Missão Prática | Nível 3 | Mundo 3

Italo Augusto Juliano Barbosa - 202303617674

Campus Aparecida de Goiânia
Back end sem banco não tem – Turma: 9001 – 3° Semestre
Link do repositório no GitHub: https://github.com/Anarquia122/trabalho-facul-cadastroBD

Objetivos 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 o uso do SQL Server no uso de persistência de dados.

1º Procedimento | Mapeamento Objeto-Relacional e DAO

Pessoa.java:

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

    public Pessoa() {}
```

```
    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;
    }

    //Getters

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

    public String getNome() {
        return this.nome;
    }

    public String getLogradouro() {
        return this.logradouro;
    }

    public String getCidade() {
        return this.cidade;
    }

    public String getEstado() {
        return this.estado;
    }

    public String getTelefone() {
        return this.telefone;
    }

    public String getEmail() {
```

```
    return this.email;
}

//Setters

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 void exibir() {
    System.out.println("--------------------");
    System.out.println("ID: "+this.id);
    System.out.println("Nome: "+this.nome);
    System.out.println("Logradouro: "+this.logradouro);
    System.out.println("Cidade: "+this.cidade);
```

```
        System.out.println("Estado: "+this.estado);
        System.out.println("Telefone: "+this.telefone);
        System.out.println("E-mail: "+this.email);
    }
}
```

PessoaFisica.java:

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

    public PessoaFisica() {
        super();
    }

    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;
    }

    //Getter

    public String getCpf() {
        return this.cpf;
    }

    //Setter

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

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

PessoaJuridica.java:

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

   public PessoaJuridica() {
      super();
   }

   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;
   }

   //Getter

   public String getCnpj() {
      return this.cnpj;
   }

   //Setter

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

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

ConectorBD.java:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.lang.ClassNotFoundException;



public class ConectorBD {


   private static final String url = "jdbc:sqlserver://localhost:1433;databaseName=Loja;encrypt=true;trustServerCertificate=true;user=sa;password=floresta2";
   private static final String usuario = "user=loja;";
   private static final String senha = "password=loja;";


   public static Connection getConnection() throws SQLException {
      try {
         Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
      } catch (ClassNotFoundException e) {
         System.out.println("Erro. Classe não encontrada: ");
         e.printStackTrace();
      }
      return DriverManager.getConnection(url);
   }


   public static PreparedStatement getPrepared(String sql) throws SQLException {
      Connection conn = getConnection();
      return conn.prepareStatement(sql);
   }


   public static ResultSet getSelect(String sql) throws SQLException {
      PreparedStatement ps = getPrepared(sql);
      return ps.executeQuery();
   }


   public static void close(Connection conn) {
      if (conn != null) {
         try {
            conn.close();
         } catch (SQLException e) {
            e.printStackTrace();
```

```
            }
        }
    }

    public static void close(Statement stmt) {
        if (stmt != null) {
            try {
                stmt.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    public static void close(ResultSet rs) {
        if (rs != null) {
            try {
                rs.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

SequenceManager.java:

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

public class SequenceManager {

    public static int getValue(String sequenceName) {
        int nextValue = 0;
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
```

```
    try {
        conn = ConectorBD.getConnection();
        String sql = "SELECT NEXT VALUE FOR " + sequenceName + " AS next_value";
        ps = conn.prepareStatement(sql);
        rs = ps.executeQuery();

        if (rs.next()) {
            nextValue = rs.getInt("next_value");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        ConectorBD.close(conn);
        ConectorBD.close(rs);
        ConectorBD.close(ps);
    }

    return nextValue;
}

public static int getNextValueA(String tableName, String idColumnName) {
    int nextId = 1;
    Connection conn = null;
    PreparedStatement ps = null;
    ResultSet rs = null;

    try {
        conn = ConectorBD.getConnection();
        String sql = "SELECT MAX (" + idColumnName + ") AS max_id FROM " + tableName;
        ps = conn.prepareStatement(sql);
        rs = ps.executeQuery();

        if (rs.next()) {
            int maxId = rs.getInt("max_id");
            nextId = maxId + 1;
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
```

```
            ConectorBD.close(conn);
            ConectorBD.close(ps);
            ConectorBD.close(rs);
        }

        return nextId;
    }
}
```

