



Estácio

## Universidade Estácio de Sá

DESENVOLVIMENTO FULL STACK- TURMA - 9001

Disciplina: RPG0016 - Back-end Sem Banco Não Tem!

Semestre Letivo: 2024.3

Repositorio Git: Leowaawe/Mundo-3 (github.com)

LEONARDO LOPES - MATRICULA: 202302731406

### 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

### Objetivos 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;
import java.io.Serializable;

/**
 *
 * @author grego
 */
public class Pessoa implements Serializable{
    private int id;
    private String nome;
    private String logradouro;
    private String cidade;
    private String estado;
    private String telefone;
    private String email;

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

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

    public void setLogradouro(String logradouro) {
        this.logradouro = logradouro;
    }
}
```

```

    }

    public void setCidade(String cidade) {
        this.cidade = cidade;
    }

    public void setEstado(String estado) {
        this.estado = estado;
    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    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;
    }

    public void exibir(){
        System.out.print("id: "+this.id + "\n" + "Nome: " + this.nome + "\n"
+
        "logradouro: "+this.logradouro+"\n"+"cidade: "+this.cidade+"\n"+
        "estado: "+this.estado+"\n" + "telefone: " + this.telefone + "\n"+
        "email: " + this.email );
    }
}

```

Classe PessoaFisica

package cadastrobd.model;

```

import java.io.Serializable;
/**
 *
 * @author grego
 */
public class PessoaFisica extends Pessoa implements Serializable {

    private String cpf;

    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;
    }

    public void exibir(){
        System.out.print("id: "+ getId()+ "\n" + "Nome: " + getNome() + "\n"
+
        "logradouro: "+getLogradouro()+"\n"+"cidade: "+getCidade()+"\n"+
        "estado: "+getEstado()+"\n" + "telefone: " + getTelefone() + "\n"+
"email: " + getEmail() + "\n"+
        "CPF: "+this.cpf + "\n");
    }
}

```

#### Classe PessoaJuridica

```

package cadastrabd.model;
import java.io.Serializable;

/**
 *
 * @author grego
 */
public class PessoaJuridica extends Pessoa implements Serializable{

    private String cnpj;

    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;
        }

        public void exibir(){
            System.out.print("id: "+ getId()+ "\n" + "Nome: " + getNome() + "\n"
+
            "logradouro: "+getLogradouro()+"\n"+"cidade: "+getCidade()+"\n"+
            "estado: "+getEstado()+"\n" + "telefone: " + getTelefone() + "\n"+
            "email: " + getEmail() + "\n"+
            "CNPJ: "+this.cnpj + "\n");
        }
    }
}

```

### Classe ConectorBD

```

package cadastro.model.util;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.swing.JOptionPane;

/**
 *
 * @author grego
 */

public class ConectorBD {

    Connection conn = null;

    //Metodo para conectar java com SQLServer

    public Connection getConnection() throws Exception{
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        Connection conn =
DriverManager.getConnection("jdbc:sqlserver://localhost\\MSSQLSERVER2019E:143
3;databaseName=Loja;encrypt=true;trustServerCertificate=true",
            "loja", "loja");
        return conn;
    }

    public void closeConnection()throws Exception{
        getConnection().close();
        //JOptionPane.showMessageDialog(null, "Conexao finalizada");
    }

    public PreparedStatement getPrepared(String sql) throws Exception {
        PreparedStatement ps = getConnection().prepareStatement(sql);
        return ps;
    }
}

```

```

public void closeStatement(String sql)throws Exception{
    getPrepared(sql).close();
    //JOptionPane.showMessageDialog(null, "Statement finalizado");
}

public ResultSet getSelect(PreparedStatement ps) throws Exception {
    ResultSet rs = ps.executeQuery();
    //ResultSet rs = getConnection().createStatement().executeQuery("");
    return rs;
}

public void closeResult(PreparedStatement ps)throws Exception{
    getSelect(ps).close();
    //JOptionPane.showMessageDialog(null, "ResultSet finalizado");
}
}

```

### Classe SequenceManager

```

package cadastro.model.util;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;

/**
 *
 * @author grego
 */
public class SequenceManager {

    public int getValue(String sequencia)throws Exception{
        int resultado = 0;
        Connection con =
DriverManager.getConnection("jdbc:sqlserver://localhost\\MSSQLSERVER2019E:143
3;databaseName=Loja;encrypt=true;trustServerCertificate=true",
        "loja", "loja");
        String sql = "SELECT NEXT VALUE FOR "+sequencia+" as proximoId";
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery(sql);
        while(rs.next())
            resultado = rs.getInt("proximoId");
        return resultado;
    }
}

```

### Classe PessoaFisicaDAO

