Universidade Estácio de Sá

DESENVOLVIMENTO FULL STACK- TURMA 9001

Disciplina: RPG0016 - BackEnd sem banco não tem

Semestre Letivo: 9001

Repositorio Git:

CARLOS HENRIQUE FERREIRA - MATRICULA: 202403622467

Objetivos da prática:

1 - Implementar persistência com base no middleware JDBC.

2 - Utilizar o padrão DAO (Data Access Object) no manuseio de dados.

3 - Implementar o mapeamento objeto-relacional em sistemas Java.

4 - Criar sistemas cadastrais com persistência em banco relacional.

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

## Pessoa.java:

src > cadastrobd.model > Pessoa.java > Pessoa

```java
package src.cadastrobd.model;

public class Pessoa {
    private Interger id;
    private String nome;
    private String logradouro;
    private String cidade;
    private String estado;
    private String telefone;
    private String email;
    private char tPessoa;

    public Pessoa(Interger id, String nome, String logradouro, String cidade, String estado, String telefone,
            String email, char tPessoa) {
        this.id = id;
        this.nome = nome;
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
        this.telefone = telefone;
        this.email = email;
        this.tPessoa = tPessoa;
    }

    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);
        System String tipo = (tPessoa == "F")? "Pessoa Física : Pessoa Jurídica";
        System.out.println("Tipo: " + tPessoa)
    }
```

```
}
public Interger getId() {
    return id;
}
public void setId(Interger id) {
    this.id = id;
}

public String getNome() {
    return nome;
}
public void setNome(String nome) {
    this.nome = nome;
}
public String getLogradouro() {
    return logradouro;
}
public void setLogradouro(String logradouro) {
    this.logradouro = logradouro;
}
public String getCidade() {
    return cidade;
}
public void setCidade(String cidade) {
    this.cidade = cidade;
}
public String getEstado() {
    return estado;
}
public void setEstado(String estado) {
    this.estado = estado;
}
public String getTelefone() {
    return Telefone;
}
public void setTelefone(String telefone) {
    this.telefone = telefone;
```

```
69      }
70      public void setTelefone(String telefone) {
71          this.telefone = telefone;
72      }
73      public String getEmail() {
74          return email;
75      }
76      public void setEmail(String email) {
77          this.email = email;
78      }
79      public char getTPessoa() {
80          return tPessoa;
81      }
82      public void setTPessoa(char tPessoa) {
83          this.tPessoa = tPessoa;
84      }
85  }
86
```

## - PessoaFisica.java

```java
package src.cadastrobd.model;

public class PessoaFisica extends Pessoa {
    private String cpf;

    public PessoaFisica(Interger id, String nome, String logradouro, String cidade, String estado, String tel
            String email, String cpf) {
        super(id, nome, logradouro, cidade, estado, telefone, email, "F");
        this.cpf = cpf;
    }

    public String getCpf() {
        return cpf;
    }

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

    @Override
    public void toString() {
        return super.toString() + "CPF: " + cpf;
    }
}
```

## - PessoaJuridica.java

```java
package src.cadastrobd.model;

public class PessoaJuridica extends Pessoa {
    Run | Debug
    public static void main(String[] args) {
    private String cnpj;


        public PessoaJuridica(Interger id, String nome, String logradouro, String cidade, String estado, Stri
            String email, String cpf, String cnpj) {
            super(id, nome, logradouro, cidade, estado, telefone, email);
            this.cnpj = cnpj;
        }

        public String getCnpj() {
            return cnpj;
        }

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

        @Override
            public String to

        String(){
            return super.toSpring() + "CNPJ: " + cnpj;
        }
    }
}
```

```java
package src.cadastrodao.model;

import cadastrobd.model.PessoaFisica;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class PessoaFisicaDAO {
    private Connection conn;

    public PessoaFisicaDAO(
    private Connection conn;

    public PessoaFisicaDAO(Connection connectionX) {
        this.conn = connectionX;
    }

    public void inserirPessoaFisica(PessoaFisica pf) throws SQLException {
        String sqlPessoa = "INSERT INTO pessoa ( nome, logradouro, cidade, estado, telefone, email, tipoPesso
        String sqlPessoaFisica = "INSERT INTO PessoaFisica (idPessoaFisica, cpf) VALUES(?,?)";
     }

    try{
          (conn.setAutoCommit(false));
          int pessoaId = 0;
           try(preparedStatement staPessoa = conn.prepareStatement(sqlPessoa,Statement.RETURN_GENERATED_KEYS
               staPessoa.setString(1, PessoaFisica.getNome());
               staPessoa.setString(2, PessoaFisica.getLogradorouro());
               staPessoa.setString(3, PessoaFisica.getCidade());
               staPessoa.setString(4, PessoaFisica.getEstado());
               staPessoa.setString(5, PessoaFisica.getTelefone());
               staPessoa.setString(6, PessoaFisica.getEmail());
               staPessoa.executeUpdate();

               try(ResultSet rSet = staPessoa.getGeneratedKeys()){
                   if(rSet.next()) {pessoaId = rSet.getInt(1);}
               }
```

