



Estácio

- Matheus Gomes Silva – Matricula 2024 0147 8822
- DESENVOLVIMENTO FULL STACK
- Disciplina: RPG0016 - Back-end Sem Banco Não Tem!
- Semestre Letivo: 2025.1
- Repositório Git: <https://github.com/matheusg999/CadastroBD>

Missão Prática | Nível 3 | Mundo 3

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

Procedimento 1: Mapeamento Objeto-Relacional e DAO

Procedimento 2: Alimentando a Base

Objetivo da Prática

- Implementar persistência com base no middleware JDBC.
- Utilizar o padrão DAO (Data Access Object) no manuseio de dados.
- Implementar o mapeamento objeto-relacional em sistemas Java.
- Criar sistemas cadastrais com persistência em banco relacional.

- No final do exercício, o aluno terá criado um aplicativo cadastral com uso do SQL Server na persistência de dados.

Códigos

Procedimento 1: Mapeamento Objeto-Relacional e DAO

Classe Pessoa:

```
package cadastrobd.model;
```

```
public abstract class Pessoa { // 'abstract' se não for instanciada diretamente
```

```
    protected int id;
```

```
    protected String nome;
```

```
    protected String logradouro;
```

```
    protected String cidade;
```

```
    protected String estado;
```

```
    protected String telefone;
```

```
    protected String email;
```

```
    public Pessoa() {
```

```
        // Construtor padrão
```

```
}
```

```
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;
```

```
}
```

```
// Getters para os atributos
```

```
public int getId() {
```

```
    return id;
```

```
}
```

```
public String getNome() {
```

```
    return nome;
```

```
}
```

```
public String getLogradouro() {
```

```
    return logradouro;
```

```
}
```

```
public String getCidade() {
```

```
    return cidade;
```

```
}
```

```
public String getEstado() {  
    return estado;  
}
```

```
public String getTelefone() {  
    return telefone;  
}
```

```
public String getEmail() {  
    return email;  
}
```

```
// Setters (opcional, adicione se precisar modificar os atributos)
```

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

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

```
// ... e assim por diante para os outros setters
```

```
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 Pessoa Física:

```
package cadastrobd.model;
```

```
public class PessoaFisica extends Pessoa {
```

```
    private String cpf;
```

```
    // Construtor padrão
```

```
    public PessoaFisica() {
```

```
        super();
```

```
    }
```

```
    // Construtor completo
```

```
    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;
```

```
    }
```

```
// Método Getter para CPF

public String getCpf() {

    return cpf;

}


// Método Setter para CPF (opcional, adicione se precisar modificar o CPF após a
criação)

public void setCpf(String cpf) {

    this.cpf = cpf;

}


@Override

public void exibir() {

    super.exibir();

    System.out.println("CPF: " + cpf);

}

}
```

Classe Pessoa Jurídica:

```
package cadastrobd.model;


public class PessoaJuridica extends Pessoa {

    private String cnpj;


    // Construtor padrão
```

```
public PessoaJuridica() {  
    super();  
}  
  
// Construtor completo  
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;  
}  
  
// Método Getter para CNPJ  
public String getCnpj() {  
    return cnpj;  
}  
  
// Método Setter para CNPJ  
public void setCnpj(String cnpj) {  
    this.cnpj = cnpj;  
}  
  
@Override  
public void exibir() {  
    super.exibir();  
    System.out.println("CNPJ: " + cnpj);  
}  
}
```

Classe ConectorBD:

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

```
import java.sql.*;
```

```
public class ConectorBD {
```

```
    private static final String URL =  
    "jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCertifica  
te=true";
```

```
    private static final String USER = "loja";
```

```
    private static final String PASSWORD = "loja";
```

```
    // Retorna uma conexão com o banco
```

```
    public static Connection getConnection() throws SQLException {
```

```
        return DriverManager.getConnection(URL, USER, PASSWORD);
```

```
    }
```

```
    // Retorna um PreparedStatement para o SQL informado e conexão fornecida
```



```
public static PreparedStatement getPrepared(String sql, Connection con) throws  
SQLException {
```

```
    return con.prepareStatement(sql);  
}
```

```
// Retorna um ResultSet para uma consulta SQL e conexão fornecida
```

```
public static ResultSet getSelect(String sql, Connection con) throws SQLException {
```

```
    Statement stmt = con.createStatement();  
    return stmt.executeQuery(sql);  
}
```

```
// Métodos close para fechar Statement, ResultSet e Connection
```

```
public static void close(Statement stmt) {
```

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

```
public static void close(ResultSet rs) {
```

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

```

        e.printStackTrace();
    }
}

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

```

Classe SequenceManager:

```

package cadastro.model.util;

import java.sql.*;

public class SequenceManager {

    // Retorna o próximo valor da sequência informada
    public static long getValue(String sequenceName) {

```

```

        long value = -1;

        String sql = "SELECT NEXT VALUE FOR " + sequenceName + " AS nextVal";

        try (Connection con = ConectorBD.getConnection();

            Statement stmt = con.createStatement();

            ResultSet rs = stmt.executeQuery(sql)) {

            if (rs.next()) {

                value = rs.getLong("nextVal");

            }

        } catch (SQLException e) {

            e.printStackTrace();

        }

        return value;

    }

}

```

Classe PessoaFisicaDAO:

```

package cadastro.model;

import cadastro.model.util.ConectorBD;
import cadastrobd.model.PessoaFisica;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

```

```
import java.util.ArrayList;

import java.util.List;

public class PessoaFisicaDAO {

    public PessoaFisica getPessoa(int id) throws Exception {

        PessoaFisica pessoa = null;

        Connection con = ConectorBD.getConnection();

        PreparedStatement ps = null;

        ResultSet resultado = null;

        // Buscando só as pessoas do tipo 'F' (física)

        String sql = "SELECT * FROM pessoa WHERE idpessoa = ? AND tipoPessoa = 'F'";

        ps = ConectorBD.getPrepared(sql, con);

        ps.setInt(1, id);

        resultado = ps.executeQuery();

        while (resultado.next()) {

            pessoa = new PessoaFisica(

                resultado.getInt("idpessoa"),

                resultado.getString("nome"),

                resultado.getString("logradouro"),

                resultado.getString("cidade"),

                resultado.getString("estado"),

                resultado.getString("telefone"),

                resultado.getString("email"),

                resultado.getString("cpf_cnpj") // Aqui usa o cpf_cnpj para cpf mesmo

            );

