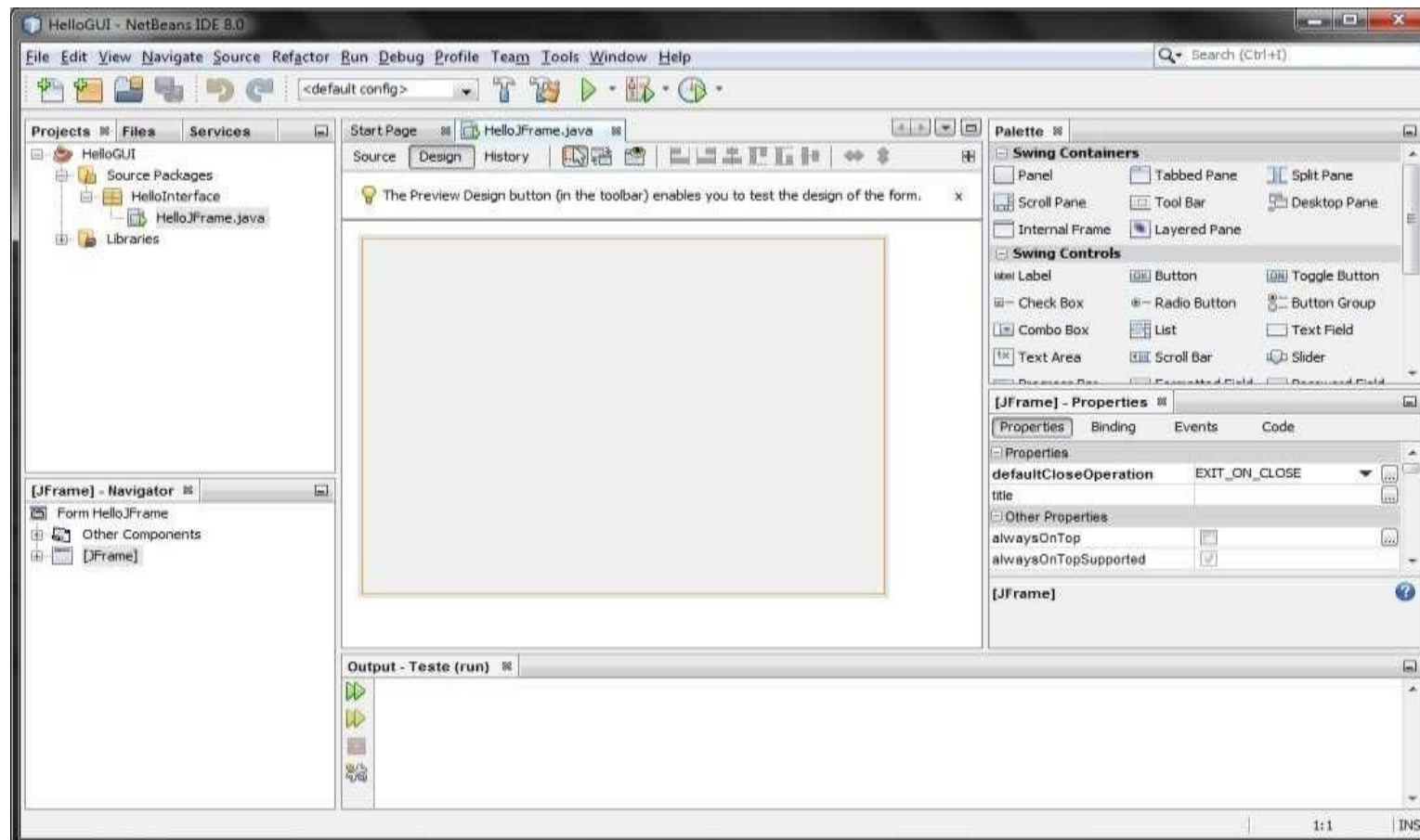


Netbeans GUI Builder – Criando Projeto

5) Crie um novo “JFrame Form”:

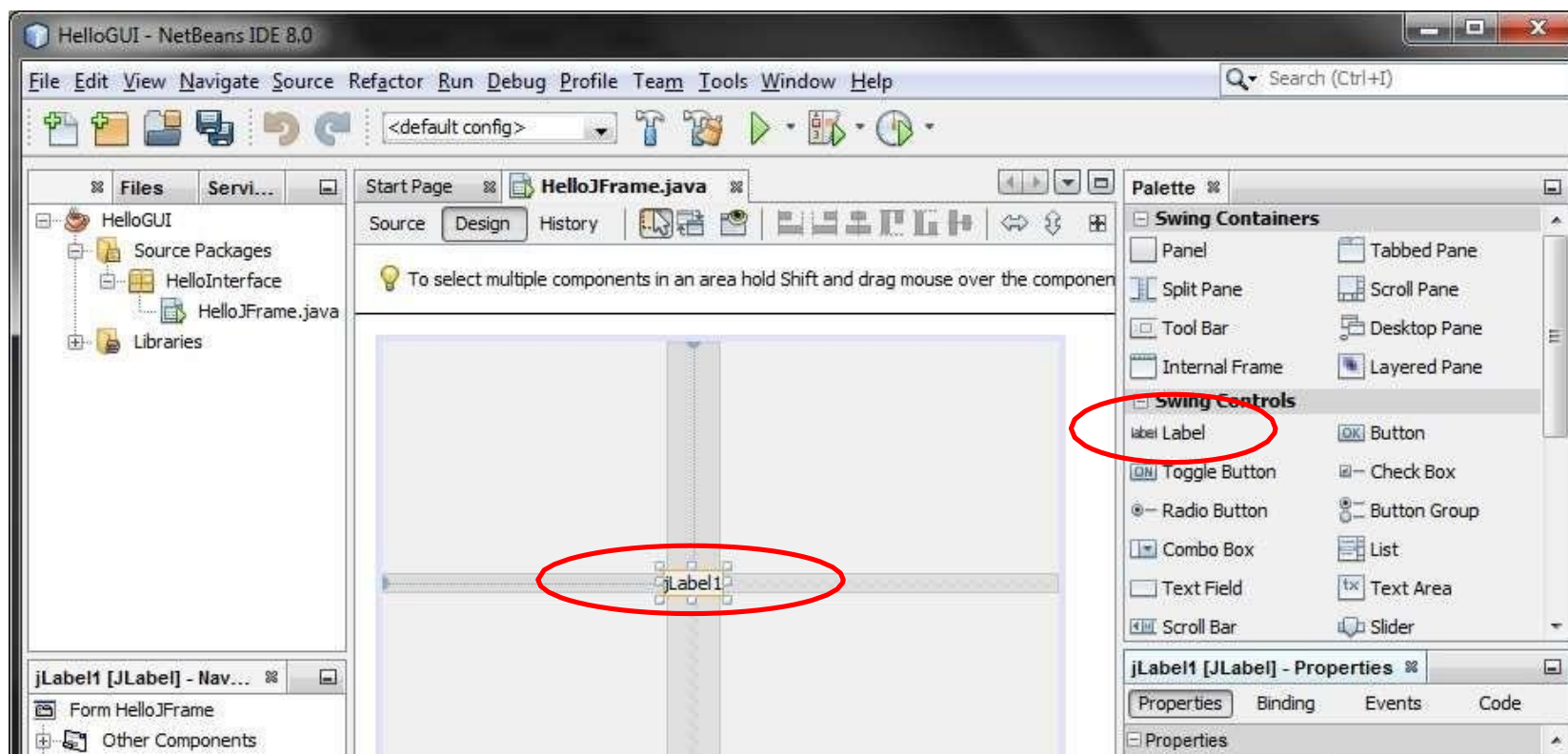


Netbeans GUI Builder – Criando Projeto



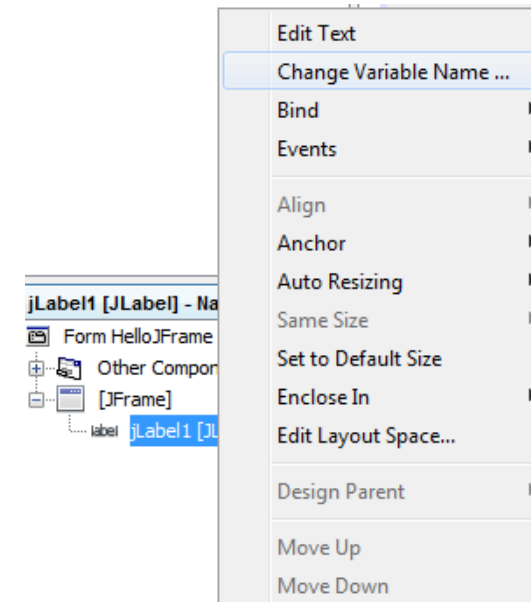
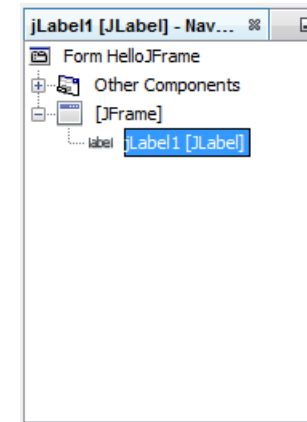
Componentes Básicos - Label

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



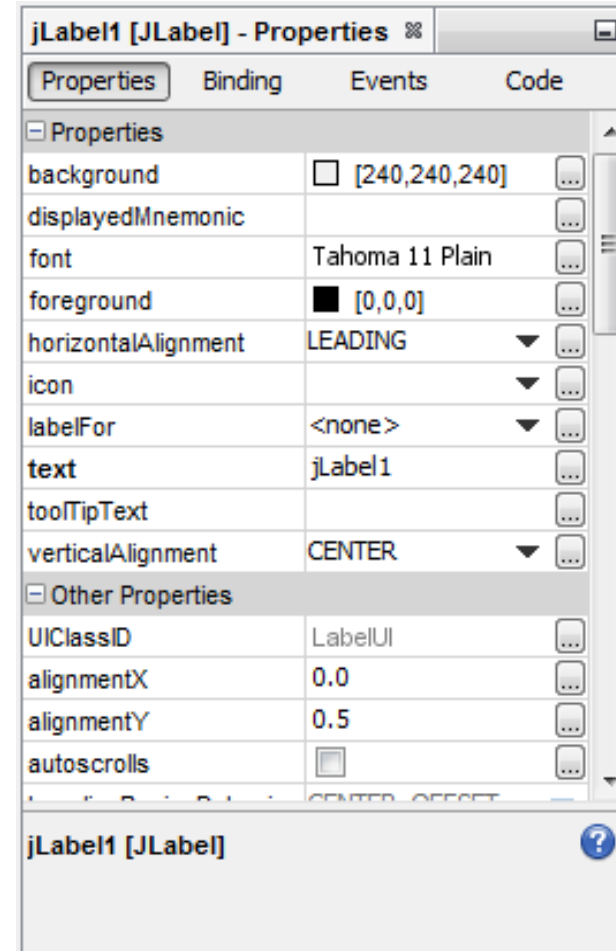
Componentes Básicos - Label

- 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.



