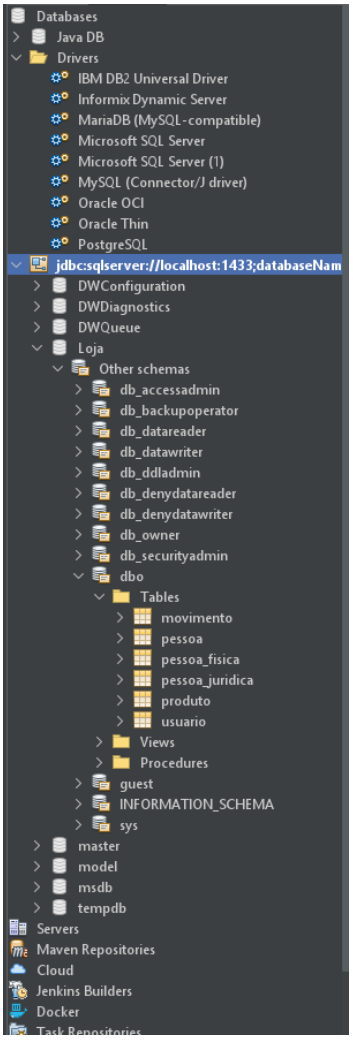
Nome do campus: Polo Pirituba
Nome do curso: Desenvolvimento Full Stack
Nome da disciplina: Nível 3: BackEnd sem banco não tem
Número da turma: EAD - 9001
Semestre letivo: 2025.1 3 semestre
Nome do integrante da prática: Guilherme Wissenbach Ferreira
Repositório Github: https://github.com/guilherme-wissenbach/college/tree/main/Terceiro%20semestre/Nivel%203

Título da prática: Missão Prática | Nível 3 | Mundo 3

Objetivo da prática:

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

Criar o projeto e configurar as bibliotecas necessárias:



Classe Pessoa

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

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

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

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

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

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

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

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

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

```
    }

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

    public int getId() {
        return id;
    }

    public String getNome() {
        return nome;
    }

    public String getLogradouro() {
        return logradouro;
    }

    public String getCidade() {
        return cidade;
    }

    public String getEstado() {
        return estado;
    }

    public String getTelefone() {
        return telefone;
    }

    public String getEmail() {
        return email;
    }

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

Classe PessoaFisica

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

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

   private String cpf;

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

    public String getCpf() {
      return cpf;
   }

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

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

Classe PessoaJuridica

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

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

   private String cnpj;

   public PessoaJuridica(int id, String nome, String logradouro, String cidade, String estado,
         String telefone, String email, String cnpj){
      super(id, nome, logradouro, cidade, estado, telefone, email);
      this.cnpj = cnpj;
   }

   public String getCnpj() {
      return cnpj;
   }

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

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

Classe ConectorBD

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

/**
 *
 * @author Guilherme
 */

 public class ConectorBD {

   Connection conn = null;

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


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

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

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

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

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

}
```

