Programação Visual

Aula 03

Prof. Ederson Cichaczewski

Conversa inicial

Nesta aula, serão abordados conceitos básicos sobre aplicações desenvolvidas em linguagem Java. Conheceremos o NetBeans IDE (ambiente de desenvolvimento integrado) enfatizando a criação de aplicações com interface gráfica em Java. Além disso, abordaremos uma importante biblioteca da linguagem C denominada OpenGL, utilizada para programação de elementos gráficos. Veremos exemplos de aplicações em linguagem C utilizando-se desta biblioteca para a criação de animações gráficas.

Antes de iniciar, assista ao vídeo do professor Ederson em que ele fala o que mais vamos estudar. Acompanhe na rota!

Contextualizando

A linguagem Java foi desenvolvida pela Sun Microsystems por James Gosling. A Oracle adquiriu a Sun Microsystems no ano de 2009, sendo assim, detentora da tecnologia Java e demais produtos Sun Microsystems.

Entre as aplicações da linguagem Java temos o desenvolvimento de interfaces gráficas, utilizando como bibliotecas a AWT e a Swing. Podemos contar com a vantagem da portabilidade da API do Java para desenvolver aplicações que vão rodar em diversos sistemas operacionais, como Windows e Linux, desde que tenha instalado o framework Java, independente da resolução da tela ou da profundidade de cores.

Indo para a área voltada para jogos, temos a biblioteca OpenGL, utilizada em programação com linguagem C. Sua vantagem é que utiliza recursos diretos da placa de vídeo (GPU) como, por exemplo, aceleração de vídeo por hardware, principalmente para gráficos 3D. A última versão da biblioteca OpenGL é a 4.5, lançada em 2014.

Veremos, nesta aula, exemplos de interface gráfica com Java Swing e exemplos de animação em linguagem C com OpenGL. Saiba mais sobre esse assunto assistindo ao vídeo do professor Ederson no material virtual.

Pesquise

Tema 01: Conceitos da Linguagem Java

A tecnologia Java é uma linguagem de programação e também uma plataforma.

Linguagem de Programação Java

A programação Java é uma linguagem de alto nível que possui as seguintes características:

- Simples
- Orientada à objeto
- Distribuída
- Multitarefa
- Dinâmica
- Arquitetura neutra
- Portátil
- Desempenho elevado
- Robusta
- Segura

Na linguagem de programação Java, todo o código fonte é escrito em arquivo texto com a extensão **.java**. As linhas deste código-fonte são então compiladas e um arquivo com extensão **.class** é criado pelo **compilador Java**, denominado **javac**. O arquivo **.class** não contém código que é nativo ao seu processador; contém preferivelmente *bytecodes* – a linguagem de máquina da **máquina virtual Java** ou simplesmente *Java VM (Virtual Machine)*. A ferramenta de inicialização denominada **java** executa então sua aplicação com uma instância da máquina virtual Java ou como vimos anteriormente, *JVM*. Veja na Figura 1 a seguir.

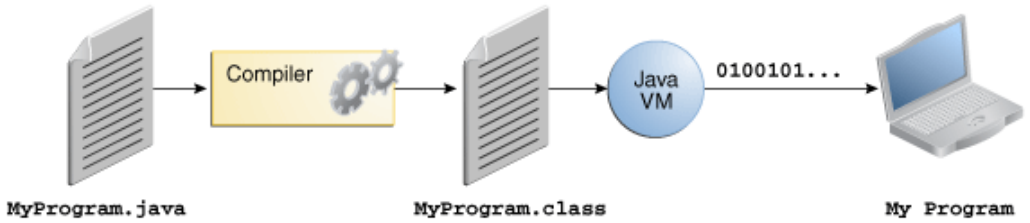

Figura 1 – Uma visão geral do processo de desenvolvimento do software.

Pelo fato de a máquina virtual Java ou JVM estar disponível em muitos sistemas operacionais diferentes, o mesmos arquivos **.class** são capazes de rodar no sistema operacional Microsoft Windows, Solaris™, Linux, ou Mac. Veja na Figura 2.

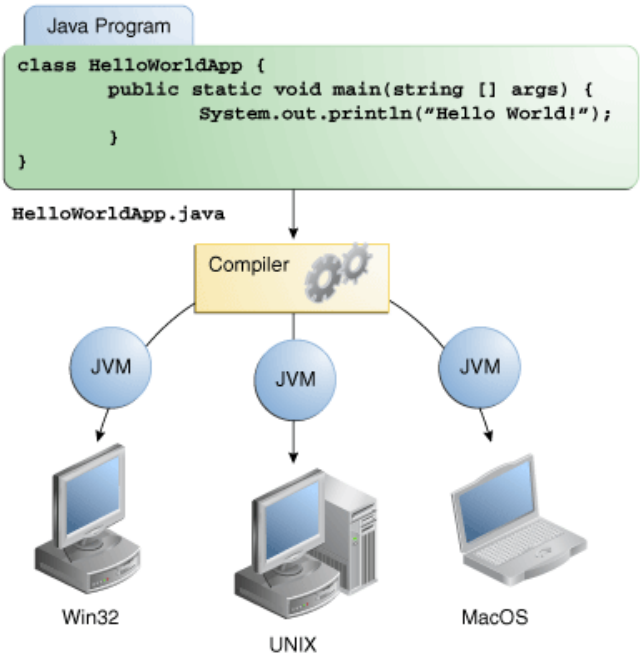

Figura 2 – Através da máquina virtual Java ou JVM, a mesma aplicação é capaz de rodar em múltiplas plataformas.

Demais considerações sobre a linguagem Java:

- Para rodar programas baseados na linguagem Java, basta ter instalado na máquina o pacote ***JRE: Java Runtime Environment*** ou mais conhecido por usuários não desenvolvedores, simplesmente por **Java**. O pacote **JRE** contém as bibliotecas (**APIs**) e a máquina virtual

Java (**JVM**) para permitir a execução de aplicações Java.

- Para compilar programas utilizando a linguagem Java, é necessário ter instalado na máquina o ***JDK: Java SE Development Kit*** de acordo com o sistema operacional em questão. Observação: a instalação **JDK** já contempla o **JRE**, portanto não é necessária a instalação deste.
- Arquivos .java são os arquivos de código fonte.
- Arquivos .class são os arquivos compilados, os *bytecodes*.
- Na linguagem Java, um código intermediário é gerado (arquivo .class), contendo o *bytecode*, a ser interpretado e executado pela máquina virtual Java (**JVM**).
- A vantagem deste arquivo com *bytecode* é o alto grau de portabilidade dos programas.

A plataforma Java

A plataforma é o ambiente de *hardware* ou de *software* em que um programa funciona. Mencionamos já algumas das plataformas mais populares como Microsoft Windows, Linux, Solaris e Mac. A maioria das plataformas podem ser descritas como a combinação do sistema operacional e o *hardware* em que rodam. A plataforma Java difere da maioria pelo fato de desagregar o *hardware* de si, ou seja, trata-se de uma plataforma de *software* que roda em cima de outras plataformas baseadas em *hardware*.

A plataforma Java possui dois componentes:

- Máquina virtual Java ou JVM.
- Interface de Programação de Aplicação Java ou Java *API* (*Java Application Programming Interface).*

A máquina virtual Java é a base para a plataforma Java e é portável em várias plataformas baseadas em *hardware*.

A API Java contém uma grande coleção de componentes de software

prontos para uso que fornecem muitas potencialidades úteis. Esses componentes são agrupados dentro de bibliotecas de classes e interfaces relacionadas; essas bibliotecas são conhecidas como **pacotes** ou termo usual em inglês *packages*. Veja a Figura 3.

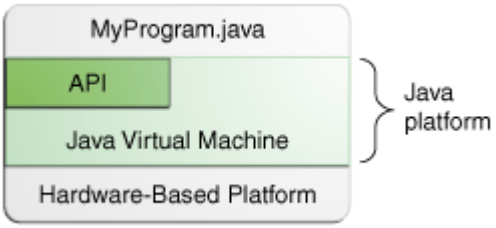

Figura 3 – A API e a máquina virtual Java separam o programa do *hardware* em si.

Como um ambiente independente de plataforma, a plataforma Java pode ser um código mais lento do que um código nativo. Entretanto, os avanços no compilador e tecnologias de máquina virtual estão trazendo desempenho próximo ao de código nativo sem ameaçar a portabilidade.

Saiba mais: acesse os sites a seguir e aprenda mais sobre a linguagem Java.

- The Java Tutorials
https://docs.oracle.com/javase/tutorial/index.html
- The Java Language Environment: Contents
http://www.oracle.com/technetwork/java/langenv-140151.html

Assista ao vídeo do professor Ederson em que ele fala mais sobre o conceito da linguagem Java. Acompanhe no seu material.

Tema 02: Java IDE e Interface Visual

Para demonstração dos componentes visuais e aplicações práticas em linguagem Java, utilizaremos o ambiente de desenvolvimento integrado denominado NetBeans.

O NetBeans IDE é gratuito e de código aberto para desenvolvedores de *software* nas linguagens Java, C, C++, PHP, Ruby, entre outras. O IDE é

executado em muitas plataformas, como Windows, Linux, Solaris e MacOS.

Sabemos que, além desta IDE, há várias outras opções para utilizarmos, tais como Eclipse, IntelliJ, JDeveloper etc., porém, a escolha do NetBeans IDE baseou-se na facilidade para a criação de aplicações baseadas em interface gráfica.

Antes de demonstrarmos o NetBeans IDE, vale uma pequena explanação sobre a construção, em linguagem Java, de interfaces gráficas para programas *desktop*.

GUI

G*raphical User Interface* – GUI, que em português significa "interface gráfica com usuário". Apresenta um mecanismo amigável ao usuário para interagir com um aplicativo. Fornece a um aplicativo uma "aparência" e "comportamento" distintos. Um aplicativo com interface gráfica onde há componentes consistentes e intuitivos que dá aos usuários um sentido de familiaridade. Além disso, o usuário pode interagir via mouse, teclado ou outra forma de entrada como, por exemplo, reconhecimento de voz.

Swing e AWT

Em Java, há duas bibliotecas principais que podemos utilizar para a construção de interfaces gráficas, são elas: AWT (*Abstract Window Toolkit*) e Swing.

AWT utiliza recursos do próprio sistema operacional para criar os componentes gráficos. Sendo assim, a sua aplicação pode exibir comportamentos diferentes caso necessite rodar em diferentes ambientes. No caso da API Swing, a maiorias dos componentes são escritos, manipulados e exibidos completamente em Java diminuindo, assim, o problema da portabilidade. Além de ser mais atual que AWT, os componentes GUI Swing permitem especificar uma aparência e comportamento uniforme para o aplicativo em todas as plataformas ou utilizar a aparência e comportamento personalizados de cada plataforma.

NetBeans IDE

Para utilizarmos a ferramenta NetBeans, o pré-requisito principal de software, conforme consta no próprio site da ferramenta, é a instalação do JDK (Java SE Development Kit), versão 6 ou superior. A versão utilizada para demonstração corresponde à versão 8.1.

A Figura 4 apresenta a tela inicial do NetBeans.

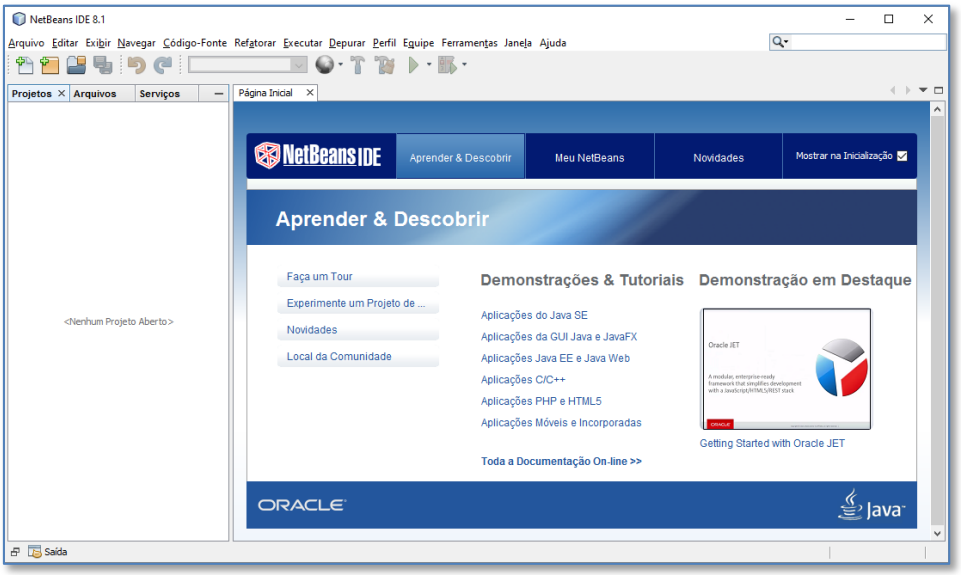

Figura 4 – Tela inicial do NetBeans.

Há duas maneiras de abrir um novo projeto. Via menu de opções ou atalho na barra de ferramentas.

Para criar um novo projeto via menu, seguir os seguintes passos:

- Clicar no menu Arquivo
- Clicar no item Novo Projeto

Para criar um novo projeto via atalho, na barra de ferramentas basta clicar no atalho, conforme demonstrado na Figura 5 a seguir.

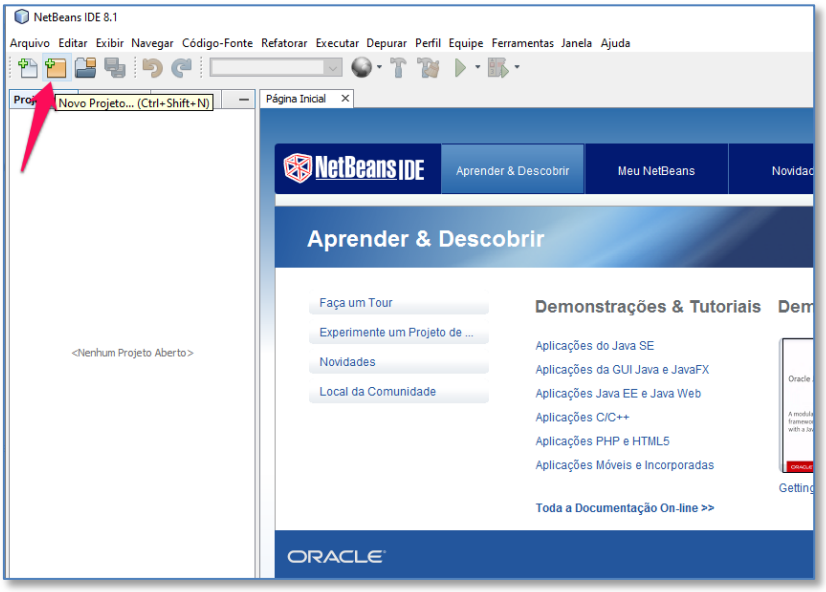

Figura 5 – Atalho para criação de um novo projeto.

Na sequência, é apresentada a tela com as opções de projeto para as diferentes linguagens de programação, conforme apresentado na Figura 6.

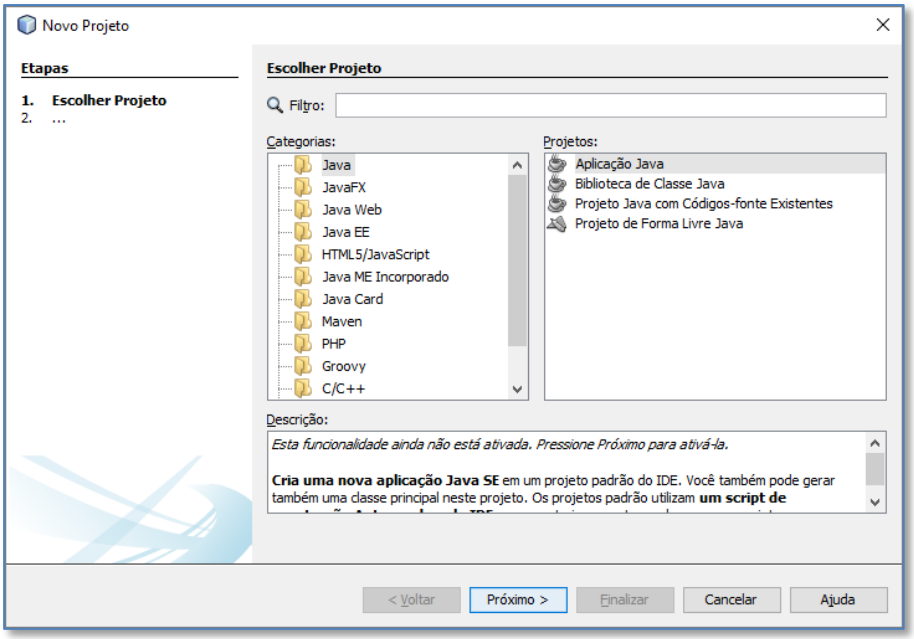

Figura 6 – Tela de opções para seleção de tipo de linguagem de programação.

