



### **Título da prática:**

Criação de aplicativo Java, com acesso ao banco de dados SQL Server através do middleware JDBC.

**CAMPUS:** Polo Planalto – BH – MG

**DISCIPLINA:** BackEnd sem banco não tem!

**TURMA:** 2025.1

**SEMESTRE LETIVO:** Primeiro Semestre (2025)

**ALUNO:** Bruno Ricardo Viana Venturelli

**Matrícula:** 202401226726

### **LINK DO MEU GITHUB:**

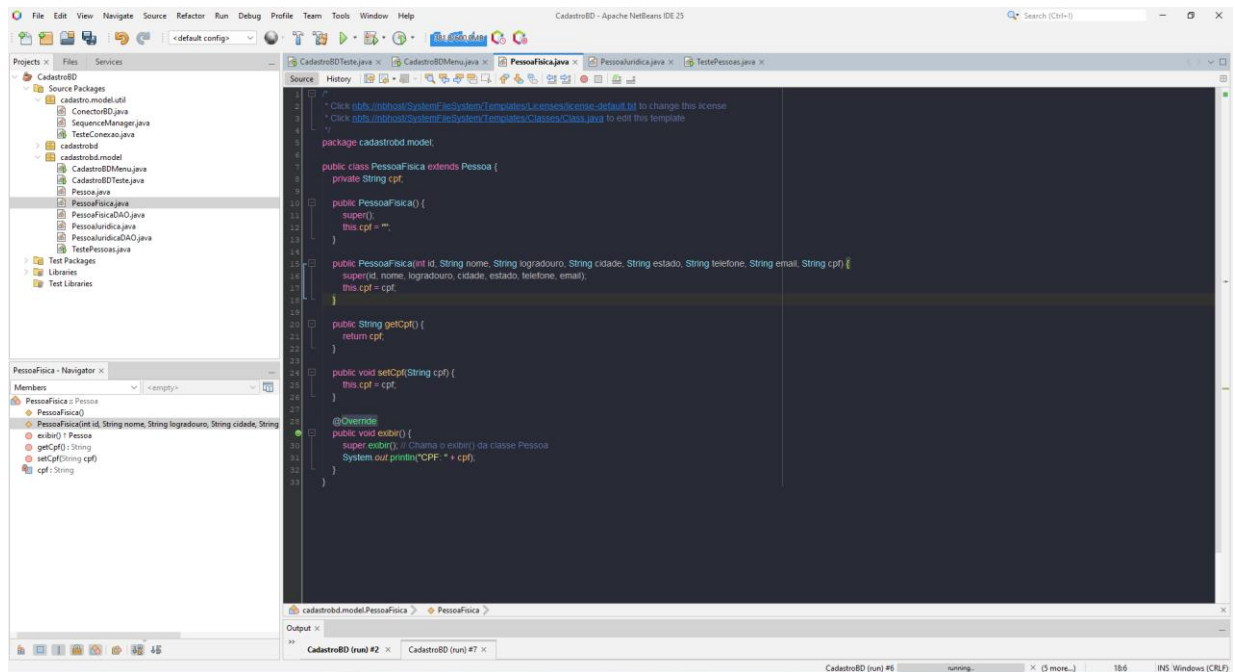
[https://github.com/DevBrN01/Trabalho\\_Pratico\\_M3\\_BackEnd\\_BancoD  
ados](https://github.com/DevBrN01/Trabalho_Pratico_M3_BackEnd_BancoDados)

### **Objetivo da Prática**

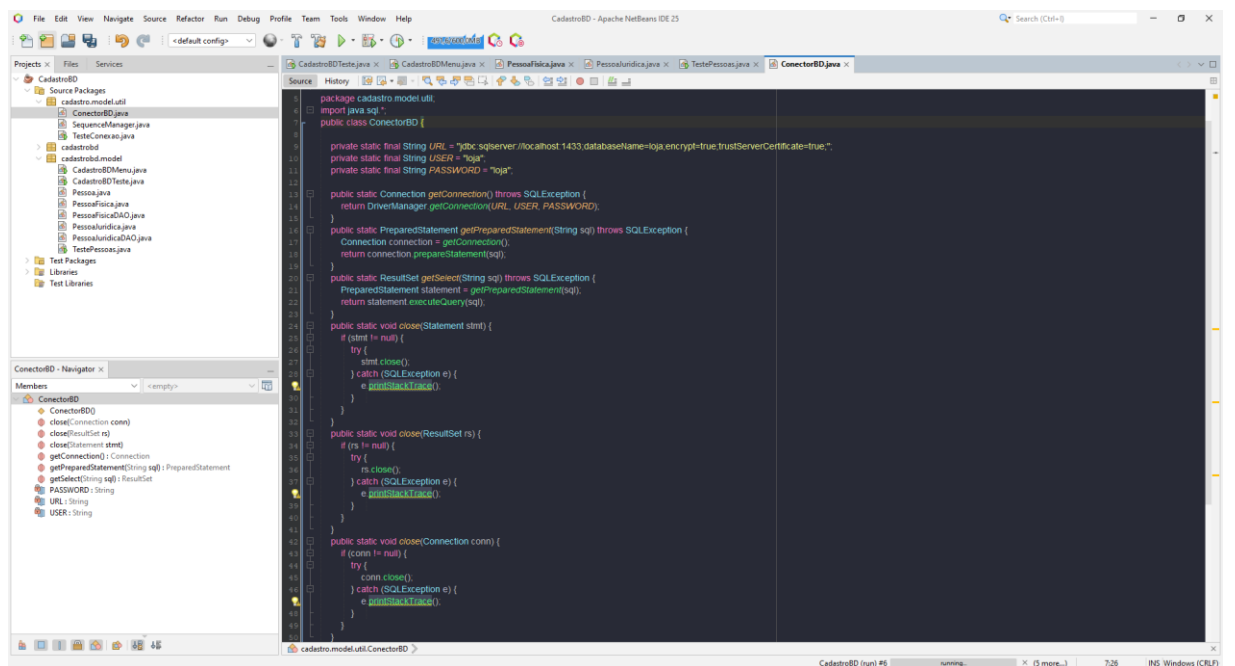
1. Implementar persistência com base no middleware JDBC.
2. Utilizar o padrão DAO (Data Access Object) no manuseio de dados.
3. Implementar o mapeamento objeto-relacional em sistemas Java.
4. Criar sistemas cadastrais com persistência em banco relacional.

