DESENVOLVIMENTO FULL STACK

CAMPOS: polo stiep gil – Salvador – BA

Disciplina: nível 3 - Back-end sem banco não tem
Turma: 9001 Semestre letivo: terceiro
Nome: David Lins do Amaral

Missão prática nível 3

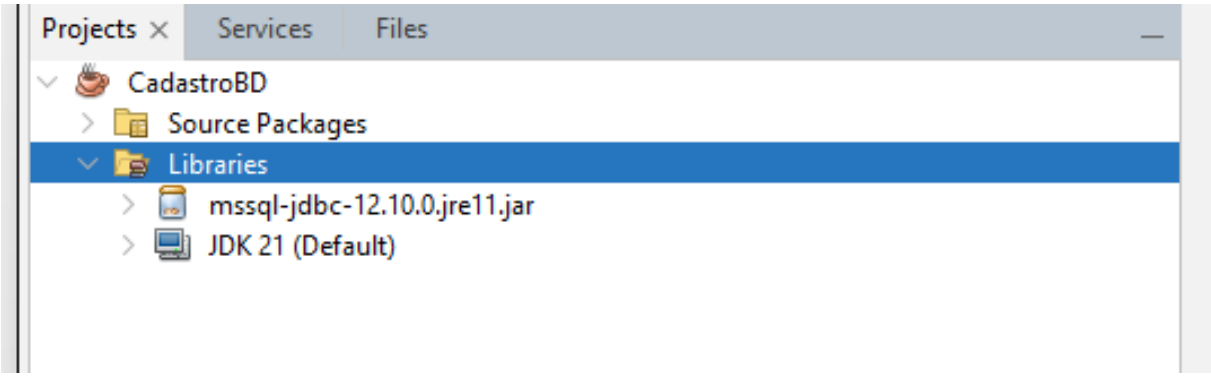
Criação de aplicativo Java, com acesso ao banco de dados SQL Server através do

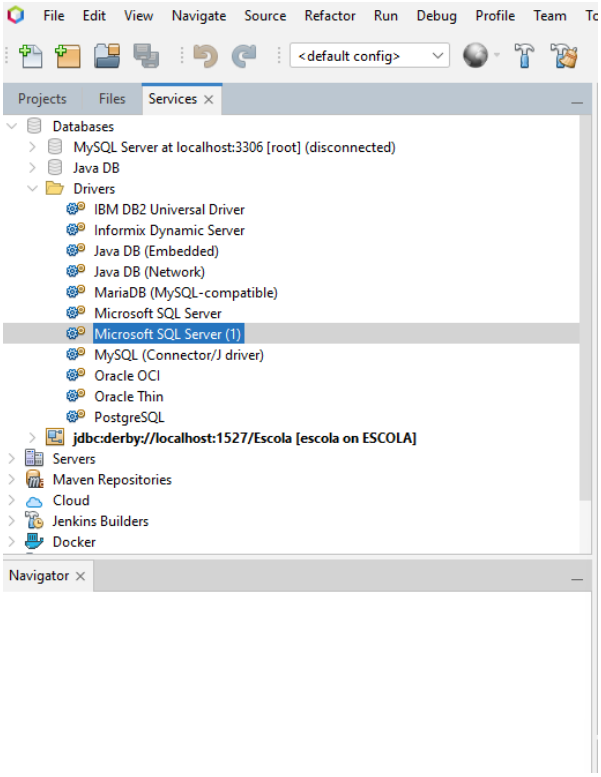
middleware JDBC.

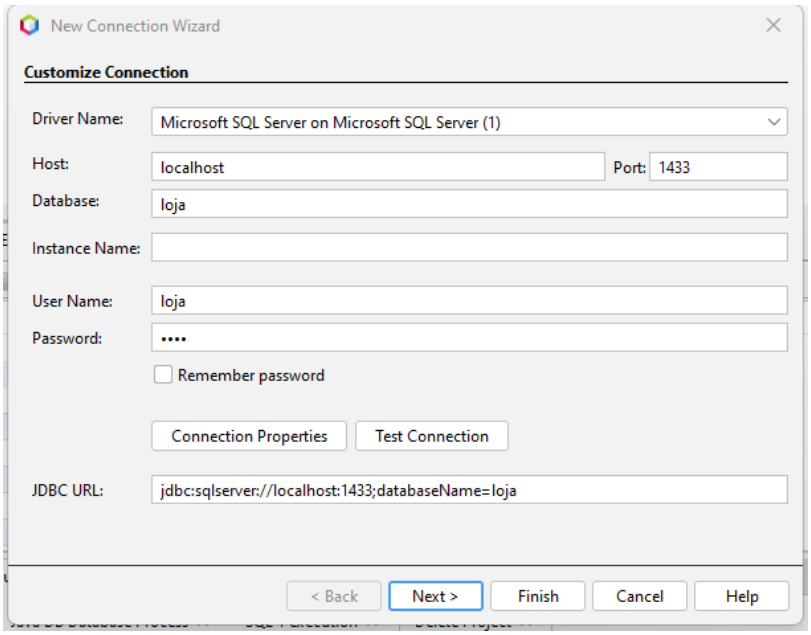
Objetivos da prática

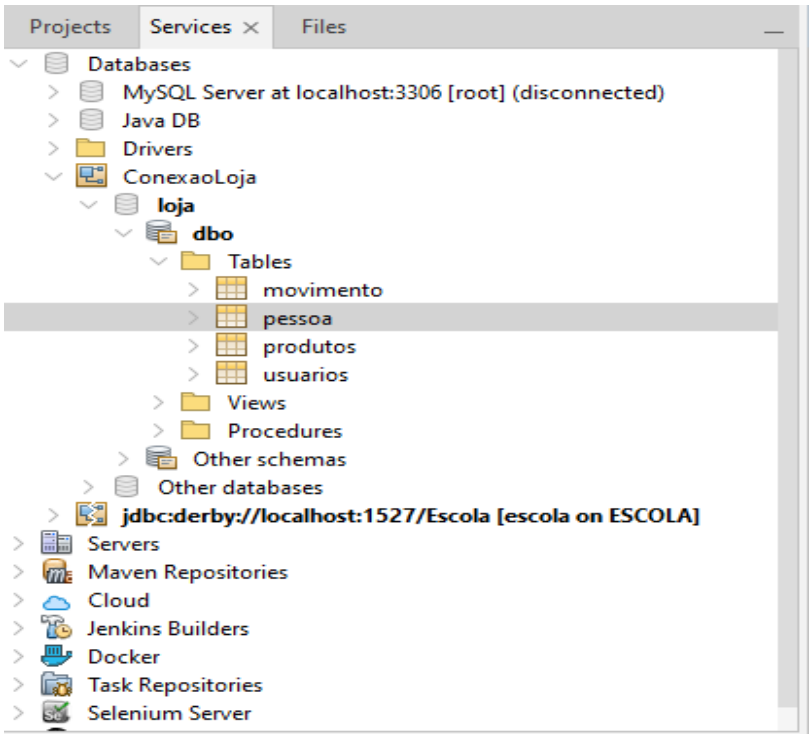
1. Implementar persistência com base no middleware JDBC.
2. Utilizar o padrão DAO (Data Access Object) no manuseio de dados.
3. Implementar o mapeamento objeto-relacional em sistemas Java.
4. Criar sistemas cadastrais com persistência em banco relacional.
5. No final do exercício, o aluno terá criado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

