Estácio

RPG0016 - BackEnd sem banco não tem

202303832338 - Rafael Rosendo Tagliaferro

Campos Interlagos
Nível 3: BackEnd sem banco não tem? – 01/2024

Objetivo da Prática

Descreva nessa seção qual o objetivo da sua prática. Todos os Relatórios de Práticas deverão ser confeccionados em arquivo no formato PDF, com a **Logo da Universidade**, **nome do Campus**, **nome do Curso**, **nome da Disciplina**, **número da Turma**, **semestre letivo**. Além disso, o projeto deve ser armazenado em um repositório no GIT e o respectivo endereço deve constar na documentação e essa documentação deve estar no no GIT. O código deve estar versionado no GIT de forma organizada.

Lembre-se que a organização contará pontos.

Esse template é um modelo a ser seguido. O aluno pode optar por seguir outro modelo, **desde que atenda a todas as etapas disponíveis na Missão Prática**. O documento final deve estar em pdf.

1º Procedimento | Mapeamento Objeto-Relacional e DAO

Inserir neste campo, **de forma organizada**, todos os códigos do roteiro do 1º Procedimento da Atividade Prática, os resultados da execução do código e a Análise e Conclusão:

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

A mais importante seria a abstração que a API oferece, pois facilmente um desenvolvedor consegue rapidamente, executar operações no banco de dados. Alem dela gerenciar a conexão com o banco e otimizar recurso.

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

O Statement serve para executar operação que não estejam parametrizadas, ja o PreparedStament permite a parametrização.

c) **Como o padrão DAO melhora a manutenibilidade do software?**

Oferece uma maneira separada de lidar com as conexão, onde em um lugar fica a logica para o gerenciamos das conexão e em outro a logica para lidar com o negocio.

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

O banco de dados precisa estar configurado da mesma forma, para que seus objetos e classes, consigam facilmente se relacionar com as tabelas, desta forma, fica simples a implementação das classes pessoas, pessoasjuridicas e pessoasfisicias.

2º Procedimento | Alimentando a Base

Inserir neste campo, **de forma organizada**, todos os códigos do roteiro do 2º Procedimento da Atividade Prática, os resultados da execução do código e a Análise e Conclusão:

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

A persistencia em arquivo, fica limitado a todo armazenamento ficando em um arquivo simples, o que dificulta em muito a leitura e escrita, alem de dificultar querys mais sofisticadas.

A Persistencia em banco, permite justamente multiplas conexão, alem de um gerencimento em memoria e outros mecanimos como backup, logs, procedures, views, chaves etc.

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

Ficou mais simples e pratico, para fazer operações diretamente no objeto por uso do linq, ou apenas retornar um objeto simples, dando mais enfase ao objeto e a necessidade, e não precisamos mais criar uma query para cada consulta especifica que precisamos fazer.

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

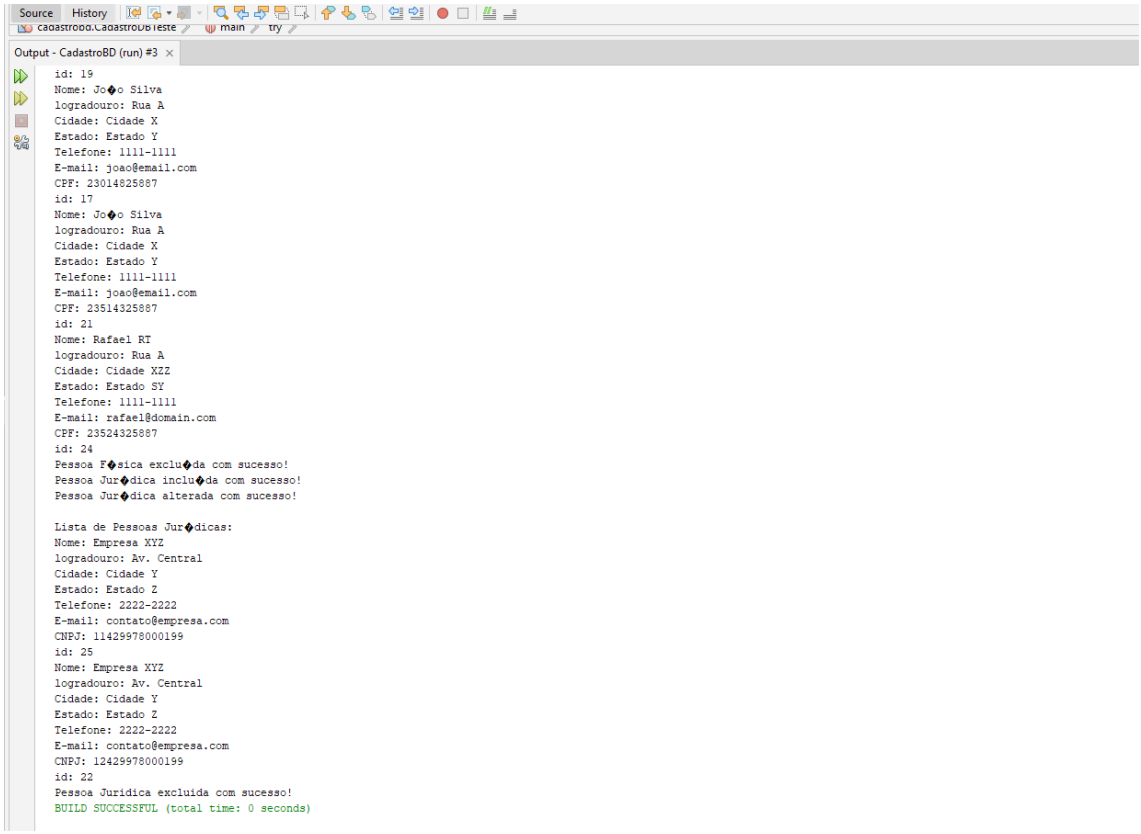
È uma caracterisca do JAVA, dado que a main é sempre static, os objetos dentro dessa classe precisam estar definidos como static

