Primeira etapa

Após ter sido criado o pacote “cadastrobd.model” no NetBeans:

1) Classe Pessoa:

```
package cadastrobd.model;


public class Pessoa {

    private int id;

    private String nome;

    private String logradouro;

    private String cidade;

    private String estado;

    private String telefone;

    private String email;


    // Construtor padrão

    public Pessoa() {

    }


    // Construtor completo

    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;
    }

    // Método para exibir os dados no console
    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);
    }
}
```

2) Classe Pessoa Física:

```
package cadastrobd.model;

public class PessoaFisica extends Pessoa {
    private String cpf;

    // Construtor padrão
    public PessoaFisica() {
        super();
    }

    // Construtor completo
```

```
    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;

    }



    // Sobrescrita do método exibir para incluir o CPF

    @Override

    public void exibir() {

        super.exibir(); // Chama o método exibir da superclasse

        System.out.println("CPF: " + cpf);

    }

}
```

3) Classe Pessoa Jurídica:

```
package cadastrobd.model;



public class PessoaJuridica extends Pessoa {

    private String cnpj;



    // Construtor padrão

    public PessoaJuridica() {

        super();

    }



    // Construtor completo

    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;
    }

    // Sobrescrita do método exibir para incluir o CNPJ
    @Override
    public void exibir() {
        super.exibir(); // Chama o método exibir da superclasse
        System.out.println("CNPJ: " + cnpj);
    }
}
```

Após ter sido criado o pacote “cadastro.model.util”, para inclusão das classes utilitárias:

Para implementar as classes no padrão DAO (Data Access Object) no pacote "cadastro.model" e as classes utilitárias no pacote "cadastro.model.util", siga as instruções abaixo:

Pacote "cadastro.model.util":

Classe ConectorBD:

```java
package cadastro.model.util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
```

```
import java.sql.SQLException;

import java.sql.Statement;



public class ConectorBD {

    private static final String URL =
"jdbc:sqlserver://localhost:1433;databaseName=nomedobanco";

    private static final String USUARIO = "seuUsuario";

    private static final String SENHA = "suaSenha";



    // Método para obter uma conexão com o banco de dados

    public static Connection getConnection() throws SQLException {

        return DriverManager.getConnection(URL, USUARIO, SENHA);

    }



    // Método para obter um PreparedStatement

    public static PreparedStatement getPrepared(String sql) throws
SQLException {

        return getConnection().prepareStatement(sql);

    }



    // Método para executar uma consulta SQL e retornar um ResultSet

    public static ResultSet getSelect(String sql) throws SQLException
{

        return getPrepared(sql).executeQuery();

    }



    // Métodos para fechar objetos de acesso ao banco de dados

    public static void close(Connection con) throws SQLException {

        if (con != null) con.close();
```

```
    }

    public static void close(Statement stmt) throws SQLException {
        if (stmt != null) stmt.close();
    }

    public static void close(ResultSet rs) throws SQLException {
        if (rs != null) rs.close();
    }
}
```

Classe SequenceManager:

```java
package cadastro.model.util;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class SequenceManager {

    // Método para obter o próximo valor de uma sequência
    public static int getValue(String sequenceName) throws
SQLException {
        String sql = "SELECT NEXT VALUE FOR " + sequenceName + " AS
value";
        try (Connection con = ConectorBD.getConnection();
```

```
             PreparedStatement pstmt = con.prepareStatement(sql);

             ResultSet rs = pstmt.executeQuery()) {

            if (rs.next()) {

                return rs.getInt("value");

            }

        }

        throw new SQLException("Erro ao obter próximo valor da
sequência.");

    }

}
```

Pacote "cadastro.model":