Criando projeto, configurando bibliotecas e conectando banco com netbeans:

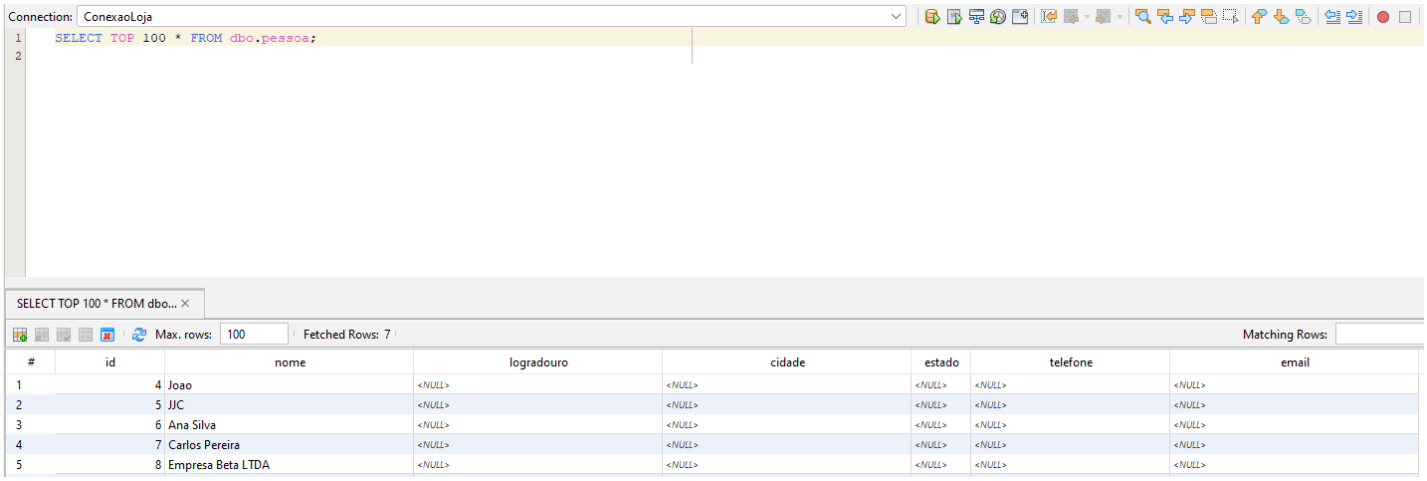








Views das tabelas e dados do banco:



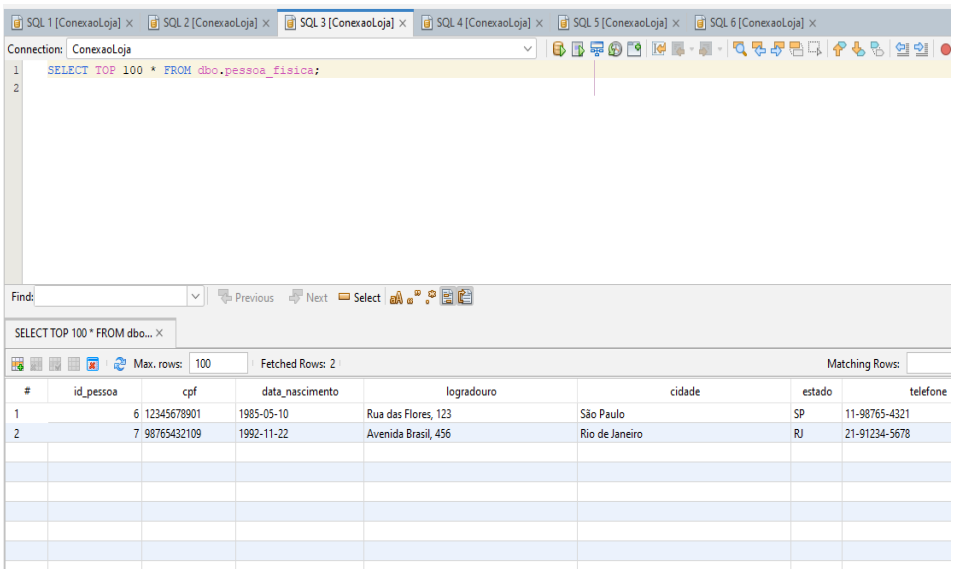
Connection: ConexaoLoja

```
SELECT TOP 100 * FROM dbo.pessoa;
```

SELECT TOP 100 * FROM dbo... ×

Max. rows: 100 Fetched Rows: 7 Matching Rows:

#	id	nome	logradouro	cidade	estado	telefone	email
1	4	Joao	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
2	5	JJC	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
3	6	Ana Silva	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
4	7	Carlos Pereira	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
5	8	Empresa Beta LTDA	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>



SQL 1 [ConexaoLoja] × SQL 2 [ConexaoLoja] × SQL 3 [ConexaoLoja] × SQL 4 [ConexaoLoja] × SQL 5 [ConexaoLoja] × SQL 6 [ConexaoLoja] ×

Connection: ConexaoLoja

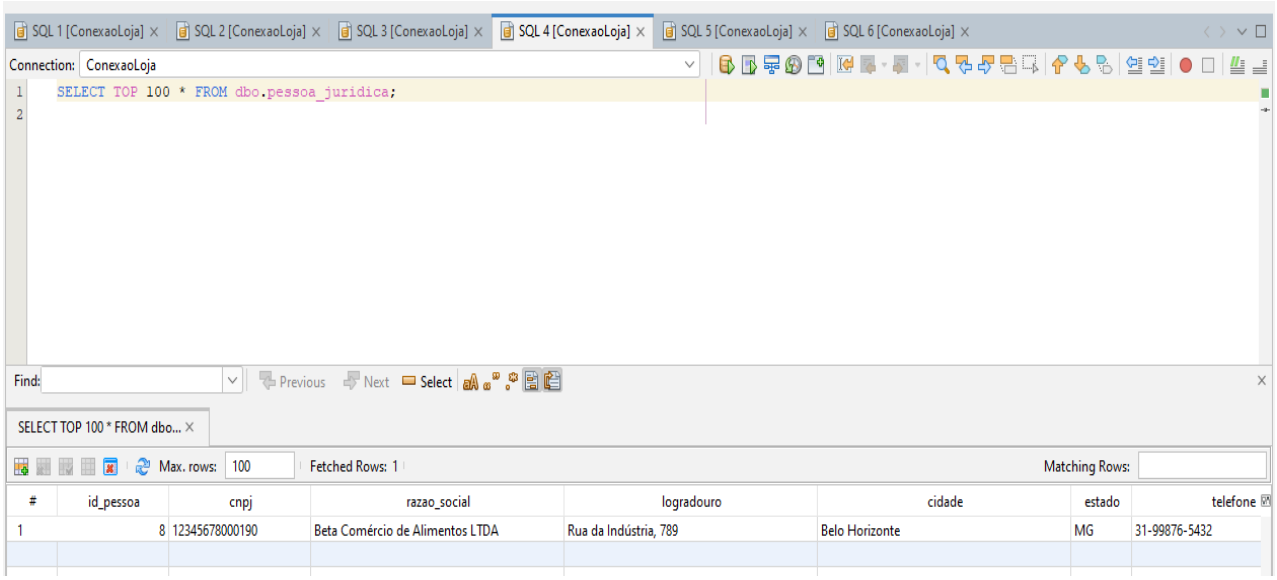
```
SELECT TOP 100 * FROM dbo.pessoa_fisica;
```

