Missão Prática | Nível 3 | Mundo 3

1º Procedimento | Mapeamento Objeto-Relacional e DAO

Objetivos da prática

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

Segue abaixo os códigos de cada classe java:

Classe Pessoa.java

```
package cadastrobd.model;

public class Pessoa {
    protected int id;
    protected String nome;
    protected String endereco;
    protected String telefone;
    protected String email;
    protected char tipo; // 'F' ou 'J'
    protected String cpf;
    protected String cnpj;

    public Pessoa() {
    }

    public Pessoa(int id, String nome, String endereco, String telefone, String email, char tipo, String
cpf, String cnpj) {
        this.id = id;
        this.nome = nome;
        this.endereco = endereco;
        this.telefone = telefone;
        this.email = email;
        this.tipo = tipo;
        this.cpf = cpf;
        this.cnpj = cnpj;
    }

    // Getters e Setters
    public int getId() {
        return id;
    }

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

    public String getNome() {
        return nome;
    }

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

    public String getEndereco() {
        return endereco;
    }

    public void setEndereco(String endereco) {
        this.endereco = endereco;
```

```
}

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;
}

public char getTipo() {
    return tipo;
}

public void setTipo(char tipo) {
    this.tipo = tipo;
}

public String getCpf() {
    return cpf;
}

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

public String getCnpj() {
    return cnpj;
}

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

public void exibir() {
    System.out.println("ID: " + id);
    System.out.println("Nome: " + nome);
    System.out.println("Endereco: " + endereco);
    System.out.println("Telefone: " + telefone);
    System.out.println("Email: " + email);
    System.out.println("Tipo: " + tipo);
    if (tipo == 'F') {
        System.out.println("CPF: " + cpf);
    } else if (tipo == 'J') {
```

```
            System.out.println("CNPJ: " + cnpj);
        }
    }
}
```

Classe PessoaFisica.java

```
package cadastrobd.model;

public class PessoaFisica extends Pessoa {

    public PessoaFisica() {
        this.tipo = 'F';
    }

    public PessoaFisica(int id, String nome, String endereco, String telefone, String email, String cpf) {
        super(id, nome, endereco, telefone, email, 'F', cpf, null);
    }

    @Override
    public void exibir() {
        super.exibir(); // já está exibindo o CPF com base no tipo
    }
}
```

Classe PessoaJuridica.java

```
package cadastrobd.model;

public class PessoaJuridica extends Pessoa {

    public PessoaJuridica() {
        this.tipo = 'J';
    }

    public PessoaJuridica(int id, String nome, String endereco, String telefone, String email, String
cnpj) {
        super(id, nome, endereco, telefone, email, 'J', null, cnpj);
    }

    @Override
    public void exibir() {
        super.exibir(); // já está exibindo o CNPJ com base no tipo
    }
}
```