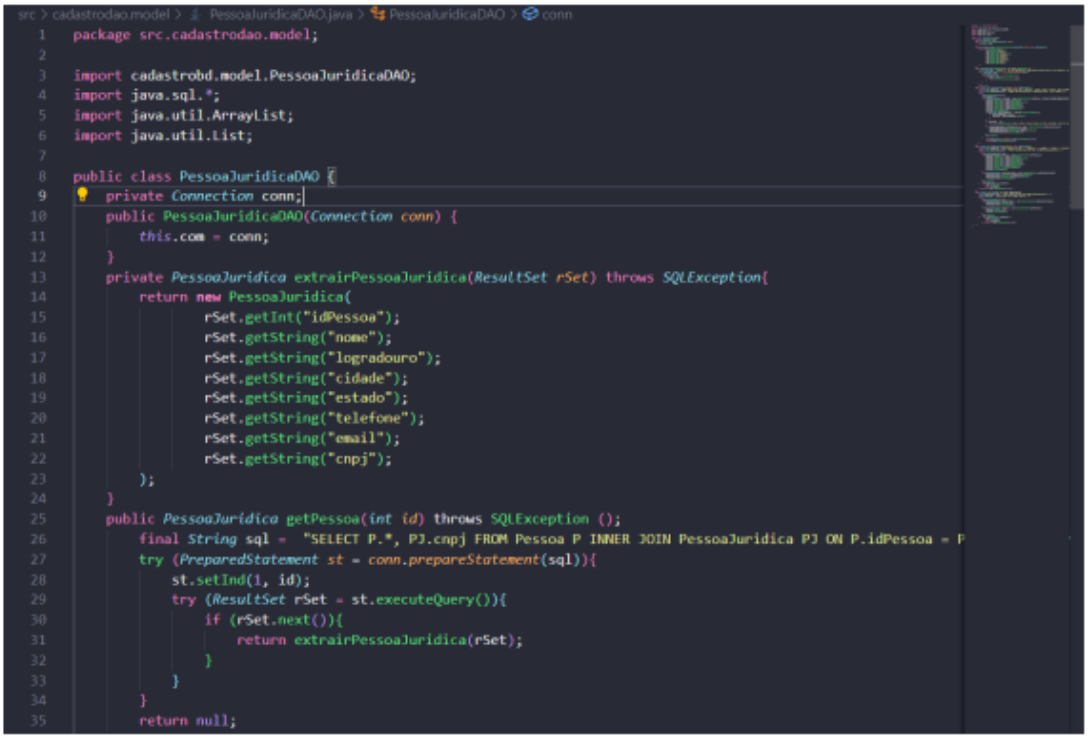
```java
               try(PreparedStatement staPessoaFisica = conn.prepareStatement(sqlPessoaFisica)){
                   staPessoaFisica.setInt(1, pessoaId);
                   staPessoaFisica.setString(2, PessoaFisica.getCpf());
                   staPessoaFisica.executeUpdate();
               }

               conn.commit();
    }catch(
SQLException exception)
{
    conn.rollback();
    throw exception;
}finally{
                   catch.setAutoCommit(true);}
}

public void alterar(PessoaFisica pf) throws SQLException {
    String sqlPessoa = "UPDATE Pessoa Set nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, em
    String sqlPessoaFisica = "UPDATE PessoaFisica SET cpf = ? WHERE idPessoaFisica = ? ";

    try{
        conn.setAutoCommit(false);
            try ( PreparedStatement staPessoa = conn.prepareStatement(sqlPessoa)){
                staPessoa.setString(1, PessoaFisica.getNome());
                staPessoa.setString(2, PessoaFisica.getLogradorouro());
                staPessoa.setString(3, PessoaFisica.getCidade());
                staPessoa.setString(4, PessoaFisica.getEstado());
                staPessoa.setString(5, PessoaFisica.getTelefone());
                staPessoa.setString(6, PessoaFisica.getEmail());
                staPessoa.setInt(7, PessoaFisica.getId());
                staPessoa.executeUpdate();
            }

            try(PreparedStatement staPessoaFisica = conn.prepareStatement(sqlPessoaFisica)){
                staPessoaFisica.setString(1, PessoaFisica.getCpf());
                staPessoaFisica.setInt(2, PessoaFisica.getId());
```

```
                staPessoaFisica.executeUpdate();
            }

            conn.commit();
        }   catch(SQLException exception) {
                conn.rollback();
                throw exception;
            } finally { conn.setAutoCommit(true);}
    }

public void excluir(Integer id) thorws SQLException{
        String sqlPessoaFisica = "Delete FROM PessoaFisica WHERE idPessoaFisica = ?";
        String sqlPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";

        try{
            conn.setAutoCommit(false);
            try(PreparedStatement staPessoaFisica = conn.prepareStatement(sqlPessoaFisica)){
                staPessoaFisica.setInt(1,id);
                staPessoaFisica.executeUpdate();
            }

            try(PreparedStatement staPessoa = conn.prepareStatement(sqlPessoa)){
                staPessoa.setInt(1, id);
                staPessoa.executeUpdate();
            }

            conn.commit();
        }    catch(SQLException exception) {
                conn.rollback();
                throw exception;
            } finally { conn.setAutoCommit(true);}
    }

public PessoaFisica getPessoa(Integer id) throws SQLException {
        String sql = "SELECT Pessoa.idPessoa, Pessoa.nome, Pessoa.logradouro, Pessoa.cidade, Pessoa.estado,
        try(PreparedStatement sta = conn.prepareStatement(sql)){
        sta.setInt(1, id);
```

```
        sta.setInt(1, id);
        try (ResultSet rSet = sta.executeQuery()){
            if (rSet.next()){
                return new PessoaFisica(
                    rSet.getInt("idPessoa");
                    rSet.getString("nome");
                    rSet.getString("logradouro");
                    rSet.getString("cidade");
                    rSet.getSpring("estado");
                    rSet.getString("telefone");
                    rSet.getstring("email");
                    rSet.getstring("cpf");
                );
            }
        }
    }
    return null;
}

public list<PessoaFisica> getPessoas() throws SQLException{
    list<PessoaFisica> list = new ArrayList<>();
    String sql = "SELECT p.*, pf.cpf FROM Pessoa AS p INNER JOIN PessoaFisica AS pf ON p.idPessoa = pf.idPes
    Try(PreparedStatement sta = conn.prepareStatement(sql));
        ResultSet rSet = (st.executeQuery()) {
            while (rSet.next()) {
                list.add(new PessoaFisica(
                    rSet.getInt("idPessoa");
                    rSet.getString("nome");
                    rSet.getString("logradouro");
                    rSet.getString("cidade");
                    rSet.getString("estado");
                    rSet.getString("telefone");
                    rSet.getString("email");
                    rSet.getString("cpf");
                ));
            }
        }
```

```
    return null;
}

public list<PessoaFisica> getPessoas() throws SQLException{
    list<PessoaFisica> list = new ArrayList<>();
    String sql = "SELECT p.*, pf.cpf FROM Pessoa AS p INNER JOIN PessoaFisica AS pf ON p.idPessoa = pf.idPes
    Try(PreparedStatement sta = conn.prepareStatement(sql));
        ResultSet rSet = (st.executeQuery()) {
            while (rSet.next()) {
                list.add(new PessoaFisica(
                    rSet.getInt("idPessoa");
                    rSet.getString("nome");
                    rSet.getString("logradouro");
                    rSet.getString("cidade");
                    rSet.getString("estado");
                    rSet.getString("telefone");
                    rSet.getString("email");
                    rSet.getString("cpf");
                ));
            }
        }
        return list;
}
```

