



Estácio

Criação de aplicativo Java, com acesso ao banco de dados SQL Server através do middleware JDBC.

Aluno: Caio Viana Castelo Branco

Matrícula: 202307465925

Disciplina: RPG0016 - BackEnd sem banco não tem

Curso: Desenvolvimento Full Stack

Turma: 9001

Semestre Letivo: 2025.1

Campus: Parangaba

1. Objetivos da prática

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

2. Códigos utilizados na prática

1º Procedimento | Mapeamento Objeto-Relacional e DAO

- Classe Pessoa

```
package cadastrbd.model;
import java.io.Serializable;

/**
 *
 * @author grego
 */
public class Pessoa implements Serializable{
    private int id;
    private String nome;
    private String logradouro;
    private String cidade;
    private String estado;
    private String telefone;
    private String email;

    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 setId(int id) {
        this.id = id;
    }
}
```

```

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

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

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

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

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

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

public int getId() {
    return id;
}

public String getNome() {
    return nome;
}

public String getLogradouro() {
    return logradouro;
}

public String getCidade() {
    return cidade;
}

public String getEstado() {
    return estado;
}

public String getTelefone() {
    return telefone;
}

public String getEmail() {
    return email;
}

public void exibir(){
    System.out.print("id: "+this.id + "\n" + "Nome: " + this.nome + "\n" +
        "logradouro: "+this.logradouro+"\n"+"cidade: "+this.cidade+"\n"+
        "estado: "+this.estado+"\n" + "telefone: " + this.telefone + "\n"+

```

```
"email: " + this.email );
    }
}
```

- Classe PessoaFisica

```
package cadastrobd.model;
import java.io.Serializable;

/**
 *
 * @author grego
 */
public class PessoaFisica extends Pessoa implements Serializable {

    private String cpf;

    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;
    }

    public void exibir(){
        System.out.print("id: " + getId() + "\n" + "Nome: " + getNome() + "\n" +
        "logradouro: "+getLogradouro()+"\n"+"cidade: "+getCidade()+"\n"+
        "estado: "+getEstado()+"\n" + "telefone: " + getTelefone() + "\n"+
        "email: " + getEmail() + "\n"+
        "CPF: "+this.cpf + "\n");
    }
}
```

- Classe PessoaJuridica

```
package cadastrobd.model;
import java.io.Serializable;
```

```

/**
 *
 * @author grego
 */
public class PessoaJuridica extends Pessoa implements Serializable{

    private String cnpj;

    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;
    }

    public void exibir(){
        System.out.print("id: " + getId() + "\n" + "Nome: " + getNome() + "\n" +
"logradouro: "+getLogradouro()+"\n"+"cidade: "+getCidade()+"\n"+
"estado: "+getEstado()+"\n" + "telefone: " + getTelefone() + "\n"+
"email: " + getEmail() + "\n"+
"CNPJ: "+this.cnpj + "\n");
    }
}

```

- Classe ConectorBD

```

package cadastro.model.util;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.swing.JOptionPane;

/**
 *
 * @author grego
 */

public class ConectorBD {

    Connection conn = null;

```

```

//Metodo para conectar java con SQLServer

public Connection getConnection() throws Exception{
    Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
    Connection conn =
DriverManager.getConnection("jdbc:sqlserver://localhost\\MSSQLSERVER2019E:1433;d
atabaseName=Loja;encrypt=true;trustServerCertificate=true",
        "loja", "loja");
    return conn;
}

public void closeConnection()throws Exception{
    getConnection().close();
    //JOptionPane.showMessageDialog(null, "Conexao finalizada");
}

public PreparedStatement getPrepared(String sql) throws Exception {
    PreparedStatement ps = getConnection().prepareStatement(sql);
    return ps;
}

public void closeStatement(String sql)throws Exception{
    getPrepared(sql).close();
    //JOptionPane.showMessageDialog(null, "Statement finalizado");
}

public ResultSet getSelect(PreparedStatement ps) throws Exception {
    ResultSet rs = ps.executeQuery();
    //ResultSet rs = getConnection().createStatement().executeQuery("");
    return rs;
}

public void closeResult(PreparedStatement ps)throws Exception{
    getSelect(ps).close();
    //JOptionPane.showMessageDialog(null, "ResultSet finalizado");
}
}

```

- Classe SequenceManager

```

package cadastro.model.util;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;

/**
 *
 * @author grego

