



Missão Prática – Nível 3

Gilvan Pereira de Oliveira – 2023.01.53256-6

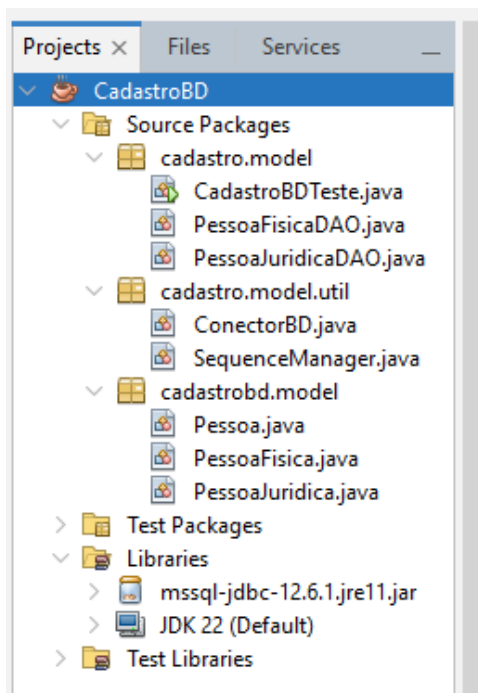
1197 – POLO CENTRO – SÃO LOURENÇO DA MATA - PE
RPG0016 - BackEnd sem banco não tem – 9001 – 2024.1

Objetivo da Prática

Exercício prático que visa implementar persistência de dados em JAVA usando o middleware JDBC, utilizando o padrão DAO (Data Access Object) para manipulação de dados e implementar o mapeamento objeto-relacional, criando um sistema cadastral para armazenar dados utilizando o SQL server como banco de dados.

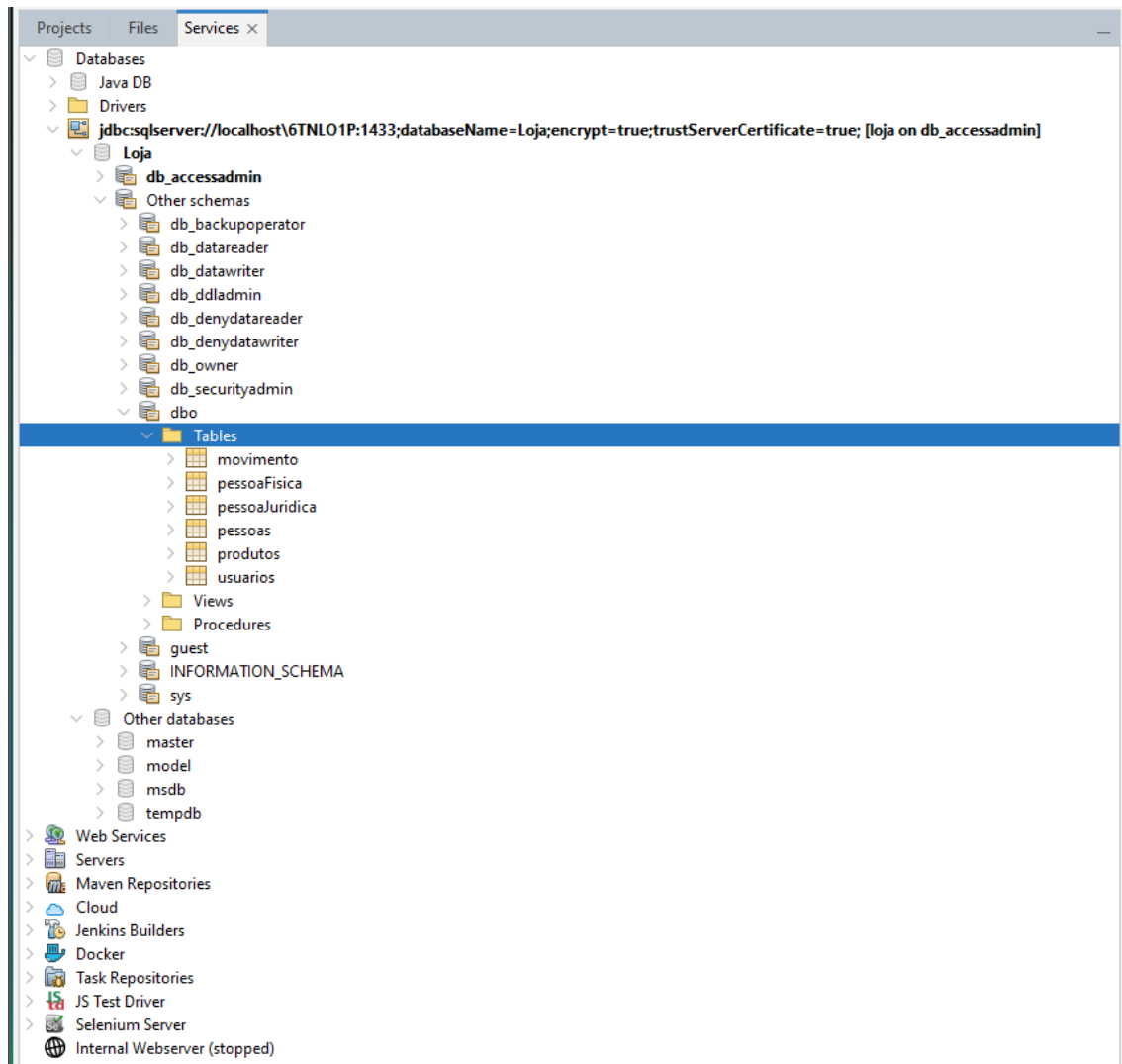
1º Procedimento | Mapeamento Objeto-Relacional e DAO