# 1º Procedimento | Mapeamento Objeto-Relacional DAO

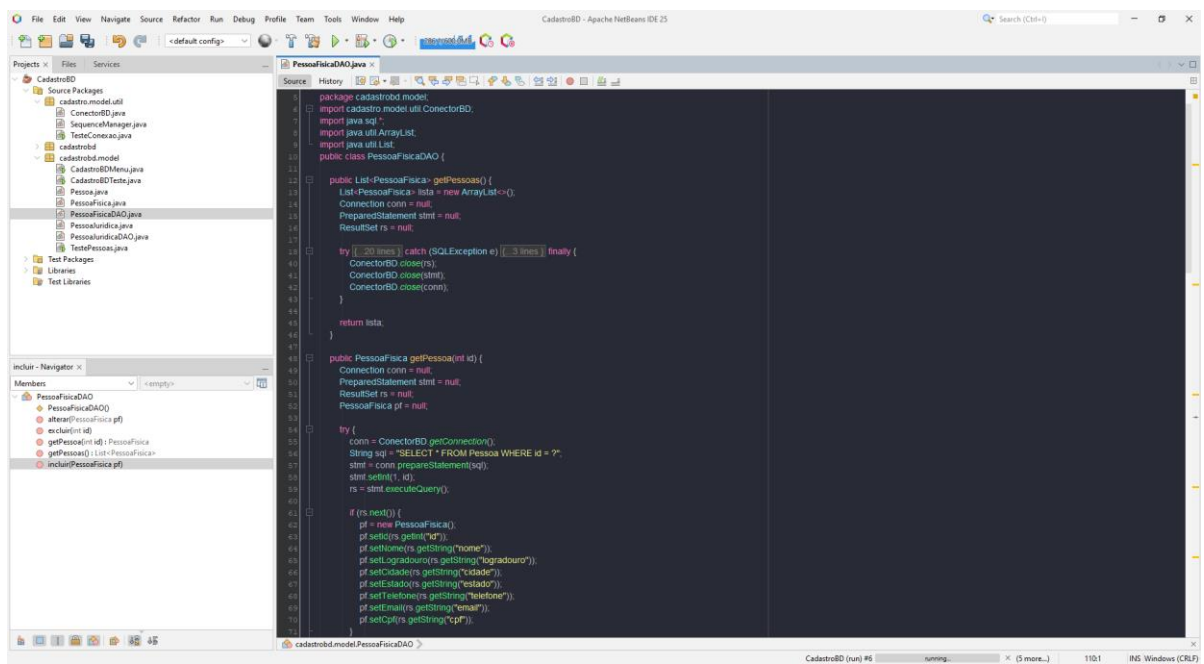
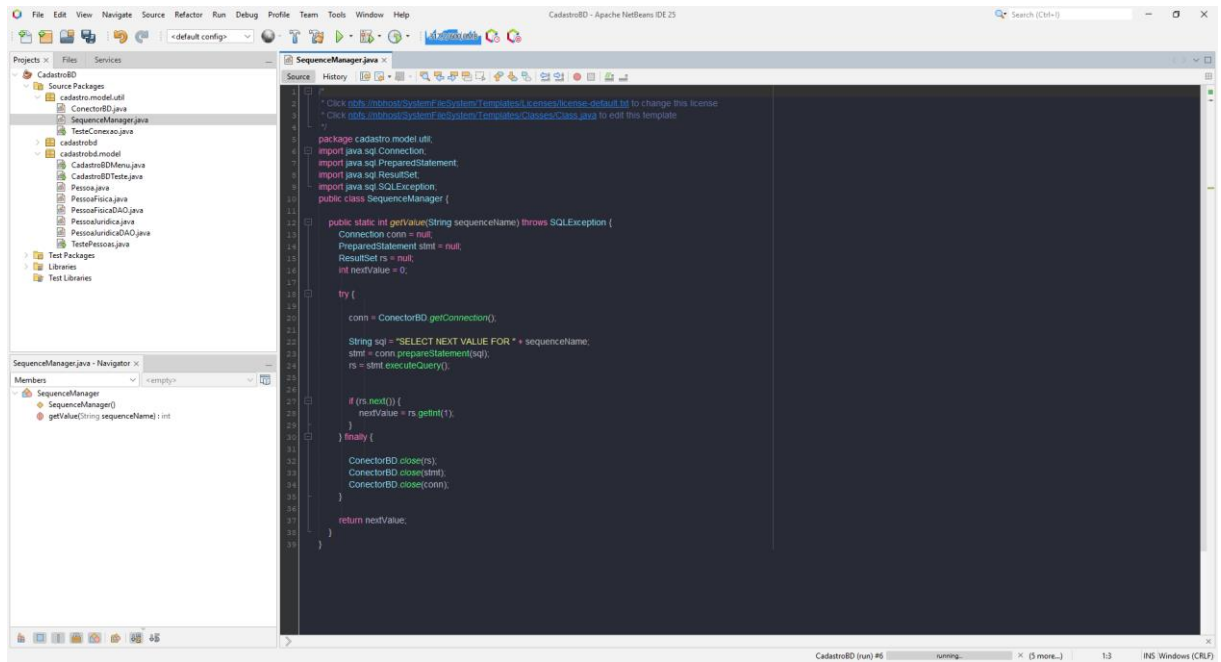
## Códigos usados neste roteiro



