Missão Prática | Nível 3 | Mundo 3

Lucas Figueirêdo Costa de Queiroz

Campos Via Corpos
Disciplina: Mundo 3 – Nível 3 BackEnd sem banco não tem

Objetivo da Prática

Criar um sistema cadastral com persistência de dados em um banco relacional através do JDBC, seguindo o padrão DAO, para o manuseio e manipulação dos dados do banco de dados Loja, criado na missão anterior.

O sistema irá se conectar ao SQL Server onda está o banco e permitirá as operações de inclsão, alteração, exclusão e consulta do registros de suas tabelas.

1º Procedimento | Mapeamento Objeto-Relacional e DAO

Conclusão:

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

O middleware permite a comunicação entre as aplicações front-end e back-end, com ele a integração é de forma transparente e mudanças de fornecedor exige pouco ou nenhuma alteração do código.

O JDBC permite o java se comunicar a diferentes tipos de bancos de dados, com ele se consulta e manipula dados usando comandos SQL em meio ao código java.

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

Ambos são executam comandos SQL através de Strings, mas o PreparedStatement permite o uso de expressões parametrizadas, logo ele pode ser usado mais vezes do que o Statement sendo necessário apenas alterar os paramêtros.

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

Ao separar método de conexão dos demais, isolando-o numa classe, a manutenção se torna muito mais simples num caso de alteração do fornecedor.

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

   A classe pai e as classes filhas terão suas tabelas, mas os atributos comuns ficarão registrados na tabela da classe pai enquanto as tabelas da classes conterão apenas seus atributos exclusivos junto com a chave estrangeira da tabela pai.

2º Procedimento | Alimentando a Base

Conclusão:

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

   O primeiro salva em arquivos sendo uma boa opção se necessário se precisam ser recuperados com frequência, o outro salva em um servidor, ele é muito mais organizado sendo bom para a análise de dados.

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

   Utilizando-o com o stream da interface Collections o código se torna mais enxutoe com uma leitura mais natural.

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

   Porque esses métodos pertecem a classe e não as suas instâncias.

1º Procedimento
Códigos:

Pessoa

```
package cadastrobd.model;

public class Pessoa {
        //Atributos
        private int id;
        private String nome;
        private String logradouro;
        private String cidade;
        private String estado;
        private String telefone;
        private String email;

        //Construtores
        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;
        }

        //getters e setters
        public void set_id(int id) {
                this.id = id;
                System.out.println("ID Atualizado.");
        }
        public int get_id() {
                return this.id;
        }
        public void set_nome(String nome) {
                this.nome = nome;
                System.out.println("Nome Atualizado.");
        }
        public String get_nome() {
                return this.nome;
        }
        public void set_logradouro(String logradouro) {
                this.logradouro = logradouro;
                System.out.println("Logradouro Atualizado.");
        }
        public String get_logradouro() {
                return this.logradouro;
        }
        public void set_cidade(String cidade) {
                this.cidade = cidade;
```

```
                System.out.println("Cidade Atualizada.");
        }
        public String get_cidade() {
                return this.cidade;
        }
        public void set_estado(String estado) {
                this.estado = estado;
                System.out.println("Estado Atualizado.");
        }
        public String get_estado() {
                return this.estado;
        }
        public void set_telefone(String telefone) {
                this.telefone = telefone;
                System.out.println("Telefone Atualizado.");
        }
        public String get_telefone() {
                return this.telefone;
        }
        public void set_email(String email) {
                this.email = email;
                System.out.println("Email Atualizado.");
        }
        public String get_email() {
                return this.email;
        }

        //Métodos
        public void exibir() {
                System.out.println("ID: " + this.id);
                System.out.println("Nome: " + this.nome);
                System.out.println("Logradouro: " + this.logradouro);
                System.out.println("Cidade: " + this.cidade);
                System.out.println("Estado: " + this.estado);
                System.out.println("Telefone: " + this.telefone);
                System.out.println("Email: " + this.email);
        }
}
```

Pessoa Física

```
package cadastrobd.model;

public class PessoaFisica extends Pessoa {
        //Atributos
        private String cpf;

        //Construtores
        public PessoaFisica() {}

        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;
```

```
        }

        //getters e setters
        public void set_cpf(String cpf) {
                this.cpf = cpf;
                System.out.println("CPF Atualizado.");
        }
        public String get_cpf() {
                return this.cpf;
        }

        //Métodos
        public void exibir() {
                super.exibir();
                System.out.println("CPF: " + this.cpf);
        }
}
```

