

# Interface Gráfica

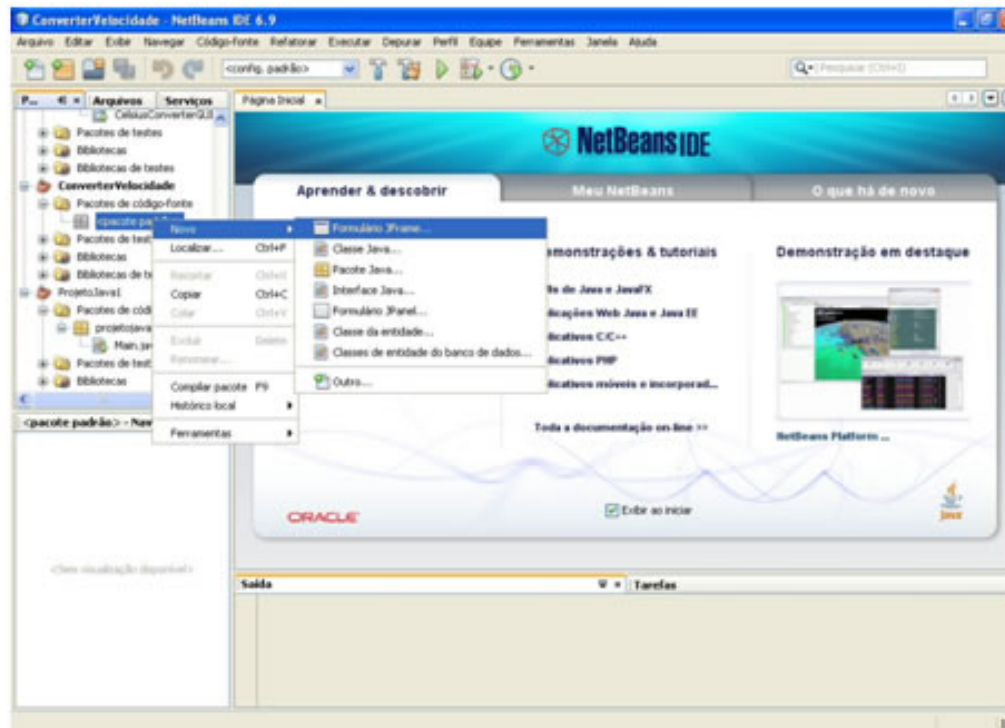
## - Swing

# Swing com o NetBeans

- Para a **construção** de **aplicativos gráficos**, o construtor de **interfaces gráficas** do **NetBeans** IDE, torna a **construção** de **interfaces** com o **usuário** uma **simples tarefa** de **arrastar** e **soltar**;
- Sua **geração automática** de **código** simplifica bastante o **processo** de **desenvolvimento** das **GUIs** (*Graphical User Interface* ou Interface Gráfica com o Usuário), permitindo o **foco** do programador na **lógica da aplicação**, e não na **infraestrutura** necessária ao aplicativo.

