Missão prática | Nível 1 | Mundo 3

CAMPUS: POLO DISTRITO JK - ANÁPOLIS - GO

CURSO: DESENVOLVIMENTO FULL STACK

NÚMERO DA TURMA: 2024.3

SEMESTRE LETIVO: Segundo semestre de 2024

ALIUNO: Henrique Rodrigues Rabello Vieira

MATRÍCULA: 202301230527

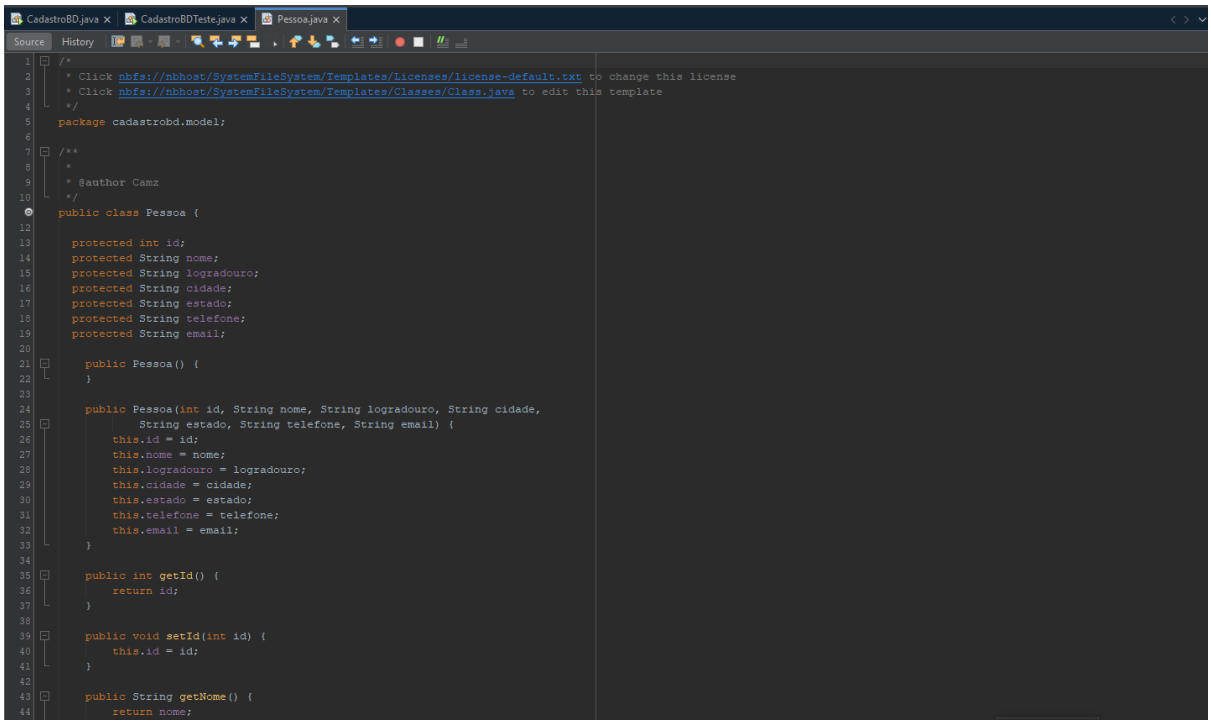
RPG0016 - BackEnd sem banco não tem

Objetivos:

- 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.

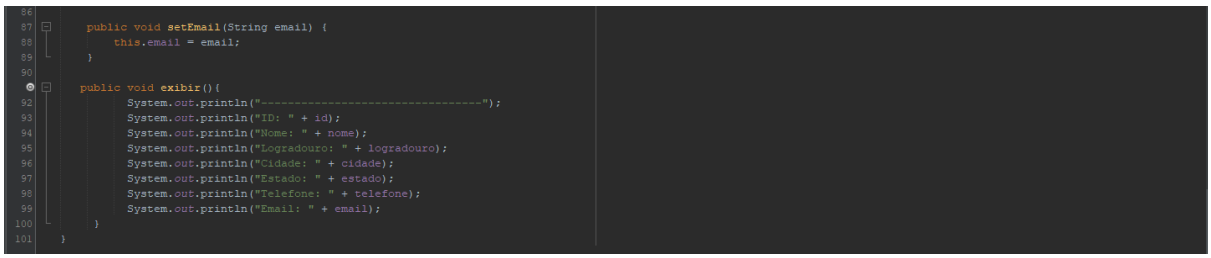
Primeiro procedimento:

Códigos solicitados:



```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 cadastrobd.model;

/**
 *
 * @author Camz
 */
public class Pessoa {

    protected int id;
    protected String nome;
    protected String logradouro;
    protected String cidade;
    protected String estado;
    protected String telefone;
    protected 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 int getId() {
        return id;
    }

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

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

```java
    }

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

    public String getLogradouro() {
        return logradouro;
    }

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

    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;
    }
```



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

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

CadastroBD.java | CadastroBDTeste.java | PessoaFisica.java

Source | History

```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 cadastrobd.model;

/**
 *
 * @author Camz
 */
public class PessoaFisica extends Pessoa {

    protected String cpf;

    public PessoaFisica() {

    }

    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;
    }

    public String getCpf() {
        return cpf;
    }

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

    @Override
    public void exibir(){
        super.exibir();
        System.out.println("CPF: " + cpf);
    }
}
```

CadastroBD.java | CadastroBDTeste.java | PessoaFisica.java | PessoaJuridica.java

Source | History

