



**Campus:** Polo Centro II - Guarulhos - SP

**Curso:** Desenvolvimento full stack

**Disciplina:** RPG0016 - Back-end sem banco não tem

**Turma:** 2024.3

**Aluno:** Pedro Wilson Araújo Avilar

- **Título da prática**

Missão prática | Mundo 3 | Nível 3

- **Material de apoio**

<https://sway.cloud.microsoft/s/G2pmewi1ksVR72or/embed>

- **Objetivo da prática**

Utilizar as ferramentas Microsoft SQL Server Management Studio (SSMS), Java Development Kit (JDK) e NetBeans para desenvolver um aplicativo em Java, implementando o padrão Data Access Object (DAO), com persistência de dados em um banco de dados relacional.

- **Repositório git**

<https://github.com/PedroAvilar/CadastroBD>

- **Softwares utilizados**

SSMS - <https://www.microsoft.com/pt-br/sql-server/sql-server-downloads>

JDK - <https://www.oracle.com/br/java/technologies/downloads/>

Apache NetBeans IDE 22 - <https://netbeans.apache.org/front/main/download/>

- **Script para a criação do banco de dados (Mundo 3 - Nível 2)**

Com o SQL Server o seguinte script para a criação do banco de dados:

---

--Criando o banco de dados loja

CREATE DATABASE loja;

GO

--Usando o banco de dados loja

USE loja;

GO

--Criando uma sequence para ID de Pessoa

CREATE SEQUENCE Sequencia\_IDPessoa

START WITH 1

INCREMENT BY 1;

GO

--Criando tabela Pessoas

CREATE TABLE Pessoas (

    IDPessoa INTEGER NOT NULL DEFAULT NEXT VALUE FOR

Sequencia\_IDPessoa,

    NomePessoa VARCHAR(255) NOT NULL,

    Email VARCHAR(255) NOT NULL,

    Telefone VARCHAR(11) NOT NULL,

    Logradouro VARCHAR(255) NOT NULL,

    Cidade VARCHAR(255) NOT NULL,

    Estado CHAR(2) NOT NULL,

    PRIMARY KEY(IDPessoa)

);

GO

--Criando tabela PessoasFisicas

CREATE TABLE PessoasFisicas (

    Pessoas\_IDPessoa INTEGER NOT NULL,

    CPF VARCHAR(11) NOT NULL UNIQUE,

    PRIMARY KEY(Pessoas\_IDPessoa),

    FOREIGN KEY(Pessoas\_IDPessoa) REFERENCES Pessoas(IDPessoa)

);

GO

--Criando tabela PessoasJuridicas

CREATE TABLE PessoasJuridicas (

    Pessoas\_IDPessoa INTEGER NOT NULL,

    CNPJ VARCHAR(14) NOT NULL UNIQUE,

    PRIMARY KEY(Pessoas\_IDPessoa),

    FOREIGN KEY(Pessoas\_IDPessoa) REFERENCES Pessoas(IDPessoa)

);

GO

--Criando tabela Usuarios

```
CREATE TABLE Usuarios (  
    IDUsuario INTEGER NOT NULL IDENTITY,  
    NomeUsuario VARCHAR(255) NOT NULL,  
    SenhaUsuario VARCHAR(20) NOT NULL,  
    PRIMARY KEY(IDUsuario)  
);  
GO
```

--Criando tabela Produtos

```
CREATE TABLE Produtos (  
    IDProduto INTEGER NOT NULL IDENTITY,  
    NomeProduto VARCHAR(255) NOT NULL,  
    QuantidadeProduto INTEGER NOT NULL,  
    PrecoVendaBase NUMERIC(6,2) NOT NULL,  
    PRIMARY KEY(IDProduto)  
);  
GO
```

--Criando tabela Movimentos

```
CREATE TABLE Movimentos (  
    IDMovimento INTEGER NOT NULL IDENTITY,  
    Usuarios_IDUsuario INTEGER NOT NULL,  
    Pessoas_IDPessoa INTEGER NOT NULL,  
    Produtos_IDProduto INTEGER NOT NULL,  
    Tipo CHAR(1) NOT NULL,  
    QuantidadeMovimentado INTEGER NOT NULL,  
    PrecoUnitario NUMERIC(6,2) NOT NULL,  
    PRIMARY KEY (IDMovimento),  
    FOREIGN KEY (Usuarios_IDUsuario) REFERENCES Usuarios(IDUsuario),  
    FOREIGN KEY (Pessoas_IDPessoa) REFERENCES Pessoas(IDPessoa),  
    FOREIGN KEY (Produtos_IDProduto) REFERENCES Produtos(IDProduto)  
);  
GO
```

---

- **1º Procedimento | Mapeamento Objeto-Relacional e DAO**

Com a IDE NetBeans foi criado o projeto adicionando o arquivo

mssql-jdbc-12.8.1.jre11.jar disponível como parte do arquivo zip no endereço

<https://learn.microsoft.com/pt-br/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server?view=sql-server-ver16>.

**Classe Pessoa** - disponível em "CadastroBD\src\cadastrobd\model\Pessoa.java"

package cadastrobd.model;

//Classe Pessoa

public class Pessoa {

    //Campos

    private int IDPessoa;

    private String NomePessoa;

    private String Email;

    private String Telefone;

    private String Logradouro;

    private String Cidade;

    private String Estado;

    //Construtor padrão

    public Pessoa() { }

    //Construtor completo

    public Pessoa (int IDPessoa, String NomePessoa, String Email, String Telefone,  
                    String Logradouro, String Cidade, String Estado) {

        this.IDPessoa = IDPessoa;

        this.NomePessoa = NomePessoa;

        this.Email = Email;

        this.Telefone = Telefone;

        this.Logradouro = Logradouro;

        this.Cidade = Cidade;

        this.Estado = Estado;

    }

    //Getters e setters

    public int getIDPessoa() {

        return IDPessoa;

    }

    public void setIDPessoa(int IDPessoa) {

        this.IDPessoa = IDPessoa;

```

}
public String getNomePessoa() {
    return NomePessoa;
}
public void setNomePessoa(String NomePessoa) {
    this.NomePessoa = NomePessoa;
}
public String getEmail() {
    return Email;
}
public void setEmail(String Email) {
    this.Email = Email;
}
public String getTelefone() {
    return Telefone;
}
public void setTelefone(String Telefone) {
    this.Telefone = Telefone;
}
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;
}

//Método para exibir os dados no console
public void exibir() {

```

```

        System.out.println("ID Pessoa: " + IDPessoa);
        System.out.println("Nome: " + NomePessoa);
        System.out.println("E-mail: " + Email);
        System.out.println("Telefone: " + Telefone);
        System.out.println("Logradouro: " + Logradouro);
        System.out.println("Cidade: " + Cidade);
        System.out.println("Estado: " + Estado);
    }
}

```

---

**Classe PessoaFisica** - disponível em

"CadastroBD\src\cadastrobd\model\PessoaFisica.java"

```
package cadastrobd.model;
```

```

public class PessoaFisica extends Pessoa {
    //Campo
    private String CPF;

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

    //Construtor completo
    public PessoaFisica(int IDPessoa, String NomePessoa, String Email, String Telefone,
        String Logradouro, String Cidade, String Estado, String CPF) {
        super(IDPessoa, NomePessoa, Email, Telefone, Logradouro, Cidade, Estado);
        this.CPF = CPF;
    }

    //Getter e setter
    public String getCPF() {
        return CPF;
    }
    public void setCPF(String CPF) {
        this.CPF = CPF;
    }

    //Método exibir que sobrescreve exibir na classe Pessoa
    @Override
    public void exibir() {

```

```

        super.exibir();
        System.out.println("CPF: " + CPF);
        System.out.println("_____");
    }
}