        }

    }

}
```

```

        );
    }

    ConectorBD.close(resultado);
    ConectorBD.close(ps);
    ConectorBD.close(con);
    return pessoa;
}

public List<PessoaFisica> getPessoas() throws Exception {
    List<PessoaFisica> lista = new ArrayList<>();
    Connection con = ConectorBD.getConnection();
    PreparedStatement ps = null;
    ResultSet resultado = null;

    String sql = "SELECT * FROM pessoa WHERE tipoPessoa = 'F'";

    ps = ConectorBD.getPrepared(sql, con);
    resultado = ps.executeQuery();

    while (resultado.next()) {
        lista.add(new PessoaFisica(
            resultado.getInt("idpessoa"),
            resultado.getString("nome"),
            resultado.getString("logradouro"),
            resultado.getString("cidade"),
            resultado.getString("estado"),
            resultado.getString("telefone"),

```

```
        resultado.getString("email"),
        resultado.getString("cpf_cnpj")
    ));
}
```

```
ConectorBD.close(resultado);
ConectorBD.close(ps);
ConectorBD.close(con);
return lista;
}
```

```
public void incluir(PessoaFisica pessoafisica) throws Exception {
```

```
    Connection con = ConectorBD.getConnection();
    PreparedStatement ps = null;
```

```
    String sql = "INSERT INTO pessoa (idpessoa, nome, logradouro, cidade, estado,
telefone, email, cpf_cnpj, tipoPessoa) VALUES (?, ?, ?, ?, ?, ?, ?, ?, 'F')";
```

```
    ps = ConectorBD.getPrepared(sql, con);
    ps.setInt(1, pessoafisica.getId());
    ps.setString(2, pessoafisica.getNome());
    ps.setString(3, pessoafisica.getLogradouro());
    ps.setString(4, pessoafisica.getCidade());
    ps.setString(5, pessoafisica.getEstado());
    ps.setString(6, pessoafisica.getTelefone());
    ps.setString(7, pessoafisica.getEmail());
    ps.setString(8, pessoafisica.getCpf()); // cpf no campo cpf_cnpj
```

```
ps.execute();
```

```
ConectorBD.close(ps);
```

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

```
}
```

```
public void alterar(int id, String nome, String logradouro, String cidade, String estado,  
String telefone, String email, String cpf) throws Exception {
```

```
    Connection con = ConectorBD.getConnection();
```

```
    PreparedStatement ps = null;
```

```
    String sql = "UPDATE pessoa SET nome=?, logradouro=?, cidade=?, estado=?,  
telefone=?, email=?, cpf_cnpj=? WHERE idpessoa=? AND tipoPessoa='F'";
```

```
    ps = ConectorBD.getPrepared(sql, con);
```

```
    ps.setString(1, nome);
```

```
    ps.setString(2, logradouro);
```

```
    ps.setString(3, cidade);
```

```
    ps.setString(4, estado);
```

```
    ps.setString(5, telefone);
```

```
    ps.setString(6, email);
```

```
    ps.setString(7, cpf);
```

```
    ps.setInt(8, id);
```

```
ps.execute();
```

```
ConectorBD.close(ps);
```

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

```
}
```

```

public void excluir(int id) throws Exception {
    Connection con = ConectorBD.getConnection();
    PreparedStatement ps = null;

    String sql = "DELETE FROM pessoa WHERE idpessoa=? AND tipoPessoa='F'";

    ps = ConectorBD.getPrepared(sql, con);
    ps.setInt(1, id);

    ps.execute();

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

```

Classe PessoaJuridicaDAO:

```

package cadastro.model;

import cadastro.model.util.ConectorBD;
import cadastrobd.model.PessoaJuridica;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;

```



```
import java.util.List;
```

```
public class PessoaJuridicaDAO {
```

```
    public PessoaJuridica getPessoa(int id) throws Exception {
```

```
        PessoaJuridica pessoa = null;
```

```
        Connection con = ConectorBD.getConnection();
```

```
        PreparedStatement ps = null;
```

```
        ResultSet resultado = null;
```

```
        String sql = "SELECT * FROM pessoa WHERE idpessoa = ? AND tipoPessoa = 'J'";
```

```
        ps = ConectorBD.getPrepared(sql, con);
```

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

```
        resultado = ps.executeQuery();
```

```
        while (resultado.next()) {
```

```
            pessoa = new PessoaJuridica(
```

```
                resultado.getInt("idpessoa"),
```

```
                resultado.getString("nome"),
```

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

```
                resultado.getString("cidade"),
```

```
                resultado.getString("estado"),
```

```
                resultado.getString("telefone"),
```

```
                resultado.getString("email"),
```

```
                resultado.getString("cpf_cnpj") // aqui é CNPJ para jurídica
```

```
            );
```

```
        }
```

```
ConectorBD.close(resultado);

ConectorBD.close(ps);

ConectorBD.close(con);

return pessoa;
}
```

```
public List<PessoaJuridica> getPessoas() throws Exception {
```

```
    List<PessoaJuridica> lista = new ArrayList<>();

    Connection con = ConectorBD.getConnection();

    PreparedStatement ps = null;

    ResultSet resultado = null;
```

```
    String sql = "SELECT * FROM pessoa WHERE tipoPessoa = 'J'";
```

```
    ps = ConectorBD.getPrepared(sql, con);

    resultado = ps.executeQuery();
```

```
    while (resultado.next()) {

        lista.add(new PessoaJuridica(

            resultado.getInt("idpessoa"),

            resultado.getString("nome"),

            resultado.getString("logradouro"),

            resultado.getString("cidade"),

            resultado.getString("estado"),

            resultado.getString("telefone"),

            resultado.getString("email"),

            resultado.getString("cpf_cnpj")
```

```
));  
}
```

```
ConectorBD.close(resultado);  
ConectorBD.close(ps);  
ConectorBD.close(con);  
return lista;  
}
```

```
public void incluir(PessoaJuridica pessoajuridica) throws Exception {  
    Connection con = ConectorBD.getConnection();  
    PreparedStatement ps = null;
```