## PessoaJuridicaDAO.java:



```
src > cadastrodao.model > PessoaJuridicaDAO.java > PessoaJuridicaDAO > conn
package src.cadastrodao.model;

import cadastrobd.model.PessoaJuridicaDAO;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class PessoaJuridicaDAO {
    private Connection conn;
    public PessoaJuridicaDAO(Connection conn) {
        this.com = conn;
    }
    private PessoaJuridica extrairPessoaJuridica(ResultSet rSet) throws SQLException{
        return new PessoaJuridica(
                rSet.getInt("idPessoa");
                rSet.getString("nome");
                rSet.getString("logradouro");
                rSet.getString("cidade");
                rSet.getString("estado");
                rSet.getString("telefone");
                rSet.getString("email");
                rSet.getString("cnpj");
        );
    }
    public PessoaJuridica getPessoa(int id) throws SQLException ();
        final String sql =  "SELECT P.*, PJ.cnpj FROM Pessoa P INNER JOIN PessoaJuridica PJ ON P.idPessoa = P
        try (PreparedStatement st = conn.prepareStatement(sql)){
            st.setInd(1, id);
            try (ResultSet rSet = st.executeQuery()){
                if (rSet.next()){
                    return extrairPessoaJuridica(rSet);
                }
            }
        }
        return null;
```