Pessoa Jurídica

```
package cadastrobd.model;

public class PessoaJuridica extends Pessoa {
        //Atributos
        private String cnpj;

        //Construtores
        public PessoaJuridica() {}

        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;
        }

        //getters e setters
        public void set_cnpj(String cnpj) {
                this.cnpj = cnpj;
                System.out.println("CNPJ Atualizado.");
        }
        public String get_cnpj() {
                return this.cnpj;
        }

        //Métodos
        public void exibir() {
                super.exibir();
                System.out.println("CNPJ: " + this.cnpj);
        }
}
```

ConectorBD

```
package cadastrobd.model.util;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public final class ConectorBD {

    public static Connection getConnection() throws ClassNotFoundException,
SQLException{
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        final String url =
"jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCerti
ficate=true";
        final String user = "loja";
        final String password = "loja";
        return DriverManager.getConnection(url, user, password);
    }

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

    public static ResultSet getSelect(PreparedStatement ps) throws SQLException,
ClassNotFoundException{
        return ps.executeQuery();
    }

    public static void close(ResultSet consulta) throws SQLException{
        if(!consulta.isClosed()){
            consulta.close();
        }
    }

    public static void close(PreparedStatement st) throws SQLException{
        if(!st.isClosed()){
            st.close();
        }
    }

    public static void close(Connection conexao) throws SQLException{
        if(conexao.isClosed()){
            conexao.close();
        }
    }
}
```

Sequence Manager

```
package cadastrobd.model.util;

import java.sql.Connection;
```

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class SequenceManager {

    public static int getValue(String sequencia_nome) throws SQLException,
ClassNotFoundException{
        int id = 0;
        //Conectando com o banco
        Connection conexao = ConectorBD.getConnection();
        //System.out.println("Conexão criada");

        //Criando preparedStatement
        String sql = "SELECT NEXT VALUE FOR " + sequencia_nome + " AS ID;";
        PreparedStatement ps = ConectorBD.getPrepared(conexao, sql);
        //System.out.println("PreparedStatement criado");

        //Recuperando valor
        ResultSet consulta = ConectorBD.getSelect(ps);
        //System.out.println("consulta retornada");
        if(consulta.next()){
            id = consulta.getInt("id");
        }
        else{
            System.out.println("Valor não encontrado");
        }

        //Fechando conexões
        ConectorBD.close(consulta);
        ConectorBD.close(ps);
        ConectorBD.close(conexao);
        //System.out.println("conexões fechadas");

        return id;
    }
}
```

Pessoa Física DAO

```
package cadastrobd.model;

import cadastrobd.model.util.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

public class PessoaFisicaDAO {

    public static PessoaFisica getPessoa(int id) throws SQLException,
ClassNotFoundException{
        PessoaFisica pessoa = new PessoaFisica();
```

```
        //Conectando com o banco
        Connection conexao = ConectorBD.getConnection();
        //System.out.println("conexão criada");

        //Criando preparedStatement
        String sql = "SELECT p.idPessoa, p.nome, p.logradouro, p.cidade,
p.estado, p.telefone, p.email, pf.cpf\n" +
"FROM Pessoa p JOIN PessoaFisica pf ON (p.idPessoa = pf.idPessoa) WHERE
p.idPessoa = ?;";
        PreparedStatement ps = ConectorBD.getPrepared(conexao, sql);
        //System.out.println("preparedStatement criado");

        //Consultando o banco
        ps.setInt(1, id);
        ResultSet consulta = ConectorBD.getSelect(ps);
        //System.out.println("consulta retornada");

        //Criando objeto pessoaFisica
        if(consulta.next()){
            String nome = consulta.getString("nome");
            String logradouro = consulta.getString("logradouro");
            String cidade = consulta.getString("cidade");
            String estado = consulta.getString("estado");
            String telefone = consulta.getString("telefone");
            String email = consulta.getString("email");
            String cpf = consulta.getString("cpf");
            pessoa = new PessoaFisica(id, nome, logradouro, cidade, estado,
telefone, email, cpf);
            //System.out.println("Objeto criado");
        }
        else{
            System.out.println("Pessoa não existe.");
        }

        //Fechando conexões
        ConectorBD.close(consulta);
        ConectorBD.close(ps);
        ConectorBD.close(conexao);
        //System.out.println("conexões fechadas");

        return pessoa;
    }

    public static ArrayList<PessoaFisica> getPessoas() throws SQLException,
ClassNotFoundException{
        ArrayList<PessoaFisica> lista = new ArrayList<PessoaFisica>();