```
    String sql = "INSERT INTO pessoa (idpessoa, nome, logradouro, cidade, estado,  
telefone, email, cpf_cnpj, tipoPessoa) VALUES (?, ?, ?, ?, ?, ?, ?, ?, 'J')";
```

```
    ps = ConectorBD.getPrepared(sql, con);  
    ps.setInt(1, pessoajuridica.getId());  
    ps.setString(2, pessoajuridica.getNome());  
    ps.setString(3, pessoajuridica.getLogradouro());  
    ps.setString(4, pessoajuridica.getCidade());  
    ps.setString(5, pessoajuridica.getEstado());  
    ps.setString(6, pessoajuridica.getTelefone());  
    ps.setString(7, pessoajuridica.getEmail());  
    ps.setString(8, pessoajuridica.getCnpj()); // cnpj no campo cpf_cnpj  
  
    ps.execute();
```

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

```
public void alterar(int id, String nome, String logradouro, String cidade, String estado,  
String telefone, String email, String cnpj) throws Exception {
```

```
    Connection con = ConectorBD.getConnection();  
    PreparedStatement ps = null;
```

```
    String sql = "UPDATE pessoa SET nome=?, logradouro=?, cidade=?, estado=?,  
telefone=?, email=?, cpf_cnpj=? WHERE idpessoa=? AND tipoPessoa='J'";
```

```
    ps = ConectorBD.getPrepared(sql, con);  
    ps.setString(1, nome);  
    ps.setString(2, logradouro);  
    ps.setString(3, cidade);  
    ps.setString(4, estado);  
    ps.setString(5, telefone);  
    ps.setString(6, email);  
    ps.setString(7, cnpj);  
    ps.setInt(8, id);
```

```
    ps.execute();
```

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

```
public void excluir(int id) throws Exception {
```

```

        Connection con = ConectorBD.getConnection();

        PreparedStatement ps = null;

        String sql = "DELETE FROM pessoa WHERE idpessoa=? AND tipoPessoa='J'";

        ps = ConectorBD.getPrepared(sql, con);
        ps.setInt(1, id);

        ps.execute();

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

```

Classe CadastroBDTeste:

```

package cadastro;

import cadastro.model.PessoaFisicaDAO;
import cadastro.model.PessoaJuridicaDAO;
import cadastro.model.util.SequenceManager;
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaJuridica;
import java.util.List;

public class CadastroBDTeste {

```

```

public static void main(String[] args) {

    try {

        System.out.println("Incluindo pessoa física...");

        PessoaFisicaDAO pfDao = new PessoaFisicaDAO();

        long idPF = SequenceManager.getValue("seq_pessoa_id");

        PessoaFisica pf = new PessoaFisica(

            (int) idPF, "Théo", "Rua 9", "São Paulo", "SP",

            "9999-9999", "theo@sp.com", "999999999999"

        );

        pfDao.incluir(pf);

        System.out.println("Pessoa física incluída com sucesso!");

    } catch (Exception e) {

        e.printStackTrace();

    }

    try {

        System.out.println("Alterando pessoa física...");

        PessoaFisicaDAO pfDao = new PessoaFisicaDAO();

        pfDao.alterar(4, "Théo Alterado", "Rua 99", "São Paulo", "SP",

            "9999-9999", "theo@spfc.com", "888888888888"

        );

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

    } catch (Exception e) {

        e.printStackTrace();

    }

}

```

```
}
```

```
try {  
    System.out.println("Excluindo pessoa física...");  
    PessoaFisicaDAO pfDao = new PessoaFisicaDAO();  
    pfDao.excluir(4);  
    System.out.println("Pessoa física excluída com sucesso!");  
} catch (Exception e) {  
    e.printStackTrace();  
}
```

```
try {  
    System.out.println("--- Pessoas Físicas cadastradas ---");  
    PessoaFisicaDAO pfDao = new PessoaFisicaDAO();  
    List<PessoaFisica> pessoasFisicas = pfDao.getPessoas();  
    for (PessoaFisica p : pessoasFisicas) {  
        System.out.println("Id: " + p.getId());  
        System.out.println("Nome: " + p.getNome());  
        System.out.println("Logradouro: " + p.getLogradouro());  
        System.out.println("Cidade: " + p.getCidade());  
        System.out.println("Estado: " + p.getEstado());  
        System.out.println("Telefone: " + p.getTelefone());  
        System.out.println("E-mail: " + p.getEmail());  
        System.out.println("CPF: " + p.getCpf());  
        System.out.println();  
    }  
}
```

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

```
try {  
    System.out.println("Incluindo pessoa jurídica...");  
    PessoaJuridicaDAO pjDao = new PessoaJuridicaDAO();  
    long idPJ = SequenceManager.getValue("seq_pessoa_id");  
    PessoaJuridica pj = new PessoaJuridica(  
        (int) idPJ, "NINES Ltda", "Avenida Central, 999", "São Paulo", "SP",  
        "2222-2222", "contato@nines.com", "12345678000199"  
    );  
    pjDao.incluir(pj);  
    System.out.println("Pessoa jurídica incluída com sucesso!");  
} catch (Exception e) {  
    e.printStackTrace();  
}
```

```
try {  
    System.out.println("--- Pessoas Jurídicas cadastradas ---");  
    PessoaJuridicaDAO pjDao = new PessoaJuridicaDAO();  
    List<PessoaJuridica> pessoasJuridicas = pjDao.getPessoas();  
    for (PessoaJuridica p : pessoasJuridicas) {  
        System.out.println("Id: " + p.getId());  
        System.out.println("Razão Social: " + p.getNome());  
        System.out.println("Logradouro: " + p.getLogradouro());  
    }  
}
```