Classe SequenceManager

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


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

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

Classe PessoaFisicaDAO

```
package cadastro.model;

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

/**
 *
 * @author Guilherme
 */
public class PessoaFisicaDAO {

   public ConectorBD connection = new ConectorBD();

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


   public List<PessoaFisica> getPessoas() throws Exception{
      List<PessoaFisica> lista = new ArrayList<>();
               String sql = "select *\n" +
               "from pessoa, pessoa_fisica\n" +
               "where pessoa.id_pessoa = pessoa_fisica.id_pessoa;";
         PreparedStatement ps = connection.getPrepared(sql);
         ResultSet resultado = ps.executeQuery();
         while(resultado.next()){
```

```
            //System.out.println(resultado.getString(5));
            lista.add(new PessoaFisica(resultado.getInt("id_pessoa"),
            resultado.getString("nome"),
            resultado.getString("logradouro"),
            resultado.getString("cidade"),
            resultado.getString("estado"),
            resultado.getString("telefone"),
            resultado.getString("email"),
            resultado.getString("cpf")));
            connection.closeConnection();
            //connection.closeResult(ps);
            connection.closeStatement(sql);
        } return lista;

    }


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

    public void alterar(int id, String cpf, String nome, String logradouro,
        String cidade, String estado, String telefone, String email) throws Exception {
    PessoaFisica pessoa = getPessoa(id);
    if (pessoa == null) {
        throw new Exception("Pessoa não encontrada!" +id);
    }

    String sqlfisica = "UPDATE pessoa_fisica SET cpf=? WHERE id_pessoa = ?";
    String sqlpessoa = "UPDATE pessoa SET nome=?, logradouro=?, cidade=?, estado=?, telefone=?,
email=? WHERE id_pessoa= ?";
```

```
        try (PreparedStatement ps = connection.getPrepared(sqlfisica);
      PreparedStatement ps1 = connection.getPrepared(sqlpessoa)) {

    ps.setString(1, cpf.isEmpty() ? pessoa.getCpf() : cpf);
    ps.setInt(2, id);
    ps.executeUpdate();

    ps1.setString(1, nome.isEmpty() ? pessoa.getNome() : nome);
    ps1.setString(2, logradouro.isEmpty() ? pessoa.getLogradouro() : logradouro);
    ps1.setString(3, cidade.isEmpty() ? pessoa.getCidade() : cidade);
    ps1.setString(4, estado.isEmpty() ? pessoa.getEstado() : estado);
    ps1.setString(5, telefone.isEmpty() ? pessoa.getTelefone() : telefone);
    ps1.setString(6, email.isEmpty() ? pessoa.getEmail() : email);
    ps1.setInt(7, id);
    ps1.executeUpdate();

  } finally {
    connection.closeConnection();
  }
}

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

}
```

Classe PessoaJuridicaDAO

```
package cadastro.model;

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

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

   public ConectorBD connection = new ConectorBD();

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

   public List<PessoaJuridica> getPessoas() throws Exception{
      List<PessoaJuridica> lista = new ArrayList<>();
      String sql = "select *\n" +
                   "from pessoa, pessoa_juridica\n" +
                   "where pessoa.id_pessoa = pessoa_juridica.id_pessoa;";
         PreparedStatement ps = connection.getPrepared(sql);
         ResultSet resultado = ps.executeQuery();
         while(resultado.next()){
            //System.out.println(resultado.getString(5));
            lista.add(new PessoaJuridica(resultado.getInt("id_pessoa"),
            resultado.getString("nome"),
```