        //Conectando com o banco
        Connection conexao = ConectorBD.getConnection();
        //System.out.println("conexão criada");

        //Criando preparedStatement
        String sql = "SELECT p.idPessoa, p.nome, p.logradouro, p.cidade,
```

```
p.estado, p.telefone, p.email, pf.cpf\n" +
"FROM Pessoa p JOIN PessoaFisica pf ON (p.idPessoa = pf.idPessoa);";
        PreparedStatement ps = ConectorBD.getPrepared(conexao, sql);
        //System.out.println("preparedStatement criado");

        //Consultando o banco
        ResultSet consulta = ConectorBD.getSelect(ps);
        //System.out.println("consulta retornada");

        //Preenchendo lista
        while(consulta.next()){
            int id = consulta.getInt("idPessoa");
            String nome = consulta.getString("nome");
            String logradouro = consulta.getString("logradouro");
            String cidade = consulta.getString("cidade");
            String estado = consulta.getString("estado");
            String telefone = consulta.getString("telefone");
            String email = consulta.getString("email");
            String cpf = consulta.getString("cpf");
            lista.add(new PessoaFisica(id, nome, logradouro, cidade, estado,
telefone, email, cpf));
        }

        //Fechando conexões
        ConectorBD.close(consulta);
        ConectorBD.close(ps);
        ConectorBD.close(conexao);
        //System.out.println("conexões fechadas");

        return lista;
    }

    public static void incluir(PessoaFisica pessoa) throws SQLException,
ClassNotFoundException{
        //Pegando o próximo id da sequência
        int id = SequenceManager.getValue("sequenciaIdPessoa");

        if(id != 0){
            //Conectando com o banco
            Connection conexao = ConectorBD.getConnection();
            //System.out.println("conexão criada");

            //Início
            conexao.setAutoCommit(false);

            //Adicionando registro a tabela Pessoa
            String sqlPessoa = "INSERT INTO Pessoa (idPessoa, nome, logradouro,
cidade, estado, telefone, email)\n" +
"VALUES (?, ?, ?, ?, ?, ?, ?);";
            PreparedStatement psPessoa = ConectorBD.getPrepared(conexao,
sqlPessoa);
            psPessoa.setInt(1, id);
            psPessoa.setString(2, pessoa.get_nome());
            psPessoa.setString(3, pessoa.get_logradouro());
```

```
            psPessoa.setString(4, pessoa.get_cidade());
            psPessoa.setString(5, pessoa.get_estado());
            psPessoa.setString(6, pessoa.get_telefone());
            psPessoa.setString(7, pessoa.get_email());
            psPessoa.executeUpdate();

            //Adicionando registro a tabela PessoaFisica
            String sqlPessoaFisica = "INSERT INTO PessoaFisica (idPessoa, cpf)
VALUES (?, ?);";
            PreparedStatement psPessoaFisica = ConectorBD.getPrepared(conexao,
sqlPessoaFisica);
            psPessoaFisica.setInt(1,id);
            psPessoaFisica.setString(2,pessoa.get_cpf());
            psPessoaFisica.executeUpdate();

            conexao.commit();
            //Fim

            System.out.println("Inclusão concluida");

            //Fechando conexões
            ConectorBD.close(psPessoa);
            ConectorBD.close(psPessoaFisica);
            ConectorBD.close(conexao);
            //System.out.println("conexões fechadas");
        }
        else{
            System.out.println("Pessoa não incluída no registro, problema no
id");
        }
    }

    public static void alterar(PessoaFisica pessoa) throws SQLException,
ClassNotFoundException{
        //Conectando com o banco
        Connection conexao = ConectorBD.getConnection();
        //System.out.println("conexão criada");

        //Início
        conexao.setAutoCommit(false);

        //Atualizando tabela Pessoa
        String sqlPessoa = "UPDATE Pessoa SET nome = ?, logradouro = ?, cidade =
?, estado = ?, telefone = ?, email = ? WHERE idPessoa = ?;";
        PreparedStatement psPessoa = ConectorBD.getPrepared(conexao, sqlPessoa);
        psPessoa.setString(1, pessoa.get_nome());
        psPessoa.setString(2, pessoa.get_logradouro());
        psPessoa.setString(3, pessoa.get_cidade());
        psPessoa.setString(4, pessoa.get_estado());
        psPessoa.setString(5, pessoa.get_telefone());
        psPessoa.setString(6, pessoa.get_email());
        psPessoa.setInt(7, pessoa.get_id());
        psPessoa.executeUpdate();
```