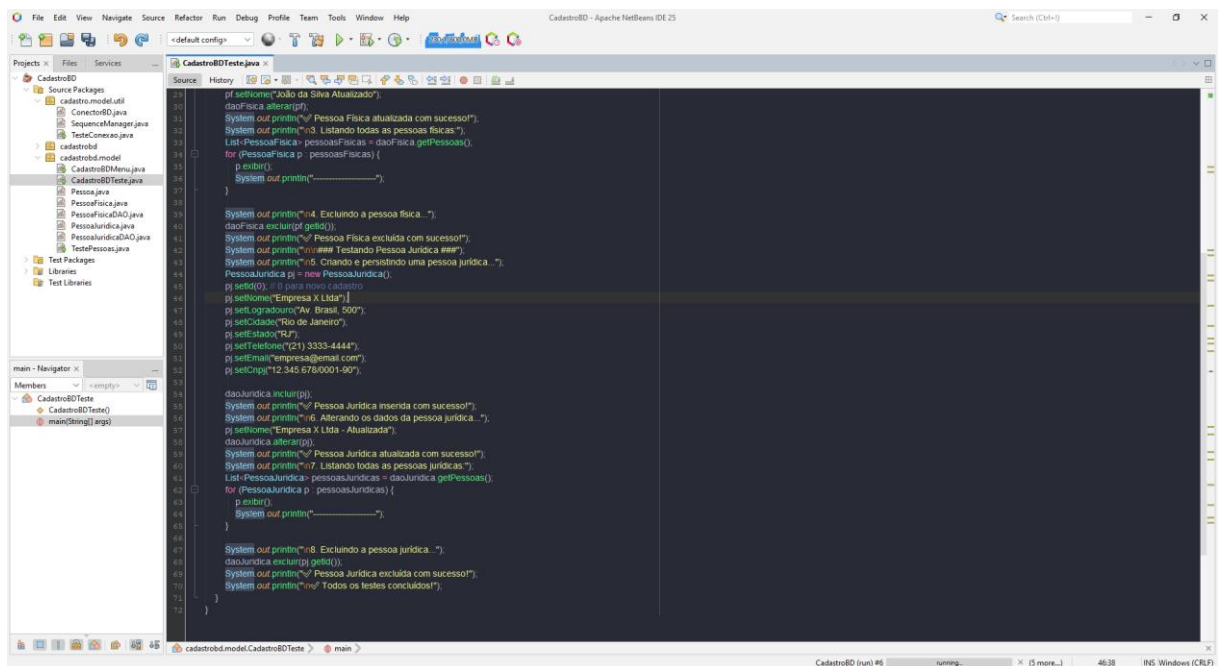
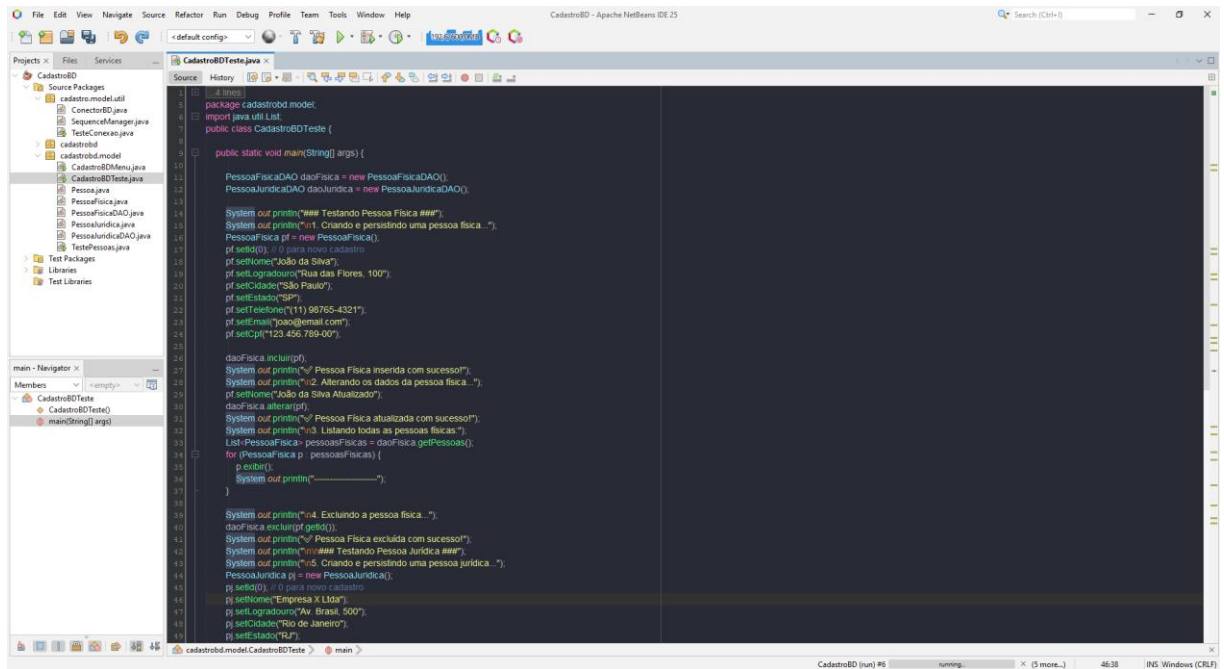
```
1 package cadastrobdt.model;
2
3 import java.util.*;
4
5 public class PessoaFisica extends Pessoa {
6     private String cpf;
7
8     public PessoaFisica() {
9         super();
10        this.cpf = "";
11    }
12
13    public PessoaFisica(int id, String nome, String logradouro, String cidade, String estado, String telefone, String email, String cpf) {
14        super(id, nome, logradouro, cidade, estado, telefone, email);
15        this.cpf = cpf;
16    }
17
18    public String getCpf() {
19        return cpf;
20    }
21
22    public void setCpf(String cpf) {
23        this.cpf = cpf;
24    }
25
26    @Override
27    public void exibir() {
28        super.exibir(); // Chama o exibir() da classe Pessoa
29        System.out.println("CPF: " + cpf);
30    }
31
32 }
```