```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 cadastrobd.model;

/**
 *
 * @author Camz
 */
public class PessoaJuridica extends Pessoa {

    protected String cnpj;

    public PessoaJuridica() {
    }

    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;
    }

    public String getCnpj() {
        return cnpj;
    }

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

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

CadastroBD.java | CadastroBDTeste.java | PessoaFisica.java | PessoaJuridica.java | ConectorBD.java

Source | History

```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 cadastrobd.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;

/**
 *
 * @author Camz
 */

public class ConectorBD {
    private static final String URL = "jdbc:sqlserver://localhost:1433;databaseName=Loja;"
            + "encrypt=true;trustServerCertificate=true";
    private static final String USER = "loja";
    private static final String PASSWORD = "loja";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }

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

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

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

CadastroBD.java | CadastroBDTeste.java | PessoaFisica.java | PessoaJuridica.java | ConectorBD.java | SequenceManager.java

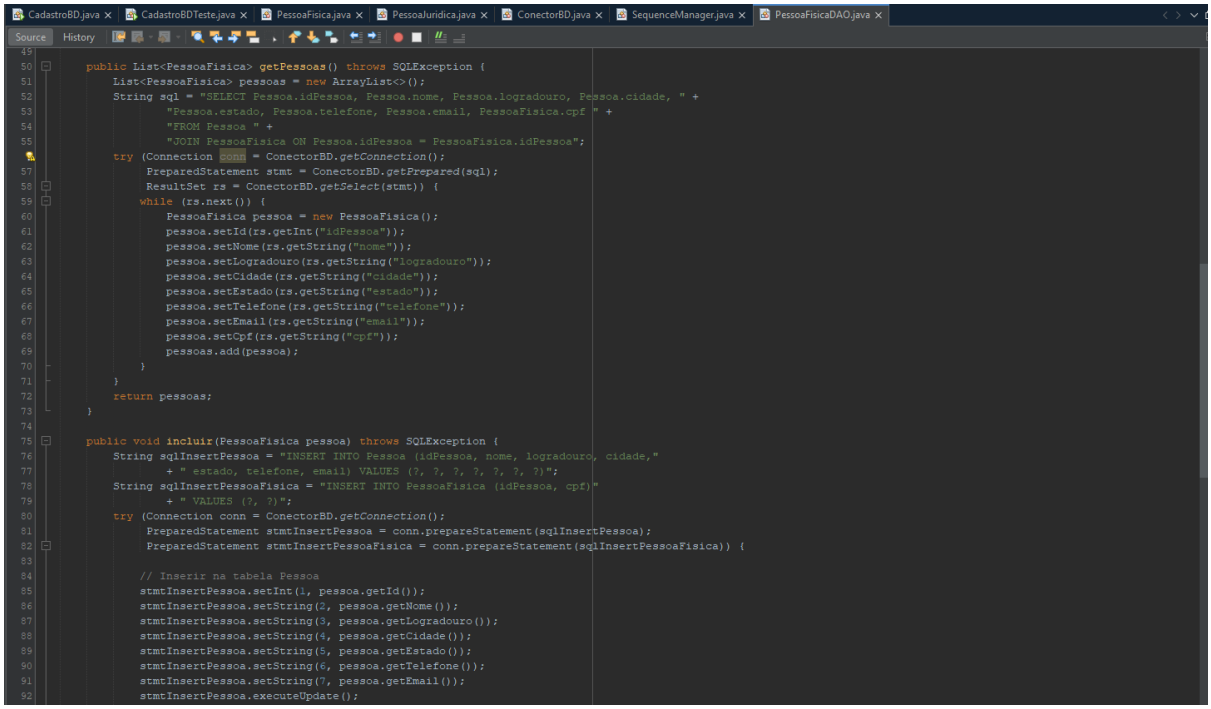
Source | History

```
/*
 * 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 cadastrobd.model.util;

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

/**
 *
 * @author Camz
 */
public class SequenceManager {

    public static int getValue(String sequenceName) throws SQLException {
        String sql = "SELECT NEXT VALUE FOR " + sequenceName + " AS nextval";
        try (Connection conn = ConectorBD.getConnection();
                PreparedStatement stmt = conn.prepareStatement(sql);
                ResultSet rs = stmt.executeQuery()) {
            if (rs.next()) {
                return rs.getInt("nextval");
            } else {
                throw new SQLException("Não foi possível obter o próximo valor da sequência "
                        + sequenceName);
            }
        }
    }
}
```

CadastroBD.java | CadastroBDTeste.java | PessoaFisica.java | PessoaJuridica.java | ConectorBD.java | SequenceManager.java | PessoaFisicaDAO.java

Source | History

