



- Criando um cadastro de cliente em JAVA

Disciplina: LP
Profa. Me Juliana Pasquini
25-3-2024

Objetivo

2

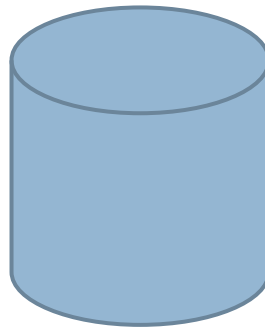
- O objetivo desta aula é desenvolver uma aplicação em JSE (Java Standard Edition) de inserção de dados utilizando alguns padrões de projeto

Roteiro

3

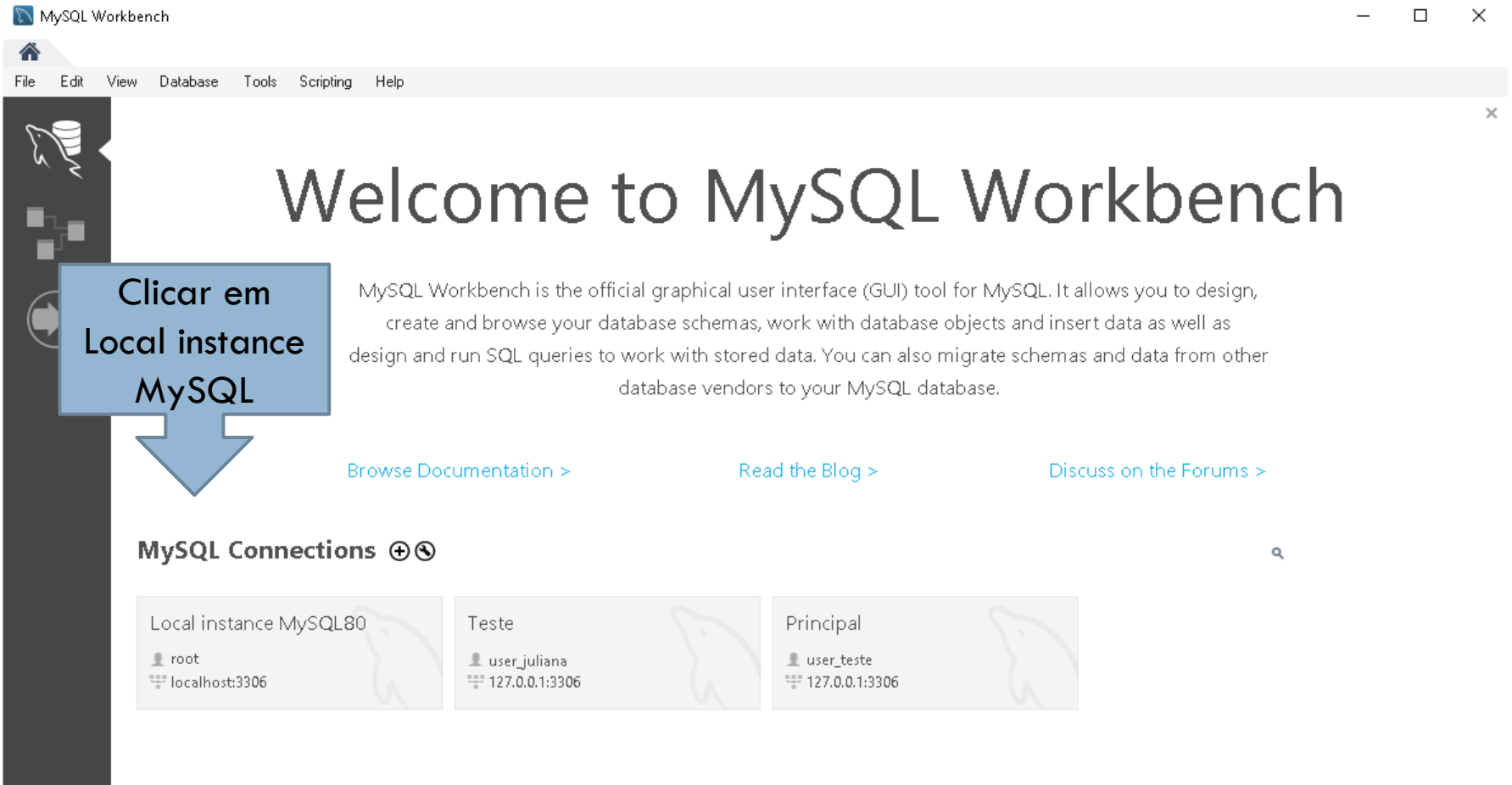
- ❑ Criar o banco de Dados Vendas e a tabela Cliente no MySQL;
- ❑ Criar o Projeto no Netbeans.

Criando o Banco de Dados e a tabela Cliente no MySQL



Acessando MySQL Workbench (1)

5



The screenshot shows the MySQL Workbench application window. The title bar reads "MySQL Workbench". The menu bar includes "File", "Edit", "View", "Database", "Tools", "Scripting", and "Help". The main area displays a "Welcome to MySQL Workbench" message, followed by a paragraph describing the tool's capabilities: "MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database." Below this text are three links: "Browse Documentation >", "Read the Blog >", and "Discuss on the Forums >". On the left side, there is a sidebar with icons for various database functions. A blue callout box with a large blue arrow pointing down contains the text "Clicar em Local instance MySQL". At the bottom, the "MySQL Connections" section shows three connection cards: "Local instance MySQL80" (root user, localhost:3306), "Teste" (user_juliana, 127.0.0.1:3306), and "Principal" (user_teste, 127.0.0.1:3306).

MySQL Workbench

File Edit View Database Tools Scripting Help

Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

[Browse Documentation >](#) [Read the Blog >](#) [Discuss on the Forums >](#)

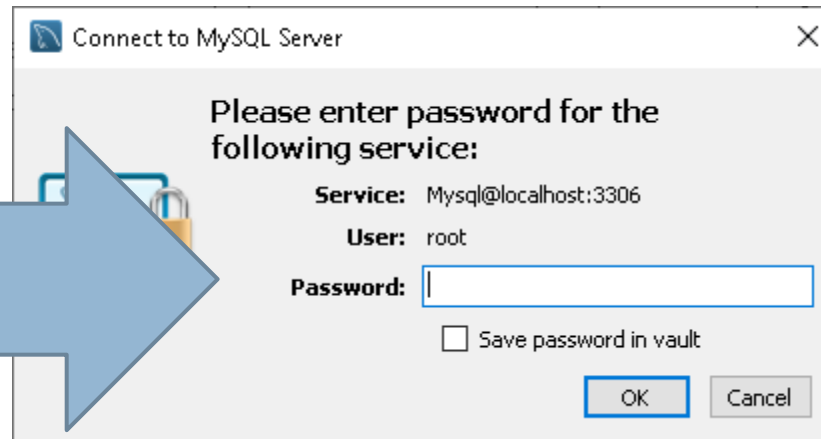
MySQL Connections

Connection Name	User	Host
Local instance MySQL80	root	localhost:3306
Teste	user_juliana	127.0.0.1:3306
Principal	user_teste	127.0.0.1:3306

Acessando MySQL Workbench(2)

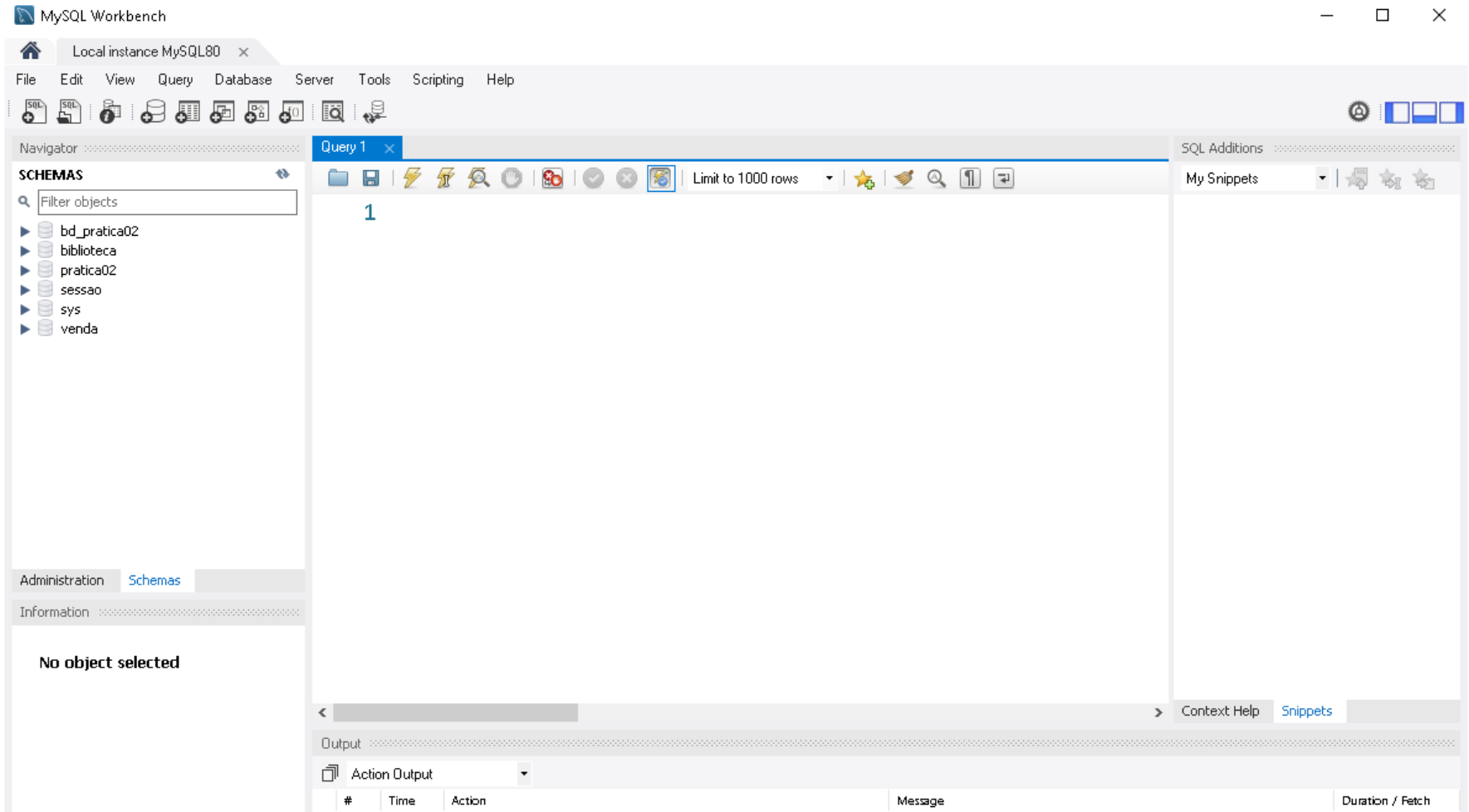
6

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



Acessando MySQL Workbench(3)

7

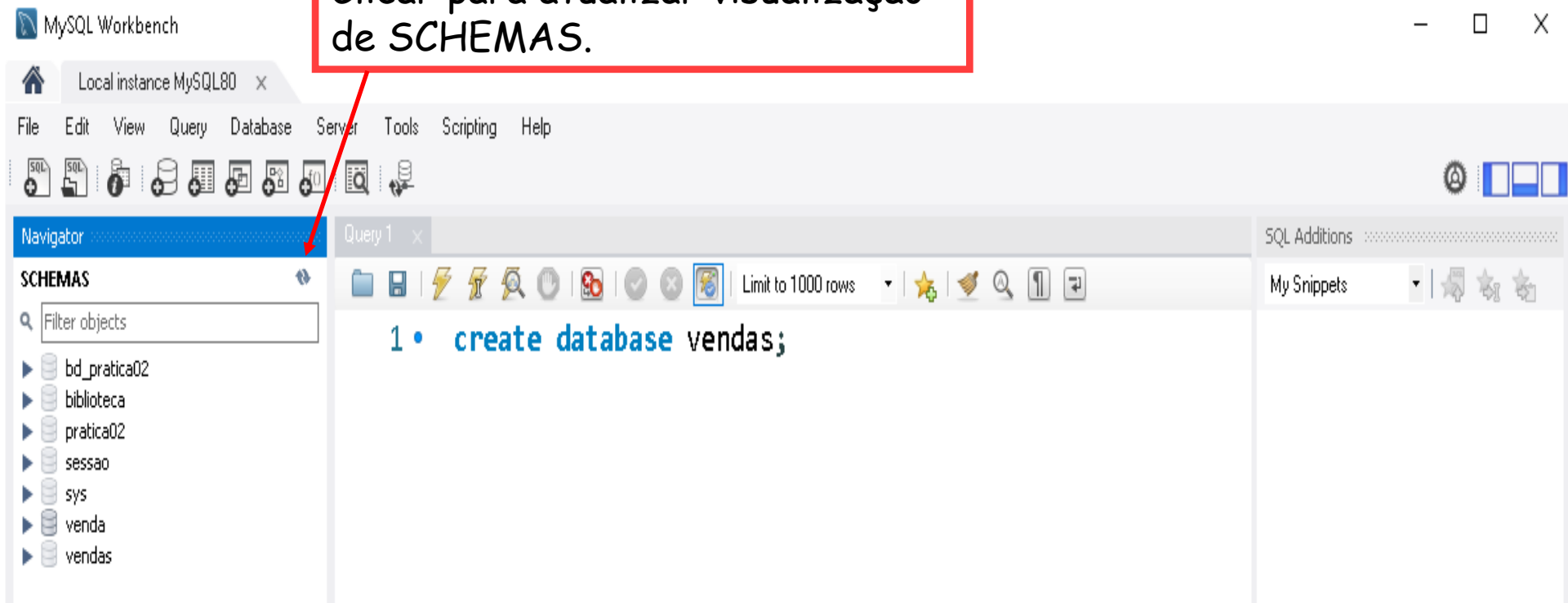


Criando o Banco de Dados Vendas

8

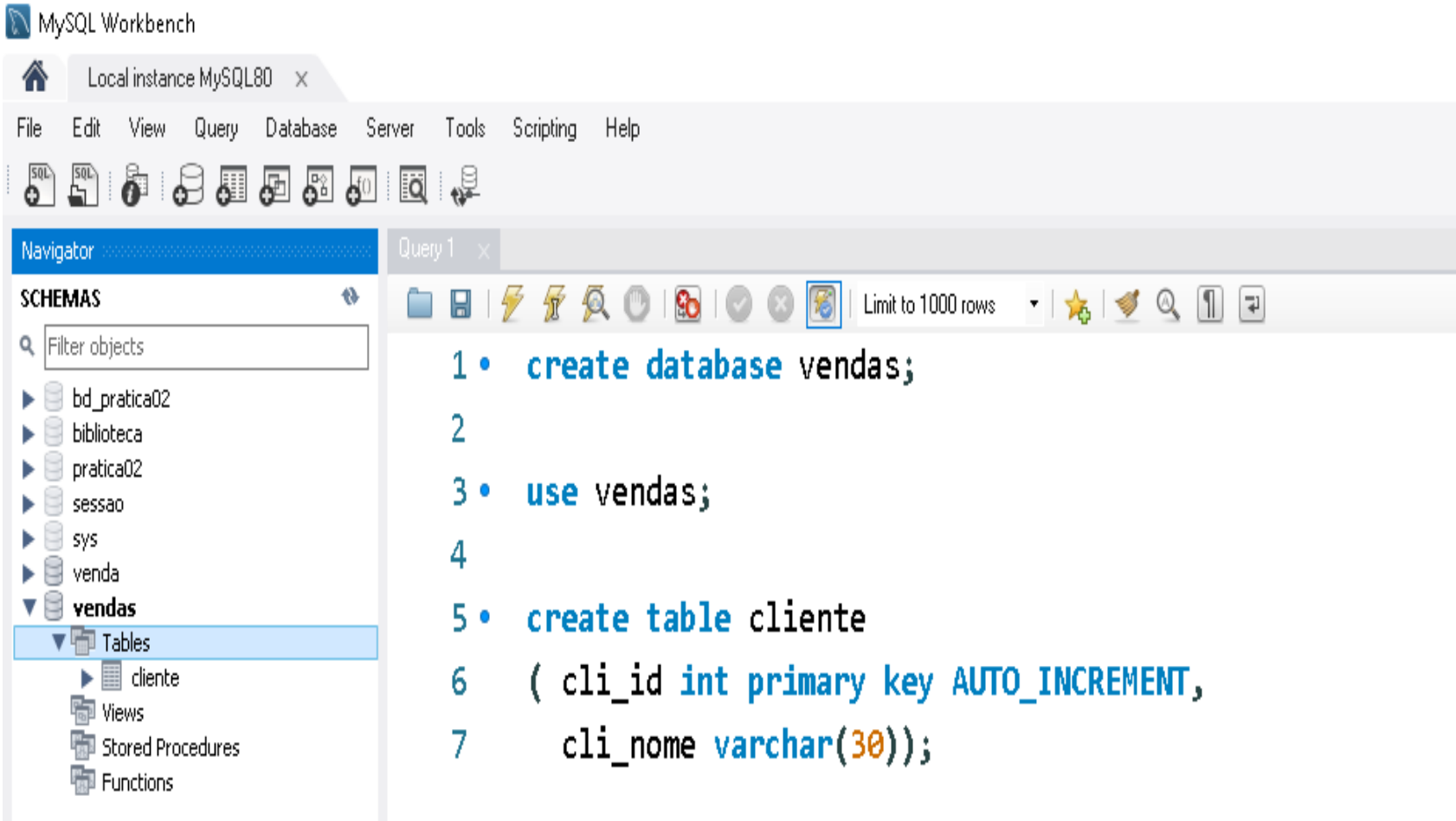
Sintaxe: create database <nome do Banco de dados>;

Clicar para atualizar visualização de SCHEMAS.