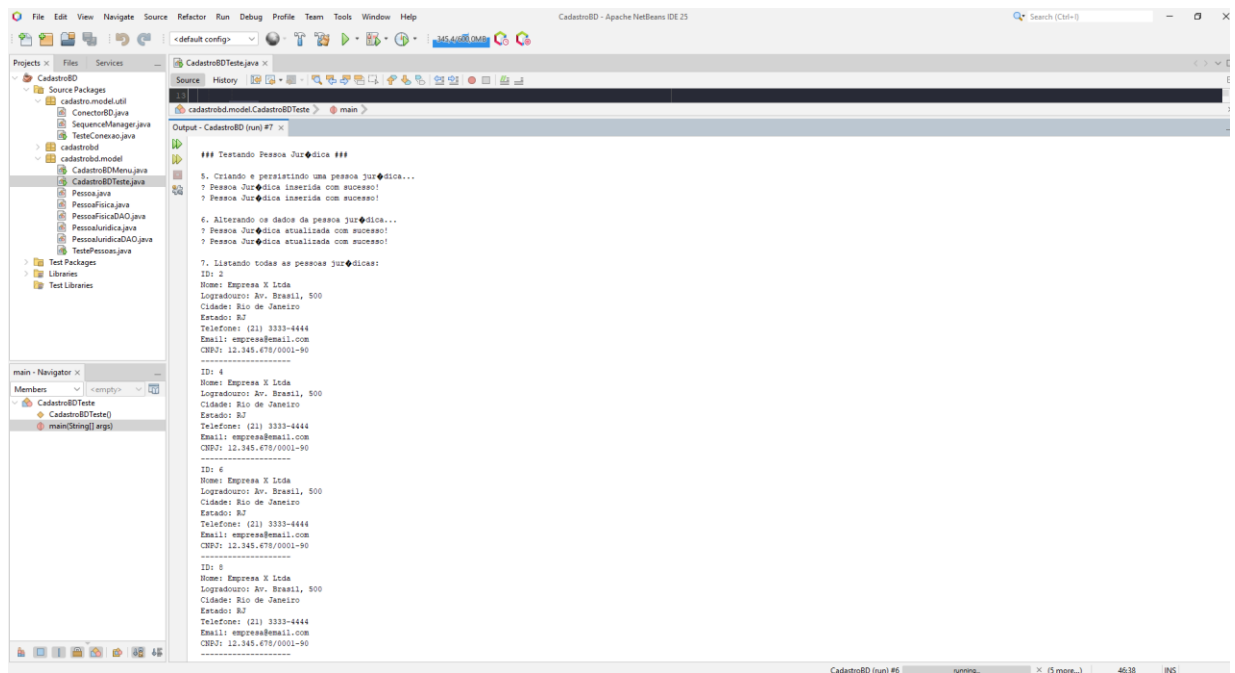
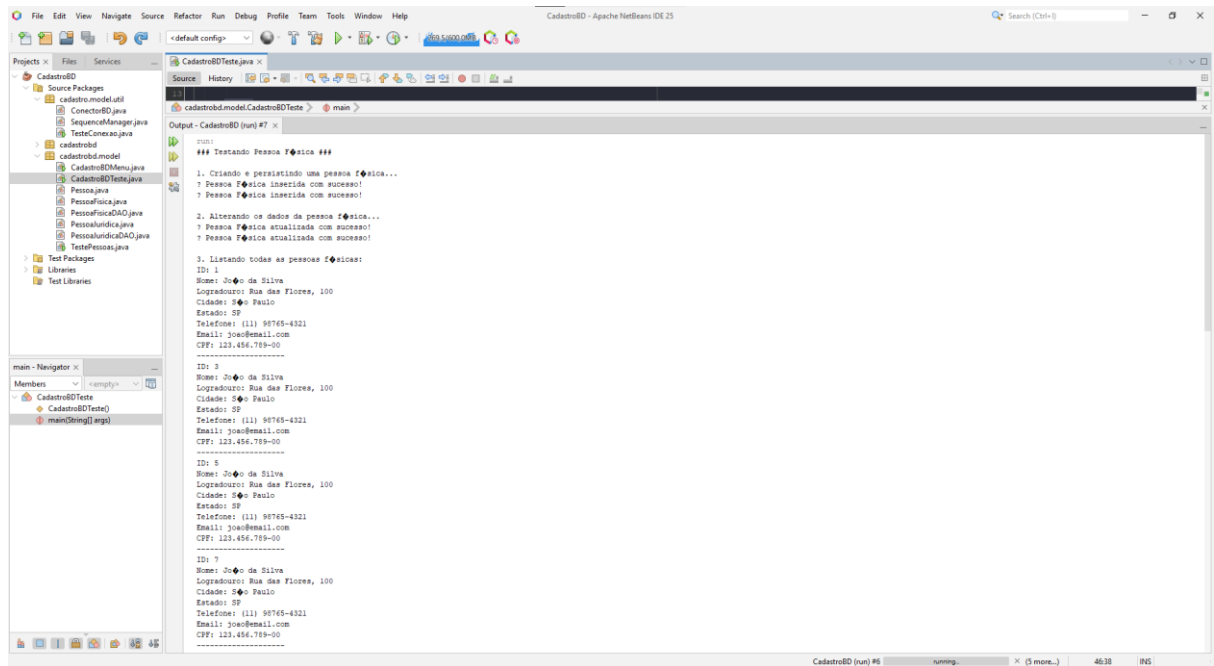


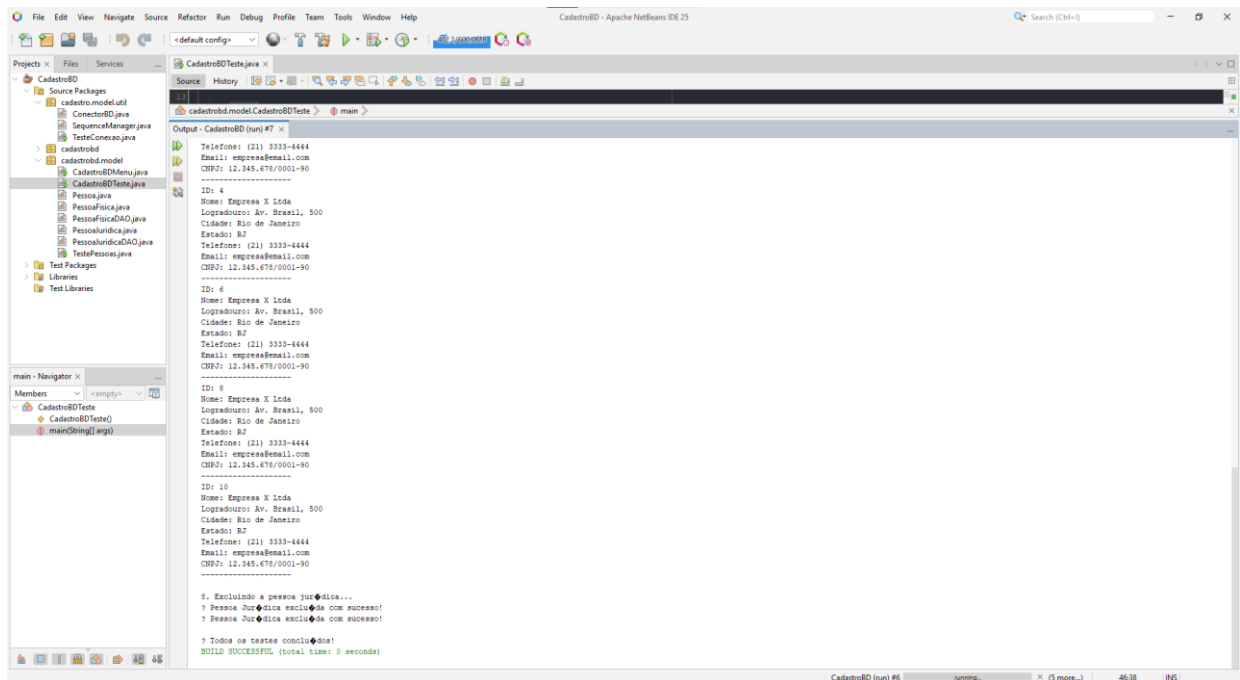
```
1 package cadastrobdt.model;
2
3 import java.sql.*;
4
5 public class ConectorBD {
6
7     private static final String URL = "jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=true";
8     private static final String USER = "loja";
9     private static final String PASSWORD = "loja";
10
11     public static Connection getConnection() throws SQLException {
12         return DriverManager.getConnection(URL, USER, PASSWORD);
13     }
14
15     public static PreparedStatement getPreparedStatement(String sql) throws SQLException {
16         Connection connection = getConnection();
17         return connection.prepareStatement(sql);
18     }
19
20     public static ResultSet getSelect(String sql) throws SQLException {
21         PreparedStatement statement = getPreparedStatement(sql);
22         return statement.executeQuery(sql);
23     }
24
25     public static void closeStatement(Statement stmt) {
26         if (stmt != null) {
27             try {
28                 stmt.close();
29             } catch (SQLException e) {
30                 e.printStackTrace();
31             }
32         }
33     }
34
35     public static void closeResultSet(ResultSet rs) {
36         if (rs != null) {
37             try {
38                 rs.close();
39             } catch (SQLException e) {
40                 e.printStackTrace();
41             }
42         }
43     }
44
45     public static void close(Connection conn) {
46         if (conn != null) {
47             try {
48                 conn.close();
49             } catch (SQLException e) {
50                 e.printStackTrace();
51             }
52         }
53     }
54 }
```