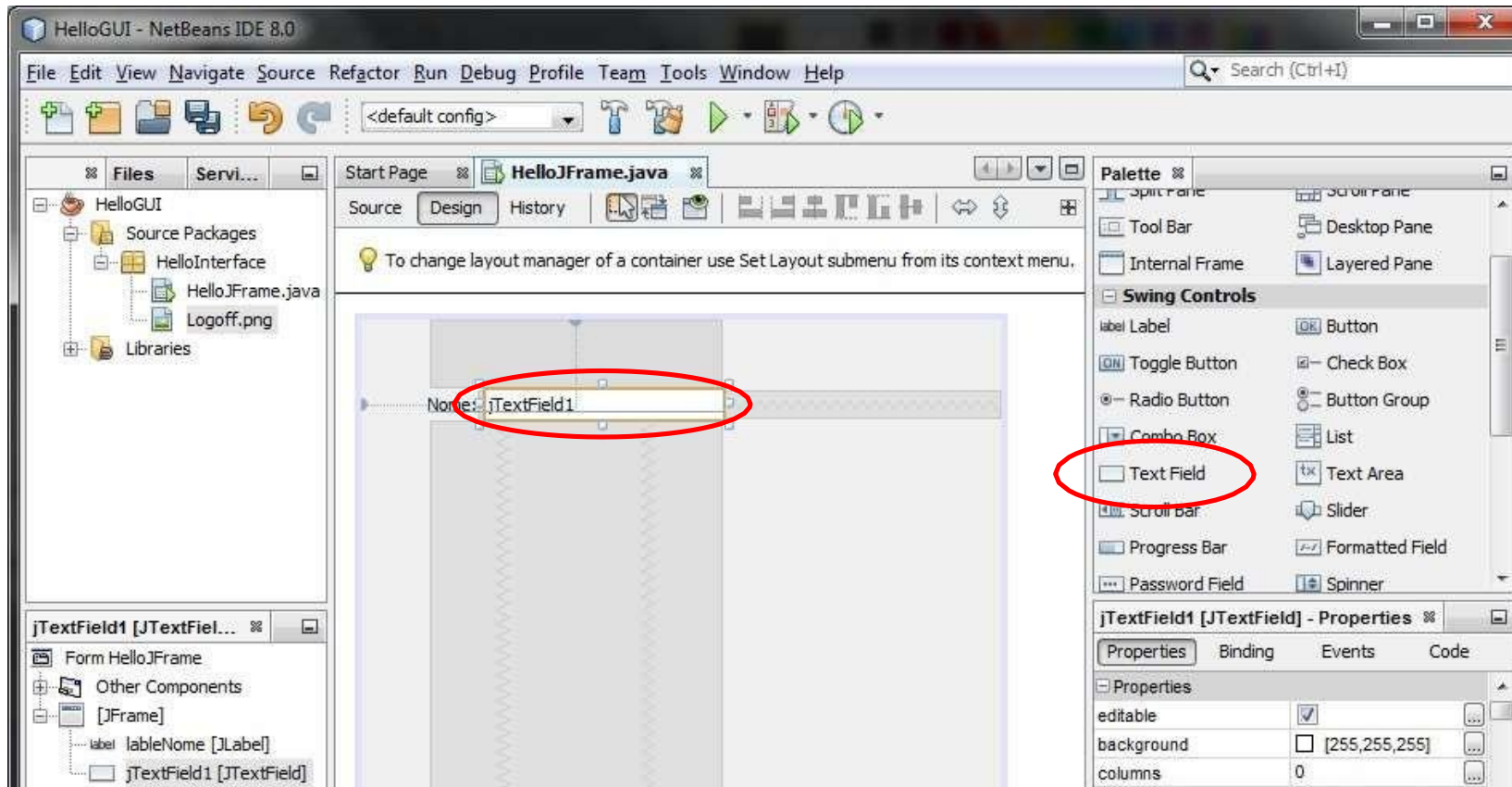
Componentes Básicos - Label

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



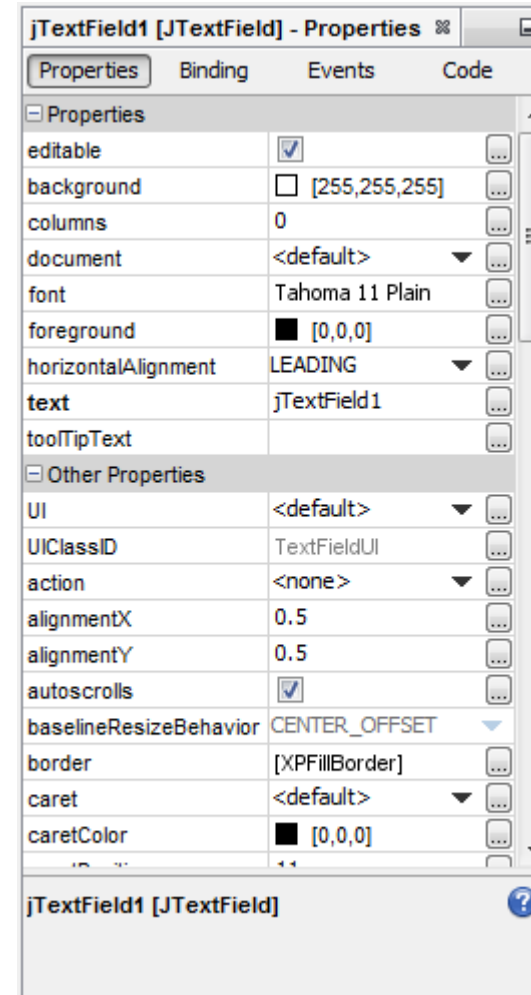
Componentes Básicos – TextField

- Componente para entrada, edição e exibição de texto.



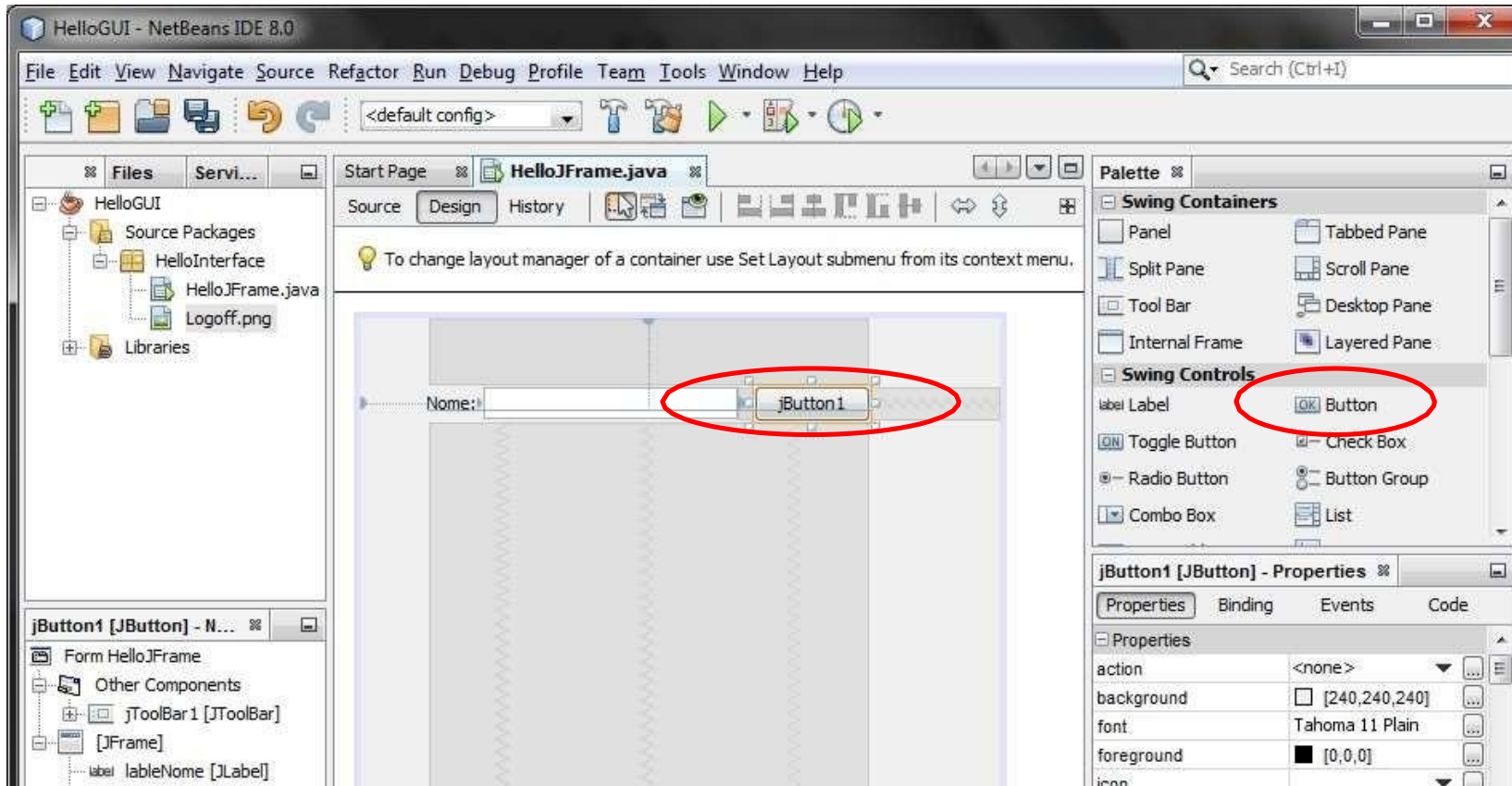
Componentes Básicos – TextField

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



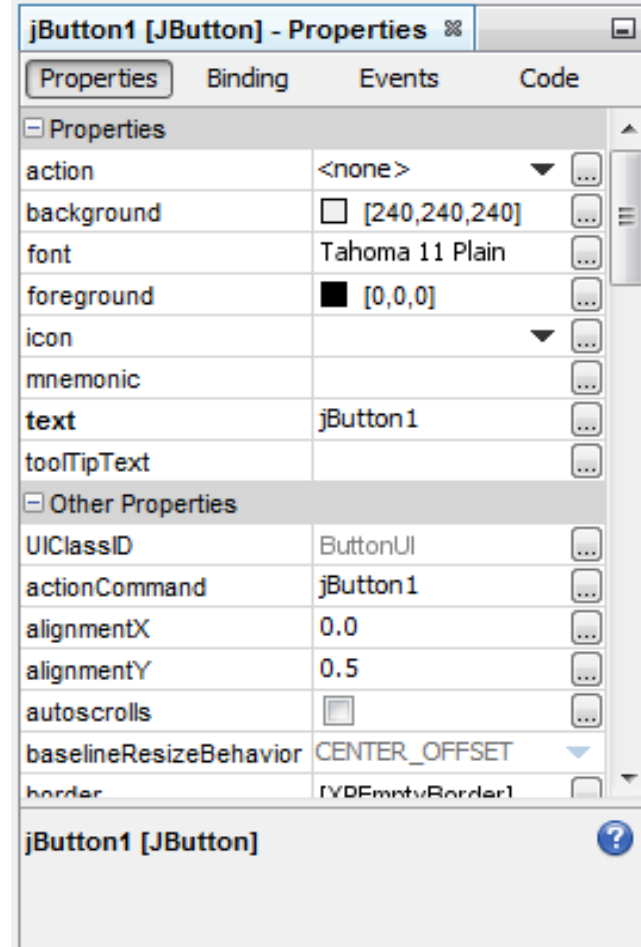
Componentes Básicos – Button

- Componente que representa um botão.