Ao clicar em uma opção de linguagem de programação como, por exemplo, Java, na região localizada à esquerda e denominada “Categorias”, são listadas na região direita denominada “Projetos” as opções de diferentes tipos de projeto para a linguagem de programação escolhida. Na região inferior, é apresentada uma descrição de cada uma dessas opções ao clicar nelas. Nesta tela, é possível também inserir no campo “Filtro” o nome da linguagem ou tipo de projeto para que a ferramenta sugira as opções de forma a filtrar os resultados.

Para este exemplo, vamos escolher a Categoria Java e projeto Aplicação Java. Veja na Figura 7.

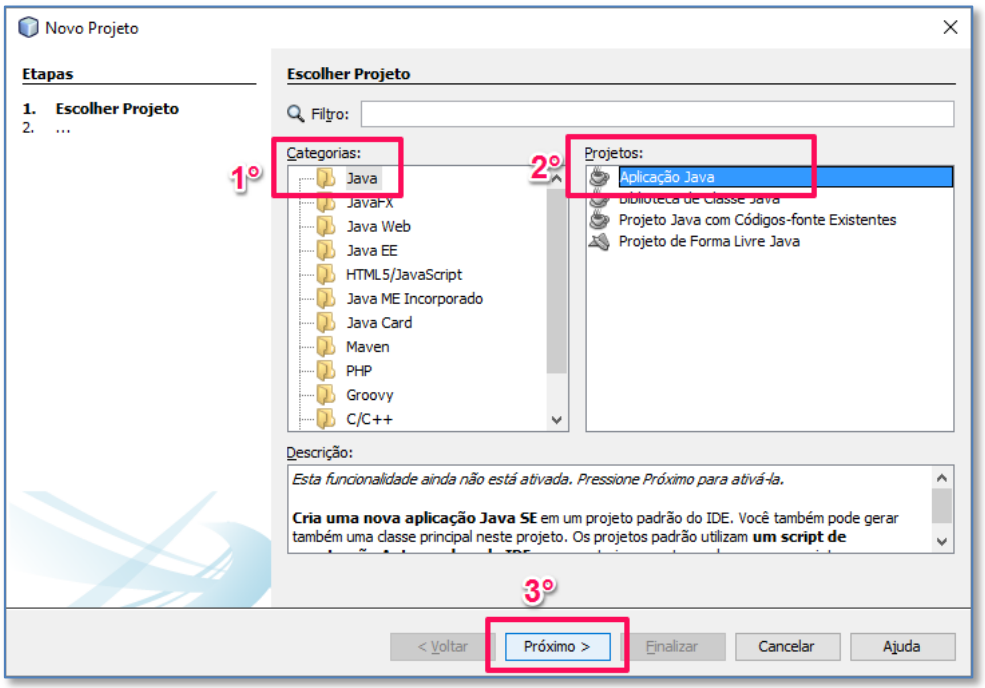

Figura 7 – Criação de um novo projeto.

Ao clicar no botão “Próximo”, será exibida a janela para inserirmos o nome do projeto, bem como as demais opções.

- Vamos colocar o nome do projeto como “JavaComponentes”;
- Desmarque a opção “Criar Classe Principal”;
- Clique em Finalizar.

A Figura 8 demonstra a janela que denota o nome do projeto e localização de gravação.

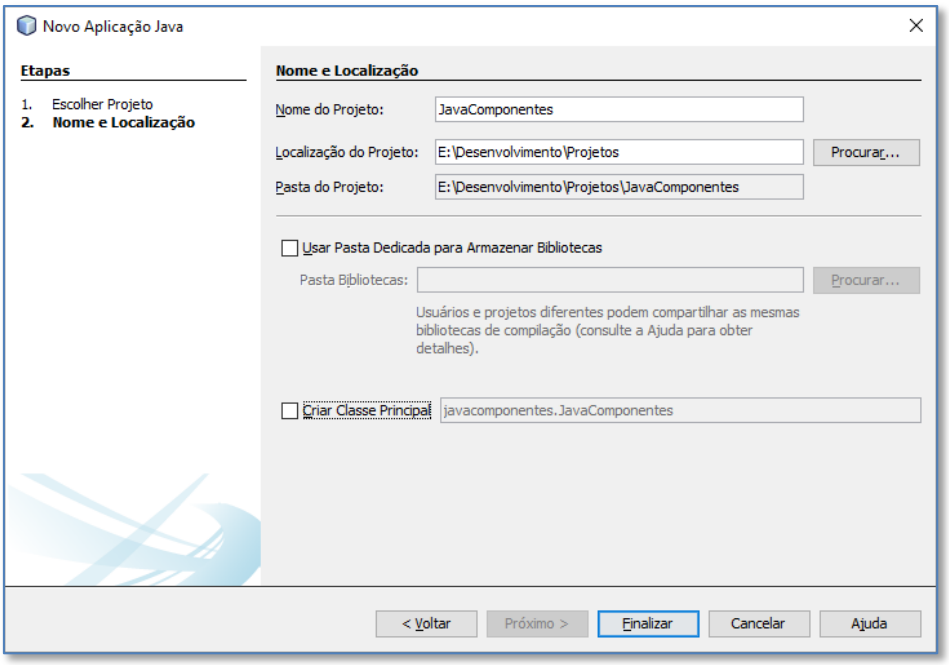

Figura 8 – Nome do projeto e localização de gravação.

Ao clicar em “Finalizar”, conforme observado na Figura 9, será exibido no canto esquerdo, no guia “Projetos”, o nome do projeto e subpastas: “Pacotes de Código-Fonte” e “Bibliotecas”. Confira a seguir.

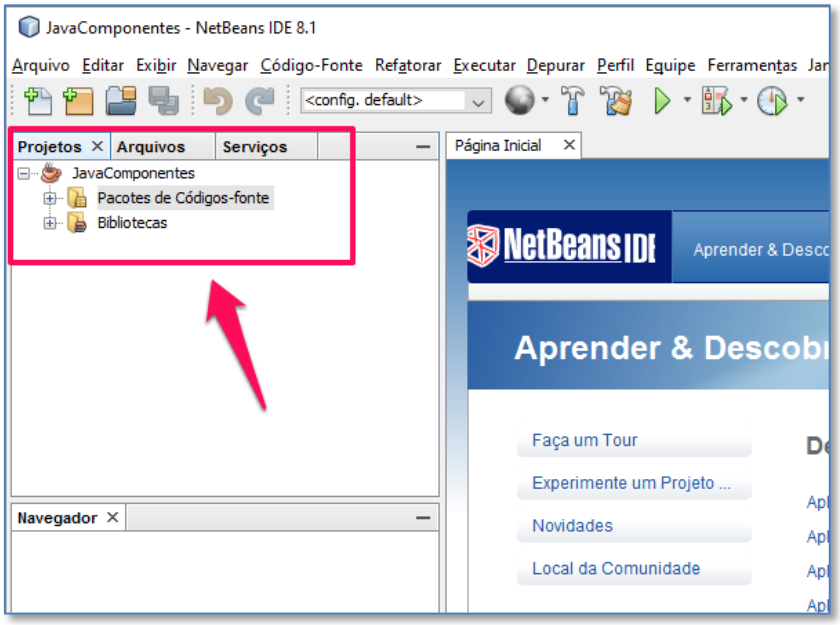

Figura 9 – Informações do projeto na aba “Projetos”.

Agora, vamos criar um pacote Java.

Para isso, clique na subpasta “Pacotes de Códigos-fonte” com o botão direito e selecione a opção “Novo > Pacote Java”.

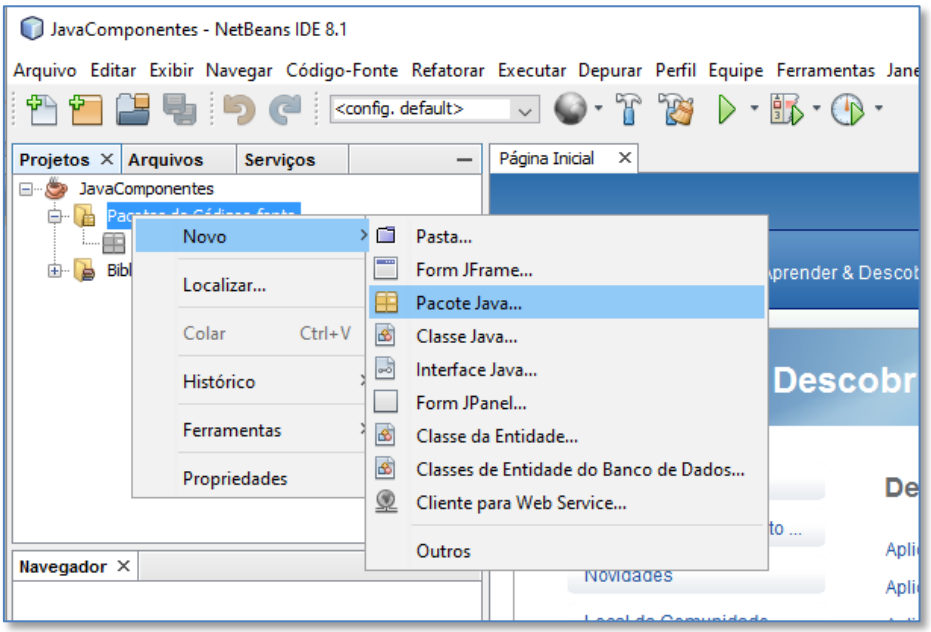

Figura 10 – Criação de pacote Java.

Na janela que será exibida, insira o nome “view” (em minúsculo) no campo “Nome do Pacote”. Depois, clique em “Finalizar”. Veja a Figura 11.

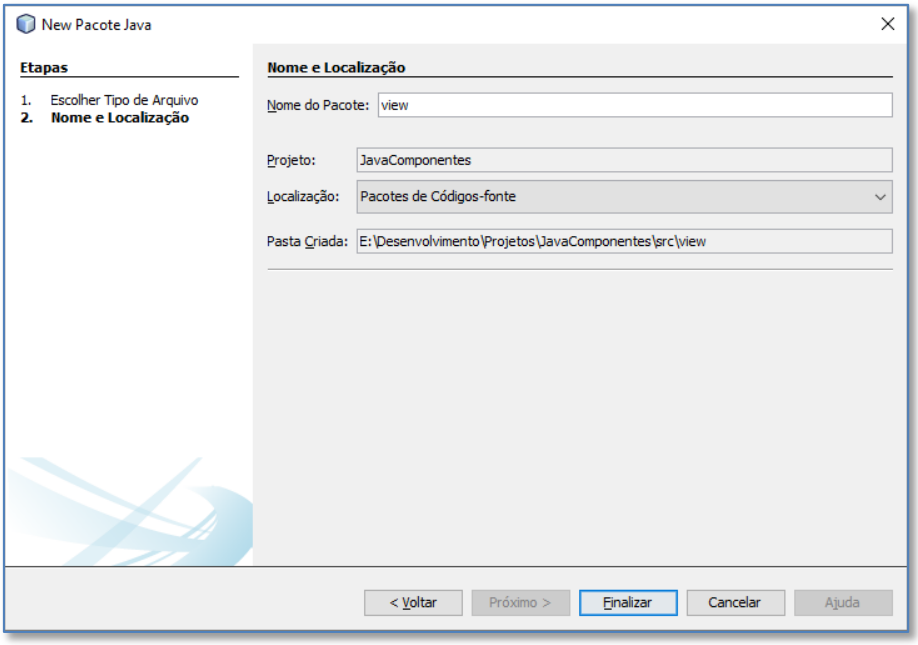

Figura 11 – Novo Pacote nomeado “view”.

Após criar o novo pacote denominado “view”, selecioná-lo na guia esquerda conforme mostra a Figura 12. Vamos, agora, criar um novo componente para inserir no projeto.

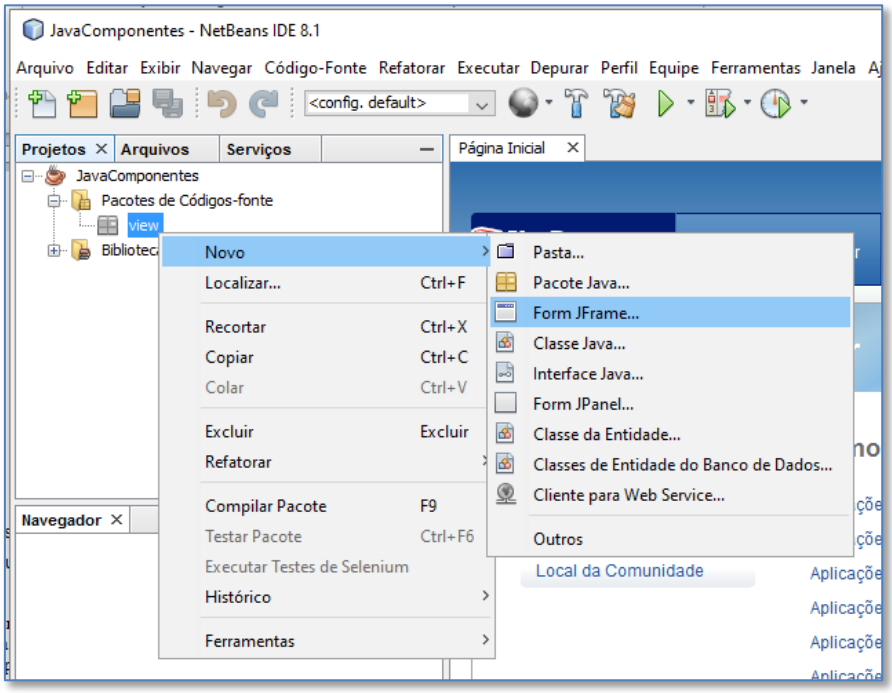

Figura 12 – Criação de componente Form JFrame.

Clicar com o botão direito no pacote “view” e selecionar menu “Novo > Form JFrame”. O nome do novo componente “Form JFrame” compreende uma janela de nível superior autônoma como a interface principal do usuário para uma aplicação. A maioria das aplicações Swing são construídas iniciando a partir deste componente. Após clicar na opção “Form JFrame”, será exibida uma janela para nomear a classe a ser criada. Então, daremos o nome da classe, conforme veremos na Figura 13, como “FormPrincipal”.

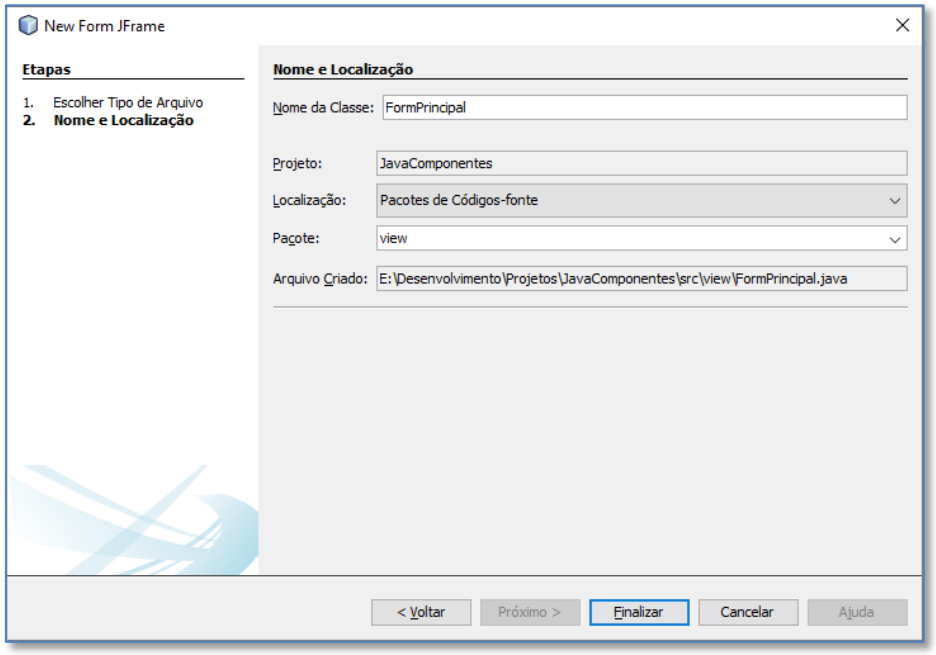

Figura 13 – Criação de nova classe, FormPrincipal.