PessoaFisicaDAO.java:

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.sql.Statement;

public class PessoaFisicaDAO {

    public PessoaFisica getPessoa(int id) {
        PessoaFisica pessoa = null;
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;

        try {
            conn = ConectorBD.getConnection();
            String sqlPessoaFisica = "SELECT * FROM PessoaFisica WHERE Pessoa_idPessoa = ?";
            ps = conn.prepareStatement(sqlPessoaFisica);
            ps.setInt(1, id);
            rs = ps.executeQuery();

            pessoa = new PessoaFisica();

            if (rs.next()) {
                try {
                    pessoa.setId(rs.getInt("Pessoa_idPessoa"));
                    pessoa.setCpf(rs.getString("cpf"));
```

```
            } catch (SQLException e) {
                System.out.println("Erro ao receber os dados da tabela PessoaFisica: ");
                e.printStackTrace();
            }
        }

        String sqlPessoa = "SELECT * FROM Pessoa WHERE idPessoa = ?";
        ps = conn.prepareStatement(sqlPessoa);
        ps.setInt(1, id);
        rs = ps.executeQuery();

        if (rs.next()) {
            try {
                pessoa.setNome(rs.getString("nome"));
                pessoa.setLogradouro(rs.getString("logradouro"));
                pessoa.setCidade(rs.getString("cidade"));
                pessoa.setEstado(rs.getString("estado"));
                pessoa.setTelefone(rs.getString("telefone"));
                pessoa.setEmail(rs.getString("email"));
            } catch (SQLException e) {
                System.out.println("Erro ao receber os dados da tabela Pessoa: ");
                e.printStackTrace();
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        ConectorBD.close(rs);
        ConectorBD.close(ps);
        ConectorBD.close(conn);
    }

    return pessoa;
}

public List<PessoaFisica> getPessoas() {
    List<PessoaFisica> pessoas = new ArrayList<>();
    Connection conn = null;
    PreparedStatement ps = null;
```

```
        ResultSet rs = null;

        try {
            conn = ConectorBD.getConnection();
            String sql = "SELECT p.idPessoa, p.nome, p.logradouro, p.cidade, p.estado, p.telefone, p.email,
pf.cpf FROM Pessoa p INNER JOIN PessoaFisica pf ON p.idPessoa = pf.Pessoa_idPessoa";
            ps = conn.prepareStatement(sql);
            rs = ps.executeQuery();

            while (rs.next()) {
                PessoaFisica pessoa = new PessoaFisica();
                pessoa.setId(rs.getInt("idPessoa"));
                pessoa.setNome(rs.getString("nome"));
                pessoa.setLogradouro(rs.getString("logradouro"));
                pessoa.setCidade(rs.getString("cidade"));
                pessoa.setEstado(rs.getString("estado"));
                pessoa.setTelefone(rs.getString("telefone"));
                pessoa.setEmail(rs.getString("email"));
                pessoa.setCpf(rs.getString("cpf"));

                pessoas.add(pessoa);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(rs);
            ConectorBD.close(ps);
            ConectorBD.close(conn);
        }

        return pessoas;
    }

    public void inserir(PessoaFisica pessoa) {
        Connection conn = null;
        PreparedStatement ps = null;
        //ResultSet rs = null;

        try {
```

```
            conn = ConectorBD.getConnection();
            conn.setAutoCommit(false);

            //int pessoaId = SequenceManager.getValue("seq_pessoa");
            int pessoaId = SequenceManager.getNextValueA("Pessoa", "idPessoa");

            String sqlPessoa = "INSERT INTO Pessoa (idPessoa, nome, logradouro, cidade, estado, telefone,
email) VALUES (?, ?, ?, ?, ?, ?, ?)";
            ps = conn.prepareStatement(sqlPessoa, PreparedStatement.RETURN_GENERATED_KEYS);
            ps.setInt(1, pessoaId);
            ps.setString(2, pessoa.getNome());
            ps.setString(3, pessoa.getLogradouro());
            ps.setString(4, pessoa.getCidade());
            ps.setString(5, pessoa.getEstado());
            ps.setString(6, pessoa.getTelefone());
            ps.setString(7, pessoa.getEmail());
            ps.executeUpdate();