```

```

*/
public class SequenceManager {

    public int getValue(String sequencia)throws Exception{
        int resultado = 0;
        Connection con =
DriverManager.getConnection("jdbc:sqlserver://localhost\\MSSQLSERVER2019E:1433;d
atabaseName=Loja;encrypt=true;trustServerCertificate=true",
        "loja", "loja");
        String sql = "SELECT NEXT VALUE FOR "+sequencia+" as proximoId";
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery(sql);
        while(rs.next())
            resultado = rs.getInt("proximoId");
        return resultado;
    }
}

```

- Classe PessoaFisicaDAO

```

package cadastro.model;
import cadastrobd.model.PessoaFisica;
import java.util.ArrayList;
import java.util.List;
import cadastro.model.util.ConectorBD;
import com.sun.jdi.connect.spi.Connection;
import java.sql.ResultSet;
import java.sql.PreparedStatement;

public class PessoaFisicaDAO {

    public ConectorBD connection = new ConectorBD();

    public PessoaFisica getPessoa(int id)throws Exception {
        PessoaFisica pessoa = null;
        String sql = "select *\n" +
            "from pessoa, pessoa_fisica\n" +
            "where pessoa.idpessoa = "+ id + "AND " +
            "pessoa.idpessoa = pessoa_fisica.idpessoa;";
        PreparedStatement ps = connection.getPrepared(sql);
        ResultSet resultado = ps.executeQuery();
        while(resultado.next()){
            pessoa = new PessoaFisica(resultado.getInt("idpessoa"),
            resultado.getString("nome"),
            resultado.getString("logradouro"),
            resultado.getString("cidade"),
            resultado.getString("estado"),
            resultado.getString("telefone"),
            resultado.getString("email"),
            resultado.getString("cpf"));
            connection.closeConnection();
            //connection.closeResult(ps);
        }
    }
}

```

```

        connection.closeStatement(sql);
    } return pessoa;
}

public List<PessoaFisica> getPessoas() throws Exception{
    List<PessoaFisica> lista = new ArrayList<>();
    String sql = "select *\n" +
        "from pessoa, pessoa_fisica\n" +
        "where pessoa.idpessoa = pessoa_fisica.idpessoa;";
    PreparedStatement ps = connection.getPrepared(sql);
    ResultSet resultado = ps.executeQuery();
    while(resultado.next()){
        //System.out.println(resultado.getString(5));
        lista.add(new PessoaFisica(resultado.getInt("idpessoa"),
            resultado.getString("nome"),
            resultado.getString("logradouro"),
            resultado.getString("cidade"),
            resultado.getString("estado"),
            resultado.getString("telefone"),
            resultado.getString("email"),
            resultado.getString("cpf")));
        connection.closeConnection();
        //connection.closeResult(ps);
        connection.closeStatement(sql);
    } return lista;
}

public void incluir(PessoaFisica pessoafisica) throws Exception{
    String sqlfisica = "insert into pessoa_fisica (idpessoa, cpf) values
(?,?)";
    String sqlpessoa = "insert into pessoa (idpessoa,nome,logradouro,
cidade,"
        + "estado, telefone, email ) values (?,?,?,?,?,?,?)";
    PreparedStatement ps = connection.getPrepared(sqlfisica);
    PreparedStatement ps1 = connection.getPrepared(sqlpessoa);
    //ResultSet resultado = ps.executeQuery();
    ps.setInt(1, pessoafisica.getId());
    ps.setString(2, pessoafisica.getCpf());
    ps1.setInt(1, pessoafisica.getId());
    ps1.setString(2, pessoafisica.getNome());
    ps1.setString(3, pessoafisica.getLogradouro());
    ps1.setString(4, pessoafisica.getCidade());
    ps1.setString(5, pessoafisica.getEstado());
    ps1.setString(6, pessoafisica.getTelefone());
    ps1.setString(7, pessoafisica.getEmail());
    ps1.execute();
    ps.execute();
    connection.closeConnection();
    //connection.closeResult(ps);
    connection.closeStatement(sqlfisica);
}

public void alterar(int id, String cpf, String nome, String logradouro,
    String cidade, String estado,String telefone, String email) throws
Exception{