```
        //Atualizando tabela PessoaFisica
        String sqlPessoaFisica = "UPDATE PessoaFisica SET cpf = ? WHERE idPessoa
= ?;";
        PreparedStatement psPessoaFisica = ConectorBD.getPrepared(conexao,
sqlPessoaFisica);
        psPessoaFisica.setString(1,pessoa.get_cpf());
        psPessoaFisica.setInt(2,pessoa.get_id());
        psPessoaFisica.executeUpdate();

        conexao.commit();
        //Fim

        System.out.println("Atualização concluida");

        //Fechando conexões
        ConectorBD.close(psPessoa);
        ConectorBD.close(psPessoaFisica);
        ConectorBD.close(conexao);
        //System.out.println("conexões fechadas");
    }

    public static void excluir(int id) throws SQLException,
ClassNotFoundException{
        //Conectando com o banco
        Connection conexao = ConectorBD.getConnection();
        //System.out.println("conexão criada");

        //Início
        conexao.setAutoCommit(false);

        //Excluindo registro da tabela PessoaFisica
        String sqlPessoaFisica = "DELETE FROM PessoaFisica WHERE idPessoa = ?;";
        PreparedStatement psPessoaFisica = ConectorBD.getPrepared(conexao,
sqlPessoaFisica);
        psPessoaFisica.setInt(1,id);
        psPessoaFisica.executeUpdate();

        //Excluindo registro da tabela Pessoa
        String sqlPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";
        PreparedStatement psPessoa = ConectorBD.getPrepared(conexao, sqlPessoa);
        psPessoa.setInt(1, id);
        psPessoa.executeUpdate();

        conexao.commit();
        //Fim

        System.out.println("Exclusão concluida");

        //Fechando conexões
        ConectorBD.close(psPessoa);
        ConectorBD.close(psPessoaFisica);
        ConectorBD.close(conexao);
        //System.out.println("conexões fechadas");
    }
```

```
}
```

Pessoa Jurídica DAO

```
package cadastrobd.model;

import cadastrobd.model.util.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

public class PessoaJuridicaDAO {

    public static PessoaJuridica getPessoa(int id) throws SQLException,
ClassNotFoundException{
        PessoaJuridica pessoa = new PessoaJuridica();

        //Conectando com o banco
        Connection conexao = ConectorBD.getConnection();
        //System.out.println("conexão criada");

        //Criando preparedStatement
        String sql = "SELECT p.idPessoa, p.nome, p.logradouro, p.cidade,
p.estado, p.telefone, p.email, pj.cnpj\n" +
"FROM Pessoa p JOIN PessoaJuridica pj ON (p.idPessoa = pj.idPessoa) WHERE
p.idPessoa = ?;";
        PreparedStatement ps = ConectorBD.getPrepared(conexao, sql);
        //System.out.println("preparedStatement criado");

        //Consultando o banco
        ps.setInt(1, id);
        ResultSet consulta = ConectorBD.getSelect(ps);
        //System.out.println("consulta retornada");

        //Criando objeto pessoaJuridica
        if(consulta.next()){
            String nome = consulta.getString("nome");
            String logradouro = consulta.getString("logradouro");
            String cidade = consulta.getString("cidade");
            String estado = consulta.getString("estado");
            String telefone = consulta.getString("telefone");
            String email = consulta.getString("email");
            String cnpj = consulta.getString("cnpj");
            pessoa = new PessoaJuridica(id, nome, logradouro, cidade, estado,
telefone, email, cnpj);
            //System.out.println("Objeto criado");
        }
        else{
            System.out.println("Pessoa não existe.");
        }

        //Fechando conexões
```

```
        ConectorBD.close(consulta);
        ConectorBD.close(ps);
        ConectorBD.close(conexao);
        //System.out.println("conexões fechadas");

        return pessoa;
    }

    public static ArrayList<PessoaJuridica> getPessoas() throws SQLException,
ClassNotFoundException{
        ArrayList<PessoaJuridica> lista = new ArrayList<PessoaJuridica>();

        //Conectando com o banco
        Connection conexao = ConectorBD.getConnection();
        //System.out.println("conexão criada");

