



Estácio

CAMPUS POLO AUSTIN - NOVA IGUAÇU – RJ

DESENVOLVIMENTO FULL STACK

Nível 3: Back-end Sem Banco Não Tem

RPG0016 9001

2024/2

SALOMÃO ISAAC CARVALHO GARCIA

**Desenvolvimento de Aplicativo Java com Acesso a Banco de Dados SQL
Server Utilizando JDBC e Padrão DAO**

Nova Iguaçu

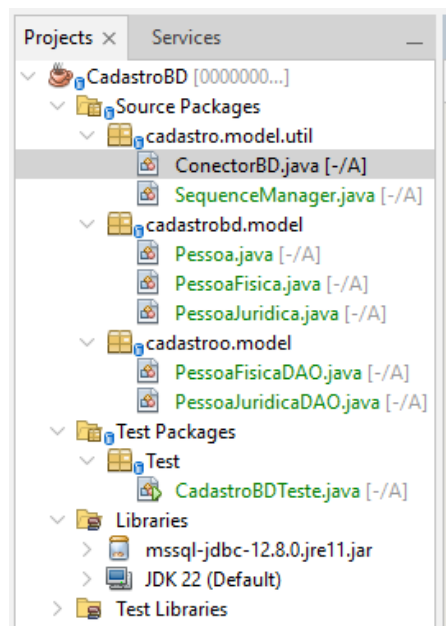
15/08/2024

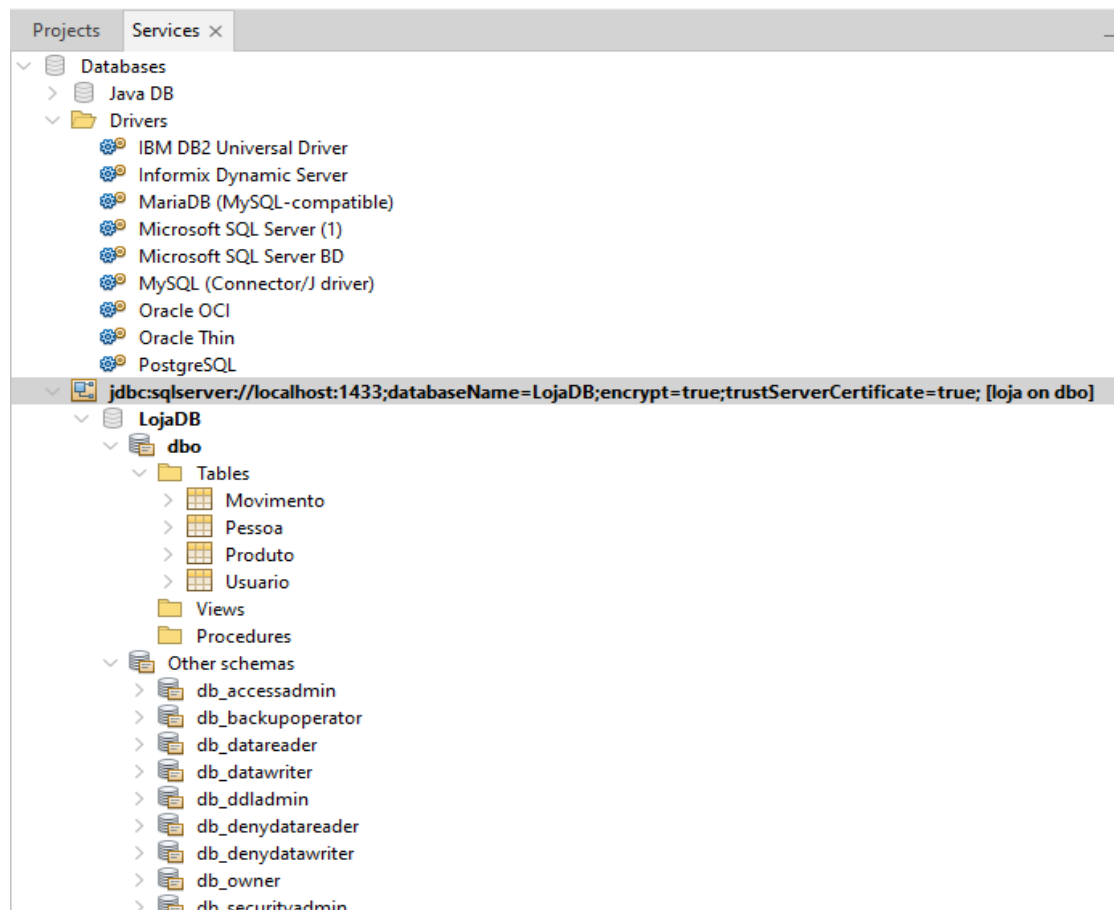
Desenvolvimento de Aplicativo Java com Acesso a Banco de Dados SQL Server Utilizando JDBC e Padrão DAO