Classe PessoaFisicaDAO:

```java
package cadastro.model;


import cadastro.model.util.ConectorBD;

import cadastro.model.util.SequenceManager;


import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;
```

```
public class PessoaFisicaDAO {

    // Método para obter uma pessoa física pelo ID

    public PessoaFisica getPessoa(int id) throws SQLException {

        String sql = "SELECT * FROM PessoaFisica WHERE id = ?";

        try (Connection con = ConectorBD.getConnection();

             PreparedStatement pstmt = con.prepareStatement(sql)) {

            pstmt.setInt(1, id);

            try (ResultSet rs = pstmt.executeQuery()) {

                if (rs.next()) {

                    return criarPessoaFisica(rs);

                }

            }

        }

        return null;

    }


    // Método para obter todas as pessoas físicas

    public List<PessoaFisica> getPessoas() throws SQLException {

        String sql = "SELECT * FROM PessoaFisica";

        List<PessoaFisica> pessoas = new ArrayList<>();

        try (Connection con = ConectorBD.getConnection();

             PreparedStatement pstmt = con.prepareStatement(sql);

             ResultSet rs = pstmt.executeQuery()) {

            while (rs.next()) {

                pessoas.add(criarPessoaFisica(rs));

            }

        }

        return pessoas;
```

```
    }

    // Método para incluir uma pessoa física
    public void incluir(PessoaFisica pessoa) throws SQLException {
        String sql = "INSERT INTO Pessoa (id, nome, logradouro,
cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
        try (Connection con = ConectorBD.getConnection();
             PreparedStatement pstmt = con.prepareStatement(sql)) {
            int id = SequenceManager.getValue("nomeDaSequencia");
            pstmt.setInt(1, id);
            pstmt.setString(2, pessoa.getNome());
            pstmt.setString(3, pessoa.getLogradouro());
            pstmt.setString(4, pessoa.getCidade());
            pstmt.setString(5, pessoa.getEstado());
            pstmt.setString(6, pessoa.getTelefone());
            pstmt.setString(7, pessoa.getEmail());
            pstmt.executeUpdate();

            sql = "INSERT INTO PessoaFisica (id, cpf) VALUES (?, ?)";
            try (PreparedStatement pstmt2 = con.prepareStatement(sql))
{
                pstmt2.setInt(1, id);
                pstmt2.setString(2, pessoa.getCpf());
                pstmt2.executeUpdate();
            }
        }
    }

    // Outros métodos: alterar, excluir...
```

```
    // Método privado para criar um objeto PessoaFisica a partir do
ResultSet

    private PessoaFisica criarPessoaFisica(ResultSet rs) throws
SQLException {

        int id = rs.getInt("id");

        String nome = rs.getString("nome");

        String logradouro = rs.getString("logradouro");

        String cidade = rs.getString("cidade");

        String estado = rs.getString("estado");

        String telefone = rs.getString("telefone");

        String email = rs.getString("email");

        String cpf = rs.getString("cpf");

        return new PessoaFisica(id, nome, logradouro, cidade, estado,
telefone, email, cpf);

    }

}
```

Classe PessoaJuridicaDAO:

```java
package cadastro.model;


import cadastro.model.util.ConectorBD;

import cadastro.model.util.SequenceManager;


import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;
```

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

public class PessoaJuridicaDAO {
    // Método para obter uma pessoa jurídica pelo ID
    public PessoaJuridica getPessoa(int id) throws SQLException {
        String sql = "SELECT * FROM PessoaJuridica WHERE id = ?";
        try (Connection con = ConectorBD.getConnection();
             PreparedStatement pstmt = con.prepareStatement(sql)) {
            pstmt.setInt(1, id);
            try (ResultSet rs = pstmt.executeQuery()) {
                if (rs.next()) {
                    return criarPessoaJuridica(rs);
                }
            }
        }
        return null;
    }

    // Método para obter todas as pessoas jurídicas
    public List<PessoaJuridica> getPessoas() throws SQLException {
        String sql = "SELECT * FROM PessoaJuridica";
        List<PessoaJuridica> pessoas = new ArrayList<>();
        try (Connection con = ConectorBD.getConnection();
             PreparedStatement pstmt = con.prepareStatement(sql);
             ResultSet rs = pstmt.executeQuery()) {
            while (rs.next()) {
```

```
                pessoas.add(criarPessoaJuridica(rs));
            }
        }
        return pessoas;
    }

