

Java

Utilizando Swing

O Java é como um carro básico que vem sem ar-condicionado, vidro elétrico, trava e rodas estilizadas.



Se você deseja ter ar-condicionado, pode mandar adicionar o ar-condicionado ou dar um “import” no kit, ou seja, no Pacote do ar-condicionado. Sendo assim, “import arCondicionado”.



Desta forma, todos os recursos extras precisam ser importados.



Já os recursos básicos, NÃO precisam ser importados, pois já estão funcionais. Por exemplo, ainda no carro básico, todo carro já possui farol; desta forma, não é necessário importar esse recurso – A menos que deseje possuir um farol de milha, por exemplo.

O Java é como um carro popular; ele não possui a maior parte dos Pacotes necessários. Logo, bastará dar um “import” quando precisar.



Pacote Java.lang

Por padrão, o Java já vem com o Pacote Java.lang ativo.

Este Pacote já vem com as instruções básicas de funcionamento do Java. **Estas instruções são ditas Essenciais**, como realizar operações aritméticas, fazer testes condicionais, escrever na tela. Ou seja, coisas bem simples já vêm inclusas no Java, mas na grande maioria dos casos irá precisar de acessórios adicionais, precisará de Pacotes.



Dado esse entendimento, NÃO será necessário importar o Java.lang.



Vamos ver outros exemplos de Pacotes:

Java.applet para criar Aplicativos.

Java.util para utilitários. Por exemplo, monitoramento de entrada como o teclado.

Java.math para funções matemáticas.

Java.net par redes.

Javax.sound para bibliotecas estendidas de sons.

Javax.media para mídia.

Javax.swing para algumas aplicações em janela.

JavaFX.fxml para alguns recursos de ambientes gráficos.

Biblioteca Swing

A biblioteca Swing permite criarmos Interfaces Gráficas para janelas.

Logo, poderemos criar Interfaces Gráficas para todos os ambientes em janela, como: Windows, Linux, Mac, Solaris ou qualquer sistema que possua Ambiente Gráfico.



A Biblioteca Swing provem de uma Biblioteca mais antiga, a AWT – Abstract Window Toolkit.

A AWT foi uma das primeiras Bibliotecas Gráficas para poder criar Aplicativos para ambientes de Sistemas Operacionais que possuam Janelas.

O problema da AWT é que ela deixava por conta do Sistema Operacional a aparência da Janela.

Por exemplo, a aparência de um Botão no Mac é diferente de aparência de um Botão no Windows. Desta fora, a AWT deixava a cargo do Sistema Operacional a decisão da aparência dos componentes.

Compreende como uma característica NEGATIVA da AWT a mudança das características dos componentes.

A biblioteca AWT ainda existe e pode ser utilizada através do comando “import java.awt”.



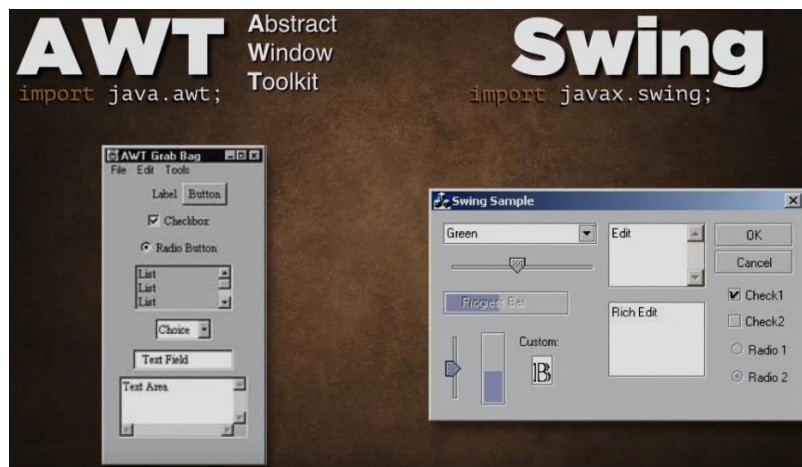
Uma tela em AWT teria uma tela semelhante à exibida na imagem abaixo.



Para resolver os problemas gerados pela Biblioteca AWT, surgiu a Biblioteca Swing.

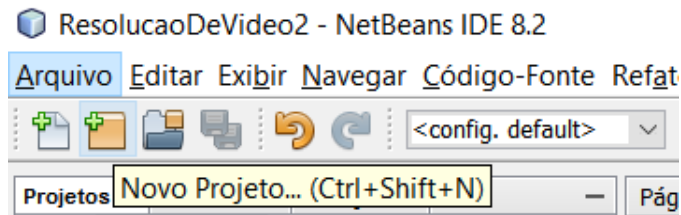
Para utilizar a Biblioteca Swing, basta inserir o comando “**import javax.swing;**”.

Uma tela possuindo Swing é mais caprichada; possui componentes visuais melhores. Ou seja, a **Biblioteca Swing permite criar Interfaces Gráficas**.

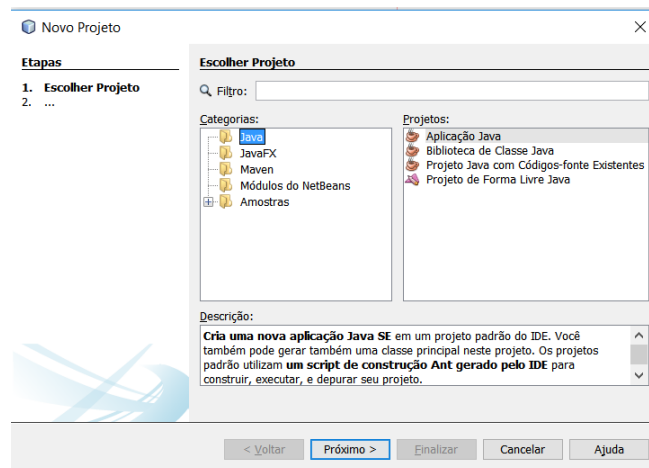


Tarefa Prática

Abra o Netbeans e clique sobre o botão Novo Projeto.



Mantenha selecionada a Categoria Java e em Projetos, Aplicação Java.

