

APLICAÇÃO DA LINGUAGEM DE PROGRAMAÇÃO ORIENTADA À OBJETOS - ALPOO

Aula - Componentes (Java Swing)

Prof. Danilo Pereira - danilo.pereira1@docente.unip.br



TÓPICOS DA AULA

- Java Swing usando NetBeans IDE e Eclipse
- Revisar as etapas de um desenvolvimento usando Java Swing
- Conhecer os principais componentes
- Exercícios práticos usando os componentes

INTRODUÇÃO A INTERFACES GRÁFICAS EM JAVA

O desenvolvimento de uma interface GUI em Swing se baseia nos conceitos de **janela e **eventos****

- Uma janela é um contêiner de objetos gráficos
- Os objetos devem ser anexados ao contêiner para que sejam exibidos
- Existem diferentes classes que podem representar uma janela, no entanto, a classe JFrame fornece o padrão de janela comum da

UM APLICATIVO GUI - SWING

1ª etapa:

- Criação da janela que conterá os demais objetos gráficos da aplicação

2ª etapa:

- Inserção dos componentes da interface

3ª etapa

- Tratamento de eventos

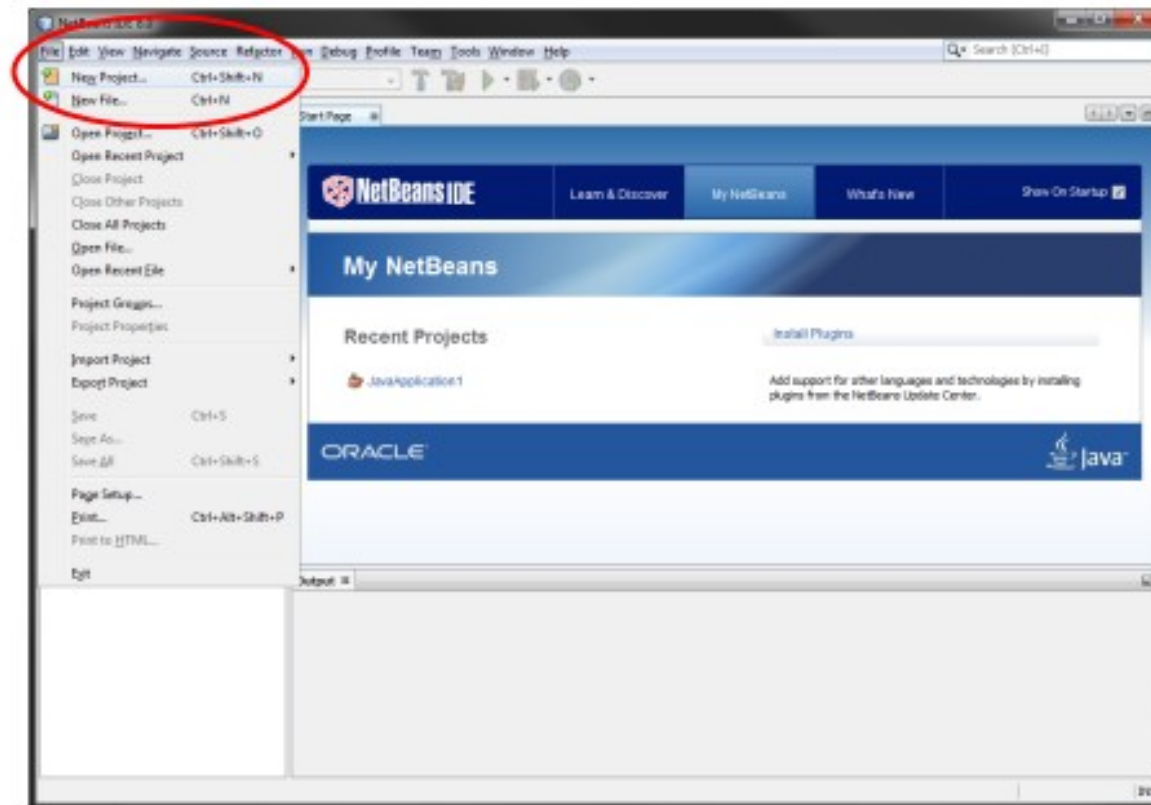
4ª etapa

- Lógica do programa

Java Swing usando NetBeans IDE

Netbeans GUI Builder – Criando Projeto

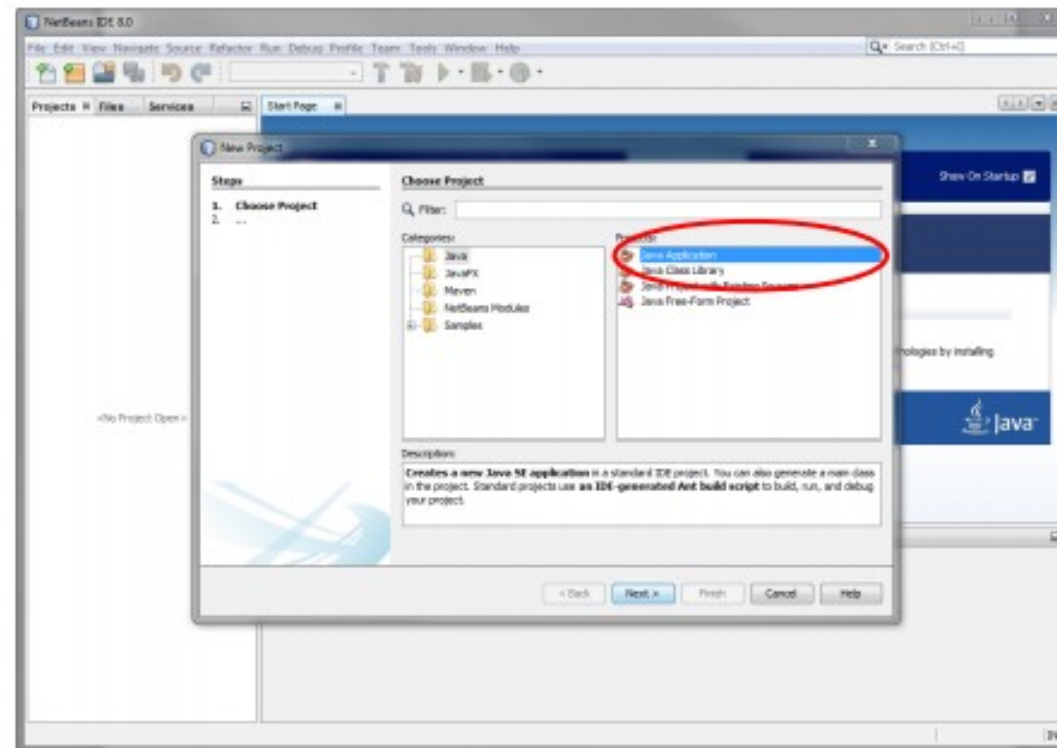
1) Acesse o menu File->New Project...



Java Swing usando **NetBeans IDE**

Netbeans GUI Builder – Criando Projeto

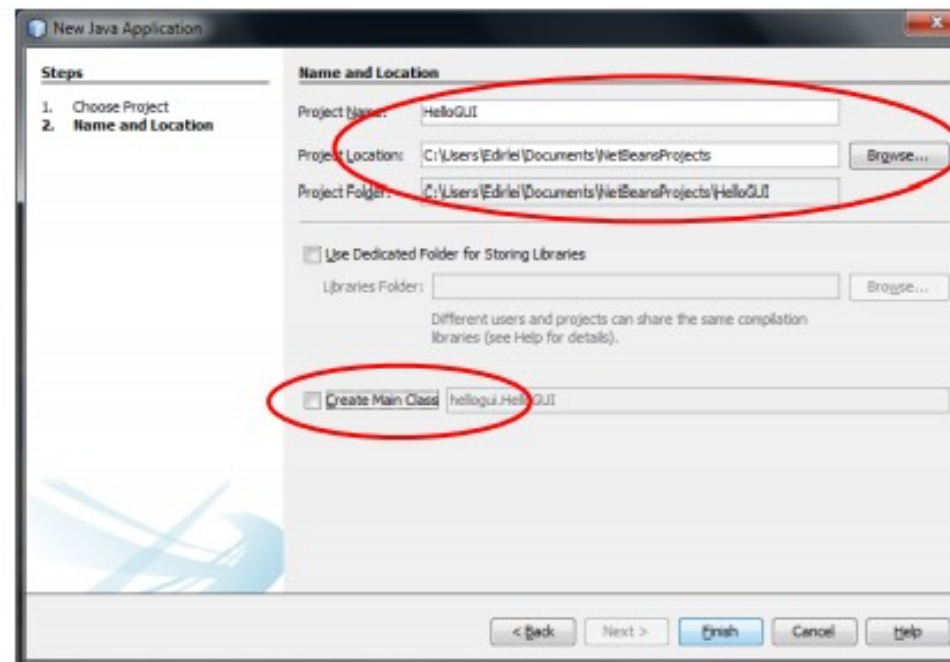
- 2) Selecione o tipo de projeto “Java Application” e em seguida clique em “Next”:



Java Swing usando **NetBeans IDE**

Netbeans GUI Builder – Criando Projeto

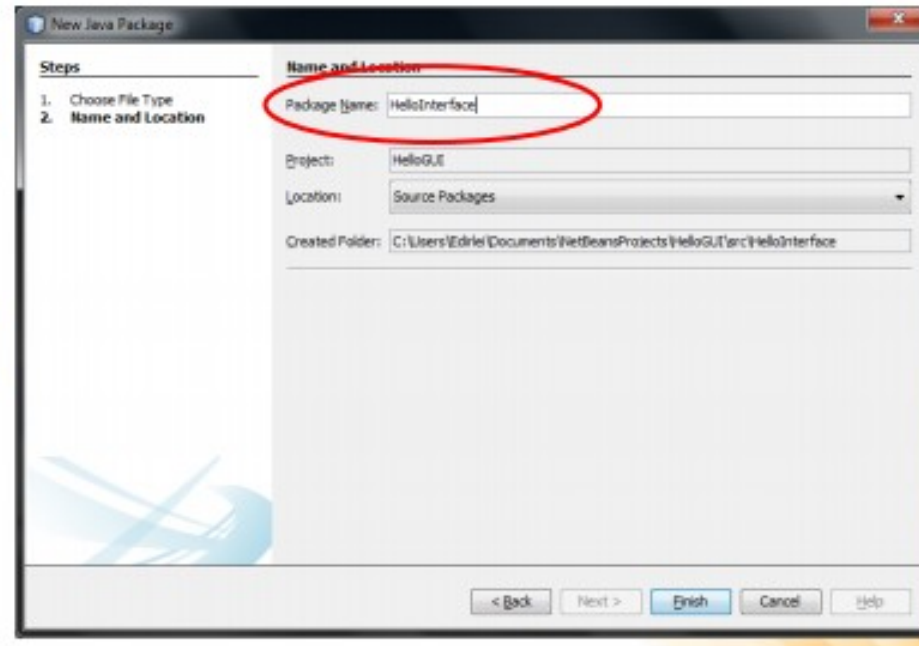
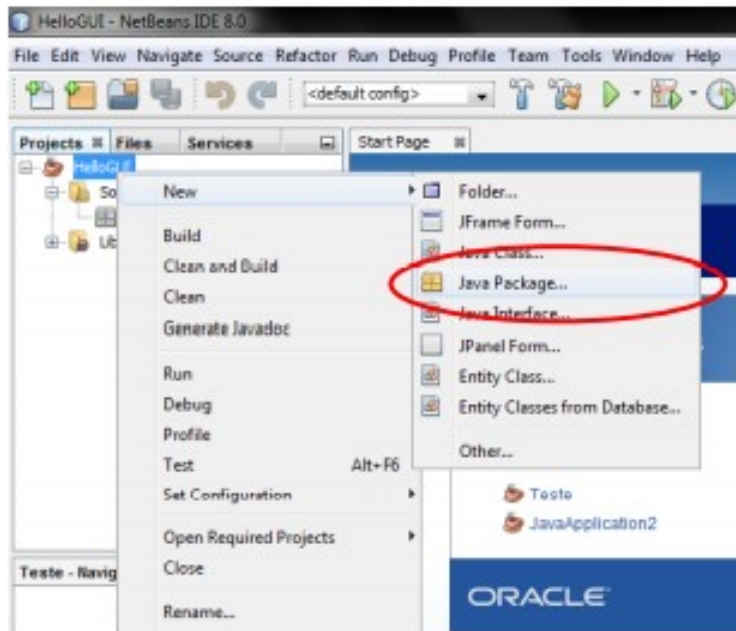
- 3) De um nome para o projeto, selecione o local onde ele será salvo e desmarque a opção “Create Main Class”. Em seguida clique em “Finish”:



Java Swing usando **NetBeans IDE**

Netbeans GUI Builder – Criando Projeto

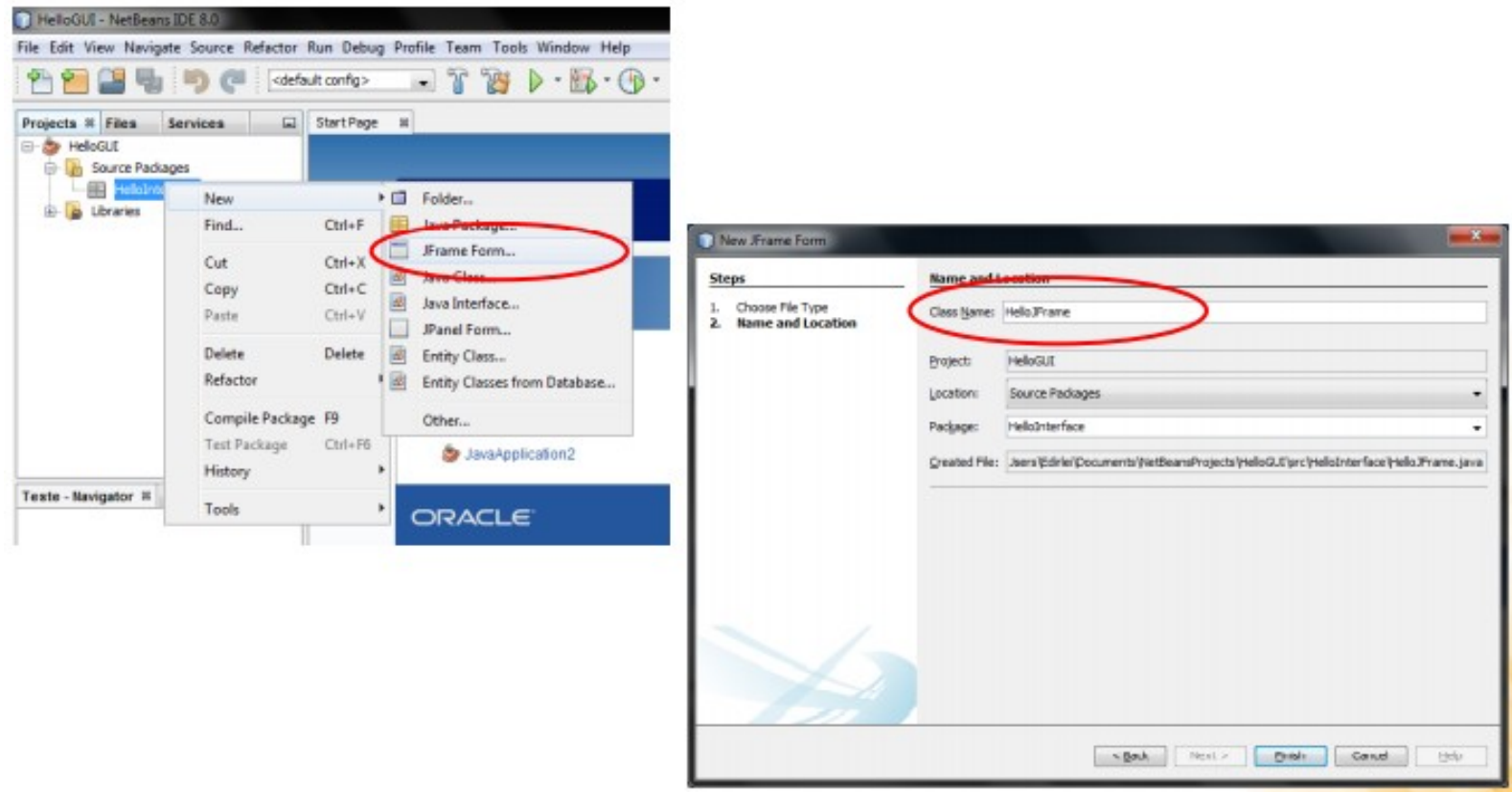
4) Crie um novo “Java Package” no projeto:



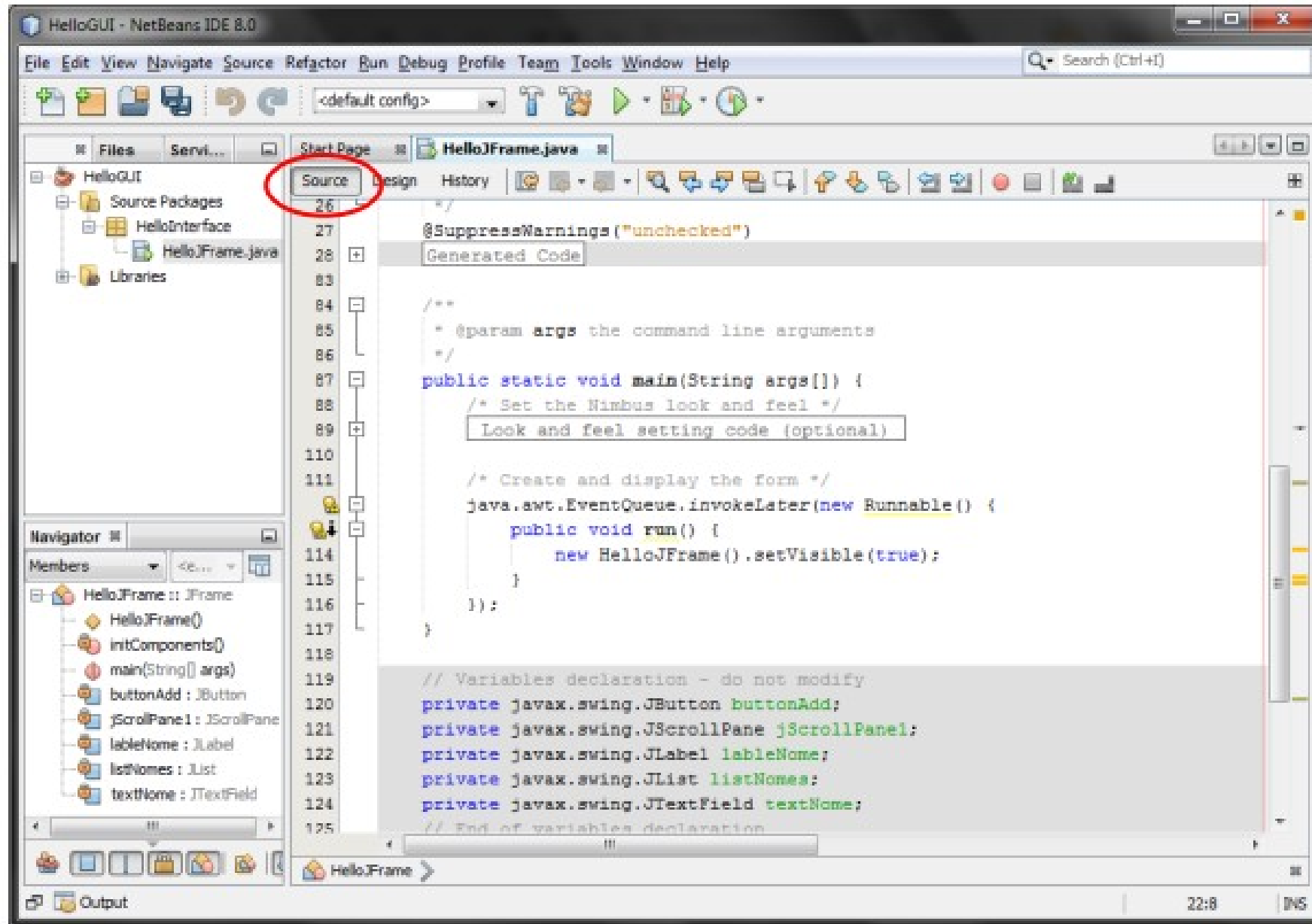
Java Swing usando **NetBeans IDE**

Netbeans GUI Builder – Criando Projeto

5) Crie um novo “JFrame Form”:

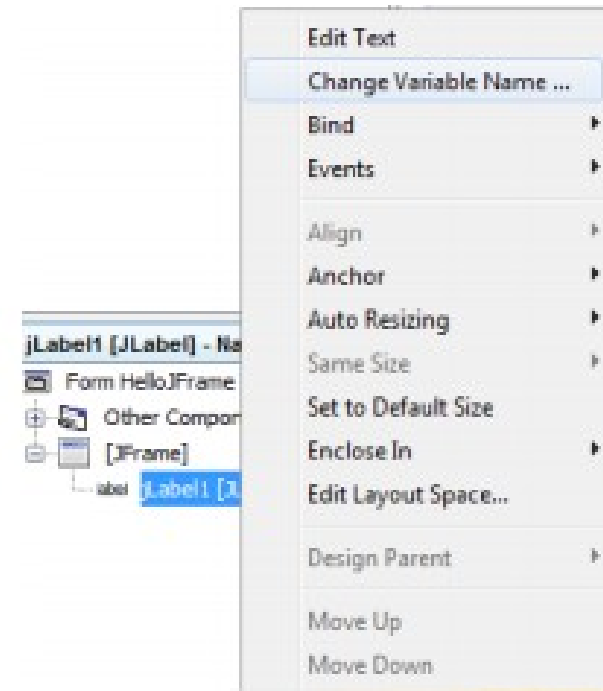
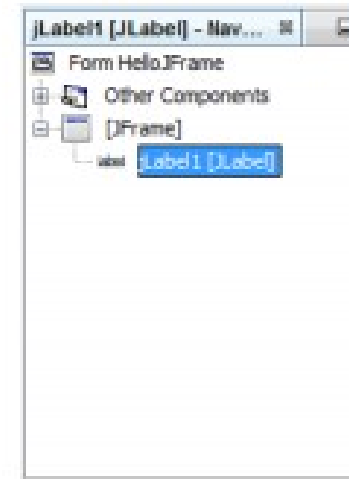


