- Criando um cadastro de cliente em JAVA

Disciplina: LP
Profa. Me Juliana Pasquini
8/9/2025

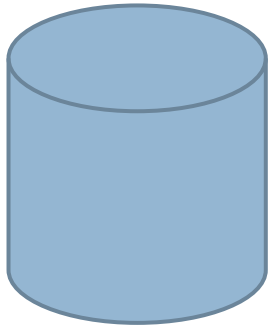
Objetivo

- O objetivo desta aula é desenvolver uma aplicação em JSE (Java Standard Edition) de inserção de dados utilizando alguns padrões de projeto

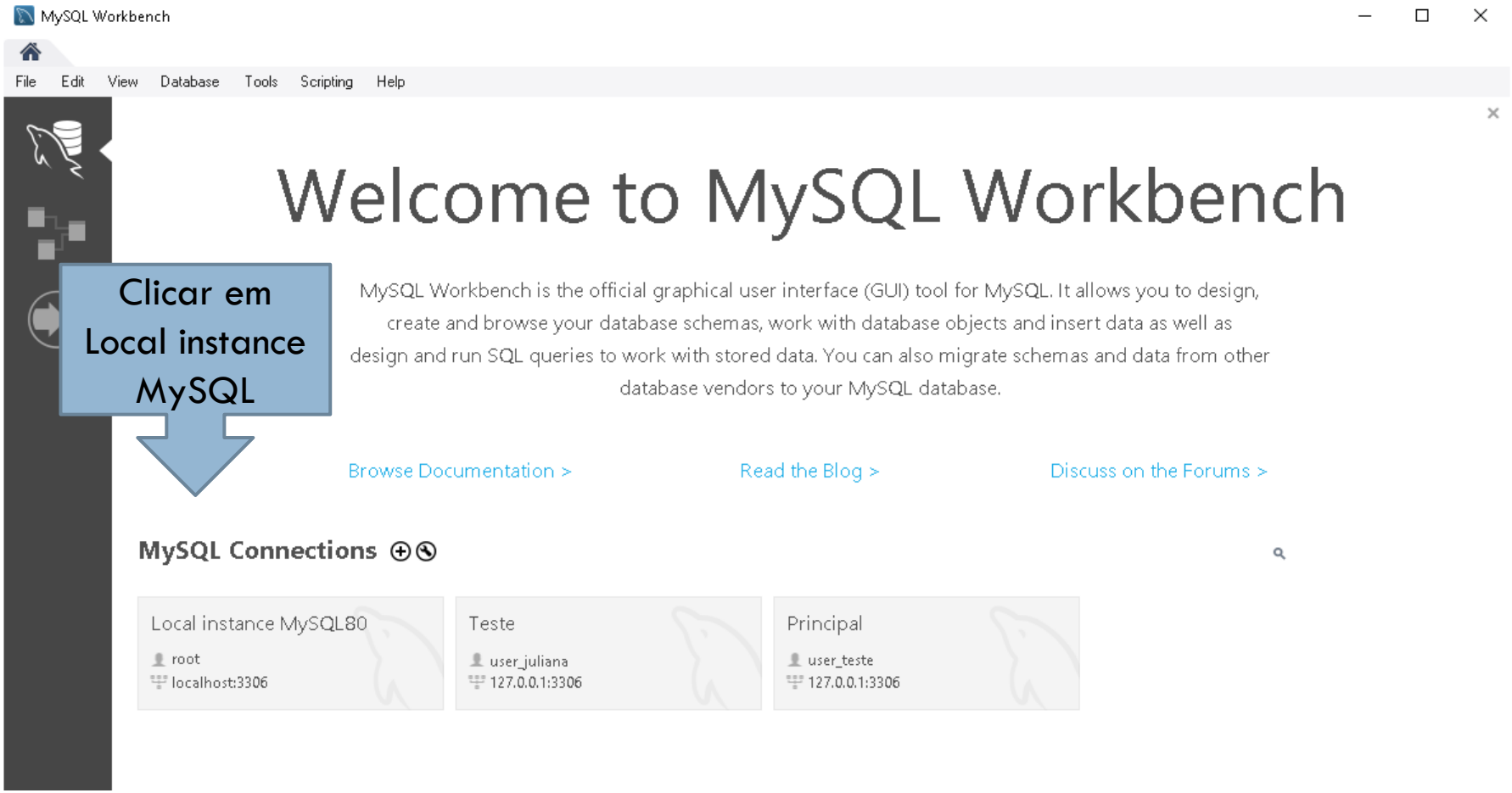
Roteiro

- Criar o banco de Dados Vendas e a tabela Cliente no MySQL;
- Criar o Projeto no Netbeans.

Criando o Banco de Dados e a tabela Cliente no MySQL



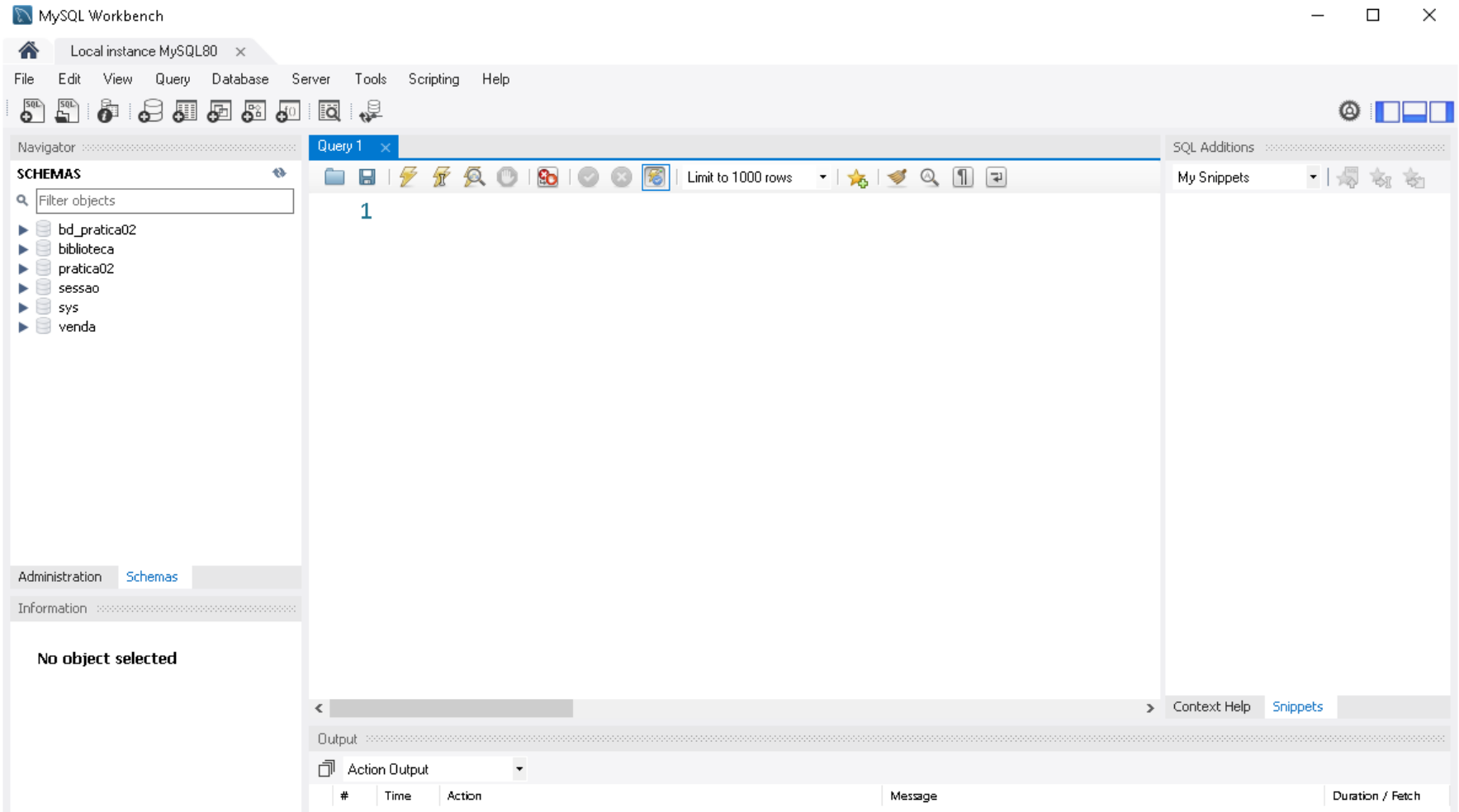
Acessando MySQL Workbench (1)



Acessando MySQL Workbench(2)

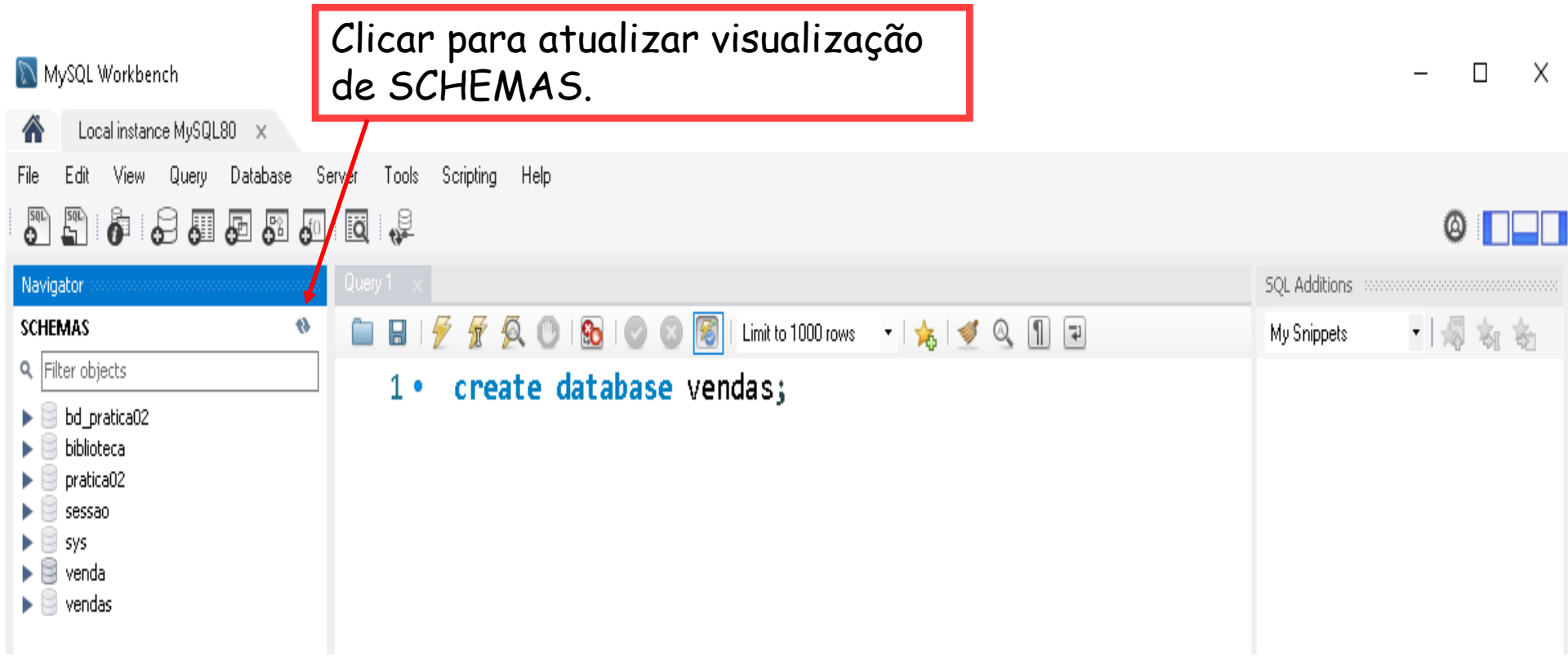
Entrar com a senha, informada na instalação. Nos laboratórios da Fatec colocar fatec.

Acessando MySQL Workbench(3)

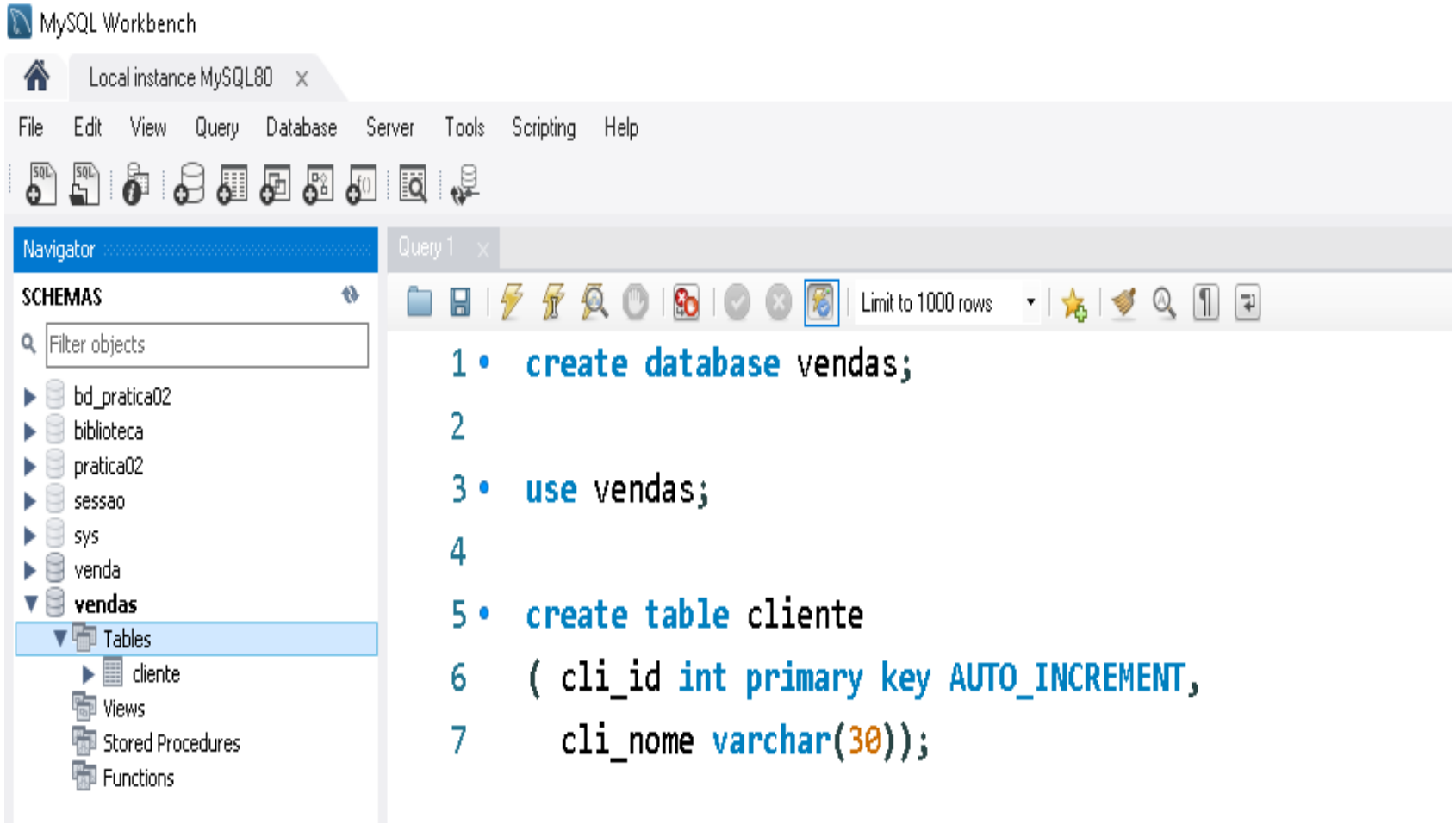


Criando o Banco de Dados Vendas

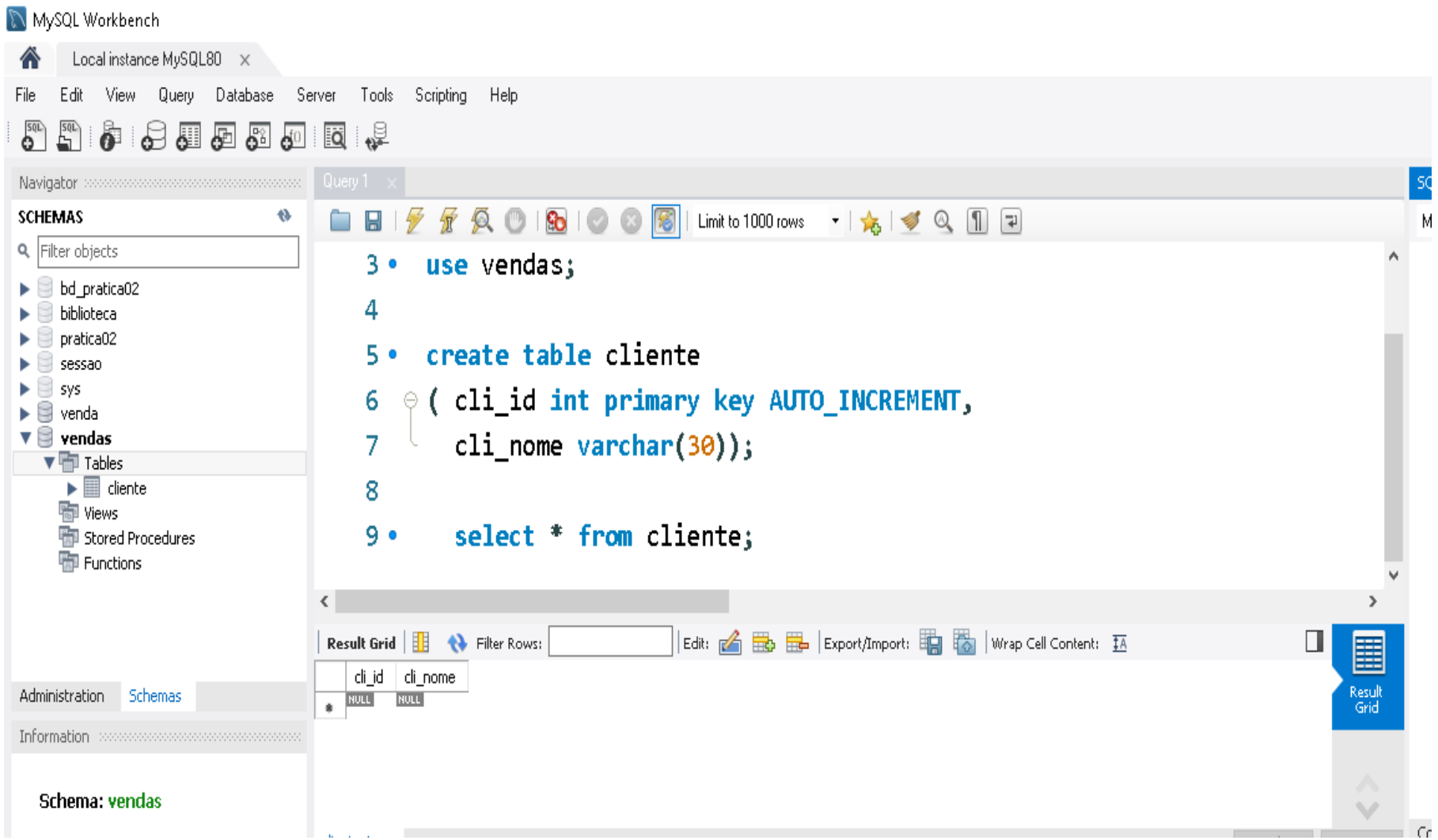
Sintaxe: create database <nome do Banco de dados>;



Criando a tabela cliente

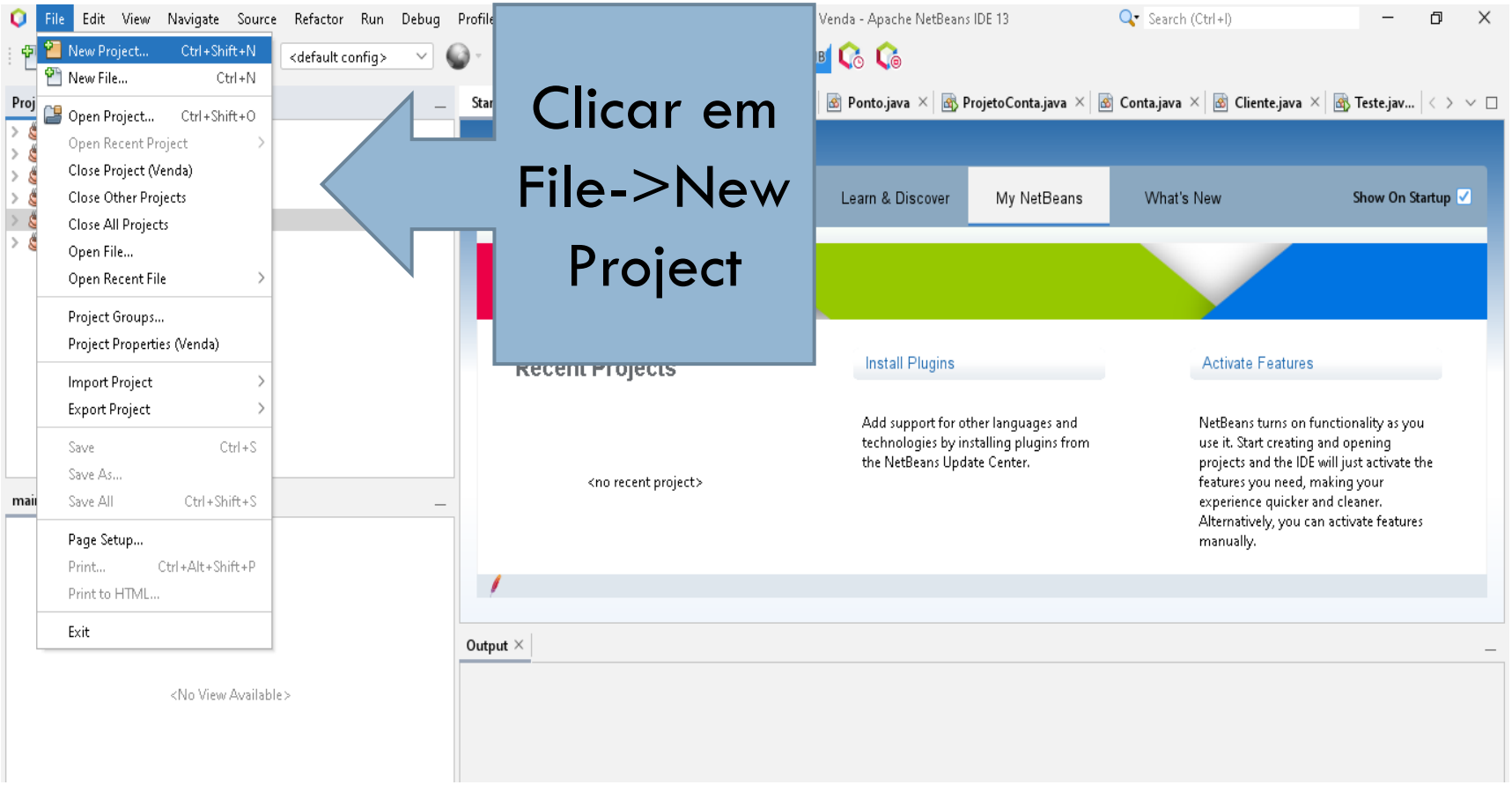


Usando o comando Select



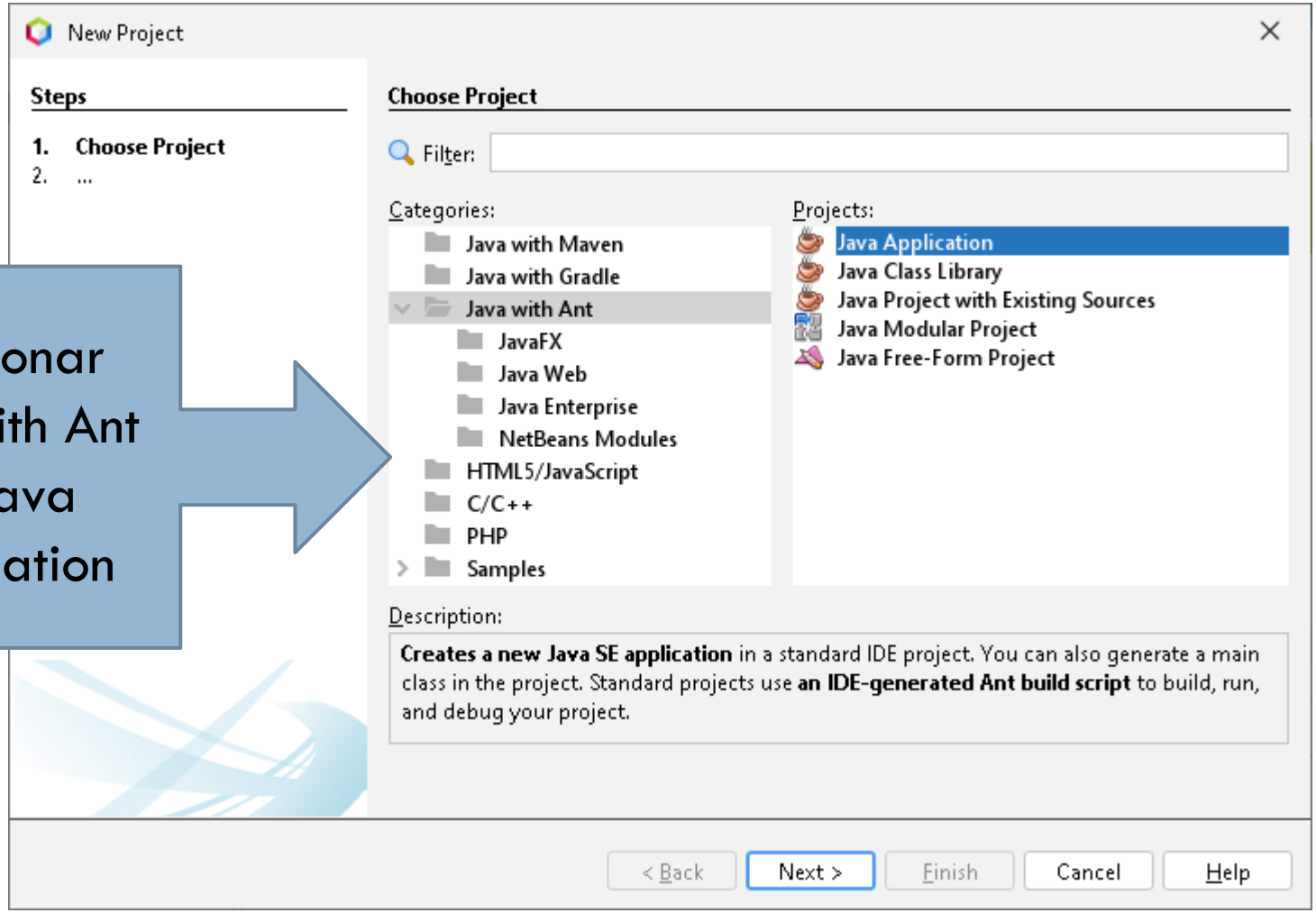
Criando o Projeto Venda no NetBeans

Criando um Novo Projeto (1)