```



```

PessoaFisica pessoa = getPessoa(id);
String sqlfisica = "UPDATE pessoa_fisica SET cpf=? where idpessoa =
"+id;
String sqlpessoa = "UPDATE pessoa SET nome=?, logradouro=?, cidade=?,
+ "estado=?, telefone=?, email=? WHERE idpessoa= "+id;
PreparedStatement ps = connection.prepareStatement(sqlfisica);
PreparedStatement ps1 = connection.prepareStatement(sqlpessoa);
if(cpf.equals("")){
    ps.setString(1, pessoa.getCpf());
} else{
    ps.setString(1, cpf);
}

if(nome.equals("")){
    ps1.setString(1, pessoa.getNome());
} else{
    ps1.setString(1, nome);
}

if(logradouro.equals("")){
    ps1.setString(2, pessoa.getLogradouro());
} else{
    ps1.setString(2, logradouro);
}

if(cidade.equals("")){
    ps1.setString(3, pessoa.getCidade());
} else{
    ps1.setString(3, cidade);
}

if(estado.equals("")){
    ps1.setString(4, pessoa.getEstado());
} else{
    ps1.setString(4, estado);
}

if(telefone.equals("")){
    ps1.setString(5, pessoa.getTelefone());
} else{
    ps1.setString(5, telefone);
}
if(email.equals("")){
    ps1.setString(6, pessoa.getEmail());
} else{
    ps1.setString(6, email);
}
ps.execute();
ps1.execute();
connection.closeConnection();
//connection.closeResult(ps);
connection.closeStatement(sqlfisica);
}

public void excluir(int id)throws Exception{
    String sqlfisica = "DELETE FROM pessoa_fisica WHERE idpessoa="+id;
    String sqlpessoa = "DELETE FROM pessoa WHERE idpessoa="+id;
    PreparedStatement ps = connection.prepareStatement(sqlfisica);

```

```

        PreparedStatement ps1 = connection.prepareStatement(sqlpessoa);
        ps.execute();
        ps1.execute();
        connection.closeConnection();
        //connection.closeResult(ps);
        connection.closeStatement(sqlfisica);
    }
}

```

- Classe PessoaJuridicaDAO

```

package cadastro.model;

import cadastro.model.util.ConectorBD;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import cadastrobd.model.PessoaJuridica;
import java.util.ArrayList;
import java.util.List;

public class PessoaJuridicaDAO {

    public ConectorBD connection = new ConectorBD();

    public PessoaJuridica getPessoa(int id) throws Exception {
        PessoaJuridica pessoa = null;
        String sql = "select *\n" +
            "from pessoa, pessoa_juridica\n" +
            "where pessoa.idpessoa = " + id + "AND " +
            "pessoa.idpessoa = pessoa_juridica.idpessoa;";
        PreparedStatement ps = connection.prepareStatement(sql);
        ResultSet resultado = ps.executeQuery();
        while(resultado.next()){
            pessoa = new PessoaJuridica(resultado.getInt("idpessoa"),
                resultado.getString("nome"),
                resultado.getString("logradouro"),
                resultado.getString("cidade"),
                resultado.getString("estado"),
                resultado.getString("telefone"),
                resultado.getString("email"),
                resultado.getString("cnpj"));
            connection.closeConnection();
            //connection.closeResult(ps);
            connection.closeStatement(sql);
        } return pessoa;
    }

    public List<PessoaJuridica> getPessoas() throws Exception{
        List<PessoaJuridica> lista = new ArrayList<>();
        String sql = "select *\n" +
            "from pessoa, pessoa_juridica\n" +
            "where pessoa.idpessoa = pessoa_juridica.idpessoa;";
        PreparedStatement ps = connection.prepareStatement(sql);
        ResultSet resultado = ps.executeQuery();
    }
}

