RPG0016 - BackEnd sem banco não tem

Fábio Henrique Morales Prado | 202211280754

Sudoeste
Desenvolvimento Full Stack – 9001 – 2023.3
Github*: [fabiomprado/M3-Pratica3 (github.com)]*

Objetivo da Prática

- Implementar persistência com base no middleware JDBC.

- 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.

1º Procedimento | Mapeamento Objeto-Relacional e DAO

Arquivo: CadastroDB.java

```
package cadastrobd;



public class CadastroBD {
    public static void main(String[] args) {

    }
}
```

Arquivo: CadastroBDTeste.java

```
package cadastrobd;

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


public class CadastroBDTeste {

    public static void main(String[] args) {

        ConectorBD conectorBD = new ConectorBD();



        //a. Instanciar uma pessoa fisica e persistir no banco de
dados.
        System.out.println("\n");
        PessoaFisica pessoaFisica = new PessoaFisica();
        PessoaFisicaDAO pessoaFisicaDAO = new
PessoaFisicaDAO(conectorBD);
        pessoaFisica.setNome("TESTE");;
        pessoaFisica.setLogradouro("RUA DO TESTE");
        pessoaFisica.setCidade("TESTE DO SUL");
        pessoaFisica.setEstado("PA");
        pessoaFisica.setTelefone("33333");
        pessoaFisica.setEmail("TESTE@TESTE.COM");
        pessoaFisica.setCpf("3333333");
        int idRegistro = pessoaFisicaDAO.incluir(pessoaFisica);
        pessoaFisica.exibir();
        System.out.println("Criamos uma Pessoa Fisica com dados acima
```

```
e ID: " + idRegistro + "\n");

        //b. alterar os dados da pessoa fisica no banco.
        //pessoaFisicaDAO.getPessoa(idRegistro);
        pessoaFisica.setIdPessoa(idRegistro);
        pessoaFisica.setNome("TESTE1");
        pessoaFisica.setLogradouro("RUA DO TESTE1");
        pessoaFisica.setCidade("TESTE1 DO SUL");
        pessoaFisica.setEstado("PA");
        pessoaFisica.setTelefone("33333");
        pessoaFisica.setEmail("TESTE1@TESTE1.COM");
        pessoaFisica.setCpf("33333334");
        pessoaFisica.exibir();
        pessoaFisicaDAO.alterar(pessoaFisica);
        System.out.println("Observe que alteramos a Pessoa Fisica
criada anteriormente com ID: " + idRegistro + "\n");

        //c. Consultar todas as pessoas físicas do banco de dados e
listar no console.
        List<PessoaFisica> pessoasFisicas =
pessoaFisicaDAO.getPessoas();
        for (PessoaFisica pf : pessoasFisicas) {
            pf.exibir();
        }

        //d. Excluir a pessoa física criada anteriormente no banco.
        PessoaFisica pessoaExcluir =
pessoaFisicaDAO.getPessoa(idRegistro);
        pessoaFisicaDAO.excluir(pessoaExcluir);
        System.out.println("Acima temos todas as Pessoa Física, mas
iremos excluir a criada anteriormente com ID: " + idRegistro +"\n");
        pessoaFisicaDAO.getPessoas();
        List<PessoaFisica> pessoasFisicasNovas =
pessoaFisicaDAO.getPessoas();
        for (PessoaFisica pf : pessoasFisicasNovas) {
            pf.exibir();
        }
        System.out.println("Observe acima que agora não mais existe
Pessoa Física com o ID: " + idRegistro +"\n\n");

        //e. Instanciar uma pessoa fisica e persistir no banco de
dados.
        System.out.println("\n");
        PessoaJuridica pessoaJuridica = new PessoaJuridica();
        PessoaJuridicaDAO pessoaJuridicaDAO = new
PessoaJuridicaDAO(conectorBD);
        pessoaJuridica.setNome("TESTE");;
        pessoaJuridica.setLogradouro("RUA DO TESTE");
        pessoaJuridica.setCidade("TESTE DO SUL");
        pessoaJuridica.setEstado("PA");
        pessoaJuridica.setTelefone("33333");
        pessoaJuridica.setEmail("TESTE@TESTE.COM");
        pessoaJuridica.setCnpj("3333333");
        int idRegistroPJ = pessoaJuridicaDAO.incluir(pessoaJuridica);
        pessoaJuridica.exibir();
        System.out.println("Criamos uma Pessoa Juridica com dados
acima e ID: " + idRegistroPJ + "\n");

        //f. alterar os dados da pessoa fisica no banco.
        //pessoaJuridicaDAO.getPessoa(idRegistro);
```

```
        pessoaJuridica.setIdPessoa(idRegistroPJ);
        pessoaJuridica.setNome("TESTE1");
        pessoaJuridica.setLogradouro("RUA DO TESTE1");
        pessoaJuridica.setCidade("TESTE1 DO SUL");
        pessoaJuridica.setEstado("PA");
        pessoaJuridica.setTelefone("33333");
        pessoaJuridica.setEmail("TESTE1@TESTE1.COM");
        pessoaJuridica.setCnpj("33333334");
        pessoaJuridica.exibir();
        pessoaJuridicaDAO.alterar(pessoaJuridica);
        System.out.println("Observe que alteramos a Pessoa Juridica
criada anteriormente com ID: " + idRegistroPJ + "\n");

        //g. Consultar todas as pessoas físicas do banco de dados e
listar no console.
        List<PessoaJuridica> pessoasJuridicas =
pessoaJuridicaDAO.getPessoas();
        for (PessoaJuridica pj : pessoasJuridicas) {
            pj.exibir();
        }

        //h. Excluir a pessoa física criada anteriormente no banco.
        PessoaJuridica pessoaExcluirPJ =
pessoaJuridicaDAO.getPessoa(idRegistroPJ);
        pessoaJuridicaDAO.excluir(pessoaExcluirPJ);
        System.out.println("Acima temos todas as Pessoa Juridica, mas
iremos excluir a criada anteriormente com ID: " + idRegistroPJ +"\n");
        pessoaJuridicaDAO.getPessoas();
        List<PessoaJuridica> pessoasJuridicasNovas =
pessoaJuridicaDAO.getPessoas();
        for (PessoaJuridica pj : pessoasJuridicasNovas) {
            pj.exibir();
        }
        System.out.println("Observe acima que agora não mais existe
Pessoa Juridica com o ID: " + idRegistroPJ +"\n");

        System.out.println("programa encerrado!!");
    }
}
```