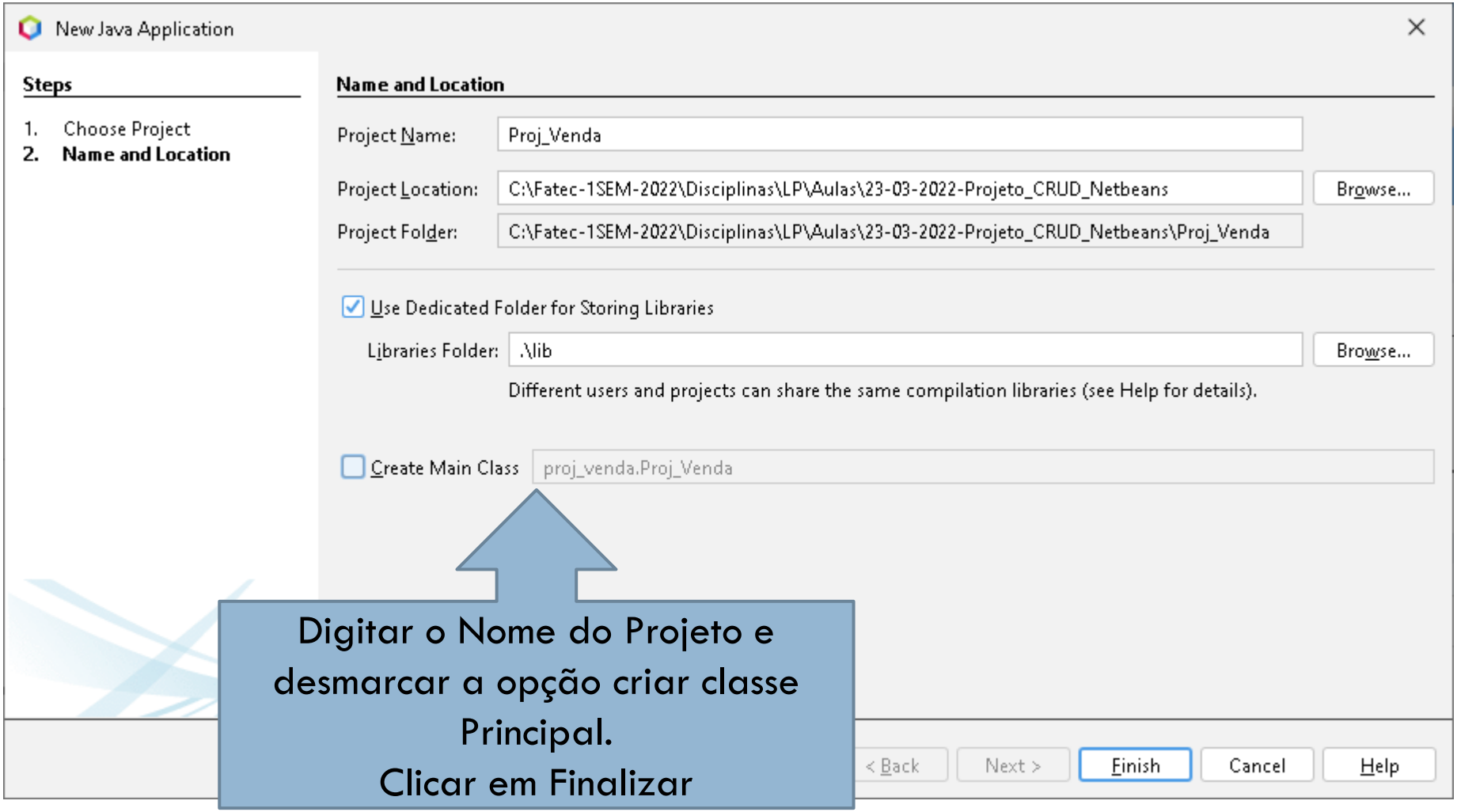


Criando um Novo Projeto (2)

Selecionar Java with Ant -> Java Application

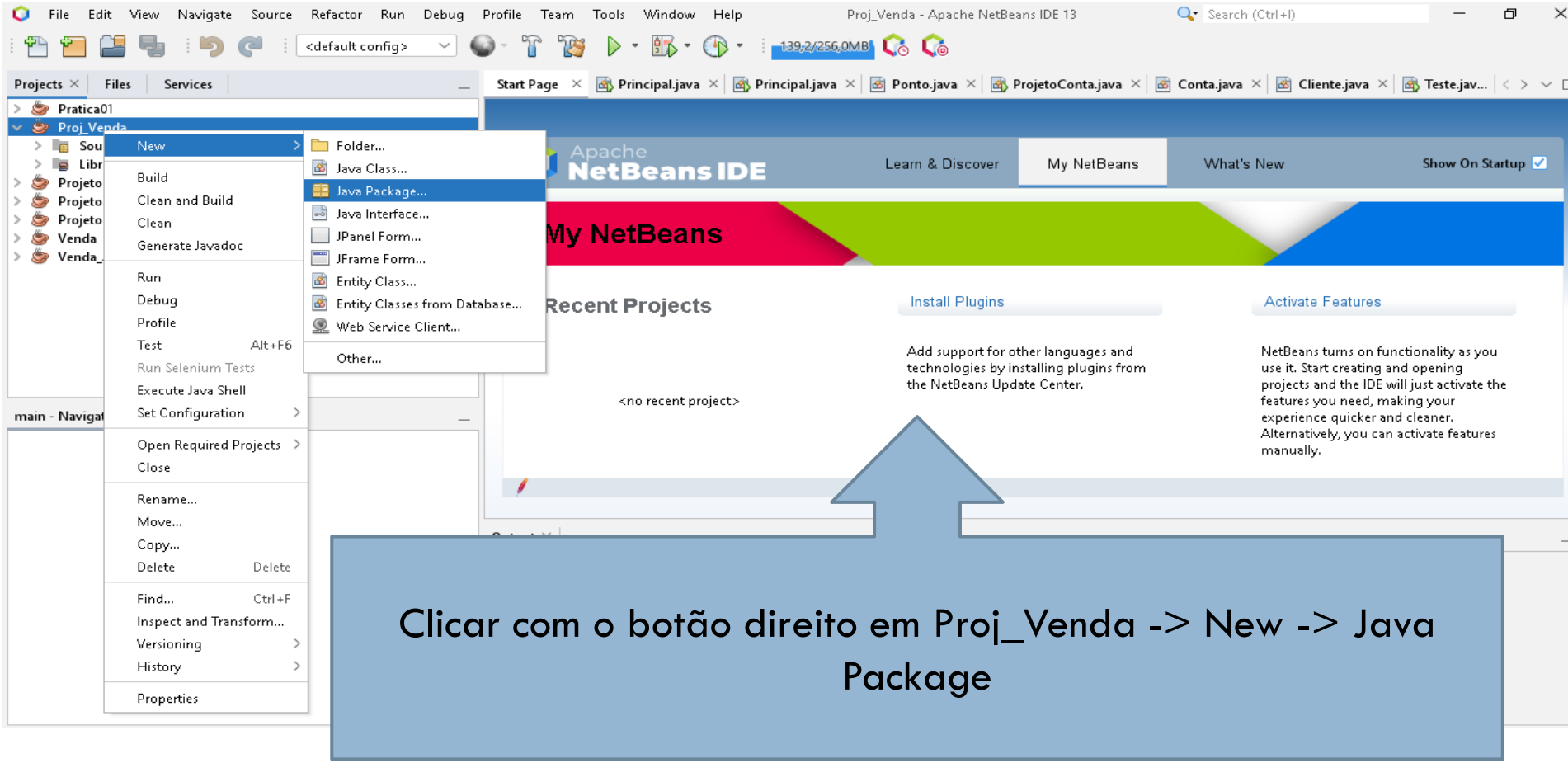


Criando um Projeto (3)



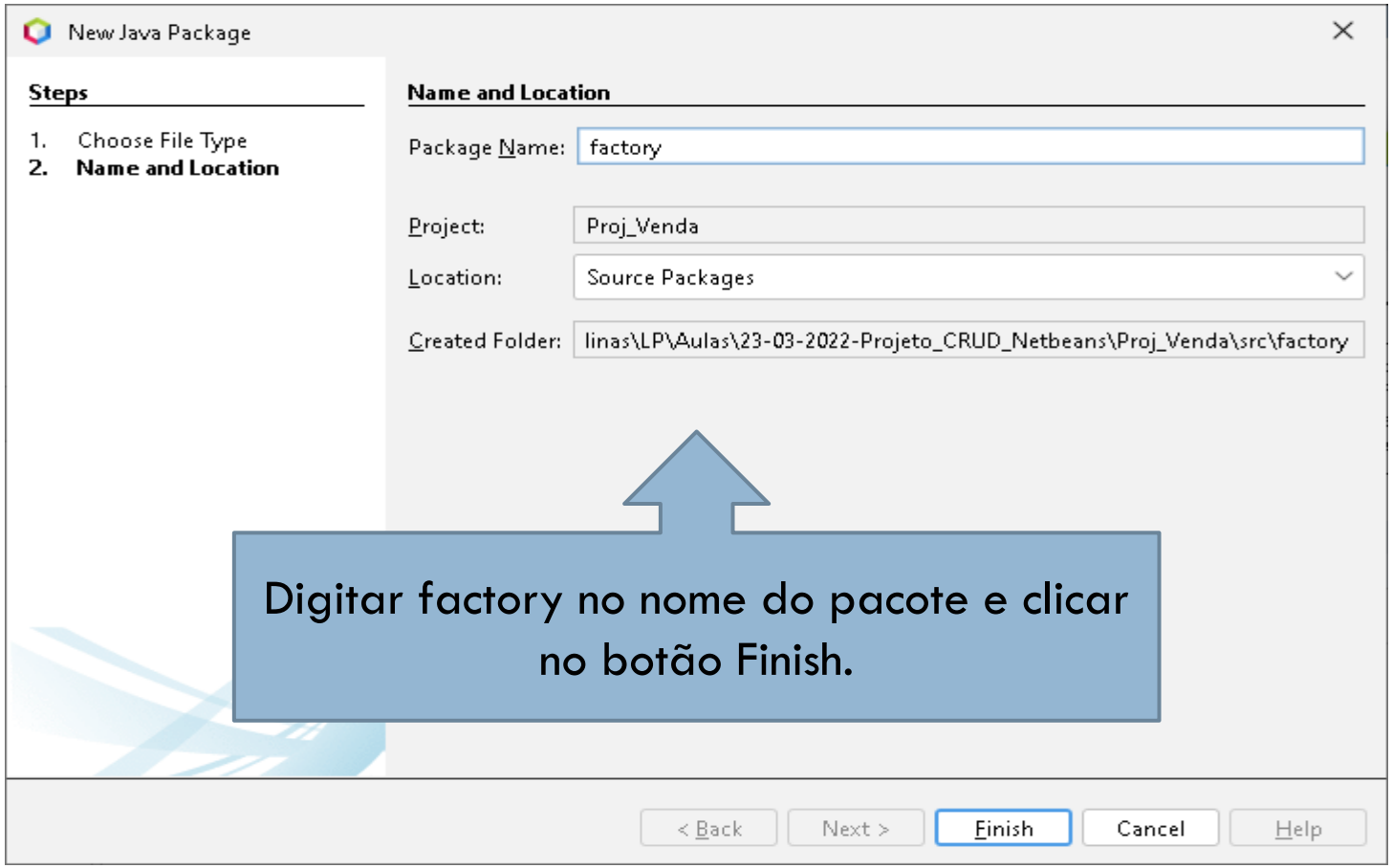
Digitar o Nome do Projeto e desmarcar a opção criar classe Principal.
Clicar em Finalizar

Criando Pacotes (Packages) (1)



Clicar com o botão direito em Proj_Venda -> New -> Java Package

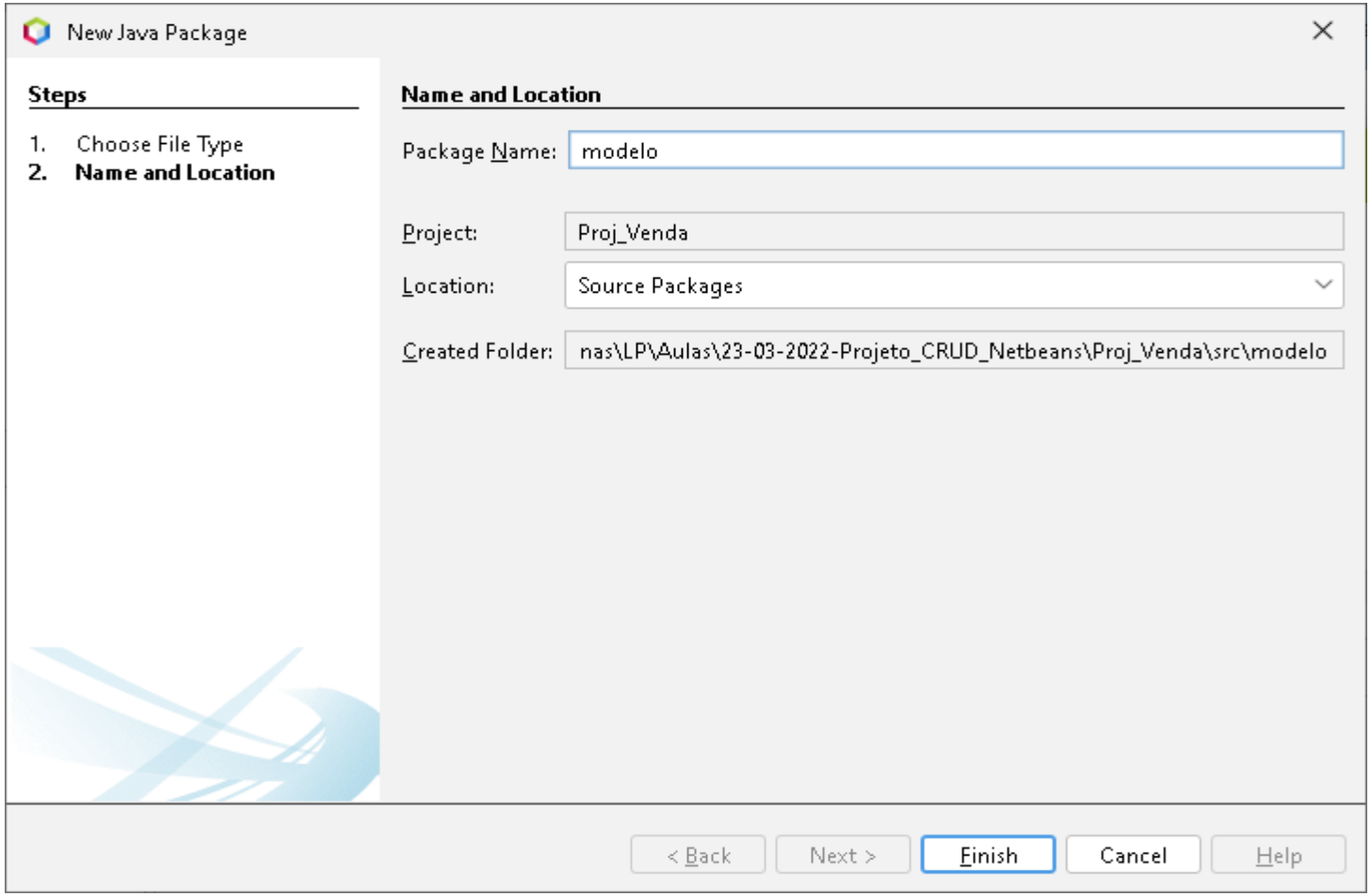
Criando Pacotes (Packages) (2)



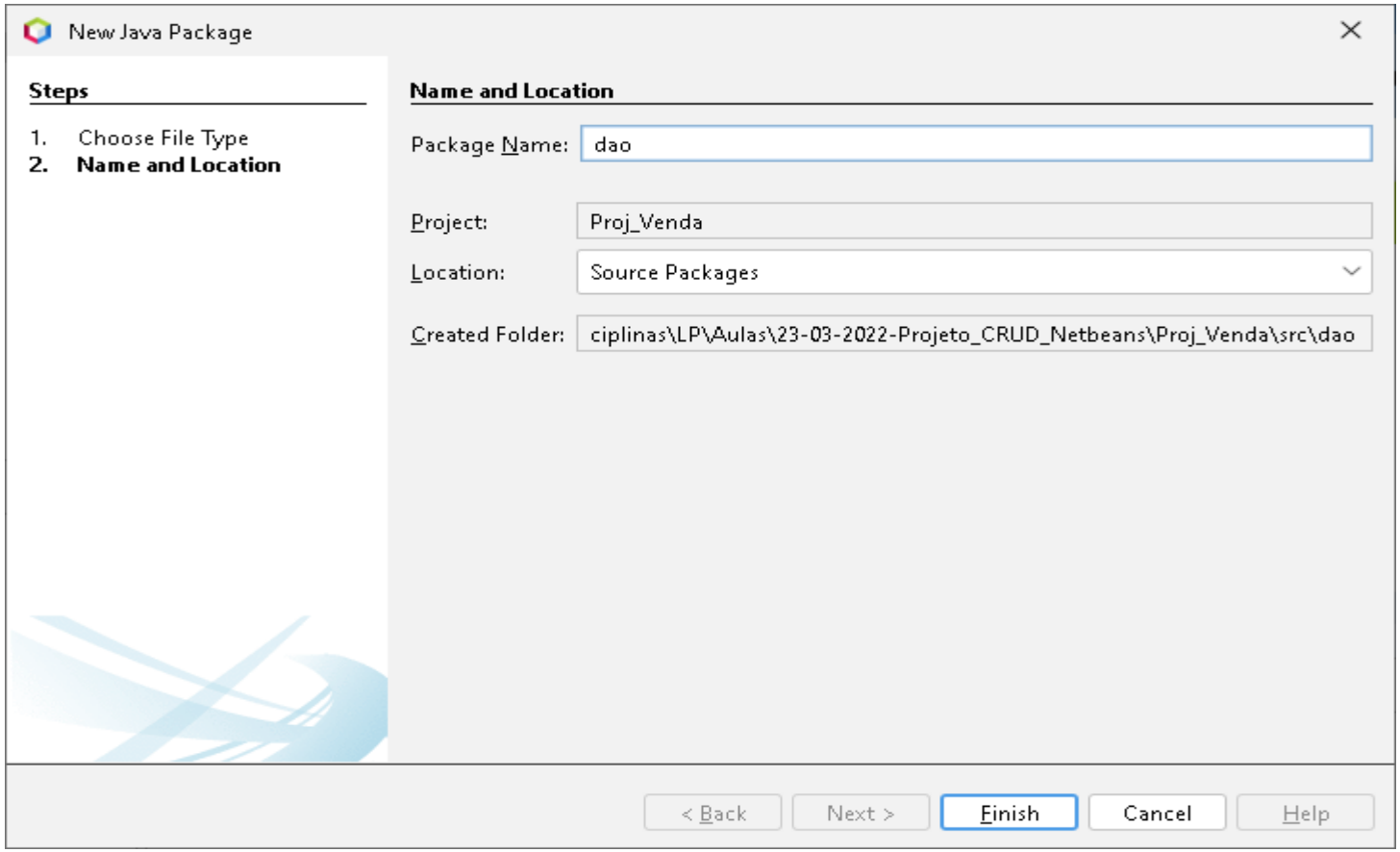
Criando Pacotes (Packages) (3)

Vamos repetir o processo de criação de pacote, criando os seguintes pacotes, além do pacote factory:**modelo**, **dao**, **gui**.