```

```

        while(resultado.next()){
            //System.out.println(resultado.getString(5));
            lista.add(new PessoaJuridica(resultado.getInt("idpessoa"),
            resultado.getString("nome"),
            resultado.getString("logradouro"),
            resultado.getString("cidade"),
            resultado.getString("estado"),
            resultado.getString("telefone"),
            resultado.getString("email"),
            resultado.getString("cnpj")));
            connection.closeConnection();
            //connection.closeResult(ps);
            connection.closeStatement(sql);
        } return lista;
    }

    public void incluir(PessoaJuridica pessoajuridica)throws Exception{
        String sqljuridica = "insert into pessoa_juridica (idpessoa, cnpj)
values (?,?)";
        String sqlpessoa = "insert into pessoa (idpessoa,nome,logradouro,
cidade,"
            + "estado, telefone, email ) values (?,?,?,?,?,?,?)";
        PreparedStatement ps = connection.getPrepared(sqljuridica);
        PreparedStatement ps1 = connection.getPrepared(sqlpessoa);
        //ResultSet resultado = ps.executeQuery();
        ps.setInt(1, pessoajuridica.getId());
        ps.setString(2, pessoajuridica.getCnpj());
        ps1.setInt(1, pessoajuridica.getId());
        ps1.setString(2, pessoajuridica.getNome());
        ps1.setString(3, pessoajuridica.getLogradouro());
        ps1.setString(4, pessoajuridica.getCidade());
        ps1.setString(5, pessoajuridica.getEstado());
        ps1.setString(6, pessoajuridica.getTelefone());
        ps1.setString(7, pessoajuridica.getEmail());
        ps1.execute();
        ps.execute();
        connection.closeConnection();
        //connection.closeResult(ps);
        connection.closeStatement(sqljuridica);
    }

    public void alterar(int id, String cnpj, String nome, String logradouro,
        String cidade, String estado,String telefone, String email)throws
Exception{
        PessoaJuridica pessoa = getPessoa(id);
        String sqljuridica = "UPDATE pessoa_juridica SET cnpj=? where idpessoa =
"+id;
        String sqlpessoa = "UPDATE pessoa SET nome=?, logradouro=?, cidade=?,"
            + "estado=?, telefone=?, email=? WHERE idpessoa= "+id;
        PreparedStatement ps = connection.getPrepared(sqljuridica);
        PreparedStatement ps1 = connection.getPrepared(sqlpessoa);
        if(cnpj.equals("")){
            ps.setString(1, pessoa.getCnpj());
        } else{
            ps.setString(1, cnpj);
        }
    }

```

```

        if(nome.equals("")){
            ps1.setString(1, pessoa.getNome());
        } else{
            ps1.setString(1, nome);
        }

        if(logradouro.equals("")){
            ps1.setString(2, pessoa.getLogradouro());
        } else{
            ps1.setString(2, logradouro);
        }

        if(cidade.equals("")){
            ps1.setString(3, pessoa.getCidade());
        } else{
            ps1.setString(3, cidade);
        }

        if(estado.equals("")){
            ps1.setString(4, pessoa.getEstado());
        } else{
            ps1.setString(4, estado);
        }

        if(telefone.equals("")){
            ps1.setString(5, pessoa.getTelefone());
        } else{
            ps1.setString(5, telefone);
        }

        if(email.equals("")){
            ps1.setString(6, pessoa.getEmail());
        } else{
            ps1.setString(6, email);
        }
        ps.execute();
        ps1.execute();
        connection.closeConnection();
        //connection.closeResult(ps);
        connection.closeStatement(sqljuridica);
    }

    public void excluir(int id)throws Exception{
        String sqljuridica = "DELETE FROM pessoa_juridica WHERE idpessoa="+id;
        String sqlpessoa = "DELETE FROM pessoa WHERE idpessoa="+id;
        PreparedStatement ps = connection.getPrepared(sqljuridica);
        PreparedStatement ps1 = connection.getPrepared(sqlpessoa);
        ps.execute();
        ps1.execute();
        connection.closeConnection();
        //connection.closeResult(ps);
        connection.closeStatement(sqljuridica);
    }
}

