



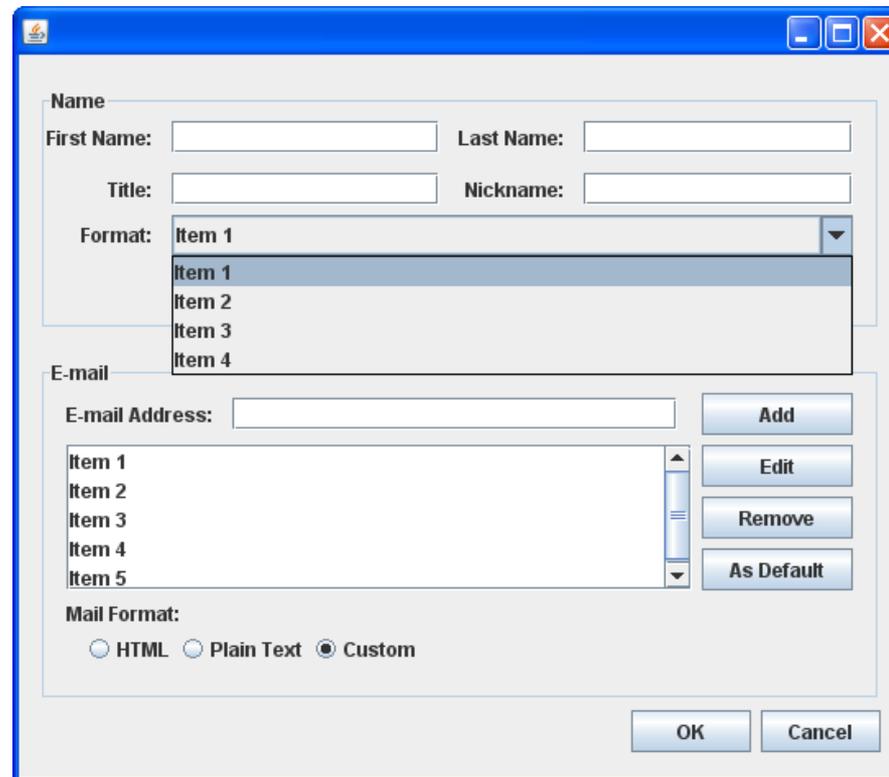
Técnicas de Programação II

Aula 03 – GUI e Swing

Edirlei Soares de Lima
<edirlei.lima@uniriotec.br>

GUI – Graphical User Interface

- A API Java fornece diversas classes destinadas a criação de interfaces gráficas.

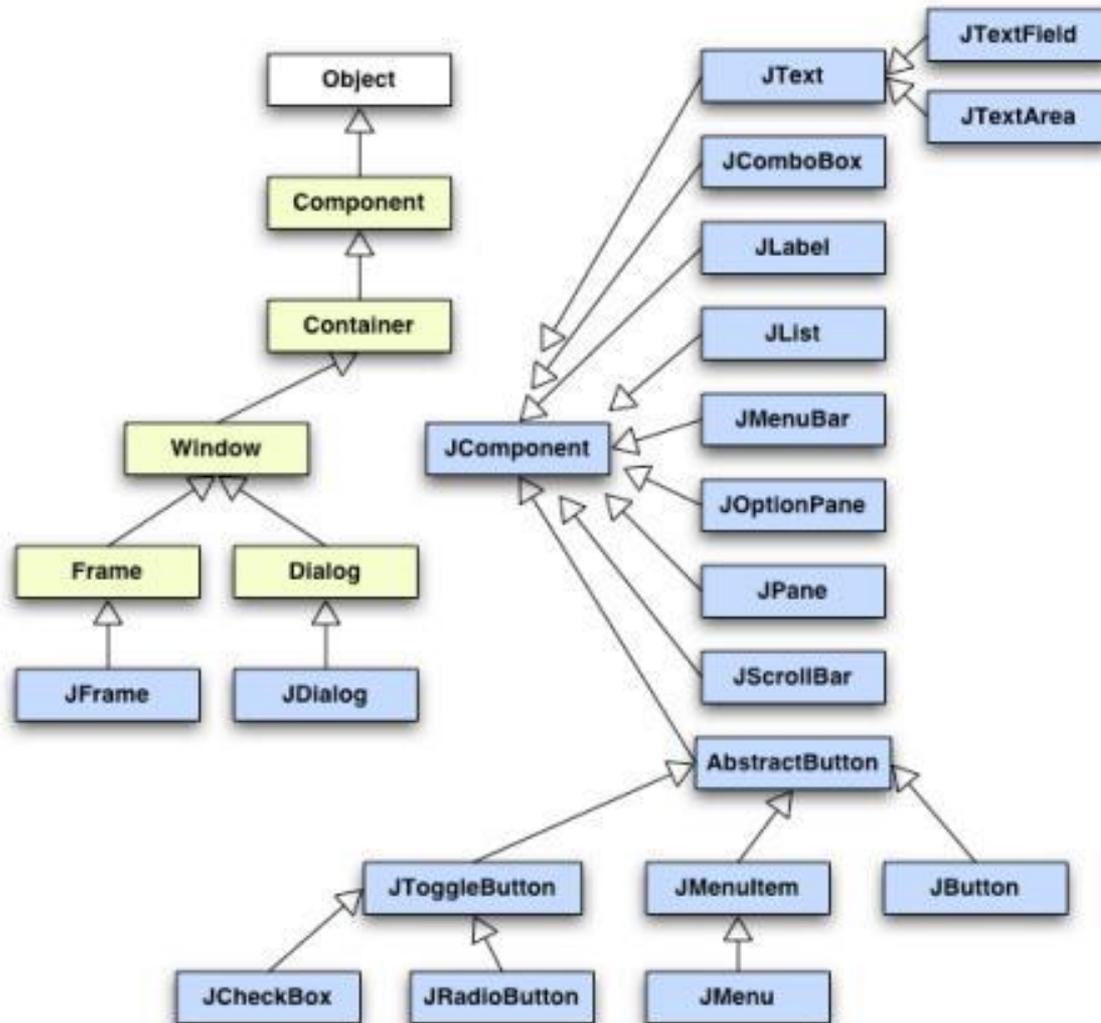


The image shows a Java Swing dialog box with a blue title bar and standard window controls (minimize, maximize, close). The dialog is titled "Name" and contains several input fields and a list. The "Name" section includes fields for "First Name:", "Last Name:", "Title:", and "Nickname:". Below these is a "Format:" dropdown menu currently showing "Item 1", with a list of options: "Item 1", "Item 2", "Item 3", and "Item 4". The "E-mail" section features an "E-mail Address:" field, a list of five items ("Item 1" through "Item 5"), and four buttons: "Add", "Edit", "Remove", and "As Default". At the bottom, there are radio buttons for "Mail Format:" with options "HTML", "Plain Text", and "Custom" (which is selected). "OK" and "Cancel" buttons are located at the bottom right of the dialog.

AWT e Swing

- **AWT (Abstract Window Toolkit):**
 - API padrão para criação de componentes GUI no início da plataforma Java (entre 1995 até 1998);
 - Os objetos AWT são construídos sobre objetos de código nativo do Sistema Operacional em uso;
 - Os componentes GUI originais do pacote `java.awt` estão diretamente associados com as capacidades de GUI da plataforma local.
- **Swing:**
 - API escrita puramente em Java (padrão desde 1998);
 - `JLabel`, `JButton`, `JTextField`, etc, são componentes GUI do pacote `javax.swing`.
 - Mais flexível que o `java.awt` porque é implementada toda em Java, enquanto que `java.awt` é implementada em código nativo.

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.

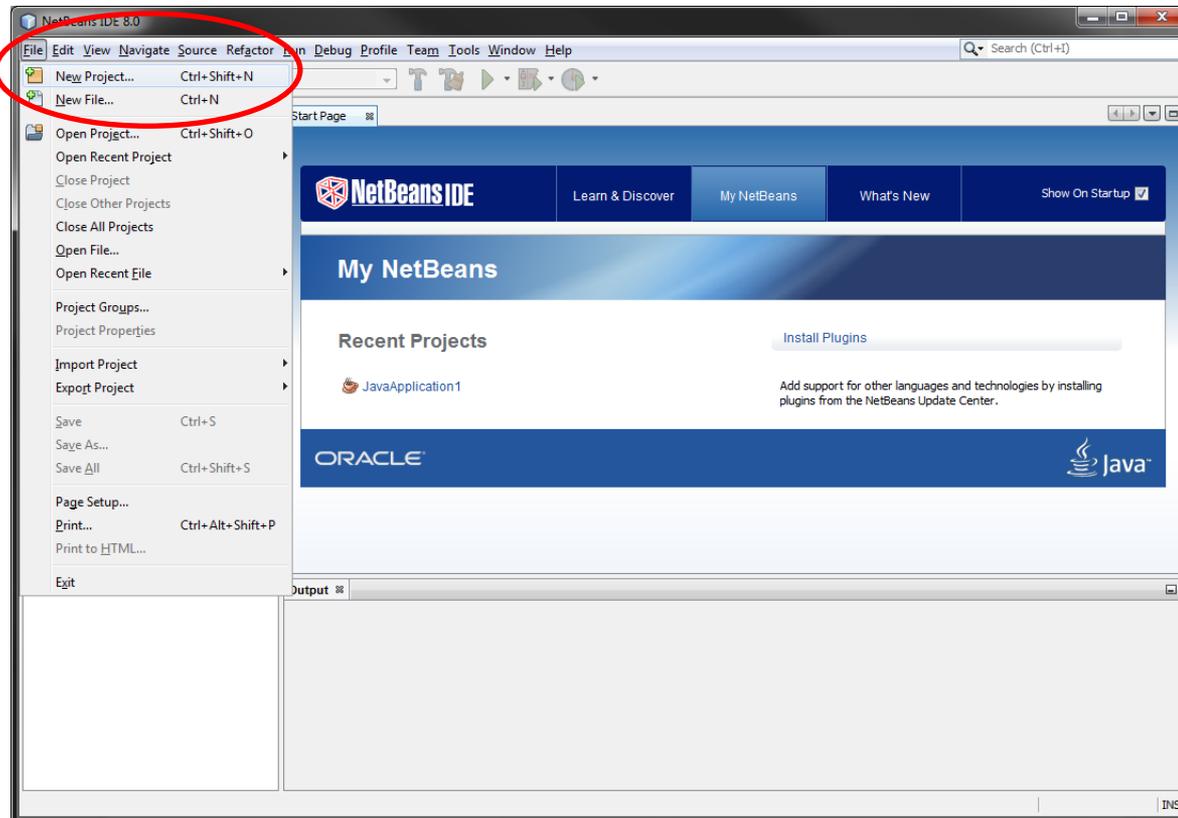
Netbeans GUI Builder

The screenshot displays the NetBeans IDE interface for the GUI Builder. The main window shows the design of the 'Antenna.java' form. The form is divided into two sections: 'Position/Direction' and 'System'. The 'Position/Direction' section includes a 'Direction [°]' field set to 140.000, a 'Height [m]' field set to 110.000, and a checkbox for 'Height is Lower Edge (Not Center)'. The 'System' section includes fields for 'Channels' (2), 'Watts' (12.000), 'Antenna Type' (Kathrein 742151), 'Electrical Downtilt From [°]' (0.000) to 'To' (10.000), 'Polarization' (X +45°), and 'Frequency From [MHz]' (943.000) to 'To' (951.000). Each numerical field has an 'Adjust' button next to it. The 'OK' and 'Cancel' buttons are at the bottom of the form.

The left sidebar shows the project structure for 'GUIFormExamples', including 'Source Packages', 'examples', and 'Libraries'. The 'examples' folder contains 'Antenna.java', 'Antenna.properties', 'ContactEditor.java', 'ContactEditor.properties', 'Find.java', and 'NewJFrame.java'. The 'jLabel2 [JLabel] - Navigator' window shows the component hierarchy for the 'Form Antenna', including 'JFrame', 'jPanel1 [JPanel]', 'jLabel1 [JLabel]', 'jLabel2 [JLabel]', 'jTextField1 [JTextField]', 'jTextField2 [JTextField]', 'jCheckBox1 [JCheckBox]', 'jPanel2 [JPanel]', 'jButton3 [JButton]', and 'jButton4 [JButton]'. The 'jLabel2 [JLabel] - Properties' window shows the properties for the selected 'jLabel2' component, including 'background', 'displayedMnemonic', 'font' (Tahoma 11 Plain), 'foreground' (black), 'horizontalAlignment' (LEADING), 'icon', 'labelFor' (<none>), 'text' (Height [m]:), 'toolTipText', and 'verticalAlignment' (CENTER).

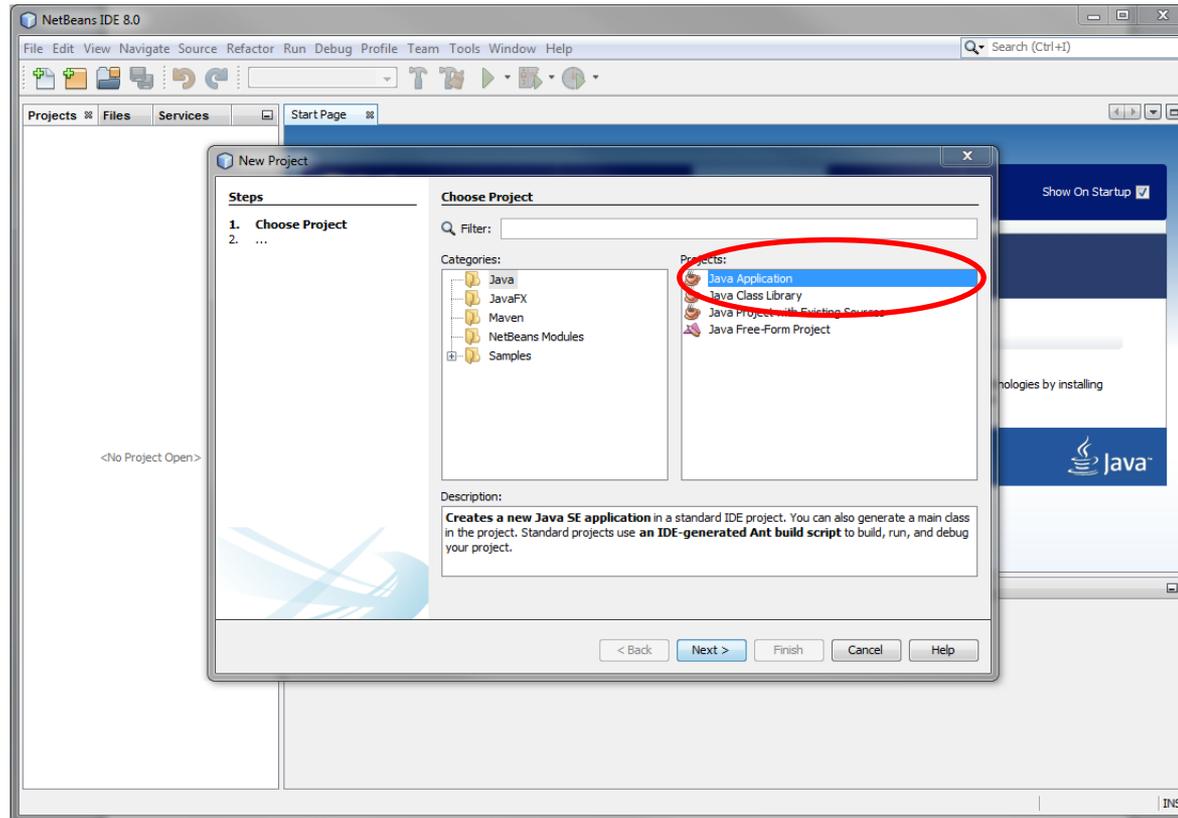
Netbeans GUI Builder – Criando Projeto

1) Acesse o menu File->New Project...