```

        System.out.println("Cidade: " + p.getCidade());
        System.out.println("Estado: " + p.getEstado());
        System.out.println("Telefone: " + p.getTelefone());
        System.out.println("E-mail: " + p.getEmail());
        System.out.println("CNPJ: " + p.getCnpj());
        System.out.println();
    }
} catch (Exception e) {
    e.printStackTrace();
}

try {
    System.out.println("Alterando pessoa jurídica...");
    PessoaJuridicaDAO pjDao = new PessoaJuridicaDAO();
    pjDao.alterar(5, "NINES Comércio Ltda", "Av. Paulista, 1500", "São Paulo", "SP",
        "3333-3333", "contato@ninescomercial.com", "8765432000188"
    );
    System.out.println("Pessoa jurídica alterada com sucesso!");
} catch (Exception e) {
    e.printStackTrace();
}

try {
    System.out.println("--- Pessoas Jurídicas atualizadas ---");
    PessoaJuridicaDAO pjDao = new PessoaJuridicaDAO();
    List<PessoaJuridica> pessoasJuridicasAtualizadas = pjDao.getPessoas();

```

```

        for (PessoaJuridica p : pessoasJuridicasAtualizadas) {

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

            System.out.println("Razão Social: " + p.getNome());

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

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

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

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

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

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

            System.out.println();

        }

    } catch (Exception e) {

        e.printStackTrace();

    }

}

try {

    System.out.println("Excluindo pessoa jurídica...");

    PessoaJuridicaDAO pjDao = new PessoaJuridicaDAO();

    pjDao.excluir(5);

    System.out.println("Pessoa jurídica excluída com sucesso!");

} catch (Exception e) {

    e.printStackTrace();

}

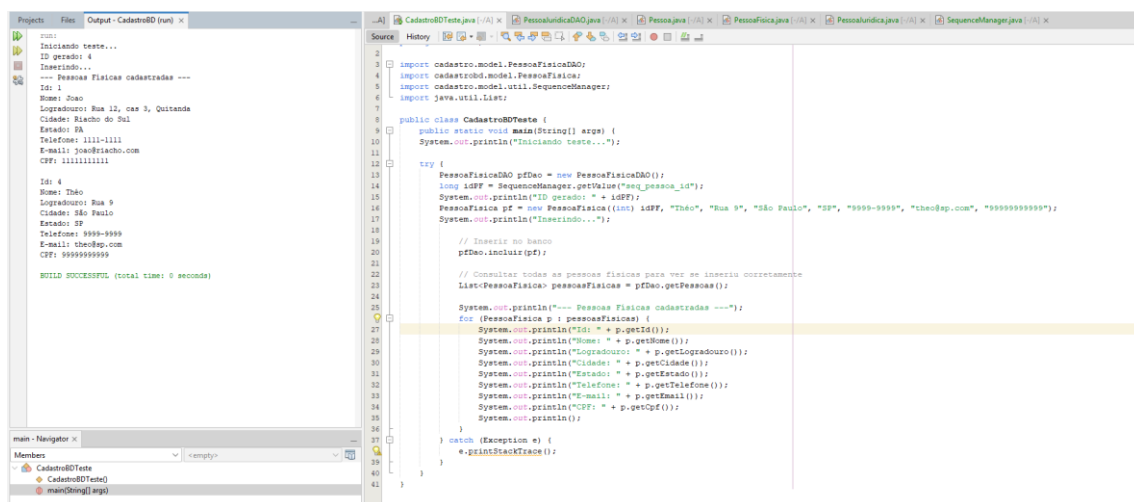
}

}

```

Resultado Procedimento 1:

Instanciar uma pessoa física e persistir no banco de dados:



The screenshot displays an IDE with two main panels. The left panel shows the output of a Java application, and the right panel shows the source code of the `CadastroBDTeste` class.

Output Panel (Left):

```
run:
Iniciando teste...
ID gerado: 4
Inserindo...
--- Pessoas Fisicas cadastradas ---
Id: 1
Nome: Joao
Logradouro: Rua 11, cas 3, Quitande
Cidade: Riacho do Sul
Estado: BA
Telefone: 1111-1111
E-mail: joao@riacho.com
CPF: 111111111111

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

Source Code Panel (Right):

```
1 import cadastro.modelo.PessoaFisicaDAO;
2 import cadastro.modelo.PessoaFisica;
3 import cadastro.modelo.util.SequenciaManager;
4 import java.util.List;
5
6 public class CadastroBDTeste {
7     public static void main(String[] args) {
8         System.out.println("Iniciando teste...");
9
10        try {
11            PessoaFisicaDAO pFdao = new PessoaFisicaDAO();
12            long idPF = SequenciaManager.getValue("seq_pessoa_id");
13            System.out.println("ID gerado: " + idPF);
14            PessoaFisica pf = new PessoaFisica(idPF, "Théo", "Rua 9", "São Paulo", "SP", "9999-9999", "theo@sp.com", "999999999999");
15            System.out.println("Inserindo...");
16
17            // Inserir no banco
18            pFdao.incluir(pf);
19
20            // Consultar todas as pessoas físicas para ver se inseriu corretamente
21            List<PessoaFisica> pessoasFisicas = pFdao.getPessoas();
22
23            System.out.println("--- Pessoas Fisicas cadastradas ---");
24            for (PessoaFisica p : pessoasFisicas) {
25                System.out.println("Id: " + p.getId());
26                System.out.println("Nome: " + p.getNome());
27                System.out.println("Logradouro: " + p.getLogradouro());
28                System.out.println("Cidade: " + p.getCidade());
29                System.out.println("Estado: " + p.getEstado());
30                System.out.println("Telefone: " + p.getTelefone());
31                System.out.println("E-mail: " + p.getEmail());
32                System.out.println("CPF: " + p.getCpf());
33            }
34        } catch (Exception e) {
35            e.printStackTrace();
36        }
37    }
38 }
```

SQLQuery1.sql - loca...ESS.loja (loja (60))*

```
use loja;
select * from pessoa;
```

159 %

Resultados Mensagens

	idpessoa	nome	logradouro	cidade	estado	telefone	email	cpf_cnpj	tipoPessoa
1	1	Joao	Rua 12, cas 3, Quitanda	Riacho do Sul	PA	1111-1111	joao@riacho.com	11111111111	F
2	2	JJC	Rua 11, Centro	Riacho do Norte	PA	1212-1212	jic@riacho.com	22222222222	J
3	4	Théo	Rua 9	São Paulo	SP	9999-9999	theo@sp.com	99999999999	F

Alterar os dados da pessoa física no banco:

Projects Files Output - Cadastro00 (run) x

```
run:
Iniciando teste...
Alterando...
--- Pessoas Fisicas cadastradas ---
Id: 1
Nome: Joao
Logradouro: Rua 12, cas 3, Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
E-mail: joao@riacho.com
CPF: 11111111111

Id: 4
Nome: Theo Alterado
Logradouro: Rua 99
Cidade: São Paulo
Estado: SP
Telefone: 9999-9999
E-mail: theo@sp.com
CPF: 99999999999

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

Source History

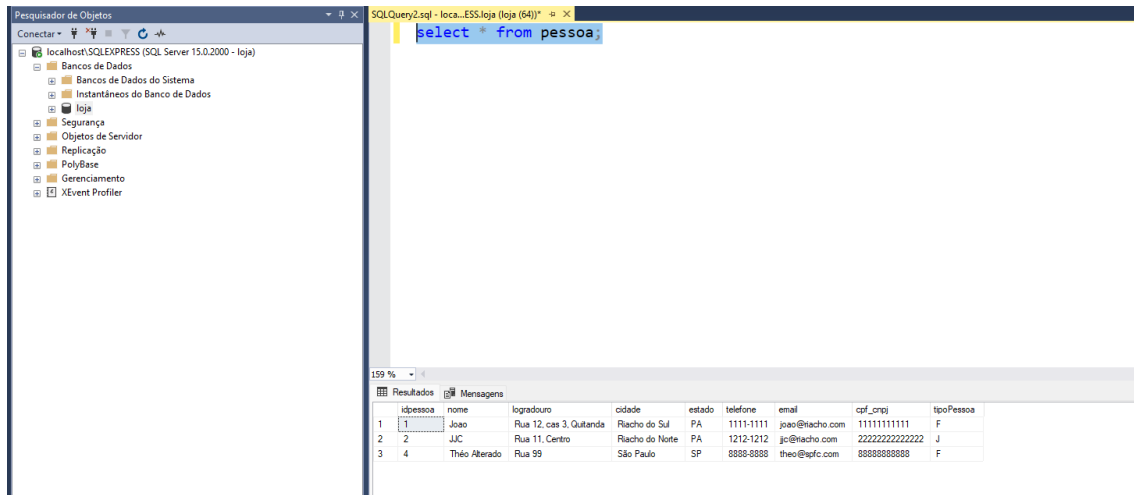
```
import java.util.List;

public class Cadastro00Teste {
    public static void main(String[] args) {
        System.out.println("Iniciando teste...");
    }
}

try {
    PessoaFisicaDAO pfDao = new PessoaFisicaDAO();

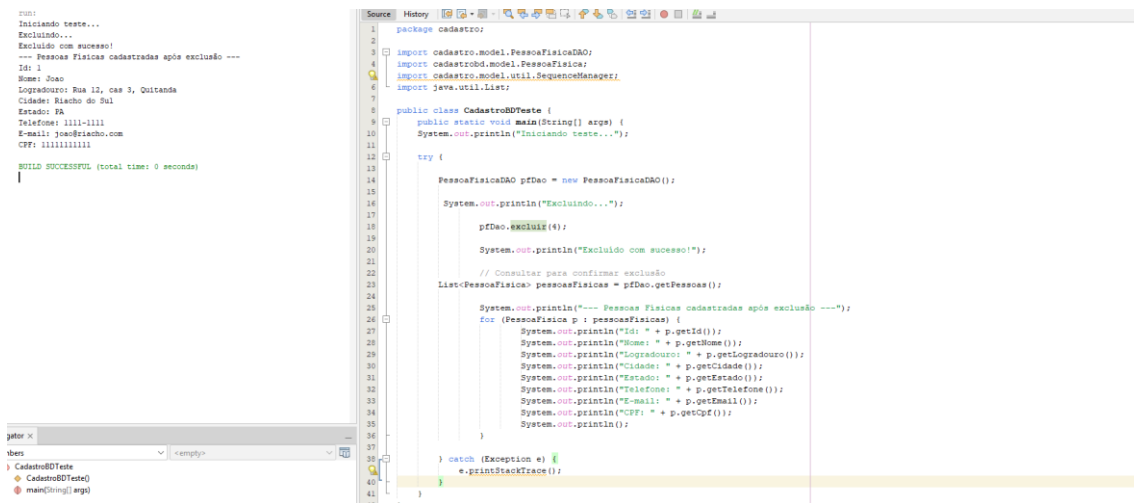
    System.out.println("Alterando...");
    pfDao.alterar(4, "Theo Alterado", "Rua 99", "São Paulo", "SP", "9999-9999", "theo@sp.com", "99999999999");

    // Listar todas as pessoas fisicas para confirmar alteração
    List<PessoaFisica> pessoasFisicas = pfDao.getPessoas();
    System.out.println("--- Pessoas Fisicas cadastradas ---");
    for (PessoaFisica p : pessoasFisicas) {
        System.out.println("Id: " + p.getId());
        System.out.println("Nome: " + p.getNome());
        System.out.println("Logradouro: " + p.getLogradouro());
        System.out.println("Cidade: " + p.getCidade());
        System.out.println("Estado: " + p.getEstado());
        System.out.println("Telefone: " + p.getTelefone());
        System.out.println("E-mail: " + p.getEmail());
        System.out.println("CPF: " + p.getCpf());
        System.out.println();
    }
} catch (Exception e) {
    e.printStackTrace();
}
```



(Nos prints já possuem a consulta feita no console e banco)

Excluir pessoa física criada:



Instanciar pessoa Jurídica:

Terminal Output:

```
INFO:
Iniciando teste...
Incluindo pessoa juridica...
--- Pessoas Juridicas cadastradas ---
Id: 2
Razão Social: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
E-mail: jc@riacho.com
CPF: 222222222222222222

Id: 5
Razão Social: NINES Ltda
Logradouro: Avenida Central, 999
Cidade: São Paulo
Estado: SP
Telefone: 2222-2222
E-mail: contato@nines.com
CPF: 12345678000199

