



# Estácio

## Missão Prática Nível 3 – Mundo 3

Fernanda Canto P. da Costa - 202208379788

Campus de Ipanema

BackEnd sem banco não tem – 22.3 – 3º semestre

Github: <https://github.com/nandacpc/Missao-Nv3-M3>

### Objetivos 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.

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

Arquivo Pessoa.java:

```
package cadastrobd.model;

/**
 *
 * @author Nanda
 */
public class Pessoa {
    private int idPessoa;
    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.idPessoa = id;
        this.nome = nome;
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
        this.telefone = telefone;
        this.email = email;
    }
    public int getId() {
        return idPessoa;
    }
    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 setId(int id) {
        this.idPessoa = 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("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("Email: " + email);
    }
}

```

### Arquivo PessoaFisica.java:

```

package cadastrbd.model;

/**
 *
 * @author Nanda
 */

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;
    }
    public void setCpf(String cpf) {
        this.cpf = cpf;
    }
    public String getCpf() {
        return cpf;
    }
    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CPF: " + cpf);
    }
}

```

### Arquivo PessoaJuridica.java:

```
package cadastrobd.model;

/**
 *
 * @author Nanda
 */

public class PessoaJuridica extends Pessoa {
    private String cnpj;

    public PessoaJuridica() {
        super();
    }

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

    public String getCnpj() {
        return cnpj;
    }

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

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

### Arquivo ConectorBD.java:

```
package cadastro.model.util;

/**
 *
 * @author Nanda
 */

import java.sql.Statement;
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:62208;databaseName=loja;encrypt=true;trustSer
verCertificate=true;";
    private static final String USUARIO = "loja";
    private static final String SENHA = "loja";

    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(PreparedStatement prepared) {
        try {
            return prepared.executeQuery();
        } catch (SQLException e) {
            System.out.println("Erro ao preparar: " + e.getMessage());
            return null;
        }
    }
    public static void close(Statement stmt) {
        try {
            if (stmt != null && !stmt.isClosed()) {
                stmt.close();
            }
        } catch (SQLException e) {
        }
    }
    public static void close(ResultSet rs) {
        try {
            if (rs != null && !rs.isClosed()) {
                rs.close();
            }
        } catch (SQLException e) {
        }
    }
    public static void close(Connection conn) {
        try {
            if (conn != null && !conn.isClosed()) {
                conn.close();
            }
        } catch (SQLException e) {
        }
    }
}

```

## Arquivo SequenceManager.java:

```
package cadastro.model.util;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
/**
 *
 * @author Nanda
 */

public class SequenceManager {
    private static final String SELECT_SEQUENCE_SQL = "SELECT
sequence_value FROM sequences WHERE sequence_name = ?";
    private static final String UPDATE_SEQUENCE_SQL = "UPDATE sequences
SET sequence_value = ? WHERE sequence_name = ?";

    public int getValue(String sequenceName) {
        Connection connection = null;
        PreparedStatement selectStatement = null;
        PreparedStatement updateStatement = null;
        ResultSet resultSet = null;

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

            selectStatement =
connection.prepareStatement(SELECT_SEQUENCE_SQL);
            selectStatement.setString(1, sequenceName);
            resultSet = selectStatement.executeQuery();

            int nextValue = 1;

            if (resultSet.next()) {
                nextValue = resultSet.getInt("sequence_value") + 1;

                updateStatement =
connection.prepareStatement(UPDATE_SEQUENCE_SQL);
                updateStatement.setInt(1, nextValue);
                updateStatement.setString(2, sequenceName);
                updateStatement.executeUpdate();
            }

            connection.commit();
        }
    }
}
```

```

        return nextValue;
    } catch (SQLException e) {
        if (connection != null) {
            try {
                connection.rollback();
            } catch (SQLException rollbackException) {
            }
        }
        return 0;
    } finally {
        ConectorBD.close(resultSet);
        ConectorBD.close(selectStatement);
        ConectorBD.close(updateStatement);
        ConectorBD.close(connection);
    }
}
}
}

```

#### 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;

/**
 *
 * @author Nanda
 */

public class PessoaFisicaDAO {
    private ConectorBD conector;
    private SequenceManager sequenceManager;

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

    public PessoaFisica getPessoa(int idPessoa) {
        Connection connection = null;
    }
}