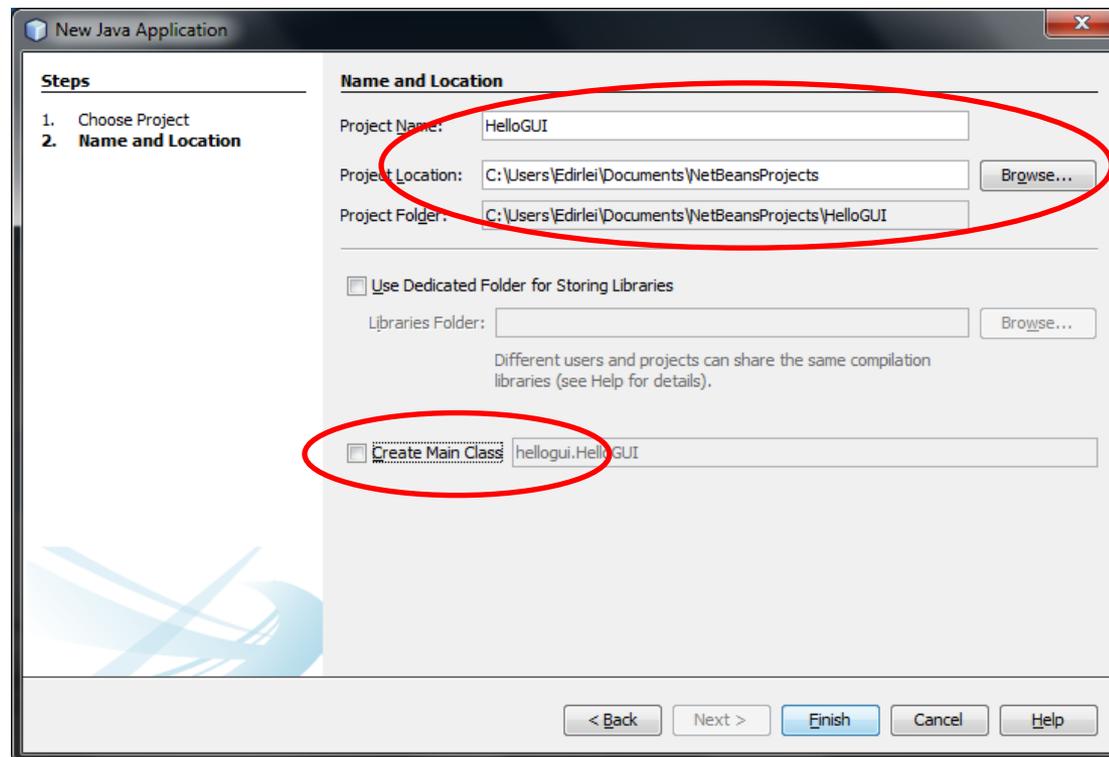
Netbeans GUI Builder – Criando Projeto

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



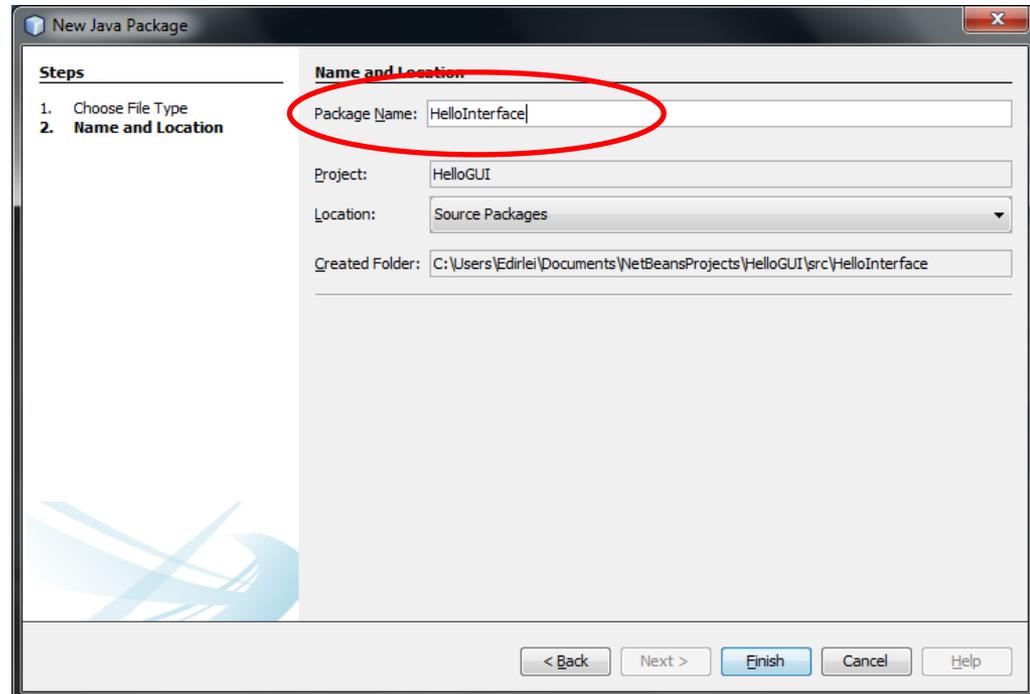
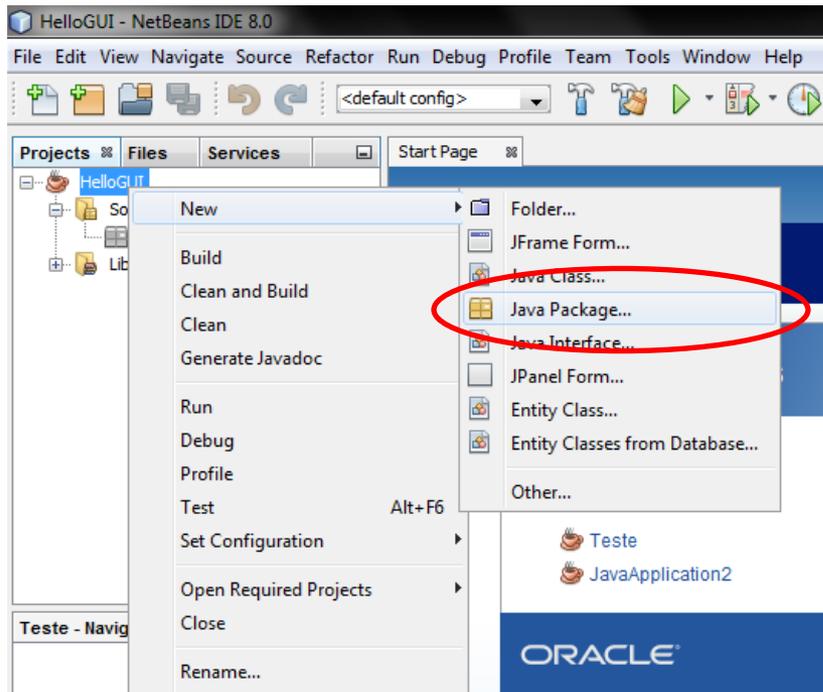
Netbeans GUI Builder – Criando Projeto

- 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”:



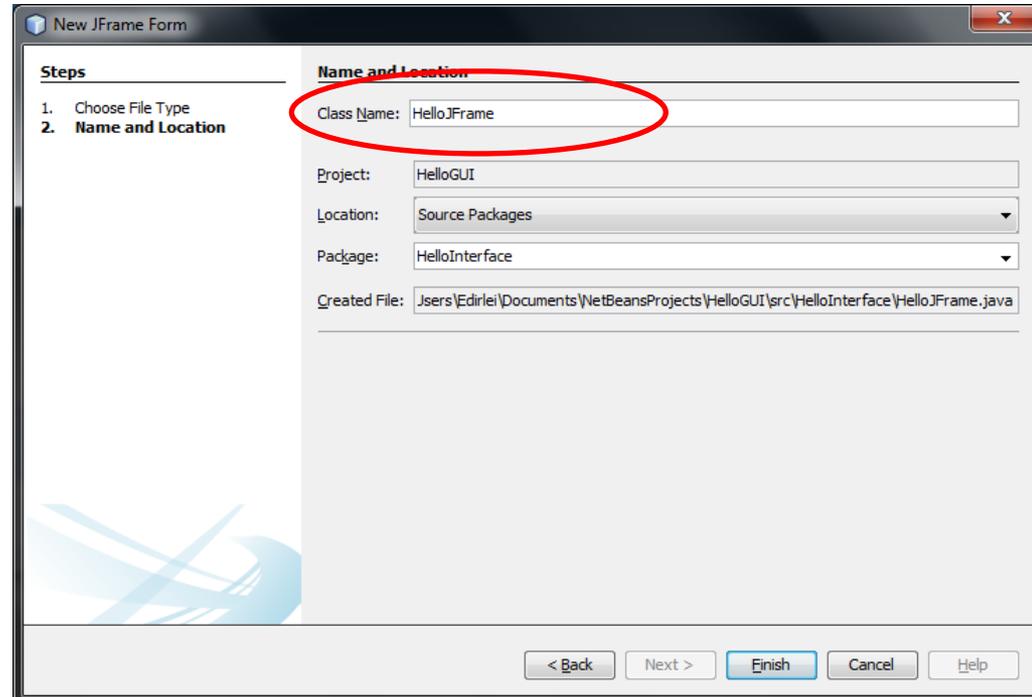
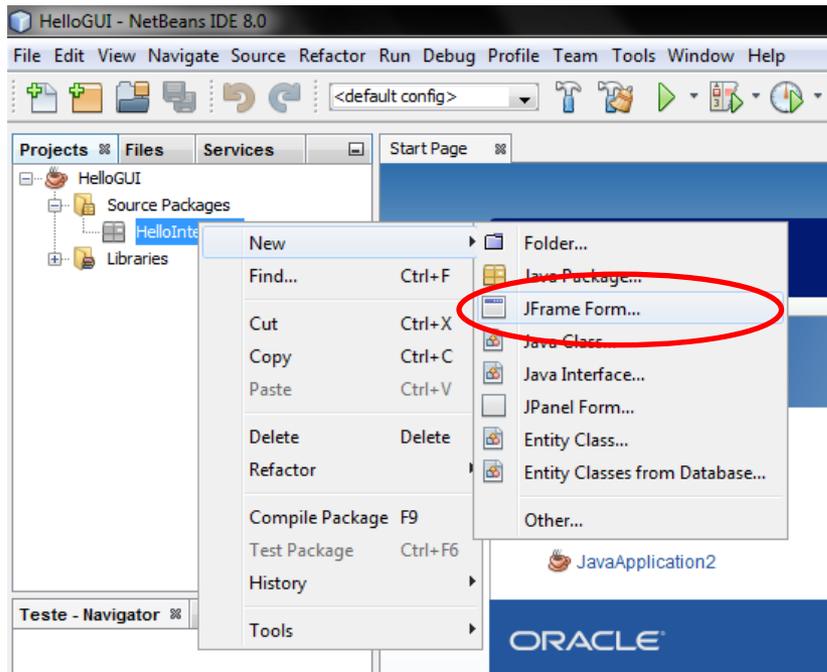
Netbeans GUI Builder – Criando Projeto