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

Source Code:

```
13
14 try {
15     System.out.println("Incluindo pessoa juridica...");
16
17     PessoaJuridicaDAO pJDao = new PessoaJuridicaDAO();
18     long idPJ = SequenceManager.getValue("seq_pessoa_id");
19
20     PessoaJuridica pJ = new PessoaJuridica(
21         (int) idPJ,
22         "NINES Ltda",
23         "Avenida Central, 999",
24         "São Paulo",
25         "SP",
26         "2222-2222",
27         "contato@nines.com",
28         "12345678000199"
29     );
30
31     pJDao.incluir(pJ);
32
33     System.out.println("--- Pessoas Juridicas cadastradas ---");
34     List<PessoaJuridica> pessoasJuridicas = pJDao.getPessoas();
35     for (PessoaJuridica p : pessoasJuridicas) {
36         System.out.println("Id: " + p.getId());
37         System.out.println("Razão Social: " + p.getNome());
38         System.out.println("Logradouro: " + p.getLogradouro());
39         System.out.println("Cidade: " + p.getCidade());
40         System.out.println("Estado: " + p.getEstado());
41         System.out.println("Telefone: " + p.getTelefone());
42         System.out.println("E-mail: " + p.getEmail());
43         System.out.println("CPF: " + p.getCnpj());
44     }
45 } catch (Exception e) {
46     e.printStackTrace();
47 }
```

SQL Query:

```
SELECT * FROM pessoa WHERE tipoPessoa = 'J';
```

Database Explorer:

- localhost\SQLEXPRESS (SQL Server 15.0.2000 - loja)
- Bancos de Dados
- Bancos de Dados do Sistema
- Instantâneos do Banco de Dados
- loja
- Segurança
- Objetos de Servidor
- Replicação
- PolyBase
- Gerenciamento
- XEvent Profiler

Results:

IdPessoa	nome	logradouro	cidade	estado	telefone	email	cpf_cnpj	tipoPessoa
2	JJC	Rua 11, Centro	Riacho do Norte	PA	1212-1212	jc@riacho.com	222222222222222222	J
5	NINES Ltda	Avenida Central, 999	São Paulo	SP	2222-2222	contato@nines.com	12345678000199	J

Alterar Pessoa Jurídica:

```

1  Iniciando teste...
2  Incluindo pessoa jurídica...
3  --- Pessoas Jurídicas atualizadas ---
4  Id: 5
5  Razão Social: NINES Comércio Ltda
6  Logradouro: Rua 11, Centro
7  Cidade: Riacho do Norte
8  Estado: PA
9  Telefone: 1212-1212
10 E-mail: jc@riacho.com
11 CNPJ: 2222222222222
12
13 Id: 5
14 Razão Social: NINES Comércio Ltda
15 Logradouro: Av. Paulista, 1500
16 Cidade: São Paulo
17 Estado: SP
18 Telefone: 3333-3333
19 E-mail: contato@ninescomercial.com
20 CNPJ: 8765432000188
21
22 BUILD SUCCESSFUL (total time: 0 seconds)

```

```

1 package cadastro;
2
3 import cadastro.model.PessoaFisicaDAO;
4 import cadastro.model.PessoaJuridicaDAO;
5 import cadastro.model.PessoaFisica;
6 import cadastro.model.util.SequenceManager;
7 import cadastro.model.PessoaJuridica;
8 import java.util.List;
9
10 public class CadastroBDTeste {
11     public static void main(String[] args) {
12         System.out.println("Iniciando teste...");
13
14         try {
15             System.out.println("Incluindo pessoa jurídica...");
16
17             PessoaJuridicaDAO pJDao = new PessoaJuridicaDAO();
18             pJDao.alterar(
19                 5,
20                 "NINES Comércio Ltda",
21                 "Av. Paulista, 1500",
22                 "São Paulo",
23                 "SP",
24                 "3333-3333",
25                 "contato@ninescomercial.com",
26                 "8765432000188"
27             );
28
29             // Listar novamente para verificar alteração
30             System.out.println("--- Pessoas Jurídicas atualizadas ---");
31             List<PessoaJuridica> pessoasJuridicasAtualizadas = pJDao.getPessoas();
32             for (PessoaJuridica p : pessoasJuridicasAtualizadas) {
33                 System.out.println("Id: " + p.getId());
34                 System.out.println("Razão Social: " + p.getNome());
35                 System.out.println("Logradouro: " + p.getLogradouro());
36                 System.out.println("Cidade: " + p.getCidade());
37                 System.out.println("Estado: " + p.getEstado());
38                 System.out.println("Telefone: " + p.getTelefone());
39                 System.out.println("E-mail: " + p.getEmail());
40                 System.out.println("CNPJ: " + p.getCnpj());
41                 System.out.println();
42             }
43
44             } catch (Exception e) {
45                 e.printStackTrace();
46             }
47         }
48     }
49 }

```

```

SELECT * FROM pessoa WHERE tipoPessoa = 'J'

```

159 %

Resultados Mensagens

	idPessoa	nome	logradouro	cidade	estado	telefone	email	cpf_cnpj	tipoPessoa
1	2	JUC	Rua 11, Centro	Riacho do Norte	PA	1212-1212	jc@riacho.com	222222222222222	J
2	5	NINES Comércio Ltda	Av. Paulista, 1500	São Paulo	SP	3333-3333	contato@ninescomercial.com	8765432000188	J

(Nos prints já possuem a consulta feita no console e banco)

Excluir pessoa jurídica criada:

```

FUN:
Iniciando teste...
Excluindo pessoa jurídica...
Pessoa jurídica excluída com sucesso.
BUILD SUCCESSFUL (total time: 0 seconds)

```

```

1 package cadastro;
2
3 import cadastro.model.PessoaFisicaDAO;
4 import cadastro.model.PessoaJuridicaDAO;
5 import cadastrobd.model.PessoaFisica;
6 import cadastro.model.util.SequenceManager;
7 import cadastrobd.model.PessoaJuridica;
8 import java.util.List;
9
10 public class CadastroBDTeste {
11     public static void main(String[] args) {
12         System.out.println("Iniciando teste...");
13
14         try {
15             PessoaJuridicaDAO pjDao = new PessoaJuridicaDAO();
16             System.out.println("Excluindo pessoa jurídica...");
17
18             pjDao.excluir((int) 5);
19
20             System.out.println("Pessoa jurídica excluída com sucesso.");
21
22         } catch (Exception e) {
23             e.printStackTrace();
24         }
25     }
26 }

```

Pesquisador de Objetos

Conectar

localhost\SQLEXPRESS (SQL Server 15.0.2000 - loja)