```

- Classe CadastroBDTeste

```

import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaJuridica;
import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;
import cadastro.model.PessoaFisicaDAO;
import cadastro.model.PessoaJuridicaDAO;
import java.util.List;

public class CadastroBDTeste {

    public static void main(String[] args)throws Exception {
        // a. Instanciar uma pessoa física e persistir no banco de dados
        //Instanciando a sequencia
        SequenceManager seq = new SequenceManager();

        PessoaFisica pessoaIncluir = new
PessoaFisica(seq.getValue("seq_Pessoa"),"Gregorio", "Rua 360, Centro",
"Recife", "PE", "1212-1212","Gregorio@recife.com","98765432112");
        PessoaFisicaDAO pessoaPF = new PessoaFisicaDAO();
        pessoaPF.incluir(pessoaIncluir);

        // b.Alterar os dados da pessoa física no banco.
        // Alterando pessoa e pessoaFisica pelo id--> 3 . Mudando nome, cpf e
telefone
        PessoaFisicaDAO pessoaPF1 = new PessoaFisicaDAO();
        pessoaPF1.alterar( 3, "12345678998", "Emerson Gregorio", "", "", "",
"123456789", "");

        // c.Consultar todas as pessoas físicas do banco de dados e listar no
console.
        // Retorno de todas as pessoas físicas do banco de dados
        System.out.println("Pessoas fisicas:");
        PessoaFisicaDAO pessoasPF = new PessoaFisicaDAO();
        List<PessoaFisica> resultado = pessoasPF.getPessoas();
        for (PessoaFisica pessoaFisica : resultado) {
            pessoaFisica.exibir();
        }

        //d. Excluir a pessoa física criada anteriormente no banco.
        // Excluindo pessoaFisica e Pessoa pelo id.
        PessoaFisicaDAO pessoaPF2 = new PessoaFisicaDAO();
        pessoaPF2.excluir(3);

        // e.Instanciar uma pessoa jurídica e persistir no banco de dados.

        //Incluir pessoa juridica e pessoa
        PessoaJuridica pessoaIncluir2 = new
PessoaJuridica(seq.getValue("seq_Pessoa"),"GREG LTDA",
"Rua Gregorio, Centro","Maceio", "AL", "9898-
9898","GREGLTDA@maceio.com","55555555555555");
        PessoaJuridicaDAO pessoaPJ = new PessoaJuridicaDAO();
        pessoaPJ.incluir(pessoaIncluir2);
    }
}

```

```

// f.Alterar os dados da pessoa jurídica no banco.

// Alterando pessoa e pessoaJuridica pelo id--> 4 . Mudando nome e cnpj
PessoaJuridicaDAO pessoaPJ2 = new PessoaJuridicaDAO();
pessoaPJ2.alterar( 4, "99999999999999", "Gregorio LTDA ", "", "", "",
"", "");

// g.Consultar todas as pessoas jurídicas do banco e listar no console.

// Retorno de todas as pessoas juridicas do banco de dados
System.out.println("Pessoas juridicas:");
PessoaJuridicaDAO pessoasPJ = new PessoaJuridicaDAO();
List<PessoaJuridica> resultado2 = pessoasPJ.getPessoas();
for (PessoaJuridica pessoaJuridica : resultado2) {
    pessoaJuridica.exibir();
}

// h.Excluir a pessoa jurídica criada anteriormente no banco.

// Excluindo pessoa juridida e Pessoa pelo id.
PessoaJuridicaDAO pessoaPJ3 = new PessoaJuridicaDAO();
pessoaPJ3.excluir(4);

}
}

```

2º Procedimento | Alimentando a Base

- Classe CadastroBDTeste2

```

import cadastro.model.PessoaFisicaDAO;
import cadastro.model.PessoaJuridicaDAO;
import cadastro.model.util.SequenceManager;
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaJuridica;
import java.util.List;
import java.util.Scanner;

public class CadastroBDTeste2 {

    public static void main(String[] args) throws Exception {

        Scanner scan = new Scanner(System.in);
        String escolha;

        do {
            System.out.println("=====");
            System.out.println("1 - Incluir Pessoa");
            System.out.println("2 - Alterar Pessoa");
            System.out.println("3 - Excluir Pessoa");
            System.out.println("4 - Buscar pelo Id");
            System.out.println("5 - Exibir Todos");
            System.out.println("0 - Finalizar Programa");
            System.out.println("=====");

            escolha = scan.next();
            SequenceManager seq = new SequenceManager();

            switch (escolha) {

                // Incluir
                case "1":
                    do {
                        System.out.println("=====");
                        System.out.println("F - Pessoa Fisica | J - Pessoa
Juridica | M - Menu");

                        escolha = scan.next();
                        scan.nextLine();

                        switch (escolha.toUpperCase()) {

                            case "F":
                                System.out.println("Insira os dados... ");
                                System.out.print("Nome: ");
                                String nome = scan.nextLine();
                                System.out.print("Logradouro: ");
                                String logradouro = scan.nextLine();
                                System.out.print("Cidade: ");
                                String cidade = scan.nextLine();
                                System.out.print("Estado: ");
                                String estado = scan.nextLine();
                                System.out.print("Telefone: ");
                                String telefone = scan.nextLine();
                                System.out.print("Email: ");
                                String email = scan.nextLine();
                                System.out.print("CPF: ");

