



Missão Prática | Nível 3 | Mundo 3

Eduardo Eugênio de Araujo e Silva Domingues 202302579043

**1172 POLO CASA CAIADA – OLINDA – PE
RPG0016 - BackEnd sem banco não tem – 2024.1**

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.

1º Procedimento | Mapeamento Objeto-Relacional e DAO

Pessoa.java

```
package cadastrbd.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;
```

```
    public Pessoa() {  
    }  
}
```



```
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;  
}  
}
```

PessoaFisica.java

```
package cadastrbd.model;  
  
public class PessoaFisica extends Pessoa {  
    private String cpf;
```



```
public PessoaFisica() {  
}  
  
public String getCpf() {  
    return cpf;  
}  
  
public void setCpf(String cpf) {  
    this.cpf = cpf;  
}  
}
```

PessoaFisicaDAO.java

```
package cadastrbd.model;  
  
import cadastro.model.util.ConectorBD;  
  
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 {  
    public PessoaFisica getPessoa(int id) {  
        PessoaFisica pessoa = null;  
        Connection conn = null;  
        PreparedStatement pstmt = null;  
        ResultSet rs = null;
```



```
try {
    conn = ConectorBD.getConnection();
    String sql = "SELECT * FROM pessoa INNER JOIN pessoa_fisica ON
pessoa.idPessoa = pessoa_fisica.idPessoa WHERE pessoa.idPessoa = ?";
    pstmt = conn.prepareStatement(sql);
    pstmt.setInt(1, id);
    rs = pstmt.executeQuery();

    if (rs.next()) {
        pessoa = new PessoaFisica();
        pessoa.setIdPessoa(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"));
    }
} catch (SQLException e) {
    e.printStackTrace();
} finally {
    ConectorBD.close(rs);
    ConectorBD.close(pstmt);
    ConectorBD.close(conn);
}

return pessoa;
}

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



Estácio

```
PreparedStatement pstmt = null;
```

```
ResultSet rs = null;
```

```
try {
```

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

```
    String sql = "SELECT * FROM pessoa INNER JOIN pessoa_fisica ON  
pessoa.idPessoa = pessoa_fisica.idPessoa";
```

```
    pstmt = conn.prepareStatement(sql);
```

```
    rs = pstmt.executeQuery();
```

```
    while (rs.next()) {
```

```
        PessoaFisica pessoa = new PessoaFisica();
```

```
        pessoa.setIdPessoa(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(pstmt);
```

```
    ConectorBD.close(conn);
```

```
}
```

```
return pessoas;
```

```
}
```

```
public boolean incluir(PessoaFisica pessoaFisica) {
```



```
boolean sucesso = false;
Connection conn = null;
PreparedStatement pstmtPessoa = null;
PreparedStatement pstmtPessoaFisica = null;
ResultSet rs = null;

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

    String sqlSeq = "SELECT NEXT VALUE FOR SEQ_idPessoa AS id";
    pstmtPessoa = conn.prepareStatement(sqlSeq);
    rs = pstmtPessoa.executeQuery();
    int idPessoa = 0;
    if (rs.next()) {
        idPessoa = rs.getInt("id");
    }

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

    String sqlPessoaFisica = "INSERT INTO pessoa_fisica (idPessoa, cpf) VALUES
(?, ?)";
    pstmtPessoaFisica = conn.prepareStatement(sqlPessoaFisica);
    pstmtPessoaFisica.setInt(1, idPessoa);
```



Estácio

```
pstmtPessoaFisica.setString(2, pessoaFisica.getCpf());
int linhasAfetadasPessoaFisica = pstmtPessoaFisica.executeUpdate();

if (linhasAfetadasPessoa > 0 && linhasAfetadasPessoaFisica > 0) {
    conn.commit();
    sucesso = true;
} else {
    conn.rollback();
}
} catch (SQLException e) {
    e.printStackTrace();
    try {
        if (conn != null) {
            conn.rollback();
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
} finally {
    try {
        if (conn != null) {
            conn.setAutoCommit(true);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    ConectorBD.close(rs);
    ConectorBD.close(pstmtPessoaFisica);
    ConectorBD.close(pstmtPessoa);
    ConectorBD.close(conn);
}

return sucesso;
```




```
}
```

```
public boolean alterar(PessoaFisica pessoaFisica) {
```

```
    String sql = "UPDATE pessoa SET nome = ?, logradouro = ?, cidade = ?, estado =  
    ?, telefone = ?, email = ? WHERE idPessoa = ?";
```