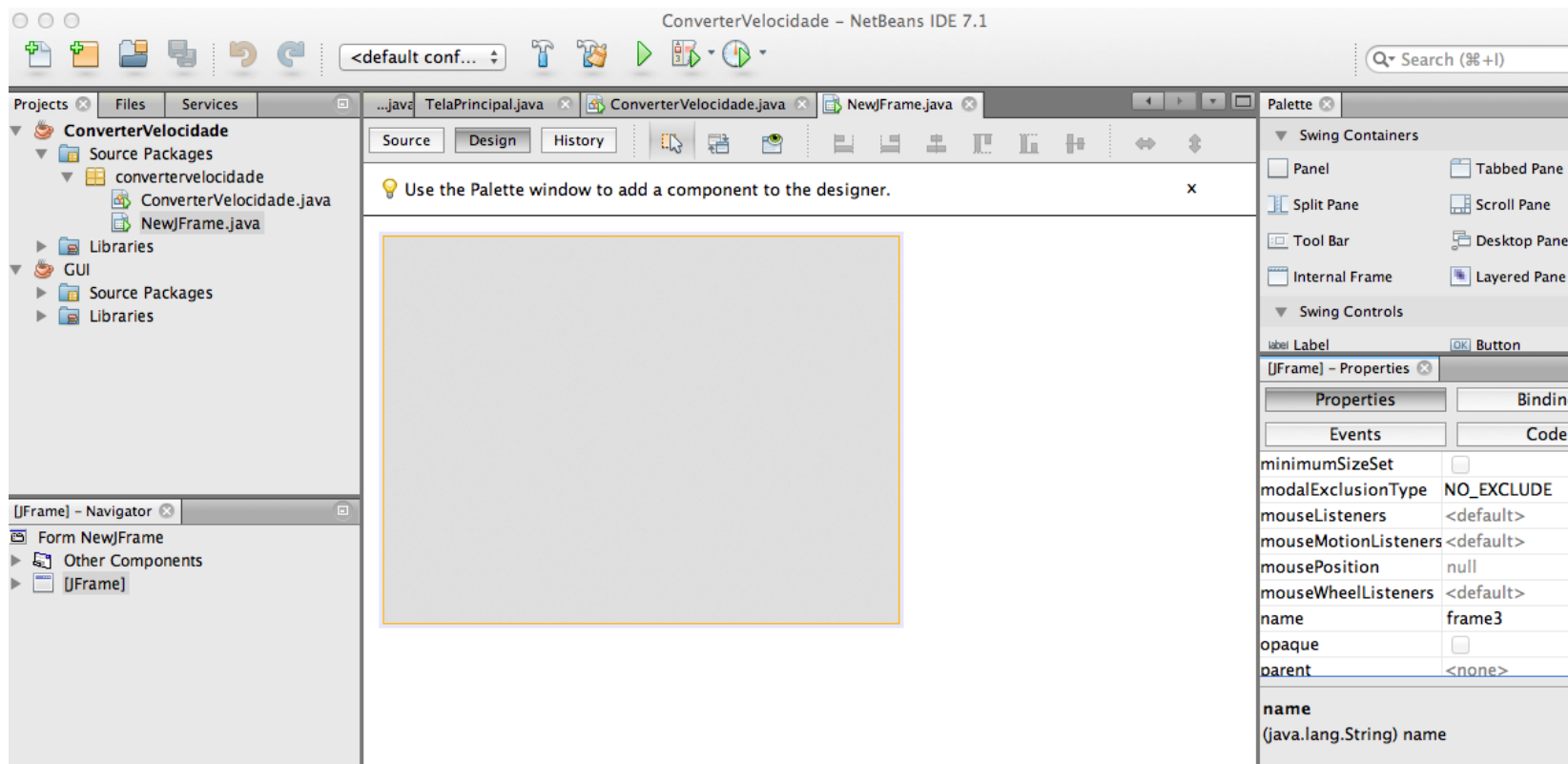
# Aplicação Simples

- ❑ Criem um **novo projeto** com o nome **ConverterVelocidade**;
- ❑ Certifique-se de que a opção (*check box*) **Criar classe principal** esteja **desmarcada**.
- ❑ Adicione um novo formulário **JFrame** clicando com o botão direito do mouse sobre o nome do projeto (**ConverterVelocidade**) e seguindo o caminho,



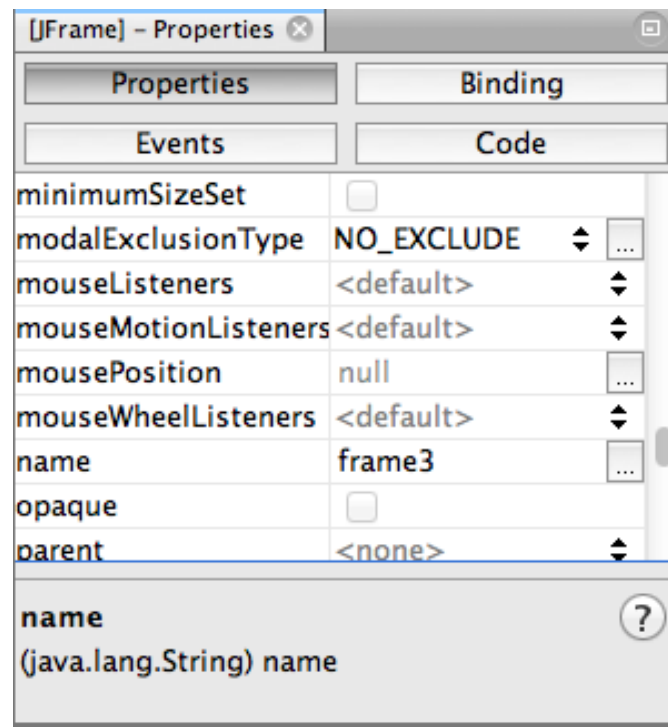
# Opções de Visualização

- O **NetBeans** disponibiliza **duas opções** de **visualização**: a vista **gráfica** do projeto e a vista do **código** fonte;



# Propriedades

- No modo de **visualização Projeto** clicando no **componente JFrame** aparecerá, no canto **inferior direito**, o painel de **propriedades** com várias **características** do componente que **poderão** ser **alteradas**