Arquivo: ConectorBD.java

```
package cadastro.model.util;

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

public class ConectorBD {
    private static final String URL =
"jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustS
erverCertificate=true;";
    private static final String USUARIO = "loja";
    private static final String SENHA = "loja";

    static{
```

```
        try{

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        }catch (ClassNotFoundException e){
            e.printStackTrace();
        }
    }

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USUARIO, SENHA);
    }

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

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

    public static void close(PreparedStatement statement, ResultSet
resultSet, Connection connection) {
        try {
            if (resultSet != null) {
                resultSet.close();
            }
            if (statement != null) {
                statement.close();
            }
            if (connection != null) {
                connection.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Arquivo: SequenceManager.java

```
package cadastro.model.util;

import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class SequenceManager {
    public static int getValue(String sequenceName) {
        int value = -1;
        try {
            String query = "SELECT NEXT VALUE FOR " + sequenceName;
            PreparedStatement preparedStatement =
ConectorBD.getPrepared(query);
            ResultSet resultSet = preparedStatement.executeQuery();
            if (resultSet.next()) {
                value = resultSet.getInt(1);
            }
```

```
            resultSet.close();
            preparedStatement.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return value;
    }
}
```

Arquivo: Pessoa.java

```
package cadastrobd.model;


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

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


    // Construtor completo
    public Pessoa(int idPessoa, String nome, String logradouro, String
cidade, String estado, String telefone, String email) {
        this.idPessoa = idPessoa;
        this.nome = nome;
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
        this.telefone = telefone;
        this.email = email;
    }

    // Métodos getters e setters para todos os campos
    public int getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(int idPessoa) {
        this.idPessoa = idPessoa;
    }

    public String getNome() {
        return nome;
    }

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

```
    public String getLogradouro() {
        return logradouro;
    }

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

    public String getCidade() {
        return cidade;
    }

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

    public String getEstado() {
        return estado;
    }

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

    public String getTelefone() {
        return telefone;
    }

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

    public String getEmail() {
        return email;
    }

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

    // Método para exibir os dados no console
    public void exibir() {
        System.out.println("Id: " + idPessoa);
        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("E-mail: " + email);
    }
}
```

Arquivo: PessoaFisica.java

```
package cadastrobd.model;


public class PessoaFisica extends Pessoa {
```

```
    private String cpf;

    // Construtor padrão
    public PessoaFisica() {

    }

    // Construtor completo
    public PessoaFisica(int idPessoa, String nome, String logradouro,
String cidade, String estado, String telefone, String email, String
cpf) {
        super(idPessoa, nome, logradouro, cidade, estado, telefone,
email); // Chama o construtor completo da classe Pessoa
        this.cpf = cpf;
    }

    // Métodos getters e setters para o campo cpf
    public String getCpf() {
        return cpf;
    }

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

    // Sobrescrita do método exibir para incluir o CPF
    @Override
    public void exibir() {
        super.exibir(); // Chama o método exibir da classe Pessoa
        System.out.println("CPF: " + cpf + "\n");
    }
}
```