```
    try (PreparedStatement stmt = ConectorBD.getPrepared(sql)) {
```

```
        stmt.setString(1, pessoaFisica.getNome());
```

```
        stmt.setString(2, pessoaFisica.getLogradouro());
```

```
        stmt.setString(3, pessoaFisica.getCidade());
```

```
        stmt.setString(4, pessoaFisica.getEstado());
```

```
        stmt.setString(5, pessoaFisica.getTelefone());
```

```
        stmt.setString(6, pessoaFisica.getEmail());
```

```
        stmt.setInt(7, pessoaFisica.getIdPessoa());
```

```
        int rowsAffected = stmt.executeUpdate();
```

```
        return rowsAffected > 0;
```

```
    } catch (SQLException e) {
```

```
        e.printStackTrace();
```

```
        return false;
```

```
    }
```

```
}
```

```
public boolean excluir(int id) {
```

```
    boolean sucesso = false;
```

```
    Connection conn = null;
```

```
    PreparedStatement pstmtPessoaFisica = null;
```

```
    PreparedStatement pstmtPessoa = null;
```

```
    try {
```

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

```
        conn.setAutoCommit(false);
```

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

```
        pstmtPessoaFisica = conn.prepareStatement(sqlPessoaFisica);
```

```
        pstmtPessoaFisica.setInt(1, id);
```



Estácio

```
int linhasAfetadasPessoaFisica = pstmtPessoaFisica.executeUpdate();

String sqlPessoa = "DELETE FROM pessoa WHERE idPessoa=?";
pstmtPessoa = conn.prepareStatement(sqlPessoa);
pstmtPessoa.setInt(1, id);
int linhasAfetadasPessoa = pstmtPessoa.executeUpdate();

if (linhasAfetadasPessoaFisica > 0 && linhasAfetadasPessoa > 0) {
    conn.commit();
    sucesso = true;
} else {
    conn.rollback();
}
} catch (SQLException e) {
    e.printStackTrace();
    try {
        if (conn != null) {
            conn.rollback();
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
} finally {
    try {
        if (conn != null) {
            conn.setAutoCommit(true);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
ConectorBD.close(pstmtPessoaFisica);
ConectorBD.close(pstmtPessoa);
ConectorBD.close(conn);
```



```
}  
  
    return sucesso;  
}  
}
```

PessoaJuridica.java

```
package cadastrbd.model;  
  
public class PessoaJuridica extends Pessoa {  
    private String cnpj;  
  
    public PessoaJuridica() {  
    }  
  
    public String getCnpj() {  
        return cnpj;  
    }  
  
    public void setCnpj(String cnpj) {  
        this.cnpj = cnpj;  
    }  
}
```

PessoaJuridicaDAO.java

```
package cadastrbd.model;
```



Estácio

```
import cadastro.model.util.ConectorBD;

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 pessoaJuridica = null;

        Connection conn = null;

        PreparedStatement pstmt = null;

        ResultSet rs = null;

        try {

            conn = ConectorBD.getConnection();

            String sql = "SELECT * FROM pessoa INNER JOIN pessoa_juridica ON
pessoa.idPessoa = pessoa_juridica.idPessoa WHERE pessoa.idPessoa = ?";

            pstmt = conn.prepareStatement(sql);

            pstmt.setInt(1, id);

            rs = pstmt.executeQuery();

            if (rs.next()) {
```



Estácio

```
    pessoaJuridica = new PessoaJuridica();

    pessoaJuridica.setIdPessoa(rs.getInt("idPessoa"));

    pessoaJuridica.setNome(rs.getString("nome"));

    pessoaJuridica.setLogradouro(rs.getString("logradouro"));

    pessoaJuridica.setCidade(rs.getString("cidade"));

    pessoaJuridica.setEstado(rs.getString("estado"));

    pessoaJuridica.setTelefone(rs.getString("telefone"));

    pessoaJuridica.setEmail(rs.getString("email"));

    pessoaJuridica.setCnpj(rs.getString("cnpj"));

}

} catch (SQLException e) {

    e.printStackTrace();