Conclusão

A Implementação desde codigo em JAVA me mostrou o quanto o JAVA evoluiu, pois não é a minha linguagem que estou naturalmente acostumado a desenvolver (.net).

A implementação de conceitos como herança e polimorfismo é bem simples, assim como no C#, bem como a tratativa em manipular os bancos de dados e o uso de lambdas

PRINT EXECUÇÃO ENTREGA 1 :



PRINT EXECUÇÃO ENTREGA 2 :

1 – Codigo – Atividade 1: Classe de Teste

```
package cadastrobd;

import cadastrodb.model.PessoaFisica;
import cadastrodb.model.PessoaFisicaDAO;
import cadastrodb.model.PessoaJuridica;
import cadastrodb.model.PessoaJuridicaDAO;
import java.sql.SQLException;
import java.util.List;

/**
 *
 * @author rafae
 */
public class CadastroDBTeste {

    public static void main(String[] args) {
        try {
            PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
            PessoaFisica pessoaFisica = new PessoaFisica(0, "Rafael RT", "Rua A", "Cidade XZZ", "Estado SY", "1111-1111", "rafael@domain.com", "23514325887");
```

```
pessoaFisicaDAO.incluir(pessoaFisica);
System.out.println("Pessoa Física incluída com sucesso!");

pessoaFisica.setNome("Rafael Silva 2");
pessoaFisica.setEmail("rafael@domain.com.br");
pessoaFisicaDAO.alterar(pessoaFisica);
System.out.println("Pessoa Física alterada com sucesso!");

List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();
System.out.println("\nLista de Pessoas Físicas:");
for (PessoaFisica pf : pessoasFisicas) {
    System.out.println("Nome: " + pf.getNome());
    System.out.println("logradouro: " + pf.getLogradouro());
    System.out.println("Cidade: " + pf.getCidade());
    System.out.println("Estado: " + pf.getEstado());
    System.out.println("Telefone: " + pf.getTelefone());
    System.out.println("E-mail: " + pf.getEmail());
    System.out.println("CPF: " + pf.getCpf());
    System.out.println("id: " + pf.getId());
}

pessoaFisicaDAO.excluir(pessoaFisica.getId());
System.out.println("Pessoa Física excluída com sucesso!");

PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();
```

```
        PessoaJuridica pessoaJuridica = new PessoaJuridica(0, "Empresa XYZ", "Av. Central", "Cidade Y", "Estado Z", "2222-2222", "contato@empresa.com", "12429978000199");
        pessoaJuridicaDAO.incluir(pessoaJuridica);
        System.out.println("Pessoa Jurídica incluída com sucesso!");

        pessoaJuridica.setNome("Empresa XYZ Ltda");
        pessoaJuridica.setEmail("contato@xyz.com");
        pessoaJuridicaDAO.alterar(pessoaJuridica);
        System.out.println("Pessoa Jurídica alterada com sucesso!");

        List<PessoaJuridica> pessoasJuridicas = pessoaJuridicaDAO.getPessoas();
        System.out.println("\nLista de Pessoas Jurídicas:");
        for (PessoaJuridica pj : pessoasJuridicas) {
            System.out.println("Nome: " + pj.getNome());
            System.out.println("logradouro: " + pj.getLogradouro());
            System.out.println("Cidade: " + pj.getCidade());
            System.out.println("Estado: " + pj.getEstado());
            System.out.println("Telefone: " + pj.getTelefone());
            System.out.println("E-mail: " + pj.getEmail());
            System.out.println("CNPJ: " + pj.getCnpj());
            System.out.println("id: " + pj.getId());
        }

        pessoaJuridicaDAO.excluir(pessoaJuridica.getId());
```

```
            System.out.println("Pessoa Juridica excluida com sucesso!");


        } catch (SQLException e) {
            System.err.println("Erro: " + e.getMessage());
        }
    }
}
```