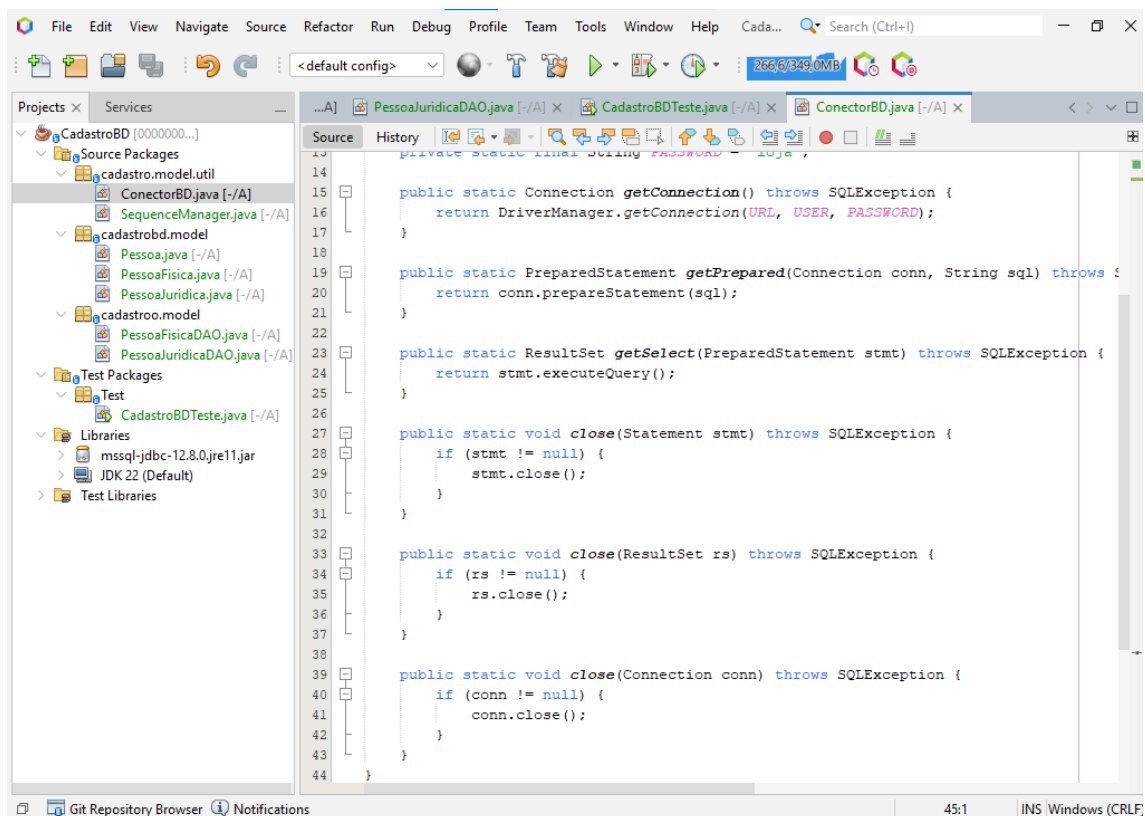
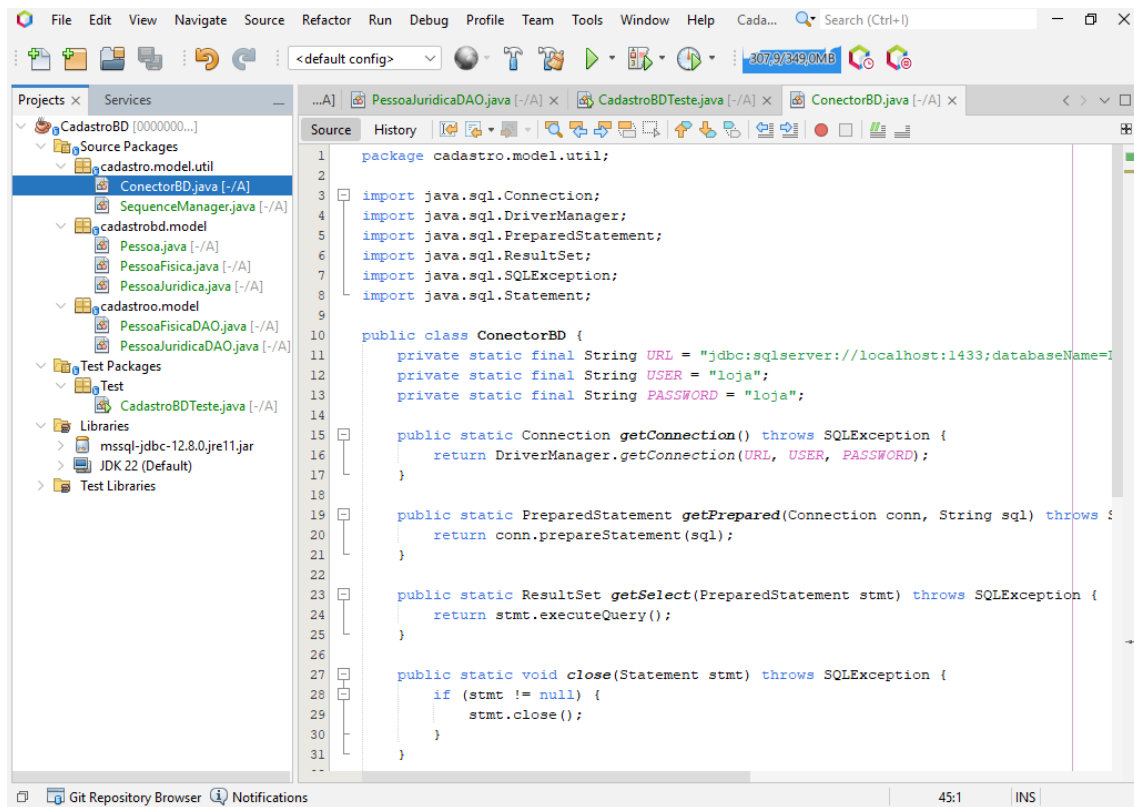
Objetivo da Prática: Este relatório descreve a prática de desenvolvimento de um aplicativo Java com acesso a um banco de dados SQL Server através do JDBC (Java Database Connectivity). A prática envolve a implementação do padrão DAO (Data Access Object) para gerenciar a persistência de dados e a manipulação de informações armazenadas no banco de dados. O objetivo principal é demonstrar a capacidade de integrar um aplicativo Java com um banco de dados relacional e aplicar boas práticas de programação e design de software. O projeto inclui a criação e a execução de testes para garantir a funcionalidade correta das operações de CRUD (Create, Read, Update, Delete), bem como a análise do impacto do padrão DAO na manutenibilidade do código e na organização da aplicação.

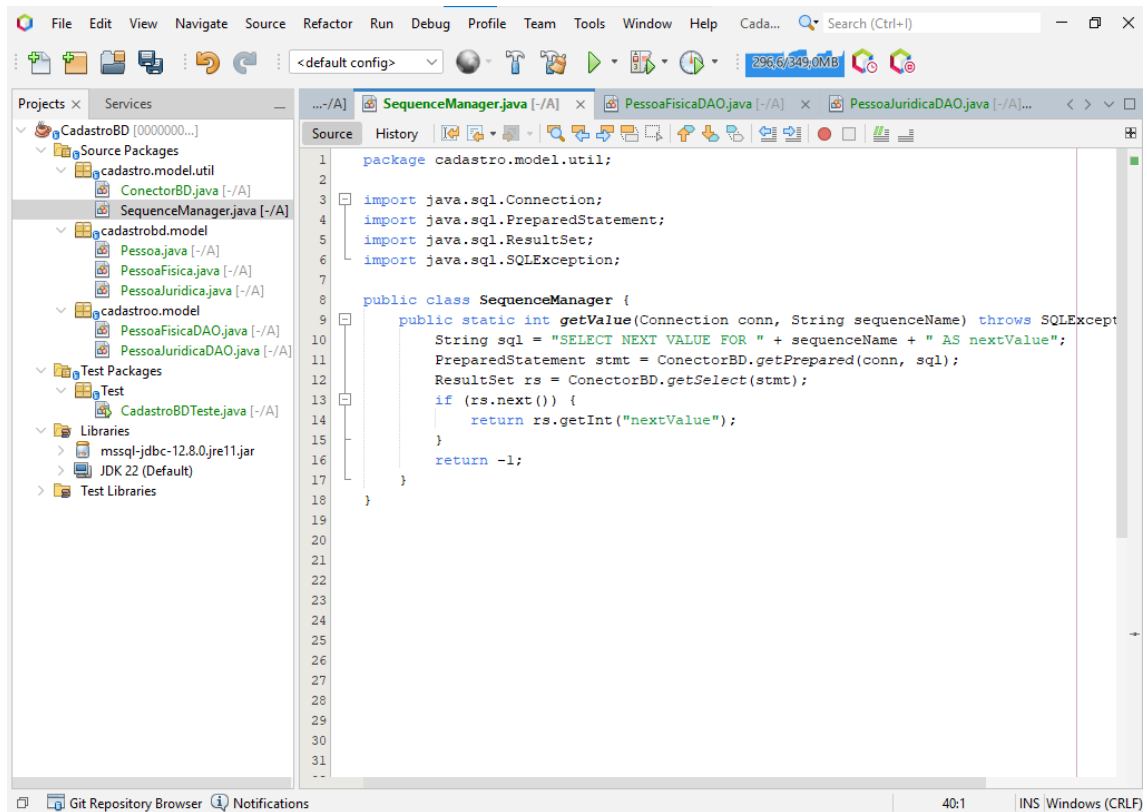
<https://github.com/SaloGarcia/nivel03mundo03.git>