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

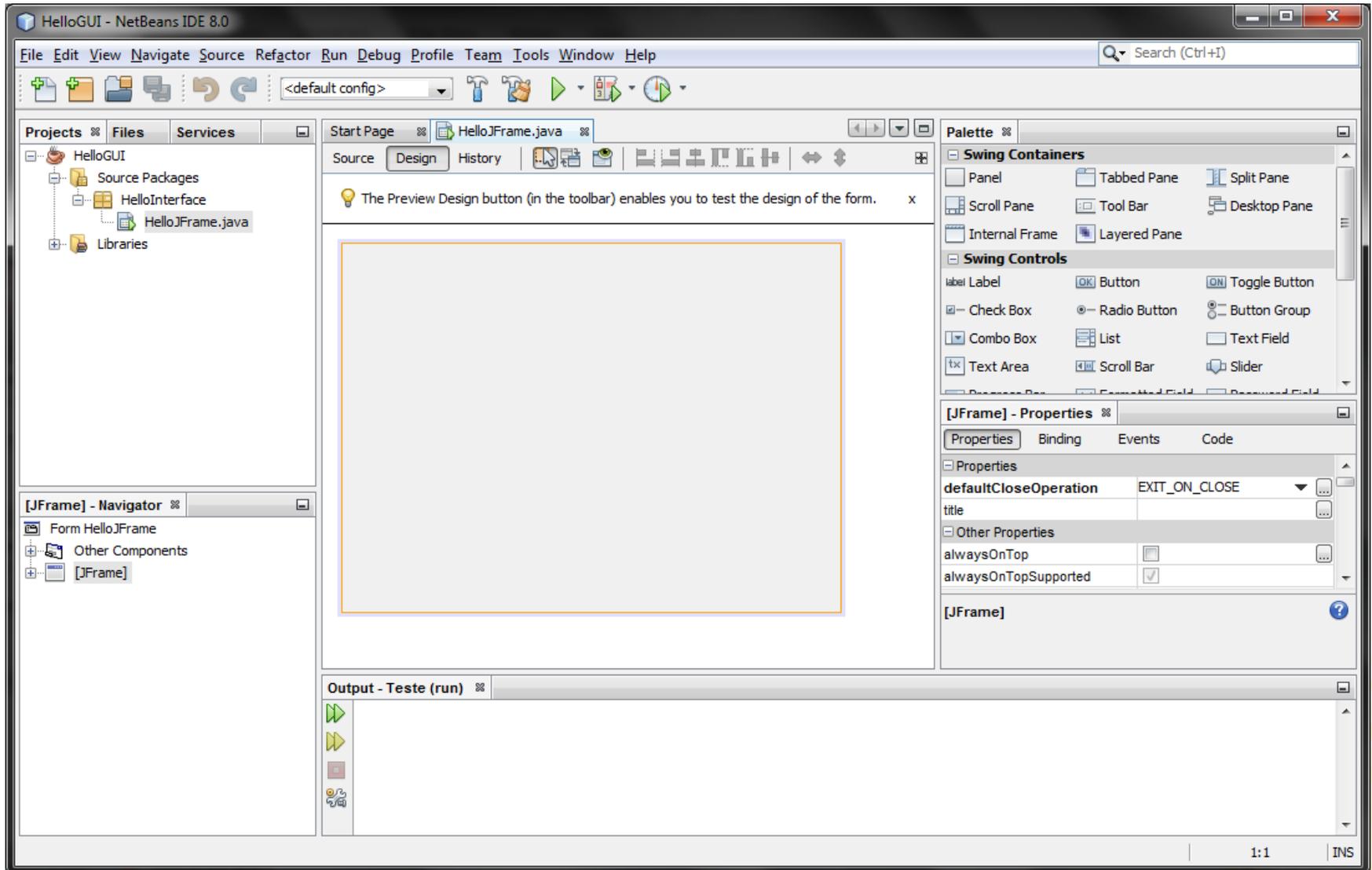


Netbeans GUI Builder – Criando Projeto

5) Crie um novo “JFrame Form”:

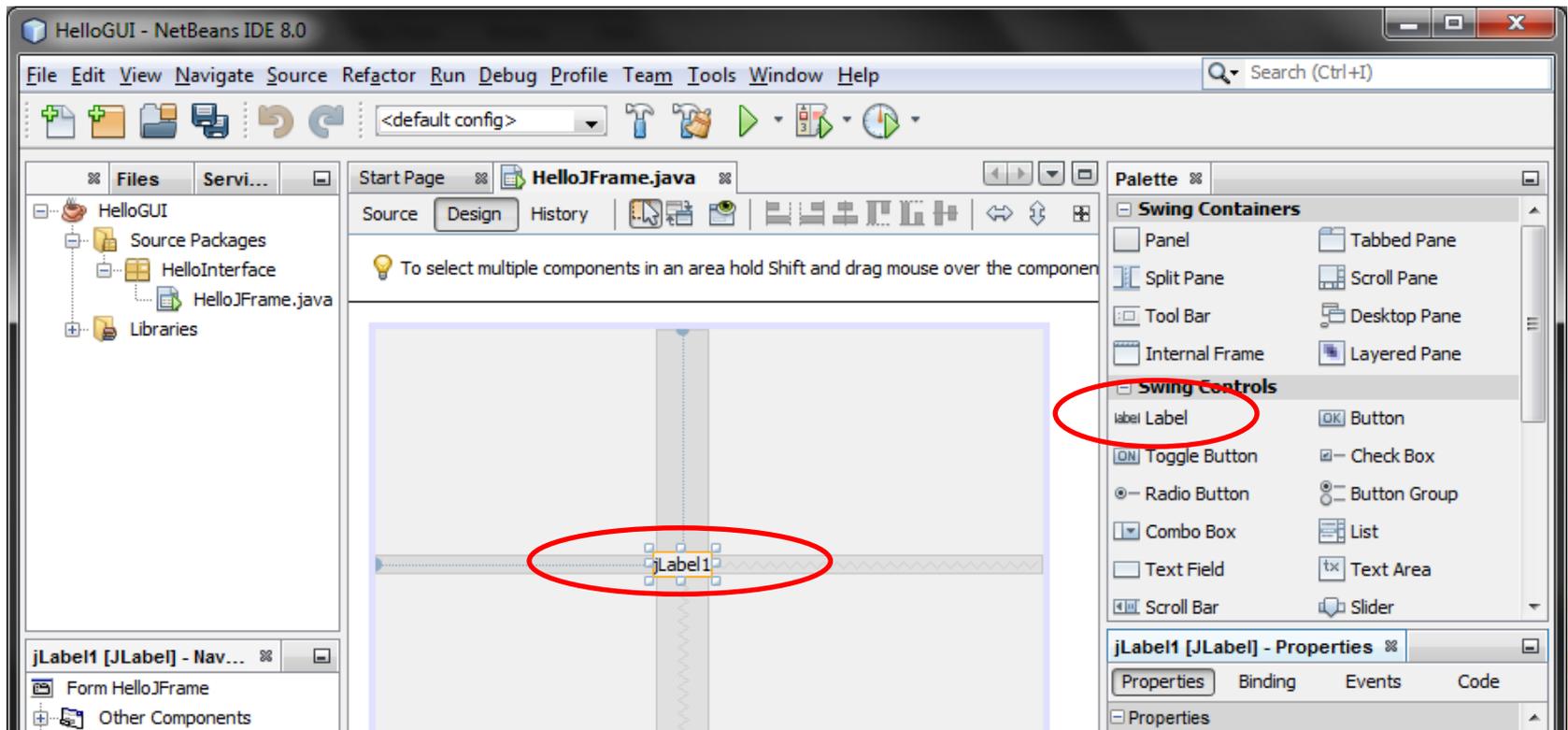


Netbeans GUI Builder



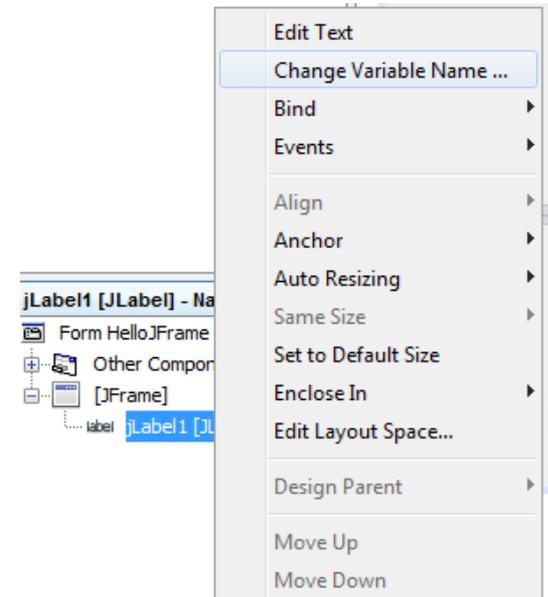
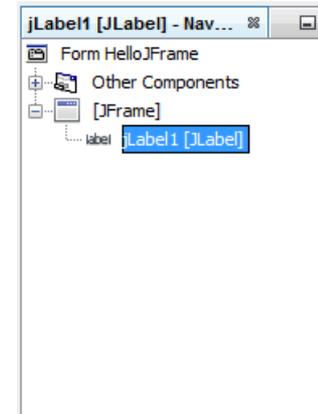
Componentes Básicos - Label

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



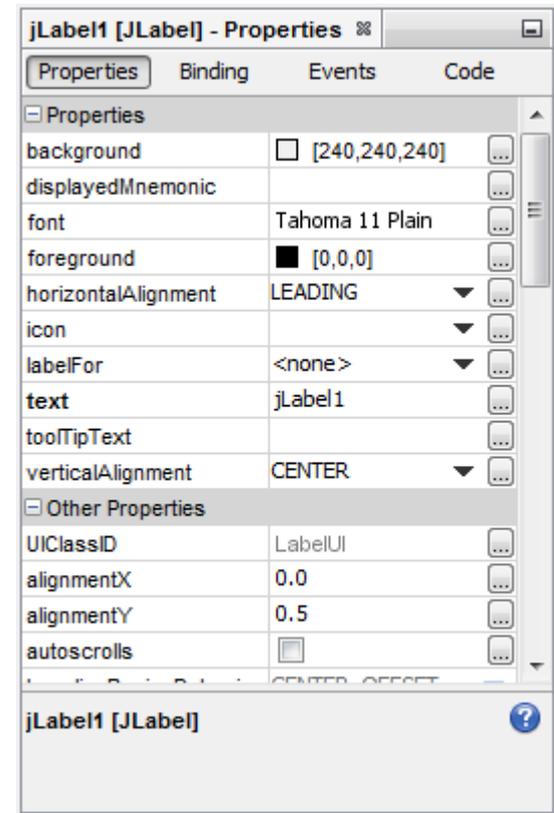
Componentes Básicos - Label

- 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.



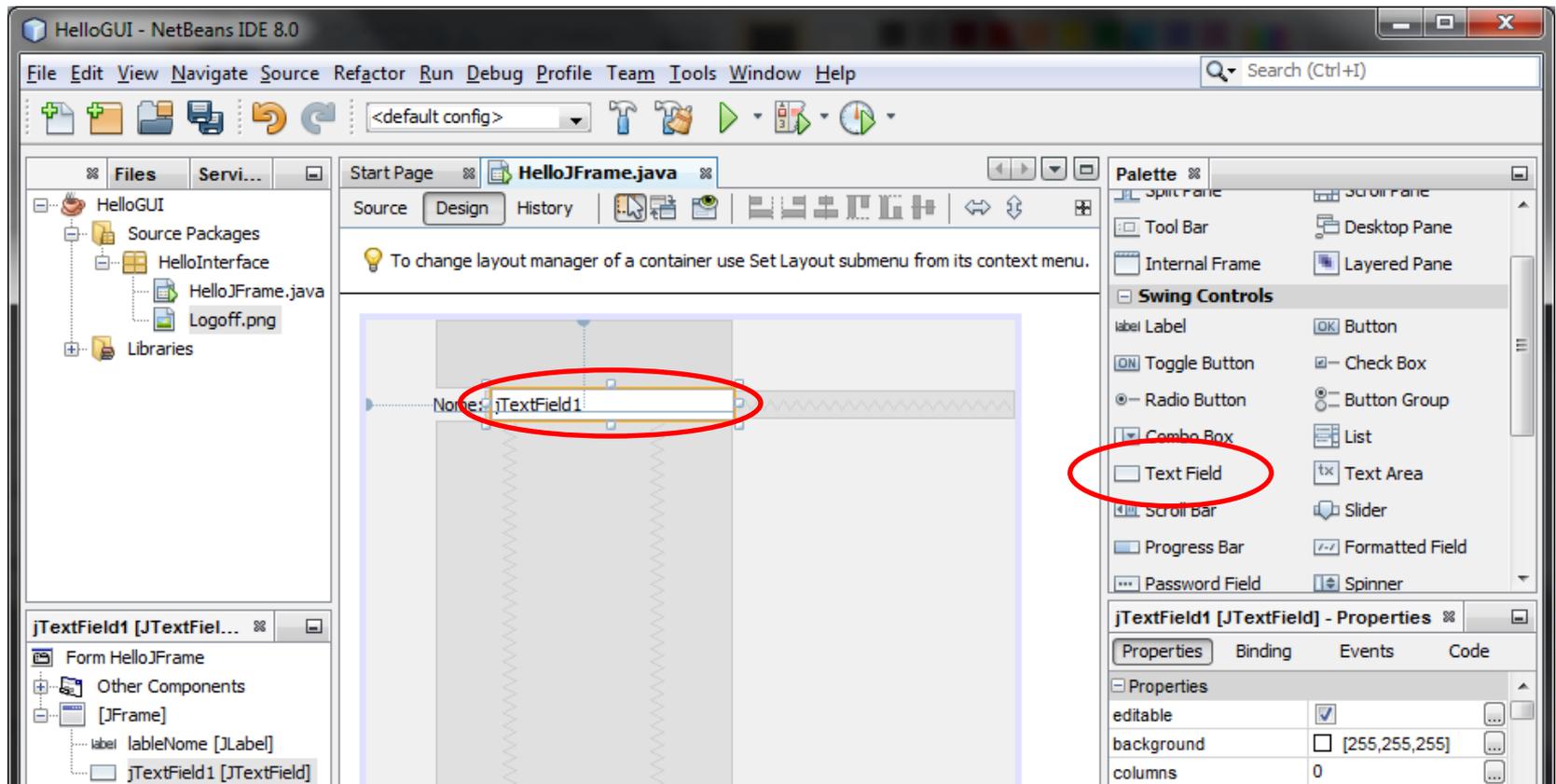
Componentes Básicos - Label

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



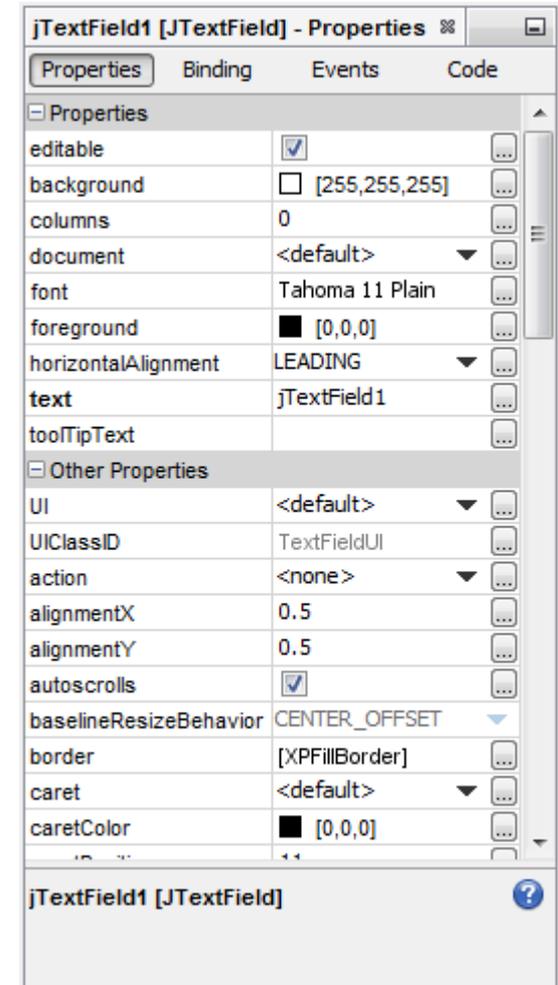
Componentes Básicos – TextField

- Componente para entrada, edição e exibição de texto.