        //Criando preparedStatement
        String sql = "SELECT p.idPessoa, p.nome, p.logradouro, p.cidade,
p.estado, p.telefone, p.email, pj.cnpj\n" +
"FROM Pessoa p JOIN PessoaJuridica pj ON (p.idPessoa = pj.idPessoa);";
        PreparedStatement ps = ConectorBD.getPrepared(conexao, sql);
        //System.out.println("preparedStatement criado");

        //Consultando o banco
        ResultSet consulta = ConectorBD.getSelect(ps);
        //System.out.println("consulta retornada");

        //Preenchendo lista
        while(consulta.next()){
            int id = consulta.getInt("idPessoa");
            String nome = consulta.getString("nome");
            String logradouro = consulta.getString("logradouro");
            String cidade = consulta.getString("cidade");
            String estado = consulta.getString("estado");
            String telefone = consulta.getString("telefone");
            String email = consulta.getString("email");
            String cnpj = consulta.getString("cnpj");
            lista.add(new PessoaJuridica(id, nome, logradouro, cidade, estado,
telefone, email, cnpj));
        }

        //Fechando conexões
        ConectorBD.close(consulta);
        ConectorBD.close(ps);
        ConectorBD.close(conexao);
        //System.out.println("conexões fechadas");

        return lista;
    }

    public static void incluir(PessoaJuridica pessoa) throws SQLException,
ClassNotFoundException{
        //Pegando o próximo id da sequência
        int id = SequenceManager.getValue("sequenciaIdPessoa");
```

```
        if(id != 0){
            //Conectando com o banco
            Connection conexao = ConectorBD.getConnection();
            //System.out.println("conexão criada");

            //Início
            conexao.setAutoCommit(false);

            //Adicionando registro a tabela Pessoa
            String sqlPessoa = "INSERT INTO Pessoa (idPessoa, nome, logradouro,
cidade, estado, telefone, email)\n" +
"VALUES (?, ?, ?, ?, ?, ?, ?);";
            PreparedStatement psPessoa = ConectorBD.getPrepared(conexao,
sqlPessoa);
            psPessoa.setInt(1, id);
            psPessoa.setString(2, pessoa.get_nome());
            psPessoa.setString(3, pessoa.get_logradouro());
            psPessoa.setString(4, pessoa.get_cidade());
            psPessoa.setString(5, pessoa.get_estado());
            psPessoa.setString(6, pessoa.get_telefone());
            psPessoa.setString(7, pessoa.get_email());
            psPessoa.executeUpdate();

            //Adicionando registro a tabela PessoaJuridica
            String sqlPessoaJuridica = "INSERT INTO PessoaJuridica (idPessoa,
cnpj) VALUES (?, ?);";
            PreparedStatement psPessoaJuridica = ConectorBD.getPrepared(conexao,
sqlPessoaJuridica);
            psPessoaJuridica.setInt(1,id);
            psPessoaJuridica.setString(2,pessoa.get_cnpj());
            psPessoaJuridica.executeUpdate();

            conexao.commit();
            //Fim

            System.out.println("Inclusão concluida");

            //Fechando conexões
            ConectorBD.close(psPessoa);
            ConectorBD.close(psPessoaJuridica);
            ConectorBD.close(conexao);
            //System.out.println("conexões fechadas");
        }
        else{
            System.out.println("Pessoa não incluída no registro, problema no
id");
        }
    }

    public static void alterar(PessoaJuridica pessoa) throws SQLException,
ClassNotFoundException{
        //Conectando com o banco
        Connection conexao = ConectorBD.getConnection();
```

```
        //System.out.println("conexão criada");

        //Início
        conexao.setAutoCommit(false);

        //Atualizando tabela Pessoa
        String sqlPessoa = "UPDATE Pessoa SET nome = ?, logradouro = ?, cidade =
?, estado = ?, telefone = ?, email = ? WHERE idPessoa = ?;";
        PreparedStatement psPessoa = ConectorBD.getPrepared(conexao, sqlPessoa);
        psPessoa.setString(1, pessoa.get_nome());
        psPessoa.setString(2, pessoa.get_logradouro());
        psPessoa.setString(3, pessoa.get_cidade());
        psPessoa.setString(4, pessoa.get_estado());
        psPessoa.setString(5, pessoa.get_telefone());
        psPessoa.setString(6, pessoa.get_email());
        psPessoa.setInt(7, pessoa.get_id());
        psPessoa.executeUpdate();