```

package cadastro.model;
import cadastrobd.model.PessoaFisica;
import java.util.ArrayList;
import java.util.List;
import cadastro.model.util.ConectorBD;
import com.sun.jdi.connect.spi.Connection;
import java.sql.ResultSet;

```

```

import java.sql.PreparedStatement;

/**
 *
 * @author grego
 */
public class PessoaFisicaDAO {

    public ConectorBD connection = new ConectorBD();

    public PessoaFisica getPessoa(int id) throws Exception {
        PessoaFisica pessoa = null;
        String sql = "select *\n" +
            "from pessoa, pessoa_fisica\n" +
            "where pessoa.idpessoa = "+ id + "AND " +
            "pessoa.idpessoa = pessoa_fisica.idpessoa;";
        PreparedStatement ps = connection.getPrepared(sql);
        ResultSet 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"));
            connection.closeConnection();
            //connection.closeResult(ps);
            connection.closeStatement(sql);
        } return pessoa;
    }

    public List<PessoaFisica> getPessoas() throws Exception{
        List<PessoaFisica> lista = new ArrayList<>();
        String sql = "select *\n" +
            "from pessoa, pessoa_fisica\n" +
            "where pessoa.idpessoa = pessoa_fisica.idpessoa;";
        PreparedStatement ps = connection.getPrepared(sql);
        ResultSet resultado = ps.executeQuery();
        while(resultado.next()){
            //System.out.println(resultado.getString(5));
            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")));
            connection.closeConnection();
            //connection.closeResult(ps);
            connection.closeStatement(sql);
        } return lista;
    }
}

```

```

        public void incluir(PessoaFisica pessoafisica)throws Exception{
            String sqlfisica = "insert into pessoa_fisica (idpessoa, cpf) values
            (?,?)";
            String sqlpessoa = "insert into pessoa (idpessoa,nome,logradouro,
            cidade,"
                + "estado, telefone, email ) values (?,?,?,?,?,?,?)";
            PreparedStatement ps = connection.getPrepared(sqlfisica);
            PreparedStatement ps1 = connection.getPrepared(sqlpessoa);
            //ResultSet resultado = ps.executeQuery();
            ps.setInt(1, pessoafisica.getId());
            ps.setString(2, pessoafisica.getCpf());
            ps1.setInt(1, pessoafisica.getId());
            ps1.setString(2, pessoafisica.getNome());
            ps1.setString(3, pessoafisica.getLogradouro());
            ps1.setString(4, pessoafisica.getCidade());
            ps1.setString(5, pessoafisica.getEstado());
            ps1.setString(6, pessoafisica.getTelefone());
            ps1.setString(7, pessoafisica.getEmail());
            ps1.execute();
            ps.execute();
            connection.closeConnection();
            //connection.closeResult(ps);
            connection.closeStatement(sqlfisica);
        }

        public void alterar(int id, String cpf, String nome, String logradouro,
            String cidade, String estado,String telefone, String email)throws
            Exception{
            PessoaFisica pessoa = getPessoa(id);
            String sqlfisica = "UPDATE pessoa_fisica SET cpf=? where idpessoa =
            "+id;
            String sqlpessoa = "UPDATE pessoa SET nome=?, logradouro=?,
            cidade=?,"
                + "estado=?, telefone=?, email=? WHERE idpessoa= "+id;
            PreparedStatement ps = connection.getPrepared(sqlfisica);
            PreparedStatement ps1 = connection.getPrepared(sqlpessoa);
            if(cpf.equals("")){
                ps.setString(1, pessoa.getCpf());
            } else{
                ps.setString(1, cpf);
            }

            if(nome.equals("")){
                ps1.setString(1, pessoa.getNome());
            } else{
                ps1.setString(1, nome);
            }

            if(logradouro.equals("")){
                ps1.setString(2, pessoa.getLogradouro());
            } else{
                ps1.setString(2, logradouro);
            }

            if(cidade.equals("")){
                ps1.setString(3, pessoa.getCidade());
            } else{
                ps1.setString(3, cidade);
            }
        }
    }

```