Find: Previous Next Select

SELECT TOP 100 * FROM dbo... ×

Max. rows: 100 Fetched Rows: 2 Matching Rows:

#	id_pessoa	cpf	data_nascimento	logradouro	cidade	estado	telefone
1	6	12345678901	1985-05-10	Rua das Flores, 123	São Paulo	SP	11-98765-4321
2	7	98765432109	1992-11-22	Avenida Brasil, 456	Rio de Janeiro	RJ	21-91234-5678



SQL 1 [ConexaoLoja] × SQL 2 [ConexaoLoja] × SQL 3 [ConexaoLoja] × SQL 4 [ConexaoLoja] × SQL 5 [ConexaoLoja] × SQL 6 [ConexaoLoja] ×

Connection: ConexaoLoja

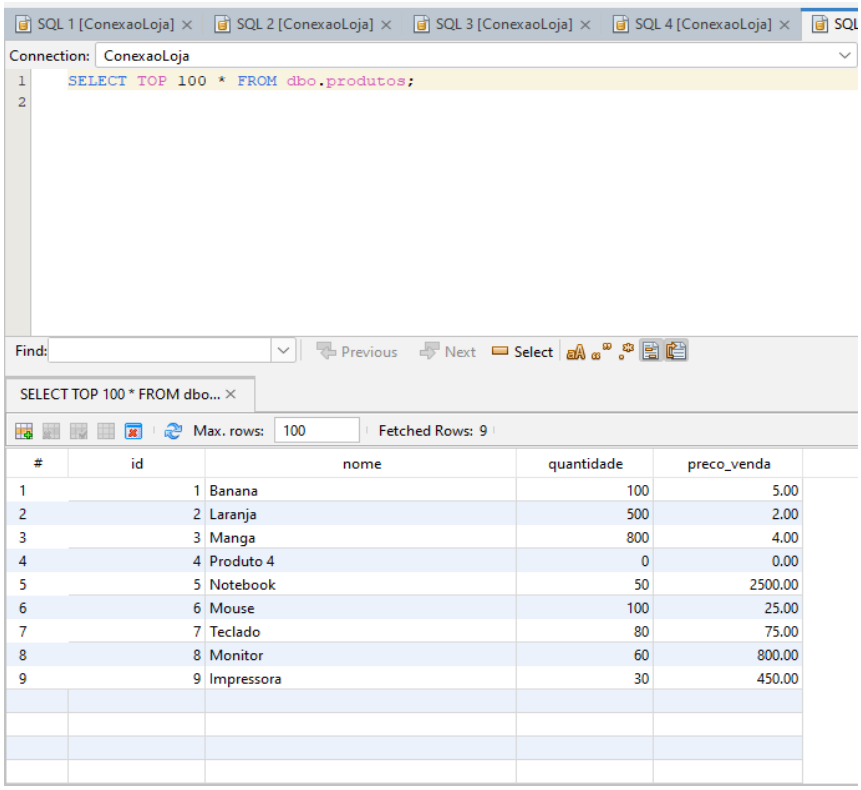
```
SELECT TOP 100 * FROM dbo.pessoa_juridica;
```