Projeto criado:



Conexão feita com o banco de dados criado no SQL management, baseado na missão prática anterior:

jdbc:sqlserver://localhost\6TNLO1P:1433;databaseName=Loja;encrypt=true;trustServerCertificate=true;



Tabelas acessadas e consultadas:

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Services' tab is active, displaying a tree view of databases and tables. The 'Loja' database is expanded, showing various schemas and tables. The 'movimento' table is selected, and a context menu is open with options like 'View Data...', 'Execute Command...', 'Add Column...', 'Refresh', 'Delete', 'Grab Structure...', 'Recreate Table...', and 'Properties'. The main window displays a query: `SELECT TOP 100 * FROM dbo.movimento;`. Below the query, the results are shown in a table with columns: idMovimento, idUsuario, idPessoa, idProduto, quantidade, and status. The output window at the bottom shows the execution details: `[1:1] Executed successfully in 0,012 s. Fetching resultset took 0,001 s. Execution finished after 0,35 s, no errors occurred.`

idMovimento	idUsuario	idPessoa	idProduto	quantidade	status
2	1	1	1	20	S
3	1	1	3	15	S
4	2	2	3	10	S
5	1	2	3	15	E
6	1	2	4	20	E

Agora, dando início ao projeto da missão prática atual:

Criando o pacote cadastrobd.model com as classes:

Classe Pessoa:

```
package cadastrobd.model;
```

```
public class Pessoa {
```

```
    private int id;
```

```
    private String nome;
```

```
    private String logradouro;
```

```
    private String cidade;
```

```
    private String estado;
```

```
    private String telefone;
```

```
private String email;
```

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

```
public Pessoa(int id, String nome, String logradouro, String cidade, String estado,  
String telefone, String email) {  
    this.id = id;  
    this.nome = nome;  
    this.logradouro = logradouro;  
    this.cidade = cidade;  
    this.estado = estado;  
    this.telefone = telefone;  
    this.email = email;  
}
```

```
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 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;  
}
```

```
public void exibir() {  
    System.out.println("ID: " + id);  
    System.out.println("Nome: " + nome);  
    System.out.println("Logradouro: " + logradouro);  
    System.out.println("Cidade: " + cidade);  
    System.out.println("Estado: " + estado);  
}
```

```
        System.out.println("Telefone: " + telefone);

        System.out.println("E-mail: " + email);
    }
}
```

Classe PessoaFisica:

```
package cadastrabd.model;

import cadastrabd.model.Pessoa;

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 String getCpf() {

        return cpf;
    }

    public void setCpf(String cpf) {
```

```
        this.cpf = cpf;
    }
}
```

@Override

```
public String toString() {
    return "Id: " + id + "\n" +
        "Nome: " + nome + "\n" +
        "Logradouro: " + logradouro + "\n" +
        "Cidade: " + cidade + "\n" +
        "Estado: " + estado + "\n" +
        "Telefone: " + telefone + "\n" +
        "E-mail: " + email + "\n" +
        "CPF: " + cpf + "\n";
}
}
```

Classe PessoaJuridica:

```
package cadastrbd.model;

public class PessoaJuridica extends Pessoa {
    private String cnpj;

    public PessoaJuridica() {
        super();
    }
}
```



```
public PessoaJuridica(int id, String nome, String logradouro, String cidade, String
estado, String telefone, String email, String cnpj) {

    super(id, nome, logradouro, cidade, estado, telefone, email);

    this.cnpj = cnpj;
}
```

```
public String getCnpj() {

    return cnpj;
}
```

```
public void setCnpj(String cnpj) {

    this.cnpj = cnpj;
}
```

@Override

```
public String toString() {

    return "Id: " + id + "\n" +

        "Nome: " + nome + "\n" +

        "Logradouro: " + logradouro + "\n" +

        "Cidade: " + cidade + "\n" +

        "Estado: " + estado + "\n" +

        "Telefone: " + telefone + "\n" +

        "E-mail: " + email + "\n" +

        "CNPJ: " + cnpj + "\n";

}

}
```

Criando o pacote cadastro.model.util com as classes:

Classe ConectorBD:

```
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\\6TNLO1P:1433;databaseName=Loja;encrypt=true;trustServ
erCertificate=true;";

    private static final String USER = "loja";

    private static final String PASSWORD = "cadastrobd";

    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) {  
    if (stmt != null) {  
        try {  
            stmt.close();  
        } catch (SQLException e) {  
        }  
    }  
}
```