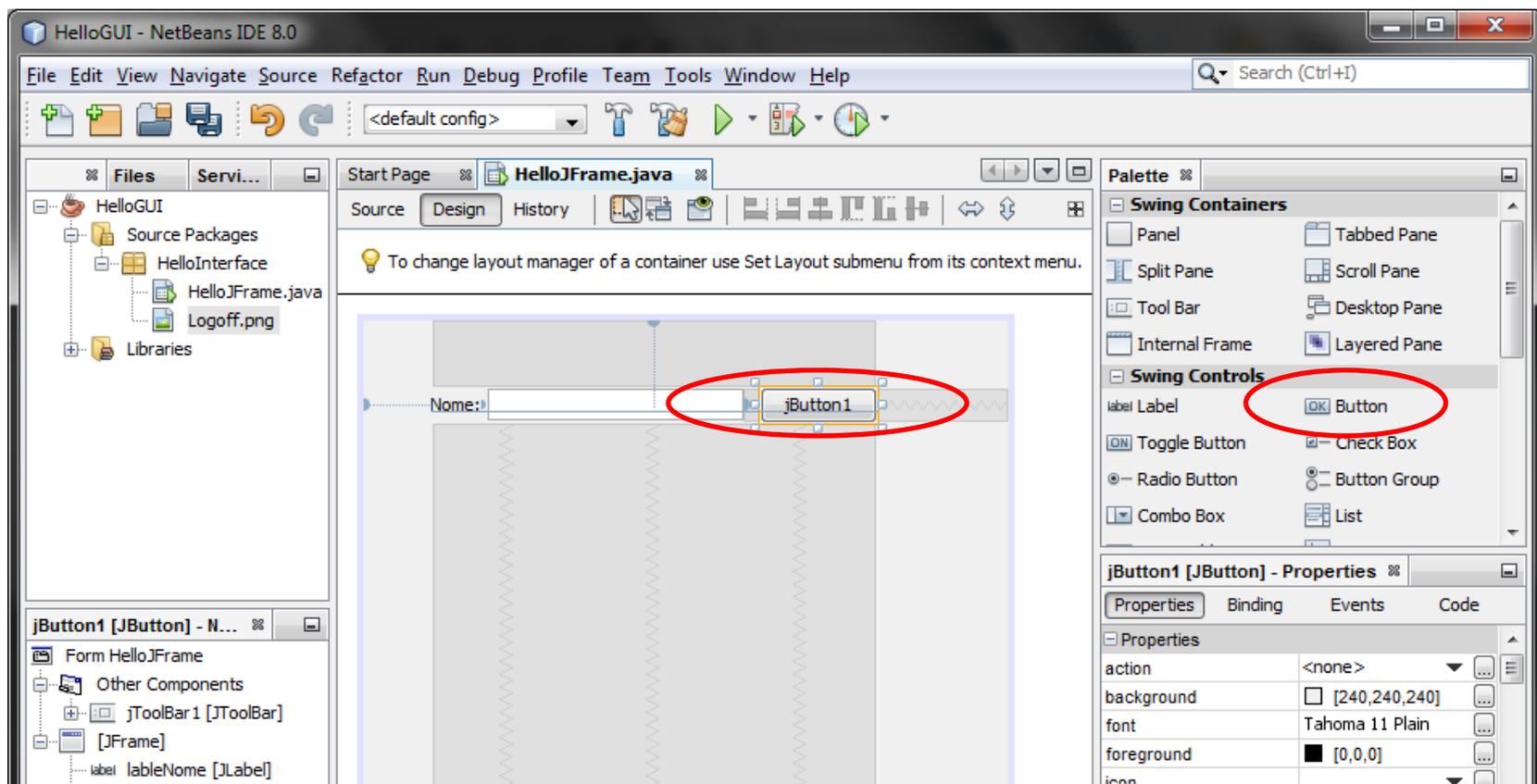
Componentes Básicos – TextField

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



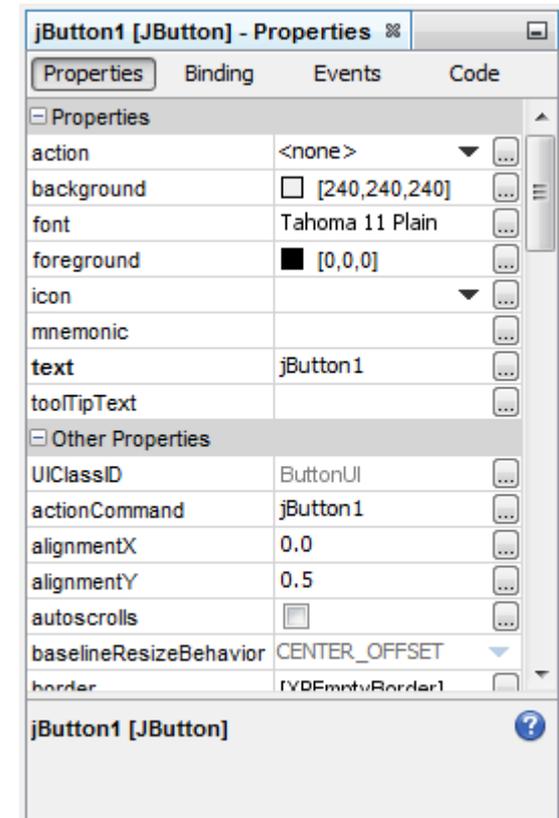
Componentes Básicos – Button

- Componente que representa um botão.



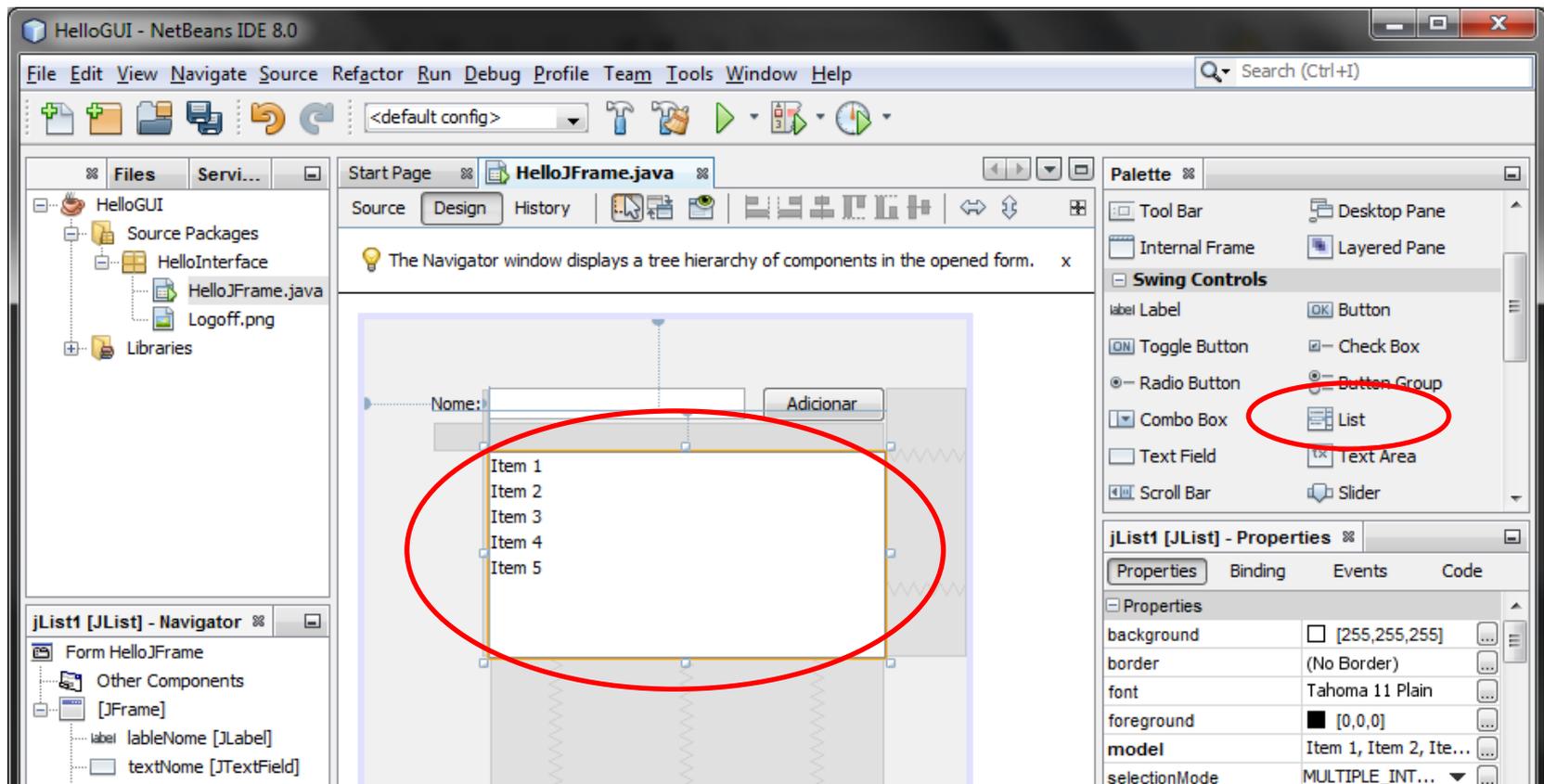
Componentes Básicos – Button

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



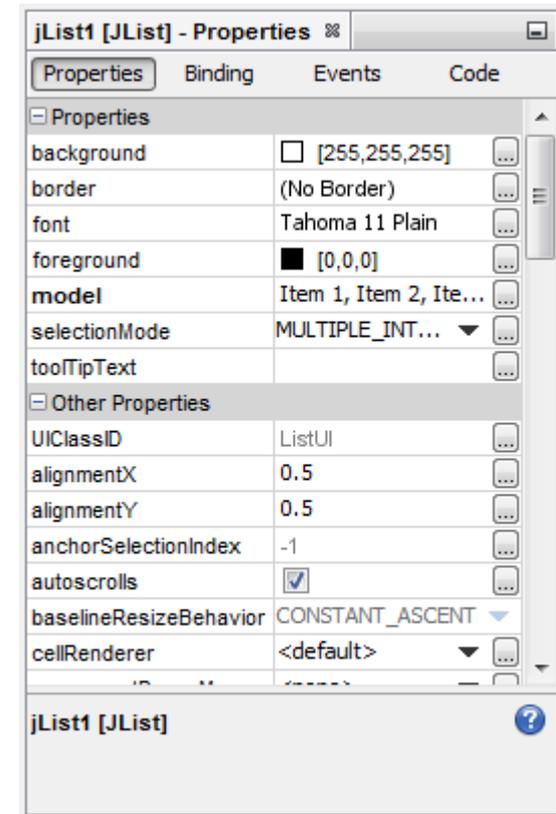
Componentes Básicos – List

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

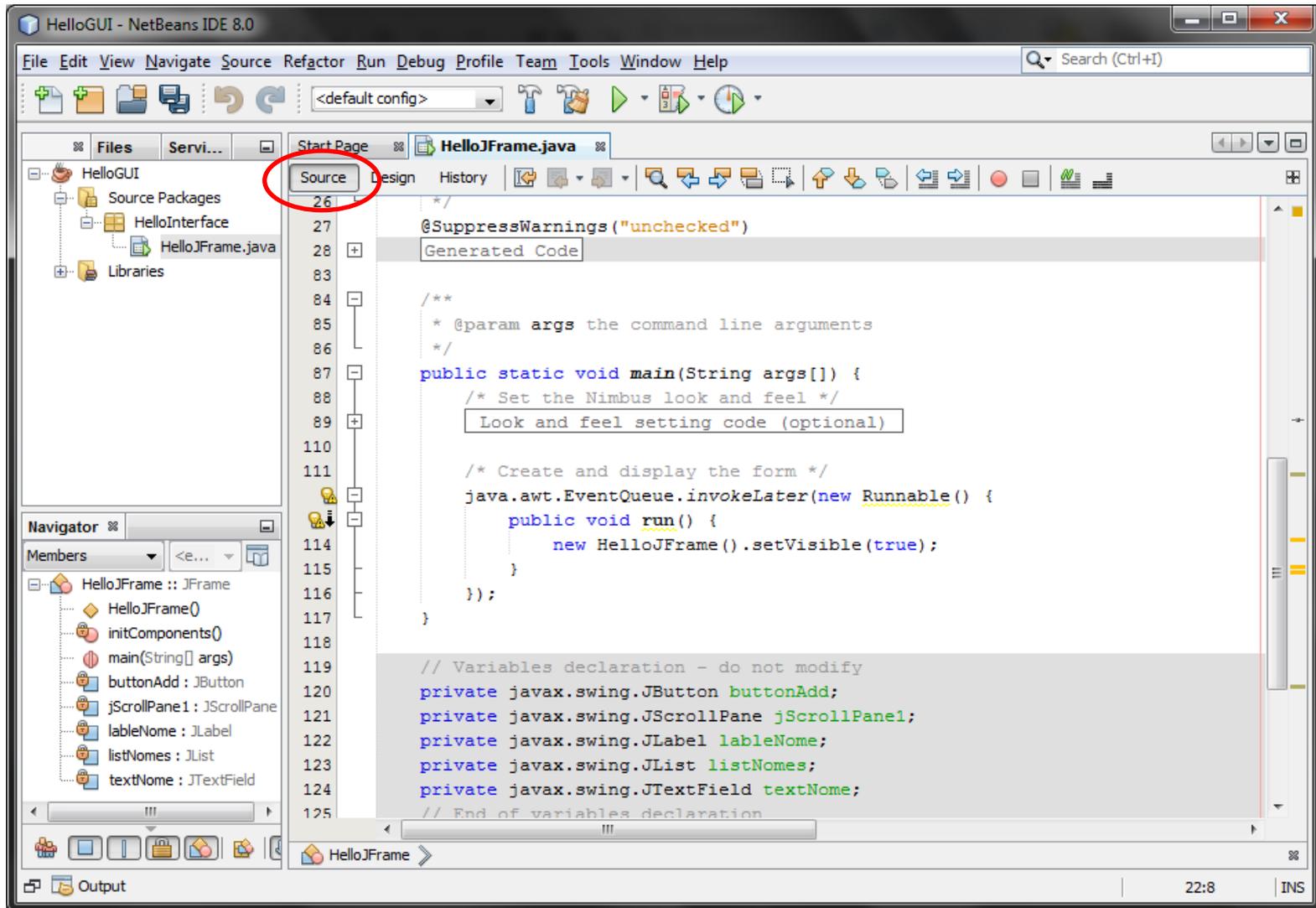


Componentes Básicos – List

- Principais Propriedades (JList):
 - model;
 - selectionMode;
 - selectedIndex;
 - visibleRowCount;
 - foreground;
 - background;
 - font;
 - toolTipText;
 - border;
 - enabled;