```

---

**Classe PessoaJuridica** - disponível em  
 "CadastroBD\src\cadastrobd\model\PessoaJuridica.java"  
 package cadastrobd.model;

```

public class PessoaJuridica extends Pessoa {
    //Campo
    private String CNPJ;

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

    //Construtor completo
    public PessoaJuridica(int IDPessoa, String NomePessoa, String Email, String Telefone,
        String Logradouro, String Cidade, String Estado, String CNPJ) {
        super(IDPessoa, NomePessoa, Email, Telefone, Logradouro, Cidade, Estado);
        this.CNPJ = CNPJ;
    }

    //Getter e setter
    public String getCNPJ() {
        return CNPJ;
    }
    public void setCNPJ(String CNPJ) {
        this.CNPJ = CNPJ;
    }

    //Método exibir que sobreescreve exibir na classe Pessoa
    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CNPJ: " + CNPJ);
        System.out.println("_____");
    }
}

```

```
}
```

---

**Classe ConectorBD** - disponível em

"CadastroBD\src\cadastro\model\util\ConectorBD.java"

```
package cadastro.model.util;
```

```
import java.sql.*;
```

```
public class ConectorBD {
```

```
    private Connection conn;
```

```
    private PreparedStatement pst;
```

```
    private ResultSet rs;
```

```
    //Constantes para a conexão com o banco de dados
```

```
    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";
```

```
    //Método para retornar a conexão
```

```
    public Connection getConnection() throws SQLException {
```

```
        conn = DriverManager.getConnection(URL, USER, PASSWORD);
```

```
        return conn;
```

```
    }
```

```
    //Método para retornar um objeto do tipo PreparedStatement
```

```
    public PreparedStatement getPrepared(String sql) throws SQLException {
```

```
        pst = getConnection().prepareStatement(sql);
```

```
        return pst;
```

```
    }
```

```
    //Método para retornar o ResultSet relacionado a uma consulta
```

```
    public ResultSet getSelect(String sql) throws SQLException {
```

```
        pst = getPrepared(sql);
```

```
        rs = pst.executeQuery();
```

```
        return rs;
```

```
    }
```

```
    //Métodos para fechar os recursos
```

```
    public void close(Statement st) {
```

```
        if (st != null) {
```



```

        try {
            st.close();
        } catch (SQLException e) {
            System.out.println("Erro ao fechar o Statement: " + e);
        }
    }
}

public void close(ResultSet rs) {
    if (rs != null) {
        try {
            rs.close();
        } catch (SQLException e) {
            System.out.println("Erro ao fechar o ResultSet: " + e);
        }
    }
}

public void close(Connection conn) {
    if (conn != null) {
        try {
            conn.close();
        } catch (SQLException e) {
            System.out.println("Erro ao fechar a conexão: " + e);
        }
    }
}
}
}

```

---

**Classe SequenceManager** - disponível em  
 "CadastroBD\src\cadastro\model\util\SequenceManager.java"  
 package cadastro.model.util;

```

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

```

```

public class SequenceManager {

```

```

    // Método para obter o próximo valor de uma sequência

```

```

    public static int getValue(String sequenciaNome) throws SQLException {
        String sql = "SELECT NEXT VALUE FOR " + sequenciaNome;
    }
}

```

```

int nextValue = -1;

//Instância para ConectorBD
ConectorBD conectorBD = new ConectorBD();

//Acesso e retorno do banco de dados
try (Connection conn = conectorBD.getConnection();
    PreparedStatement pst = conn.prepareStatement(sql);
    ResultSet rs = pst.executeQuery()) {
    if (rs.next()) {
        nextValue = rs.getInt(1);
    } else {
        throw new SQLException("Nenhum valor retornado para a sequência: " +
sequenciaNome);
    }
} catch (SQLException e) {
    System.out.println("Erro ao obter valor da sequência: " + e.getMessage());
    throw e;
}
return nextValue;
}
}

```

---

**Classe PessoaFisicaDAO** - disponível em  
"CadastroBD\src\cadastro\model\PessoaFisicaDAO.java"

```

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

public class PessoaFisicaDAO {
    //Instância para ConectorBD
    private final ConectorBD conectorBD = new ConectorBD();

    //Método para retornar uma pessoa física pelo ID
    public PessoaFisica getPessoa(int IDPessoa) throws SQLException {
        String sql = "SELECT p.*, pf.CPF "
            + "FROM Pessoas p "

```

```

        + "INNER JOIN PessoasFisicas pf "
        + "ON p.IDPessoa = pf.Pessoas_IDPessoa "
        + "WHERE p.IDPessoa = ?";
Connection conn = null;
PreparedStatement pst = null;
ResultSet rs = null;
PessoaFisica pessoaFisica = null;
try {
    conn = conectorBD.getConnection();
    pst = conn.prepareStatement(sql);
    pst.setInt(1, IDPessoa);
    rs = pst.executeQuery();
    if (rs.next()) {
        pessoaFisica = new PessoaFisica(
            rs.getInt("IDPessoa"),
            rs.getString("NomePessoa"),
            rs.getString("Email"),
            rs.getString("Telefone"),
            rs.getString("Logradouro"),
            rs.getString("Cidade"),
            rs.getString("Estado"),
            rs.getString("CPF")
        );
    }
} catch (SQLException e) {
    System.out.println("Erro ao buscar a pessoa fisica pelo ID: " + e.getMessage());
    throw e;
} finally {
    conectorBD.close(rs);
    conectorBD.close(pst);
    conectorBD.close(conn);
}
return pessoaFisica;
}

```

//Método para retornar todas as pessoas físicas do banco de dados

```

public List<PessoaFisica> getPessoas() throws SQLException {
    String sql = "SELECT p.*, pf.CPF "
        + "FROM Pessoas p "
        + "INNER JOIN PessoasFisicas pf "

```

```

        + "ON p.IDPessoa = pf.Pessoas_IDPessoa";
Connection conn = null;
PreparedStatement pst = null;
ResultSet rs = null;
List<PessoaFisica> listaPf = new ArrayList<>();
try {
    conn = conectorBD.getConnection();
    pst = conn.prepareStatement(sql);
    rs = pst.executeQuery();
    while (rs.next()) {
        PessoaFisica pessoaFisica = new PessoaFisica(
            rs.getInt("IDPessoa"),
            rs.getString("NomePessoa"),
            rs.getString("Email"),
            rs.getString("Telefone"),
            rs.getString("Logradouro"),
            rs.getString("Cidade"),
            rs.getString("Estado"),
            rs.getString("CPF")
        );
        listaPf.add(pessoaFisica);
    }
} catch (SQLException e){
    System.out.println("Erro ao buscar todas as pessoas fisicas: " + e.getMessage());
    throw e;
} finally {
    conectorBD.close(rs);
    conectorBD.close(pst);
    conectorBD.close(conn);
}
return listaPf;
}

//Método para incluir uma pessoa física
public void incluir(PessoaFisica pessoaFisica) throws SQLException {
    String sqlPessoa = "INSERT INTO Pessoas (IDPessoa, NomePessoa, Email, Telefone,
Logradouro, Cidade, Estado) "
        + "VALUES (?, ?, ?, ?, ?, ?, ?)";
    String sqlPessoaFisica = "INSERT INTO PessoasFisicas (Pessoas_IDPessoa, CPF) "
        + "VALUES (?,?)";