Após clicar em finalizar, será exibido na IDE as informações a seguir:

- **Painel superior esquerdo:** nome da classe, aba “Projetos”.
- **Painel central:** exibirá a classe no modo “Projeto”, ou seja, exibirá a interface visual.
- **Painel superior direito:** exibirá na aba “Paleta” os componentes para adição no formulário.
- **Painel inferior direito:** exibirá na aba “Nome do componente – Propriedades” as propriedades do componente selecionado onde “Nome do Componente” corresponde ao componente em questão.

Para melhor entendimento, visualize a Figura 14 a seguir.

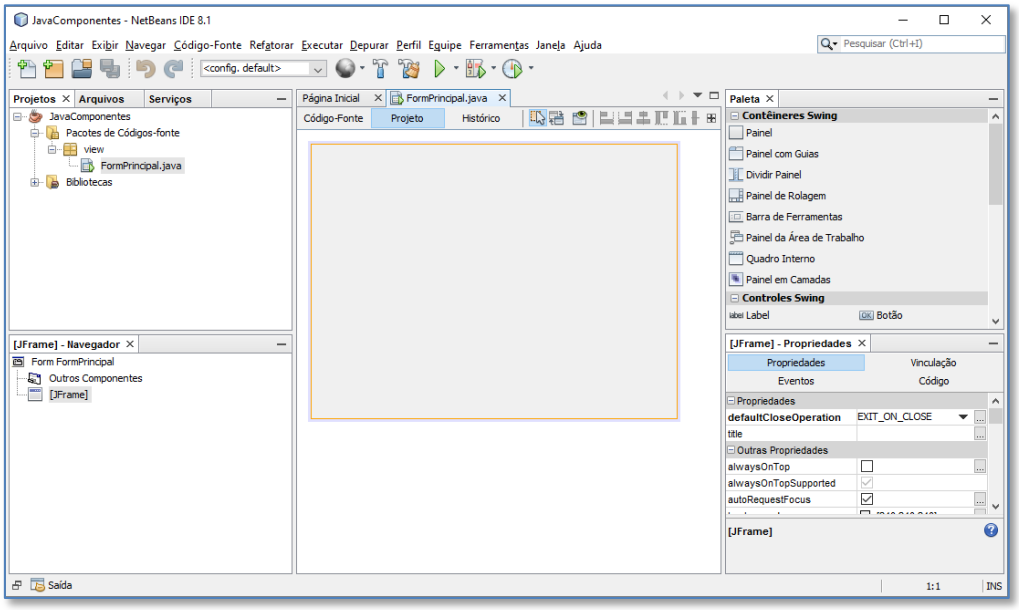

Figura 14 – Informações do Projeto.

O NetBeans conta com uma opção muito prática que pode ser utilizada para visualização prévia de como ficará a interface gráfica quando o projeto for executado. Para isso, basta clicar na opção “Visualizar Design” como mostra a Figura 15.

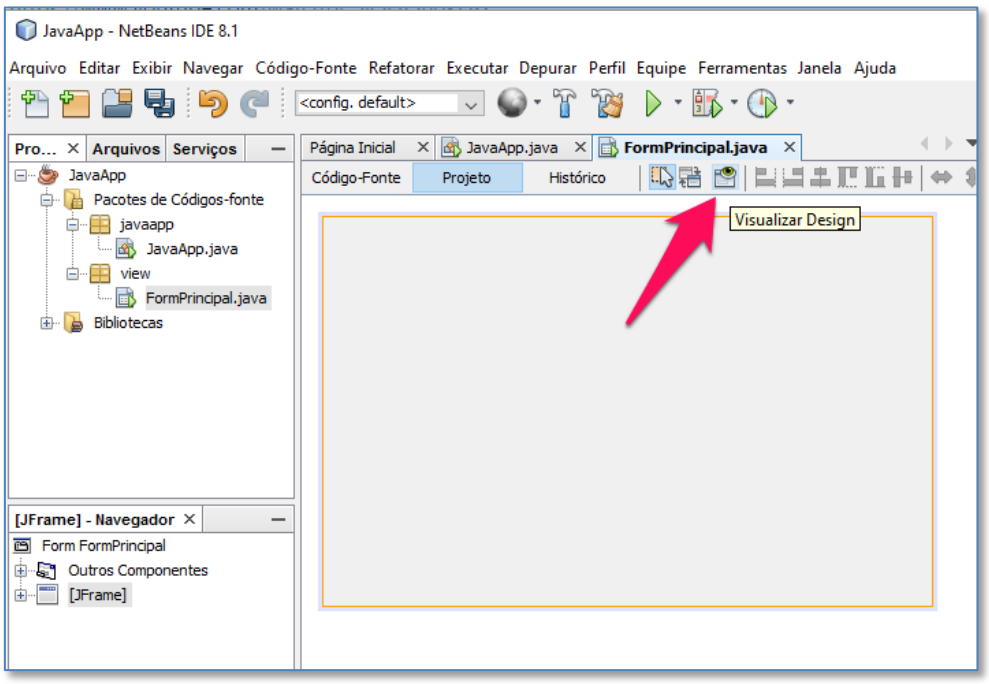

Figura 15 – Opção “Visualizar Design”.

Após clicar na opção “Visualizar Design”, o NetBeans apresenta a interface gráfica do projeto como é possível visualizar na Figura 16.

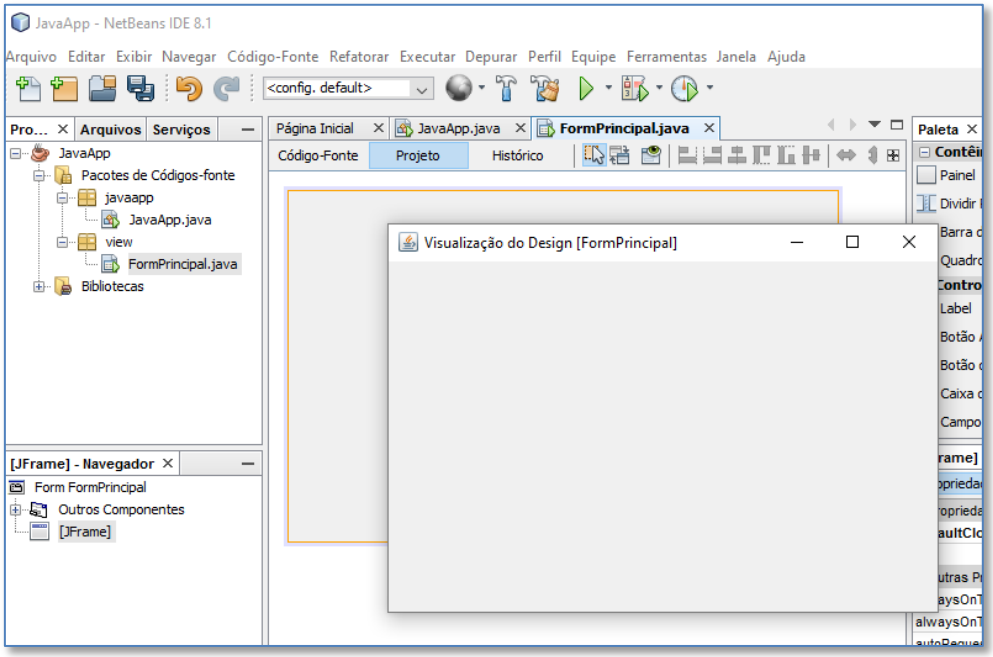

Figura 16 – Prévia de exibição de aplicação Java Swing.

Quando criamos um componente Form JFrame, vimos que foi necessário criar um nome para uma classe Java a ser criada. Foi necessário realizar a criação dessa classe denominada, nesse exemplo, como “FormPrincipal”, pois é ela que armazenará as informações deste componente.

Mas como essa classe soube renderizar corretamente o componente na tela para a inserção de outros componentes ou efetuar ações neste mesmo componente?

O que ocorreu aqui foi uma herança de classe já contida no pacote Swing. Como foi selecionado o componente Form JFrame, a ferramenta NetBeans realizou, de forma automática, a herança a partir da classe **JFrame** contida no pacote ***javax.swing***.

Para comprovar essa informação, podemos mudar a visualização de exibição da classe clicando no item “Código-Fonte” ao lado do item já selecionado “Projeto”. Assim, a classe será exibida em seu código-fonte e não

mais na interface visual. Para voltar à interface visual, basta clicar novamente no item “Projeto”. A Figura 15 denota o código fonte da classe “FormPrincipal.java” e a herança que possibilita a correta renderização do componente JFrame.

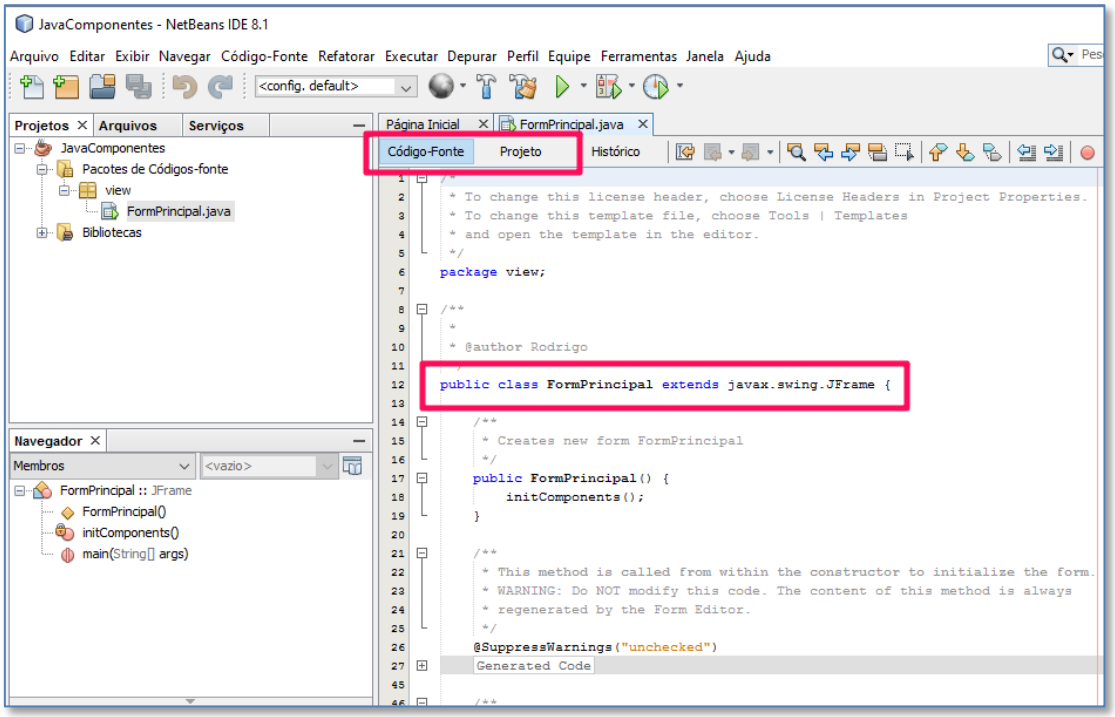

Figura 17 – Código fonte, classe “FormPrincipal.java” e herança, classe “JFrame” da biblioteca Swing.

Observação: quando no modo “Código-fonte”, o painel lateral direito contendo a paleta de componentes e propriedades não é exibido, a exibição está condicionada ao modo de visualização “Projeto”.

Clicar novamente no item “Projeto” para que seja possível visualizar o painel com os componentes. Note no painel superior direito que há vários componentes disponíveis para utilização. Para visualizar melhor esses componentes, minimizar a guia “Propriedades” como mostra a Figura 18.

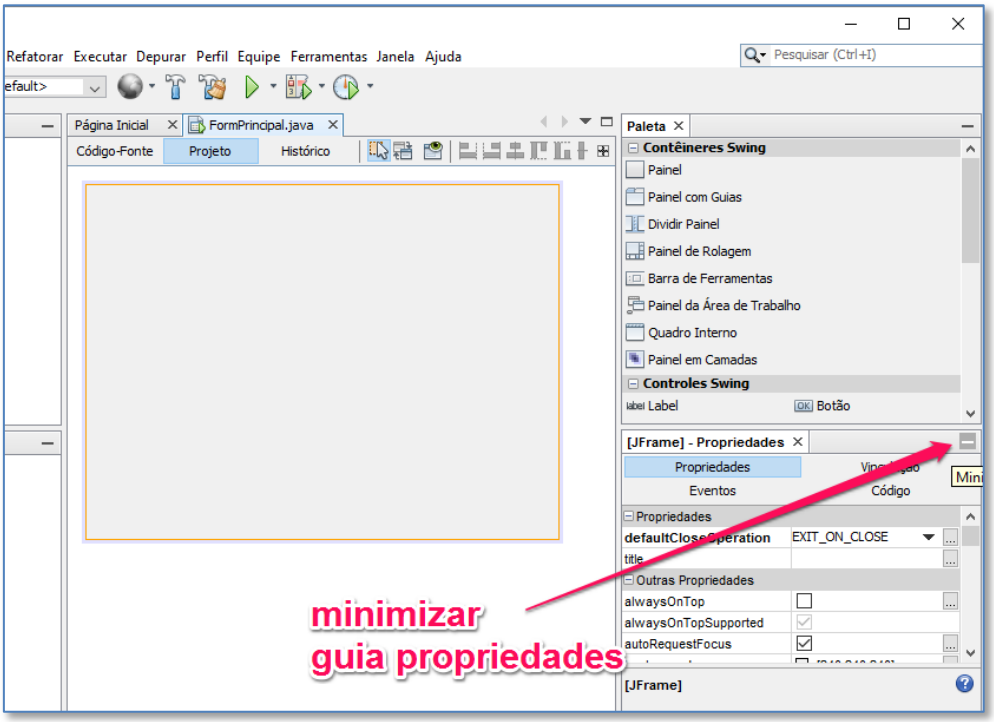

Figura 18 – Minimizar guia “Propriedades”.

Ao minimizar a guia “Propriedades”, será possível visualizar uma série de componentes divididos por categorias. As principais categorias são: “Contêineres Swing” e “Controles Swing”.

A ideia é de que os componentes na categoria “Contêineres Swing” sejam utilizados para armazenar os componentes de controle localizados na categoria “Controles Swing”. Por exemplo, podemos inserir um controle do tipo “Label” e “Campo de Texto” dentro de um contêiner do tipo “Painel” ou até mesmo dentro de um “Painel com Guias”.

A Figura 19 demonstra as categorias citadas e componentes disponíveis para utilização.

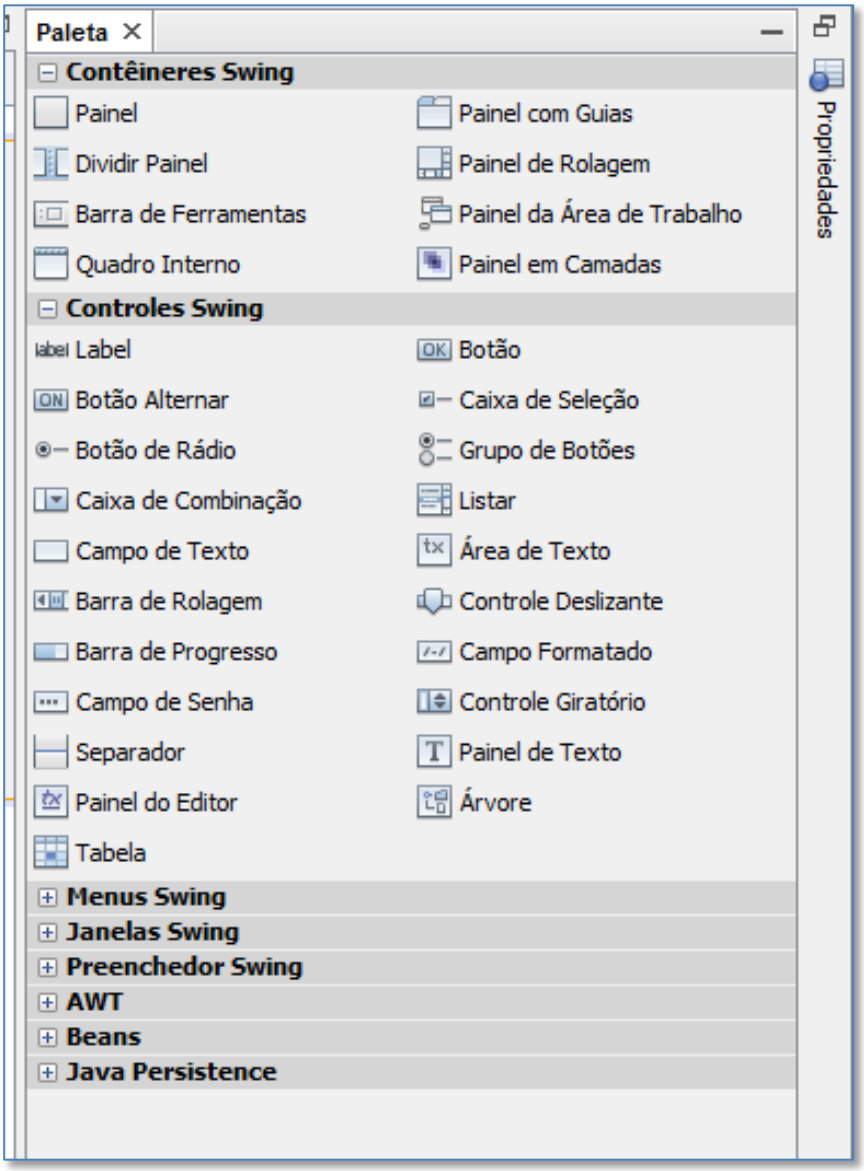

Figura 19 – Paleta de componentes disponíveis para utilização.