        //Atualizando tabela PessoaJuridica
        String sqlPessoaJuridica = "UPDATE PessoaJuridica SET cnpj = ? WHERE
idPessoa = ?;";
        PreparedStatement psPessoaJuridica = ConectorBD.getPrepared(conexao,
sqlPessoaJuridica);
        psPessoaJuridica.setString(1,pessoa.get_cnpj());
        psPessoaJuridica.setInt(2,pessoa.get_id());
        psPessoaJuridica.executeUpdate();

        conexao.commit();
        //Fim

        System.out.println("Atualização concluida");

        //Fechando conexões
        ConectorBD.close(psPessoa);
        ConectorBD.close(psPessoaJuridica);
        ConectorBD.close(conexao);
        //System.out.println("conexões fechadas");
    }

    public static void excluir(int id) throws SQLException,
ClassNotFoundException{
        //Conectando com o banco
        Connection conexao = ConectorBD.getConnection();
        //System.out.println("conexão criada");

        //Início
        conexao.setAutoCommit(false);

        //Excluindo registro da tabela PessoaJuridica
        String sqlPessoaJuridica = "DELETE FROM PessoaJuridica WHERE idPessoa =
?;";
        PreparedStatement psPessoaJuridica = ConectorBD.getPrepared(conexao,
sqlPessoaJuridica);
        psPessoaJuridica.setInt(1,id);
```

```
        psPessoaJuridica.executeUpdate();

        //Excluindo registro da tabela Pessoa
        String sqlPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";
        PreparedStatement psPessoa = ConectorBD.getPrepared(conexao, sqlPessoa);
        psPessoa.setInt(1, id);
        psPessoa.executeUpdate();

        conexao.commit();
        //Fim

        System.out.println("Exclusão concluida");

        //Fechando conexões
        ConectorBD.close(psPessoa);
        ConectorBD.close(psPessoaJuridica);
        ConectorBD.close(conexao);
        //System.out.println("conexões fechadas");
    }
}
```

CadastroBD Teste

```
import cadastrobd.model.*;
import cadastrobd.model.util.SequenceManager;
import java.sql.SQLException;


public class CadastroBDTeste {

    public static void main(String[] args){
        try{
            //Instanciar e persistir pessoa física no banco
            PessoaFisica pessoa1 = new PessoaFisica(0, "xxxxx", "xxxx", "xxx",
"xx", "xxxx-xxxx", "x@x.com", "xxxxxxxxxxx");
            PessoaFisicaDAO.incluir(pessoa1);

            //Atualizar pessoa física
            int id = SequenceManager.getValue("sequenciaIdPessoa") - 1;
            PessoaFisica pessoa2 = new PessoaFisica(id, "yyyyy", "yyyy", "yyy",
"yy", "yyyy-yyyy", "y@y.com", "yyyyyyyyyyy");
            PessoaFisicaDAO.alterar(pessoa2);

            //Cinsultar e exibir todas as pessoas físicas
            PessoaFisicaDAO.getPessoas().forEach(pessoa -> pessoa.exibir());

            //Excluir pessoa física
            PessoaFisicaDAO.excluir(id);

            //Instanciar e persistir pessoa jurídica no banco
            PessoaJuridica pessoa3 = new PessoaJuridica(0, "xxxxx", "xxxx",
"xxx", "xx", "xxxx-xxxx", "x@x.com", "xxxxxxxxxxxxx");
            PessoaJuridicaDAO.incluir(pessoa3);
```

```
            //Atualizar pessoa jurídica
            id = SequenceManager.getValue("sequenciaIdPessoa") - 1;
            PessoaJuridica pessoa4 = new PessoaJuridica(id, "yyyyy", "yyyy",
"yyy", "yy", "yyyy-yyyy", "y@y.com", "yyyyyyyyyyyyyy");
            PessoaJuridicaDAO.alterar(pessoa4);

            //Cinsultar e exibir todas as pessoas juridica
            PessoaJuridicaDAO.getPessoas().forEach(pessoa -> pessoa.exibir());

            //Excluir pessoa juridica
            PessoaJuridicaDAO.excluir(id);
        }
        catch(SQLException e){
            System.out.println("Algum problema com o SQL");
        }
        catch(Exception e){
            System.out.println("Erro");
        }
    }
}
```

Resultado:

```
run:
Inclusão concluida
Atualização concluida
ID: 2
Nome: Joao
Logradouro: Rua 12, casa 3, Quintanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: joao@riacho.com
CPF: 11111111111
ID: 26
Nome: yyyyy
Logradouro: yyyy
Cidade: yyy
Estado: yy
Telefone: yyyy-yyyy
Email: y@y.com
CPF: yyyyyyyyyyy
Exclusão concluida
Inclusão concluida
Atualização concluida
ID: 3
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
Email: jjc@riacho.com
CNPJ: 22222222222222
```