Criando a tabela cliente

9



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'vendas' database selected. The main editor window, titled 'Query 1', contains the following SQL code:

```
1 • create database vendas;  
2  
3 • use vendas;  
4  
5 • create table cliente  
6   ( cli_id int primary key AUTO_INCREMENT,  
7     cli_nome varchar(30));
```

Usando o comando Select

10

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bd_pratica02
- biblioteca
- pratica02
- sessao
- sys
- venda
- vendas**
 - Tables
 - cliente
 - Views
 - Stored Procedures
 - Functions

Query 1 x

Limit to 1000 rows

```
3 • use vendas;  
4  
5 • create table cliente  
6 ( cli_id int primary key AUTO_INCREMENT,  
7   cli_nome varchar(30));  
8  
9 • select * from cliente;
```

Result Grid

	cli_id	cli_nome
*	NULL	NULL

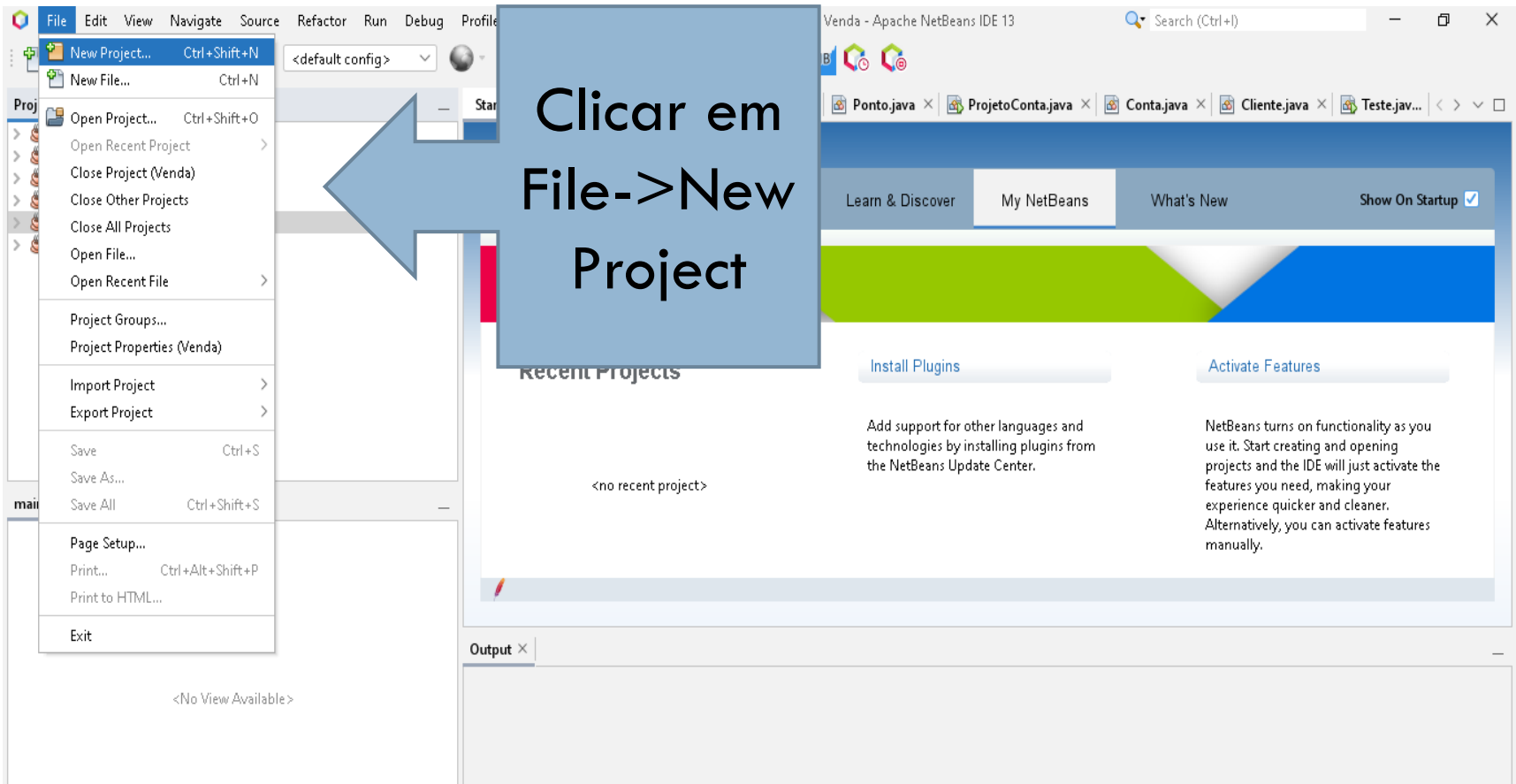
Administration Schemas Information

Schema: vendas

Criando o Projeto Venda no NetBeans

Criando um Novo Projeto (1)

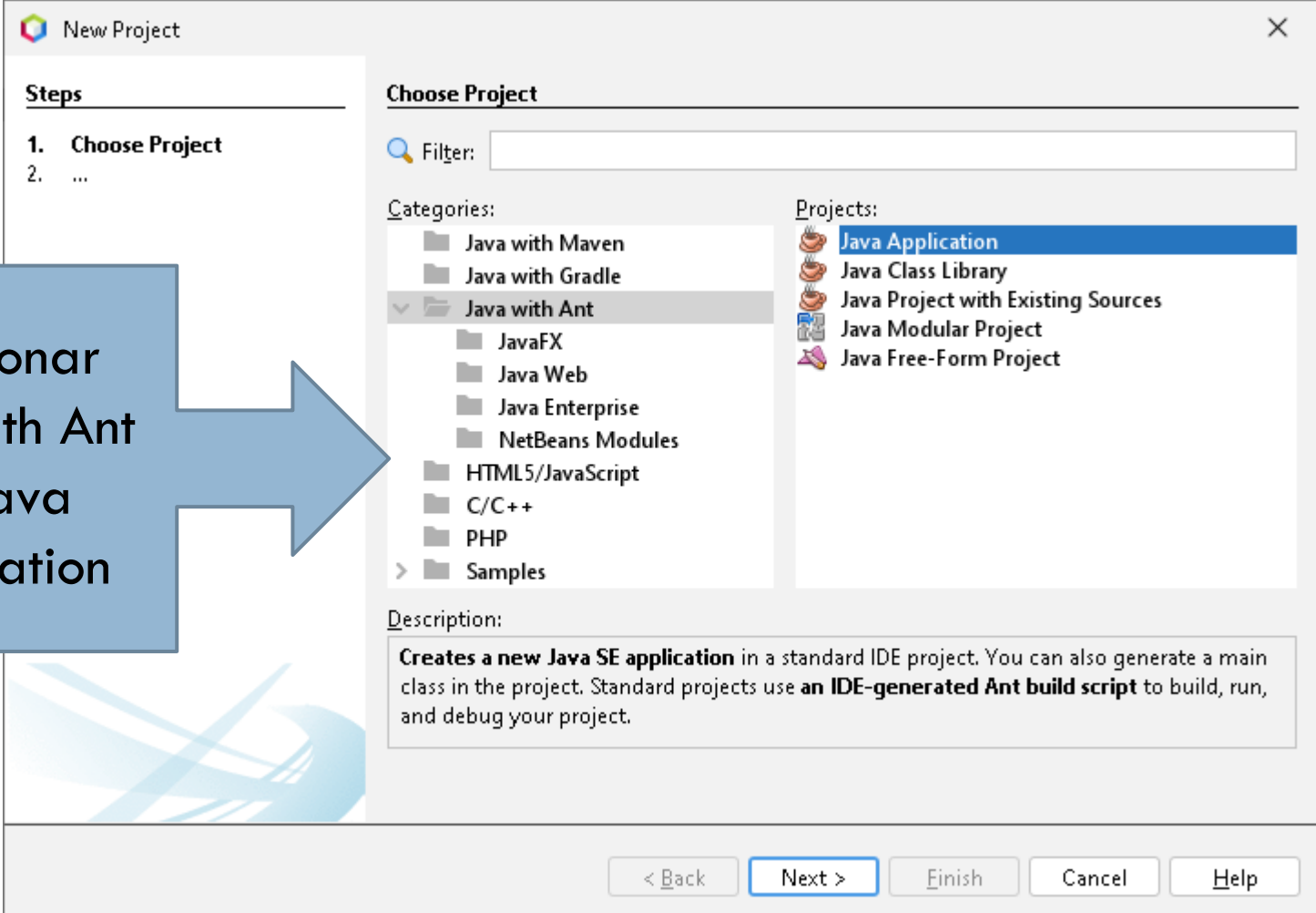
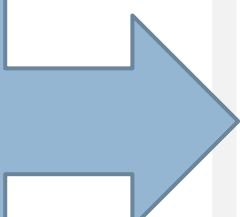
12



Criando um Novo Projeto (2)

13

Selecionar
Java with Ant
-> Java
Application



Steps

1. Choose Project
2. ...

Choose Project

Filter:

Categories:

- Java with Maven
- Java with Gradle
- Java with Ant
 - JavaFX
 - Java Web
 - Java Enterprise
 - NetBeans Modules
 - HTML5/JavaScript
 - C/C++
 - PHP
 - Samples

Projects:

- Java Application
- Java Class Library
- Java Project with Existing Sources
- Java Modular Project
- Java Free-Form Project

Description:

Creates a new **Java SE application** in a standard IDE project. You can also generate a main class in the project. Standard projects use an **IDE-generated Ant build script** to build, run, and debug your project.

< Back Next > Finish Cancel Help

Criando um Projeto (3)

14

New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: Proj_Venda

Project Location: C:\Fatec-1SEM-2022\Disciplinas\LP\Aulas\23-03-2022-Projeto_CRUD_Netbeans Browse...

Project Folder: C:\Fatec-1SEM-2022\Disciplinas\LP\Aulas\23-03-2022-Projeto_CRUD_Netbeans\Proj_Venda

☒ Use Dedicated Folder for Storing Libraries

Libraries Folder: .lib Browse...

Different users and projects can share the same compilation libraries (see Help for details).

☐ Create Main Class proj_venda.Proj_Venda

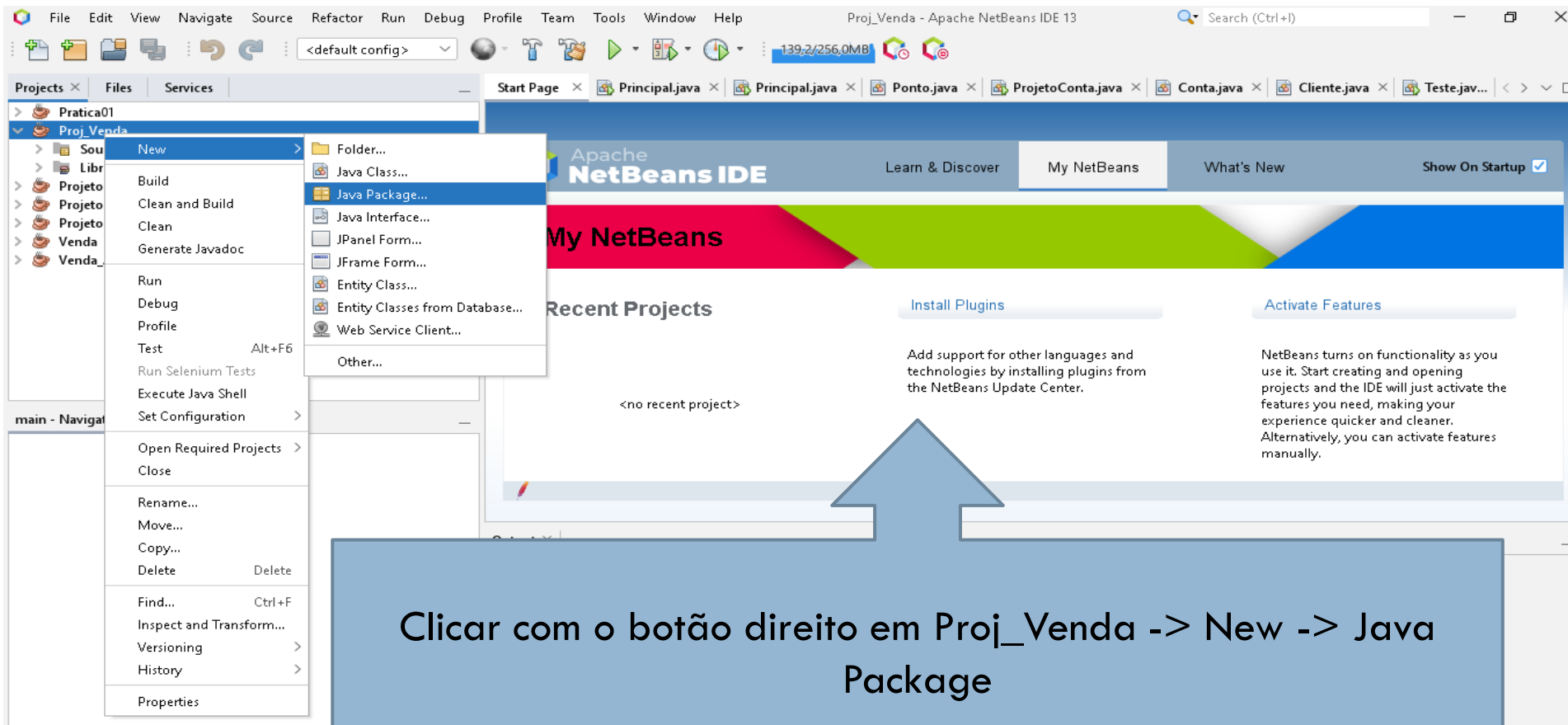
Instructions:

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

< Back Next > **Finish** Cancel Help

Criando Pacotes (Packages) (1)

15



Criando Pacotes (Packages) (2)

16

New Java Package

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Package Name:

Project:

Location:

Created Folder:

Digitar factory no nome do pacote e clicar no botão Finish.

< Back Next > **Finish** Cancel Help

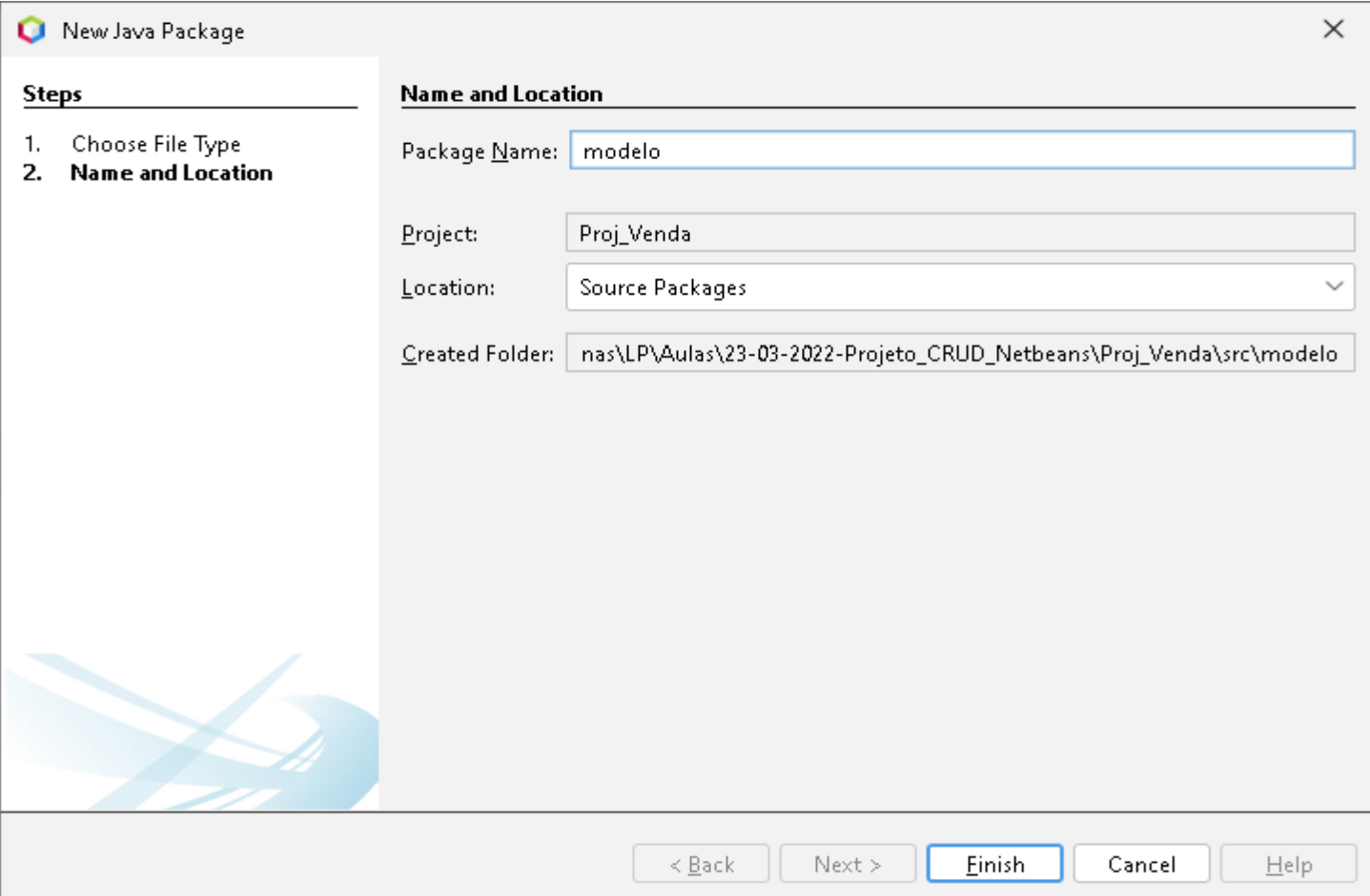
Criando Pacotes (Packages) (3)

17

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

Criando o Pacote modelo

18



New Java Package

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Package Name:

Project:

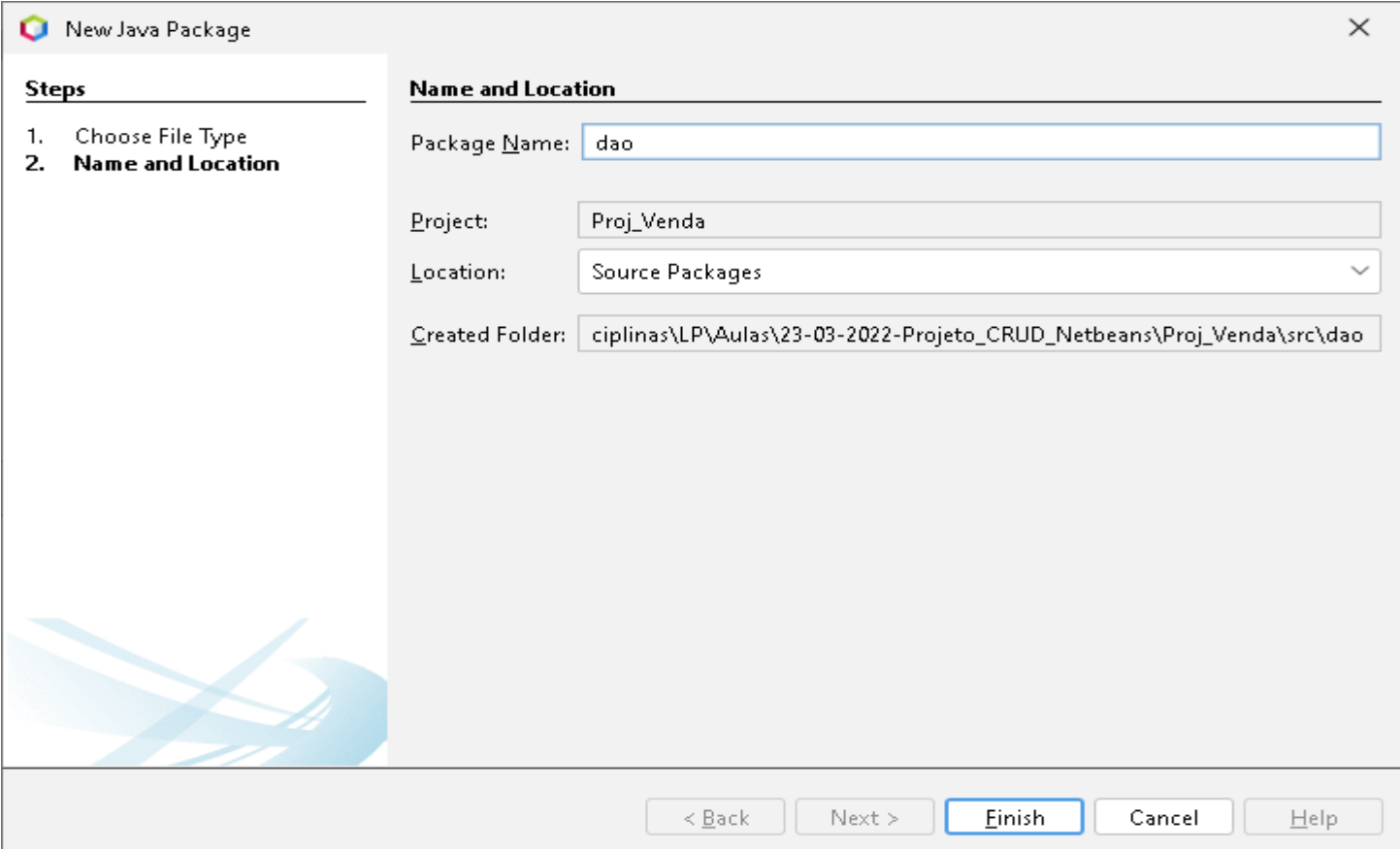
Location:

Created Folder:

< Back Next > **Finish** Cancel Help

Criando o Pacote dao

19



The image shows a 'New Java Package' dialog box from an IDE. It has a title bar with a close button. On the left, a 'Steps' pane shows two steps: '1. Choose File Type' and '2. Name and Location', with the second step being active. The main area is titled 'Name and Location' and contains four input fields: 'Package Name' (containing 'dao'), 'Project' (containing 'Proj_Venda'), 'Location' (a dropdown menu showing 'Source Packages'), and 'Created Folder' (containing a full file path). At the bottom, there are five buttons: '< Back', 'Next >', 'Finish' (which is highlighted with a blue border), 'Cancel', and 'Help'.