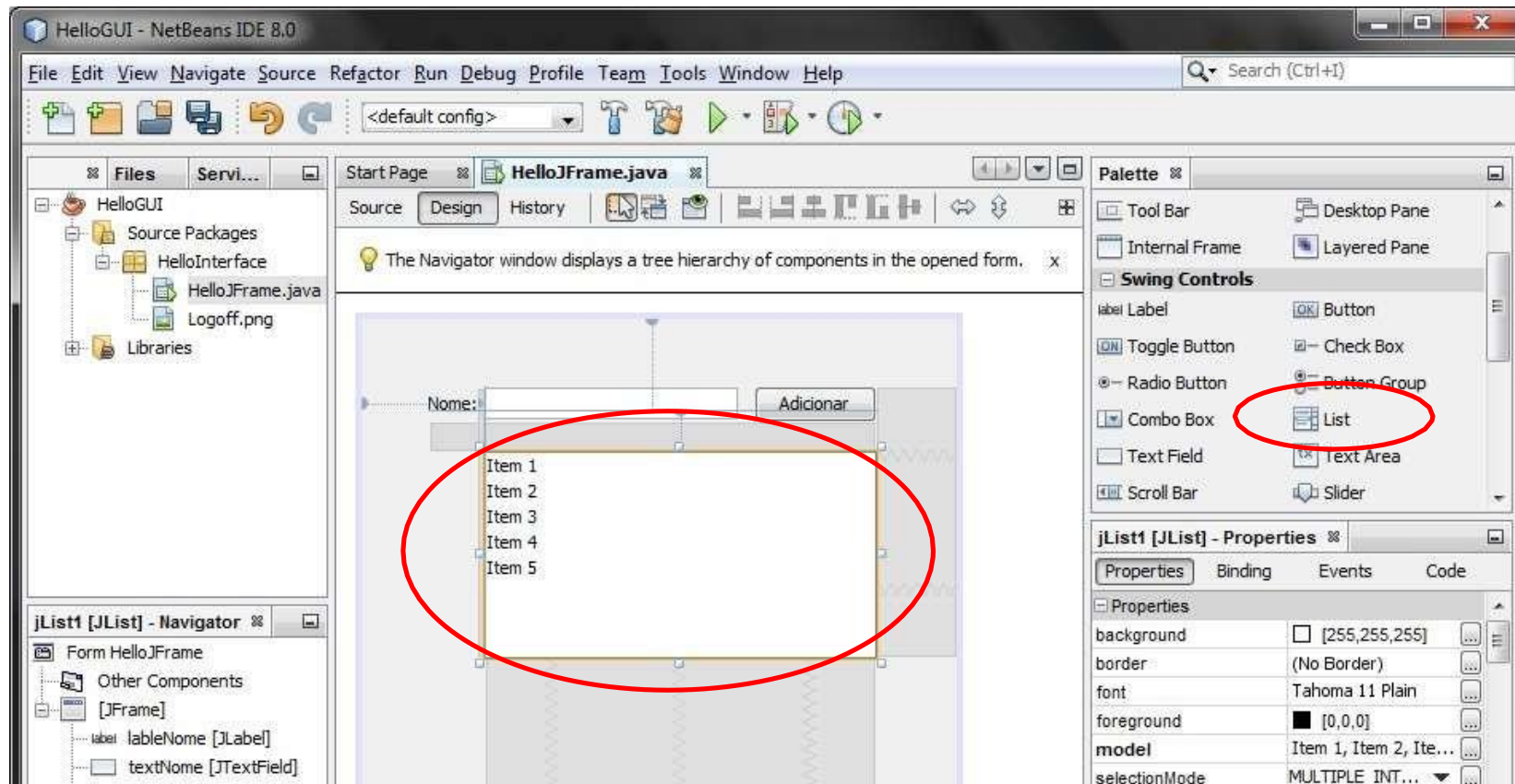
Componentes Básicos – Button

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



Componentes Básicos – List

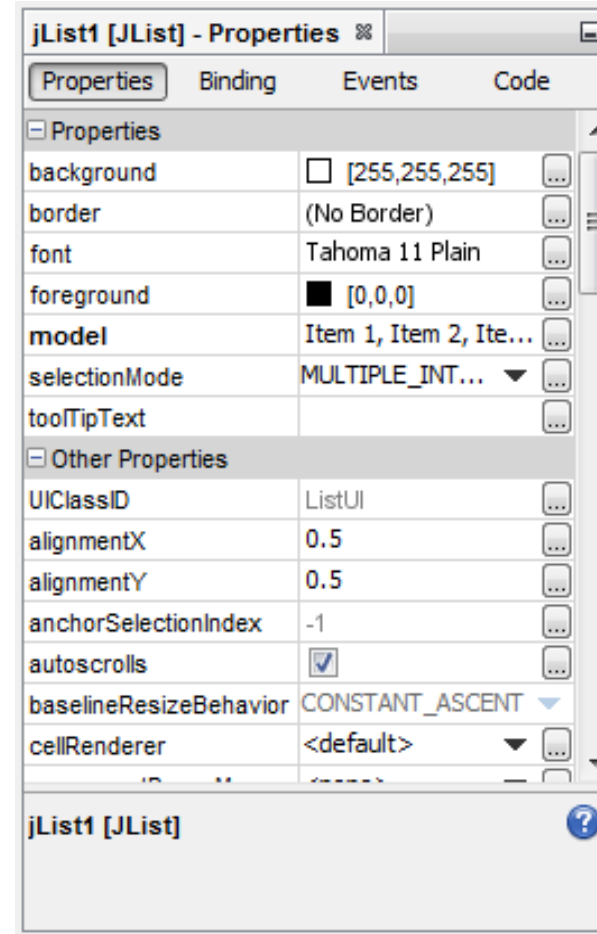
- Componente que exibe uma lista de itens e permite que o usuário possa seleciona-los.



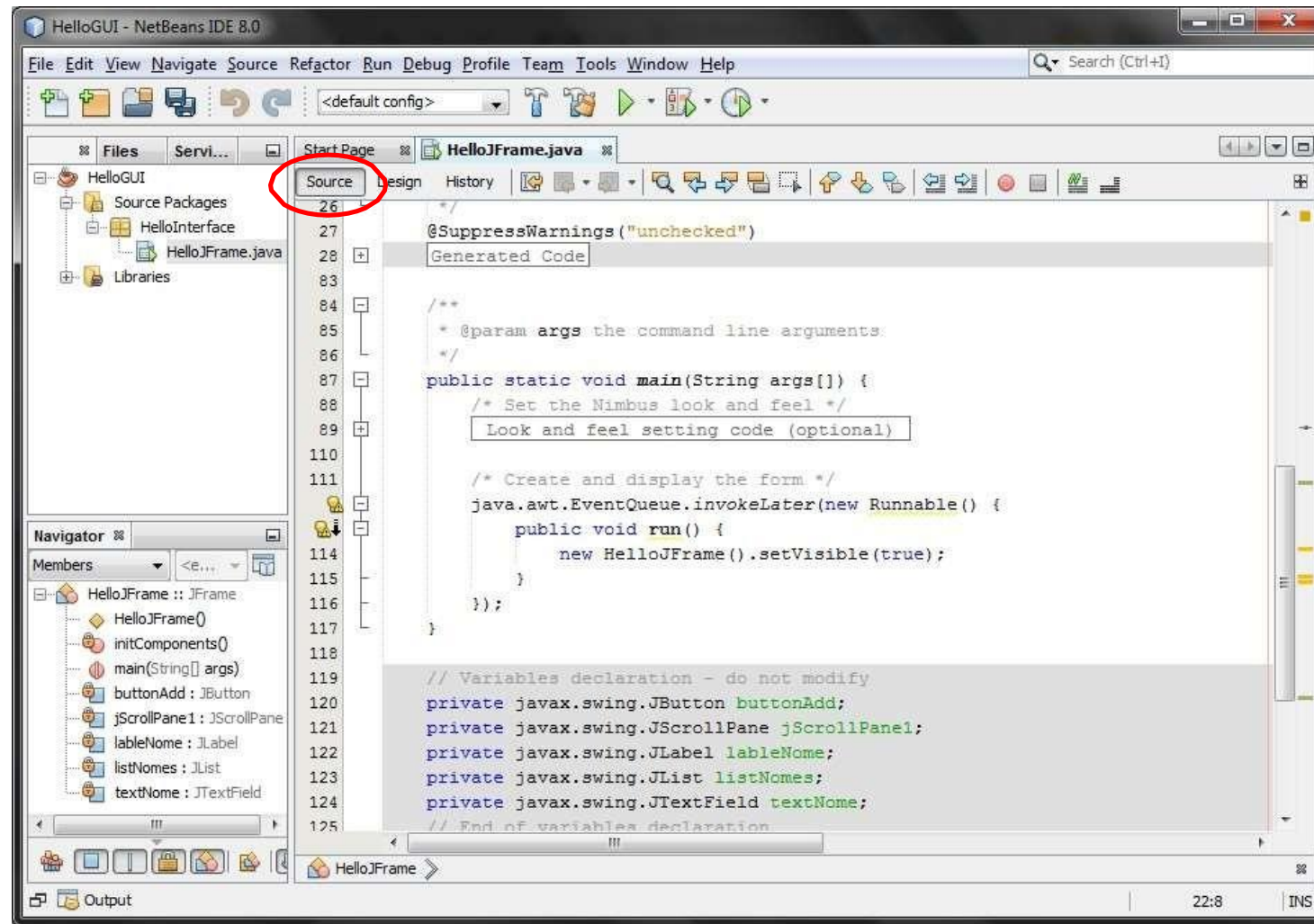
Componentes Básicos – List

- Principais Propriedades (JList):