```
/*
 * 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 cadastrobd.model;

import cadastrobd.model.util.ConectorBD;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Camz
 */

public class PessoaFisicaDAO {

    public PessoaFisica getPessoa(int id) throws SQLException {
        String sql = "SELECT Pessoa.idPessoa, Pessoa.nome, Pessoa.logradouro, Pessoa.cidade, " +
                "Pessoa.estado, Pessoa.telefone, Pessoa.email, PessoaFisica.cpf " +
                "FROM Pessoa " +
                "JOIN PessoaFisica ON Pessoa.idPessoa = PessoaFisica.idPessoa " +
                "WHERE Pessoa.idPessoa = ?";
        try (Connection conn = ConectorBD.getConnection();
             PreparedStatement stmt = ConectorBD.getPrepared(sql)) {
            stmt.setInt(1, id);
            try (ResultSet rs = ConectorBD.getSelect(stmt)) {
                if (rs.next()) {
                    PessoaFisica pessoa = new PessoaFisica();
                    pessoa.setId(rs.getInt("idPessoa"));
                    pessoa.setNome(rs.getString("nome"));
                    pessoa.setLogradouro(rs.getString("logradouro"));
                    pessoa.setCidade(rs.getString("cidade"));
                    pessoa.setEstado(rs.getString("estado"));
                    pessoa.setTelefone(rs.getString("telefone"));
                    pessoa.setEmail(rs.getString("email"));
                    pessoa.setCpf(rs.getString("cpf"));
                    return pessoa;
                }
            }
        }
        return null;
    }
```




```
    public List<PessoaFisica> getPessoas() throws SQLException {
        List<PessoaFisica> pessoas = new ArrayList<>();
        String sql = "SELECT Pessoa.idPessoa, Pessoa.nome, Pessoa.logradouro, Pessoa.cidade, " +
                "Pessoa.estado, Pessoa.telefone, Pessoa.email, PessoaFisica.cpf " +
                "FROM Pessoa " +
                "JOIN PessoaFisica ON Pessoa.idPessoa = PessoaFisica.idPessoa";
        try (Connection conn = ConectorBD.getConnection();
             PreparedStatement stmt = ConectorBD.getPrepared(sql);
             ResultSet rs = ConectorBD.getSelect(stmt)) {
            while (rs.next()) {
                PessoaFisica pessoa = new PessoaFisica();
                pessoa.setId(rs.getInt("idPessoa"));
                pessoa.setNome(rs.getString("nome"));
                pessoa.setLogradouro(rs.getString("logradouro"));
                pessoa.setCidade(rs.getString("cidade"));
                pessoa.setEstado(rs.getString("estado"));
                pessoa.setTelefone(rs.getString("telefone"));
                pessoa.setEmail(rs.getString("email"));
                pessoa.setCpf(rs.getString("cpf"));
                pessoas.add(pessoa);
            }
        }
        return pessoas;
    }

    public void incluir(PessoaFisica pessoa) throws SQLException {
        String sqlInsertPessoa = "INSERT INTO Pessoa (idPessoa, nome, logradouro, cidade,"
                + " estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
        String sqlInsertPessoaFisica = "INSERT INTO PessoaFisica (idPessoa, cpf)"
                + " VALUES (?, ?)";
        try (Connection conn = ConectorBD.getConnection();
             PreparedStatement stmtInsertPessoa = conn.prepareStatement(sqlInsertPessoa);
             PreparedStatement stmtInsertPessoaFisica = conn.prepareStatement(sqlInsertPessoaFisica)) {

            // Inserir na tabela Pessoa
            stmtInsertPessoa.setInt(1, pessoa.getId());
            stmtInsertPessoa.setString(2, pessoa.getNome());
            stmtInsertPessoa.setString(3, pessoa.getLogradouro());
            stmtInsertPessoa.setString(4, pessoa.getCidade());
            stmtInsertPessoa.setString(5, pessoa.getEstado());
            stmtInsertPessoa.setString(6, pessoa.getTelefone());
            stmtInsertPessoa.setString(7, pessoa.getEmail());
            stmtInsertPessoa.executeUpdate();
```

```
            // Inserir na tabela PessoaFisica
            stmtInsertPessoaFisica.setInt(1, pessoa.getId());
            stmtInsertPessoaFisica.setString(2, pessoa.getCpf());
            stmtInsertPessoaFisica.executeUpdate();
        }
    }

    public void alterar(PessoaFisica pessoa, String novoNome, String novoLogradouro, String novaCidade,
        String novoEstado,String novoTelefone, String novoEmail, String novoCpf) throws SQLException {
        String sql = "UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?,"
                + " estado = ?, telefone = ?, email = ? WHERE idPessoa = ?";
        String sqlFisica = "UPDATE PessoaFisica SET cpf = ? WHERE idPessoa = ?";

        try (Connection conn = ConectorBD.getConnection();
             PreparedStatement stmt = conn.prepareStatement(sql);
             PreparedStatement stmtFisica = conn.prepareStatement(sqlFisica)) {

            // Atualizar dados na tabela Pessoa
            stmt.setString(1, novoNome);
            stmt.setString(2, novoLogradouro);
            stmt.setString(3, novaCidade);
            stmt.setString(4, novoEstado);
            stmt.setString(5, novoTelefone);
            stmt.setString(6, novoEmail);
            stmt.setInt(7, pessoa.getId());
            stmt.executeUpdate();

            // Atualizar CPF na tabela PessoaFisica
            stmtFisica.setString(1, novoCpf);
            stmtFisica.setInt(2, pessoa.getId());
            stmtFisica.executeUpdate();
        }
    }

    public void excluir(int id) throws SQLException {
    String sql = "DELETE FROM PessoaFisica WHERE idPessoa = ?";
    String sqlPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";

    try (Connection conn = ConectorBD.getConnection();
         PreparedStatement stmt = conn.prepareStatement(sql);
         PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa)) {
        // Excluir da tabela PessoaFisica
        stmt.setInt(1, id);
        stmt.executeUpdate();
```

