Campos : Polo Barbosa II - Marília - SP
Curso : Desenvolvedor Full Stack
Turma : 2023.2
Integrante : Rafael Leal Altero

1º Procedimento :

Mapeamento Objeto-Relacional e DAO

Objetivo da Prática :

1 - Integração com Banco de Dados: A prática visa ensinar como criar uma aplicação Java que se conecta a um banco de dados SQL Server. Onde é necessário armazenar e recuperar dados persistentes.

2 - Mapeamento Objeto-Relacional: Ao criar classes como Pessoa, PessoaFisica, e PessoaJuridica, nós mapeamos as estruturas de objetos em estruturas de tabelas em um banco de dados relacional. Isso é importante para tornar a comunicação entre a camada de aplicação e o banco de dados mais eficiente e coeso.

3 - ConnectorBD : A conexão estabelecida com o banco de dados SQL server usando os conectores do JDBC, foi feita de forma separada para a abstração desses componentes.

4 - Padrão DAO e Operações CRUD: A implementação do padrão DAO ajuda a separar a lógica de acesso ao banco de dados da lógica de negócios principal da aplicação.Crud é fundamental para qualquer sistema que precise gerenciar dados persistentes.

segue o ***LINK***👇 **Codigos**

Códigos solicitados no roteiro de aula:

Link GetHub :
https://github.com/Rafa1a/CadastroBD_SQL/tree/main/src

Código e Resultados da execução dos códigos:

cadastrobd.model:

Class Pessoa :

```
public class Pessoa {
    private int id;
    private String nome, logradouro, cidade, estado,
telefone, email;
        public Pessoa() {
        this.id = 0;
        this.nome = "";
        this.logradouro = "";
        this.cidade = "";
        this.estado = "";
        this.telefone = "";
        this.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;
        }

        //GET SET ID
```

```java
    public int getId(){
        return this.id;
    }

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

    // GET SET nome
    public String getNome(){
        return this.nome;
    }
    public void setNome(String nome){
        this.nome = nome;
    }

    // GET SET logradouro
    public String getLogradouro() {
        return this.logradouro;
    }
    public void setLogradouro (String logradouro) {
        this.logradouro = logradouro;
    }

    //GET SET logradouro
    public String getCidade(){
        return this.cidade;
    }
    public void setCidade (String cidade){
        this.cidade = cidade;
    }

    //GET SET estado
    public String getEstado() {
        return this.estado;
    }
    public void setEstado(String estado) {
        this.estado = estado;
    }
```

```
        //GET SET telefone
        public String getTelefone(){
            return this.telefone;
        }
        public void setTelefone(String telefone){
            this.telefone = telefone;
        }

        //GET SET email
        public String getEmail(){
            return this.email;
        }
        public void setEmail(String email){
            this.email = email;
        }

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

        }
}
```

Class PessoaJuridica :

```
public class PessoaJuridica extends Pessoa{
    private String cnpj;

    public PessoaJuridica(){
        super();
        this.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;
    }
    //Get set Pessoa Fisica
    public String getCnpj(){
        return this.cnpj;
    }
    public void setCnpj(String cnpj){
        this.cnpj = cnpj;
    }

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

System.out.println("=====================================")
;
    }
}
```

Class PessoaFisica :

```
public class PessoaFisica extends Pessoa {
    private String cpf;

    public PessoaFisica(){
        super();
        this.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;
    }
    //Get set Pessoa Fisica
    public String getCpf(){
        return this.cpf;
    }
    public void setCpf(String cpf){
        this.cpf = cpf;
    }

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

System.out.println("=======================================")
;

    }

}
```

cadastro.model.util :

Class ConectorBD :

```
package cadastro.model.util;

/**
 *
 * @author Windows 10
 */
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=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(String sql) throws
SQLException {
        return getPrepared(sql).executeQuery();
    }

    public static void closeConnection(Connection
connection) {
        try {
            if (connection != null) {
                connection.close();
            }
        } catch (SQLException e) {
            System.out.println(e);
        }
    }
```

```
    public static void closeStatement(Statement statement)
{
        try {
            if (statement != null) {
                statement.close();
            }
        } catch (SQLException e) {
            System.out.println(e);
        }
    }
     public static void closeResultset(ResultSet resultSet)
{
        try {
            if (resultSet != null) {
                resultSet.close();
            }
        } catch (SQLException e) {
            System.out.println(e);
        }
    }
}
```