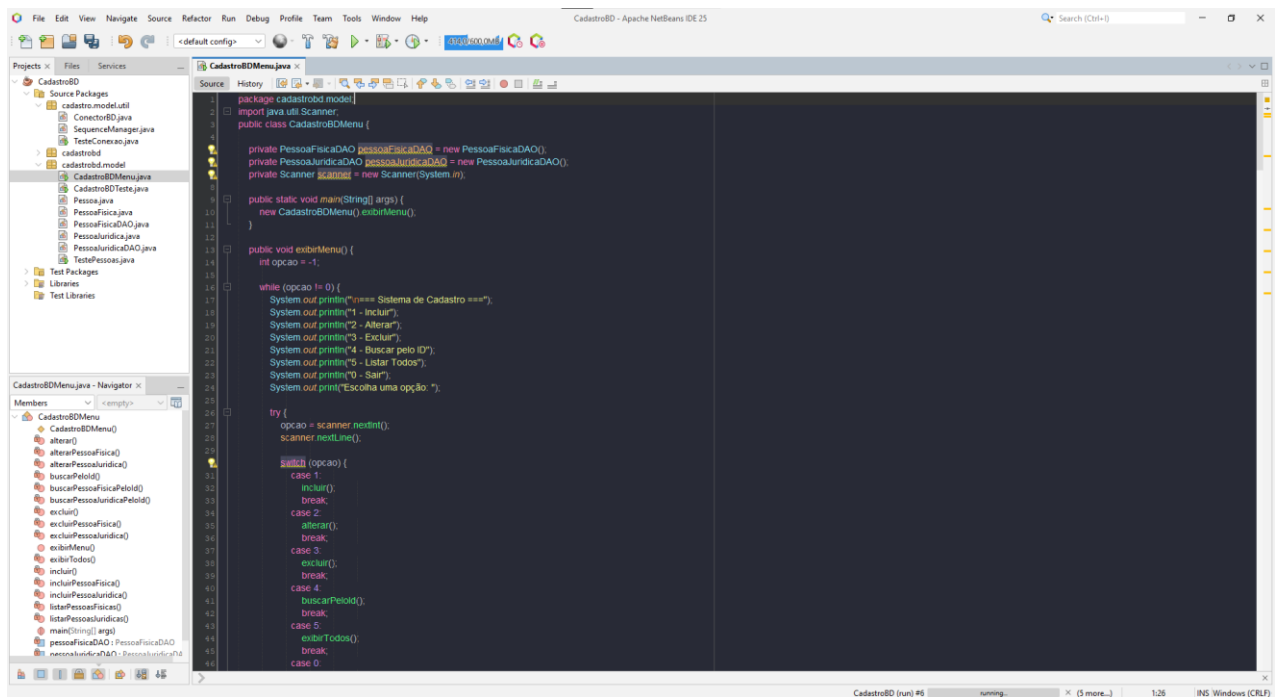


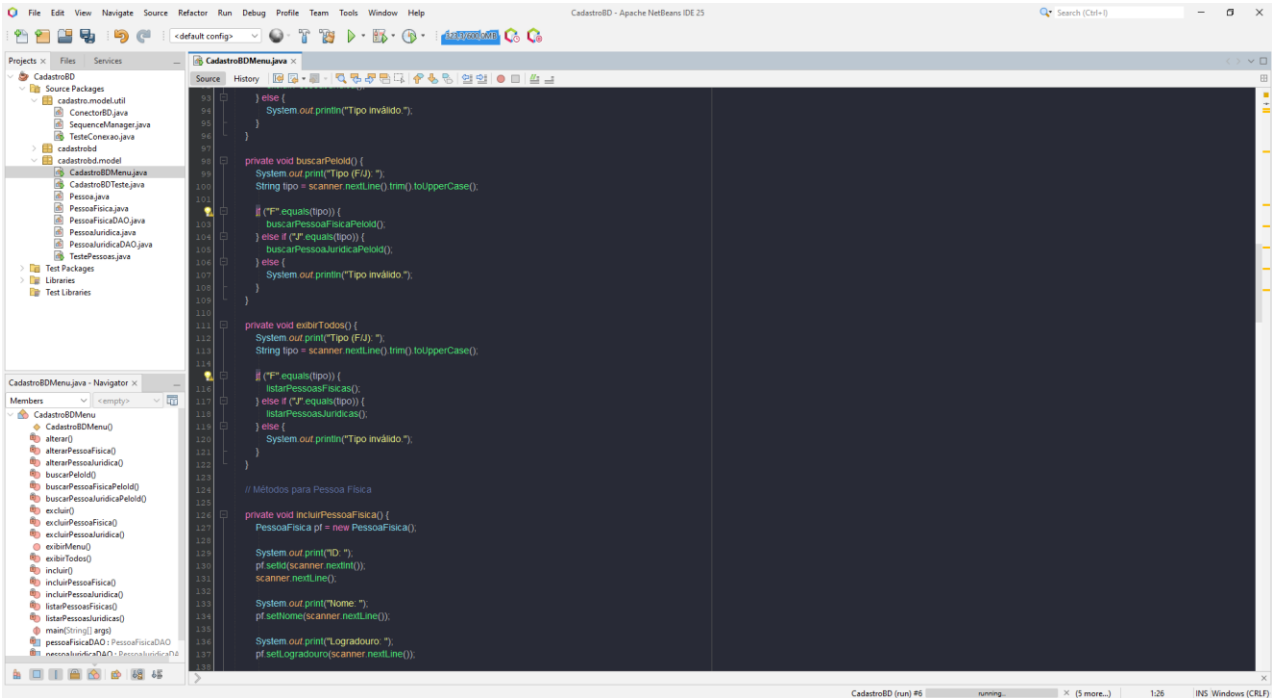