Java Swing usando NetBeans IDE - CÓDIGO GERADO



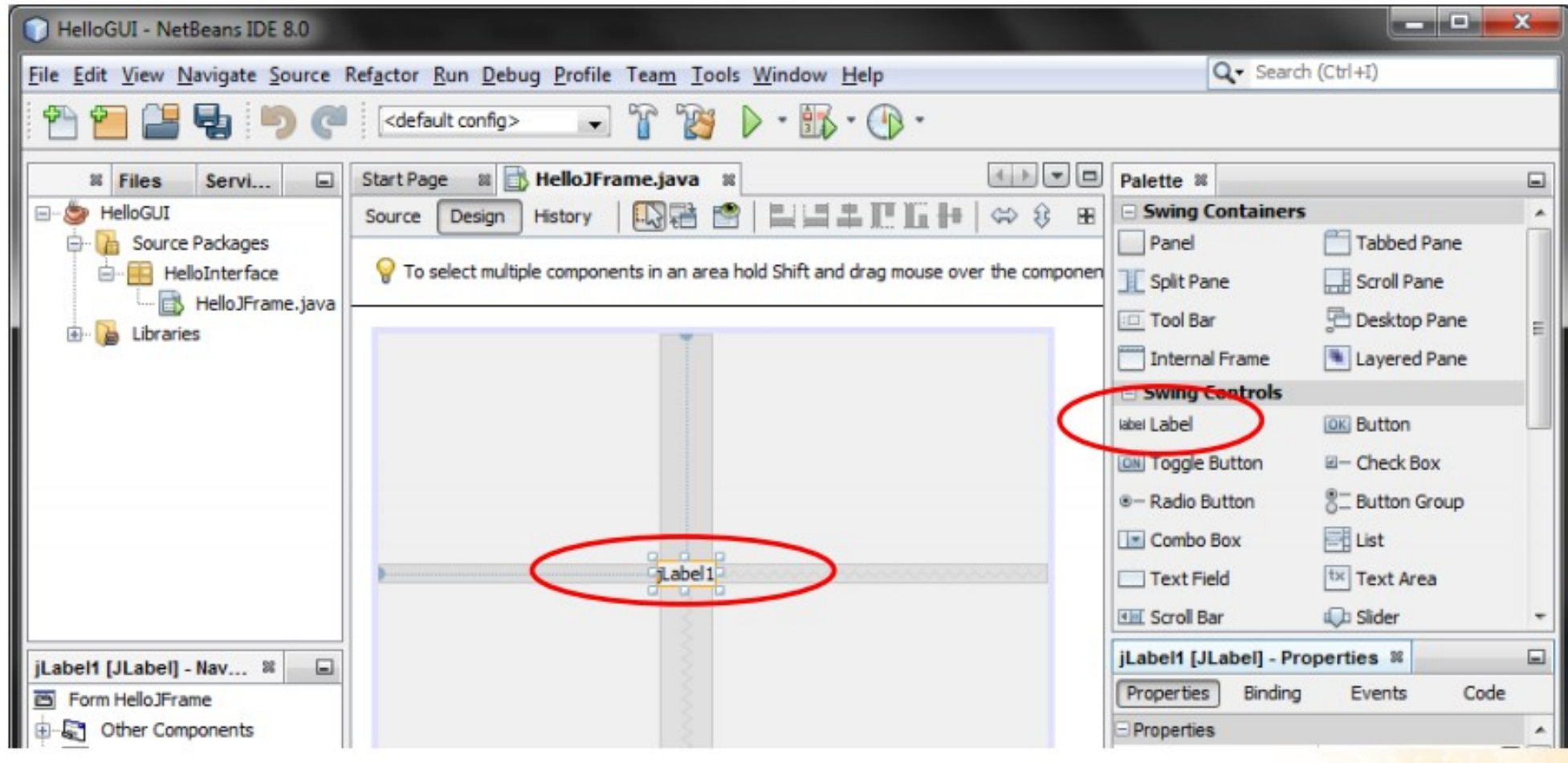
Componentes básicos - LABEL

- 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.



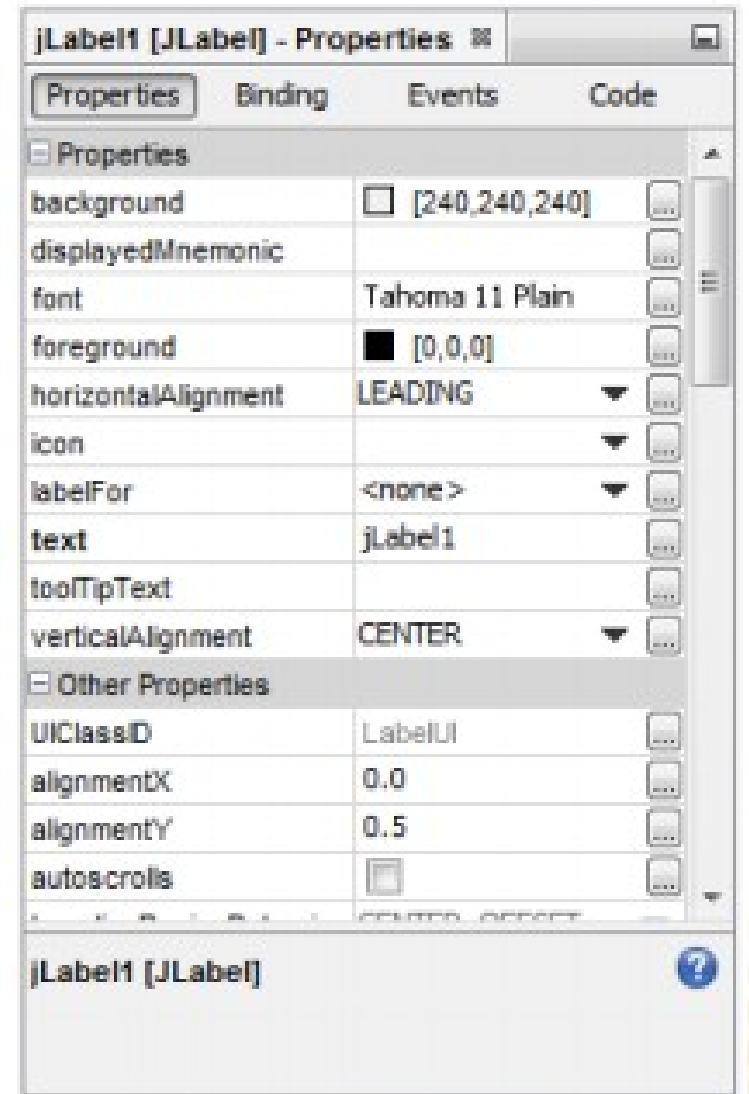
Componentes básicos - LABEL

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



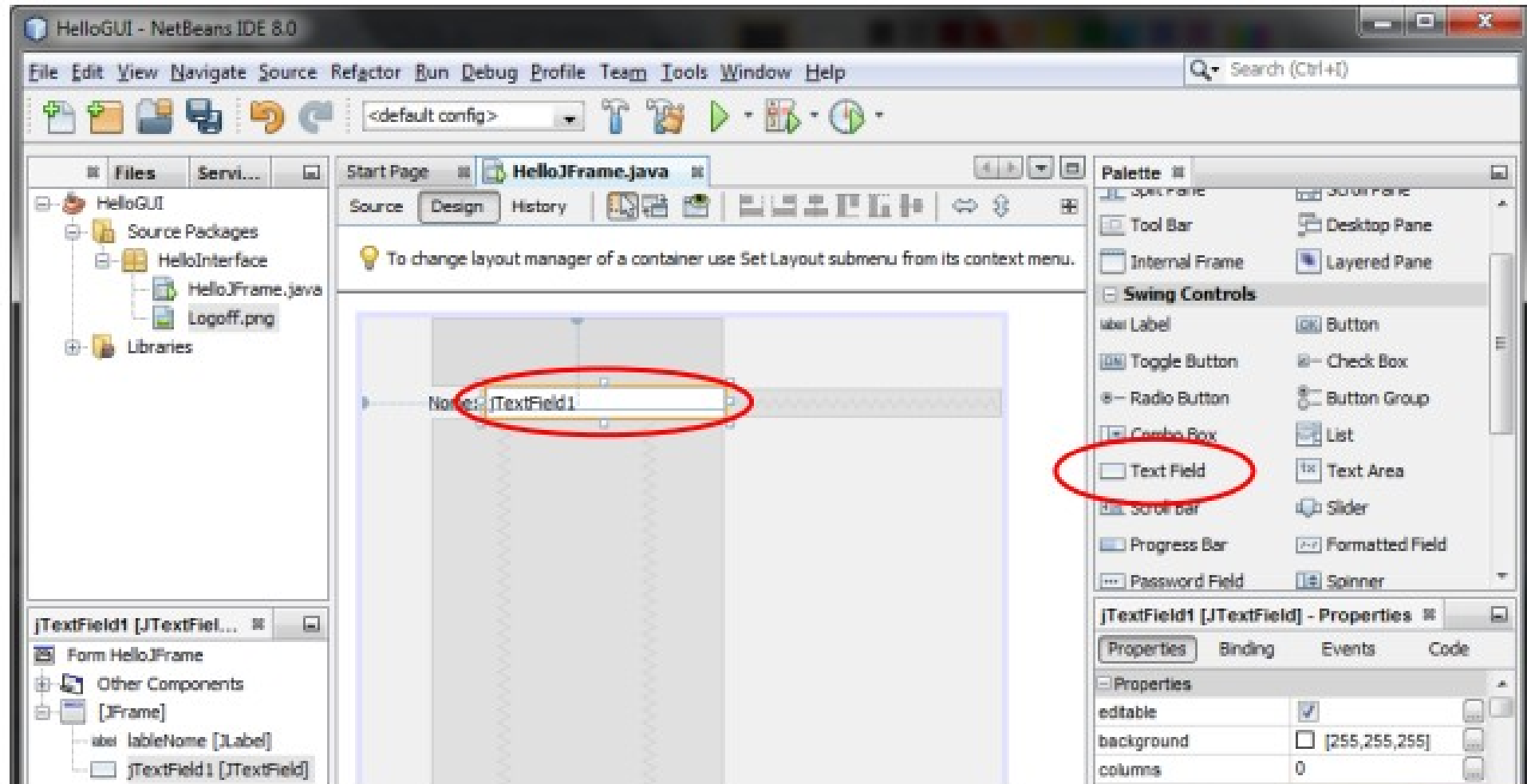
Componentes básicos - LABEL

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



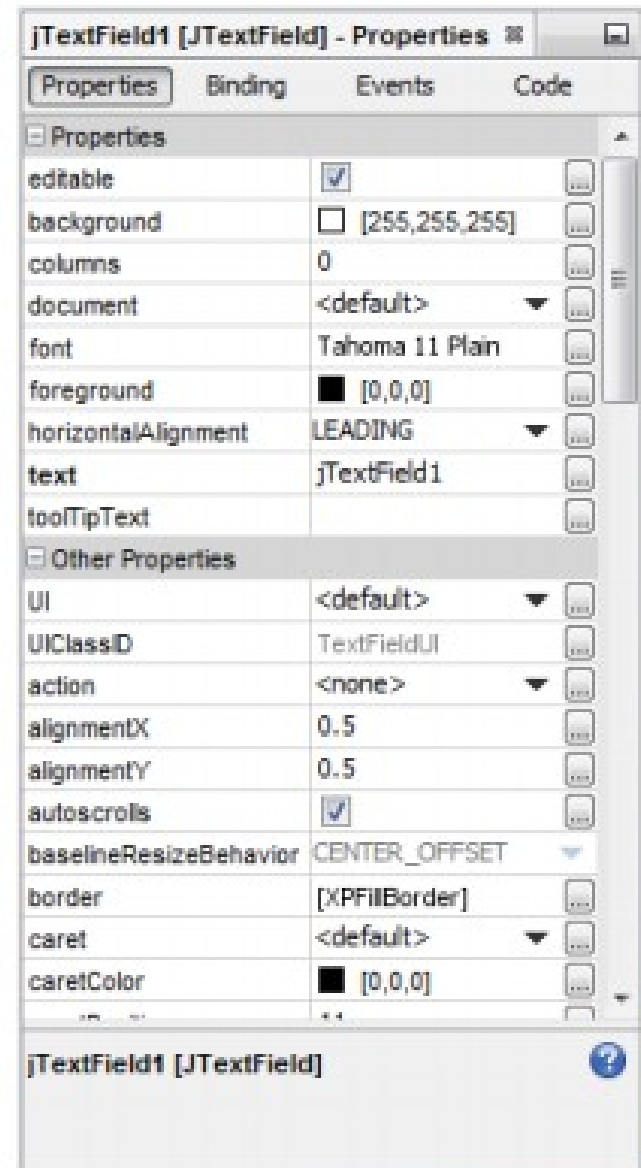
Componentes básicos - TEXTFIELD

- Componente para entrada, edição e exibição de texto.