} finally {

    ConectorBD.close(rs);

    ConectorBD.close(pstmt);

    ConectorBD.close(conn);

}

return pessoaJuridica;

}

public List<PessoaJuridica> getPessoas() {

    List<PessoaJuridica> pessoasJuridicas = new ArrayList<>();

    Connection conn = null;

    PreparedStatement pstmt = null;
```



```
ResultSet rs = null;

try {

    conn = ConectorBD.getConnection();

    String sql = "SELECT * FROM pessoa INNER JOIN pessoa_juridica ON
pessoa.idPessoa = pessoa_juridica.idPessoa";

    pstmt = conn.prepareStatement(sql);

    rs = pstmt.executeQuery();

    while (rs.next()) {

        PessoaJuridica pessoaJuridica = new PessoaJuridica();

        pessoaJuridica.setIdPessoa(rs.getInt("idPessoa"));

        pessoaJuridica.setNome(rs.getString("nome"));

        pessoaJuridica.setLogradouro(rs.getString("logradouro"));

        pessoaJuridica.setCidade(rs.getString("cidade"));

        pessoaJuridica.setEstado(rs.getString("estado"));

        pessoaJuridica.setTelefone(rs.getString("telefone"));

        pessoaJuridica.setEmail(rs.getString("email"));

        pessoaJuridica.setCnpj(rs.getString("cnpj"));

        pessoasJuridicas.add(pessoaJuridica);

    }

} catch (SQLException e) {

    e.printStackTrace();

} finally {

    ConectorBD.close(rs);
```



```
ConectorBD.close(pstmt);

ConectorBD.close(conn);

}

return pessoasJuridicas;

}

public boolean incluir(PessoaJuridica pessoaJuridica) {

    boolean sucesso = false;

    Connection conn = null;

    PreparedStatement pstmtPessoa = null;

    PreparedStatement pstmtPessoaJuridica = null;

    ResultSet rs = null;

    try {

        conn = ConectorBD.getConnection();

        conn.setAutoCommit(false);

        String sqlSeq = "SELECT NEXT VALUE FOR SEQ_idPessoa AS id";

        pstmtPessoa = conn.prepareStatement(sqlSeq);

        rs = pstmtPessoa.executeQuery();

        int idPessoa = 0;

        if (rs.next()) {

            idPessoa = rs.getInt("id");

        }

    }
```



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

```
pstmtPessoa = conn.prepareStatement(sqlPessoa);  
pstmtPessoa.setInt(1, idPessoa);  
pstmtPessoa.setString(2, pessoaJuridica.getNome());  
pstmtPessoa.setString(3, pessoaJuridica.getLogradouro());  
pstmtPessoa.setString(4, pessoaJuridica.getCidade());  
pstmtPessoa.setString(5, pessoaJuridica.getEstado());  
pstmtPessoa.setString(6, pessoaJuridica.getTelefone());  
pstmtPessoa.setString(7, pessoaJuridica.getEmail());  
int linhasAfetadasPessoa = pstmtPessoa.executeUpdate();
```

```
String sqlPessoaJuridica = "INSERT INTO pessoa_juridica (idPessoa, cnpj) VALUES (?, ?)";
```

```
pstmtPessoaJuridica = conn.prepareStatement(sqlPessoaJuridica);  
pstmtPessoaJuridica.setInt(1, idPessoa);  
pstmtPessoaJuridica.setString(2, pessoaJuridica.getCnpj());  
int linhasAfetadasPessoaJuridica = pstmtPessoaJuridica.executeUpdate();
```

```
if (linhasAfetadasPessoa > 0 && linhasAfetadasPessoaJuridica > 0) {  
    conn.commit();  
    sucesso = true;  
} else {  
    conn.rollback();
```




```
}  
  
} catch (SQLException e) {  
    e.printStackTrace();  
  
    try {  
        if (conn != null) {  
            conn.rollback();  
        }  
    } catch (SQLException ex) {  
        ex.printStackTrace();  
    }  
}  
  
} finally {  
    try {  
        if (conn != null) {  
            conn.setAutoCommit(true);  
        }  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
  
    ConectorBD.close(rs);  
  
    ConectorBD.close(pstmtPessoaJuridica);  
  
    ConectorBD.close(pstmtPessoa);  
  