```

        if(estado.equals("")){
            ps1.setString(4, pessoa.getEstado());
        } else{
            ps1.setString(4, estado);
        }

        if(telefone.equals("")){
            ps1.setString(5, pessoa.getTelefone());
        } else{
            ps1.setString(5, telefone);
        }
        if(email.equals("")){
            ps1.setString(6, pessoa.getEmail());
        } else{
            ps1.setString(6, email);
        }
        ps.execute();
        ps1.execute();
        connection.closeConnection();
        //connection.closeResult(ps);
        connection.closeStatement(sqlfisica);
    }

    public void excluir(int id)throws Exception{
        String sqlfisica = "DELETE FROM pessoa_fisica WHERE idpessoa="+id;
        String sqlpessoa = "DELETE FROM pessoa WHERE idpessoa="+id;
        PreparedStatement ps = connection.getPrepared(sqlfisica);
        PreparedStatement ps1 = connection.getPrepared(sqlpessoa);
        ps.execute();
        ps1.execute();
        connection.closeConnection();
        //connection.closeResult(ps);
        connection.closeStatement(sqlfisica);
    }
}

```

### Classe PessoaJuridicaDAO

```

package cadastro.model;

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

/**
 *
 * @author grego
 */
public class PessoaJuridicaDAO {

    public ConectorBD connection = new ConectorBD();

    public PessoaJuridica getPessoa(int id)throws Exception {

```

```

PessoaJuridica pessoa = null;
String sql = "select *\n" +
            "from pessoa, pessoa_juridica\n" +
            "where pessoa.idpessoa = "+ id + "AND " +
            "pessoa.idpessoa = pessoa_juridica.idpessoa;";
PreparedStatement ps = connection.getPreparedStatement(sql);

ResultSet 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("cnpj"));
    connection.closeConnection();
    //connection.closeResult(ps);
    connection.closeStatement(sql);
} return pessoa;
}

public List<PessoaJuridica> getPessoas() throws Exception{
    List<PessoaJuridica> lista = new ArrayList<>();
    String sql = "select *\n" +
        "from pessoa, pessoa_juridica\n" +
        "where pessoa.idpessoa = pessoa_juridica.idpessoa;";
    PreparedStatement ps = connection.getPreparedStatement(sql);
    ResultSet resultado = ps.executeQuery();
    while(resultado.next()){
        //System.out.println(resultado.getString(5));
        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("cnpj")));
        connection.closeConnection();
        //connection.closeResult(ps);
        connection.closeStatement(sql);
    } return lista;
}

public void incluir(PessoaJuridica pessoajuridica) throws Exception{
    String sqljuridica = "insert into pessoa_juridica (idpessoa, cnpj)
values (?,?)";
    String sqlpessoa = "insert into pessoa (idpessoa,nome,logradouro,
cidade,"
        + "estado, telefone, email ) values (?,?,,?,,?,,?)";
    PreparedStatement ps = connection.getPreparedStatement(sqljuridica);
    PreparedStatement ps1 = connection.getPreparedStatement(sqlpessoa);
    //ResultSet resultado = ps.executeQuery();
    ps.setInt(1, pessoajuridica.getId());
    ps.setString(2, pessoajuridica.getCnpj());
    ps1.setInt(1, pessoajuridica.getId());

```

```

        ps1.setString(2, pessoajuridica.getLogradouro());
        ps1.setString(4, pessoajuridica.getCidade());
        ps1.setString(5, pessoajuridica.getEstado());
        ps1.setString(6, pessoajuridica.getTelefone());
        ps1.setString(7, pessoajuridica.getEmail());
        ps1.execute();

        ps.execute();
        connection.closeConnection();
        //connection.closeResult(ps);
        connection.closeStatement(sqljuridica);
    }

    public void alterar(int id, String cnpj, String nome, String logradouro,
        String cidade, String estado,String telefone, String email)throws
    Exception{
        PessoaJuridica pessoa = getPessoa(id);
        String sqljuridica = "UPDATE pessoa_juridica SET cnpj=? where
idpessoa = "+id;
        String sqlpessoa = "UPDATE pessoa SET nome=?, logradouro=?,
cidade=?, "
            + "estado=?, telefone=?, email=? WHERE idpessoa= "+id;
        PreparedStatement ps = connection.getPrepared(sqljuridica);
        PreparedStatement ps1 = connection.getPrepared(sqlpessoa);
        if(cnpj.equals("")){
            ps.setString(1, pessoa.getCnpj());
        } else{
            ps.setString(1, cnpj);
        }

        if(nome.equals("")){
            ps1.setString(1, pessoa.getNome());
        } else{
            ps1.setString(1, nome);
        }

        if(logradouro.equals("")){
            ps1.setString(2, pessoa.getLogradouro());
        } else{
            ps1.setString(2, logradouro);
        }

        if(cidade.equals("")){
            ps1.setString(3, pessoa.getCidade());
        } else{
            ps1.setString(3, cidade);
        }

        if(estado.equals("")){
            ps1.setString(4, pessoa.getEstado());
        } else{
            ps1.setString(4, estado);
        }

        if(telefone.equals("")){
            ps1.setString(5, pessoa.getTelefone());
        } else{
            ps1.setString(5, telefone);
        }
    }

```