Arquivo: PessoaFisicaDAO.java

```
package cadastrobd.model;


import cadastro.model.util.ConectorBD;
//import cadastro.model.util.SequenceManager;

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 PessoaFisicaDAO {
    private ConectorBD conectorBD;

    public PessoaFisicaDAO(ConectorBD conectorBD) {
        this.conectorBD = conectorBD;
    }

    public PessoaFisica getPessoa(int idPessoa) {
        Connection connection = null;
        PreparedStatement statement = null;
```

```
        ResultSet resultSet = null;
        try {
            connection = ConectorBD.getConnection();
            String sql = "SELECT p.nome, p.logradouro, p.cidade,
p.estado, p.telefone, p.email, pf.cpf " +
                    "FROM dbo.Pessoa p " +
                    "INNER JOIN dbo.PessoaFisica pf ON p.idPessoa =
pf.Pessoa_idPessoa " +
                    "WHERE p.idPessoa = ?";
            statement = connection.prepareStatement(sql);
            statement.setInt(1, idPessoa);
            resultSet = statement.executeQuery();
            if (resultSet.next()) {
                String nome = resultSet.getString("nome");
                String logradouro = resultSet.getString("logradouro");
                String cidade = resultSet.getString("cidade");
                String estado = resultSet.getString("estado");
                String telefone = resultSet.getString("telefone");
                String email = resultSet.getString("email");
                String cpf = resultSet.getString("cpf");
                return new PessoaFisica(idPessoa, nome, logradouro,
cidade, estado, telefone, email, cpf);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(statement, resultSet, connection);
        }
        return null;
    }

    public List<PessoaFisica> getPessoas() {
        List<PessoaFisica> pessoas = new ArrayList<>();
        Connection connection = null;
        PreparedStatement statement = null;
        ResultSet resultSet = null;
        try {
            connection = ConectorBD.getConnection();
            String sql = "SELECT p.idPessoa, p.nome, p.logradouro,
p.cidade, p.estado, p.telefone, p.email, pf.cpf FROM dbo.Pessoa p
INNER JOIN dbo.PessoaFisica pf ON p.idPessoa = pf.Pessoa_idPessoa";
            statement = connection.prepareStatement(sql);
            resultSet = statement.executeQuery();
            while (resultSet.next()) {
                int idf = resultSet.getInt("idPessoa");
                String nome = resultSet.getString("nome");
                String logradouro = resultSet.getString("logradouro");
                String cidade = resultSet.getString("cidade");
                String estado = resultSet.getString("estado");
                String telefone = resultSet.getString("telefone");
                String email = resultSet.getString("email");
                String cpf = resultSet.getString("cpf");
                PessoaFisica pessoa = new PessoaFisica(idf, nome,
logradouro, cidade, estado, telefone, email, cpf);
                pessoas.add(pessoa);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(statement, resultSet, connection);
```

```
        }
        return pessoas;
    }

    public int incluir(PessoaFisica pessoa) {
        Connection connection = null;
        PreparedStatement statement = null;
        ResultSet generatedKeys = null;
        int idd = 0;
        try {
            connection = ConectorBD.getConnection();
            String sqlPessoa = "INSERT INTO dbo.Pessoa (nome,
logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?,
?)";
            statement = connection.prepareStatement(sqlPessoa, new
String[]{"idPessoa"});
            //int id = SequenceManager.getValue("PessoaSeq");
            //statement.setInt(1, id);
            statement.setString(1, pessoa.getNome());
            statement.setString(2, pessoa.getLogradouro());
            statement.setString(3, pessoa.getCidade());
            statement.setString(4, pessoa.getEstado());
            statement.setString(5, pessoa.getTelefone());
            statement.setString(6, pessoa.getEmail());
            statement.executeUpdate();

            generatedKeys = statement.getGeneratedKeys();
            if (generatedKeys.next()) {
                idd = generatedKeys.getInt(1);
                String sqlPessoaFisica = "INSERT INTO dbo.PessoaFisica
(Pessoa_idPessoa, cpf) VALUES (?, ?)";
                statement =
connection.prepareStatement(sqlPessoaFisica);
                statement.setInt(1, idd);
                statement.setString(2, pessoa.getCpf());
                statement.executeUpdate();
            }

        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            if (generatedKeys != null) {
                try {
                    generatedKeys.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
            ConectorBD.close(statement, null, connection);
        }
        return idd;
    }


    public void alterar(PessoaFisica pessoa) {
        Connection connection = null;
        PreparedStatement statement = null;
        try {
            connection = ConectorBD.getConnection();
            String sqlPessoa = "UPDATE dbo.Pessoa SET nome = ?,
```

```
logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE
idPessoa = ?";
            statement = connection.prepareStatement(sqlPessoa);
            statement.setString(1, pessoa.getNome());
            statement.setString(2, pessoa.getLogradouro());
            statement.setString(3, pessoa.getCidade());
            statement.setString(4, pessoa.getEstado());
            statement.setString(5, pessoa.getTelefone());
            statement.setString(6, pessoa.getEmail());
            statement.setInt(7, pessoa.getIdPessoa());
            statement.executeUpdate();

            String sqlPessoaFisica = "UPDATE dbo.PessoaFisica SET cpf
= ? WHERE Pessoa_idPessoa = ?";
            statement = connection.prepareStatement(sqlPessoaFisica);
            statement.setString(1, pessoa.getCpf());
            statement.setInt(2, pessoa.getIdPessoa());
            statement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(statement, null, connection);
        }
    }

    public void excluir(PessoaFisica pessoa) {
        Connection connection = null;
        PreparedStatement statement = null;
        try {
            connection = ConectorBD.getConnection();
            String sqlPessoaFisica = "DELETE FROM dbo.PessoaFisica
WHERE Pessoa_idPessoa = ?";
            statement = connection.prepareStatement(sqlPessoaFisica);
            statement.setInt(1, pessoa.getIdPessoa());
            statement.executeUpdate();

            String sqlPessoa = "DELETE FROM dbo.Pessoa WHERE idPessoa
= ?";
            statement = connection.prepareStatement(sqlPessoa);
            statement.setInt(1, pessoa.getIdPessoa());
            statement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(statement, null, connection);
        }
    }
}
```

Arquivo: PessoaJuridica.java

```
package cadastrobd.model;

public class PessoaJuridica extends Pessoa {
    private String cnpj;

    // Construtor padrão
    public PessoaJuridica() {
```

```
    }

    // Construtor completo
    public PessoaJuridica(int idPessoa, String nome, String
logradouro, String cidade, String estado, String telefone, String
email, String cnpj) {
        super(idPessoa, nome, logradouro, cidade, estado, telefone,
email); // Chama o construtor completo da classe Pessoa
        this.cnpj = cnpj;
    }

    // Métodos getters e setters para o campo cnpj
    public String getCnpj() {
        return cnpj;
    }

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

    // Sobrescrita do método exibir para incluir o CNPJ
    @Override
    public void exibir() {
        super.exibir(); // Chama o método exibir da classe Pessoa
        System.out.println("CNPJ: " + cnpj+ "\n");
    }
}
```