```
        // Excluir da tabela Pessoa
        stmtPessoa.setInt(1, id);
        stmtPessoa.executeUpdate();

        System.out.println("Pessoa fisica excluida com ID: " + id);
        }
    }
}
```

CadastroBD.java | CadastroBDTeste.java | PessoaFisica.java | PessoaJuridica.java | ConectorBD.java | SequenceManager.java | PessoaFisicaDAO.java | PessoaJuridicaDAO.java

Source | History

```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 cadastrobd.model;

import cadastrobd.model.util.ConectorBD;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Camz
 */
public class PessoaJuridicaDAO {

    public PessoaJuridica getPessoa(int id) throws SQLException {
        String sql = "SELECT Pessoa.idPessoa, Pessoa.nome, Pessoa.logradouro, Pessoa.cidade, " +
                "Pessoa.estado, Pessoa.telefone, Pessoa.email, PessoaJuridica.cnpj " +
                "FROM Pessoa " +
                "JOIN PessoaJuridica ON Pessoa.idPessoa = PessoaJuridica.idPessoa " +
                "WHERE Pessoa.idPessoa = ?";
        try (Connection conn = ConectorBD.getConnection();
             PreparedStatement stmt = ConectorBD.getPrepared(sql)) {
            stmt.setInt(1, id);
            try (ResultSet rs = ConectorBD.getSelect(stmt)) {
                if (rs.next()) {
                    PessoaJuridica pessoa = new PessoaJuridica();
                    pessoa.setId(rs.getInt("idPessoa"));
                    pessoa.setNome(rs.getString("nome"));
                    pessoa.setLogradouro(rs.getString("logradouro"));
                    pessoa.setCidade(rs.getString("cidade"));
                    pessoa.setEstado(rs.getString("estado"));
                    pessoa.setTelefone(rs.getString("telefone"));
                    pessoa.setEmail(rs.getString("email"));
                    pessoa.setCnpj(rs.getString("cnpj"));
                    return pessoa;
                }
            }
        }
        return null;
    }
```

```java

    public List<PessoaJuridica> getPessoas() throws SQLException {
        List<PessoaJuridica> pessoas = new ArrayList<>();
        String sql = "SELECT Pessoa.idPessoa, Pessoa.nome, Pessoa.logradouro, Pessoa.cidade, " +
                "Pessoa.estado, Pessoa.telefone, Pessoa.email, PessoaJuridica.cnpj " +
                "FROM Pessoa " +
                "JOIN PessoaJuridica ON Pessoa.idPessoa = PessoaJuridica.idPessoa";
        try (Connection conn = ConectorBD.getConnection();
             PreparedStatement stmt = ConectorBD.getPrepared(sql);
             ResultSet rs = ConectorBD.getSelect(stmt)) {
            while (rs.next()) {
                PessoaJuridica pessoa = new PessoaJuridica();
                pessoa.setId(rs.getInt("idPessoa"));
                pessoa.setNome(rs.getString("nome"));
                pessoa.setLogradouro(rs.getString("logradouro"));
                pessoa.setCidade(rs.getString("cidade"));
                pessoa.setEstado(rs.getString("estado"));
                pessoa.setTelefone(rs.getString("telefone"));
                pessoa.setEmail(rs.getString("email"));
                pessoa.setCnpj(rs.getString("cnpj"));
                pessoas.add(pessoa);
            }
        }
        return pessoas;
    }

    public void incluir(PessoaJuridica pessoa) throws SQLException {
        String sqlInsertPessoa = "INSERT INTO Pessoa (idPessoa, nome, logradouro, cidade,"
                + " estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
        String sqlInsertPessoaJuridica = "INSERT INTO PessoaJuridica (idPessoa, cnpj)"
                + " VALUES (?, ?)";
        try (Connection conn = ConectorBD.getConnection();
             PreparedStatement stmtInsertPessoa = conn.prepareStatement(sqlInsertPessoa);
             PreparedStatement stmtInsertPessoaJuridica = conn.prepareStatement(sqlInsertPessoaJuridica)) {

            // Inserir na tabela Pessoa
            stmtInsertPessoa.setInt(1, pessoa.getId());
            stmtInsertPessoa.setString(2, pessoa.getNome());
            stmtInsertPessoa.setString(3, pessoa.getLogradouro());
            stmtInsertPessoa.setString(4, pessoa.getCidade());
            stmtInsertPessoa.setString(5, pessoa.getEstado());
            stmtInsertPessoa.setString(6, pessoa.getTelefone());
            stmtInsertPessoa.setString(7, pessoa.getEmail());
            stmtInsertPessoa.executeUpdate();

```