            //rs = ps.getGeneratedKeys();
            //int pessoaId = 0;
            //if (rs.next()) {
               //pessoaId = rs.getInt(1);
            //}

            int pessoaFisicaId = SequenceManager.getNextValueA("PessoaFisica", "idPessoaFisica");

            String sqlPessoaFisica = "INSERT INTO PessoaFisica (idPessoaFisica, Pessoa_idPessoa, cpf)
VALUES (?, ?, ?)";
            ps = conn.prepareStatement(sqlPessoaFisica);
            ps.setInt(1, pessoaFisicaId);
            ps.setInt(2, pessoaId);
            ps.setString(3, pessoa.getCpf());
            ps.executeUpdate();

            conn.commit();
        } catch (SQLException e) {
            if (conn != null) {
               try {
                  conn.rollback();
               } catch (SQLException ex) {
```

```
                System.out.println("Erro ao tentar o rollback: ");
                ex.printStackTrace();
            }
        }
        e.printStackTrace();
    } finally {
        ConectorBD.close(conn);
        ConectorBD.close(ps);
        //ConectorBD.close(rs);
    }
  }

  public void alterar(PessoaFisica pessoa) {
    Connection conn = null;
    PreparedStatement ps = null;

    try {
        int idPessoa = pessoa.getId();

        conn = ConectorBD.getConnection();
        String sqlPessoa = "UPDATE Pessoa SET nome=?, logradouro=?, cidade=?, estado=?, telefone=?, email=? WHERE idPessoa=?";
        ps = conn.prepareStatement(sqlPessoa);
        ps.setString(1, pessoa.getNome());
        ps.setString(2, pessoa.getLogradouro());
        ps.setString(3, pessoa.getCidade());
        ps.setString(4, pessoa.getEstado());
        ps.setString(5, pessoa.getTelefone());
        ps.setString(6, pessoa.getEmail());
        ps.setInt(7, idPessoa);
        ps.executeUpdate();

        String sqlPessoaFisica = "UPDATE PessoaFisica SET cpf=? WHERE Pessoa_idPessoa=?";
        ps = conn.prepareStatement(sqlPessoaFisica);
        ps.setString(1, pessoa.getCpf());
        ps.setInt(2, idPessoa);
        ps.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
```

```
    } finally {
        ConectorBD.close(conn);
        ConectorBD.close(ps);
    }
}

public void excluir(int id) {
    Connection conn = null;
    PreparedStatement ps = null;

    try {
        conn = ConectorBD.getConnection();
        conn.setAutoCommit(false);

        String sqlPessoaFisica = "DELETE FROM PessoaFisica WHERE Pessoa_idPessoa = ?";
        ps = conn.prepareStatement(sqlPessoaFisica);
        ps.setInt(1, id);
        ps.executeUpdate();

        String sqlPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";
        ps = conn.prepareStatement(sqlPessoa);
        ps.setInt(1, id);
        ps.executeUpdate();

        conn.commit();
    } catch (SQLException e) {
        if (conn != null) {
            try {
                conn.rollback();
            } catch (SQLException ex) {
                System.out.println("Erro ao tentar rollback: ");
                ex.printStackTrace();
            }
        }
        e.printStackTrace();
    } finally {
        ConectorBD.close(conn);
        ConectorBD.close(ps);
```