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

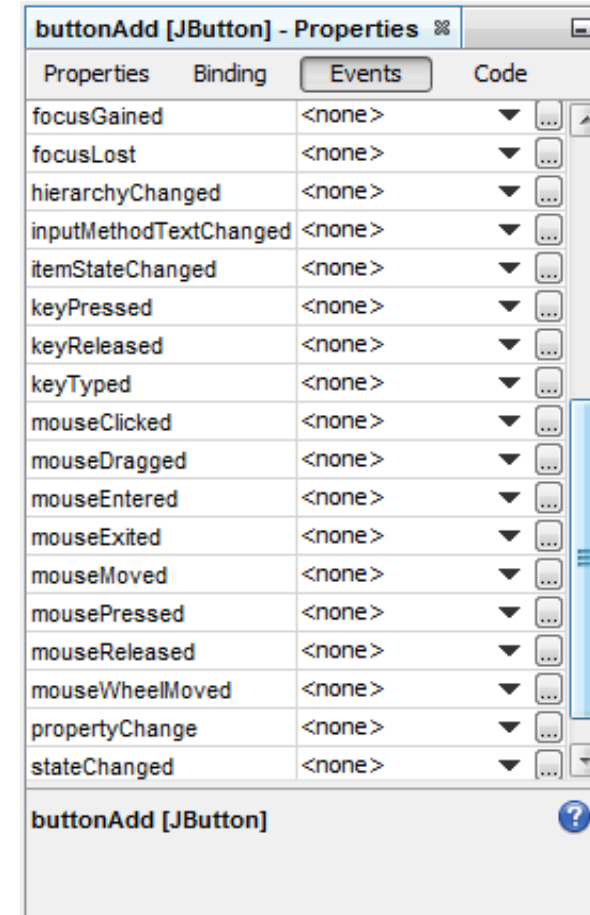


Netbeans GUI Builder – Código Gerado



Eventos – Button

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



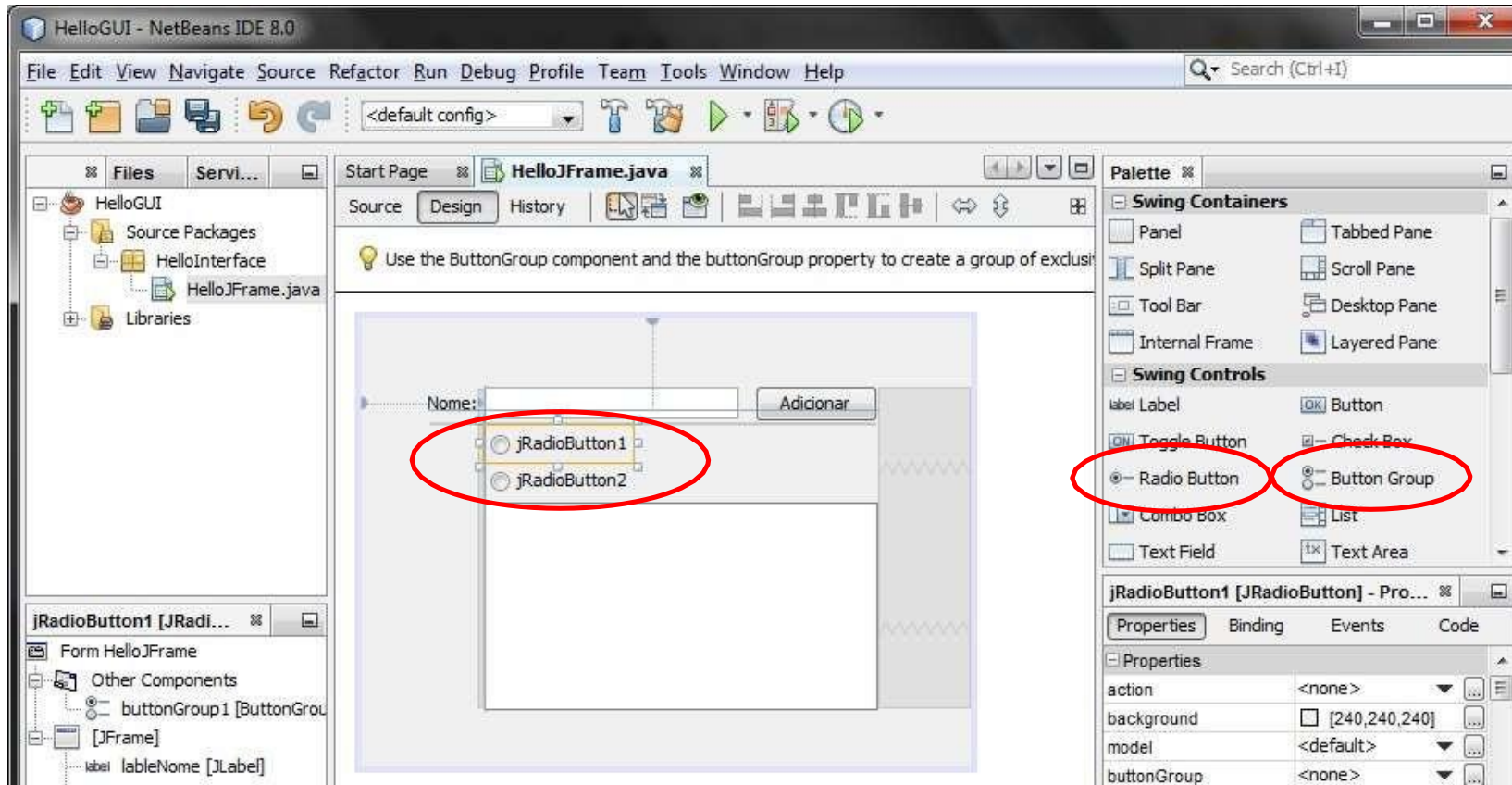
Evento actionPerformed – Button

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

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

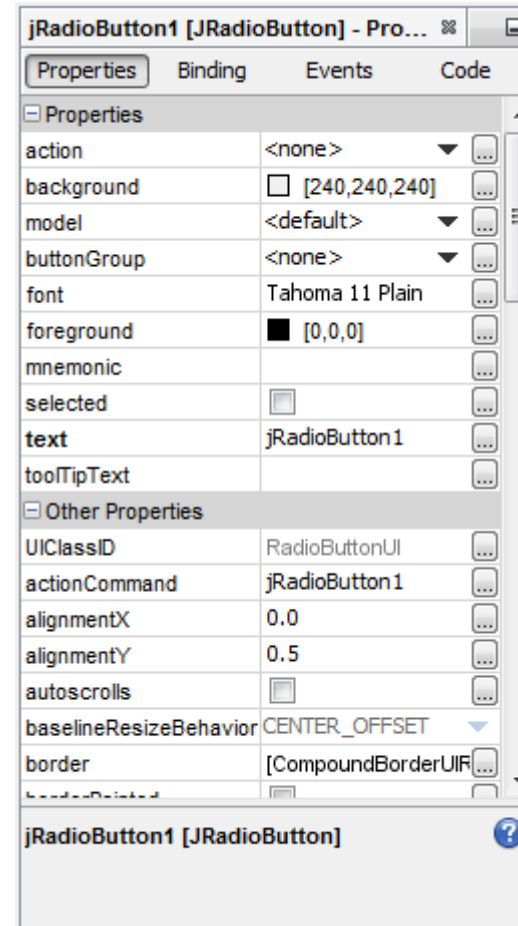
Componentes Básicos – RadioButton e ButtonGroup

- Componentes que permitem a seleção de opções.



Componentes Básicos – RadioButton

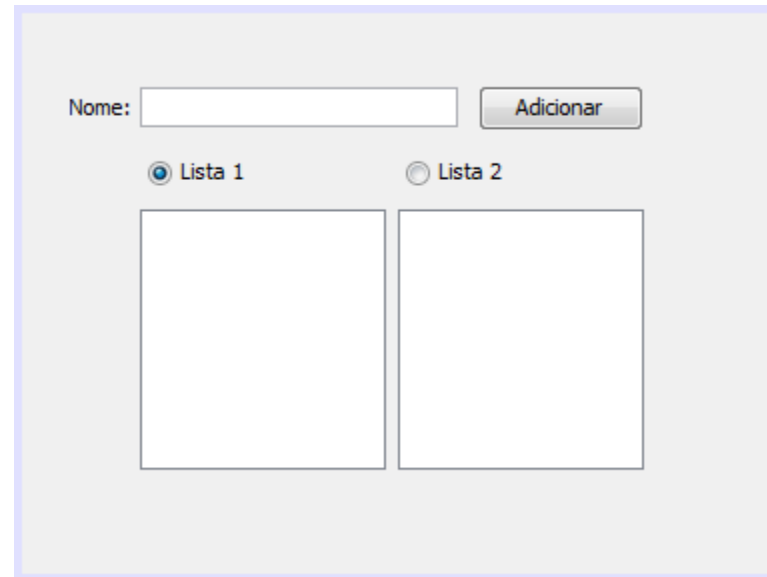
- Principais Propriedades (JRadioButton):
 - text;
 - buttonGroup;
 - Selected;
 - foreground;
 - background;
 - font;
 - icon;
 - toolTipText;
 - border;
 - enabled;



Usando o RadioButton

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

– **Interface:**



The image shows a user interface with a light gray background. At the top, there is 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 radio button is selected (indicated by a blue dot) and is labeled "Lista 1". The second radio button is not selected (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

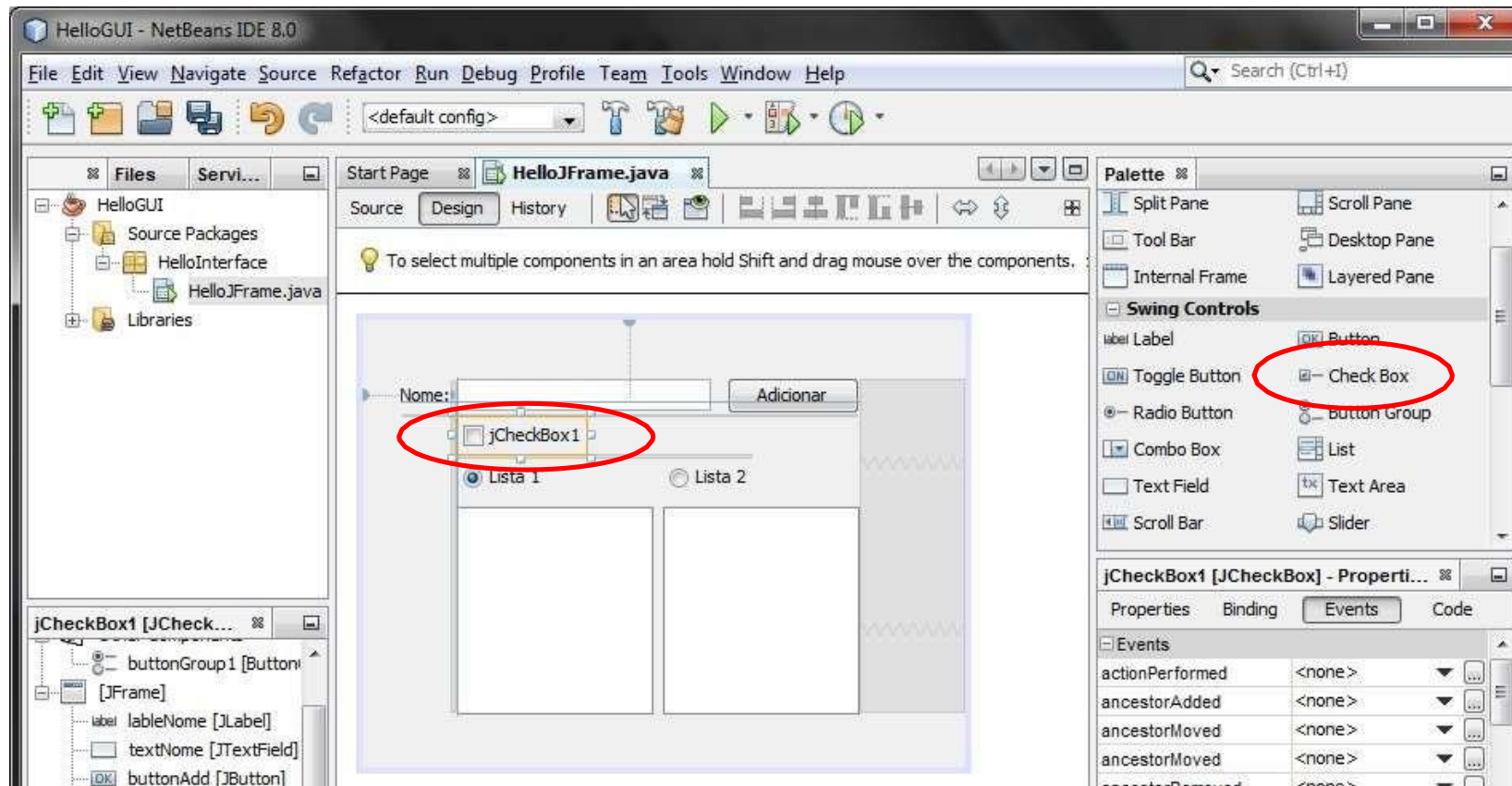
- **Exemplo** – Adicionar o conteúdo do TextField nas Lists de acordo com a opção selecionada no RadioButton.

```
...  
  
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("");  
}  
  
...
```