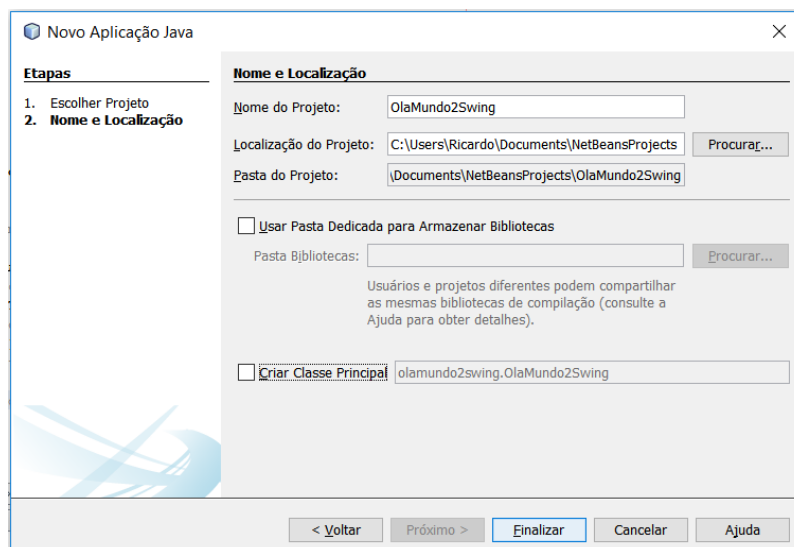


Nomeie o projeto como OlaMundo2Swing e DESMARQUE a caixa de seleção Criar Classe Principal.

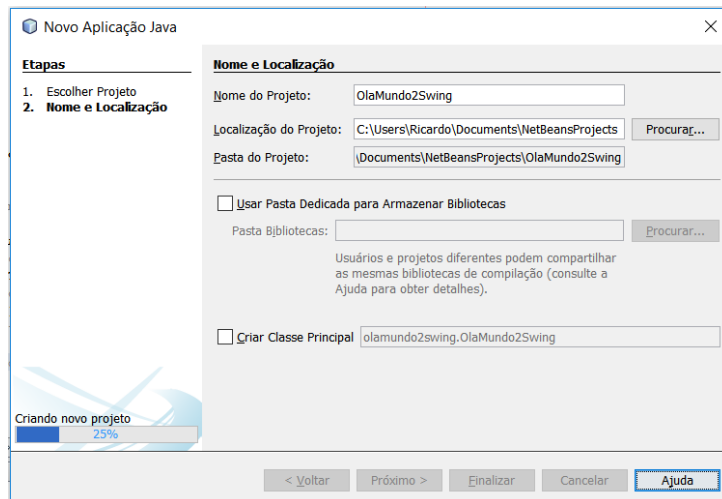
Desmarcar CRIAR CLASSE PRINCIPAL é muito importante quando criar interface Swing, pois a janela será a nossa própria Classe.

Não criando a Classe Principal, antes de finalizar, o próprio Netbeans irá questionar qual a é a Classe Principal.

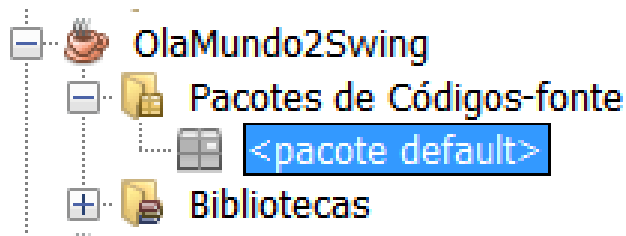
Após nomear o projeto e desmarcar Criar Classe Principal, clique no botão Finalizar.



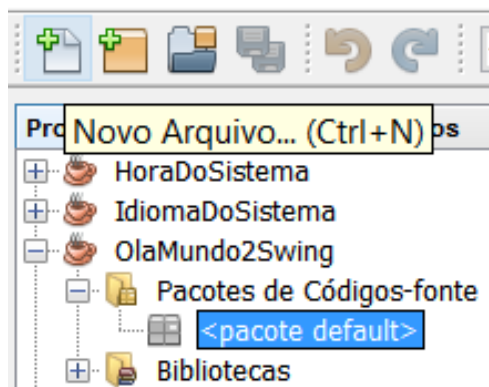
Aguarde o processo de criação dos Pacotes das Classes.



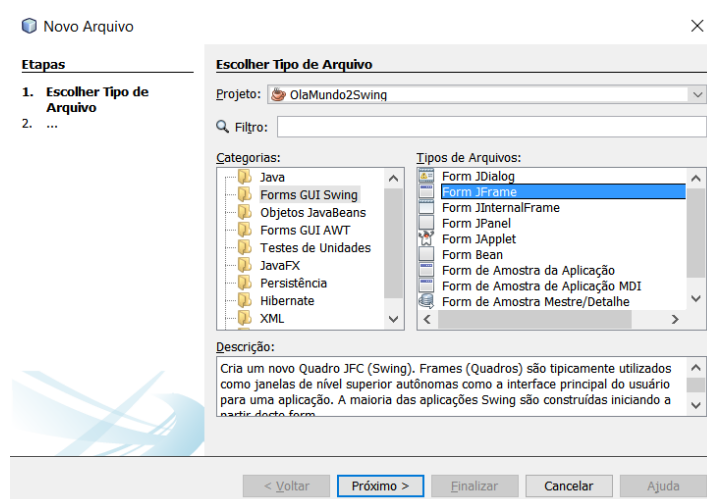
Após a criação, o Pacote Default está vazio.



Crie a primeira Classe clicando no botão Novo Arquivo, localizado da barra de tarefas.

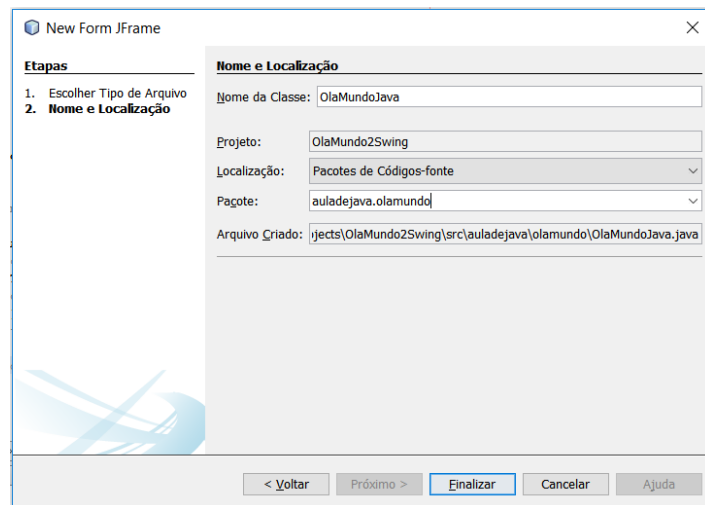


Em categoria, selecione Forms GUI Swing e em Tipos de Arquivos, selecione Form JFrame, que é uma Janela, e clique no botão Próximo.