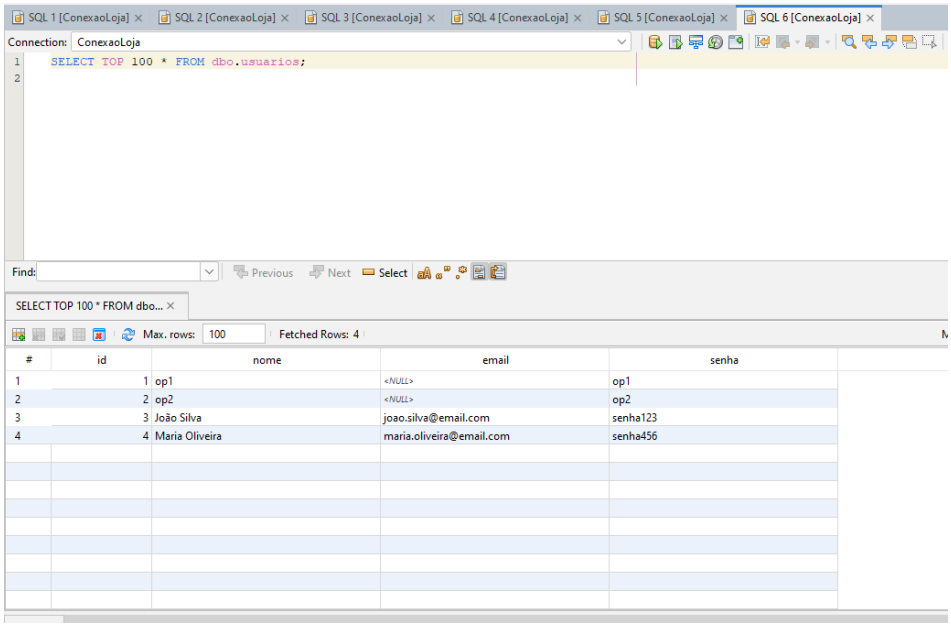
Find: Previous Next Select

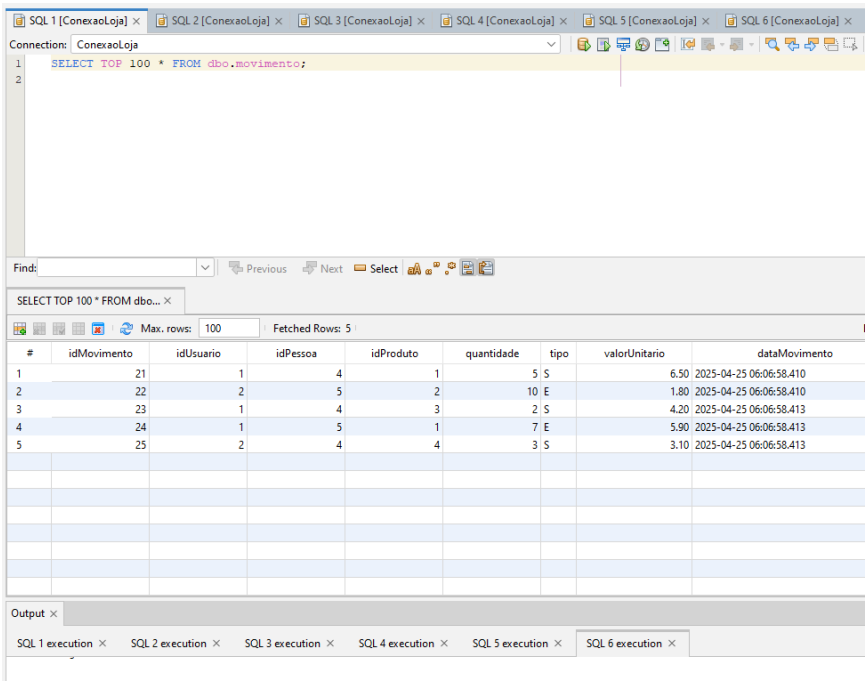
SELECT TOP 100 * FROM dbo... ×

Max. rows: 100 Fetched Rows: 1 Matching Rows:

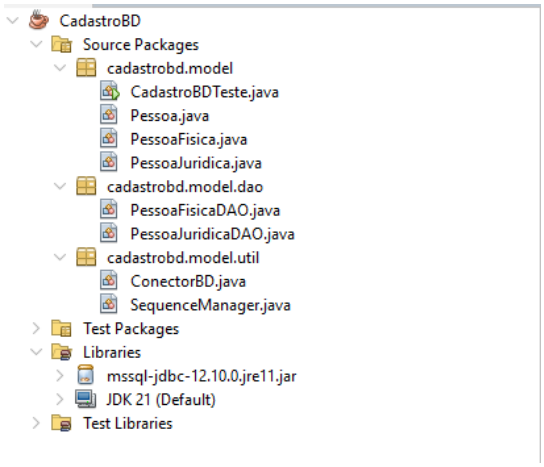
#	id_pessoa	cnpj	razao_social	logradouro	cidade	estado	telefone
1	8	12345678000190	Beta Comércio de Alimentos LTDA	Rua da Indústria, 789	Belo Horizonte	MG	31-99876-5432


SQL 1 [ConexaoLoja]
SQL 2 [ConexaoLoja]
SQL 3 [ConexaoLoja]
SQL 4 [ConexaoLoja]
Connection: ConexaoLoja
SELECT TOP 100 * FROM dbo.produtos;
Find:
Previous
Next
Select
SELECT TOP 100 * FROM dbo...
Max. rows: 100
Fetched Rows: 9
id nome quantidade preco_venda
1 1 Banana 100 5.00
2 2 Laranja 500 2.00
3 3 Manga 800 4.00
4 4 Produto 4 0 0.00
5 5 Notebook 50 2500.00
6 6 Mouse 100 25.00
7 7 Teclado 80 75.00
8 8 Monitor 60 800.00
9 9 Impressora 30 450.00


SQL 1 [ConexaoLoja]
SQL 2 [ConexaoLoja]
SQL 3 [ConexaoLoja]
SQL 4 [ConexaoLoja]
SQL 5 [ConexaoLoja]
SQL 6 [ConexaoLoja]
Connection: ConexaoLoja
SELECT TOP 100 * FROM dbo.usuarios;
Find:
Previous
Next
Select
SELECT TOP 100 * FROM dbo...
Max. rows: 100
Fetched Rows: 4
id nome email senha
1 1 op1 <NULL> op1
2 2 op2 <NULL> op2
3 3 João Silva joao.silva@email.com senha123
4 4 Maria Oliveira maria.oliveira@email.com senha456


SQL 1 [ConexaoLoja]
SQL 2 [ConexaoLoja]
SQL 3 [ConexaoLoja]
SQL 4 [ConexaoLoja]
SQL 5 [ConexaoLoja]
SQL 6 [ConexaoLoja]
Connection: ConexaoLoja
SELECT TOP 100 * FROM dbo.movimento;
Find:
Previous
Next
Select
SELECT TOP 100 * FROM dbo...
Max. rows: 100
Fetched Rows: 5
idMovimento idUsuario idPessoa idProduto quantidade tipo valorUnitario dataMovimento
1 21 1 4 1 5 S 6.50 2025-04-25 06:06:58.410
2 22 2 5 2 10 E 1.80 2025-04-25 06:06:58.410
3 23 1 4 3 2 S 4.20 2025-04-25 06:06:58.413
4 24 1 5 1 7 E 5.90 2025-04-25 06:06:58.413
5 25 2 4 4 3 S 3.10 2025-04-25 06:06:58.413
Output
SQL 1 execution
SQL 2 execution
SQL 3 execution
SQL 4 execution
SQL 5 execution
SQL 6 execution

Estrutura do projeto e Códigos solicitados:



Pessoa.java

Pessoa.java

Source History

```java
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 void exibir() {
        System.out.println("ID: " + id);
        System.out.println("Nome: " + nome);
        System.out.println("Endereço: " + logradouro + ", " + cidade + " - " + estado);
        System.out.println("Telefone: " + telefone);
        System.out.println("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; }
```

PessoaFisica.java

PessoaFisica.java

Source | History

```java
package cadastrobd.model;

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;
    }

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CPF: " + cpf);
    }

    public String getCpf() {
        return cpf;
    }

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

PessoaJuridica.java

PessoaJuridica.java

Source | History

```java
package cadastrobd.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;
    }

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

    // Getter e Setter
    public String getCnpj() {
        return cnpj;
    }

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

PessoaFisicaDAO.java PARTE 1

PessoaFisicaDAO.java

Source | History

```java
package cadastrobd.model.dao;

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

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

public class PessoaFisicaDAO {

    public PessoaFisica getPessoa(int id) {
        String sql = "SELECT p.id, p.nome, p.logradouro, p.cidade, p.estado, p.telefone, p.email, f.cpf " +
                     "FROM Pessoa p JOIN PessoaFisica f ON p.id = f.id WHERE p.id = ?";
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
        PessoaFisica pessoa = null;

        try {
            conn = ConectorBD.getConnection();
            stmt = ConectorBD.getPrepared(sql, conn);
            stmt.setInt(1, id);
            rs = ConectorBD.getSelect(stmt);

            if (rs.next()) {
                pessoa = new PessoaFisica();
                pessoa.setId(rs.getInt("id"));
                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(stmt);
            ConectorBD.close(conn);
        }

        return pessoa;
    }
```