2 – Codigo Main – Entrega 2

```
/*
 * 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 cadastrobd;

import cadastrodb.model.PessoaFisica;
import cadastrodb.model.PessoaFisicaDAO;
import cadastrodb.model.PessoaJuridica;
import cadastrodb.model.PessoaJuridicaDAO;
import java.util.List;
import java.util.Scanner;
import java.sql.SQLException;
```

```
/**
 *
 * @author rafae
 */
public class CadastroBD {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();

        try {
            int opcao;
            do {
                System.out.println("\n--- Menu de Opcoes ---");
                System.out.println("1. Incluir");
                System.out.println("2. Alterar");
                System.out.println("3. Excluir");
                System.out.println("4. Exibir pelo ID");
                System.out.println("5. Exibir todos");
                System.out.println("0. Sair");
                System.out.print("Escolha uma opcao: ");
                opcao = scanner.nextInt();
                scanner.nextLine();
```

```
            switch (opcao) {
                case 1:
                    incluir(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                    break;
                case 2:
                    alterar(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                    break;
                case 3:
                    excluir(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                    break;
                case 4:
                    exibirPorId(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                    break;
                case 5:
                    exibirTodos(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                    break;
                case 0:
                    System.out.println("Encerrando o sistema...");
                    break;
                default:
                    System.out.println("Opção inválida. Tente novamente.");
            }
        } while (opcao != 0);

    } catch (SQLException e) {
```

```
            System.err.println("Erro de SQL: " + e.getMessage());
        } catch (Exception e) {
            System.err.println("Erro: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
```

```
    private static void incluir(Scanner scanner, PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) throws SQLException {
        System.out.println("Incluir Pessoa (1. Fisica, 2. Juridica): ");
        int tipo = scanner.nextInt();
        scanner.nextLine();

        if (tipo == 1) {
            System.out.println("Inserir nova Pessoa Fisica");
            PessoaFisica pf = capturarPessoaFisica(scanner);
            pessoaFisicaDAO.incluir(pf);
            System.out.println("Pessoa Fisica incluída com sucesso!");
        } else if (tipo == 2) {
            System.out.println("Inserir nova Pessoa Juridica");
            PessoaJuridica pj = capturarPessoaJuridica(scanner);
            pessoaJuridicaDAO.incluir(pj);
            System.out.println("Pessoa Juridica incluída com sucesso!");
        } else {
```

```
            System.out.println("Tipo inválido!");
        }
    }

    private static void alterar(Scanner scanner, PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) throws SQLException {
        System.out.println("Alterar Pessoa (1. Fisica, 2. Juridica): ");
        int tipo = scanner.nextInt();
        scanner.nextLine();

        if (tipo == 1) {
            System.out.print("Informe o ID da Pessoa Fisica: ");
            int id = scanner.nextInt();
            scanner.nextLine();
            PessoaFisica pf = pessoaFisicaDAO.getPessoa(id);
            if (pf != null) {
                System.out.println("Dados atuais: " + pf);
                System.out.println("Digite os novos dados:");
                pf = capturarPessoaFisica(scanner, pf);
                pessoaFisicaDAO.alterar(pf);
                System.out.println("Pessoa Fisica alterada com sucesso!");
            } else {
                System.out.println("Pessoa Fisica nao encontrada.");
            }
        } else if (tipo == 2) {
```

```
            System.out.print("Informe o ID da Pessoa Juridica: ");
            int id = scanner.nextInt();
            scanner.nextLine();
            PessoaJuridica pj = pessoaJuridicaDAO.getPessoa(id);
            if (pj != null) {
                System.out.println("Dados atuais: " + pj);
                System.out.println("Digite os novos dados:");
                pj = capturarPessoaJuridica(scanner, pj);
                pessoaJuridicaDAO.alterar(pj);
                System.out.println("Pessoa Juridica alterada com sucesso!");
            } else {
                System.out.println("Pessoa Juridica nao encontrada.");
            }
        } else {
            System.out.println("Tipo inválido!");
        }
    }
```

```
    private static void excluir(Scanner scanner, PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) throws SQLException {
        System.out.println("Excluir Pessoa (1. Fisica, 2. Juridica): ");
        int tipo = scanner.nextInt();
        scanner.nextLine();

        if (tipo == 1) {
```

```
            System.out.print("Informe o ID da Pessoa Fisica a ser excluída: ");
            int id = scanner.nextInt();
            pessoaFisicaDAO.excluir(id);
            System.out.println("Pessoa Fisica excluída com sucesso!");
        } else if (tipo == 2) {
            System.out.print("Informe o ID da Pessoa Juridica a ser excluída: ");
            int id = scanner.nextInt();
            pessoaJuridicaDAO.excluir(id);
            System.out.println("Pessoa Juridica excluída com sucesso!");
        } else {
            System.out.println("Tipo inválido!");
        }
    }

    private static void exibirPorId(Scanner scanner, PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) throws SQLException {
        System.out.println("Exibir Pessoa por ID (1. Fisica, 2. Juridica): ");
        int tipo = scanner.nextInt();