    ConectorBD.close(conn);  
}  
  
return sucesso;
```



```
}
```

```
public boolean alterar(PessoaJuridica pessoaJuridica) {  
    boolean sucesso = false;  
  
    Connection conn = null;  
  
    PreparedStatement pstmtPessoa = null;  
    PreparedStatement pstmtPessoaJuridica = null;  
  
    try {  
        conn = ConectorBD.getConnection();  
        conn.setAutoCommit(false);  
  
        String sqlPessoa = "UPDATE pessoa SET nome=?, logradouro=?, cidade=?,  
estado=?, telefone=?, email=? WHERE idPessoa=?";  
  
        pstmtPessoa = conn.prepareStatement(sqlPessoa);  
  
        pstmtPessoa.setString(1, pessoaJuridica.getNome());  
        pstmtPessoa.setString(2, pessoaJuridica.getLogradouro());  
        pstmtPessoa.setString(3, pessoaJuridica.getCidade());  
        pstmtPessoa.setString(4, pessoaJuridica.getEstado());  
        pstmtPessoa.setString(5, pessoaJuridica.getTelefone());  
        pstmtPessoa.setString(6, pessoaJuridica.getEmail());  
        pstmtPessoa.setInt(7, pessoaJuridica.getIdPessoa());  
  
        int linhasAfetadasPessoa = pstmtPessoa.executeUpdate();  
    }  
}
```



Estácio

```
String sqlPessoaJuridica = "UPDATE pessoa_juridica SET cnpj=? WHERE  
idPessoa=?";
```

```
pstmtPessoaJuridica = conn.prepareStatement(sqlPessoaJuridica);
```

```
pstmtPessoaJuridica.setString(1, pessoaJuridica.getCnpj());
```

```
pstmtPessoaJuridica.setInt(2, pessoaJuridica.getIdPessoa());
```

```
int linhasAfetadasPessoaJuridica = pstmtPessoaJuridica.executeUpdate();
```

```
if (linhasAfetadasPessoa > 0 && linhasAfetadasPessoaJuridica > 0) {
```

```
    conn.commit();
```

```
    sucesso = true;
```

```
} else {
```

```
    conn.rollback();
```

```
}
```

```
} catch (SQLException e) {
```

```
    e.printStackTrace();
```

```
try {
```

```
    if (conn != null) {
```

```
        conn.rollback();
```

```
    }
```

```
} catch (SQLException ex) {
```

```
    ex.printStackTrace();
```

```
}
```

```
} finally {
```

```
try {
```

```
    if (conn != null) {
```



Estácio

```
        conn.setAutoCommit(true);

    }

    } catch (SQLException e) {

        e.printStackTrace();

    }

    ConectorBD.close(pstmtPessoaJuridica);

    ConectorBD.close(pstmtPessoa);

    ConectorBD.close(conn);

}

return sucesso;

}

public boolean excluir(int id) {

    boolean sucesso = false;

    Connection conn = null;

    PreparedStatement pstmtPessoaJuridica = null;

    PreparedStatement pstmtPessoa = null;

    try {

        conn = ConectorBD.getConnection();

        conn.setAutoCommit(false);

        String sqlPessoaJuridica = "DELETE FROM pessoa_juridica WHERE
idPessoa=?";
```



Estácio

```
pstmtPessoaJuridica = conn.prepareStatement(sqlPessoaJuridica);

pstmtPessoaJuridica.setInt(1, id);

int linhasAfetadasPessoaJuridica = pstmtPessoaJuridica.executeUpdate();


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

pstmtPessoa = conn.prepareStatement(sqlPessoa);

pstmtPessoa.setInt(1, id);

int linhasAfetadasPessoa = pstmtPessoa.executeUpdate();


if (linhasAfetadasPessoaJuridica > 0 && linhasAfetadasPessoa > 0) {

    conn.commit();

    sucesso = true;

} else {

    conn.rollback();

}

} catch (SQLException e) {

    e.printStackTrace();

    try {

        if (conn != null) {

            conn.rollback();

        }

    } catch (SQLException ex) {

        ex.printStackTrace();

    }

} finally {
```



```
try {  
    if (conn != null) {  
        conn.setAutoCommit(true);  
    }  
} catch (SQLException e) {  
    e.printStackTrace();  
}  
  