PessoaFisicaDAO.java PARTE 2

```java
    public List<PessoaFisica> getPessoas() {
        List<PessoaFisica> lista = new ArrayList<>();
        String sql = "SELECT p.id, p.nome, p.logradouro, p.cidade, p.estado, p.telefone, p.email, f.cpf " +
                     "FROM Pessoa p JOIN PessoaFisica f ON p.id = f.id";
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;

        try {
            conn = ConectorBD.getConnection();
            stmt = ConectorBD.getPrepared(sql, conn);
            rs = ConectorBD.getSelect(stmt);

            while (rs.next()) {
                PessoaFisica pessoa = new PessoaFisica();
                pessoa.setId(rs.getInt("id"));
                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"));
                lista.add(pessoa);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(rs);
            ConectorBD.close(stmt);
            ConectorBD.close(conn);
        }

        return lista;
    }

    public void incluir(PessoaFisica pessoa) {
        int id = SequenceManager.getValue("seq_pessoa");
        String sqlPessoa = "INSERT INTO Pessoa (id, nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
        String sqlFisica = "INSERT INTO PessoaFisica (id, cpf) VALUES (?, ?)";
        Connection conn = null;
        PreparedStatement stmt = null;

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

            stmt = ConectorBD.getPrepared(sqlPessoa, conn);
            stmt.setInt(1, id);
```

PessoaFisicaDAO.java PARTE 3

```
            stmt.setString(i: 2, string: pessoa.getNome());
            stmt.setString(i: 3, string: pessoa.getLogradouro());
            stmt.setString(i: 4, string: pessoa.getCidade());
            stmt.setString(i: 5, string: pessoa.getEstado());
            stmt.setString(i: 6, string: pessoa.getTelefone());
            stmt.setString(i: 7, string: pessoa.getEmail());
            stmt.executeUpdate();

            stmt = ConectorBD.getPrepared(sql:sqlFisica, conn);
            stmt.setInt(i: 1, il: id);
            stmt.setString(i: 2, string: pessoa.getCpf());
            stmt.executeUpdate();

            conn.commit();
        } catch (SQLException e) {
            try { if (conn != null) conn.rollback(); } catch (SQLException ignored) {}
            e.printStackTrace();
        } finally {
            ConectorBD.close(stmt);
            ConectorBD.close(conn);
        }
    }

    public void alterar(PessoaFisica pessoa) {
        String sqlPessoa = "UPDATE Pessoa SET nome=?, logradouro=?, cidade=?, estado=?, telefone=?, email=? WHERE id=?";
        String sqlFisica = "UPDATE PessoaFisica SET cpf=? WHERE id=?";
        Connection conn = null;
        PreparedStatement stmt = null;

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

            stmt = ConectorBD.getPrepared(sql:sqlPessoa, conn);
            stmt.setString(i: 1, string: pessoa.getNome());
            stmt.setString(i: 2, string: pessoa.getLogradouro());
            stmt.setString(i: 3, string: pessoa.getCidade());
            stmt.setString(i: 4, string: pessoa.getEstado());
            stmt.setString(i: 5, string: pessoa.getTelefone());
            stmt.setString(i: 6, string: pessoa.getEmail());
            stmt.setInt(i: 7, il: pessoa.getId());
            stmt.executeUpdate();

            stmt = ConectorBD.getPrepared(sql:sqlFisica, conn);
            stmt.setString(i: 1, string: pessoa.getCpf());
            stmt.setInt(i: 2, il: pessoa.getId());
            stmt.executeUpdate();

            conn.commit();
```

PessoaFisicaDAO.java PARTE 4

```
        } catch (SQLException e) {
            try { if (conn != null) conn.rollback(); } catch (SQLException ignored) {}
            e.printStackTrace();
        } finally {
            ConectorBD.close(stmt);
            ConectorBD.close(conn);
        }
    }

    public void excluir(int id) {
        String sqlFisica = "DELETE FROM PessoaFisica WHERE id=?";
        String sqlPessoa = "DELETE FROM Pessoa WHERE id=?";
        Connection conn = null;
        PreparedStatement stmt = null;

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

            stmt = ConectorBD.getPrepared(sql:sqlFisica, conn);
            stmt.setInt(i: 1, il: id);
            stmt.executeUpdate();

            stmt = ConectorBD.getPrepared(sql:sqlPessoa, conn);
            stmt.setInt(i: 1, il: id);
            stmt.executeUpdate();

            conn.commit();
        } catch (SQLException e) {
            try { if (conn != null) conn.rollback(); } catch (SQLException ignored) {}
            e.printStackTrace();
        } finally {
            ConectorBD.close(stmt);
            ConectorBD.close(conn);
        }
    }
}
```

PessoaJuridicaDAO.java PARTE 01

PessoaJuridicaDAO.java

Source | History

```java
package cadastrobd.model.dao;

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

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

public class PessoaJuridicaDAO {

    public PessoaJuridica getPessoa(int id) {
        String sql = "SELECT p.id, p.nome, p.logradouro, p.cidade, p.estado, p.telefone, p.email, j.cnpj " +
                     "FROM Pessoa p JOIN PessoaJuridica j ON p.id = j.id WHERE p.id = ?";
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
        PessoaJuridica pessoa = null;

        try {
            conn = ConectorBD.getConnection();
            stmt = ConectorBD.getPrepared(sql, conn);
            stmt.setInt(i: 1, il: id);
            rs = ConectorBD.getSelect(stmt);

            if (rs.next()) {
                pessoa = new PessoaJuridica();
                pessoa.setId(id: rs.getInt(string: "id"));
                pessoa.setNome(nome: rs.getString(string: "nome"));
                pessoa.setLogradouro(logradouro: rs.getString(string: "logradouro"));
                pessoa.setCidade(cidade: rs.getString(string: "cidade"));
                pessoa.setEstado(estado: rs.getString(string: "estado"));
                pessoa.setTelefone(telefone: rs.getString(string: "telefone"));
                pessoa.setEmail(email: rs.getString(string: "email"));
                pessoa.setCnpj(cnpj: rs.getString(string: "cnpj"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(rs);
            ConectorBD.close(stmt);
            ConectorBD.close(conn);
        }

        return pessoa;
    }