New Java Package

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Package Name:

Project:

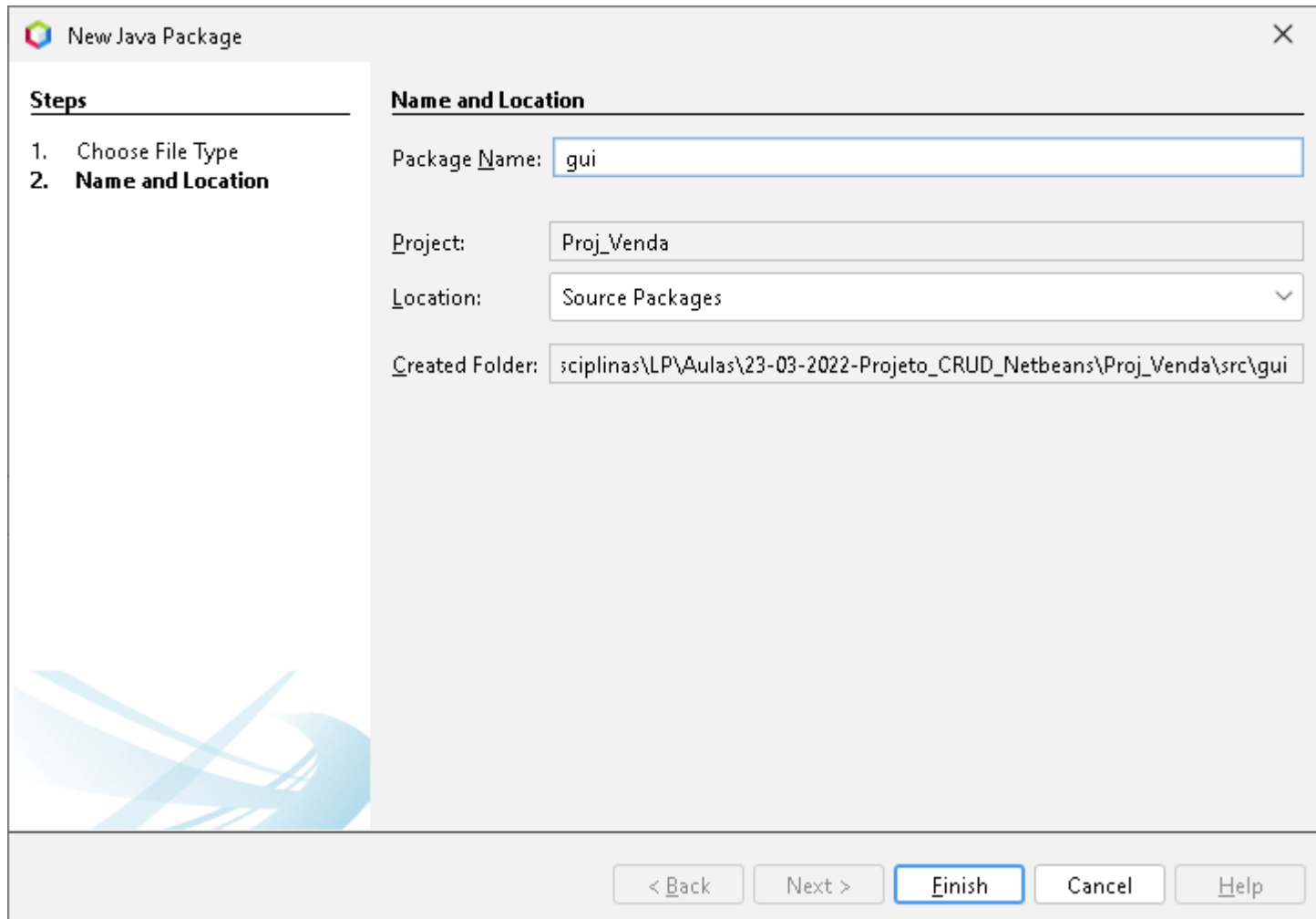
Location:

Created Folder:

< Back Next > **Finish** Cancel Help

Criando o pacote gui

20



The image shows a 'New Java Package' dialog box in an IDE. It has a title bar with a close button. On the left, a 'Steps' pane shows two steps: '1. Choose File Type' and '2. Name and Location', with the second step being the active one. The main area is titled 'Name and Location' and contains four fields: 'Package Name' (text input with 'gui'), 'Project' (text input with 'Proj_Venda'), 'Location' (dropdown menu with 'Source Packages'), and 'Created Folder' (text input with a full file path). At the bottom, there are five buttons: '< Back', 'Next >', 'Finish' (highlighted with a blue border), 'Cancel', and 'Help'.

New Java Package

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Package Name:

Project:

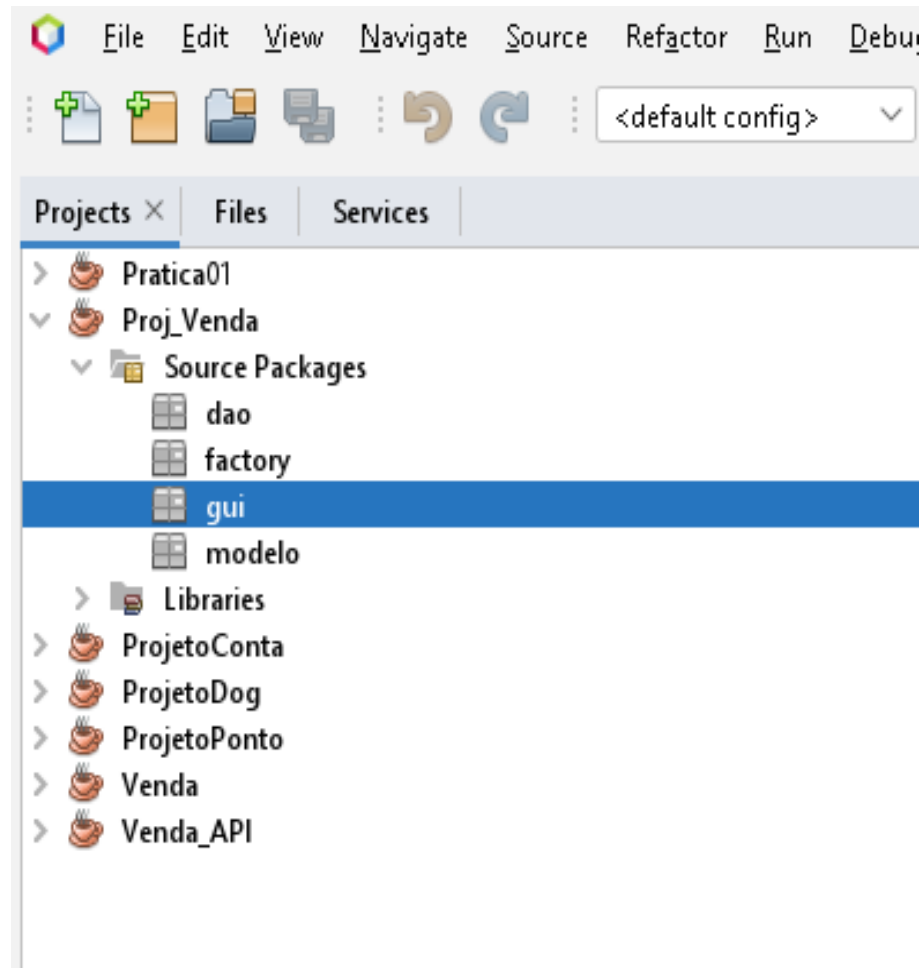
Location:

Created Folder:

< Back Next > **Finish** Cancel Help

Pacotes Criados

21



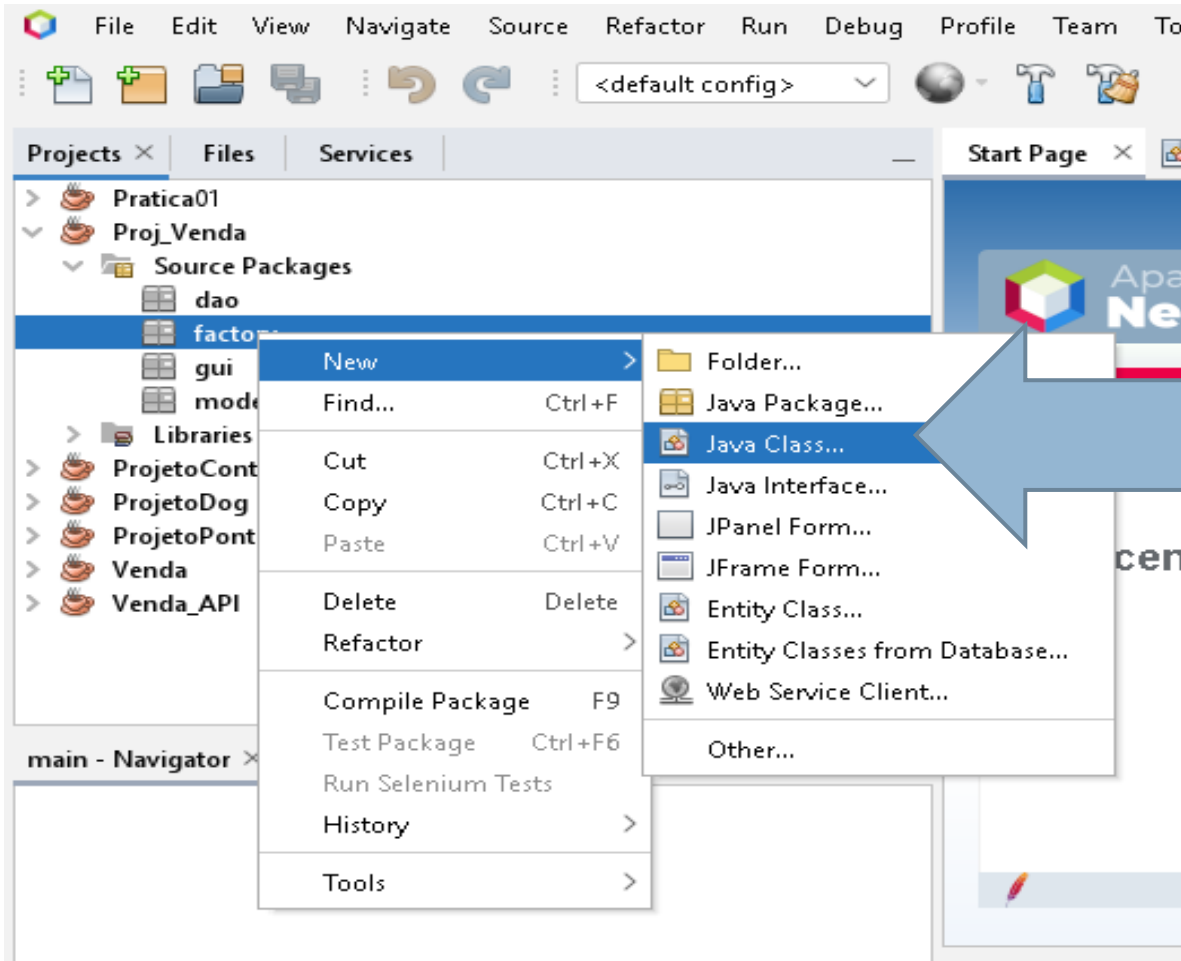
Factory

22

- **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)

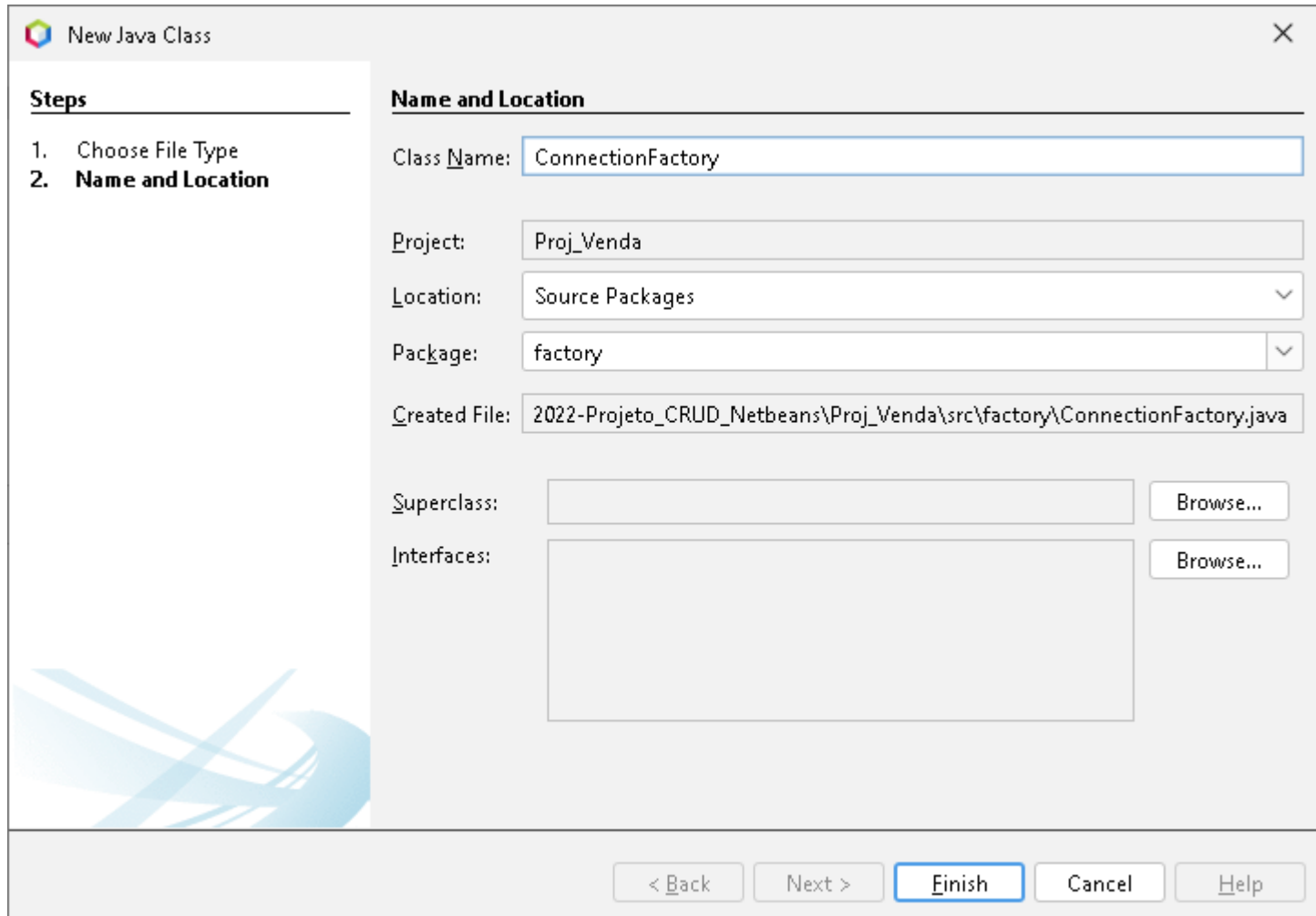
23



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

Criando a classe ConnectionFactory (2)

24



The image shows a 'New Java Class' dialog box with a light gray background. On the left, a 'Steps' panel lists '1. Choose File Type' and '2. Name and Location'. The main area is titled 'Name and Location' and contains several input fields: 'Class Name' (text box with 'ConnectionFactory'), 'Project' (text box with 'Proj_Venda'), 'Location' (dropdown menu with 'Source Packages'), 'Package' (dropdown menu with 'factory'), and 'Created File' (text box with the full file path). Below these are 'Superclass' and 'Interfaces' sections, each with a text box and a 'Browse...' button. At the bottom, there are five buttons: '< Back', 'Next >', 'Finish' (highlighted with a blue border), 'Cancel', and 'Help'.

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name: ConnectionFactory

Project: Proj_Venda

Location: Source Packages

Package: factory

Created File: 2022-Projeto_CRUD_Netbeans\Proj_Venda\src\factory\ConnectionFactory.java

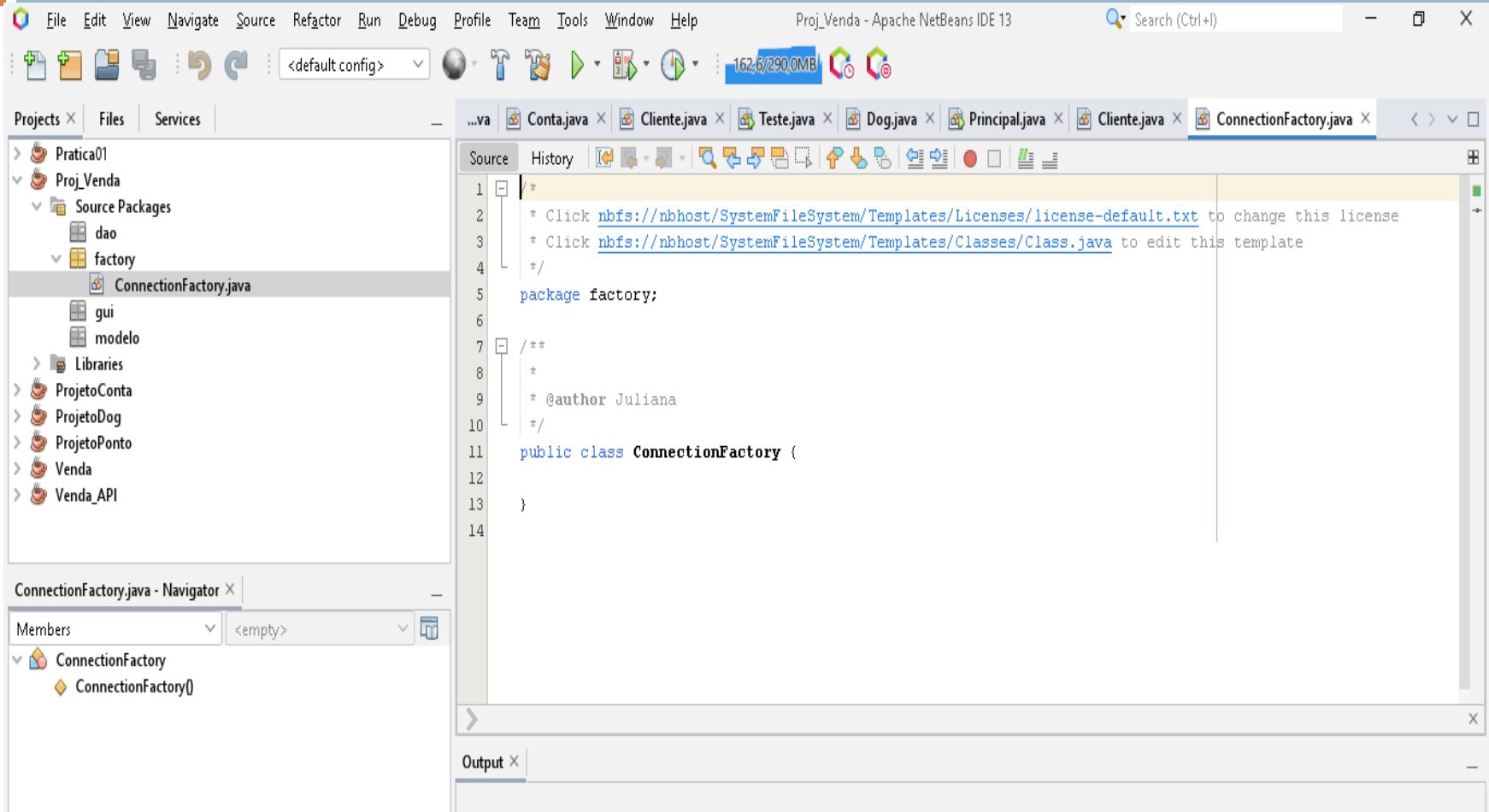
Superclass: Browse...

Interfaces: Browse...

< Back Next > **Finish** Cancel Help

Criando a classe ConnectionFactory (3)

25



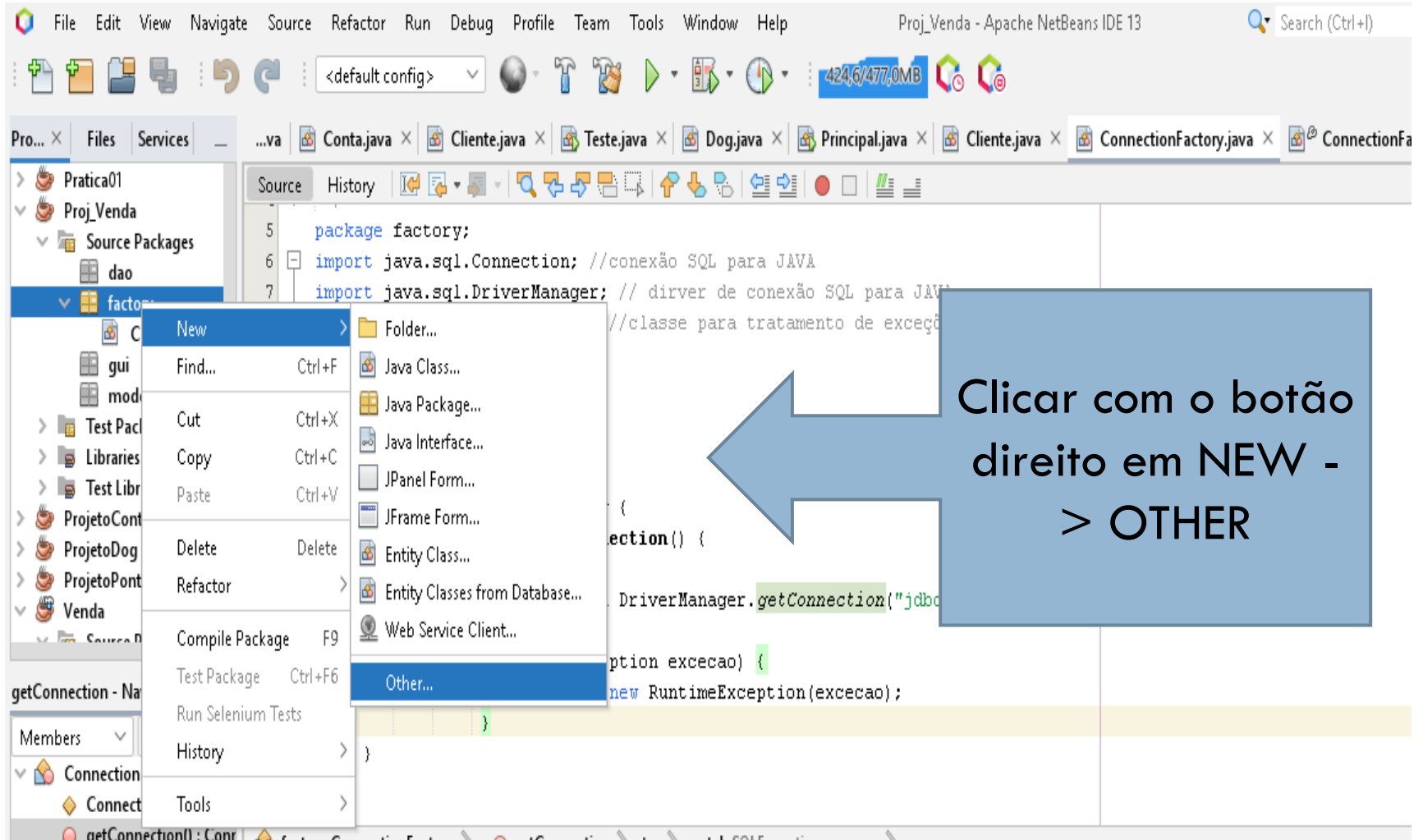
Criando a classe ConnectionFactory (4)

26

```
5 package factory;
6 import java.sql.Connection; //conexão SQL para JAVA
7 import java.sql.DriverManager; // driver 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     }
23 }
```

Criando a classe Main TestaConexao (1)

27



Criando a classe Main TestaConexao (2)

28

Steps

1. Choose File Type
2. ...

Choose File Type

Project: Proj_Venda

Filter:

Categories:

- Java
- Swing GUI Forms
- JavaBeans Objects
- AWT GUI Forms
- Unit Tests
- Web
- Struts
- Spring Framework
- Persistence
- ...

