 <b>Estácio</b>	<p style="text-align: center;">Universidade Estácio Campus Belém Curso de Desenvolvimento Full Stack Relatório da Missão Prática 3 - Mundo 3</p>
Disciplina:	RPG0016 - Backend sem banco não tem
Nome:	Breno Ambrosim Louzada
Turma:	2023.1

Criação de aplicativo Java, com acesso ao BD SQL Server através do middleware JDBC

## 1. Título da Prática: “1º Procedimento | Mapeamento Objeto-Relacional e DAO”

## 2. Objetivo da Prática

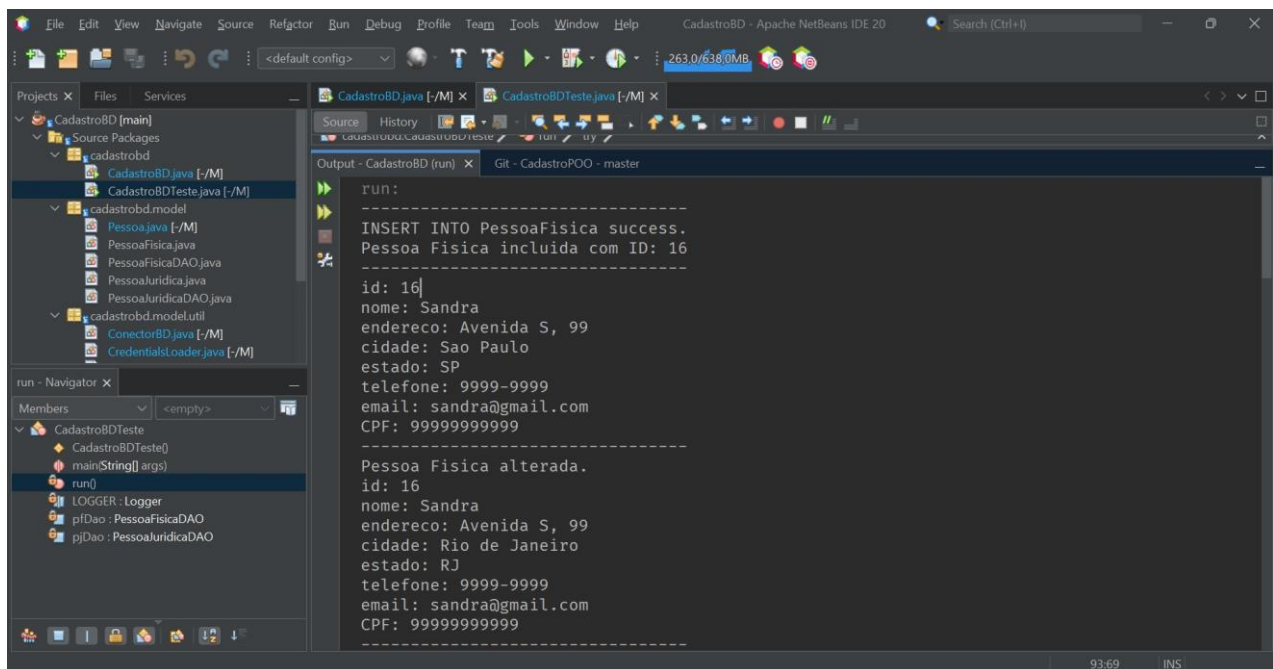
- Implementar persistência com base no middleware JDBC.
- Utilizar o padrão DAO (Data Access Object) no manuseio de dados.
- Implementar o mapeamento objeto-relacional em sistemas Java.
- Criar sistemas cadastrais com persistência em banco relacional.
- No final do exercício, o aluno terá criado um aplicativo cadastral com uso do SQL Server na persistência de dados.

## 3. Códigos solicitados: estão no repositório

<https://github.com/BrenoAmbrosim/CadastroBD>

e também no Anexo I, no final do relatório.

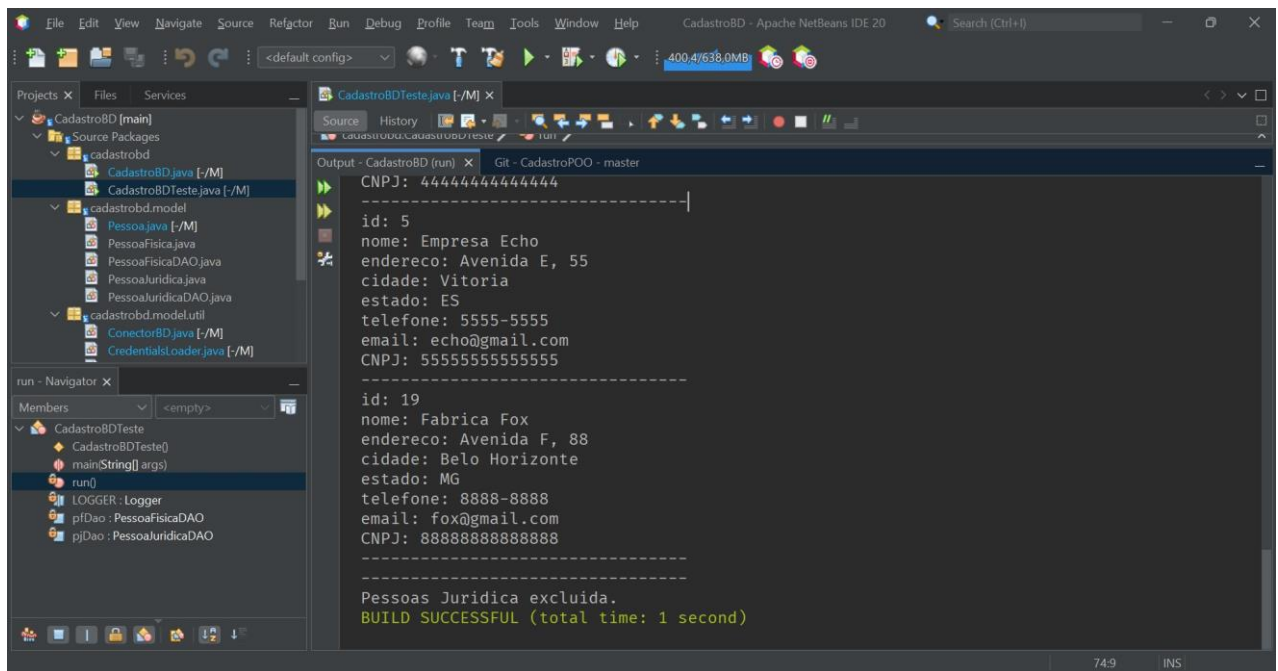
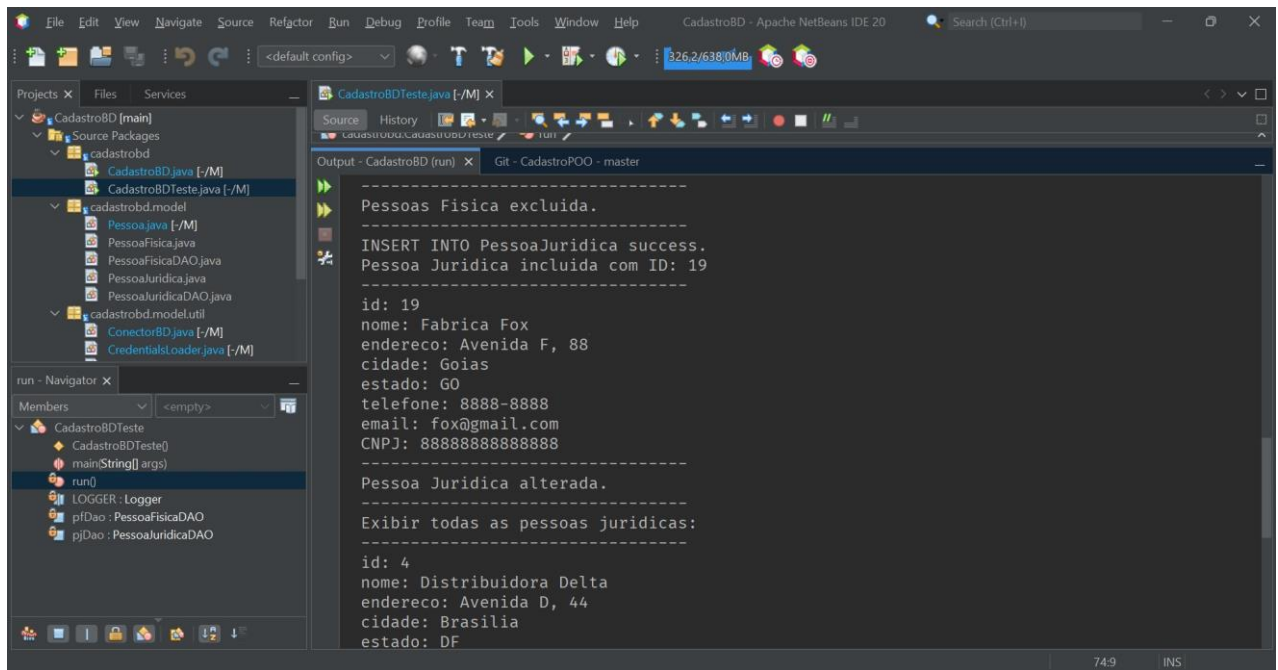
## 4. Resultados da execução dos códigos



```

run:
-----
INSERT INTO PessoaFisica success.
Pessoa Física incluída com ID: 16
-----
id: 16
nome: Sandra
endereco: Avenida S, 99
cidade: Sao Paulo
estado: SP
telefone: 9999-9999
email: sandra@gmail.com
CPF: 999999999999
-----
Pessoa Física alterada.
id: 16
nome: Sandra
endereco: Avenida S, 99
cidade: Rio de Janeiro
estado: RJ
telefone: 9999-9999
email: sandra@gmail.com
CPF: 999999999999
-----