```
        }
    }
}
```

PessoaJuridicaDAO.java:

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class PessoaJuridicaDAO {

    public PessoaJuridica getPessoa(int id) {
        PessoaJuridica pessoa = null;
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;

        try {
            conn = ConectorBD.getConnection();
            String sqlPessoaJuridica = "SELECT * FROM PessoaJuridica WHERE Pessoa_idPessoa = ?";
            ps = conn.prepareStatement(sqlPessoaJuridica);
            ps.setInt(1, id);
            rs = ps.executeQuery();

            pessoa = new PessoaJuridica();

            if (rs.next()) {
                try {
                    pessoa.setId(rs.getInt("Pessoa_idPessoa"));
                    pessoa.setCnpj(rs.getString("cnpj"));
                } catch (SQLException e) {
                    System.out.println("Erro ao receber os dados da tabela PessoaJuridica: ");
                    e.printStackTrace();
                }
            }
```

```
            String sqlPessoa = "SELECT * FROM Pessoa WHERE idPessoa = ?";
            ps = conn.prepareStatement(sqlPessoa);
            ps.setInt(1, id);
            rs = ps.executeQuery();

            if (rs.next()) {
               try {
                  pessoa.setNome(rs.getString("nome"));
                  pessoa.setLogradouro(rs.getString("logradouro"));
                  pessoa.setCidade(rs.getString("cidade"));
                  pessoa.setEstado(rs.getString("estado"));
                  pessoa.setTelefone(rs.getString("telefone"));
                  pessoa.setEmail(rs.getString("email"));
               } catch (SQLException e) {
                  System.out.println("Erro ao receber os dados da tabela Pessoa: ");
                  e.printStackTrace();
               }
            }
         } catch (SQLException e) {
            e.printStackTrace();
         } finally {
            ConectorBD.close(rs);
            ConectorBD.close(ps);
            ConectorBD.close(conn);
         }

         return pessoa;
      }

      public List<PessoaJuridica> getPessoas() {
         List<PessoaJuridica> pessoas = new ArrayList<>();
         Connection conn = null;
         PreparedStatement ps = null;
         ResultSet rs = null;

         try {
            conn = ConectorBD.getConnection();
            String sql = "SELECT p.idPessoa, p.nome, p.logradouro, p.cidade, p.estado, p.telefone, p.email,
pj.cnpj FROM Pessoa p INNER JOIN PessoaJuridica pj ON p.idPessoa = pj.Pessoa_idPessoa";
```

```
            ps = conn.prepareStatement(sql);
            rs = ps.executeQuery();

            while (rs.next()) {
                PessoaJuridica pessoa = new PessoaJuridica();
                pessoa.setId(rs.getInt("idPessoa"));
                pessoa.setNome(rs.getString("nome"));
                pessoa.setLogradouro(rs.getString("logradouro"));
                pessoa.setCidade(rs.getString("cidade"));
                pessoa.setEstado(rs.getString("estado"));
                pessoa.setTelefone(rs.getString("telefone"));
                pessoa.setEmail(rs.getString("email"));
                pessoa.setCnpj(rs.getString("cnpj"));

                pessoas.add(pessoa);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(rs);
            ConectorBD.close(ps);
            ConectorBD.close(conn);
        }

        return pessoas;
    }

    public void inserir (PessoaJuridica pessoa) {
        Connection conn = null;
        PreparedStatement ps = null;

        try {
            conn = ConectorBD.getConnection();
            conn.setAutoCommit(false);

            int pessoaId = SequenceManager.getNextValueA("Pessoa", "idPessoa");

            String sqlPessoa = "INSERT INTO Pessoa (idPessoa, nome, logradouro, cidade, estado, telefone,
email) VALUES (?, ?, ?, ?, ?, ?, ?)";
```

```
            ps = conn.prepareStatement(sqlPessoa, PreparedStatement.RETURN_GENERATED_KEYS);
            ps.setInt(1, pessoaId);
            ps.setString(2, pessoa.getNome());
            ps.setString(3, pessoa.getLogradouro());
            ps.setString(4, pessoa.getCidade());
            ps.setString(5, pessoa.getEstado());
            ps.setString(6, pessoa.getTelefone());
            ps.setString(7, pessoa.getEmail());
            ps.executeUpdate();

            int pessoaJuridicaId = SequenceManager.getNextValueA("PessoaJuridica", "idPessoaJuridica");

            String sqlPessoaJuridica = "INSERT INTO PessoaJuridica (idPessoaJuridica, Pessoa_idPessoa,
cnpj) VALUES (?, ?, ?)";
            ps = conn.prepareStatement(sqlPessoaJuridica);
            ps.setInt(1, pessoaJuridicaId);
            ps.setInt(2, pessoaId);
            ps.setString(3, pessoa.getCnpj());
            ps.executeUpdate();

            conn.commit();
        } catch (SQLException e) {
            if (conn != null) {
                try {
                    conn.rollback();
                } catch (SQLException ex) {
                    System.out.println("Erro ao tentar o rollback: ");
                    ex.printStackTrace();
                }
            }
            e.printStackTrace();
        } finally {
            ConectorBD.close(conn);
            ConectorBD.close(ps);
        }
    }

    public void alterar(PessoaJuridica pessoa) {
        Connection conn = null;
```

```
        PreparedStatement ps = null;

        try {
            int idPessoa = pessoa.getId();

            conn = ConectorBD.getConnection();
            String sqlPessoa = "UPDATE Pessoa SET nome=?, logradouro=?, cidade=?, estado=?, telefone=?, email=? WHERE idPessoa=?";
            ps = conn.prepareStatement(sqlPessoa);
            ps.setString(1, pessoa.getNome());
            ps.setString(2, pessoa.getLogradouro());
            ps.setString(3, pessoa.getCidade());
            ps.setString(4, pessoa.getEstado());
            ps.setString(5, pessoa.getTelefone());
            ps.setString(6, pessoa.getEmail());
            ps.setInt(7, idPessoa);
            ps.executeUpdate();

            String sqlPessoaJuridica = "UPDATE PessoaJuridica SET cnpj=? WHERE Pessoa_idPessoa=?";
            ps = conn.prepareStatement(sqlPessoaJuridica);
            ps.setString(1, pessoa.getCnpj());
            ps.setInt(2, idPessoa);
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(conn);
            ConectorBD.close(ps);
        }
    }

    public void excluir(int id) {
        Connection conn = null;
        PreparedStatement ps = null;

        try {
            conn = ConectorBD.getConnection();
            conn.setAutoCommit(false);
```

```
            String sqlPessoaJuridica = "DELETE FROM PessoaJuridica WHERE Pessoa_idPessoa = ?";
            ps = conn.prepareStatement(sqlPessoaJuridica);
            ps.setInt(1, id);
            ps.executeUpdate();

            String sqlPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";
            ps = conn.prepareStatement(sqlPessoa);
            ps.setInt(1, id);
            ps.executeUpdate();

            conn.commit();
        } catch (SQLException e) {
            if (conn != null) {
                try {
                    conn.rollback();
                } catch (SQLException ex) {
                    System.out.println("Erro ao tentar rollback: ");
                    ex.printStackTrace();
                }
            }
            e.printStackTrace();
        } finally {
            ConectorBD.close(conn);
            ConectorBD.close(ps);

        }
    }
}
```