File Types:

- Java Class
- Java Interface
- Java Enum
- Java Annotation Type
- Java Exception
- Java Package Info
- Java Module Info
- JApplet
- Applet
- Java Main Class**
- Java Singleton Class

Description:

Creates a new Java class with a `main` method permitting it to be run as a console application. If you want to design a visual application, you might prefer to use the `JFrame` template under Java GUI Forms, or an application skeleton under Java GUI Forms | Sample

< Back **Next >** Finish Cancel Help

Selecionar Java -> Java Main Class

Criando a classe Main TestaConexao (3)

29

New Java Main Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

Superclass:

Interfaces:

Criar a classe TestaConexao dentro do pacote factory

< Back Next > **Finish** Cancel Help

Criando a classe Main TestaConexao (4)

30



The screenshot shows an IDE with a project named 'Pratica01'. The left sidebar displays a package structure: 'Proj_Venda' contains 'Source Packages' (dao, factory, ConnectionFactory, TestaConexao.java, gui, modelo), 'Test Packages', 'Libraries', and 'Test Libraries'. The 'TestaConexao.java' file is selected. The main editor shows the source code for the 'TestaConexao' class. The code includes a package declaration, a license notice, an author tag, and a main method with a TODO comment.

```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
4   */
5  package factory;
6
7  /**
8   *
9   * @author Juliana
10  */
11  public class TestaConexao {
12
13      /**
14       * @param args the command line arguments
15       */
16      public static void main(String[] args) {
17          // TODO code application logic here
18      }
19
20  }
```

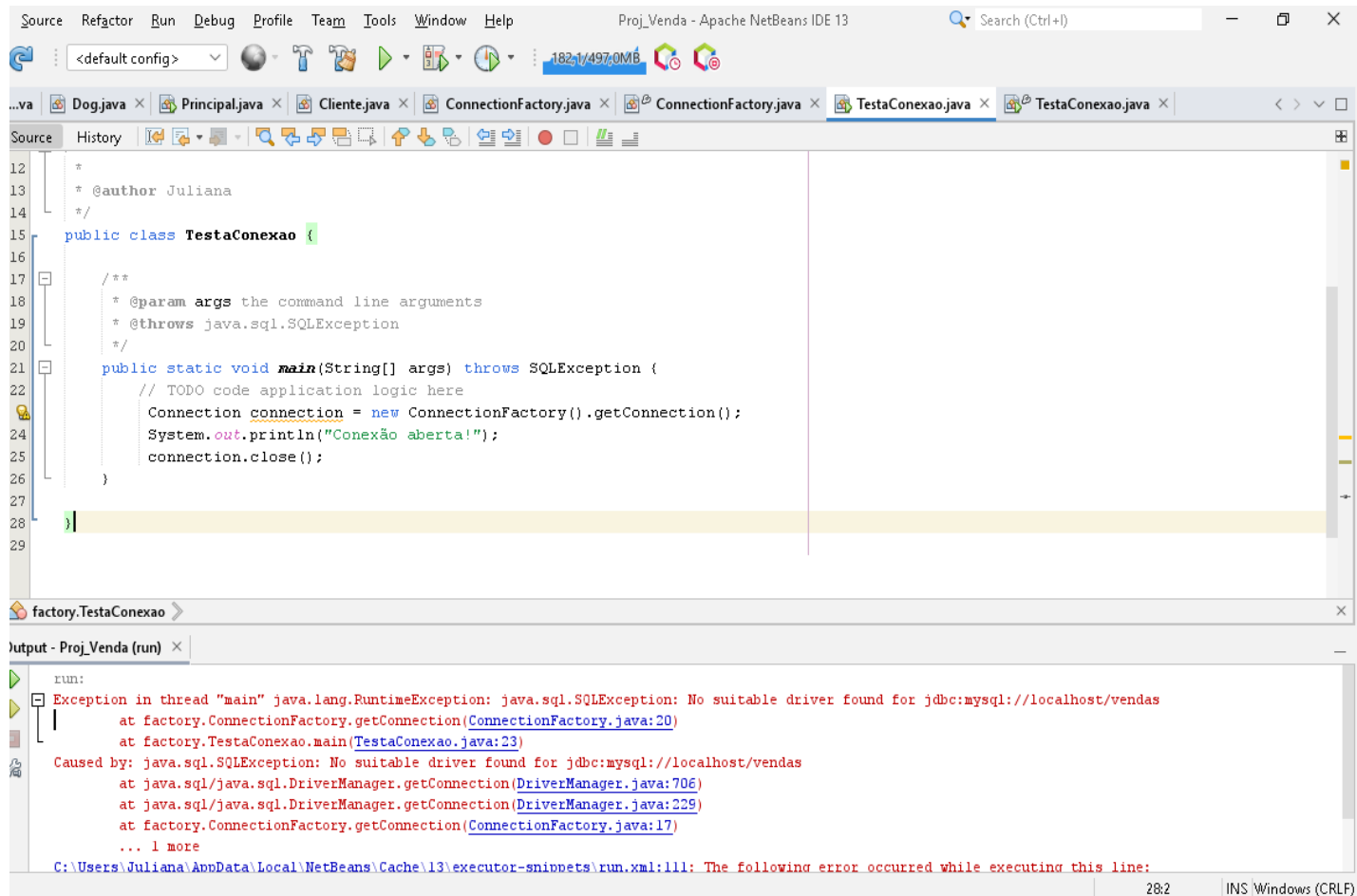
Criando a classe Main TestaConexao (5)

31

```
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
23         Connection connection = new ConnectionFactory().getConnection();
24         System.out.println("Conexão aberta!");
25         connection.close();
26     }
27
28 }
29
```

Executando a Classe TestaConexão

32



The screenshot displays the Apache NetBeans IDE interface. The main editor window shows the source code of the `TestaConexao` class. The code includes a package declaration, a class comment, and a `main` method that attempts to connect to a database using `ConnectionFactory`. The `main` method is as follows:

```
12  *  
13  * @author Juliana  
14  */  
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  
23          Connection connection = new ConnectionFactory().getConnection();  
24          System.out.println("Conexão aberta!");  
25          connection.close();  
26      }  
27  }  
28  
29
```

Below the editor, the `Output - Proj_Venda (run)` window shows the execution results. It reports a `java.lang.RuntimeException: java.sql.SQLException: No suitable driver found for jdbc:mysql://localhost/vendas` at line 20 of `ConnectionFactory.java` and line 23 of `TestaConexao.java`. The stack trace indicates the exception was caused by `java.sql.SQLException: No suitable driver found for jdbc:mysql://localhost/vendas` at line 706 of `DriverManager.java` and line 229 of `DriverManager.java`. The error message is repeated at line 17 of `ConnectionFactory.java`.

At the bottom right of the IDE, the status bar shows the text "28:2 INS Windows (CRLF)".

Executando o Projeto

33

- ❑ 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)

34

The screenshot shows the Apache NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The main window displays the source code of the `TestaConexao` class in the `factory` package. The code includes a `main` method that attempts to establish a database connection using `ConnectionFactory`. The IDE's left sidebar shows the project structure with `Proj_Venda` and its sub-packages. A context menu is open over the `Libraries` folder, with the option `Add JAR/Folder...` selected. The bottom right pane shows the output window with a runtime exception: `java.sql.SQLException: No suitable driver found for jdbc:mysql://localhost/vendas`. The exception stack trace points to the `getConnection` method in `ConnectionFactory` and the `main` method in `TestaConexao`.

```
12  *
13  * @author Juliana
14  */
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
23          Connection connection = new ConnectionFactory().getConnection();
24          System.out.println("Conexão aberta!");
25          connection.close();
26      }
27  }
28
29
```

TestaConexao - Navig... x

Members

TestaConexao

- TestaConexao()
- main(String[] args)

factory.TestaConexao

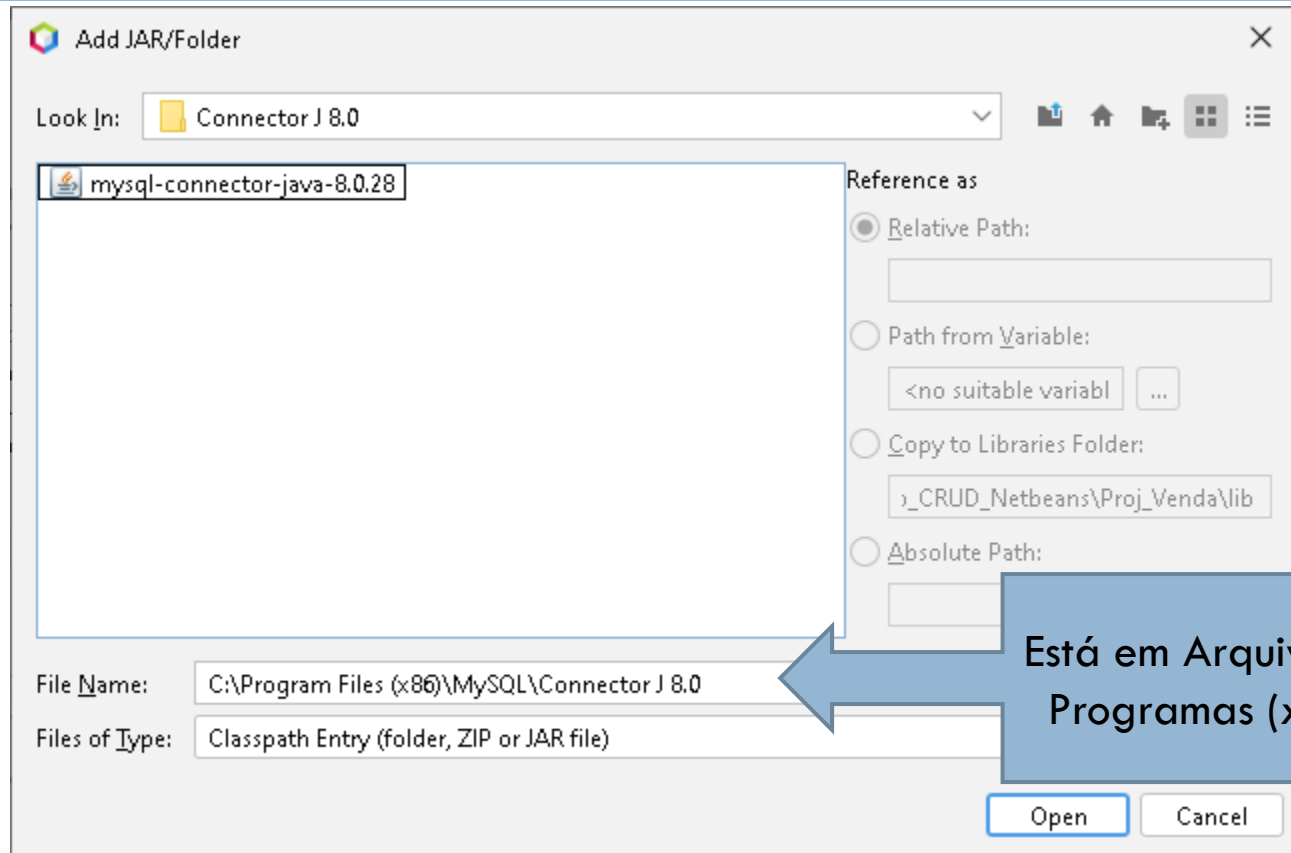
Output - Proj_Venda (run) x

run:

```
Exception in thread "main" java.lang.RuntimeException: java.sql.SQLException: No suitable driver found for jdbc:mysql://localhost/vendas
    at factory.ConnectionFactory.getConnection(ConnectionFactory.java:20)
    at factory.TestaConexao.main(TestaConexao.java:23)
Caused by: java.sql.SQLException: No suitable driver found for jdbc:mysql://localhost/vendas
    at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:706)
    at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:229)
    at factory.ConnectionFactory.getConnection(ConnectionFactory.java:17)
    ... 1 more
C:\Users\Juliana\AppData\Local\NetBeans\Cache\13\executor-snippets\run.xml:111: The following error occurred while executing this line:
```

Adicionando o drive Connector/J 8.0.28 (2)

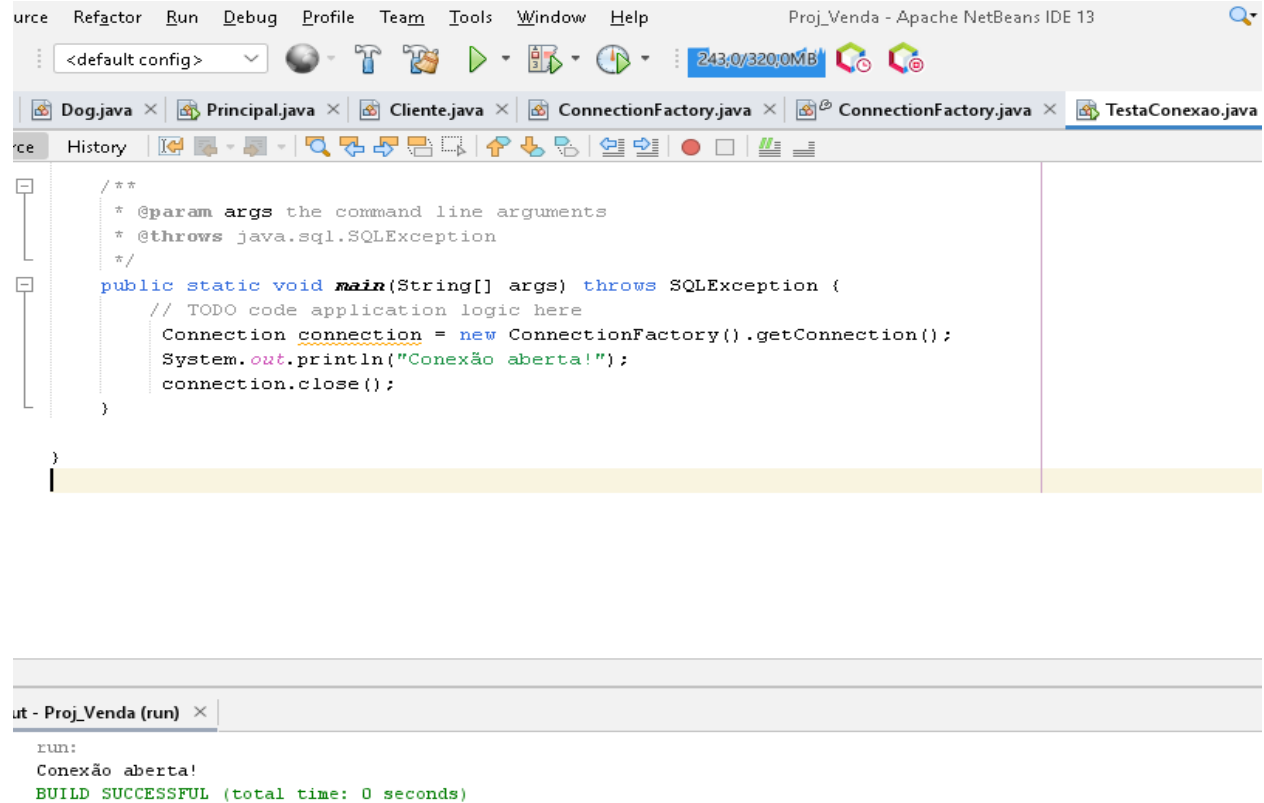
35



Está em Arquivo de Programas (x86)

Executando o Projeto

36



The screenshot shows the Apache NetBeans IDE 13 interface. The top menu bar includes File, Edit, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The toolbar contains icons for running and debugging. The project name is Proj_Venda. The tabs at the top show Dog.java, Principal.java, Cliente.java, ConnectionFactory.java, and TestaConexao.java. The TestaConexao.java file is open, showing the following code:

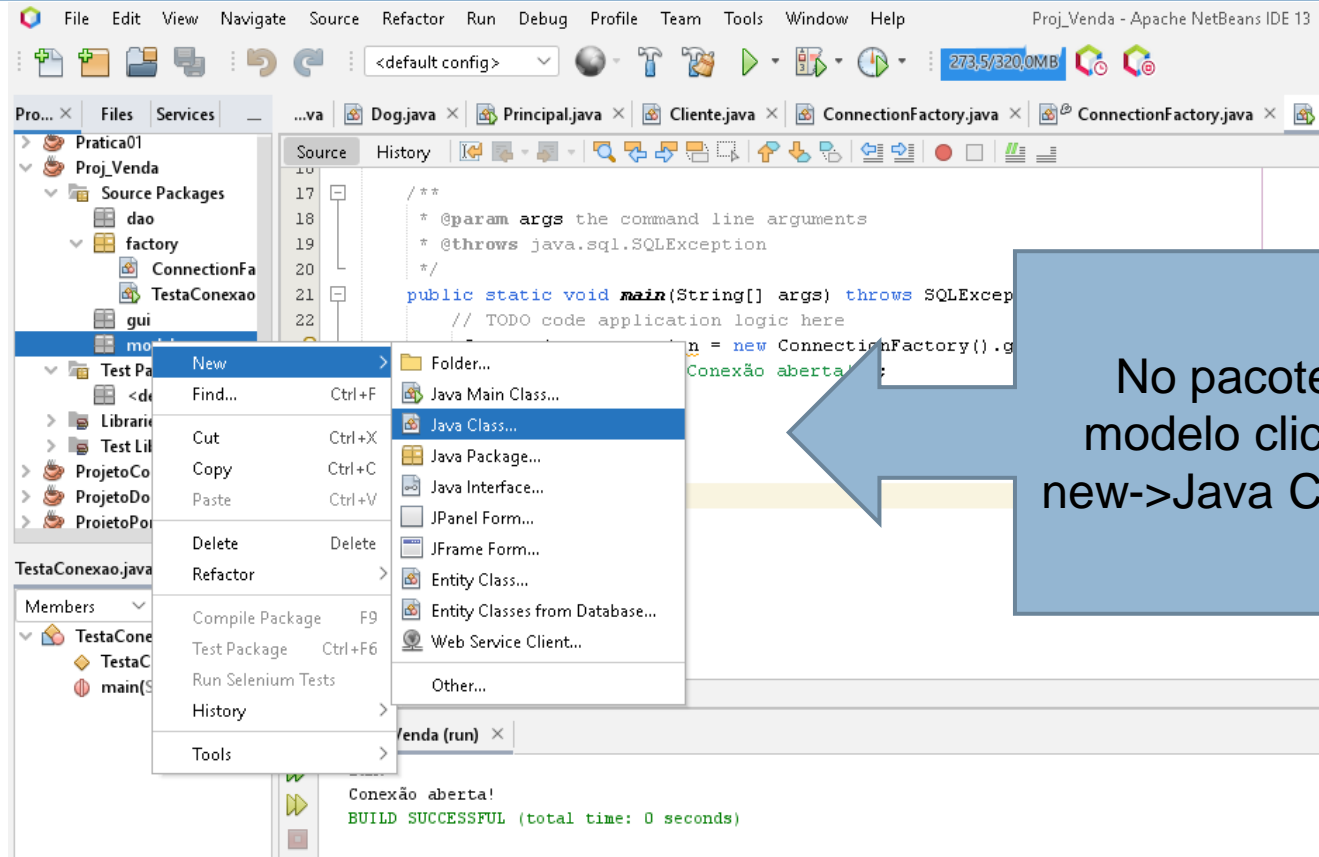
```
/**
 * @param args the command line arguments
 * @throws java.sql.SQLException
 */
public static void main(String[] args) throws SQLException {
    // TODO code application logic here
    Connection connection = new ConnectionFactory().getConnection();
    System.out.println("Conexão aberta!");
    connection.close();
}
```

