1. Título da Prática: **“1º Procedimento | Mapeamento Objeto-Relacional e DAO”**

2. Objetivo da prática :
    - Implementar persistência com base no middleware JDBC manuseio de dados;
    - 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;

3. Códigos Solicitados:

```java
package cadastrobd.model;

public class Pessoa {
    public String nome;
    public String logradouro;
    public String cidade;
    public String estado;
    public int telefone;
    public String email;
    public int id;

    public Pessoa() {
    }

    public Pessoa(String nome, String logradouro, String cidade, int telefone, String email, int id) {
        this.nome = nome;
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.telefone =  telefone;
        this.email = email;
        this.id = id ;
        this.estado = estado;

    }

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

    }
```

```java
package cadastrobd.model;


public class PessoaFisica extends Pessoa{
    int cpf;

    public PessoaFisica(){}

    public PessoaFisica(String nome, String logradouro, String cidade, int telefone, String email, int id, int cpf){
        super(nome,logradouro,cidade,telefone,email,id);
        this.cpf = cpf;
    }

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

    }

    int getId() {
        return super.id;
    }


}
```

```java
package cadastrobd.model;

import cadastrobd.model.util.ConectorBD;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import javax.sql.DataSource;

public class PessoaFisicaDAO {
    private ConectorBD conectorBD;
    private ArrayList<PessoaFisica> lista;

    public PessoaFisicaDAO(ConectorBD conectorBD) {
        this.conectorBD = conectorBD;
        this.lista = new ArrayList<>();
    }

    public ArrayList<PessoaFisica> readAll() throws ClassNotFoundException {
        try {
            String SQL = "SELECT * FROM pessoafisica";
            PreparedStatement ps = conectorBD.getDataSource().getConnection().prepareStatement(SQL);
            ResultSet rs = ps.executeQuery();

            ArrayList<PessoaFisica> lista = new ArrayList<>();

            while (rs.next()) {
                PessoaFisica pef = new PessoaFisica();

                pef.id = rs.getInt("id");
                pef.nome = rs.getString("nome");
                pef.cpf = rs.getInt("cpf");
                pef.telefone = rs.getInt("telefone");
                pef.email = rs.getString("email");
                pef.logradouro = rs.getString("logradouro");
                pef.cidade = rs.getString("cidade");
                pef.estado = rs.getString("estado");

                lista.add(pef);
            }
```

```
            }
            return lista;
        } catch (SQLException ex) {
            System.out.println("Erro ao recuperar " + ex.getMessage());
        }
        return null;
    }

    public void incluir(PessoaFisica pessoaFisica) throws ClassNotFoundException {
        try {
            String SQL = "INSERT INTO pessoafisica (nome, cpf, telefone, email, logradouro, cidade, estado) VALUES (?, ?, ?, ?, ?, ?, ?)";
            PreparedStatement ps = conectorBD.getDataSource().getConnection().prepareStatement(SQL);
            ps.setString(1, pessoaFisica.nome);
            ps.setInt(2, pessoaFisica.cpf);
            ps.setInt(3, pessoaFisica.telefone);
            ps.setString(4, pessoaFisica.email);
            ps.setString(5, pessoaFisica.logradouro);
            ps.setString(6, pessoaFisica.cidade);
            ps.setString(7, pessoaFisica.estado);
            ps.executeUpdate();
        } catch (SQLException ex) {
            System.out.println("Erro ao incluir " + ex.getMessage());
        }
    }

    public void alterar(PessoaFisica pessoaFisica) throws ClassNotFoundException {
        try {
            String SQL = "UPDATE pessoafisica SET nome=?, cpf=?, telefone=?, email=?, logradouro=?, cidade=?, estado=? WHERE id=?";
            PreparedStatement ps = conectorBD.getDataSource().getConnection().prepareStatement(SQL);
            ps.setString(1, pessoaFisica.nome);
            ps.setInt(2, pessoaFisica.cpf);
            ps.setInt(3, pessoaFisica.telefone);
            ps.setString(4, pessoaFisica.email);
            ps.setString(5, pessoaFisica.logradouro);
            ps.setString(6, pessoaFisica.cidade);
            ps.setString(7, pessoaFisica.estado);
            ps.setInt(8, pessoaFisica.id);
            ps.executeUpdate();
        } catch (SQLException ex) {
            System.out.println("Erro ao alterar " + ex.getMessage());
        }
    }

    public void excluir(int id) throws ClassNotFoundException {
        try {