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

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

```

        } catch (SQLException e) {

        }

    }

}

}

```

Classe SequenceManager:

```
package cadastro.model.util;
```

```
import java.sql.Connection;
```

```
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
```

```
import java.sql.Statement;
```

```
public class SequenceManager {
```

```
    public static int getValue(String sequenceName) {
```

```
        int value = 0;
```

```
        Connection conn = null;
```

```
        Statement stmt = null;
```

```
        ResultSet rs = null;
```

```
        try {
```

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

```
            stmt = conn.createStatement();
```

```
            rs = stmt.executeQuery("SELECT NEXT VALUE FOR " + sequenceName);
```

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

```
                value = rs.getInt(1);
```

```

        }
    } catch (SQLException e) {
    } finally {
        ConectorBD.close(rs);
        ConectorBD.close(stmt);
        ConectorBD.close(conn);
    }

    return value;
}
}

```

Criando o pacote cadastro.model com as classes:

Classe PessoaFisicaDAO:

```

package cadastro.model;

import cadastrobd.model.PessoaFisica;
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 stmt = null;

ResultSet rs = null;

try {

    conn = ConectorBD.getConnection();

    stmt = conn.prepareStatement("SELECT * FROM pessoas JOIN pessoaFisica
ON pessoas.idPessoa = pessoaFisica.idPessoa WHERE pessoas.idPessoa = ?");

    stmt.setInt(1, id);

    rs = stmt.executeQuery();

    if (rs.next()) {

        pessoa = new PessoaFisica(

            rs.getInt("idPessoa"),

            rs.getString("nome"),

            rs.getString("logradouro"),

            rs.getString("cidade"),

            rs.getString("estado"),

            rs.getString("telefone"),

            rs.getString("email"),

            rs.getString("cpf")

        );

    }

} catch (SQLException e) {

} finally {

    ConectorBD.close(rs);

```

```

        ConectorBD.close(stmt);

        ConectorBD.close(conn);

    }

    return pessoa;
}

public List<PessoaFisica> getPessoas() {

    List<PessoaFisica> pessoas = new ArrayList<>();

    Connection conn = null;

    PreparedStatement stmt = null;

    ResultSet rs = null;

    try {

        conn = ConectorBD.getConnection();

        stmt = conn.prepareStatement("SELECT * FROM pessoas JOIN pessoaFisica
ON pessoas.idPessoa = pessoaFisica.idPessoa");

        rs = stmt.executeQuery();

        while (rs.next()) {

            PessoaFisica pessoa = new PessoaFisica(

                rs.getInt("idPessoa"),

                rs.getString("nome"),

                rs.getString("logradouro"),

                rs.getString("cidade"),

                rs.getString("estado"),

```

```

        rs.getString("telefone"),
        rs.getString("email"),
        rs.getString("cpf")
    );
    pessoas.add(pessoa);
}
} catch (SQLException e) {
} finally {
    ConectorBD.close(rs);
    ConectorBD.close(stmt);
    ConectorBD.close(conn);
}

return pessoas;
}

```

```

public void incluir(PessoaFisica pessoa) {
    Connection conn = null;
    PreparedStatement stmt = null;

    try {
        conn = ConectorBD.getConnection();

        stmt = conn.prepareStatement("INSERT INTO pessoas (nome, logradouro,
cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?)",
PreparedStatement.RETURN_GENERATED_KEYS);

        stmt.setString(1, pessoa.getNome());
    }
}

```



```

        stmt.setString(2, pessoa.getLogradouro());

        stmt.setString(3, pessoa.getCidade());

        stmt.setString(4, pessoa.getEstado());

        stmt.setString(5, pessoa.getTelefone());

        stmt.setString(6, pessoa.getEmail());

        stmt.executeUpdate();

        ResultSet rs = stmt.getGeneratedKeys();

        if (rs.next()) {

            int idPessoa = rs.getInt(1);

            stmt = conn.prepareStatement("INSERT INTO pessoaFisica (idPessoa, cpf)
VALUES (?, ?)");

            stmt.setInt(1, idPessoa);

            stmt.setString(2, pessoa.getCpf());

            stmt.executeUpdate();

            pessoa.setId(idPessoa);

        }

    } catch (SQLException e) {

    } finally {

        ConectorBD.close(stmt);

        ConectorBD.close(conn);

    }

}

public void alterar(PessoaFisica pessoa) {

    Connection conn = null;