        scanner.nextLine();

        if (tipo == 1) {
            System.out.print("Informe o ID da Pessoa Fisica: ");
            int id = scanner.nextInt();
            PessoaFisica pf = pessoaFisicaDAO.getPessoa(id);
            if (pf != null) {
```

```
            System.out.println("Pessoa Fisica: " + pf);
        } else {
            System.out.println("Pessoa Fisica nao encontrada.");
        }
    } else if (tipo == 2) {
        System.out.print("Informe o ID da Pessoa Juridica: ");
        int id = scanner.nextInt();
        PessoaJuridica pj = pessoaJuridicaDAO.getPessoa(id);
        if (pj != null) {
            System.out.println("Pessoa Juridica: " + pj);
        } else {
            System.out.println("Pessoa Juridica nao encontrada.");
        }
    } else {
        System.out.println("Tipo invalido!");
    }
}

    private static void exibirTodos(Scanner scanner, PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) throws SQLException {
    System.out.println("Exibir todas as Pessoas (1. Fisica, 2. Juridica): ");
    int tipo = scanner.nextInt();
    scanner.nextLine();

    if (tipo == 1) {
```

```
            List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();
            System.out.println("Lista de Pessoas Fisicas:");
            for (PessoaFisica pf : pessoasFisicas) {
                System.out.println("Nome: " + pf.getNome());
                System.out.println("logradouro: " + pf.getLogradouro());
                System.out.println("Cidade: " + pf.getCidade());
                System.out.println("Estado: " + pf.getEstado());
                System.out.println("Telefone: " + pf.getTelefone());
                System.out.println("E-mail: " + pf.getEmail());
                System.out.println("CPF: " + pf.getCpf());
                System.out.println("id: " + pf.getId());
            }
        } else if (tipo == 2) {
                                        List<PessoaJuridica>  pessoasJuridicas  =
pessoaJuridicaDAO.getPessoas();
            System.out.println("Lista de Pessoas Juridicas:");
            for (PessoaJuridica pj : pessoasJuridicas) {
                System.out.println("Nome: " + pj.getNome());
                System.out.println("logradouro: " + pj.getLogradouro());
                System.out.println("Cidade: " + pj.getCidade());
                System.out.println("Estado: " + pj.getEstado());
                System.out.println("Telefone: " + pj.getTelefone());
                System.out.println("E-mail: " + pj.getEmail());
                System.out.println("CNPJ: " + pj.getCnpj());
                System.out.println("id: " + pj.getId());
```

```
            }
        } else {
            System.out.println("Tipo inválido!");
        }
    }

    private static PessoaFisica capturarPessoaFisica(Scanner scanner) {
        System.out.print("Nome: ");
        String nome = scanner.nextLine();
        System.out.print("Logradouro: ");
        String logradouro = scanner.nextLine();
        System.out.print("Cidade: ");
        String cidade = scanner.nextLine();
        System.out.print("Estado: ");
        String estado = scanner.nextLine();
        System.out.print("Telefone: ");
        String telefone = scanner.nextLine();
        System.out.print("Email: ");
        String email = scanner.nextLine();
        System.out.print("CPF: ");
        String cpf = scanner.nextLine();
            return new PessoaFisica(0, nome, logradouro, cidade, estado, telefone,
email, cpf);
    }
```

```
    private static PessoaJuridica capturarPessoaJuridica(Scanner scanner) {
        System.out.print("Nome: ");
        String nome = scanner.nextLine();
        System.out.print("Logradouro: ");
        String logradouro = scanner.nextLine();
        System.out.print("Cidade: ");
        String cidade = scanner.nextLine();
        System.out.print("Estado: ");
        String estado = scanner.nextLine();
        System.out.print("Telefone: ");
        String telefone = scanner.nextLine();
        System.out.print("Email: ");
        String email = scanner.nextLine();
        System.out.print("CNPJ: ");
        String cnpj = scanner.nextLine();
         return new PessoaJuridica(0, nome, logradouro, cidade, estado, telefone,
email, cnpj);
    }


          private static PessoaFisica capturarPessoaFisica(Scanner scanner,
PessoaFisica pf) {
        System.out.print("Nome (" + pf.getNome() + "): ");
        pf.setNome(scanner.nextLine());
        System.out.print("Logradouro (" + pf.getLogradouro() + "): ");
        pf.setLogradouro(scanner.nextLine());
        System.out.print("Cidade (" + pf.getCidade() + "): ");
```

```
        pf.setCidade(scanner.nextLine());
        System.out.print("Estado (" + pf.getEstado() + "): ");
        pf.setEstado(scanner.nextLine());
        System.out.print("Telefone (" + pf.getTelefone() + "): ");
        pf.setTelefone(scanner.nextLine());
        System.out.print("Email (" + pf.getEmail() + "): ");
        pf.setEmail(scanner.nextLine());
        System.out.print("CPF (" + pf.getCpf() + "): ");
        pf.setCpf(scanner.nextLine());
        return pf;
    }

    private static PessoaJuridica capturarPessoaJuridica(Scanner scanner,
PessoaJuridica pj) {
        System.out.print("Nome (" + pj.getNome() + "): ");
        pj.setNome(scanner.nextLine());
        System.out.print("Logradouro (" + pj.getLogradouro() + "): ");
        pj.setLogradouro(scanner.nextLine());
        System.out.print("Cidade (" + pj.getCidade() + "): ");
        pj.setCidade(scanner.nextLine());
        System.out.print("Estado (" + pj.getEstado() + "): ");
        pj.setEstado(scanner.nextLine());
        System.out.print("Telefone (" + pj.getTelefone() + "): ");
        pj.setTelefone(scanner.nextLine());
        System.out.print("Email (" + pj.getEmail() + "): ");
```

```
        pj.setEmail(scanner.nextLine());
        System.out.print("CNPJ (" + pj.getCnpj() + "): ");
        pj.setCnpj(scanner.nextLine());
        return pj;
    }

}
```