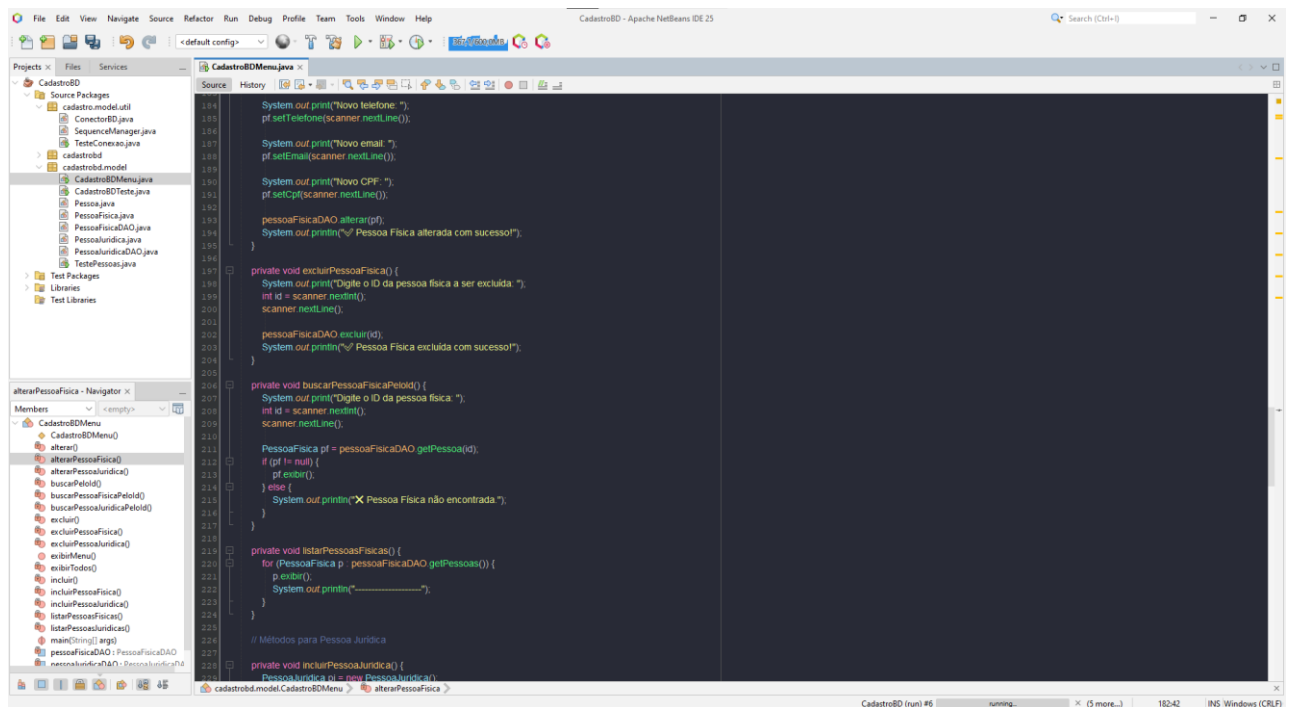
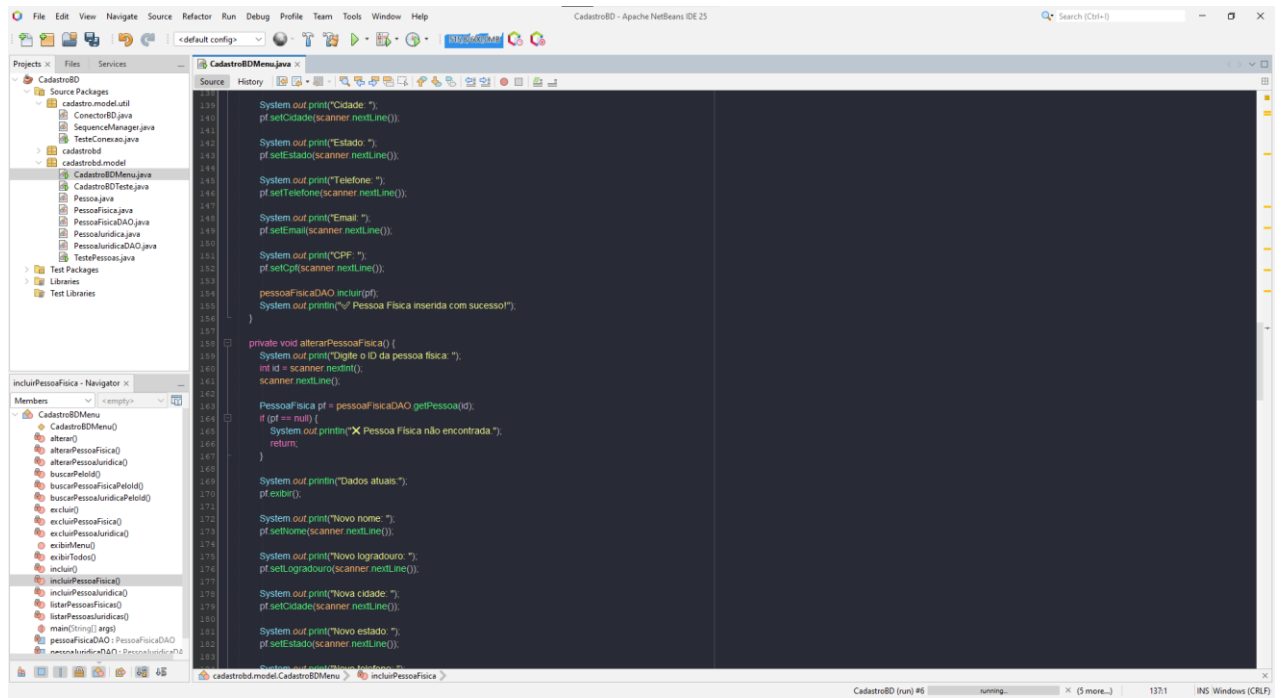
## 2º Procedimento | Alimentando a Base