```

        if(!email.equals("")){
            ps1.setString(6, pessoa.getEmail());
        } else{
            ps1.setString(6, email);
        }
        ps.execute();
        ps1.execute();

        connection.closeConnection();
        //connection.closeResult(ps);
        connection.closeStatement(sqljuridica);
    }

    public void excluir(int id)throws Exception{
        String sqljuridica = "DELETE FROM pessoa_juridica WHERE
idpessoa="+id;
        String sqlpessoa = "DELETE FROM pessoa WHERE idpessoa="+id;
        PreparedStatement ps = connection.getPrepared(sqljuridica);
        PreparedStatement ps1 = connection.getPrepared(sqlpessoa);
        ps.execute();
        ps1.execute();
        connection.closeConnection();
        //connection.closeResult(ps);
        connection.closeStatement(sqljuridica);
    }
}

```

#### Classe CadastroBDTeste

```

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

/*
 * 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
 */

/**
 *
 * @author grego
 */
public class CadastroBDTeste {

    public static void main(String[] args)throws Exception {
        // a. Instanciar uma pessoa física e persistir no banco de dados
        //Instanciando a sequencia
        SequenceManager seq = new SequenceManager();

        PessoaFisica pessoaIncluir = new
PessoaFisica(seq.getValue("seq_Pessoa"),"Gregorio", "Rua 360, Centro",
"Recife", "PE", "1212-1212","Gregorio@recife.com","98765432112");
    }
}

```

```

        PessoaFisicaDAO pessoaPF1 = new PessoaFisicaDAO();

        // b.Alterar os dados da pessoa física no banco.
        // Alterando pessoa e pessoaFisica pelo id--> 3 . Mudando nome, cpf e
telefone
        PessoaFisicaDAO pessoaPF1 = new PessoaFisicaDAO();

        pessoaPF1.alterar( 3, "12345678998", "Emerson Gregorio", "", "",
"", "123456789", "");

        // c.Consultar todas as pessoas físicas do banco de dados e listar no
console.
        // Retorno de todas as pessoas físicas do banco de dados
        System.out.println("Pessoas fisicas:");
        PessoaFisicaDAO pessoasPF = new PessoaFisicaDAO();
        List<PessoaFisica> resultado = pessoasPF.getPessoas();
        for (PessoaFisica pessoaFisica : resultado) {
            pessoaFisica.exibir();
        }

        //d. Excluir a pessoa física criada anteriormente no banco.
        // Excluindo pessoaFisica e Pessoa pelo id.
        PessoaFisicaDAO pessoaPF2 = new PessoaFisicaDAO();
        pessoaPF2.excluir(3);

        // e.Instanciar uma pessoa jurídica e persistir no banco de dados.

        //Incluir pessoa juridica e pessoa
        PessoaJuridica pessoaIncluir2 = new
PessoaJuridica(seq.getValue("seq_Pessoa"),"GREG LTDA",
        "Rua Gregorio, Centro","Maceio", "AL", "9898-
9898","GREGLTDA@maceio.com","55555555555555");
        PessoaJuridicaDAO pessoaPJ = new PessoaJuridicaDAO();
        pessoaPJ.incluir(pessoaIncluir2);

        // f.Alterar os dados da pessoa jurídica no banco.

        // Alterando pessoa e pessoaJuridica pelo id--> 4 . Mudando nome e cnpj
        PessoaJuridicaDAO pessoaPJ2 = new PessoaJuridicaDAO();
        pessoaPJ2.alterar( 4, "999999999999999", "Gregorio LTDA ", "", "",
"", "", "");

        // g.Consultar todas as pessoas jurídicas do banco e listar no console.

        // Retorno de todas as pessoas juridicas do banco de dados
        System.out.println("Pessoas juridicas:");
        PessoaJuridicaDAO pessoasPJ = new PessoaJuridicaDAO();
        List<PessoaJuridica> resultado2 = pessoasPJ.getPessoas();
        for (PessoaJuridica pessoaJuridica : resultado2) {
            pessoaJuridica.exibir();
        }

```

```

        // h.Excluir a pessoa jurídica criada anteriormente no banco.