```

```

        PreparedStatement preparedStatement = null;
        ResultSet resultSet = null;

        try {
            connection = conector.getConnection();
            String sql = "SELECT * FROM pessoa_fisica WHERE idPessoa =
?";

            preparedStatement = connection.prepareStatement(sql);
            preparedStatement.setInt(1, idPessoa);
            resultSet = preparedStatement.executeQuery();

            if (resultSet.next()) {
                return buildPessoaFisicaFromResultSet(resultSet);
            }
        } catch (SQLException e) {
        } finally {
            conector.close(resultSet);
            conector.close(preparedStatement);
            conector.close(connection);
        }

        return null;
    }

    public List<PessoaFisica> getPessoas() {
        Connection connection = null;
        PreparedStatement preparedStatement = null;
        ResultSet resultSet = null;
        List<PessoaFisica> pessoasFisicas = new ArrayList<>();

        try {
            connection = conector.getConnection();
            String sql = "SELECT * FROM pessoa_fisica";
            preparedStatement = connection.prepareStatement(sql);
            resultSet = preparedStatement.executeQuery();

            while (resultSet.next()) {
                pessoasFisicas.add(buildPessoaFisicaFromResultSet(resultSet));
            }
        } catch (SQLException e) {
        } finally {
            conector.close(resultSet);
            conector.close(preparedStatement);
            conector.close(connection);
        }

        return pessoasFisicas;
    }
}

```



```

public void incluir(PessoaFisica pessoaFisica) {
    Connection connection = null;
    PreparedStatement preparedStatement = null;

    try {
        connection = conector.getConnection();
        connection.setAutoCommit(false);

        int idPessoa = sequenceManager.getValue("pessoa_fisica_seq");

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

        preparedStatement = connection.prepareStatement(sqlPessoa);
        preparedStatement.setInt(1, idPessoa);
        preparedStatement.setString(2, pessoaFisica.getNome());
        preparedStatement.setString(3, pessoaFisica.getLogradouro());
        preparedStatement.setString(4, pessoaFisica.getCidade());
        preparedStatement.setString(5, pessoaFisica.getEstado());
        preparedStatement.setString(6, pessoaFisica.getTelefone());
        preparedStatement.setString(7, pessoaFisica.getEmail());
        preparedStatement.executeUpdate();

        String sqlPessoaFisica = "INSERT INTO pessoa_fisica
(idPessoa, cpf) VALUES (?, ?)";
        preparedStatement =
connection.prepareStatement(sqlPessoaFisica);
        preparedStatement.setInt(1, idPessoa);
        preparedStatement.setString(2, pessoaFisica.getCpf());
        preparedStatement.executeUpdate();

        connection.commit();
    } catch (SQLException e) {
        if (connection != null) {
            try {
                connection.rollback();
            } catch (SQLException rollbackException) {
            }
        }
    } finally {
        conector.close(preparedStatement);
        conector.close(connection);
    }
}

public void alterar(PessoaFisica pessoaFisica) {
    Connection connection = null;
    PreparedStatement preparedStatement = null;

```

```

        try {
            connection = conector.getConnection();
            connection.setAutoCommit(false);

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

            String sqlPessoaFisica = "UPDATE pessoa_fisica SET cpf = ?
WHERE idPessoa = ?";
            preparedStatement =
connection.prepareStatement(sqlPessoaFisica);
            preparedStatement.setString(1, pessoaFisica.getCpf());
            preparedStatement.setInt(2, pessoaFisica.getId());
            preparedStatement.executeUpdate();

            connection.commit();
        } catch (SQLException e) {
            if (connection != null) {
                try {
                    connection.rollback();
                } catch (SQLException rollbackException) {
                }
            }
        } finally {
            conector.close(preparedStatement);
            conector.close(connection);
        }
    }

    public void excluir(int idPessoa) {
        Connection connection = null;
        PreparedStatement preparedStatement = null;

        try {
            connection = conector.getConnection();
            connection.setAutoCommit(false);

            String sqlPessoaFisica = "DELETE FROM pessoa_fisica WHERE
idPessoa = ?";