The bottom output window, titled 'Output - Proj_Venda (run)', shows the following output:

```
run:
Conexão aberta!
BUILD SUCCESSFUL (total time: 0 seconds)
```

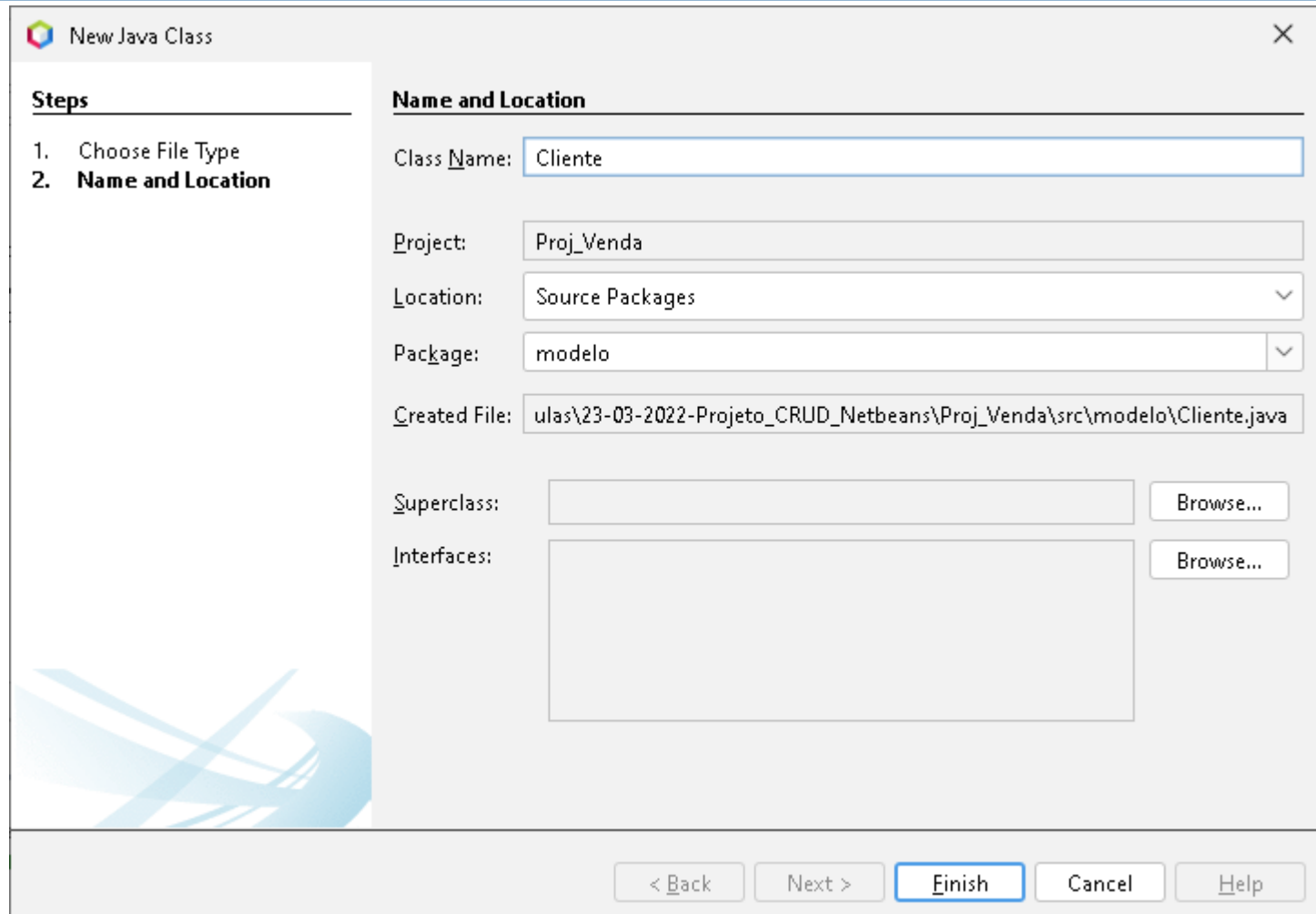
Criando a classe Cliente no Pacote Modelo (1)

37



Criando a classe Cliente no Pacote Modelo (2)

38



The image shows a 'New Java Class' dialog box in an IDE. The 'Steps' panel on the left indicates the current step is '2. Name and Location'. The 'Name and Location' section contains the following fields: 'Class Name' with the value 'Cliente', 'Project' with 'Proj_Venda', 'Location' with 'Source Packages', and 'Package' with 'modelo'. The 'Created File' field shows the full path: 'ulas\23-03-2022-Projeto_CRUD_Netbeans\Proj_Venda\src\modelo\Cliente.java'. There are 'Browse...' buttons for 'Superclass' and 'Interfaces'. At the bottom, there are navigation buttons: '< Back', 'Next >', 'Finish' (highlighted with a blue border), 'Cancel', and 'Help'.

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

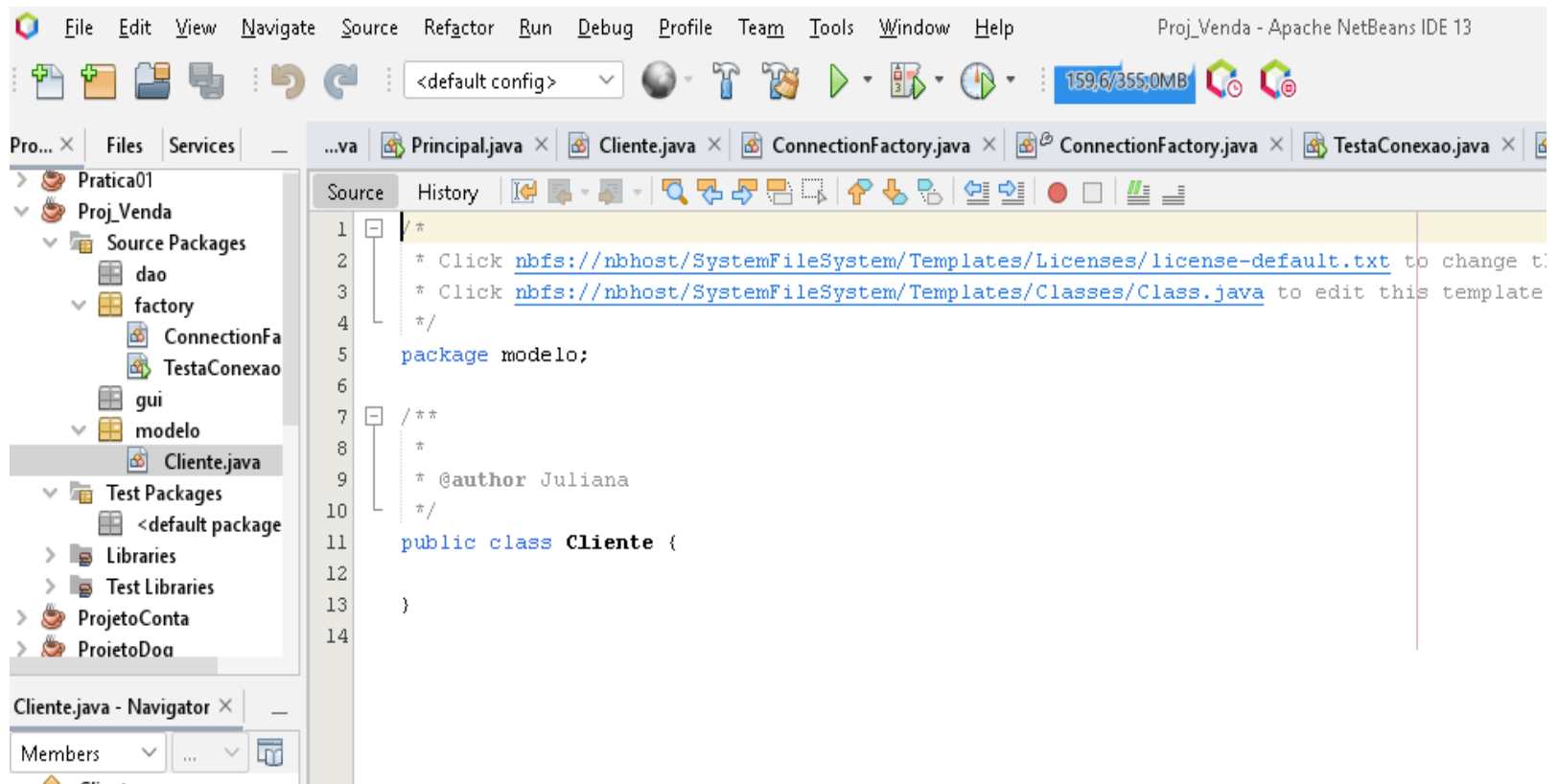
Superclass:

Interfaces:

< Back Next > **Finish** Cancel Help

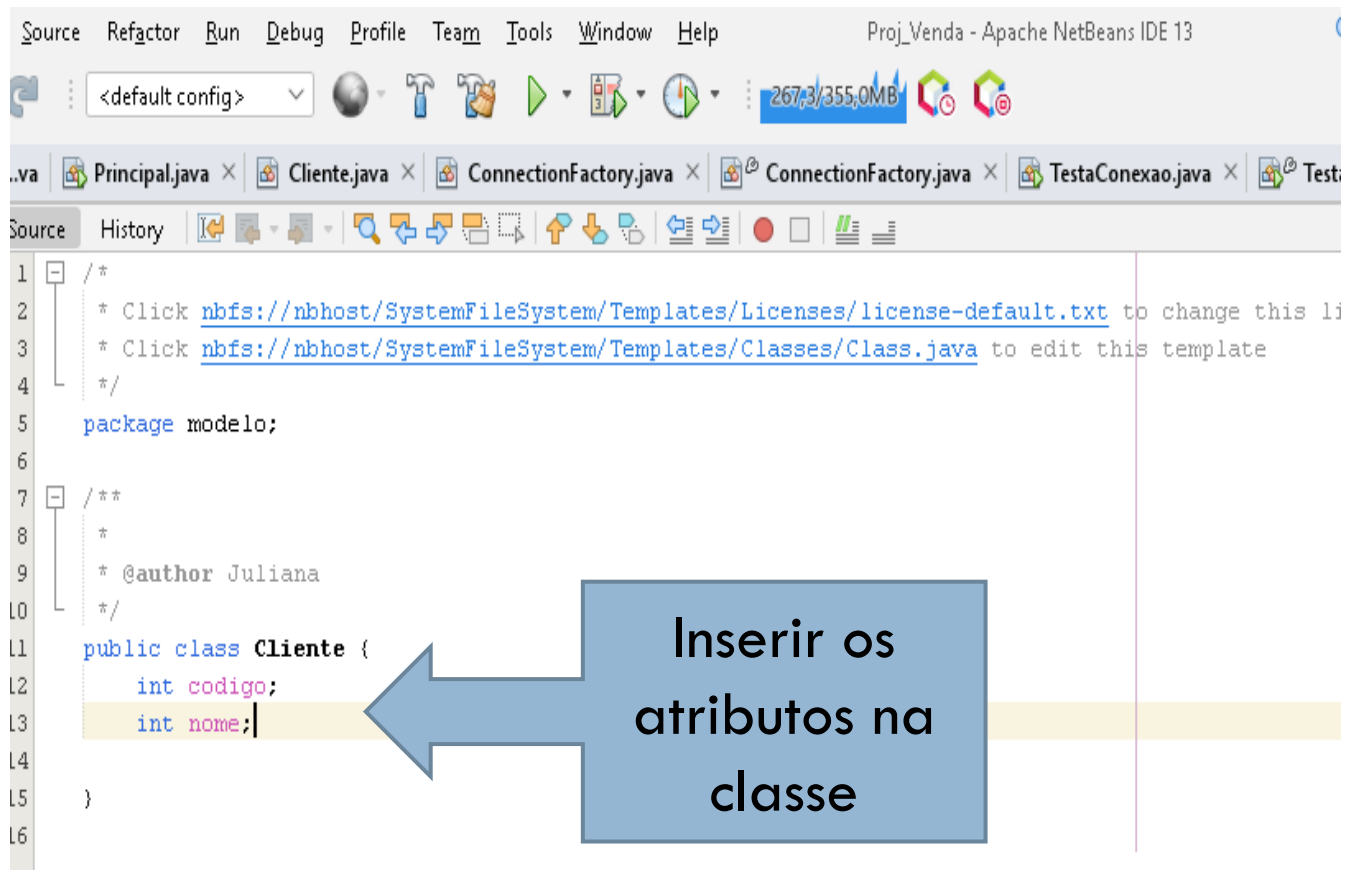
Classe Cliente

39



Classe Cliente (Atributos)

40



Source Refactor Run Debug Profile Team Tools Window Help Proj_Venda - Apache NetBeans IDE 13

<default config> 267,3/355,0MB

Principal.java x Cliente.java x ConnectionFactory.java x ConnectionFactory.java x TestaConexao.java x TestaConexao.java x

Source History

```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5   package modelo;
6
7   /**
8    *
9    * @author Juliana
10   */
11  public class Cliente {
12      int codigo;
13      int nome;
14  }
15
16
```

Inserir os atributos na classe

Classe Clientes (métodos)

41

Proj_Venda - Apache NetBeans IDE 13

271.8/355.0MB

Source History

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package modelo;
6
7  /**
8   *
9   * @author Juliana
10  */
11  public class Cliente {
12      int codigo;
13      int nome;
14  }
15
16
```

Cliente - Navigator

Members

Cliente

- Cliente()
- codigo : int
- nome : int

Output - Proj_Venda (run)

```
run:
Conexão aberta!
BUILD SUCCESSFUL (total time: 0 s)
```

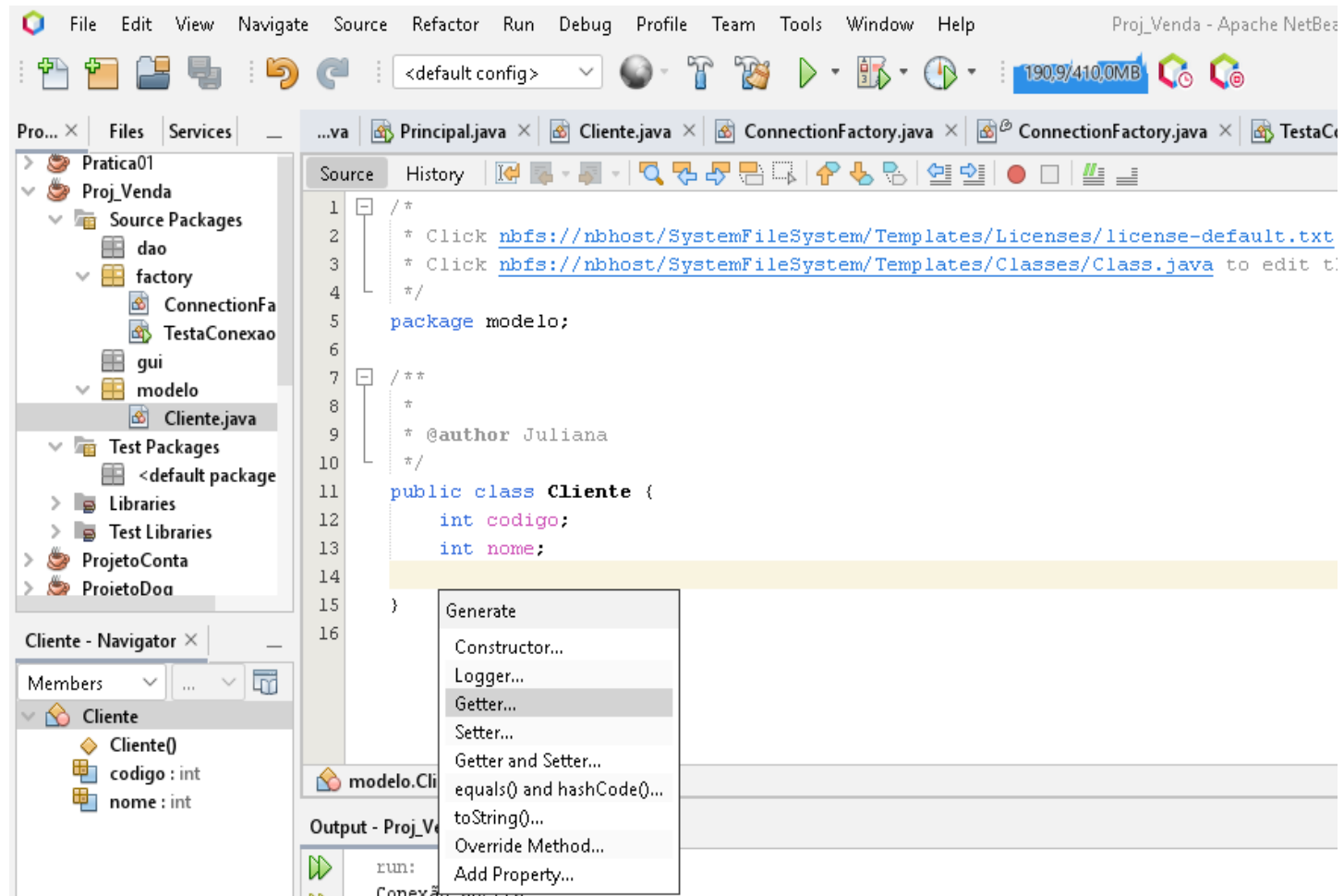
Context Menu:

- Navigate
- Show Javadoc Alt+F1
- Find Usages Alt+F7
- Call Hierarchy
- Insert Code... Alt+Insert**
- Fix Imports Ctrl+Shift+I
- Refactor
- Format Alt+Shift+F
- Run File Shift+F6
- Debug File Ctrl+Shift+F5
- Test File Ctrl+F6
- Debug Test File Ctrl+Shift+F6
- Run Focused Test Method
- Debug Focused Test Method
- Run Into Method
- New Watch... Ctrl+Shift+F7
- Toggle Line Breakpoint Ctrl+F8
- Profile
- Cut Ctrl+X
- Copy Ctrl+C
- Paste Ctrl+V
- Code Folds

Clicar com o botão direito na classe depois dos atributos -> insert code

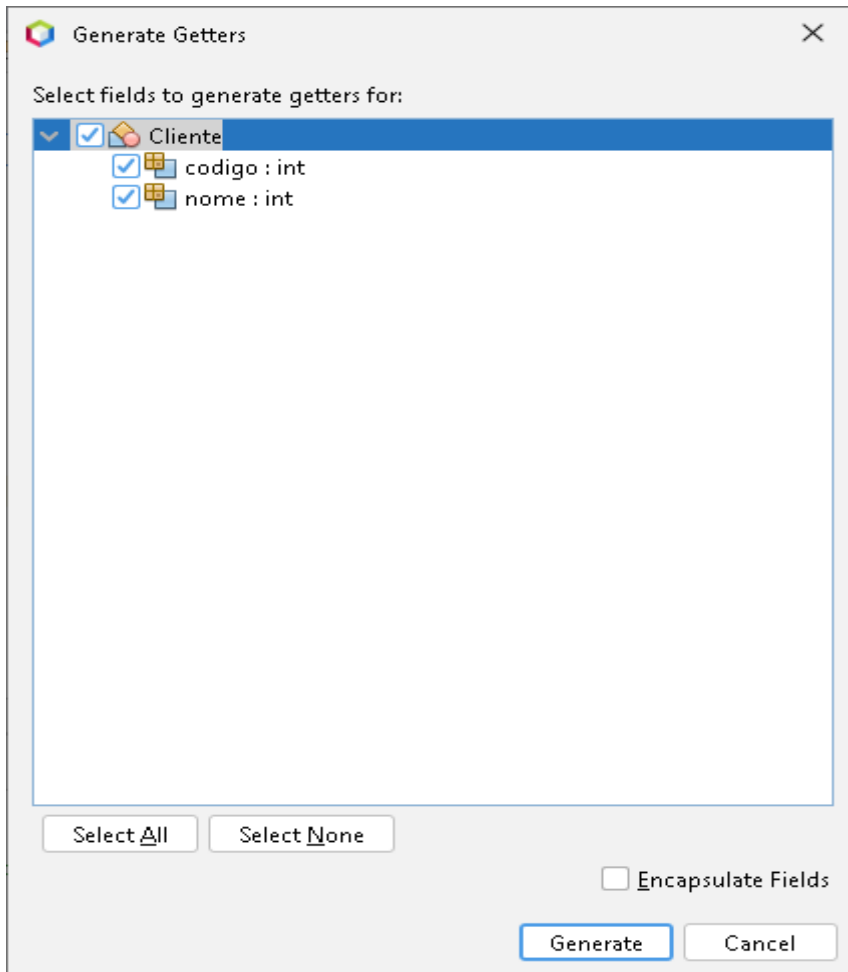
Classe Clientes (Get)

42



Classe Clientes (Get)

43



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

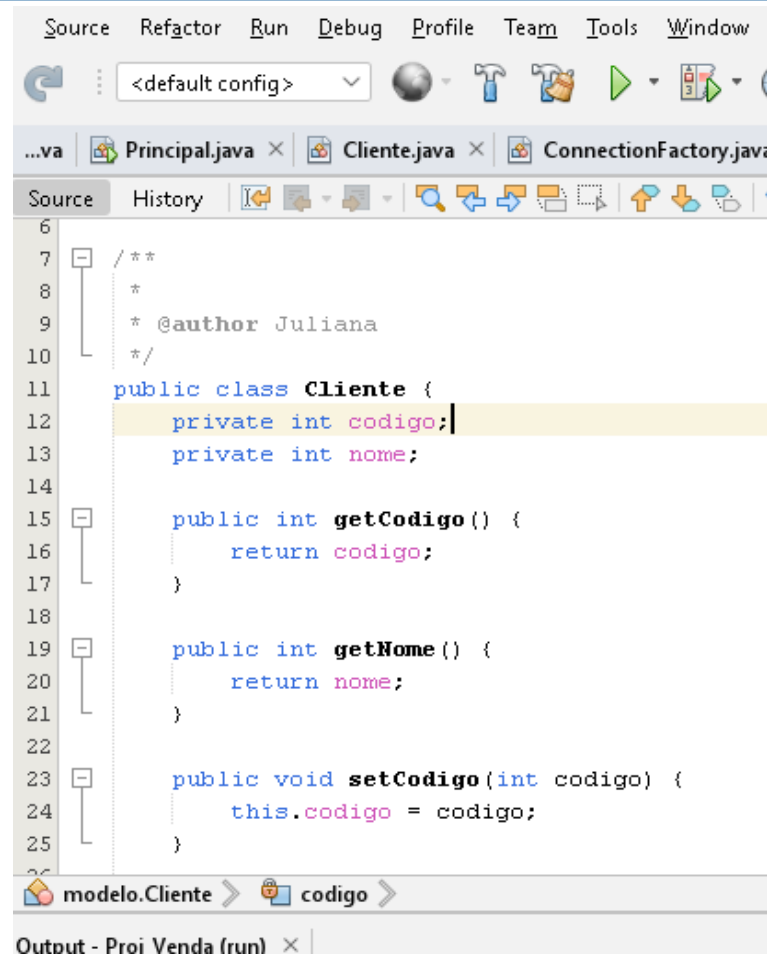
Classe Cliente (Setters)

44

- Repedir o processo anterior para gerar os métodos setters.

Classe Cliente

45