```



## 5. Análise e Conclusão

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

Os componentes de middleware, como o JDBC (Java Database Connectivity), são essenciais porque atuam como intermediários entre diferentes sistemas de software, facilitando a comunicação e o intercâmbio de dados. No caso do JDBC, ele permite que aplicações Java se conectem e executem operações em bancos de dados de maneira padronizada, independente do sistema de gestão de banco de dados (SGBD) utilizado. Isso simplifica o desenvolvimento, aumenta a portabilidade das aplicações e promove a interoperabilidade entre sistemas heterogêneos.

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

A diferença principal entre Statement e PreparedStatement no Java JDBC está na performance e segurança: 1. PreparedStatement: É compilado pela base de dados antecipadamente, o que significa que ao utilizá-lo com parâmetros, ele pode ser reutilizado com diferentes valores, otimizando a performance. Além disso, oferece proteção contra injeção de SQL, pois os valores dos parâmetros são tratados de forma segura pelo driver JDBC. 2. Statement: É adequado para consultas que não necessitam de parâmetros e serão executadas uma única vez. Não é pré-compilado, o que pode ser menos eficiente para execuções repetidas, e não tem a mesma proteção contra injeção de SQL, pois a consulta é montada como uma única String, podendo ser vulnerável se os dados não forem adequadamente sanitizados. PreparedStatement é geralmente recomendado para operações que incluem dados dinâmicos devido à segurança e performance, enquanto Statement pode ser usado para consultas estáticas simples.

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

O padrão DAO (Data Access Object) melhora a manutenibilidade do software separando a lógica de acesso a dados da lógica de negócio. Isso é feito encapsulando o código de acesso a dados em classes específicas (DAOs). As principais vantagens incluem: Abstração e Encapsulamento: Ao centralizar o acesso aos dados, o DAO esconde os detalhes específicos do mecanismo de armazenamento ou da fonte de dados utilizada. Facilidade de Mudança: Mudanças no esquema do banco de dados ou na tecnologia de persistência impactam apenas o DAO, não o restante do código. Reusabilidade: O mesmo DAO pode ser usado em diferentes partes do sistema, evitando a duplicação de código. Testabilidade: Com o acesso a dados isolado, é mais fácil mockar essas camadas para testar componentes de negócio de forma independente. Organização: O código fica mais organizado com responsabilidades bem definidas, facilitando o entendimento e a manutenção do software.

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

No banco de dados relacional, a herança é emulada, pois não há suporte nativo como em linguagens de programação orientadas a objetos. Isso é geralmente feito usando três técnicas: a primeira, Tabela Única, onde uma única tabela contém as colunas para todas as propriedades de todas as classes na hierarquia; a segunda, Tabela por Classe, onde cada classe tem sua própria tabela contendo todas as suas propriedades, incluindo as herdadas; e a terceira, Tabela por Subclasse, onde a superclasse tem uma tabela e cada subclasse tem sua própria tabela contendo apenas as propriedades que não estão na superclasse. A escolha entre essas técnicas depende do equilíbrio desejado entre redundância de dados, performance de consultas e complexidade na manutenção do esquema do banco.

---

## 1. Título da Prática: “2º Procedimento | Alimentando a Base”

### 2. Objetivo da Prática

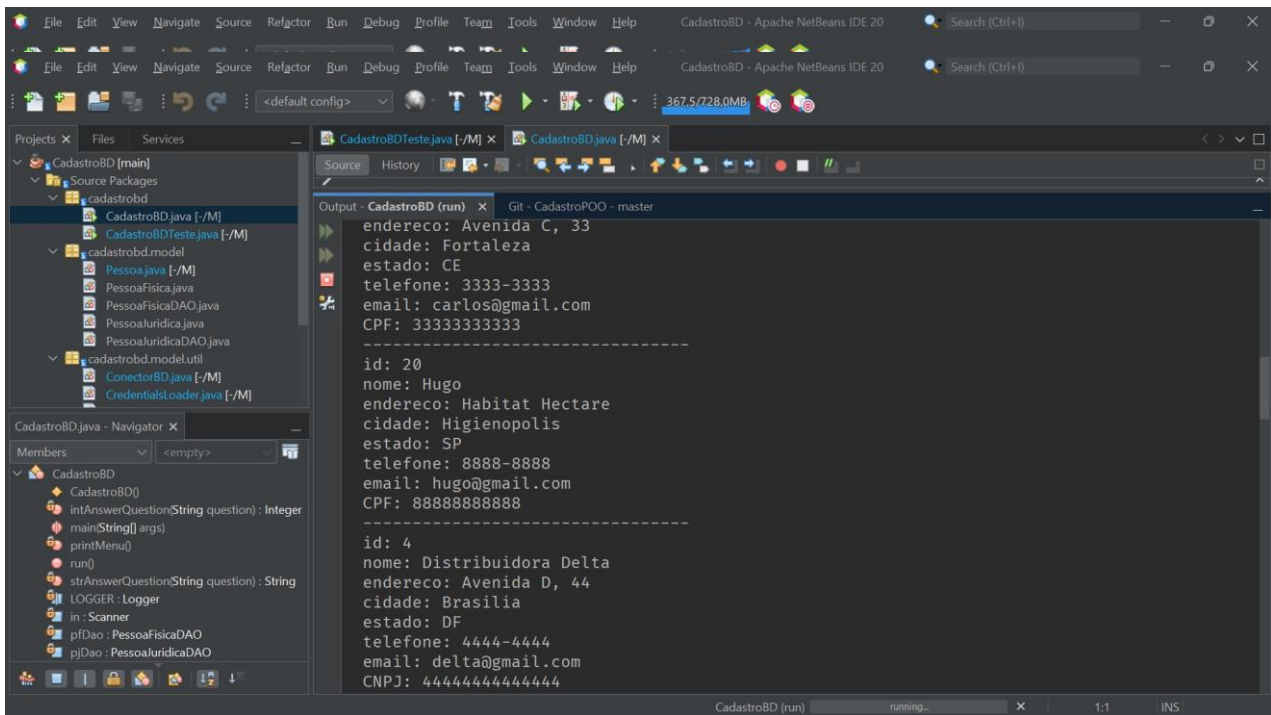
- Identificar os requisitos de um sistema e transformá-los no modelo adequado.
- Utilizar ferramentas de modelagem para bases de dados relacionais.
- Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
- Explorar a sintaxe SQL na consulta e manipulação de dados (DML)

- No final do exercício, o aluno terá vivenciado a experiência de modelar a base de dados para um sistema simples, além de implementá-la, através da sintaxe SQL, na plataforma do SQL Server.