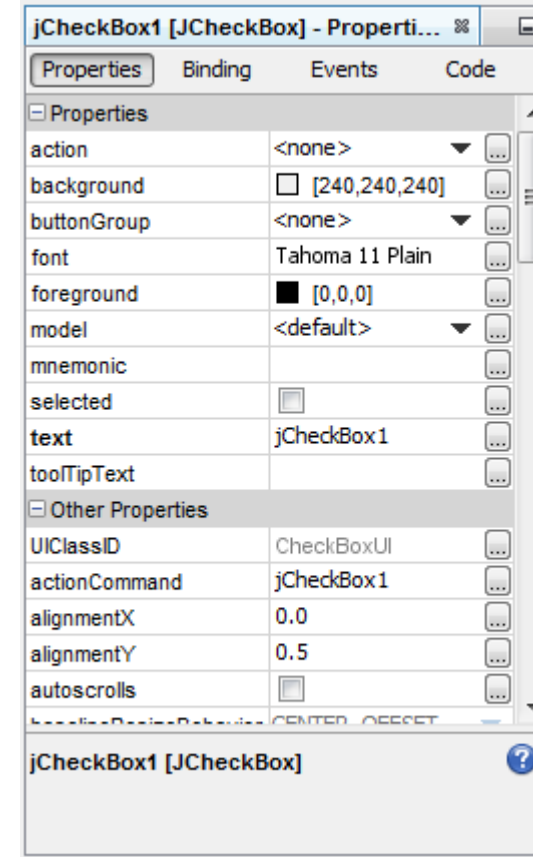
Componentes Básicos – CheckBox

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



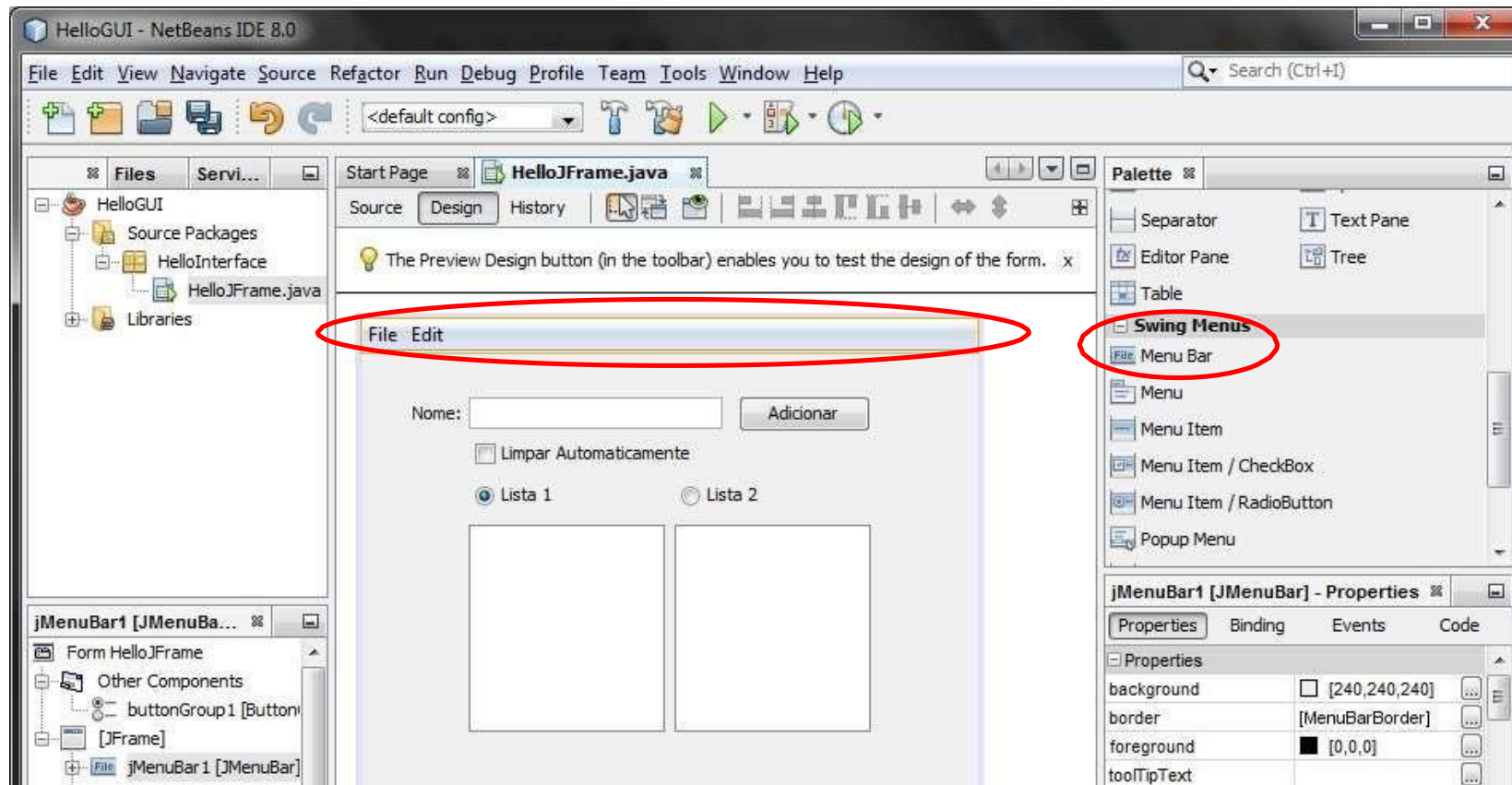
Componentes Básicos – CheckBox

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



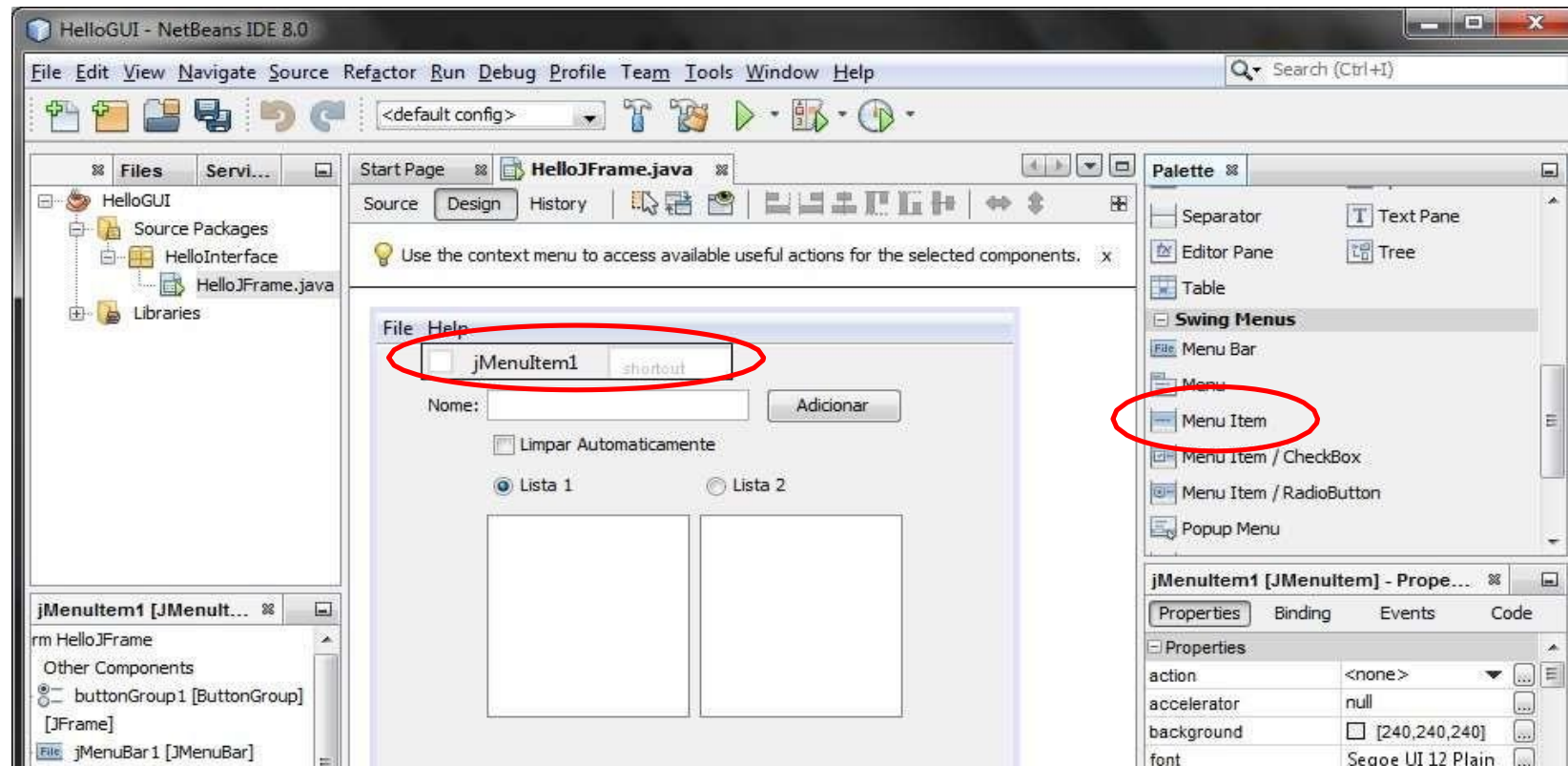
Componentes Básicos – MenuBar

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



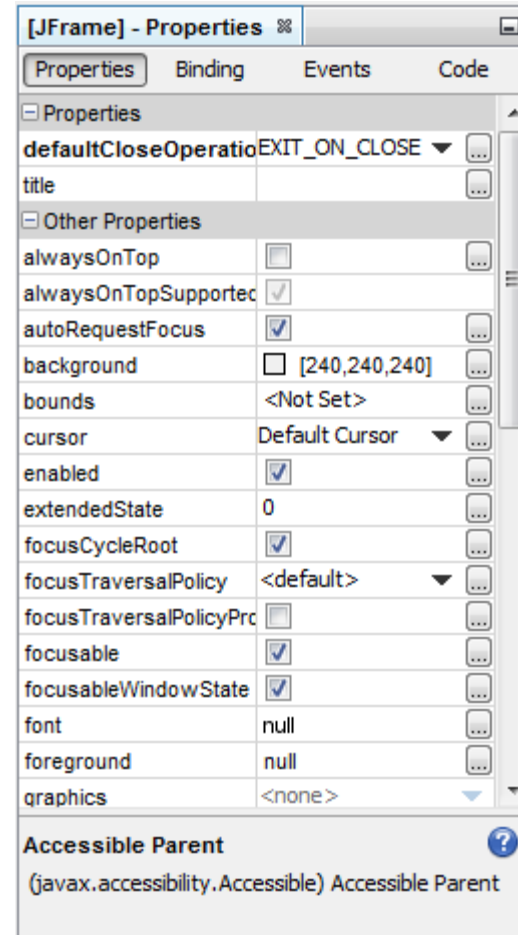
Componentes Básicos – MenuItem