        // Excluindo pessoa jurídica e Pessoa pelo id.
        PessoaJuridicaDAO pessoaPJ3 = new PessoaJuridicaDAO();
        pessoaPJ3.excluir(4);
    }
}

```

## Procedimento 2: Alimentando a Base

### Classe CadastroBDTeste2 (Procedimento 2)

```

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;

/*
 * 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
 */

/**
 *
 * @author grego
 */
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 'F':
            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(seq.getValue("seq_Pessoa"),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(seq.getValue("seq_Pessoa"),nomej,
logradouroj,cidadej, estadoj,

```

```

telefonej,emailj,cnpj));
PessoaJuridicaDAO();

PessoaJuridicaDAO pessoaPJ = new
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();
//PessoaFisica pessoaFisicaLocalizada =
pfRepo.obter(idPessoaFisica);

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();

```

```

        pessoaJuridicaLocalizada.getCidade());
        System.out.println("Cidade: " +
        System.out.print("Nova Cidade: ");
        String novoCidade = scan.nextLine();

        pessoaJuridicaLocalizada.getEstado());
        System.out.println("Estado: " +
        System.out.print("Novo Estado: ");
        String novoEstado = scan.nextLine();

        pessoaJuridicaLocalizada.getTelefone());
        System.out.println("Telefone: " +
        System.out.print("Novo Telefone: ");
        String novoTelefone = scan.nextLine();

        pessoaJuridicaLocalizada.getEmail());
        System.out.println("Email: " +
        System.out.print("Novo Email: ");
        String novoEmail = scan.nextLine();

        pessoaJuridicaLocalizada.getCnpj());
        System.out.println("CNPJ atual: " +
        System.out.print("Novo CNPJ: ");
        String novoCNPJ = scan.nextLine();

        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);