CÓDIGOS

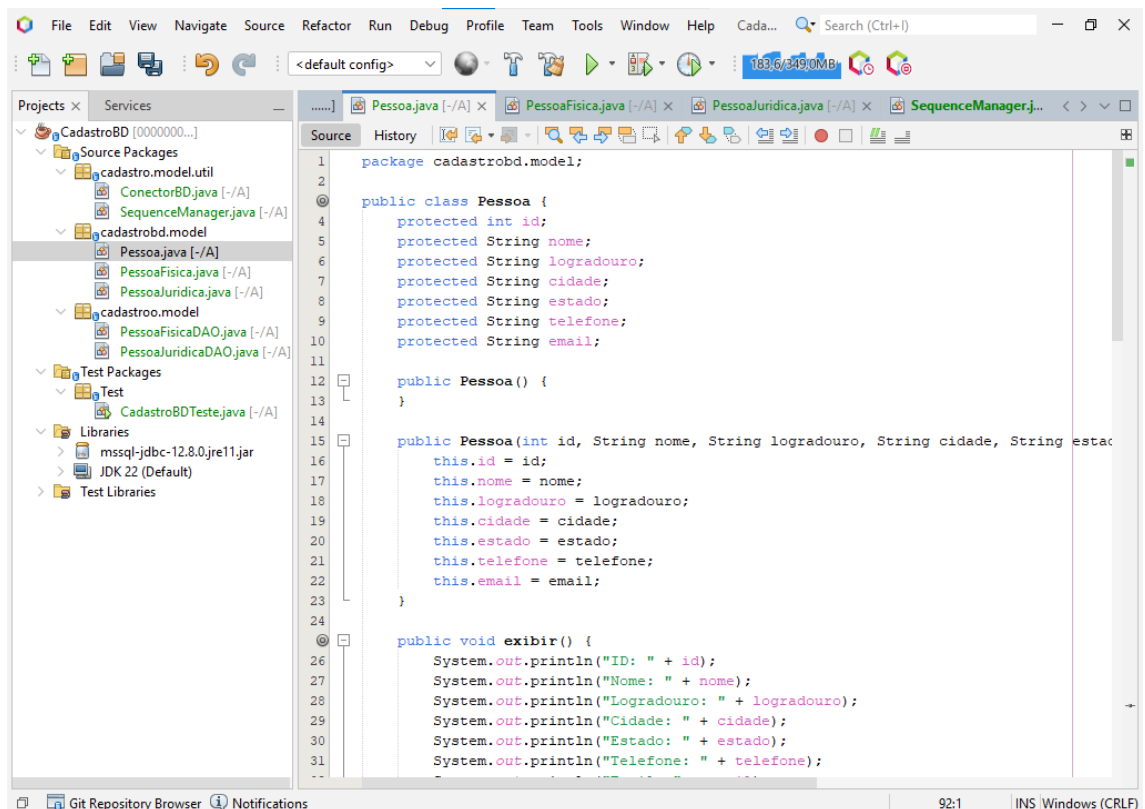




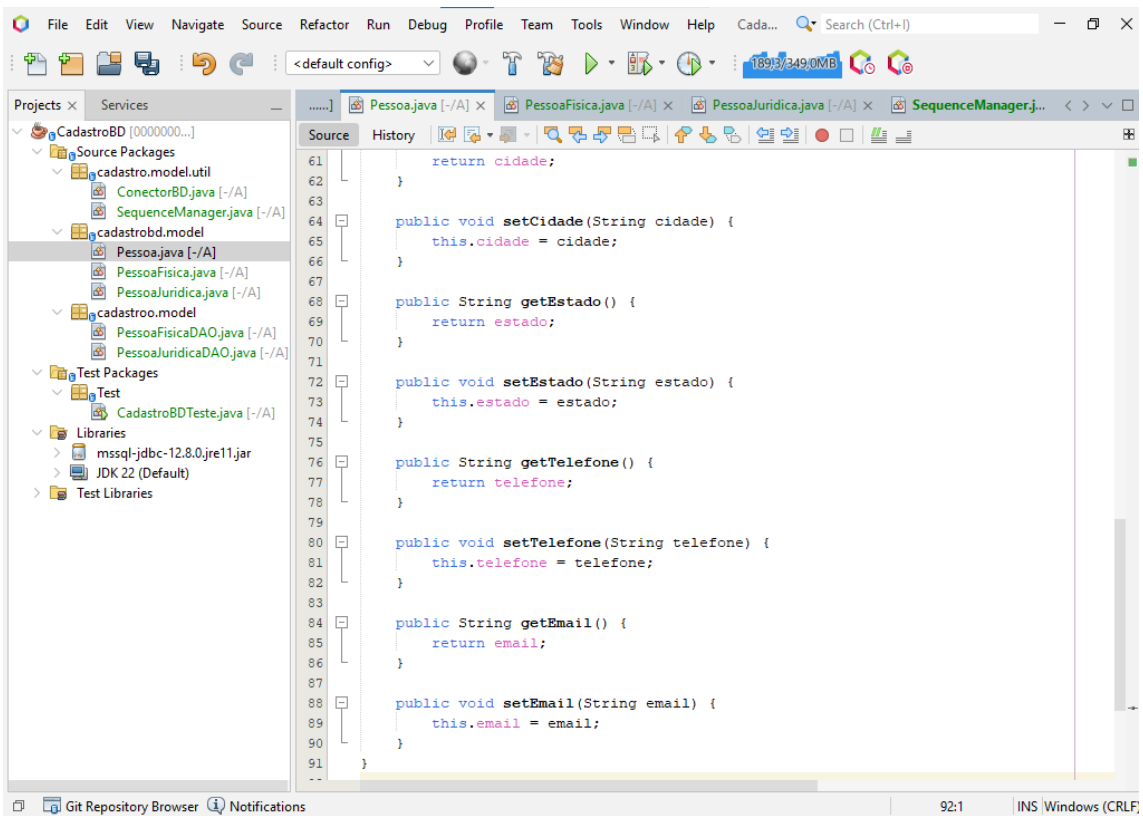
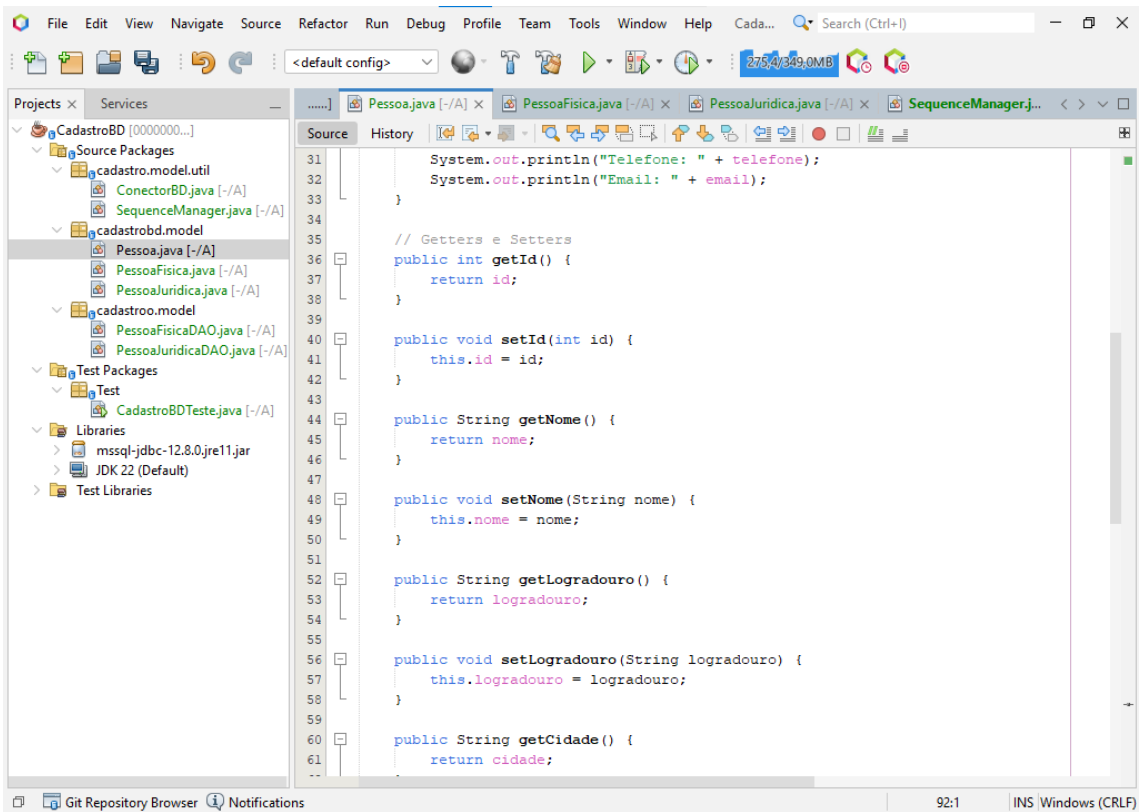