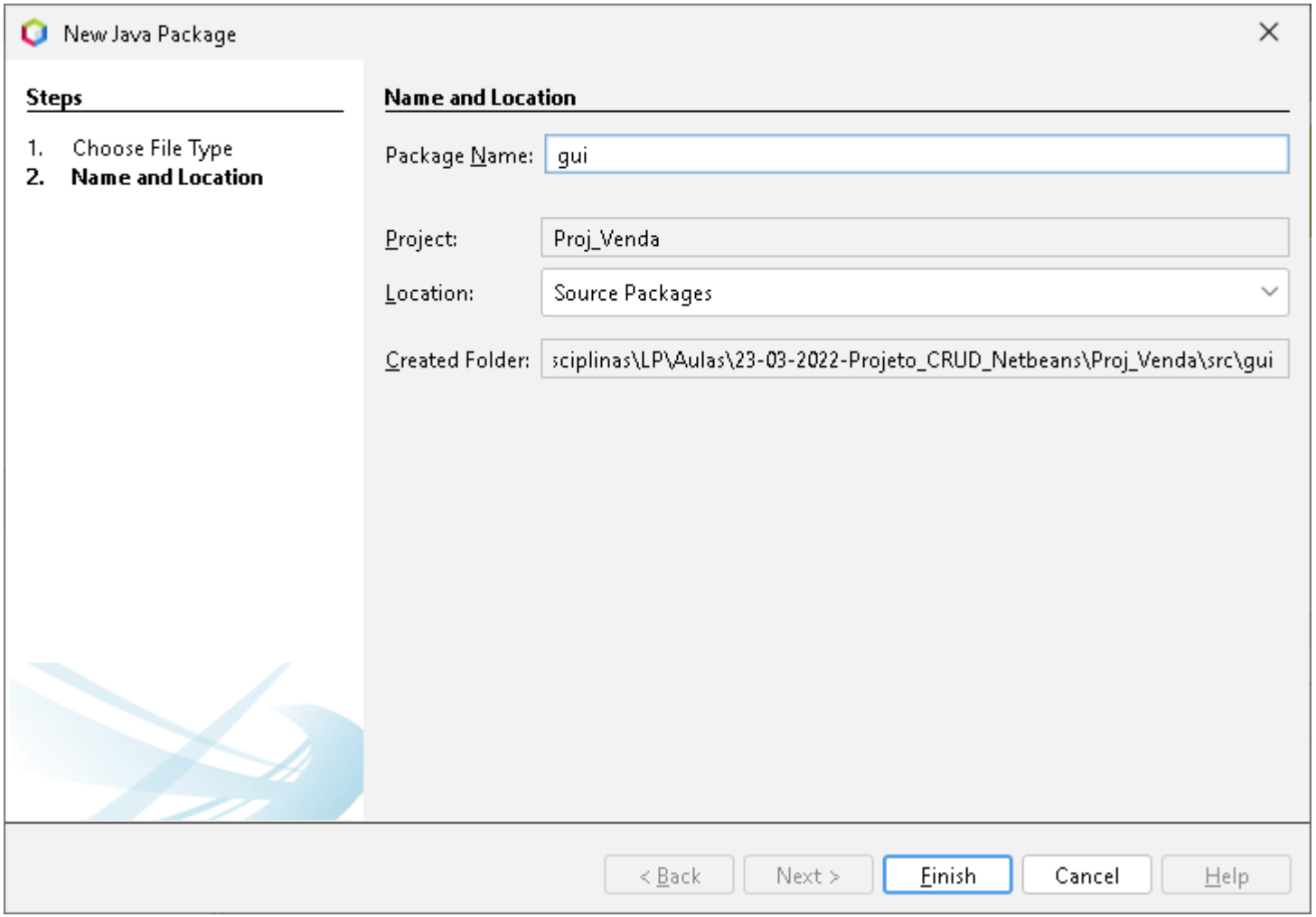
Criando o Pacote modelo



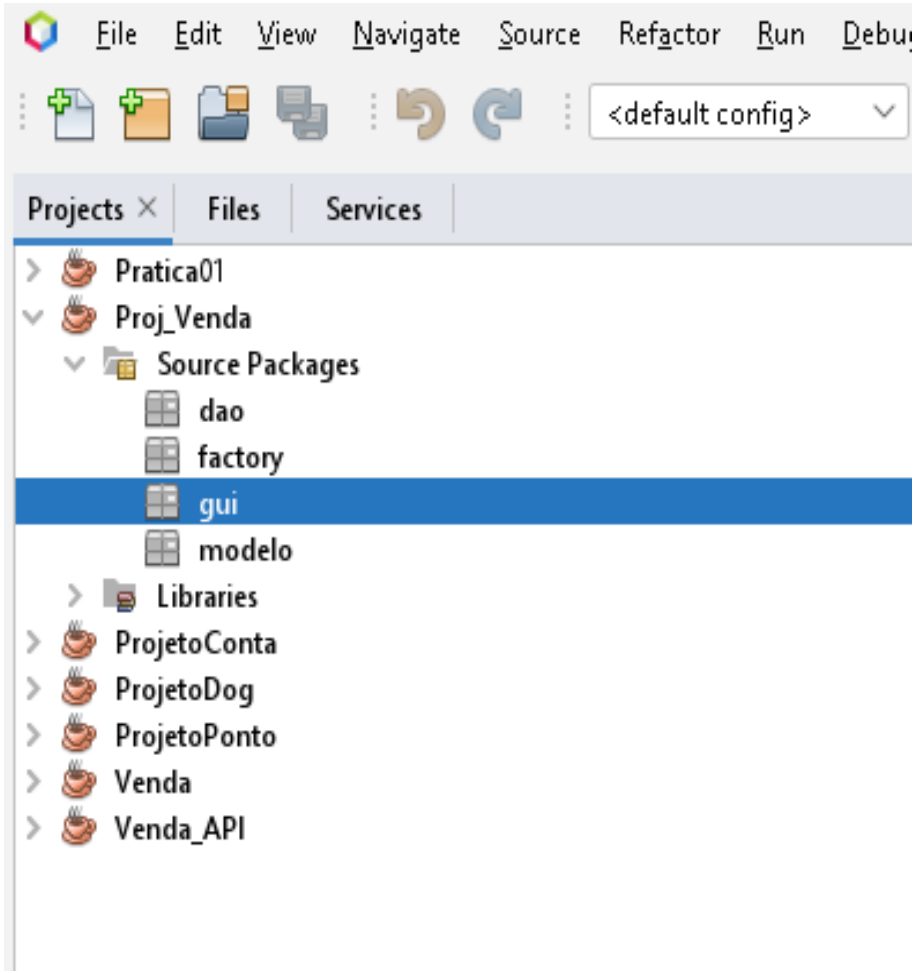
Criando o Pacote dao



Criando o pacote gui



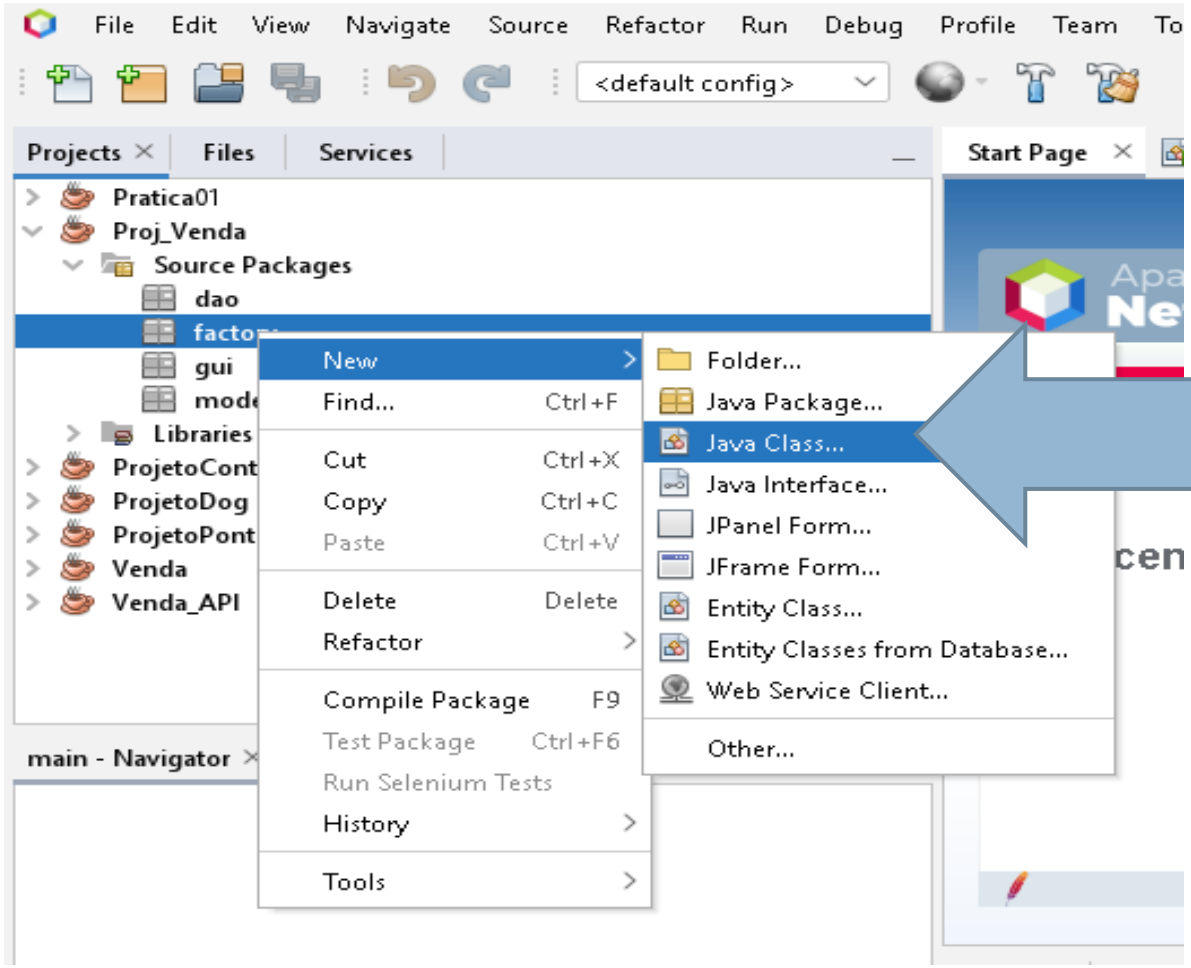
Pacotes Criados



Factory

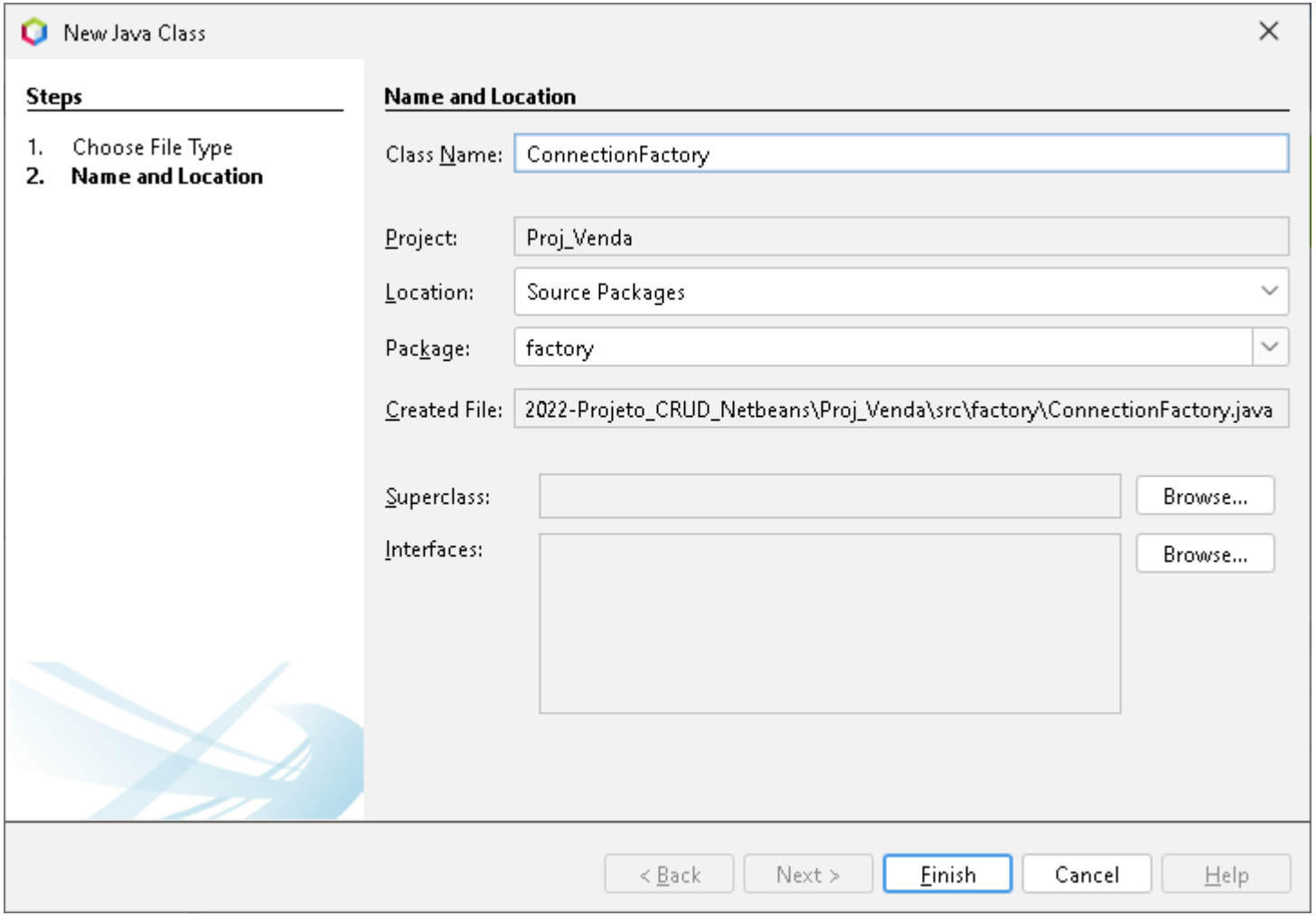
- **Factory** significa "fábrica" e **ConnectionFactory** significa fábrica de conexões.
- **Factory** será o nome do pacote e **ConnectionFactory** o nome da classe que fará a interface com o **driver JDBC de conexão a qualquer banco que desejar.**
- Por isso o nome "fábrica", pois o JDBC permite a conexão a qualquer banco: MySQL, Postgree, Oracle, SQL Server, etc., somente alterando a linha do método "getConnection".

Criando a classe ConnectionFactory (1)

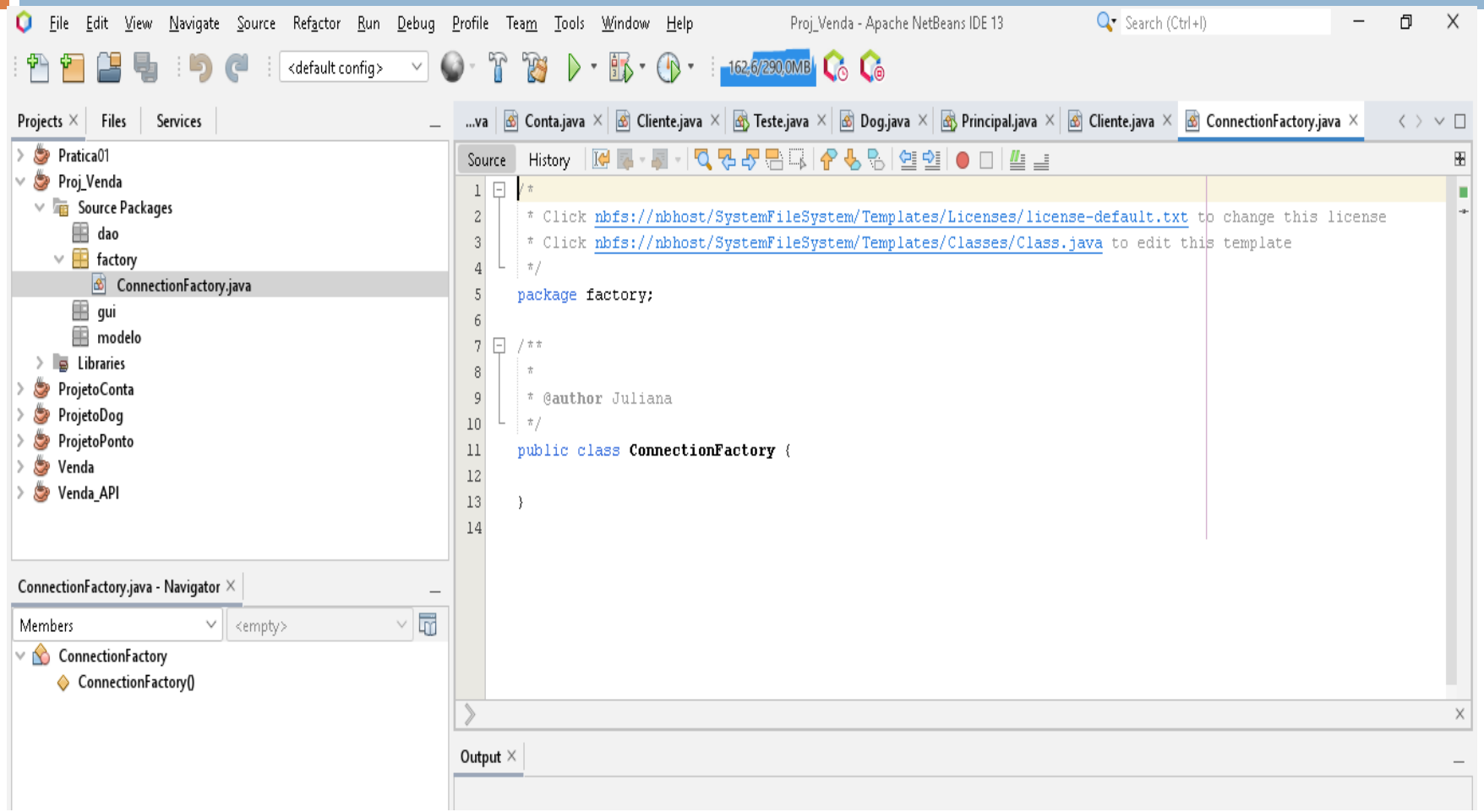


Clique com o botão direito no pacote factory ->new-> Java Classe

Criando a classe ConnectionFactory (2)



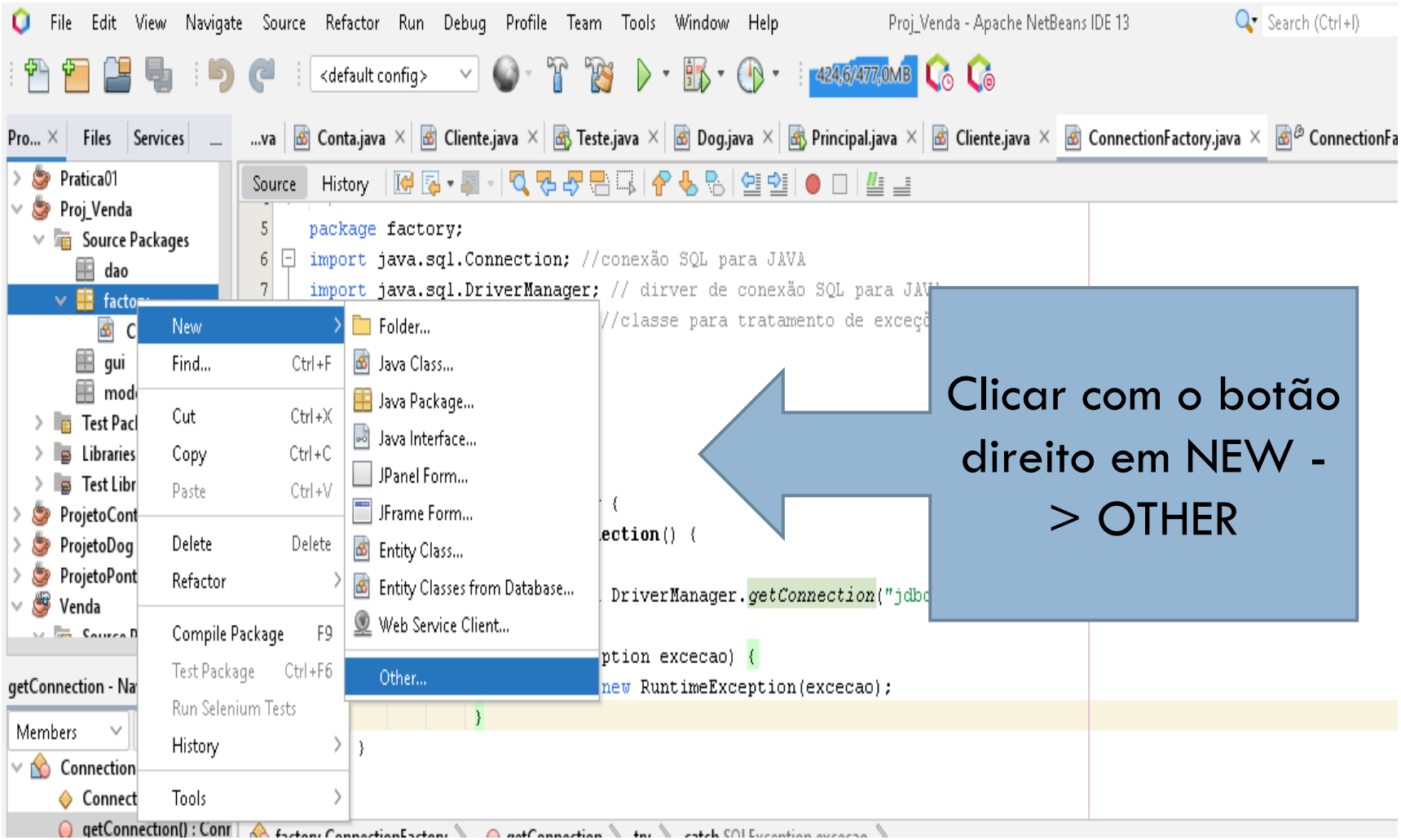
Criando a classe ConnectionFactory (3)



Criando a classe ConnectionFactory (4)

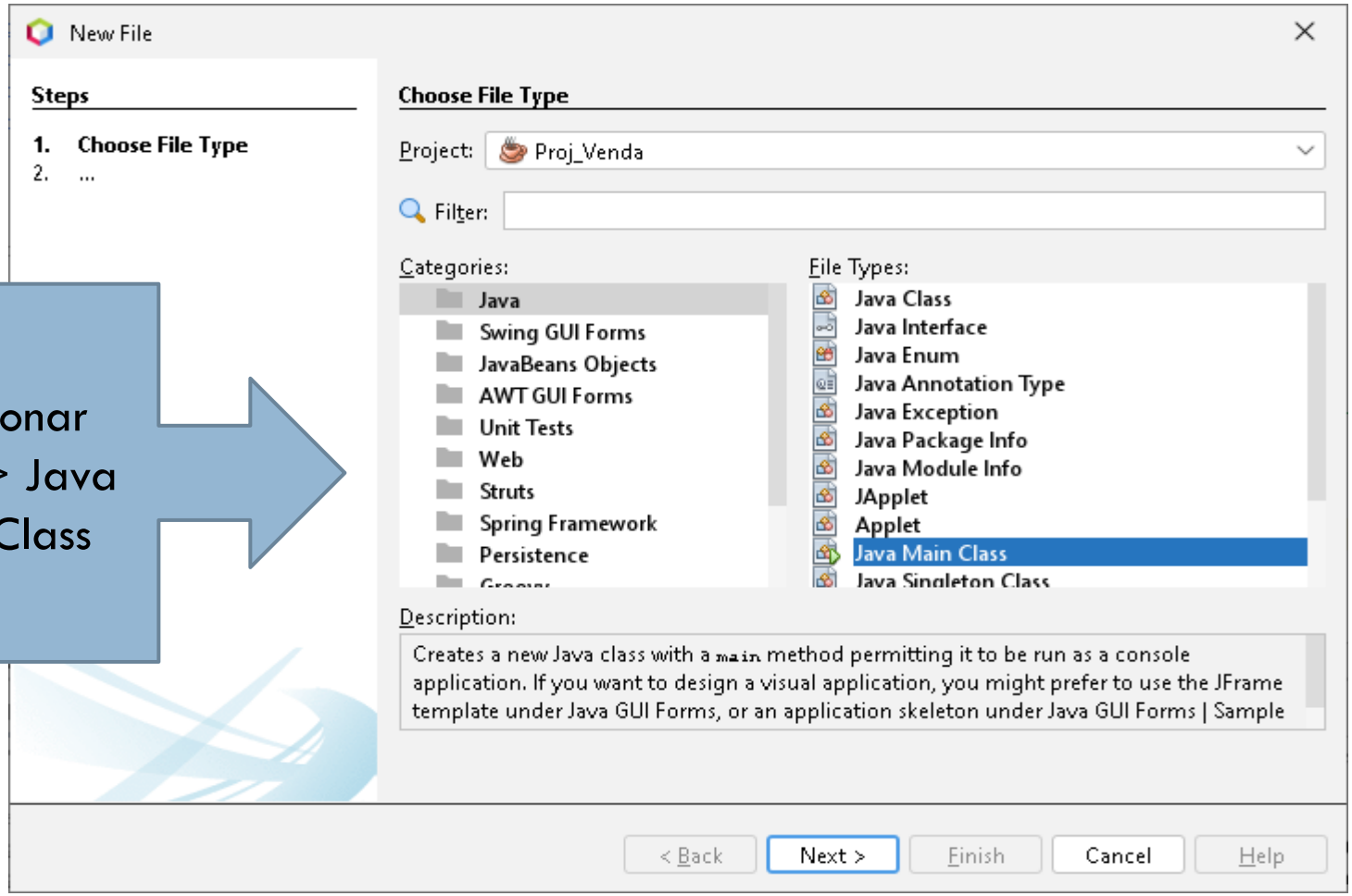

```
5   package factory;
6   import java.sql.Connection; //conexão SQL para JAVA
7   import java.sql.DriverManager; // dirver de conexão SQL para JAVA
8   import java.sql.SQLException; //classe para tratamento de exceções
9
10  /**
11   *
12   * @author Juliana
13   */
14  public class ConnectionFactory {
15      public Connection getConnection() {
16              try {
17                  return DriverManager.getConnection("jdbc:mysql://localhost/vendas","root","fatec");
18              }
19              catch(SQLException excecao) {
20                  throw new RuntimeException(excecao);
21              }
22      }
```

Criando a classe Main TestaConexao (1)

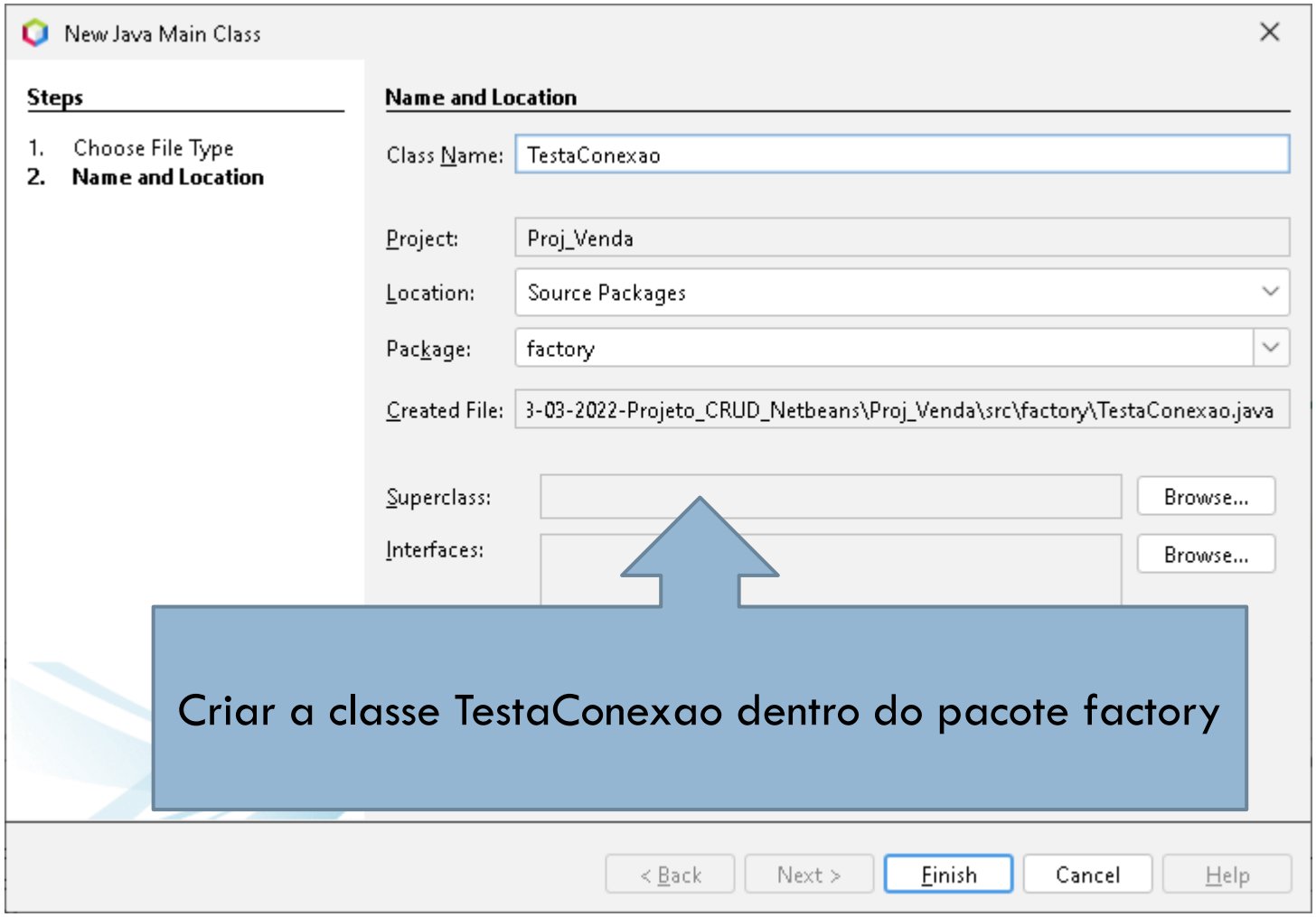


Criando a classe Main TestaConexao (2)