```
Source Refactor Run Debug Profile Team Tools Window
<default config>
...va Principal.java x Cliente.java x ConnectionFactory.java
Source History
6
7 /**
8  *
9  * @author Juliana
10 */
11 public class Cliente {
12     private int codigo;
13     private int nome;
14
15     public int getCodigo() {
16         return codigo;
17     }
18
19     public int getNome() {
20         return nome;
21     }
22
23     public void setCodigo(int codigo) {
24         this.codigo = codigo;
25     }
26
27     modelo.Cliente > codigo >
Output - Proi Venda (run) x
```

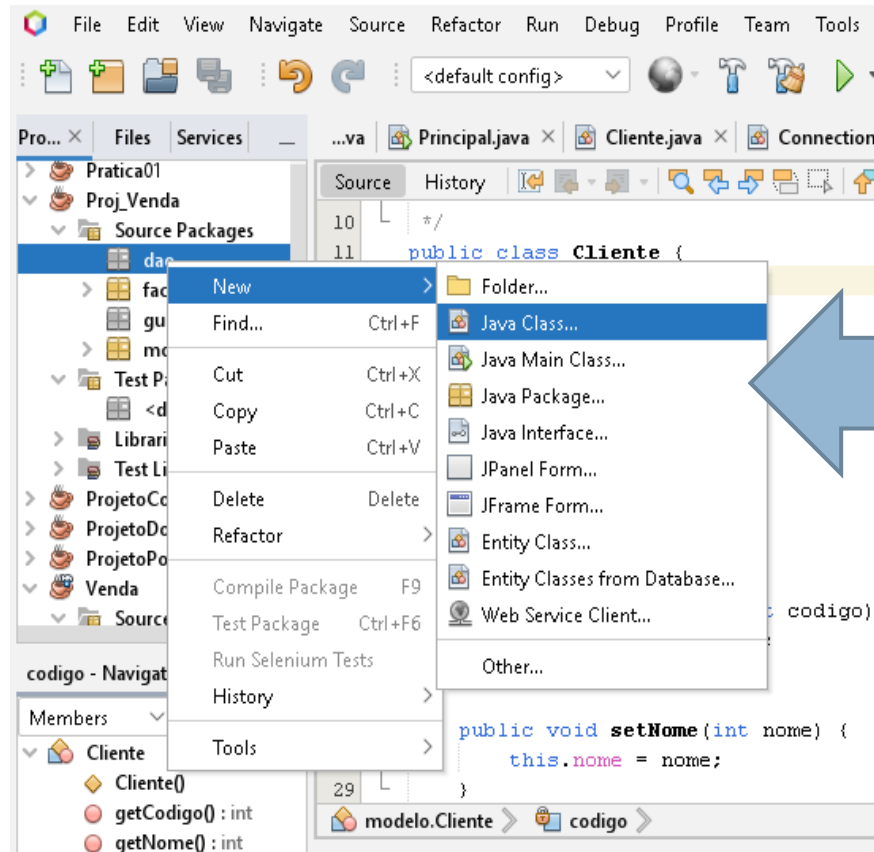
Classe ClienteDAO

46

- Crie no **pacote DAO** a classe ClienteDAO: **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

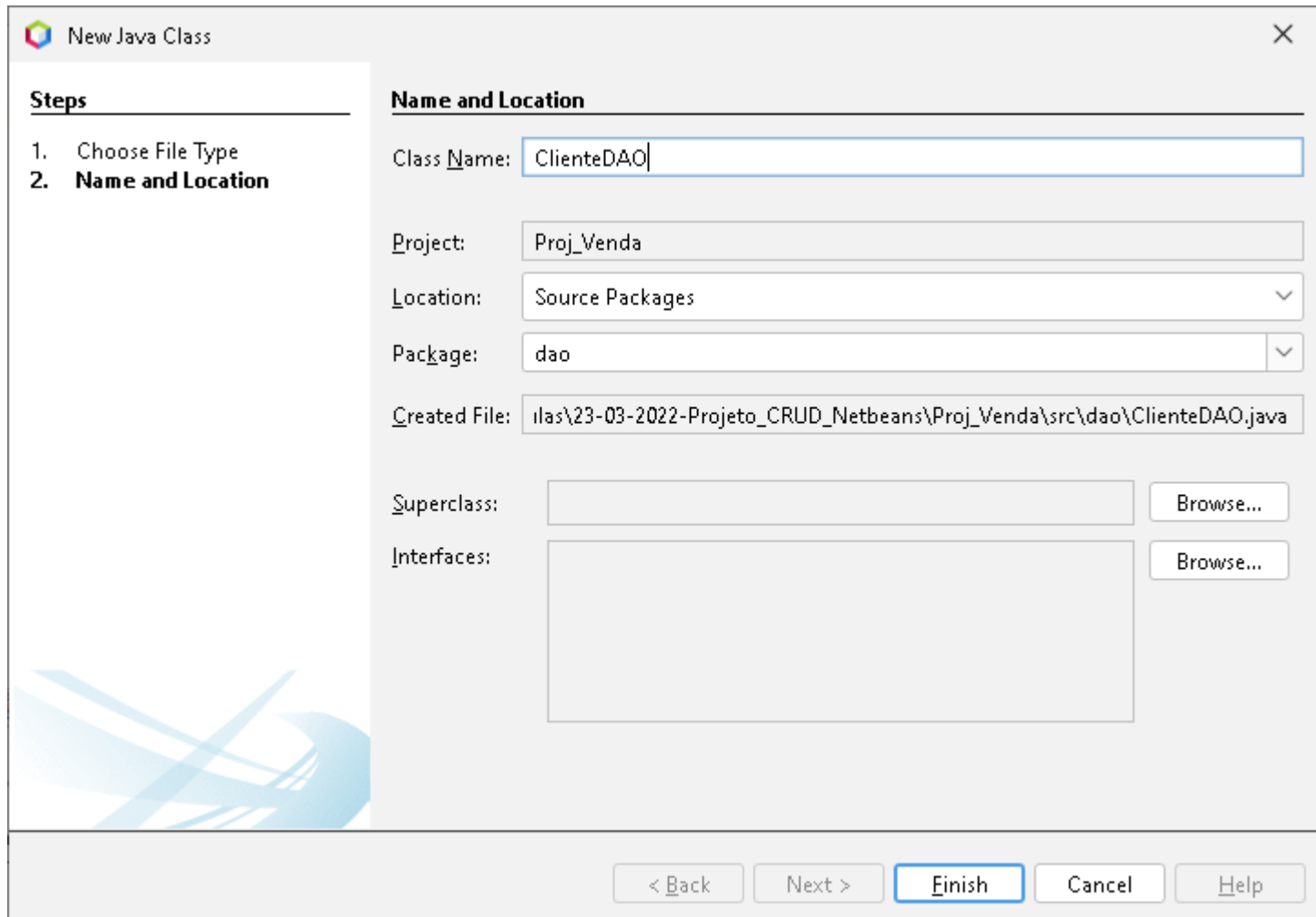
47



No pacote dao
clique new->Java
Class

Classe ClienteDAO

48



The image shows a 'New Java Class' dialog box with a light gray background and a blue header bar. The title bar says 'New Java Class' with a close button (X) on the right. The dialog is divided into two main sections: 'Steps' on the left and 'Name and Location' on the right. The 'Steps' section has a list with two items: '1. Choose File Type' and '2. Name and Location', with the second item being bolded. The 'Name and Location' section contains several fields: 'Class Name' with the text 'ClienteDAO', 'Project' with 'Proj_Venda', 'Location' with a dropdown menu showing 'Source Packages', 'Package' with a dropdown menu showing 'dao', and 'Created File' with a text field showing the full path. Below these are 'Superclass' and 'Interfaces' sections, each with a text field and a 'Browse...' button. At the bottom, there are five buttons: '< Back', 'Next >', 'Finish' (highlighted with a blue border), 'Cancel', and 'Help'.

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

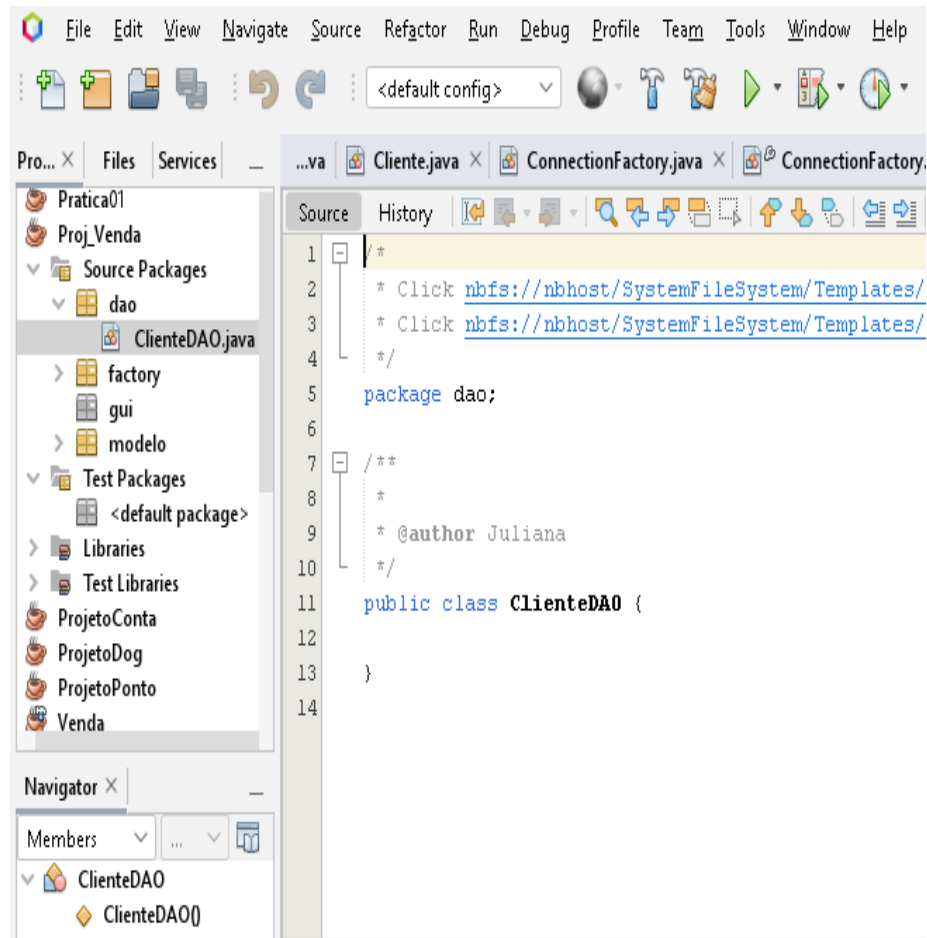
Created File:

Superclass:

Interfaces:

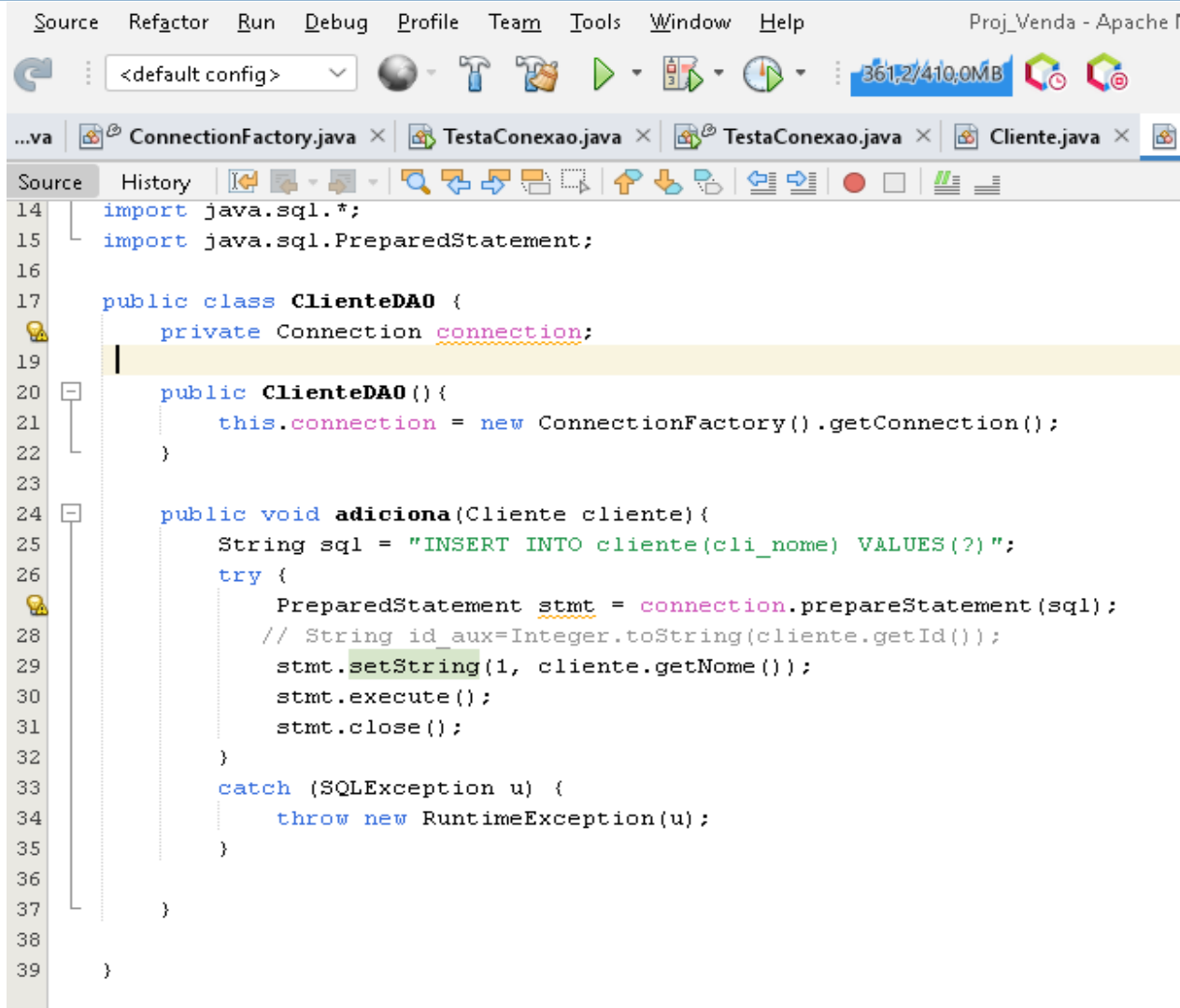
Classe ClienteDAO

49



Classe ClienteDAO

50