```

```

PreparedStatement stmt = null;

try {

    conn = ConectorBD.getConnection();

    stmt = conn.prepareStatement("UPDATE pessoas SET nome=?, logradouro=?,
cidade=?, estado=?, telefone=?, email=? WHERE idPessoa=?");

    stmt.setString(1, pessoa.getNome());

    stmt.setString(2, pessoa.getLogradouro());

    stmt.setString(3, pessoa.getCidade());

    stmt.setString(4, pessoa.getEstado());

    stmt.setString(5, pessoa.getTelefone());

    stmt.setString(6, pessoa.getEmail());

    stmt.setInt(7, pessoa.getId());

    stmt.executeUpdate();

    stmt = conn.prepareStatement("UPDATE pessoaFisica SET cpf=? WHERE
idPessoa=?");

    stmt.setString(1, pessoa.getCpf());

    stmt.setInt(2, pessoa.getId());

    stmt.executeUpdate();

} catch (SQLException e) {

} finally {

    ConectorBD.close(stmt);

    ConectorBD.close(conn);

}

}

```

```

public void excluir(int id) {

    Connection conn = null;

    PreparedStatement stmt = null;

    try {

        conn = ConectorBD.getConnection();

        stmt = conn.prepareStatement("DELETE FROM pessoaFisica WHERE
idPessoa=?");

        stmt.setInt(1, id);

        stmt.executeUpdate();

        stmt = conn.prepareStatement("DELETE FROM pessoas WHERE
idPessoa=?");

        stmt.setInt(1, id);

        stmt.executeUpdate();

    } catch (SQLException e) {

    } finally {

        ConectorBD.close(stmt);

        ConectorBD.close(conn);

    }

}

```

Classe PessoaJuridicaDAO:

```

package cadastro.model;

import cadastrobd.model.PessoaJuridica;

```

```
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 pessoa = null;

        Connection conn = null;

        PreparedStatement stmt = null;

        ResultSet rs = null;

        try {

            conn = ConectorBD.getConnection();

            stmt = conn.prepareStatement("SELECT * FROM pessoas JOIN pessoaJuridica  
ON pessoas.idPessoa = pessoaJuridica.idPessoa WHERE pessoas.idPessoa = ?");

            stmt.setInt(1, id);

            rs = stmt.executeQuery();

            if (rs.next()) {

                pessoa = new PessoaJuridica(

                    rs.getInt("idPessoa"),

                    rs.getString("nome"),

                    rs.getString("logradouro"),
```

```

        rs.getString("cidade"),

        rs.getString("estado"),

        rs.getString("telefone"),

        rs.getString("email"),

        rs.getString("cnpj")

    );

}

} catch (SQLException e) {

} finally {

    ConectorBD.close(rs);

    ConectorBD.close(stmt);

    ConectorBD.close(conn);

}

return pessoa;

}

public List<PessoaJuridica> getPessoas() {

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

    Connection conn = null;

    PreparedStatement stmt = null;

    ResultSet rs = null;

    try {

        conn = ConectorBD.getConnection();

```

```

        stmt = conn.prepareStatement("SELECT * FROM pessoas JOIN pessoaJuridica
ON pessoas.idPessoa = pessoaJuridica.idPessoa");

        rs = stmt.executeQuery();

        while (rs.next()) {

            PessoaJuridica pessoa = new PessoaJuridica(

                rs.getInt("idPessoa"),

                rs.getString("nome"),

                rs.getString("logradouro"),

                rs.getString("cidade"),

                rs.getString("estado"),

                rs.getString("telefone"),

                rs.getString("email"),

                rs.getString("cnpj")

            );

            pessoas.add(pessoa);

        }

    } catch (SQLException e) {

    } finally {

        ConectorBD.close(rs);

        ConectorBD.close(stmt);

        ConectorBD.close(conn);

    }

    return pessoas;

}

```

```
public void incluir(PessoaJuridica pessoa) {

    Connection conn = null;

    PreparedStatement stmt = null;

    try {

        conn = ConectorBD.getConnection();

        stmt = conn.prepareStatement("INSERT INTO pessoas (nome, logradouro,
cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?)",
PreparedStatement.RETURN_GENERATED_KEYS);

        stmt.setString(1, pessoa.getNome());

        stmt.setString(2, pessoa.getLogradouro());

        stmt.setString(3, pessoa.getCidade());

        stmt.setString(4, pessoa.getEstado());

        stmt.setString(5, pessoa.getTelefone());

        stmt.setString(6, pessoa.getEmail());

        stmt.executeUpdate();

        ResultSet rs = stmt.getGeneratedKeys();

        if (rs.next()) {

            int idPessoa = rs.getInt(1);

            stmt = conn.prepareStatement("INSERT INTO pessoaJuridica (idPessoa,
cnpj) VALUES (?, ?)");

            stmt.setInt(1, idPessoa);

            stmt.setString(2, pessoa.getCnpj());

            stmt.executeUpdate();

        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

```

        pessoa.setId(idPessoa);
    }
} catch (SQLException e) {
} finally {
    ConectorBD.close(stmt);
    ConectorBD.close(conn);
}
}

```