Classe PessoaFisicaDAO.java

```
package cadastrobd.model;

import cadastrobd.model.util.ConectorBD;
import cadastrobd.model.util.SequenceManager;

import java.sql.*;
import java.util.ArrayList;

public class PessoaFisicaDAO {

    public Pessoa getPessoa(int id) {
        Pessoa pessoa = null;
        try {
            Connection con = ConectorBD.getConnection();
            String sql = "SELECT * FROM Pessoa WHERE id_pessoa = ? AND tipo = 'F'";
            PreparedStatement ps = ConectorBD.getPrepared(con, sql);
            ps.setInt(1, id);
            ResultSet rs = ps.executeQuery();

            if (rs.next()) {
                pessoa = new PessoaFisica(
                    rs.getInt("id_pessoa"),
                    rs.getString("nome"),
                    rs.getString("endereco"),
                    rs.getString("telefone"),
                    rs.getString("email"),
                    rs.getString("cpf")
                );
            }

            ConectorBD.close(rs);
            ConectorBD.close(ps);
            ConectorBD.close(con);

        } catch (SQLException e) {
            System.out.println("Erro ao buscar pessoa física: " + e.getMessage());
        }
        return pessoa;
    }

    public ArrayList<Pessoa> getPessoas() {
        ArrayList<Pessoa> lista = new ArrayList<>();
        try {
            Connection con = ConectorBD.getConnection();
            String sql = "SELECT * FROM Pessoa WHERE tipo = 'F'";
            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery(sql);

            while (rs.next()) {
                PessoaFisica pf = new PessoaFisica(
```

```
                rs.getInt("id_pessoa"),
                rs.getString("nome"),
                rs.getString("endereco"),
                rs.getString("telefone"),
                rs.getString("email"),
                rs.getString("cpf")
            );
            lista.add(pf);
        }

        ConectorBD.close(rs);
        ConectorBD.close(st);
        ConectorBD.close(con);

    } catch (SQLException e) {
        System.out.println("Erro ao listar pessoas físicas: " + e.getMessage());
    }
    return lista;
}

public void incluir(Pessoa p) {
    try {
        PessoaFisica pf = (PessoaFisica) p;
        Connection con = ConectorBD.getConnection();
        String sql = "INSERT INTO Pessoa (id_pessoa, nome, endereco, telefone, email, tipo, cpf,
cnpj) VALUES (?, ?, ?, ?, ?, 'F', ?, NULL)";
        PreparedStatement ps = ConectorBD.getPrepared(con, sql);
        int id = SequenceManager.getValue("seq_id_pessoa");
        ps.setInt(1, id);
        ps.setString(2, pf.getNome());
        ps.setString(3, pf.getEndereco());
        ps.setString(4, pf.getTelefone());
        ps.setString(5, pf.getEmail());
        ps.setString(6, pf.getCpf());
        ps.executeUpdate();

        ConectorBD.close(ps);
        ConectorBD.close(con);

        pf.setId(id);

    } catch (SQLException e) {
        System.out.println("Erro ao incluir pessoa física: " + e.getMessage());
    }
}

public void alterar(Pessoa p) {
    try {
        PessoaFisica pf = (PessoaFisica) p;
        Connection con = ConectorBD.getConnection();
        String sql = "UPDATE Pessoa SET nome=?, endereco=?, telefone=?, email=?, cpf=? WHERE
id_pessoa=? AND tipo='F'";
```

```
            PreparedStatement ps = ConectorBD.getPrepared(con, sql);
            ps.setString(1, pf.getNome());
            ps.setString(2, pf.getEndereco());
            ps.setString(3, pf.getTelefone());
            ps.setString(4, pf.getEmail());
            ps.setString(5, pf.getCpf());
            ps.setInt(6, pf.getId());
            ps.executeUpdate();

            ConectorBD.close(ps);
            ConectorBD.close(con);

        } catch (SQLException e) {
            System.out.println("Erro ao alterar pessoa física: " + e.getMessage());
        }
    }

    public void excluir(int id) {
        try {
            Connection con = ConectorBD.getConnection();
            String sql = "DELETE FROM Pessoa WHERE id_pessoa = ? AND tipo = 'F'";
            PreparedStatement ps = ConectorBD.getPrepared(con, sql);
            ps.setInt(1, id);
            ps.executeUpdate();

            ConectorBD.close(ps);
            ConectorBD.close(con);

        } catch (SQLException e) {
            System.out.println("Erro ao excluir pessoa física: " + e.getMessage());
        }
    }
}
```