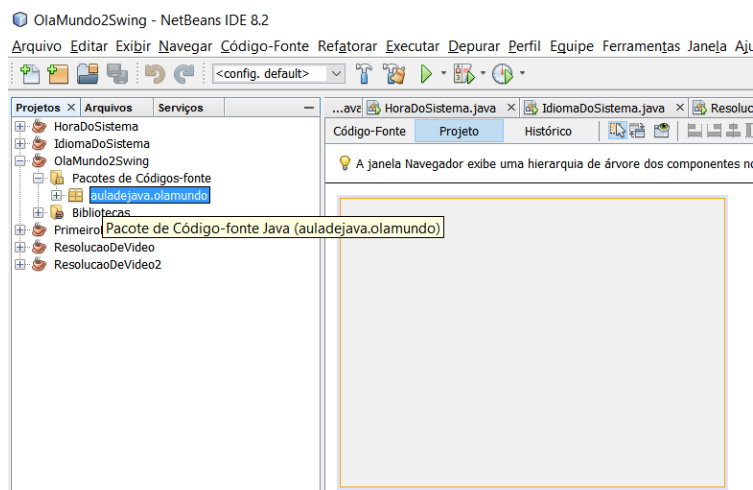


Nomeie a Classe como `OlaMundoJava` e nomeie o Pacote como `auladejava.olamundo`. Após, clique no botão Finalizar.

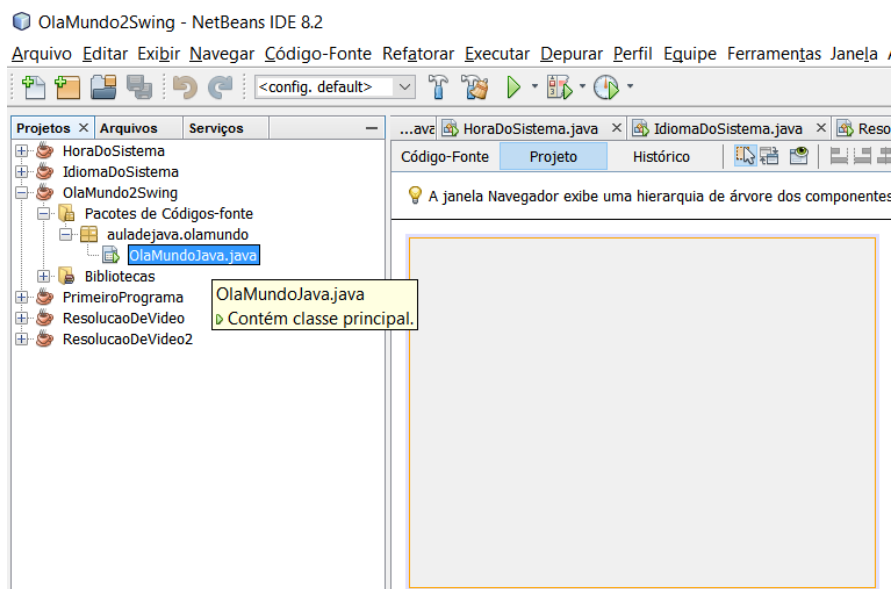
Lembrando: Classe sempre com a primeira letra maiúscula e todas as outras minúsculas, utilizando a regra do CamelCase. Nomes de Pacotes com tudo em letra minúscula.



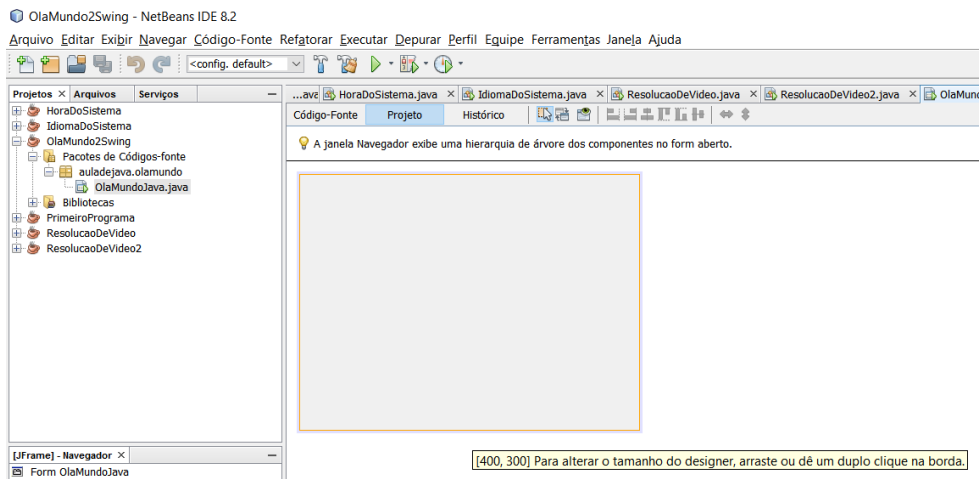
Foi criado o Pacote.



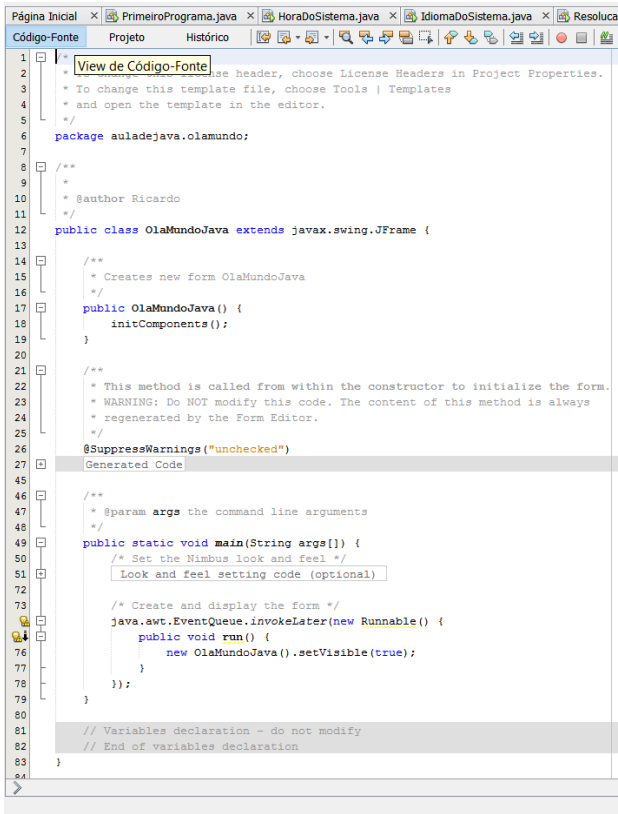
Também, foi criada a Classe.



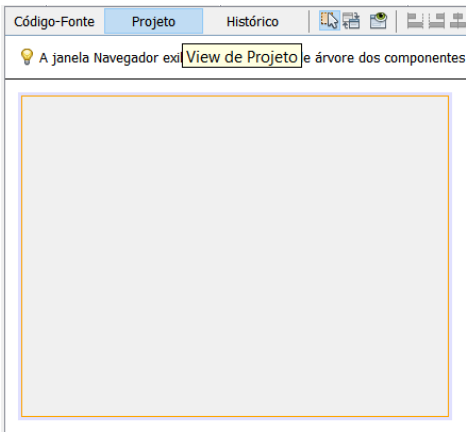
E foi criada a Janela.