    // Método para incluir uma pessoa jurídica
    public void incluir(PessoaJuridica pessoa) throws SQLException {
        String sql = "INSERT INTO Pessoa (id, nome, logradouro,
cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
        try (Connection con = ConectorBD.getConnection();
             PreparedStatement pstmt = con.prepareStatement(sql)) {
            int id = SequenceManager.getValue("nomeDaSequencia");
            pstmt.setInt(1, id);
            pstmt.setString(2, pessoa.getNome());
            pstmt.setString(3, pessoa.getLogradouro());
            pstmt.setString(4, pessoa.getCidade());
            pstmt.setString(5, pessoa.getEstado());
            pstmt.setString(6, pessoa.getTelefone());
            pstmt.setString(7, pessoa.getEmail());
            pstmt.executeUpdate();

            sql = "INSERT INTO PessoaJuridica (id, cnpj) VALUES (?,
?)";
            try (PreparedStatement pstmt2 = con.prepareStatement(sql))
{
                pstmt2.setInt(1, id);
                pstmt2.setString(2, pessoa.getCnpj());
                pstmt2.executeUpdate();
```

```
            }

        }

    }



    // Outros métodos: alterar, excluir...



    // Método privado para criar um objeto PessoaJuridica a partir do
ResultSet

    private PessoaJuridica criarPessoaJuridica(ResultSet rs) throws
SQLException {

        int id = rs.getInt("id");

        String nome = rs.getString("nome");

        String logradouro = rs.getString("logradouro");

        String cidade = rs.getString("cidade");

        String estado = rs.getString("estado");

        String telefone = rs.getString("telefone");

        String email = rs.getString("email");

        String cnpj = rs.getString("cnpj");

        return new PessoaJuridica(id, nome, logradouro, cidade,
estado, telefone, email, cnpj);

    }

}
```

Criação de classe principal de testes “CadastroBDTeste”, considerando as operações no método main:

```
package cadastro.model;



import cadastro.model.util.ConectorBD;

import java.sql.SQLException;
```

```
import java.util.List;


public class CadastroBDTeste {


    public static void main(String[] args) {

        try {

            // Instanciando e persistindo uma pessoa física

            PessoaFisica pessoaFisica = new PessoaFisica();

            pessoaFisica.setNome("Fulano");

            pessoaFisica.setLogradouro("Rua A");

            pessoaFisica.setCidade("Cidade A");

            pessoaFisica.setEstado("Estado A");

            pessoaFisica.setTelefone("123456789");

            pessoaFisica.setEmail("fulano@example.com");

            pessoaFisica.setCpf("123.456.789-01");


            PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();

            pessoaFisicaDAO.incluir(pessoaFisica);


            // Alterando os dados da pessoa física

            pessoaFisica.setNome("Fulano da Silva");

            pessoaFisicaDAO.alterar(pessoaFisica);


            // Consultando e listando todas as pessoas físicas do
banco de dados

            List<PessoaFisica> pessoasFisicas =
pessoaFisicaDAO.getPessoas();

            System.out.println("Pessoas Físicas:");

            for (PessoaFisica pf : pessoasFisicas) {
```

```
            pf.exibir();
        }

        // Excluindo a pessoa física criada anteriormente
        pessoaFisicaDAO.excluir(pessoaFisica.getId());

        // Instanciando e persistindo uma pessoa jurídica
        PessoaJuridica pessoaJuridica = new PessoaJuridica();
        pessoaJuridica.setNome("Empresa XYZ");
        pessoaJuridica.setLogradouro("Rua B");
        pessoaJuridica.setCidade("Cidade B");
        pessoaJuridica.setEstado("Estado B");
        pessoaJuridica.setTelefone("987654321");
        pessoaJuridica.setEmail("empresa@example.com");
        pessoaJuridica.setCnpj("12.345.678/0001-90");

        PessoaJuridicaDAO pessoaJuridicaDAO = new
PessoaJuridicaDAO();
        pessoaJuridicaDAO.incluir(pessoaJuridica);