```

```

        preparedStatement =
connection.prepareStatement(sqlPessoaFisica);
        preparedStatement.setInt(1, idPessoa);
        preparedStatement.executeUpdate();

        String sqlPessoa = "DELETE FROM pessoa WHERE idPessoa = ?";
        preparedStatement = connection.prepareStatement(sqlPessoa);
        preparedStatement.setInt(1, idPessoa);
        preparedStatement.executeUpdate();

        connection.commit();
    } catch (SQLException e) {
        if (connection != null) {
            try {
                connection.rollback();
            } catch (SQLException rollbackException) {
            }
        }
    } finally {
        conector.close(preparedStatement);
        conector.close(connection);
    }
}

private PessoaFisica buildPessoaFisicaFromResultSet(ResultSet
resultSet) throws SQLException {
    int idPessoa = 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 pessoaFisica = new PessoaFisica(idPessoa, nome,
logradouro, cidade, estado, telefone, email, cpf);
    return pessoaFisica;
}
}

```

## Arquivo PessoaJuridicaDAO.java:

```
package cadastrodb.modelo;

import cadastrodb.modelo.util.ConectorBD;
import cadastrodb.modelo.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;

/**
 *
 * @author Nanda
 */

public class PessoaJuridicaDAO {
    private final ConectorBD conector;
    private final SequenceManager sequenceManager;

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

    public PessoaJuridica getPessoa(int idPessoaJuridica) {
        Connection connection = null;
        PreparedStatement preparedStatement = null;
        ResultSet resultSet = null;

        try {
            connection = conector.getConnection();
            String sql = "SELECT * FROM pessoa_juridica WHERE idPessoaJuridica = ?";
            preparedStatement = connection.prepareStatement(sql);
            preparedStatement.setInt(1, idPessoaJuridica);
            resultSet = preparedStatement.executeQuery();

            if (resultSet.next()) {
                return buildPessoaJuridicaFromResultSet(resultSet);
            }
        } catch (SQLException e) {
        } finally {
            conector.close(resultSet);
            conector.close(preparedStatement);
            conector.close(connection);
        }
    }
}
```

```

        return null;
    }

    public List<PessoaJuridica> getPessoas() {
        Connection connection = null;
        PreparedStatement preparedStatement = null;
        ResultSet resultSet = null;
        List<PessoaJuridica> pessoasJuridicas = new ArrayList<>();

        try {
            connection = conector.getConnection();
            String sql = "SELECT * FROM pessoa_juridica";
            preparedStatement = connection.prepareStatement(sql);
            resultSet = preparedStatement.executeQuery();

            while (resultSet.next()) {

pessoasJuridicas.add(buildPessoaJuridicaFromResultSet(resultSet));
            }
        } catch (SQLException e) {
        } finally {
            conector.close(resultSet);
            conector.close(preparedStatement);
            conector.close(connection);
        }

        return pessoasJuridicas;
    }

    public void incluir(PessoaJuridica pessoaJuridica) {
        Connection connection = null;
        PreparedStatement preparedStatement = null;

        try {
            connection = conector.getConnection();
            connection.setAutoCommit(false);

            int idPessoaJuridica =
sequenceManager.getValue("pessoa_juridica_seq");

            String sqlPessoa = "INSERT INTO pessoa (idPessoaJuridica,
nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?,
?, ?)";

            preparedStatement = connection.prepareStatement(sqlPessoa);
            preparedStatement.setInt(1, idPessoaJuridica);
            preparedStatement.setString(2, pessoaJuridica.getNome());
            preparedStatement.setString(3,
pessoaJuridica.getLogradouro());
            preparedStatement.setString(4, pessoaJuridica.getCidade());