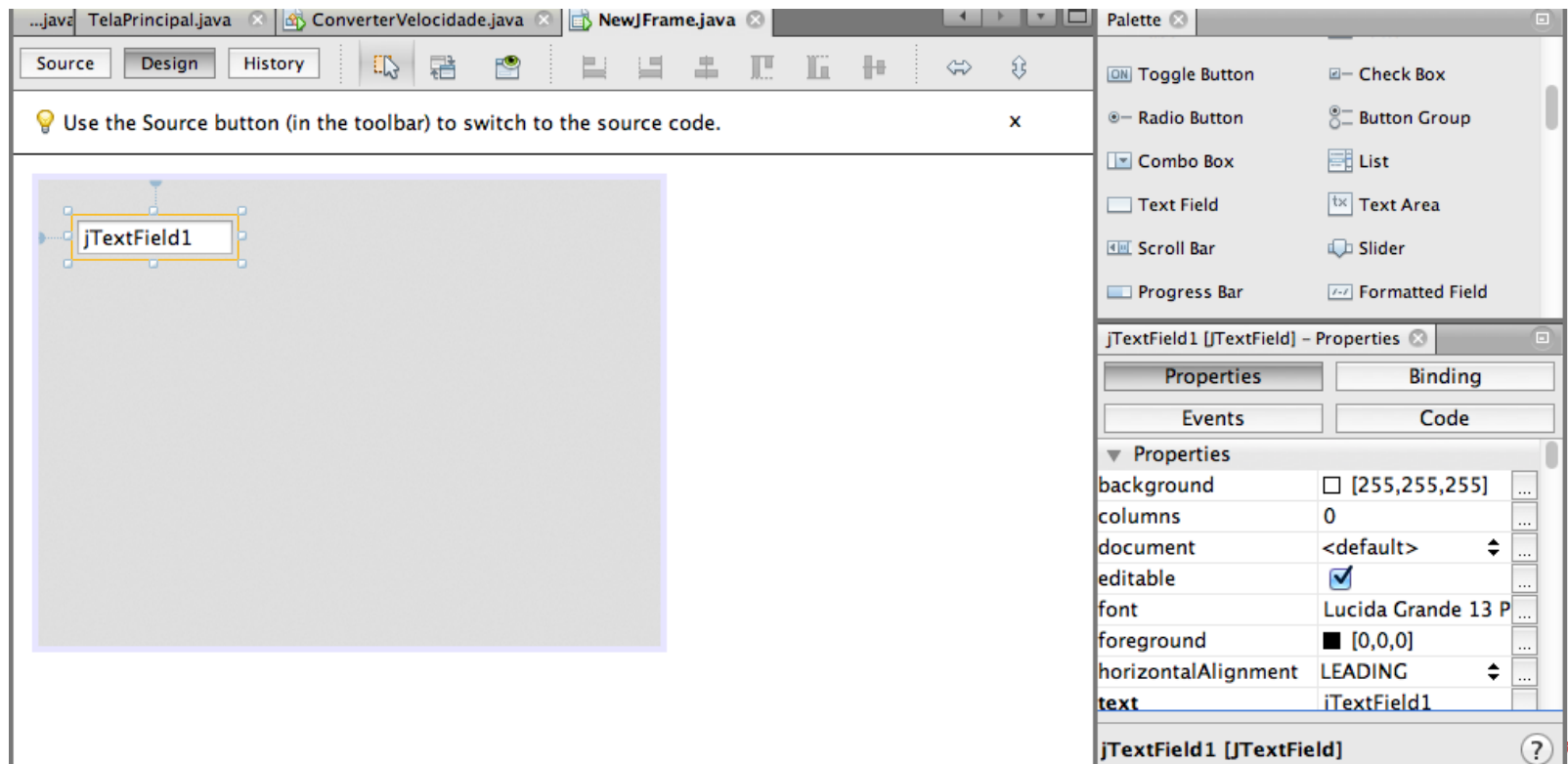
# Propriedades do JFrame



- ❑ No subitem **Propriedades** do menu, você poderá utilizar:
  - **title** – permite **inserir** um **texto** que aparecerá no **topo** da **janela** do **aplicativo**.
  
- ❑ Na seção '**Outras propriedades**', você poderá alterar também:
  - **reizable** – permite o usuário redimensionar a janela do aplicativo.
  
  - **undecorated** – define a exibição ou não da borda da janela. Por padrão, essa opção se apresenta desmarcada, exibindo a borda externa ao **JFrame**.

# Campo de texto (TextField)

- Em seguida, na janela à direita, clique e **arraste** um campo de **texto** (**JTextField**) para **dentro** da área do **JFrame**.