O botão View de Código-Fonte é utilizado para exibir o código do programa e permitir alterações no código.



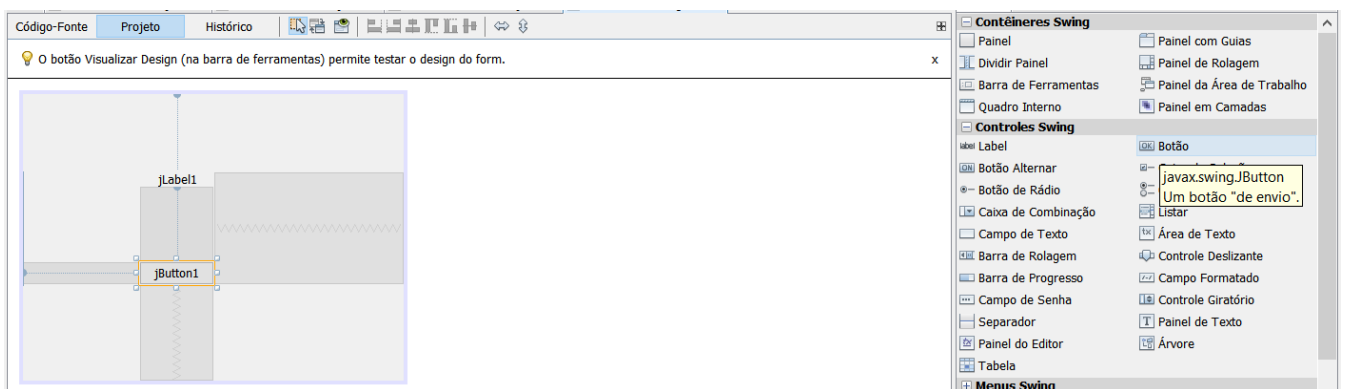
O botão View de Projeto permite alterações na Janela.



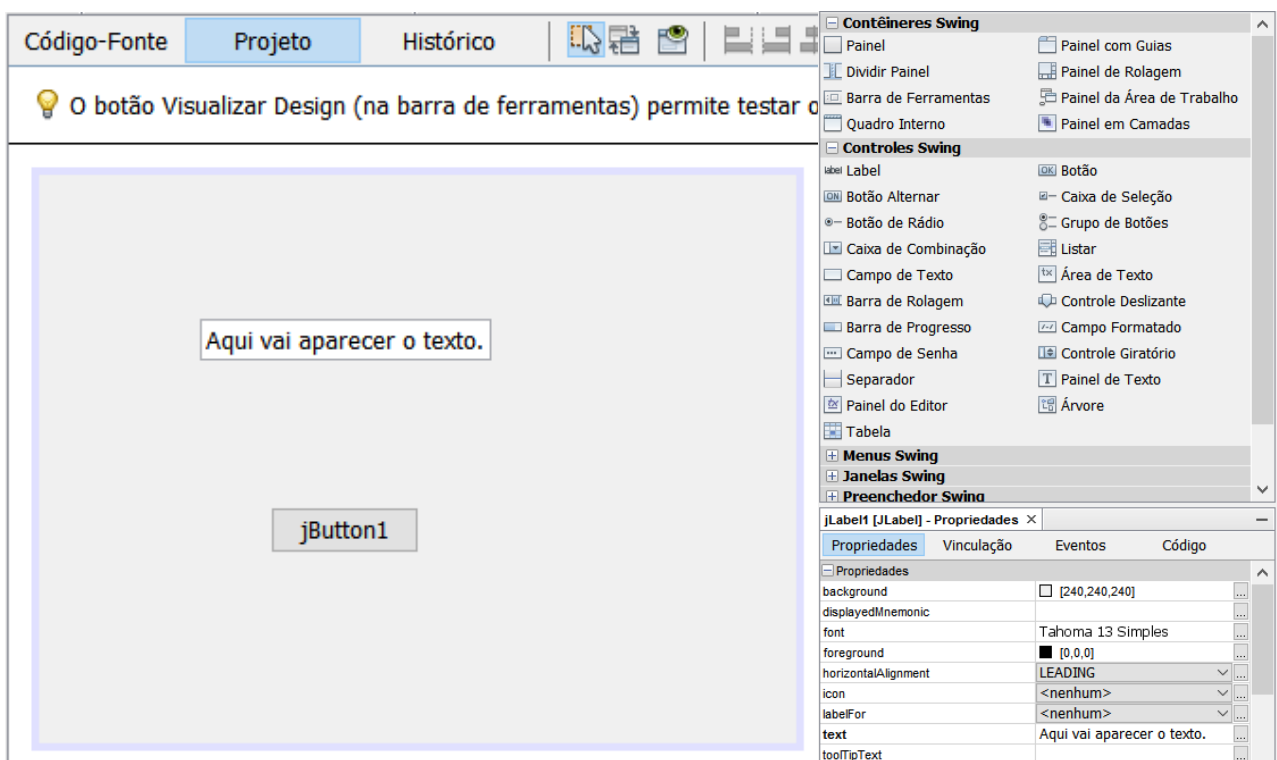
Utilizando as Paletas, clique e arraste Label para a Janela.



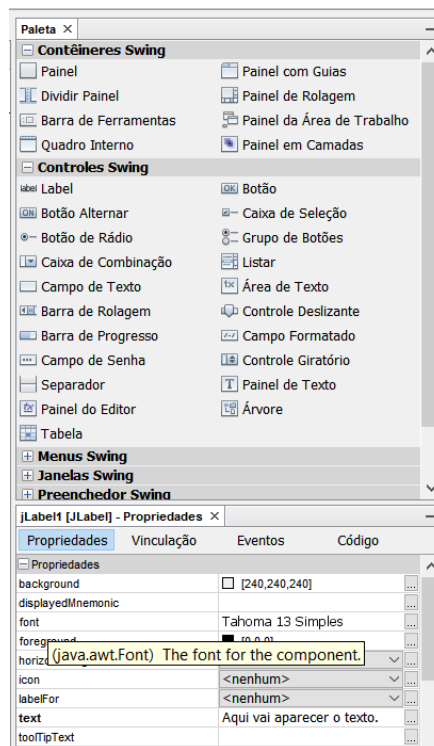
Ainda utilizando as Paletas, clique e arraste um Botão.