Class SequenceManager :

```
package cadastro.model.util;

/**
 *
 * @author Windows 10
 */

import java.sql.ResultSet;
import java.sql.SQLException;
public class SequenceManager {
    private static final String sqln = "SELECT NEXT VALUE FOR
sequencia";
```

```
    public static int getValue() throws SQLException {

        ResultSet resultSet = ConectorBD.getSelect(sqln);

        int nextValue = 0;
        if (resultSet.next()) {
            nextValue = resultSet.getInt(1);
        }
        ConectorBD.closeResultset(resultSet);

        return nextValue;
    }

}
```

cadastro.modelDAO :

Class PessoaFisicaDAO :

```
package cadastro.modelDAO;
import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;
import cadastrobd.model.PessoaFisica;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
/**
 *
 * @author Windows 10
 */
public class PessoaFisicaDAO  {
    private static final String TABLE_PESSOA =
"DBM.Pessoa";
```

```
    private static final String TABLE_NAME =
"DBM.Pessoa_fisica";
    private static final String ID_COLUMN_NAME =
"id_pessoa";
    private static final String NOME_COLUMN_NAME =
"nome";
    private static final String LOGRADOURO_COLUMN_NAME =
"logradouro";
    private static final String CIDADE_COLUMN_NAME =
"cidade";
    private static final String ESTADO_COLUMN_NAME =
"estado";
    private static final String TELEFONE_COLUMN_NAME =
"telefone";
    private static final String EMAIL_COLUMN_NAME =
"email";
    private static final String CPF_COLUMN_NAME =
"id_cpf";

    public PessoaFisica getPessoa(int id) throws
SQLException{

        ResultSet resultPf = ConectorBD.getSelect("SELECT
* FROM " + TABLE_NAME + " WHERE " + ID_COLUMN_NAME + " =
" + id);
        ResultSet resultP = ConectorBD.getSelect("SELECT
* FROM " + TABLE_PESSOA + " WHERE " + ID_COLUMN_NAME + "
= " + id);

        PessoaFisica pessoafisica=null;

        if(resultP.next() &&  resultPf.next()){
            int Id = resultP.getInt(ID_COLUMN_NAME);
            String nome =
resultP.getString(NOME_COLUMN_NAME);
            String logradouro =
```

```
resultP.getString(LOGRADOURO_COLUMN_NAME);
            String cidade =
resultP.getString(CIDADE_COLUMN_NAME);
            String estado =
resultP.getString(ESTADO_COLUMN_NAME);
            String telefone =
resultP.getString(TELEFONE_COLUMN_NAME);
            String email =
resultP.getString(EMAIL_COLUMN_NAME);
            String cpf =
resultPf.getString(CPF_COLUMN_NAME);

            // Construa o objeto PessoaFisica usando os
valores obtidos
            pessoafisica = new PessoaFisica(Id, nome,
logradouro, cidade, estado, telefone, email,cpf);
        }else {
            System.out.println("Pessoa nao cadastrada no
cpf");
        }

        // Feche o ResultSet
        ConectorBD.closeResultset(resultPf);
        ConectorBD.closeResultset(resultP);
        return pessoafisica;
    }

    public List<PessoaFisica> getPessoas() throws
SQLException{
        ResultSet resultPf = ConectorBD.getSelect("SELECT
* FROM " + TABLE_NAME);
        ResultSet resultP = ConectorBD.getSelect("SELECT
* FROM " + TABLE_PESSOA);

        List<PessoaFisica> allpessoas = new
ArrayList<>();
```

```
        List<PessoaFisica> pessoasFisicas = new
ArrayList<>();
        while (resultP.next()) {
            int Id = resultP.getInt(ID_COLUMN_NAME);
            String nome =
resultP.getString(NOME_COLUMN_NAME);
            String logradouro =
resultP.getString(LOGRADOURO_COLUMN_NAME);
            String cidade =
resultP.getString(CIDADE_COLUMN_NAME);
            String estado =
resultP.getString(ESTADO_COLUMN_NAME);
            String telefone =
resultP.getString(TELEFONE_COLUMN_NAME);