```

```

        preparedStatement.setString(5, pessoaJuridica.getEstado());
        preparedStatement.setString(6, pessoaJuridica.getTelefone());
        preparedStatement.setString(7, pessoaJuridica.getEmail());
        preparedStatement.executeUpdate();

        String sqlPessoaJuridica = "INSERT INTO pessoa_juridica
(idPessoaJuridica, cnpj) VALUES (?, ?)";
        preparedStatement =
connection.prepareStatement(sqlPessoaJuridica);
        preparedStatement.setInt(1, idPessoaJuridica);
        preparedStatement.setString(2, pessoaJuridica.getCnpj());
        preparedStatement.executeUpdate();

        connection.commit();
    } catch (SQLException e) {
        if (connection != null) {
            try {
                connection.rollback();
            } catch (SQLException rollbackException) {
            }
        }
    } finally {
        conector.close(preparedStatement);
        conector.close(connection);
    }
}

public void alterar(PessoaJuridica pessoaJuridica) {
    Connection connection = null;
    PreparedStatement preparedStatement = null;

    try {
        connection = conector.getConnection();
        connection.setAutoCommit(false);

        String sqlPessoa = "UPDATE pessoa SET nome = ?, logradouro =
?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE idPessoaJuridica
= ?";

        preparedStatement = connection.prepareStatement(sqlPessoa);
        preparedStatement.setString(1, pessoaJuridica.getNome());
        preparedStatement.setString(2,
pessoaJuridica.getLogradouro());
        preparedStatement.setString(3, pessoaJuridica.getCidade());
        preparedStatement.setString(4, pessoaJuridica.getEstado());
        preparedStatement.setString(5, pessoaJuridica.getTelefone());
        preparedStatement.setString(6, pessoaJuridica.getEmail());
        preparedStatement.setInt(7, pessoaJuridica.getId());
        preparedStatement.executeUpdate();
    }
}

```

```

        String sqlPessoaJuridica = "UPDATE pessoa_juridica SET cnpj =
? WHERE idPessoaJuridica = ?";
        preparedStatement =
connection.prepareStatement(sqlPessoaJuridica);
        preparedStatement.setString(1, pessoaJuridica.getCnpj());
        preparedStatement.setInt(2, pessoaJuridica.getId());
        preparedStatement.executeUpdate();

        connection.commit();
    } catch (SQLException e) {
        if (connection != null) {
            try {
                connection.rollback();
            } catch (SQLException rollbackException) {
            }
        }
    } finally {
        conector.close(preparedStatement);
        conector.close(connection);
    }
}

public void excluir(int idPessoaJuridica) {
    Connection connection = null;
    PreparedStatement preparedStatement = null;

    try {
        connection = conector.getConnection();
        connection.setAutoCommit(false);

        String sqlPessoaJuridica = "DELETE FROM pessoa_juridica WHERE
idPessoaJuridica = ?";
        preparedStatement =
connection.prepareStatement(sqlPessoaJuridica);
        preparedStatement.setInt(1, idPessoaJuridica);
        preparedStatement.executeUpdate();

        String sqlPessoa = "DELETE FROM pessoa WHERE idPessoaJuridica
= ?";

        preparedStatement = connection.prepareStatement(sqlPessoa);
        preparedStatement.setInt(1, idPessoaJuridica);
        preparedStatement.executeUpdate();

        connection.commit();
    } catch (SQLException e) {
        if (connection != null) {
            try {
                connection.rollback();
            } catch (SQLException rollbackException) {
            }
        }
    }
}

```

```

        }
    }
    } finally {
        conector.close(preparedStatement);
        conector.close(connection);
    }
}

private PessoaJuridica buildPessoaJuridicaFromResultSet(ResultSet
resultSet) throws SQLException {
    int idPessoaJuridica = resultSet.getInt("idPessoaJuridica");
    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 pessoaJuridica = new
PessoaJuridica(idPessoaJuridica, nome, logradouro, cidade, estado,
telefone, email, cnpj);
    return pessoaJuridica;
}
}

```

### Arquivo CadastroBD.java:

```

package cadastrobd.model;

import java.util.List;

/**
 *
 * @author Nanda
 */

public class CadastroBD {
    public static void main(String[] args) {
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();

        // Operações para Pessoa Física
        PessoaFisica pessoaFisica = new PessoaFisica(1, "Fulano", "Rua
A", "Cidade A", "Estado A", "123456789", "fulano@gmail.com",
"123.456.789-01");
        pessoaFisicaDAO.incluir(pessoaFisica);
        System.out.println("Pessoa Fisica incluida: ");
    }
}