```
            // Inserir na tabela PessoaJuridica
            stmtInsertPessoaJuridica.setInt(1, pessoa.getId());
            stmtInsertPessoaJuridica.setString(2, pessoa.getCnpj());
            stmtInsertPessoaJuridica.executeUpdate();
        }
    }

    public void alterar(PessoaJuridica pessoa, String novoNome, String novoLogradouro, String novaCidade,
        String novoEstado,String novoTelefone, String novoEmail, String novoCnpj) throws SQLException {
        String sql = "UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?, "
                + "estado = ?, telefone = ?, email = ? WHERE idPessoa = ?";
        String sqlJuridica = "UPDATE PessoaJuridica SET cnpj = ? WHERE idPessoa = ?";

        try (Connection conn = ConectorBD.getConnection();
             PreparedStatement stmt = conn.prepareStatement(sql);
             PreparedStatement stmtJuridica = conn.prepareStatement(sqlJuridica)) {

            // Atualizar dados na tabela Pessoa
            stmt.setString(1, novoNome);
            stmt.setString(2, novoLogradouro);
            stmt.setString(3, novaCidade);
            stmt.setString(4, novoEstado);
            stmt.setString(5, novoTelefone);
            stmt.setString(6, novoEmail);
            stmt.setInt(7, pessoa.getId());
            stmt.executeUpdate();

            // Atualizar CNPJ na tabela PessoaJuridica
            stmtJuridica.setString(1, novoCnpj);
            stmtJuridica.setInt(2, pessoa.getId());
            stmtJuridica.executeUpdate();
        }
    }

    public void excluir(int id) throws SQLException {
    String sql = "DELETE FROM PessoaJuridica WHERE idPessoa = ?";
    String sqlPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";

    try (Connection conn = ConectorBD.getConnection();
         PreparedStatement stmt = conn.prepareStatement(sql);
         PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa)) {
        // Excluir da tabela PessoaJuridica
        stmt.setInt(1, id);
        stmt.executeUpdate();
```

```
        // Excluir da tabela Pessoa
        stmtPessoa.setInt(1, id);
        stmtPessoa.executeUpdate();

        System.out.println("Pessoa juridica excluida com ID: " + id);
        }
    }
}
```

CadastroBD.java | CadastroBDTeste.java | PessoaFisica.java | PessoaJuridica.java | ConectorBD.java | SequenceManager.java | PessoaFisicaDAO.java | PessoaJuridicaDAO.java

Source | History

```
package cadastrobd;

/*
 * 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
 */


import java.sql.SQLException;
import java.util.List;
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridica;
import cadastrobd.model.PessoaJuridicaDAO;
import cadastrobd.model.util.ConectorBD;
import java.sql.Connection;

/**
 *
 * @author Camz
 */
public class CadastroBDTeste {

    public static void main(String[] args) {
        try {
            Connection conn = ConectorBD.getConnection();
            PessoaFisicaDAO pfDAO = new PessoaFisicaDAO();
            PessoaJuridicaDAO pjDAO = new PessoaJuridicaDAO();

            // Criar uma pessoa fisica
            PessoaFisica pf = new PessoaFisica(6, "Pedro", "Rua A, 10", "Atibaia", "SP",
                    "1234-5678", "pedro@gmail.com", "12345678910");

            // Persistir a pessoa fisica no banco de dados
            pfDAO.incluir(pf);
            System.out.println("Pessoa fisica criada:");
            pf.exibir();
            System.out.println();

            // Alterar os dados da pessoa fisica no banco
            pfDAO.alterar(pf, "Pedro Alves", "Rua B, 11", "Atibaia", "SP",
                    "9999-8888", "pedro.alves@email.com", "12345678900");
            System.out.println("-----------------------------------");
            System.out.println("Dados da pessoa fisica alterados.");
            System.out.println("-----------------------------------");
            System.out.println();
```

```java
            // Consultar todas as pessoas fisicas do banco de dados e listar no console
            List<PessoaFisica> pessoasFisicas = pfDAO.getPessoas();
            System.out.println("Todas as pessoas fisicas:");
            for (PessoaFisica pessoaFisica : pessoasFisicas) {
                pessoaFisica.exibir();
            }
            System.out.println();

            // Excluir a pessoa fisica criada anteriormente no banco
            System.out.println("---------------------------------");
            pfDAO.excluir(pf.getId());
            System.out.println("---------------------------------");
            System.out.println();

            // Criar uma pessoa juridica
            PessoaJuridica pj = new PessoaJuridica(7, "Empresa ABC", "Av. Principal, 100",
                    "Sao Paulo", "SP", "1234-5678", "empresa@abc.com", "12345678901234");

            // Persistir a pessoa juridica no banco de dados
            pjDAO.incluir(pj);
            System.out.println("Pessoa juridica criada:");
            pj.exibir();
            System.out.println();

            // Alterar os dados da pessoa jurídica no banco
            pjDAO.alterar(pj, "Companhia ABC", "Av. Nova, 200", "Rio de Janeiro", "RJ",
                    "9876-5432", "companhia@abc.com", "98765432109876");
            System.out.println("---------------------------------");
            System.out.println("Dados da pessoa juridica alterados.");
            System.out.println("---------------------------------");
            System.out.println();

            // Consultar todas as pessoas juridicas do banco de dados e listar no console
            List<PessoaJuridica> pessoasJuridicas = pjDAO.getPessoas();
            System.out.println("Todas as pessoas juridicas:");
            for (PessoaJuridica pessoaJuridica : pessoasJuridicas) {
                pessoaJuridica.exibir();
            }
            System.out.println();