        // Alterando os dados da pessoa jurídica
        pessoaJuridica.setNome("Nova Empresa XYZ");
        pessoaJuridicaDAO.alterar(pessoaJuridica);

        // Consultando e listando todas as pessoas jurídicas do
banco de dados
        List<PessoaJuridica> pessoasJuridicas =
pessoaJuridicaDAO.getPessoas();
        System.out.println("\nPessoas Jurídicas:");
```

```
            for (PessoaJuridica pj : pessoasJuridicas) {
                pj.exibir();
            }

            // Excluindo a pessoa jurídica criada anteriormente
            pessoaJuridicaDAO.excluir(pessoaJuridica.getId());

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

Segunda etapa: Alimentando a Base

```
package cadastro.model;

import cadastro.model.util.ConectorBD;
import java.sql.SQLException;
import java.util.InputMismatchException;
import java.util.Scanner;

public class CadastroBDTeste {

    public static void main(String[] args) {
        try {
            Scanner scanner = new Scanner(System.in);
            int opcao;
```

```
do {
    System.out.println("=== Menu ===");
    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 opção: ");
    opcao = scanner.nextInt();
    switch (opcao) {
        case 1:
            incluir(scanner);
            break;
        case 2:
            alterar(scanner);
            break;
        case 3:
            excluir(scanner);
            break;
        case 4:
            exibirPorId(scanner);
            break;
        case 5:
            exibirTodos();
            break;
        case 0:
```

```
                        System.out.println("Encerrando o
programa...");

                        break;

                    default:

                        System.out.println("Opção inválida. Por favor,
escolha uma opção válida.");

                        break;

                }

            } while (opcao != 0);

        } catch (SQLException e) {

            System.out.println("Erro de banco de dados: " +
e.getMessage());

        } catch (InputMismatchException e) {

            System.out.println("Erro de entrada: Por favor, insira um
número inteiro.");

        } catch (Exception e) {

            System.out.println("Ocorreu um erro inesperado: " +
e.getMessage());

        }

    }



    private static void incluir(Scanner scanner) throws SQLException {

        System.out.println("Incluir:");

        System.out.println("1 - Pessoa Física");

        System.out.println("2 - Pessoa Jurídica");

        System.out.print("Escolha o tipo de pessoa (1 ou 2): ");

        int tipoPessoa = scanner.nextInt();

        scanner.nextLine(); // Limpar o buffer do scanner



        if (tipoPessoa == 1) {
```

```
            // Incluir Pessoa Física
            PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
            PessoaFisica pessoaFisica = lerPessoaFisica(scanner);
            pessoaFisicaDAO.incluir(pessoaFisica);
        } else if (tipoPessoa == 2) {
            // Incluir Pessoa Jurídica
            PessoaJuridicaDAO pessoaJuridicaDAO = new
PessoaJuridicaDAO();
            PessoaJuridica pessoaJuridica =
lerPessoaJuridica(scanner);
            pessoaJuridicaDAO.incluir(pessoaJuridica);
        } else {
            System.out.println("Tipo de pessoa inválido.");
        }
    }

    // Métodos alterar, excluir, exibirPorId, exibirTodos...

    private static PessoaFisica lerPessoaFisica(Scanner scanner) {
        PessoaFisica pessoaFisica = new PessoaFisica();
        System.out.print("Nome: ");
        pessoaFisica.setNome(scanner.nextLine());
        System.out.print("Logradouro: ");
        pessoaFisica.setLogradouro(scanner.nextLine());
        // Ler os demais atributos...
        return pessoaFisica;
    }

    private static PessoaJuridica lerPessoaJuridica(Scanner scanner) {
```

```
        PessoaJuridica pessoaJuridica = new PessoaJuridica();
        System.out.print("Nome: ");
        pessoaJuridica.setNome(scanner.nextLine());
        System.out.print("Logradouro: ");
        pessoaJuridica.setLogradouro(scanner.nextLine());
        // Ler os demais atributos...
        return pessoaJuridica;
    }
}
```