```



```

        pessoaFisica.exibir(); // Exibir os dados da pessoa física

        // Alterar os dados da pessoa física
        pessoaFisica.setNome("Fulano da Silva");
        pessoaFisicaDAO.alterar(pessoaFisica);

        // Consultar e listar todas as pessoas físicas
        List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();
        for (PessoaFisica pf : pessoasFisicas) {
            pf.exibir();
        }

        // Excluir a pessoa física
        pessoaFisicaDAO.excluir(pessoaFisica.getId());
        System.out.println("\nPessoa Fisica excluida.");

        // Operações para Pessoa Jurídica
        PessoaJuridica pessoaJuridica = new PessoaJuridica(1, "Empresa
XYZ", "Rua B", "Cidade B", "Estado B", "987654321", "empresa@xyz.com",
"12345678901234");
        pessoaJuridicaDAO.incluir(pessoaJuridica);
        System.out.println("\nPessoa Juridica incluida: ");
        pessoaJuridica.exibir(); // Exibir os dados da pessoa jurídica

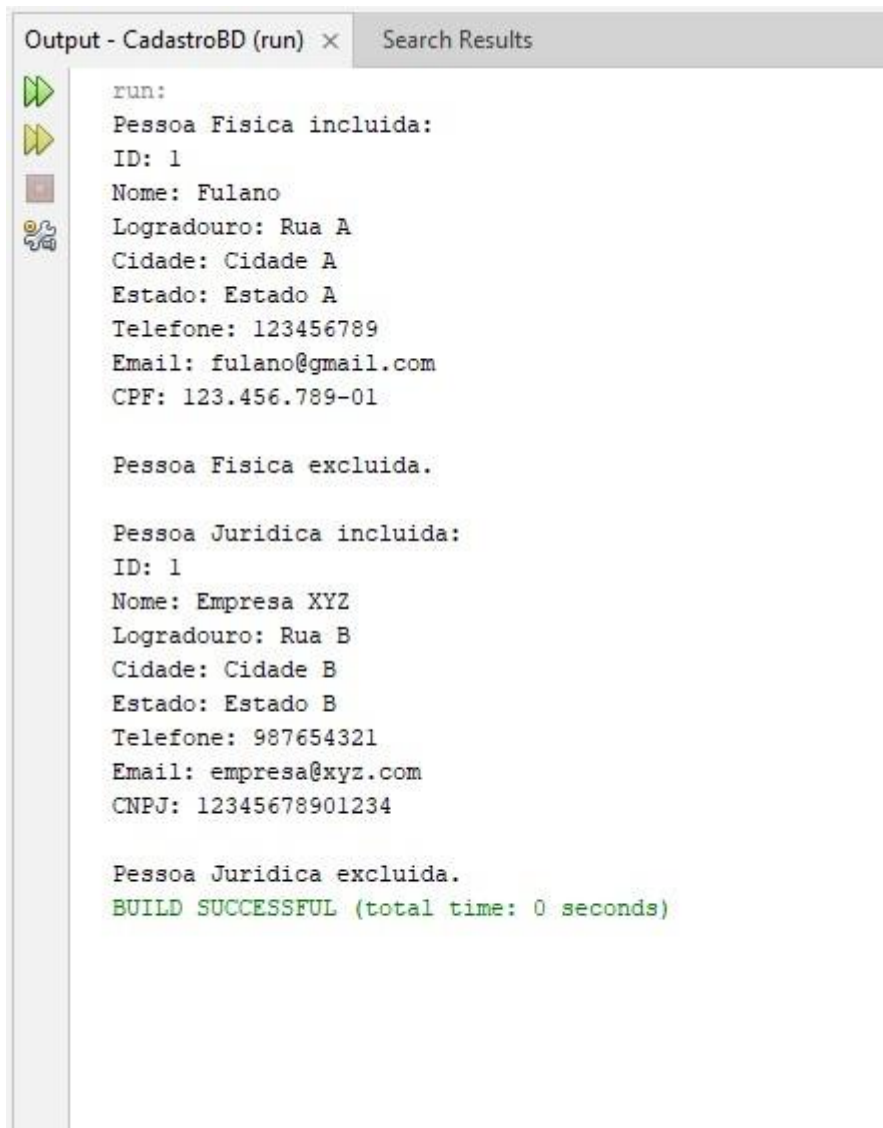
        // Alterar os dados da pessoa jurídica
        pessoaJuridica.setNome("Nova Empresa XYZ");
        pessoaJuridicaDAO.alterar(pessoaJuridica);

        // Consultar e listar todas as pessoas jurídicas
        List<PessoaJuridica> pessoasJuridicas =
pessoaJuridicaDAO.getPessoas();
        for (PessoaJuridica pj : pessoasJuridicas) {
            pj.exibir();
        }

        // Excluir a pessoa jurídica
        pessoaJuridicaDAO.excluir(pessoaJuridica.getId());
        System.out.println("\nPessoa Juridica excluida.");
    }
}