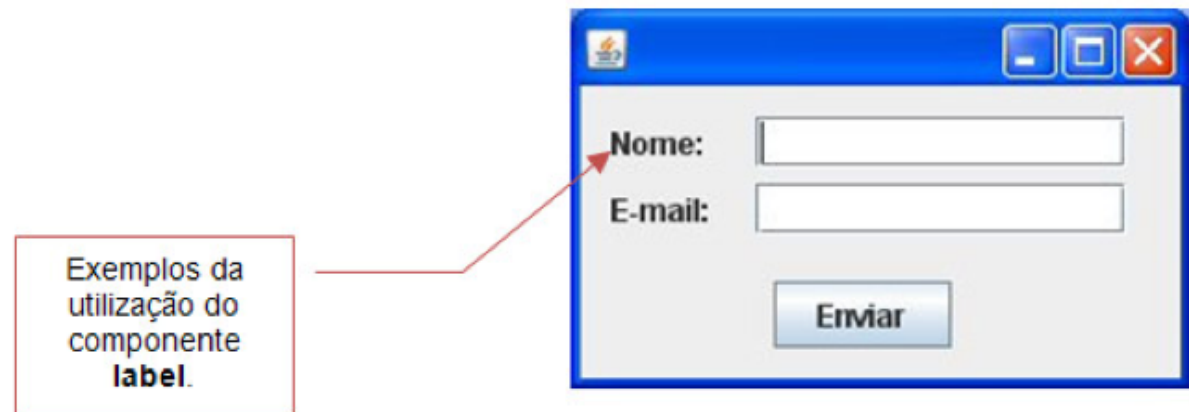
# Campo de texto (TextField)

- Esse **componente** é muito **comum** em **formulários** onde o usuário **digita** um **texto** como uma **informação** a ser **enviada**: dados de nome, telefone e e-mail em um cadastro pessoal.
  
- Algumas de suas propriedades incluem
  - **text** – essa propriedade é utilizada para você especificar um texto padrão para o componente quando a aplicação for executada, ao invés de ficar em branco.
  
  - **tamanho vertical e tamanho horizontal** – utilizado para alterar as dimensões de altura e largura do componente, permitindo mudar o tamanho padrão.
  
  - **background** – permite alteração da cor de plano de fundo do componente, disponibilizando a especificação da cor em diversos padrões.



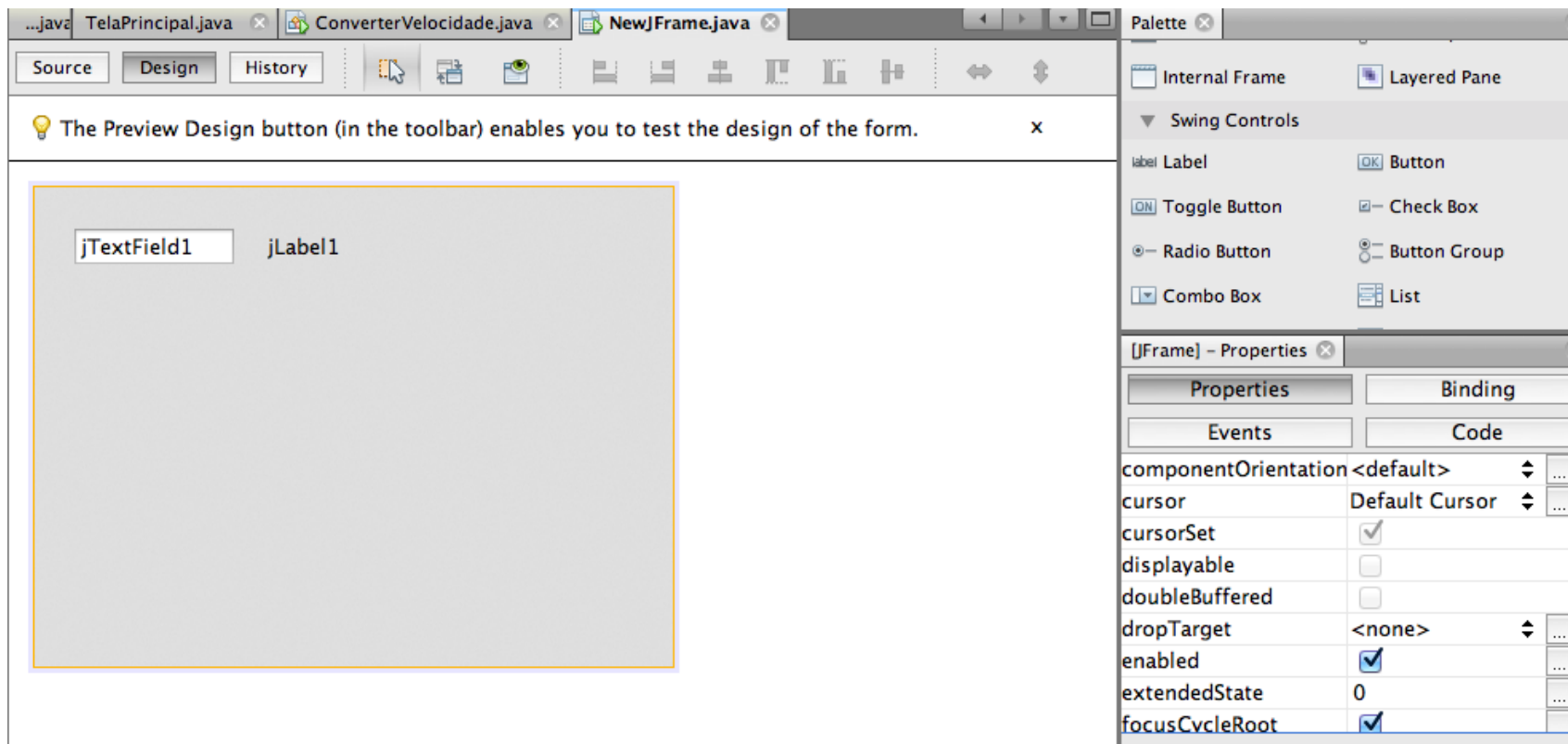
# Rótulo (label)