```
Source  Refactor  Run  Debug  Profile  Team  Tools  Window  Help  Proj_Venda - Apache I
<default config> 361,2/410,0MB
...va  ConnectionFactory.java  TestaConexao.java  TestaConexao.java  Cliente.java
Source  History
14  import java.sql.*;
15  import java.sql.PreparedStatement;
16
17  public class ClienteDAO {
18      private Connection connection;
19
20      public ClienteDAO() {
21          this.connection = new ConnectionFactory().getConnection();
22      }
23
24      public void adiciona(Cliente cliente) {
25          String sql = "INSERT INTO cliente(cli_nome) VALUES(?)";
26          try {
27              PreparedStatement stmt = connection.prepareStatement(sql);
28              // String id_aux=Integer.toString(cliente.getId());
29              stmt.setString(1, cliente.getNome());
30              stmt.execute();
31              stmt.close();
32          }
33          catch (SQLException u) {
34              throw new RuntimeException(u);
35          }
36      }
37  }
38
39  }
```

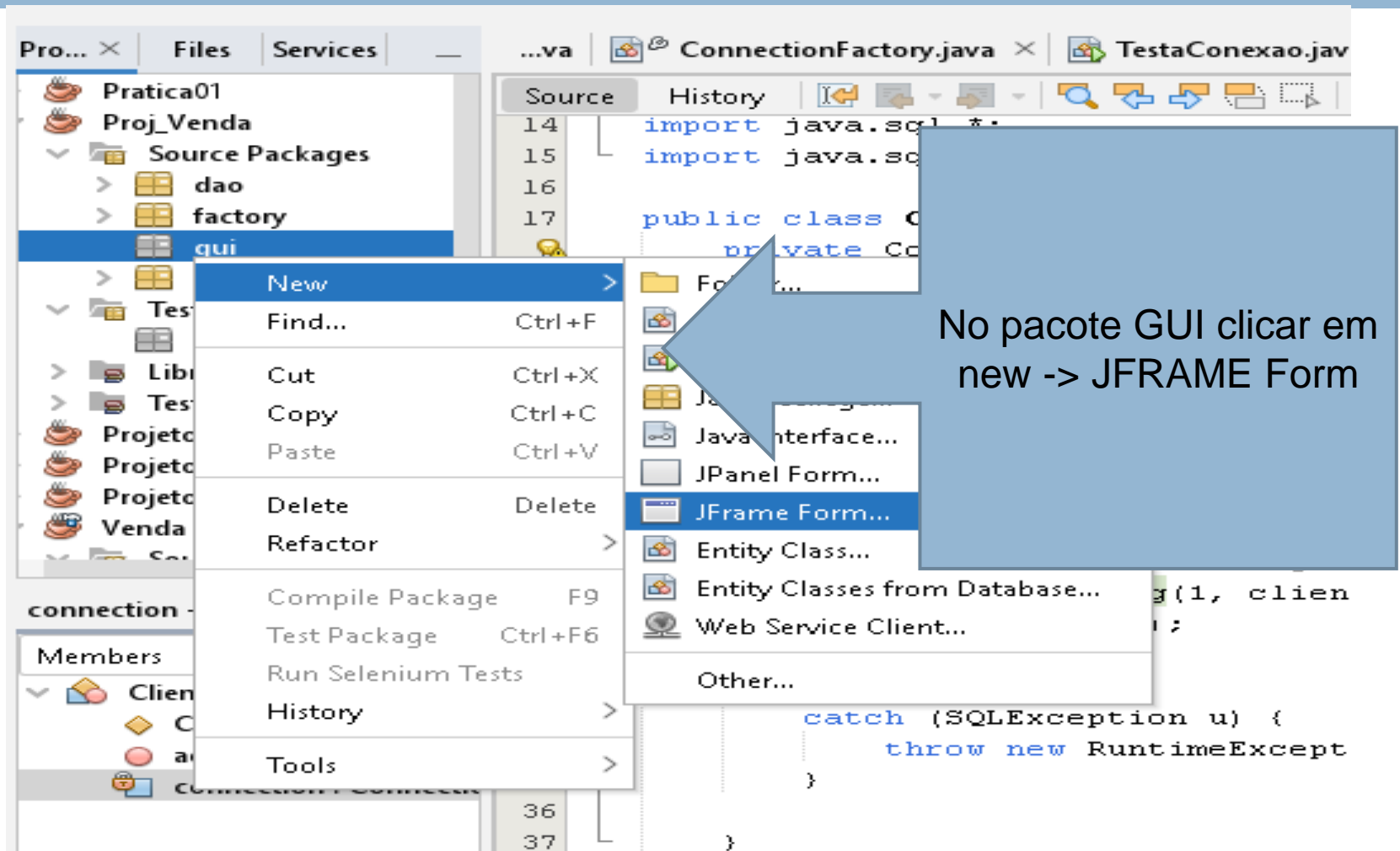
GUI (Graphical User Interface ou Interface Gráfica de Usuário)

51

- 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

52



Classe ClienteGUI

53

New JFrame Form

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

Superclass:

Interfaces:

< Back Next > **Finish** Cancel Help

Inserindo Label

54

1) Selecionar Design

The gaps between components can be selected and adjusted via mouse, or edited in a dialog.
If a gap is hard to select, invoke *Edit Layout Space* context menu action on a neighbor component.

Cadastro de Cliente

Vertical resizable gap: 268

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

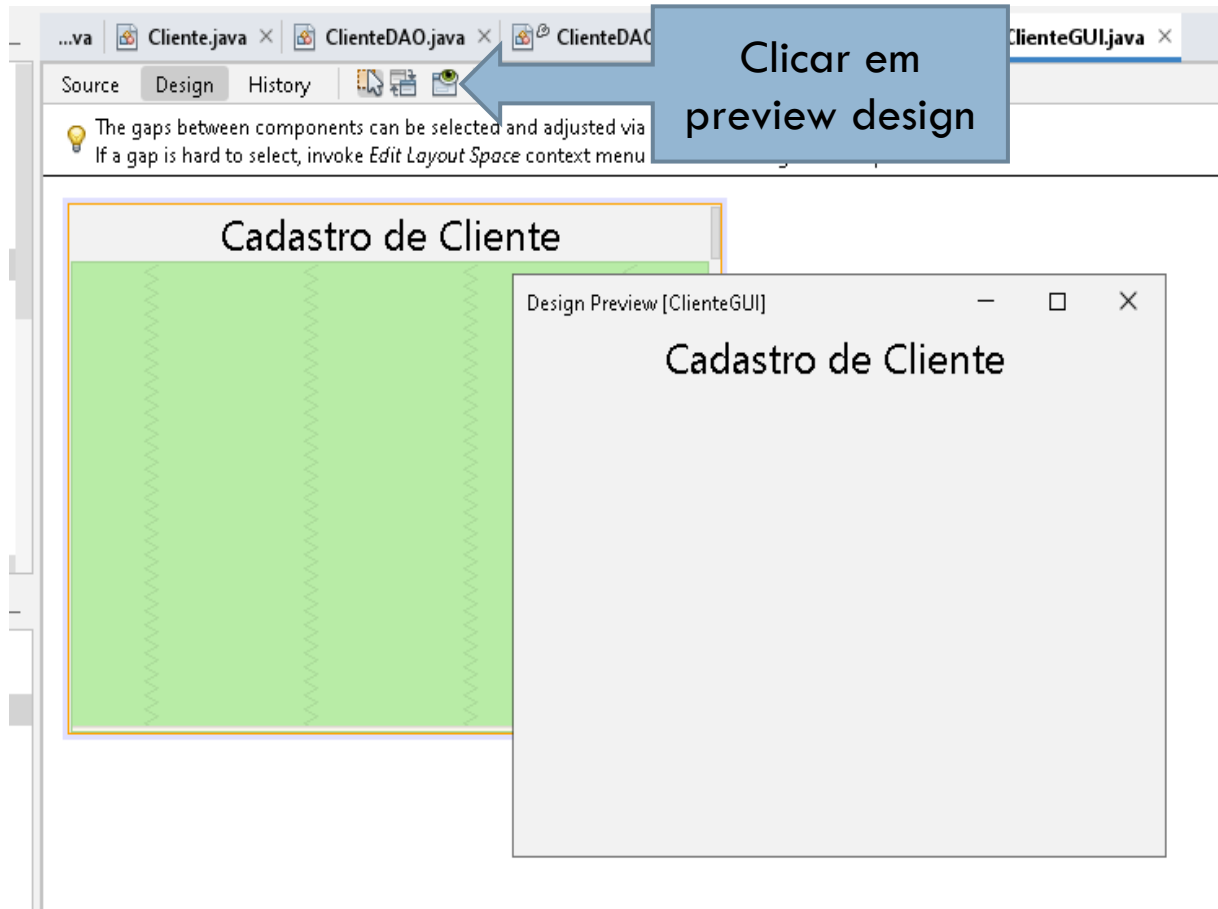
Properties

Properties	Events	Code
defaultCloseOperation		EXIT_ON_CLOSE
title		
Other Properties		
alwaysOnTop	<input type="checkbox"/>	
alwaysOnTopSupported	<input checked="" type="checkbox"/>	
autoRequestFocus	<input checked="" type="checkbox"/>	
background	<input type="checkbox"/>	[242,242,242]

[JFrame]

Visualizando o Design

55



Inserindo Panel

56

The screenshot shows the Apache NetBeans IDE 13 interface. The main window is titled 'Cadastro de Cliente' and is in Design view. A red arrow points from the 'JPanel' component in the 'Swing Containers' palette to a new panel being added to the design view. The 'Swing Containers' palette is open, showing various components like Panel, Tabbed Pane, Split Pane, Scroll Pane, Tool Bar, Desktop Pane, etc. The 'Properties' window for the selected 'JPanel' is also visible, showing properties like background, border, foreground, and alignment.

Source Design History

The Navigator window displays a tree hierarchy of components in the opened form.

Cadastro de Cliente

Swing Containers

- Panel
- Tabbed Pane
- Split Pane
- Scroll Pane
- Tool Bar
- Desktop Pane

javax.swing.JPanel
A generic lightweight container.

Swing Menus

- Label
- OK Button
- Toggle Button
- Check Box
- Radio Button
- Button Group
- Combo Box
- List
- Text Field
- Text Area
- Scroll Bar
- Slider
- Progress Bar
- Formatted Field
- Password Field
- Spinner
- Separator
- Text Pane
- Editor Pane
- Tree
- Table

jPanel1 [JPanel] - Properties

Properties	Events	Code
Properties		
background		[242,242,242]
border		(No Border)
foreground		[0,0,0]
toolTipText		
Other Properties		
UIClassID		PanelUI
alignmentX		0.5

Alterando Propriedades do Panel

57

Selecionar Titled Border e em Title digitar Cadastrar novo Cliente

1) Clicar nos (...) em border.

Proj_Venda - Apache NetBeans IDE 13

Search (Ctrl+I)

380/6/1238MB

...va ClienteDAO.java x ClienteDAO.java x ClienteGUI.java x ClienteGUI.java x

Source Design History

Use the Design This Container action from

Cadastro de C

jPanel1 [JPanel] - border

Set jPanel1's border property using: Border customizer

Available Borders

- (No Border)
- Bevel Border
- Compound Border
- Empty Border
- Etched Border
- Line Border
- Matte Border
- Soft Bevel Border
- Titled Border**

Properties

Border	[FlatLineBorder]
Title	Cadastrar novo Cliente
Color	[0,0,0]
Font	Segoe UI 12 Plain
Justification	Default Justification
Position	Default Position

Justification

(int) Horizontal justification of the title.

OK Cancel Help

Swing Containers

- Panel
- Scroll Pane
- Internal Frame
- Tabbed Pane
- Tool Bar
- Layered Pane
- Split Pane
- Desktop Pane

Swing Controls

- Label
- Check Box
- Combo Box
- Text Area
- Progress Bar
- Spinner
- Editor Pane
- OK Button
- Radio Button
- List
- Slider
- File Chooser
- ToggleButton
- Button Group
- Text Field

Swing Menus

jPanel1 [JPanel] - Properties

Properties	Events	Code
background		[242,242,242]
border		(No Border)
foreground		[0,0,0]
toolTipText		
Other Properties		
UIClassID		PanelUI
alignmentX		0.5

border

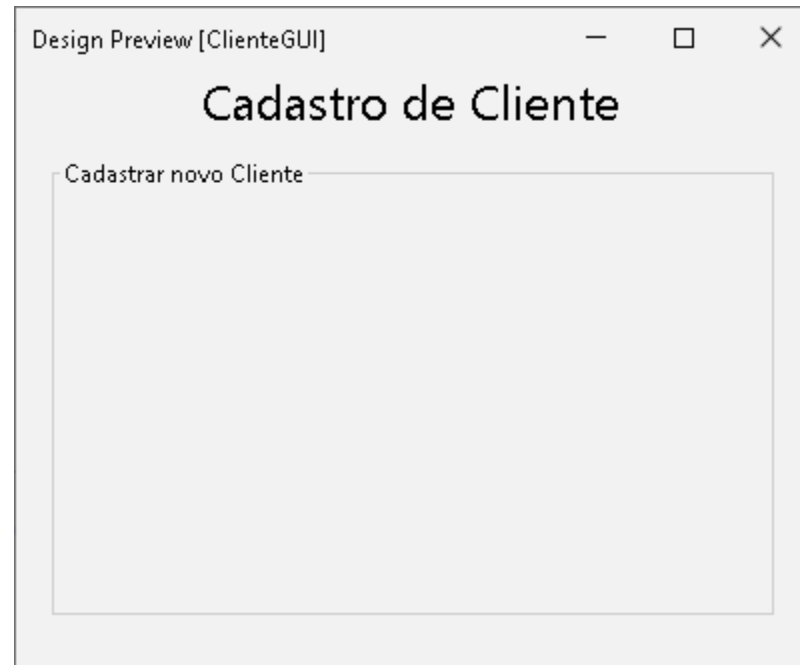
(javax.swing.border.Border) The component's border.

Output - Proj_Venda (run)

```
run:
Conexão aberta!
BUILD SUCCESSFUL (total time: 0 seconds)
```

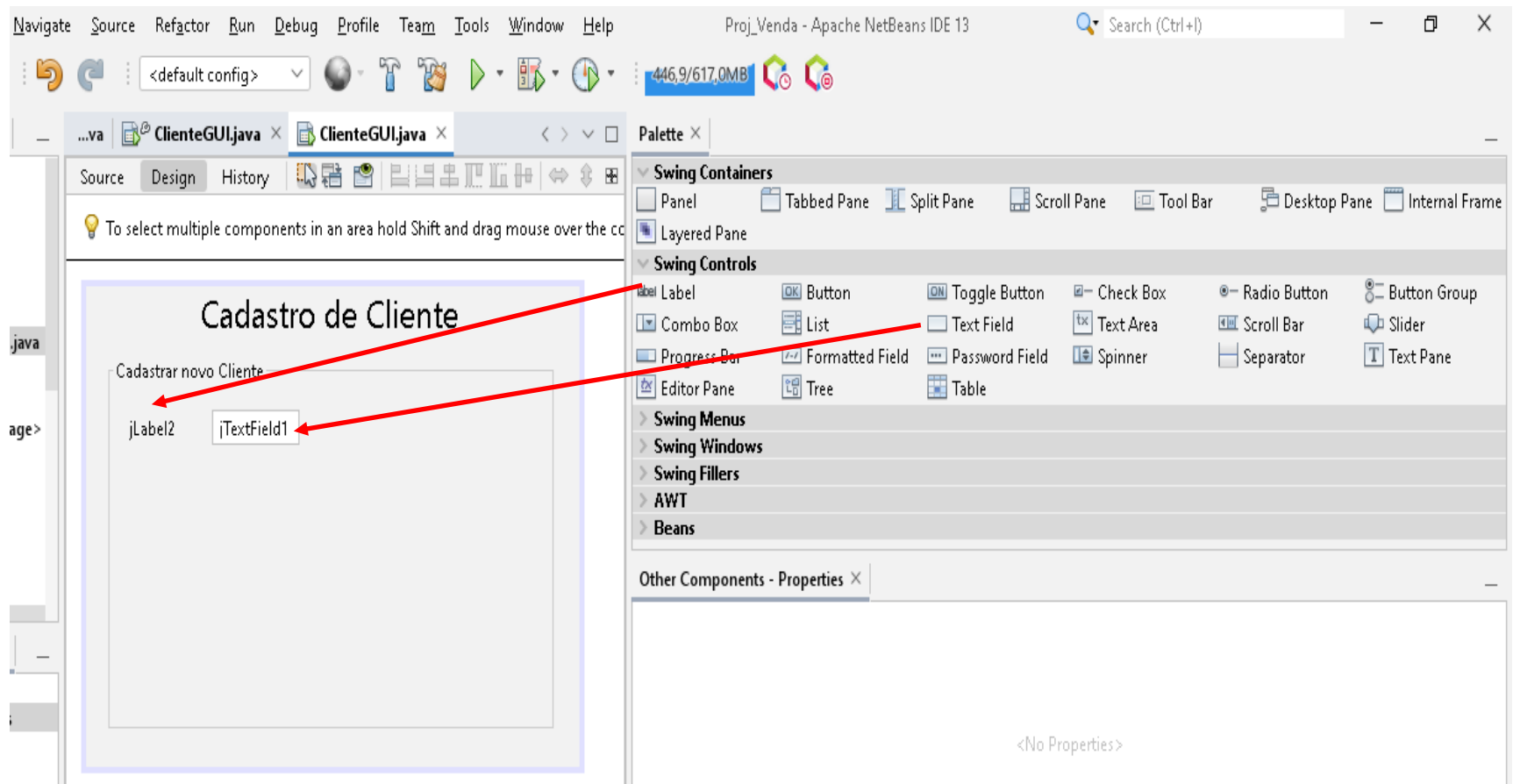
Cadastro de Cliente

58



Cadastro de Cliente

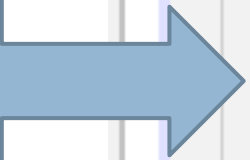
59



Cadastro de Cliente

60

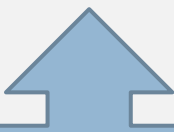
Properties
-> text =
Nome



Cadastro de Cliente

Cadastrar novo Cliente

Nome:



Properties ->
text = tirar o
texto

Cadastro de Cliente

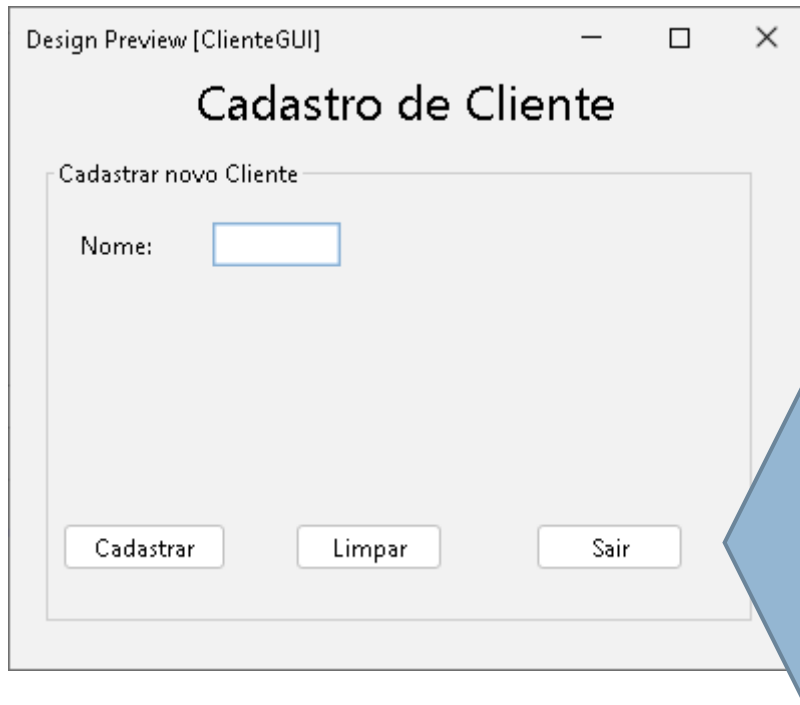
61

The screenshot shows the Apache NetBeans IDE interface. The main window displays the 'Cadastro de Cliente' GUI design, which includes a form titled 'Cadastrar novo Cliente' with a 'Nome:' label and a text field. Below the form are three buttons labeled 'jButton1', 'jButton2', and 'jButton3'. A red arrow points from the 'jButton1' button in the design to the 'jButton1 [JButton] - Properties' window on the right. The Properties window shows the 'Properties' tab for the selected button, with the 'text' property set to 'jButton1'.

Alterar em Propriedades -> text
os nomes dos botões
jButton1=Cadastrar
jButton2=Limpar
jButton3=Sair

Cadastro de Cliente – Evento Sair

62



Em Design dê um duplo clique no botão Sair.

Evento Sair

63

Proj_Venda - Apache NetBeans IDE 13

424,9/557,0MB

Pro... x Files Services

Pratica01
Proj_Venda
Source Packages
dao
factory
gui
ClienteGUI.java
modelo
Test Packages
<default package>
Libraries
Test Libraries
ProjetoConta
ProjetoDog
ProjetoPonto
Venda

Navigator x

Members

ClienteGUI :: JFrame
ClienteGUI()
initComponents()
jButton3ActionPerformed
main(String[] args)
jButton1 : JButton
jButton2 : JButton
jButton3 : JButton
jLabel1 : JLabel
jLabel2 : JLabel

Source Design History

22
23
24
25
26
117
119
120
121
122
123
124
125
126
127
148
149
152
153
154
155
156
157
158

Generated Code

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
}  
  
/**  
 * @param evt  
 */  
public void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
    // Create and display the form */  
    java.awt.EventQueue.invokeLater(new Runnable() {  
        public void run() {  
            new ClienteGUI().setVisible(true);  
        }  
    });  
}  
  
// Variables declaration - do not modify  
private javax.swing.JButton jButton1;
```