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



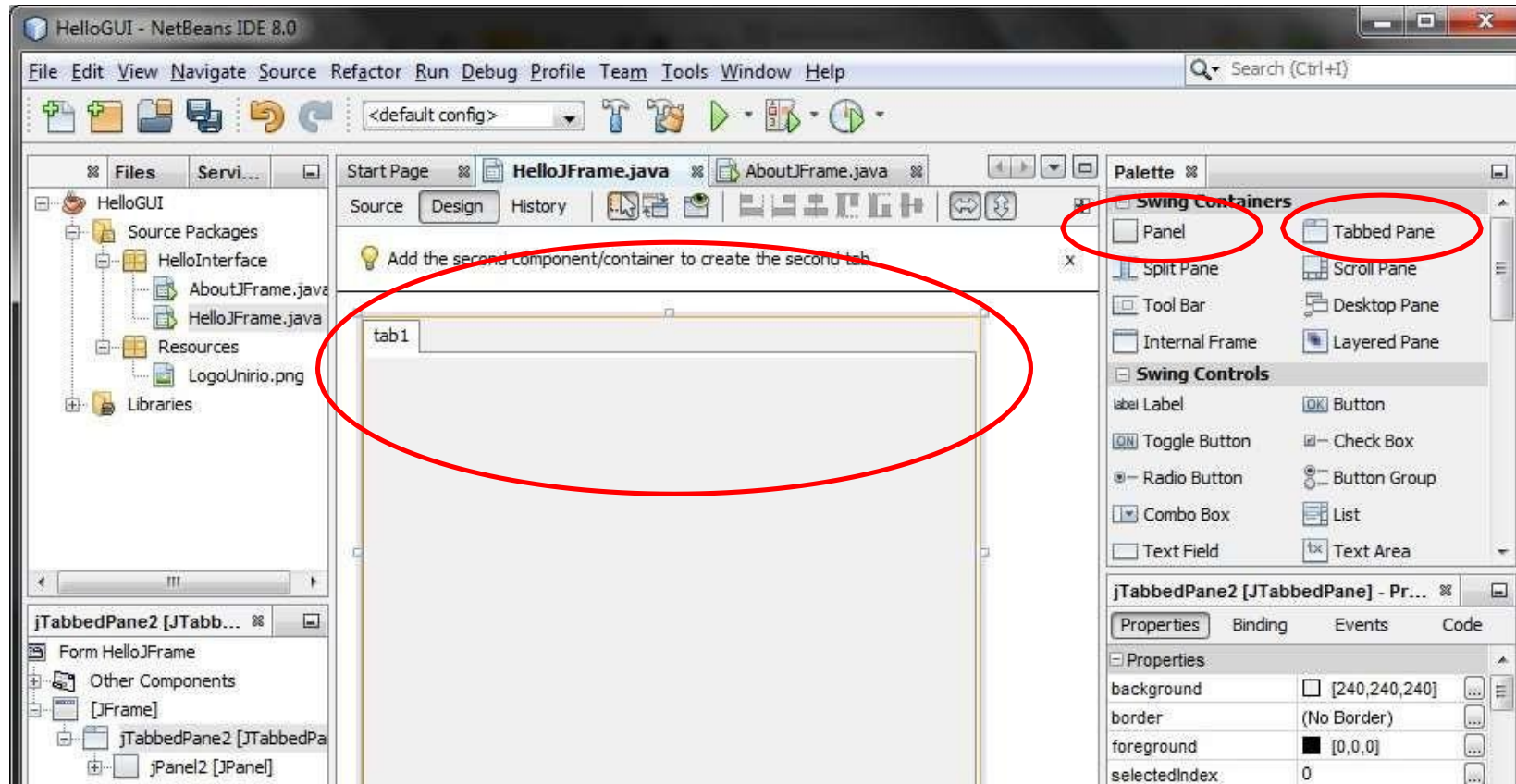
Componentes Básicos – Frame

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

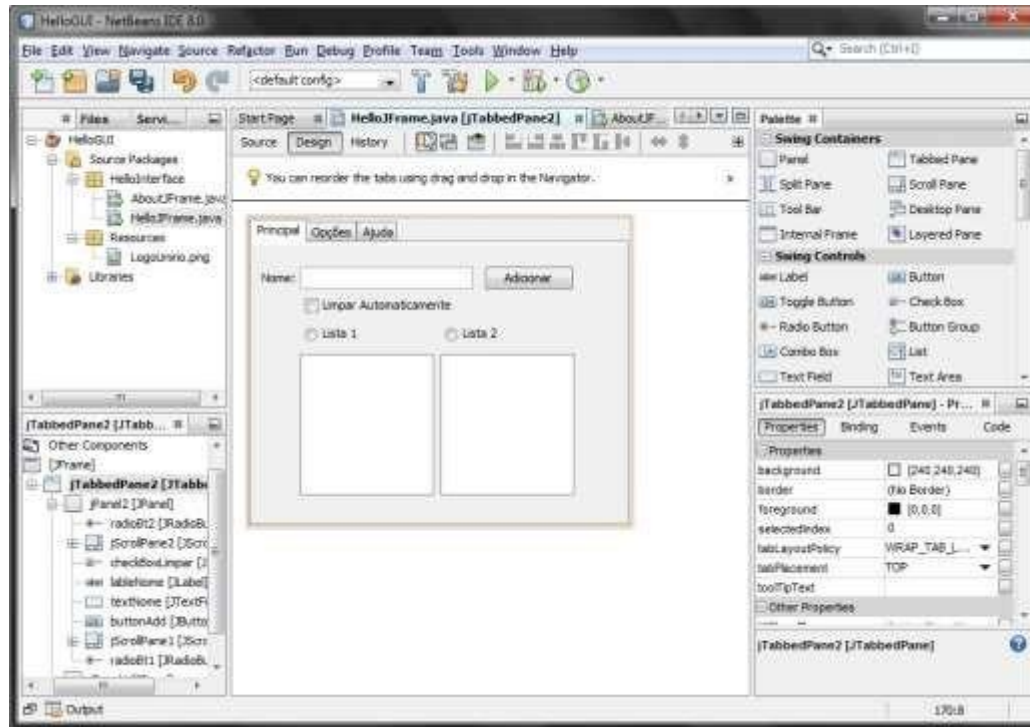


Componentes Básicos – Tabbed Pane

- 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:

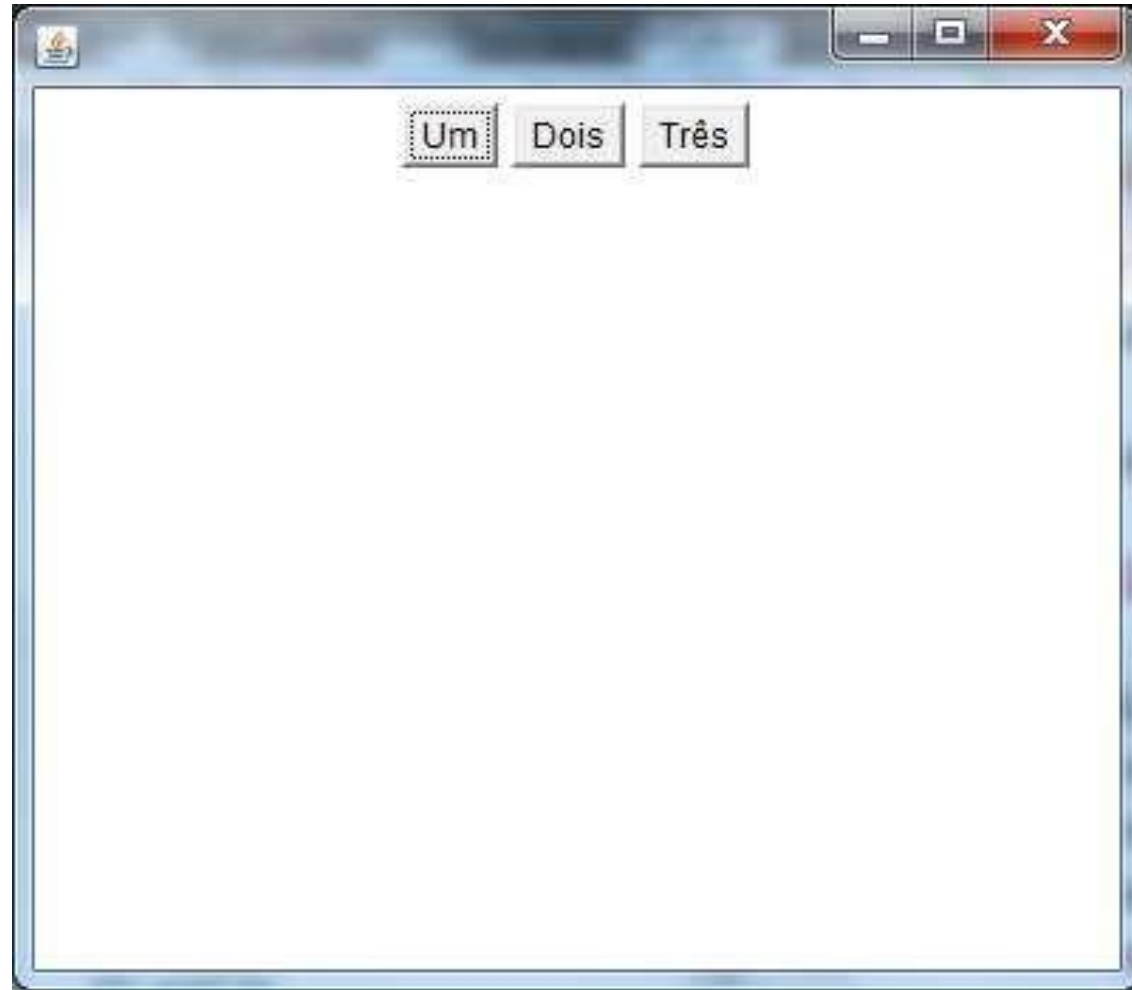
- FlowLayout
- BorderLayout
- GridLayout
- CardLayout
- GridBagLayout