            String SQL = "DELETE FROM pessoafisica WHERE id=?";
            PreparedStatement ps = conectorBD.getDataSource().getConnection().prepareStatement(SQL);
            ps.setInt(1, id);
            ps.executeUpdate();
        } catch (SQLException ex) {
            System.out.println("Erro ao excluir " + ex.getMessage());
        }
    }
}
```

```
package cadastrobd.model;

public class PessoaJuridica extends Pessoa {
    public int cnpj;

    public PessoaJuridica(){}

    public PessoaJuridica(String nome, String logradouro, String cidade, int telefone, String email, int id, int cnpj){
        super(nome,logradouro,cidade,telefone,email,id);
        this.cnpj = cnpj;
    }
    public void metodoExibir(){
        super.metodoExibir();
        System.out.println("CNPJ :"+ cnpj);
    }

    int getId() {
        return super.id;
    }
}
```

```java
package cadastrobd.model;

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

public class PessoaJuridicaDAO {

    public ArrayList<PessoaJuridica> lista;
    public Connection connection;
    public SequenceManager sequenceManager;

    public PessoaJuridicaDAO(Connection connection, SequenceManager sequenceManager) {
        this.lista = new ArrayList<>();
        this.connection = connection;
        this.sequenceManager = sequenceManager;
    }

    public ArrayList<PessoaJuridica> readAll() {
        try {
            String SQL = "SELECT * FROM pessoajuridica";
            PreparedStatement ps = connection.prepareStatement(SQL);
            ResultSet rs = ps.executeQuery();

            ArrayList<PessoaJuridica> lista = new ArrayList<>();

            while (rs.next()) {
                PessoaJuridica pj = new PessoaJuridica();

                pj.id = rs.getInt("id");
                pj.nome = rs.getString("nome");
                pj.cnpj = rs.getInt("cnpj");
                pj.telefone = rs.getInt("telefone");
                pj.email = rs.getString("email");
                pj.logradouro = rs.getString("logradouro");
                pj.cidade = rs.getString("cidade");
                lista.add(pj);
            }

            // Fechar recursos
            rs.close();
            ps.close();

            return lista;
        } catch (SQLException ex) {
            System.out.println("Erro ao recuperar: " + ex.getMessage());
        }
        return null;
    }

    public PessoaJuridica getPessoa(int id) {
        for (PessoaJuridica pessoaJuridica : lista) {
            if (pessoaJuridica.id == id) {
                return pessoaJuridica;
            }
        }
        return null;
    }

    public void incluir(PessoaJuridica pessoaJuridica) throws ClassNotFoundException {
        try {
            String SQL = "INSERT INTO pessoajuridica (nome, cnpj, telefone, email, logradouro, cidade, estado) VALUES (?, ?, ?, ?, ?, ?, ?)";
            PreparedStatement ps = connection.prepareStatement(SQL);
            ps.setString(1, pessoaJuridica.nome);
            ps.setInt(2, pessoaJuridica.cnpj);
            ps.setInt(3, pessoaJuridica.telefone);
            ps.setString(4, pessoaJuridica.email);
            ps.setString(5, pessoaJuridica.logradouro);
            ps.setString(6, pessoaJuridica.cidade);
            ps.setString(7, pessoaJuridica.estado);
            ps.executeUpdate();
            lista.add(pessoaJuridica);
        } catch (SQLException ex) {
            System.out.println("Erro ao incluir " + ex.getMessage());
        }
    }
```

```java
    public void alterar(PessoaJuridica pessoaJuridica) throws ClassNotFoundException {
        try {
            String SQL = "UPDATE pessoajuridica SET nome=?, cnpj=?, telefone=?, email=?, logradouro=?, cidade=?, estado=? WHERE id=?";
            PreparedStatement ps = connection.prepareStatement(SQL);
            ps.setString(1, pessoaJuridica.nome);
            ps.setInt(2, pessoaJuridica.cnpj);
            ps.setInt(3, pessoaJuridica.telefone);
            ps.setString(4, pessoaJuridica.email);
            ps.setString(5, pessoaJuridica.logradouro);
            ps.setString(6, pessoaJuridica.cidade);
            ps.setString(7, pessoaJuridica.estado);
            ps.setInt(8, pessoaJuridica.id);
            ps.executeUpdate();
            for (int i = 0; i < lista.size(); i++) {
                if (lista.get(i).id == pessoaJuridica.id) {
                    lista.set(i, pessoaJuridica);
                    break;
                }
            }
        } catch (SQLException ex) {
            System.out.println("Erro ao alterar " + ex.getMessage());
        }
    }

    public void excluir(int id) throws ClassNotFoundException {