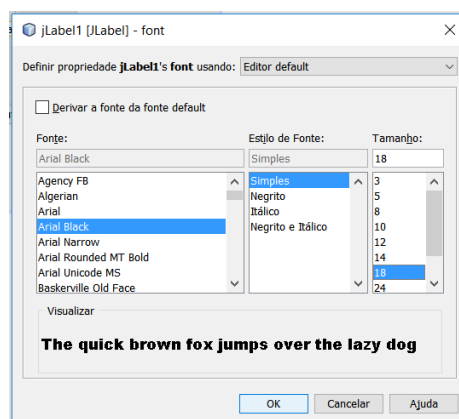
Para alterar o texto, basta dar um clique em jLabel1 ou utilizando a Janela de Propriedades, digite o texto no campo Text.



Aumente o tamanho da fonte utilizando a Paleta, Propriedade, font. Clique em “...”.

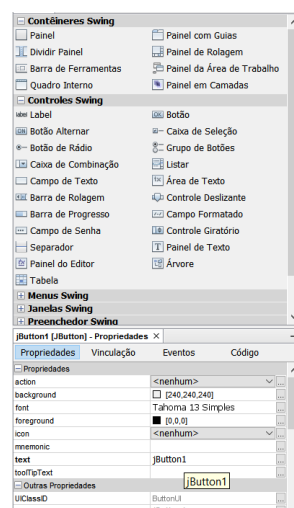


Configure as opções de Fonte e clique no botão OK.



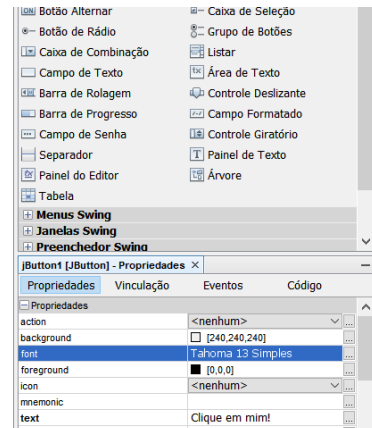
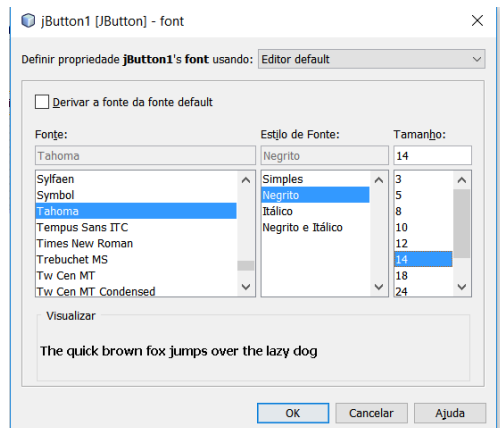
Temos estas mesmas opções no Botão.

Altere o texto do botão clicando sobre o botão ou clicando na caixa text da Paleta - Propriedade.

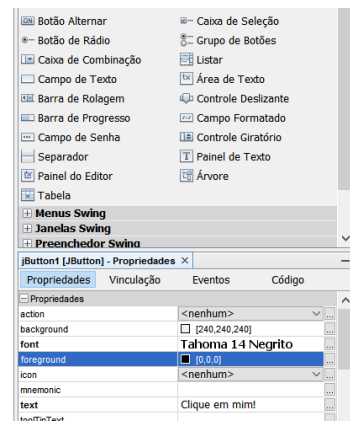
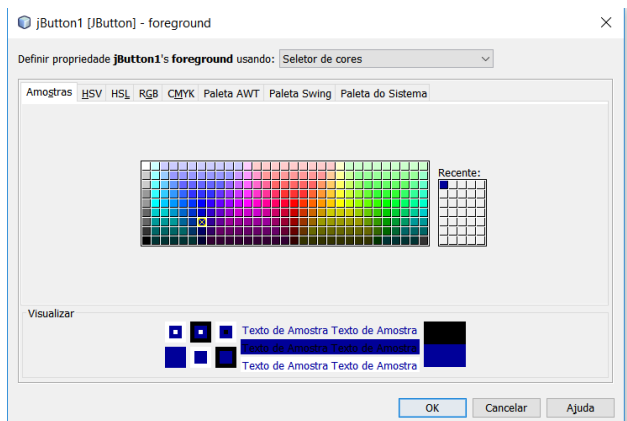


Altere o tamanho da fonte clicando em “font”. Desta vez, experimente clicar diretamente sobre a numeração do tamanho da fonte; a janela de formatação da fonte abrirá normalmente.

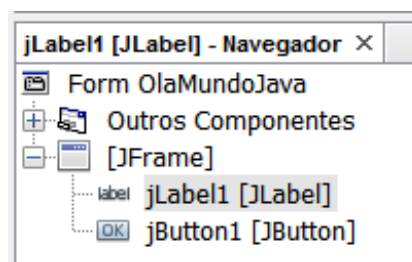
Após escolher as opções de formatação, clique no botão OK.



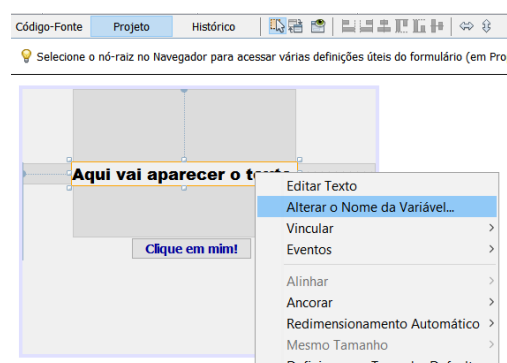
Altere a cor da fonte clicando em “foreground”.



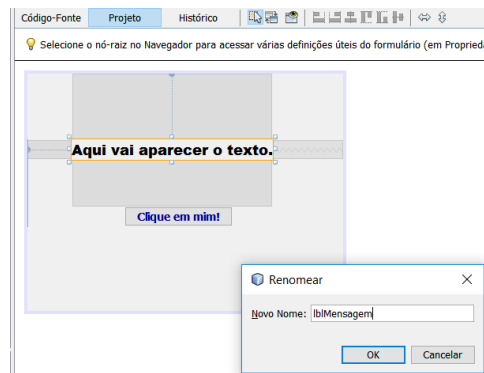
Quando inserimos o Label e o Botão, estes foram exibidos no Painel o lado esquerdo.



Para alterar o nome dos componentes, basta clicar com o botão direito do mouse sobre eles e clicar em Alterar o Nome da Variável, conforme a imagem acima; ou clicando com o botão direito do mouse diretamente sobre o objeto em Projeto, conforme imagem abaixo.

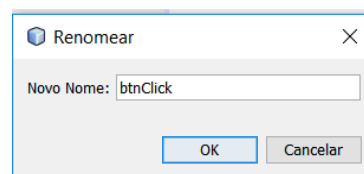


Na Janela Renomear, digite lblMensagem e clique no botão OK.