```
ID: 28
Nome: yyyyy
Logradouro: yyyy
Cidade: yyy
Estado: yy
Telefone: yyyy-yyyy
Email: y@y.com
CNPJ: yyyyyyyyyyyyyy
Exclusão concluida
BUILD SUCCESSFUL (total time: 2 seconds)
```

2º Procedimento
Códigos:

Principal

```
import cadastrobd.model.*;
import java.sql.SQLException;
import java.util.Scanner;

public class Principal {

    public static Pessoa cadastroPessoa(Scanner teclado) throws SQLException,
ClassNotFoundException{
        System.out.println("Digite o nome: ");
        String nome = teclado.nextLine();
        System.out.println("Digite o logradouro: ");
        String logradouro = teclado.nextLine();
        System.out.println("Digite a cidade: ");
        String cidade = teclado.nextLine();
        System.out.println("Digite o estado: ");
        String estado = teclado.nextLine();
        System.out.println("Digite o telefone: ");
        String telefone = teclado.nextLine();
        System.out.println("Digite o email: ");
        String email = teclado.nextLine();
        return new Pessoa(0, nome, logradouro, cidade, estado, telefone, email);

    }
    public static void main(String[] args) {
        boolean ligado = true;
        Scanner teclado = new Scanner(System.in);

        while(ligado) {
            //Display menu

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

            //Recebendo instrução selecionada
            int instrucao1 = -1;

            try {
                instrucao1 = Integer.parseInt(teclado.nextLine());
            }
            catch (Exception e) {
                System.out.println("Entrada inválida.");
            }

            //Recebendo o tipo de pessoa
            String tipo = "?";
            if(instrucao1 > 0 & instrucao1 < 6) {
                while(true) {
                    System.out.println("F - Pessoa Física | J - Pessoa
Jurídica");
                    tipo = teclado.nextLine();
                    if(tipo.equals("F") | tipo.equals("J")) {
                        break;
                    }
                    else {
                        System.out.println("Não entendi, por favor tente
novamente.");
                    }
                }
            }

            switch(instrucao1) {
            case 1:
                try{
                    Pessoa pessoa = cadastroPessoa(teclado);
                    if(tipo.equals("F")){
                        System.out.println("Digite o cpf: ");
                        String cpf = teclado.nextLine();
                        PessoaFisicaDAO.incluir(new PessoaFisica(0,
pessoa.get_nome(), pessoa.get_logradouro(), pessoa.get_cidade(),
pessoa.get_estado(), pessoa.get_telefone(), pessoa.get_email(), cpf));
                    }
                    else{
                        System.out.println("Digite o cnpj: ");
                        String cnpj = teclado.nextLine();
                        PessoaJuridicaDAO.incluir(new PessoaJuridica(0,
pessoa.get_nome(), pessoa.get_logradouro(), pessoa.get_cidade(),
pessoa.get_estado(), pessoa.get_telefone(), pessoa.get_email(), cnpj));
                    }
                }
                catch(SQLException e){
                    System.out.println("Algum problema com o SQL, inclusão não
realizada");
                }
```

```
                catch(Exception e){
                    System.out.println("Não foi possível regidtrar a pessoa");
                }
                break;

            case 2:
                try{
                    System.out.println("Digite o id: ");
                    int id = Integer.parseInt(teclado.nextLine());
                    if(tipo.equals("F")){
                        PessoaFisicaDAO.getPessoa(id).exibir();
                        Pessoa pessoa = cadastroPessoa(teclado);
                        System.out.println("Digite o cpf: ");
                        String cpf = teclado.nextLine();
                        PessoaFisicaDAO.alterar(new PessoaFisica(id,
pessoa.get_nome(), pessoa.get_logradouro(), pessoa.get_cidade(),
pessoa.get_estado(), pessoa.get_telefone(), pessoa.get_email(), cpf));
                    }
                    else{
                        PessoaJuridicaDAO.getPessoa(id).exibir();
                        Pessoa pessoa = cadastroPessoa(teclado);
                        System.out.println("Digite o cnpj: ");
                        String cnpj = teclado.nextLine();
                        PessoaJuridicaDAO.alterar(new PessoaJuridica(id,
pessoa.get_nome(), pessoa.get_logradouro(), pessoa.get_cidade(),
pessoa.get_estado(), pessoa.get_telefone(), pessoa.get_email(), cnpj));
                    }
                }
                catch(SQLException e){
                    System.out.println("Algum problema com o SQL, inclusão não
realizada");
                }
                catch(Exception e){
                    System.out.println("Não foi possível atualizar o registro");
                }
                break;