ConectorBD.close(pstmtPessoaJuridica);  
  
ConectorBD.close(pstmtPessoa);  
  
ConectorBD.close(conn);  
}  
  
return sucesso;  
}  
}
```

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;  
import java.sql.Statement;
```



```
public class ConectorBD {  
    private static final String URL =  
    "jdbc:sqlserver://localhost:1433;databaseName=Loja;encrypt=true;trustServerCertificate=true";  
    private static final String USER = "loja";  
    private static final String PASSWORD = "loja";  
  
    public static Connection getConnection() throws SQLException {  
        return DriverManager.getConnection(URL, USER, PASSWORD);  
    }  
  
    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(Statement stmt) {  
        try {  
            if (stmt != null) {  
                stmt.close();  
            }  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
    }  
  
    public static void close(ResultSet rs) {  
        try {  
            if (rs != null) {  
                rs.close();  
            }  
        }  
    }  
}
```



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

SequenceManager.java

```
package cadastro.model.util;  
  
import java.sql.Connection;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
  
public class SequenceManager {  
    public static int getValue(String SEQ_idPessoa) {  
        int nextValue = 0;  
        Connection conn = null;  
        PreparedStatement pstmt = null;  
        ResultSet rs = null;
```




```
try {
    conn = ConectorBD.getConnection();
    String sql = "SELECT NEXT VALUE FOR " + SEQ_idPessoa + " AS
next_value";
    pstmt = conn.prepareStatement(sql);
    rs = pstmt.executeQuery();
    if (rs.next()) {
        nextValue = rs.getInt("next_value");
    }
} catch (SQLException e) {
    e.printStackTrace();
} finally {
    ConectorBD.close(rs);
    ConectorBD.close(pstmt);
    ConectorBD.close(conn);
}

return nextValue;
}
}
```



Estácio

```
Output - CadastroBD (run) x
run:
Pessoas fisicas:
Id: 679
Nome: Joao
Logradouro: Blablabla
Cidade: Belem
Estado: PA
Telefone: 1111-1111
E-mail: joao@riacho.com
CPF: 11111111111

Pessoas juridicas:
Id: 687
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho
Estado: PA
Telefone: 1212-1212
E-mail: jjc@riacho.com
CNPJ: 11111111111111

BUILD SUCCESSFUL (total time: 0 seconds)
```

Conclusão:

- Os componentes de middleware, como o JDBC, facilitam a conexão e interação com diferentes bancos de dados, promovendo portabilidade e modularidade no desenvolvimento de aplicativos.
- O PreparedStatement é preferível ao Statement devido à sua capacidade de pré-compilar consultas SQL, melhorando o desempenho e prevenindo ataques de injeção de SQL.
- O padrão DAO separa a lógica de acesso aos dados da lógica de negócios, facilitando a manutenção, testabilidade e reutilização do código ao isolar as operações de banco de dados em classes específicas.
- No modelo relacional, a herança é frequentemente representada usando-se chaves estrangeiras para mapear relacionamentos entre tabelas, onde uma tabela filha referencia a tabela pai por meio de uma chave estrangeira que coincide com a chave primária da tabela pai.



Observe que os tópicos acima seguem exatamente o que está na Atividade Prática exigida.