### 3. Códigos solicitados: estão no repositório

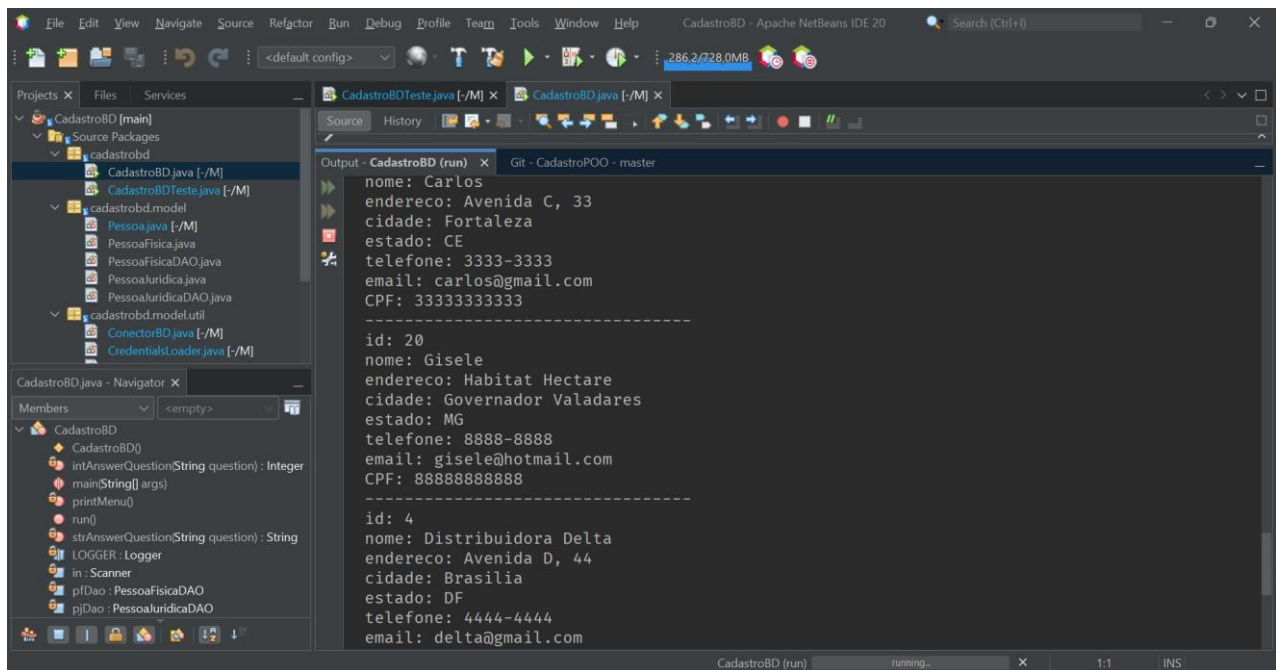
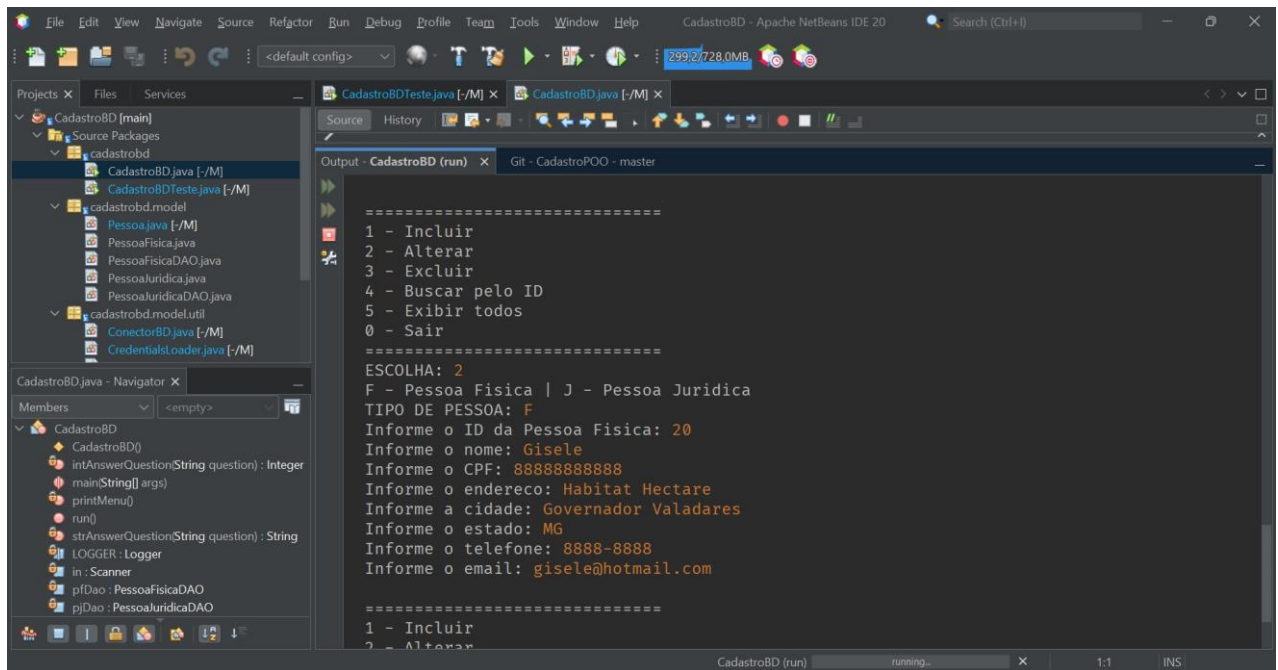
<https://github.com/BrenoAmbrosim/CadastroBD> e também no Anexo I, no final do relatório.

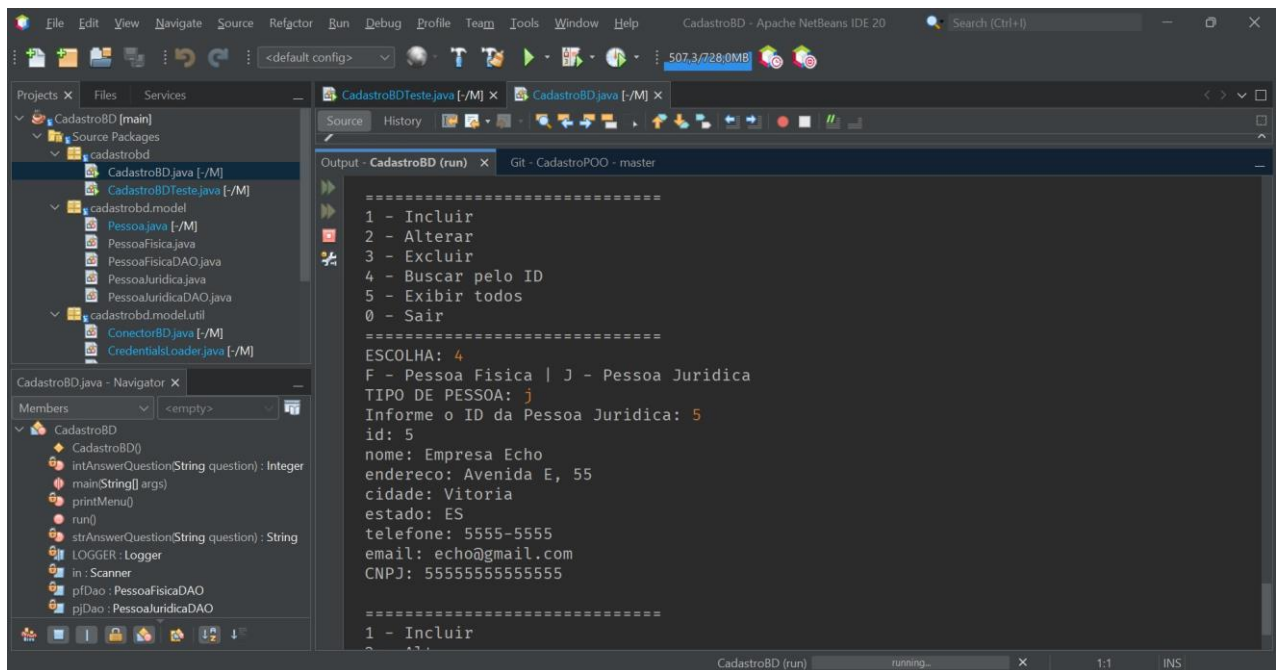
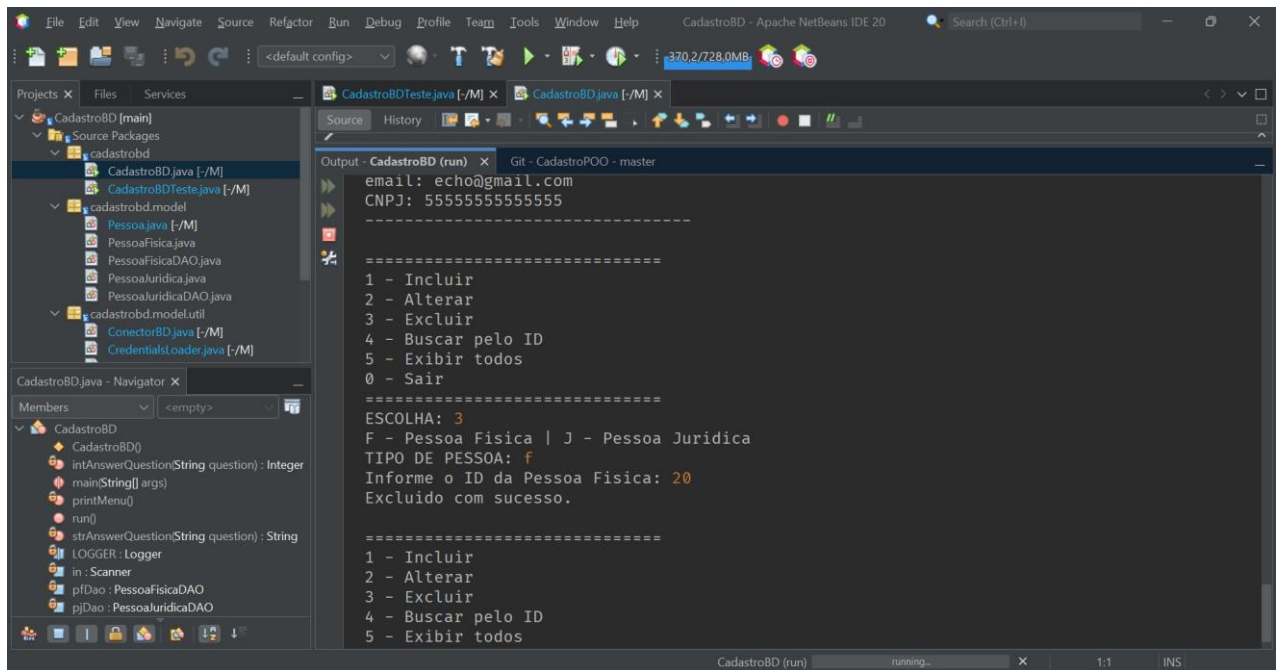
### 4. Resultados da execução dos códigos