PessoaJuridicaDAO.java

```
/*
 * 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 cadastrodb.model;

import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author rafae
```

```
 */

public class PessoaJuridicaDAO {


    public PessoaJuridica getPessoa(int id) throws SQLException {
        PessoaJuridica pessoaJuridica = null;
          String sql = "SELECT * FROM Pessoa p JOIN PessoasJuridicas pj ON p.pessoaid = pj.pessoaid WHERE p.pessoaid  = ?";


        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;


        try {
            conn = ConectorBD.getConnection();
            stmt = ConectorBD.getPrepared(conn, sql);
            stmt.setInt(1, id);
            rs = ConectorBD.getSelect(stmt);


            if (rs.next()) {
                pessoaJuridica = new PessoaJuridica(
                    rs.getInt("pessoaid"),
                    rs.getString("nome"),
                    rs.getString("logradouro"),
                    rs.getString("cidade"),
```

```
                rs.getString("estado"),
                rs.getString("telefone"),
                rs.getString("email"),
                rs.getString("cnpj")
            );
        }
    } finally {
        ConectorBD.close(rs);
        ConectorBD.close(stmt);
        ConectorBD.close(conn);
    }

    return pessoaJuridica;
}

public List<PessoaJuridica> getPessoas() throws SQLException {
    List<PessoaJuridica> pessoas = new ArrayList<>();
      String sql = "SELECT * FROM Pessoa p JOIN PessoasJuridicas pj ON
p.pessoaid = pj.pessoaid";

    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;

    try {
```

```
            conn = ConectorBD.getConnection();
            stmt = ConectorBD.getPrepared(conn, sql);
            rs = ConectorBD.getSelect(stmt);

            while (rs.next()) {
                PessoaJuridica pessoaJuridica = new PessoaJuridica(
                    rs.getInt("pessoaid"),
                    rs.getString("nome"),
                    rs.getString("logradouro"),
                    rs.getString("cidade"),
                    rs.getString("estado"),
                    rs.getString("telefone"),
                    rs.getString("email"),
                    rs.getString("cnpj")
                );
                pessoas.add(pessoaJuridica);
            }
        } finally {
            ConectorBD.close(rs);
            ConectorBD.close(stmt);
            ConectorBD.close(conn);
        }

        return pessoas;
    }
```

```
public void incluir(PessoaJuridica pessoaJuridica) throws SQLException {
    String sqlPessoa = "INSERT INTO Pessoa (pessoaid, nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
    String sqlPessoaJuridica = "INSERT INTO PessoasJuridicas (pessoaid, cnpj) VALUES (?, ?)";

    Connection conn = null;
    PreparedStatement stmtPessoa = null;
    PreparedStatement stmtPessoaJuridica = null;

    try {
        conn = ConectorBD.getConnection();
        conn.setAutoCommit(false);

        int novoId = (int) SequenceManager.getValue("Seq_PessoaID");

        stmtPessoa = ConectorBD.getPrepared(conn, sqlPessoa);
        stmtPessoa.setInt(1, novoId);
        stmtPessoa.setString(2, pessoaJuridica.getNome());
        stmtPessoa.setString(3, pessoaJuridica.getLogradouro());
        stmtPessoa.setString(4, pessoaJuridica.getCidade());
        stmtPessoa.setString(5, pessoaJuridica.getEstado());
        stmtPessoa.setString(6, pessoaJuridica.getTelefone());