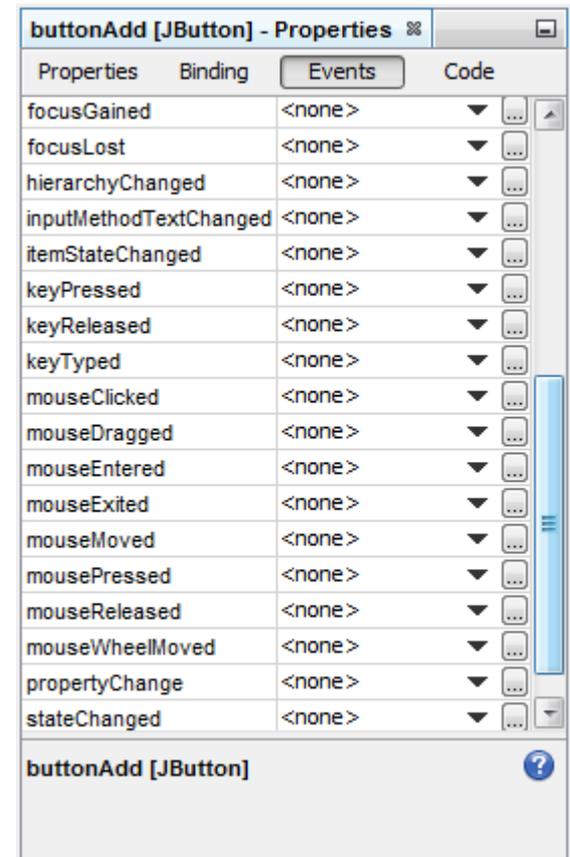


Netbeans GUI Builder – Código Gerado



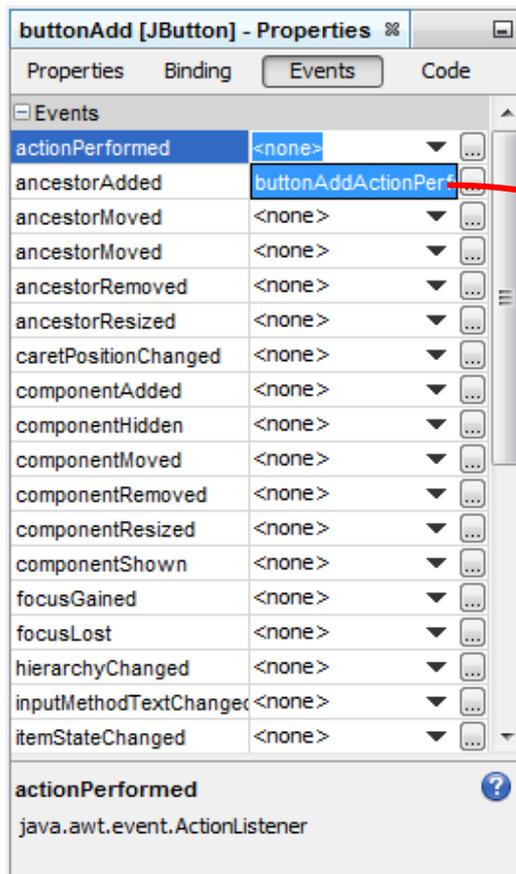
Eventos – Button

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



Evento actionPerformed – Button

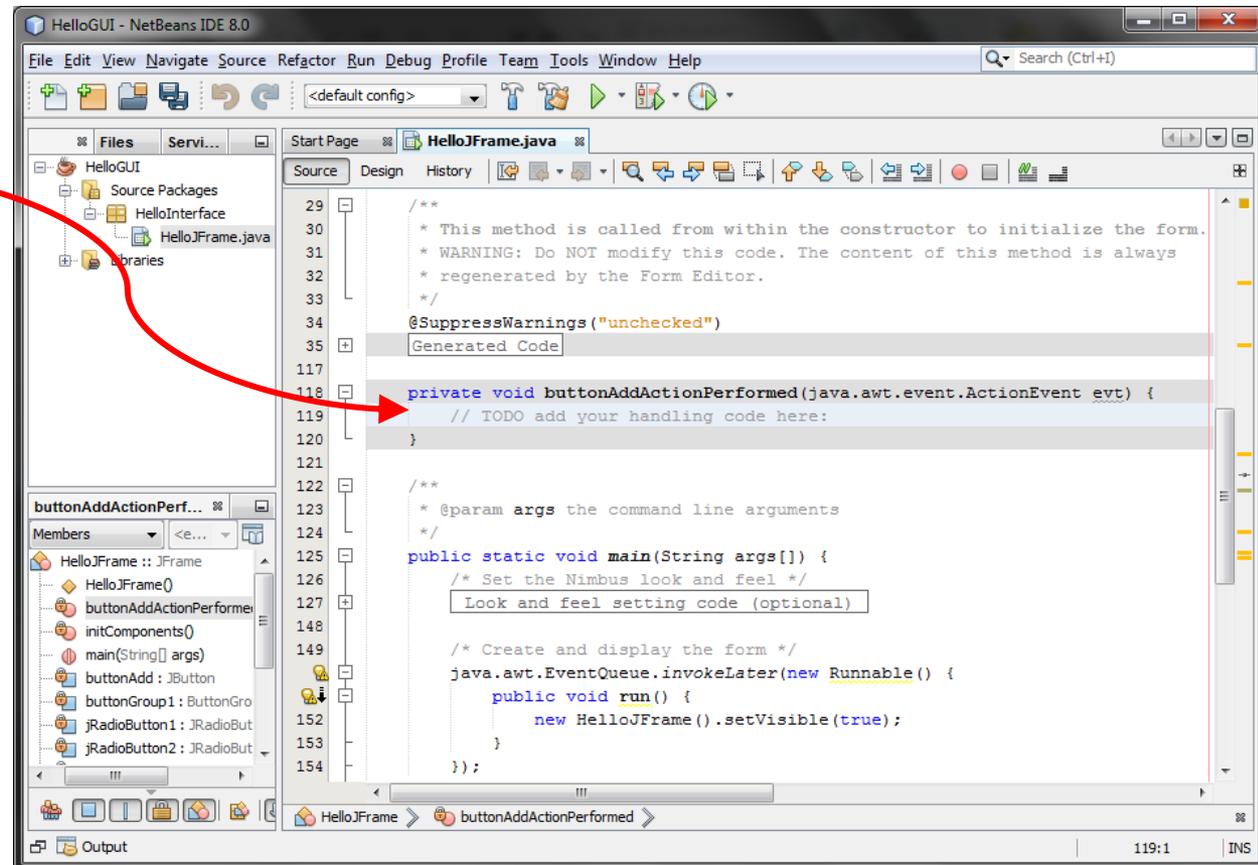
- Criando um evento de ativação para o botão (actionPerformed):



The screenshot shows the 'Properties' window for a JButton component. The 'Events' tab is selected, displaying a list of events and their current handlers. The 'actionPerformed' event is highlighted, and its handler is set to 'buttonAddActionPerformed'. Below the list, the details for the 'actionPerformed' event are shown, indicating it is implemented by 'java.awt.event.ActionListener'.

Event	Handler
actionPerformed	<none>
ancestorAdded	buttonAddActionPerformed
ancestorMoved	<none>
ancestorMoved	<none>
ancestorRemoved	<none>
ancestorResized	<none>
caretPositionChanged	<none>
componentAdded	<none>
componentHidden	<none>
componentMoved	<none>
componentRemoved	<none>
componentResized	<none>
componentShown	<none>
focusGained	<none>
focusLost	<none>
hierarchyChanged	<none>
inputMethodTextChanged	<none>
itemStateChanged	<none>

actionPerformed
java.awt.event.ActionListener



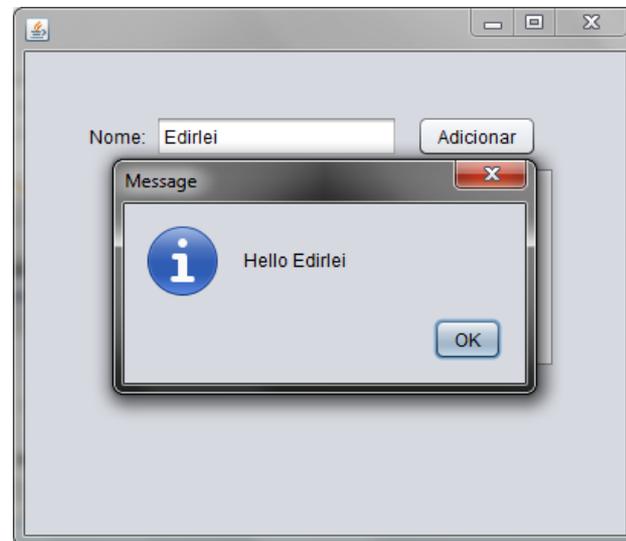
The screenshot shows the 'Source' window of the NetBeans IDE. The file 'HelloJFrame.java' is open, and the 'buttonAddActionPerformed' method is highlighted. A red arrow points from the 'buttonAddActionPerformed' handler in the Properties window to this method in the code. The code includes a comment indicating that this method is called from within the constructor to initialize the form and is regenerated by the Form Editor. The method signature is 'private void buttonAddActionPerformed(java.awt.event.ActionEvent evt)'. Below it, there is a comment '// TODO add your handling code here:'. The 'main' method is also visible, showing the creation and display of the form.

```
29 /**
30  * This method is called from within the constructor to initialize the form.
31  * WARNING: Do NOT modify this code. The content of this method is always
32  * regenerated by the Form Editor.
33  */
34  @SuppressWarnings("unchecked")
35  Generated Code
117
118 private void buttonAddActionPerformed(java.awt.event.ActionEvent evt) {
119     // TODO add your handling code here:
120 }
121
122 /**
123  * @param args the command line arguments
124  */
125 public static void main(String args[]) {
126     /* Set the Nimbus look and feel */
127     Look and feel setting code (optional)
128
129     /* Create and display the form */
130     java.awt.EventQueue.invokeLater(new Runnable() {
131         public void run() {
132             new HelloJFrame().setVisible(true);
133         }
134     });
135 }
```

Evento actionPerformed – Button

- **Exemplo 1** – Mostrar mensagem com o conteúdo do TextField:

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



Evento actionPerformed – Button

- **Exemplo 2** – Adicionar o conteúdo do TextField na List:
 - a) Crie um objeto do tipo `DefaultListModel` para armazenar os elementos da lista:

```
...
public class HelloJFrame extends javax.swing.JFrame {
    private DefaultListModel listModel = new DefaultListModel();
    ...
}
```

- b) No método construtor da classe, associe o `listModel` a lista:

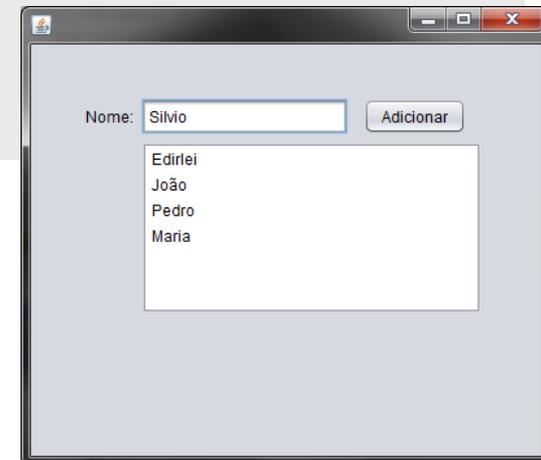
```
...
public HelloJFrame() {
    initComponents();

    listNomes.setModel(listModel);
}
...
}
```