            String email =
resultP.getString(EMAIL_COLUMN_NAME);
            String cpf = "null";

            PessoaFisica pessoas = new PessoaFisica(Id,
nome, logradouro, cidade, estado, telefone, email, cpf);
            allpessoas.add(pessoas);
        }

        while (resultPf.next()) {
            int Id = resultPf.getInt(ID_COLUMN_NAME);
            String cpf =
resultPf.getString(CPF_COLUMN_NAME);

            for (PessoaFisica pessoa : allpessoas) {
                if (pessoa.getId() == Id) {
                    pessoa.setCpf(cpf);
                    pessoasFisicas.add(pessoa);
                    break;
                }
            }
        }
```

```
    // Feche os ResultSets
    ConectorBD.closeResultset(resultPf);
    ConectorBD.closeResultset(resultP);
    return pessoasFisicas;

    }

    public void incluir (PessoaFisica pessoafisica)
throws SQLException{

        PreparedStatement statementP =
ConectorBD.getPrepared("INSERT INTO " + TABLE_PESSOA + "
(id_pessoa,nome, logradouro, cidade,
estado,telefone,email) VALUES (?,?,?,?,?,?,?)");
        int id = SequenceManager.getValue();

        statementP.setInt(1, id);
        statementP.setString(2, pessoafisica.getNome());
        statementP.setString(3,
pessoafisica.getLogradouro());
        statementP.setString(4,
pessoafisica.getCidade());
        statementP.setString(5,
pessoafisica.getEstado());
        statementP.setString(6,
pessoafisica.getTelefone());
        statementP.setString(7, pessoafisica.getEmail());

        PreparedStatement statementPf =
ConectorBD.getPrepared("INSERT INTO " + TABLE_NAME + "
(id_cpf,id_pessoa) VALUES (?,?)");
        statementPf.setString(1,pessoafisica.getCpf());
        statementPf.setInt(2,id);

        statementP.executeUpdate();
```

```
        statementPf.executeUpdate();
        ConectorBD.closeStatement(statementP);
        ConectorBD.closeStatement(statementPf);
    }

    public void alterar (PessoaFisica pessoafisica)
throws SQLException {

         PreparedStatement statementP =
ConectorBD.getPrepared("UPDATE " + TABLE_PESSOA + " SET
nome=?, logradouro=?, cidade=?, estado=?, telefone=?,
email=? WHERE " + ID_COLUMN_NAME + "=?");
    PreparedStatement statementPf =
ConectorBD.getPrepared("UPDATE " + TABLE_NAME + " SET
id_cpf=? WHERE " + ID_COLUMN_NAME + "=?");

    statementP.setString(1, pessoafisica.getNome());
    statementP.setString(2,
pessoafisica.getLogradouro());
    statementP.setString(3, pessoafisica.getCidade());
    statementP.setString(4, pessoafisica.getEstado());
    statementP.setString(5, pessoafisica.getTelefone());
    statementP.setString(6, pessoafisica.getEmail());
    statementP.setInt(7, pessoafisica.getId()); // Aqui
definimos o ID do registro a ser atualizado

    // Tabela pessoafisica
    statementPf.setString(1, pessoafisica.getCpf());
    statementPf.setInt(2, pessoafisica.getId()); // Aqui
definimos o ID do registro a ser atualizado

    statementP.executeUpdate();
    statementPf.executeUpdate();

    ConectorBD.closeStatement(statementP);
    ConectorBD.closeStatement(statementPf);
```

```
}

    public void excluir(int id) throws SQLException{
        PreparedStatement statementPF =
ConectorBD.getPrepared("DELETE FROM " + TABLE_NAME + "
WHERE id_pessoa = ?");
        PreparedStatement statementP =
ConectorBD.getPrepared("DELETE FROM " + TABLE_PESSOA + "
WHERE id_pessoa = ?");
        statementPF.setInt(1, id);
        statementP.setInt(1, id);
        statementPF.executeUpdate();
        statementP.executeUpdate();
        ConectorBD.closeStatement(statementPF);
        ConectorBD.closeStatement(statementP);
    }

}
```