            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("=====");

```

```

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

        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();
}
}

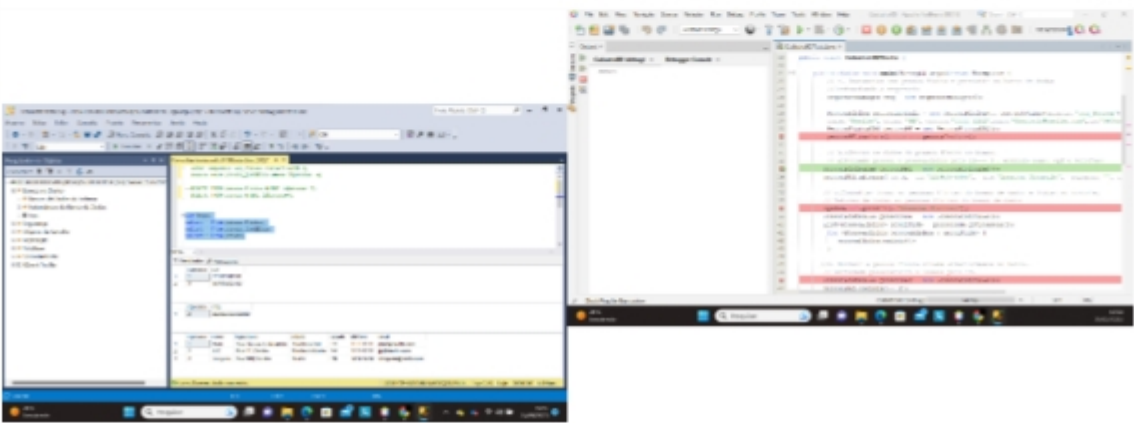
```

---

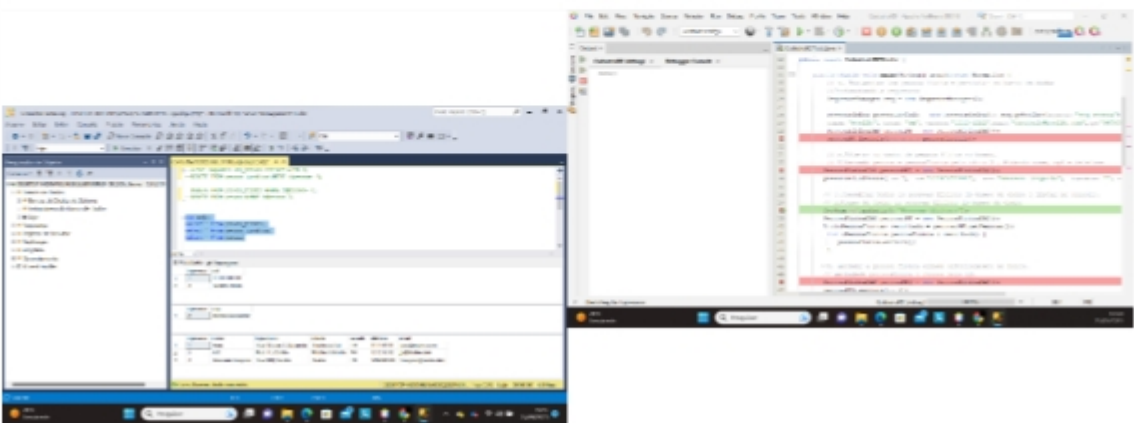
## Resultados:

Procedimento 1: <https://github.com/Gregdev22/Missao-3-Mundo-3/tree/main/Procedimento1>

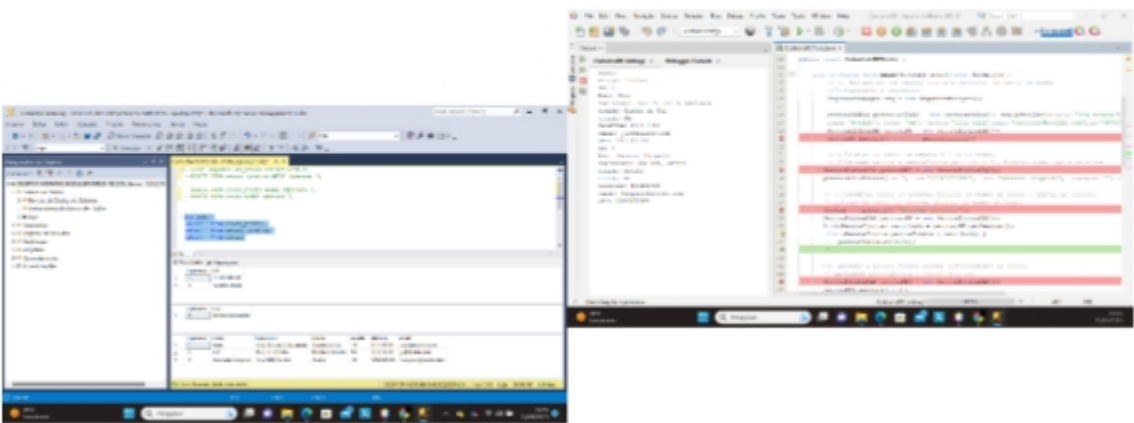
## Instanciar uma pessoa física e persistir no banco de dados



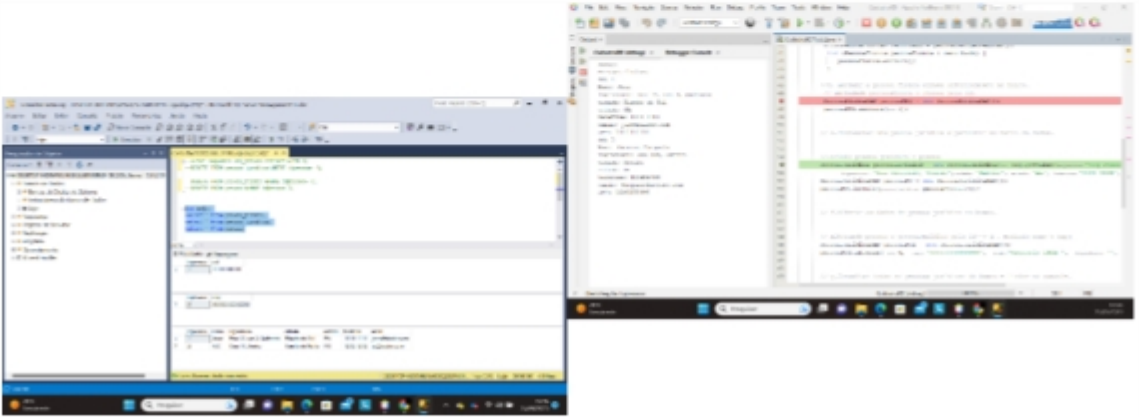
## Alterar os dados da pessoa física no banco



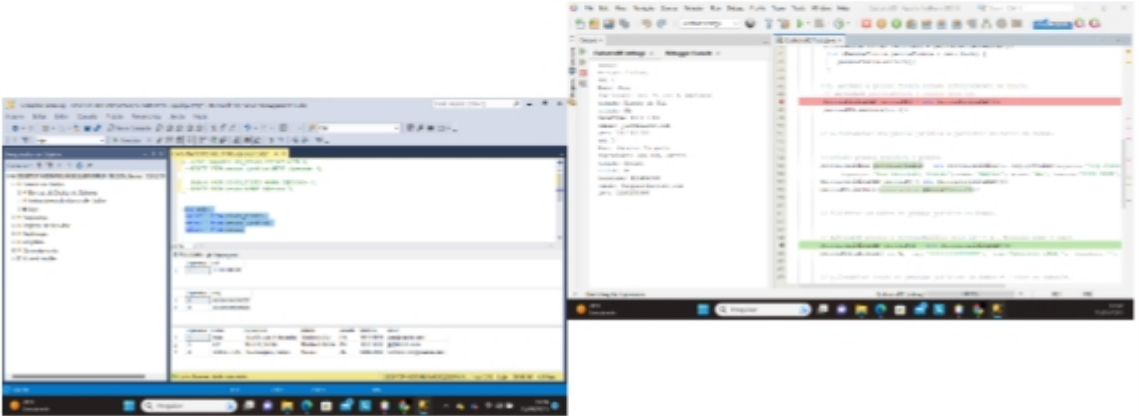
## Consultar todas as pessoas físicas do banco de dados e listar no console



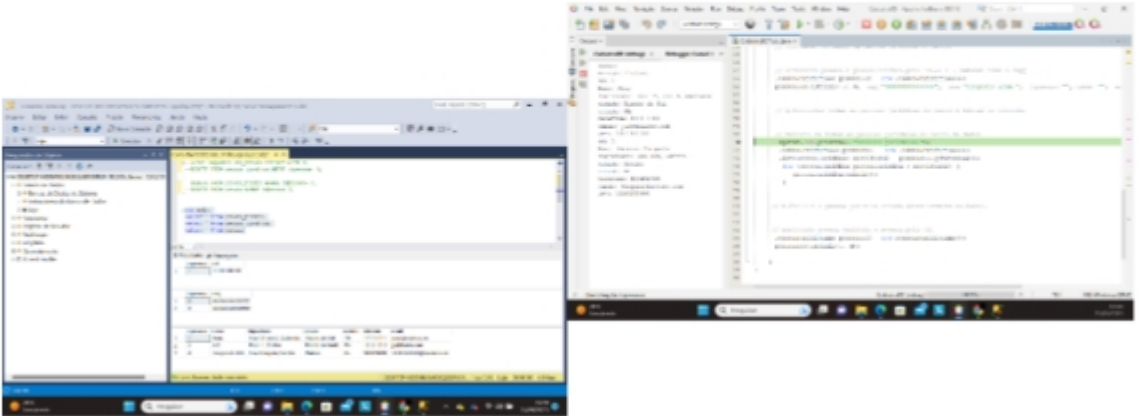
## Excluir a pessoa física criada anteriormente no banco



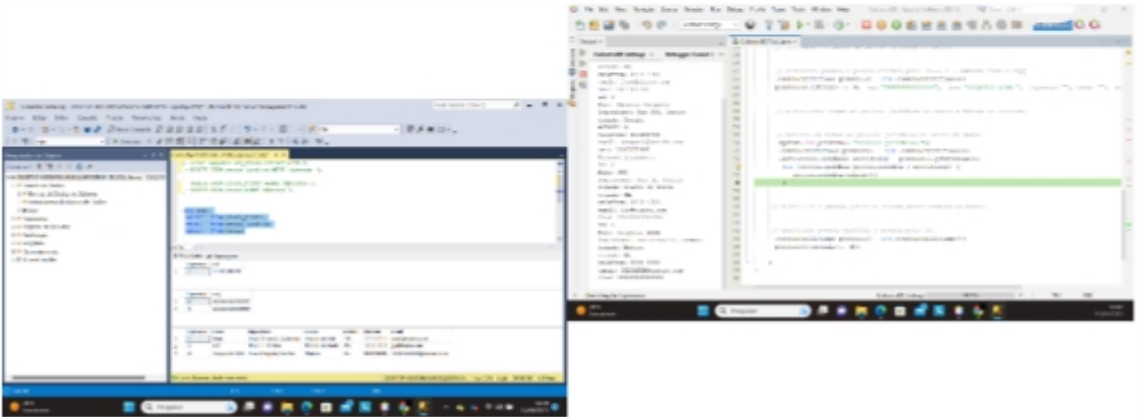
## Instanciar uma pessoa jurídica e persistir no banco de dados



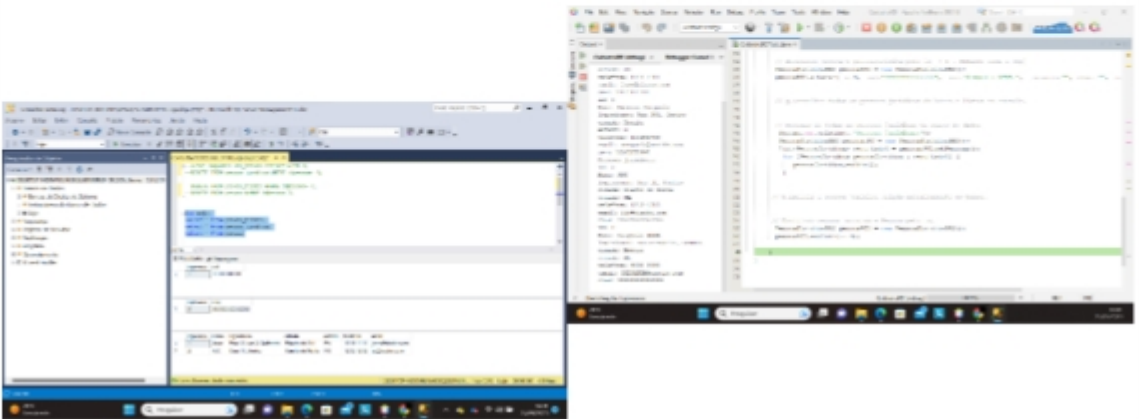
## Alterar os dados da pessoa jurídica no banco



## Consultar todas as pessoas jurídicas do banco e listar no console



## Excluir a pessoa jurídica criada anteriormente no banco



Procedimento 2: [https://github.com/Gregdev22/Missao-3-Mundo-3/blob/main/Procedimento2/CadastroBD/procedimento-2\\_1fF7rEli.mp4](https://github.com/Gregdev22/Missao-3-Mundo-3/blob/main/Procedimento2/CadastroBD/procedimento-2_1fF7rEli.mp4)

## Análise e Conclusão

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