```

```

        String cpf = scan.nextLine();

        PessoaFisica pessoaIncluir = new
PessoaFisica(seq.getValue("seq_Pessoa"),nome, logradouro,
        cidade, estado, telefone,email,cpf);
        PessoaFisicaDAO pessoaPF = new
PessoaFisicaDAO();

        pessoaPF.incluir(pessoaIncluir);

        System.out.println("Inclusao realizada com
sucesso!");

        break;

    case "J":
        System.out.println("Insira os dados... ");
        System.out.print("Nome: ");
        String nomej = scan.nextLine();
        System.out.print("Logradouro: ");
        String logradouroj = scan.nextLine();
        System.out.print("Cidade: ");
        String cidadej = scan.nextLine();
        System.out.print("Estado: ");
        String estadoj = scan.nextLine();
        System.out.print("Telefone: ");
        String telefonej = scan.nextLine();
        System.out.print("Email: ");
        String emailj = scan.nextLine();
        System.out.print("CNPJ: ");
        String cnpj = scan.nextLine();

        PessoaJuridica pessoaJIncluir = new
PessoaJuridica(seq.getValue("seq_Pessoa"),nomej,
        logradouroj,cidadej, estadoj,
        telefonej,emailj,cnpj);
        PessoaJuridicaDAO pessoaPJ = new
PessoaJuridicaDAO();

        pessoaPJ.incluir(pessoaJIncluir);

        System.out.println("Inclusao realizada com
sucesso!");

        break;

    case "M":
        break;

    default:
        System.out.println("Opcao invalida.");
        break;
    }
} while (!escolha.equalsIgnoreCase("M"));
break;

// Alterar
case "2":
do {
    System.out.println("=====");
    System.out.println("F - Pessoa Fisica | J - Pessoa

```



```

Juridica | M - Menu");

        escolha = scan.next();
        scan.nextLine();

        switch (escolha.toUpperCase()) {

            case "F":
                System.out.println("Digite o ID da pessoa: ");
                int idPessoaFisica = scan.nextInt();
                scan.nextLine();

                PessoaFisica pessoaFisicaLocalizada = new
PessoaFisicaDAO().getPessoa(idPessoaFisica);
                PessoaFisicaDAO pessoaFisicaLocalizadaAlterar =
new PessoaFisicaDAO();
                //PessoaFisica pessoaFisicaLocalizada =
pfRepo.obter(idPessoaFisica);

                if (pessoaFisicaLocalizada != null) {
                    pessoaFisicaLocalizada.exibir();

                    System.out.println("Nome atual: " +
pessoaFisicaLocalizada.getNome());
                    System.out.print("Novo nome: ");
                    String novoNome = scan.nextLine();

                    System.out.println("Logradouro: " +
pessoaFisicaLocalizada.getLogradouro());
                    System.out.print("Novo Logradouro: ");
                    String novoLogradouro = scan.nextLine();

                    System.out.println("Cidade: " +
pessoaFisicaLocalizada.getCidade());
                    System.out.print("Nova Cidade: ");
                    String novoCidade = scan.nextLine();

                    System.out.println("Estado: " +
pessoaFisicaLocalizada.getEstado());
                    System.out.print("Novo Estado: ");
                    String novoEstado = scan.nextLine();

                    System.out.println("Telefone: " +
pessoaFisicaLocalizada.getTelefone());
                    System.out.print("Novo Telefone: ");
                    String novoTelefone = scan.nextLine();

                    System.out.println("Email: " +
pessoaFisicaLocalizada.getEmail());
                    System.out.print("Novo Email: ");
                    String novoEmail = scan.nextLine();

                    System.out.println("CPF atual: " +
pessoaFisicaLocalizada.getCpf());
                    System.out.print("Novo CPF: ");
                    String novoCPF = scan.nextLine();