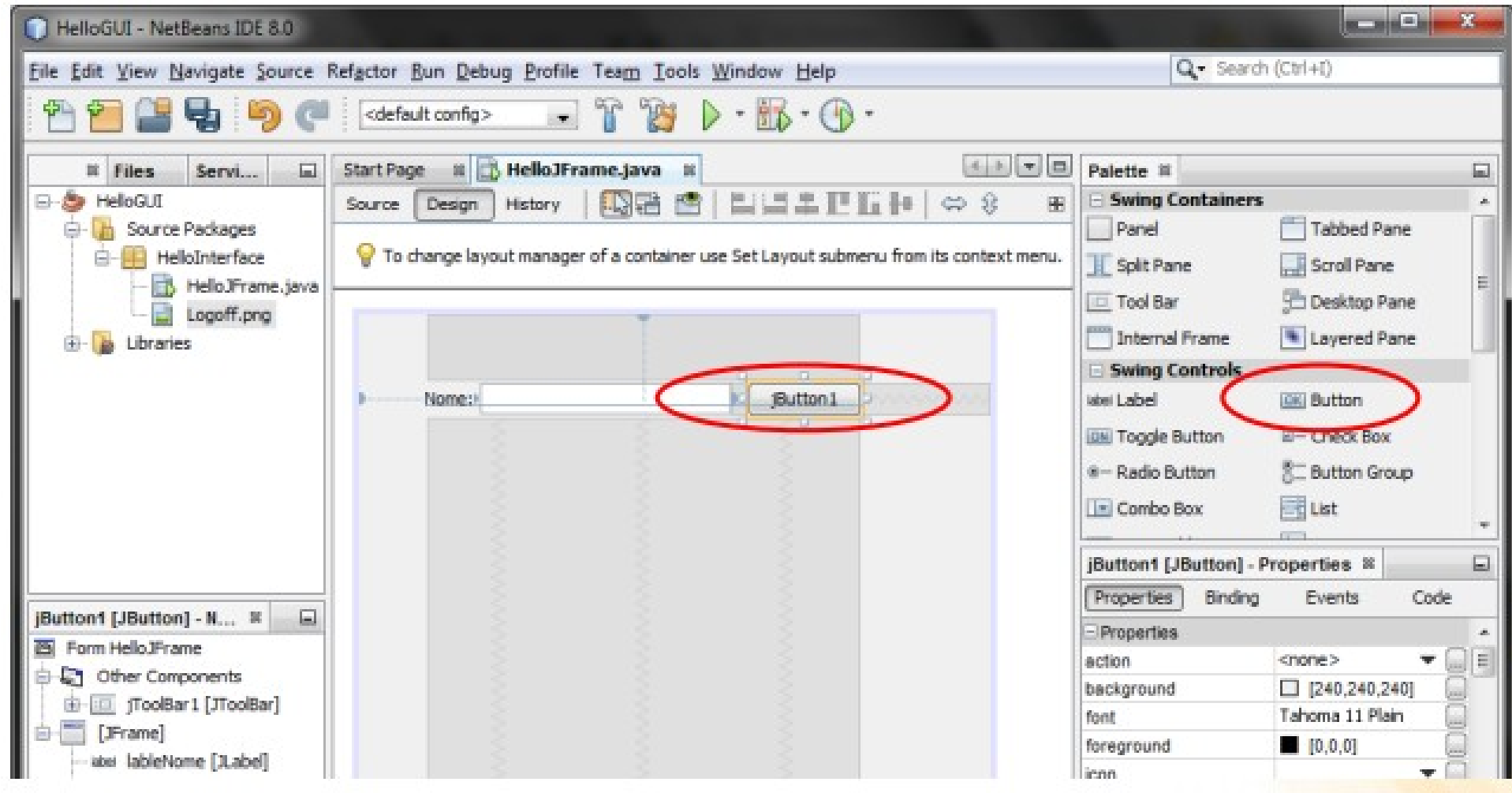
Componentes básicos - TEXTFIELD

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



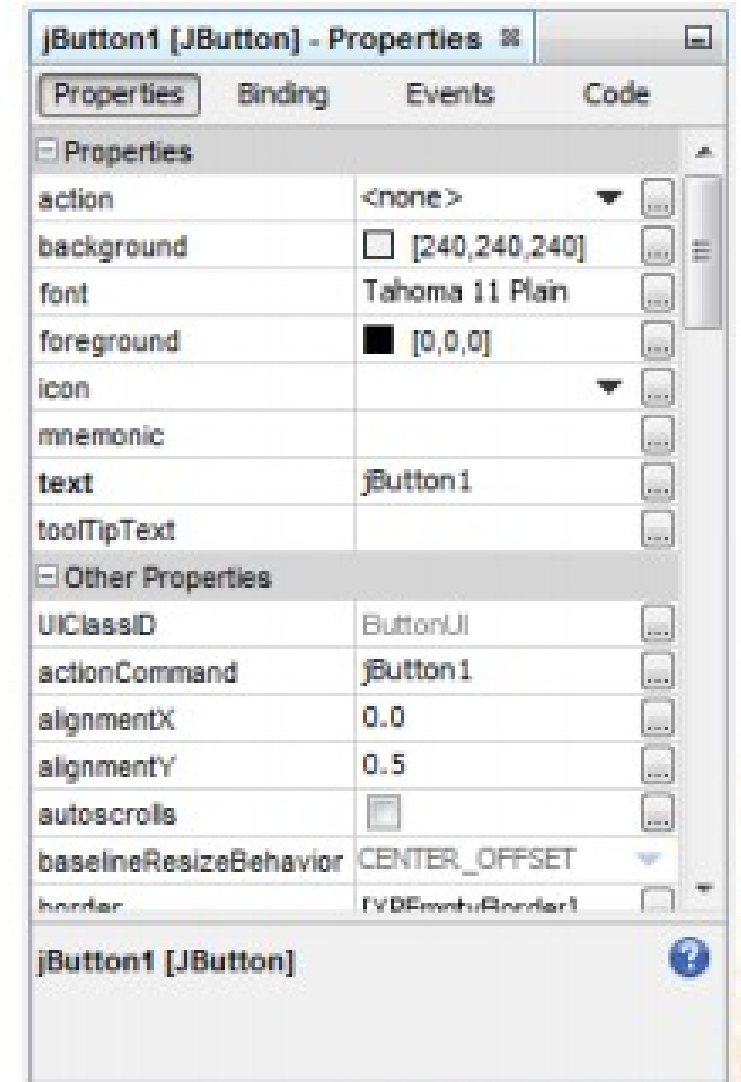
Componentes básicos - **BUTTON**

- Componente que representa um botão.