Vamos abordar a seguir alguns dos principais componentes da API Swing.

1. **JFrame (Janelas Swing > Quadro): r**epresenta a janela principal da aplicação, tal como vemos em diversas aplicações *desktop*. Normalmente, contém o título da aplicação no topo da janela, botões de comando para minimizar e fechar a aplicação.
2. **JPanel (Painel):** tipo básico de container para inserção de componentes.

3. **JLabel (Label):** representa um rótulo de texto.
4. **JTextField (Campo de Texto):** representa um campo de texto onde o usuário pode informar um texto em uma linha.
5. **JPasswordField (Campo de Senha):** representa um campo de texto protegido, subclasse de JTextField.
6. **JTextArea (Área de Texto):** representa uma caixa onde o usuário pode informar várias linhas de texto.
7. **JCheckBox (Caixa de Seleção):** representa uma caixa de seleção e permite selecionar ou não uma opção.
8. **JRadioButton (Botão de Seleção Única):** representa um componente que permite seleciona uma entre diversas opções.
9. **JComboBox (Caixa de Combinação):** representa uma caixa de combinação, da qual o usuário pode selecionar uma opção.
10. **JList (Lista de Opções):** representa uma lista de opções que permite a seleção de mais de um item simultaneamente.
11. **JButton (Botão de Ação):** representa um botão destinado a executar uma ação.

A Figura 20 apresenta exemplos de componentes em um form (JFrame).

Figura 20 – Componentes Java Swing.

Saiba Mais: acesse os sites a seguir para você aprender mais sobre o que estudamos neste tema.

- Instalação Java SE Development Kit (JDK)
  http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- NetBeans IDE
  https://netbeans.org
- About the JFC and Swing
  https://docs.oracle.com/javase/tutorial/uiswing/start/about.html
- Oracle Technology Network for Java Developers
  http://www.oracle.com/technetwork/java/index.html

Agora, o professor Ederson vai falar mais sobre Java IDE e Interface Visual. Acompanhe no vídeo!

Tema 03: Projetos com Java

Vamos agora utilizar os recursos da biblioteca Swing, recentemente vistos no tópico anterior, para a criação de uma aplicação Java com interface gráfica.

Com o NetBeans aberto, clique no atalho para a criação de um novo projeto. A Figura 19 demonstra a janela para a seleção de Categoria e Projeto.

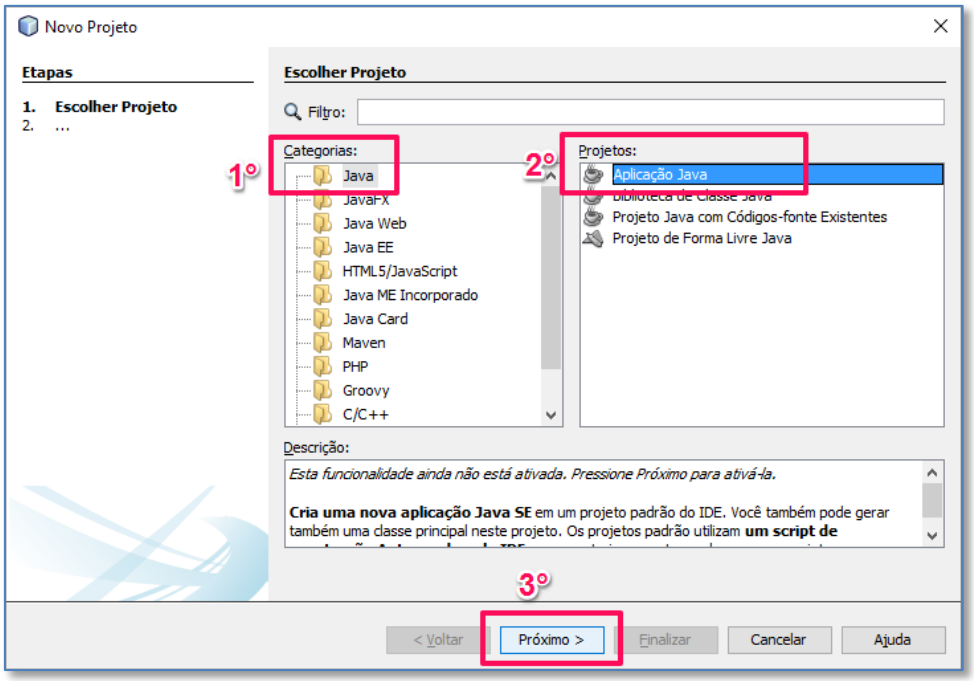

Figura 21 – Criação de um novo projeto.

Ao clicar no botão “Próximo”, será exibido janela para inserirmos o nome do projeto, bem como as demais opções, conforme a Figura 22.

- Vamos colocar o nome do projeto como “JavaApp”;
- **Não** marque a opção “Criar Classe Principal”;
- Clique em Finalizar.

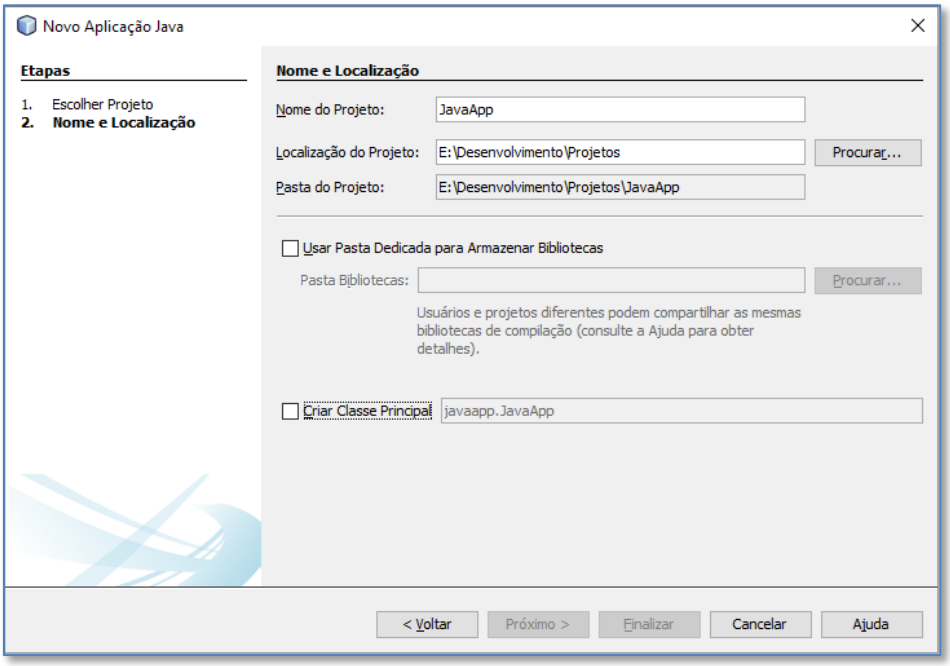

Figura 22 – Nome do projeto e localização de gravação.

Agora, vamos criar nossas classes que herdam as classes da biblioteca Swing. Porém, antes, vamos manter um padrão para a separação das classes e, para isso, criaremos pacotes.

Clique na subpasta “Pacotes de Códigos-fonte” com o botão direito e selecione a opção “Novo > Pacote Java”.

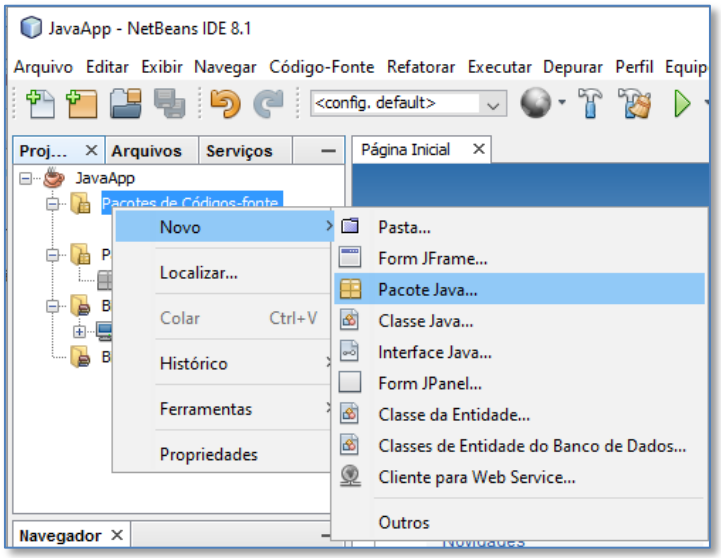

Figura 23 – Criação de pacote Java.

Na janela que será exibida, insira o nome “view” (em minúsculo) no campo “Nome do Pacote”. Clique em “Finalizar”. Veja a Figura 24.

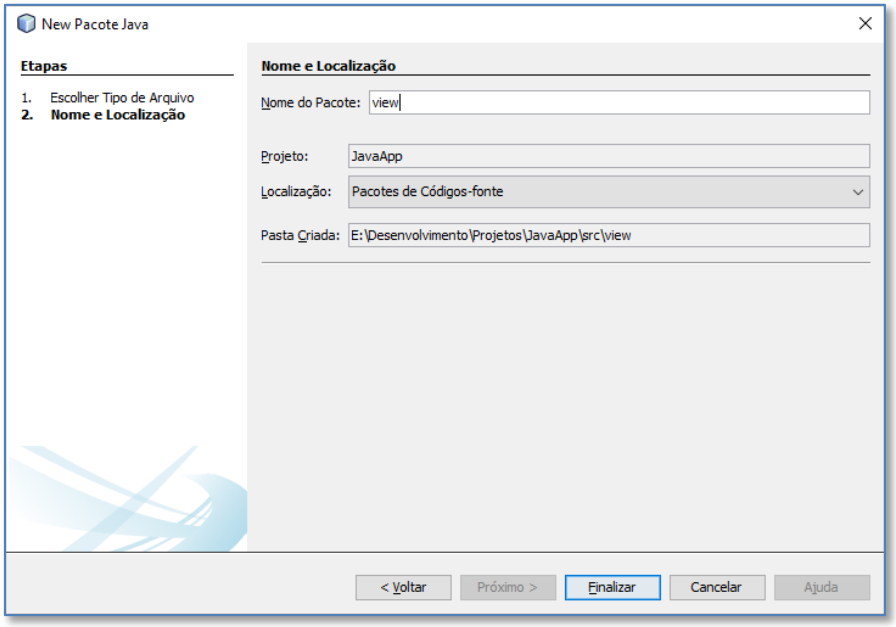

Figura 24 – Novo Pacote nomeado “view”.

Após criar o novo pacote denominado “view”, selecioná-lo na guia esquerda para a criação de um novo componente para inserir no projeto. Selecionar opção “Novo > Form JFrame”. Ao selecionar esta opção, será exibida na tela a janela como mostra a Figura 25.

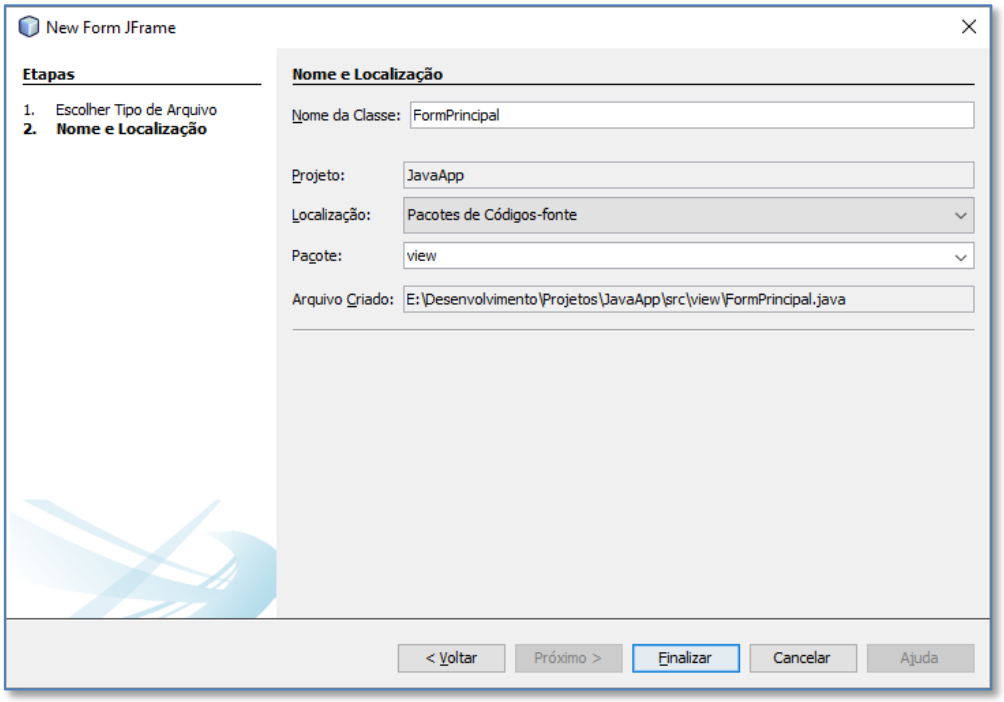

Figura 25 – Novo componente “JFrame”.

Insira o nome de classe como “FormPrincipal” e clique no botão “Finalizar”. Veremos, então, o componente “JFrame” no centro da tela conforme Figura 26.

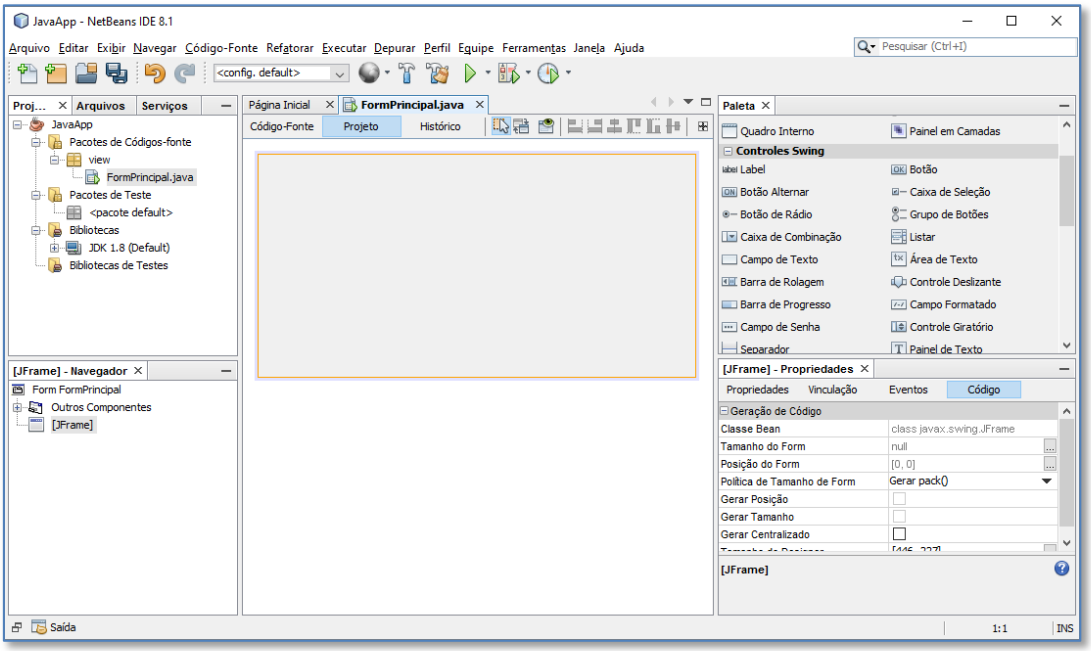

Figura 26 – Componente “JFrame” adicionado.