Classe PessoaJuridicaDAO :

```
package cadastro.modelDAO;
import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;
import cadastrobd.model.PessoaJuridica;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
/**
 *
 * @author Windows 10
 */
```

```
public class PessoaJuridicaDAO  {
    private static final String TABLE_PESSOA =
"DBM.Pessoa";
    private static final String TABLE_NAME =
"DBM.Pessoa_juridica";
    private static final String ID_COLUMN_NAME =
"id_pessoa";
    private static final String NOME_COLUMN_NAME =
"nome";
    private static final String LOGRADOURO_COLUMN_NAME =
"logradouro";
    private static final String CIDADE_COLUMN_NAME =
"cidade";
    private static final String ESTADO_COLUMN_NAME =
"estado";
    private static final String TELEFONE_COLUMN_NAME =
"telefone";
    private static final String EMAIL_COLUMN_NAME =
"email";
    private static final String CNPJ_COLUMN_NAME =
"id_cnpj";

    public PessoaJuridica getPessoa(int id) throws
SQLException{

        ResultSet resultPf = ConectorBD.getSelect("SELECT
* FROM " + TABLE_NAME + " WHERE " + ID_COLUMN_NAME + " =
" + id);
        ResultSet resultP = ConectorBD.getSelect("SELECT
* FROM " + TABLE_PESSOA + " WHERE " + ID_COLUMN_NAME + "
= " + id);

        PessoaJuridica pessoajuridica=null;

        if(resultP.next() &&  resultPf.next()){
            int Id = resultP.getInt(ID_COLUMN_NAME);
```

```
            String nome =
resultP.getString(NOME_COLUMN_NAME);
            String logradouro =
resultP.getString(LOGRADOURO_COLUMN_NAME);
            String cidade =
resultP.getString(CIDADE_COLUMN_NAME);
            String estado =
resultP.getString(ESTADO_COLUMN_NAME);
            String telefone =
resultP.getString(TELEFONE_COLUMN_NAME);
            String email =
resultP.getString(EMAIL_COLUMN_NAME);
            String cnpj =
resultPf.getString(CNPJ_COLUMN_NAME);

            // Construa o objeto PessoaFisica usando os
valores obtidos
            pessoajuridica = new PessoaJuridica(Id, nome,
logradouro, cidade, estado, telefone, email,cnpj);
        }else {
            System.out.println("Pessoa nao cadastrada no
cnpj");
        }

        // Feche o ResultSet
        ConectorBD.closeResultset(resultPf);
        ConectorBD.closeResultset(resultP);
        return pessoajuridica;
    }

    public List<PessoaJuridica> getPessoas() throws
SQLException{
        ResultSet resultPf = ConectorBD.getSelect("SELECT
* FROM " + TABLE_NAME);
        ResultSet resultP = ConectorBD.getSelect("SELECT
* FROM " + TABLE_PESSOA);
```

```
        List<PessoaJuridica> allpessoas = new
ArrayList<>();
        List<PessoaJuridica> pessoajuridicas = new
ArrayList<>();
        while (resultP.next()) {
            int Id = resultP.getInt(ID_COLUMN_NAME);
            String nome =
resultP.getString(NOME_COLUMN_NAME);
            String logradouro =
resultP.getString(LOGRADOURO_COLUMN_NAME);
            String cidade =
resultP.getString(CIDADE_COLUMN_NAME);
            String estado =
resultP.getString(ESTADO_COLUMN_NAME);
            String telefone =
resultP.getString(TELEFONE_COLUMN_NAME);
            String email =
resultP.getString(EMAIL_COLUMN_NAME);
            String cnpj = "null";

            PessoaJuridica pessoas = new
PessoaJuridica(Id, nome, logradouro, cidade, estado,
telefone, email, cnpj);
            allpessoas.add(pessoas);
        }

        while (resultPf.next()) {
            int Id = resultPf.getInt(ID_COLUMN_NAME);