```

PessoaJuridicaDAO.java PARTE 02

```java
    public List<PessoaJuridica> getPessoas() {
        List<PessoaJuridica> lista = new ArrayList<>();
        String sql = "SELECT p.id, p.nome, p.logradouro, p.cidade, p.estado, p.telefone, p.email, j.cnpj " +
                     "FROM Pessoa p JOIN PessoaJuridica j ON p.id = j.id";
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;

        try {
            conn = ConectorBD.getConnection();
            stmt = ConectorBD.getPrepared(sql, conn);
            rs = ConectorBD.getSelect(stmt);

            while (rs.next()) {
                PessoaJuridica pessoa = new PessoaJuridica();
                pessoa.setId(id: rs.getInt(string: "id"));
                pessoa.setNome(nome: rs.getString(string: "nome"));
                pessoa.setLogradouro(logradouro: rs.getString(string: "logradouro"));
                pessoa.setCidade(cidade: rs.getString(string: "cidade"));
                pessoa.setEstado(estado: rs.getString(string: "estado"));
                pessoa.setTelefone(telefone: rs.getString(string: "telefone"));
                pessoa.setEmail(email: rs.getString(string: "email"));
                pessoa.setCnpj(cnpj: rs.getString(string: "cnpj"));
                lista.add(e: pessoa);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(rs);
            ConectorBD.close(stmt);
            ConectorBD.close(conn);
        }

        return lista;
    }

    public void incluir(PessoaJuridica pessoa) {
        int id = SequenceManager.getValue(sequenceName: "seq_pessoa");
        String sqlPessoa = "INSERT INTO Pessoa (id, nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
        String sqlJuridica = "INSERT INTO PessoaJuridica (id, cnpj) VALUES (?, ?)";
        Connection conn = null;
        PreparedStatement stmt = null;

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

PessoaJuridicaDAO.java PARTE 03

```
96              stmt = ConectorBD.getPrepared(sql:sqlPessoa, conn);
97              stmt.setInt(i: 1, il: id);
98              stmt.setString(i: 2, string: pessoa.getNome());
99              stmt.setString(i: 3, string: pessoa.getLogradouro());
100             stmt.setString(i: 4, string: pessoa.getCidade());
101             stmt.setString(i: 5, string: pessoa.getEstado());
102             stmt.setString(i: 6, string: pessoa.getTelefone());
103             stmt.setString(i: 7, string: pessoa.getEmail());
104             stmt.executeUpdate();
105
106             stmt = ConectorBD.getPrepared(sql:sqlJuridica, conn);
107             stmt.setInt(i: 1, il: id);
108             stmt.setString(i: 2, string: pessoa.getCnpj());
109             stmt.executeUpdate();
110
111             conn.commit();
112         } catch (SQLException e) {
113             try { if (conn != null) conn.rollback(); } catch (SQLException ignored) {}
114             e.printStackTrace();
115         } finally {
116             ConectorBD.close(stmt);
117             ConectorBD.close(conn);
118         }
119     }
120
121     public void alterar(PessoaJuridica pessoa) {
122         String sqlPessoa = "UPDATE Pessoa SET nome=?, logradouro=?, cidade=?, estado=?, telefone=?, email=? WHERE id=?";
123         String sqlJuridica = "UPDATE PessoaJuridica SET cnpj=? WHERE id=?";
124         Connection conn = null;
125         PreparedStatement stmt = null;
126
127         try {
128             conn = ConectorBD.getConnection();
129             conn.setAutoCommit(bln:false);
130
131             stmt = ConectorBD.getPrepared(sql:sqlPessoa, conn);
132             stmt.setString(i: 1, string: pessoa.getNome());
133             stmt.setString(i: 2, string: pessoa.getLogradouro());
134             stmt.setString(i: 3, string: pessoa.getCidade());
135             stmt.setString(i: 4, string: pessoa.getEstado());
136             stmt.setString(i: 5, string: pessoa.getTelefone());
137             stmt.setString(i: 6, string: pessoa.getEmail());
138             stmt.setInt(i: 7, il: pessoa.getId());
139             stmt.executeUpdate();
140
141             stmt = ConectorBD.getPrepared(sql:sqlJuridica, conn);
142             stmt.setString(i: 1, string: pessoa.getCnpj());
143             stmt.setInt(i: 2, il: pessoa.getId());
```

PessoaJuridicaDAO.java PARTE 04

```
144             stmt.executeUpdate();
145
146             conn.commit();
147         } catch (SQLException e) {
148             try { if (conn != null) conn.rollback(); } catch (SQLException ignored) {}
149             e.printStackTrace();
150         } finally {
151             ConectorBD.close(stmt);
152             ConectorBD.close(conn);
153         }
154     }
155
156     public void excluir(int id) {
157         String sqlJuridica = "DELETE FROM PessoaJuridica WHERE id=?";
158         String sqlPessoa = "DELETE FROM Pessoa WHERE id=?";
159         Connection conn = null;
160         PreparedStatement stmt = null;
161
162         try {
163             conn = ConectorBD.getConnection();
164             conn.setAutoCommit(bln:false);
165
166             stmt = ConectorBD.getPrepared(sql:sqlJuridica, conn);
167             stmt.setInt(i: 1, il: id);
168             stmt.executeUpdate();
169
170             stmt = ConectorBD.getPrepared(sql:sqlPessoa, conn);
171             stmt.setInt(i: 1, il: id);
172             stmt.executeUpdate();
173
174             conn.commit();
175         } catch (SQLException e) {
176             try { if (conn != null) conn.rollback(); } catch (SQLException ignored) {}
177             e.printStackTrace();
178         } finally {
179             ConectorBD.close(stmt);
180             ConectorBD.close(conn);
181         }
182     }
183 }
```

SequenceManager.java

SequenceManager.java

Source | History

```java
package cadastrobd.model.util;

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

public class SequenceManager {

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

        try {
            conn = ConectorBD.getConnection();
            stmt = conn.prepareStatement("SELECT NEXT VALUE FOR " + sequenceName);
            rs = ConectorBD.getSelect(stmt);

            if (rs.next()) {
                nextVal = rs.getInt(i: 1);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            ConectorBD.close(rs);
            ConectorBD.close(stmt);
            ConectorBD.close(conn);
        }

        return nextVal;
    }
}
```

ConectorBD.java

ConectorBD.java

Source | History

```java
package cadastrobd.model.util;

import java.sql.*;

public class ConectorBD {

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

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(url:URL);
    }

    public static PreparedStatement getPrepared(String sql, Connection conn) throws SQLException {
        return conn.prepareStatement(string: sql);
    }

    public static ResultSet getSelect(PreparedStatement stmt) throws SQLException {
        return stmt.executeQuery();
    }

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

    public static void close(Statement stmt) {
        if (stmt != null) {
            try {
                stmt.close();
            } catch (SQLException ignored) {}
        }
    }