        try {
            String SQL = "DELETE FROM pessoajuridica WHERE id=?";
            PreparedStatement ps = connection.prepareStatement(SQL);
            ps.setInt(1, id);
            ps.executeUpdate();
            lista.removeIf(pj -> pj.id == id);
        } catch (SQLException ex) {
            System.out.println("Erro ao excluir " + ex.getMessage());
        }
    }
}
```

```java
package cadastrobd.model.util;

import java.sql.Connection;
import java.io.IOException;
import java.sql.SQLException;
import org.apache.commons.dbcp2.BasicDataSource;
import javax.sql.DataSource;


public class ConectorBD {

    public static ConectorBD instance;
    public DataSource dataSource;

    public ConectorBD() {
        // Configurar o DataSource
        BasicDataSource basicDataSource = new BasicDataSource();
        basicDataSource.setDriverClassName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        basicDataSource.setUrl("jdbc:sqlserver://localhost:1433;databaseName=loja;ssl=require;trustServerCertificate=true");
        basicDataSource.setUsername("admin");
        basicDataSource.setPassword("felipe011");
        this.dataSource = basicDataSource;
    }

    public static synchronized ConectorBD getInstance() {
        if (instance == null) {
            instance = new ConectorBD();
        }
        return instance;
    }
    public Connection getConnection() throws SQLException {
    return dataSource.getConnection();
}

    public DataSource getDataSource() {
        return dataSource;
    }

    public void closeConnection(Connection connection) throws IOException, SQLException {
        if (connection != null) {
            connection.close();
        }
    }
}
```

```java
package cadastrobd.model.util;

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

public class SequenceManager {

    private ConectorBD conector;

    public SequenceManager(ConectorBD conector) {
        this.conector = conector;
    }

    public int getValue(String sequenceName) throws SQLException {
        Connection conn = null;
        PreparedStatement preparedStatement = null;
        ResultSet resultSet = null;
        int nextValue = -1;

        try {
            conn = conector.getDataSource().getConnection();
            String sql = "SELECT NEXT VALUE FOR " + sequenceName;
            preparedStatement = conn.prepareStatement(sql);
            resultSet = preparedStatement.executeQuery();
            if (resultSet.next()) {
                nextValue = resultSet.getInt(1);
            }
        } finally {

            if (resultSet != null) {
                resultSet.close();
            }
            if (preparedStatement != null) {
                preparedStatement.close();
            }
            if (conn != null) {
                conn.close();
            }
        }

        return nextValue;
    }
}
```

```java
package cadastrobd.model;

import cadastrobd.model.util.ConectorBD;
import cadastrobd.model.util.SequenceManager;
import java.io.IOException;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;

public class CadastroBDTeste {

    public static void main(String[] args) throws ClassNotFoundException, SQLException, IOException {
        ConectorBD conectorBD = ConectorBD.getInstance();

        testarPessoaFisica(conectorBD);
        testarPessoaJuridica(conectorBD);
    }

    private static void testarPessoaFisica(ConectorBD conectorBD) throws ClassNotFoundException, SQLException, IOException {
        Connection connection = conectorBD.getDataSource().getConnection();
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conectorBD);
        PessoaFisica pessoaFisica = new PessoaFisica("João", "Rua A", "Cidade A", 123456789, "joao@email.com", 1, 123456789);
        pessoaFisicaDAO.incluir(pessoaFisica);
        System.out.println("--------------------------------------------------------------------------------");
        System.out.println("Pessoa Fisica incluida com sucesso:");
        System.out.println("ID: " + pessoaFisica.id);
        System.out.println("Nome: " + pessoaFisica.nome);
        System.out.println("CPF: " + pessoaFisica.cpf);
        System.out.println("Logradouro: " + pessoaFisica.logradouro);
        System.out.println("Cidade: " + pessoaFisica.cidade);
        System.out.println("Estado: " + pessoaFisica.estado);
        System.out.println("--------------------------------------------------------------------------------");

        PessoaFisica novaPessoaFisica = new PessoaFisica("João da Silva", "Rua A", "Cidade A", 123456789, "joao@email.com", 1, 123456789);
        pessoaFisicaDAO.alterar(novaPessoaFisica);