```

```

Connection conn = null;
PreparedStatement pstPessoa = null;
PreparedStatement pstPessoaFisica = null;
try {
    conn = conectorBD.getConnection();
    conn.setAutoCommit(false);
    int IDPessoa = SequenceManager.getValue("Sequencia_IDPessoa");
    pessoaFisica.setIDPessoa(IDPessoa);
    //Incluindo na tabela Pessoas
    pstPessoa = conn.prepareStatement(sqlPessoa);
    pstPessoa.setInt(1, IDPessoa);
    pstPessoa.setString(2, pessoaFisica.getNomePessoa());
    pstPessoa.setString(3, pessoaFisica.getEmail());
    pstPessoa.setString(4, pessoaFisica.getTelefone());
    pstPessoa.setString(5, pessoaFisica.getLogradouro());
    pstPessoa.setString(6, pessoaFisica.getCidade());
    pstPessoa.setString(7, pessoaFisica.getEstado());
    pstPessoa.executeUpdate();
    //Incluindo na tabela PessoasFisicas
    pstPessoaFisica = conn.prepareStatement(sqlPessoaFisica);
    pstPessoaFisica.setInt(1, IDPessoa);
    pstPessoaFisica.setString(2, pessoaFisica.getCPF());
    pstPessoaFisica.executeUpdate();
    conn.commit();
} catch (SQLException e) {
    if (conn != null) {
        try {
            conn.rollback();
        } catch (SQLException e2) {
            System.out.println("Erro ao desfazer a inclusao: " + e2.getMessage());
        }
    }
    System.out.println("Erro ao incluir a pessoa fisica: " + e.getMessage());
    throw e;
} finally {
    conectorBD.close(pstPessoaFisica);
    conectorBD.close(pstPessoa);
    conectorBD.close(conn);
}
}

```

```

//Método para alterar uma pessoa física
public void alterar(PessoaFisica pessoaFisica) throws SQLException {
    String sqlPessoa = "UPDATE Pessoas "
        + "SET NomePessoa = ?, Email = ?, Telefone = ?, Logradouro = ?, Cidade = ?,
Estado = ? "
        + "WHERE IDPessoa = ?";
    String sqlPessoaFisica = "UPDATE PessoasFisicas "
        + "SET CPF = ? "
        + "WHERE Pessoas_IDPessoa = ?";
    Connection conn = null;
    PreparedStatement pstPessoa = null;
    PreparedStatement pstPessoaFisica = null;
    try {
        conn = conectorBD.getConnection();
        conn.setAutoCommit(false);
        //Alterando na tabela Pessoas
        pstPessoa = conn.prepareStatement(sqlPessoa);
        pstPessoa.setString(1, pessoaFisica.getNomePessoa());
        pstPessoa.setString(2, pessoaFisica.getEmail());
        pstPessoa.setString(3, pessoaFisica.getTelefone());
        pstPessoa.setString(4, pessoaFisica.getLogradouro());
        pstPessoa.setString(5, pessoaFisica.getCidade());
        pstPessoa.setString(6, pessoaFisica.getEstado());
        pstPessoa.setInt(7, pessoaFisica.getIDPessoa());
        pstPessoa.executeUpdate();
        //Alterando na tabela PessoasFisicas
        pstPessoaFisica = conn.prepareStatement(sqlPessoaFisica);
        pstPessoaFisica.setString(1, pessoaFisica.getCPF());
        pstPessoaFisica.setInt(2, pessoaFisica.getIDPessoa());
        pstPessoaFisica.executeUpdate();
        conn.commit();
    } catch (SQLException e) {
        if (conn != null) {
            try {
                conn.rollback();
            } catch (SQLException e2) {
                System.out.println("Erro ao desfazer a alteracao: " + e2.getMessage());
            }
        }
    }
}

```

```

        System.out.println("Erro ao alterar a pessoa fisica: " + e.getMessage());
        throw e;
    } finally {
        conectorBD.close(pstPessoaFisica);
        conectorBD.close(pstPessoa);
        conectorBD.close(conn);
    }
}

//Método para excluir uma pessoa física
public void excluir(int IDPessoa) throws SQLException {
    String sqlPessoa = "DELETE FROM Pessoas "
        + "WHERE IDPessoa = ?";
    String sqlPessoaFisica = "DELETE FROM PessoasFisicas "
        + "WHERE Pessoas_IDPessoa = ?";
    Connection conn = null;
    PreparedStatement pstPessoa = null;
    PreparedStatement pstPessoaFisica = null;
    try {
        conn = conectorBD.getConnection();
        conn.setAutoCommit(false);
        //Deletando da tabela PessoasFisicas
        pstPessoaFisica = conn.prepareStatement(sqlPessoaFisica);
        pstPessoaFisica.setInt(1, IDPessoa);
        pstPessoaFisica.executeUpdate();
        //Deletando da tabela Pessoas
        pstPessoa = conn.prepareStatement(sqlPessoa);
        pstPessoa.setInt(1, IDPessoa);
        pstPessoa.executeUpdate();
        conn.commit();
    } catch (SQLException e) {
        if (conn != null) {
            try {
                conn.rollback();
            } catch (SQLException e2) {
                System.out.println("Erro ao desfazer a exclusao: " + e2.getMessage());
            }
        }
    }
    System.out.println("Erro ao excluir a pessoa fisica: " + e.getMessage());
    throw e;
}

```

```

    } finally {
        conectorBD.close(pstPessoaFisica);
        conectorBD.close(pstPessoa);
        conectorBD.close(conn);
    }
}
}

```

---

**Classe PessoaJuridicaDAO** - disponível em  
"CadastroBD\src\cadastro\model\PessoaJuridicaDAO.java"

```

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

public class PessoaJuridicaDAO {
    //Instância para ConectorBD
    private final ConectorBD conectorBD = new ConectorBD();

    //Método para retornar uma pessoa jurídica pelo ID
    public PessoaJuridica getPessoa(int IDPessoa) throws SQLException {
        String sql = "SELECT p.*, pj.CNPJ "
            + "FROM Pessoas p "
            + "INNER JOIN PessoasJuridicas pj "
            + "ON p.IDPessoa = pj.Pessoas_IDPessoa "
            + "WHERE p.IDPessoa = ?";

        Connection conn = null;
        PreparedStatement pst = null;
        ResultSet rs = null;
        PessoaJuridica pessoaJuridica = null;
        try {
            conn = conectorBD.getConnection();
            pst = conn.prepareStatement(sql);
            pst.setInt(1, IDPessoa);
            rs = pst.executeQuery();
            if (rs.next()) {
                pessoaJuridica = new PessoaJuridica(
                    rs.getInt("IDPessoa"),

```



```

        rs.getString("NomePessoa"),
        rs.getString("Email"),
        rs.getString("Telefone"),
        rs.getString("Logradouro"),
        rs.getString("Cidade"),
        rs.getString("Estado"),
        rs.getString("CNPJ")
    );
}
} catch (SQLException e) {
    System.out.println("Erro ao buscar a pessoa juridica pelo ID: " + e.getMessage());
    throw e;
} finally {
    conectorBD.close(rs);
    conectorBD.close(pst);
    conectorBD.close(conn);
}
return pessoaJuridica;
}

```

//Método para retornar todas as pessoas jurídicas do banco de dados

```

public List<PessoaJuridica> getPessoas() throws SQLException {
    String sql = "SELECT p.*, pj.CNPJ "
        + "FROM Pessoas p "
        + "INNER JOIN PessoasJuridicas pj "
        + "ON p.IDPessoa = pj.Pessoas_IDPessoa";

    Connection conn = null;
    PreparedStatement pst = null;
    ResultSet rs = null;
    List<PessoaJuridica> listaPj = new ArrayList<>();
    try {
        conn = conectorBD.getConnection();
        pst = conn.prepareStatement(sql);
        rs = pst.executeQuery();
        while (rs.next()) {
            PessoaJuridica pessoaJuridica = new PessoaJuridica(
                rs.getInt("IDPessoa"),
                rs.getString("NomePessoa"),
                rs.getString("Email"),
                rs.getString("Telefone"),

```

```

        rs.getString("Logradouro"),
        rs.getString("Cidade"),
        rs.getString("Estado"),
        rs.getString("CNPJ")
    );
    listaPj.add(pessoaJuridica);
}
} catch (SQLException e) {
    System.out.println("Erro ao buscar todas as pessoas juridicas: " + e.getMessage());
    throw e;
} finally {
    conectorBD.close(rs);
    conectorBD.close(pst);
    conectorBD.close(conn);
}
return listaPj;
}