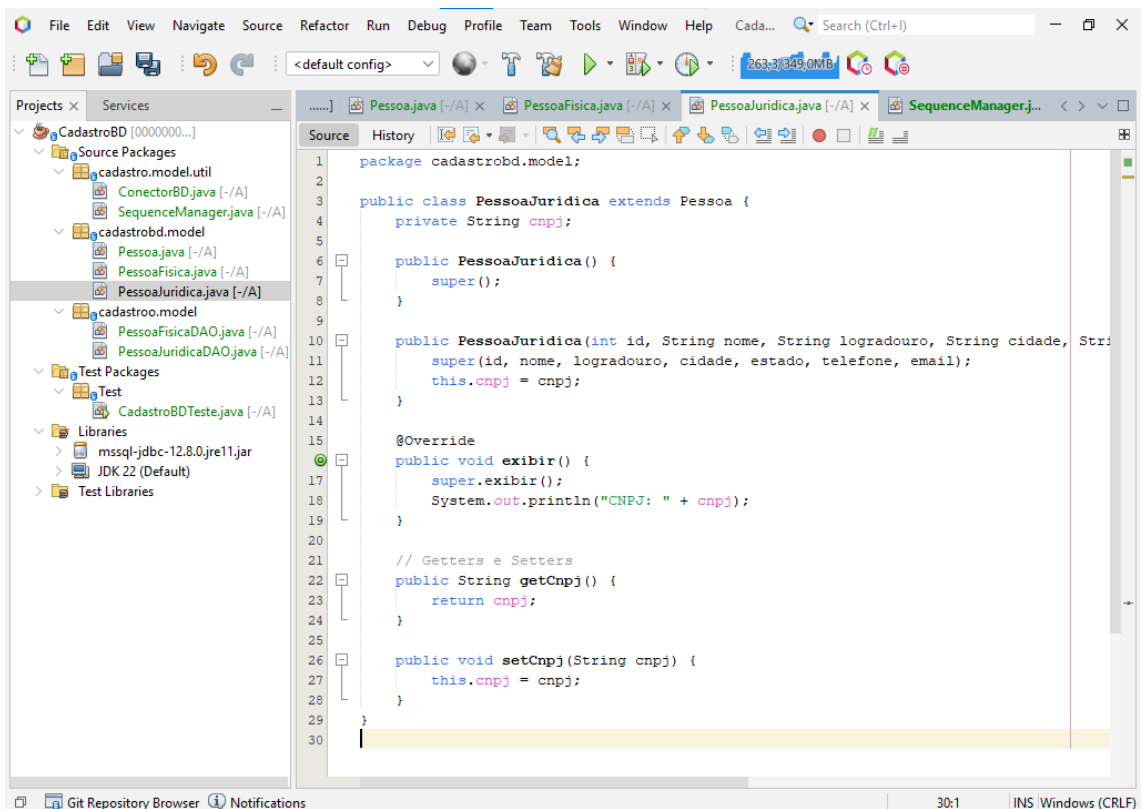
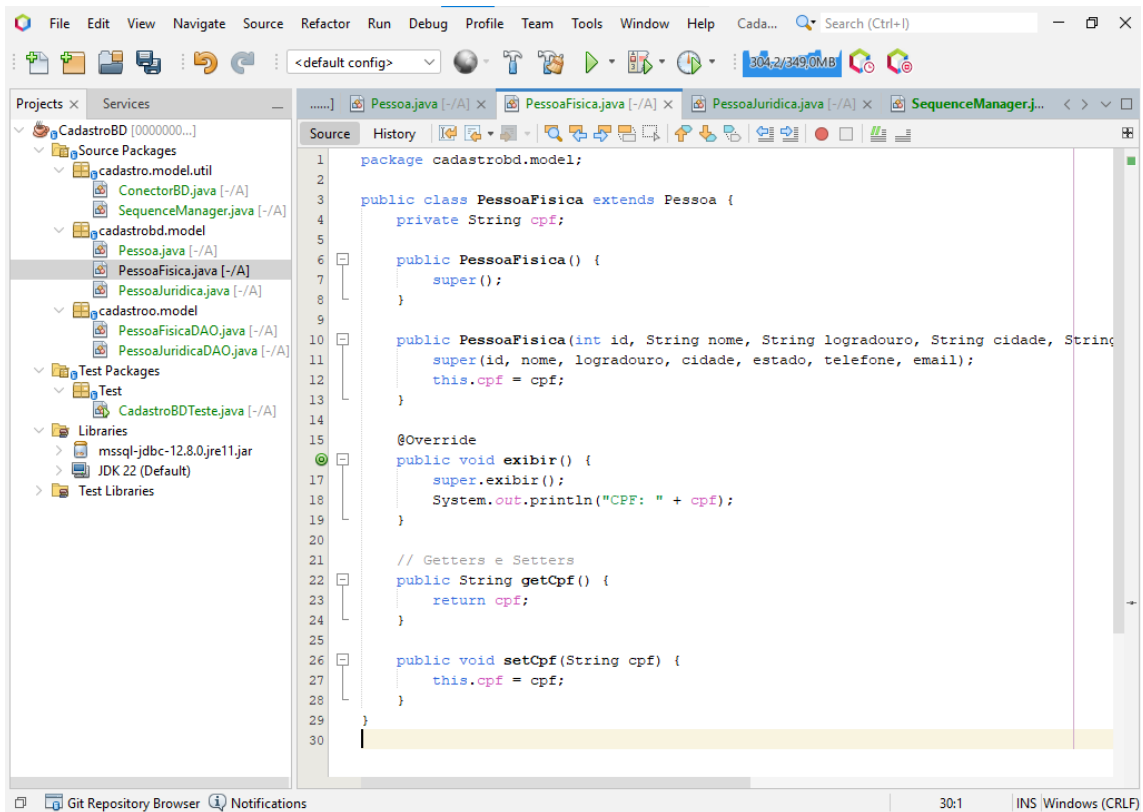