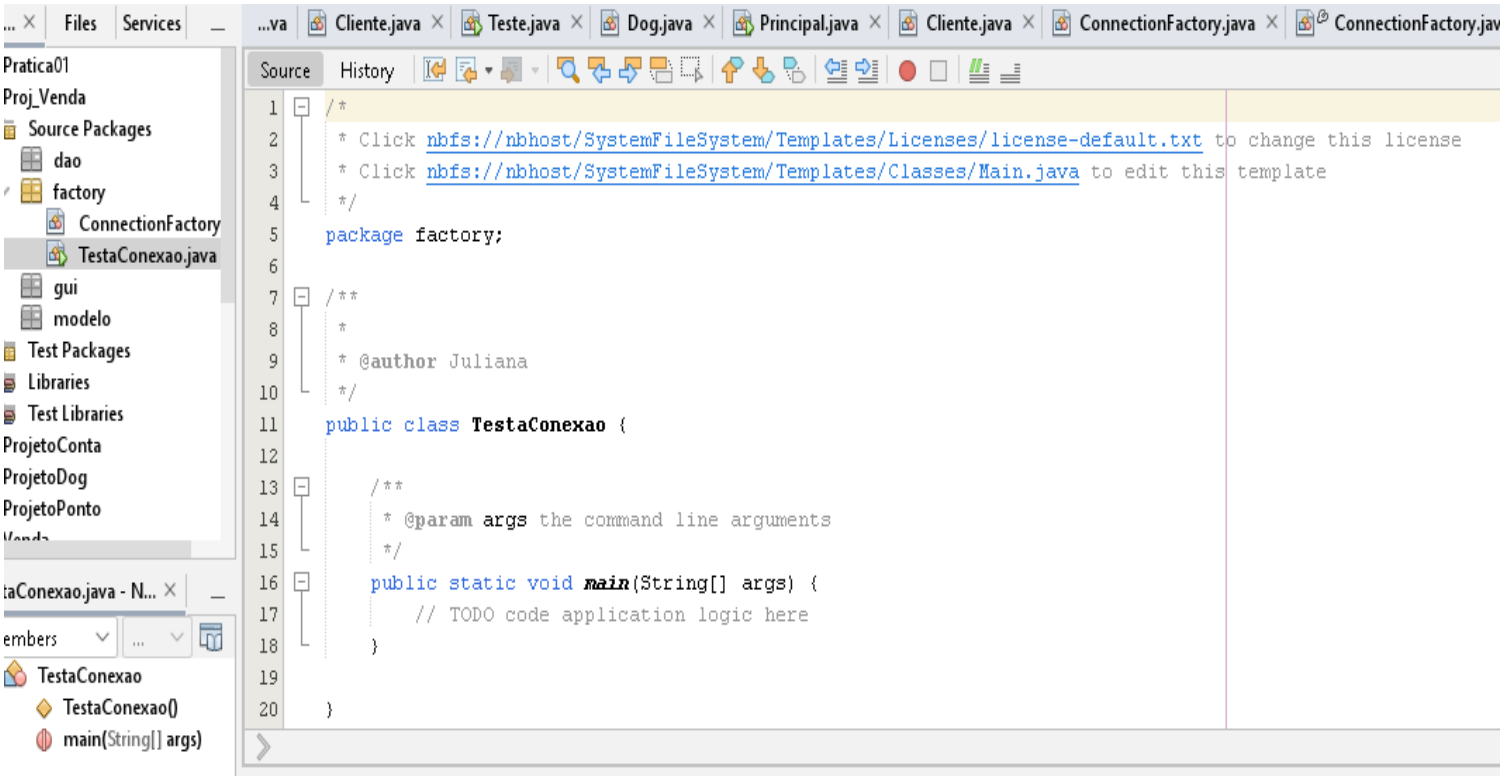
Selecionar Java -> Java Main Class



Criando a classe Main TestaConexao (3)



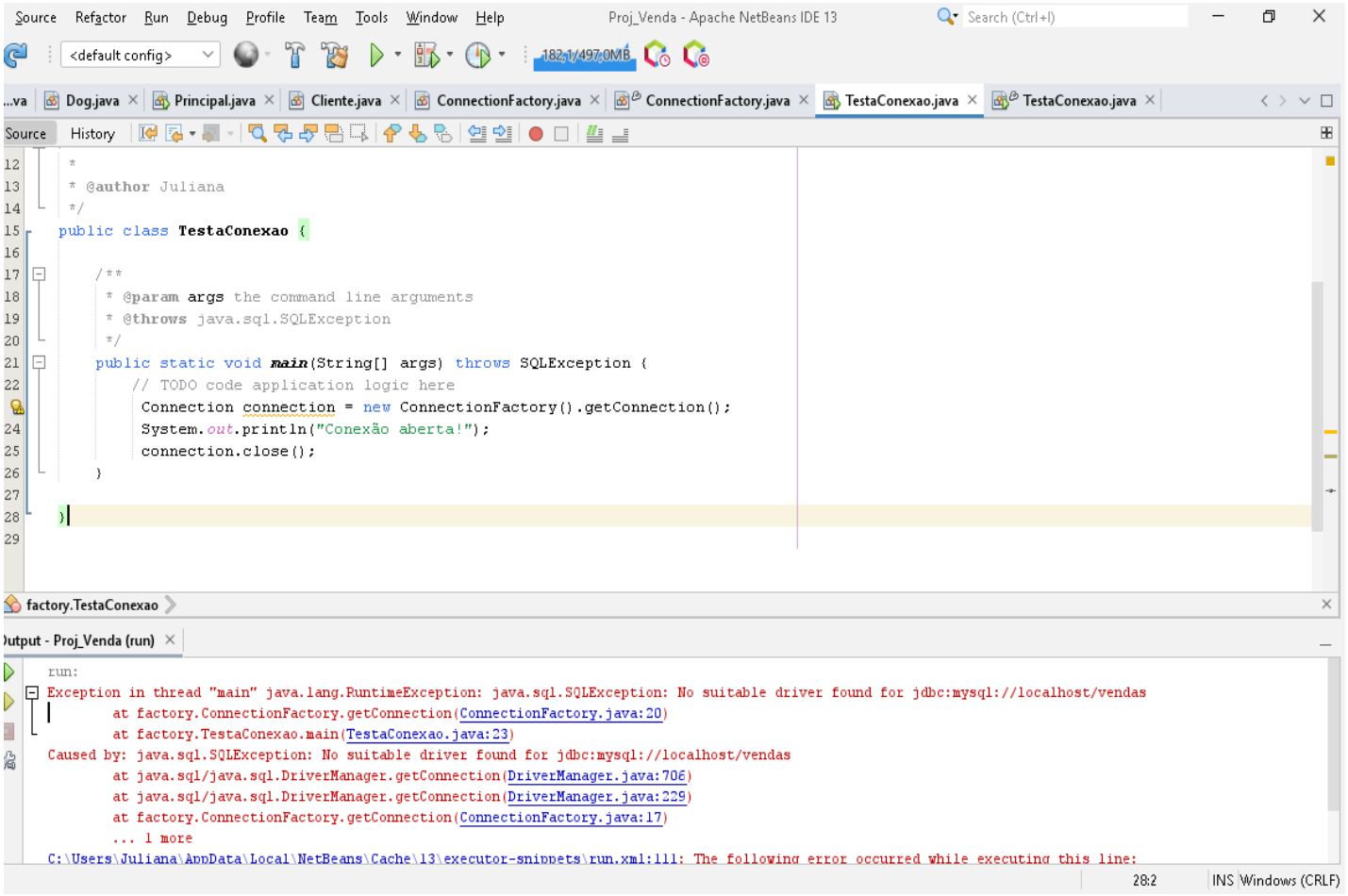
Criando a classe Main TestaConexao (4)



Criando a classe Main TestaConexao (5)


```
15  public class TestaConexao {
16
17      /**
18       * @param args the command line arguments
19       * @throws java.sql.SQLException
20       */
21      public static void main(String[] args) throws SQLException {
22          // TODO code application logic here
            Connection connection = new ConnectionFactory().getConnection();
24          System.out.println("Conexão aberta!");
25          connection.close();
26      }
27
28  }
29
```

Executando a Classe TestaConexão



Executando o Projeto

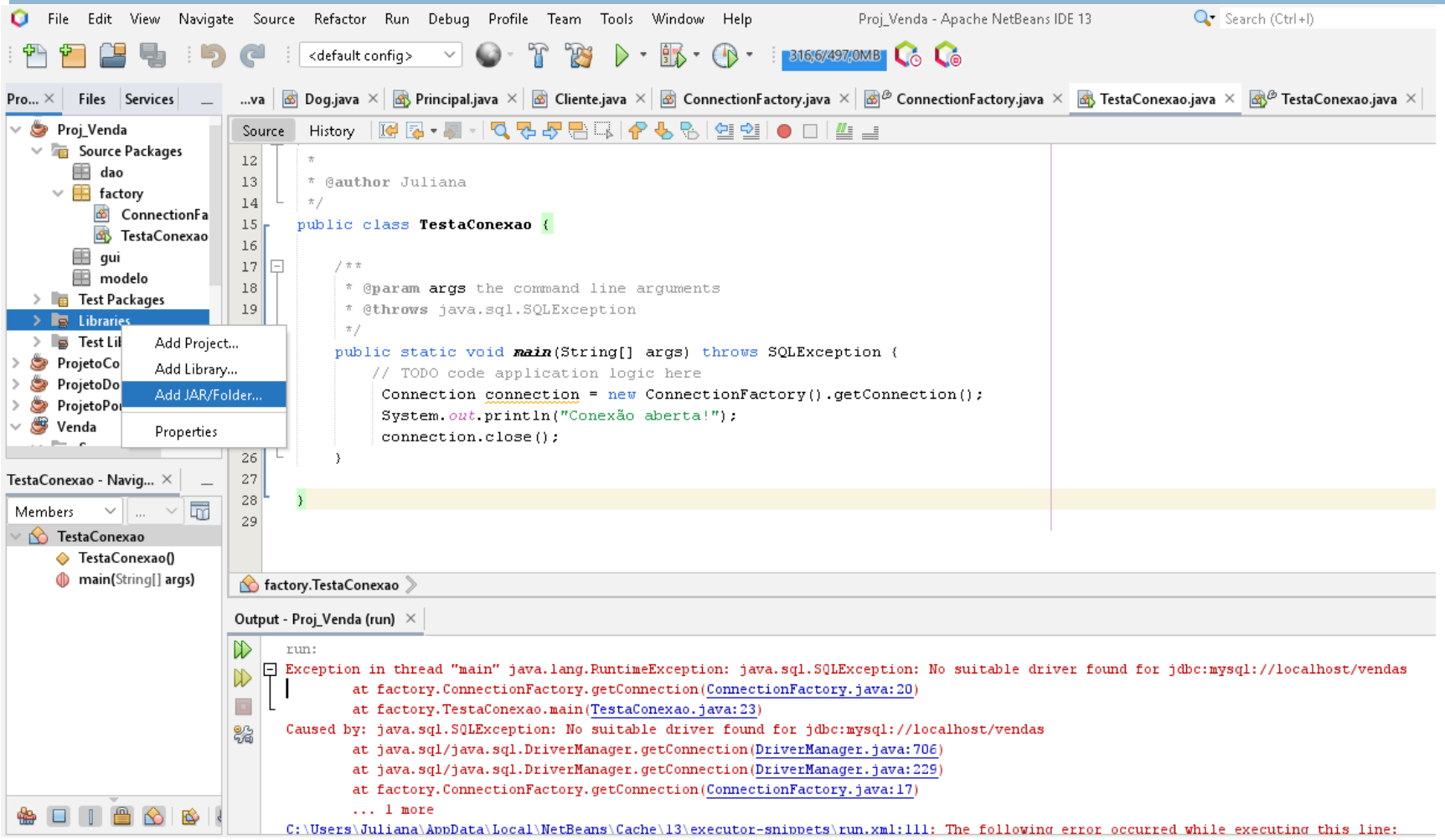
- Ao executar o projeto uma mensagem é exibida no console. Esta mensagem de erro significa ausência do driver JDBC.

- OBS: Quando instalar o MYSQL selecionar Connector/J 8.0.28

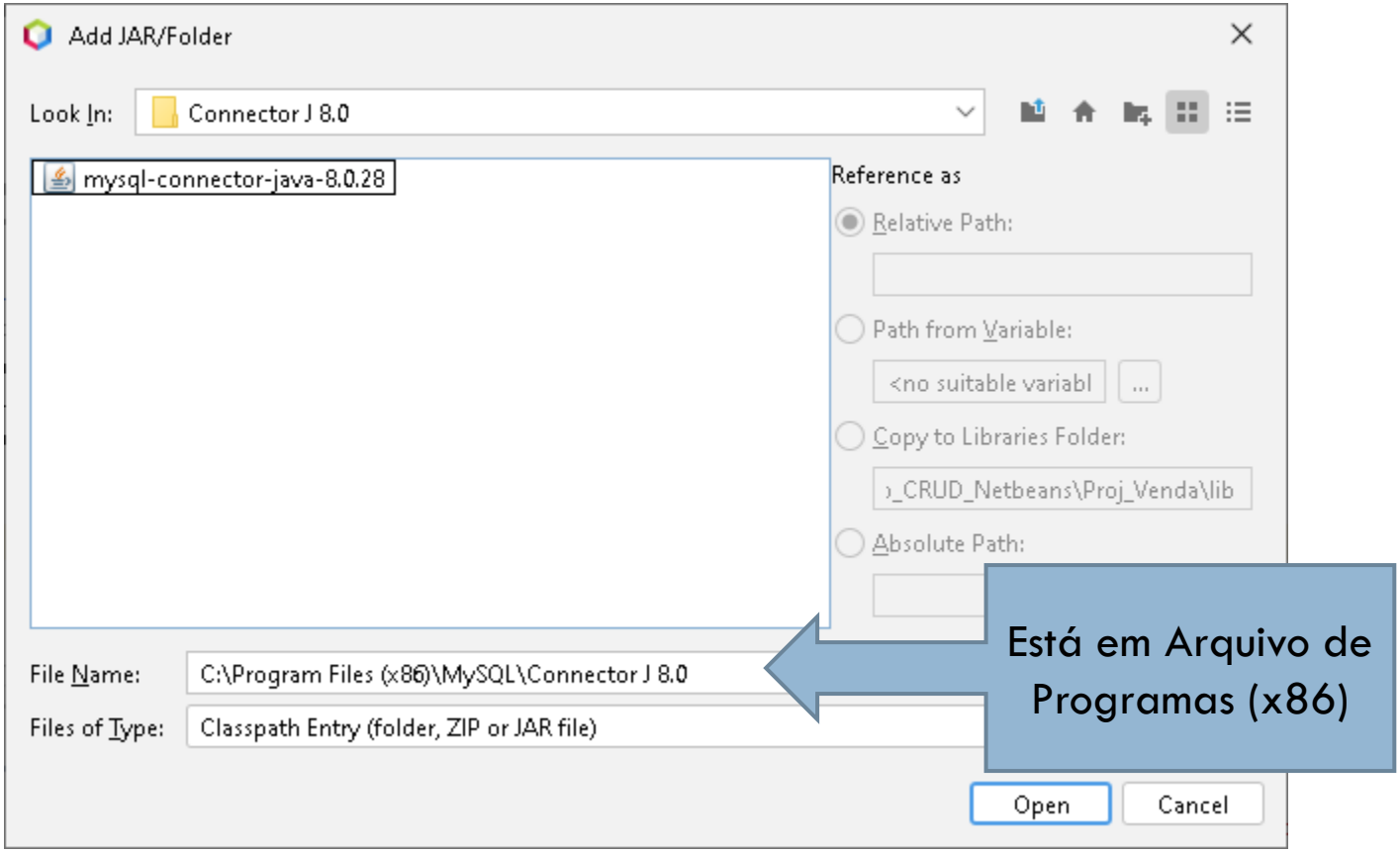
- Download do drive :

https://www.mysql.com/products/connector/

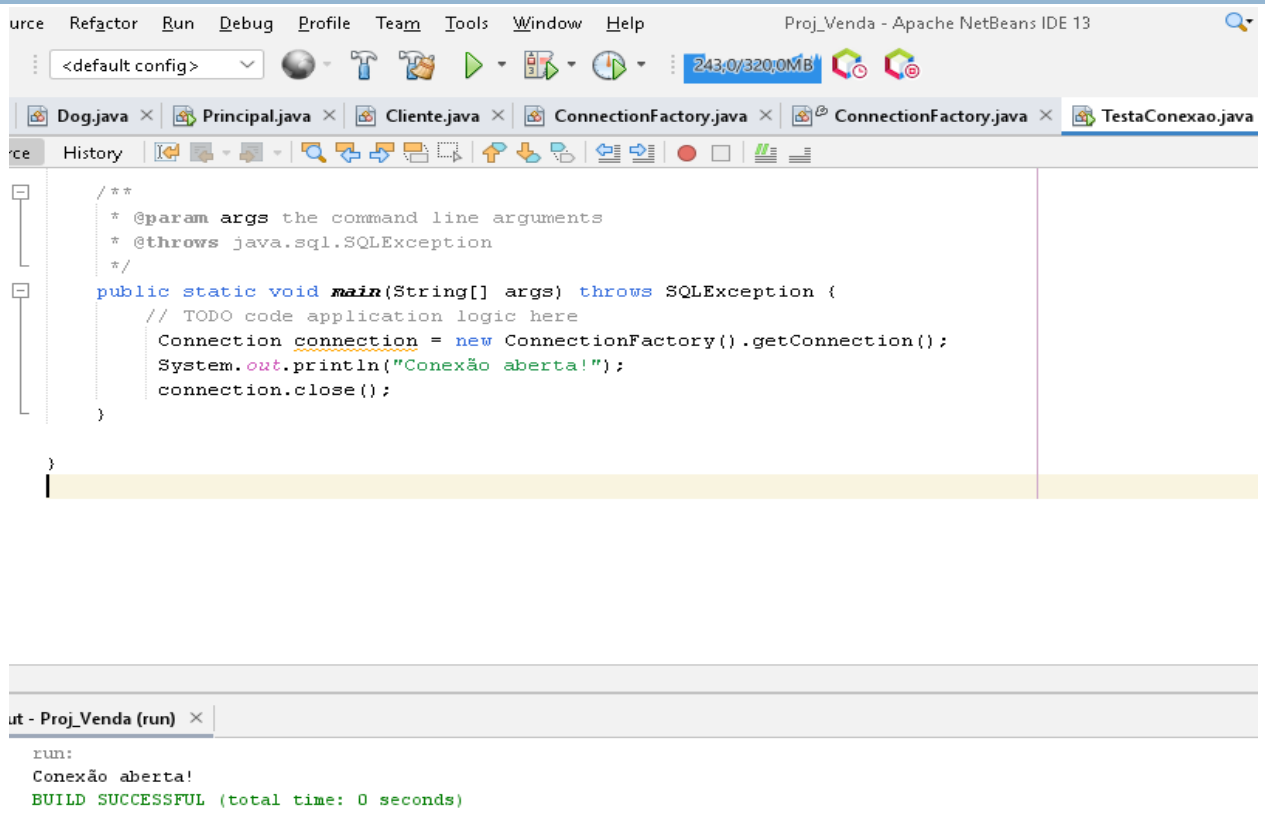
Adicionando o drive Connector/J 8.0.28 (1)



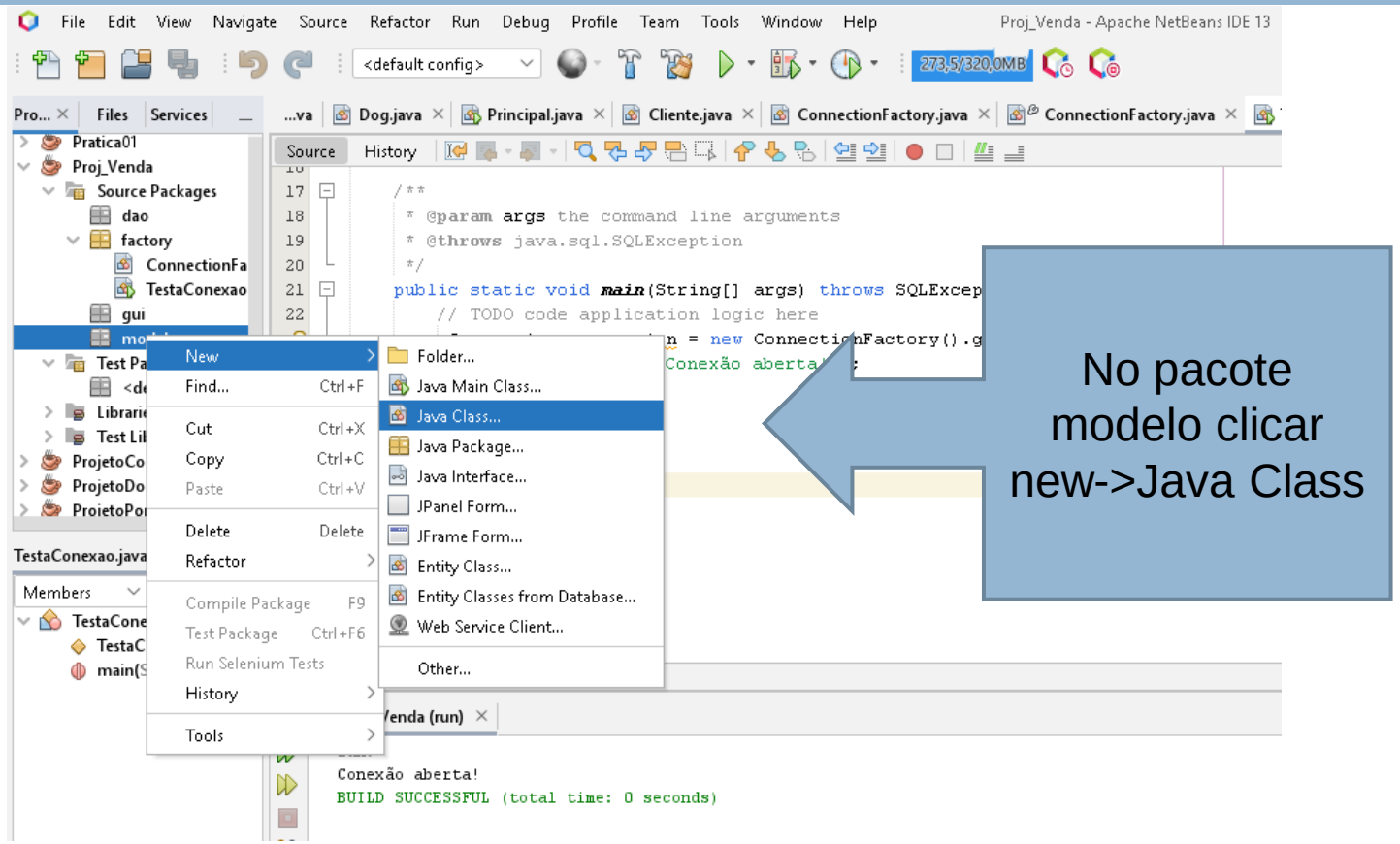
Adicionando o drive Connector/J 8.0.28 (2)