```

```

        pessoaFisicaLocalizadaAlterar.alterar(
idPessoaFisica,novoCPF, novoNome, novoLogradouro, novoCidade,
        novoEstado, novoTelefone, novoEmail );

        System.out.println("Pessoa alterada com
sucesso!");
    } else
        System.out.println("Pessoa nao localizada!
");
        break;

    case "J":
        System.out.println("Digite o ID da pessoa: ");
        int idPessoaJuridica = scan.nextInt();
        scan.nextLine();

        PessoaJuridica pessoaJuridicaLocalizada = new
PessoaJuridicaDAO().getPessoa(idPessoaJuridica);
        PessoaJuridicaDAO pessoaJuridicaLocalizadaAlterar
= new PessoaJuridicaDAO();

        if (pessoaJuridicaLocalizada != null) {
            pessoaJuridicaLocalizada.exibir();

            System.out.println("Nome atual: " +
pessoaJuridicaLocalizada.getNome());
            System.out.print("Novo nome: ");
            String novoNome = scan.nextLine();

            System.out.println("Logradouro: " +
pessoaJuridicaLocalizada.getLogradouro());
            System.out.print("Novo Logradouro: ");
            String novoLogradouro = scan.nextLine();

            System.out.println("Cidade: " +
pessoaJuridicaLocalizada.getCidade());
            System.out.print("Nova Cidade: ");
            String novoCidade = scan.nextLine();

            System.out.println("Estado: " +
pessoaJuridicaLocalizada.getEstado());
            System.out.print("Novo Estado: ");
            String novoEstado = scan.nextLine();

            System.out.println("Telefone: " +
pessoaJuridicaLocalizada.getTelefone());
            System.out.print("Novo Telefone: ");
            String novoTelefone = scan.nextLine();

            System.out.println("Email: " +
pessoaJuridicaLocalizada.getEmail());
            System.out.print("Novo Email: ");
            String novoEmail = scan.nextLine();

            System.out.println("CNPJ atual: " +

```

```

pessoaJuridicaLocalizada.getCnpj());
        System.out.print("Novo CNPJ: ");
        String novoCNPJ = scan.nextLine();

        pessoaJuridicaLocalizadaAlterar.alterar(
idPessoaJuridica, novoCNPJ, novoNome, novoLogradouro, novoCidade,
        novoEstado, novoTelefone, novoEmail);

        System.out.println("Pessoa alterada com
sucesso!");
    } else
        System.out.println("Pessoa nao
localizada!");
        break;

    case "M":
        break;

    default:
        System.out.println("Opcao invalida.");
        break;
    }
} while (!escolha.equalsIgnoreCase("M"));
break;

// EXCLUIR
case "3":
    do {
        System.out.println("=====");
        System.out.println("F - Pessoa Fisica | J - Pessoa
Juridica | M - Menu");

        escolha = scan.next();
        scan.nextLine();

        switch (escolha.toUpperCase()) {

            case "F":
                System.out.println("Digite o ID da pessoa: ");
                int idPessoaFisica = scan.nextInt();

                PessoaFisica pessoaFisicaLocalizada = new
PessoaFisicaDAO().getPessoa(idPessoaFisica);
                PessoaFisicaDAO pessoaFisicaLocalizadaExcluir =
new PessoaFisicaDAO();

                if (pessoaFisicaLocalizada != null) {
                    pessoaFisicaLocalizada.exibir();

                    pessoaFisicaLocalizadaExcluir.excluir(idPessoaFisica);

                    System.out.println("Pessoa excluida com
sucesso!");
                } else
                    System.out.println("Pessoa nao
localizada!");

```

```

        break;

        case "J":
            System.out.println("Digite o ID da pessoa: ");
            int idPessoaJuridica = scan.nextInt();

            PessoaJuridica pessoaJuridicaLocalizada = new
PessoaJuridicaDAO().getPessoa(idPessoaJuridica);
            PessoaJuridicaDAO pessoaJuridicaLocalizadaExcluir
= new PessoaJuridicaDAO();

            if (pessoaJuridicaLocalizada != null) {
                pessoaJuridicaLocalizada.exibir();

pessoaJuridicaLocalizadaExcluir.excluir(idPessoaJuridica);

                System.out.println("Pessoa excluida com
sucesso!");
            } else
                System.out.println("Pessoa nao
localizada!");

            break;

        case "M":
            break;

        default:
            System.out.println("Opcao invalida.");
            break;
    }

    } while (!escolha.equalsIgnoreCase("M"));
    break;

    // obter pelo Id
    case "4":
        do {
            System.out.println("=====");
            System.out.println("F - Pessoa Fisica | J - Pessoa
Juridica | M - Menu");

            escolha = scan.next();
            scan.nextLine();

            switch (escolha.toUpperCase()) {

                case "F":
                    System.out.println("Digite o ID da pessoa: ");
                    int idPessoaFisica = scan.nextInt();

                    PessoaFisica pessoaFisicaLocalizada = new
PessoaFisicaDAO().getPessoa(idPessoaFisica);

                    if (pessoaFisicaLocalizada != null) {
                        System.out.println("Pessoa localizada!");