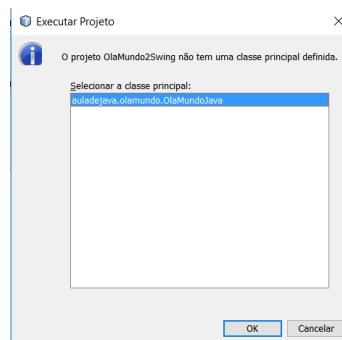


Repita os passos para o botão, nomeando como btnClick.

Note que as primeiras letras são minúsculas por ser o nome de um objeto.



Ao clicar em Executar Projeto, o Netbeans irá exibir uma mensagem questionando qual é a Classe Principal, já que desmarcamos a opção Criar Classe Principal no início do desenvolvimento do software. Isso acontece porque o Netbeans não sabe por onde começar.

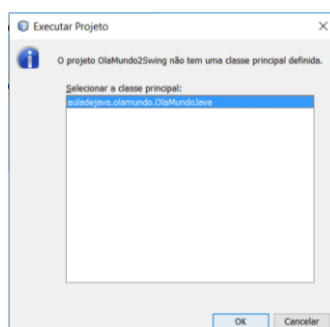


No nosso caso, a janela que acabamos de criar será a Classe Principal.

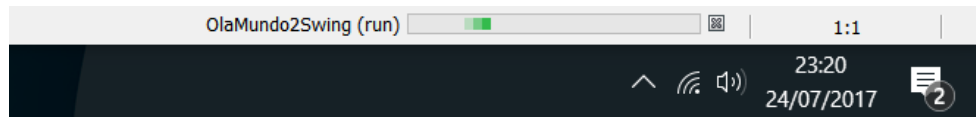
A Classe Principal será auladejava.olamundo que é o nome do Pacote e OlaMundoJava que é o nome da Classe.

Lembrando: Pacote tudo em letras minúsculas (auladejava.olamundo) e Classe com a primeira letra maiúscula, utilizando CamelCase (OlaMundoJava).

Clique no botão OK.



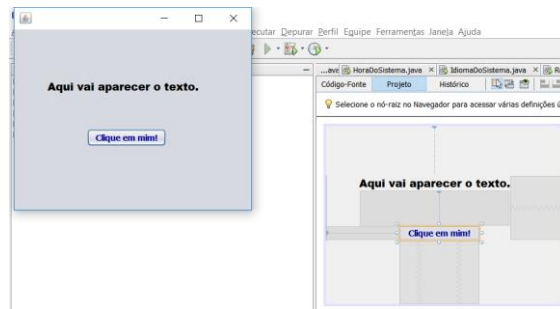
Aguarde o processo terminar.



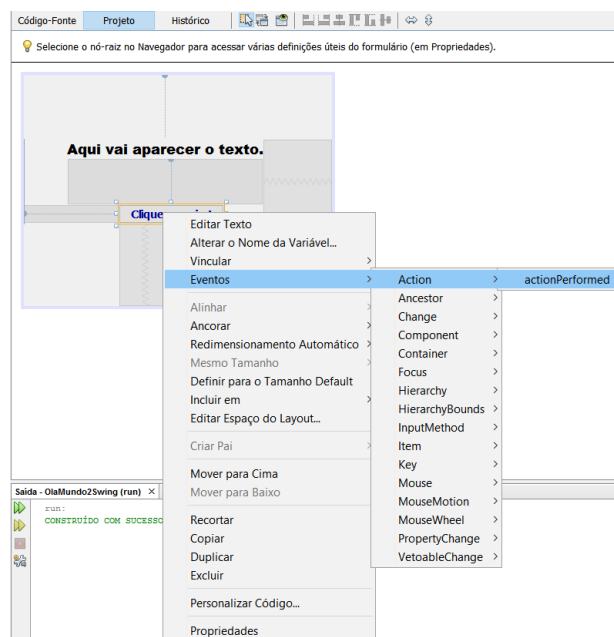
Após terminar o processo, será exibida a janela da aplicação.

Ao clicar no botão da janela criada, nada irá acontecer. Isso porque ainda não há nenhum comando para execução; não foi emitido nenhum comando solicitando que apareça algo na tela.

Feche esta nova janela.



Para programarmos uma ação para este botão, clique com o botão direito do mouse sobre o botão, aponte para Eventos, Action e clique em actionPerformed.



Ao clicar em actionPerformed, seremos automaticamente direcionados para Código-Fonte.

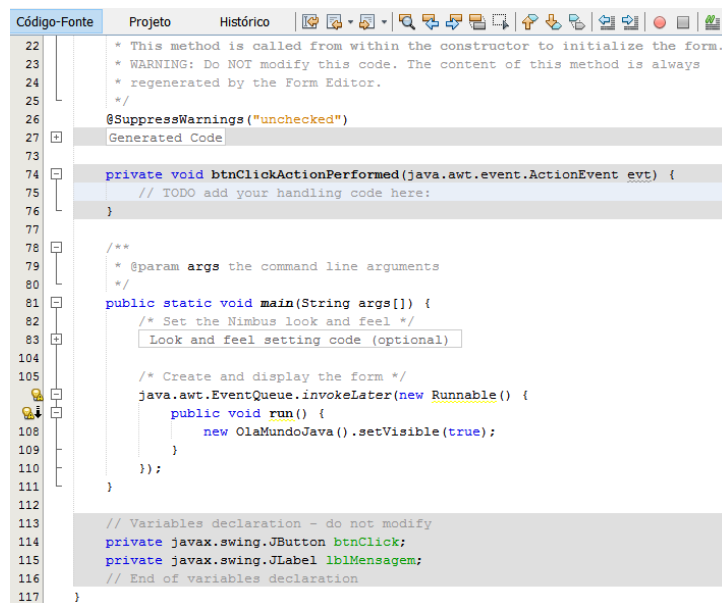


Utilize a barra de rolagem para poder visualizar melhor.

Há vários códigos na tela e outros códigos ocultos, podendo ser exibidos clicando no sinal +. Vários destes códigos foram inseridos para poder criar a janela, porém não precisamos digitar nada.