2º Procedimento | Alimentando a Base

CadastroBD.java

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

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



```
while (true) {  
    System.out.println("Selecione a 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 - Finalizar");  
    int opcao = scanner.nextInt();  
    scanner.nextLine();  
  
    switch (opcao) {  
        case 1:  
            incluir(scanner);  
            break;  
        case 2:  
            alterar(scanner);  
            break;  
        case 3:  
            excluir(scanner);  
            break;  
        case 4:  
            exibirPorId(scanner);  
            break;  
        case 5:  
            exibirTodos();  
            break;  
        case 0:  
            System.out.println("Finalizando...");  
            return;  
        default:  
            System.out.println("Opção inválida. Tente novamente.");  
    }  
}
```



```
}  
}  
}
```

```
private static void incluir(Scanner scanner) {  
    System.out.println("Selecione o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica:");  
    int tipo = scanner.nextInt();  
    scanner.nextLine();  
  
    if (tipo == 1) {  
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();  
        PessoaFisica pessoaFisica = new PessoaFisica();  
  
        System.out.println("Digite o nome:");  
        pessoaFisica.setNome(scanner.nextLine());  
  
        System.out.println("Digite o logradouro:");  
        pessoaFisica.setLogradouro(scanner.nextLine());  
  
        System.out.println("Digite a cidade:");  
        pessoaFisica.setCidade(scanner.nextLine());  
  
        System.out.println("Digite o estado:");  
        pessoaFisica.setEstado(scanner.nextLine());  
  
        System.out.println("Digite o telefone:");  
        pessoaFisica.setTelefone(scanner.nextLine());  
  
        System.out.println("Digite o email:");  
        pessoaFisica.setEmail(scanner.nextLine());  
  
        System.out.println("Digite o CPF:");  
        pessoaFisica.setCpf(scanner.nextLine());  
    }  
}
```



```
    pessoaFisicaDAO.incluir(pessoaFisica);
    System.out.println("Pessoa física incluída com sucesso.");
} else if (tipo == 2) {
    PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();
    PessoaJuridica pessoaJuridica = new PessoaJuridica();

    System.out.println("Digite o nome:");
    pessoaJuridica.setNome(scanner.nextLine());

    System.out.println("Digite o logradouro:");
    pessoaJuridica.setLogradouro(scanner.nextLine());

    System.out.println("Digite a cidade:");
    pessoaJuridica.setCidade(scanner.nextLine());

    System.out.println("Digite o estado:");
    pessoaJuridica.setEstado(scanner.nextLine());

    System.out.println("Digite o telefone:");
    pessoaJuridica.setTelefone(scanner.nextLine());

    System.out.println("Digite o email:");
    pessoaJuridica.setEmail(scanner.nextLine());

    System.out.println("Digite o CNPJ:");
    pessoaJuridica.setCnpj(scanner.nextLine());

    pessoaJuridicaDAO.incluir(pessoaJuridica);
    System.out.println("Pessoa jurídica incluída com sucesso.");
} else {
    System.out.println("Tipo inválido.");
}
```



```
}
```

```
private static void alterar(Scanner scanner) {  
    System.out.println("Selecione o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):");  
    int tipo = scanner.nextInt();  
    scanner.nextLine();  
  
    if (tipo == 1) {  
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();  
  
        System.out.println("Digite o ID da pessoa física a ser alterada:");  
        int id = scanner.nextInt();  
        scanner.nextLine();  
  
        PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);  
  
        if (pessoaFisica != null) {  
            System.out.println("Dados atuais:");  
            System.out.println(pessoaFisica);  
  
            System.out.println("Digite o novo nome:");  
            pessoaFisica.setNome(scanner.nextLine());  
  
            System.out.println("Digite o novo logradouro:");  
            pessoaFisica.setLogradouro(scanner.nextLine());  
  
            System.out.println("Digite a nova cidade:");  
            pessoaFisica.setCidade(scanner.nextLine());  
  
            System.out.println("Digite o novo estado:");  
            pessoaFisica.setEstado(scanner.nextLine());  
  
            System.out.println("Digite o novo telefone:");
```



Estácio

```
    pessoaFisica.setTelefone(scanner.nextLine());

    System.out.println("Digite o novo email:");
    pessoaFisica.setEmail(scanner.nextLine());

    pessoaFisicaDAO.alterar(pessoaFisica);
    System.out.println("Pessoa física alterada com sucesso.");
} else {
    System.out.println("Pessoa física não encontrada.");
}
} else if (tipo == 2) {
    PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();

    System.out.println("Digite o ID da pessoa jurídica a ser alterada:");
    int id = scanner.nextInt();
    scanner.nextLine();

    PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);

    if (pessoaJuridica != null) {
        System.out.println("Dados atuais:");
        System.out.println(pessoaJuridica);

        System.out.println("Digite o novo nome:");
        pessoaJuridica.setNome(scanner.nextLine());

        System.out.println("Digite o novo logradouro:");
        pessoaJuridica.setLogradouro(scanner.nextLine());

        System.out.println("Digite a nova cidade:");
        pessoaJuridica.setCidade(scanner.nextLine());

        System.out.println("Digite o novo estado:");
```