```

## Execução:



```
run:
Pessoa Fisica incluida:
ID: 1
Nome: Fulano
Logradouro: Rua A
Cidade: Cidade A
Estado: Estado A
Telefone: 123456789
Email: fulano@gmail.com
CPF: 123.456.789-01

Pessoa Fisica excluida.

Pessoa Juridica incluida:
ID: 1
Nome: Empresa XYZ
Logradouro: Rua B
Cidade: Cidade B
Estado: Estado B
Telefone: 987654321
Email: empresa@xyz.com
CNPJ: 12345678901234

Pessoa Juridica excluida.
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Análise e Conclusão:

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

R: Os componentes de middleware, como o JDBC, são cruciais porque facilitam a conexão entre aplicativos e bancos de dados, tornando a interação mais eficiente e segura.

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

R: A diferença é que o PreparedStatement é pré-compilado, o que melhora o desempenho e evita injeções de SQL, enquanto o Statement é menos seguro e mais lento.

3.Como o padrão DAO melhora a manutenibilidade do software?

R: O padrão DAO separa a lógica de acesso a dados da lógica de negócios, tornando o software mais modular e fácil de manter.

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

R: A herança no modelo relacional é geralmente refletida usando chaves estrangeiras e tabelas relacionadas para representar relacionamentos entre entidades.

## 2º Procedimento | Alimentando a base

**Arquivo CadastroBD.java:**

```
package cadastrobd.model;

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

/**
 *
 * @author Nanda
 */
```

```

public class CadastroBD {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();

        try {
            int opcao;
            do {
                System.out.println("\n===== Menu =====");
                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.print("Escolha uma opcao: ");

                opcao = scanner.nextInt();
                scanner.nextLine();

                switch (opcao) {
                    case 1:
                        incluirOpcao(scanner, pessoaFisicaDAO,
pessoaJuridicaDAO);
                        break;
                    case 2:
                        alterarOpcao(scanner, pessoaFisicaDAO,
pessoaJuridicaDAO);
                        break;
                    case 3:
                        excluirOpcao(scanner, pessoaFisicaDAO,
pessoaJuridicaDAO);
                        break;
                    case 4:
                        exibirPorIdOpcao(scanner, pessoaFisicaDAO,
pessoaJuridicaDAO);
                        break;
                    case 5:
                        exibirTodosOpcao(scanner, pessoaFisicaDAO,
pessoaJuridicaDAO);
                        break;
                    case 0:
                        System.out.println("Ate logo!");
                        break;
                    default:
                        System.out.println("Opcao invalida. Tente
novamente.");
                }
            } while (opcao != 0);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

        }
        } while (opcao != 0);
    } catch (InputMismatchException e) {
        System.out.println("Entrada invalida. Certifique-se de
inserir um numero do Menu.");
    } catch (Exception e) {
        System.out.println("Ocorreu um erro durante a execucao do
programa: " + e.getMessage());
    } finally {
        scanner.close();
    }
}

private static void incluirOpcao(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
    System.out.print("Escolha o tipo (1 - Pessoa Fisica, 2 - Pessoa
Juridica): ");
    int tipo = scanner.nextInt();
    scanner.nextLine();