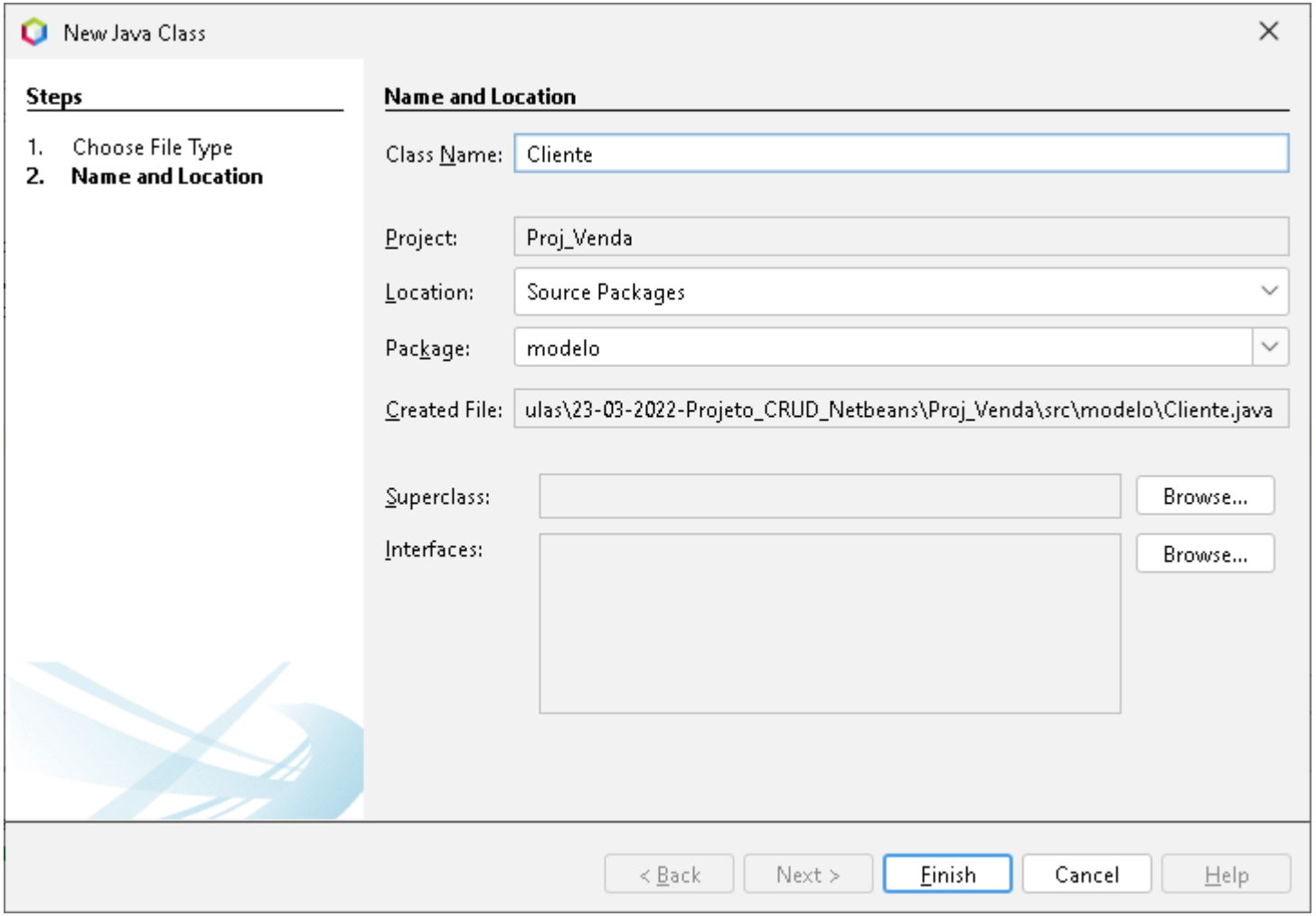
Executando o Projeto



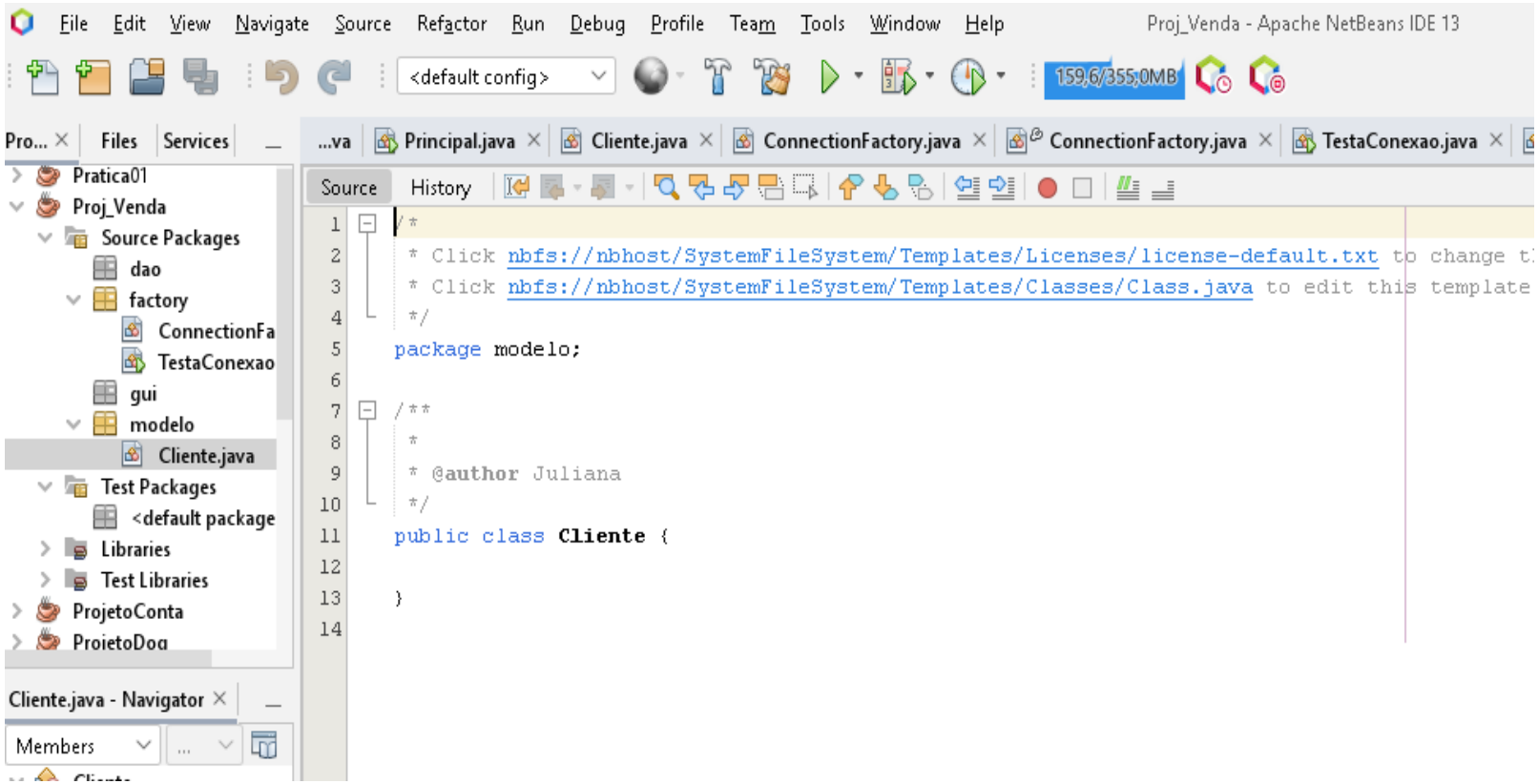
Criando a classe Cliente no Pacote Modelo (1)



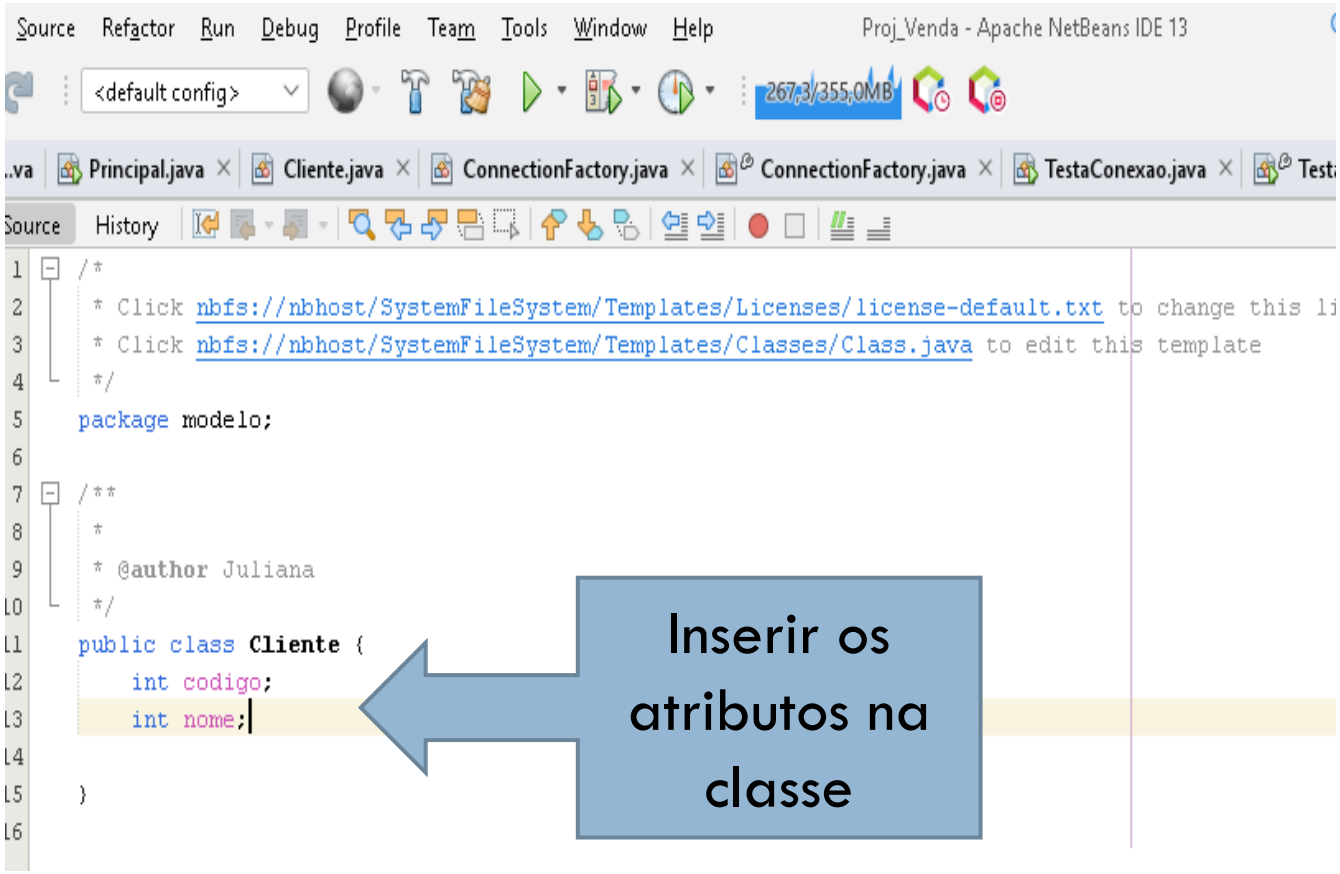
Criando a classe Cliente no Pacote Modelo (2)



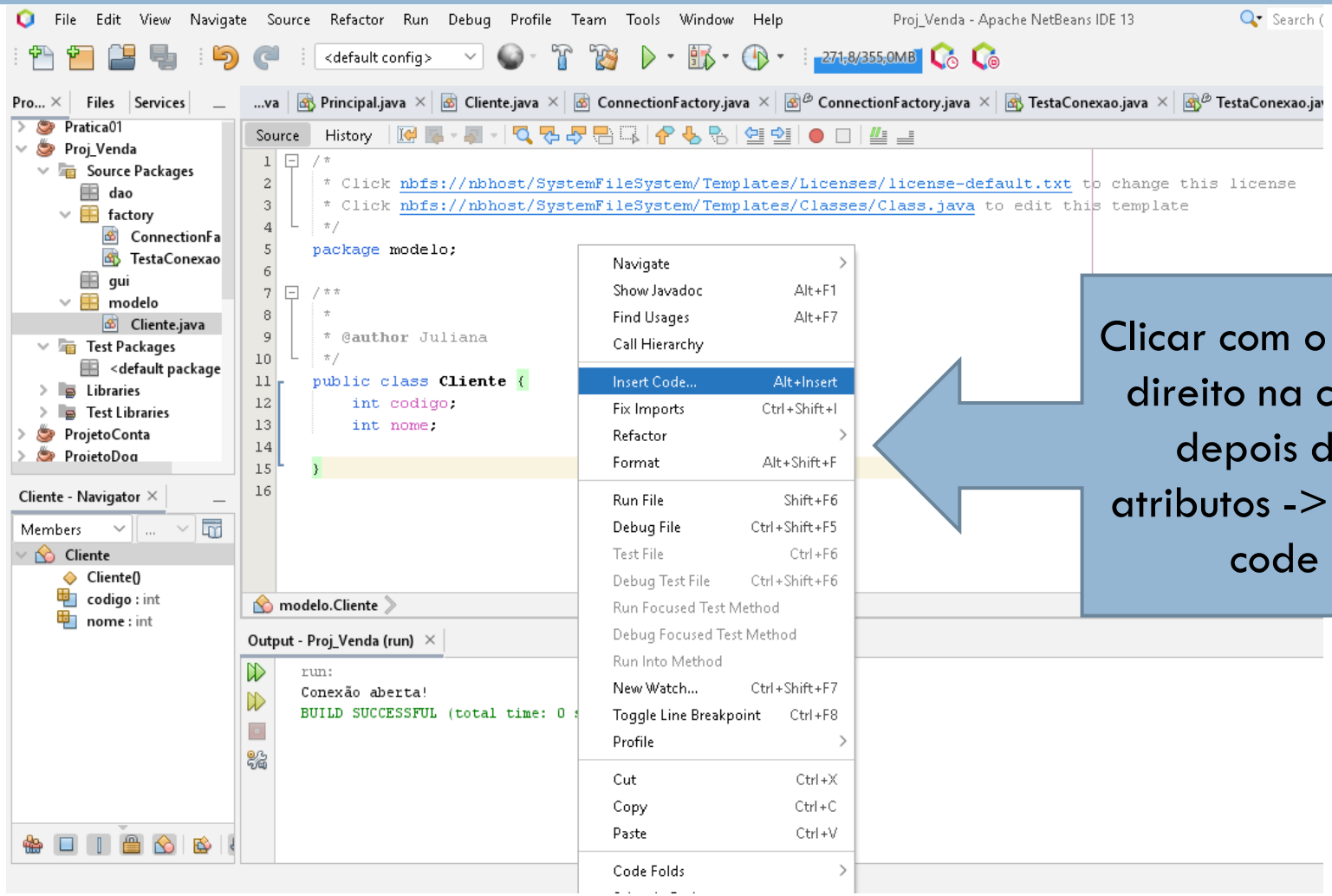
Classe Cliente



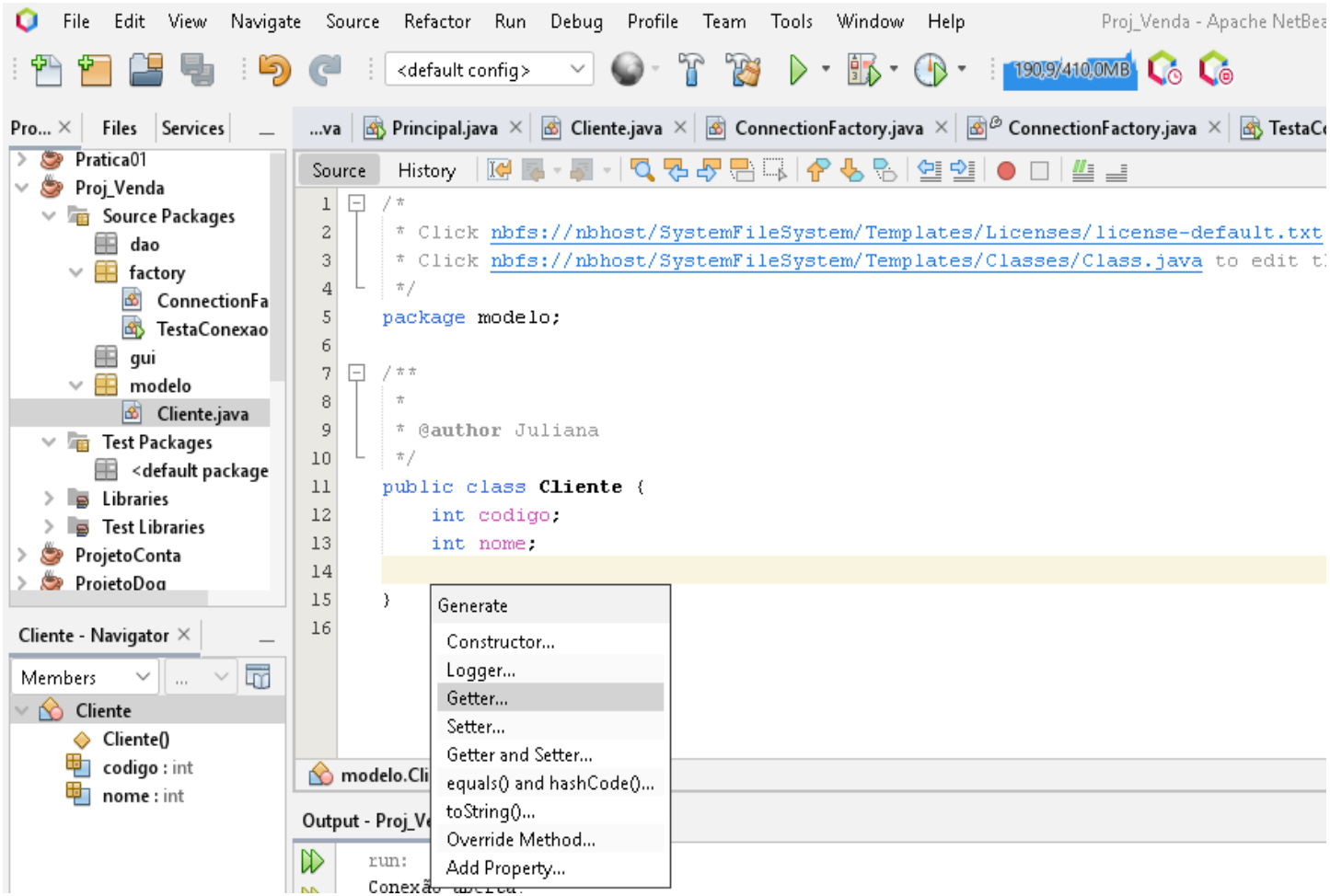
Classe Cliente (Atributos)



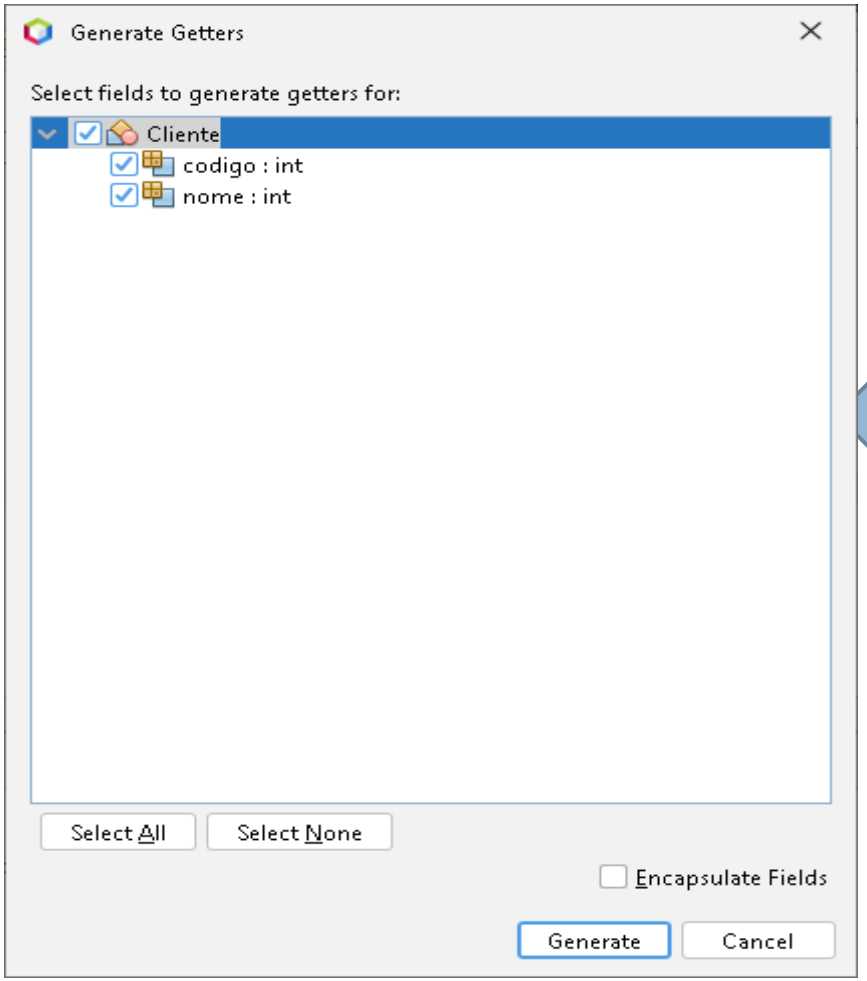
Classe Clientes (métodos)



Classe Clientes (Get)



Classe Clientes (Get)

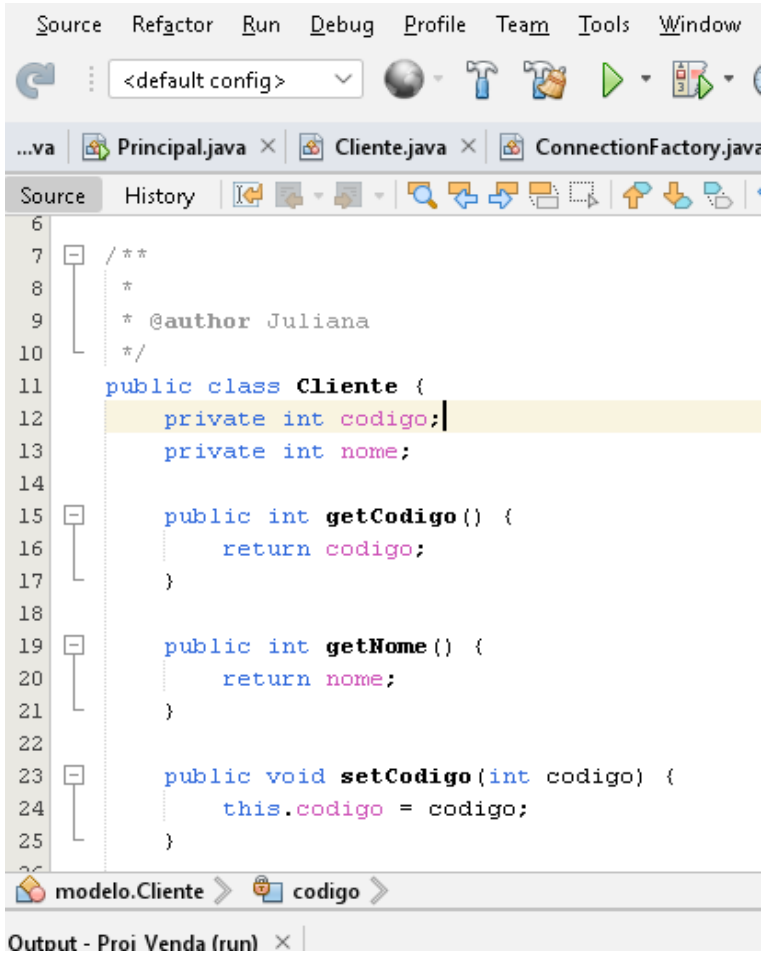


Selecionar os atributos para criação do métodos getters, clicar em encapsular campos e geral.

Classe Cliente (Setters)

- Repedir o processo anterior para gerar os métodos setters.

Classe Cliente

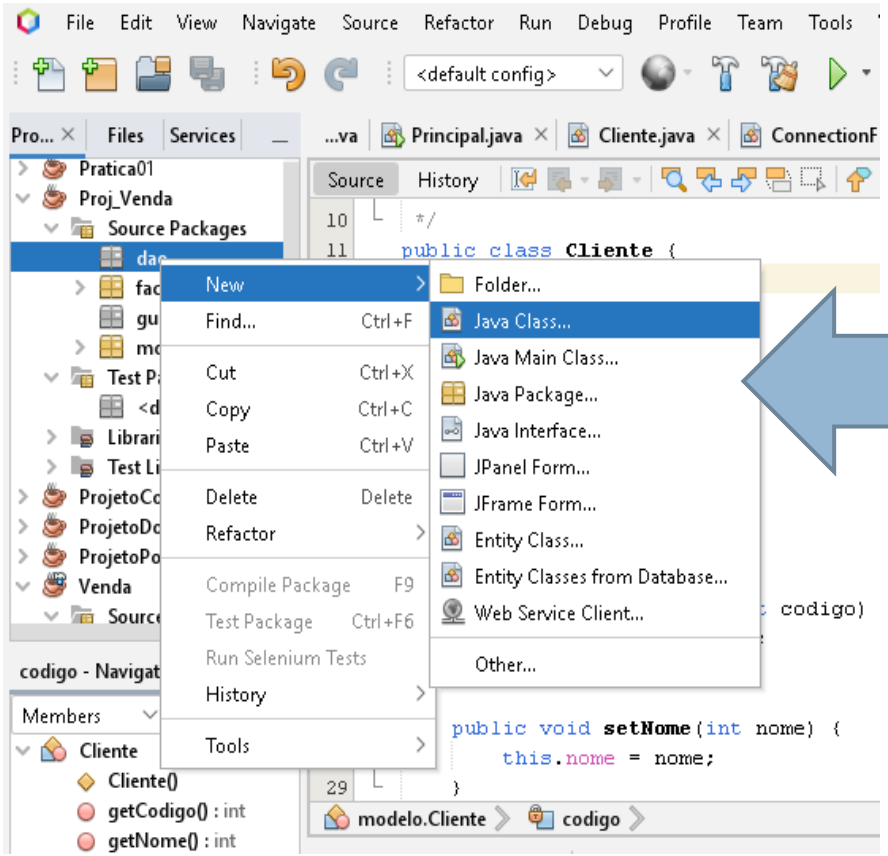


Classe ClienteDAO

- Crie no **pacote DAO** a classe ClienteoDAO: **dao > Novo > Classe Java > UsuarioDAO > Finalizar**.