Evento actionPerformed – Button

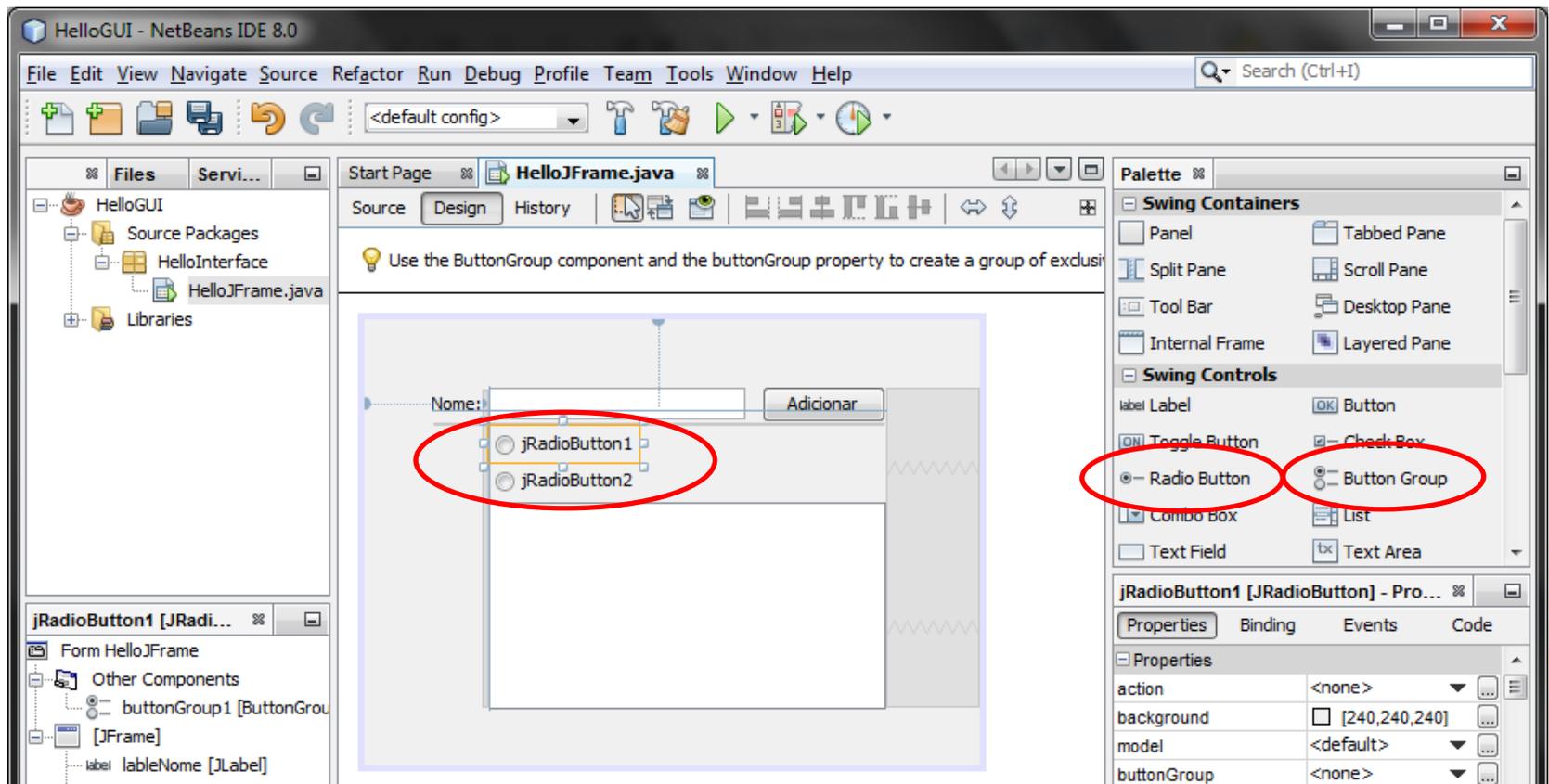
- **Exemplo 2** – Adicionar o conteúdo do TextField na List:
 - c) No evento actionPerformed, adicione o conteúdo do TextField na List:

```
...  
  
private void buttonAddActionPerformed(java.awt.event.ActionEvent evt)  
{  
    listModel.addElement(textNome.getText());  
    textNome.setText("");  
}  
  
...
```



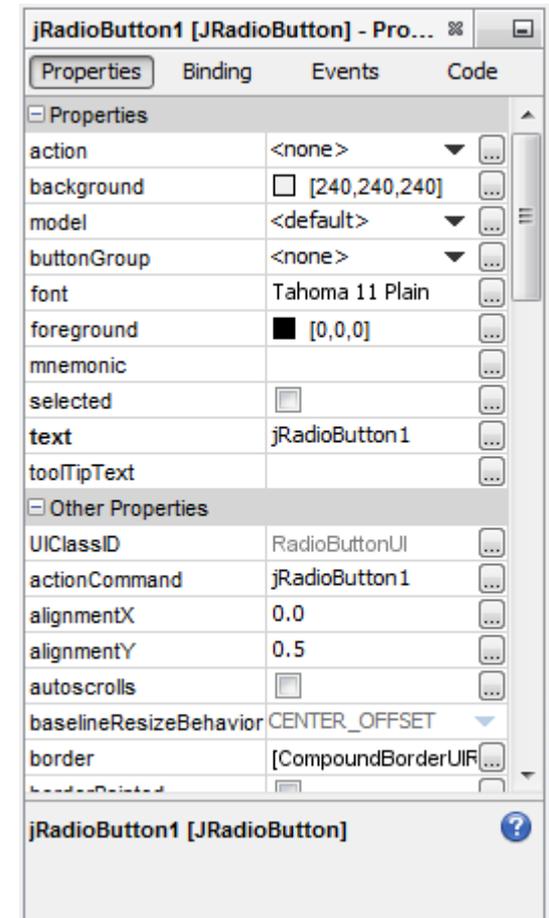
Componentes Básicos – RadioButton e ButtonGroup

- Componentes que permitem a seleção de opções.



Componentes Básicos – RadioButton

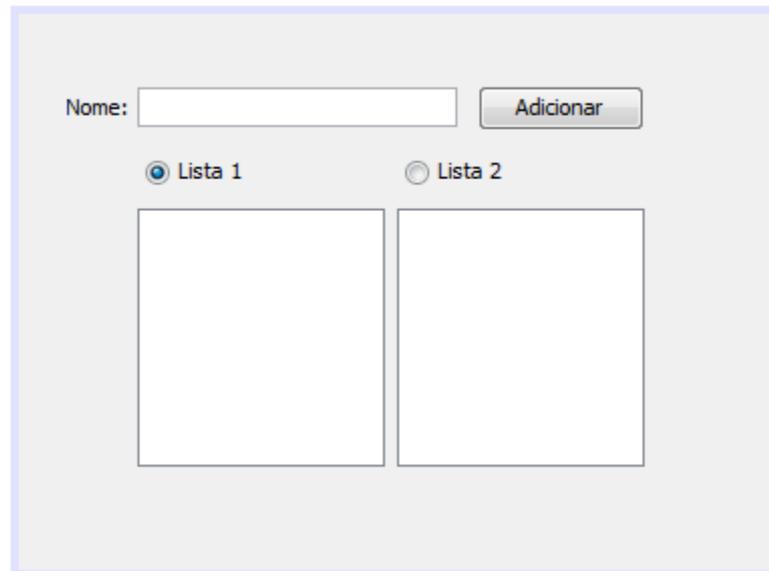
- Principais Propriedades (JRadioButton):
 - text;
 - buttonGroup;
 - Selected;
 - foreground;
 - background;
 - font;
 - icon;
 - toolTipText;
 - border;
 - enabled;



Usando o RadioButton

- **Exemplo 3** – Adicionar o conteúdo do TextField em duas List de acordo com a opção selecionada no RadioButton.

– Interface:

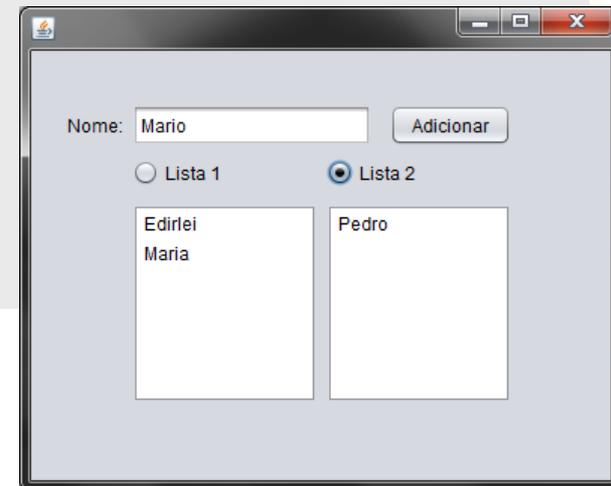


The screenshot shows a Java Swing window with a light gray background. At the top left, there is a label "Nome:" followed by a text input field. To the right of the input field is a button labeled "Adicionar". Below the input field and button, there are two radio buttons. The first radio button is selected and is labeled "Lista 1". The second radio button is unselected and is labeled "Lista 2". Below the radio buttons, there are two empty rectangular boxes, one under "Lista 1" and one under "Lista 2", representing the lists where the content from the text field will be added.

Usando o RadioButton

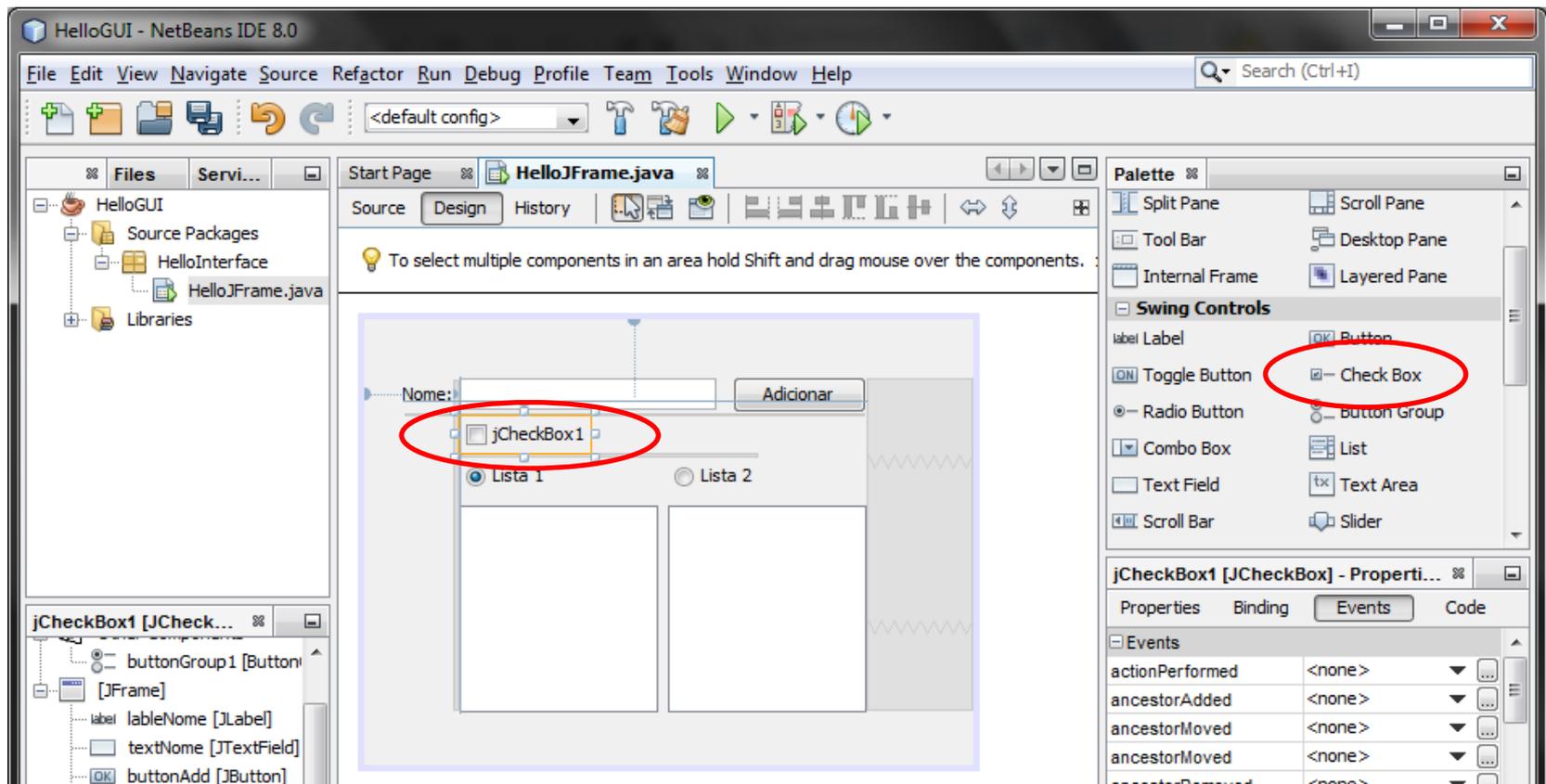
- **Exemplo 3** – Adicionar o conteúdo do TextField nas Lists de acordo com a opção selecionada no RadioButton.

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



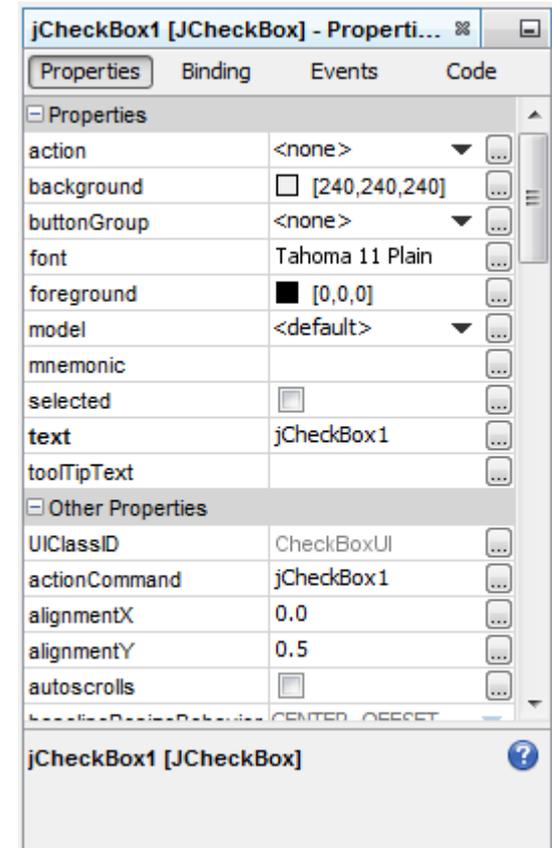
Componentes Básicos – CheckBox

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



Componentes Básicos – CheckBox

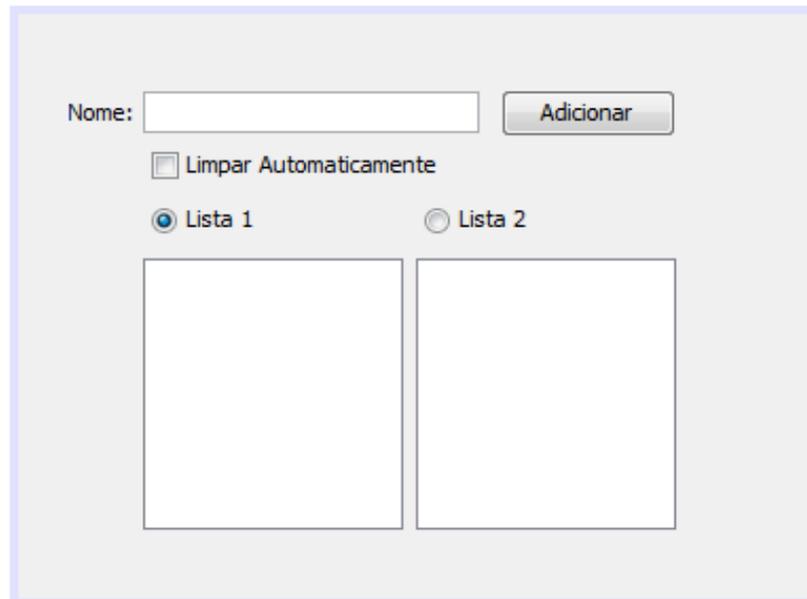
- Principais Propriedades (JCheckBox):
 - text;
 - buttonGroup;
 - Selected;
 - foreground;
 - background;
 - font;
 - icon;
 - tooltipText;
 - border;
 - enabled;



Usando o CheckBox

- **Exemplo 4** – Permitir ao usuário escolher se ele deseja limpar automaticamente o conteúdo do TextField após adicioná-lo à List.