```

//Método para incluir uma pessoa jurídica

```

public void incluir(PessoaJuridica pessoaJuridica) throws SQLException {
    String sqlPessoa = "INSERT INTO Pessoas (IDPessoa, NomePessoa, Email, Telefone,
Logradouro, Cidade, Estado) "
        + "VALUES (?, ?, ?, ?, ?, ?, ?)";
    String sqlPessoaJuridica = "INSERT INTO PessoasJuridicas (Pessoas_IDPessoa,
CNPJ) "
        + "VALUES (?, ?)";
    Connection conn = null;
    PreparedStatement pstPessoa = null;
    PreparedStatement pstPessoaJuridica = null;
    try {
        conn = conectorBD.getConnection();
        conn.setAutoCommit(false);
        int IDPessoa = SequenceManager.getValue("Sequencia_IDPessoa");
        pessoaJuridica.setIDPessoa(IDPessoa);
        //Incluindo na tabela Pessoas
        pstPessoa = conn.prepareStatement(sqlPessoa);
        pstPessoa.setInt(1, IDPessoa);
        pstPessoa.setString(2, pessoaJuridica.getNomePessoa());
        pstPessoa.setString(3, pessoaJuridica.getEmail());
        pstPessoa.setString(4, pessoaJuridica.getTelefone());
    }
}

```

```

        pstPessoa.setString(5, pessoaJuridica.getLogradouro());
        pstPessoa.setString(6, pessoaJuridica.getCidade());
        pstPessoa.setString(7, pessoaJuridica.getEstado());
        pstPessoa.executeUpdate();
        //Incluindo na tabela PessoasJuridicas
        pstPessoaJuridica = conn.prepareStatement(sqlPessoaJuridica);
        pstPessoaJuridica.setInt(1, IDPessoa);
        pstPessoaJuridica.setString(2, pessoaJuridica.getCNPJ());
        pstPessoaJuridica.executeUpdate();
        conn.commit();
    } catch (SQLException e) {
        if (conn != null) {
            try {
                conn.rollback();
            } catch (SQLException e2) {
                System.out.println("Erro ao desfazer a inclusao: " + e2.getMessage());
            }
        }
        System.out.println("Erro ao incluir a pessoa juridica: " + e.getMessage());
        throw e;
    } finally {
        conectorBD.close(pstPessoaJuridica);
        conectorBD.close(pstPessoa);
        conectorBD.close(conn);
    }
}

```

```

//Método para alterar uma pessoa jurídica
public void alterar(PessoaJuridica pessoaJuridica) throws SQLException {
    String sqlPessoa = "UPDATE Pessoas "
        + "SET NomePessoa = ?, Email = ?, Telefone = ?, Logradouro = ?, Cidade = ?,
Estado = ? "
        + "WHERE IDPessoa = ?";
    String sqlPessoaJuridica = "UPDATE PessoasJuridicas "
        + "SET CNPJ = ? "
        + "WHERE Pessoas_IDPessoa = ?";
    Connection conn = null;
    PreparedStatement pstPessoa = null;
    PreparedStatement pstPessoaJuridica = null;
    try {

```

```

conn = conectorBD.getConnection();
conn.setAutoCommit(false);
//Alterando na tabela Pessoas
pstPessoa = conn.prepareStatement(sqlPessoa);
pstPessoa.setString(1, pessoaJuridica.getNomePessoa());
pstPessoa.setString(2, pessoaJuridica.getEmail());
pstPessoa.setString(3, pessoaJuridica.getTelefone());
pstPessoa.setString(4, pessoaJuridica.getLogradouro());
pstPessoa.setString(5, pessoaJuridica.getCidade());
pstPessoa.setString(6, pessoaJuridica.getEstado());
pstPessoa.setInt(7, pessoaJuridica.getIDPessoa());
pstPessoa.executeUpdate();
//Alterando na tabela PessoasJuridicas
pstPessoaJuridica = conn.prepareStatement(sqlPessoaJuridica);
pstPessoaJuridica.setString(1, pessoaJuridica.getCNPJ());
pstPessoaJuridica.setInt(2, pessoaJuridica.getIDPessoa());
pstPessoaJuridica.executeUpdate();
conn.commit();
} catch (SQLException e) {
    if (conn != null) {
        try {
            conn.rollback();
        } catch (SQLException e2) {
            System.out.println("Erro ao desfazer a alteracao: " + e2.getMessage());
        }
    }
    System.out.println("Erro ao alterar a pessoa juridica: " + e.getMessage());
    throw e;
} finally {
    conectorBD.close(pstPessoaJuridica);
    conectorBD.close(pstPessoa);
    conectorBD.close(conn);
}
}

```

//Método para excluir uma pessoa jurídica

```

public void excluir(int IDPessoa) throws SQLException {
    String sqlPessoa = "DELETE FROM Pessoas "
        + "WHERE IDPessoa = ?";
    String sqlPessoaJuridica = "DELETE FROM PessoasJuridicas "

```

```

        + "WHERE Pessoas_IDPessoa = ?";
Connection conn = null;
PreparedStatement pstPessoa = null;
PreparedStatement pstPessoaJuridica = null;
try {
    conn = conectorBD.getConnection();
    conn.setAutoCommit(false);
    //Deletando da tabela PessoasJuridicas
    pstPessoaJuridica = conn.prepareStatement(sqlPessoaJuridica);
    pstPessoaJuridica.setInt(1, IDPessoa);
    pstPessoaJuridica.executeUpdate();
    //Deletando da tabela Pessoas
    pstPessoa = conn.prepareStatement(sqlPessoa);
    pstPessoa.setInt(1, IDPessoa);
    pstPessoa.executeUpdate();
    conn.commit();
} catch (SQLException e) {
    if (conn != null) {
        try {
            conn.rollback();
        } catch (SQLException e2) {
            System.out.println("Erro ao desfazer a exclusao: " + e2.getMessage());
        }
    }
    System.out.println("Erro ao excluir a pessoa juridica: " + e.getMessage());
    throw e;
} finally {
    conectorBD.close(pstPessoaJuridica);
    conectorBD.close(pstPessoa);
    conectorBD.close(conn);
}
}
}

```

---

**Classe CadastroBDTeste** - disponível em

"CadastroBD\src\cadastrobd\CadastroBDTeste.java"

```
package cadastrobd;
```

```
import cadastro.model.PessoaFisicaDAO;
```

```
import cadastro.model.PessoaJuridicaDAO;
```

```
import cadastrobd.model.PessoaFisica;
```

```
import cadastrobd.model.PessoaJuridica;
```

```
import java.sql.SQLException;
```

```
import java.util.List;
```

```
public class CadastroBDTeste {
```