- ❑ Esse é um outro **componente** bastante **comum**. Podemos dizer até que está **presente** em toda **interface gráfica**.
- ❑ O **Rótulo**, mais conhecido pelo termo em inglês **label**, é todo o texto que **encontramos** na interface que tem a **função** de informar o usuário.



# Rótulo (label)

- Agora adicione o rótulo (**JLabel**) seguindo as linhas-guia, posicionando-o ao lado do **JTextField**;



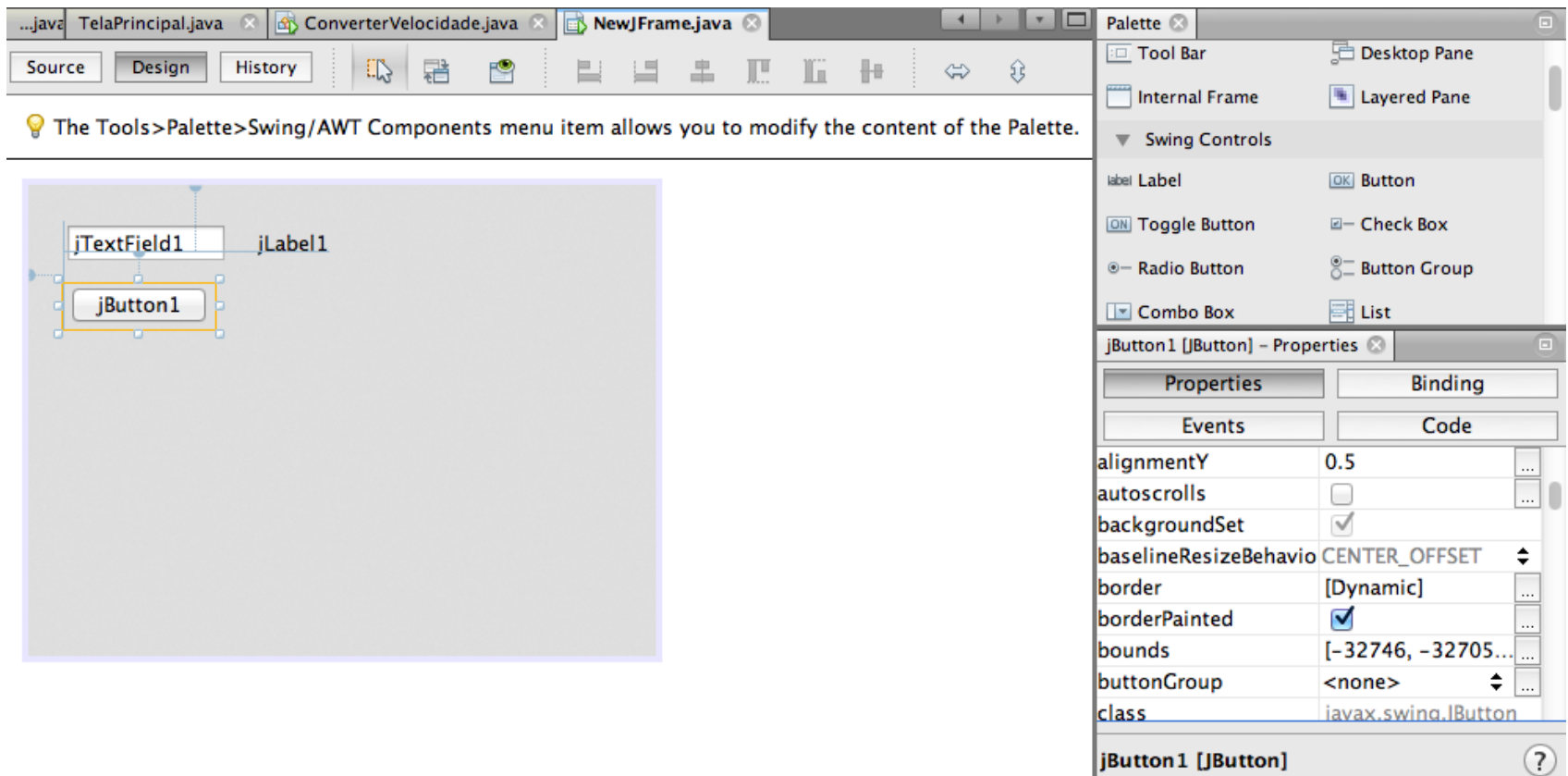
The screenshot shows an IDE window titled "NewJFrame.java" in Design mode. The main canvas displays a window with a text field labeled "jTextField1" and a label "jLabel1" positioned to its right. The "Palette" panel on the right shows "Swing Controls" with "Label" selected. Below the palette, the "Properties" panel for the selected component is visible, showing various attributes like "componentOrientation", "cursor", "cursorSet", "displayable", "doubleBuffered", "dropTarget", "enabled", "extendedState", and "focusCycleRoot".