            String cnpj =
resultPf.getString(CNPJ_COLUMN_NAME);

            for (PessoaJuridica pessoa : allpessoas) {
                if (pessoa.getId() == Id) {
                    pessoa.setCnpj(cnpj);
                    pessoajuridicas.add(pessoa);
```

```
                    break;
                }
            }
        }

        // Feche os ResultSets
        ConectorBD.closeResultset(resultPf);
        ConectorBD.closeResultset(resultP);
        return pessoajuridicas;

    }

    public void incluir (PessoaJuridica pessoajuridica)
throws SQLException{

        PreparedStatement statementP =
ConectorBD.getPrepared("INSERT INTO " + TABLE_PESSOA + "
(id_pessoa,nome, logradouro, cidade,
estado,telefone,email) VALUES (?,?,?,?,?,?,?)");
        int id = SequenceManager.getValue();

        statementP.setInt(1, id);
        statementP.setString(2,
pessoajuridica.getNome());
        statementP.setString(3,
pessoajuridica.getLogradouro());
        statementP.setString(4,
pessoajuridica.getCidade());
        statementP.setString(5,
pessoajuridica.getEstado());
        statementP.setString(6,
pessoajuridica.getTelefone());
        statementP.setString(7,
pessoajuridica.getEmail());

        PreparedStatement statementPf =
```

```
ConectorBD.getPrepared("INSERT INTO " + TABLE_NAME + "
(id_cnpj,id_pessoa) VALUES (?,?)");

statementPf.setString(1,pessoajuridica.getCnpj());
        statementPf.setInt(2,id);

        statementP.executeUpdate();
        statementPf.executeUpdate();
        ConectorBD.closeStatement(statementP);
        ConectorBD.closeStatement(statementPf);
    }

    public void alterar (PessoaJuridica pessoajuridica)
throws SQLException {

        PreparedStatement statementP =
ConectorBD.getPrepared("UPDATE " + TABLE_PESSOA + " SET
nome=?, logradouro=?, cidade=?, estado=?, telefone=?,
email=? WHERE " + ID_COLUMN_NAME + "=?");
        PreparedStatement statementPf =
ConectorBD.getPrepared("UPDATE " + TABLE_NAME + " SET
id_cnpj=? WHERE " + ID_COLUMN_NAME + "=?");

        statementP.setString(1,
pessoajuridica.getNome());
        statementP.setString(2,
pessoajuridica.getLogradouro());
        statementP.setString(3,
pessoajuridica.getCidade());
        statementP.setString(4,
pessoajuridica.getEstado());
        statementP.setString(5,
pessoajuridica.getTelefone());
        statementP.setString(6,
pessoajuridica.getEmail());
        statementP.setInt(7, pessoajuridica.getId()); //
```

```
Aqui definimos o ID do registro a ser atualizado

        // Tabela pessoafisica
        statementPf.setString(1,
pessoajuridica.getCnpj());
        statementPf.setInt(2, pessoajuridica.getId()); //
Aqui definimos o ID do registro a ser atualizado

        statementP.executeUpdate();
        statementPf.executeUpdate();

        ConectorBD.closeStatement(statementP);
        ConectorBD.closeStatement(statementPf);
}

    public void excluir(int id) throws SQLException{
        PreparedStatement statementPF =
ConectorBD.getPrepared("DELETE FROM " + TABLE_NAME + "
WHERE id_pessoa = ?");
        PreparedStatement statementP =
ConectorBD.getPrepared("DELETE FROM " + TABLE_PESSOA + "
WHERE id_pessoa = ?");
        statementPF.setInt(1, id);
        statementP.setInt(1, id);
        statementPF.executeUpdate();
        statementP.executeUpdate();
        ConectorBD.closeStatement(statementPF);
        ConectorBD.closeStatement(statementP);
    }

}
```