gui.ClienteGUI > jButton3ActionPerformed >

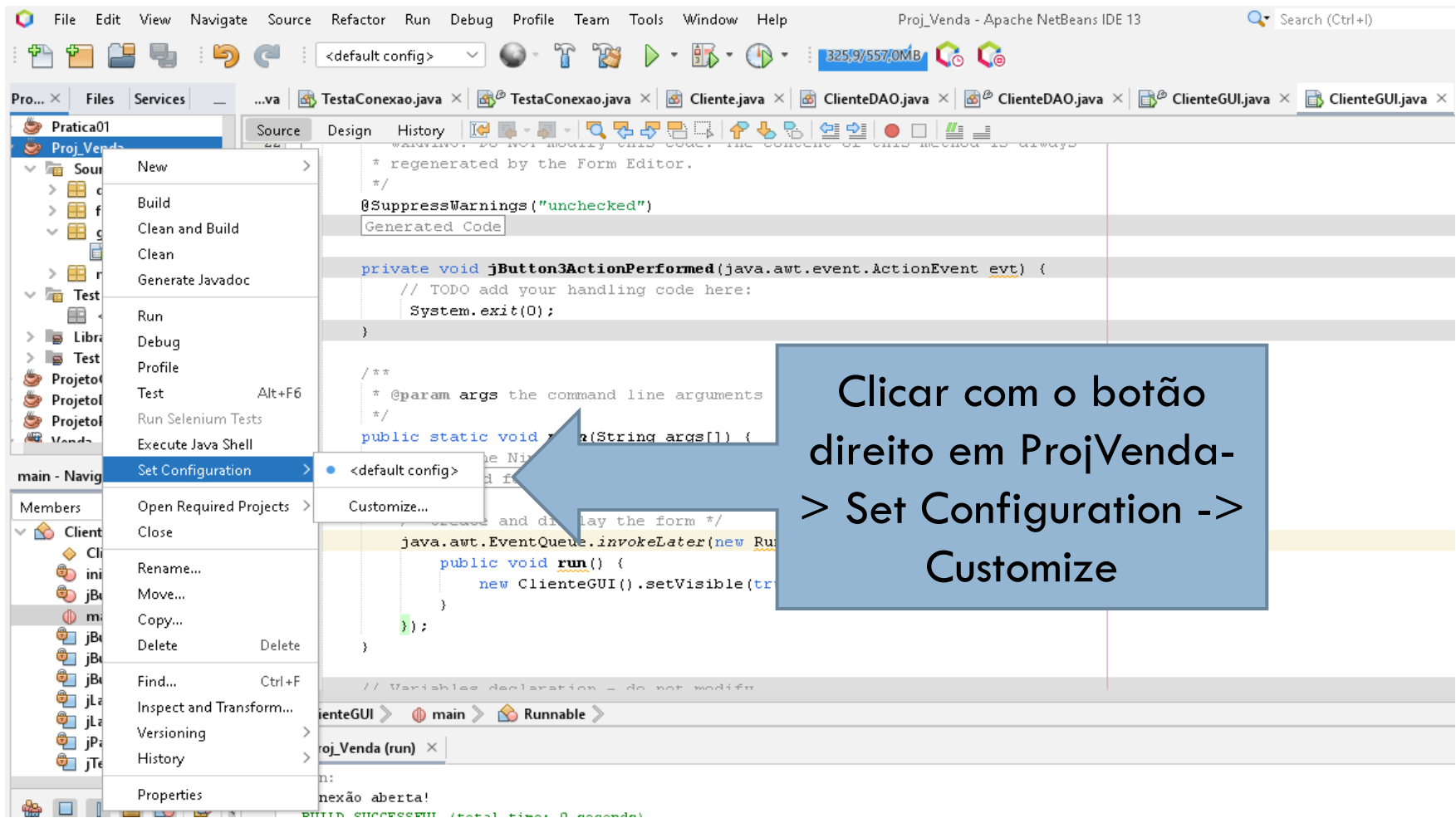
Evento Sair

64

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

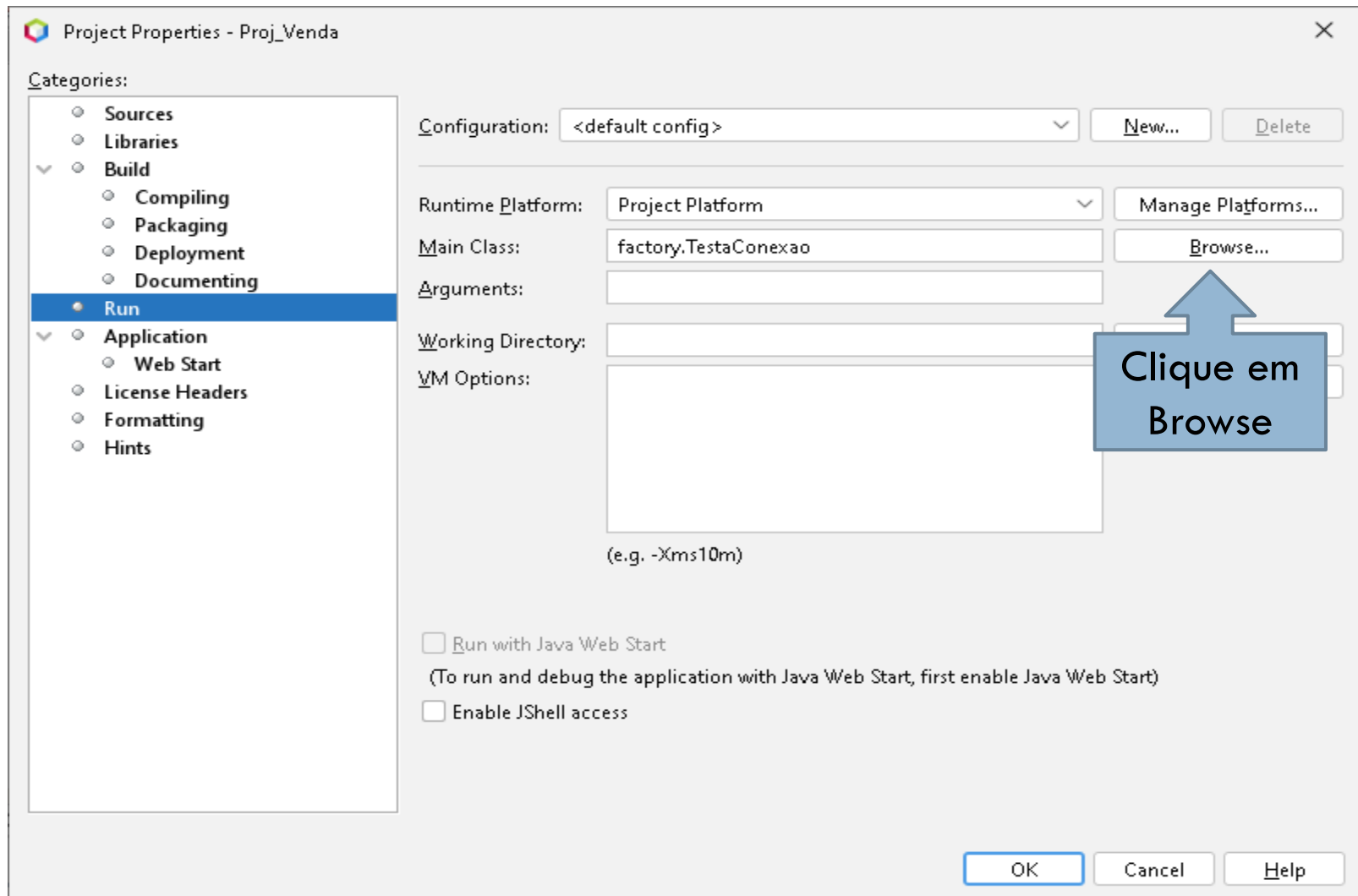

Alterando a Classe Main

65



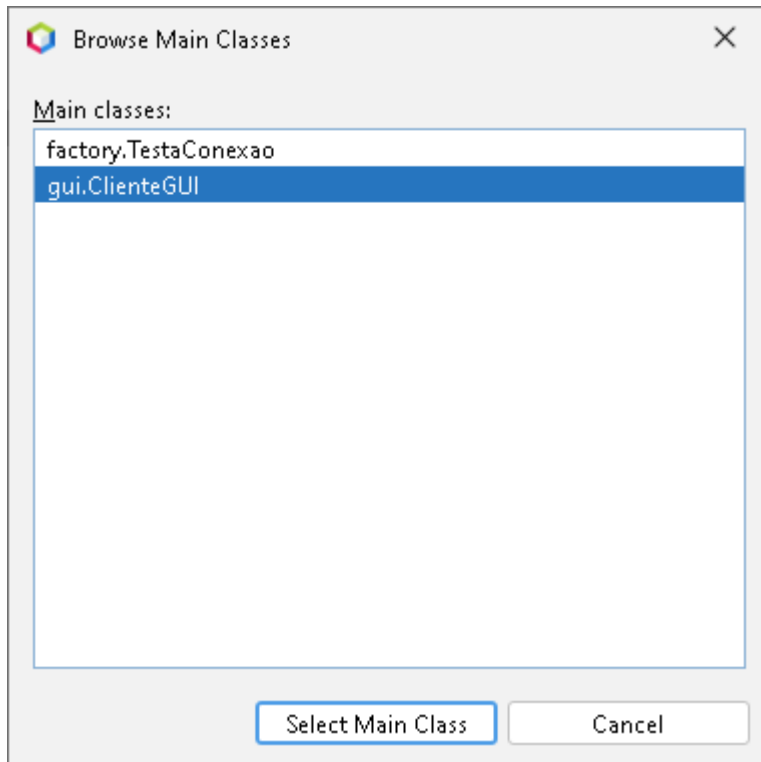
Alterando a Classe Main

66



Alterando a Classe Main

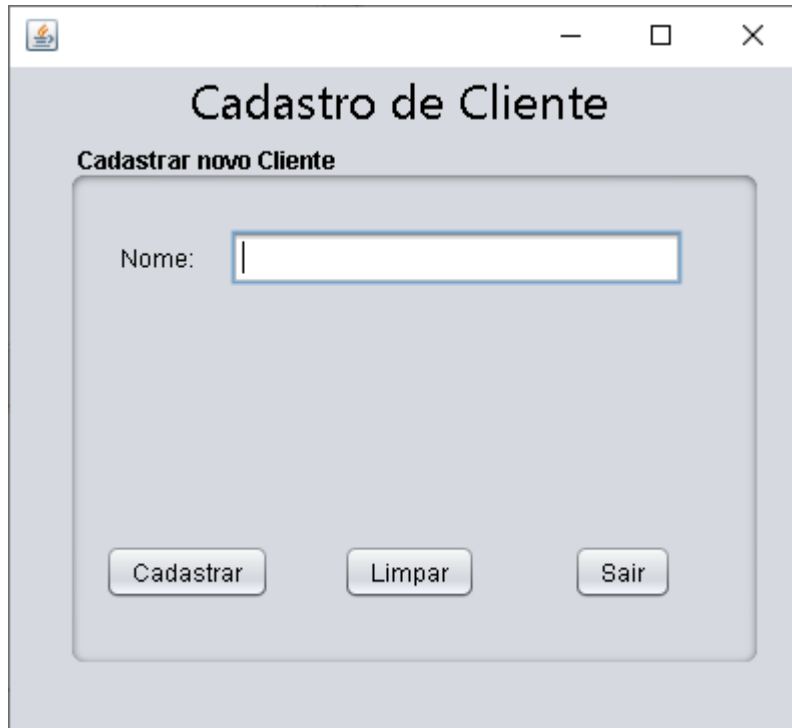
67



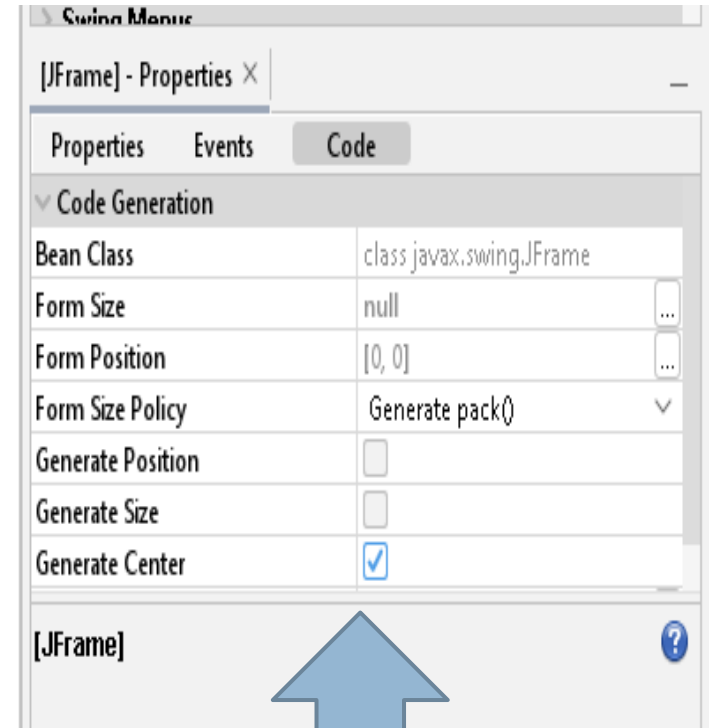
Selecione
gui.ClienteGUI e
clique no botão
Select Main Class

Executando

68



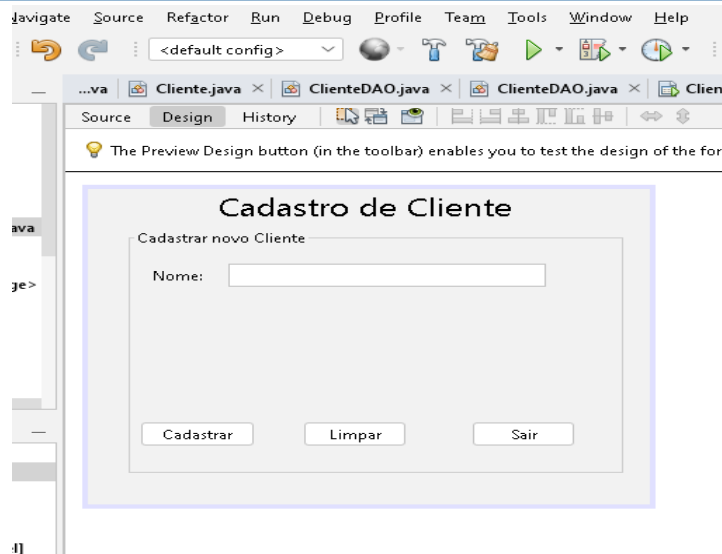
A screenshot of a Java Swing window titled "Cadastro de Cliente". Inside the window, there is a sub-panel titled "Cadastrar novo Cliente". This panel contains a text input field labeled "Nome:" and three buttons at the bottom: "Cadastrar", "Limpar", and "Sair".



Para centralizar
clique em
Generate Center

Evento Limpar

69

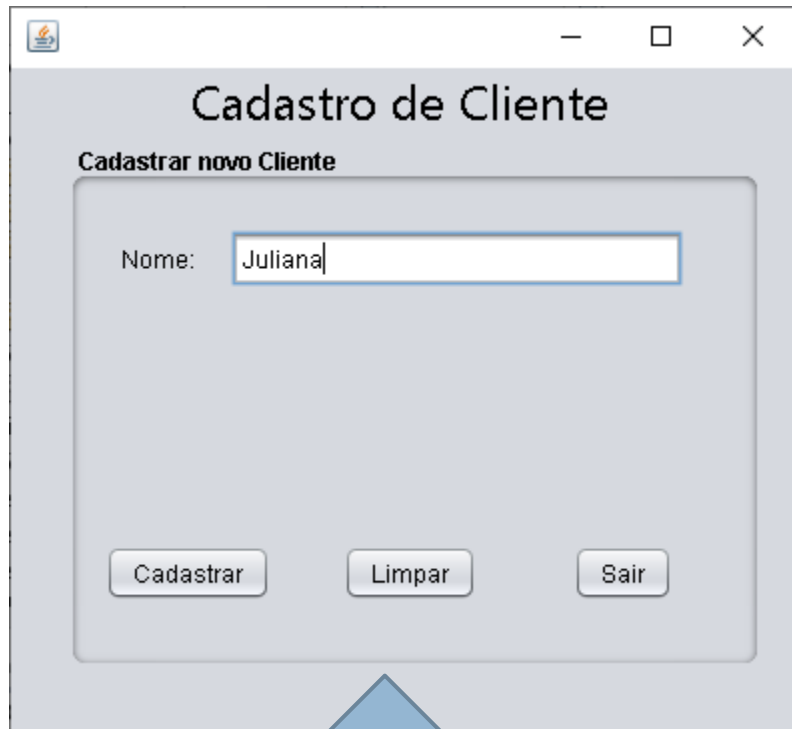


Em Design dê um duplo clique no botão Limpar

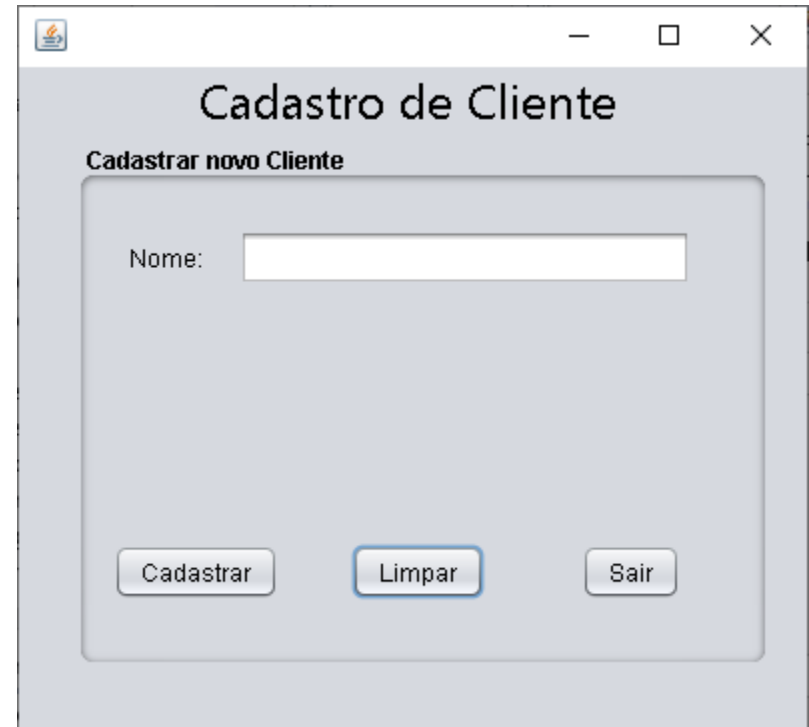
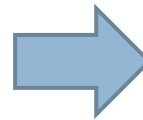
```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    jTextField1.setText("");  
  
}
```

Executando

70



The screenshot shows a window titled "Cadastro de Cliente" with a subtitle "Cadastrar novo Cliente". Inside, there is a text input field labeled "Nome:" containing the text "Juliana". Below the input field are three buttons: "Cadastrar", "Limpar", and "Sair". The "Limpar" button is highlighted with a blue border.



The screenshot shows the same "Cadastro de Cliente" window after the "Limpar" button was clicked. The "Nome:" input field is now empty. The "Limpar" button remains highlighted with a blue border.

**Digitar um Nome e
clicar em Limpar**

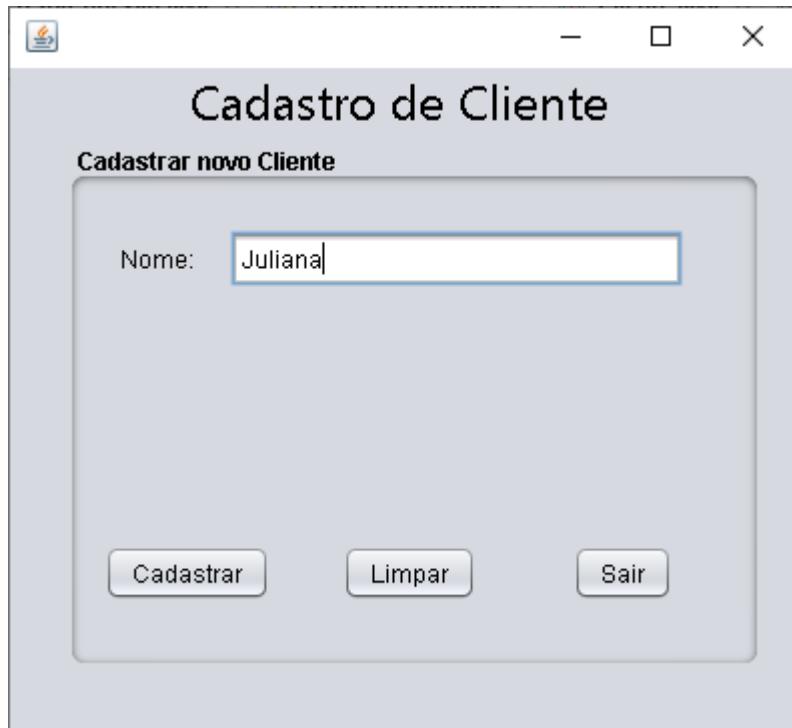
Evento Cadastrar

71

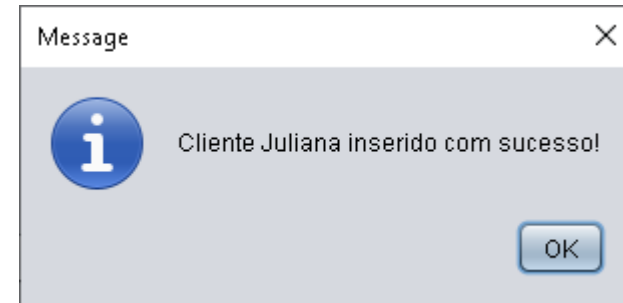
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    Cliente cliente = new Cliente();  
    cliente.setNome(jTextField1.getText());  
  
    // fazendo a validação dos dados  
    if (jTextField1.getText().isEmpty()) {  
        JOptionPane.showMessageDialog(null, "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

72



The screenshot shows a window titled "Cadastro de Cliente" with a subtitle "Cadastrar novo Cliente". Inside the window, there is a text input field labeled "Nome:" containing the text "Juliana". Below the input field, there are three buttons: "Cadastrar", "Limpar", and "Sair".



Digitar o Nome e clicar
em Cadastrar

Visualizando no Banco de Dados

73

The screenshot shows a database management interface. On the left, a tree view displays the database structure: 'vendas' (expanded) contains 'Tables' (expanded), which includes 'cliente'. Below 'Tables' are 'Views', 'Stored Procedures', and 'Functions'. At the bottom, there are tabs for 'Administration' and 'Schemas', and an 'Information' section.

On the right, a SQL query is entered in a text area:

```
5 • select * from cliente;
```

Below the query, a toolbar contains icons for 'Result Grid', 'Filter Rows' (with a search box), 'Edit', and 'Export/Import'.

The 'Result Grid' displays the following data:

	cli_id	cli_nome
▶	1	Juliana
•	NULL	NULL