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

ConectorBDTeste.java PARTE 01

CadastroBDTeste.java

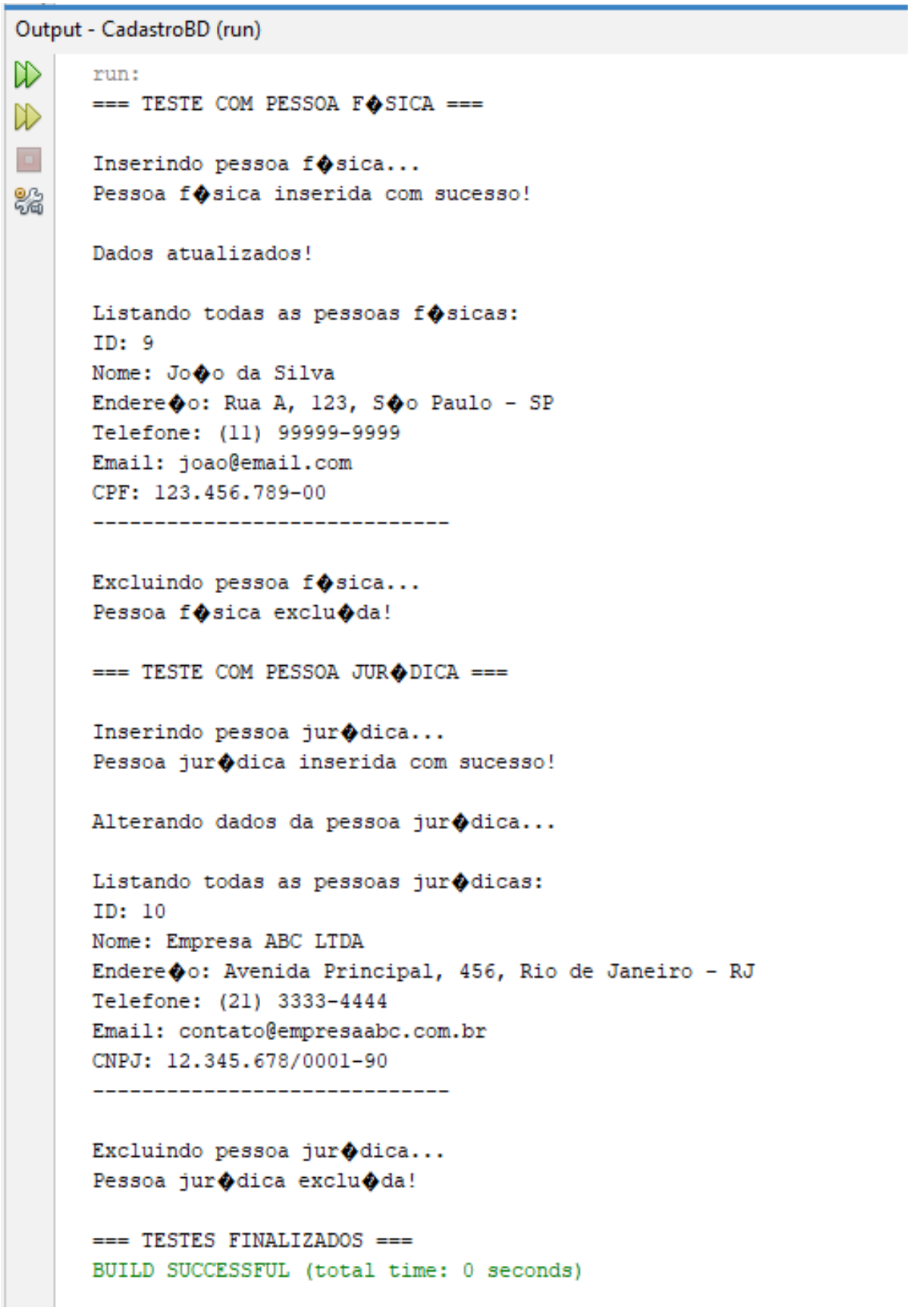
Source | History

```java
package cadastrobd.model;

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

public class CadastroBDTeste {

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

        System.out.println(x: "=== TESTE COM PESSOA FÍSICA ===");

        PessoaFisica pf = new PessoaFisica();
        pf.setNome(nome: "João da Silva");
        pf.setLogradouro(logradouro: "Rua A, 123");
        pf.setCidade(cidade: "São Paulo");
        pf.setEstado(estado: "SP");
        pf.setTelefone(telefone: "(11) 99999-9999");
        pf.setEmail(email: "joao@email.com");
        pf.setCpf(cpf: "123.456.789-00");

        System.out.println(x: "\nInserindo pessoa física...");
        pessoaFisicaDAO.incluir(pessoa: pf);
        System.out.println(x: "Pessoa física inserida com sucesso!");

        pf.setTelefone(telefone: "(11) 88888-8888");
        pessoaFisicaDAO.alterar(pessoa: pf);
        System.out.println(x: "\nDados atualizados!");

        System.out.println(x: "\nListando todas as pessoas físicas:");
        for (PessoaFisica p : pessoaFisicaDAO.getPessoas()) {
            p.exibir();
            System.out.println(x: "-----------------------------");
        }
```

ConectorBDTeste.java PARTE 02

```java

        System.out.println(x: "\nExcluindo pessoa física...");
        pessoaFisicaDAO.excluir(id: pf.getId());
        System.out.println(x: "Pessoa física excluída!");

        System.out.println(x: "\n=== TESTE COM PESSOA JURÍDICA ===");

        PessoaJuridica pj = new PessoaJuridica();
        pj.setNome(nome: "Empresa ABC LTDA");
        pj.setLogradouro(logradouro: "Avenida Principal, 456");
        pj.setCidade(cidade: "Rio de Janeiro");
        pj.setEstado(estado: "RJ");
        pj.setTelefone(telefone: "(21) 3333-4444");
        pj.setEmail(email: "contato@empresaabc.com.br");
        pj.setCnpj(cnpj: "12.345.678/0001-90");

        System.out.println(x: "\nInserindo pessoa jurídica...");
        pessoaJuridicaDAO.incluir(pessoa: pj);
        System.out.println(x: "Pessoa jurídica inserida com sucesso!");

        pj.setTelefone(telefone: "(21) 2222-3333");
        pessoaJuridicaDAO.alterar(pessoa: pj);
        System.out.println(x: "\nAlterando dados da pessoa jurídica...");

        System.out.println(x: "\nListando todas as pessoas jurídicas:");
        for (PessoaJuridica p : pessoaJuridicaDAO.getPessoas()) {
            p.exibir();
            System.out.println(x: "-----------------------------");
        }

        System.out.println(x: "\nExcluindo pessoa jurídica...");
        pessoaJuridicaDAO.excluir(id: pj.getId());
        System.out.println(x: "Pessoa jurídica excluída!");

        System.out.println(x: "\n=== TESTES FINALIZADOS ===");
    }
}
```

Resultado da saída no Console:



1. Análise e Conclusão:

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

O JDBC é super importante porque ele é uma ponte que o Java usa pra conversar com o banco de dados. Sem ele, o Java não conseguiria mandar comandos SQL nem pegar os dados de volta. Então, o JDBC é essencial pra qualquer programa em Java que precise guardar ou buscar informações num banco.

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

A diferença no uso de Statement ou PreparedStatement para a manipulação de dados é que o Statement é usado para comandos SQL simples, onde os valores são concatenados diretamente na string do comando, o que pode deixar o código vulnerável a ataques de injeção de SQL. Já o PreparedStatement é mais seguro e eficiente, pois permite o uso de parâmetros que são tratados separadamente do comando SQL, prevenindo a injeção de SQL e permitindo que o banco de dados compile o comando antecipadamente, otimizando a execução de consultas repetidas.