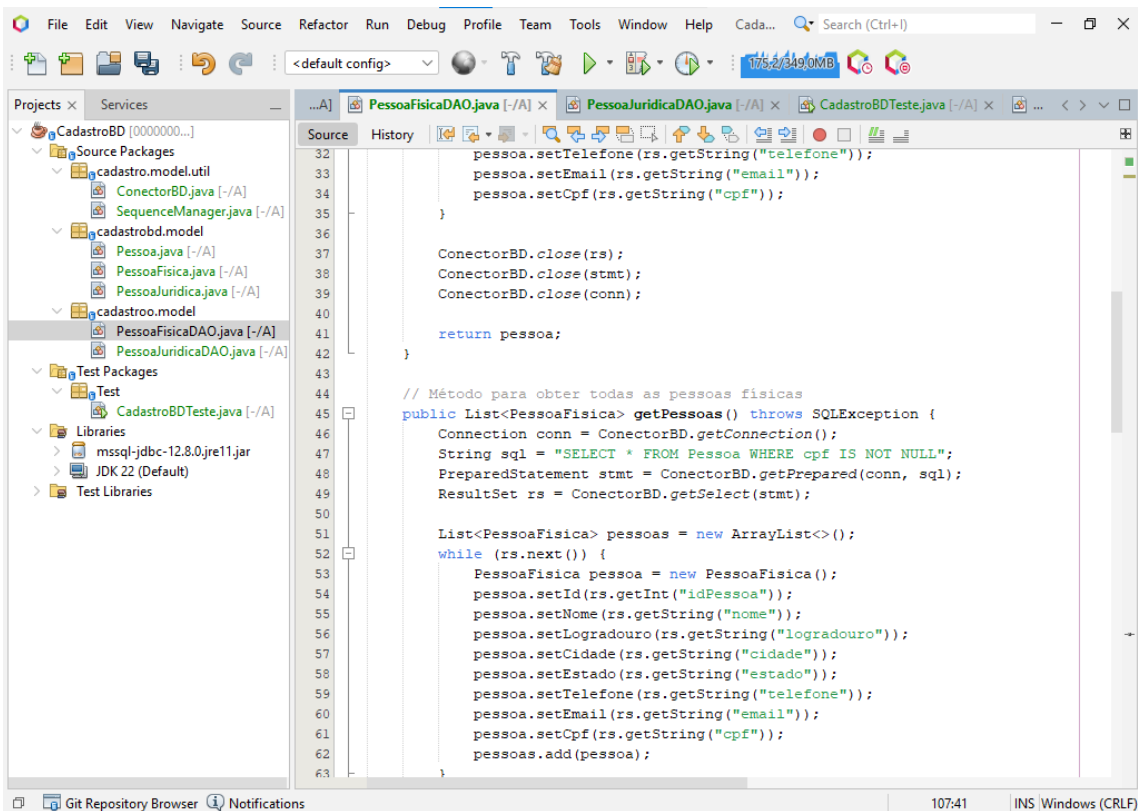
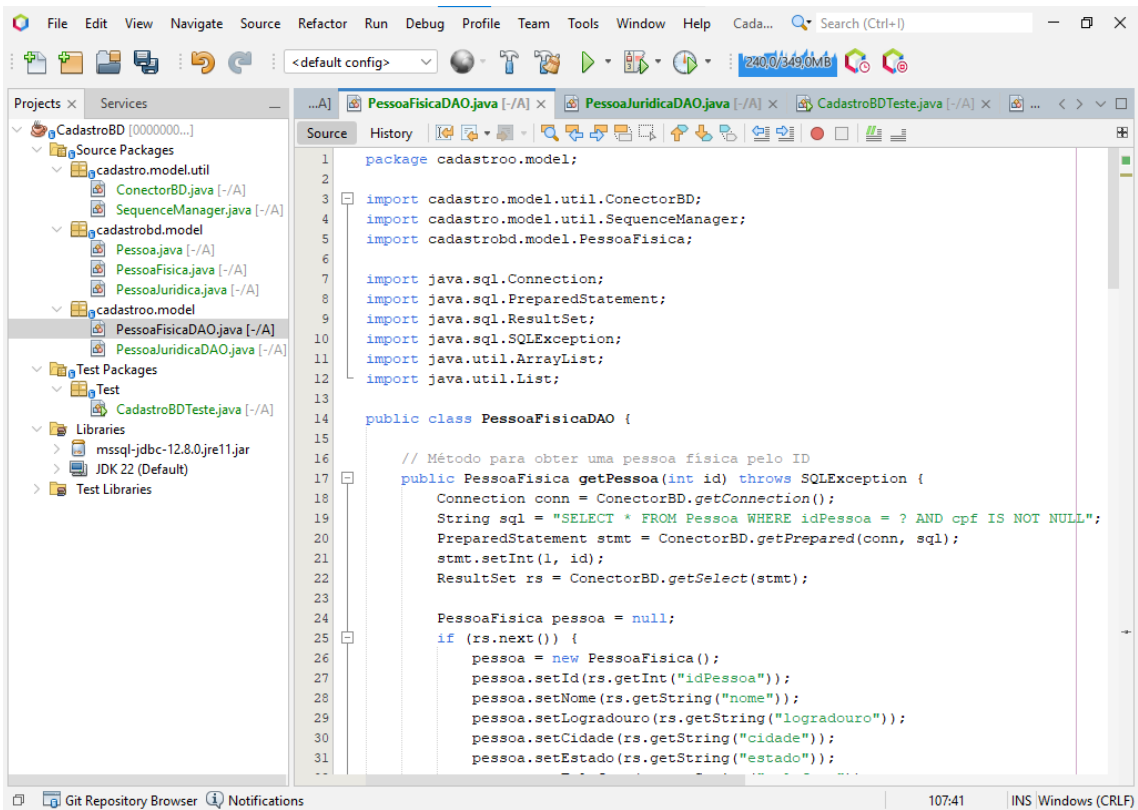
```
1 package cadastro.model.util;
2
3 import java.sql.Connection;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7
8 public class SequenceManager {
9     public static int getValue(Connection conn, String sequenceName) throws SQLException {
10         String sql = "SELECT NEXT VALUE FOR " + sequenceName + " AS nextValue";
11         PreparedStatement stmt = ConectorBD.getPrepared(conn, sql);
12         ResultSet rs = ConectorBD.getSelect(stmt);
13         if (rs.next()) {
14             return rs.getInt("nextValue");
15         }
16         return -1;
17     }
18 }
```

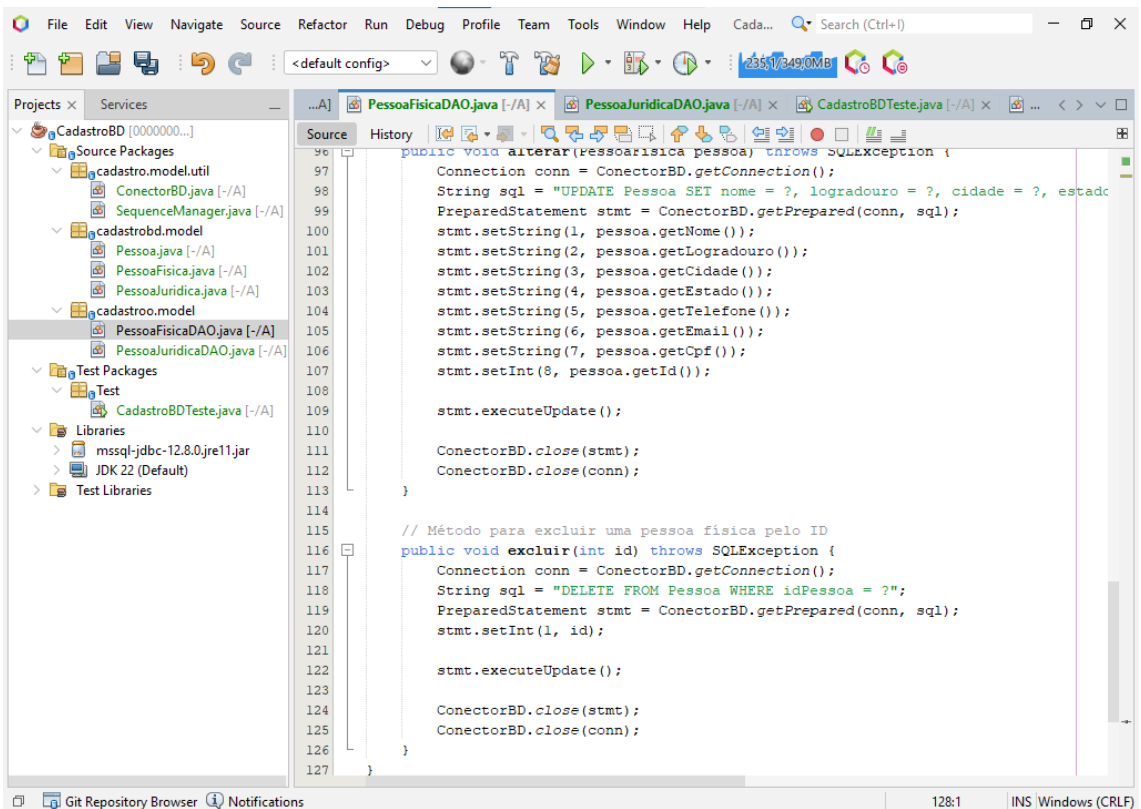
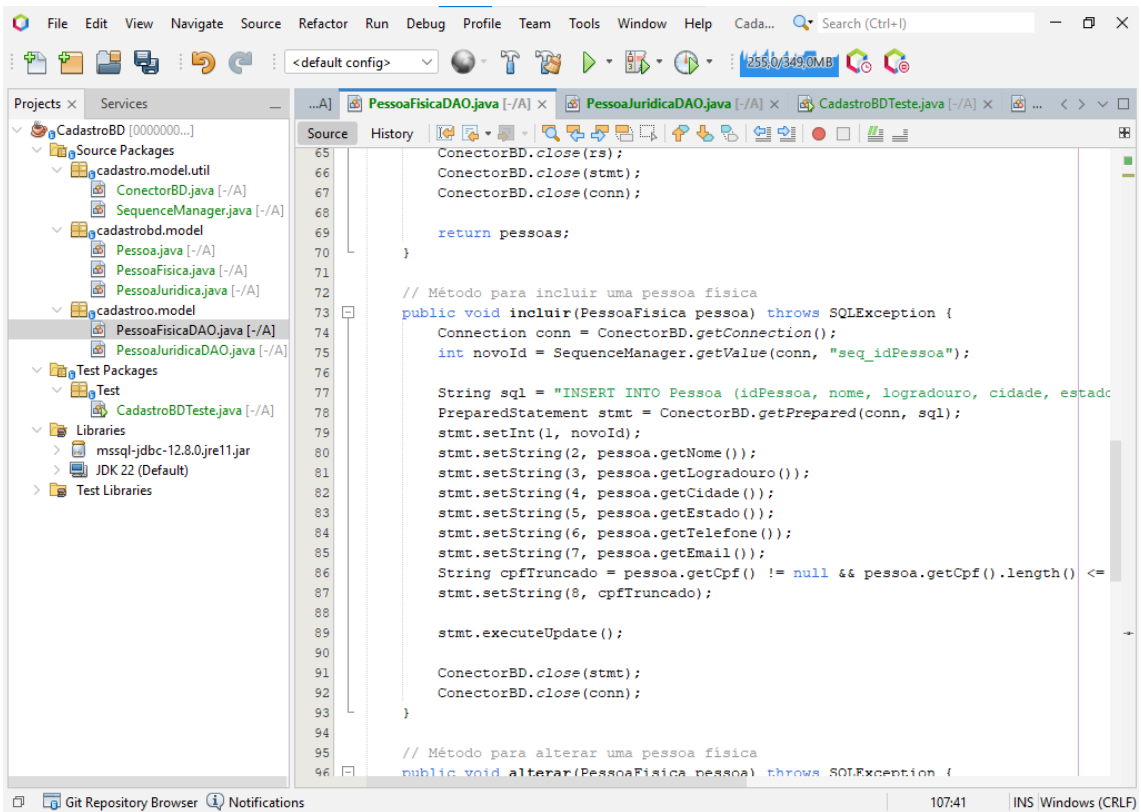