        stmtPessoa.setString(7, pessoaJuridica.getEmail());
        stmtPessoa.executeUpdate();
```

```
                stmtPessoaJuridica = ConectorBD.getPrepared(conn,
sqlPessoaJuridica);
            stmtPessoaJuridica.setInt(1, novoId);
            stmtPessoaJuridica.setString(2, pessoaJuridica.getCnpj());
            stmtPessoaJuridica.executeUpdate();

            conn.commit();

        } catch (SQLException e) {
            if (conn != null) conn.rollback();
            throw e;
        } finally {
            ConectorBD.close(stmtPessoaJuridica);
            ConectorBD.close(stmtPessoa);
            ConectorBD.close(conn);
        }
    }

    public void alterar(PessoaJuridica pessoaJuridica) throws SQLException {
        String sqlPessoa = "UPDATE Pessoa SET nome = ?, logradouro = ?,
cidade = ?, estado = ?, telefone = ?, email = ? WHERE pessoaid = ?";
        String sqlPessoaJuridica = "UPDATE PessoasJuridicas SET cnpj = ?
WHERE pessoaid = ?";

        Connection conn = null;
```

```
        PreparedStatement stmtPessoa = null;
        PreparedStatement stmtPessoaJuridica = null;

        try {
            conn = ConectorBD.getConnection();
            conn.setAutoCommit(false);

            stmtPessoa = ConectorBD.getPrepared(conn, sqlPessoa);
            stmtPessoa.setString(1, pessoaJuridica.getNome());
            stmtPessoa.setString(2, pessoaJuridica.getLogradouro());
            stmtPessoa.setString(3, pessoaJuridica.getCidade());
            stmtPessoa.setString(4, pessoaJuridica.getEstado());
            stmtPessoa.setString(5, pessoaJuridica.getTelefone());
            stmtPessoa.setString(6, pessoaJuridica.getEmail());
            stmtPessoa.setInt(7, pessoaJuridica.getId());
            stmtPessoa.executeUpdate();

                        stmtPessoaJuridica = ConectorBD.getPrepared(conn,
sqlPessoaJuridica);
            stmtPessoaJuridica.setString(1, pessoaJuridica.getCnpj());
            stmtPessoaJuridica.setInt(2, pessoaJuridica.getId());
            stmtPessoaJuridica.executeUpdate();

            conn.commit();
```

```
        } catch (SQLException e) {
            if (conn != null) conn.rollback();
            throw e;
        } finally {
            ConectorBD.close(stmtPessoaJuridica);
            ConectorBD.close(stmtPessoa);
            ConectorBD.close(conn);
        }
    }

    public void excluir(int id) throws SQLException {
        String sqlPessoaJuridica = "DELETE FROM PessoasJuridicas WHERE pessoaid = ?";
        String sqlPessoa = "DELETE FROM Pessoa WHERE pessoaid = ?";

        Connection conn = null;
        PreparedStatement stmtPessoaJuridica = null;
        PreparedStatement stmtPessoa = null;

        try {
            conn = ConectorBD.getConnection();
            conn.setAutoCommit(false);

            stmtPessoaJuridica = ConectorBD.getPrepared(conn, sqlPessoaJuridica);
            stmtPessoaJuridica.setInt(1, id);
```

```
            stmtPessoaJuridica.executeUpdate();


            stmtPessoa = ConectorBD.getPrepared(conn, sqlPessoa);
            stmtPessoa.setInt(1, id);
            stmtPessoa.executeUpdate();


            conn.commit();


        } catch (SQLException e) {
            if (conn != null) conn.rollback();
            throw e;
        } finally {
            ConectorBD.close(stmtPessoaJuridica);
            ConectorBD.close(stmtPessoa);
            ConectorBD.close(conn);
        }
    }
}
```