Arquivo: PessoaJuridicaDAO.java

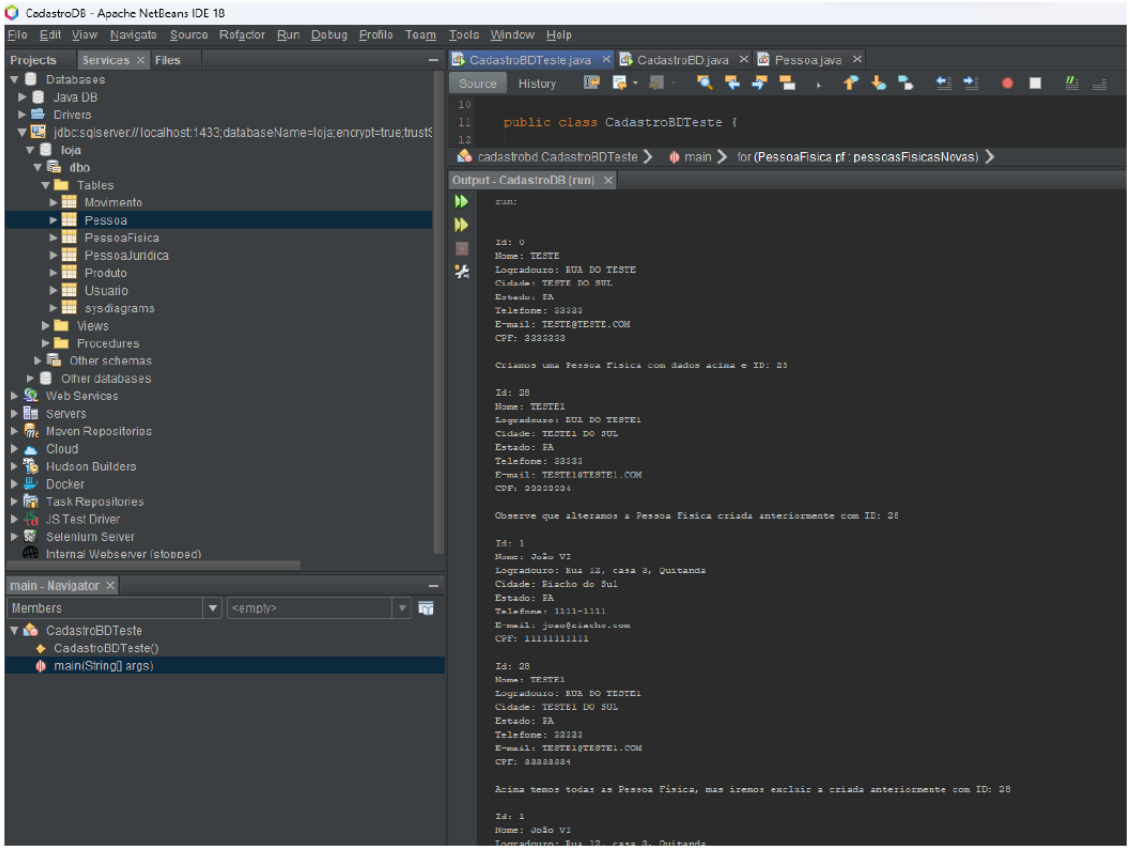
```
package cadastrobd.model;


import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;

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 {
    private ConectorBD conectorBD;

    public PessoaJuridicaDAO(ConectorBD conectorBD) {
        this.conectorBD = conectorBD;
    }

    public PessoaJuridica getPessoa(int idPessoa) {
        Connection connection = null;
        PreparedStatement statement = null;
        ResultSet resultSet = null;
        try {
            connection = ConectorBD.getConnection();
            String sql = "SELECT p.nome, p.logradouro, p.cidade,
```

```
p.estado, p.telefone, p.email, pj.cnpj " +
                    "FROM dbo.Pessoa p " +
                    "INNER JOIN dbo.PessoaJuridica pj ON p.idPessoa =
pj.Pessoa idPessoa " +
                    "WHERE p.idPessoa = ?";
            statement = connection.prepareStatement(sql);
            statement.setInt(1, idPessoa);
            resultSet = statement.executeQuery();
            if (resultSet.next()) {
                String nome = resultSet.getString("nome");
                String logradouro = resultSet.getString("logradouro");
                String cidade = resultSet.getString("cidade");
                String estado = resultSet.getString("estado");
                String telefone = resultSet.getString("telefone");
                String email = resultSet.getString("email");
                String cnpj = resultSet.getString("cnpj");
                return new PessoaJuridica(idPessoa, nome, logradouro,
cidade, estado, telefone, email, cnpj);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(statement, resultSet, connection);
        }
        return null;
    }

    public List<PessoaJuridica> getPessoas() {
        List<PessoaJuridica> pessoas = new ArrayList<>();
        Connection connection = null;
        PreparedStatement statement = null;
        ResultSet resultSet = null;
        try {
            connection = ConectorBD.getConnection();
            String sql = "SELECT p.idPessoa, p.nome, p.logradouro,
p.cidade, p.estado, p.telefone, p.email, pj.cnpj FROM dbo.Pessoa p
INNER JOIN dbo.PessoaJuridica pj ON p.idPessoa = pj.Pessoa idPessoa";
            statement = connection.prepareStatement(sql);
            resultSet = statement.executeQuery();
            while (resultSet.next()) {
                int idj = resultSet.getInt("idPessoa");
                String nome = resultSet.getString("nome");
                String logradouro = resultSet.getString("logradouro");
                String cidade = resultSet.getString("cidade");
                String estado = resultSet.getString("estado");
                String telefone = resultSet.getString("telefone");
                String email = resultSet.getString("email");
                String cnpj = resultSet.getString("cnpj");
                PessoaJuridica pessoa = new PessoaJuridica(idj, nome,
logradouro, cidade, estado, telefone, email, cnpj);
                pessoas.add(pessoa);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(statement, resultSet, connection);
        }
        return pessoas;
    }
```