            case 3:
                try{
                    System.out.println("Digite o id: ");
                    int id = Integer.parseInt(teclado.nextLine());
                    if(tipo.equals("F")){
                        PessoaFisicaDAO.excluir(id);
                    }
                    else{
                        PessoaJuridicaDAO.excluir(id);
                    }
                }
                catch(SQLException e){
                    System.out.println("Algum problema com o SQL, inclusão não
realizada");
                }
                catch(Exception e){
                    System.out.println("Não foi possível excluir a pessoa");
```

```
                }
                break;

            case 4:
                try{
                    System.out.println("Digite o id: ");
                    int id = Integer.parseInt(teclado.nextLine());
                    if(tipo.equals("F")){
                        PessoaFisicaDAO.getPessoa(id).exibir();
                    }
                    else{
                        PessoaJuridicaDAO.getPessoa(id).exibir();
                    }
                }
                catch(SQLException e){
                    System.out.println("Algum problema com o SQL, inclusão não
realizada");
                }
                catch(Exception e){
                    System.out.println("Não foi possível encontrar a pessoa");
                }
                break;

            case 5:
                try{
                    if(tipo.equals("F")){
                        PessoaFisicaDAO.getPessoas().forEach(pesssoa ->
pesssoa.exibir());
                    }
                    else{
                        PessoaJuridicaDAO.getPessoas().forEach(pesssoa ->
pesssoa.exibir());
                    }
                }
                catch(SQLException e){
                    System.out.println("Algum problema com o SQL, inclusão não
realizada");
                }
                catch(Exception e){
                    System.out.println("Não foi possível encontrar a pessoa");
                }
                break;

            case 0:
                ligado = false;
                System.out.println("Fechando programa.");
                break;

            default:
                System.out.println("Por favor tente novamente.");
                break;
            }
        }
        teclado.close();
```

```
    }
}
```

Resultado:

```
run:
==========================================================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==========================================================================
1
F - Pessoa Física | J - Pessoa Jurídica
J
Digite o nome:
Luiz
Digite o logradouro:
Rua Y Casa X
Digite a cidade:
Diacho do Sul
Digite o estado:
KK
Digite o telefone:
1414-1414
Digite o email:
luiz@diacho.com
Digite o cnpj:
77777777777777
Inclusão concluida
==========================================================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==========================================================================
2
F - Pessoa Física | J - Pessoa Jurídica
J
Digite o id:
31
ID: 31
Nome: Luiz
Logradouro: Rua Y Casa X
Cidade: Diacho do Sul
Estado: KK
Telefone: 1414-1414
Email: luiz@diacho.com
CNPJ: 77777777777777
```

```
Digite o nome:
Luiz2
Digite o logradouro:
Rua Z Casa W
Digite a cidade:
Diacho do Norte
Digite o estado:
HA
Digite o telefone:
1515-1515
Digite o email:
luiz2@diacho.com
Digite o cnpj:
99999999999999
Atualização concluida
========================================================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
========================================================================
5
F - Pessoa Física | J - Pessoa Jurídica
J
ID: 3
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
Email: jjc@riacho.com
CNPJ: 22222222222222
ID: 31
Nome: Luiz2
Logradouro: Rua Z Casa W
Cidade: Diacho do Norte
Estado: HA
Telefone: 1515-1515
Email: luiz2@diacho.com
CNPJ: 99999999999999
========================================================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
========================================================================
3
F - Pessoa Física | J - Pessoa Jurídica
J
Digite o id:
```

```
31
Exclusão concluida
=====================================================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
=====================================================================
5
F - Pessoa Física | J - Pessoa Jurídica
J
ID: 3
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
Email: jjc@riacho.com
CNPJ: 22222222222222
=====================================================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
=====================================================================
4
F - Pessoa Física | J - Pessoa Jurídica
J
Digite o id:
3
ID: 3
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
Email: jjc@riacho.com
CNPJ: 22222222222222
=====================================================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
=====================================================================
0
Fechando programa.
BUILD SUCCESSFUL (total time: 4 minutes 49 seconds)
```