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

```
        //Instâncias dos objetos DAO
```

```
        PessoaFisicaDAO pfDAO = new PessoaFisicaDAO();
```

```
        PessoaJuridicaDAO pjDAO = new PessoaJuridicaDAO();
```

```
        try {
```

```
            //Instância de pessoa física persistida no banco de dados
```

```
            PessoaFisica pf = new PessoaFisica(0, "Claudio Pontes", "claudio06@email.com",  
                "11955664489", "Rua Moraes, 17", "Lapa", "SP", "44888959610");
```

```
            pfDAO.incluir(pf);
```

```
            System.out.println("Pessoa física incluída com sucesso: ");
```

```
            pf.exibir();
```

```
            //Alterar os dados da pessoa física no banco de dados
```

```
            pf.setNomePessoa("Claudio Santos");
```

```
            pf.setLogradouro("Rua Moraes, 71");
```

```
            pfDAO.alterar(pf);
```

```
            System.out.println("Pessoa física alterada com sucesso: ");
```

```
            pf.exibir();
```

```
            //Consultar todas as pessoas físicas no banco de dados
```

```
            List<PessoaFisica> listaPf = pfDAO.getPessoas();
```

```
            System.out.println("Lista de todas as pessoas físicas: ");
```

```
            for (PessoaFisica pessoa : listaPf) {
```

```
                pessoa.exibir();
```

```
            }
```

```
            //Excluir a pessoa física criada anteriormente
```

```
            pfDAO.excluir(pf.getIDPessoa());
```

```

System.out.println("Pessoa fisica excluida com sucesso.");
System.out.println("_____");

//Instância de uma pessoa jurídica persistida no banco de dados
PessoaJuridica pj = new PessoaJuridica(0, "Sacola Verde", "sacver@email.com",
    "3333445566", "Avenida Kimber, 5", "Rio Claro", "MG", "12345678508199");
pjDAO.incluir(pj);
System.out.println("Pessoa juridica incluida com sucesso: ");
pj.exibir();

//Alterar os dados da pessoa jurídica no banco de dados
pj.setLogradouro("Rua Kimber, 5");
pj.setCidade("Ceú Claro");
pjDAO.alterar(pj);
System.out.println("Pessoa juridica alterada com sucesso: ");
pj.exibir();

//Consultar todas as pessoas jurídicas no banco de dados
List<PessoaJuridica> listaPj = pjDAO.getPessoas();
System.out.println("Lista de todas as pessoas juridicas: ");
for (PessoaJuridica pessoa : listaPj) {
    pessoa.exibir();
}

//Excluir a pessoa jurídica criada anteriormente
pjDAO.excluir(pj.getIDPessoa());
System.out.println("Pessoa juridica excluida com sucesso.");
System.out.println("_____");
} catch (SQLException e) {
    System.out.println("Erro durante os testes: " + e.getMessage());
}
}
}

```

---

- Resultados da execução | Procedimento 1

```
Projects | Files | Services | Output - CadastroBD (run) x
```

```
run:
Pessoa fisica incluida com sucesso:
ID Pessoa: 62
Nome: Claudio Pontes
E-mail: claudio06@email.com
Telefone: 11955664489
Logradouro: Rua Moraes, 17
Cidade: Lapa
Estado: SP
CPF: 44888959610

Pessoa fisica alterada com sucesso:
ID Pessoa: 62
Nome: Claudio Santos
E-mail: claudio06@email.com
Telefone: 11955664489
Logradouro: Rua Moraes, 71
Cidade: Lapa
Estado: SP
CPF: 44888959610

Lista de todas as pessoas fisicas:
ID Pessoa: 1
Nome: Pedro Avilar
E-mail: pe.avilar@email.com
Telefone: 11971667717
Logradouro: Rua Vera, 35
Cidade: Guarulhos
Estado: SP
CPF: 12121313910

ID Pessoa: 62
Nome: Claudio Santos
E-mail: claudio06@email.com
Telefone: 11955664489
Logradouro: Rua Moraes, 71
Cidade: Lapa
Estado: SP
CPF: 44888959610

ID Pessoa: 2
Nome: Wilson Araujo
E-mail: wilson.ar@email.com
Telefone: 11955544555
Logradouro: Estrada Rio, 7
Cidade: Rio Claro
Estado: MG
CPF: 54335499206

Pessoa fisica excluida com sucesso.
```



```

Pessoa juridica incluida com sucesso:
ID Pessoa: 63
Nome: Sacola Verde
E-mail: sacver@email.com
Telefone: 3333445566
Logradouro: Avenida Kimber, 5
Cidade: Rio Claro
Estado: MG
CNPJ: 12345678508199

Pessoa juridica alterada com sucesso:
ID Pessoa: 63
Nome: Sacola Verde
E-mail: sacver@email.com
Telefone: 3333445566
Logradouro: Rua Kimber, 5
Cidade: Ceu Claro
Estado: MG
CNPJ: 12345678508199

Lista de todas as pessoas juridicas:
ID Pessoa: 63
Nome: Sacola Verde
E-mail: sacver@email.com
Telefone: 3333445566
Logradouro: Rua Kimber, 5
Cidade: Ceu Claro
Estado: MG
CNPJ: 12345678508199

ID Pessoa: 4
Nome: Panificadora Ysa
E-mail: pan_ysa@email.com
Telefone: 11912348765
Logradouro: Avenida Jane, 5760
Cidade: Gaspar
Estado: SC
CNPJ: 12568905100001

ID Pessoa: 3
Nome: Feira Vanessa Cot
E-mail: van_cot@email.com
Telefone: 11922334455
Logradouro: Rua Emilio, 850
Cidade: Mombaca
Estado: CE
CNPJ: 55884466000001

Pessoa juridica excluida com sucesso.

BUILD SUCCESSFUL (total time: 5 seconds)
```

- **Análise e conclusão | Procedimento 1**

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

Os componentes de middleware, como o JDBC, são importantes para permitir a comunicação entre uma aplicação Java e o banco de dados relacional.

O JDBC fornece uma API padrão para realizar operações no banco de dados pela aplicação Java, para criação, leitura, alteração e exclusão, oferecendo métodos para conexões e transações. Conseguindo um código mais flexível e reutilizável para diversos Sistemas de Gerenciamento de Banco de Dados com poucas alterações.

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

A diferença entre o uso de Statement e PreparedStatement está na segurança, desempenho e praticidade.

O PreparedStatement oferece maior segurança, usando parâmetros que podem ser substituídos e definidos de forma separada, tendo uma consulta pré-compilada, prevenindo a injeção de SQL, onde dados maliciosos podem alterar a lógica SQL. No Statement, por permitir a execução de consultas de forma direta, tendo o código SQL construído de forma dinâmica, e não pré-compilado, resulta em menos segurança.

Sempre que uma consulta é executada com o uso de Statement, o SQL é enviado ao banco de dados para ser compilado e executado, assim, temos um desempenho inferior quando comparado ao uso de PreparedStatement, que tem as consultas pré-compiladas, onde a mesma consulta com parâmetros diferentes não precisa ser recompilada a cada instrução.

Quando precisamos implementar consultas com múltiplos valores, ou dados complexos, o uso de Statement se torna complicado, sendo mais prático para implementar e fazer manutenção no uso de PreparedStatement.

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

O padrão DAO melhora a manutenibilidade por manter de forma separada a lógica de acesso aos dados do resto da aplicação, permitindo que mudanças na estrutura do banco de dados ou a troca do tipo de banco usado não afetem de forma direta a lógica de negócios. Com essa separação temos um software com menor complexidade, facilitando testes e a evolução do software.

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

A herança é refletida em tabelas e relacionamentos, considerando opções como uma tabela única, podendo resultar em campos vazios para algumas ou várias colunas, uma tabela por classe, onde evita campos e colunas vazias e é necessário o uso de JOINS para consultas completas dos dados, e o uso de tabelas por subclasse, contendo todos os dados de específicos e herdados, causando dados duplicados.