Bancos de Dados

Bancos de Dados do Sistema

Instantâneos do Banco de Dados

loja

Segurança

Objetos de Servidor

Replicação

PolyBase

Gerenciamento

XEvent Profiler

SQLQuery2.sql - loca...ESS.loja (loja (64))

SELECT * FROM pessoa WHERE tipoPessoa = 'J';

159 %

Resultados

Mensagens

	idpessoa	nome	logradouro	cidade	estado	telefone	email	cpf_cnpj	tipoPessoa
1	2	JJC	Rua 11, Centro	Riacho do Norte	PA	1212-1212	jic@hachos.com	222222222222222222	J

Procedimento 2: Alimentando a base:

Código:

Classe CadastroBDTeste2:

```
import cadastro.model.PessoaFisicaDAO;
```

```
import cadastro.model.PessoaJuridicaDAO;
```

```
import cadastro.model.util.SequenceManager;
```

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



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

```
import java.util.List;
```

```
import java.util.Scanner;
```

```
public class CadastroBDTeste2 {
```

```
    public static void main(String[] args) throws Exception {
```

```
        Scanner scan = new Scanner(System.in);
```

```
        String escolha;
```

```
        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("=====");
```

```
            escolha = scan.next();
```

```
            SequenceManager seq = new SequenceManager();
```

```
            switch(escolha) {
```

```
// Incluir

case "1":

    do {

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

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


        escolha = scan.next();

        scan.nextLine();


        switch (escolha.toUpperCase()) {

            case "F":

                System.out.println("Insira os dados... ");

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

                String nome = scan.nextLine();

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

                String logradouro = scan.nextLine();

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

                String cidade = scan.nextLine();

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

                String estado = scan.nextLine();

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

                String telefone = scan.nextLine();

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

                String email = scan.nextLine();

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

                String cpf = scan.nextLine();
```

```
        PessoaFisica pessoaIncluir = new PessoaFisica((int)
seq.getValue("seq_pessoa_id"),nome, logradouro,
        cidade, estado, telefone,email,cpf);

        PessoaFisicaDAO pessoaPF = new PessoaFisicaDAO();
        pessoaPF.incluir(pessoaIncluir);

        System.out.println("Inclusao realizada com sucesso!");
        break;
```

```
case "J":
```

```
        System.out.println("Insira os dados... ");
        System.out.print("Nome: ");
        String nomej = scan.nextLine();
        System.out.print("Logradouro: ");
        String logradouroj = scan.nextLine();
        System.out.print("Cidade: ");
        String cidadej = scan.nextLine();
        System.out.print("Estado: ");
        String estadoj = scan.nextLine();
        System.out.print("Telefone: ");
        String telefonej = scan.nextLine();
        System.out.print("Email: ");
        String emailj = scan.nextLine();
        System.out.print("CNPJ: ");
        String cnpj = scan.nextLine();
```

```
        PessoaJuridica pessoaJIncluir = new PessoaJuridica((int)
seq.getValue("seq_pessoa_id"),nomej,
        logradouroj,cidadej, estadoj, telefonej,emailj,cnpj);
```

```
PessoaJuridicaDAO pessoaPJ = new PessoaJuridicaDAO();  
pessoaPJ.incluir(pessoaJIncluir);
```

```
System.out.println("Inclusao realizada com sucesso!");  
break;
```

```
case "M":  
    break;
```

```
default:  
    System.out.println("Opcao invalida.");  
    break;
```

```
}
```

```
} while (!escolha.equalsIgnoreCase("M"));  
break;
```

```
// Alterar
```

```
case "2":
```

```
do {
```

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

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

```
    escolha = scan.next();
```

```
    scan.nextLine();
```

```
    switch (escolha.toUpperCase()) {
```

```
case "F":

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

    int idPessoaFisica = scan.nextInt();

    scan.nextLine();


    PessoaFisica pessoaFisicaLocalizada = new
PessoaFisicaDAO().getPessoa(idPessoaFisica);

    PessoaFisicaDAO pessoaFisicaLocalizadaAlterar = new
PessoaFisicaDAO();


    if (pessoaFisicaLocalizada != null) {

        pessoaFisicaLocalizada.exibir();


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

        System.out.print("Novo nome: ");

        String novoNome = scan.nextLine();


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

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

        String novoLogradouro = scan.nextLine();


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

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

        String novoCidade = scan.nextLine();
```

```

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

        System.out.print("Novo Estado: ");

        String novoEstado = scan.nextLine();


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

        System.out.print("Novo Telefone: ");

        String novoTelefone = scan.nextLine();


        System.out.println("Email: " + pessoaFisicaLocalizada.getEmail());

        System.out.print("Novo Email: ");

        String novoEmail = scan.nextLine();


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

        System.out.print("Novo CPF: ");

        String novoCPF = scan.nextLine();


        pessoaFisicaLocalizadaAlterar.alterar( idPessoaFisica,novoCPF,
        novoNome, novoLogradouro, novoCidade,

        novoEstado, novoTelefone, novoEmail );


        System.out.println("Pessoa alterada com sucesso!");
    } else

        System.out.println("Pessoa nao localizada! ");

    break;

case "J":

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

```

```
int idPessoaJuridica = scan.nextInt();  
  
scan.nextLine();
```

```
PessoaJuridica pessoaJuridicaLocalizada = new  
PessoaJuridicaDAO().getPessoa(idPessoaJuridica);
```

```
PessoaJuridicaDAO pessoaJuridicaLocalizadaAlterar = new  
PessoaJuridicaDAO();
```

```
if (pessoaJuridicaLocalizada != null) {  
  
    pessoaJuridicaLocalizada.exibir();
```

```
    System.out.println("Nome atual: " +  
pessoaJuridicaLocalizada.getNome());
```

```
    System.out.print("Novo nome: ");
```

```
    String novoNome = scan.nextLine();
```

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

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

```
    String novoLogradouro = scan.nextLine();
```

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

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

```
    String novoCidade = scan.nextLine();
```

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

```
    System.out.print("Novo Estado: ");
```

```
    String novoEstado = scan.nextLine();
```

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

        System.out.print("Novo Telefone: ");

        String novoTelefone = scan.nextLine();