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

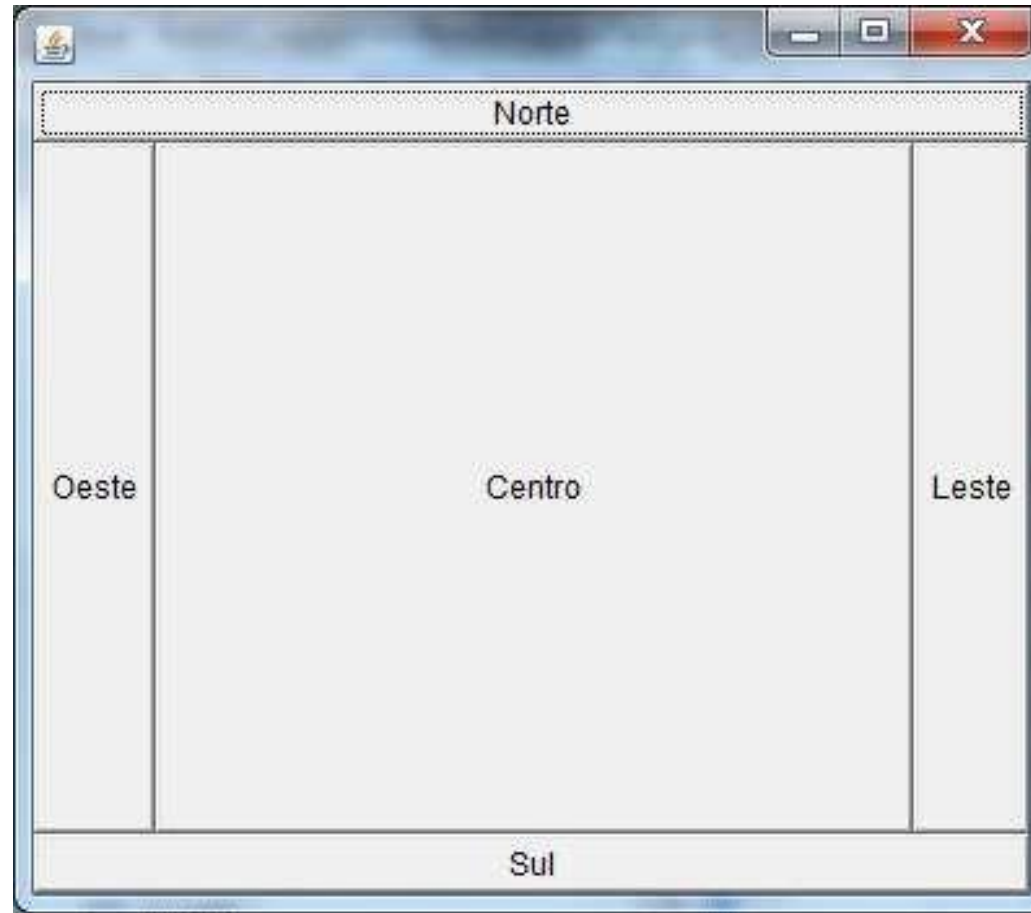
EAST - borda esquerda

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



JButton

Um objeto da classe **JButton** é um botão que o usuário pode clicar e, então, uma ação pode ser executada.

CONSTRUTOR

String texto

Icon imagem

```
JButton btn = new JButton(texto);
```

```
JButton btn = new JButton(texto, imagem);
```

```
JButton btn = new JButton(imagem);
```

JButton



JLabel

JLabel

Um objeto da classe **JLabel** é um dos componentes mais simples, já que servem apenas para exibir textos na tela. O usuário não consegue editar o texto, porém, o programa pode alterar suas características.

CONSTRUTOR

```
String texto  
Icon    imagem  
int     alinhamento // JLabel.LEFT, JLabel.CENTER ou JLabel.RIGHT
```

```
JLabel lbl = new JLabel(texto);  
JLabel lbl = new JLabel(texto, alinhamento);  
JLabel lbl = new JLabel(imagem);  
JLabel lbl = new JLabel(imagem, alinhamento);  
JLabel lbl = new JLabel(texto, imagem, alinhamento);
```

JCheckbox

JCheckBox

Um objeto da classe **JCheckBox** é um componente que representa dois estados: selecionado ou não selecionado. O usuário pode alterar o estado da caixa clicando sobre ela. O estado da caixa de seleção é representado por um valor do tipo boolean, sendo: **true** (caixa selecionada) e **false** (caixa não selecionada).

CONSTRUTOR

```
String    texto;    // texto do CheckBox  
boolean   estado;   // estado do CheckBox (true ou false)
```

```
JCheckBox cb = new JCheckBox(texto);  
JCheckBox cb = new JCheckBox(texto, estado);
```

JCheckbox



JRadioButton

Um objeto da classe **JRadioButton** é um componente que representa um grupo de opções (**ButtonGroup**), onde apenas uma delas podem ser selecionada. Inicialmente os botões vêm desmarcados para o usuário que poderá, através do mouse, clicar sobre a sua opção.

Um **ButtonGroup** reúne uma quantidade (2 ou mais) de **JRadioButton**. Para tanto, é necessário criar um objeto do tipo **ButtonGroup** e a ele adicionar os **JRadioButtons** desejados.

```
ButtonGroup bgroup = new ButtonGroup();  
bgroup.add(radioButton1);  
bgroup.add(radioButton2);  
bgroup.add(radioButton3);
```

CONSTRUTOR

```
String      texto;      // texto do RadioButton  
boolean     estado;     // estado do RadioButton (true ou false)  
Icon imagem;           // imagem associada ao RadioButton
```

```
JRadioButton rb = new JRadioButton(texto);  
JRadioButton rb = new JRadioButton(texto, estado);  
JRadioButton rb = new JRadioButton(imagem);  
JRadioButton rb = new JRadioButton(imagem, estado);  
JRadioButton rb = new JRadioButton(texto, imagem);  
JRadioButton rb = new JRadioButton(texto, imagem, estado);
```


JRadioButton



JScrollPane

JScrollPane

Um objeto da classe **JScrollPane** é um componente que permite a exibição de um conteúdo em uma janela com barras de rolagem. Este objeto é útil quando temos uma porção pequena da tela e necessitamos exibir uma grande quantidade de informação ou imagem.

CONSTRUTOR

```
JScrollPane sp = new JScrollPane(imagem);  
JScrollPane sp = new JScrollPane(imagem, vsbPolicy, hsbPolicy);  
JScrollPane sp = new JScrollPane(vsbPolicy, hsbPolicy);
```

Onde:

- **vsbPolicy** → VERTICAL_SCROLLBAR_AS_NEEDED
VERTICAL_SCROLLBAR_NEVER
VERTICAL_SCROLLBAR_ALWAYS
- **hsbPolicy** → HORIZONTAL_SCROLLBAR_AS_NEEDED
HORIZONTAL_SCROLLBAR_NEVER
HORIZONTAL_SCROLLBAR_ALWAYS

JScrollPane



Ainda bem que
é apenas um
exemplo e não
condiz com a
realidade!!!!



JTextFieldComponent

Um objeto da classe **JTextField** permite a edição de textos pelo usuário. Os componentes pertencentes à esta classe são:

- **JTextField**
- **JTextArea**
- **JEditorPane**
- **JPasswordField**

JTextField

Componentes **JTextField** têm dois tipos de uso:

ENTRADA → o usuário pode entrar com uma linha de texto (String)

SAÍDA → exibe o conteúdo de uma linha de texto

CONSTRUTOR

```
JTextField tf = new JTextField(int colunas);  
JTextField tf = new JTextField(String inicial);  
JTextField tf = new JTextField(String inicial, int colunas);
```

JTextComponent

*** CADASTRO DE ALUNOS ***



NOME

SEXO

☐ Feminino ☐ Masculino

IDADE

OBSERVAÇÃO

Rapazes! Alistem-se no serviço militar

JTextArea

JTextArea

Componentes **JTextArea** tem praticamente a mesma funcionalidade de um **JTextField**, porém, pode conter múltiplas linhas. Sendo assim, muitos métodos são comuns aos dois componentes.

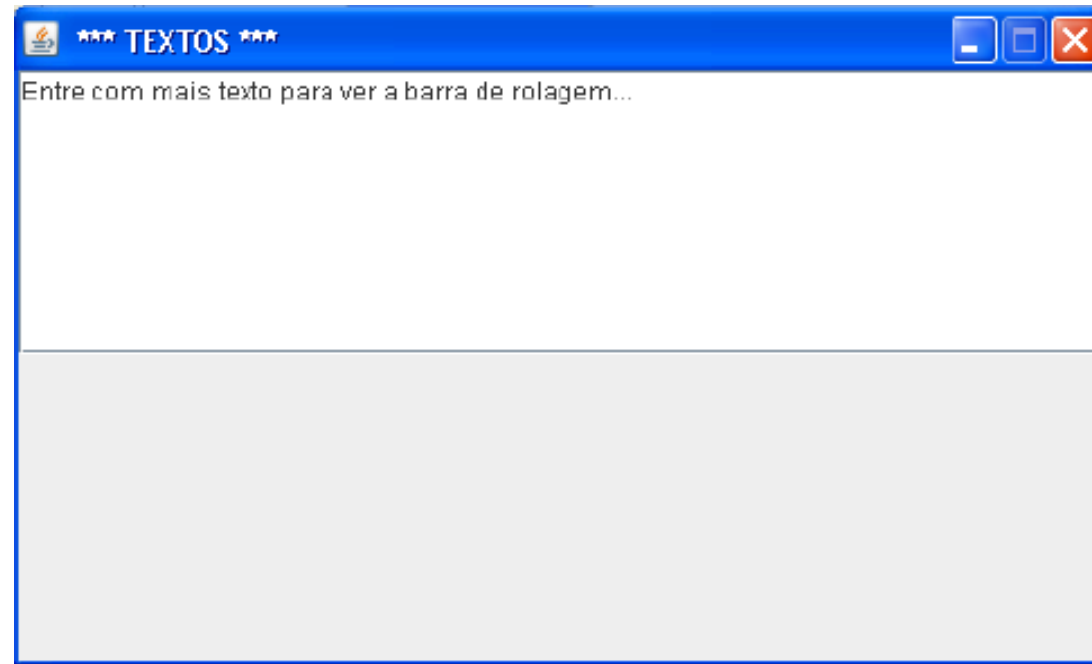
CONSTRUTOR

```
JTextArea ta = new JTextArea(int linhas, int colunas)
```

```
JTextArea ta = new JTextArea(String conteúdo)
```

```
JTextArea ta = new JTextArea(String conteúdo, int linhas, int colunas)
```

JTextArea

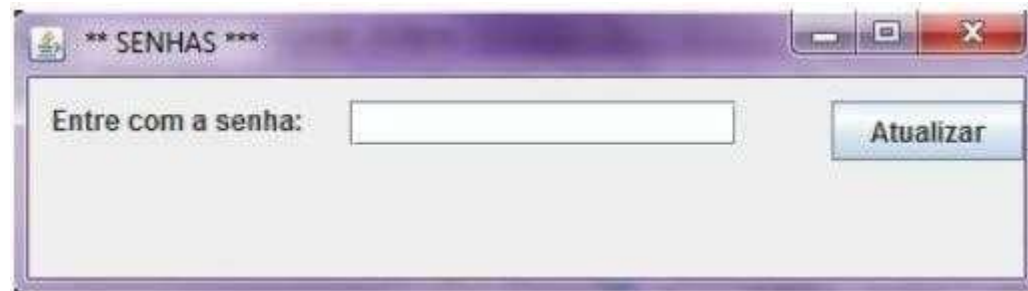


JPasswordField

Componentes **JPasswordField** tem a mesma funcionalidade de um **JTextField**, porém, os caracteres digitados não aparecem ("ecoam") na tela. O caractere de máscara padrão é o * (asterisco), mas é possível modificar este caractere com o método apropriado.