- **2º Procedimento | Alimentando a Base**

**Classe CadastroBDMain**- disponível em  
"CadastroBD\src\cadastrobd\CadastroBDMain.java"

```
package cadastrobd;
import cadastro.model.PessoaFisicaDAO;
import cadastro.model.PessoaJuridicaDAO;
import java.util.InputMismatchException;
import java.util.Scanner;
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaJuridica;
import java.sql.SQLException;

public class CadastroBDMain {
    private static Scanner scanner = new Scanner(System.in);

    //Método para tratamento de exceção de leitura de números
    private static int entradaInt(String mensagem) {
        int valor = -1;
        while (valor == -1) {
            System.out.print(mensagem);
            try {
                valor = scanner.nextInt();
                scanner.nextLine();
            } catch (InputMismatchException e) {
                System.out.println("Entrada invalida. Insira um numero inteiro.");
                scanner.nextLine();
            }
        }
        return valor;
    }
}
```

```

//Método auxiliar para exibir mensagens e receber dados do usuário
private static String entradaString(String mensagem) {
    System.out.print(mensagem);
    return scanner.nextLine();
}

public static void main(String[] args) {
    //Instâncias dos objetos DAO
    PessoaFisicaDAO pfDAO = new PessoaFisicaDAO();
    PessoaJuridicaDAO pjDAO = new PessoaJuridicaDAO();

    //Laço que mantém o programa em execução
    while (true) {
        System.out.println("=====");
        System.out.println("MENU");
        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("=====");
        int opcao = entradaInt("Digite o numero da opcao: ");

        switch (opcao) {
            //Incluir
            case 1: {
                System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
                String tipoIncluir = entradaString("Tipo de pessoa: ");
                if (tipoIncluir.equalsIgnoreCase("F")) { //Pessoa Física
                    System.out.println("Insira os dados da pessoa fisica.\n");
                    //Capturando e inserindo pessoa física
                    int id = 0; //ID será gerado pelo SequenceManager
                    String nome = entradaString("Nome: ");
                    String email = entradaString("E-mail: ");
                    String telefone = entradaString("Telefone: ");
                    String logradouro = entradaString("Logradouro: ");
                    String cidade = entradaString("Cidade: ");
                    String estado = entradaString("Estado: ");
                }
            }
        }
    }
}

```

```

String cpf = entradaString("CPF: ");
try {
    PessoaFisica pf = new PessoaFisica(id, nome, email,
        telefone, logradouro, cidade, estado, cpf);
    pfDAO.incluir(pf);
    System.out.println("Pessoa fisica incluida com sucesso.\n");
} catch (SQLException e) {
    System.out.println("Erro ao incluir pf pela classe principal: " +
e.getMessage());
}
} else if (tipoIncluir.equalsIgnoreCase("J")) { //Pessoa Jurídica
    System.out.println("Insira os dados da pessoa juridica.\n");
    //Capturando e inserindo pessoa jurídica
    int id = 0; //ID será gerado pelo SequenceManager
    String nome = entradaString("Nome: ");
    String email = entradaString("E-mail: ");
    String telefone = entradaString("Telefone: ");
    String logradouro = entradaString("Logradouro: ");
    String cidade = entradaString("Cidade: ");
    String estado = entradaString("Estado: ");
    String cnpj = entradaString("CNPJ: ");
    try {
        PessoaJuridica pj = new PessoaJuridica(id, nome, email,
            telefone, logradouro, cidade, estado, cnpj);
        pjDAO.incluir(pj);
        System.out.println("Pessoa juridica incluida com sucesso.\n");
    } catch (SQLException e) {
        System.out.println("Erro ao incluir pj pela classe principal: " +
e.getMessage());
    }
} else {
    System.out.println("Opcao invalida.");
}
} break;

//Alterar
case 2: {
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    String tipoAlterar = entradaString("Tipo de pessoa: ");
    if (tipoAlterar.equalsIgnoreCase("F")) { //Pessoa Física

```

```

try {
    int id = entradaInt("Digite o ID da pessoa fisica para alterar: ");
    PessoaFisica pf = pfDAO.getPessoa(id);
    //Exibe os dados atuais e exige os novos
    if (pf != null) {
        System.out.println("DADOS ATUAIS: ");
        pf.exibir();
        System.out.println("Insira os novos dados.");
        pf.setNomePessoa(entradaString("Novo nome: "));
        pf.setEmail(entradaString("Novo e-mail: "));
        pf.setTelefone(entradaString("Novo telefone: "));
        pf.setLogradouro(entradaString("Novo logradouro: "));
        pf.setCidade(entradaString("Nova cidade: "));
        pf.setEstado(entradaString("Novo estado: "));
        pf.setCPF(entradaString("Novo CPF: "));
        pfDAO.alterar(pf);
        System.out.println("Pessoa fisica alterada com sucesso.\n");
    } else {
        System.out.println("Nao foi encontrada uma pessoa fisica com o ID " +
id);
    }
} catch (SQLException e) {
    System.out.println("Erro ao alterar pf na classe principal: " +
e.getMessage());
}
} else if (tipoAlterar.equalsIgnoreCase("J")) { //Pessoa Jurídica
    try {
        int id = entradaInt("Digite o ID da pessoa juridica para alterar: ");
        PessoaJuridica pj = pjDAO.getPessoa(id);
        //Exibe os dados atuais e exige os novos
        if (pj != null) {
            System.out.println("Dados atuais: ");
            pj.exibir();
            System.out.println("Insira os novos dados.\n");
            pj.setNomePessoa(entradaString("Novo nome: "));
            pj.setEmail(entradaString("Novo e-mail: "));
            pj.setTelefone(entradaString("Novo telefone: "));
            pj.setLogradouro(entradaString("Novo logradouro: "));
            pj.setCidade(entradaString("Nova cidade: "));
            pj.setEstado(entradaString("Novo estado: "));

```

```

        pj.setCNPJ(entradaString("Novo CNPJ: "));
        pjDAO.alterar(pj);
        System.out.println("Pessoa juridica alterada com sucesso.\n");
    } else {
        System.out.println("Nao foi encontrada uma pessoa juridica com o ID " +
id);
    }
} catch (SQLException e) {
    System.out.println("Erro ao alterar pj na classe principal: " +
e.getMessage());
}
} else {
    System.out.println("Opcao invalida.");
}
} break;

//Excluir
case 3: {
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    String tipoExcluir = entradaString("Tipo de pessoa: ");
    if (tipoExcluir.equalsIgnoreCase("F")) { //Pessoa Física
        try {
            int id = entradaInt("Digite o ID da pessoa fisica para excluir: ");
            PessoaFisica pf = pfDAO.getPessoa(id);
            if (pf != null) {
                pfDAO.excluir(id);
                System.out.println("Pessoa fisica excluida com sucesso.\n");
            } else {
                System.out.println("ID nao encontrado.");
            }
        } catch (SQLException e) {
            System.out.println("Erro ao excluir pf na classe principal: " +
e.getMessage());
        }
    } else if (tipoExcluir.equalsIgnoreCase("J")) { //Pessoa Jurídica
        try {
            int id = entradaInt("Digite o ID da pessoa juridica para excluir: ");
            PessoaJuridica pj = pjDAO.getPessoa(id);
            if (pj != null) {
                pjDAO.excluir(id);

```

```

        System.out.println("Pessoa juridica excluida com sucesso.\n");
    } else {
        System.out.println("ID nao encontrado.");
    }
} catch (SQLException e) {
    System.out.println("Erro ao excluir pj na classe principal: " +
e.getMessage());
}
} else {
    System.out.println("Opcao invalida.");
}
} break;

//Buscar
case 4: {
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    String tipoBuscar = entradaString("Tipo de pessoa: ");
    if (tipoBuscar.equalsIgnoreCase("F")) { //Pessoa Física
        try {
            int id = entradaInt("Digite o ID da pessoa fisica para exibir: ");
            PessoaFisica pf = pfDAO.getPessoa(id);
            if (pf != null) {
                System.out.println("DADOS DA PESSOA FISICA\n");
                pf.exibir();
            } else {
                System.out.println("Pessoa fisica nao encontrada.");
            }
        } catch (SQLException e) {
            System.out.println("Erro ao buscar pf na classe principal: " +
e.getMessage());
        }
    } else if (tipoBuscar.equalsIgnoreCase("J")) { //Pessoa Jurídica
        try {
            int id = entradaInt("Digite o ID da pessoa juridica para exibir: ");
            PessoaJuridica pj = pjDAO.getPessoa(id);
            if (pj != null) {
                System.out.println("DADOS DA PESSOA JURIDICA\n");
                pj.exibir();
            } else {
                System.out.println("Pessoa juridica nao encontrada.");
            }
        }
    }
}
}