CadastroBDTeste :

Teste no MAINtest do Código :

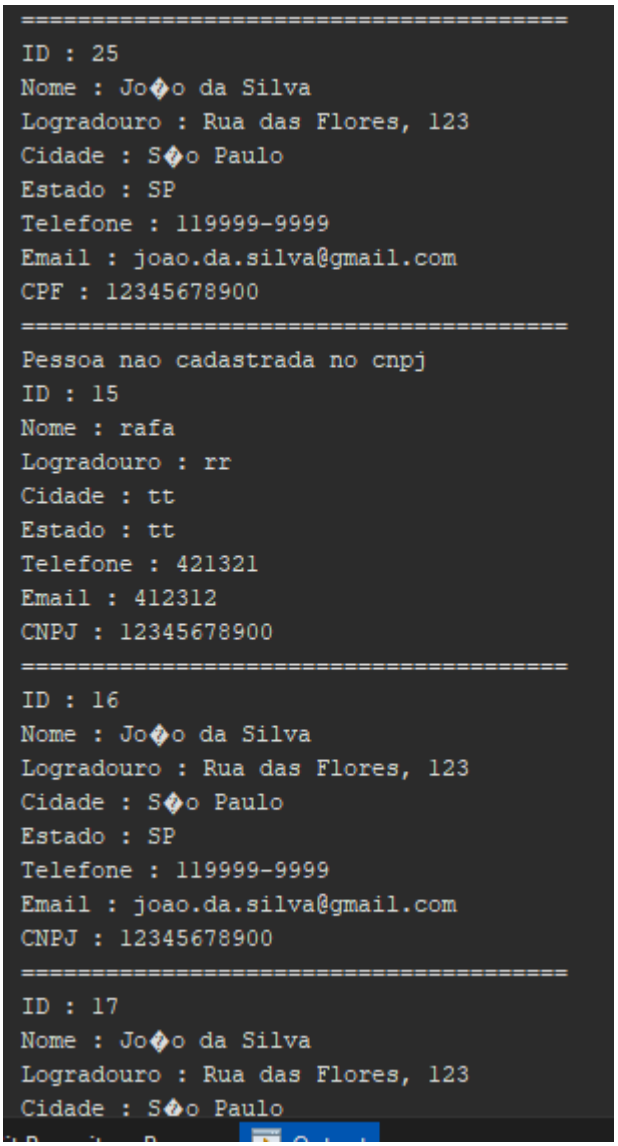
```
package cadastrobd;
import cadastro.modelDAO.PessoaFisicaDAO;
import cadastro.modelDAO.PessoaJuridicaDAO;
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaJuridica;
import java.sql.SQLException;
import java.util.List;
/**
 *
 * @author Windows 10
 */
public class CadastroBDTest {
    public static void main(String[] args)  {
            try{
             // INCLUIR CPF
==============================================

            PessoaFisica pessoaFisica = new
PessoaFisica();
            PessoaFisicaDAO pessoaFisicaDAO = new
PessoaFisicaDAO();

            pessoaFisica.setNome("João da Silva");
            pessoaFisica.setLogradouro("Rua das Flores,
123");
            pessoaFisica.setCidade("São Paulo");
            pessoaFisica.setEstado("SP");
            pessoaFisica.setTelefone("119999-9999");

pessoaFisica.setEmail("joao.da.silva@gmail.com");
            pessoaFisica.setCpf("12345678900");
```

```
            pessoaFisicaDAO.incluir(pessoaFisica);

            //OBTEM ATRAVEZ DO ID
==================================

            // Obtém a pessoa com ID 10 do banco de dados
            PessoaFisica pessoa =
pessoaFisicaDAO.getPessoa(10);

            // Verifica se a pessoa foi encontrada antes
de imprimir o nome
            if (pessoa != null) {
                pessoa.exibir();
            }

            // OBTEM TODAS AS PESSOAS DO BANCO DE
DADOS================

               List<PessoaFisica> pessoas =
pessoaFisicaDAO.getPessoas();
            if (pessoas != null) {
                for (PessoaFisica pessoaa : pessoas) {
                    pessoaa.exibir();
                }
            }else {
                System.out.println("Nenhuma pessoa
encontrada.");
            }
           //ALTERAR
=================================================
            pessoaFisica.setId(6);//ID da pessoa q vai
alterar
            pessoaFisica.setNome("rafa");
            pessoaFisica.setLogradouro("Rua das bc,
```

```
123");
            pessoaFisica.setCidade("São pao");
            pessoaFisica.setEstado("SP");
            pessoaFisica.setTelefone("119999-9999");

pessoaFisica.setEmail("joao.da.ocu@gmaibosta.cu");
            pessoaFisica.setCpf("333333333");

            pessoaFisicaDAO.alterar(pessoaFisica);

            // EXCLUIR
==============================================

            pessoaFisicaDAO.excluir(10);

             // INCLUIR
CNPJ================================================

            PessoaJuridica pessoajuridica = new
PessoaJuridica();
            PessoaJuridicaDAO pessoajuridicaDAO = new
PessoaJuridicaDAO();

            pessoajuridica.setNome("João da Silva");
            pessoajuridica.setLogradouro("Rua das Flores,
123");
            pessoajuridica.setCidade("São Paulo");
            pessoajuridica.setEstado("SP");
            pessoajuridica.setTelefone("119999-9999");

pessoajuridica.setEmail("joao.da.silva@gmail.com");
            pessoajuridica.setCnpj("12345678900");


            pessoajuridicaDAO.incluir(pessoajuridica);
             //OBTEM ATRAVEZ DO ID
```

```
==================================

            // Obtém a pessoa com ID 10 do banco de dados

            PessoaJuridica pessoaJ =
pessoajuridicaDAO.getPessoa(12);

            // Verifica se a pessoa foi encontrada antes
de imprimir o nome
            if (pessoaJ != null) {
                pessoaJ.exibir();
            }

            // OBTEM TODAS AS PESSOAS DO BANCO DE
DADOS================

               List<PessoaJuridica> pessoasJ =
pessoajuridicaDAO.getPessoas();
            if (pessoasJ != null) {
                for (PessoaJuridica pessoaa : pessoasJ) {
                    pessoaa.exibir();
                }
            }else {
                System.out.println("Nenhuma pessoa
encontrada.");
            }
             // EXCLUIR
================================================

            pessoajuridicaDAO.excluir(13);

          }catch (SQLException e) {
            System.out.println(e);
            e.printStackTrace();
        }
    }}
```