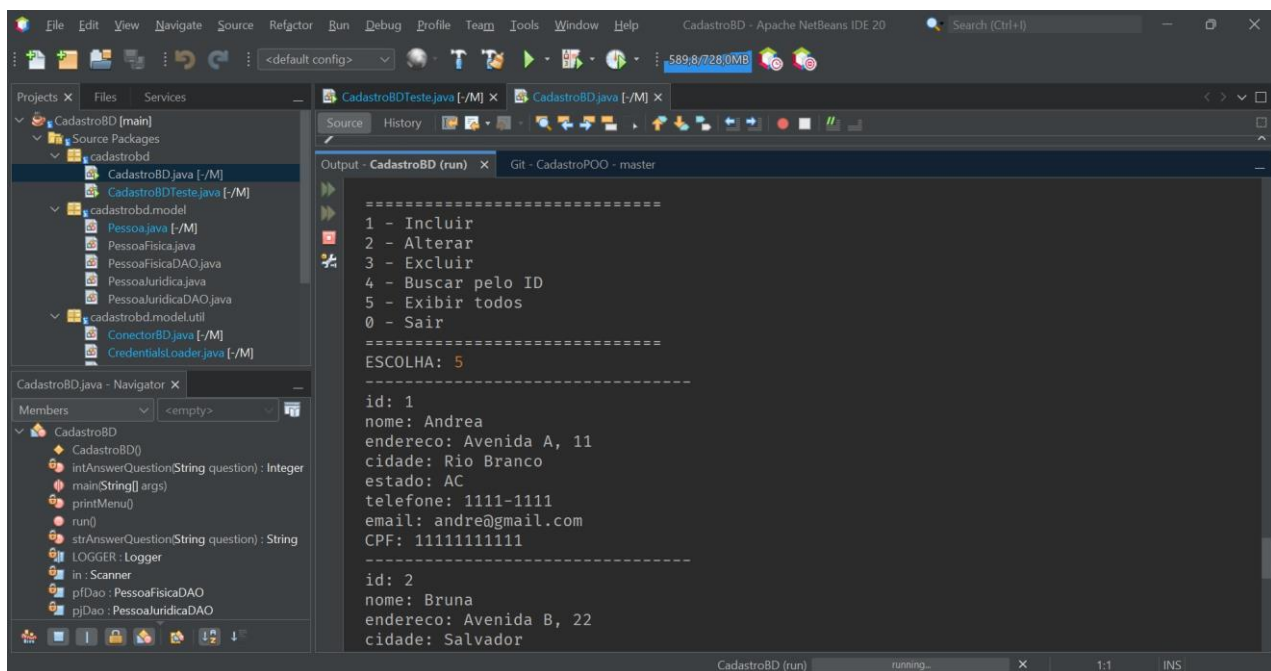
```

Output - CadastroBD (run) x  Git - CadastroPOO - master
endereco: Avenida C, 33
cidade: Fortaleza
estado: CE
telefone: 3333-3333
email: carlos@gmail.com
CPF: 33333333333
-----
id: 20
nome: Hugo
endereco: Habitat Hectare
cidade: Higienopolis
estado: SP
telefone: 8888-8888
email: hugo@gmail.com
CPF: 88888888888
-----
id: 4
nome: Distribuidora Delta
endereco: Avenida D, 44
cidade: Brasilia
estado: DF
telefone: 4444-4444
email: delta@gmail.com
CNPJ: 444444444444444
  
```









## 5. Análise e Conclusão

(a) Quais as diferenças entre a persistência em arquivo e a persistência em banco de dados?

A persistência em arquivos consiste no armazenamento de dados em um sistema de arquivos em formatos binário, por exemplo, arquivos com extensão .dat, .bin, .xls, .xlsx; ou arquivos texto, por exemplo, com extensões .csv, .json, .xml, são reconhecidamente arquivos de dados. É uma forma simples e direta de salvar dados, mas não é facilmente escalável, inicialmente segura ou eficiente, para consultas complexas e manipulação de dados quando comparada à persistência em banco de dados.

Por outro lado, a persistência em bancos de dados utiliza sistemas de gerenciamento de banco de dados (SGBD) para armazenar, recuperar e gerenciar dados de maneira estruturada, com uso de tabelas, índices, suporta operações complexas e transações. Também oferece recursos avançados como atomicidade, consistência, isolamento, durabilidade (ACID), além de segurança, backup e otimizações para acessos simultâneos.

(b) 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, a partir do Java 8, simplificou a impressão de valores contidos em entidades, ao permitir que desenvolvedores escrevam código mais legível e conciso, sem a necessidade de criar classes anônimas para operações simples. Com operadores lambda, é possível passar comportamentos de forma direta e declarativa.