CadastroBDTeste.java:

```
public class CadastroBDTeste {

    public static void main(String[] args) {
        PessoaFisica pessoaF = new PessoaFisica();
        pessoaF.setNome("Luffy");
        pessoaF.setLogradouro("Maca");
        pessoaF.setCidade("Lapa");
        pessoaF.setEstado("GO");
        pessoaF.setTelefone("1234-5678");
```

```
        pessoaF.setEmail("luffy@email.com");
        pessoaF.setCpf("12345678900");

        PessoaFisicaDAO pessoaFDao = new PessoaFisicaDAO();
        pessoaFDao.inserir(pessoaF);

        pessoaFDao.excluir(10);
        pessoaFDao.excluir(15);

        List<PessoaFisica> pessoasFisicas = pessoaFDao.getPessoas();
        System.out.println("Pessoas Fisicas: ");
        for (PessoaFisica pf : pessoasFisicas) {
            pf.exibir();
        }

        PessoaJuridica pessoaJ = new PessoaJuridica();
        pessoaJ.setNome("Baratie");
        pessoaJ.setLogradouro("East Blue");
        pessoaJ.setCidade("Rio");
        pessoaJ.setEstado("OP");
        pessoaJ.setTelefone("1111-1111");
        pessoaJ.setEmail("sanji@opmail.com");
        pessoaJ.setCnpj("1111111111/1111");

        PessoaJuridicaDAO pessoaJDao = new PessoaJuridicaDAO();
        pessoaJDao.inserir(pessoaJ);

        pessoaJDao.excluir(14);

        List<PessoaJuridica> pessoasJuridicas = pessoaJDao.getPessoas();
        System.out.println("Pessoas Juridicas: ");
        for (PessoaJuridica pj : pessoasJuridicas) {
            pj.exibir();
        }
    }
}
```

Resultado:

```
run:
Pessoas Fisicas:
--------------------
ID: 1
Nome: Rogerin
Logradouro: rua 4
Cidade: cidade dos aviao
Estado: SP
Telefone: (55)91111-1111
E-mail: rogerin@gmail.com
CPF: 111.111.111-00
--------------------
--------------------
ID: 3
Nome: Italo
Logradouro: Rua A
Cidade: Cidade A
Estado: GO
Telefone: 1111-1111
E-mail: italo@gmail.com
CPF: 111.111.111-00
--------------------
```

Conclusão:

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

Os componentes de middleware tem um grande papel, pois ele facilita a comunicação entre aplicativos java e o banco de dados.