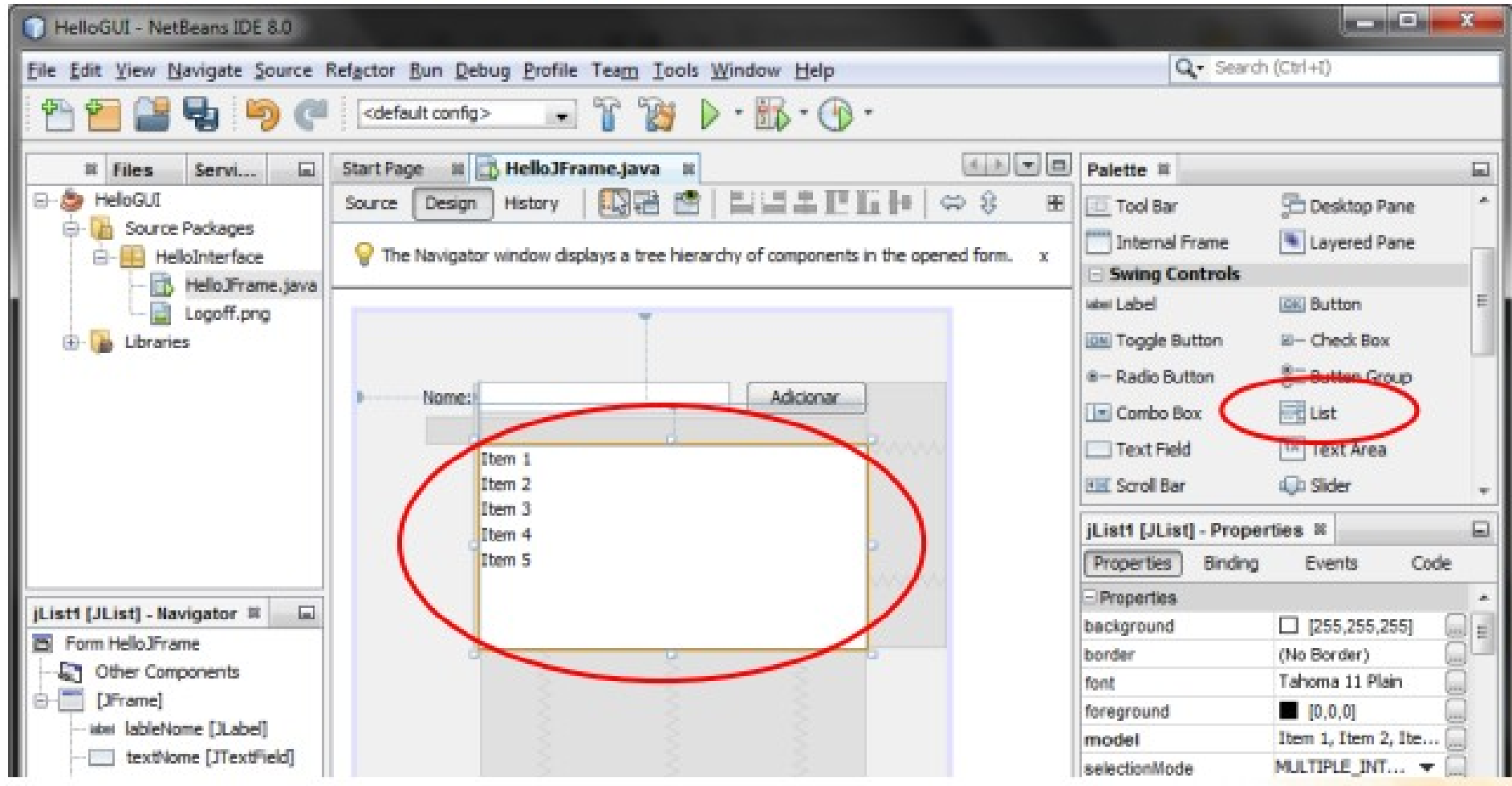
Componentes básicos - **BUTTON**

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



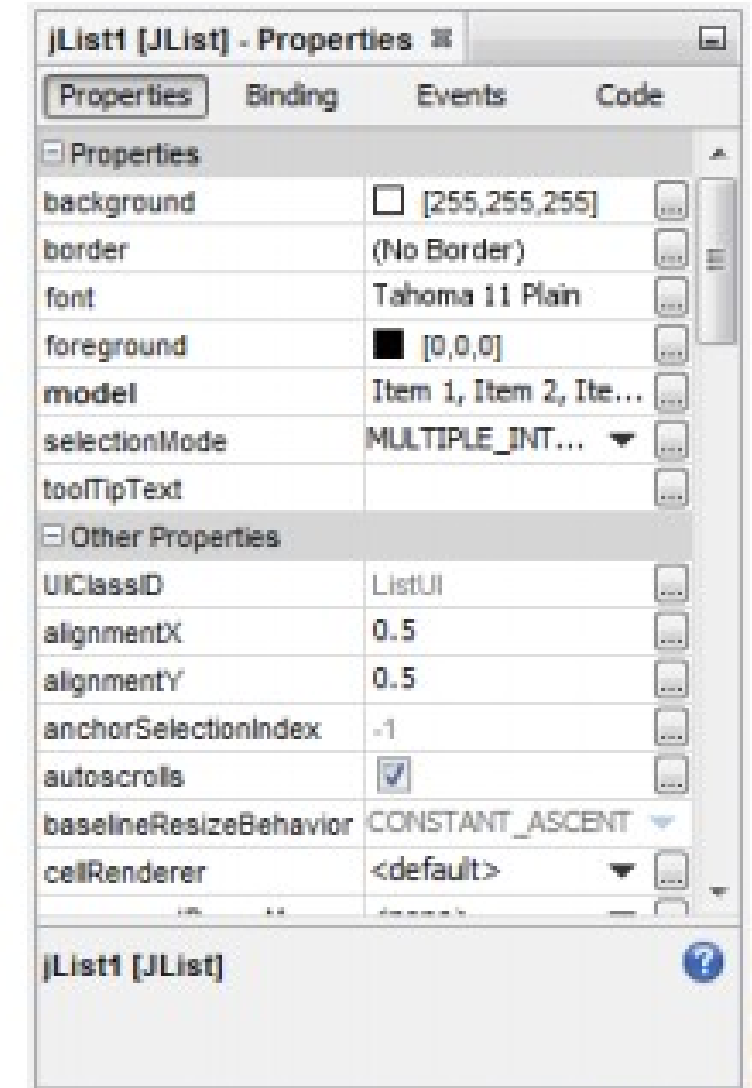
Componentes básicos - LIST

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



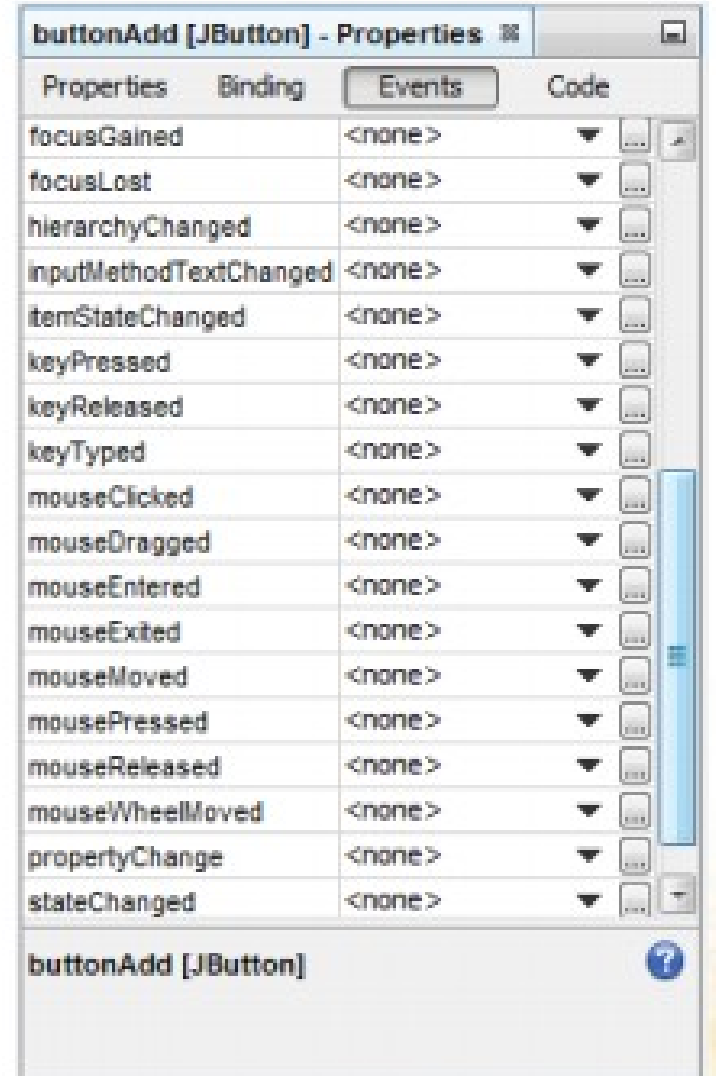
Componentes básicos - LIST

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



Tratamento de Eventos - Button

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

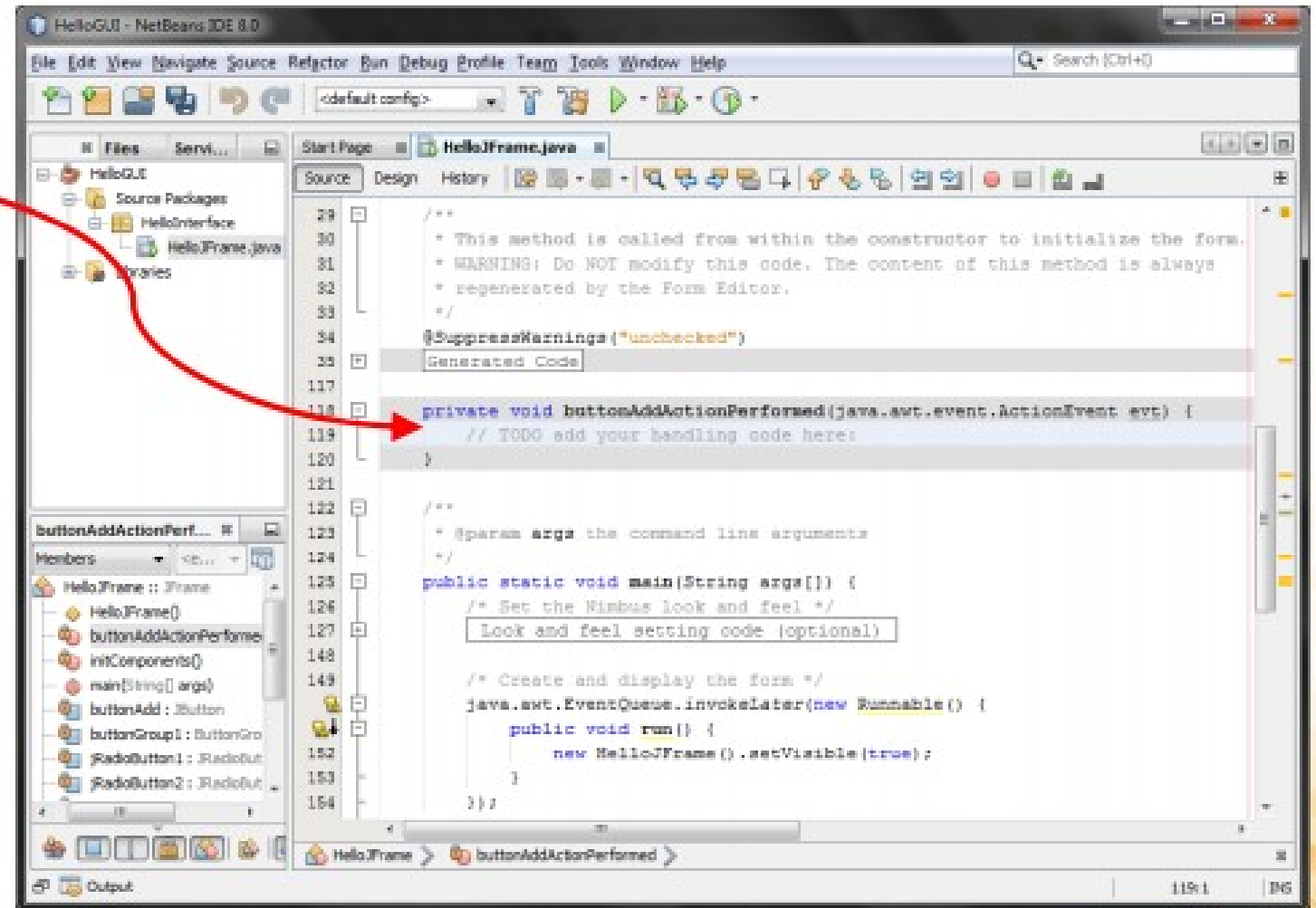
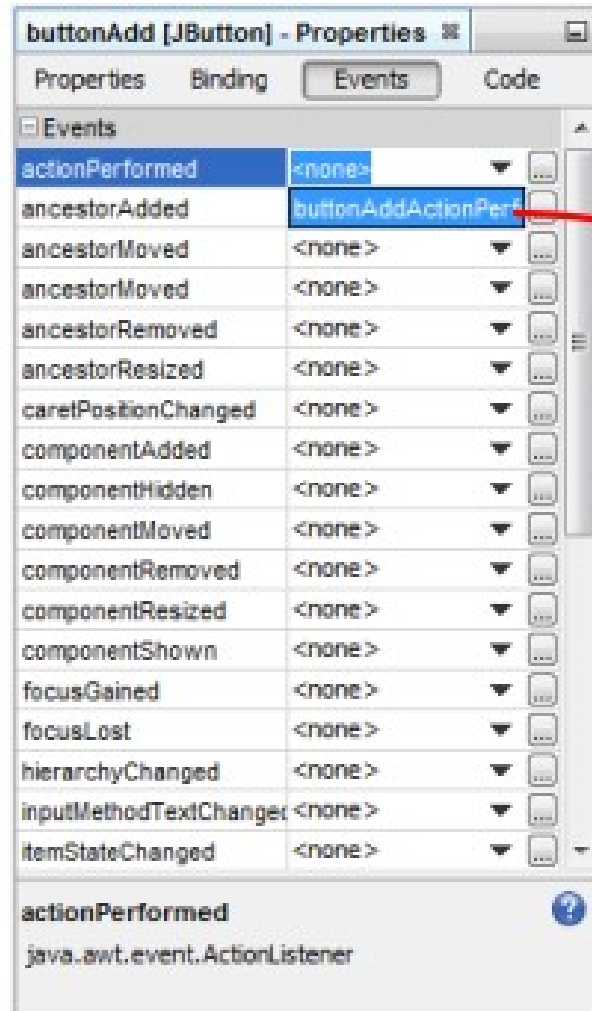


Properties	Binding	Events	Code
focusGained		<none>	...
focusLost		<none>	...
hierarchyChanged		<none>	...
inputMethodTextChanged		<none>	...
itemStateChanged		<none>	...
keyPressed		<none>	...
keyReleased		<none>	...
keyTyped		<none>	...
mouseClicked		<none>	...
mouseDragged		<none>	...
mouseEntered		<none>	...
mouseExited		<none>	...
mouseMoved		<none>	...
mousePressed		<none>	...
mouseReleased		<none>	...
mouseWheelMoved		<none>	...
propertyChange		<none>	...
stateChanged		<none>	...