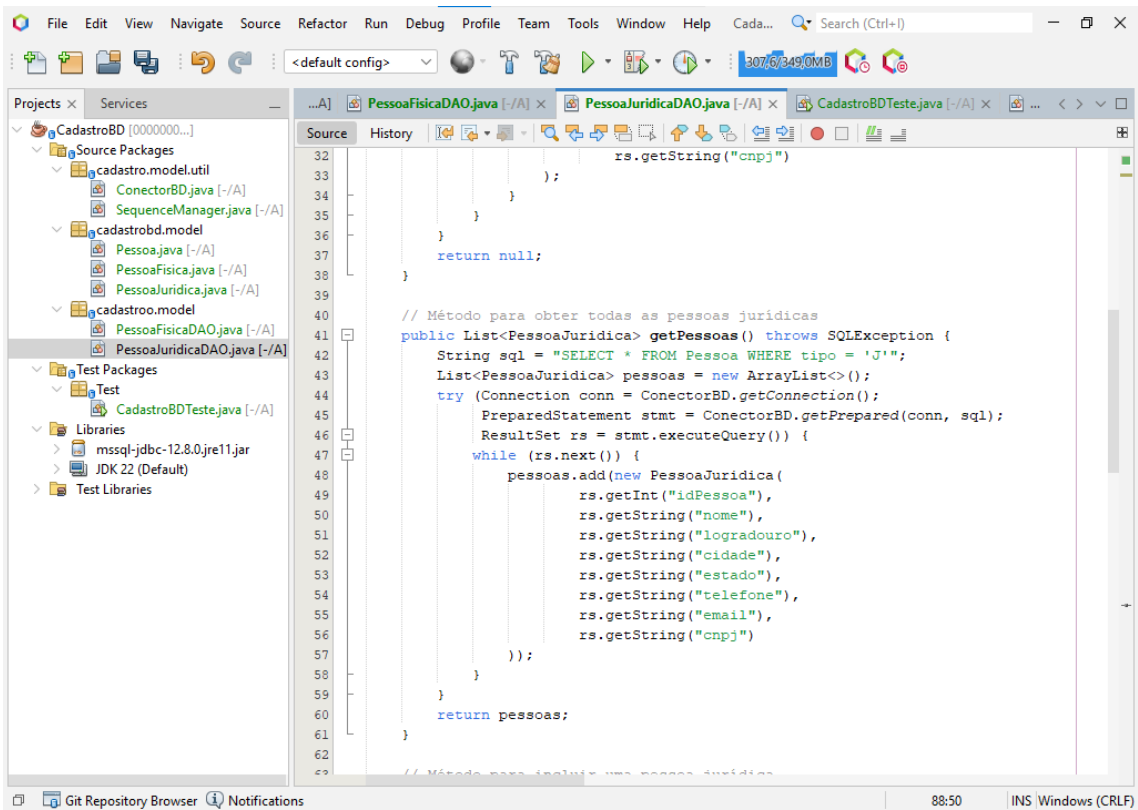
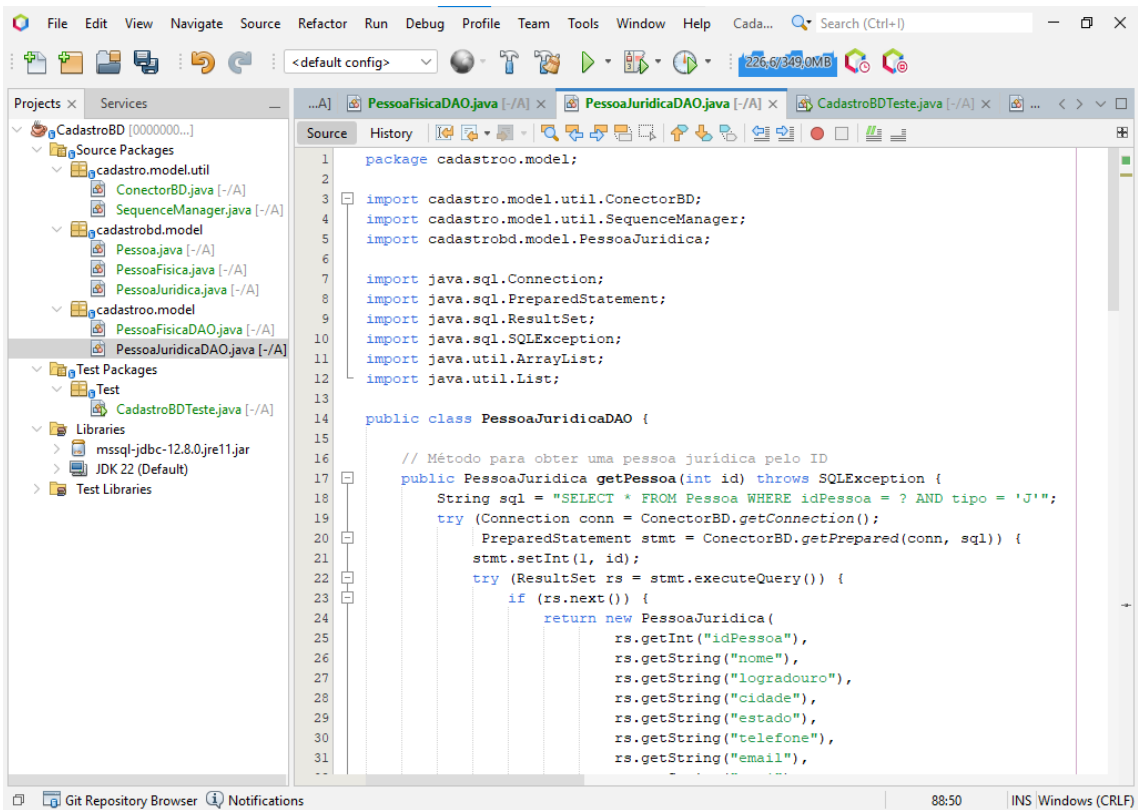
```
1 package cadastrobd.model;
2
3 public class Pessoa {
4     protected int id;
5     protected String nome;
6     protected String logradouro;
7     protected String cidade;
8     protected String estado;
9     protected String telefone;
10    protected String email;
11
12    public Pessoa() {
13    }
14
15    public Pessoa(int id, String nome, String logradouro, String cidade, String estado, String telefone, String email) {
16        this.id = id;
17        this.nome = nome;
18        this.logradouro = logradouro;
19        this.cidade = cidade;
20        this.estado = estado;
21        this.telefone = telefone;
22        this.email = email;
23    }
24
25    @Override
26    public void exibir() {
27        System.out.println("ID: " + id);
28        System.out.println("Nome: " + nome);
29        System.out.println("Logradouro: " + logradouro);
30        System.out.println("Cidade: " + cidade);
31        System.out.println("Estado: " + estado);
32        System.out.println("Telefone: " + telefone);
33        System.out.println("Email: " + email);
34    }
35 }
```