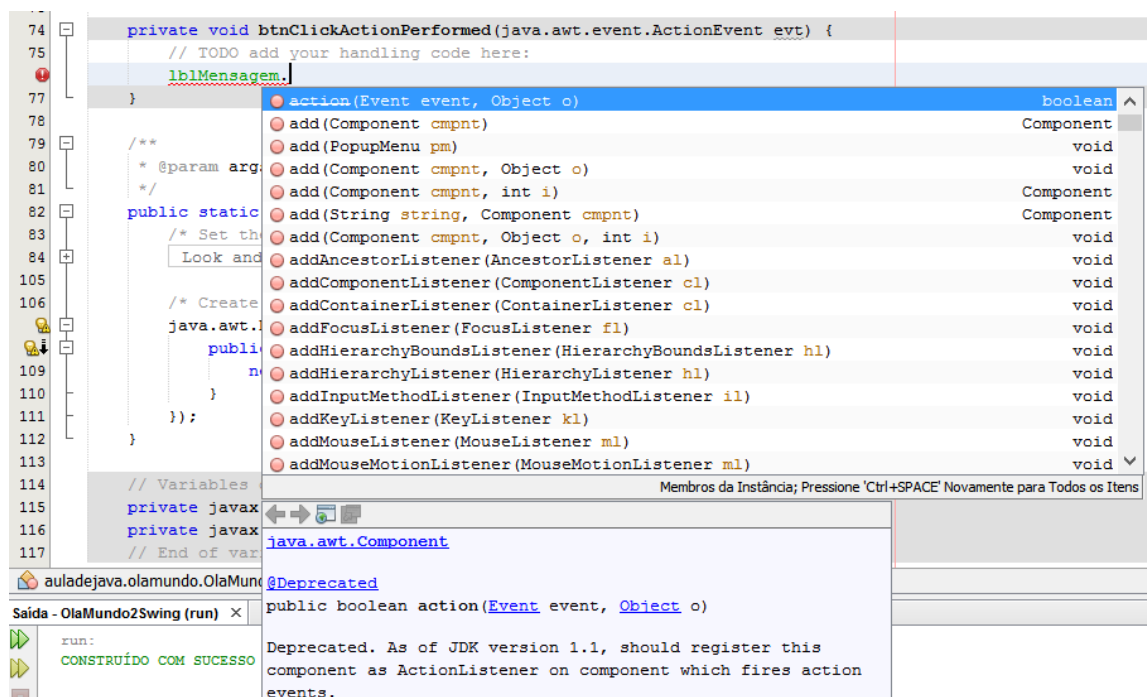
Observe que o cursor do mouse já está posicionado na linha do botão.

Podemos observar que é exibido o nome do botão, btnClick, e ActionPerformed que é um evento. Confirmamos que é um evento em awt.event. Não esqueçam que o Swing evoluiu a partir da AWT.



```
22  * This method is called from within the constructor to initialize the form.
23  * WARNING: Do NOT modify this code. The content of this method is always
24  * regenerated by the Form Editor.
25  */
26  @SuppressWarnings("unchecked")
27  Generated Code
73
74  private void btnClickActionPerformed(java.awt.event.ActionEvent evt) {
75      // TODO add your handling code here:
76  }
77
78  /**
79   * @param args the command line arguments
80   */
81  public static void main(String args[]) {
82      /* Set the Nimbus look and feel */
83      Look and feel setting code (optional)
104
105      /* Create and display the form */
106      java.awt.EventQueue.invokeLater(new Runnable() {
107          public void run() {
108              new OlaMundoJava().setVisible(true);
109          }
110      });
111  }
112
113  // Variables declaration - do not modify
114  private javax.swing.JButton btnClick;
115  private javax.swing.JLabel lblMensagem;
116  // End of variables declaration
117  }
```

Dentro deste campo entre { }, digite o nome do botão “lblMensagem.” e aguarde o menu de ajuda para completar.



```
74  private void btnClickActionPerformed(java.awt.event.ActionEvent evt) {
75      // TODO add your handling code here:
76      lblMensagem.
77  }
78
79  /**
80   * @param args
81   */
82  public static
83      /* Set th
84      Look and
105
106      /* Create
107      java.awt.
108      publi
109      n
110      });
111  }
112
113  // Variables
114  private javax
115  private javax
116  // End of var
117  }
```

action(Event event, Object o) boolean
add(Component cmpnt) Component
add(PopupMenu pm) void
add(Component cmpnt, Object o) void
add(Component cmpnt, int i) Component
add(String string, Component cmpnt) Component
add(Component cmpnt, Object o, int i) void
addAncestorListener(AncestorListener al) void
addComponentListener(ComponentListener cl) void
addContainerListener(ContainerListener cl) void
addFocusListener(FocusListener fl) void
addHierarchyBoundsListener(HierarchyBoundsListener hl) void
addHierarchyListener(HierarchyListener hl) void
addInputMethodListener(InputMethodListener il) void
addKeyListener(KeyListener kl) void
addMouseListener(MouseListener ml) void
addMouseMotionListener(MouseMotionListener ml) void

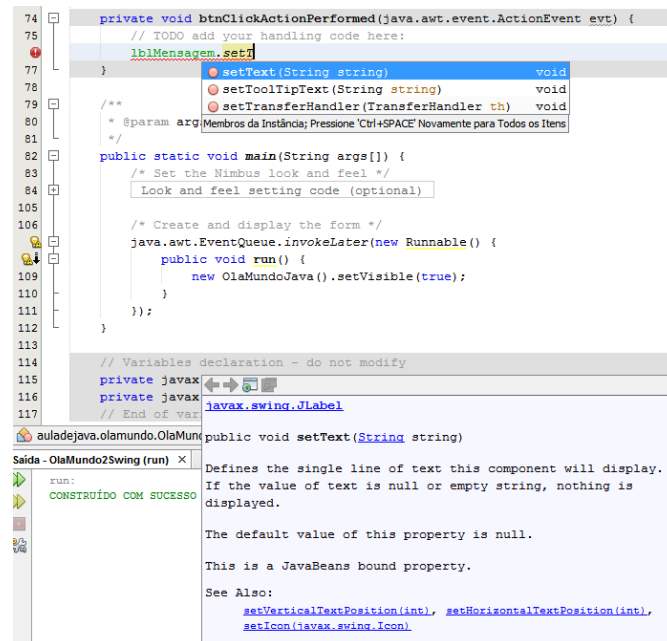
Membros da Instância; Pressione 'Ctrl+SPACE' Novamente para Todos os Itens

java.awt.Component

@Deprecated
public boolean action(Event event, Object o)

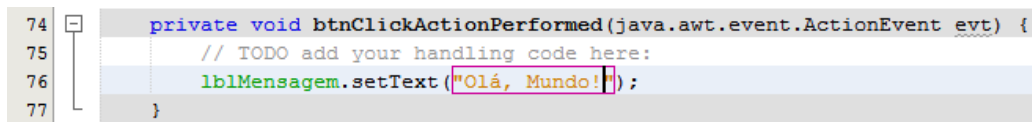
run: CONSTRUÍDO COM SUCESSO

Digite “setT”. Aparecerá setText na ajuda, basta apertar a tecla Enter.

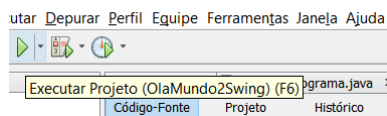


Digite “Olá, Mundo!”.