```

```

        pessoaFisicaLocalizada.exibir();
    } else
        System.out.println("Pessoa nao
localizada!");

    break;

    case "J":
        System.out.println("Digite o ID da pessoa: ");
        int idPessoaJuridica = scan.nextInt();

        PessoaJuridica pessoaJuridicaLocalizada = new
PessoaJuridicaDAO().getPessoa(idPessoaJuridica);

        if (pessoaJuridicaLocalizada != null) {
            System.out.println("Pessoa localizada!");
            pessoaJuridicaLocalizada.exibir();
        } else
            System.out.println("Pessoa nao
localizada!");

        break;

    case "M":
        break;

    default:
        System.out.println("Opcao invalida.");
        break;
    }

    } while (!escolha.equalsIgnoreCase("M"));
    break;

    //obterTodos
    case "5":
        do {
            System.out.println("=====");
            System.out.println("F - Pessoa Fisica | J - Pessoa
Juridica | M - Menu");

            escolha = scan.next();
            scan.nextLine();

            switch (escolha.toUpperCase()) {

                case "F":
                    System.out.println("Pessoas fisicas:");
                    PessoaFisicaDAO pessoasFisica = new
PessoaFisicaDAO();
                    List<PessoaFisica> resultado =
pessoasFisica.getPessoas();

                    for (PessoaFisica pessoaFisica : resultado) {
                        pessoaFisica.exibir();
                    }
                    break;

                case "J":

```

```

        System.out.println("Pessoas juridicas:");
        PessoaJuridicaDAO pessoasJuridica = new
PessoaJuridicaDAO();
        List<PessoaJuridica> resultado2 =
        for (PessoaJuridica pessoaJuridica :
        pessoaJuridica.exibir();
        }
        break;

        case "M":
        break;

        default:
        System.out.println("Opcao invalida");
        break;
    }

    } while (!escolha.equalsIgnoreCase("M"));
    break;
    case "0":
    System.out.println("Sistema Finalizado com sucesso.");
    break;

    default:
    System.out.println("Opcao invalida");
    break;
    }
    } while (!escolha.equals("0"));
    scan.close();
}
}

```

3. Análise e Conclusão

1º Procedimento:

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

O JDBC permite a comunicação entre aplicações Java e bancos de dados, abstraindo detalhes de conexão e execução de queries. Ele padroniza o acesso a dados, facilitando a portabilidade entre diferentes SGBDs.

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

- **Statement:** Executa queries estáticas, vulneráveis a SQL injection e menos eficiente para queries repetidas.
- **PreparedStatement:** Pré-compila queries, permite parâmetros dinâmicos e evita SQL injection e é mais eficiente para execuções repetidas.

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

O padrão DAO melhora a manutenibilidade do software ao separar a lógica de acesso a dados da lógica de negócios, centralizando operações de CRUD em componentes reutilizáveis e isolados, o que facilita modificações no banco de dados sem impactar outras camadas.

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

Em um modelo estritamente relacional, a herança é refletida por meio de estratégias como:

- Table per Hierarchy: uma única tabela com uma coluna discriminadora e todos os atributos das subclasses.
- Table per Class: uma tabela para cada classe, inclusive a superclasse, com joins necessários para reconstruir os objetos.
- Table per Concrete Class: uma tabela para cada subclasse contendo todos os atributos, inclusive os herdados.

Cada abordagem possui vantagens e desvantagens em termos de normalização, desempenho e complexidade das consultas.

2º Procedimento

a) 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 simples, geralmente no formato texto ou binário, sendo adequada para aplicações pequenas e com baixa complexidade, mas com limitações quanto à segurança, concorrência e organização dos dados. Já a persistência em banco de dados oferece uma estrutura mais robusta, permitindo acesso eficiente, consultas complexas, controle de

concorrência, integridade referencial e escalabilidade, o que a torna mais adequada para aplicações de médio a grande porte.

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

O uso do operador lambda em Java permite escrever funções anônimas de forma concisa, facilitando a manipulação de coleções com a API de Streams. Com isso, a impressão dos valores das entidades pode ser feita de forma mais simples e legível.

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

Métodos acionados diretamente pelo main precisam ser static porque o método main também é estático e, por definição, métodos estáticos pertencem à classe, e não a uma instância dela. Como não há objeto instanciado naquele momento, só é possível chamar outros métodos que também pertençam à classe, ou seja, que sejam marcados como static.