Por exemplo, para imprimir os valores de uma lista, antes do Java 8, seria necessário criar um loop explícito. Com lambdas, isso pode ser feito de maneira simplificada usando métodos de stream e operações como **forEach**

```
lista.forEach(elemento -> System.out.println(elemento));
```

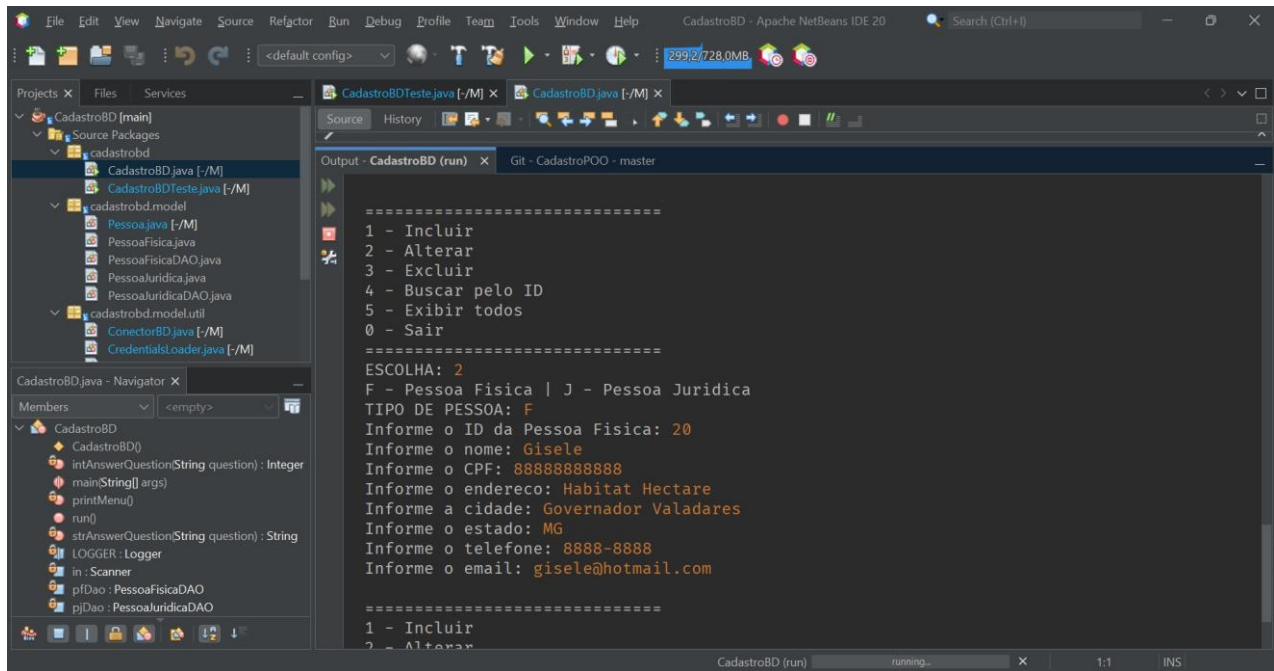


Esse código substitui várias linhas de um loop for ou loop aprimorado for-each, tradicionais, ao mesmo tempo que torna o código mais expressivo, focado no que realmente se deseja realizar (a ação de imprimir), e não em como realizar o loop ou controle de fluxo.

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

O método main é o primeiro método a ser executado quando uma classe Java é compilada e executada, caso, obviamente, a classe contenha um método main. A principal razão pela qual o método main é definido como estático, é justamente para conceder a possibilidade de ser diretamente executado sem necessidade de instanciar um objeto do tipo da classe, para depois invocar o método main, ou seja, uma comodidade para evitar código desnecessário.

Por definição, atributos ou métodos estáticos pertencem à classe onde são definidos, e não ao objeto instanciado a partir dessa classe. Assim, se o atributo ou método forem públicos, podem ser referenciados externamente por outras classes, através somente do nome da classe e não por objeto. Contudo, métodos estáticos, em suas implementações, ao tentar invocar outros métodos sem uso de objetos de referência, somente podem invocar diretamente outros métodos estáticos. Logo, ambos métodos precisam ser estáticos, tanto o método que invoca o outro método, como o método invocado.



```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package cadastrabd;

import java.util.ArrayList;
import java.util.Scanner;

import cadastrabd.model.PessoaFisica;
import cadastrabd.model.PessoaFisicaDAO;
import cadastrabd.model.PessoaJuridica;
import cadastrabd.model.PessoaJuridicaDAO;

import java.sql.SQLException;
import java.util.logging.Logger;
import java.util.logging.Level;

public class CadastroBD {

    private static final Logger LOGGER = Logger.getLogger(CadastroBD.class.getName());
    private Scanner in;
    private PessoaFisicaDAO pfDao;
    private PessoaJuridicaDAO pjDao;

    public CadastroBD() {
        in = new Scanner(System.in);
        pfDao = new PessoaFisicaDAO();
        pjDao = new PessoaJuridicaDAO();
    }

    private String strAnswerQuestion(String question) {
        System.out.print(question);
        return in.nextLine();
    }

    private Integer intAnswerQuestion(String question) {
        System.out.print(question);
        String strValue = in.nextLine();
        Integer intValue = 0;
        try {
            intValue = Integer.valueOf(strValue);
        }
        catch (NumberFormatException e) {
            LOGGER.log(Level.SEVERE, e.toString(), e);
        }
        return intValue;
    }

    private void printMenu() {
        System.out.println("\n=====");
        System.out.println("1 - Incluir");
        System.out.println("2 - Alterar");
        System.out.println("3 - Excluir");
        System.out.println("4 - Buscar pelo ID");
        System.out.println("5 - Exibir todos");
        System.out.println("0 - Sair");
        System.out.println("=====");
    }

    public void run() {

```

```

int opcao = -1;
while (opcao != 0) {
    printMenu();
    opcao = intAnswerQuestion("ESCOLHA: ");
    switch (opcao) {
        case 1: {
            System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
            String escolhaIncluir = strAnswerQuestion("TIPO DE PESSOA: ").toUpperCase();
            if (escolhaIncluir.equals("F")) {
                String nome = strAnswerQuestion("Informe o nome: ");
                String cpf = strAnswerQuestion("Informe o CPF: ");
                String endereco = strAnswerQuestion("Informe o endereco: ");
                String cidade = strAnswerQuestion("Informe a cidade: ");
                String estado = strAnswerQuestion("Informe o estado: ");
                String telefone = strAnswerQuestion("Informe o telefone: ");
                String email = strAnswerQuestion("Informe o email: ");
                try {
                    pfDao.incluir(new PessoaFisica(null, nome, endereco, cidade, estado, telefone, email, cpf));
                }
                catch (SQLException e) {
                    LOGGER.log(Level.SEVERE, e.toString(), e);
                }
            }
            else if (escolhaIncluir.equals("J")) {
                String nome = strAnswerQuestion("Informe o nome: ");
                String cnpj = strAnswerQuestion("Informe o CNPJ: ");
                String endereco = strAnswerQuestion("Informe o endereco: ");
                String cidade = strAnswerQuestion("Informe a cidade: ");
                String estado = strAnswerQuestion("Informe o estado: ");
                String telefone = strAnswerQuestion("Informe o telefone: ");
                String email = strAnswerQuestion("Informe o email: ");
                try {
                    pjDao.incluir(new PessoaJuridica(null, nome, endereco, cidade, estado, telefone, email, cnpj));
                }
                catch (SQLException e) {
                    LOGGER.log(Level.SEVERE, e.toString(), e);
                }
            }
            else {
                System.out.println("Erro: Escolha Invalida!");
            }
        }; break;
        case 2: {
            System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
            String escolhaAlterar = strAnswerQuestion("TIPO DE PESSOA: ").toUpperCase();
            if (escolhaAlterar.equals("F")) {
                try {
                    Integer id = intAnswerQuestion("Informe o ID da Pessoa Fisica: ");
                    PessoaFisica pf = pfDao.getPessoa(id);
                    if (pf != null) {
                        pf.setNome(strAnswerQuestion("Informe o nome: "));
                        pf.setCpf(strAnswerQuestion("Informe o CPF: "));
                        pf.setEndereco(strAnswerQuestion("Informe o endereco: "));
                        pf.setCidade(strAnswerQuestion("Informe a cidade: "));
                        pf.setEstado(strAnswerQuestion("Informe o estado: "));
                        pf.setTelefone(strAnswerQuestion("Informe o telefone: "));
                        pf.setEmail(strAnswerQuestion("Informe o email: "));
                        pfDao.alterar(pf);
                    }
                    else {
                        System.out.println("ID nao encontrado!");
                    }
                }
            }
        }
    }
}