```

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

    try {
        conn = ConectorBD.getConnection();

        stmt = conn.prepareStatement("UPDATE pessoas SET nome=?, logradouro=?,
cidade=?, estado=?, telefone=?, email=? WHERE idPessoa=?");

        stmt.setString(1, pessoa.getNome());
        stmt.setString(2, pessoa.getLogradouro());
        stmt.setString(3, pessoa.getCidade());
        stmt.setString(4, pessoa.getEstado());
        stmt.setString(5, pessoa.getTelefone());
        stmt.setString(6, pessoa.getEmail());
        stmt.setInt(7, pessoa.getId());

        stmt.executeUpdate();
    }
}

```



```

        stmt = conn.prepareStatement("UPDATE pessoaJuridica SET cnpj=? WHERE
idPessoa=?");

        stmt.setString(1, pessoa.getCnpj());

        stmt.setInt(2, pessoa.getId());

        stmt.executeUpdate();
    } catch (SQLException e) {
    } finally {

        ConectorBD.close(stmt);

        ConectorBD.close(conn);

    }
}

```

```

public void excluir(int id) {

    Connection conn = null;

    PreparedStatement stmt = null;

    try {

        conn = ConectorBD.getConnection();

        stmt = conn.prepareStatement("DELETE FROM pessoaJuridica WHERE
idPessoa=?");

        stmt.setInt(1, id);

        stmt.executeUpdate();

        stmt = conn.prepareStatement("DELETE FROM pessoas WHERE
idPessoa=?");

        stmt.setInt(1, id);
    }
}

```

```

        stmt.executeUpdate();
    } catch (SQLException e) {
    } finally {
        ConectorBD.close(stmt);
        ConectorBD.close(conn);
    }
}
}
}

```

Criando a classe de testes CadastroBDTeste:

```
package cadastro.model;
```

```
import cadastrobd.model.PessoaJuridica;
```

```
import cadastrobd.model.PessoaFisica;
```

```

public class CadastroBDTeste {

    public static void main(String[] args) {

        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();

        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();

        // Inclusão de Pessoa Fisica

        PessoaFisica pessoaFisica = new PessoaFisica();

        pessoaFisica.setNome("Joao da Silva");

        pessoaFisica.setLogradouro("Rua 10, casa 6, Quitanda");

        pessoaFisica.setCidade("Riacho do Leste");

        pessoaFisica.setEstado("PA");
    }
}

```

```
    pessoaFisica.setTelefone("3333-3333");  
    pessoaFisica.setEmail("joao.silva@riacho.com");  
    pessoaFisica.setCpf("33333333333");
```

```
    pessoaFisicaDAO.incluir(pessoaFisica);  
    System.out.println("Pessoa física criada:");  
    pessoaFisica.exibir();
```

```
// Inclusão de Pessoa Juridica
```

```
PessoaJuridica pessoaJuridica = new PessoaJuridica();  
pessoaJuridica.setNome("JJCSilva");  
pessoaJuridica.setLogradouro("Rua 13, Centro");  
pessoaJuridica.setCidade("Riacho do Oeste");  
pessoaJuridica.setEstado("PA");  
pessoaJuridica.setTelefone("4444-4444");  
pessoaJuridica.setEmail("jjc.silva@riacho.com");  
pessoaJuridica.setCnpj("4444444444444444");
```

```
    pessoaJuridicaDAO.incluir(pessoaJuridica);  
    System.out.println("Pessoa jurídica criada:");  
    pessoaJuridica.exibir();
```

```
// Consulta todas as pessoas físicas
```

```
System.out.println("\nPessoas Físicas:");  
for (PessoaFisica pf : pessoaFisicaDAO.getPessoas()) {  
    pf.exibir();
```

```
        System.out.println();
    }

    // Consulta todas as pessoas jurídicas
    System.out.println("\nPessoas Juridicas:");
    for (PessoaJuridica pj : pessoaJuridicaDAO.getPessoas()) {
        pj.exibir();
        System.out.println();
    }

    // Exclui a pessoa física e jurídica criadas
    pessoaFisicaDAO.excluir(pessoaFisica.getId());
    pessoaJuridicaDAO.excluir(pessoaJuridica.getId());
}
}
```

Resultado obtido após todo o procedimento:



```
Output - CadastroBD (run) ×

Pessoas Fisicas:
ID: 1
Nome: Joao
Logradouro: Rua 12, casa 3, Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
E-mail: joao@riacho.com
CPF: 1111111111

ID: 11
Nome: Joao da Silva
Logradouro: Rua 10, casa 6, Quitanda
Cidade: Riacho do Leste
Estado: PA
Telefone: 3333-3333
E-mail: joao.silva@riacho.com
CPF: 3333333333

Pessoas Juridicas:
ID: 2
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
E-mail: jjc@riacho.com
CNPJ: 222222222222

ID: 12
Nome: JJCSilva
Logradouro: Rua 13, Centro
Cidade: Riacho do Oeste
Estado: PA
Telefone: 4444-4444
E-mail: jjc.silva@riacho.com
CNPJ: 44444444444444

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

Conclusão:

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

Os componentes, como JDBC, são importantes porque fornecem uma camada de abstração que simplifica a comunicação entre o aplicativo e o banco de dados, promovendo eficiência e segurança no acesso aos dados.