            // Excluir a pessoa juridica criada anteriormente no banco
            System.out.println("---------------------------------");
            pjDAO.excluir(pj.getId());
            System.out.println("---------------------------------");
            System.out.println();
```

```java
            ConectorBD.close(conn);

        } catch (SQLException e) {
            System.out.println("Ocorreu um erro: " + e.getMessage());
        }
    }
}
```

Resultados:

```
Pessoa juridica criada:
==============================
ID: 7
Nome: Empresa ABC
Logradouro: Av. Principal, 100
Cidade: Sao Paulo
Estado: SP
Telefone: 1234-5678
E-mail: empresa@abc.com
CPF: 12345678901234

==============================
Dados da pessoa juridica alterados.
==============================

Todas as pessoas juridicas
==============================
ID: 4
Nome: Distribuidora Diamante
Logradouro: Avenida A, 40
Cidade: Curitiba
Estado: PR
Telefone: 4444-4444
E-mail: diamante@gmail.com
CPF: 44444444444
==============================
ID: 5
Nome: Empresa Estrela
Logradouro: Avenida B, 50
Cidade: Recife
Estado: PE
Telefone: 5555-5555
E-mail: estrela@gmail.com
CPF: 55555555555
==============================
ID: 7
Nome: Companhia ABC
Logradouro: Av. Nova, 200
Cidade: Rio de Janeiro
Estado: RJ
Telefone: 9876-5432
E-mail: companhia@abc.com
CPF: 98765432109876

==============================
Pessoa juridica excluida com ID: 7
```

Análise e conclusão:

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

O JDBC é importante para a comunicação entre a aplicação JAVA e o banco de dados .

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

O preparedStatement é mais seguro e eficiente que statement, pois ele evita a injeção de SQL e permite a definição de paramêtros.

Como o padrão DAO melhora a manutenibilidade do software?

Melhora separando a lógica de acesso da lógica de negócios.

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

Pode ser feita de diversas formas, todas as classes compartilhando uma tabela, cada classe ter sua própria tabela, tendo apenas campos exclusivos e uma chave estrangeira para a tabela da superclasse, ou cada subclasse ter sua tabela o que impacta diretamente o sistema, cada um tendo as suas vantagens.

Segundo procedimento:

Códigos solicitados:

CadastroBDTeste.java | CadastroBD.java

Source | History

```java
/*
 * 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 cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridica;
import cadastrobd.model.PessoaJuridicaDAO;
import java.util.Scanner;
import java.sql.SQLException;
import java.util.ArrayList;

/**
 *
 * @author Camz
 */

public class CadastroBD {

    private static final Scanner sc = new Scanner(System.in);
    private static final PessoaFisicaDAO pfDao = new PessoaFisicaDAO();
    private static final PessoaJuridicaDAO pjDao = new PessoaJuridicaDAO();

    public static void main(String[] args) {
        int opcao = -1;
        while (opcao != 0) {
            printMenu();
            opcao = inputInt("ESCOLHA: ");
            switch (opcao) {
                case 1 -> incluir();
                case 2 -> alterar();
                case 3 -> excluir();
                case 4 -> buscarPeloId();
                case 5 -> exibirTodos();
                case 0 -> System.out.println("Finalizando...");
                default -> System.out.println("Escolha invalida!");
            }
        }
    }

    private static 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("==============================");
    }

    private static String input(String prompt) {
        System.out.print(prompt);
        return sc.nextLine();
    }

    private static int inputInt(String prompt) {
        System.out.print(prompt);
        try {
            return Integer.parseInt(sc.nextLine());
        } catch (NumberFormatException e) {
            System.out.println("Erro: Entrada invalida. Tente novamente.");
            return inputInt(prompt);
        }
    }

    private static void incluir() {
        System.out.println("\nIncluindo pessoa...");
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        String tipoPessoa = input("TIPO DE PESSOA: ").toUpperCase();
        Integer id = inputInt("Informe o ID: ");
        switch (tipoPessoa) {
            case "F" -> {
                try {
                    pfDao.incluir(criarPessoaFisica(id));
                    System.out.println("Pessoa fisica incluida com sucesso!");
                } catch (SQLException e) {
                    System.out.println("Erro ao incluir pessoa fisica: " + e.getMessage());
                }
            }
            case "J" -> {
                try {
                    pjDao.incluir(criarPessoaJuridica(id));
                    System.out.println("Pessoa juridica incluida com sucesso!");
                } catch (SQLException e) {
                    System.out.println("Erro ao incluir pessoa juridica: " + e.getMessage());
                }
            }
            default -> System.out.println("Tipo de pessoa invalido!");
        }
    }

    private static PessoaFisica criarPessoaFisica(Integer id) {
        System.out.println("Criando Pessoa Fisica...");
        String nome = input("Informe o nome: ");
        String logradouro = input("Informe o logradouro: ");
        String cidade = input("Informe a cidade: ");
        String estado = input("Informe o estado: ");
        String telefone = input("Informe o telefone: ");
        String email = input("Informe o email: ");
        String cpf = input("Informe o CPF: ");
        return new PessoaFisica(id, nome, logradouro, cidade, estado, telefone, email, cpf);
    }

    private static PessoaJuridica criarPessoaJuridica(Integer id) {
        System.out.println("Criando Pessoa Juridica...");
        String nome = input("Informe o nome: ");
        String logradouro = input("Informe o logradouro: ");
        String cidade = input("Informe a cidade: ");
        String estado = input("Informe o estado: ");
        String telefone = input("Informe o telefone: ");
        String email = input("Informe o email: ");
        String cnpj = input("Informe o CNPJ: ");
        return new PessoaJuridica(id, nome, logradouro, cidade, estado, telefone, email, cnpj);
    }