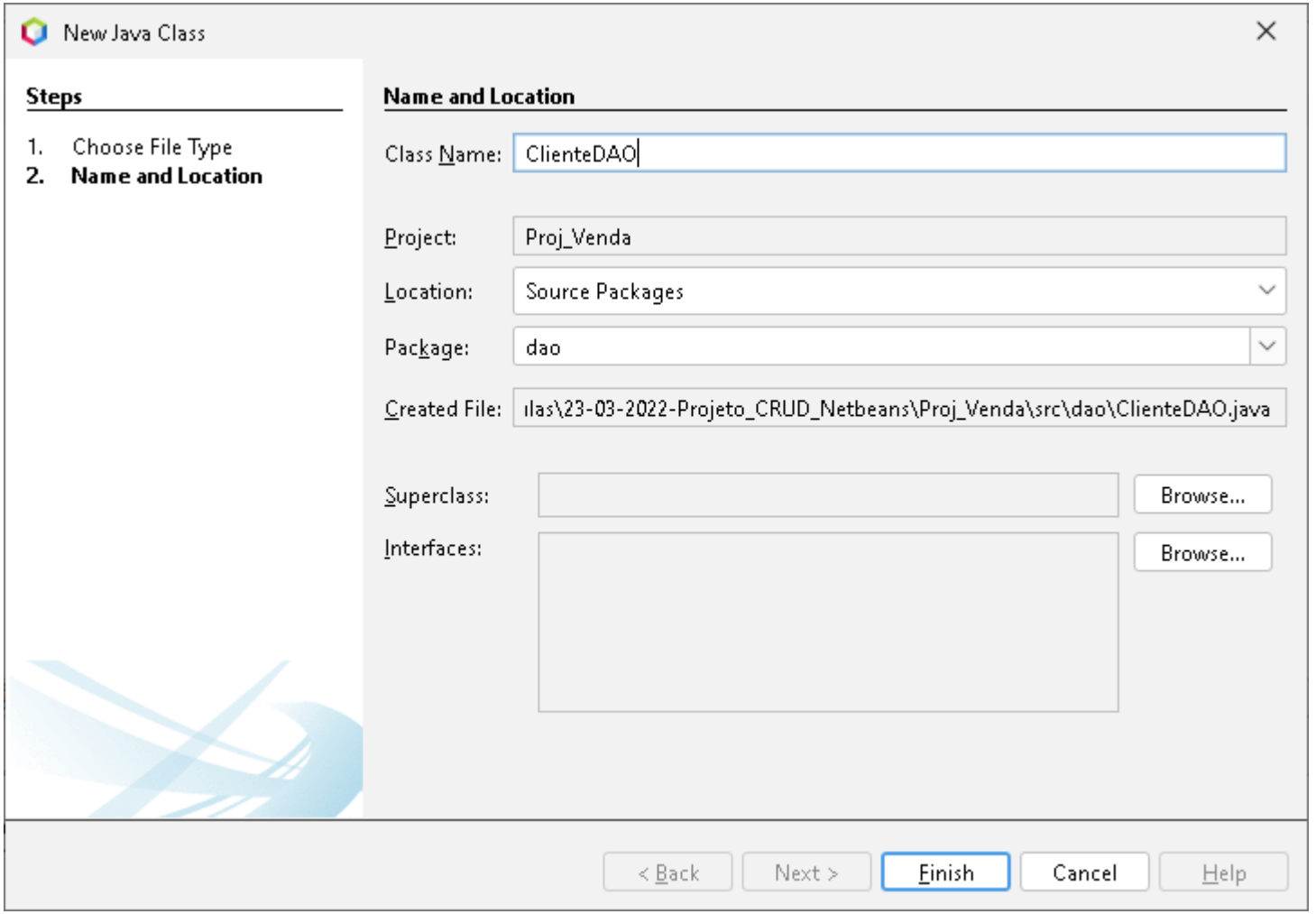
- Nesse pacote ficam as classes que são responsáveis pelo **CRUD** (Create, Read, Update, Delete - ou - Criar, Consultar, Alterar, Deletar), isto é, dados de persistência.

Classe ClienteDAO

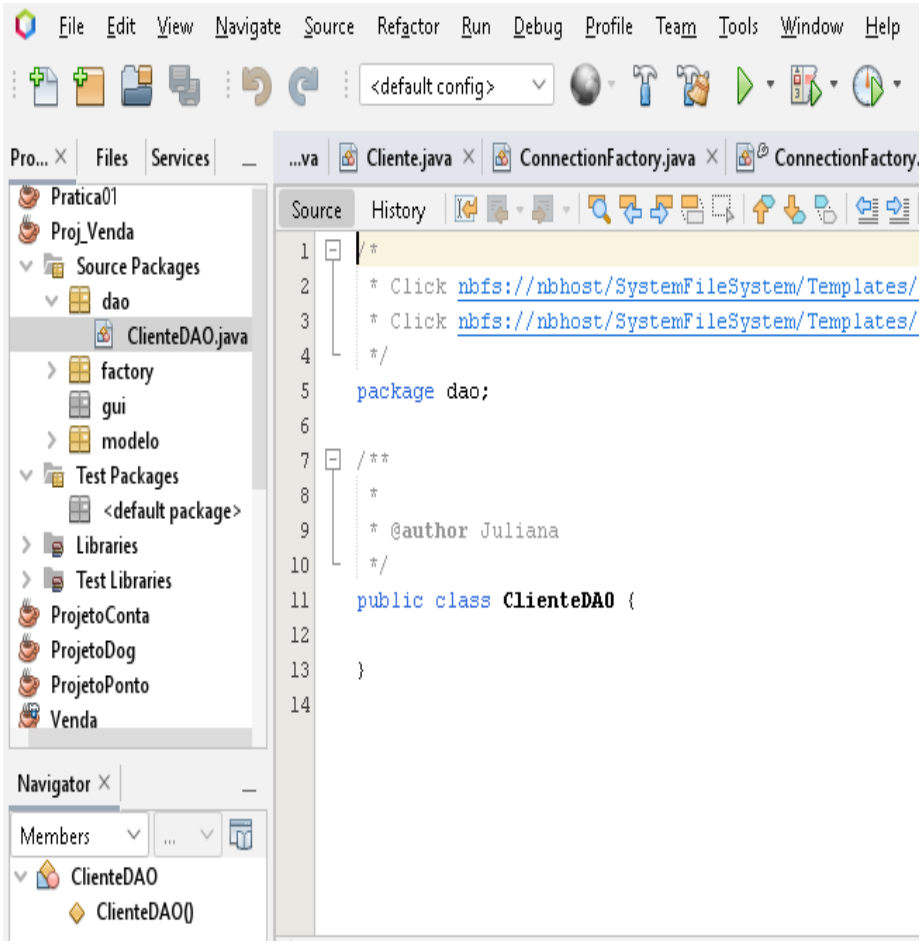


No pacote dao clicar new->Java Class

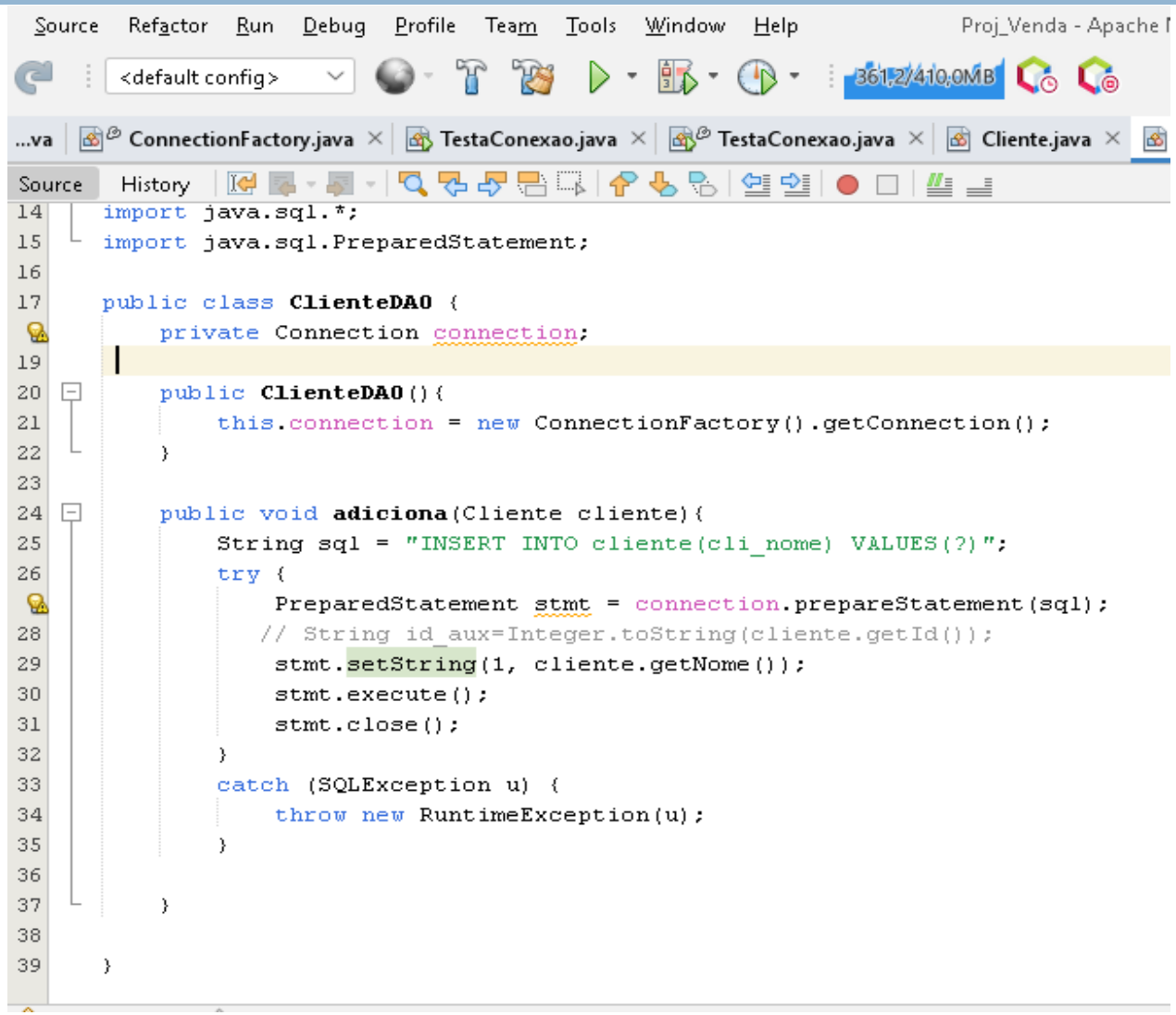
Classe ClienteDAO



Classe ClienteDAO



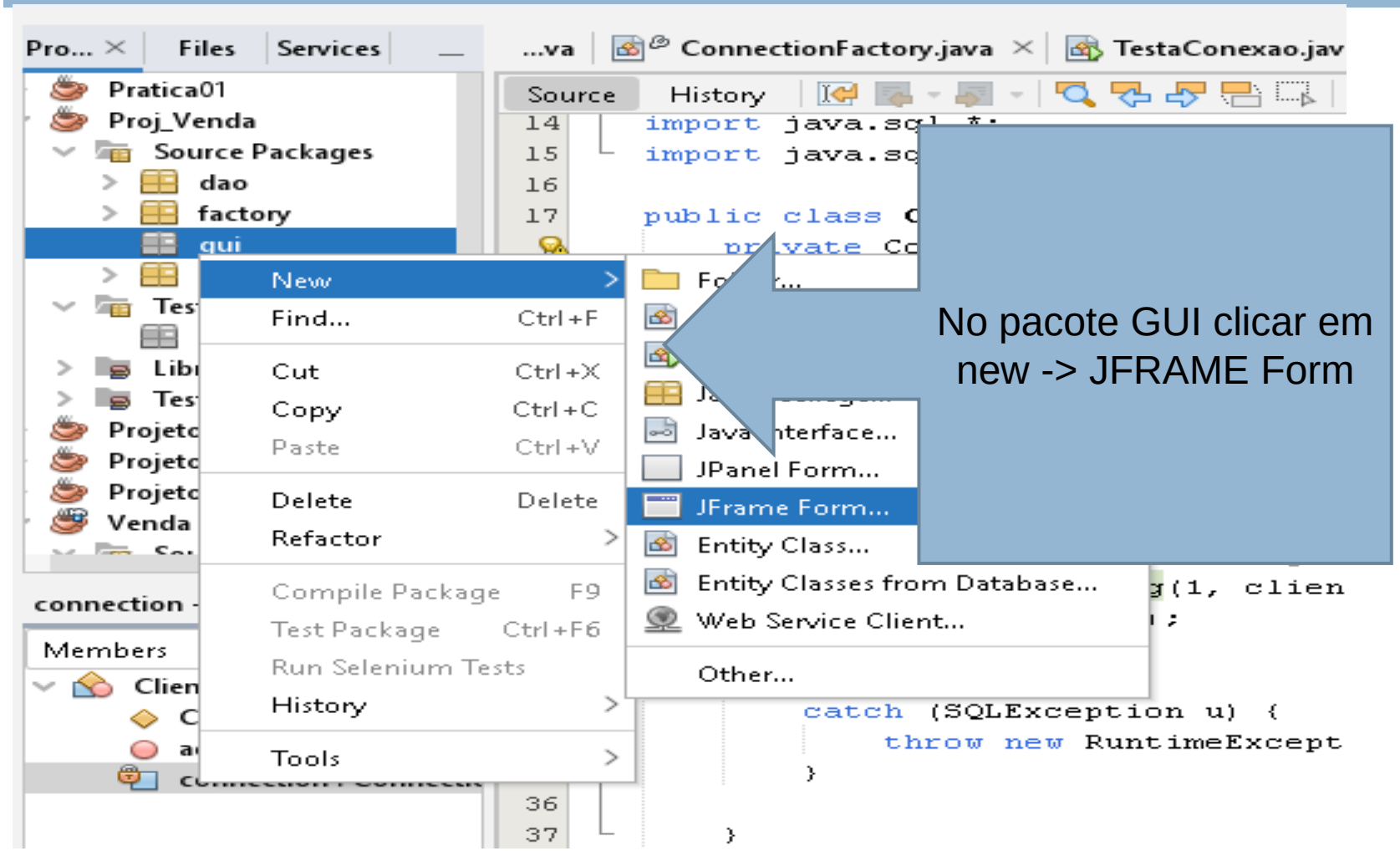
Classe ClienteDAO



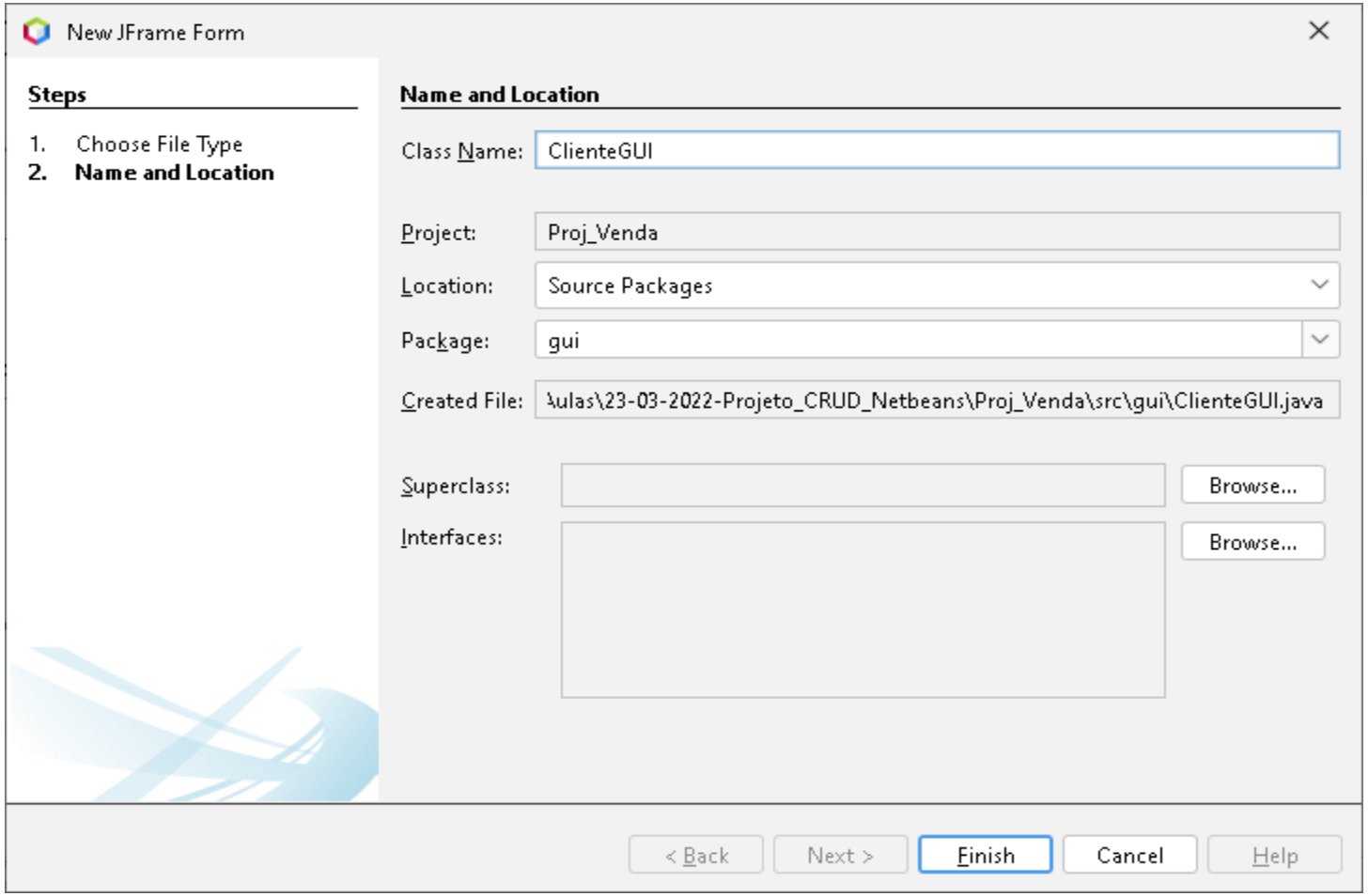
GUI (Graphical User Interface ou Interface Gráfica de Usuário)

- A aplicação back-end está finalizada.
- Vamos fazer o front-end, isto é, a interface de usuário - GUI, ou seja, a classe que será responsável pela interação com o usuário.