```

```

    }
    catch (NullPointerException | SQLException e) {
        LOGGER.log(Level.SEVERE, e.toString(), e);
    }
}

else if (escolhaAlterar.equals("J")) {
    try {
        Integer id = intAnswerQuestion("Informe o ID da Pessoa Juridica: ");
        PessoaJuridica pj = pjDao.getPessoa(id);
        if (pj != null) {
            pj.setNome(strAnswerQuestion("Informe o nome: "));
            pj.setCnpj(strAnswerQuestion("Informe o CNPJ: "));
            pj.setEndereco(strAnswerQuestion("Informe o endereco: "));
            pj.setCidade(strAnswerQuestion("Informe a cidade: "));
            pj.setEstado(strAnswerQuestion("Informe o estado: "));
            pj.setTelefone(strAnswerQuestion("Informe o telefone: "));
            pj.setEmail(strAnswerQuestion("Informe o email: "));
            pjDao.alterar(pj);
        }
        else {
            System.out.println("ID nao encontrado!");
        }
    }
    catch (NullPointerException | SQLException e) {
        LOGGER.log(Level.SEVERE, e.toString(), e);
    }
}

else {
    System.out.println("Erro: Escolha Invalida!");
}

}; break;
case 3: {
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    String escolhaExcluir = strAnswerQuestion("TIPO DE PESSOA: ").toUpperCase();
    if (escolhaExcluir.equals("F")) {
        try {
            Integer id = intAnswerQuestion("Informe o ID da Pessoa Fisica: ");
            PessoaFisica pf = pfDao.getPessoa(id);
            if (pf != null) {
                pfDao.excluir(pf);
                System.out.println("Excluido com sucesso.");
            }
            else {
                System.out.println("ID nao encontrado.");
            }
        }
        catch (NullPointerException | SQLException e) {
            LOGGER.log(Level.SEVERE, e.toString(), e);
        }
    }
    else if (escolhaExcluir.equals("J")) {
        try {
            Integer id = intAnswerQuestion("Informe o ID da Pessoa Juridica: ");
            PessoaJuridica pj = pjDao.getPessoa(id);
            if (pj != null) {
                pjDao.excluir(pj);
                System.out.println("Excluido com sucesso.");
            }
            else {
                System.out.println("ID nao encontrado.");
            }
        }
    }
}

```

```

    }
    catch (NullPointerException | SQLException e) {
        LOGGER.log(Level.SEVERE, e.toString(), e);
    }
}
else {
    System.out.println("Erro: Escolha Invalida!");
}
}; break;
case 4: {
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    String escolhaExibir = strAnswerQuestion("TIPO DE PESSOA: ").toUpperCase();
    if (escolhaExibir.equals("F")) {
        try {
            PessoaFisica pf = pfDao.getPessoa(intAnswerQuestion("Informe o ID da Pessoa Fisica: "));
            if (pf != null) {
                pf.exibir();
            }
        }
        catch (SQLException e){
            System.err.println("Pessoa nao encontrada!");
            LOGGER.log(Level.SEVERE, e.toString(), e);
        }
    }
    else if (escolhaExibir.equals("J")) {
        try {
            PessoaJuridica pj = pjDao.getPessoa(intAnswerQuestion("Informe o ID da Pessoa Juridica: "));
            if (pj != null) {
                pj.exibir();
            }
        }
        catch (SQLException e){
            System.err.println("Pessoa nao encontrada!");
            LOGGER.log(Level.SEVERE, e.toString(), e);
        }
    }
    else {
        System.out.println("Erro: Escolha Invalida!");
    }
}; break;
case 5: {
    try {
        ArrayList<PessoaFisica> listaPf = pfDao.getPessoas();
        for (PessoaFisica pessoa : listaPf) {
            System.out.println("-----");
            pessoa.exibir();
        }
        System.out.println("-----");
        ArrayList<PessoaJuridica> listaPj = pjDao.getPessoas();
        for (PessoaJuridica pessoa : listaPj) {
            pessoa.exibir();
            System.out.println("-----");
        }
    }
    catch (SQLException e) {
        LOGGER.log(Level.SEVERE, e.toString(), e);
    }
}; break;
default: System.out.println("Escolha invalida!");
}
}
}

```



```

/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    new CadastroBD().run();
}
}

```

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastrabd;

import java.sql.SQLException;
import java.util.ArrayList;

import cadastrabd.model.PessoaFisica;
import cadastrabd.model.PessoaFisicaDAO;
import cadastrabd.model.PessoaJuridica;
import cadastrabd.model.PessoaJuridicaDAO;

import java.util.logging.Logger;
import java.util.logging.Level;

public class CadastroBDTeste {

    private static final Logger LOGGER = Logger.getLogger(CadastroBDTeste.class.getName());

    private final PessoaFisicaDAO pfDao;
    private final PessoaJuridicaDAO pjDao;

    public CadastroBDTeste() {
        pfDao = new PessoaFisicaDAO();
        pjDao = new PessoaJuridicaDAO();
    }

    private void run() {
        PessoaFisica pf = new PessoaFisica(null, "Sandra", "Avenida S, 99", "Sao Paulo",
            "SP", "9999-9999", "sandra@gmail.com", "999999999999");

        if (pf.getNome() == null || pf.getNome().trim().isEmpty()) {
            System.out.println("'nome' cannot be empty or null.");
            return;
        }

        try {
            System.out.println("-----");
            System.out.println("Pessoa Fisica incluida com ID: " + pfDao.incluir(pf));
            System.out.println("-----");
            pf.exibir();
            pf.setCidade("Rio de Janeiro");
            pf.setEstado("RJ");

```

```

        pfDao.alterar(pf);
        System.out.println("-----");
        System.out.println("Pessoa Fisica alterada.");
        pf.exibir();
        ArrayList<PessoaFisica> listaPf = pfDao.getPessoas();
        System.out.println("-----");
        System.out.println("Exibir todas as pessoas fisicas:");
        for (PessoaFisica pessoa : listaPf) {
            System.out.println("-----");
            pessoa.exibir();
        }
        System.out.println("-----");
        pfDao.excluir(pf);
        System.out.println("-----");
        System.out.println("Pessoa Fisica excluida.");
        pfDao.close();
    }
    catch (SQLException e) {
        LOGGER.log(Level.SEVERE, e.toString(), e);
    }
}

PessoaJuridica pj = new PessoaJuridica(null, "Fabrica Fox", "Avenida F, 88", "Goias",
    "GO", "8888-8888", "fox@gmail.com", "8888888888888888");

if (pj.getNome() == null || pj.getNome().trim().isEmpty()) {
    System.out.println("'nome' cannot be empty or null.");
    return;
}

try {
    System.out.println("-----");
    System.out.println("Pessoa Juridica incluida com ID: " + pjDao.incluir(pj));
    System.out.println("-----");
    pj.exibir();
    pj.setCidade("Belo Horizonte");
    pj.setEstado("MG");
    pjDao.alterar(pj);
    System.out.println("-----");
    System.out.println("Pessoa Juridica alterada.");
    ArrayList<PessoaJuridica> listaPj = pjDao.getPessoas();
    System.out.println("-----");
    System.out.println("Exibir todas as pessoas juridicas:");
    for (PessoaJuridica pessoa : listaPj) {
        System.out.println("-----");
        pessoa.exibir();
    }
    System.out.println("-----");
    pjDao.excluir(pj);
    System.out.println("-----");
    System.out.println("Pessoa Juridica excluida.");
    pjDao.close();
}
catch (SQLException e) {
    LOGGER.log(Level.SEVERE, e.toString(), e);
}

}

public static void main(String[] args) {
    new CadastroBDTeste().run();
}

```

```
}
```

```
/*  
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license  
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template  
 */  
package cadastrbd.model;
```

```
public class Pessoa {
```

```
    private Integer id;  
    private String nome;  
    private String endereco;  
    private String cidade;  
    private String estado;  
    private String telefone;  
    private String email;
```

```
    public Pessoa() {
```

```
    }
```

```
    public Pessoa(Integer id, String nome, String endereco, String cidade,  
        String estado, String telefone, String email) {  
        this.id = id;  
        this.nome = nome;  
        this.endereco = endereco;  
        this.cidade = cidade;  
        this.estado = estado;  
        this.telefone = telefone;  
        this.email = email;  
    }
```

```
    public void exibir() {  
        System.out.println(this);  
    }
```

```
    public Integer getId() {  
        return id;  
    }
```

```
    public void setId(Integer id) {  
        this.id = id;  
    }
```

```
    public String getNome() {  
        return nome;  
    }
```

```
    public void setNome(String nome) {  
        this.nome = nome;  
    }
```

```
    public String getEndereco() {  
        return endereco;  
    }
```

```

public void setEndereco(String endereco) {
    this.endereco = endereco;
}

public String getCidade() {
    return cidade;
}

public void setCidade(String cidade) {
    this.cidade = cidade;
}

public String getEstado() {
    return estado;
}

public void setEstado(String estado) {
    this.estado = estado;
}

public String getTelefone() {
    return telefone;
}

public void setTelefone(String telefone) {
    this.telefone = telefone;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