        System.out.println("Email: " + pessoaJuridicaLocalizada.getEmail());

        System.out.print("Novo Email: ");

        String novoEmail = scan.nextLine();


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

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

        String novoCNPJ = scan.nextLine();


        pessoaJuridicaLocalizadaAlterar.alterar( idPessoaJuridica, novoCNPJ,
        novoNome, novoLogradouro, novoCidade,
                novoEstado, novoTelefone, novoEmail);


        System.out.println("Pessoa alterada com sucesso!");
    } else

        System.out.println("Pessoa nao localizada!");

    break;


case "M":

    break;


default:

    System.out.println("Opcao invalida.");
```



```

        break;
    }
} while (!escolha.equalsIgnoreCase("M"));
break;

// EXCLUIR
case "3":
    do {
        System.out.println("=====");
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica | M - Menu");

        escolha = scan.next();
        scan.nextLine();

        switch (escolha.toUpperCase()) {

            case "F":
                System.out.println("Digite o ID da pessoa: ");
                int idPessoaFisica = scan.nextInt();

                PessoaFisica pessoaFisicaLocalizada = new
PessoaFisicaDAO().getPessoa(idPessoaFisica);

                PessoaFisicaDAO pessoaFisicaLocalizadaExcluir = new
PessoaFisicaDAO();

                if (pessoaFisicaLocalizada != null) {
                    pessoaFisicaLocalizada.exibir();
                    pessoaFisicaLocalizadaExcluir.excluir(idPessoaFisica);
                }
            }
        }
    } while (true);
}

```

```

        System.out.println("Pessoa excluida com sucesso!");
    } else
        System.out.println("Pessoa nao localizada!");
    break;

case "J":
    System.out.println("Digite o ID da pessoa: ");
    int idPessoaJuridica = scan.nextInt();

    PessoaJuridica pessoaJuridicaLocalizada = new
PessoaJuridicaDAO().getPessoa(idPessoaJuridica);

    PessoaJuridicaDAO pessoaJuridicaLocalizadaExcluir = new
PessoaJuridicaDAO();

    if (pessoaJuridicaLocalizada != null) {
        pessoaJuridicaLocalizada.exibir();
        pessoaJuridicaLocalizadaExcluir.excluir(idPessoaJuridica);

        System.out.println("Pessoa excluida com sucesso!");
    } else
        System.out.println("Pessoa nao localizada!");
    break;

case "M":
    break;

default:
    System.out.println("Opcao invalida.");

```

```

        break;
    }

} while (!escolha.equalsIgnoreCase("M"));

break;

// obter pelo Id
case "4":
    do {
        System.out.println("=====");
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica | M - Menu");

        escolha = scan.next();
        scan.nextLine();

        switch (escolha.toUpperCase()) {

            case "F":
                System.out.println("Digite o ID da pessoa: ");
                int idPessoaFisica = scan.nextInt();

                PessoaFisica pessoaFisicaLocalizada = new
PessoaFisicaDAO().getPessoa(idPessoaFisica);

                if (pessoaFisicaLocalizada != null) {
                    System.out.println("Pessoa localizada!");
                    pessoaFisicaLocalizada.exibir();
                } else

```

```
        System.out.println("Pessoa nao localizada!");
        break;

    case "J":

        System.out.println("Digite o ID da pessoa: ");
        int idPessoaJuridica = scan.nextInt();

        PessoaJuridica pessoaJuridicaLocalizada = new
PessoaJuridicaDAO().getPessoa(idPessoaJuridica);

        if (pessoaJuridicaLocalizada != null) {
            System.out.println("Pessoa localizada!");
            pessoaJuridicaLocalizada.exibir();
        } else
            System.out.println("Pessoa nao localizada!");
        break;

    case "M":
        break;

    default:
        System.out.println("Opcao invalida.");
        break;
    }

} while (!escolha.equalsIgnoreCase("M"));
break;
```

```
//obterTodos

case "5":

    do {

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

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


        escolha = scan.next();

        scan.nextLine();


        switch (escolha.toUpperCase()) {

            case "F":

                System.out.println("Pessoas fisicas:");

                PessoaFisicaDAO pessoasFisica = new PessoaFisicaDAO();

                List<PessoaFisica> resultado = pessoasFisica.getPessoas();

                for (PessoaFisica pessoaFisica : resultado) {

                    pessoaFisica.exibir();

                }

                break;

            case "J":

                System.out.println("Pessoas juridicas:");

                PessoaJuridicaDAO pessoasJuridica = new PessoaJuridicaDAO();

                List<PessoaJuridica> resultado2 = pessoasJuridica.getPessoas();

                for (PessoaJuridica pessoaJuridica : resultado2) {

                    pessoaJuridica.exibir();

                }

                break;

        }

    }

}
```

```
        case "M":
            break;

        default:
            System.out.println("Opcao invalida");
            break;
    }

    } while (!escolha.equalsIgnoreCase("M"));

    break;

case "0":

    System.out.println("Sistema Finalizado com sucesso.");

    break;

    default:

        System.out.println("Opcao invalida");

        break;

    }

} while (!escolha.equals("0"));

scan.close();

}

}
```

Análise e conclusão:

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

A persistência de dados refere-se à capacidade de armazenar informações de forma que elas permaneçam disponíveis mesmo após o término de um programa. As duas formas mais comuns são a persistência em arquivo e a persistência em banco de dados.

Como o operador lambda simplificou a impressão de valores no Java?

Antes do Java 8, para imprimir cada item de uma lista, você precisava de um for loop, que era um pouco mais "falante".

Com o operador lambda (->), o Java ficou mais direto. Agora, você pode dizer à lista: "Para cada item que você tem, apenas imprima-o".

Por que métodos chamados diretamente do main precisam ser static?

Quando seu programa Java começa, o método main é o primeiro a ser executado. Nesse momento, a Máquina Virtual Java (JVM) ainda não criou nenhum objeto da sua classe.

Métodos marcados como static são especiais: eles pertencem à classe em si, não a um objeto específico. Pense neles como algo que está pronto para usar desde o início, sem precisar "montar" nada.

Então, para o main poder chamar um método logo de cara, esse método precisa ser static, porque ele não tem um objeto para chamá-lo.