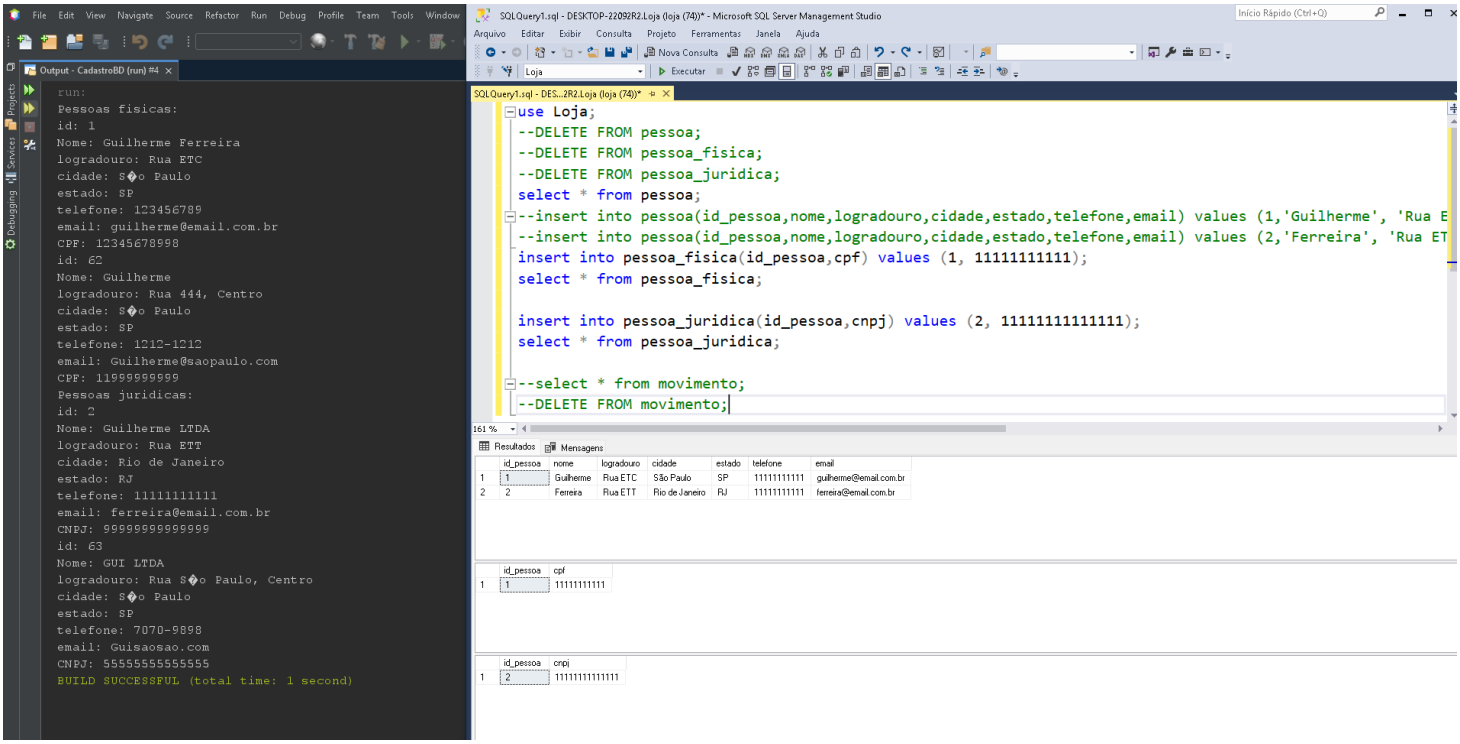
```
resultado.getString("logradouro"),
            resultado.getString("cidade"),
            resultado.getString("estado"),
            resultado.getString("telefone"),
            resultado.getString("email"),
            resultado.getString("cnpj")));
            connection.closeConnection();
            //connection.closeResult(ps);
            connection.closeStatement(sql);
        } return lista;

  }

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

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

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

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

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

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

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

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

}
```

Classe CadastroBDTeste

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

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */

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

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

        PessoaFisica pessoaIncluir = new PessoaFisica(seq.getValue("seq_Pessoa"),"Guilherme", "Rua 444, Centro",
        "São Paulo", "SP", "1212-1212","Guilherme@saopaulo.com","11999999999");
        PessoaFisicaDAO pessoaPF = new PessoaFisicaDAO();
        pessoaPF.incluir(pessoaIncluir);

        // b.Alterar os dados da pessoa física no banco.
        // Alterando pessoa e pessoaFisica pelo id--> 3 . Mudando nome, cpf e telefone
       PessoaFisicaDAO pessoaPF1 = new PessoaFisicaDAO();
       pessoaPF1.alterar( 1,  "12345678998",  "Guilherme Ferreira",  "", "",  "", "123456789",  "");