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

Statement: Concatena os parâmetros diretamente na consulta SQL, vulnerável a injeção de SQL e menos eficiente para consultas repetidas.

PreparedStatement: Parametriza parâmetros na consulta SQL, mais seguro contra injeção de SQL e mais eficiente para consultas repetidas.

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

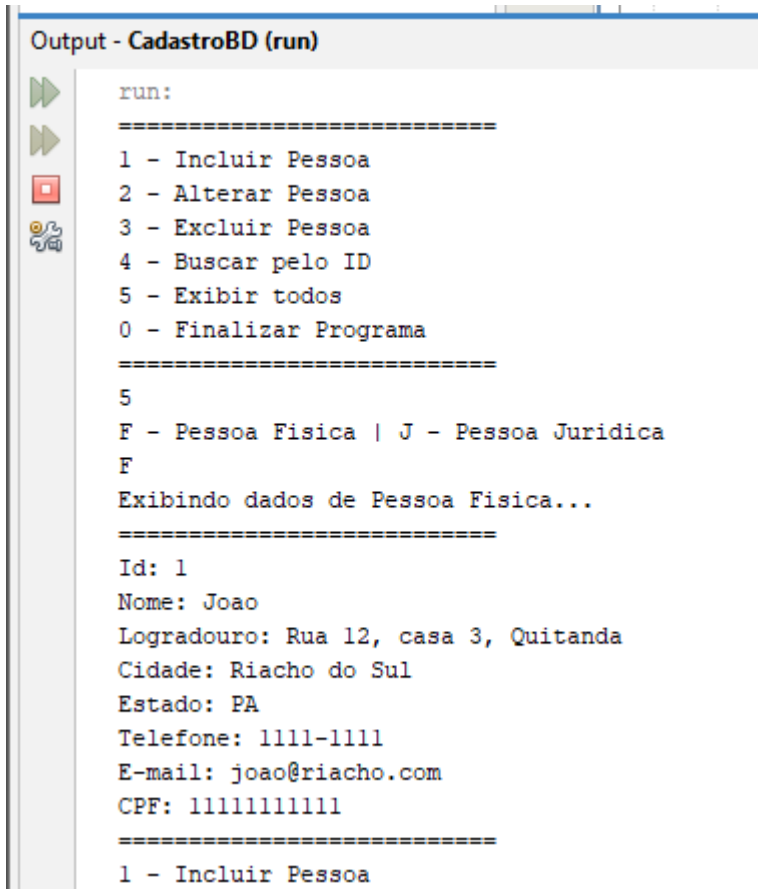
Separando a lógica de acesso a dados da lógica de negócios, facilitando a modificação e a manutenção do código, permitindo a reutilização e mudanças na estrutura do banco de dados com menor impacto.

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

É refletida criando uma tabela para cada classe ou subclasse, ou uma tabela única para toda a hierarquia de herança, dependendo da abordagem escolhida.

2º Procedimento | Alimentando a Base

Alterando o método main da classe principal CadastroBD, Output:



```
Output - CadastroBD (run)

run:
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir todos
0 - Finalizar Programa
=====
5
F - Pessoa Fisica | J - Pessoa Juridica
F
Exibindo dados de Pessoa Fisica...
=====
Id: 1
Nome: Joao
Logradouro: Rua 12, casa 3, Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
E-mail: joao@riacho.com
CPF: 11111111111
=====
1 - Incluir Pessoa
```

Código do segundo procedimento com os métodos incluir, alterar, excluir, exibir e sair:

```
/*
```

```
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
```

```
* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this
template
```

```
*/
```

```
package cadastro.model;
```

```
import cadastrobd.model.PessoaFisica;
```

```
import cadastrobd.model.PessoaJuridica;

import java.util.List;

import java.util.Scanner;

/**
 *
 * @author gilvan
 */

public class CadastroBD {

    public static void main(String[] args) {

        try (Scanner scanner = new Scanner(System.in)) {

            PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();

            PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();

            int opcao;

            do {

                System.out.println("=====");

                System.out.println("1 - Incluir Pessoa");

                System.out.println("2 - Alterar Pessoa");

                System.out.println("3 - Excluir Pessoa");

                System.out.println("4 - Buscar pelo ID");

                System.out.println("5 - Exibir todos");

                System.out.println("0 - Finalizar Programa");

                System.out.println("=====");
```



```

opcao = scanner.nextInt();

switch (opcao) {
    case 1 -> incluir(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
    case 2 -> alterar(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
    case 3 -> excluir(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
    case 4 -> obter(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
    case 5 -> obterTodos(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
    case 0 -> System.out.println("Programa finalizado.");
    default -> System.out.println("Opcao invalida.");
}
} while (opcao != 0);
}
}

```

```

private static void incluir(Scanner scanner, PessoaFisicaDAO pessoaFisicaDAO,
PessoaJuridicaDAO pessoaJuridicaDAO) {

```

```

    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");

```

```

    String tipo = scanner.next();

```

```

    try {

```

```

        switch (tipo.toUpperCase()) {

```

```

            case "F" -> {

```

```

                System.out.println("Cadastro de Pessoa Fisica:");