– Interface:

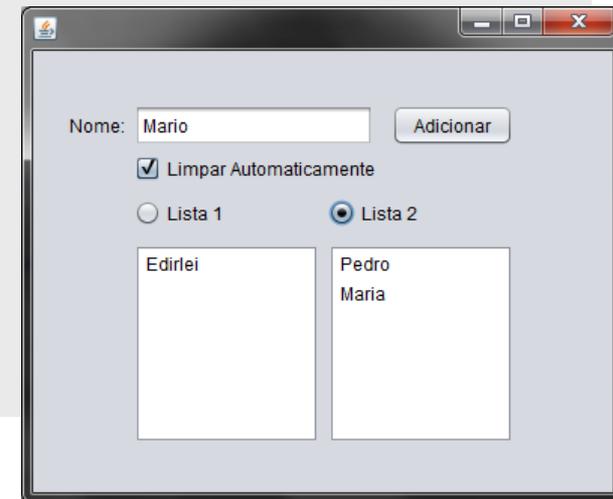


The image shows a user interface for adding items to a list. It features a text input field labeled "Nome:" with a button labeled "Adicionar" to its right. Below the input field is a checkbox labeled "Limpar Automaticamente". Underneath the checkbox are two radio buttons: "Lista 1" (which is selected) and "Lista 2". At the bottom of the interface are two empty rectangular boxes, one under each radio button, representing the lists.

Usando o CheckBox

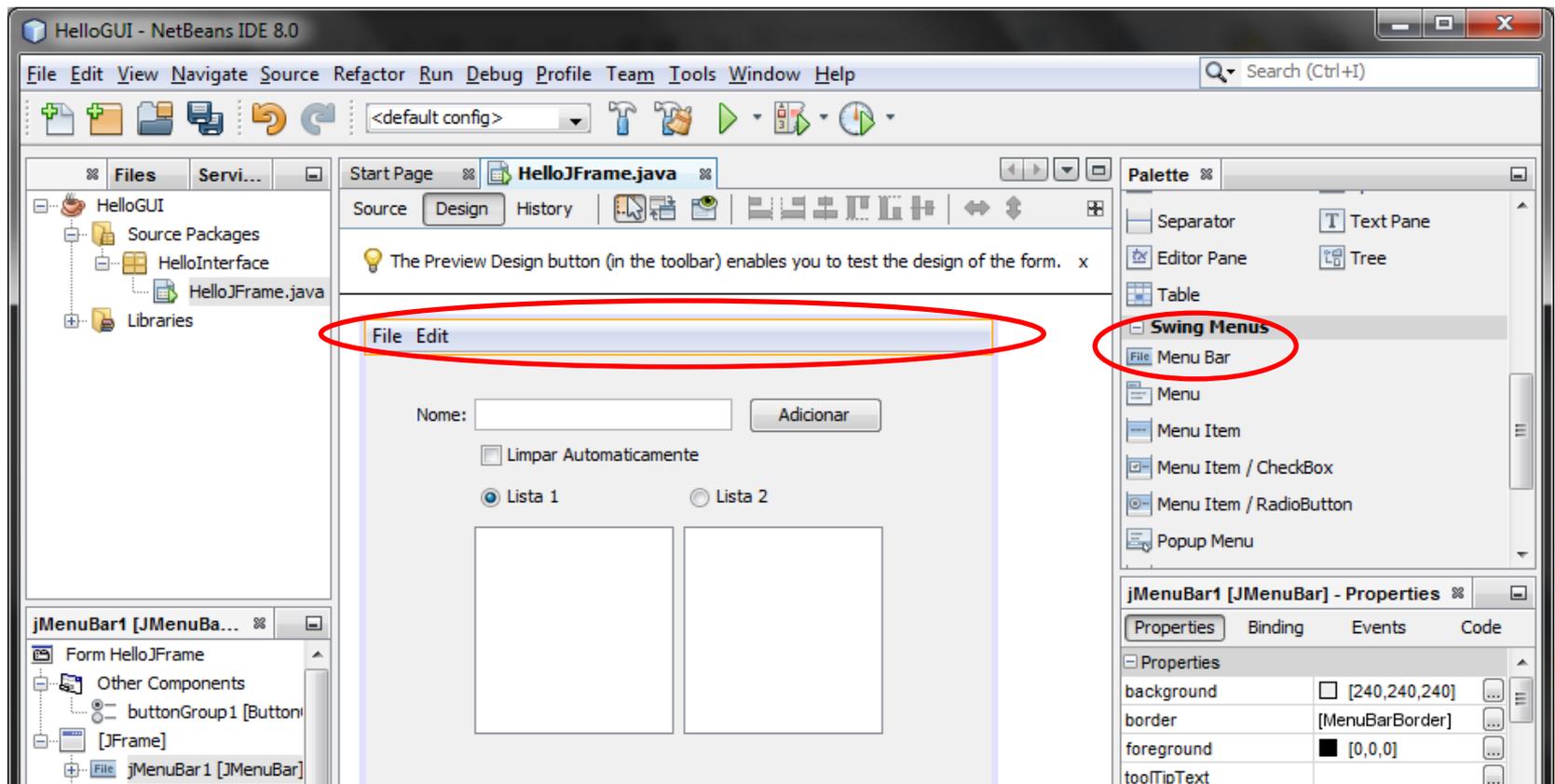
- **Exemplo 4** – Permitir ao usuário escolher se ele deseja limpar automaticamente o conteúdo do TextField após adicioná-lo à List.

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



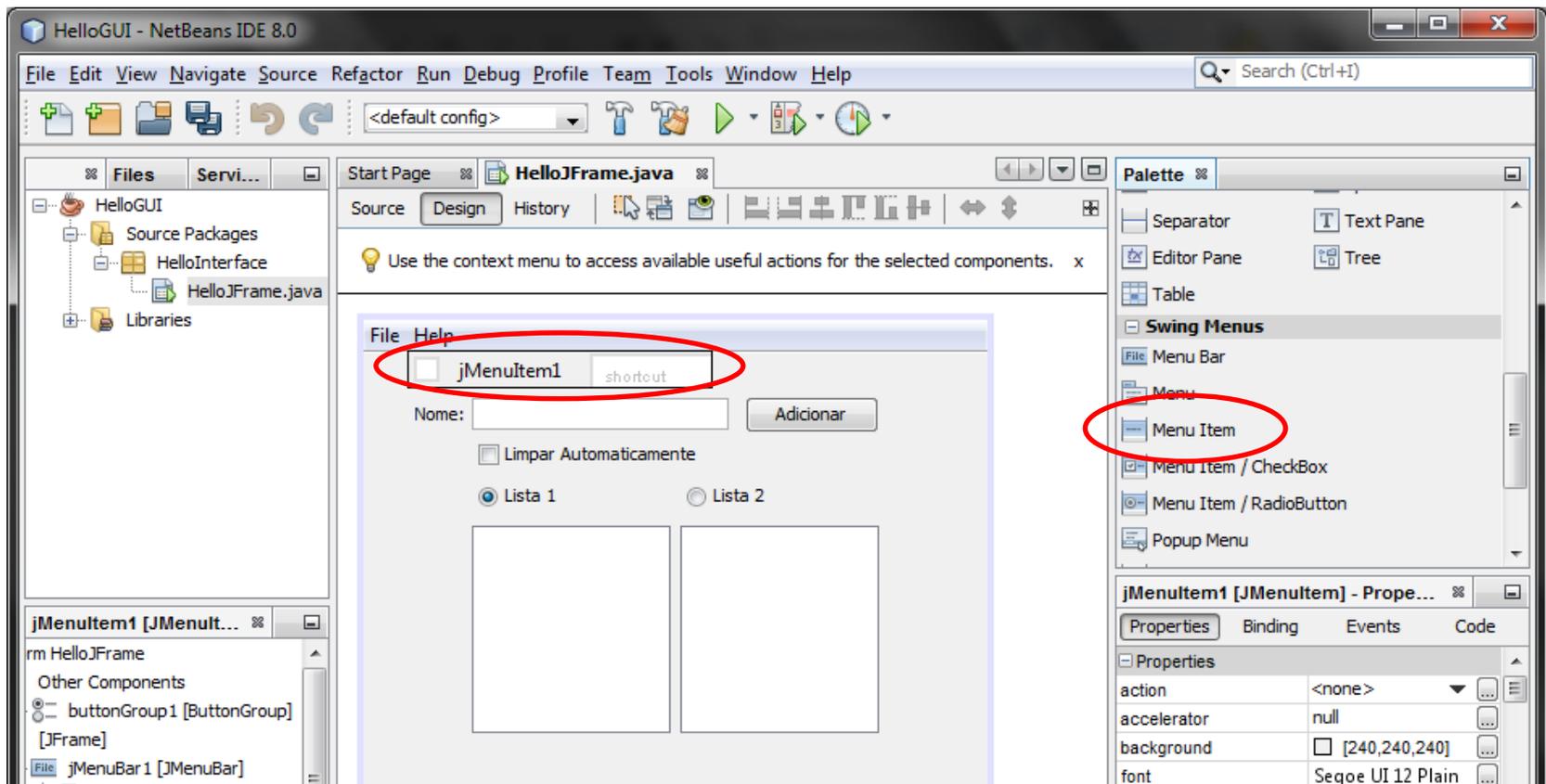
Componentes Básicos – MenuBar

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



Componentes Básicos – MenuItem

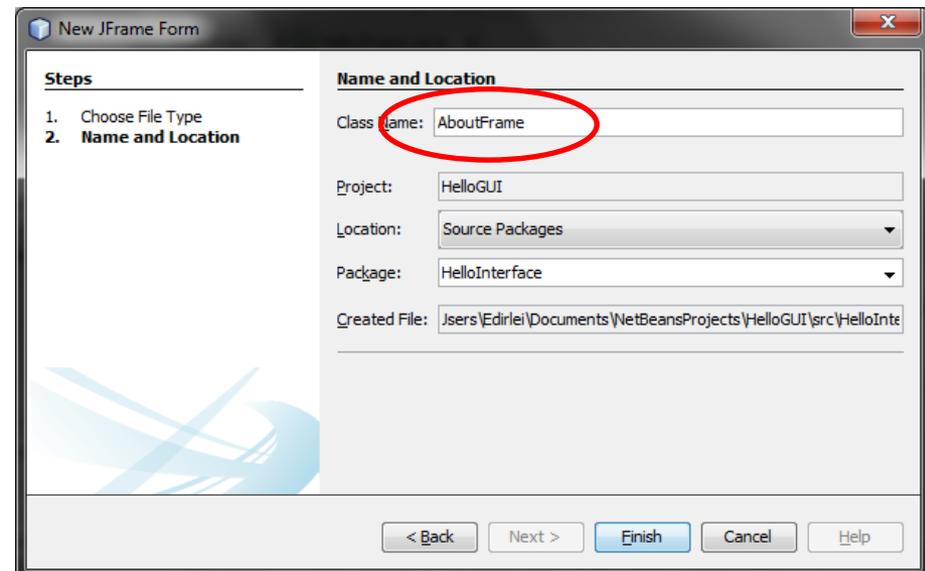
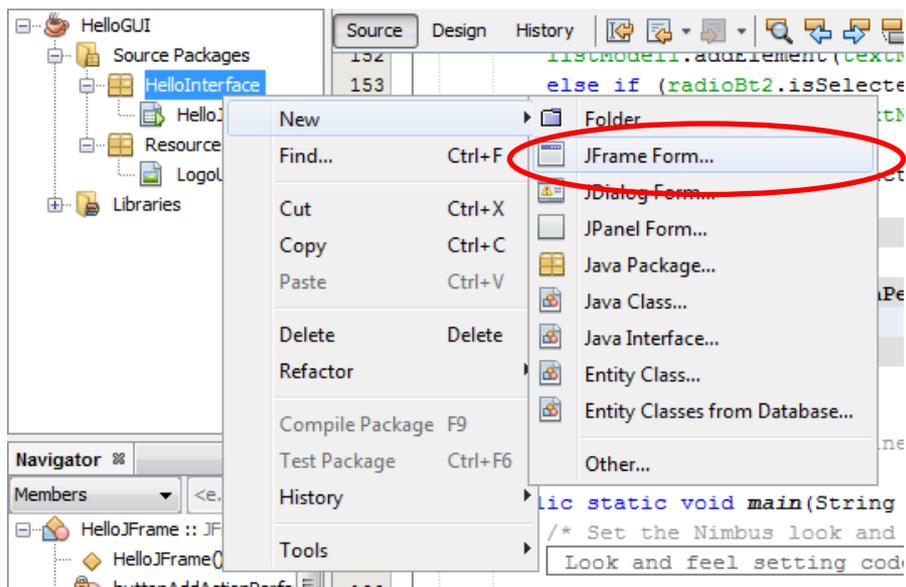
- Componente que permite a criação de itens para a barra de menu.