buttonAdd [JButton]

Evento actionPerformed - **BUTTON**

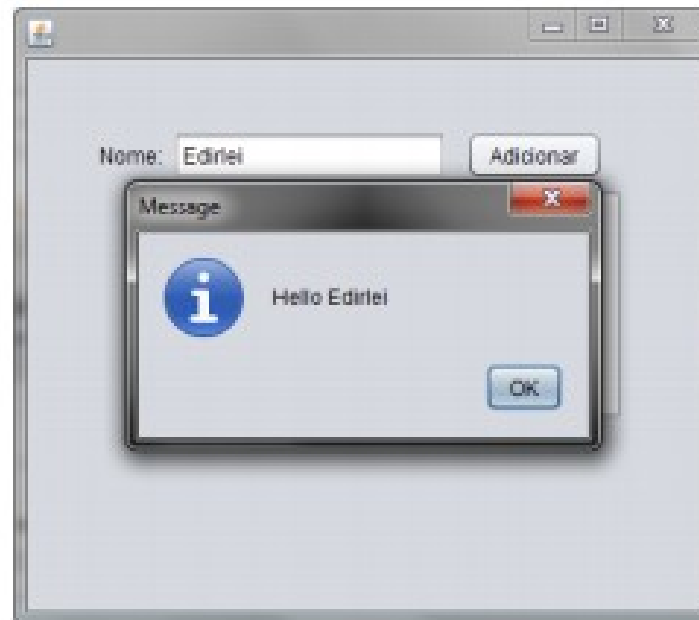
- Criando um evento de ativação para o botão (actionPerformed):



Evento actionPerformed - **BUTTON**

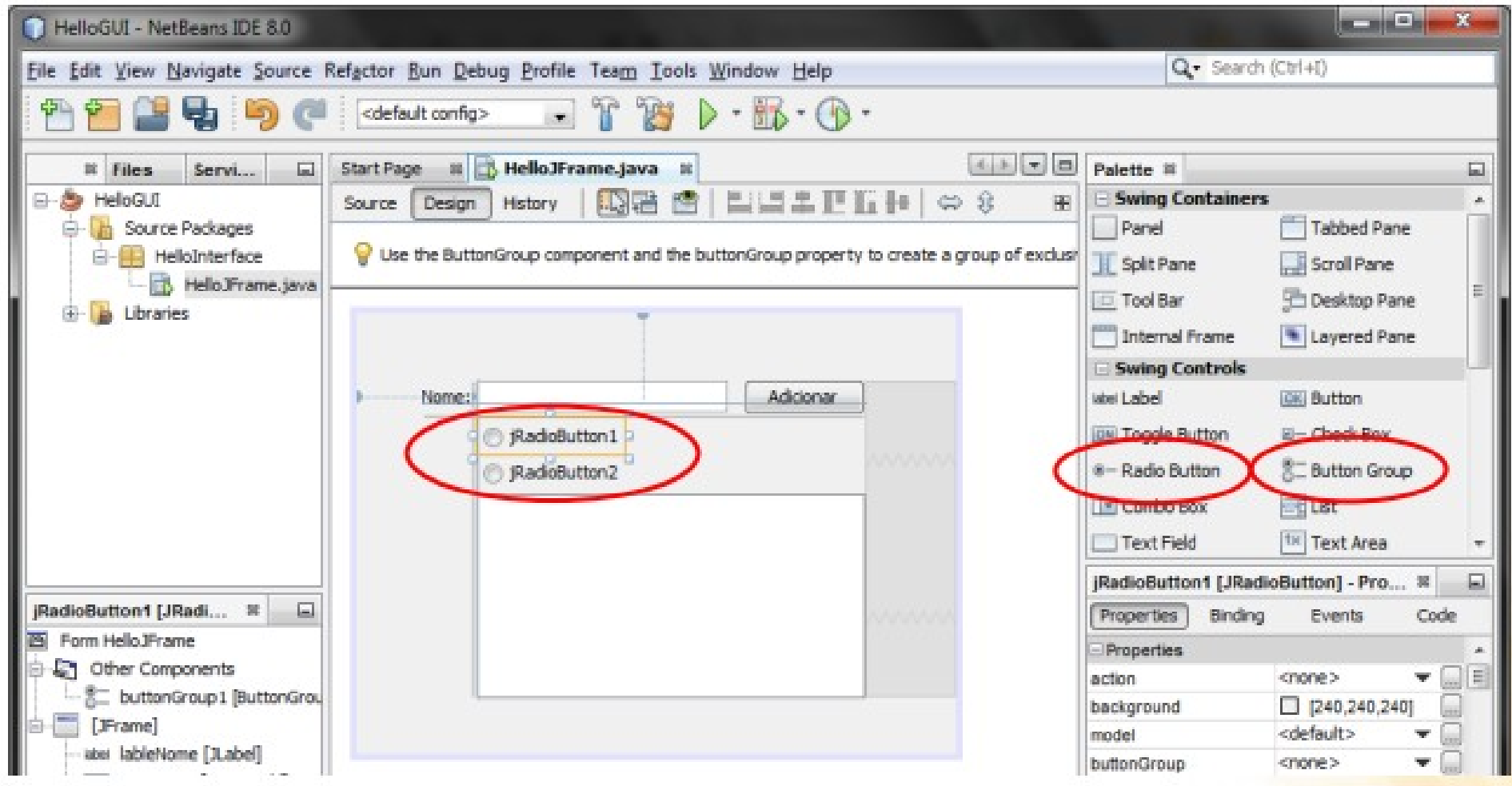
- **Exemplo 1** – Mostrar mensagem com o conteúdo do TextField:

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



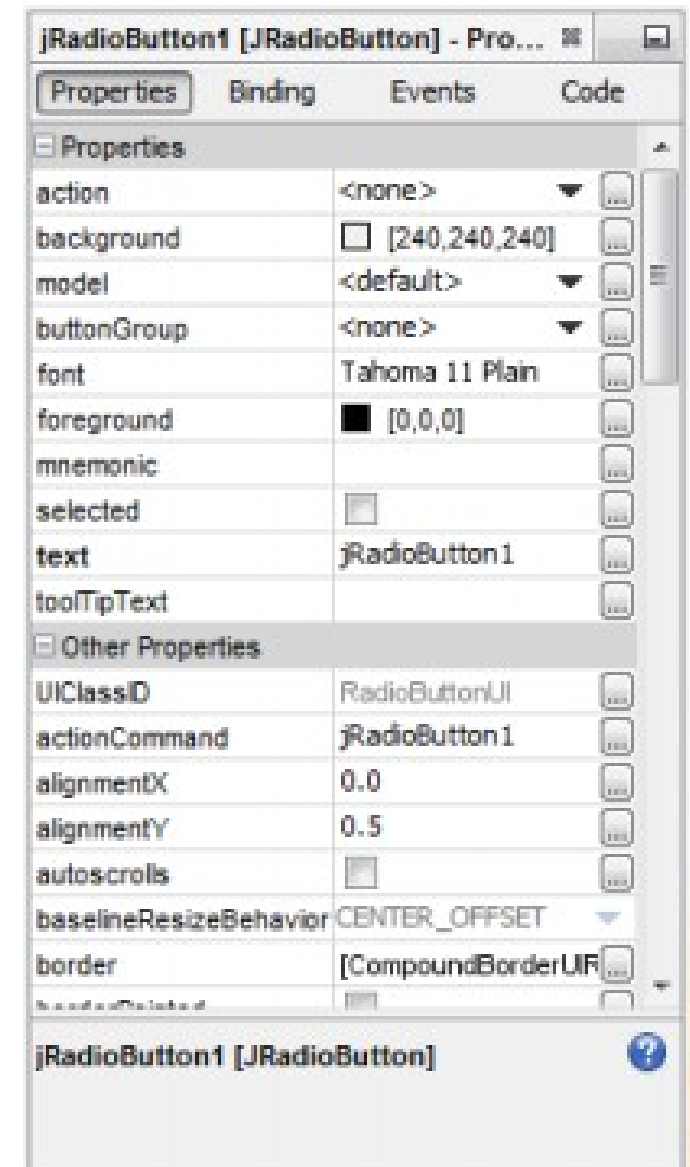
Componentes básicos - **RADIOBUTTON** e **BUTTONGROUP**

- Componentes que permitem a seleção de opções.



Componentes básicos - **RADIOBUTTON**

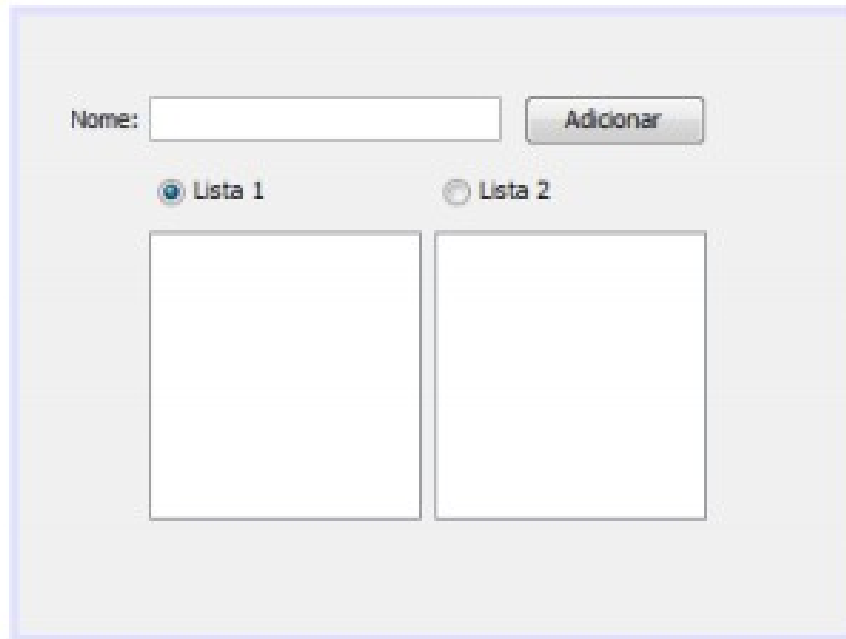
- Principais Propriedades (JRadioButton):
 - text;
 - buttonGroup;
 - Selected;
 - foreground;
 - background;
 - font;
 - icon;
 - toolTipText;
 - border;
 - enabled;



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 text input field. To the right of the input field is a button labeled "Adicionar". Below the input field, there are two radio buttons. The first radio button is selected and is labeled "Lista 1". The second radio button is not selected and is labeled "Lista 2". Below each radio button is a large, empty rectangular box, presumably for displaying the contents of the lists.