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

O Statement é muito bom para consultas simples e estáticas, mas, por estar sujeita a injeção de sql e não é ideal para consultas parametrizadas.

Já o PreparedStatement oferece segurança contra injeção de sql e melhor desempenho em consultas parametrizadas.

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

O padrão DAO melhora a manutenibilidade do código, permitindo que alterações na estrutura do banco de dados sejam isoladas e refletidas apenas na implementação do DAO.

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

Utilizar herança perminte que consultas sejam feitas de forma eficiente, mas pode levar a complexidades de junção em consultas que envolvem todas as subclasses.

2º Procedimento | Alimentando a Base

CadastroRun.java:

```
import java.util.Scanner;
import java.util.List;
import java.util.InputMismatchException;

//Nome - Logradouro - Cidade - Estado - Telefone - Email - Cpf
//Nome - Logradouro - Cidade - Estado - Telefone - Email - Cnpj
public class CadastroRun {

   private static Scanner entrada = new Scanner(System.in);
   private static PessoaFisicaDAO pfDAO = new PessoaFisicaDAO();
   private static PessoaJuridicaDAO pjDAO = new PessoaJuridicaDAO();
   private static int acao;

   public static void main (String[] args) {
      do {
         try {
            System.out.println("Para continuar digite: ");
            System.out.println("============================");
            System.out.println("1 - Incluir");
            System.out.println("2 - Aterar");
            System.out.println("3 - Exibir");
            System.out.println("4 - Excluir");
            System.out.println("5 - Exibir Todos");
            System.out.println("0 - Finalizar");
            System.out.println("============================");

            acao = entrada.nextInt();
            entrada.nextLine();
```

```
switch (acao) {
   //incluir
   case 1 :{
      System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
      String tipoPessoa = entrada.next();
      entrada.nextLine();

      if (tipoPessoa.equalsIgnoreCase("f") || tipoPessoa.equalsIgnoreCase("F")) {
         String nome, logradouro, cidade, estado, telefone, email, cpf;
         PessoaFisica pf = new PessoaFisica();

         System.out.println("Insira os dados da Pessoa Fisica.");

         System.out.println("Nome: ");
         nome = entrada.nextLine();
         pf.setNome(nome);

         System.out.println("Logradouro: ");
         logradouro = entrada.nextLine();
         pf.setLogradouro(logradouro);

         System.out.println("Cidade: ");
         cidade = entrada.nextLine();
         pf.setCidade(cidade);

         System.out.println("Estado (max 2 caracteres): ");
         estado = entrada.nextLine();
         pf.setEstado(estado);

         System.out.println("Telefone: ");
         telefone = entrada.nextLine();
         pf.setTelefone(telefone);

         System.out.println("Email: ");
         email = entrada.nextLine();
         pf.setEmail(email);

         System.out.println("CPF: ");
         cpf = entrada.nextLine();
```

```
            pf.setCpf(cpf);

            pfDAO.inserir(pf);
            System.out.println("Pessoa inserida com sucesso!");
        } else if (tipoPessoa.equalsIgnoreCase("j") || tipoPessoa.equalsIgnoreCase("J")) {
            String nome, logradouro, cidade, estado, telefone, email, cnpj;
            PessoaJuridica pj = new PessoaJuridica();

            System.out.println("Insira os dados de Pessoa Juridica.");

            System.out.println("Nome: ");
            nome = entrada.nextLine();
            pj.setNome(nome);

            System.out.println("Logradouro: ");
            logradouro = entrada.nextLine();
            pj.setLogradouro(logradouro);

            System.out.println("Cidade: ");
            cidade = entrada.nextLine();
            pj.setCidade(cidade);

            System.out.println("Estado (max 2 caracteres): ");
            estado = entrada.nextLine();
            pj.setEstado(estado);

            System.out.println("Telefone: ");
            telefone = entrada.nextLine();
            pj.setTelefone(telefone);

            System.out.println("Email: ");
            email = entrada.nextLine();
            pj.setEmail(email);

            System.out.println("CNPJ: ");
            cnpj = entrada.nextLine();
            pj.setCnpj(cnpj);

            pjDAO.inserir(pj);
```

```
            System.out.println("Pessoa inseridica com sucesso!");
        } else {
            System.out.println("Valor inserido é diferente do esperado.");
        }
        break;
    }
    //alterar
    case 2 :{
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        String tipoPessoa = entrada.next();
        entrada.nextLine();

        if (tipoPessoa.equalsIgnoreCase("f") || tipoPessoa.equalsIgnoreCase("F")) {
            System.out.println("Insira a ID da Pessoa Fisica que deseja alterar.");
            int id = entrada.nextInt();
            entrada.nextLine();

            PessoaFisica pf = pfDAO.getPessoa(id);
            System.out.println("Esses são os dados atuais dessa pessoa.");
            pf.exibir();

            System.out.println("Insira os novos dados: ");

            String nome, logradouro, cidade, estado, telefone, email, cpf;

            System.out.println("Nome: ");
            nome = entrada.nextLine();
            pf.setNome(nome);

            System.out.println("Logradouro: ");
            logradouro = entrada.nextLine();
            pf.setLogradouro(logradouro);

            System.out.println("Cidade: ");
            cidade = entrada.nextLine();
            pf.setCidade(cidade);

            System.out.println("Estado (max 2 caracteres): ");