PessoaFisicaDAO.java

```
/*
 * 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 cadastrodb.model;


import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;


/**
 *
 * @author rafae
 */



public class PessoaFisicaDAO {


    public PessoaFisica getPessoa(int id) throws SQLException {
        PessoaFisica pessoaFisica = null;
            String sql = "SELECT * FROM Pessoa p JOIN PessoasFisicas pf ON
p.pessoaid = pf.pessoaid WHERE p.pessoaid = ?";


        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
```

```
try {
    conn = ConectorBD.getConnection();
    stmt = ConectorBD.getPrepared(conn, sql);
    stmt.setInt(1, id);
    rs = ConectorBD.getSelect(stmt);

    if (rs.next()) {
        pessoaFisica = new PessoaFisica(
            rs.getInt("pessoaid"),
            rs.getString("nome"),
            rs.getString("logradouro"),
            rs.getString("cidade"),
            rs.getString("estado"),
            rs.getString("telefone"),
            rs.getString("email"),
            rs.getString("cpf")
        );
    }
} finally {
    ConectorBD.close(rs);
    ConectorBD.close(stmt);
    ConectorBD.close(conn);
}

return pessoaFisica;
```

```
    }

    public List<PessoaFisica> getPessoas() throws SQLException {
        List<PessoaFisica> pessoas = new ArrayList<>();
            String sql = "SELECT * FROM Pessoa p JOIN PessoasFisicas pf ON
p.pessoaid = pf.pessoaid";

        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;

        try {
            conn = ConectorBD.getConnection();
            stmt = ConectorBD.getPrepared(conn, sql);
            rs = ConectorBD.getSelect(stmt);

            while (rs.next()) {
                PessoaFisica pessoaFisica = new PessoaFisica(
                    rs.getInt("pessoaid"),
                    rs.getString("nome"),
                    rs.getString("logradouro"),
                    rs.getString("cidade"),
                    rs.getString("estado"),
                    rs.getString("telefone"),
                    rs.getString("email"),
```

```
                rs.getString("cpf")
            );
            pessoas.add(pessoaFisica);
        }
    } finally {
        ConectorBD.close(rs);
        ConectorBD.close(stmt);
        ConectorBD.close(conn);
    }

    return pessoas;
}

public void incluir(PessoaFisica pessoaFisica) throws SQLException {
    String sqlPessoa = "INSERT INTO Pessoa (pessoaid, nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
    String sqlPessoaFisica = "INSERT INTO PessoasFisicas (pessoaid, cpf) VALUES (?, ?)";

    Connection conn = null;
    PreparedStatement stmtPessoa = null;
    PreparedStatement stmtPessoaFisica = null;

    try {
        conn = ConectorBD.getConnection();
        conn.setAutoCommit(false);
```

```
    int novoId = (int) SequenceManager.getValue("Seq_PessoaID");

    stmtPessoa = ConectorBD.getPrepared(conn, sqlPessoa);
    stmtPessoa.setInt(1, novoId);
    stmtPessoa.setString(2, pessoaFisica.getNome());
    stmtPessoa.setString(3, pessoaFisica.getLogradouro());
    stmtPessoa.setString(4, pessoaFisica.getCidade());
    stmtPessoa.setString(5, pessoaFisica.getEstado());
    stmtPessoa.setString(6, pessoaFisica.getTelefone());
    stmtPessoa.setString(7, pessoaFisica.getEmail());
    stmtPessoa.executeUpdate();

    stmtPessoaFisica = ConectorBD.getPrepared(conn, sqlPessoaFisica);
    stmtPessoaFisica.setInt(1, novoId);
    stmtPessoaFisica.setString(2, pessoaFisica.getCpf());
    stmtPessoaFisica.executeUpdate();

    conn.commit();

} catch (SQLException e) {
    if (conn != null) conn.rollback();
    throw e;
} finally {
    ConectorBD.close(stmtPessoaFisica);
```

```
            ConectorBD.close(stmtPessoa);
            ConectorBD.close(conn);
        }
    }

    public void alterar(PessoaFisica pessoaFisica) throws SQLException {
        String sqlPessoa = "UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE pessoaid = ?";
        String sqlPessoaFisica = "UPDATE PessoasFisicas SET cpf = ? WHERE pessoaid = ?";

        Connection conn = null;
        PreparedStatement stmtPessoa = null;
        PreparedStatement stmtPessoaFisica = null;

        try {
            conn = ConectorBD.getConnection();
            conn.setAutoCommit(false);

            stmtPessoa = ConectorBD.getPrepared(conn, sqlPessoa);
            stmtPessoa.setString(1, pessoaFisica.getNome());
            stmtPessoa.setString(2, pessoaFisica.getLogradouro());
            stmtPessoa.setString(3, pessoaFisica.getCidade());
            stmtPessoa.setString(4, pessoaFisica.getEstado());
            stmtPessoa.setString(5, pessoaFisica.getTelefone());
            stmtPessoa.setString(6, pessoaFisica.getEmail());
```

```
            stmtPessoa.setInt(7, pessoaFisica.getId());
            stmtPessoa.executeUpdate();

            stmtPessoaFisica = ConectorBD.getPrepared(conn, sqlPessoaFisica);
            stmtPessoaFisica.setString(1, pessoaFisica.getCpf());
            stmtPessoaFisica.setInt(2, pessoaFisica.getId());
            stmtPessoaFisica.executeUpdate();

            conn.commit();

        } catch (SQLException e) {
            if (conn != null) conn.rollback();
            throw e;
        } finally {
            ConectorBD.close(stmtPessoaFisica);
            ConectorBD.close(stmtPessoa);
            ConectorBD.close(conn);
        }
    }

    public void excluir(int id) throws SQLException {
            String sqlPessoaFisica = "DELETE FROM PessoasFisicas WHERE
pessoaid = ?";
        String sqlPessoa = "DELETE FROM Pessoa WHERE pessoaid = ?";
```

```
Connection conn = null;
PreparedStatement stmtPessoaFisica = null;
PreparedStatement stmtPessoa = null;

try {
    conn = ConectorBD.getConnection();
    conn.setAutoCommit(false);

    stmtPessoaFisica = ConectorBD.getPrepared(conn, sqlPessoaFisica);
    stmtPessoaFisica.setInt(1, id);
    stmtPessoaFisica.executeUpdate();

    stmtPessoa = ConectorBD.getPrepared(conn, sqlPessoa);
    stmtPessoa.setInt(1, id);
    stmtPessoa.executeUpdate();

    conn.commit();

} catch (SQLException e) {
    if (conn != null) conn.rollback();
    throw e;
} finally {
    ConectorBD.close(stmtPessoaFisica);
    ConectorBD.close(stmtPessoa);
    ConectorBD.close(conn);
```

```
        }
    }
}
```