Estácio

```
    pessoaJuridica.setEstado(scanner.nextLine());
```

```
    System.out.println("Digite o novo telefone:");
    pessoaJuridica.setTelefone(scanner.nextLine());
```

```
    System.out.println("Digite o novo email:");
    pessoaJuridica.setEmail(scanner.nextLine());
```

```
    pessoaJuridicaDAO.alterar(pessoaJuridica);
    System.out.println("Pessoa jurídica alterada com sucesso.");
} else {
    System.out.println("Pessoa jurídica não encontrada.");
}
} else {
    System.out.println("Tipo inválido.");
}
}
```

```
private static void excluir(Scanner scanner) {
    System.out.println("Selecione o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica:");
    int tipo = scanner.nextInt();
    scanner.nextLine();
```

```
    if (tipo == 1) {
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();

        System.out.println("Digite o ID da pessoa física a ser excluída:");
        int id = scanner.nextInt();
        scanner.nextLine();

        PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);

        if (pessoaFisica != null) {
```



Estácio

```
        pessoaFisicaDAO.excluir(id);
        System.out.println("Pessoa física excluída com sucesso.");
    } else {
        System.out.println("Pessoa física não encontrada.");
    }
} else if (tipo == 2) {
    PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();

    System.out.println("Digite o ID da pessoa jurídica a ser excluída:");
    int id = scanner.nextInt();
    scanner.nextLine();

    PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);

    if (pessoaJuridica != null) {
        pessoaJuridicaDAO.excluir(id);
        System.out.println("Pessoa jurídica excluída com sucesso.");
    } else {
        System.out.println("Pessoa jurídica não encontrada.");
    }
} else {
    System.out.println("Tipo inválido.");
}
}

private static void exibirPorId(Scanner scanner) {
    System.out.println("Selecione o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):");
    int tipo = scanner.nextInt();
    scanner.nextLine();

    if (tipo == 1) {
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
```



```
System.out.println("Digite o ID da pessoa física a ser exibida:");
int id = scanner.nextInt();
scanner.nextLine();

PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);

if (pessoaFisica != null) {
    System.out.println("Pessoa física encontrada:");
    imprimirPessoa(pessoaFisica);
} else {
    System.out.println("Pessoa física não encontrada.");
}
} else if (tipo == 2) {
    PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();

    System.out.println("Digite o ID da pessoa jurídica a ser exibida:");
    int id = scanner.nextInt();
    scanner.nextLine();

    PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);

    if (pessoaJuridica != null) {
        System.out.println("Pessoa jurídica encontrada:");
        imprimirPessoa(pessoaJuridica);
    } else {
        System.out.println("Pessoa jurídica não encontrada.");
    }
} else {
    System.out.println("Tipo inválido.");
}
}
```



Estácio

```
private static void exibirTodos() {
    Scanner scanner = new Scanner(System.in);

    System.out.println("Selecione o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):");
    int tipo = scanner.nextInt();
    scanner.nextLine();

    if (tipo == 1) {
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();

        List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();
        if (!pessoasFisicas.isEmpty()) {
            System.out.println("Pessoas físicas encontradas:");
            for (PessoaFisica pf : pessoasFisicas) {
                imprimirPessoa(pf);
            }
        } else {
            System.out.println("Nenhuma pessoa física encontrada.");
        }
    } else if (tipo == 2) {
        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();

        List<PessoaJuridica> pessoasJuridicas = pessoaJuridicaDAO.getPessoas();
        if (!pessoasJuridicas.isEmpty()) {
            System.out.println("Pessoas jurídicas:");
            for (PessoaJuridica pj : pessoasJuridicas) {
                imprimirPessoa(pj);
            }
        } else {
            System.out.println("Nenhuma pessoa jurídica encontrada.");
        }
    } else {
        System.out.println("Tipo inválido.");
    }
}
```



```
}  
}
```

```
private static void imprimirPessoa(PessoaFisica pessoa) {  
    System.out.println("Id: " + pessoa.getIdPessoa());  
    System.out.println("Nome: " + pessoa.getNome());  
    System.out.println("Logradouro: " + pessoa.getLogradouro());  
    System.out.println("Cidade: " + pessoa.getCidade());  
    System.out.println("Estado: " + pessoa.getEstado());  
    System.out.println("Telefone: " + pessoa.getTelefone());  
    System.out.println("E-mail: " + pessoa.getEmail());  
    System.out.println("CPF: " + pessoa.getCpf());  
    System.out.println();  
}
```

```
private static void imprimirPessoa(PessoaJuridica pessoa) {  
    System.out.println("Id: " + pessoa.getIdPessoa());  
    System.out.println("Nome: " + pessoa.getNome());  
    System.out.println("Logradouro: " + pessoa.getLogradouro());  
    System.out.println("Cidade: " + pessoa.getCidade());  
    System.out.println("Estado: " + pessoa.getEstado());  
    System.out.println("Telefone: " + pessoa.getTelefone());  
    System.out.println("E-mail: " + pessoa.getEmail());  
    System.out.println("CNPJ: " + pessoa.getCnpj());  
    System.out.println();  
}  
}
```