        System.out.println("--------------------------------------------------------------------------------");
        System.out.println("Pessoa Fisica alterada com sucesso:");
        System.out.println("ID: " + novaPessoaFisica.id);
        System.out.println("Nome: " + novaPessoaFisica.nome);
        System.out.println("CPF: " + novaPessoaFisica.cpf);
        System.out.println("Logradouro: " + novaPessoaFisica.logradouro);
```

```
        System.out.println("Cidade: " + novaPessoaFisica.cidade);
        System.out.println("Estado: " + novaPessoaFisica.estado);
        System.out.println("--------------------------------------------------------------------------------");

        ArrayList<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.readAll();
        System.out.println("Listagem de Pessoas Fisicas:");
        for (PessoaFisica pf : pessoasFisicas) {
            System.out.println("--------------------------------------------------------------------------------");
            System.out.println("ID: " + pf.id);
            System.out.println("Nome: " + pf.nome);
            System.out.println("CPF: " + pf.cpf);
            System.out.println("Logradouro: " + pf.logradouro);
            System.out.println("Cidade: " + pf.cidade);
            System.out.println("Estado: " + pf.estado);


        }

        pessoaFisicaDAO.excluir(pessoaFisica.getId());
        System.out.println("Pessoa Fisica excluida com sucesso!");

        conectorBD.closeConnection(connection);
    }

    private static void testarPessoaJuridica(ConectorBD conectorBD) throws ClassNotFoundException, SQLException, IOException {
        Connection connection = conectorBD.getConnection();
        SequenceManager sequenceManager = new SequenceManager(conectorBD);

        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(connection, sequenceManager);

        PessoaJuridica pessoaJuridica = new PessoaJuridica("Empresa X", "Av. B", "Cidade B", 987654321, "empresa@email.com", 1, 987654321);
        pessoaJuridicaDAO.incluir(pessoaJuridica);
        System.out.println("--------------------------------------------------------------------------------");
        System.out.println("Pessoa Juridica incluida com sucesso:");
        System.out.println("ID: " + pessoaJuridica.id);
        System.out.println("CNPJ: " + pessoaJuridica.cnpj);
        System.out.println("Nome: " + pessoaJuridica.nome);
        System.out.println("Endereço: " + pessoaJuridica.logradouro);
        System.out.println("Cidade " + pessoaJuridica.cidade);
        System.out.println("Cidade " + pessoaJuridica.cidade);
        System.out.println("Estado " + pessoaJuridica.estado);
        System.out.println("--------------------------------------------------------------------------------");

        PessoaJuridica novaPessoaJuridica = new PessoaJuridica("Empresa Y", "Av. B", "Cidade B", 987654321, "empresa@email.com", 1, 987654321);
        pessoaJuridicaDAO.alterar(novaPessoaJuridica);
        System.out.println("--------------------------------------------------------------------------------");
        System.out.println("Pessoa Juridica alterada com sucesso:");
        System.out.println("ID: " + novaPessoaJuridica.id);
        System.out.println("Nome: " + novaPessoaJuridica.nome);
        System.out.println("CNPJ: " + novaPessoaJuridica.cnpj);
        System.out.println("Endereço: " + novaPessoaJuridica.logradouro);
        System.out.println("Cidade " + novaPessoaJuridica.cidade);
        System.out.println("Estado " + novaPessoaJuridica.estado);
        System.out.println("--------------------------------------------------------------------------------");

        ArrayList<PessoaJuridica> pessoasJuridicas = pessoaJuridicaDAO.readAll();
        System.out.println("--------------------------------------------------------------------------------");
        System.out.println("Listagem de Pessoas Juridicas:");
        for (PessoaJuridica pj : pessoasJuridicas) {
            System.out.println("ID: " + pj.id);
            System.out.println("Nome: " + pj.nome);
            System.out.println("CNPJ: " + pj.cnpj);
            System.out.println("Logradouro: " + pj.logradouro);
            System.out.println("Cidade " + pj.cidade);
            System.out.println("Estado " + pj.estado);

        }

        pessoaJuridicaDAO.excluir(pessoaJuridica.getId());
        System.out.println("--------------------------------------------------------------------------------");
        System.out.println("Pessoa Juridica excluida com sucesso!");

        conectorBD.closeConnection(connection);
    }
}
```