### Códigos usados neste roteiro

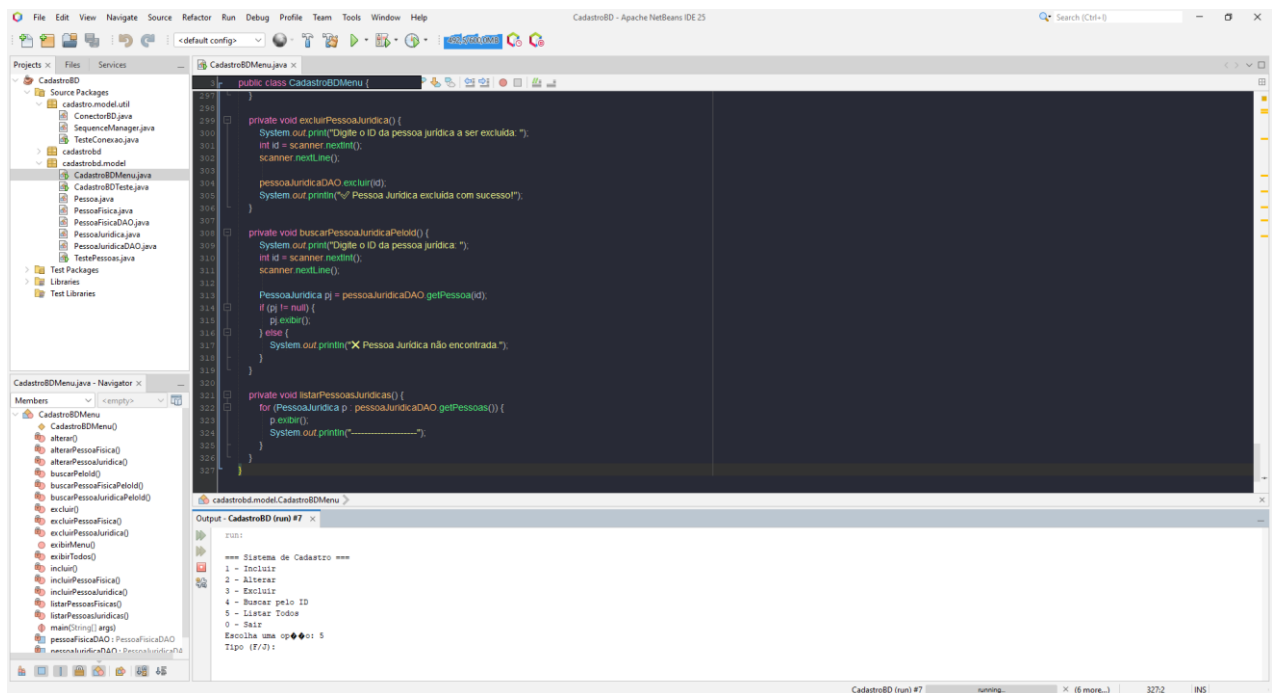
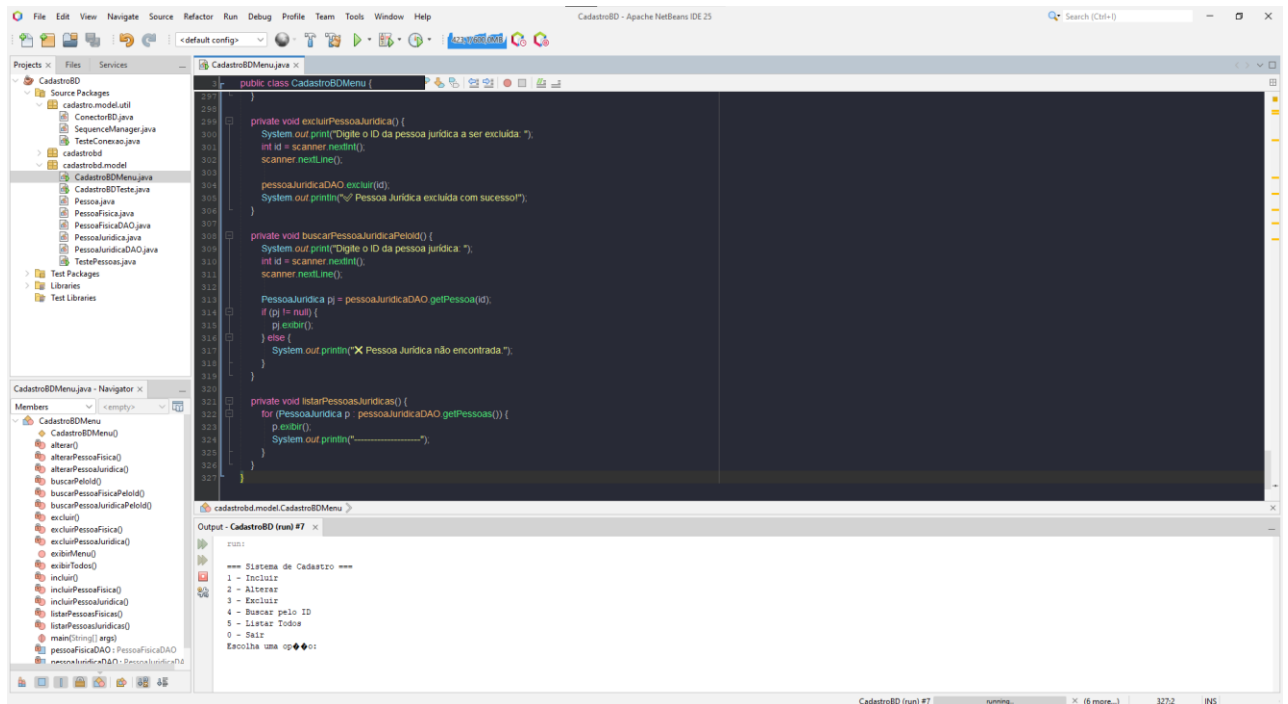