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

      //d. Excluir a pessoa física criada anteriormente no banco.
       // Excluindo pessoaFisica e Pessoa pelo id.
       PessoaFisicaDAO pessoaPF2 = new PessoaFisicaDAO();
       pessoaPF2.excluir(1);
```

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


    //Incluir pessoa juridica e pessoa
    PessoaJuridica pessoaIncluir2 = new PessoaJuridica(seq.getValue("seq_Pessoa"),"GUI LTDA",
        "Rua São Paulo, Centro","São Paulo", "SP", "7070-9898","Guisaosao.com","55555555555555");
    PessoaJuridicaDAO pessoaPJ = new PessoaJuridicaDAO();
    pessoaPJ.incluir(pessoaIncluir2);


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


    // Alterando pessoa e pessoaJuridica pelo id--> 4 . Mudando nome e cnpj
    PessoaJuridicaDAO pessoaPJ2 = new PessoaJuridicaDAO();
    pessoaPJ2.alterar( 2,  "99999999999999",  "Guilherme LTDA ",  "",  "",  "",  "",  "");


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


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


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


    // Excluindo pessoa juridida e Pessoa pelo id.
     PessoaJuridicaDAO pessoaPJ3 = new PessoaJuridicaDAO();
     pessoaPJ3.excluir(2);

   }
}
```



Análise e Conclusão:

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

Abstração do acesso ao banco de dados, independência de banco de dados, gerenciamento de conexões, segurança e controle e facilita a manutenção e evolução

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

A diferença entre o uso de Statement e PreparedStatement no JDBC está relacionada principalmente à segurança, eficiência e facilidade de uso. Ambos servem para executar comandos SQL no banco de dados, mas possuem comportamentos distintos que afetam diretamente o desempenho e a proteção da aplicação contra vulnerabilidades como SQL Injection.

Como o padrão DAO melhora a manutenibilidade do software?

Melhora significativamente a manutenibilidade do software ao isolar a lógica de acesso a dados do restante da aplicação. Ele faz parte das boas práticas de design de software, principalmente em aplicações que acessam bancos de dados.

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

A herança é refletida por meio de Identificadores em comum. Onde há uma tabela base (Pessoa) e outras derivadas (Pessoa Fisica e Pessoa Juridica).

Classe CadastroBDTeste2 (Procedimento 2)

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

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */

/**
 *
 * @author Guilherme
 */
public class CadastroBDTeste2 {

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

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

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

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

            switch (escolha) {

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

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

```
switch (escolha.toUpperCase()) {

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

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

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

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

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

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

```
            case "M":
                break;