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

O padrão DAO ajuda muito na manutenibilidade porque ele separa a lógica de acesso aos dados do resto do programa. Isso significa que, se você precisar mudar a forma como os dados são armazenados (por exemplo, de um banco de dados para outro), você só precisa mexer nas classes DAO, sem alterar as outras partes do código. Facilita bastante a vida do programador na hora de fazer mudanças e corrigir problemas!

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

Em programação orientada a objetos, a herança é um mecanismo que permite a uma classe (subclasse) herdar atributos e métodos de outra classe (superclasse). No entanto, o modelo relacional de banco de dados não oferece suporte direto a esse conceito. Portanto, a representação da herança em um modelo relacional requer a utilização de estratégias específicas. As três abordagens principais são:

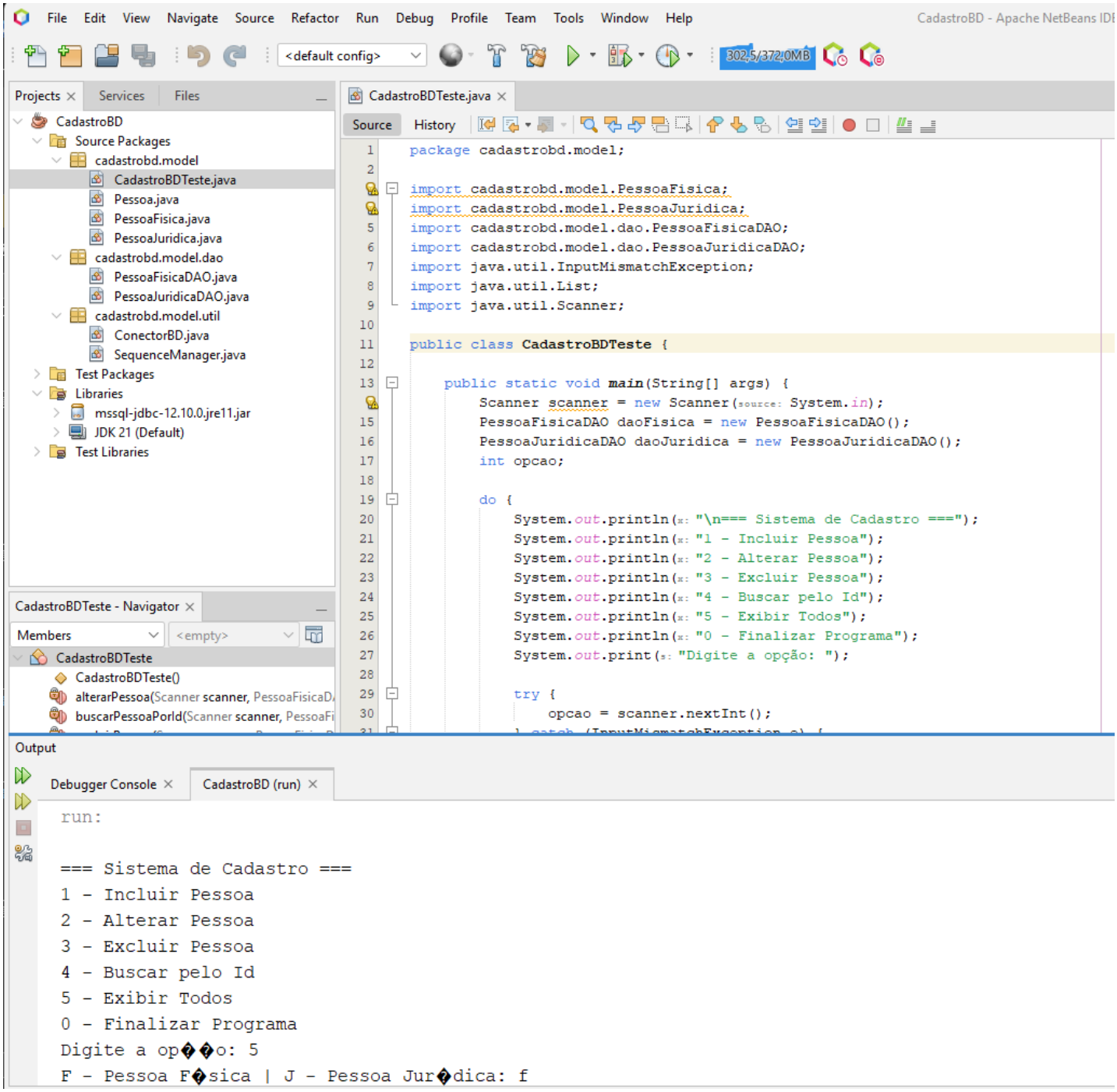
1. **Tabela Única: Nessa estratégia, todas as classes da hierarquia são mapeadas para uma única tabela. A tabela contém todos os atributos da superclasse e das subclasses, e uma coluna discriminadora é utilizada para indicar a qual subclasse pertence cada registro.**

2. **Tabela por Classe Concreta: Nessa abordagem, cada classe concreta (ou seja, as classes que podem ser instanciadas) é mapeada para uma tabela separada. Cada tabela contém todos os atributos da classe, incluindo os atributos herdados da superclasse.**

3. **Tabela por Classe: Nessa estratégia, cada classe da hierarquia, incluindo a superclasse e as subclasses, é mapeada para uma tabela. A tabela da superclasse contém os atributos comuns a todas as subclasses, enquanto as tabelas das subclasses contêm apenas os atributos específicos de cada subclasse, além de uma chave estrangeira que referencia a tabela da superclasse.**

A escolha da estratégia mais adequada depende dos requisitos da aplicação, incluindo a complexidade da hierarquia de classes, os padrões de consulta e as considerações de desempenho.

2 PROCEDIMENTO: Alimentando base



Análise e Conclusão:

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

- **Arquivo: Armazena dados em arquivos no sistema de arquivos (txt, csv, etc.). É mais simples para volumes pequenos, mas oferece consultas limitadas, baixa concorrência (dificuldade de acesso simultâneo) e não suporta transações (garantia de operações completas).**

- **Banco de dados: Utiliza um SGBD para armazenar dados de forma estruturada. Adequado para grandes volumes, permite consultas complexas via SQL, oferece alta concorrência com controle de acesso e suporta transações para garantir a integridade dos dados.**

❓ Como o uso de operador lambda simplificou a

impressão dos valores contidos nas entidades, nas versões mais recentes do Java?
Simplificou o código, permitindo a criação de funções anônimas concisas. Em vez de loops verbosos, usa-se forEach() com uma expressão lambda para iterar sobre a coleção e especificar a ação (imprimir o valor) de forma mais direta e legível.

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

Porque main é chamado diretamente pela JVM, que não instancia a classe que o contém. O modificador static indica que o método pertence à classe em si, e não a uma instância dela, permitindo que seja chamado diretamente pelo nome da classe, sem a necessidade de um objeto.