# Rótulo (label)

- Vejamos mais algumas **propriedades** que podemos **manipular** para **personalizar** o **componente**.
  - **font** – com essa **propriedade**, é possível **configurar diversas** outras **propriedades** em relação à fonte, como **tipo**, **estilo** e **tamanho**.
  - **foreground** – utilizado para **alterar** a **cor** da **fonte**.
  - **horizontalAlignment** – permite o **alinhamento horizontal** do texto, da propriedade **text** em relação às **dimensões** do **componente**.
  - **verticalAlignment** – permite o **alinhamento** vertical do texto, da propriedade **text** em relação às **dimensões** do componente.

# Botão (Jbutton)

- Agora, adicione um botão (**JButton**) logo abaixo do **JTextField**. Lembre-se de utilizar as linhas guia para alinhar os componentes.



The screenshot shows an IDE window with three tabs: TelaPrincipal.java, ConverterVelocidade.java, and NewJFrame.java. The Design view is active, showing a window with a JTextField1, a JLabel1, and a JButton1. The JButton1 is positioned below the JTextField1. A tooltip message reads: "The Tools>Palette>Swing/AWT Components menu item allows you to modify the content of the Palette." The Palette window on the right shows the Swing Controls section, with the JButton component selected. The Properties window for JButton1 [JButton] is also visible, showing various properties such as alignmentY, autoscrolls, backgroundSet, baselineResizeBehavior, border, borderPainted, bounds, buttonGroup, and class.