Estácio

Output - CadastroBD (run) ×



run:

Selecione a opção:

1 - Incluir

2 - Alterar

3 - Excluir

4 - Exibir pelo ID

5 - Exibir todos

0 - Finalizar

4

Selecione o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):

1

Digite o ID da pessoa física a ser exibida:

719

Pessoa física encontrada:

Id: 719

Nome: Eduardo

Logradouro: Gomes

Cidade: Olinda

Estado: PE

Telefone: 1212-1212

E-mail: eeadomingues@gmail.com

CPF: 04444444422



Estácio

Output - CadastroBD (run) x



```
Estado: PE
Telefone: 1212-1212
E-mail: eeadomingues@gmail.com
CPF: 044444444422

Selecione a opção:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
0 - Finalizar
4
Selecione o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):
2
Digite o ID da pessoa jurídica a ser exibida:
711
Pessoa jurídica encontrada:
Id: 711
Nome: ED Studio
Logradouro: Nereu
Cidade: Recife
Estado: PE
Telefone: 5454-5454
E-mail: edstudio@edstudio.com
CNPJ: 44444444442

Selecione a opção:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
0 - Finalizar
0
Finalizando...
BUILD SUCCESSFUL (total time: 37 seconds)
```



Estácio

```
Output - CadastroBD (run) x
run:
Selecione a opção:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
0 - Finalizar
5
Selecione o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):
1
Pessoas físicas encontradas:
Id: 719
Nome: Eduardo
Logradouro: Gomes
Cidade: Olinda
Estado: PE
Telefone: 1212-1212
E-mail: eeadomingues@gmail.com
CPF: 04444444422
```

```
Selecione a opção:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
0 - Finalizar
3
Selecione o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):
1
Digite o ID da pessoa física a ser excluída:
719
Pessoa física excluída com sucesso.
Selecione a opção:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
0 - Finalizar
5
Selecione o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):
1
Nenhuma pessoa física encontrada.
Selecione a opção:
```




Estácio

```
Output - CadastroBD (run) x
5 - Exibir todos
0 - Finalizar
1
Selecione o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):
1
Digite o nome:
Eduardo
Digite o logradouro:
Nereu
Digite a cidade:
Sao Paulo
Digite o estado:
SP
Digite o telefone:
3434-3434
Digite o email:
edomingues@gmail.com
Digite o CPF:
5555555525
Pessoa física incluída com sucesso.
Selecione a opção:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
0 - Finalizar
4
Selecione o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):
1
Digite o ID da pessoa física a ser exibida:
727
Pessoa física encontrada:
Id: 727
Nome: Eduardo
Logradouro: Nereu
Cidade: Sao Paulo
Estado: SP
Telefone: 3434-3434
E-mail: edomingues@gmail.com
CPF: 5555555525
```



Conclusão:

- a) A persistência em arquivo armazena dados em formato de arquivo no sistema de arquivos, enquanto a persistência em banco de dados armazena dados em tabelas estruturadas, proporcionando maior capacidade de consulta e gerenciamento.
- b) O uso de operadores lambda no Java simplifica a sintaxe para expressar operações em coleções de dados, proporcionando uma abordagem mais concisa e legível para a iteração e processamento de elementos.
- c) Métodos acionados diretamente pelo método main precisam ser marcados como static porque o método main é estático e pode ser chamado sem instanciar a classe. Métodos estáticos pertencem à classe, não a instâncias específicas, e podem ser chamados sem criar objetos da classe.

Observe que os tópicos acima seguem exatamente o que está na Atividade Prática exigida.