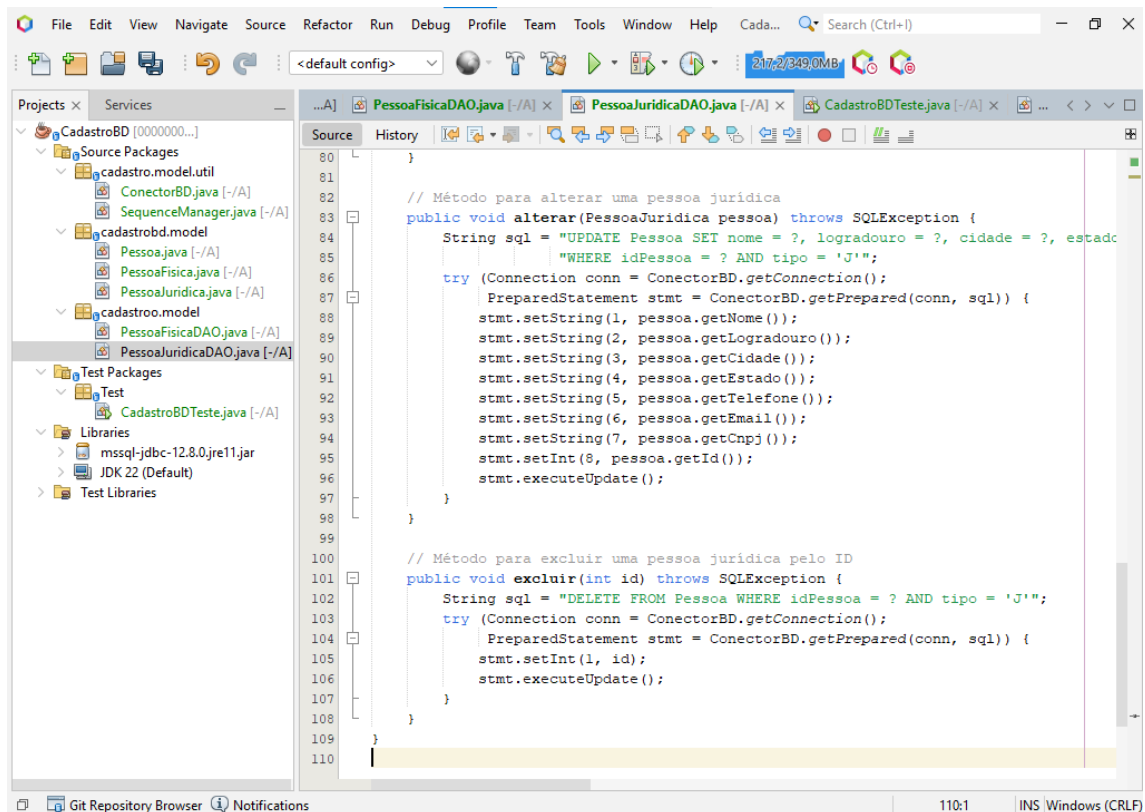
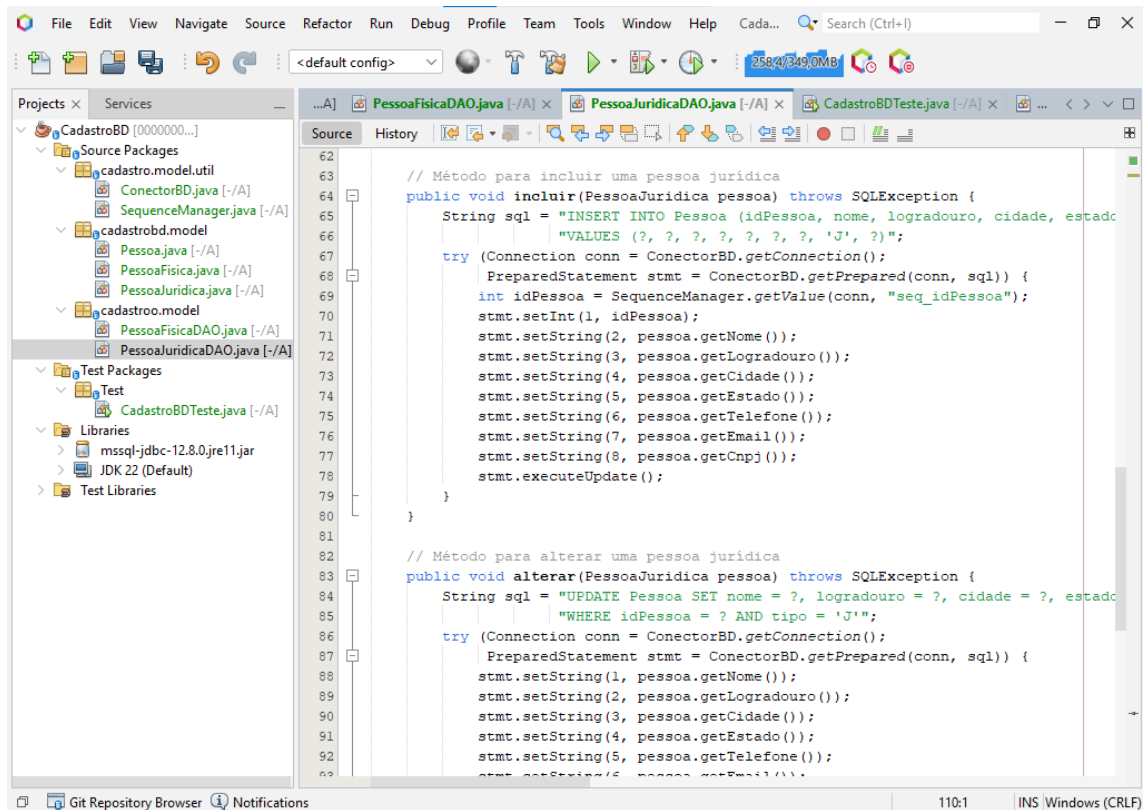