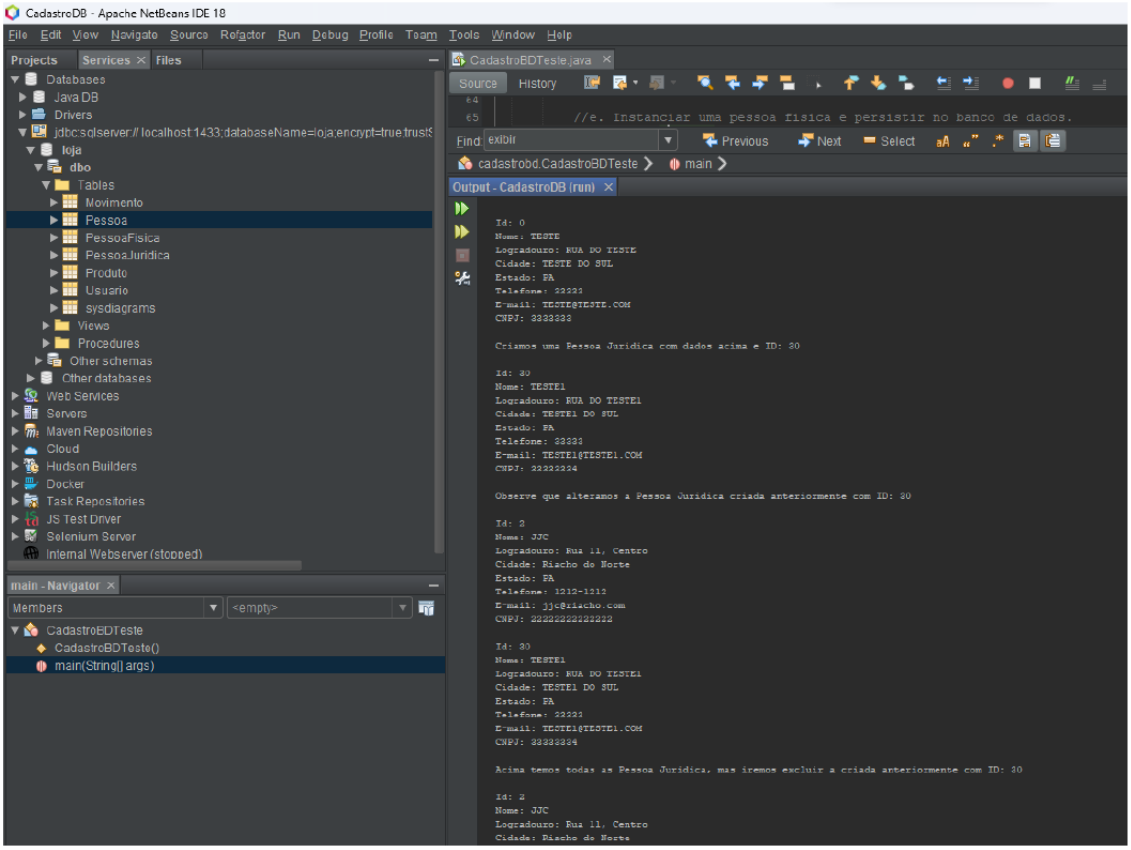
```
    public int incluir(PessoaJuridica pessoa) {
        Connection connection = null;
        PreparedStatement statement = null;
        ResultSet generatedKeys = null;
        int idd = 0;
        try {
            connection = ConectorBD.getConnection();
            String sqlPessoa = "INSERT INTO dbo.Pessoa (nome,
logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?,
?)";
            statement = connection.prepareStatement(sqlPessoa, new
String[] {"idPessoa"});
            //int id = SequenceManager.getValue("PessoaSeq");
            //statement.setInt(1, id);
            statement.setString(1, pessoa.getNome());
            statement.setString(2, pessoa.getLogradouro());
            statement.setString(3, pessoa.getCidade());
            statement.setString(4, pessoa.getEstado());
            statement.setString(5, pessoa.getTelefone());
            statement.setString(6, pessoa.getEmail());
            statement.executeUpdate();

            generatedKeys = statement.getGeneratedKeys();
            if (generatedKeys.next()){
                idd = generatedKeys.getInt(1);
                String sqlPessoaFisica = "INSERT INTO
dbo.PessoaJuridica(Pessoa idPessoa, cnpj) VALUES (?, ?)";
                statement =
connection.prepareStatement(sqlPessoaFisica);
                statement.setInt(1, idd);
                statement.setString(2, pessoa.getCnpj());
                statement.executeUpdate();
            }

        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            if (generatedKeys != null) {
                try {
                    generatedKeys.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
            ConectorBD.close(statement, null, connection);
        }
        return idd;
    }

    public void alterar(PessoaJuridica pessoa) {
        Connection connection = null;
        PreparedStatement statement = null;
        try {
            connection = ConectorBD.getConnection();
            String sqlPessoa = "UPDATE dbo.Pessoa SET nome = ?,
logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE
idPessoa = ?";
            statement = connection.prepareStatement(sqlPessoa);
            statement.setString(1, pessoa.getNome());
            statement.setString(2, pessoa.getLogradouro());
```

```
            statement.setString(3, pessoa.getCidade());
            statement.setString(4, pessoa.getEstado());
            statement.setString(5, pessoa.getTelefone());
            statement.setString(6, pessoa.getEmail());
            statement.setInt(7, pessoa.getIdPessoa());
            statement.executeUpdate();

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

    public void excluir(PessoaJuridica pessoa) {
        Connection connection = null;
        PreparedStatement statement = null;
        try {
            connection = ConectorBD.getConnection();
            String sqlPessoaJuridica = "DELETE FROM dbo.PessoaJuridica
WHERE Pessoa_idPessoa = ?";
            statement =
connection.prepareStatement(sqlPessoaJuridica);
            statement.setInt(1, pessoa.getIdPessoa());
            statement.executeUpdate();

            String sqlPessoa = "DELETE FROM dbo.Pessoa WHERE idPessoa
= ?";
            statement = connection.prepareStatement(sqlPessoa);
            statement.setInt(1, pessoa.getIdPessoa());
            statement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(statement, null, connection);
        }
    }
}
```