Classe PessoaJuridicaDAO.java

```
package cadastrobd.model;

import cadastrobd.model.util.ConectorBD;
import cadastrobd.model.util.SequenceManager;

import java.sql.*;
import java.util.ArrayList;

public class PessoaJuridicaDAO {

    public Pessoa getPessoa(int id) {
        PessoaJuridica pj = null;
        Connection con = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;

        try {
            con = ConectorBD.getConnection();
            stmt = con.prepareStatement("SELECT * FROM Pessoa WHERE id_pessoa = ? AND tipo =
'J'");
            stmt.setInt(1, id);
            rs = stmt.executeQuery();

            if (rs.next()) {
                pj = new PessoaJuridica(
                    rs.getInt("id_pessoa"),
                    rs.getString("nome"),
                    rs.getString("endereco"),
                    rs.getString("telefone"),
                    rs.getString("email"),
                    rs.getString("cnpj")
                );
            }

        } catch (SQLException e) {
            System.out.println("Erro ao buscar pessoa jurÃdica: " + e.getMessage());
        } finally {
            ConectorBD.close(rs);
            ConectorBD.close(stmt);
            ConectorBD.close(con);
        }

        return pj;
    }

    public ArrayList<Pessoa> getPessoas() {
        ArrayList<Pessoa> lista = new ArrayList<>();
        Connection con = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
```

```
    try {
        con = ConectorBD.getConnection();
        stmt = con.prepareStatement("SELECT * FROM Pessoa WHERE tipo = 'J'");
        rs = stmt.executeQuery();

        while (rs.next()) {
            PessoaJuridica pj = new PessoaJuridica(
                rs.getInt("id_pessoa"),
                rs.getString("nome"),
                rs.getString("endereco"),
                rs.getString("telefone"),
                rs.getString("email"),
                rs.getString("cnpj")
            );
            lista.add(pj);
        }

    } catch (SQLException e) {
        System.out.println("Erro ao listar pessoas jurÃdicas: " + e.getMessage());
    } finally {
        ConectorBD.close(rs);
        ConectorBD.close(stmt);
        ConectorBD.close(con);
    }

    return lista;
}

public void incluir(Pessoa p) {
    PessoaJuridica pj = (PessoaJuridica) p;
    Connection con = null;
    PreparedStatement stmt = null;

    try {
        int id = SequenceManager.getValue("seq_id_pessoa");
        con = ConectorBD.getConnection();

        stmt = con.prepareStatement(
            "INSERT INTO Pessoa (id_pessoa, nome, endereco, telefone, email, tipo, cpf, cnpj) " +
            "VALUES (?, ?, ?, ?, ?, 'J', NULL, ?)"
        );
        stmt.setInt(1, id);
        stmt.setString(2, pj.getNome());
        stmt.setString(3, pj.getEndereco());
        stmt.setString(4, pj.getTelefone());
        stmt.setString(5, pj.getEmail());
        stmt.setString(6, pj.getCnpj());
        stmt.executeUpdate();

        pj.setId(id);
```

```
        } catch (SQLException e) {
            System.out.println("Erro ao incluir pessoa jurÃdica: " + e.getMessage());
        } finally {
            ConectorBD.close(stmt);
            ConectorBD.close(con);
        }
    }

    public void alterar(Pessoa p) {
        PessoaJuridica pj = (PessoaJuridica) p;
        Connection con = null;
        PreparedStatement stmt = null;

        try {
            con = ConectorBD.getConnection();

            stmt = con.prepareStatement(
                "UPDATE Pessoa SET nome = ?, endereco = ?, telefone = ?, email = ?, cnpj = ? " +
                "WHERE id_pessoa = ? AND tipo = 'J'"
            );
            stmt.setString(1, pj.getNome());
            stmt.setString(2, pj.getEndereco());
            stmt.setString(3, pj.getTelefone());
            stmt.setString(4, pj.getEmail());
            stmt.setString(5, pj.getCnpj());
            stmt.setInt(6, pj.getId());
            stmt.executeUpdate();

        } catch (SQLException e) {
            System.out.println("Erro ao alterar pessoa jurÃdica: " + e.getMessage());
        } finally {
            ConectorBD.close(stmt);
            ConectorBD.close(con);
        }
    }

    public void excluir(int id) {
        Connection con = null;
        PreparedStatement stmt = null;

        try {
            con = ConectorBD.getConnection();

            stmt = con.prepareStatement("DELETE FROM Pessoa WHERE id_pessoa = ? AND tipo =
'J'");
            stmt.setInt(1, id);
            stmt.executeUpdate();

        } catch (SQLException e) {
            System.out.println("Erro ao excluir pessoa jurÃdica: " + e.getMessage());
        } finally {
            ConectorBD.close(stmt);
```

```
            ConectorBD.close(con);
        }
    }
}
```

Classe ConectorBD.java

```
package cadastrobd.model.util;

import java.sql.*;

public class ConectorBD {

    private static final String URL =
"jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=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);
    }

    // Método para obter PreparedStatement a partir de uma conexão
    public static PreparedStatement getPrepared(Connection con, String sql) throws SQLException {
        return con.prepareStatement(sql);
    }

    // Método adicional para uso rápido (abre uma conexão interna)
    public static PreparedStatement getPrepared(String sql) throws SQLException {
        return getConnection().prepareStatement(sql);
    }

    // Método para executar SELECT diretamente (abre conexão internamente)
    public static ResultSet getSelect(String sql) throws SQLException {
        PreparedStatement ps = getPrepared(sql);
        return ps.executeQuery();
    }

    // Métodos de fechamento seguros

    public static void close(Connection conn) {
        try {
            if (conn != null && !conn.isClosed()) {
                conn.close();
            }
        } catch (SQLException e) {
            System.out.println("Erro ao fechar Connection: " + e.getMessage());
        }
    }

    public static void close(PreparedStatement ps) {
        try {
            if (ps != null && !ps.isClosed()) {
                ps.close();
            }
        } catch (SQLException e) {
            System.out.println("Erro ao fechar PreparedStatement: " + e.getMessage());
```

```
        }
    }

    public static void close(ResultSet rs) {
        try {
            if (rs != null && !rs.isClosed()) {
                rs.close();
            }
        } catch (SQLException e) {
            System.out.println("Erro ao fechar ResultSet: " + e.getMessage());
        }
    }

    public static void close(Statement st) {
        try {
            if (st != null && !st.isClosed()) {
                st.close();
            }
        } catch (SQLException e) {
            System.out.println("Erro ao fechar Statement: " + e.getMessage());
        }
    }
}
```