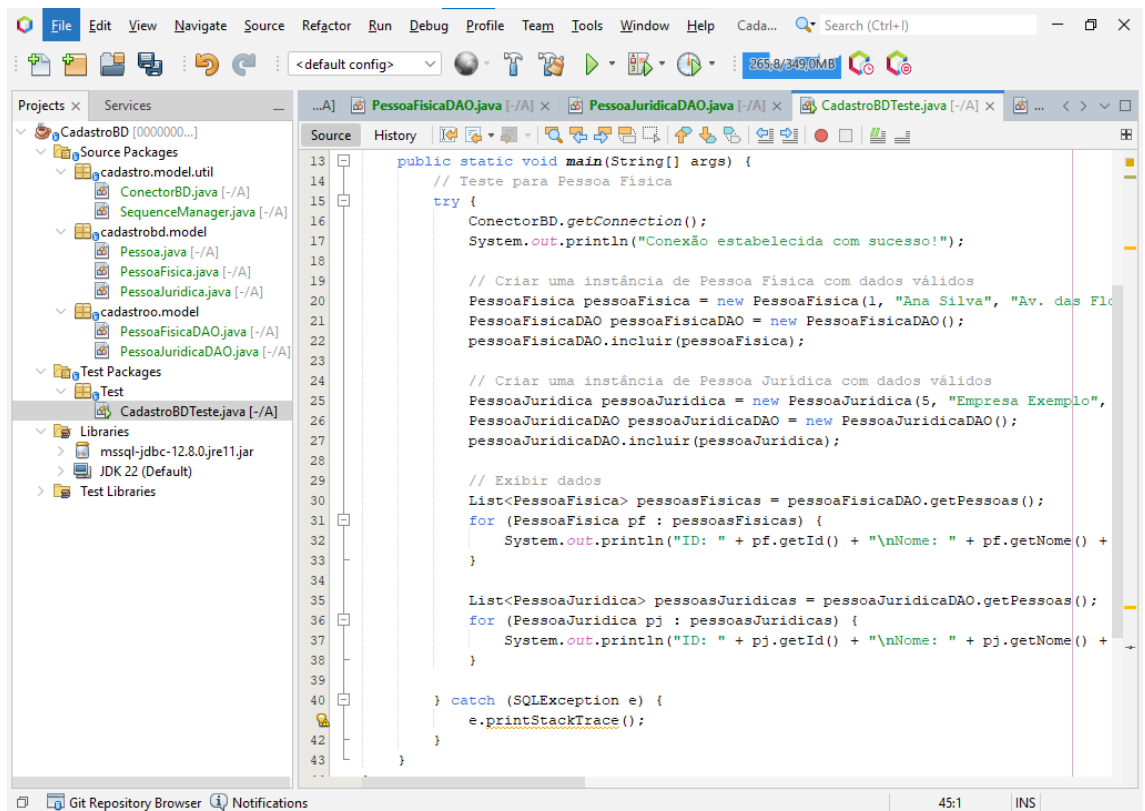
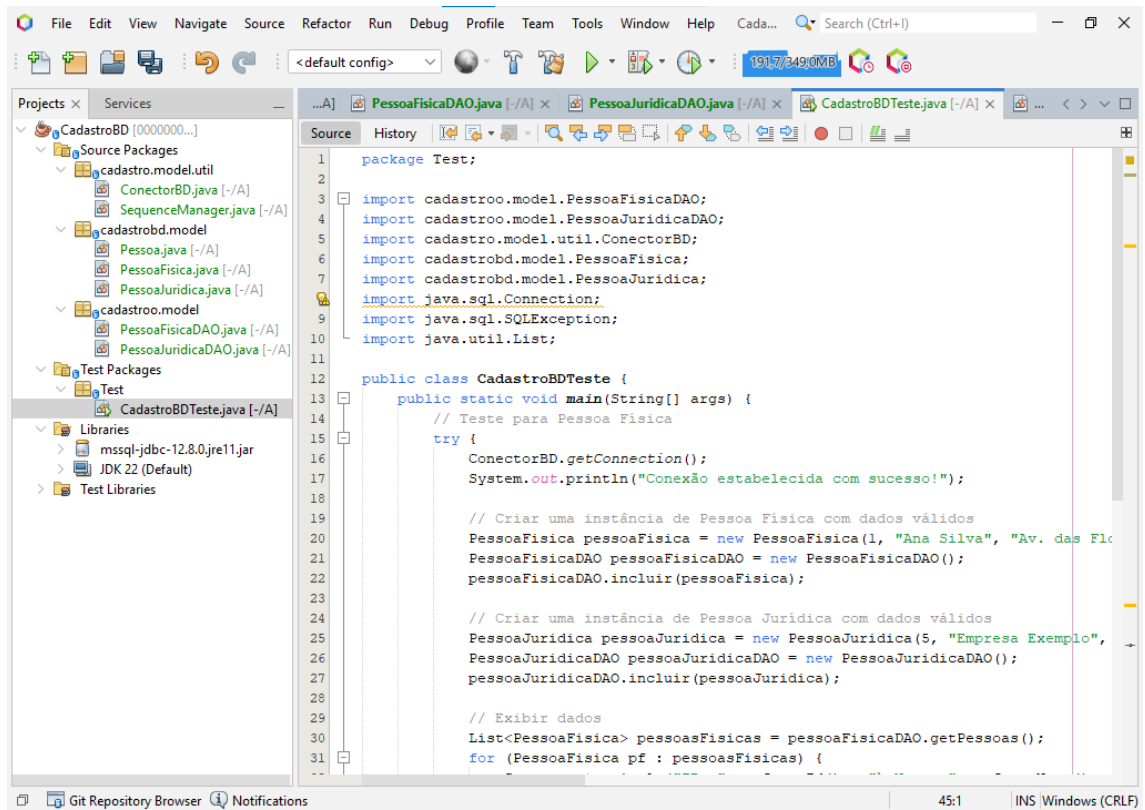












1. Qual a importância dos componentes de middleware, como o JDBC?

O JDBC (Java Database Connectivity) é um componente de middleware crucial para a interação entre aplicações Java e bancos de dados relacionais. Sua importância reside na capacidade de fornecer uma interface padronizada e independente de banco de dados para executar operações de SQL, como consultas e atualizações. O JDBC abstrai as complexidades da comunicação direta com o banco de dados, permitindo que os desenvolvedores se concentrem na lógica de negócios da aplicação. Ele oferece um meio eficiente e consistente para acessar, manipular e gerenciar dados, o que é fundamental para a construção de aplicações robustas e escaláveis.

2. Qual a diferença no uso de Statement ou PreparedStatement para a manipulação de dados?

- **Statement:** O Statement é utilizado para executar consultas SQL que não possuem parâmetros. Ele é adequado para consultas simples e ad hoc, mas não é tão eficiente para consultas repetidas. Além disso, o Statement é vulnerável a injeções SQL, o que pode comprometer a segurança da aplicação.
- **PreparedStatement:** O PreparedStatement é uma extensão do Statement que permite a execução de consultas SQL parametrizadas. Ele é mais eficiente para consultas repetidas, pois o banco de dados pode pré-compilar a consulta e reutilizar o plano de execução. Além disso, o PreparedStatement oferece proteção contra injeções SQL, uma vez que os parâmetros são tratados separadamente da consulta SQL. Isso melhora a segurança e o desempenho da aplicação.

3. Como o padrão DAO melhora a manutenibilidade do software?

O padrão DAO (Data Access Object) melhora a manutenibilidade do software ao separar a lógica de acesso a dados da lógica de negócios. Com o DAO, a manipulação dos dados é abstraída em objetos específicos que são responsáveis por interagir com a camada de persistência. Isso resulta em uma

arquitetura mais modular e organizada, onde alterações na forma como os dados são acessados ou persistidos não afetam diretamente a lógica de negócios. O DAO facilita a manutenção e evolução do software, uma vez que permite mudanças na camada de acesso a dados sem impactar outras partes da aplicação, e promove uma melhor organização do código.

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

Em um modelo relacional estrito, a herança é geralmente refletida por meio da criação de tabelas distintas para cada entidade, com possíveis relacionamentos entre elas. Existem diferentes abordagens para implementar herança em bancos de dados relacionais:

- **Tabela por Subclasse:** Cada subclasse é representada por uma tabela separada que inclui uma chave estrangeira para a tabela da superclasse. Isso permite armazenar atributos específicos de cada subclasse e herdar atributos da superclasse.
- **Tabela por Classe Única:** Uma única tabela é criada para armazenar todas as informações da superclasse e das subclasses, com colunas adicionais para armazenar atributos específicos das subclasses. Isso pode resultar em tabelas com muitas colunas nulas se as subclasses tiverem muitos atributos distintos.
- **Tabela por Superclasse e Subclasses:** Uma tabela para a superclasse e tabelas separadas para cada subclasse, com relacionamentos de chave estrangeira entre a tabela da superclasse e as tabelas das subclasses. Isso permite uma estrutura mais organizada, mas pode exigir joins complexos para consultas que envolvem herança.

Essas abordagens têm diferentes implicações para a normalização e a eficiência das consultas no banco de dados.

<https://github.com/SaloGarcia/nivel03mundo03.git>