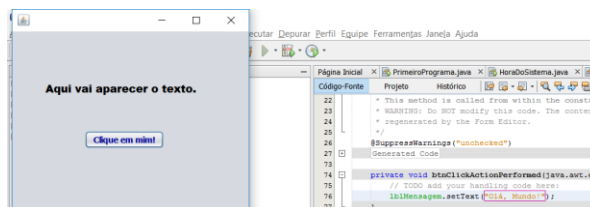
Desta forma, está configurada o texto de lblMensagem que é o nosso Label.



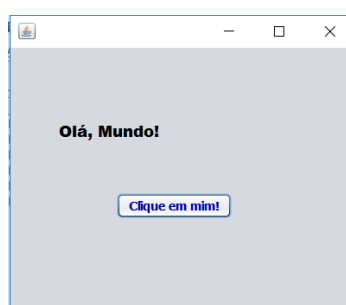
Aperte o botão Executar Programa.



Aparecerá a janela.



Clicando em “Clique em mim!”, o evento será disparado e aparecerá “Olá, Mundo!” na tela.



Agora, com o programa funcional, vamos analisar algumas linhas de código.

No final, onde temos “private.javax.swing.JButton btnClick” e “private.javax.swing.JLabel lblMensagem” são linhas que identificam os controles.

```
114 // Variables declaration - do not modify
115 private javax.swing.JButton btnClick;
116 private javax.swing.JLabel lblMensagem;
117 // End of variables declaration
```

Basicamente, o nosso programa é composto pelo seguinte:

Temos a Classe TelaSwing e aparece uma palavra nova “extends”.

Assim como a Classe, o extends está relacionado ao conceito de Programação Orientada a Objeto.

O conceito de Programação Orientada a Objeto relacionado a extends se chama Herança. Neste caso, a Classe pública TelaSwing tem como Herança java.swing.JFrame.

Assim como em nossa família nós herdamos os bens dos nossos antecessores, tudo que um JFrame tiver vai passar para a nossa Tela. Assim, não precisamos ficar criando telas e criando os comandos para as telas; nós dizemos que essa tela que irá dar Olá, Mundo em Swing irá herdar coisas de JFrame; desta forma, podemos pegar tudo que o JFrame possui e utilizar na nossa tela de Swing.

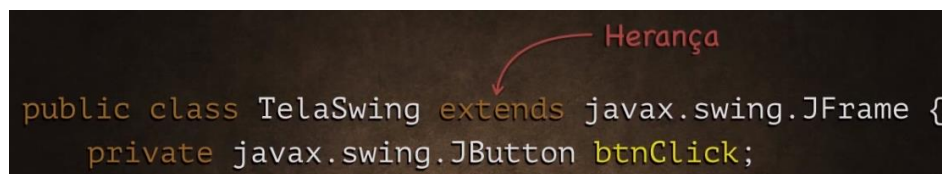
NÃO é necessário recriar, basta herdar as características.



```
public class TelaSwing extends javax.swing.JFrame {
```

Dentro da Classe há as especificações de cada controle, como por exemplo, “private javax.swing.JButton btnClick”.

Podemos perceber que btnClick é o nome de um JButton que é do tipo Swing.

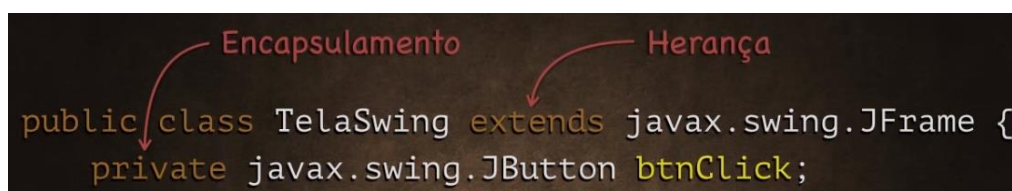


```
public class TelaSwing extends javax.swing.JFrame {
    private javax.swing.JButton btnClick;
```

Observe que temos as nomes “public” e “private”.

“public” quando todos podem ter acesso e “private” quando o acesso é privado aquele objeto.

O nome que se dá para tornar coisas privadas a um objeto é Encapsulamento.



```
public class TelaSwing extends javax.swing.JFrame {
    private javax.swing.JButton btnClick;
```


De forma similar, além do botão temos “private javax.swing.JLabel lblMensagem”.

“lblMensagem” é um “JLabel”.

```
public class TelaSwing extends javax.swing.JFrame {  
    private javax.swing.JButton btnClick;  
    private javax.swing.JLabel lblMensagem;
```

Diagram illustrating the code snippet above. A red arrow labeled "Encapsulamento" points to the private variables `btnClick` and `lblMensagem`. Another red arrow labeled "Herança" points to the `extends javax.swing.JFrame` line.

Temos o código do Método “private void btnClickActionPerformed(...)”.

`btnClick` que é o nome do nosso objeto.

`ActionPerformed` que é ação realizada ou ação performada.

Logo temos uma Classe.

Dentro da Classe temos comandos dizendo que vamos ter um Botão e um Label e vamos ter o código de um Método que será executado ao clicar sobre o botão.

```
public class TelaSwing extends javax.swing.JFrame {  
    private javax.swing.JButton btnClick;  
    private javax.swing.JLabel lblMensagem;  
    private void btnClickActionPerformed(...) {
```

Diagram illustrating the code snippet above. A red arrow labeled "Encapsulamento" points to the private variables `btnClick` and `lblMensagem`. Another red arrow labeled "Herança" points to the `extends javax.swing.JFrame` line.

Sendo assim, “ActionPerformed” é um Evento, que na verdade é um Método em resposta a alguma coisa.

```
public class TelaSwing extends javax.swing.JFrame {  
    private javax.swing.JButton btnClick;  
    private javax.swing.JLabel lblMensagem;  
    private void btnClickActionPerformed(...) {
```

Diagram illustrating the code snippet above. A red arrow labeled "Encapsulamento" points to the private variables `btnClick` and `lblMensagem`. Another red arrow labeled "Herança" points to the `extends javax.swing.JFrame` line. A third red arrow labeled "Evento" points to the `btnClickActionPerformed` method name.

O comando no interior é “`lblMensagem.setText(“Olá, Mundo!”)`”. Onde, “`setText`” é o Método do Objeto, que vai poder modificar o texto que está dentro dele.

Então, repetindo, “`setText`” é um Método.

```
public class TelaSwing extends javax.swing.JFrame {  
    private javax.swing.JButton btnClick;  
    private javax.swing.JLabel lblMensagem;  
    private void btnClickActionPerformed(...) {  
        lblMensagem.setText(“Olá, Mundo!”);  
    }  
}
```

Diagram illustrating the code snippet above. A red arrow labeled "Encapsulamento" points to the private variables `btnClick` and `lblMensagem`. Another red arrow labeled "Herança" points to the `extends javax.swing.JFrame` line. A third red arrow labeled "Evento" points to the `btnClickActionPerformed` method name. A fourth red arrow points to the `setText` method call inside the `btnClickActionPerformed` method.