4. Os resultados da execução dos códigos também devem ser apresentados;
    a) Qual a importância dos componentes de middleware, como o JDBC?

        Middleware, como o JDBC, é essencial porque faz a ponte entre diferentes partes do software, tornando tudo mais fácil de integrar e gerenciar. Ele facilita a comunicação, padroniza processos e economiza tempo e esforço no desenvolvimento de aplicações.

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

        O Statement é bom para consultas simples e únicas, mas menos seguro e menos eficiente para consultas repetidas. Já o PreparedStatement é Melhor para consultas com parâmetros, mais seguro contra SQL Injection e mais eficiente para consultas 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, facilitando a reutilização de código, testabilidade e manutenção do sistema. Ele permite uma organização mais clara e modular, tornando o software mais fácil de evoluir e de manter ao longo do tempo.

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

Quando lidamos com herança em um modelo estritamente relacional, temos que mapear a hierarquia de classes de orientação a objetos para tabelas no banco de dados. Existem três abordagens principais para fazer isso: Tabela Única por Hierarquia (Single Table Inheritance)**,** Tabela por Subclasse (Class Table Inheritance) **e** Tabela por Concreta Classe (Concrete Table Inheritance)**.**

Relatório discente de acompanhamento

1. Título da Prática: “**2º Procedimento | Alimentando a Base**

2. Todos os códigos solicitados neste roteiro de aula:

```java
package cadastrobd.model;

import cadastrobd.model.util.ConectorBD;
import cadastrobd.model.util.SequenceManager;
import java.io.IOException;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.Scanner;

public class CadastroBDTeste {

    public static void main(String[] args) throws ClassNotFoundException, SQLException, IOException {
        ConectorBD conectorBD = ConectorBD.getInstance();
        Scanner scanner = new Scanner(System.in);
        int option;

        do {
            System.out.println("------------------------------------------------");
            System.out.println("Selecione uma opçao:");
            System.out.println("1 - Incluir");
            System.out.println("2 - Alterar");
            System.out.println("3 - Excluir");
            System.out.println("4 - Exibir pelo ID");
            System.out.println("5 - Exibir todos");
            System.out.println("0 - Sair");
            System.out.println("------------------------------------------------");
            option = scanner.nextInt();
            scanner.nextLine();

            switch (option) {
                case 1:
                    incluir(conectorBD, scanner);
                    break;
                case 2:
                    alterar(conectorBD, scanner);
                    break;
                case 3:
                    excluir(conectorBD, scanner);
                    break;
                case 4:
                    exibirPeloId(conectorBD, scanner);
                    break;
                case 5:
                    exibirTodos(conectorBD, scanner);
                    break;
                case 0:
                    System.out.println("Saindo...");
                    break;
                default:
                    System.out.println("Opçao invalida!");
                    break;
            }
        } while (option != 0);

        scanner.close();
    }

    private static void incluir(ConectorBD conectorBD, Scanner scanner) throws ClassNotFoundException, SQLException {
        System.out.println("Escolha o tipo: 1 - Fisica, 2 - Juridica");
        int tipo = scanner.nextInt();
        scanner.nextLine();

        if (tipo == 1) {
            PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conectorBD);
            PessoaFisica pessoaFisica = new PessoaFisica();

            System.out.println("Digite o nome:");
            pessoaFisica.nome
                    = scanner.nextLine();
            System.out.println("Digite o logradouro:");
            pessoaFisica.logradouro = scanner.nextLine();
            System.out.println("Digite a cidade:");
            pessoaFisica.cidade= scanner.nextLine();
            System.out.println("Digite o Estado:");
            pessoaFisica.estado = scanner.nextLine();
            System.out.println("Digite o CPF:");
            pessoaFisica.cpf = scanner.nextLong();
            scanner.nextLine();
            System.out.println("Digite o email:");
            pessoaFisica.email = scanner.nextLine();
            System.out.println("Digite o telefone:");
            pessoaFisica.telefone = scanner.nextLong();
            System.out.println("Digite o ID:");
            pessoaFisica.id = scanner.nextInt();
            pessoaFisicaDAO.incluir(pessoaFisica);
        } else if (tipo == 2) {
            SequenceManager sequenceManager = new SequenceManager(conectorBD);