Facilitar as estratégias de comunicação dos bancos de dados, com procedimentos de solicitação de informações e execução de atividades.



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

Statement usa consultas SQL baseadas em strings, quando você pretende executar instruções SQL fixas ou seja instruções de texto puro. Já o PreparedStatement é usado quando você pretende executar instruções SQL parametrizadas e possibilita que você especifique o tipo do parâmetro passado como Int, String, Float etc.

Como o padrão DAO melhora a manutenibilidade do software?

Possibilidade de reutilização dos componentes e distribuição entre distintas aplicações. Possibilidade de dividir projetos grandes e complexos em projetos mais simples e atribuí-los a diferentes times ou recursos garantindo a unicidade dos componentes de acesso a dados e regras de negócio. Caso você tenha uma camada DAO separada da sua camada lógica a alteração da fonte de dados se torna simplesmente uma questão de criar uma nova interface de acesso para a nova fonte, mantendo suas regras de negócio intocadas.

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

A herança é refletida por meio de Identificadores em comum. Onde há uma tabela base (Pessoa) e outras derivadas (Pessoa Física e Pessoa Jurídica).

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

Persistência em arquivo: Arquivos nada custam pois são inerentemente suportados pelo sistema operacional; arquivos não têm custo de aquisição mas têm um alto custo de manutenção; Arquivos são limitados em tamanho pelo sistema operacional; Arquivos podem ser rápidos se couberem na memória ou se forem eficientemente acessados seqüencial ou randomicamente; Arquivos de backup são necessários para recuperação de crash; Arquivos podem ser protegidos pelo sistema operacional (mas sem sofisticação).

Persistência em banco de dados: Pode ser muito caro; podem armazenar gigantescas quantidades de informação; têm algoritmos e estruturas de dados especiais para lidar com grandes quantidades de dados; oferecem independência de dados de forma que o desenvolvedor pode se concentrar nos aspectos lógicos dos dados, sem preocupação imediata com detalhes de implementação física; têm lógica sofisticada para

recuperação de crash; oferecem o suporte a regras de integridade (definidas pelo programador) Suporte a transações; oferecem linguagens de consulta para facilitar o manuseio de dados;podem implementar segurança com passwords, views, etc.

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

É um mecanismo bastante poderoso, que facilita muito a escrita de código conciso e evita que o programador seja obrigado a escrever um monte de código “inútil”, principalmente em operações simples, além de flexibilizar o mesmo.

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

Porque métodos Static se refere à classe e executam quando a classe for carregada.