Neste ponto, realizaremos a inserção de novos componentes no componente principal “JFrame”.

Insira um componente do tipo “JPanel” ou “Painel”. Para esta ação, basta clicar no componente “Painel” na paleta de componentes. Após clicar neste componente, clique em qualquer área do nosso “FormPrincipal (JFrame)”. O NetBeans realizará a inserção do componente “JPanel” conforme podemos visualizar na Figura 27.

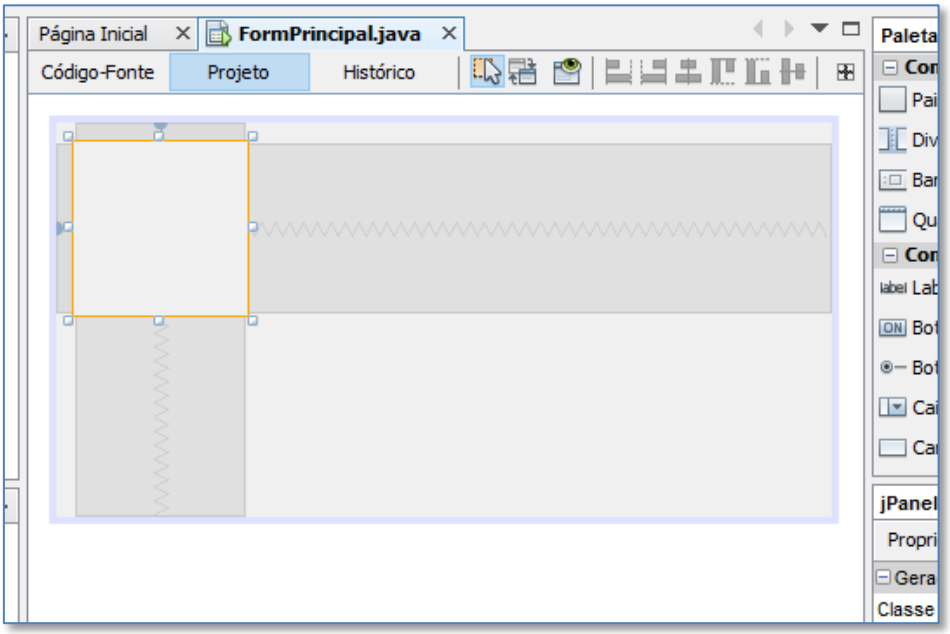

Figura 27 – Componente JPanel.

Redimensione o componente “JPanel” para que fique conforme Figura 28.

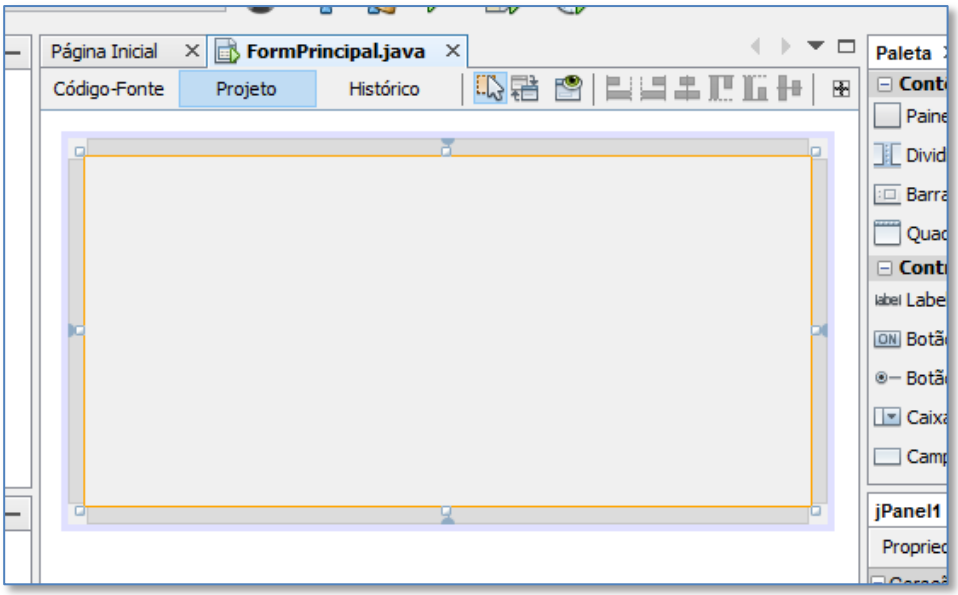

Figura 28 – Componente “JPanel” redimensionado.

Para inserir efeito de borda no “Painel”, basta clicar no componente e selecionar, na janela de propriedades, a propriedade “border”.

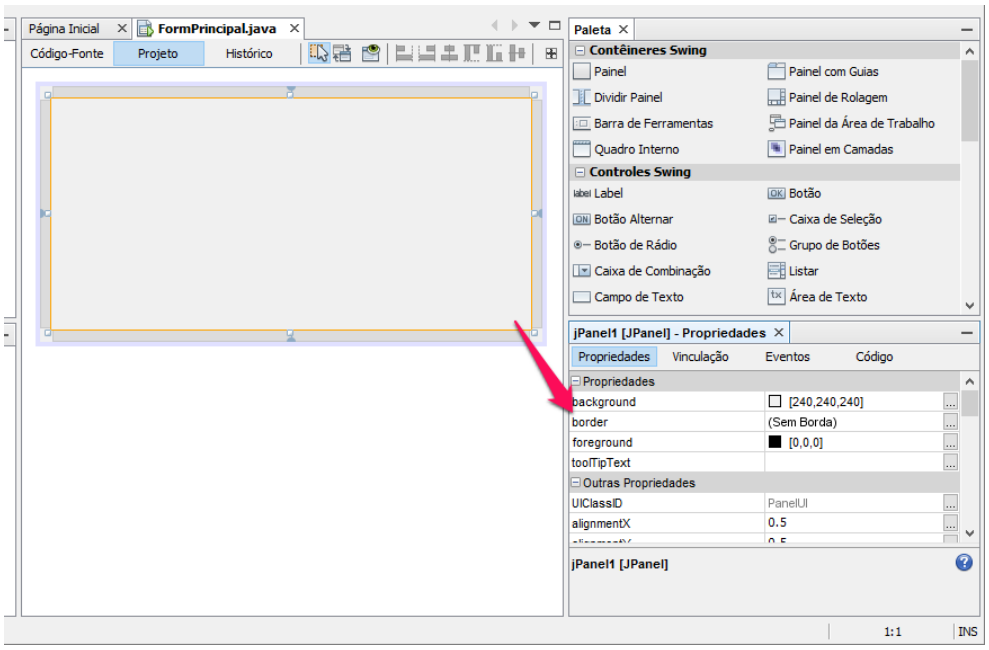

Figura 29 – Painel e propriedade “border”.

Clique na propriedade e, após exibição de janela, selecione a opção “borda gravada”.

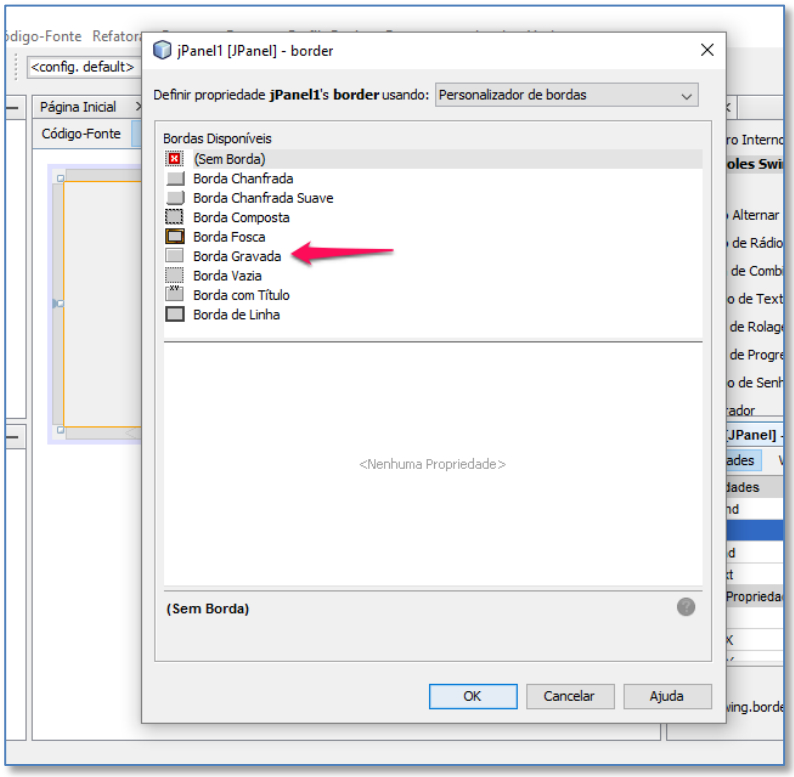

Figura 30 – Seleção de opção “Borda gravada”.

O componente “Painel” será exibido, agora conforme ilustrado na Figura 31.

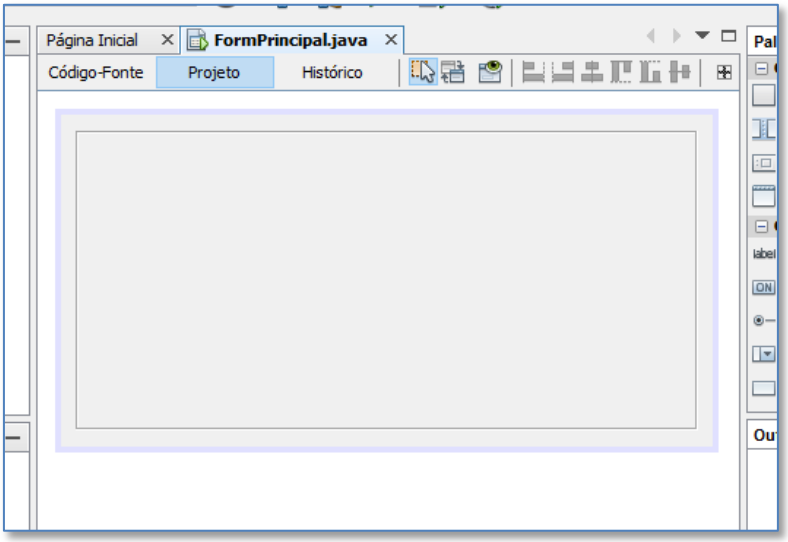

Figura 31 – Painel com opção de borda “Borda gravada”.

Inserir demais componentes no Painel de modo que seja exibido conforme Figura 32.

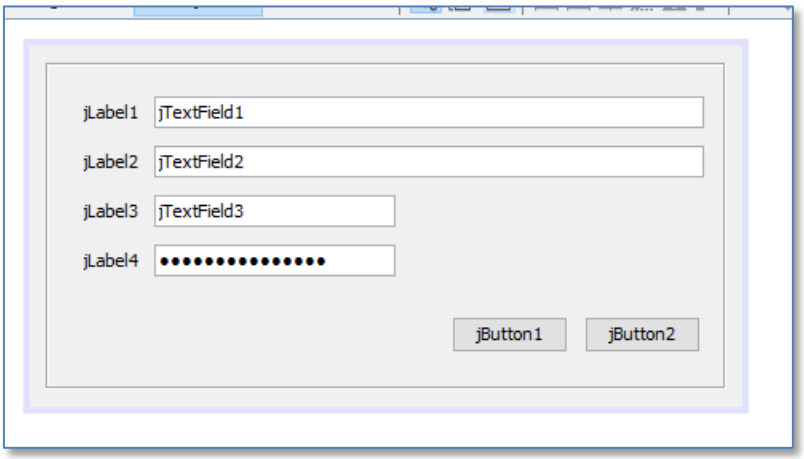

Figura 32 – Formulário Exemplo.

Neste formulário, foram inseridos componentes “JLabel” (Label), “JTextField” (Campo de Texto), “JPasswordField” (Campo de Senha) e “JButton” (Botão).

Note que os textos dos componentes, quando inseridos, são exibidos

com o texto igual ao nome da variável. Por exemplo, para o componente adicionado “jButton1”, o texto corresponde ao mesmo nome da variável, a qual o componente será identificado na programação. E como realizamos a alteração do texto? Para alterar o texto dos componentes, basta clicar no componente e localizar a propriedade “text” na janela de propriedades como mostra a Figura 33.

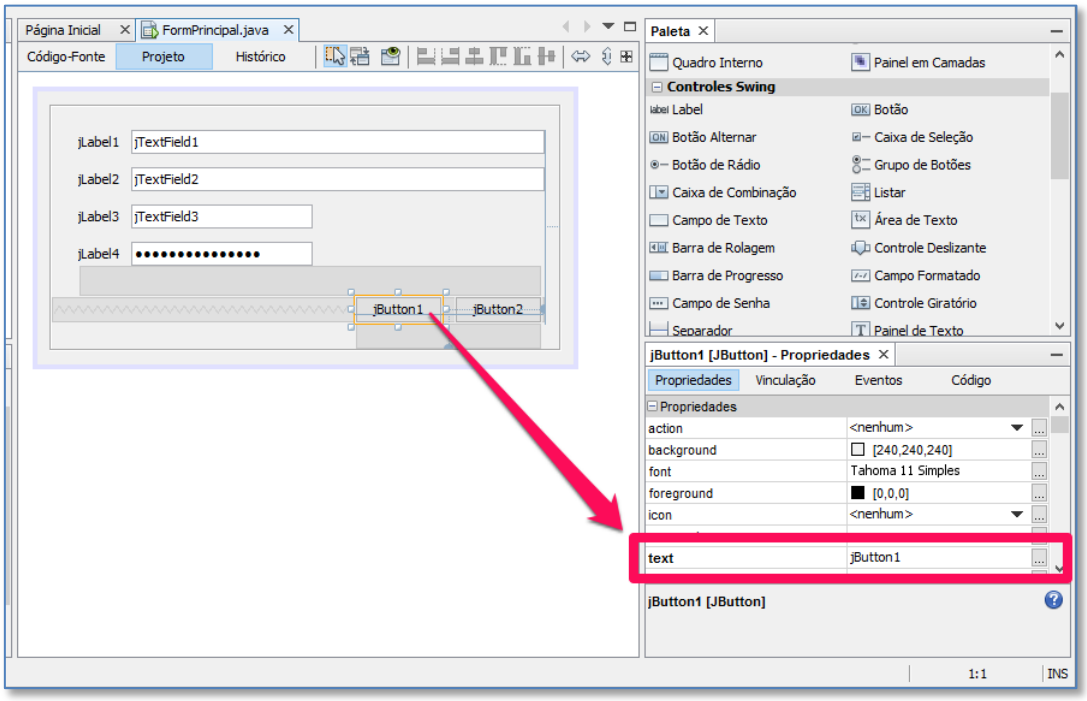

Figura 33 – Alteração de texto de componentes.

A imagem a seguir, Figura 34, demonstra o formulário já com todos os textos dos componentes alterados.

Observação: para os componentes de inserção de valores, mantenha a propriedade “text” vazia.

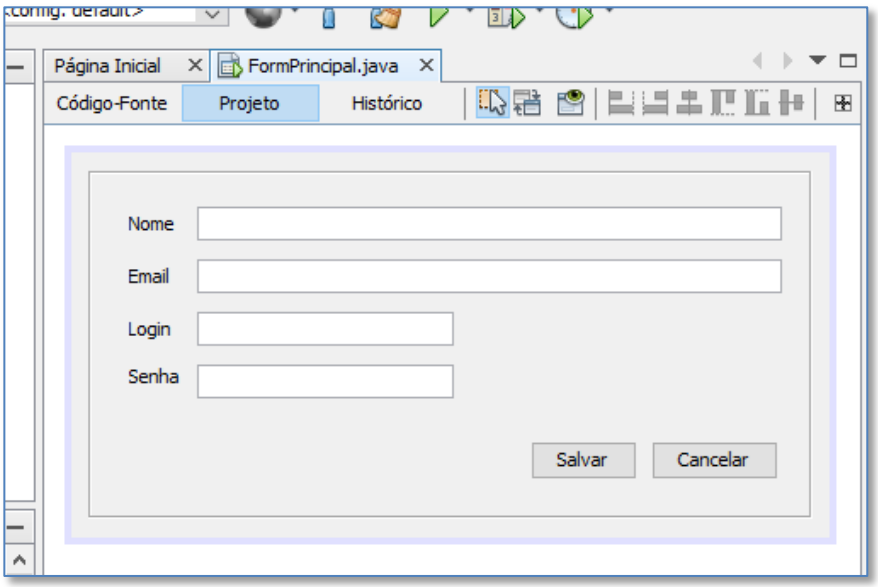

Figura 34 – Formulário com textos de componentes alterados.