            Connection connection = conectorBD.getConnection();
            PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(connection, sequenceManager);
            PessoaJuridica pessoaJuridica = new PessoaJuridica();
            System.out.println("Digite o nome:");
            pessoaJuridica.nome = scanner.nextLine();
            System.out.println("Digite o logradouro:");
            pessoaJuridica.logradouro = scanner.nextLine();
            System.out.println("Digite a cidade:");

            pessoaJuridica.cidade = scanner.nextLine();
            System.out.println("Digite o Estado:");
            pessoaJuridica.estado = scanner.nextLine();
            System.out.println("Digite o CNPJ:");
            pessoaJuridica.cnpj = scanner.nextLong();
            scanner.nextLine();
            System.out.println("Digite o email:");
            pessoaJuridica.email = scanner.nextLine();
            System.out.println("Digite o telefone:");
            pessoaJuridica.telefone = scanner.nextInt();
            System.out.println("Digite o ID:");
            pessoaJuridica.id = scanner.nextInt();
            pessoaJuridicaDAO.incluir(pessoaJuridica);
            connection.close();
        } else {
            System.out.println("Tipo invalido!");
        }
```

```
    }

    private static void alterar(ConectorBD conectorBD, Scanner scanner) throws ClassNotFoundException, SQLException {
        System.out.println("Escolha o tipo: 1 - Fisica, 2 - Juridica");
        int tipo = scanner.nextInt();
        scanner.nextLine(); // Consumir a nova linha

        if (tipo == 1) {
            PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conectorBD);
            System.out.println("Digite o ID:");
            int id = scanner.nextInt();
            scanner.nextLine();
            PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);
            if (pessoaFisica == null) {
                System.out.println("Pessoa Fisica nao encontrada!");
                return;
            }
            System.out.println("Nome atual: " + pessoaFisica.nome);
            System.out.println("Digite o novo nome:");
            pessoaFisica.nome = scanner.nextLine();
            System.out.println("CPF atual: " + pessoaFisica.cpf);
            pessoaFisica.cpf = scanner.nextLong();
            System.out.println("Telefone atual: " + pessoaFisica.telefone);
            pessoaFisica.telefone = scanner.nextLong();
            scanner.nextLine();
            System.out.println("Email atual: " + pessoaFisica.email);
            pessoaFisica.email = scanner.nextLine();
            System.out.println("Endereço atual: " + pessoaFisica.logradouro + ", " + pessoaFisica.cidade + ", " + pessoaFisica.estado);
            System.out.println("Digite o novo logradouro:");
            pessoaFisica.logradouro = scanner.nextLine();
            System.out.println("Digite uma nova cidade:");
            pessoaFisica.cidade = scanner.nextLine();
            System.out.println("Digite um novo estado:");
            pessoaFisica.estado = scanner.nextLine();
            pessoaFisicaDAO.alterar(pessoaFisica);
        } else if (tipo == 2) {
            SequenceManager sequenceManager = new SequenceManager(conectorBD);
            Connection connection = conectorBD.getConnection();
            PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(connection, sequenceManager);
            System.out.println("Digite o ID:");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consumir a nova linha
            PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);
            if (pessoaJuridica == null) {
                System.out.println("Pessoa Jurídica não encontrada!");
                return;
            }
            System.out.println("Nome atual: " + pessoaJuridica.nome);
            System.out.println("Digite o novo nome:");
            pessoaJuridica.nome = scanner.nextLine();
            System.out.println("CNPJ atual: " + pessoaJuridica.cnpj);
            pessoaJuridica.cnpj = scanner.nextLong();
            System.out.println("Telefone atual: " + pessoaJuridica.telefone);
            pessoaJuridica.telefone = scanner.nextLong();
            scanner.nextLine();
            System.out.println("Email atual: " + pessoaJuridica.email);
            pessoaJuridica.email = scanner.nextLine();
            System.out.println("Endereço atual: " + pessoaJuridica.logradouro + ", " + pessoaJuridica.cidade + ", " + pessoaJuridica.estado);
            System.out.println("Digite o novo logradouro:");
            pessoaJuridica.logradouro = scanner.nextLine();
            System.out.println("Digite uma nova cidade:");
            pessoaJuridica.cidade = scanner.nextLine();
            System.out.println("Digite um novo estado:");
            pessoaJuridica.estado = scanner.nextLine();
            pessoaJuridicaDAO.alterar(pessoaJuridica);