Classe ClienteGUI

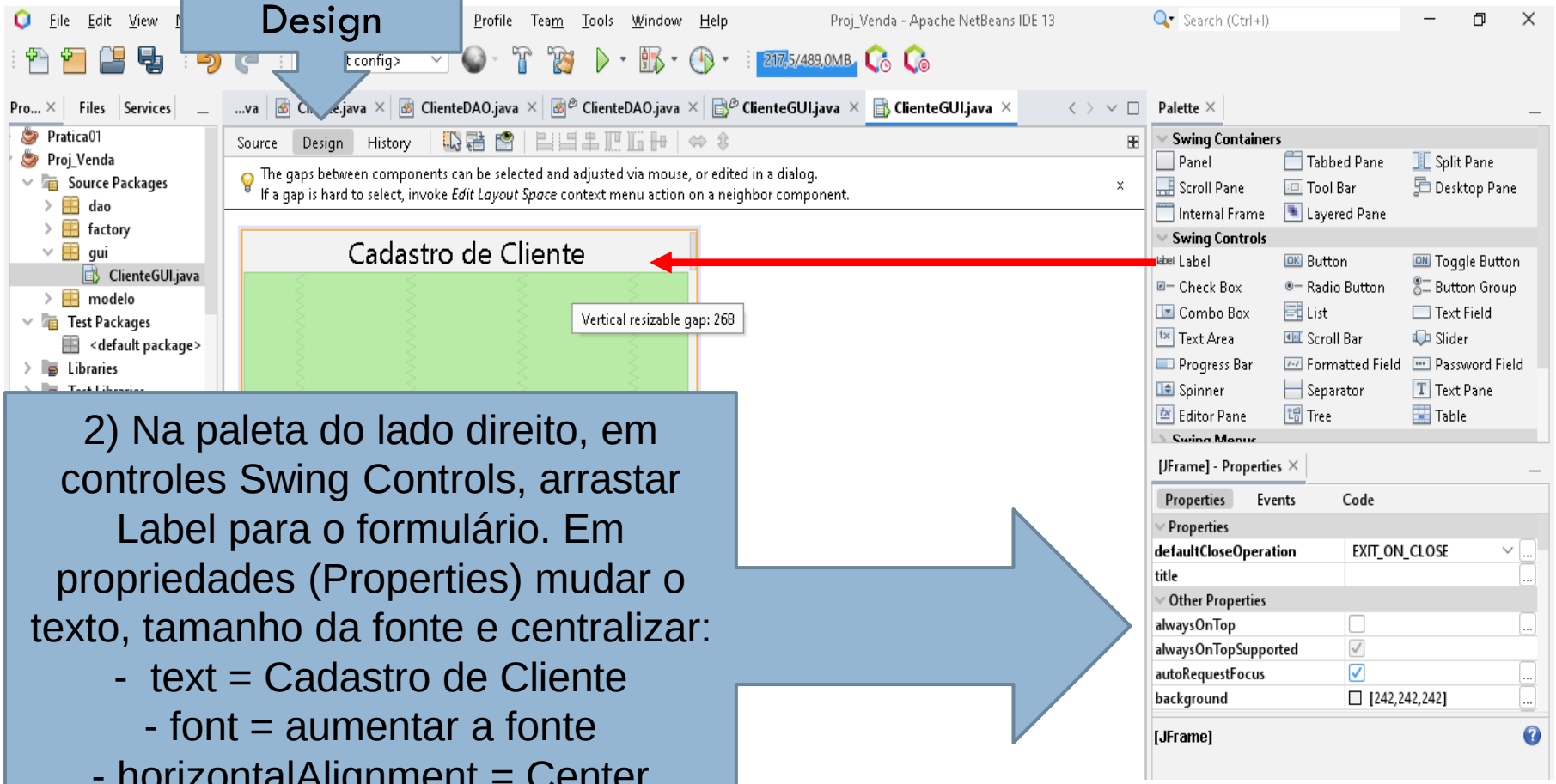


Classe ClienteGUI



Inserindo Label

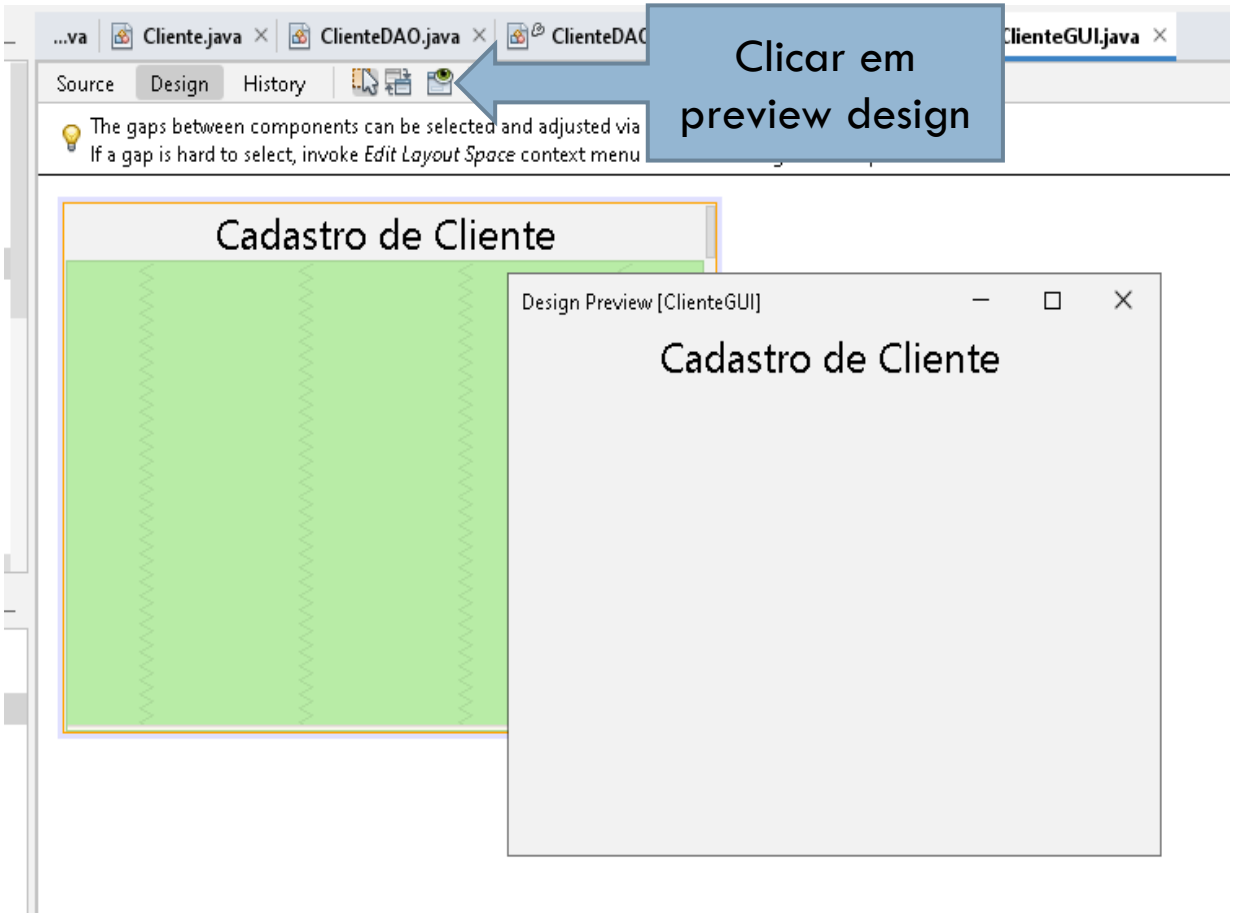
1) Selecionar Design



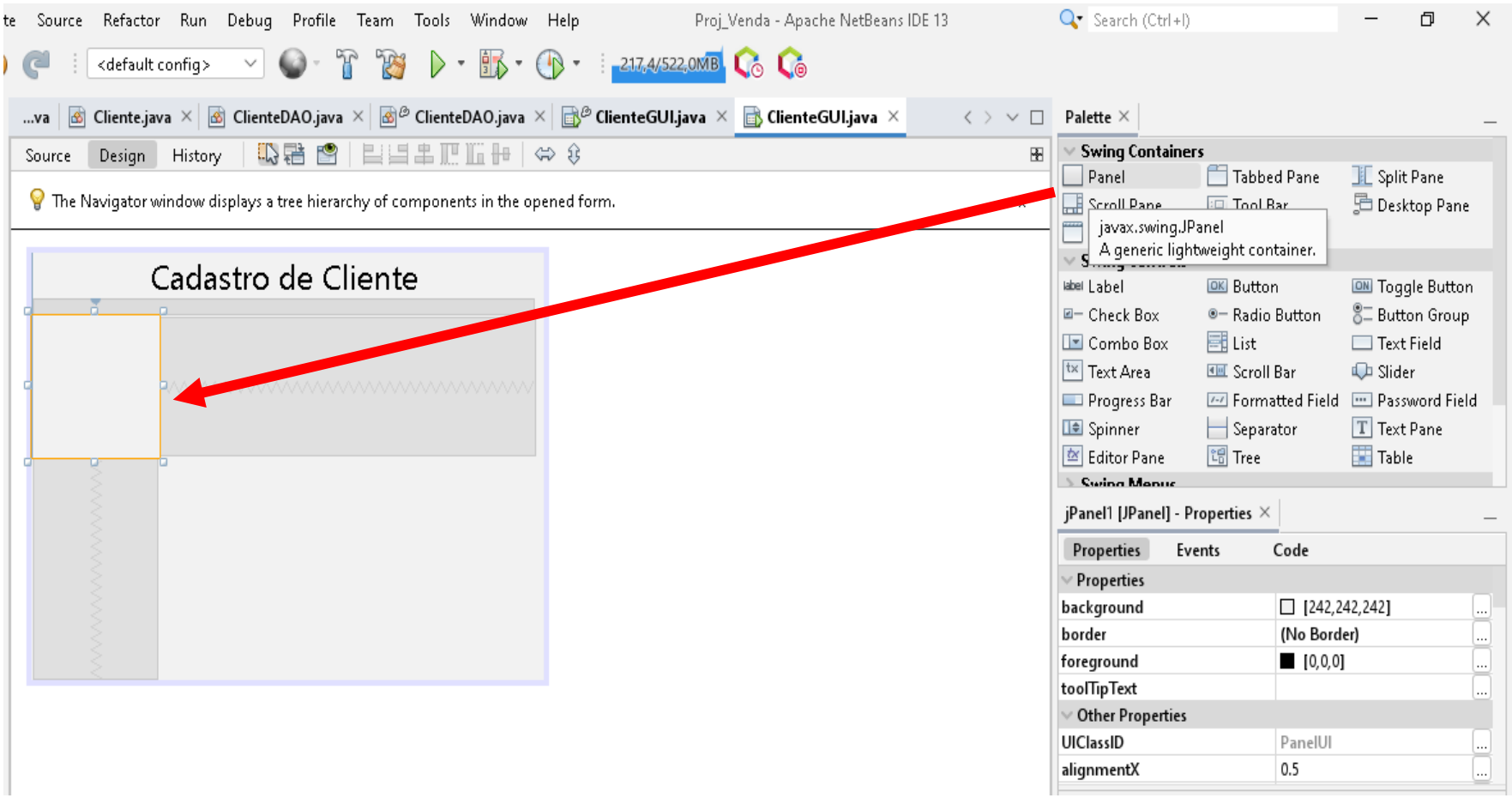
2) Na paleta do lado direito, em controles Swing Controls, arrastar Label para o formulário. Em propriedades (Properties) mudar o texto, tamanho da fonte e centralizar:

- text = Cadastro de Cliente
- font = aumentar a fonte
- horizontalAlignment = Center

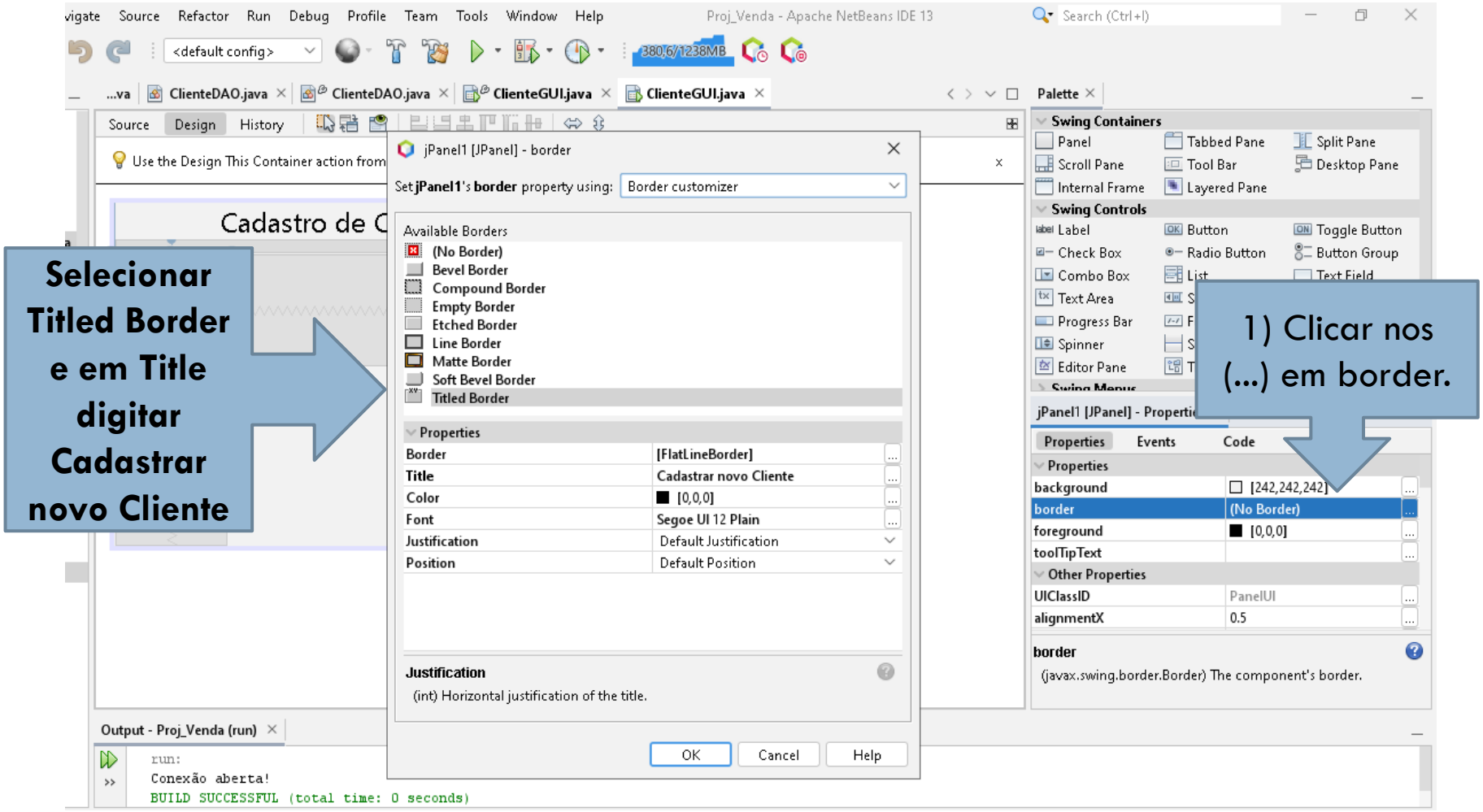
Visualizando o Design



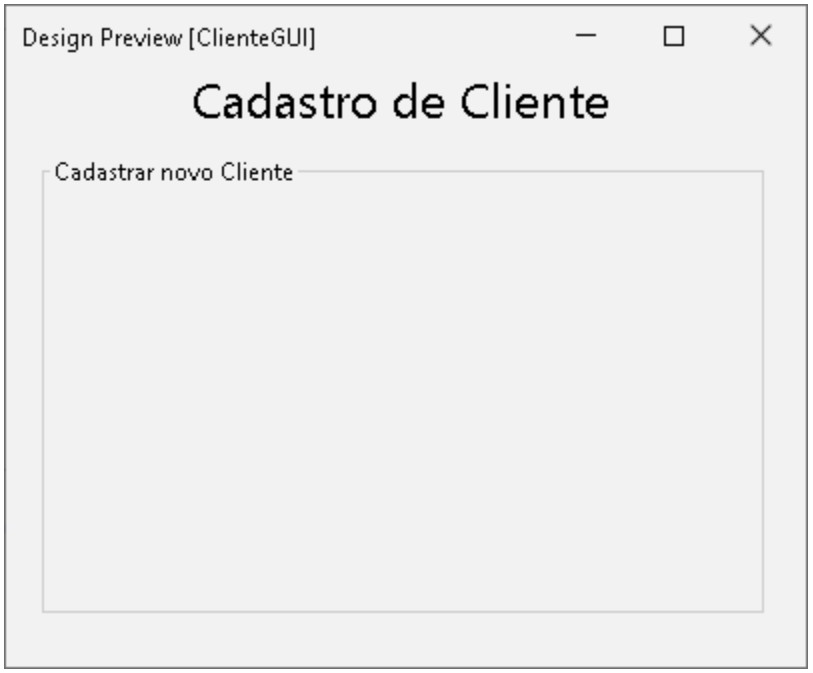
Inserindo Panel



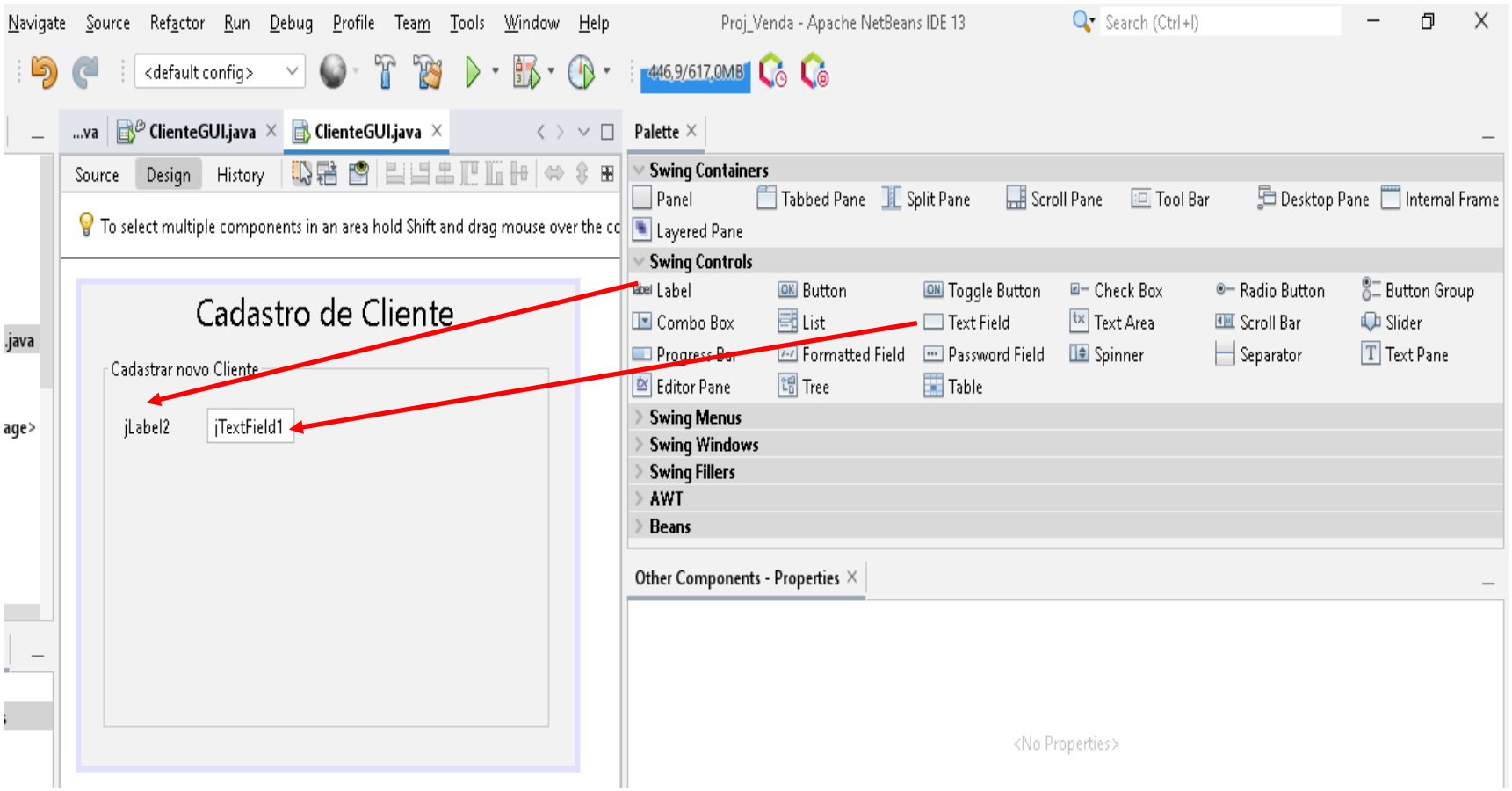
Alterando Propriedades do Panel



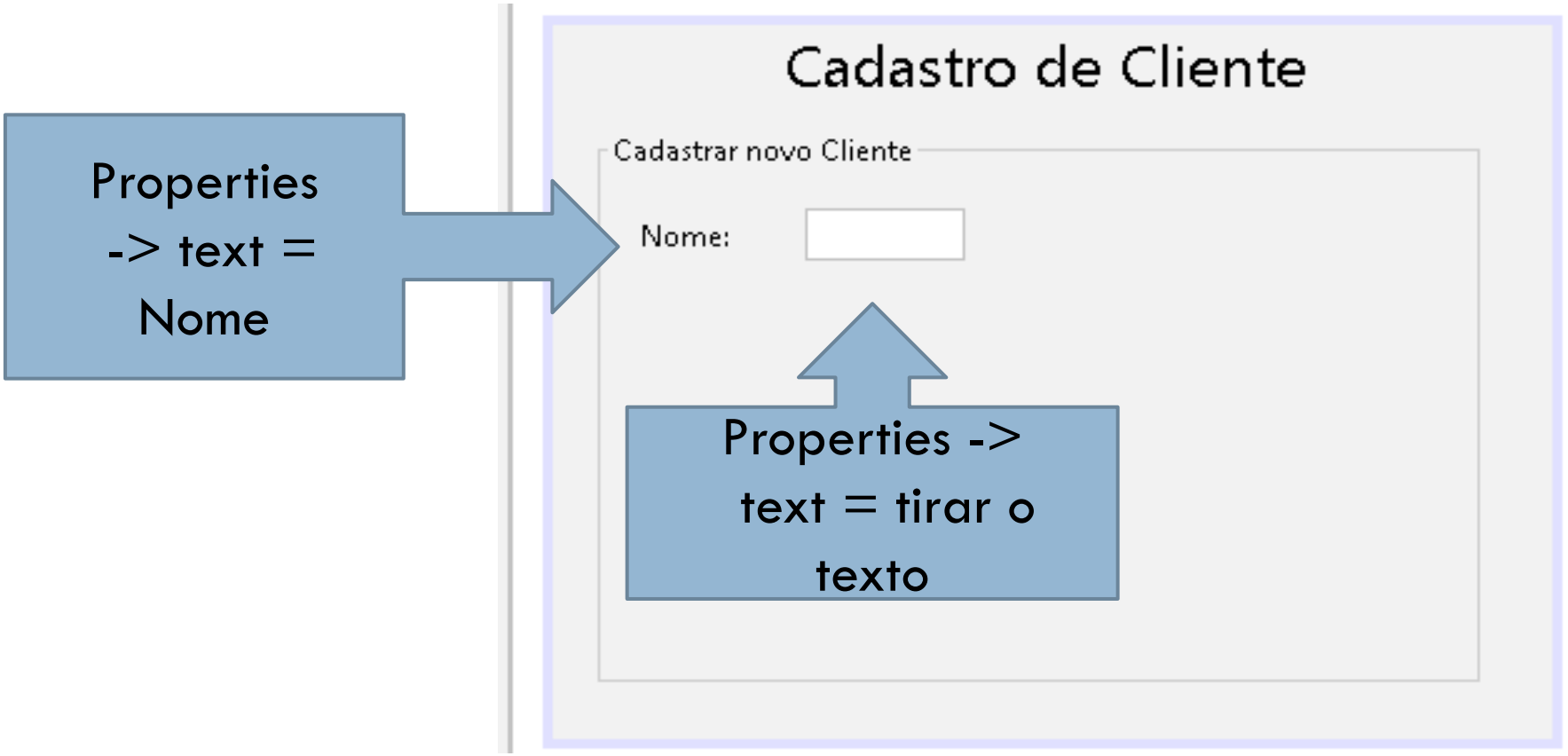
Cadastro de Cliente



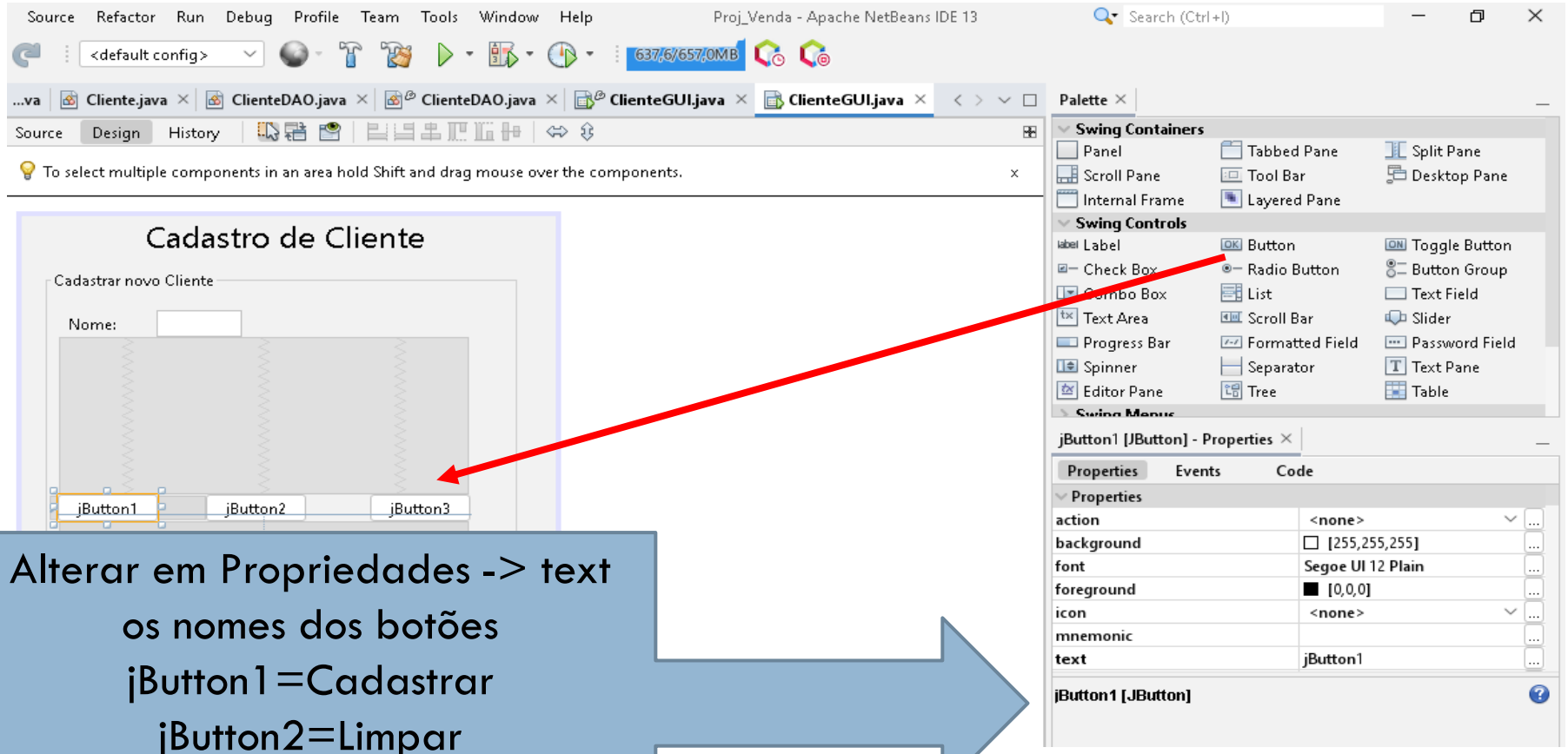
Cadastro de Cliente



Cadastro de Cliente



Cadastro de Cliente



Alterar em Propriedades -> text

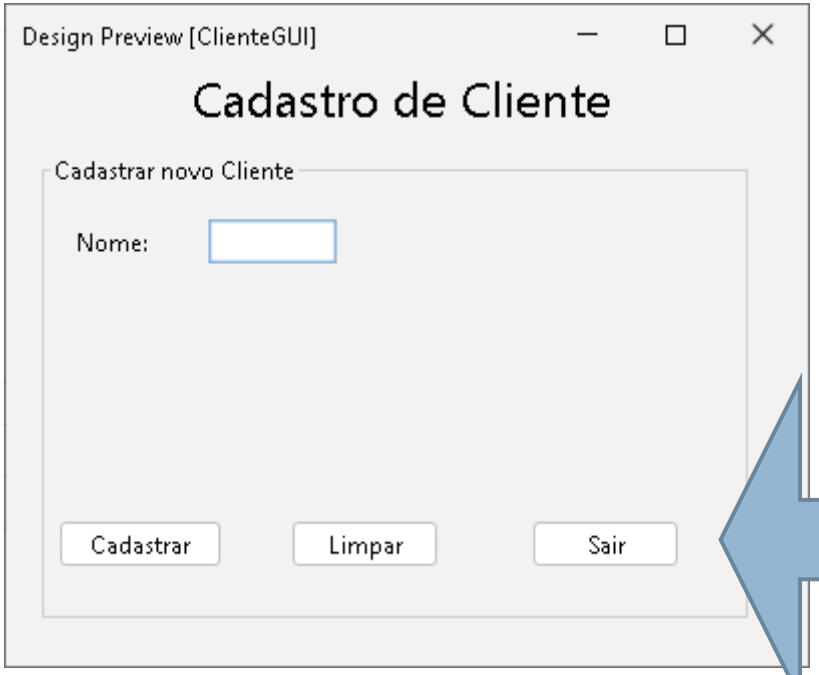
os nomes dos botões

jButton1=Cadastrar

jButton2=Limpar

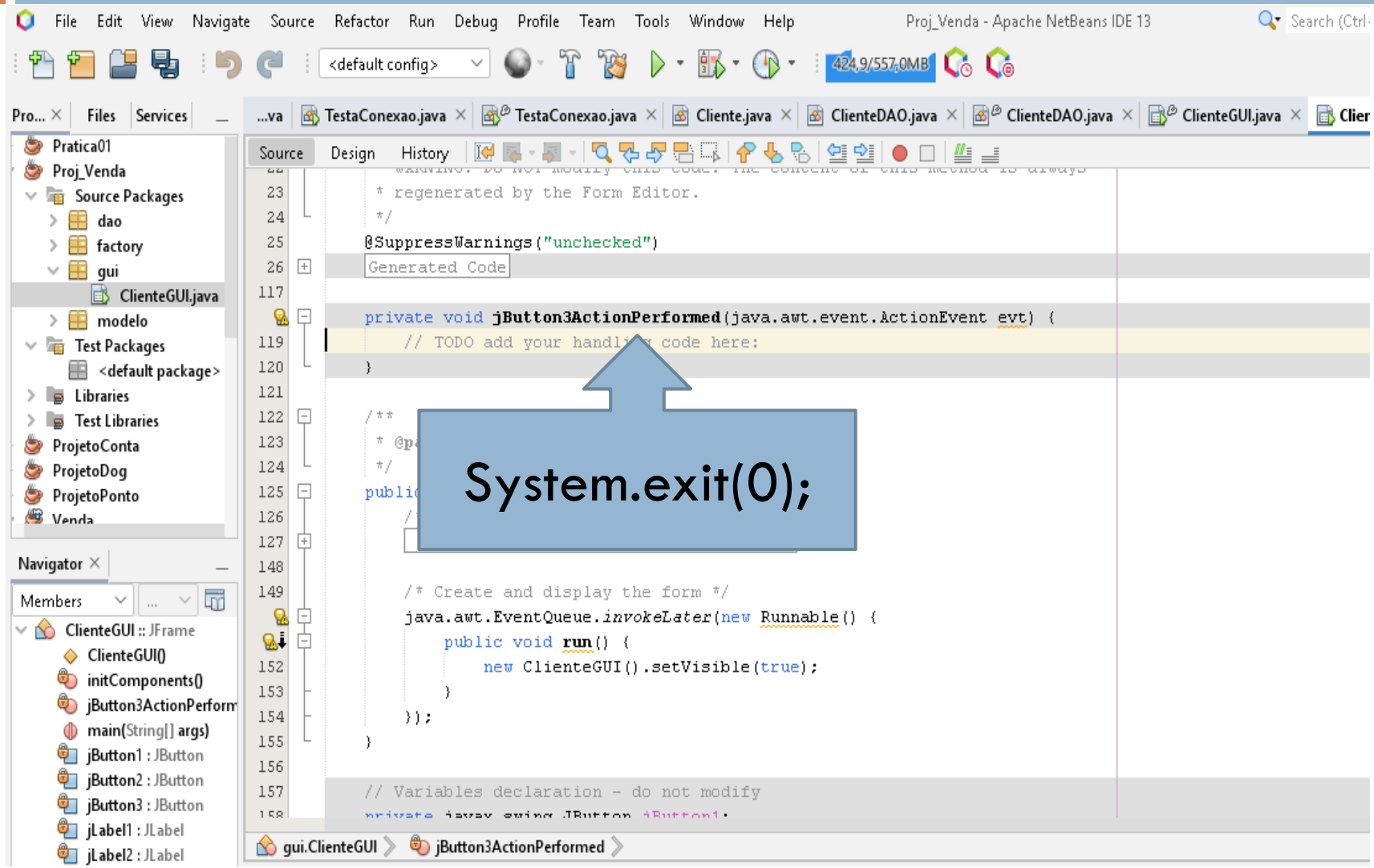
jButton3=Sair

Cadastro de Cliente – Evento Sair



Em Design dê um duplo clique no botão Sair.