Com os componentes já criados, faz-se necessário alterar o nome dos componentes que utilizaremos para recuperação e também o nome dos botões. Para alterar o nome do componente, ou “nome da variável”, basta clicar no nome do componente e, na janela de propriedades, clicar na guia “Código”.

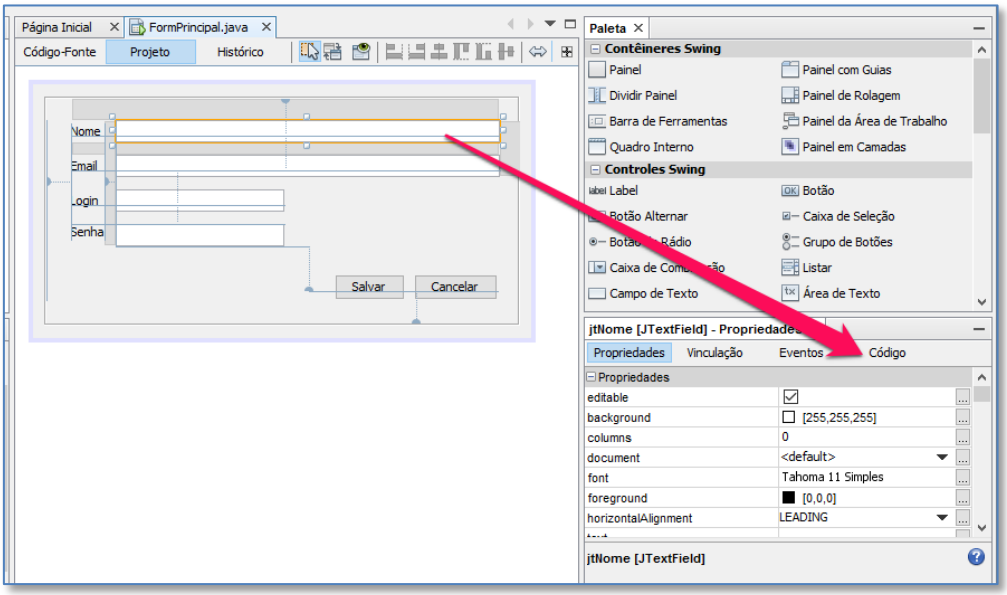

Figura 35 – Alteração de nome dos componentes.

A escolha do nome do componente deve condizer com o propósito do componente. Neste nosso exemplo, realizaremos a alteração dos nomes como no exemplo a seguir.

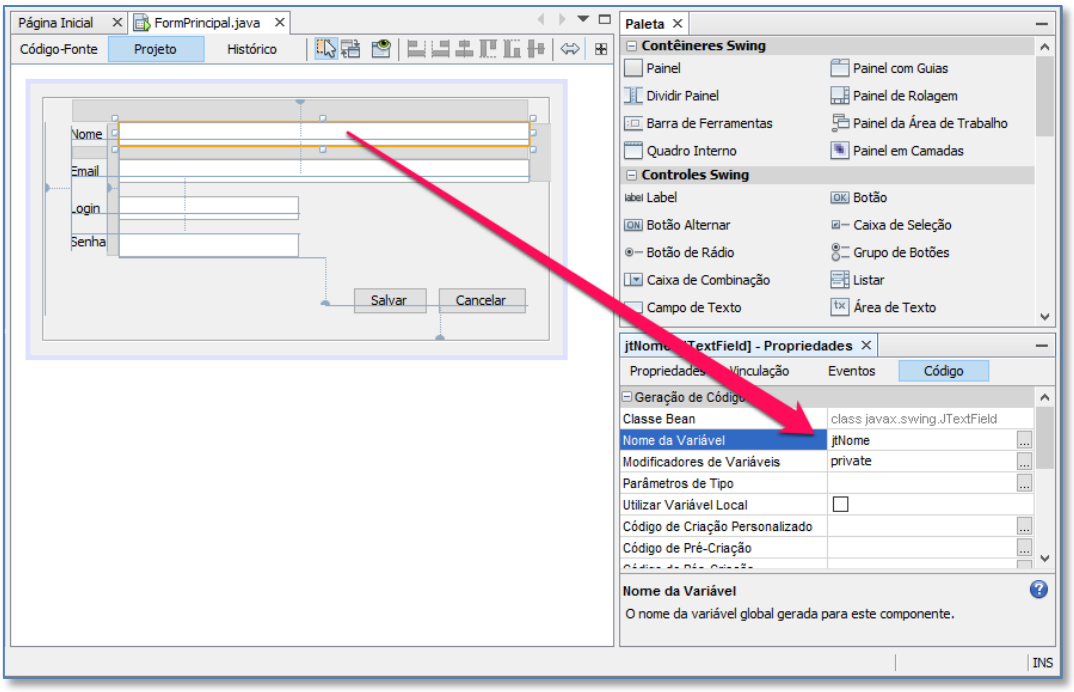

Figura 36 – Componente com nome de variável alterado.

O próximo passo é executar a aplicação. Para este passo, basta clicar no ícone “Executar Projeto” localizado na barra de ferramentas ou tecla de atalho F6.

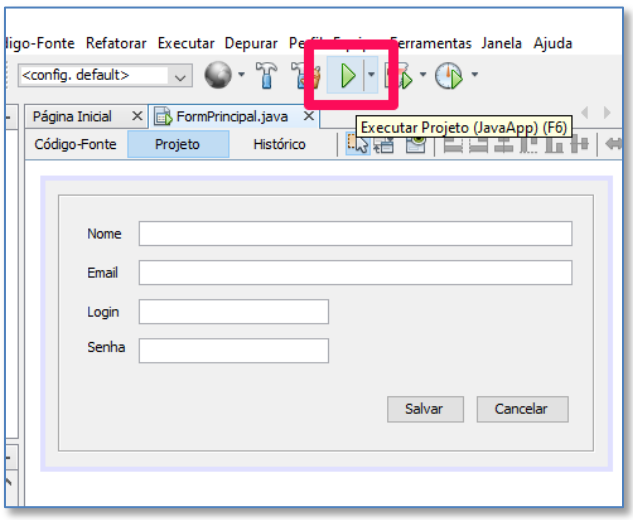

Figura 37 – Botão “Executar Projeto”.

Como no projeto em questão, não definimos uma classe principal, o NetBeans exibirá um questionamento sobre definir a classe “FormPrincipal.java” como classe principal para execução.

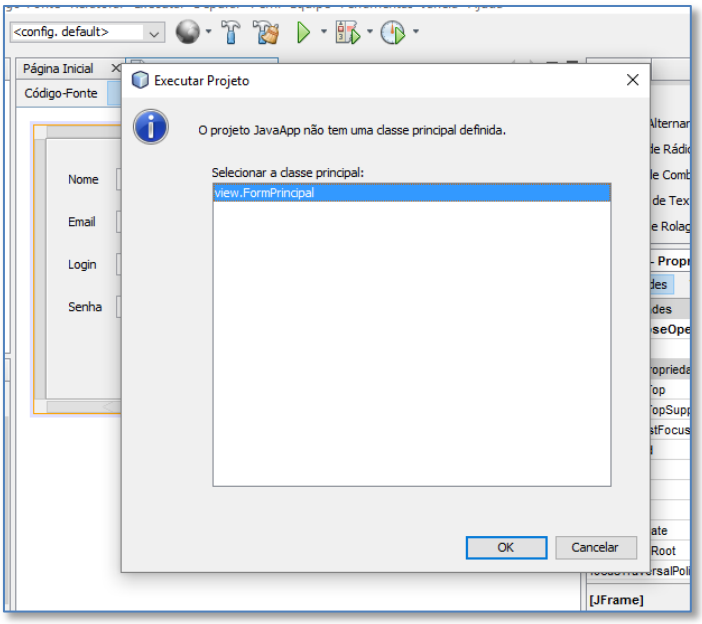

Figura 38 – Seleção de classe principal.

Após selecionar a classe principal, será exibido o formulário conforme ilustração da Figura 39.

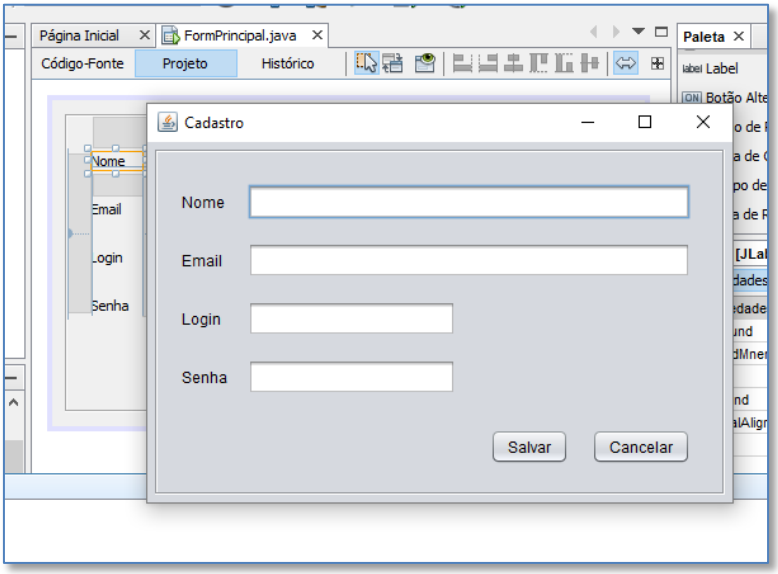

Figura 39 – Execução de projeto JavaApp.

Clique no ícone “X” para fechar a janela do formulário e encerrar a aplicação Java.

Tendo visualizado o formulário conforme Figura 39, partimos para o próximo passo que corresponde à criação de dois eventos para os botões “Salvar” e “Cancelar”.

Para o botão “Cancelar”, criaremos um evento para fechar o formulário e encerrar a aplicação, assim como se tivéssemos clicado no ícone “X” da janela.

Com o formulário aberto no modo “Projeto”, clique no botão “Cancelar” com o botão direito e selecione a opção “Eventos > Mouse > mousePressed”.

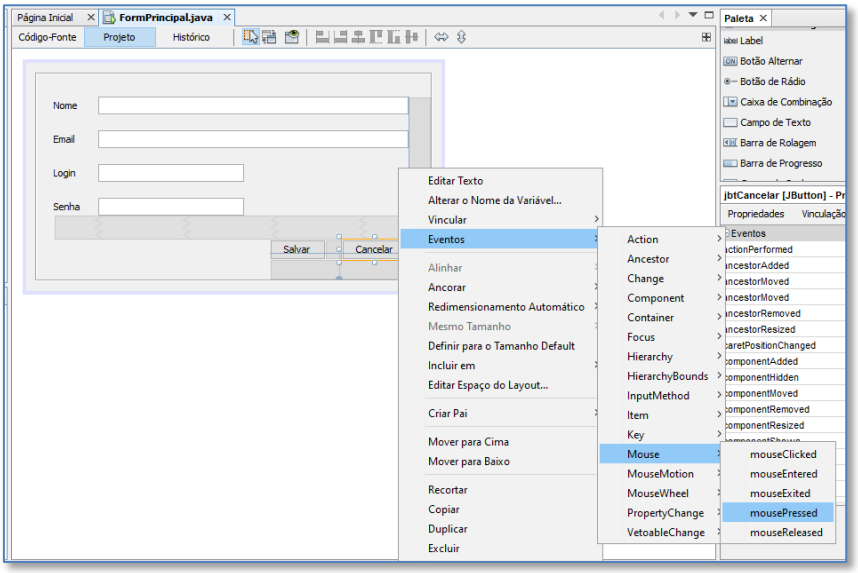

Figura 40 – Criação de evento para o botão “Cancelar”.

Quando selecionado o evento mencionado, o NetBeans abrirá a classe “FormPrincipal.java” no modo “Código”. Veremos também que o próprio NetBeans terá criado o método “**jbtCancelarMousePressed**”. Ou seja, foi criado um método com nome concatenado ao nome da variável do componente e ao nome do evento em questão (jbtCancelar + MousePressed). Veja a Figura 41.

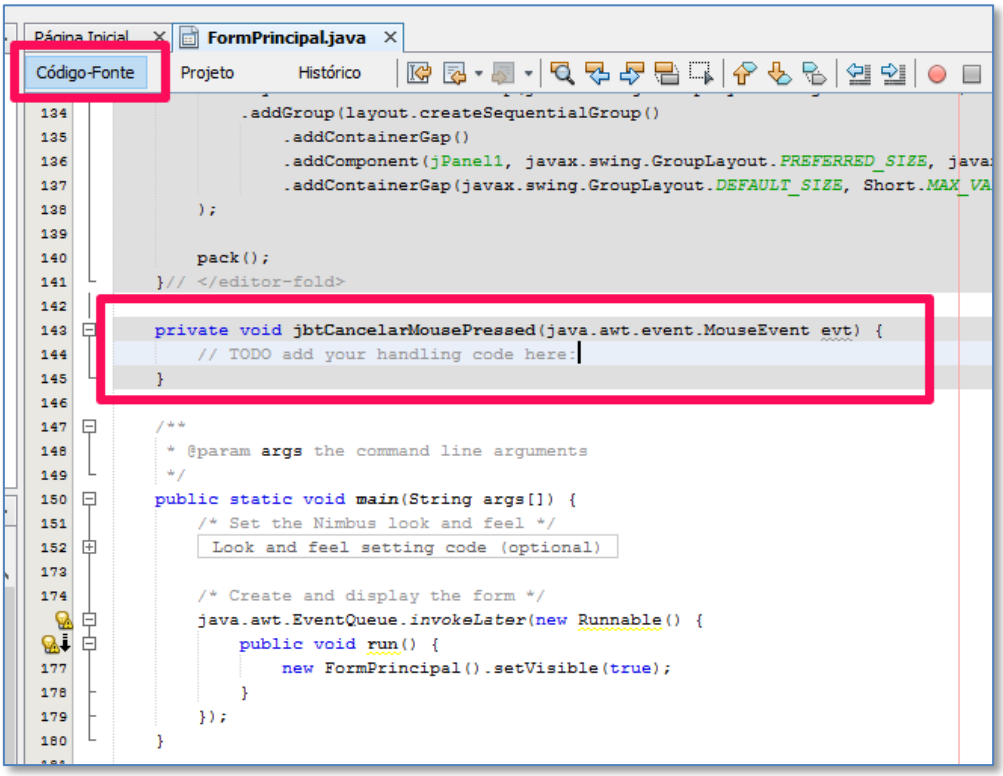

Figura 41 – Método “jbtCancelarMousePressed” criado.

Com o método “jbtCancelarMousePressed” criado, colar o seguinte conteúdo dentro do método:

```
        String message = "Tem certeza que deseja sair?";
        String title = "JavaApp";
        int reply = JOptionPane.showConfirmDialog(null, message, title,
JOptionPane.YES_NO_OPTION);
        if (reply == JOptionPane.YES_OPTION)
        {
         System.exit(0);
        }
```

Após a inclusão do trecho, o método será exibido conforme Figura 42.

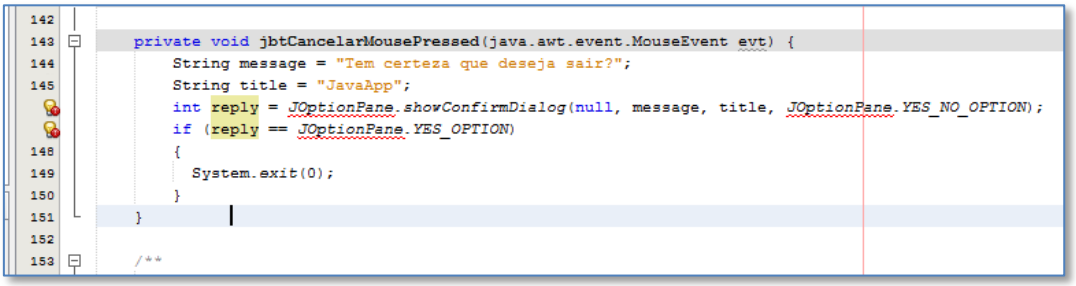

Figura 42 – Trecho de código inserido no método “jbtCancelarMousePressed”.