            connection.close();
        } else {
            System.out.println("Tipo invalido!");
        }
    }

    private static void excluir(ConectorBD conectorBD, Scanner scanner) throws ClassNotFoundException, SQLException {
        System.out.println("Escolha o tipo: 1 - Fisica, 2 - Juridica");
        int tipo = scanner.nextInt();
        scanner.nextLine();

        if (tipo == 1) {
            PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conectorBD);
            System.out.println("Digite o ID:");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consumir a nova linha
            pessoaFisicaDAO.excluir(id);
        } else if (tipo == 2) {
            SequenceManager sequenceManager = new SequenceManager(conectorBD);
            Connection connection = conectorBD.getConnection();
            PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(connection, sequenceManager);
            System.out.println("Digite o ID:");
            int id = scanner.nextInt();
            scanner.nextLine();
            pessoaJuridicaDAO.excluir(id);
            connection.close();
        } else {
            System.out.println("Tipo inválido!");
        }
    }

    private static void exibirPeloId(ConectorBD conectorBD, Scanner scanner) throws ClassNotFoundException, SQLException {
        System.out.println("Escolha o tipo: 1 - Fisica, 2 - Juridica");
        int tipo = scanner.nextInt();
        scanner.nextLine(); // Consumir a nova linha

        if (tipo == 1) {
            PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conectorBD);
            System.out.println("Digite o ID:");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consumir a nova linha
            PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);
            if (pessoaFisica != null) {
                System.out.println("ID: " + pessoaFisica.id);
                System.out.println("Nome: " + pessoaFisica.nome);
                System.out.println("CPF: " + pessoaFisica.cpf);
                System.out.println("Endereço: " + pessoaFisica.logradouro + ", " + pessoaFisica.cidade + ", " + pessoaFisica.estado);

            } else {
                System.out.println("Pessoa Fisica nao encontrada!");
            }
        } else if (tipo == 2) {
```

```
            SequenceManager sequenceManager = new SequenceManager(conectorBD);
            Connection connection = conectorBD.getConnection();
            PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(connection, sequenceManager);
            System.out.println("Digite o ID:");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consumir a nova linha
            PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);
            if (pessoaJuridica != null) {
                System.out.println("ID: " + pessoaJuridica.id);
                System.out.println("Nome: " + pessoaJuridica.nome);
                System.out.println("CNPJ: " + pessoaJuridica.cnpj);
                System.out.println("Endereço: " + pessoaJuridica.logradouro + ", " + pessoaJuridica.cidade + ", " + pessoaJuridica.estado);

            } else {
                System.out.println("Pessoa Juridica nao encontrada!");
            }
            connection.close();
        } else {
            System.out.println("Tipo invalido!");
        }
    }

    private static void exibirTodos(ConectorBD conectorBD, Scanner scanner) throws ClassNotFoundException, SQLException {
        System.out.println("Escolha o tipo: 1 - Fisica, 2 - Juridica");
        int tipo = scanner.nextInt();
        scanner.nextLine();

        if (tipo == 1) {
            PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conectorBD);
            ArrayList<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.readAll();
            for (PessoaFisica pf : pessoasFisicas) {
                System.out.println("ID: " + pf.id);
                System.out.println("Nome: " + pf.nome);
                System.out.println("CPF: " + pf.cpf);
                System.out.println("Endereço: " + pf.logradouro + ", " + pf.cidade + ", " + pf.estado);

            }
        } else if (tipo == 2) {
            SequenceManager sequenceManager = new SequenceManager(conectorBD);
            Connection connection = conectorBD.getConnection();
            PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(connection, sequenceManager);
            ArrayList<PessoaJuridica> pessoasJuridicas = pessoaJuridicaDAO.readAll();
            for (PessoaJuridica pj : pessoasJuridicas) {
                System.out.println("ID: " + pj.id);
                System.out.println("Nome: " + pj.nome);
                System.out.println("CNPJ: " + pj.cnpj);
                System.out.println("Endereço: " + pj.logradouro + ", " + pj.cidade + ", " + pj.estado);
            }
            connection.close();
        } else {
            System.out.println("Tipo invalido!");
        }
    }
}
```