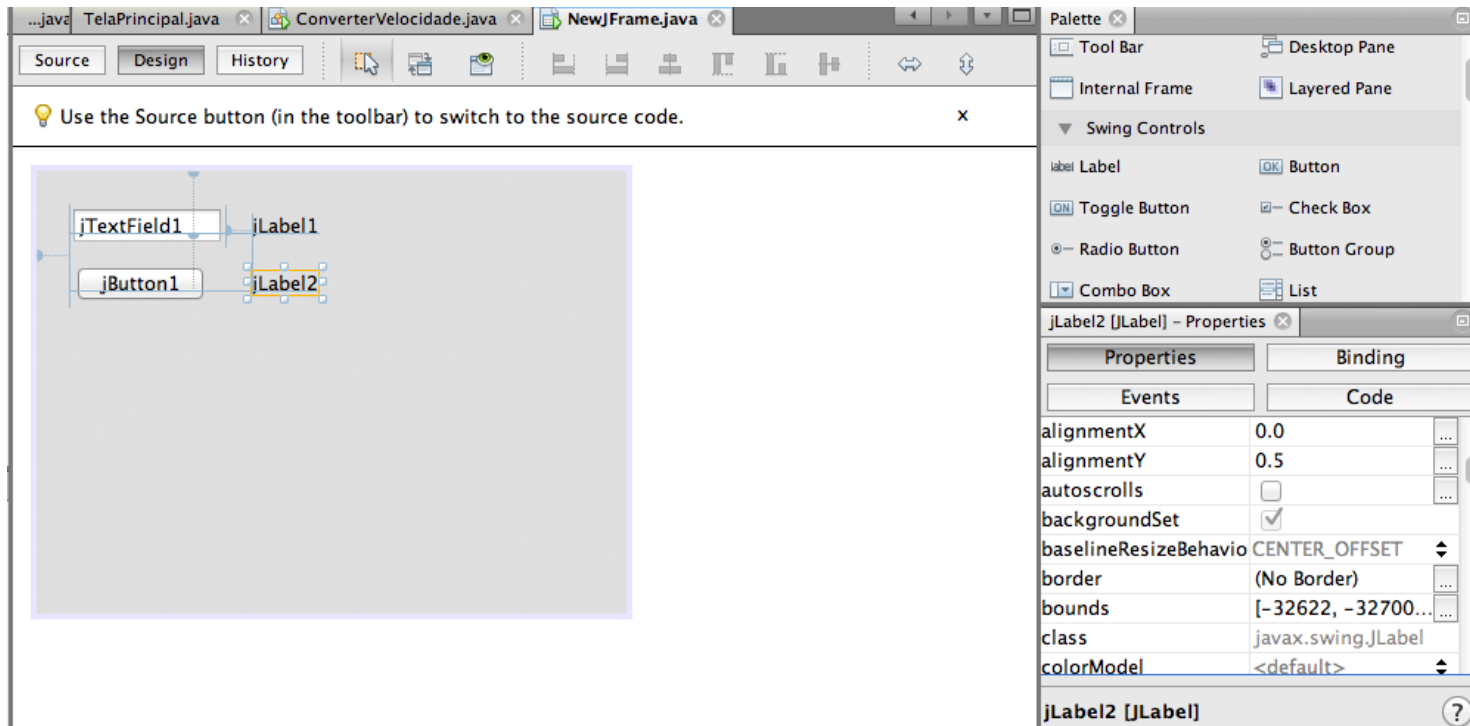
Properties	Binding
alignmentY	0.5
autoscrolls	<input type="checkbox"/>
backgroundSet	<input checked="" type="checkbox"/>
baselineResizeBehavior	CENTER_OFFSET
border	[Dynamic]
borderPainted	<input checked="" type="checkbox"/>
bounds	[-32746, -32705...]
buttonGroup	<none>
class	javax.swing.JButton

# Botão (Jbutton)

- Na lista de propriedades **disponíveis**, você poderá **personalizar** o seu **botão** e deixá-lo do seu jeito. Algumas **propriedades** deste componente **incluem**:
  - **toolTipTex** – essa propriedade permite **incluir** um **texto explicativo** que aparece para o usuário quando o **cursor** passa por cima do **componente**;
  - **enabled** – se esse **check box** estiver **marcado** significa que o componente ficará **habilitado**, ou seja, no caso do **botão** poder ser **clicado**, caso contrário, ficará **desabilitado** e não será **executada nenhuma ação** definida para esse componente. Por padrão, esse **check box já é marcado**.

# Fazendo Aplicação

- ▣ Agora adicione um segundo **JLabel** ao lado do botão;

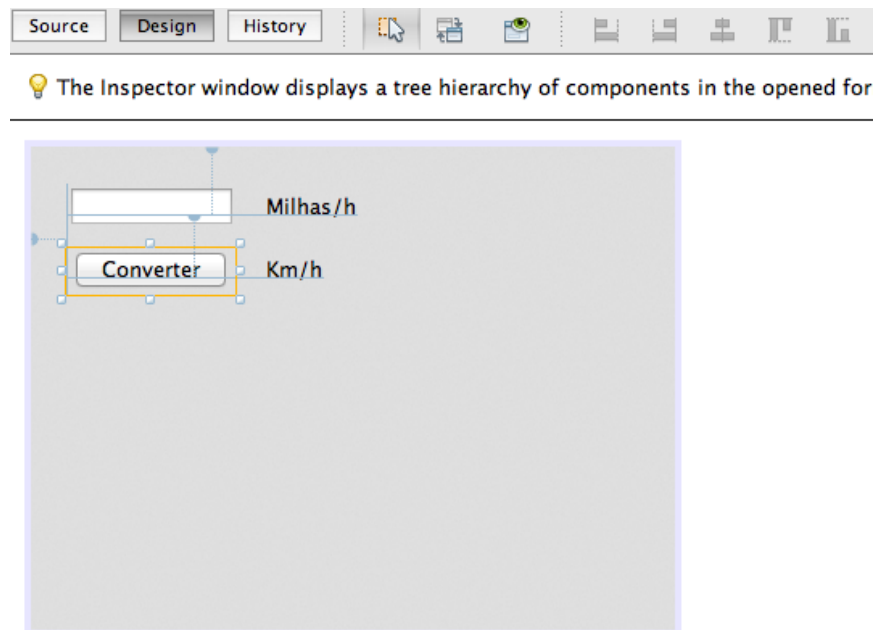


The screenshot shows an IDE with three open Java files: TelaPrincipal.java, ConverterVelocidade.java, and NewJFrame.java. The main window is in Design mode, displaying a GUI with a JTextField, a JButton, and two JLabels. A Palette on the right shows various Swing Controls, and a Properties window for JLabel2 is open, showing properties like alignmentX, alignmentY, autoscrolls, backgroundSet, baselineResizeBehavior, border, bounds, class, and colorModel.