Podemos notar que as linhas 146 e 147 estão agora com erro. Isso ocorreu, pois como utilizaremos a classe “JOptionPane” para exibição de mensagem customizada, precisaremos importar ela na nossa classe “FormPrincipal”. Para tal ação, basta clicar no ícone contido na linha 146 e selecionar a opção “Adicionar importação para javax.swing.JOptionPane”.

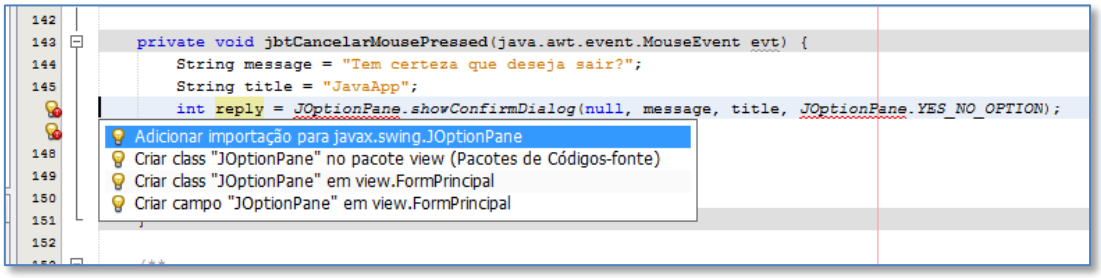

Figura 43 – Adicionar importação de classe “JOptionPane”.

Após a importação da classe mencionada, salve o projeto e execute para testar a inclusão de evento ao botão “Cancelar”. Ao clicar no botão “Cancelar”, deverá ser exibida na tela a mensagem solicitando a ação opção “Sim” ou “Não”.

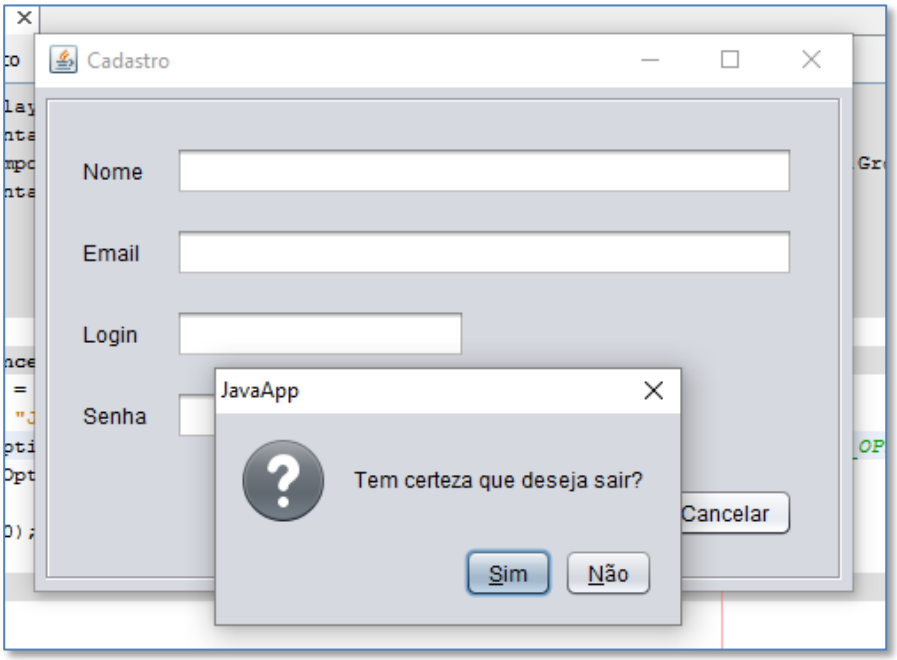

Figura 44 – Mensagem de confirmação.

Caso seja selecionada a opção “Sim”, a aplicação será encerrada. Caso seja selecionada a opção “Não”, a aplicação continuará em execução.

Da mesma maneira que criamos o evento para o botão “Cancelar”, vamos criar o evento para o botão “Salvar”. A Figura 45 demonstra o método já criado após selecionar o evento “mousePressed” e com o código a seguir que deverá ser inserido.

```
// Inicio Log verificacao
System.out.println(jtNome.getText());
System.out.println(jtEmail.getText());
System.out.println(jtLogin.getText());
System.out.println(jpSenha.getPassword());
// Fim Log verificacao

String message = "Operação realizada com sucesso";
String title = "JavaApp";
JOptionPane.showMessageDialog(null, message, title,
JOptionPane.INFORMATION_MESSAGE);
```

Figura 45 – Método “jbtSalvarMousePressed” criado.

Após a inclusão do trecho mencionado, salve o projeto e execute para testar a inclusão de evento ao botão “Salvar”.

Preencher os campos do formulário e clicar no botão “Salvar”. A aplicação exibirá mensagem de sucesso e realizará o log dos dados do formulário no console conforme mostra a Figura 46.

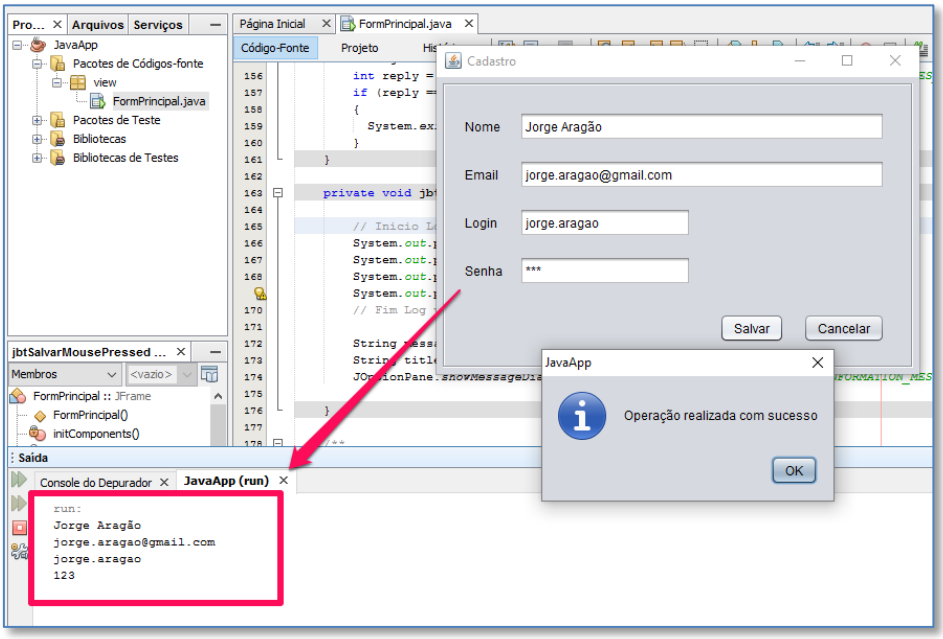

Figura 46 – Mensagem de informação e log dos dados preenchidos no console.

Vamos aprender mais com o professor Ederson, ele vai explicar detalhes sobre os projetos com Java no vídeo que está disponível na rota.

Tema 04: A Biblioteca Gráfica OpenGL para Programação em Linguagem C

OpenGL

OpenGL (*Open Graphical Library*) é uma API (*Application Programming Interface*) criada em 1992 que permite a criação de imagens através da definição de objetos (2D/3D) usando um conjunto de formas primitivas geométricas e rotinas para manipulação. Essa API possui as seguintes características:

- Segue a convenção de chamada de biblioteca da linguagem C.
- Aproximadamente 120 comandos para especificar objetos e operações de aplicações interativas 3D ou 2D.
- Algoritmos otimizados pela *Silicon Graphics*.

- Primitivas vetoriais e matriciais (imagens).
- Capaz de gerar imagens de alta qualidade.
- Comumente implementado de forma a tirar partido da aceleração gráfica (se disponível).
- Independente de plataforma.
- Independente de sistema de janelas.
- Foi adotado mundialmente como um padrão aberto para *hardware* gráfico.
- Aplicações em jogos, ferramentas CAD, imagens médicas ou modelagem de efeitos especiais para filmes (ex.: Jurassic Park, Star Wars).
- Dentre os programas de modelagem, estão 3D Max, Maya, Rhino 3D etc.
- Dentre os jogos, estão Quake, Half-Life, NFL etc.

A API OpenGL dá suporte, além de primitivas gráficas com linhas e polígonos, à iluminação, sombreamento, textura, transparência, animação, eventos de entrada por teclado e mouse. Executa também transformações de translação, escala e rotação.

Funciona como uma máquina de estados (variáveis de estado como cor, textura, fontes de iluminação etc.).

As funções da API fornecem acesso direto à praticamente todos os recursos do *hardware* gráfico.

Trabalha com desenho tanto na forma vetorial (definida por vértices) como na forma matricial (definida pixel a pixel). As medidas de ângulos são em graus. A representação de cada componente de cor é definida por um número de ponto flutuante de 0 a 1. As coordenadas na tela funcionam como um plano cartesiano, sendo definido pelo usuário, por exemplo, pode-se configurar o centro correspondente a 0, o topo da janela 1 e a base da janela -1. Sempre ao

iniciar uma aplicação, é necessário definir o sistema de coordenadas, se 2D (gluOrtho2D) ou 3D (glOrtho).

A seguir, algumas bibliotecas relacionadas à API OpenGL:

- GLU (*OpenGL Utility Library*)
  - Parte do padrão OpenGL.
  - Contém funções utilitárias, principalmente para mapeamento de janelas, coordenadas etc.
- GLEW (*OpenGL Extension Wrangler Library*)
  - Biblioteca multiplataforma utilizada para consultar e carregar extensões OpenGL.
- GLUT (*OpenGL Utility Toolkit*)
  - Biblioteca da Silicon Graphics.
  - API portátil de acesso aos sistemas de janelas.
  - Tratamento de eventos como mouse, teclado.
  - Encapsula e esconde as camadas proprietárias.
  - Possui seus tipos de dados independentes de plataforma, como GLboolean, GLdouble, GLint, GLbyte e GLfloat.

Primitivas Geométricas

A API OpenGL oferece uma série de primitivas geométricas para definição de objetos gráficos. Basicamente, existem três tipos de primitivas: pontos, linhas ou superfícies (triângulos, quadriláteros e polígonos convexos). A escolha de qual primitiva utilizar vai depender da aplicação.

Para a criação de primitivas, é necessário delimitá-las por meio dos comandos glBegin e glEnd, conforme exemplo a seguir:

```
glBegin (nome da primitiva);
especif de vértices, cores, coord de textura, propriedades de material
glEnd ();
```

Entre os comandos glBegin e glEnd, apenas alguns comandos podem

ser usados. São eles:

- glMaterial
- glNormal
- glTexCoord
- glVertex

Para definição dos vértices, utiliza-se o comando Vertex. A seguir, exemplos do comando Vertex:

```
glVertex2f( float x, float y); //vértice para um eixo 2D
glVertex3d(double x,double y, double z); //vértice para um exio 3D
```

Uma vez emitido um vértice (glVertex), este é desenhado com as propriedades (cor, material, normal, coordenadas de textura etc.) registradas nas variáveis de estado correspondentes. Portanto, antes de emitir um vértice, assegurar-se que cor, material, normal etc. contêm o valor correto. A descrição dos parâmetros dos comandos pode ser consultada no seguinte endereço: http://www.opengl.org/documentation/specs/man_pages/hardcopy/GL/html/gl

Dentre os três tipos de primitivas citadas (ponto, linha e polígonos), temos as subdivisões a seguir:

- Um tipo de ponto
- Três tipos de linhas
- Seis tipos de polígonos

Ponto

A primitiva GL_POINTS é responsável por desenhar pontos na tela. O código no exemplo a seguir desenha três pontos na tela. Cada vértice torna-se um ponto.

```
glBegin( GL_POINTS );
   glVertex2f( xf, yf);
```

```
   glVertex2f( xf, yf);
   glVertex2f( xf, yf);
glEnd();
```

O tamanho do ponto pode ser modificado através do comando glPointSize (GLint tamanho), bastando passar como parâmetro o tamanho do ponto.

Linhas

- **GL_LINES:** O terceiro ponto é ignorado, pois a linha é formada por dois vértices. Se houvesse um quarto vértice, uma nova linha entre o terceiro e quarto vértice seria exibida.
- **GL_LINE_STRIP:** Cria linhas consecutivas, ligando o primeiro vértice com o segundo, o segundo com o terceiro e assim por diante.
- **GL_LINE_LOOP:** Funciona de maneira semelhante a primitiva anterior, porém o último vértice é ligado a primeira, devido a isso o LOOP no seu nome.

A espessura de uma linha pode ser modificada através do comando glLineWidth (GLint espessura), bastando passar como parâmetro a espessura da linha.

Exemplo de uso da primitiva GL_LINES:

```
glLineWidth(10.0) // default é 4.0
glBegin(GL_LINES);
   glVertex2i(40, 100);
   glVertex2i(500, 400);
glEnd();
```

Polígonos

São áreas formadas por várias linhas conectadas onde as arestas do polígono não podem se cruzar. Devem ser áreas convexas e não há restrição quanto ao número de segmentos do polígono.

- **GL_TRIANGLES:** Desenha triângulos a cada 3 vértices fornecidos.

- **GL_TRIANGLE_STRIP:** Uma série de triângulos conectados. Após o desenho do primeiro triângulo, cada vértice adicional forma um novo triângulo com dois últimos pontos fornecidos.
- **GL_TRIANGLE_FAN:** Uma série de triângulos com um único vértice em comum. O vértice comum é o primeiro vértice fornecido.
- **GL_QUADS:** Desenha um quadrilátero a cada 4 vértices fornecidos.
- **GL_QUAD_STRIP:** Desenha uma série de quadriláteros. Após o primeiro, apenas mais 2 vértices precisam ser fornecidos para o desenho do segundo quadrilátero.
- **GL_POLYGON:** Desenha polígonos convexos simples com um número arbitrário de vértices.

Exemplo de uso da primitiva GL_POLYGON:

```
glBegin(GL_POLYGON);
   glVertex3f (1, 2, 0.0);
   glVertex3f (2, 2, 0.0);
   glVertex3f (2, 3, 0.0);
   glVertex3f (1, 3, 0.0);
glEnd();
```

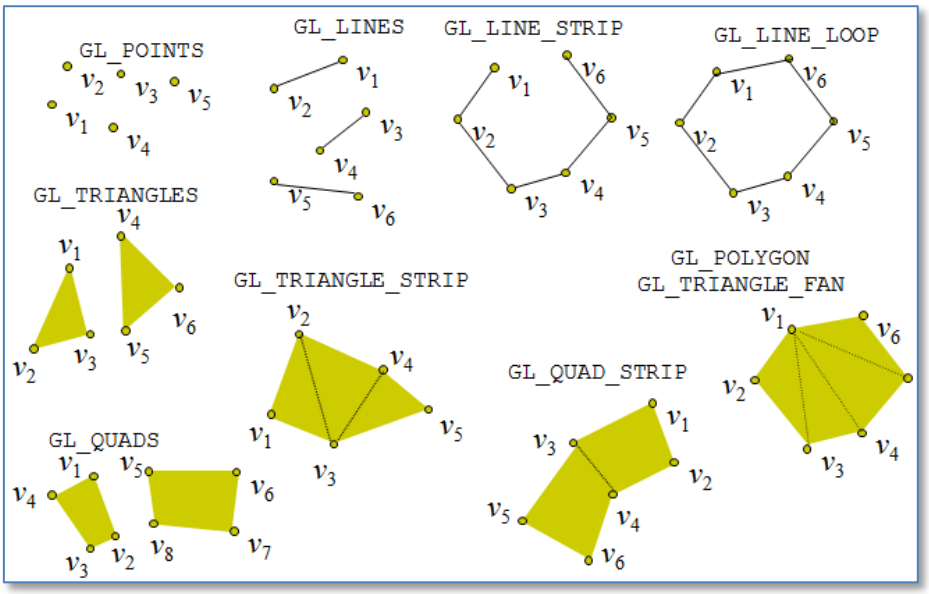

Figura 47 – Primitivas Geométricas.

Especialmente para um retângulo, ou seja, que possui 4 vértices, há o comando glRect, que é executado em apenas uma linha de código.

Transformações Geométricas – Translação e Rotação

As transformações geométricas no espaço são representadas por matrizes.