Testes Pessoa Física (item A à D)



Testes Pessoa Juridica (item E à H)



- ANÁLISE E CONCLUSÃO:

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

Componentes de middleware, como o JDBC, são cruciais para conectar aplicativos a bancos de dados, fornecendo abstração, gerenciamento de conexões, desempenho e segurança. Eles facilitam a integração de dados, promovem a portabilidade e seguem padrões, contribuindo para aplicativos robustos e eficazes.

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

PreparedStatement permite consultas parametrizadas, enquanto Statement incorpora diretamente valores, tornando-o vulnerável e menos eficiente. PreparedStatement é preferível ao Statement em JDBC devido à segurança contra injeção SQL, melhor desempenho, legibilidade e manutenção de código.

- 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, promovendo a reutilização de código, facilitando a troca de fontes de dados, melhorando a testabilidade, padronizando o código e fornecendo documentação clara. Isso simplifica a manutenção e a evolução do software.

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

Em um modelo estritamente relacional, a herança pode ser refletida criando-se tabelas separadas para cada classe concreta na hierarquia de herança, com atributos específicos e herdados.

2º Procedimento | Alimentando a Base

Arquivo: CadastroBD.java

```java
package cadastrobd;


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

public class CadastroBD {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        ConectorBD conectorBD = new ConectorBD();

        PessoaFisicaDAO pessoaFisicaDAO = new
PessoaFisicaDAO(conectorBD);
        PessoaJuridicaDAO pessoaJuridicaDAO = new
PessoaJuridicaDAO(conectorBD);

        int opcao;
        do {
            System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-");
            System.out.println("Selecione uma opção:");
            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("-=-=-=-=-=-=-=-=-=-=-=-=-");
            opcao = scanner.nextInt();
            scanner.nextLine(); // Limpar o buffer

            switch (opcao) {
                case 1:
                    System.out.println("Escolha o tipo (F - Física, J
- Jurídica):");
                    String tipo = scanner.nextLine().toUpperCase();

                    if (tipo.equalsIgnoreCase("F")) {
                        PessoaFisica pessoaFisica = new
PessoaFisica();
                        System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-
");
                        System.out.print("Nome: ");
                        pessoaFisica.setNome(scanner.nextLine());
```

```
                        System.out.print("Logradouro: ");
pessoaFisica.setLogradouro(scanner.nextLine());

                        System.out.print("Cidade: ");
                        pessoaFisica.setCidade(scanner.nextLine());

                        System.out.print("Estado: ");
                        pessoaFisica.setEstado(scanner.nextLine());

                        System.out.print("Telefone: ");
                        pessoaFisica.setTelefone(scanner.nextLine());

                        System.out.print("E-mail: ");
                        pessoaFisica.setEmail(scanner.nextLine());

                        System.out.print("CPF: ");
                        pessoaFisica.setCpf(scanner.nextLine());
                        System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-
");

                        pessoaFisicaDAO.incluir(pessoaFisica);
                        System.out.println("Pessoa Física incluída com
sucesso!");

                    } else if (tipo.equalsIgnoreCase("J")) {
                        PessoaJuridica pessoaJuridica = new
PessoaJuridica();
                        System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-
");
                        System.out.print("Nome: ");
                        pessoaJuridica.setNome(scanner.nextLine());

                        System.out.print("Logradouro: ");
pessoaJuridica.setLogradouro(scanner.nextLine());

                        System.out.print("Cidade: ");
                        pessoaJuridica.setCidade(scanner.nextLine());

                        System.out.print("Estado: ");
                        pessoaJuridica.setEstado(scanner.nextLine());

                        System.out.print("Telefone: ");
pessoaJuridica.setTelefone(scanner.nextLine());

                        System.out.print("E-mail: ");
                        pessoaJuridica.setEmail(scanner.nextLine());

                        System.out.print("CNPJ: ");
                        pessoaJuridica.setCnpj(scanner.nextLine());
                        System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-
");

                        pessoaJuridicaDAO.incluir(pessoaJuridica);
                        System.out.println("Pessoa Jurídica incluída
com sucesso!");

                    } else {
```