}
```

```
    private static void alterar() {
        System.out.println("\nAlterando pessoa...");
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        String tipoPessoa = input("TIPO DE PESSOA: ").toUpperCase();
        if (tipoPessoa.equals("F")) {
            try {
                Integer id = inputInt("Informe o ID da Pessoa Fisica: ");
                PessoaFisica pf = pfDao.getPessoa(id);
                if (pf != null) {
                    System.out.println("Dados atuais da Pessoa Fisica:");
                    pf.exibir();

                    String novoNome = input("Informe o novo nome: ");
                    String novoLogradouro = input("Informe o novo logradouro: ");
                    String novaCidade = input("Informe a nova cidade: ");
                    String novoEstado = input("Informe o novo estado: ");
                    String novoTelefone = input("Informe o novo telefone: ");
                    String novoEmail = input("Informe o novo email: ");
                    String novoCpf = input("Informe o novo CPF: ");

                    pfDao.alterar(pf, novoNome, novoLogradouro, novaCidade,
                            novoEstado, novoTelefone, novoEmail, novoCpf);
                    System.out.println("Pessoa fisica alterada com sucesso!");
                } else {
                    System.out.println("ID errado!");
                }
            } catch (NullPointerException | SQLException e) {
                System.out.println("Erro ao alterar pessoa fisica: " + e.getMessage());
            }
        } else if (tipoPessoa.equals("J")) {
            try {
                Integer id = inputInt("Informe o ID da Pessoa Juridica: ");
                PessoaJuridica pj = pjDao.getPessoa(id);
                if (pj != null) {
                    System.out.println("Dados atuais da Pessoa Juridica:");
                    pj.exibir();

                    String novoNome = input("Informe o novo nome: ");
                    String novoLogradouro = input("Informe o novo logradouro: ");
                    String novaCidade = input("Informe a nova cidade: ");
                    String novoEstado = input("Informe o novo estado: ");
                    String novoTelefone = input("Informe o novo telefone: ");
                    String novoEmail = input("Informe o novo email: ");
                    String novoCnpj = input("Informe o novo CNPJ: ");

                    pjDao.alterar(pj, novoNome, novoLogradouro, novaCidade,
                            novoEstado, novoTelefone, novoEmail, novoCnpj);
                    System.out.println("Pessoa juridica alterada com sucesso!");
                } else {
                    System.out.println("ID errado!");
                }
            } catch (NullPointerException | SQLException e) {
                System.out.println("Erro ao alterar pessoa juridica: " + e.getMessage());
            }
        } else {
            System.out.println("Tipo de pessoa invalido!");
        }
    }

    private static void excluir() {
        System.out.println("\nExcluindo pessoa...");
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        String tipoPessoa = input("TIPO DE PESSOA: ").toUpperCase();
        switch (tipoPessoa) {
            case "F" -> {
                try {
                    Integer id = inputInt("Informe o ID da Pessoa Fisica: ");
                    PessoaFisica pf = pfDao.getPessoa(id);
                    if (pf != null) {
                        pfDao.excluir(pf.getId());
                        System.out.println("Sucesso ao excluir!");
                    } else {
                        System.out.println("ID errado!");
                    }
                } catch (NullPointerException | SQLException e) {
                    System.out.println("Erro ao excluir pessoa fisica: " + e.getMessage());
                }
            }
            case "J" -> {
                try {
                    Integer id = inputInt("Informe o ID da Pessoa Juridica: ");
                    PessoaJuridica pj = pjDao.getPessoa(id);
                    if (pj != null) {
                        pjDao.excluir(pj.getId());
                        System.out.println("Sucesso ao excluir!");
                    } else {
                        System.out.println("ID errado!");
                    }
                } catch (NullPointerException | SQLException e) {
                    System.out.println("Erro ao excluir pessoa juridica: " + e.getMessage());
                }
            }
```

```
            default -> System.out.println("Tipo de pessoa invalido!");
        }
    }

    private static void buscarPeloId() {
        System.out.println("\nBuscando pessoa pelo ID...");
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        String tipoPessoa = input("TIPO DE PESSOA: ").toUpperCase();
        switch (tipoPessoa) {
            case "F" -> {
                try {
                    Integer id = inputInt("Informe o ID da Pessoa Fisica: ");
                    PessoaFisica pf = pfDao.getPessoa(id);
                    if (pf != null) {
                        pf.exibir();
                    } else {
                        System.err.println("Pessoa fisica com o ID " + id + " nao encontrada!");
                    }
                } catch (SQLException e) {
                    System.err.println("Erro ao buscar pessoa fisica: " + e.getMessage());
                }
            }
            case "J" -> {
                try {
                    Integer id = inputInt("Informe o ID da Pessoa Juridica: ");
                    PessoaJuridica pj = pjDao.getPessoa(id);
                    if (pj != null) {
                        pj.exibir();
                    } else {
                        System.err.println("Pessoa juridica com o ID " + id + " nao encontrada!");
                    }
                } catch (SQLException e) {
                    System.err.println("Erro ao buscar pessoa juridica: " + e.getMessage());
                }
            }
            default -> System.out.println("Tipo de pessoa invalido!");
        }
    }

```