SequenceManager.java

```
package cadastrobd.model.util;

import java.sql.*;

public class SequenceManager {

    public static int getValue(String sequenceName) {
        int valor = -1;
        Connection con = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;

        try {
            con = ConectorBD.getConnection();
            stmt = con.prepareStatement("SELECT NEXT VALUE FOR " + sequenceName);
            rs = stmt.executeQuery();
            if (rs.next()) {
                valor = rs.getInt(1);
            }
        } catch (SQLException e) {
            System.out.println("Erro ao obter valor da sequência: " + e.getMessage());
        } finally {
            ConectorBD.close(rs);
            ConectorBD.close(stmt);
            ConectorBD.close(con);
        }

        return valor;
    }
}
```

CadastroBDTeste.java

```
package cadastrobd;

import cadastrobd.model.Pessoa;
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridica;
import cadastrobd.model.PessoaJuridicaDAO;

import java.util.ArrayList;

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

        // ============================
        // Pessoa Física
        // ============================
        PessoaFisicaDAO pfDAO = new PessoaFisicaDAO();

        // Incluir
        PessoaFisica pf = new PessoaFisica();
        pf.setNome("Maria Silva");
        pf.setEndereco("Rua das Flores, 123");
        pf.setTelefone("(11) 99999-0000");
        pf.setEmail("maria@teste.com");
        pf.setCpf("123.456.789-00");

        pfDAO.incluir(pf);
        System.out.println("Pessoa Fisica incluida:");
        pf.exibir();

        // Alterar
        pf.setTelefone("(11) 90000-1111");
        pf.setEmail("maria.silva@novoemail.com");
        pfDAO.alterar(pf);
        System.out.println("\nPessoa Fisica alterada:");
        pf.exibir();

        // Consultar todas
        System.out.println("\nLista de Pessoas Fisicas:");
        ArrayList<Pessoa> listaPF = pfDAO.getPessoas();
        for (Pessoa pessoa : listaPF) {
            // Cast para PessoaFisica para chamar exibir()
            ((PessoaFisica) pessoa).exibir();
        }

        // Excluir
        pfDAO.excluir(pf.getId());
        System.out.println("\nPessoa Fisica excluida: ID " + pf.getId());

        // ============================
```

```
        // Pessoa Jurídica
        // ============================
        PessoaJuridicaDAO pjDAO = new PessoaJuridicaDAO();

        // Incluir
        PessoaJuridica pj = new PessoaJuridica();
        pj.setNome("Empresa Exemplo LTDA");
        pj.setEndereco("Avenida Central, 500");
        pj.setTelefone("(21) 88888-0000");
        pj.setEmail("contato@empresa.com.br");
        pj.setCnpj("12.345.678/0001-99");

        pjDAO.incluir(pj);
        System.out.println("\nPessoa Juridica incluida:");
        pj.exibir();

        // Alterar
        pj.setTelefone("(21) 91111-2222");
        pj.setEmail("novoemail@empresa.com.br");
        pjDAO.alterar(pj);
        System.out.println("\nPessoa Juridica alterada:");
        pj.exibir();

        // Consultar todas
        System.out.println("\nLista de Pessoas Juridicas:");
        ArrayList<Pessoa> listaPJ = pjDAO.getPessoas();
        for (Pessoa pessoa : listaPJ) {
            // Cast para PessoaJuridica para chamar exibir()
            ((PessoaJuridica) pessoa).exibir();
        }

        // Excluir
        pjDAO.excluir(pj.getId());
        System.out.println("\nPessoa Juridica excluida: ID " + pj.getId());
    }
}
```

Resultado da execução da classe CadastroBDTeste.java

Pessoa Fisica incluida:
ID: 29
Nome: Maria Silva
Endereco: Rua das Flores, 123
Telefone: (11) 99999-0000
Email: maria@teste.com
Tipo: F
CPF: 123.456.789-00

Pessoa Fisica alterada:
ID: 29
Nome: Maria Silva
Endereco: Rua das Flores, 123
Telefone: (11) 90000-1111
Email: maria.silva@novoemail.com
Tipo: F
CPF: 123.456.789-00