Usando o RadioButton

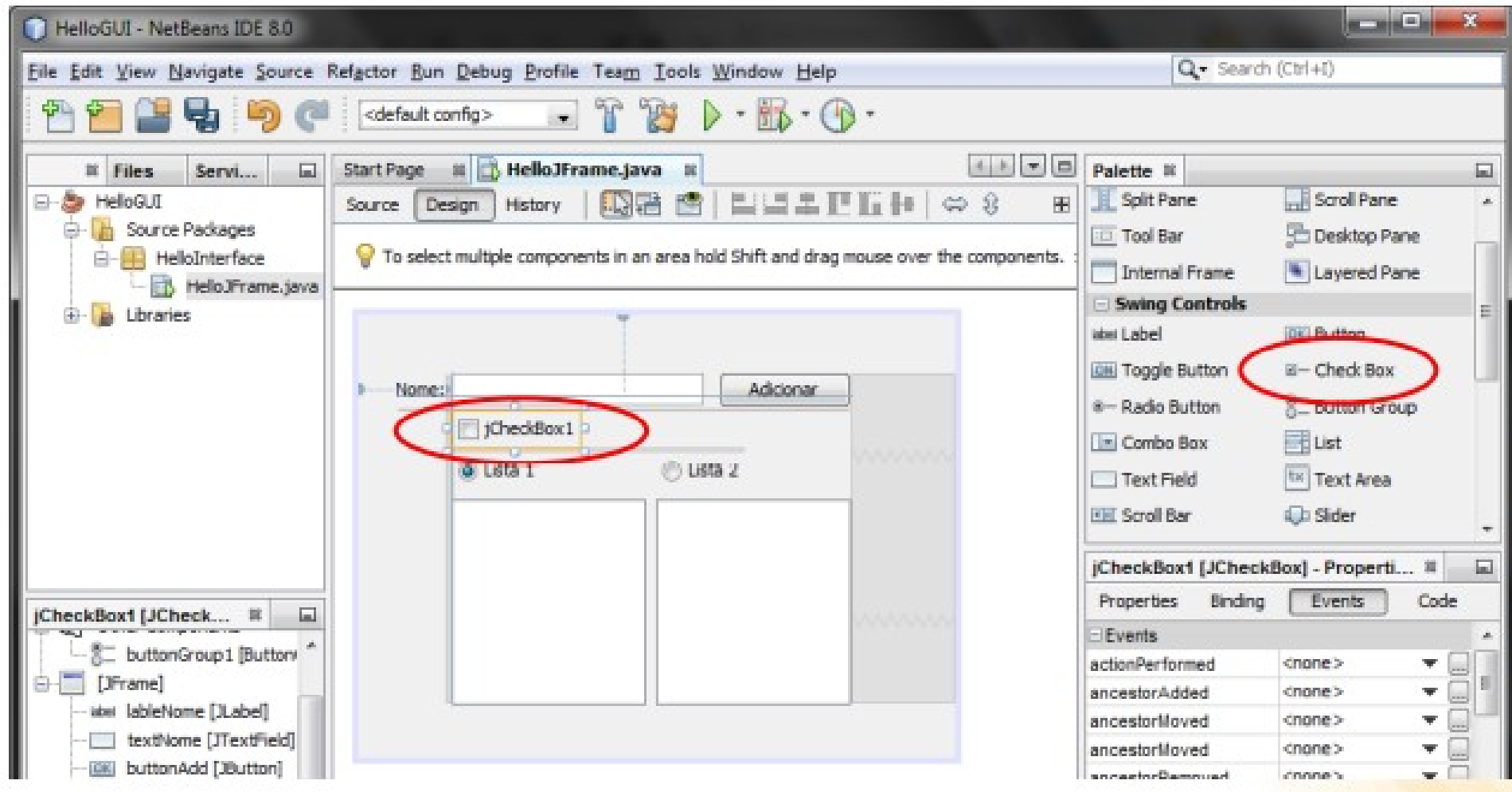
- **Exemplo** – Adicionar o conteúdo do TextField nas Lists de acordo com a opção selecionada no RadioButton.

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



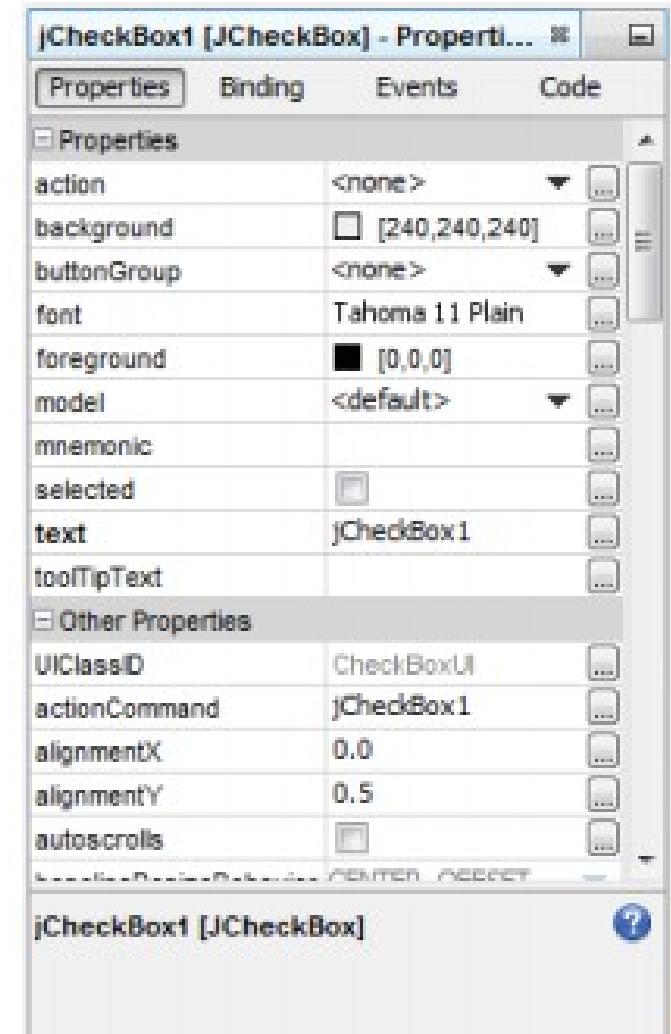
Componentes básicos - CHECKBOX

- Componente que permite a seleção de opções



Componentes básicos - CHECKBOX

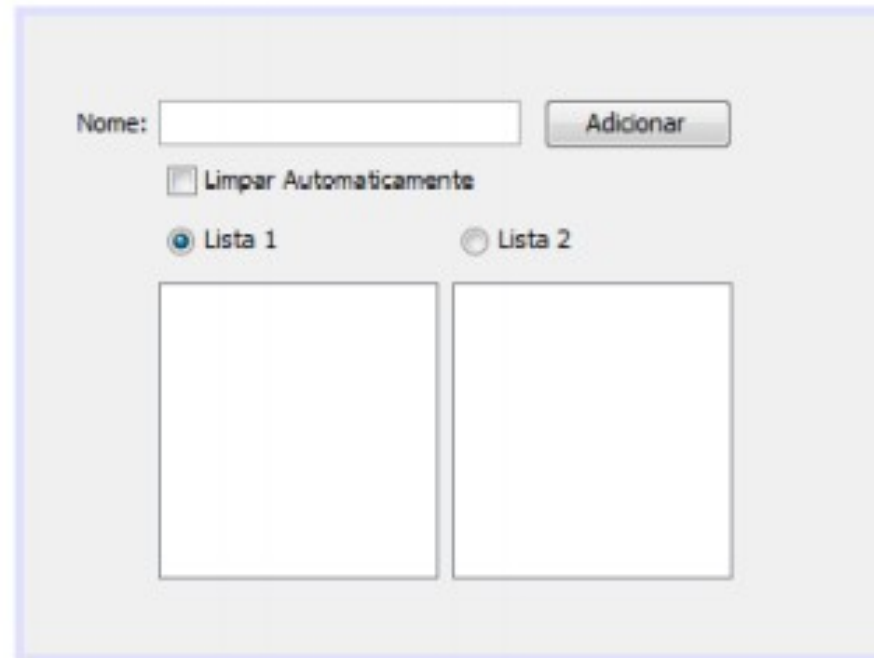
- Principais Propriedades (JCheckBox):
 - text;
 - buttonGroup;
 - Selected;
 - foreground;
 - background;
 - font;
 - icon;
 - toolTipText;
 - border;
 - enabled;



Usando o CheckBox

- **Exemplo** – 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, there are two empty rectangular boxes representing the lists.