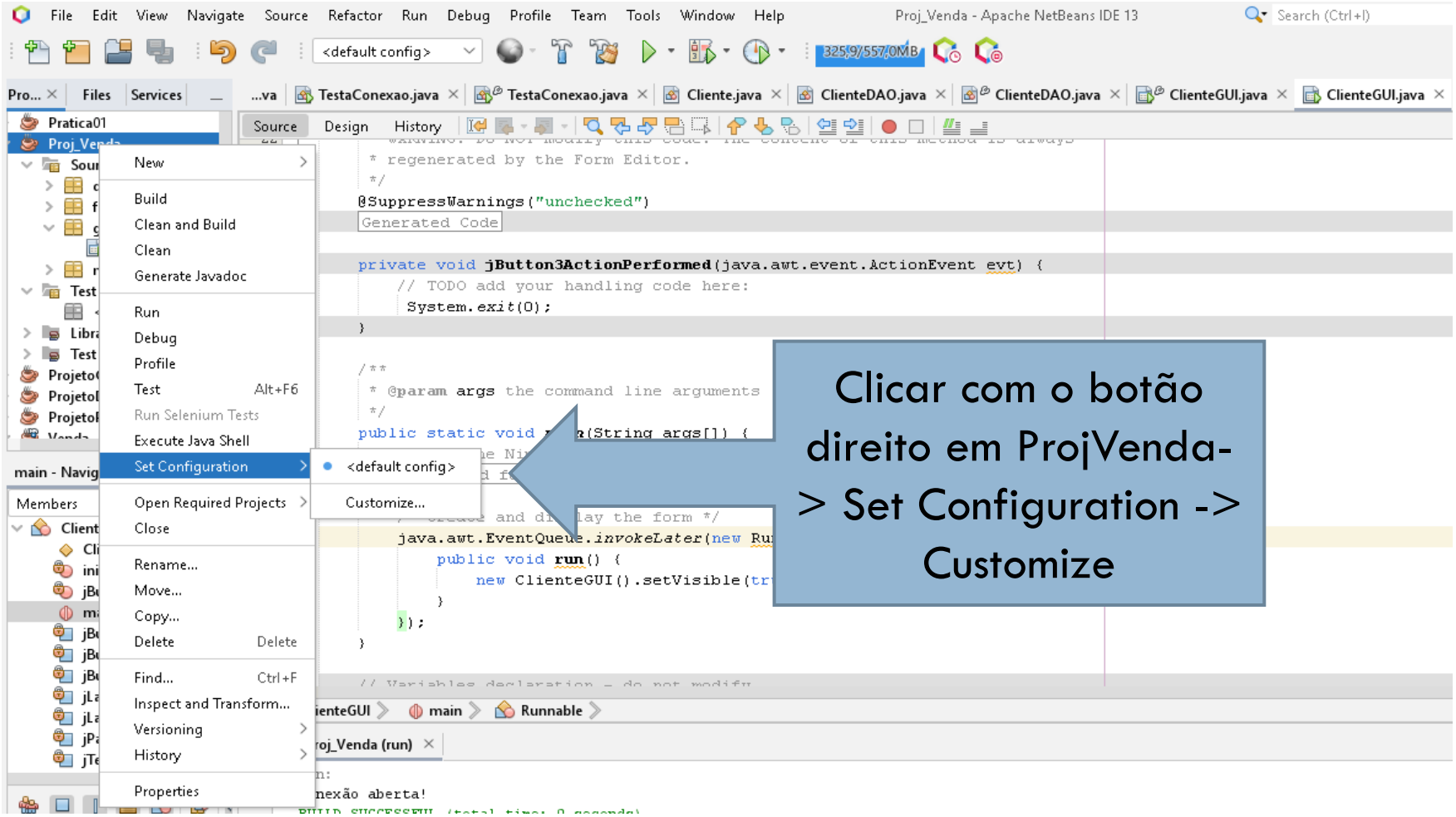
Evento Sair


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Proj_Venda - Apache NetBeans IDE 13
<default config>
Pro... Files Services
Pratica01
Proj_Venda
Source Packages
dao
factory
gui
ClienteGUI.java
modelo
Test Packages
<default package>
Libraries
Test Libraries
ProjetoConta
ProjetoDog
ProjetoPonto
Navigator
Members
ClienteGUI :: JFrame
ClienteGUI()
initComponents()
jButton3ActionPerform
main(String[] args)
jButton1 : JButton
jButton2 : JButton
jButton3 : JButton
jLabel1 : JLabel
jLabel2 : JLabel
TestaConexao.java
TestaConexao.java
Cliente.java
ClienteDAO.java
ClienteDAO.java
ClienteGUI.java
Source Design History
* regenerated by the Form Editor.
*/
@SuppressWarnings("unchecked")
Generated Code
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
}
/**
*/
/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
public void run() {
new ClienteGUI().setVisible(true);
}
});
}
// Variables declaration - do not modify
gui.ClienteGUI
jButton3ActionPerformed
System.exit(0);

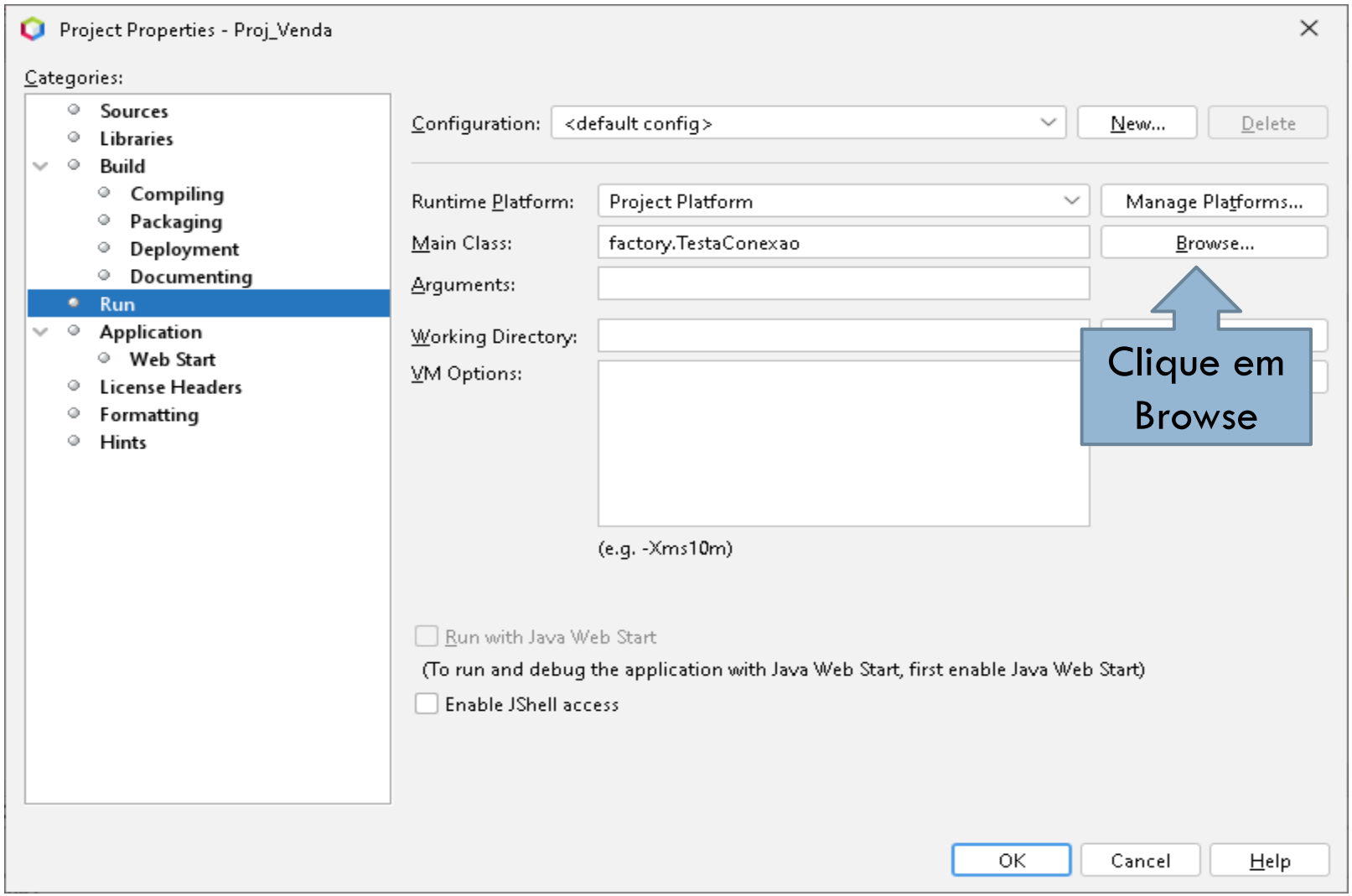
Evento Sair


```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    System.exit(0);
}
```

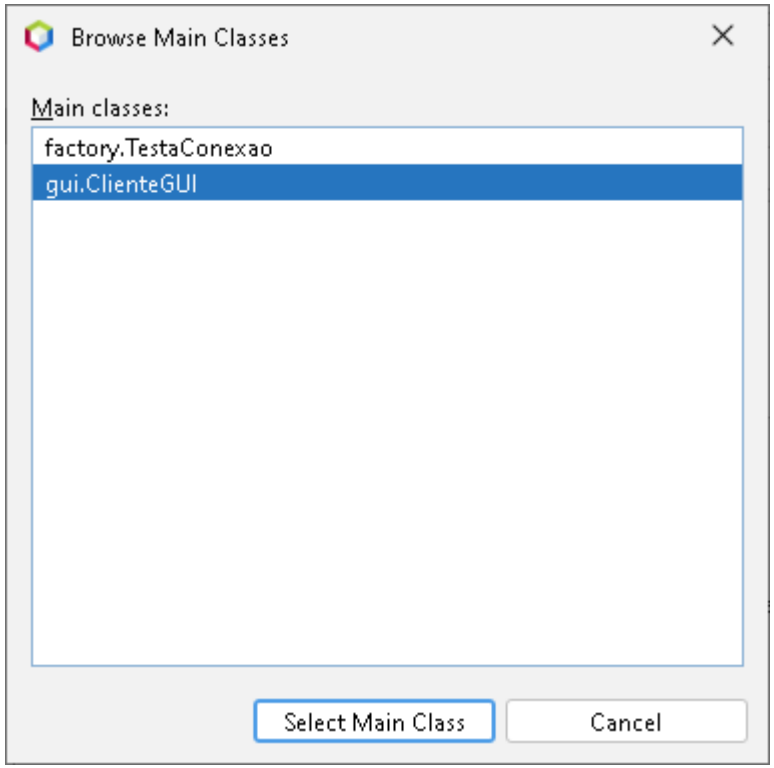
Alterando a Classe Main



Alterando a Classe Main

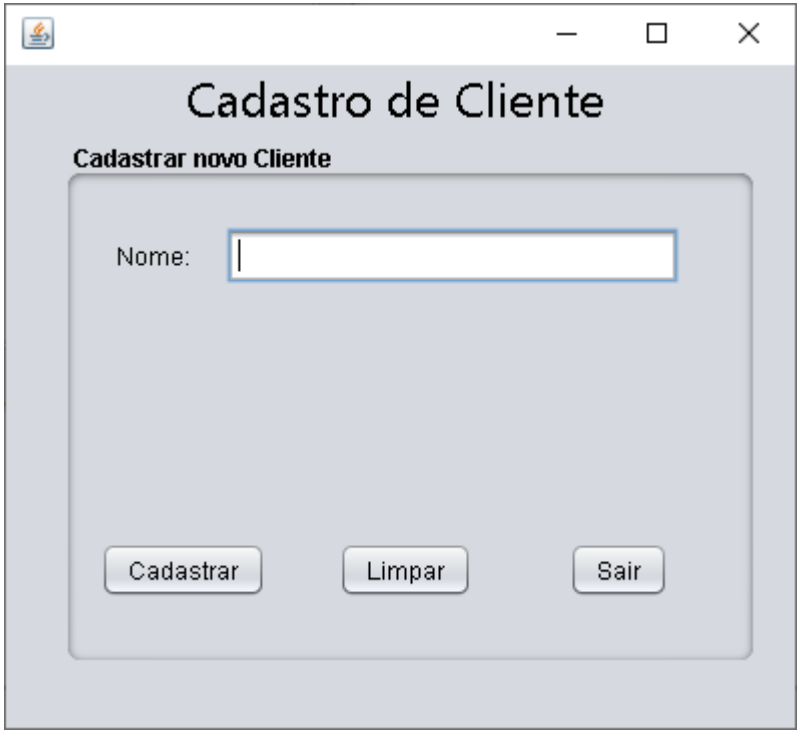


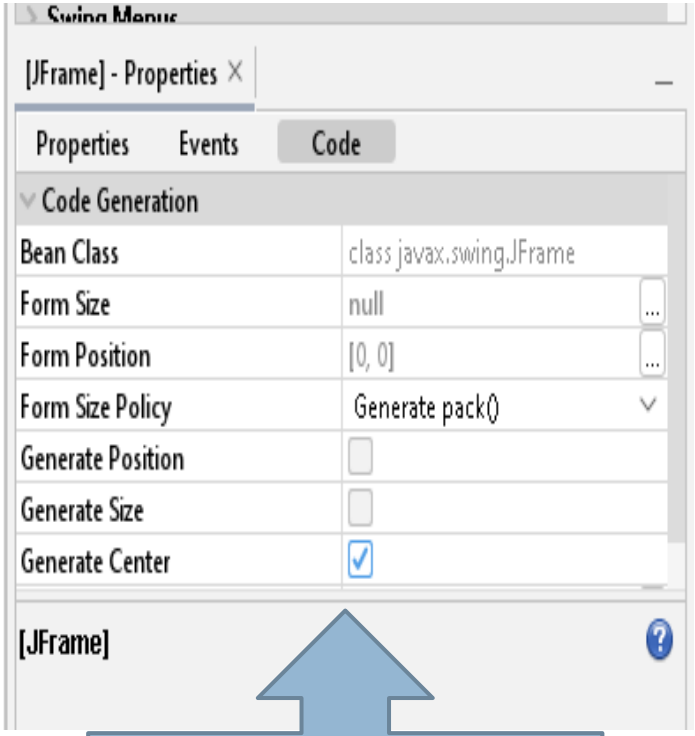
Alterando a Classe Main



Selecione gui.ClienteGUI e clique no botão Select Main Class

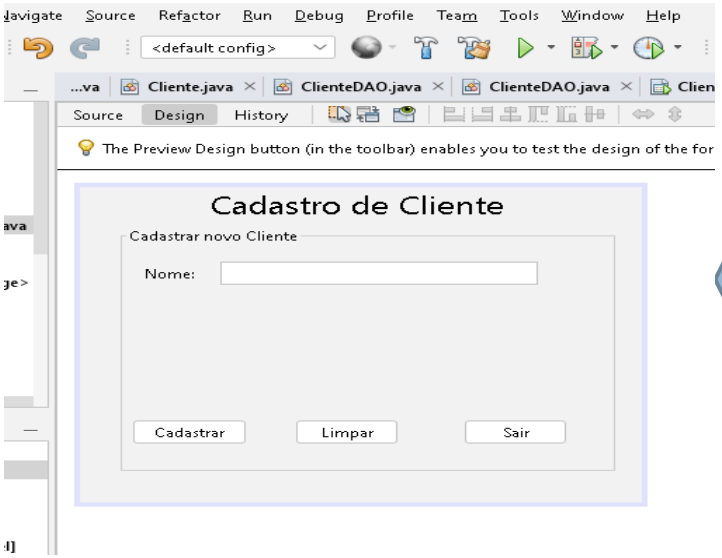
Executando





Para centralizar clicar em Generate Center

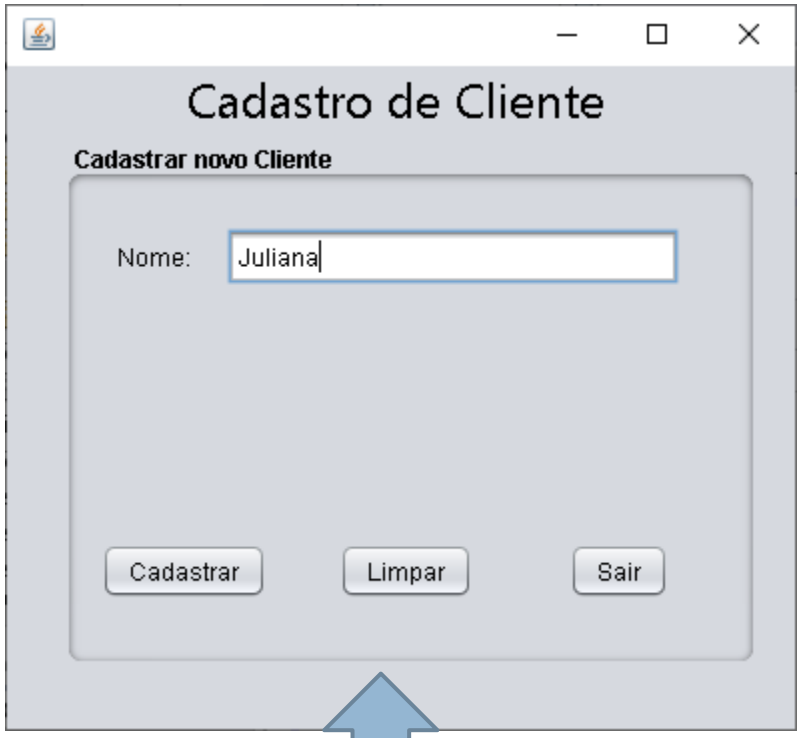
Evento Limpar

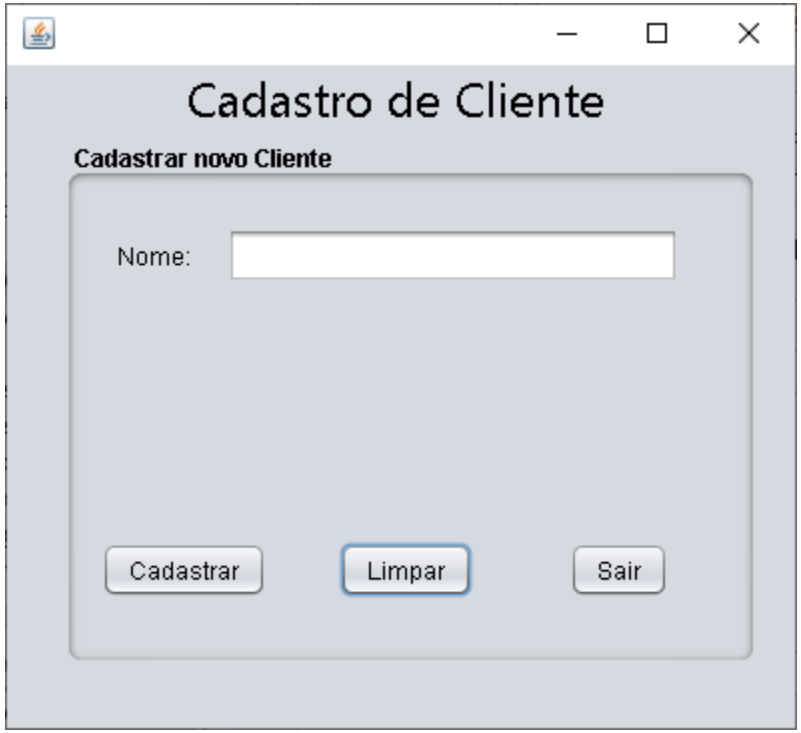


Em Design dê um duplo clique no botão Limpar

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jTextField1.setText("");

}
```

Executando



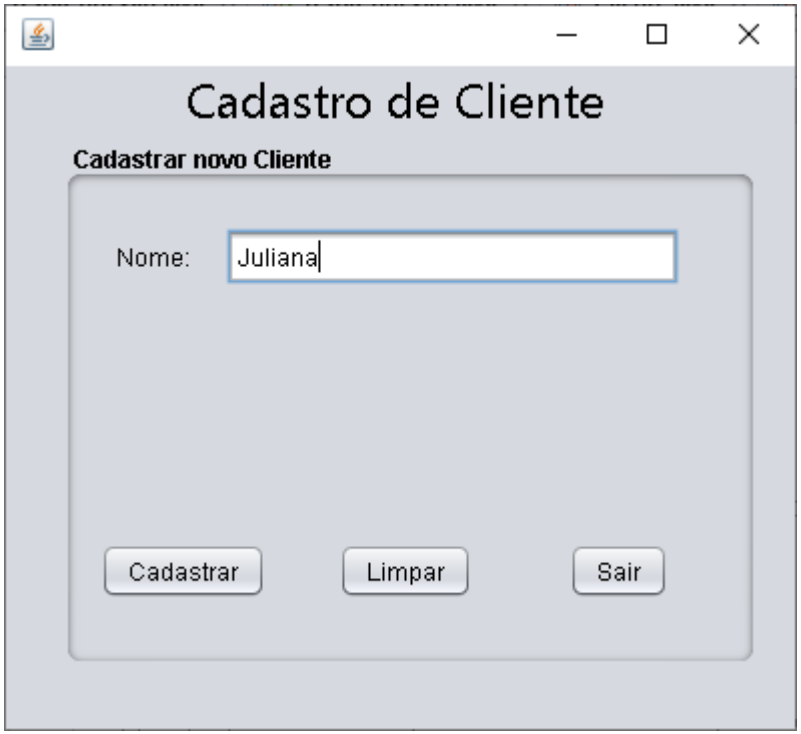


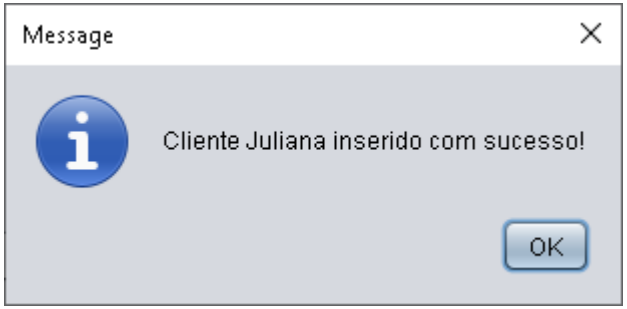
Digitar um Nome e clicar em Limpar

Evento Cadastrar


```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
   Cliente cliente = new Cliente();
       cliente.setNome(jTextField1.getText());

   // fazendo a validação dos dados
   if ((jTextField1.getText().isEmpty())) {
         JOptionPane.showMessageDialog(null, "O campo não podem retornar vazio");
    }
    else {

     // instanciando a classe ClienteDAO do pacote dao e criando seu objeto dao
     ClienteDAO dao = new ClienteDAO();
     dao.adiciona(cliente);
     JOptionPane.showMessageDialog(null, "Cliente "+jTextField1.getText()+" inserido com sucesso! ");
    }

   // apaga os dados preenchidos nos campos de texto
   jTextField1.setText("");
}
```

Executando





Digitar o Nome e clicar em Cadastrar

Visualizando no Banco de Dados