```
    private static void exibirTodos() {
        System.out.println("\nExibindo todas as pessoas...");
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        String tipoPessoa = input("TIPO DE PESSOA: ").toUpperCase();
        try {
            switch (tipoPessoa) {
                case "F" -> {
                    ArrayList<PessoaFisica> listaPf = (ArrayList<PessoaFisica>) pfDao.getPessoas();
                    for (PessoaFisica pessoa : listaPf) {
                        pessoa.exibir();
                    }
                }
                case "J" -> {
                    ArrayList<PessoaJuridica> listaPj = (ArrayList<PessoaJuridica>) pjDao.getPessoas();
                    for (PessoaJuridica pessoa : listaPj) {
                        pessoa.exibir();
                    }
                }
                default -> System.out.println("Tipo de pessoa invalido!");
            }
        } catch (SQLException e) {
            System.out.println("Erro ao exibir pessoas: " + e.getMessage());
        }
    }
}
```

Resultados:

```
============================
1 - Incluir
2 - Alterar
3 - Excluir
4 - Buscar pelo ID
5 - Exibir todos
0 - Sair
============================
ESCOLHA: 1

Incluindo pessoa...
F - Pessoa Fisica | J - Pessoa Juridica
TIPO DE PESSOA: F
Informe o ID: 11
Criando pessoa fisica...
Informe o nome: Vera
Informe o logradouro: Rua A, 02
Informe a cidade: Rio de Janeiro
Informe o estado: RJ
Informe o telefone: 2222-1111
Informe o e-mail: vera@gmail.com
Informe o CPF: 11221122112
Pessoa fisica incluida com sucesso!
```

```
============================
1 - Incluir
2 - Alterar
3 - Excluir
4 - Buscar pelo ID
5 - Exibir todos
0 - Sair
============================
ESCOLHA: 2

Alterando pessoa...
F - Pessoa Fisica | J - Pessoa Juridica
TIPO DE PESSOA: F
Informe o ID da Pessoa Fisica: 11
Dados atuais da Pessoa Fisica:
----------------------------
ID: 11
Nome: Vera
Logradouro: Rua A, 02
Cidade: Rio de Janeiro
Estado: RJ
Telefone: 2222-1111
E-mail: vera@gmail.com
CPF: 11221122112
Informe o novo nome: Vera Duarte
Informe o novo logradouro: Rua A, 02
Informe a nova cidade: Rio de Janeiro
Informe o novo estado: RJ
Informe o novo telefone: 2222-1111
Informe o novo e-mail: veraduarte@gmail.com
Informe o novo CPF: 11221122112
Pessoa fisica alterada com sucesso!
```

```
============================
1 - Incluir
2 - Alterar
3 - Excluir
4 - Buscar pelo ID
5 - Exibir todos
0 - Sair
============================
ESCOLHA: 3

Excluindo pessoa...
P - Pessoa Fisica | J - Pessoa Juridica
TIPO DE PESSOA: J
Informe o ID da Pessoa Juridica: 15
Pessoa Juridica excluida com o ID: 15
Sucesso ao excluir!
```

```
============================
1 - Incluir
2 - Alterar
3 - Excluir
4 - Buscar pelo ID
5 - Exibir todos
0 - Sair
============================
ESCOLHA: 4

Busanco pessoa pelo ID...
P - Pessoa Fisica | J - Pessoa Juridica
TIPO DE PESSOA: J
Informe o ID da Pessoa Juridica: 15
```

```
Pessoa juridica com o ID 15 nao encontrada!
```

Resultados | Mensagens

	idPessoa	nome	logradouro	cidade	estado	telefone	email	cpf	cnpj
1	1	Alana	Rua X, 10	Manaus	AM	1111-1111	alana@gmail.com	11111111111	NULL
2	2	Breno	Rua Y, 20	Rio de Janeiro	RJ	2222-2222	breno@gmail.com	22222222222	NULL
3	3	Caio	Rua Z, 30	Porto Alegre	RS	3333-3333	caio@gmail.com	33333333333	NULL
4	4	Distribuidora Diamante	Avenida A, 40	Curitiba	PR	4444-4444	diamante@gmail.com	NULL	44444444444444
5	5	Empresa Estrela	Avenida B, 50	Recife	PE	5555-5555	estrela@gmail.com	NULL	55555555555555
6	10	Carlos Silva	Rua A, 01	Osasco	SP	1111-2222	carlos2024@gmail.com	10101010101	NULL

Análise e conclusão:

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

A persistência em arquivo consiste em armazenar dados em arquivos, podendo ser em xml, json e até arquivos binários. Já em banco de dados as informações são armazenadas de forma estruturada, oferecendo vários recursos e sendo mais adequada para aplicativos em grande escala.

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

Reduzindo a necessidade de classes anônimas, simplificando a sintaxe para definir comportamentos que podem ser passados como argumentos para métodos ou armazenados em variáveis.

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

Os métodos chamados pelo main precisam ser estáticos porque são chamados no contexto da classe, não de uma instância, podendo ser chamados sem ter que criar um objeto para instanciar a classe.