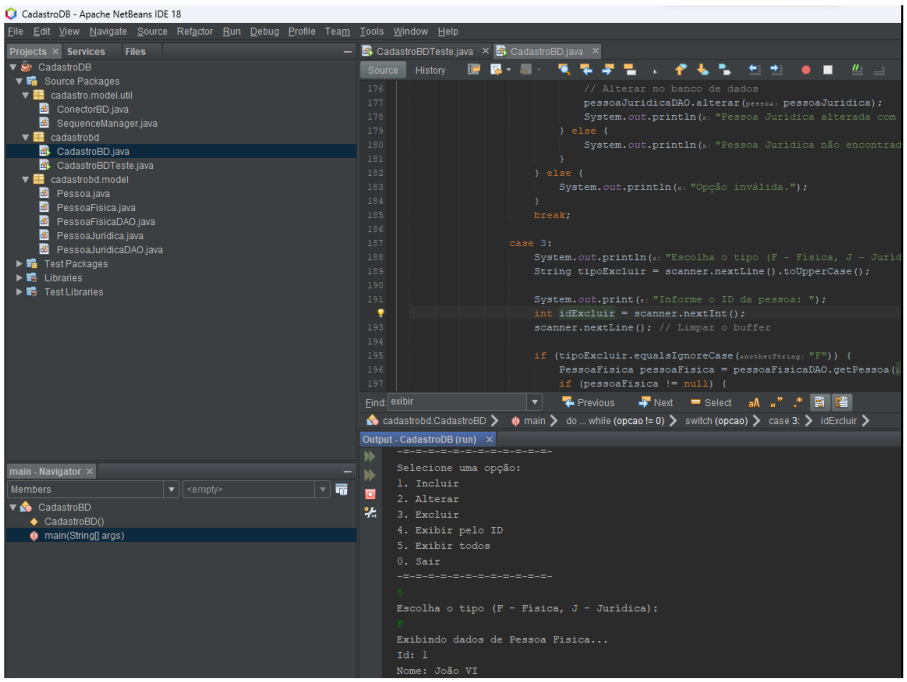
```
                        System.out.println("Opção inválida.");
                    }
                    break;

                case 2:
                    System.out.println("Escolha o tipo (F - Física, J
- Jurídica):");
                    String tipoAlterar =
scanner.nextLine().toUpperCase();

                    System.out.print("Informe o ID da pessoa: ");
                    int idAlterar = scanner.nextInt();
                    scanner.nextLine(); // Limpar o buffer

                    if (tipoAlterar.equalsIgnoreCase("F")) {
                        PessoaFisica pessoaFisica =
pessoaFisicaDAO.getPessoa(idAlterar);
                        if (pessoaFisica != null) {
                            // Exibir dados atuais da pessoa
                            pessoaFisica.exibir();

                            // Receber novos dados
                            System.out.println("-=-=-=-=-=-=-=-=-=-=-
=-=-");
                            System.out.print("Novo Nome: ");
                            pessoaFisica.setNome(scanner.nextLine());

                            System.out.print("Novo Logradouro: ");

pessoaFisica.setLogradouro(scanner.nextLine());

                            System.out.print("Nova Cidade: ");

pessoaFisica.setCidade(scanner.nextLine());

                            System.out.print("Novo Estado: ");

pessoaFisica.setEstado(scanner.nextLine());

                            System.out.print("Novo Telefone: ");

pessoaFisica.setTelefone(scanner.nextLine());

                            System.out.print("Novo E-mail: ");
                            pessoaFisica.setEmail(scanner.nextLine());

                            System.out.print("Novo CPF: ");
                            pessoaFisica.setCpf(scanner.nextLine());
                            System.out.println("-=-=-=-=-=-=-=-=-=-=-
=-=-");

                            // Alterar no banco de dados
                            pessoaFisicaDAO.alterar(pessoaFisica);
                            System.out.println("Pessoa Física alterada
com sucesso!");

                        } else {
                            System.out.println("Pessoa Física não
encontrada.");
                        }
```

```
                    } else if (tipoAlterar.equalsIgnoreCase("J")) {
                        PessoaJuridica pessoaJuridica =
pessoaJuridicaDAO.getPessoa(idAlterar);
                        if (pessoaJuridica != null) {
                            // Exibir dados atuais da pessoa jurídica
                            pessoaJuridica.exibir();

                            // Receber novos dados
                            System.out.println("-=-=-=-=-=-=-=-=-=-=-
=-=-");
                            System.out.print("Novo Nome: ");

pessoaJuridica.setNome(scanner.nextLine());

                            System.out.print("Novo Logradouro: ");

pessoaJuridica.setLogradouro(scanner.nextLine());

                            System.out.print("Nova Cidade: ");

pessoaJuridica.setCidade(scanner.nextLine());

                            System.out.print("Novo Estado: ");

pessoaJuridica.setEstado(scanner.nextLine());

                            System.out.print("Novo Telefone: ");

pessoaJuridica.setTelefone(scanner.nextLine());

                            System.out.print("Novo E-mail: ");

pessoaJuridica.setEmail(scanner.nextLine());

                            System.out.print("Novo "
                                    + "CNPJ: ");

pessoaJuridica.setCnpj(scanner.nextLine());
                            System.out.println("-=-=-=-=-=-=-=-=-=-=-
=-=-");

                            // Alterar no banco de dados
                            pessoaJuridicaDAO.alterar(pessoaJuridica);
                            System.out.println("Pessoa Jurídica
alterada com sucesso!");
                        } else {
                            System.out.println("Pessoa Jurídica não
encontrada.");
                        }
                    } else {
                        System.out.println("Opção inválida.");
                    }
                    break;

                case 3:
                    System.out.println("Escolha o tipo (F - Física, J
- Jurídica):");
                    String tipoExcluir =
scanner.nextLine().toUpperCase();
```

```
                    System.out.print("Informe o ID da pessoa: ");
                    int idExcluir = scanner.nextInt();
                    scanner.nextLine(); // Limpar o buffer

                    if (tipoExcluir.equalsIgnoreCase("F")) {
                        PessoaFisica pessoaFisica =
pessoaFisicaDAO.getPessoa(idExcluir);
                        if (pessoaFisica != null) {
                            pessoaFisicaDAO.excluir(pessoaFisica);
                            System.out.println("Pessoa Física excluída
com sucesso!");
                        } else {
                            System.out.println("Pessoa Física não
encontrada.");
                        }
                    } else if (tipoExcluir.equalsIgnoreCase("J")) {
                        PessoaJuridica pessoaJuridica =
pessoaJuridicaDAO.getPessoa(idExcluir);
                        if (pessoaJuridica != null) {
                            pessoaJuridicaDAO.excluir(pessoaJuridica);
                            System.out.println("Pessoa Jurídica
excluída com sucesso!");
                        } else {
                            System.out.println("Pessoa Jurídica não
encontrada.");
                        }
                    } else {
                        System.out.println("Opção inválida.");
                    }
                    break;

                case 4:
                    System.out.println("Escolha o tipo (F - Física, J
- Jurídica):");
                    String tipoExibirId =
scanner.nextLine().toUpperCase();

                    System.out.print("Informe o ID da pessoa: ");
                    int idExibirId = scanner.nextInt();
                    scanner.nextLine(); // Limpar o buffer

                    if (tipoExibirId.equalsIgnoreCase("F")) {
                        PessoaFisica pessoaFisica =
pessoaFisicaDAO.getPessoa(idExibirId);
                        if (pessoaFisica != null) {
                            pessoaFisica.exibir();
                        } else {
                            System.out.println("Pessoa Física não
encontrada.");
                        }
                    } else if (tipoExibirId.equalsIgnoreCase("J")) {
                        PessoaJuridica pessoaJuridica =
pessoaJuridicaDAO.getPessoa(idExibirId);
                        if (pessoaJuridica != null) {
                            pessoaJuridica.exibir();
                        } else {
                            System.out.println("Pessoa Jurídica não
encontrada.");
                        }
                    } else {
```

```
                        System.out.println("Opção inválida.");
                    }
                    break;

                case 5:
                    System.out.println("Escolha o tipo (F - Física, J
- Jurídica):");
                    String tipoExibirTodos =
scanner.nextLine().toUpperCase();

                    if (tipoExibirTodos.equalsIgnoreCase("F")) {
                        System.out.println("Exibindo dados de Pessoa
Fisica...");
                        List<PessoaFisica> pessoasFisicas =
pessoaFisicaDAO.getPessoas();
                        for (PessoaFisica pf : pessoasFisicas) {
                            pf.exibir();
                        }
                    } else if (tipoExibirTodos.equalsIgnoreCase("J"))
{
                        System.out.println("Exibindo dados de Pessoa
Juridica...");
                        List<PessoaJuridica> pessoasJuridicas =
pessoaJuridicaDAO.getPessoas();
                        for (PessoaJuridica pj : pessoasJuridicas) {
                            pj.exibir();
                        }
                    } else {
                        System.out.println("Opção inválida.");
                    }
                    break;

                case 0:
                    System.out.println("Encerrando o programa.");
                    break;

                default:
                    System.out.println("Opção inválida.");
                    break;
            }

        } while (opcao != 0);

        scanner.close();
    }
}
```