```

```

                PessoaFisica pessoaFisica = lerDadosPessoaFisica(scanner);

```

```

                pessoaFisicaDAO.incluir(pessoaFisica);
            }
        }
    }
}

```

```

        System.out.println("Pessoa fisica incluida com sucesso...");

        System.out.println("=====");
    }

    case "J" -> {

        System.out.println("Cadastro de Pessoa Juridica:");

        PessoaJuridica pessoaJuridica = lerDadosPessoaJuridica(scanner);

        pessoaJuridicaDAO.incluir(pessoaJuridica);

        System.out.println("Pessoa juridica incluida com sucesso...");

        System.out.println("=====");

    }

    default -> System.out.println("Tipo de pessoa invalido.");

}

} catch (Exception e) {

    System.out.println("Ocorreu um erro ao tentar incluir a pessoa: " +
e.getMessage());

}

}

```

```

private static PessoaFisica lerDadosPessoaFisica(Scanner scanner) {

    PessoaFisica pessoaFisica = new PessoaFisica();

    scanner.nextLine();

    System.out.print("Nome: ");

    pessoaFisica.setNome(scanner.nextLine());

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

    pessoaFisica.setLogradouro(scanner.nextLine());

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

```

```
    pessoaFisica.setCidade(scanner.nextLine());  
  
    System.out.print("Estado: ");  
  
    pessoaFisica.setEstado(scanner.nextLine());  
  
    System.out.print("Telefone: ");  
  
    pessoaFisica.setTelefone(scanner.nextLine());  
  
    System.out.print("Email: ");  
  
    pessoaFisica.setEmail(scanner.nextLine());  
  
    System.out.print("CPF: ");  
  
    pessoaFisica.setCpf(scanner.nextLine());  
  
    return pessoaFisica;  
}
```

```
private static PessoaJuridica lerDadosPessoaJuridica(Scanner scanner) {  
  
    PessoaJuridica pessoaJuridica = new PessoaJuridica();  
  
    scanner.nextLine();  
  
    System.out.print("Nome: ");  
  
    pessoaJuridica.setNome(scanner.nextLine());  
  
    System.out.print("Logradouro: ");  
  
    pessoaJuridica.setLogradouro(scanner.nextLine());  
  
    System.out.print("Cidade: ");  
  
    pessoaJuridica.setCidade(scanner.nextLine());  
  
    System.out.print("Estado: ");  
  
    pessoaJuridica.setEstado(scanner.nextLine());  
  
    System.out.print("Telefone: ");  
  
    pessoaJuridica.setTelefone(scanner.nextLine());  
  
    System.out.print("Email: ");
```

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

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

    pessoaJuridica.setCnpj(scanner.nextLine());

    return pessoaJuridica;
}
```

```
private static void alterar(Scanner scanner, PessoaFisicaDAO pessoaFisicaDAO,
PessoaJuridicaDAO pessoaJuridicaDAO) {
```

```
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
```

```
    String tipo = scanner.next();
```

```
    System.out.print("Digite o ID da pessoa: ");
```

```
    int id = scanner.nextInt();
```

```
    try {
```

```
        switch (tipo.toUpperCase()) {
```

```
            case "F" -> {
```

```
                PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);
```

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

```
                    System.out.println("Dados atuais da Pessoa Fisica:");
```

```
                    System.out.println(pessoaFisica);
```

```
                    PessoaFisica novosDados = lerDadosPessoaFisica(scanner);
```

```
                    novosDados.setId(id);
```

```
                    pessoaFisicaDAO.alterar(novosDados);
```

```
                    System.out.println("Pessoa fisica alterada com sucesso.");
```

```

    } else {

        System.out.println("Pessoa fisica nao encontrada.");

    }

}

case "J" -> {

    PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);

    if (pessoaJuridica != null) {

        System.out.println("Dados atuais da Pessoa Juridica:");

        System.out.println(pessoaJuridica);

        PessoaJuridica novosDados = lerDadosPessoaJuridica(scanner);

        novosDados.setId(id);

        pessoaJuridicaDAO.alterar(novosDados);

        System.out.println("Pessoa juridica alterada com sucesso.");

    } else {

        System.out.println("Pessoa juridica nao encontrada.");

    }

}

default -> System.out.println("Tipo de pessoa invalido.");

}

} catch (Exception e) {

    System.out.println("Ocorreu um erro ao tentar alterar a pessoa: " +
e.getMessage());

}

}

```

```

private static void excluir(Scanner scanner, PessoaFisicaDAO pessoaFisicaDAO,
PessoaJuridicaDAO pessoaJuridicaDAO) {

    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");

    String tipo = scanner.next();

    System.out.print("Digite o ID da pessoa: ");

    int id = scanner.nextInt();

    try {

        switch (tipo.toUpperCase()) {

            case "F" -> {

                pessoaFisicaDAO.excluir(id);

                System.out.println("Pessoa fisica excluida com sucesso.");