Lista de Pessoas Fisicas:
ID: 1
Nome: Maria Silva
Endereco: Rua A, 123
Telefone: 11999999999
Email: maria@email.com
Tipo: F
CPF: 123.456.789-00
ID: 2
Nome: Joao Souza
Endereco: Rua B, 456
Telefone: 11888888888
Email: joao@email.com
Tipo: F
CPF: 987.654.321-00
ID: 3
Nome: Ana Lima
Endereco: Rua C, 789
Telefone: 11777777777
Email: ana@email.com
Tipo: F
CPF: 111.222.333-44
ID: 4
Nome: Ana Carolina
Endereco: Rua D, 123
Telefone: 11666666666
Email: ana@email.com
Tipo: F
CPF: 333.555.666-55
ID: 7
Nome: Maria Silva
Endereco: Rua A, 123

Telefone: 11999999999
Email: maria@email.com
Tipo: F
CPF: 123.456.789-00
ID: 8
Nome: Joao Souza
Endereco: Rua B, 456
Telefone: 11888888888
Email: joao@email.com
Tipo: F
CPF: 987.654.321-00
ID: 9
Nome: Ana Lima
Endereco: Rua C, 789
Telefone: 11777777777
Email: ana@email.com
Tipo: F
CPF: 111.222.333-44
ID: 10
Nome: Ana Carolina
Endereco: Rua D, 123
Telefone: 11666666666
Email: ana@email.com
Tipo: F
CPF: 333.555.666-55
ID: 29
Nome: Maria Silva
Endereco: Rua das Flores, 123
Telefone: (11) 90000-1111
Email: maria.silva@novoemail.com
Tipo: F
CPF: 123.456.789-00

Pessoa Fisica excluida: ID 29

Pessoa Juridica incluida:
ID: 30
Nome: Empresa Exemplo LTDA
Endereco: Avenida Central, 500
Telefone: (21) 88888-0000
Email: contato@empresa.com.br
Tipo: J
CNPJ: 12.345.678/0001-99

Pessoa Juridica alterada:
ID: 30
Nome: Empresa Exemplo LTDA
Endereco: Avenida Central, 500
Telefone: (21) 91111-2222
Email: novoemail@empresa.com.br
Tipo: J
CNPJ: 12.345.678/0001-99

Lista de Pessoas Juridicas:
ID: 5
Nome: Empresa X
Endereco: Av. Central, 1000
Telefone: 1133333333
Email: contato@empresax.com
Tipo: J
CNPJ: 12.345.678/0001-99
ID: 6
Nome: Empresa Y
Endereco: Av. Paulista, 2000
Telefone: 1144444444
Email: vendas@empresay.com
Tipo: J
CNPJ: 98.765.432/0001-88
ID: 11
Nome: Empresa X
Endereco: Av. Central, 1000
Telefone: 1133333333
Email: contato@empresax.com
Tipo: J
CNPJ: 12.345.678/0001-99
ID: 12
Nome: Empresa Y
Endereco: Av. Paulista, 2000
Telefone: 1144444444
Email: vendas@empresay.com
Tipo: J
CNPJ: 98.765.432/0001-88
ID: 30
Nome: Empresa Exemplo LTDA
Endereco: Avenida Central, 500
Telefone: (21) 91111-2222
Email: novoemail@empresa.com.br
Tipo: J
CNPJ: 12.345.678/0001-99

Pessoa Juridica excluida: ID 30
BUILD SUCCESSFUL (total time: 0 seconds)

Análise e Conclusão

1. Qual a importância dos componentes de middleware, como o JDBC?
O JDBC é fundamental para garantir portabilidade, flexibilidade e facilidade no desenvolvimento de aplicações que interagem com bancos de dados. Ele permite que desenvolvedores usem uma interface única para acessar diferentes sistemas gerenciadores de banco de dados, promovendo interoperabilidade e reduzindo o esforço necessário para lidar com detalhes de baixo nível do protocolo de comunicação.

2. Qual a diferença no uso de Statement ou PreparedStatement para a manipulação de dados?
O uso de `PreparedStatement` é geralmente preferível, pois melhora a eficiência do banco de dados, permite o reuso de comandos SQL, e aumenta a segurança da aplicação ao prevenir injeção de SQL. O `Statement` é indicado apenas para comandos simples, executados uma única vez, sem parâmetros.

3. Como o padrão DAO melhora a manutenibilidade do software?
Ao separar a camada de persistência, o padrão DAO promove baixo acoplamento e alta coesão, facilitando manutenção, testes, e evolução do software. Mudanças na forma de acesso a dados ou troca de banco exigem alterações restritas apenas no DAO, sem impactar outras partes do sistema.

4. Como a herança é refletida no banco de dados, quando lidamos com um modelo estritamente relacional?
A herança em bancos relacionais é simulada por meio do design cuidadoso das tabelas e relações, mas isso pode aumentar a complexidade das consultas e da manutenção. É fundamental escolher a estratégia adequada ao contexto para equilibrar desempenho e simplicidade.