CONSTRUTOR

```
JPasswordField pf = new JPasswordField (int colunas)  
JPasswordField pf = new JPasswordField (String texto)  
JPasswordField pf = new JPasswordField (String texto, int colunas)
```



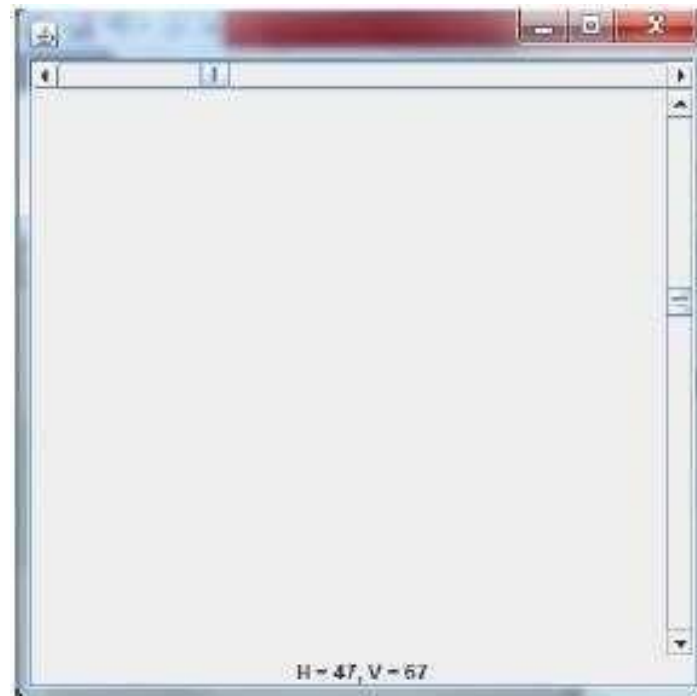
JScrollBar

JScrollBar

O **JScrollBar** permite que o usuário selecione graficamente um valor deslizando um botão dentro de um intervalo limitado.

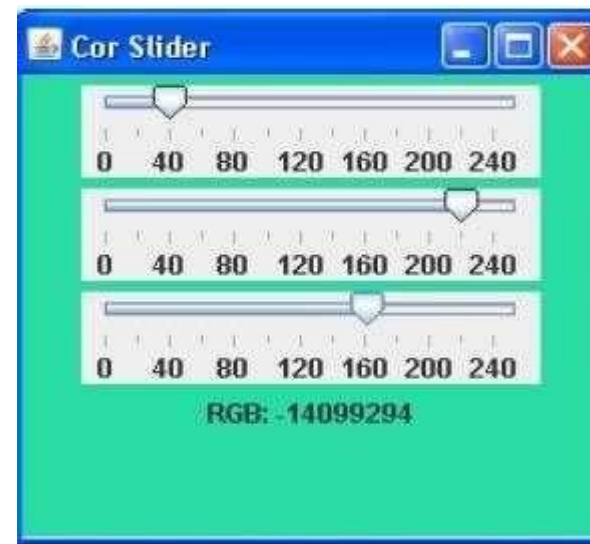
Para trabalhar com eles, você deve utilizar um `AdjustmentListener` através do método `JScrollbar.addAdjustmentListener()`. Quando ocorrer algum evento de ajuste, o método ouvinte será chamado.

JScrollBar



JSlider

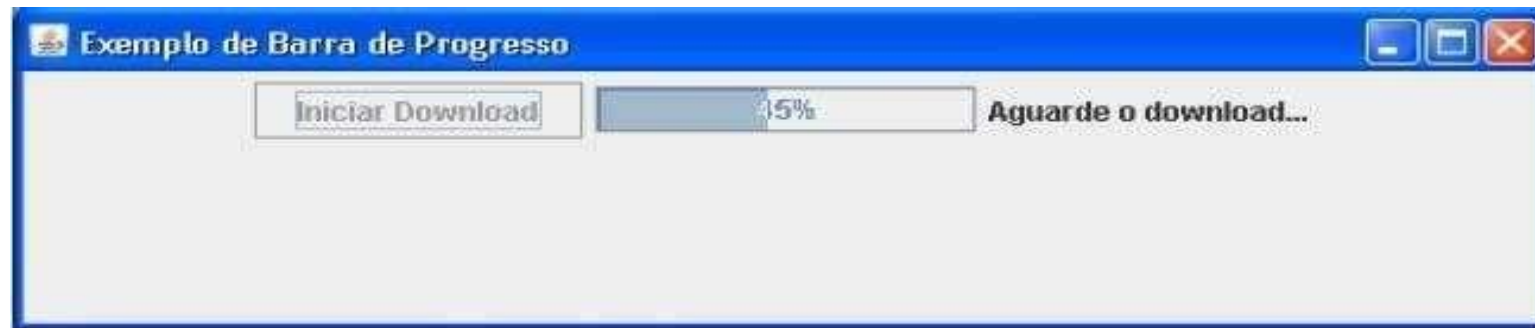
O **JSlider** permite que o usuário selecione graficamente um valor, através do deslizamento de um botão dentro de um intervalo limitado. As marcas da escala (principais e menores) podem ser mostradas, ambas com intervalos variáveis.



JProgressBar

JProgressBar

Um objeto do tipo **JProgressBar** cria uma barra que mostra o progresso da execução de uma determinada tarefa/atividade.



JComboBox

JComboBox

Um objeto do tipo **JComboBox** cria uma caixa de combinação (lista drop-down) onde o usuário pode escolher apenas uma opção. Esta lista pode ser editável ou não.



JList

JList

Um objeto do tipo **JList** cria uma lista suspensa onde o usuário pode selecionar um ou mais itens.



Menus

Menus

A classe Swing oferece uma série de componentes para a construção de menus. Os itens de um menu, na verdade, tem seu funcionamento e estrutura semelhantes aos botões (JButton).

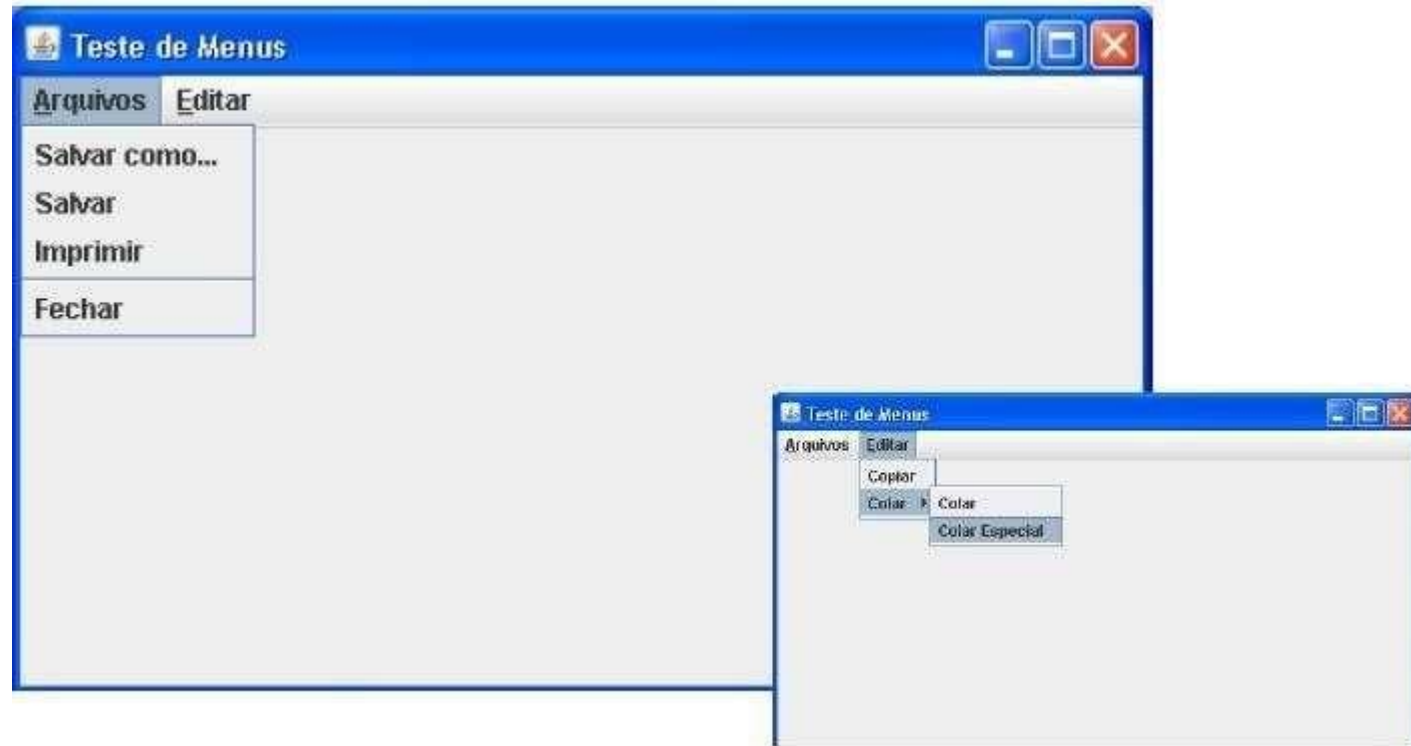
Observações:

- Um item de um menu pode ter texto e ícones
- Os itens de um menu podem ser RadioButton ou CheckBox
- Cada item de menu pode ter uma combinação de teclas de atalho associado à ele

Os componentes são:

JMenuItem
JCheckBoxMenuItem
JMenu
JMenuBar

Menus



Dúvidas?