    if (tipo == 1) {
        PessoaFisica pessoaFisica = obterDadosPessoaFisica(scanner);
        pessoaFisicaDAO.incluir(pessoaFisica);
        System.out.println("Pessoa Fisica incluida com sucesso!");
    } else if (tipo == 2) {
        PessoaJuridica pessoaJuridica =
obterDadosPessoaJuridica(scanner);
        pessoaJuridicaDAO.incluir(pessoaJuridica);
        System.out.println("Pessoa Juridica incluida com sucesso!");
    } else {
        System.out.println("Opcao invalida. Tente novamente.");
    }
}

private static void alterarOpcao(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
    System.out.print("Escolha o tipo (1 - Pessoa Fisica, 2 - Pessoa
Juridica): ");
    int tipo = scanner.nextInt();
    scanner.nextLine();

    System.out.print("Digite o ID da pessoa que deseja alterar: ");
    int id = scanner.nextInt();
    scanner.nextLine();

    if (tipo == 1) {
        PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);
        if (pessoaFisica != null) {
            System.out.println("Dados atuais da Pessoa Fisica:");
            pessoaFisica.exibir();
        }
    }
}

```

```

        System.out.println("Digite os novos dados:");
        PessoaFisica novosDados =
obterDadosPessoaFisica(scanner);
        novosDados.setId(id);
        pessoaFisicaDAO.alterar(novosDados);
        System.out.println("Pessoa Fisica alterada com
sucesso!");
    } else {
        System.out.println("Pessoa Fisica nao encontrada.");
    }
} else if (tipo == 2) {
    PessoaJuridica pessoaJuridica =
pessoaJuridicaDAO.getPessoa(id);
    if (pessoaJuridica != null) {
        System.out.println("Dados atuais da Pessoa Juridica:");
        pessoaJuridica.exibir();

        System.out.println("Digite os novos dados:");
        PessoaJuridica novosDados =
obterDadosPessoaJuridica(scanner);
        novosDados.setId(id);
        pessoaJuridicaDAO.alterar(novosDados);
        System.out.println("Pessoa Juridica alterada com
sucesso!");
    } else {
        System.out.println("Pessoa Juridica nao encontrada.");
    }
} else {
    System.out.println("Opcao invalida. Tente novamente.");
}
}

private static void excluirOpcao(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
    System.out.print("Escolha o tipo (1 - Pessoa Fisica, 2 - Pessoa
Juridica): ");
    int tipo = scanner.nextInt();
    scanner.nextLine();

    System.out.print("Digite o ID da pessoa que deseja excluir: ");
    int id = scanner.nextInt();
    scanner.nextLine();

    if (tipo == 1) {
        pessoaFisicaDAO.excluir(id);
        System.out.println("Pessoa Fisica excluida com sucesso!");
    } else if (tipo == 2) {
        pessoaJuridicaDAO.excluir(id);
        System.out.println("Pessoa Juridica excluida com sucesso!");
    }
}

```

```

    } else {
        System.out.println("Opcao invalida. Tente novamente.");
    }
}

private static void exibirPorIdOpcao(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
    System.out.print("Escolha o tipo (1 - Pessoa Fisica, 2 - Pessoa
Juridica): ");
    int tipo = scanner.nextInt();
    scanner.nextLine();

    System.out.print("Digite o ID da pessoa que deseja exibir: ");
    int id = scanner.nextInt();
    scanner.nextLine();

    if (tipo == 1) {
        PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);
        if (pessoaFisica != null) {
            System.out.println("Dados da Pessoa Fisica:");
            pessoaFisica.exibir();
        } else {
            System.out.println("Pessoa Fisica nao encontrada.");
        }
    } else if (tipo == 2) {
        PessoaJuridica pessoaJuridica =
pessoaJuridicaDAO.getPessoa(id);
        if (pessoaJuridica != null) {
            System.out.println("Dados da Pessoa Juridica:");
            pessoaJuridica.exibir();
        } else {
            System.out.println("Pessoa Juridica nao encontrada.");
        }
    } else {
        System.out.println("Opcao invalida. Tente novamente.");
    }
}

private static void exibirTodosOpcao(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
    System.out.print("Escolha o tipo (1 - Pessoa Fisica, 2 - Pessoa
Juridica): ");
    int tipo = scanner.nextInt();
    scanner.nextLine();

    if (tipo == 1) {
        List<PessoaFisica> pessoasFisicas =
pessoaFisicaDAO.getPessoas();
        System.out.println("\nTodas as Pessoas Fisicas no banco:");
        for (PessoaFisica pf : pessoasFisicas) {

```