Abrindo uma Nova Janela

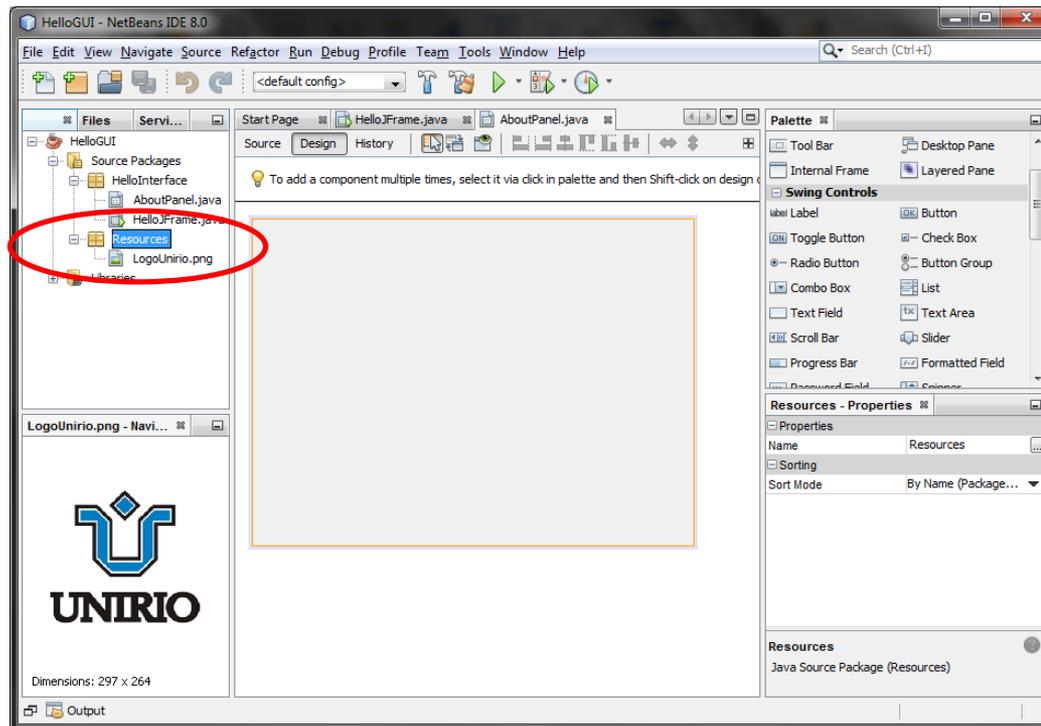
- **Exemplo 5** – Criando e abrindo uma nova janela.

a) Crie um novo JFrame;



Abrindo uma Nova Janela

- **Exemplo 5** – Criando e abrindo uma nova janela.
 - a) Crie um novo projeto e abra o HelloGUI; **crie um novo Package** e **arraste o arquivo de uma imagem** para o projeto;
 - b) Crie um novo Package e arraste o arquivo de uma imagem para o projeto;



Abrindo uma Nova Janela

- **Exemplo 5** – Criando e abrindo uma nova janela.
 - c) Adicione um novo label ao formulário, colocando a imagem previamente selecionada como ícone. Adicione também um botão para fechar a janela.



Abrindo uma Nova Janela

- **Exemplo 5** – Criando e abrindo uma nova janela.
 - d) Na janela principal, crie um novo evento para criar e abrir a nova janela quando o usuário clicar no item “About” do menu “Help”:

```
private void menuAboutActionPerformed(java.awt.event.ActionEvent evt)
{

    AboutJFrame aboutJFrame = new AboutJFrame();

    aboutJFrame.setVisible(true);

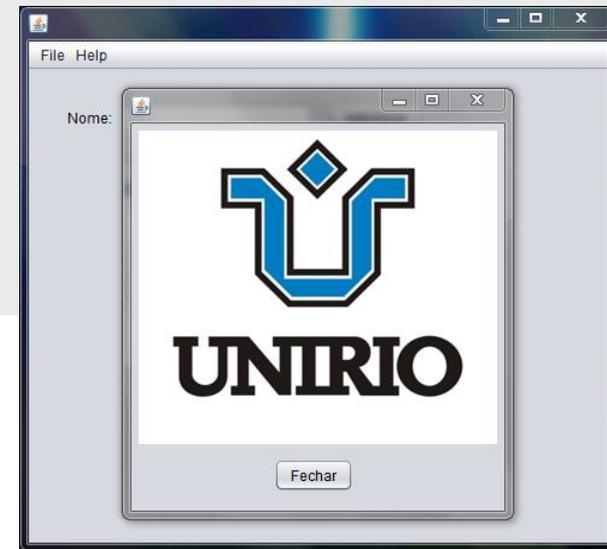
}
```

Abrindo uma Nova Janela

- **Exemplo 5** – Criando e abrindo uma nova janela.
 - e) Crie um evento “actionPerformed” para o botão fechar, ocultando e destruindo a janela quando o usuário clicar no botão:

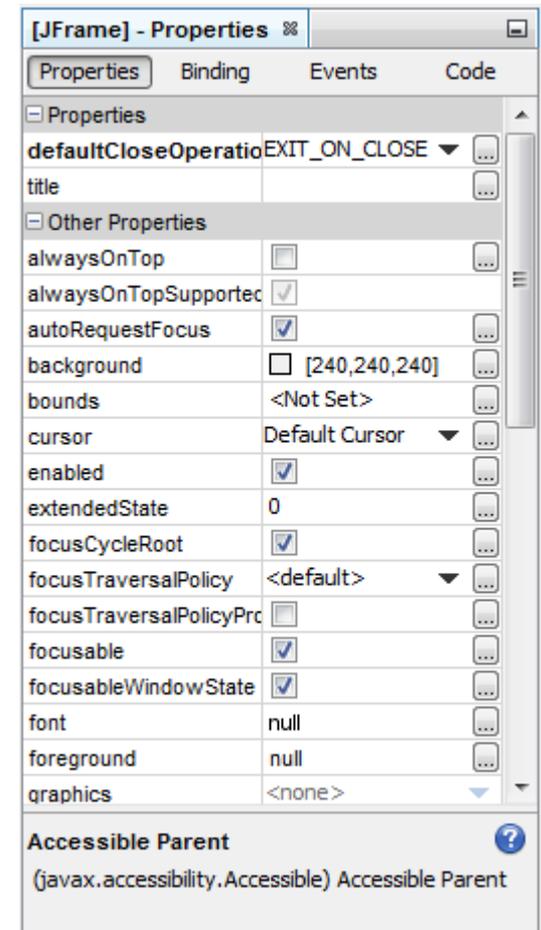
```
private void buttonFecharActionPerformed(java.awt.event.ActionEvent evt)
{
    this.setVisible(false);

    this.dispose();
}
```



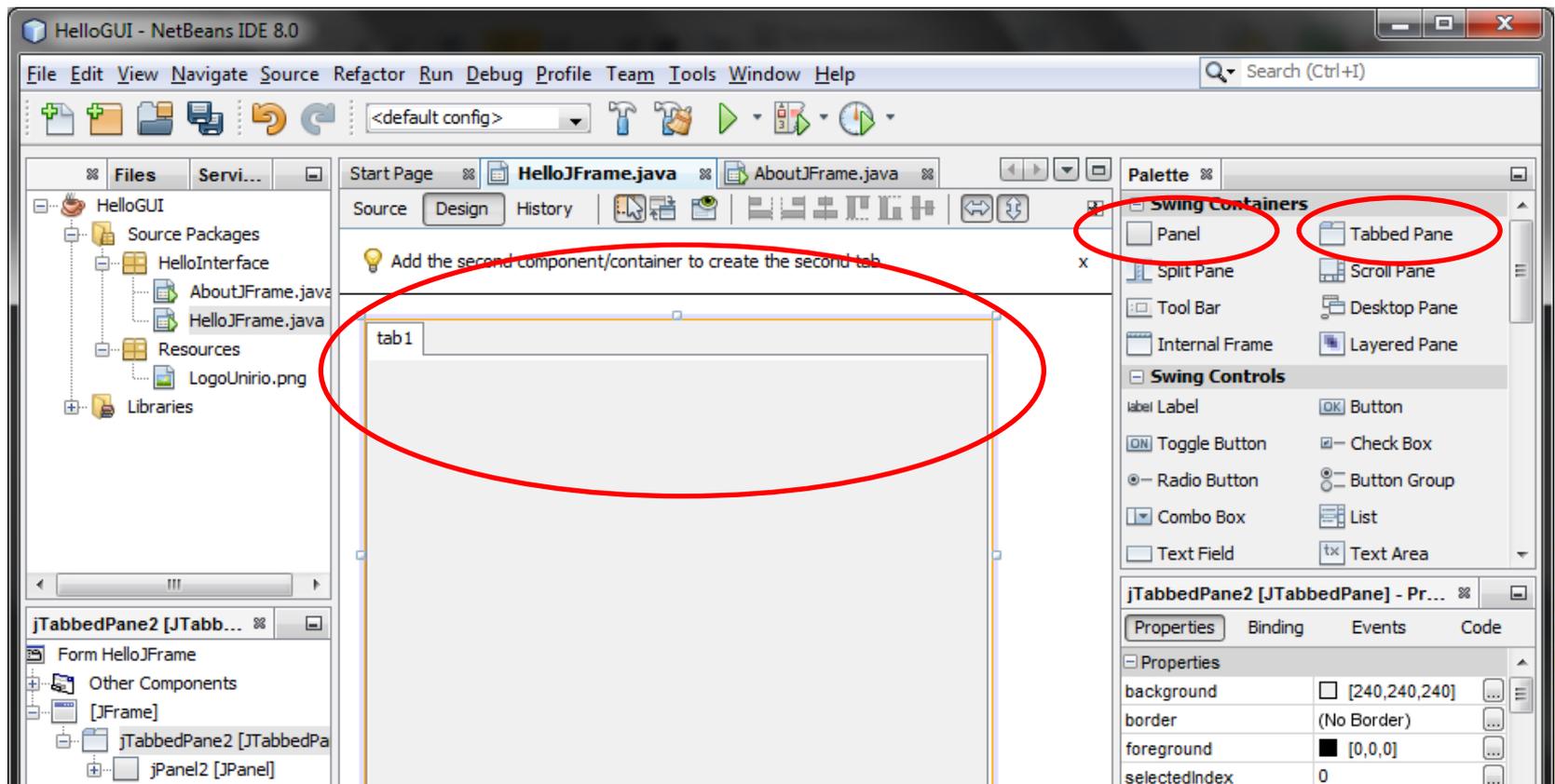
Componentes Básicos – Frame

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

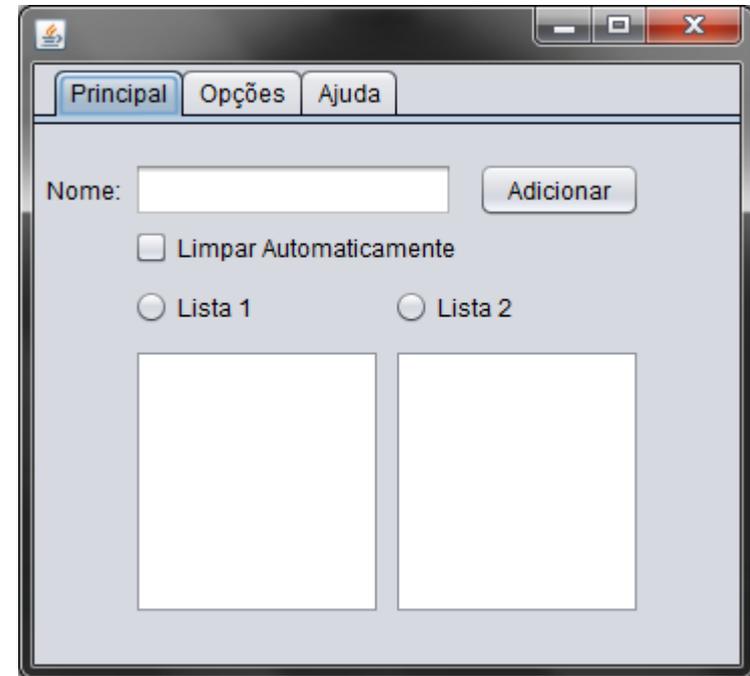
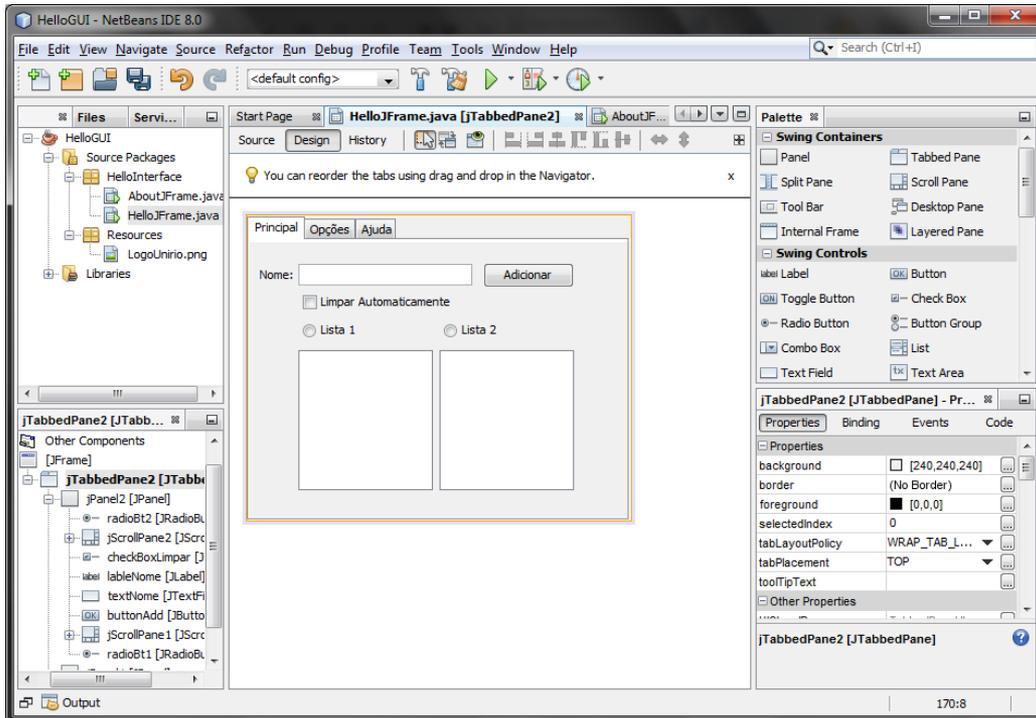


Componentes Básicos – Tabbed Pane

- Componente que permite a criação de telas com abas.



Componentes Básicos – Tabbed Pane



Exercícios

Lista de Exercícios 03 – GUI e Swing

<http://uniriodb2.uniriotec.br>