```



```

    }
    } catch (SQLException e) {
        System.out.println("Erro ao buscar pj na classe principal: " +
e.getMessage());
    }
    } else {
        System.out.println("Opcao invalida.");
    }
} break;

//Exibir todos
case 5: {
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    String tipoExibirTodos = entradaString("Tipo de pessoa: ");
    if (tipoExibirTodos.equalsIgnoreCase("F")) { //Pessoa Física
        try {
            System.out.println("DADOS DAS PESSOAS FISICAS\n");
            pfDAO.getPessoas().forEach(pf -> pf.exibir());
        } catch (SQLException e) {
            System.out.println("Erro ao exibir todas pf na classe principal: " +
e.getMessage());
        }
    } else if (tipoExibirTodos.equalsIgnoreCase("J")) { //Pessoa Jurídica
        try {
            System.out.println("DADOS DAS PESSOAS JURIDICAS\n");
            pjDAO.getPessoas().forEach(pj -> pj.exibir());
        } catch (SQLException e) {
            System.out.println("Erro ao exibir todas pj na classe principal: " +
e.getMessage());
        }
    } else {
        System.out.println("Opcao invalida.");
    }
} break;

//Finalizar o programa
case 0: {
    System.out.println("Programa finalizado.");
    scanner.close();
    System.exit(0);
}

```

```
    } break;

    default: {
        System.out.println("Opcao invalida. Tente novamente.");
    }
}
}
}
}
```

---

- Resultados da execução | Procedimento 2

- Incluindo uma pessoa física.

Banco de dados inicial:

	IDPessoa	NomePessoa	Email	Telefone	Logradouro	Cidade	Estado	CPF
1	1	Pedro Avilar	pe.avilar@email.com	11971667717	Rua Vera, 35	Guarulhos	SP	12121313910
2	2	Wilson Araujo	wilson.ar@email.com	11955544555	Estrada Rio, 7	Rio Claro	MG	54335499206

Execução do programa:

```
run:
=====
MENU
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
Digite o numero da opcao: 1
F - Pessoa Fisica | J - Pessoa Juridica
Tipo de pessoa: f
Insira os dados da pessoa fisica.

Nome: Jane Branca
E-mail: j.branca@email.com
Telefone: 11920203344
Logradouro: Travessa Sato
Cidade: Itaquaquecetuba
Estado: SP
CPF: 88556499102
Pessoa fisica incluida com sucesso.
```

Resultado no banco de dados:

Resultados		Mensagens						
	IDPessoa	NomePessoa	Email	Telefone	Logradouro	Cidade	Estado	CPF
1	1	Pedro Avilar	pe.avilar@email.com	11971667717	Rua Vera, 35	Guarulhos	SP	12121313910
2	2	Wilson Araujo	wilson.ar@email.com	11955544555	Estrada Rio, 7	Rio Claro	MG	54335499206
3	64	Jane Branca	j.branca@email.com	11920203344	Travessa Sato	Itaquaquecetuba	SP	88556499102

- **Incluindo uma pessoa jurídica.**

Banco de dados inicial:

	IDPessoa	NomePessoa	Email	Telefone	Logradouro	Cidade	Estado	CNPJ
1	3	Feira Vanessa	van_cot@email.com	11900110000	Rua Emidio, 850	Mombaca	CE	55880000000001
2	4	Panificadora Ysa	pan_ysa@email.com	11912348765	Avenida Jane, 5760	Gaspar	SC	12568905100001

Execução do programa:

```

=====
MENU
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
Digite o numero da opcao: 1
F - Pessoa Fisica | J - Pessoa Juridica
Tipo de pessoa: J
Insira os dados da pessoa juridica.

Nome: Clara Verdes
E-mail: clara.v@email.com
Telefone: 11988559030
Logradouro: Rua Roxa, 7
Cidade: Mooca
Estado: SP
CNPJ: 155699000000001
Pessoa juridica incluída com sucesso.

```

Resultado no banco de dados:

	IDPessoa	NomePessoa	Email	Telefone	Logradouro	Cidade	Estado	CNPJ
1	3	Feira Vanessa	van_cot@email.com	11900110000	Rua Emidio, 850	Mombaca	CE	55880000000001
2	4	Panificadora Ysa	pan_ysa@email.com	11912348765	Avenida Jane, 5760	Gaspar	SC	12568905100001
3	66	Clara Verdes	clara.v@email.com	11988559030	Rua Roxa, 7	Mooca	SP	155699000000001

- **Alterando uma pessoa física.**

Execução do programa:

```
=====
MENU
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
Digite o numero da opcao: 2
F - Pessoa Fisica | J - Pessoa Juridica
Tipo de pessoa: F
Digite o ID da pessoa fisica para alterar: 64
DADOS ATUAIS:
ID Pessoa: 64
Nome: Jane Branca
E-mail: j.branca@email.com
Telefone: 11920203344
Logradouro: Travessa Sato
Cidade: Itaquaquecetuba
Estado: SP
CPF: 88556499102

-----
Insira os novos dados.
Novo nome: Branca Jane
Novo e-mail: j.branca@email.com
Novo telefone: 11980005000
Novo logradouro: Rua Sato
Nova cidade: Itaquaquecetuba
Novo estado: SP
Novo CPF: 88556499102
Pessoa fisica alterada com sucesso.
```

Resultado no banco de dados:

Resultados		Mensagens						
	IDPessoa	NomePessoa	Email	Telefone	Logradouro	Cidade	Estado	CPF
1	1	Pedro Avilar	pe.avilar@email.com	11971667717	Rua Vera, 35	Guarulhos	SP	12121313910
2	2	Wilson Araujo	wilson.ar@email.com	11955544555	Estrada Rio, 7	Rio Claro	MG	54335499206
3	64	Branca Jane	j.branca@email.com	11980005000	Rua Sato	Itaquaquecetuba	SP	88556499102

- **Alterando uma pessoa jurídica.**

Execução do programa:

```
=====
MENU
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
Digite o numero da opcao: 2
F - Pessoa Fisica | J - Pessoa Juridica
Tipo de pessoa: J
Digite o ID da pessoa juridica para alterar: 4
Dados atuais:
ID Pessoa: 4
Nome: Panificadora Ysa
E-mail: pan_ysa@email.com
Telefone: 11912348765
Logradouro: Avenida Jane, 5760
Cidade: Gaspar
Estado: SC
CNPJ: 12568905100001

-----
Insira os novos dados.

Novo nome: Ysa Confeitaria
Novo e-mail: ysa_conf@email.com
Novo telefone: 47950209911
Novo logradouro: Avenida Jane, 5760
Nova cidade: Gaspar
Novo estado: SC
Novo CNPJ: 12559473000001
Pessoa juridica alterada com sucesso.
```

Resultado no banco de dados:

	Resultados	Mensagens							
	IDPessoa	NomePessoa	Email	Telefone	Logradouro	Cidade	Estado	CNPJ	
1	3	Feira Vanessa	van_cot@email.com	11900110000	Rua Emidio, 850	Mombaca	CE	55880000000001	
2	4	Ysa Confeitaria	ysa_conf@email.com	47950209911	Avenida Jane, 5760	Gaspar	SC	12559473000001	
3	66	Clara Verdes	clara.v@email.com	11988559030	Rua Roxa, 7	Mooca	SP	15569900000001	

- **Excluindo uma pessoa física.**

Banco de dados inicial:

	IDPessoa	NomePessoa	Email	Telefone	Logradouro	Cidade	Estado	CPF
1	1	Pedro Avilar	pe.avilar@email.com	11971667717	Rua Vera, 35	Guarulhos	SP	12121313910
2	2	Wilson Araujo	wilson.ar@email.com	11955544555	Estrada Rio, 7	Rio Claro	MG	54335499206
3	64	Branca Jane	j.branca@email.com	11980005000	Rua Sato	Itaquaquecetuba	SP	88556499102

Execução do programa:

```

run:
=====
MENU
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
Digite o numero da opcao: 3
F - Pessoa Fisica | J - Pessoa Juridica
Tipo de pessoa: F
Digite o ID da pessoa fisica para excluir: 64
Pessoa fisica excluida com sucesso.