PessoaFisica.java

```
/*
 * 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 cadastrodb.model;

/**
 *
 * @author rafae
 */
public class PessoaFisica extends Pessoa {
    private String cpf;

    // Construtor padrão
    public PessoaFisica() {
        super();
    }
      public PessoaFisica(int id, String nome, String logradouro, String cidade, String estado, String telefone, String email, String cpf) {
```

```
        super(id, nome, logradouro, cidade, estado, telefone, email);
        this.cpf = cpf;
    }
    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CPF: " + cpf);
    }

    public String getCpf()
    {
        return cpf;
    }

    public void setCpf(String doc)
    {
        cpf = doc;
    }
}
```

PessoaJuridica.java

```
/*
 * 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 cadastrodb.model;

/**
 *
 * @author rafae
 */
public class PessoaJuridica extends Pessoa {
    private String cnpj;

    public PessoaJuridica() {
        super();
    }

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

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
```

```
    }

    public String getCnpj()
    {
        return cnpj;
    }

    public void setCnpj(String doc)
    {
        cnpj = doc;
    }
}
```

Pessoa.java

```
 /*
  * 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 cadastrodb.model;

/**
 *
 * @author rafae
```

```
 */


public class Pessoa {
    private int id;
    private String nome;
    private String logradouro;
    private String cidade;
    private String estado;
    private String telefone;
    private String email;

    public Pessoa() {
    }

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

```
public void exibir() {
    System.out.println("ID: " + id);
    System.out.println("Nome: " + nome);
    System.out.println("Logradouro: " + logradouro);
    System.out.println("Cidade: " + cidade);
    System.out.println("Estado: " + estado);
    System.out.println("Telefone: " + telefone);
    System.out.println("Email: " + email);
}

public int getId()
{
    return id;
}

public String getNome()
{
    return nome;
}

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

```
public String getLogradouro()
{
    return logradouro;
}

public void setLogradouro(String logradouro)
{
     logradouro = this.logradouro;
}

public String getCidade()
{
    return cidade;
}

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

public String getEstado()
{
    return estado;
}
```

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

    public String getTelefone()
    {
        return telefone;
    }

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

    public String getEmail()
    {
        return email;
    }

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

SequenceManager.java

```
/*
 * 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 cadastro.model.util;

import java.sql.*;

/**
 *
 * @author rafae
 */

public class SequenceManager {

    public static long getValue(String sequenceName) throws SQLException {
        long sequenceValue = 0;
        String sql = "SELECT next value for " + sequenceName;
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
```

```
        try {
            conn = ConectorBD.getConnection();
            stmt = ConectorBD.getPrepared(conn, sql);
            rs = ConectorBD.getSelect(stmt);
            if (rs.next()) {
                sequenceValue = rs.getLong(1);
            }
        } finally {
            ConectorBD.close(rs);
            ConectorBD.close(stmt);
            ConectorBD.close(conn);
        }

        return sequenceValue;
    }
}
```

ConectorBD.java

```
/*
 * 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 cadastro.model.util;
```

```
import java.sql.*;
/**
 *
 * @author rafae
 */

public class ConectorBD {

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

    public static PreparedStatement getPrepared(Connection conn, String sql)
throws SQLException {
        return conn.prepareStatement(sql);
    }

    public static ResultSet getSelect(PreparedStatement stmt) throws
SQLException {
        return stmt.executeQuery();
    }
```

```
public static void close(Statement stmt) {
    if (stmt != null) {
        try {
            stmt.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

public static void close(ResultSet rs) {
    if (rs != null) {
        try {
            rs.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

public static void close(Connection conn) {
    if (conn != null) {
        try {
            conn.close();
```

```
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```