3. Os resultados da execução dos códigos também devem ser apresentados

```
-----------------------------------------------
Selecione uma op□ao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
0 - Sair
-----------------------------------------------
5
Escolha o tipo: 1 - Fisica, 2 - Juridica
1
ID: 1010
Nome:
CPF: 3
Endere□o: , ,
ID: 1011
Nome: sarah
CPF: 67969696
Endere□o: rua tanana, ali logo ali, sp
-----------------------------------------------
Selecione uma op□ao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
0 - Sair
-----------------------------------------------
```

4. Análise e Conclusão:
    a) Quais as diferenças entre a persistência em arquivo e a persistência em banco de dados?

    A persistência em arquivos são simples e bons para pequenos volumes de dados; já em bancos de dados são melhores para grandes volumes e operações complexas.

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

    O uso de expressões lambda no Java, a partir da versão 8, simplificou a impressão dos valores contidos nas entidades. Antes, era usado loops explícitos para iterar e imprimir valores. Já os Lambdas reduzem o código e melhoram a legibilidade ao permitir uma sintaxe mais curta e clara para operações comuns.

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

    Quando um método é chamado diretamente pelo método main em Java, ele precisa ser marcado como static porque o main é estático e pertence à classe, não a uma instância específica. Isso permite que o método seja acessado sem a necessidade de criar uma instância da classe, garantindo a correta execução do programa.