```

Resultado no banco de dados:

	IDPessoa	NomePessoa	Email	Telefone	Logradouro	Cidade	Estado	CPF
1	1	Pedro Avilar	pe.avilar@email.com	11971667717	Rua Vera, 35	Guarulhos	SP	12121313910
2	2	Wilson Araujo	wilson.ar@email.com	11955544555	Estrada Rio, 7	Rio Claro	MG	54335499206

- **Excluindo uma pessoa jurídica.**

Banco de dados inicial:

	IDPessoa	NomePessoa	Email	Telefone	Logradouro	Cidade	Estado	CNPJ
1	3	Feira Vanessa	van_cot@email.com	11900110000	Rua Emidio, 850	Mombaca	CE	55880000000001
2	4	Ysa Confeitaria	ysa_conf@email.com	47950209911	Avenida Jane, 5760	Gaspar	SC	12559473000001
3	66	Clara Verdes	clara.v@email.com	11988559030	Rua Roxa, 7	Mooca	SP	15569900000001

Execução do programa:

```

=====
MENU
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
Digite o numero da opcao: 3
F - Pessoa Fisica | J - Pessoa Juridica
Tipo de pessoa: J
Digite o ID da pessoa juridica para excluir: 66
Pessoa juridica excluida com sucesso.

```

Resultado no banco de dados:

	IDPessoa	NomePessoa	Email	Telefone	Logradouro	Cidade	Estado	CNPJ
1	3	Feira Vanessa	van_cot@email.com	11900110000	Rua Emidio, 850	Mombaca	CE	55880000000001
2	4	Ysa Confeitaria	ysa_conf@email.com	47950209911	Avenida Jane, 5760	Gaspar	SC	12559473000001



- **Buscar pelo ID uma pessoa física.**

Execução do programa:

```
MENU
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
Digite o numero da opcao: 4
F - Pessoa Fisica | J - Pessoa Juridica
Tipo de pessoa: F
Digite o ID da pessoa fisica para exibir: 1
DADOS DA PESSOA FISICA

ID Pessoa: 1
Nome: Pedro Avilar
E-mail: pe.avilar@email.com
Telefone: 11971667717
Logradouro: Rua Vera, 35
Cidade: Guarulhos
Estado: SP
CPF: 12121313910
```

- **Buscar pelo ID uma pessoa jurídica.**

Execução do programa:

```
MENU
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
Digite o numero da opcao: 4
F - Pessoa Fisica | J - Pessoa Juridica
Tipo de pessoa: J
Digite o ID da pessoa juridica para exibir: 3
DADOS DA PESSOA JURIDICA

ID Pessoa: 3
Nome: Feira Vanessa
E-mail: van_cot@email.com
Telefone: 11900110000
Logradouro: Rua Emidio, 850
Cidade: Mombaca
Estado: CE
CNPJ: 558800000000001
```

- **Exibir todos para pessoas físicas.**

Execução do programa:

```
=====
MENU
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
Digite o numero da opcao: 5
F - Pessoa Fisica | J - Pessoa Juridica
Tipo de pessoa: f
DADOS DAS PESSOAS FISICAS

ID Pessoa: 1
Nome: Pedro Avilar
E-mail: pe.avilar@email.com
Telefone: 11971667717
Logradouro: Rua Vera, 35
Cidade: Guarulhos
Estado: SP
CPF: 12121313910

ID Pessoa: 2
Nome: Wilson Araujo
E-mail: wilson.ar@email.com
Telefone: 11955544555
Logradouro: Estrada Rio, 7
Cidade: Rio Claro
Estado: MG
CPF: 54335499206
```

- **Exibir todos para pessoas jurídicas.**

Execução do programa:

```
=====
MENU
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
Digite o numero da opcao: 5
F - Pessoa Fisica | J - Pessoa Juridica
Tipo de pessoa: j
DADOS DAS PESSOAS JURIDICAS

ID Pessoa: 4
Nome: Ysa Confeitaria
E-mail: ysa_conf@email.com
Telefone: 47950209911
Logradouro: Avenida Jane, 5760
Cidade: Gaspar
Estado: SC
CNPJ: 125594730000001

-----
ID Pessoa: 3
Nome: Feira Vanessa
E-mail: van_cot@email.com
Telefone: 11900110000
Logradouro: Rua Emidio, 850
Cidade: Mombaca
Estado: CE
CNPJ: 558800000000001
-----
```

- **Finalizar o programa.**

```
=====
MENU
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
Digite o numero da opcao: 0
Programa finalizado.
BUILD SUCCESSFUL (total time: 7 minutes 24 seconds)
```

- **Exemplos de tratamentos de exceções.**

Entrada de opção no menu:

```
run:
=====
MENU
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
Digite o numero da opcao: L
Entrada invalida. Insira um numero inteiro.
Digite o numero da opcao: 7
Opcao invalida. Tente novamente.
=====
MENU
1 - Incluir Pessoa
```

Selecionar o tipo de pessoa:

```
=====
MENU
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
Digite o numero da opcao: 2
F - Pessoa Fisica | J - Pessoa Juridica
Tipo de pessoa: 7
Opcao invalida.
=====
MENU
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
Digite o numero da opcao: 1
F - Pessoa Fisica | J - Pessoa Juridica
Tipo de pessoa: K
Opcao invalida.
=====
MENU
1 - Incluir Pessoa
```

IDs inexistentes:

```
run:
=====
MENU
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
Digite o numero da opcao: 2
F - Pessoa Fisica | J - Pessoa Juridica
Tipo de pessoa: F
Digite o ID da pessoa fisica para alterar: 100
Nao foi encontrada uma pessoa fisica com o ID 100
=====
MENU
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
Digite o numero da opcao: 3
F - Pessoa Fisica | J - Pessoa Juridica
Tipo de pessoa: J
Digite o ID da pessoa juridica para excluir: 100
ID nao encontrado.
=====
MENU
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
Digite o numero da opcao: 4
F - Pessoa Fisica | J - Pessoa Juridica
Tipo de pessoa: F
Digite o ID da pessoa fisica para exibir: 101
Pessoa fisica nao encontrada.
=====
MENU
1 - Incluir Pessoa
```

- **Análise e conclusão | Procedimento 2**

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

A persistência em arquivo é mais simples de implementar, ideal para pequenos dados e sem complexidade, sendo para operações básicas de leitura e escrita. Possui menor segurança e confiabilidade, não oferecendo mecanismos internos de recuperação de falhas e integridade referencial, sendo mais vulnerável à corrupção ou perda.

A persistência em banco de dados é mais complexa de implementar, exige um Sistema de Gerenciamento de Banco de Dados, podendo ter configurações adicionais, sendo mais otimizado para armazenamento, busca e manipulação até para grande volume de dados. Oferece mais segurança e confiabilidade, com controle de acesso, criptografia, integridade referencial e mecanismos de backup, recuperação e controle de concorrência.

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

O uso de operador lambda simplificou e reduziu a complexidade do código, principalmente em operações para manipulações em coleções, trazendo uma redução no código, deixando com uma sintaxe mais clara e concisa.

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

Métodos marcados como static são estáticos e pertencem à classe e não a uma instância da classe, sendo executados sem a criação de um objeto da classe. O método main é static, chamado pela Java Virtual Machine (JVM), qualquer método que for chamado pelo main precisa ser estático, garantindo que métodos que manipulam dados globais ou operações gerais não dependam do estado específico de uma instância. Não sendo um método estático é necessário criar de forma explícita um objeto para acessar esses métodos.