O OpenGL usa uma pilha de matrizes de transformação para ajudar a lembrar a sequência de transformações realizadas. Essa funcionalidade é provida pelas funções “glPushMatrix()” e “glPopMatrix()”. Para o desenho de determinadas formas geométricas tridimensionais, possui algumas funções já prontas. Os objetos são: cone, icosahedron, teapot, cube, octahedron, tetrahedron, dodecahedron, sphere, torus. É possível também criar o desenho de um *wireframe* ou preenchimento sólido, por exemplo, para um cubo utilizando as seguintes funções “glutWireCube(GLdouble size)” e “glutSolidCube(GLdouble size)”.

Exemplo de um cubo em *wireframe* com algumas transformações:

```
glPushMatrix();
glTranslatef (-2.0, 0.0, 0.0);
glScalef (3.0, 2.0, 5.0);
glutWireCube (1.0);
glPopMatrix();
```

Explicando o exemplo, a origem do sistema de coordenadas é levada para o ponto (x,y,z) = (-2.0,0.0,0.0) através da função “glTranslatef()”, definindo a coordenada de origem para o desenho do cubo. Em seguida, usando a função “glScalef()”, o sistema de coordenadas é ampliado em (Sx,Sy,Sz) = (3.0,2.0,5.0). Finalmente, desenhado um cubo aramado através da função “glutWireCube()”.

Para realizar o processo de rotação, utiliza-se a seguinte função “glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z)”. Quando chamada, efetua uma rotação de *angle* graus no sistema de coordenadas na direção contra o sentido do relógio em torno de um vetor que vai da origem ao ponto

(x,y,z).

Exemplo de rotação:

```
glPushMatrix();
glRotatef (25.0, 0.0, 0.0, 1.0);
glTranslatef (2.0, 0.0, 0.0);
glScalef (2.0, 1.0, 4.0);
glutWireCube (1.0);
glPopMatrix();
```

Animação

Para realizar uma animação, é necessário que haja uma imagem sendo mostrada na tela e uma imagem sendo processada para ocupar o lugar da imagem atual. Para tal ação, o OpenGL possui uma técnica chamada de *double-buffering*. Sendo assim, os buffers alternam entre mostrar e processar imagem. Para efetuar essa ação, utiliza-se a função “glutSwapBuffers()”. É necessário levar em conta a taxa de atualização entre processar e mostrar a imagem. A taxa de atualização de um monitor usual de microcomputador é em torno de 60 frames por segundo (fps) ou um pouco mais. Caso o processamento seja muito demorado, a taxa de atualização pode cair, por exemplo, para 30 fps, 20 fps, chegando a um ponto em que as transições são perceptíveis pelo olho humano.

A seguir, veja o exemplo de uma animação para girar um quadrado.

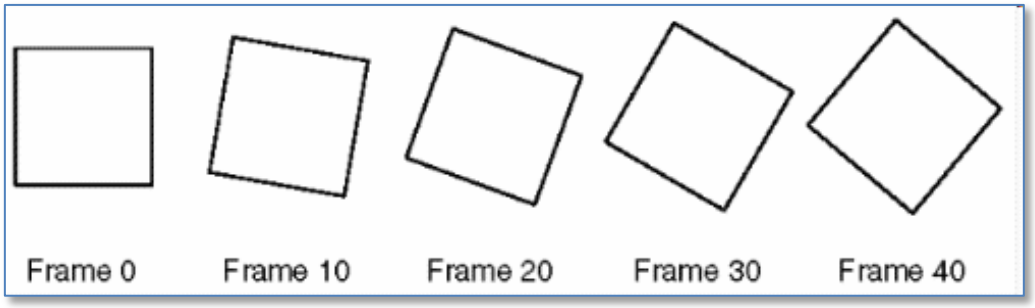

Figura 48 – Animação para girar um quadrado.

Uma forma de gerenciar a animação é utilizar a função “glutIdleFunc()”, passando a função de animação como parâmetro e dentro da função de animação, após realizar os cálculos, chamar a função “glutPostRedisplay()”.

Cores e Iluminação

O OpenGL utiliza o formato de cores RGBA, onde “A” significa alfa, que configura transparência. Para preencher um polígono ou desenhar uma linha, pode-se definir o tipo de sombreamento (*shading*), podendo ser somente uma cor (*flat*) ou uma combinação de várias cores (*smooth*). Isto é feito pela função “glShadeModel()”. No caso de uma primitiva gerada a partir de vértices, o *smooth shading* interpola as cores provenientes de cada vértice. Define-se “glShadeModel” (GL_SMOOTH).

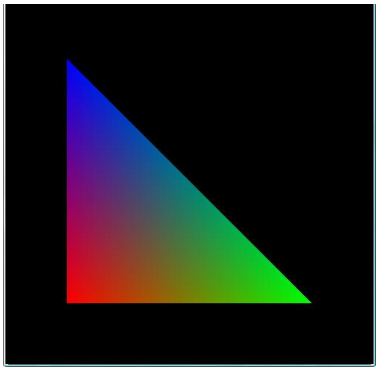

Figura 49 – Cores e Iluminação, primitiva GL_SMOOTH

Para utilizar a propriedades alfa, é necessário habilitar a opção *blend*. Isso é feito chamando a função “glEnable” (GL_BLEND) e utilizando a função “glBlendFunc” (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA). A seguir, exemplo de sobreposição de triângulos com a opção *blend*.

Figura 50 – Cores e iluminação, sobreposição de triângulos com a opção *blend*.

A iluminação é caracterizada de 3 formas:

- **Ambiente:** luz dispersa igualmente em todas as direções.
- **Difusa:** vem de uma direção.
- **Especular:** é como uma luz refletida em um espelho, um brilho.

Para determinar as propriedades de reflexão do material, também há cores ambiente, difusa e especular.

A reflexão ambiente e difusa é geralmente similar à cor do objeto, mas a reflexão especular é geralmente branca ou cinza. A seguir, veja um exemplo de renderização de uma esfera.

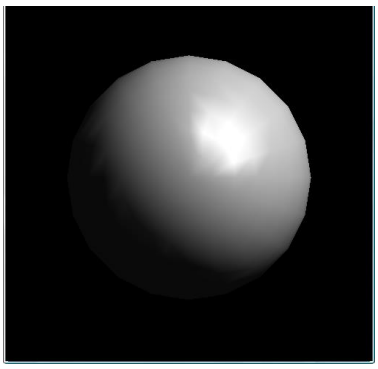

Figura 51 – Cores e Iluminação, renderização de uma esfera.

Passos para adicionar luz à cena:

- Definir o vetor normal de cada vértice de todos os objetos, para determinar a orientação do objeto relativa à fonte de luz;
- Criar, selecionar e posicionar uma ou mais fontes de luz;
- Criar e selecionar um modelo de iluminação;
- Definir as propriedades de material do objeto da cena.

Observação: no caso de utilizar uma função de desenho pronta, por exemplo, “glutSolidSphere()”, a definição dos vetores normais é parte da função.

Saiba Mais: Acesse os sites a seguir referentes à biblioteca OpenGL para você conhecer mais sobre.

- http://www.opengl.org
- http://www.lighthouse3d.com/tutorials/glut-tutorial/

No vídeo, o professor Ederson vai comentar mais sobre a biblioteca gráfico openGL em linguagem C. Confira!

Tema 05: Projetos Utilizando OpenGL para Animações Gráficas

A seguir, veremos um exemplo de aplicação na linguagem C utilizando OpenGL. O ambiente para rodar as aplicações é o *Visual Studio Community 2015*. O tipo de projeto é Visual C++: Win32 Console Application. Também é necessário instalar um pacote de bibliotecas, por meio da ferramenta *NuGet Package Manger*, então, procure "nupengl.core" e "nupengl.core.redist", associe ao projeto criado e clique em "Install" para cada um desses 2 pacotes.

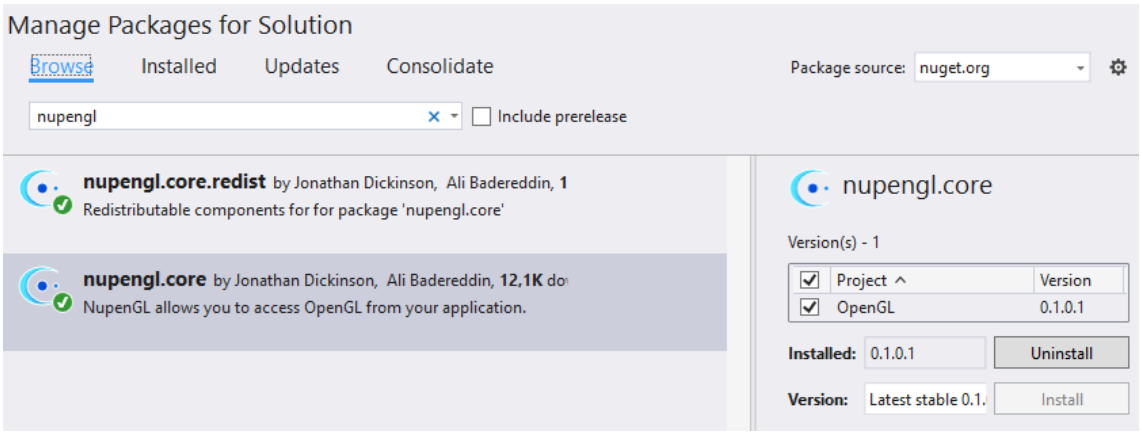

Figura 52 – Pacote OpenGL para Visual Studio Community.

O exemplo consiste em um quadrado que será animado, rotacionado continuamente. Serão utilizados os conceitos de primitivas geométricas, transformações geométricas, animação e interação com o mouse.

Veja abaixo a função de inicialização:

```
static GLfloat spin = 0.0;
int main(int argc, char** argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
```

```
        glutInitWindowSize(250, 250);
        glutInitWindowPosition(100, 100);
        glutCreateWindow(argv[0]);
        glClearColor(0.0, 0.0, 0.0, 0.0);
        glShadeModel(GL_FLAT);
        glutDisplayFunc(display);
        glutReshapeFunc(reshape);
        glutMouseFunc(mouse);
        glutMainLoop();
        return 0;
}
```

O comando “glutInitDisplayMode” configura o sistema de cores para RGB (GLUT_RGB) e também habilita a buferização dupla (GLUT_DOUBLE) para podermos trabalhar com animação. Em seguida, são inicializados o tamanho e a posição da janela a ser criada, assim como a cor de fundo. Temos também, fora da função, a declaração de uma variável *global spin*.

Então, é registrada a função *display* como *callback* de desenho por meio da chamada “glutDisplayFunc”, sendo o código da função *display*:

```
void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT);
        glPushMatrix();
        glRotatef(spin, 0.0, 0.0, 1.0); //rotação
        glColor3f(1.0, 1.0, 1.0); // branca
        glRectf(-25.0, -25.0, 25.0, 25.0);
        glPopMatrix();
        glutSwapBuffers();
}
```

Nessa função é desenhado o retângulo branco com fundo preto, tendo na variável *spin* o ângulo de rotação, e a técnica de troca de *buffers* para animação, por meio da chamada “glutSwapBuffers”.

A próxima função registrada como “callback" é a “reshape”, que trata o redimensionamento da janela, por meio da chamada “glutReshapeFunc”, sendo o código da função “reshape”:

```
void reshape(int w, int h)
{
        glViewport(0, 0, (GLsizei)w, (GLsizei)h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(-50.0, 50.0, -50.0, 50.0, -1.0, 1.0);
        glMatrixMode(GL_MODELVIEW);
```

```
        glLoadIdentity();
}
```

Na sequência, é registrada a função “mouse” como “callback” de interação por meio da chamada “glutMouseFunc”, sendo o código da função “mouse”:

```
void mouse(int button, int state, int x, int y)
{
        switch (button) {
        case GLUT_LEFT_BUTTON:
                if (state == GLUT_DOWN)
                        glutIdleFunc(spinDisplay);
                break;
        case GLUT_RIGHT_BUTTON:
                if (state == GLUT_DOWN)
                        glutIdleFunc(NULL);
                break;
        default:
                break;
        }
}
```

Nessa função, tratamos os cliques dos botões do mouse, sendo o botão direito desativando a animação de rotação e o botão esquerdo ativando. A ativação da animação é feita por meio da função “spinDisplay”:

```
void spinDisplay(void)
{
        spin = spin + 2.0;
        if (spin > 360.0)
                spin = spin - 360.0;
        glutPostRedisplay();
}
```

Por fim, é chamada a função “glutMainLoop” que coloca o programa em *loop* e aguardando as ocorrências de chamadas gráficas e interações do usuário.

A tela em execução deste exemplo é mostrada em dois momentos indicando a rotação do quadrado na Figura a seguir.

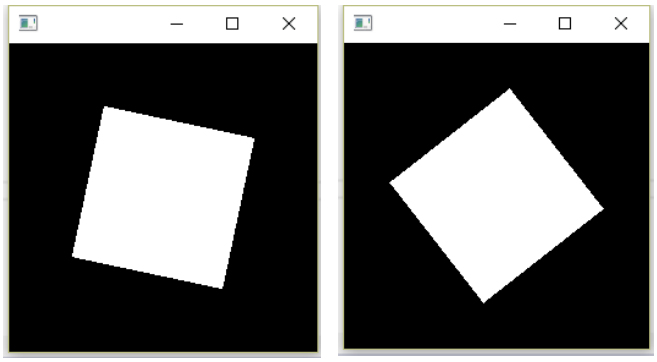

Figura 53 – Aplicação de animação em OpenGL.

Compreenda mais sobre os Projetos Utilizando OpenGL para Animações Gráficas assistindo ao vídeo do professor Ederson no vídeo deste tema.

Na prática

Java Swing - Calculadora

Implemente em Java Swing uma calculadora de 4 operações, incluindo no mínimo os seguintes recursos:

- Adição - soma dois números: n1 + n2.
- Subtração - subtrai o número 2 do número 1: n1 – n2.
- Multiplicação - multiplica dois números: n1 * n2.
- Divisão - divide o número 1 pelo número 2: n1 / n2.
- Botões - números de 0 a 9, operações de +, -, * e /, separador decimal, resultado =, e limpar C.
- Menus - Sair e Sobre.

OpenGL - Teste de Sobreposição

Temos em OpenGL uma propriedade que permite a sobreposição de cores com transparência chamada *blend*. Isto é feito chamando a função “glEnable” (GL_BLEND). E configurando o *blend* utilizando a função “glBlendFunc” (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA).

Como exercício, implemente em linguagem C, utilizando a biblioteca OpenGL, uma aplicação que faça o desenho de dois triângulos se sobrepondo, conforme a figura abaixo.

Figura 54 – Teste de Sobreposição.

Síntese

Nesta aula, vimos os conceitos de linguagem Java, explanamos a NetBeans IDE para desenvolvimento de interface visual Java Swing. Vimos também os elementos de computação gráfica baseados em OpenGL, bem como exemplos de animações gráficas utilizando a biblioteca GLUT.

Iniciamos com uma introdução à linguagem Java e a sua ferramenta de programação visual NetBeans IDE, por meio da qual foram feitos exemplos com Form utilizando a biblioteca Java Swing. Vale lembrar também que a linguagem Java é totalmente orientada a objetos. Conforme vimos, essas aplicações são portáveis para qualquer sistema operacional que possui a máquina virtual do Java, o que é uma grande vantagem.

Na sequência, conhecemos a biblioteca OpenGL e seus recursos de programação que acessam diretamente a placa de vídeo do computador, com implementação em linguagem C, onde foi utilizado o ambiente *Visual Studio Community*.

O bom entendimento desta aula é fundamental, visto que trata dos recursos distintos de interface visual, tanto em linguagem Java quanto em linguagem C. Na questão de interface visual Java, podemos tirar proveito de componentes da biblioteca Swing para o desenvolvimento de aplicativos *desktop*. Já para a linguagem C, a API OpenGL é utilizada na criação de animações gráficas destinadas aos jogos, ferramentas CAD, imagens médicas ou modelagem de efeitos especiais.

O conhecimento adquirido nesta aula é fundamental para o seu aprendizado, de forma a ter em suas mãos as ferramentas disponíveis conforme as necessidades de projeto que surgirem, assim como será utilizado ao longo do curso.

Não fique com dúvidas sobre o assunto desta aula, estude o tema consultando as referências bibliográficas e também outras fontes de pesquisa.

Para finalizar, assista ao vídeo do professor Ederson, ele vai retomar o que estudamos nesta aula.

Referências

DEITEL, P. J.; DEITEL, H. M. **Java:** Como Programar. 8ª Ed. São Paulo: Pearson, 2010.