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

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

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

        switch (escolha.toUpperCase()) {

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

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

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

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

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

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

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

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

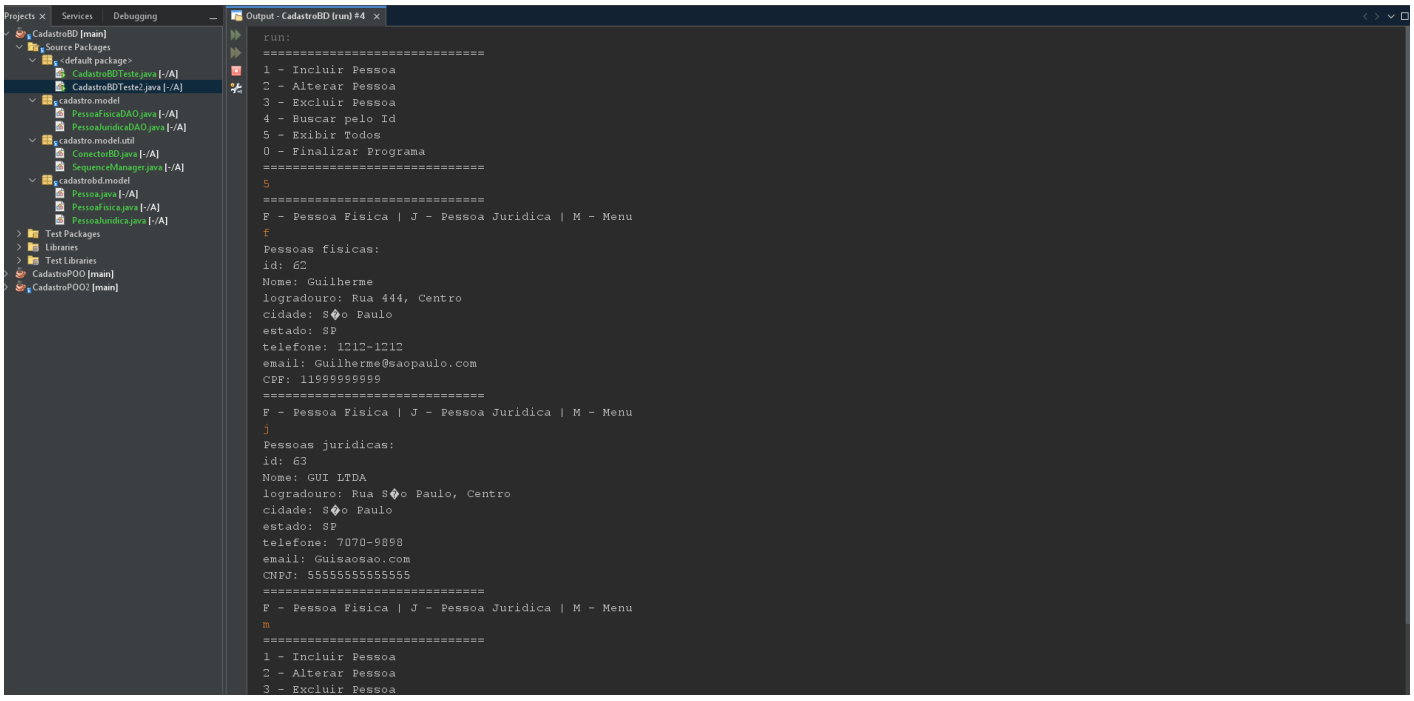
```
        System.out.println("Email: " + pessoaFisicaLocalizada.getEmail());
        System.out.print("Novo Email: ");
        String novoEmail = scan.nextLine();

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

            case "M":
              break;

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

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

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

    switch (escolha.toUpperCase()) {

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

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

        if (pessoaFisicaLocalizada != null) {
          pessoaFisicaLocalizada.exibir();
          pessoaFisicaLocalizadaExcluir.excluir(idPessoaFisica);

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

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

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

                if (pessoaJuridicaLocalizada != null) {
                    pessoaJuridicaLocalizada.exibir();
                    pessoaJurdicaLocalizadaExcluir.excluir(idPessoaJuridica);

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

            case "M":
                break;

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

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

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

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

        switch (escolha.toUpperCase()) {

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

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

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

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

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

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

            case "M":
                break;

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

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

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

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

        switch (escolha.toUpperCase()) {

            case "F":
                System.out.println("Pessoas fisicas:");
                PessoaFisicaDAO pessoasFisica = new PessoaFisicaDAO();
                List<PessoaFisica> resultado = pessoasFisica.getPessoas();
                for (PessoaFisica pessoaFisica : resultado) {
                    pessoaFisica.exibir();
                }
                break;

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

```
                case "M":
                    break;

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

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

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

 }
}
```



Estácio

Análise e Conclusão:

Quais as diferenças entre a persistência em arquivo e a persistência em banco de dados?
A persistência de dados é o processo de salvar informações de forma que possam ser recuperadas e reutilizadas depois que o programa termina e a persistência em banco de dados é o processo de armazenar dados de forma permanente em um sistema de gerenciamento de banco de dados.

Como o uso de operador lambda simplificou a impressão dos valores contidos nas entidades, nas versões mais recentes do Java?
É um mecanismo bastante poderoso, que facilita muito a escrita de código conciso e evita que o programador seja obrigado a escrever um monte de código "inútil", principalmente em operações simples, além de flexibilizar o mesmo.

Por que métodos acionados diretamente pelo método main, sem o uso de um objeto, precisam ser marcados como static?
Porque métodos Static se refere à classe e executam quando a classe for carregada e também o método main em Java também é static, e um método static só pode chamar diretamente outros métodos static sem criar objetos.