@Override
public String toString() {
    String output = "id: ".concat(id.toString());
    output = output.concat("\nnome: ".concat(nome));
    output = output.concat("\nendereco: ".concat(endereco));
    output = output.concat("\ncidade: ".concat(cidade));
    output = output.concat("\nestado: ".concat(estado));
    output = output.concat("\ntelefone: ".concat(telefone));
    output = output.concat("\nemail: ".concat(email));
    return output;
}
}

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastrbd.model;

public class PessoaFisica extends Pessoa {

    private String cpf;

    public PessoaFisica() {

```

```

    }

    public PessoaFisica(Integer id, String nome, String endereco, String cidade,
        String estado, String telefone, String email, String cpf) {
        super(id, nome, endereco, cidade, estado, telefone, email);
        this.cpf = cpf;
    }

    @Override
    public void exibir() {
        System.out.println(this);
    }

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    @Override
    public String toString() {
        String output = super.toString();
        output = output.concat("\nCPF: ".concat(cpf));
        return output;
    }
}

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastrbd.model;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import cadastrbd.model.util.ConectorBD;

public class PessoaFisicaDAO {

    private final ConectorBD connector;

    public PessoaFisicaDAO() {
        connector = new ConectorBD();
    }

    public PessoaFisica getPessoa(Integer id) throws SQLException {
        String sql = "SELECT pf.FK_Pessoa_idPessoa, pf.cpf, p.nome, p.endereco, p.cidade, p.estado, p.telefone, p.email "
            + "FROM PessoaFisica pf "
            + "INNER JOIN Pessoa p ON pf.FK_Pessoa_idPessoa = p.idPessoa "
            + "WHERE pf.FK_Pessoa_idPessoa = ?";
        try (Connection con = connector.getConnection(); PreparedStatement stmt = con.prepareStatement(sql))

```

```

{
    stmt.setInt(1, id);
    try (ResultSet rs = stmt.executeQuery()) {
        if (rs.next()) {
            return new PessoaFisica(
                rs.getInt("FK_Pessoa_idPessoa"),
                rs.getString("nome"),
                rs.getString("endereco"),
                rs.getString("cidade"),
                rs.getString("estado"),
                rs.getString("telefone"),
                rs.getString("email"),
                rs.getString("cpf")
            );
        }
    }
}
return null;
}

public ArrayList<PessoaFisica> getPessoas() throws SQLException {
    ArrayList<PessoaFisica> list = new ArrayList<>();
    String sql = "SELECT pf.FK_Pessoa_idPessoa, pf.cpf, p.nome, p.endereco, p.cidade, p.estado, p.telefone,
p.email "
        + "FROM PessoaFisica pf "
        + "INNER JOIN Pessoa p ON pf.FK_Pessoa_idPessoa = p.idPessoa";
    try (Connection con = connector.getConnection(); PreparedStatement stmt =
        con.prepareStatement(sql); ResultSet rs = stmt.executeQuery()) {
        while (rs.next()) {
            list.add(new PessoaFisica(
                rs.getInt("FK_Pessoa_idPessoa"),
                rs.getString("nome"),
                rs.getString("endereco"),
                rs.getString("cidade"),
                rs.getString("estado"),
                rs.getString("telefone"),
                rs.getString("email"),
                rs.getString("cpf")));
        }
    }
    return list;
}

public int incluir(PessoaFisica pf) throws SQLException {
    if (pf.getNome() == null || pf.getNome().trim().isEmpty()) {
        throw new IllegalArgumentException("'nome' cannot be empty or null.");
    }
    String sqlInsertPessoa = "INSERT INTO Pessoa(nome, endereco, cidade, estado, telefone, email)
VALUES(?, ?, ?, ?, ?, ?)";
    String sqlInsertPessoaFisica = "INSERT INTO PessoaFisica(FK_Pessoa_idPessoa, cpf) VALUES(?, ?)";
    try (Connection con = connector.getConnection(); PreparedStatement stmtPessoa =
        con.prepareStatement(sqlInsertPessoa, Statement.RETURN_GENERATED_KEYS)) {
        String[] pfArray = {"", pf.getNome(), pf.getEndereco(), pf.getCidade(), pf.getEstado(), pf.getTelefone(),
pf.getEmail()};
        for(int i = 1; i < 7; i++) {
            stmtPessoa.setString(i, pfArray[i]);
        }
        if (stmtPessoa.executeUpdate() != 0) {
            System.out.println("INSERT INTO PessoaFisica success.");
        } else {
            throw new SQLException("Creating user failed, no rows affected.");
        }
    }
}

```



```

        try (ResultSet generatedKeys = stmtPessoa.getGeneratedKeys()) {
            if (generatedKeys.next()) {
                int idNovaPessoa = generatedKeys.getInt(1);
                pf.setId(idNovaPessoa);
                try (PreparedStatement stmtPessoaFisica = con.prepareStatement(sqlInsertPessoaFisica,
Statement.RETURN_GENERATED_KEYS)) {
                    stmtPessoaFisica.setInt(1, idNovaPessoa);
                    stmtPessoaFisica.setString(2, pf.getCpf());
                    stmtPessoaFisica.executeUpdate();
                }
                return idNovaPessoa;
            } else {
                throw new SQLException("Creating user failed. No ID obtained.");
            }
        }
    }
}

public void alterar(PessoaFisica pf) throws SQLException {
    String sqlUpdatePessoa = "UPDATE Pessoa SET nome = ?, endereco = ?, cidade = ?, estado = ?, telefone = ?,
email = ? WHERE idPessoa = ?;";
    String sqlUpdatePessoaFisica = "UPDATE PessoaFisica SET cpf = ? WHERE FK_Pessoa_idPessoa = ?;";
    try (Connection con = connector.getConnection();
        PreparedStatement stmtPessoa = con.prepareStatement(sqlUpdatePessoa);
        PreparedStatement stmtPessoaFisica = con.prepareStatement(sqlUpdatePessoaFisica)) {
        String[] pfArray = {"", pf.getNome(), pf.getEndereco(), pf.getCidade(), pf.getEstado(), pf.getTelefone(),
pf.getEmail()};
        for(int i = 1; i < 7; i++) {
            stmtPessoa.setString(i, pfArray[i]);
        }
        stmtPessoa.setInt(7, pf.getId());
        stmtPessoa.executeUpdate();
        stmtPessoaFisica.setString(1, pf.getCpf());
        stmtPessoaFisica.setInt(2, pf.getId());
        stmtPessoaFisica.executeUpdate();
    }
}

public void excluir(PessoaFisica pf) throws SQLException {
    String sqlDeletePessoaFisica = "DELETE FROM PessoaFisica WHERE FK_Pessoa_idPessoa = ?;";
    String sqlDeletePessoa = "DELETE FROM Pessoa WHERE idPessoa = ?;";
    try (Connection con = connector.getConnection(); PreparedStatement stmtPessoaFisica =
con.prepareStatement(sqlDeletePessoaFisica); PreparedStatement stmtPessoa =
con.prepareStatement(sqlDeletePessoa)) {
        stmtPessoaFisica.setInt(1, pf.getId());
        stmtPessoaFisica.executeUpdate();
        stmtPessoa.setInt(1, pf.getId());
        stmtPessoa.executeUpdate();
    }
}

public void close() throws SQLException {
    connector.close();
}
}

```

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

```

*/
package cadastrbd.model;

public class PessoaJuridica extends Pessoa {

    private String cnpj;

    public PessoaJuridica() {

    }

    public PessoaJuridica(Integer id, String nome, String endereco, String cidade,
        String estado, String telefone, String email, String cnpj) {
        super(id, nome, endereco, cidade, estado, telefone, email);
        this.cnpj = cnpj;
    }

    @Override
    public void exibir() {
        System.out.println(this);
    }

    public String getCnpj() {
        return cnpj;
    }

    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }

    @Override
    public String toString() {
        String output = super.toString();
        output = output.concat("\nCNPJ: ".concat(cnpj));
        return output;
    }
}

```

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastrbd.model;

import cadastrbd.model.util.ConectorBD;

import java.sql.*;
import java.util.ArrayList;

public class PessoaJuridicaDAO {

    private ConectorBD connector;

    public PessoaJuridicaDAO() {
        connector = new ConectorBD();
    }

    public PessoaJuridica getPessoa(Integer id) throws SQLException {