            estado = entrada.nextLine();
```

```
            pf.setEstado(estado);

            System.out.println("Telefone: ");
            telefone = entrada.nextLine();
            pf.setTelefone(telefone);

            System.out.println("Email: ");
            email = entrada.nextLine();
            pf.setEmail(email);

            System.out.println("CPF: ");
            cpf = entrada.nextLine();
            pf.setCpf(cpf);

            pfDAO.alterar(pf);
            System.out.println("Pessoa alterada com sucesso!");
        } else if (tipoPessoa.equalsIgnoreCase("j") || tipoPessoa.equalsIgnoreCase("J")) {
            System.out.println("Insira a IA da Pessoa Juridica que deseja alterar.");
            int id = entrada.nextInt();
            entrada.nextLine();

            PessoaJuridica pj = pjDAO.getPessoa(id);
            System.out.println("Esses são os dados atuais dessa pessoa");
            pj.exibir();

            System.out.println("Insira os novos dados.");

            String nome, logradouro, cidade, estado, telefone, email, cnpj;

            System.out.println("Nome: ");
            nome = entrada.nextLine();
            pj.setNome(nome);

            System.out.println("Logradouro: ");
            logradouro = entrada.nextLine();
            pj.setLogradouro(logradouro);

            System.out.println("Cidade: ");
            cidade = entrada.nextLine();
```

```
            pj.setCidade(cidade);

            System.out.println("Estado (max 2 caracteres): ");
            estado = entrada.nextLine();
            pj.setEstado(estado);

            System.out.println("Telefone: ");
            telefone = entrada.nextLine();
            pj.setTelefone(telefone);

            System.out.println("Email: ");
            email = entrada.nextLine();
            pj.setEmail(email);

            System.out.println("CNPJ: ");
            cnpj = entrada.nextLine();
            pj.setCnpj(cnpj);

            pjDAO.alterar(pj);
            System.out.println("Pessoa alterada com sucesso!");
        } else {
            System.out.println("Valor inserido diferente do solicitado.");
        }
        break;
    }
    //exibir
    case 3 :{
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        String tipoPessoa = entrada.next();
        entrada.nextLine();

        if (tipoPessoa.equalsIgnoreCase("f") || tipoPessoa.equalsIgnoreCase("F")) {
            System.out.println("Insira a ID da Pessoa Fisica.");
            int id = entrada.nextInt();
            entrada.nextLine();

            PessoaFisica pf = pfDAO.getPessoa(id);
            System.out.println("Buscando dados...");
```

```
            pf.exibir();
        } else if (tipoPessoa.equalsIgnoreCase("j") || tipoPessoa.equalsIgnoreCase("J")) {
            System.out.println("Insira a ID da Pessoa Juridica.");
            int id = entrada.nextInt();
            entrada.nextLine();

            PessoaJuridica pj = pjDAO.getPessoa(id);
            System.out.println("Buscando dados...");

            pj.exibir();
        } else {
            System.out.println("Valor inserido diferente do solicitado.");
        }
        break;
    }
    //excluir
    case 4 :{
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        String tipoPessoa = entrada.next();
        entrada.nextLine();

        if (tipoPessoa.equalsIgnoreCase("f") || tipoPessoa.equalsIgnoreCase("F")) {
            System.out.println("Insira a ID da Pessoa Fisica.");
            int id = entrada.nextInt();
            entrada.nextLine();

            PessoaFisica pf = pfDAO.getPessoa(id);
            System.out.println("Esses sao os dados da pessoa:");
            pf.exibir();

            System.out.println("Deseja excluir? S - sim | N - nao");
            String conf = entrada.next();
            entrada.nextLine();

            if (conf.equalsIgnoreCase("S") || conf.equalsIgnoreCase("s")) {
                pfDAO.excluir(id);
                System.out.println("Pessoa Excluida com sucesso!");
            } else if (conf.equalsIgnoreCase("N") || conf.equalsIgnoreCase("n")) {
                System.out.println("Pessoa intacta!");
```

```
            } else {
                System.out.println("Valor inserido diferente do solicitado.");
            }
        } else if (tipoPessoa.equalsIgnoreCase("j") || tipoPessoa.equalsIgnoreCase("J")) {
            System.out.println("Insira a ID da Pessoa Juridica");
            int id = entrada.nextInt();
            entrada.nextLine();

            PessoaJuridica pj = pjDAO.getPessoa(id);
            System.out.println("Esses sao os dados atuais da pessoa:");
            pj.exibir();

            System.out.println("Deseja excluir? S - sim | N - nao");
            String conf = entrada.next();
            entrada.nextLine();

            if (conf.equalsIgnoreCase("S") || conf.equalsIgnoreCase("s")) {
                pjDAO.excluir(id);
                System.out.println("Pessoa excluida com sucesso!");
            } else if (conf.equalsIgnoreCase("N") || conf.equalsIgnoreCase("n")) {
                System.out.println("Pessoa intacta!");
            } else {
                System.out.println("Valor inserido diferente do solicitado.");
            }
        } else {
            System.out.println("Valor inserido diferente do solicitado.");
        }
        break;
    }
    //exibir todos
    case 5 :{
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        String tipoPessoa = entrada.next();
        entrada.nextLine();

        if (tipoPessoa.equalsIgnoreCase("f") || tipoPessoa.equalsIgnoreCase("F")) {
            System.out.println("Buscando dados das Pessoas Fisicas...");

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