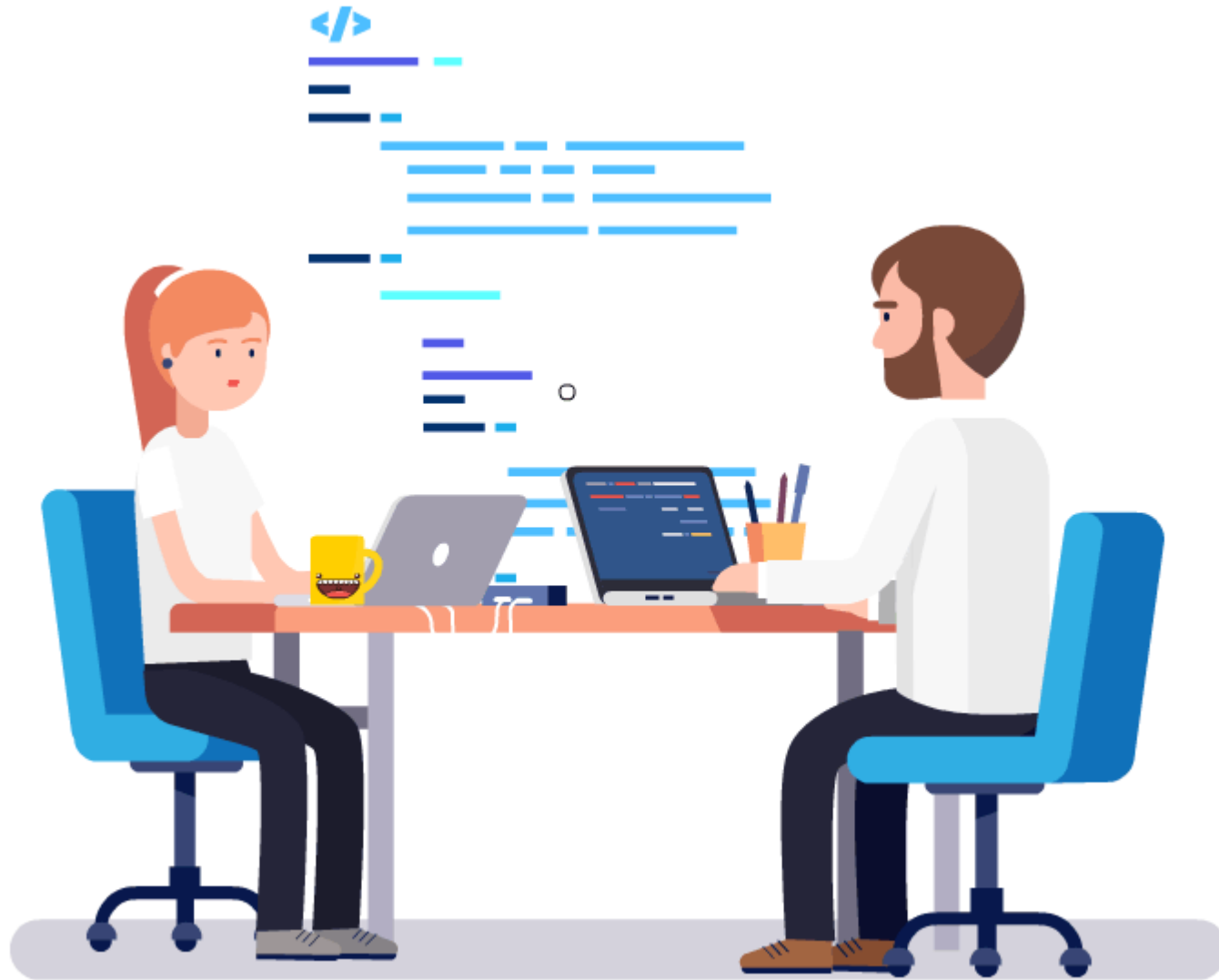
Usando o CheckBox

- **Exemplo** – 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("");  
}  
  
...
```

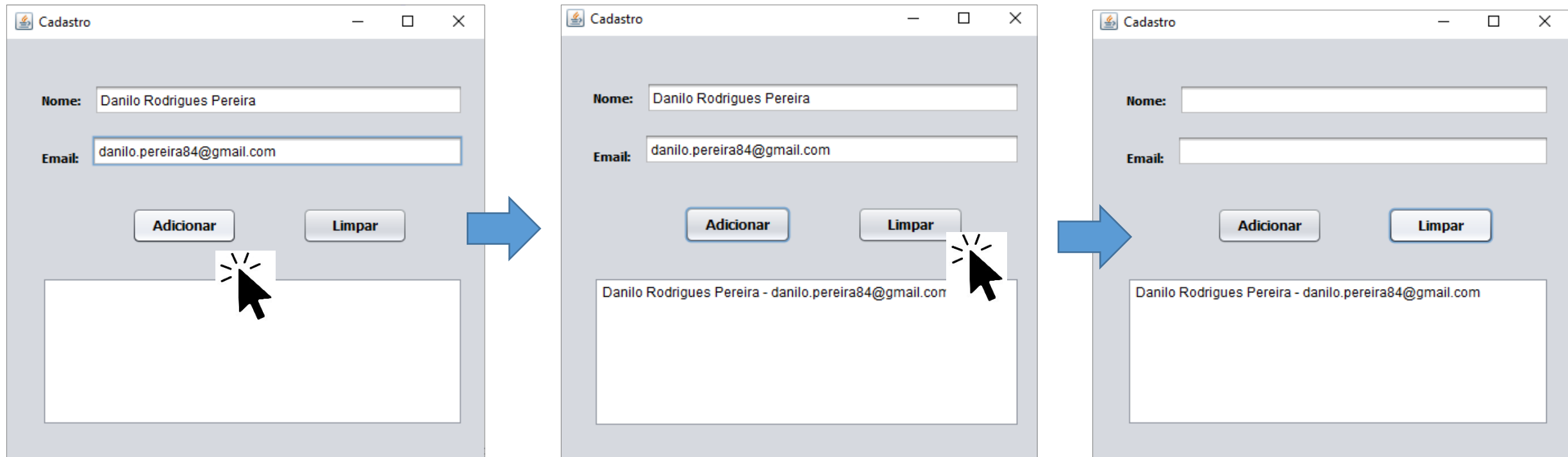


VAMOS PRATICAR ?



EXERCICIO DE FIXAÇÃO

- 1) Para fixarmos o conteúdo que aprendemos nessa aula, vamos criar uma aplicação Swing que:
- **Botão Adicionar:** Obter o nome e email de um usuário e adiciona na lista
 - **Botão Limpar:** Limpar o nome e usuário

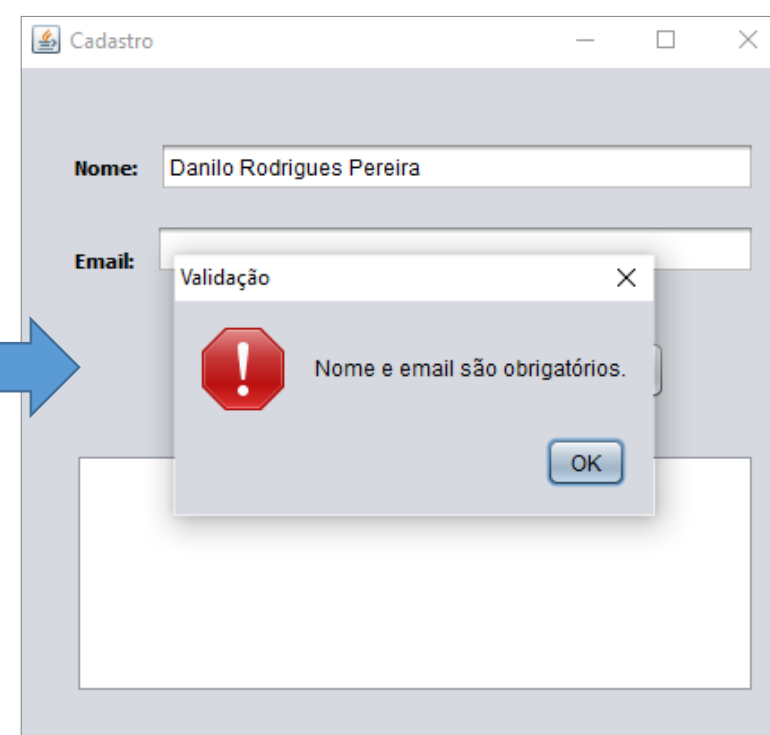
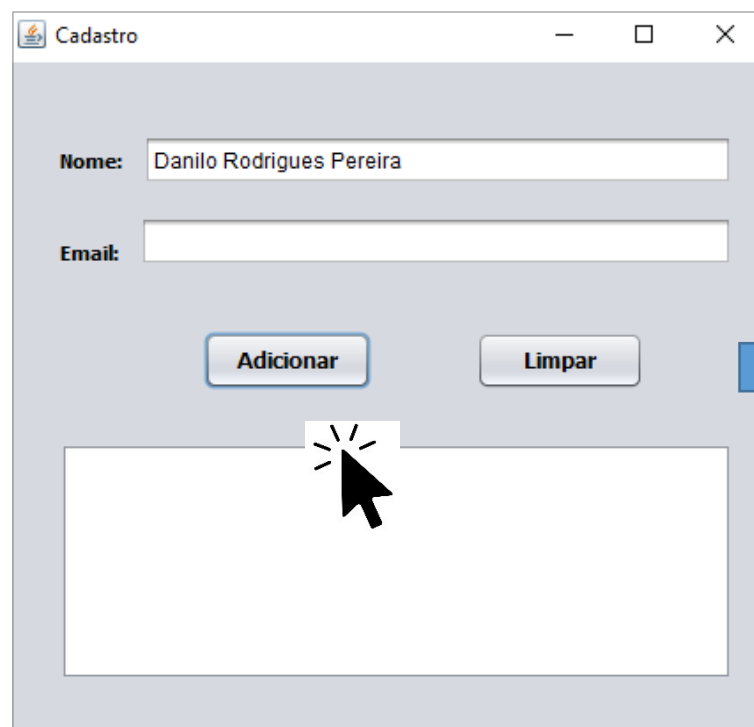
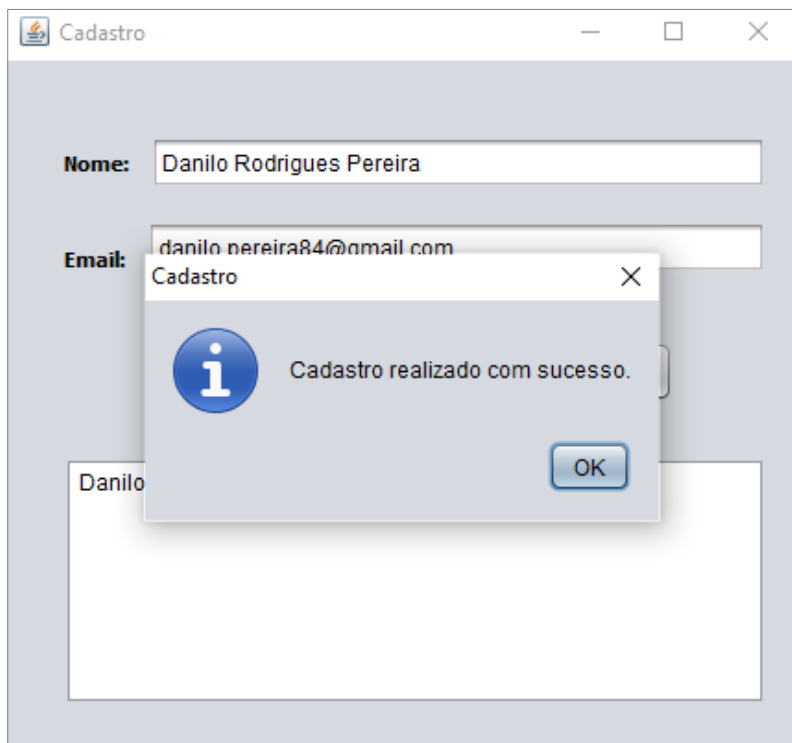


EXERCICIO DE FIXAÇÃO

2) Modificar o exercício anterior de acordo com a regra a seguir.

- **Botão Adicionar:**

- Obter o nome e email de um usuário e adiciona na lista **e exibir uma mensagem de sucesso**
- **Caso os campos nome e email, não forem preenchidos, exibir mensagem informando que os campos são obrigatórios**



EXERCICIO DE FIXAÇÃO

2) Desenvolva uma janela para a aplicação de cálculo da poupança. Essa janela deve ter a aparência da figura abaixo. Para calcular o total poupado é necessário:

- a. Para cada mês de contribuição é necessário calcular os juros (valor * juros);
- b. O percentual é informado em % (ex: 5%);
- c. O valor do depósito mensal é informado em "R\$";



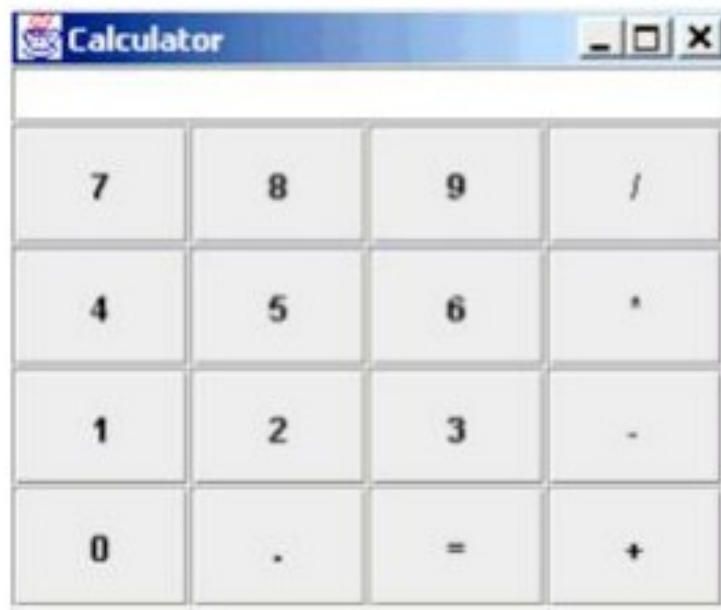
The image shows a screenshot of a software window titled "PoupeX". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. Inside the window, there are four input fields arranged vertically, each preceded by a label: "Juros ao mês %:", "Num. de anos:", "Depósito mensal R\$:", and "Total poupado R\$:". The "Total poupado R\$" field is disabled, indicated by a light gray background. At the bottom right of the window is an "OK" button.

el "Total poupado

EXERCICIO DE FIXAÇÃO

3) Implementar uma calculadora.

- a. O campo de texto não pode ser editável através do teclado;
- b. Os botões devem alterar o valor do campo de texto;



EXERCICIO DE FIXAÇÃO

4) Escreva um programa que jogue “adivinhe o número”: o programa escolhe um número a ser adivinhado, selecionando-o randomicamente no intervalo de 1- 100. O programa mostra uma mensagem (JLabel) – Eu tenho um número entre 1 e 100, você pode adivinhá-lo? Entre com seu chute.

- Um JTextField deve ser usado na captura do valor. Para cada tentativa de adivinhação um Label deve apresentar o intervalo em que o número encontra-se.
- Quando a resposta estiver correta, você deve mostrar a frase “Correto!” e não permitir mais a edição no JTextField.
- Um JButton deve permitir um novo jogo. Quando o JButton é clicado, um novo número randômico deve ser gerado e o JTextField ser editável.