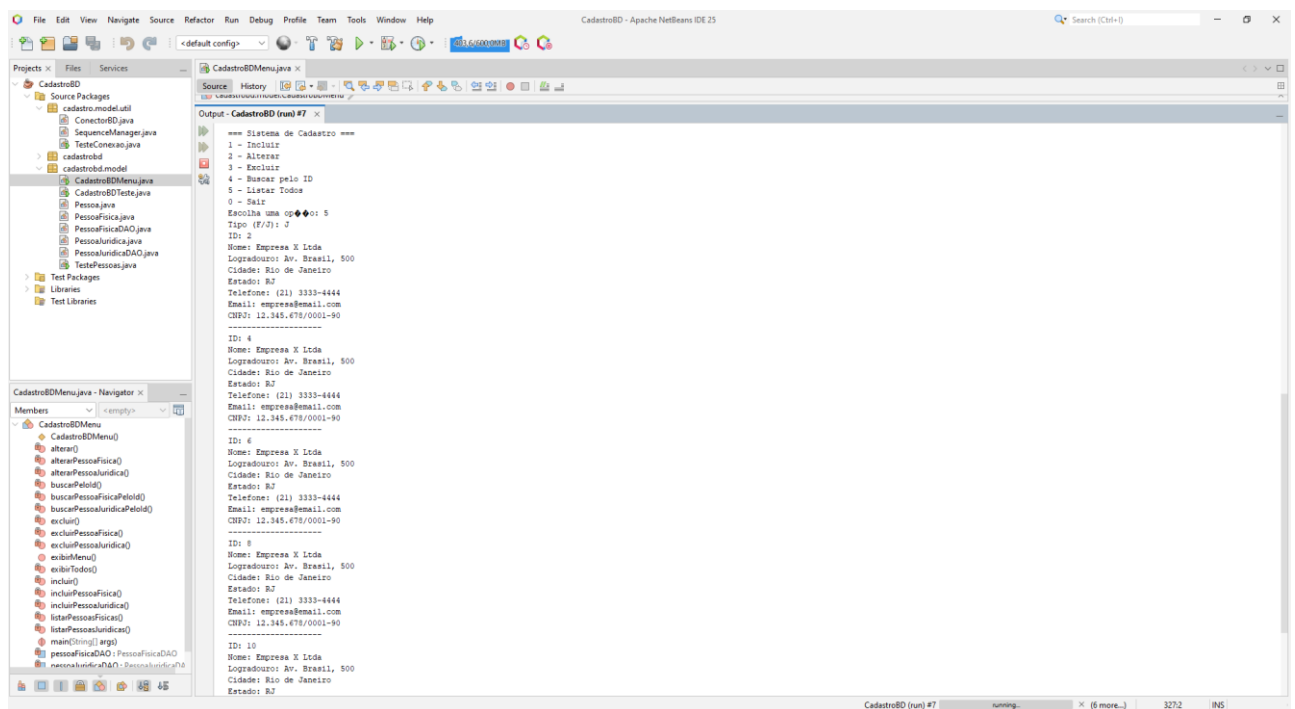
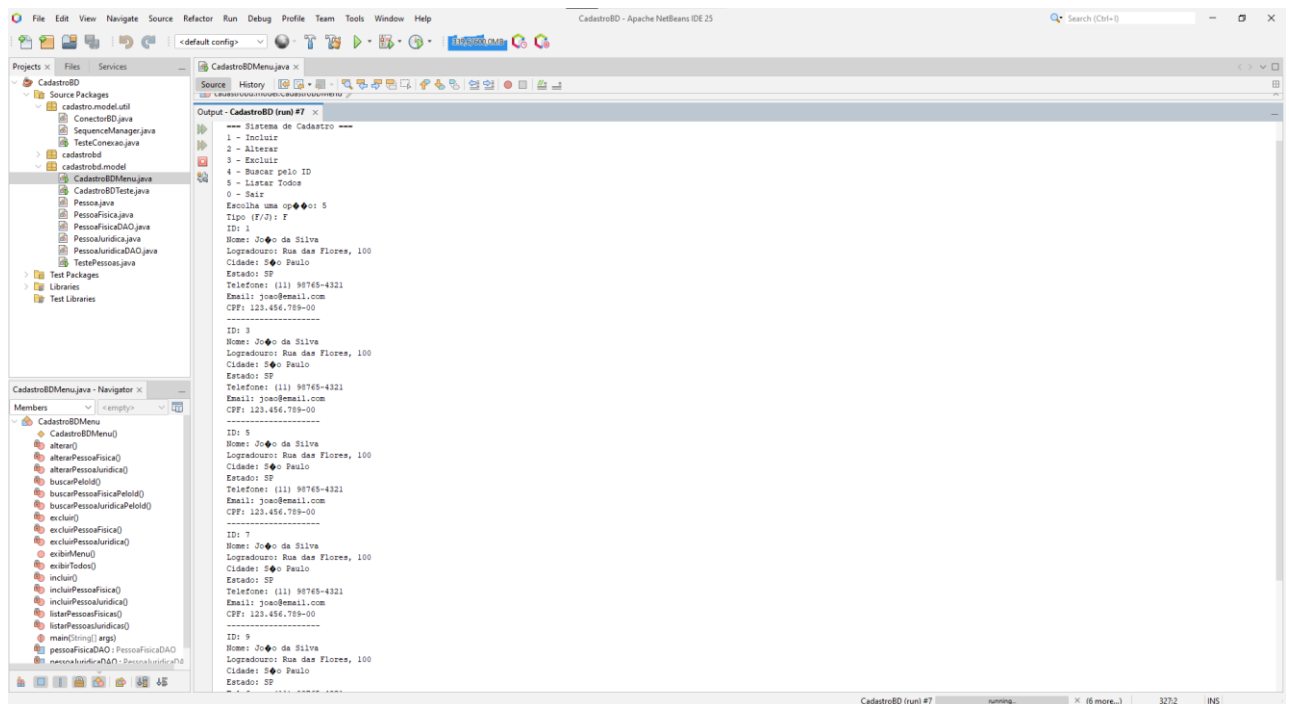












## Conclusão do exercício, 1º Procedimento:

Qual a importância dos componentes de middleware, como o JDBC?

Os componentes de middleware, como o JDBC (Java Database

Connectivity), têm papel fundamental na comunicação entre a aplicação e o banco de dados. Eles atuam como uma ponte de integração, abstraindo detalhes técnicos e facilitando o desenvolvimento.

### **Qual a diferença no uso de Statement ou PreparedStatement para a manipulação de dados?**

A principal diferença entre Statement e PreparedStatement no JDBC está em segurança, desempenho e facilidade de uso.

### **Como o padrão DAO melhora a manutenibilidade do software?**

O padrão DAO (Data Access Object) melhora significativamente a manutenibilidade do software ao separar a lógica de acesso a dados da lógica de negócios.

### **Análise e conclusão:**

#### **Como a herança é refletida no banco de dados, quando lidamos com um modelo estritamente relacional?**

Quando lidamos com herança em um modelo orientado a objetos (como em Java), mas usamos um banco de dados relacional (que não tem herança nativamente), precisamos simular esse comportamento com estratégias específicas de modelagem.

### **Conclusão do exercício, 2º Procedimento:**

#### **Qual a importância dos componentes de middleware, como o JDBC?**

Os componentes de middleware, como o JDBC (Java Database Connectivity), têm papel fundamental na comunicação entre a aplicação e o banco de dados. Eles atuam como uma ponte de integração, abstraindo detalhes técnicos e facilitando o desenvolvimento.

**Como o uso de operador lambda simplificou a impressão dos valores contidos nas entidades, nas versões mais recentes do Java?**

O uso de operadores lambda no Java (introduzidos a partir do Java 8) trouxe uma forma muito mais concisa, expressiva e moderna de trabalhar com coleções, inclusive para imprimir valores de entidades..

**Análise e conclusão:**

**Por que métodos acionados diretamente pelo método main, sem o uso de um objeto, precisam ser marcados como static?**

Métodos chamados diretamente a partir do método main precisam ser static porque o main também é um método estático e métodos estáticos só podem acessar diretamente outros métodos ou atributos que também sejam estáticos.