Print do TEST :



Todas as funções foram testadas e estão devidamente funcionais.

```
INCLUIR(PessoaFisica || PessoaJuridica),
ALTERAR(PessoaFisica || PessoaJuridica),
GETPESSOA(ID), GETPESSOAS(), EXCLUIR(ID).
```

Para ver o CadastroDB original Acesse o Procedimento 2, para a implementação final.

Perguntas Feitas da Faculdade e Respondidas :

Análise e Conclusão:

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

O JDBC é um componente de middleware que fornece uma API padrão para acessar bancos de dados relacionais. Isso facilita para os desenvolvedores trabalharem com diferentes bancos de dados, pois não precisam se preocupar com a implementação específica do banco de dados.

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

Statement :

- **Usado para executar consultas SQL diretas.**
- **Requer que os valores sejam inseridos diretamente na string SQL.**
- **Pode estar vulnerável a ataques de injeção de SQL se os valores não forem tratados corretamente.**

PreparedStatement :

- **Usado para executar consultas parametrizadas.**
- **Permite a definição de parâmetros na consulta usando placeholders (?), o que ajuda a evitar ataques de injeção de SQL.**
- **Geralmente é considerado mais seguro e eficiente**

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

- **Abstração de Banco de Dados**
- **Reutilização de Código**

Manutenção Centralizada: Com o padrão DAO, as mudanças no acesso aos dados são limitadas às classes DAO relevantes. Isso facilita a manutenção, pois você não precisa procurar e modificar várias partes do código sempre que ocorrer uma mudança na estrutura ou na lógica de acesso aos dados.

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

A herança no banco de dados seria a base de relacionamento com suas chaves primárias e estrangeiras respectivamente as estrangeiras seriam as tabelas “filhas” da tabela “pai” que possui a chave primária correspondente.

Essa abordagem permite que as tabelas "filhas" herdem os atributos e relacionamentos da tabela "pai". As chaves estrangeiras nas tabelas "filhas" também garantem a integridade referencial, mantendo a consistência dos dados entre as tabelas.