```

```

String sql = "SELECT pj.FK_Pessoa_idPessoa, pj.cnpj, p.nome, p.endereco, p.cidade, p.estado, p.telefone,
p.email "
+ "FROM PessoaJuridica pj "
+ "INNER JOIN Pessoa p ON pj.FK_Pessoa_idPessoa = p.idPessoa "
+ "WHERE pj.FK_Pessoa_idPessoa = ?";
try (Connection con = connector.getConnection();
    PreparedStatement stmt = con.prepareStatement(sql)) {
    stmt.setInt(1, id);
    try (ResultSet rs = stmt.executeQuery()) {
        if (rs.next()) {
            return new PessoaJuridica(
                rs.getInt("FK_Pessoa_idPessoa"),
                rs.getString("nome"),
                rs.getString("endereco"),
                rs.getString("cidade"),
                rs.getString("estado"),
                rs.getString("telefone"),
                rs.getString("email"),
                rs.getString("cnpj")
            );
        }
    }
}
return null;
}

```

```

public ArrayList<PessoaJuridica> getPessoas() throws SQLException {
    ArrayList<PessoaJuridica> list = new ArrayList<>();
    String sql = "SELECT pj.FK_Pessoa_idPessoa, pj.cnpj, p.nome, p.endereco, p.cidade, p.estado, p.telefone,
p.email "
+ "FROM PessoaJuridica pj "
+ "INNER JOIN Pessoa p ON pj.FK_Pessoa_idPessoa = p.idPessoa";
    try (Connection con = connector.getConnection();
        PreparedStatement stmt = con.prepareStatement(sql);
        ResultSet rs = stmt.executeQuery()) {
        while (rs.next()) {
            list.add(new PessoaJuridica(
                rs.getInt("FK_Pessoa_idPessoa"),
                rs.getString("nome"),
                rs.getString("endereco"),
                rs.getString("cidade"),
                rs.getString("estado"),
                rs.getString("telefone"),
                rs.getString("email"),
                rs.getString("cnpj")
            ));
        }
    }
    return list;
}

```

```

public int incluir(PessoaJuridica pj) throws SQLException {
    if (pj.getNome() == null || pj.getNome().trim().isEmpty()) {
        throw new IllegalArgumentException("'nome' cannot be empty or null.");
    }
    String sqlInsertPessoa = "INSERT INTO Pessoa(nome, endereco, cidade, estado, telefone, email)
VALUES(?, ?, ?, ?, ?, ?)";
    String sqlInsertPessoaJuridica = "INSERT INTO PessoaJuridica(FK_Pessoa_idPessoa, cnpj) VALUES(?, ?)";

    try (Connection con = connector.getConnection();
        PreparedStatement stmtPessoa =
con.prepareStatement(sqlInsertPessoa, Statement.RETURN_GENERATED_KEYS)) {
        String[] pfArray = {"", pj.getNome(), pj.getEndereco(), pj.getCidade(), pj.getEstado(), pj.getTelefone(),

```

```

pj.getEmail());
    for(int i = 1; i < 7; i++) {
        stmtPessoa.setString(i, pfArray[i]);
    }
    if (stmtPessoa.executeUpdate() != 0) {
        System.out.println("INSERT INTO PessoaJuridica success.");
    }
    else {
        throw new SQLException("Creating user failed, no rows affected.");
    }
    try (ResultSet generatedKeys = stmtPessoa.getGeneratedKeys()) {
        if (generatedKeys.next()) {
            int idNovaPessoa = generatedKeys.getInt(1);
            pj.setId(idNovaPessoa);
            try (PreparedStatement stmtPessoaFisica =
con.prepareStatement(sqlInsertPessoaJuridica,Statement.RETURN_GENERATED_KEYS)) {
                stmtPessoaFisica.setInt(1, idNovaPessoa);
                stmtPessoaFisica.setString(2, pj.getCnpj());
                stmtPessoaFisica.executeUpdate();
            }
            return idNovaPessoa;
        } else {
            throw new SQLException("Creating user failed. No ID obtained.");
        }
    }
}
}
}

```

```

public void alterar(PessoaJuridica pj) throws SQLException {
    String sqlUpdatePessoa = "UPDATE Pessoa SET nome = ?, endereco = ?, cidade = ?, estado = ?, telefone = ?,
email = ? WHERE idPessoa = ?";
    String sqlUpdatePessoaJuridica = "UPDATE PessoaJuridica SET cnpj = ? WHERE FK_Pessoa_idPessoa = ?";
    try (Connection con = connector.getConnection();
        PreparedStatement stmtPessoa =
con.prepareStatement(sqlUpdatePessoa,Statement.RETURN_GENERATED_KEYS);
        PreparedStatement stmtPessoaJuridica =
con.prepareStatement(sqlUpdatePessoaJuridica,Statement.RETURN_GENERATED_KEYS)) {
        String[] pfArray = {"", pj.getNome(), pj.getEndereco(), pj.getCidade(), pj.getEstado(), pj.getTelefone(),
pj.getEmail()};
        for(int i = 1; i < 7; i++) {
            stmtPessoa.setString(i, pfArray[i]);
        }
        stmtPessoa.setInt(7, pj.getId());
        stmtPessoa.executeUpdate();
        stmtPessoaJuridica.setString(1, pj.getCnpj());
        stmtPessoaJuridica.setInt(2, pj.getId());
        stmtPessoaJuridica.executeUpdate();
    }
}
}

```

```

public void excluir(PessoaJuridica pj) throws SQLException {
    String sqlDeletePessoaJuridica = "DELETE FROM PessoaJuridica WHERE FK_Pessoa_idPessoa = ?";
    String sqlDeletePessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";
    try (Connection con = connector.getConnection();
        PreparedStatement stmtPessoaJuridica =
con.prepareStatement(sqlDeletePessoaJuridica,Statement.RETURN_GENERATED_KEYS);
        PreparedStatement stmtPessoa =
con.prepareStatement(sqlDeletePessoa,Statement.RETURN_GENERATED_KEYS)) {
        stmtPessoaJuridica.setInt(1, pj.getId());
        stmtPessoaJuridica.executeUpdate();
        stmtPessoa.setInt(1, pj.getId());
        stmtPessoa.executeUpdate();
    }
}

```

```

    }
}

public void close() throws SQLException {
    connector.close();
}

}

```

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastrbd.model.util;

```

```

import java.sql.*;

```

```

public class ConectorBD {

```

```

    public Connection con;
    public PreparedStatement stmt;
    public ResultSet rs;

```

```

    public Connection getConnection() throws SQLException {
        String url =
"jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=true";
        String user = "sa";
        String password = "loja";
        con = DriverManager.getConnection(url, user, password);
        return con;
    }

```

```

    public ResultSet getSelect(String sql) throws SQLException {
        stmt = getConnection().prepareStatement(sql);
        rs = stmt.executeQuery();
        return rs;
    }

```

```

    public int insert(String sql) throws SQLException {
        stmt = getConnection().prepareStatement(sql, Statement.RETURN_GENERATED_KEYS);
        stmt.executeUpdate();
        rs = stmt.getGeneratedKeys();
        if (rs.next()) {
            return rs.getInt(1);
        } else {
            throw new SQLException("Falha ao inserir dados.");
        }
    }

```

```

    public boolean update(String sql) throws SQLException {
        stmt = getConnection().prepareStatement(sql);
        int affectedRows = stmt.executeUpdate();
        return affectedRows > 0;
    }

```

```

    public void close() throws SQLException {
        if (rs != null && !rs.isClosed()) rs.close();
    }

```

```
        if (stmt != null && !stmt.isClosed()) stmt.close();
        if (con != null && !con.isClosed()) con.close();
    }
}
```

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastrbd.model.util;

import java.sql.ResultSet;
import java.sql.SQLException;

public class SequenceManager {

    ConectorBD cnx = new ConectorBD();

    public int getValue(String sequenceName) throws SQLException {
        String sql = "SELECT NEXT VALUE FOR " + sequenceName;
        ResultSet rs = cnx.getSelect(sql);
        if (rs.next()) {
            return rs.getInt(1);
        } else {
            throw new SQLException("Não foi possível obter o próximo valor da sequência: " + sequenceName);
        }
    }
}
```