Properties	Binding
alignmentX	0.0
alignmentY	0.5
autoscrolls	<input type="checkbox"/>
backgroundSet	<input checked="" type="checkbox"/>
baselineResizeBehavior	CENTER_OFFSET
border	(No Border)
bounds	[-32622, -32700...]
class	javax.swing.JLabel
colorModel	<default>

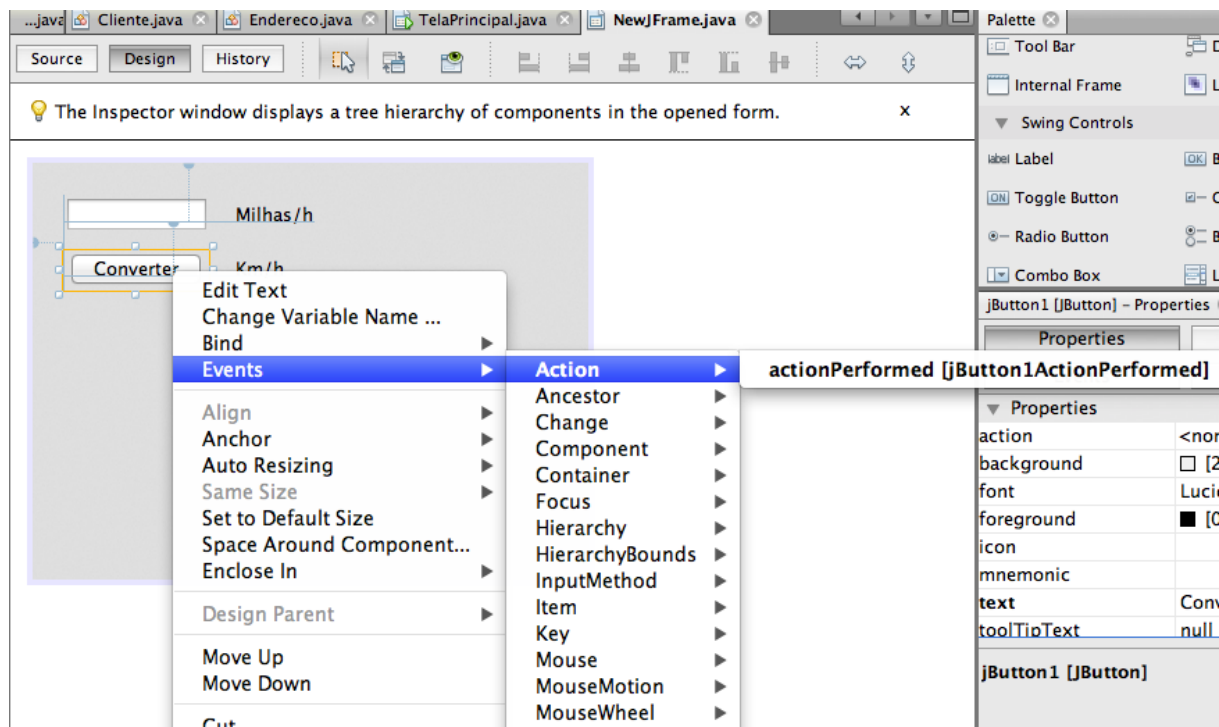
# Fazendo Aplicação

- ❑ Para alterar o texto do **JTextField**, clique no componente e procure no painel de propriedades o campo **text**. Ao apagar o texto padrão **JTextField1**, que constava anteriormente,
- ❑ Como não queremos nenhum texto inicial no **JTextField**, manteremos esse campo vazio. Agora, de forma similar, altere o campo **text** dos outros componentes adicionando os nomes (**JLabel1** "milhas/h",  **JButton** "Converter" e **JLabel2** "Km/h").



# Fazendo Aplicação

- ❑ A parte realizada na GUI está completa. Vamos, agora, concluir o aplicativo inserindo os comandos de conversão ao botão.
- ❑ Clique no botão **Converter** com o botão direito do mouse.





# Fazendo Aplicação

- Aparecerá na tela a área onde deverá ser incluído o código que será executado quando o botão for apertado.
- Insira a linha de comando especificada abaixo
- **int tempFahr = (int)  
((Double.parseDouble(jTextField1.getText())) \*  
1.609344);  
jLabel2.setText(tempFahr + " Km/h");**