```

        pf.exibir();
    }
    } else if (tipo == 2) {
        List<PessoaJuridica> pessoasJuridicas =
        pessoaJuridicaDAO.getPessoas();
        System.out.println("\nTodas as Pessoas Juridicas no banco:");
        for (PessoaJuridica pj : pessoasJuridicas) {
            pj.exibir();
        }
    } else {
        System.out.println("Opcao invalida. Tente novamente.");
    }
}

private static PessoaFisica obterDadosPessoaFisica(Scanner scanner) {
    System.out.print("Digite o nome: ");
    String nome = scanner.nextLine();

    System.out.print("Digite o logradouro: ");
    String logradouro = scanner.nextLine();

    System.out.print("Digite a cidade: ");
    String cidade = scanner.nextLine();

    System.out.print("Digite o estado: ");
    String estado = scanner.nextLine();

    System.out.print("Digite o telefone: ");
    String telefone = scanner.nextLine();

    System.out.print("Digite o e-mail: ");
    String email = scanner.nextLine();

    System.out.print("Digite o CPF: ");
    String cpf = scanner.nextLine();

    return new PessoaFisica(0, nome, logradouro, cidade, estado,
    telefone, email, cpf);
}

private static PessoaJuridica obterDadosPessoaJuridica(Scanner
scanner) {
    System.out.print("Digite o nome: ");
    String nome = scanner.nextLine();

    System.out.print("Digite o logradouro: ");
    String logradouro = scanner.nextLine();

    System.out.print("Digite a cidade: ");
    String cidade = scanner.nextLine();

```



```
        System.out.print("Digite o estado: ");
        String estado = scanner.nextLine();

        System.out.print("Digite o telefone: ");
        String telefone = scanner.nextLine();

        System.out.print("Digite o e-mail: ");
        String email = scanner.nextLine();

        System.out.print("Digite o CNPJ: ");
        String cnpj = scanner.nextLine();

        return new PessoaJuridica(0, nome, logradouro, cidade, estado,
        telefone, email, cnpj);
    }
}
```

## Execução:

```
Output - CadastroBD (run) × Search Results

===== Menu =====
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir Todos
0 - Sair
Escolha uma opcao: 1
Escolha o tipo (1 - Pessoa Fisica, 2 - Pessoa Juridica): 1
Digite o nome: Joana Silva
Digite o logradouro: Rua comprida, 27
Digite a cidade: Lagoinha
Digite o estado: LO
Digite o telefone: 235416874
Digite o e-mail: joana@email.com
Digite o CPF: 16524895321
Pessoa Fisica incluida com sucesso!

===== Menu =====
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir Todos
0 - Sair
Escolha uma opcao: 1
Escolha o tipo (1 - Pessoa Fisica, 2 - Pessoa Juridica): 2
Digite o nome: Empresa LTDA
Digite o logradouro: Rua Movimentada, 90
Digite a cidade: Cidade
Digite o estado: CI
Digite o telefone: 956321485
Digite o e-mail: empresa@email.com
Digite o CNPJ: 32584951265732
Pessoa Juridica incluida com sucesso!

===== Menu =====
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir Todos
0 - Sair
Escolha uma opcao: 0
Ate logo!
BUILD SUCCESSFUL (total time: 2 minutes 24 seconds)
```

### **Análise e Conclusão:**

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

R: A persistência em arquivo envolve armazenar dados em arquivos no sistema de arquivos, enquanto a persistência em banco de dados usa um sistema de gerenciamento de banco de dados para armazenar dados de forma estruturada e mais eficiente.

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

R: O uso de operadores lambda simplificou a impressão de valores nas entidades do Java, permitindo que você defina uma função inline para processar e imprimir os valores de forma mais concisa.

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

R: Porque o método main é estático e só pode chamar outros métodos estáticos. Isso permite que o método main seja invocado sem a necessidade de criar uma instância da classe.