RESULTADOS DA EXECUÇÃO:



Conclusão

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

A persistência em arquivo envolve armazenar dados em arquivos no sistema de arquivos, exigindo acesso direto e controle manual. Em contrapartida, a persistência em banco de dados usa um Sistema de Gerenciamento de Banco de Dados (SGBD) para armazenar dados em estruturas tabulares, fornecendo acesso simplificado, segurança avançada, suporte a transações e escalabilidade, sendo mais adequada para aplicações complexas e de grande escala.

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

O uso de operadores lambda no Java simplificou a impressão de valores em entidades, eliminando a necessidade de código extenso. Isso torna o código mais conciso e legível, melhorando a expressividade e a eficiência em operações em coleções de dados.

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

Métodos acionados diretamente pelo método main em Java precisam ser marcados como static porque o main é um método estático e não possui acesso a instâncias de objetos. A

marcação como static permite que esses métodos sejam chamados independentemente de objetos e simplifica a chamada direta a partir do main.

- Concluímos que:

Componentes de middleware, como o JDBC, são fundamentais para conectar aplicativos a bancos de dados, facilitando a integração e o gerenciamento de conexões e o padrão DAO melhora a manutenibilidade ao separar a lógica de acesso a dados. A persistência de dados pode ocorrer tanto em arquivos como em bancos de dados, com diferenças marcantes em termos de estrutura e complexidade. Resumindo, escolher a abordagem certa de persistência, aproveitar os recursos do Java e seguir boas práticas de design contribuem para sistemas mais eficientes e de fácil manutenção.