            }

            case "J" -> {

                pessoaJuridicaDAO.excluir(id);

                System.out.println("Pessoa juridica excluida com sucesso.");

            }

            default -> System.out.println("Tipo de pessoa invalido.");

        }

    } catch (Exception e) {

        System.out.println("Ocorreu um erro ao tentar excluir a pessoa: " +
e.getMessage());

    }

}

```

```

private static void obter(Scanner scanner, PessoaFisicaDAO pessoaFisicaDAO,
PessoaJuridicaDAO pessoaJuridicaDAO) {

    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");

    String tipo = scanner.next();

    System.out.print("Digite o ID da pessoa: ");

    int id = scanner.nextInt();

    try {

        switch (tipo.toUpperCase()) {

            case "F" -> {

                PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);

                if (pessoaFisica != null) {

                    System.out.println("Pessoa fisica encontrada:");

                    System.out.println(pessoaFisica);

                } else {

                    System.out.println("Pessoa fisica nao encontrada.");

                }

            }

            case "J" -> {

                PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);

                if (pessoaJuridica != null) {

                    System.out.println("Pessoa juridica encontrada:");

                    System.out.println(pessoaJuridica);

                } else {

                    System.out.println("Pessoa juridica nao encontrada.");

                }

            }

        }

    } catch (Exception e) {

        System.out.println("Erro: " + e.getMessage());

    }

}

```

```

        }

    }

    default -> System.out.println("Tipo de pessoa invalido.");

}

} catch (Exception e) {

    System.out.println("Ocorreu um erro ao tentar buscar a pessoa: " +
e.getMessage());

}

}

```

```

private static void obterTodos(Scanner scanner, PessoaFisicaDAO pessoaFisicaDAO,
PessoaJuridicaDAO pessoaJuridicaDAO) {

```

```

    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");

```

```

    String tipo = scanner.next();

```

```

    try {

```

```

        switch (tipo.toUpperCase()) {

```

```

            case "F" -> {

```

```

                List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();

```

```

                System.out.println("Exibindo dados de Pessoa Fisica...");

```

```

                System.out.println("=====");

```

```

                for (PessoaFisica pf : pessoasFisicas) {

```

```

                    System.out.println("Id: " + pf.getId());

```

```

                    System.out.println("Nome: " + pf.getNome());

```

```

                    System.out.println("Logradouro: " + pf.getLogradouro());

```

```

                    System.out.println("Cidade: " + pf.getCidade());

```



```

        System.out.println("Estado: " + pf.getEstado());

        System.out.println("Telefone: " + pf.getTelefone());

        System.out.println("E-mail: " + pf.getEmail());

        System.out.println("CPF: " + pf.getCpf());

    }

}

case "J" -> {

    List<PessoaJuridica> pessoasJuridicas = pessoaJuridicaDAO.getPessoas();

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

    System.out.println("=====");

    for (PessoaJuridica pj : pessoasJuridicas) {

        System.out.println("Id: " + pj.getId());

        System.out.println("Nome: " + pj.getNome());

        System.out.println("Logradouro: " + pj.getLogradouro());

        System.out.println("Cidade: " + pj.getCidade());

        System.out.println("Estado: " + pj.getEstado());

        System.out.println("Telefone: " + pj.getTelefone());

        System.out.println("E-mail: " + pj.getEmail());

        System.out.println("CNPJ: " + pj.getCnpj());

    }

}

default -> System.out.println("Tipo de pessoa invalido.");

}

} catch (Exception e) {

    System.out.println("Ocorreu um erro ao tentar exibir as pessoas: " +
e.getMessage());

```

```
}  
  
}  
  
}
```

Conclusão:

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

A persistência em arquivo ela armazena os dados em arquivos do sistema do computador, enquanto a persistência em banco de dados armazena dados em um sistema de gerenciamento de banco de dados, permitindo consultas mais complexas, relacionamento entre os dados com melhor controle e integridade dos dados.

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

O operador Lambda simplifica a impressão dos valores contidos nas entidades ao reduzir a quantidade de código necessária para iterar sobre uma coleção e realizar uma operação, tornando o código mais conciso e legível.

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

Métodos acionados diretamente pela main, sem uso de um objeto, precisam ser estáticos, para que possam ser chamados sem a necessidade de criar uma instância de classe que os contém. Pois os métodos estáticos podem ser chamados diretamente devido a JVM (Java Virtual Machine), já os métodos não estáticos só podem ser chamados em instâncias de objeto.

Conclusão

A missão prática apresenta uma abordagem detalhada para implementar persistência de dados em um banco de dados SQL criado na missão prática anterior, utilizando JDBC e o padrão DAO para programar em Java sua estrutura. Os procedimentos detalham desde a criação das classes necessárias para o mapeamento objeto-relacional, implementação das operações (criar, alterar, excluir e buscar), e a realização de testes para garantir o aprendizado e o funcionamento correto do sistema. Demonstrando assim não só conceitos teóricos, mas também fornecendo uma experiência prática na construção de uma aplicação em Java interagindo com um banco de dados.