```
                    for (PessoaFisica pf : pessoas) {
                        pf.exibir();
                    }
                } else if (tipoPessoa.equalsIgnoreCase("j") || tipoPessoa.equalsIgnoreCase("J")) {
                    System.out.println("Buscando dados das Pessoas Juridicas...");

                    List<PessoaJuridica> pessoas = pjDAO.getPessoas();
                    for (PessoaJuridica pj : pessoas) {
                        pj.exibir();
                    }
                } else {
                    System.out.println("Valor inserido diferente do solicitado.");
                }
                break;
            }
            case 0 :{
                System.out.println("Programa encerrado.");
                break;
            }
            default :{
                System.out.println("Valor inserido diferente do solicitado.");
                break;
            }
        }

    } catch (InputMismatchException e) {
        System.out.println("Erro: Entrada nao permitida: " + e.getMessage());
        break;
    }
  } while (acao != 0);
 }
}
```

Conclusão:

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

A persistência em banco de dados oferece uma maior segurança, pois além de não criar um arquivo visível a todos não tem necessidade de lembrar o nome do banco dados para recuperar os dados.

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

O uso de operações lambda simplificou a impressão dos operadores, pois permite uma abordagem mais contida e legível para a iteração sobre coleções de objetos.

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

Porque o próprio método *main* já é *static* e é chamado diretamente pela JVM durente a inicialização do programa. Isso permite que o método main seja executado sem criar uma instância da classe que o contem.

Conclusão

Após a conclusão da missão prática percebi a importância de um banco de dados para armazenar as informações, pois dessa forma a organização e a segunrança fica bem melhor do que na persistência em arquivo. E apesar de todos os problemas que houve durante a codificação, não pude deixar de notar o quanto isso interessante.