src > cadastrodao.model > PessoaJuridicaDAO.java > PessoaJuridicaDAO > conn

```
38        final String sqlPessoaJuridica = "INSERT INTO PessoaJuridica (idPessoaJuridica, cnpj) VALUES(?,?)";
39        try {
40            conn.setAutoCommit(false);
41            int pessoaid = 0;
42            try (PreparedStatement stPessoa = conn.prepareStatement(sqlPessoa, Statement.RETURN_GENERATED_KEYS)
43                stPessoa.setString(1, pessoa.getNome());
44                stPessoa.setString(2, pessoa.getLogradorouro());
45                stPessoa.setString(3, pessoa.getCidade());
46                stPessoa.setString(4, pessoa.getEstado());
47                stPessoa.setString(5, pessoa.getTelefone());
48                stPessoa.setString(6, pessoa.getEmail());
49                stPessoa.executeUpdate();
50                try(ResultSet generatedKeySet = stPessoa.getGeneratedKeys()){
51                    if( generatedKeySet.next()){
52                        pessoaId = generatedKeySet.getInt(1);
53                    }
54                }
55                if (pessoaId == 0){
56                    throw new SQLDataException("Ops! Falha ao inserir pessoa, ID não foi gerado ou não foi enco
57                }
58                try (preparedStatement stPessoaJuridica = conn. prepareStatement(sqlPessoaJuridica)){
59                    stPessoaPessoaJuridica.setInt(1, pessoaid);
60                    stPessoaPessoaJuridica.setString(2, pessoa.getCnpj());
61                    stPessoaPessoaJuridica.executeUpdate();
62                }
63                conn.commit();
64            }
65                catch(SQLException exception) {conn.setAutoCommit(true);}
66                finally{conn.setAutoCommit(true);}
67        }
68    }
69    public void alterar(PessoaJuridica pessoa) throws SQLException{
70        final String sqlPessoa = "UPDATE Pessoa Set nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone
71        final String sqlPessoaJuridica = "UPDATE PessoaJuridica SET cnpj =? WHERE idPessoaJuridica = ?";
72        try{
73            conn.setAutoCommit(false);
```

src > cadastrodao.model > PessoaJuridicaDAO.java > PessoaJuridicaDAO > alterar(PessoaJuridica)

```
74            try(preparedStatement stPessoa = conn.prepareStatement(sqlPessoa)){
75                stPessoa.setString(1, pessoa.getNome());
76                stPessoa.setString(2, pessoa.getLogradorouro());
77                stPessoa.setString(3, pessoa.getCidade());
78                stPessoa.setString(4, pessoa.getEstado());
79                stPessoa.setString(5, pessoa.getTelefone());
80                stPessoa.setString(6, pessoa.getEmail());
81                stPessoa.setInt(7, PessoaFisicapate());
82                stPessoa.executeUpdate();
83            }
84            try(PreparedStatement stPessoaJuridica = conn.prepareStatement(sqlPessoaJuridica)){
85                stPessoaJuridica.setString(1, PessoaFisica.getCpf());
86                stPessoaJuridica.setInt(2, PessoaFisica.getId());
87                stPessoaJuridica.executeUpdate();
88            }
89            conn.commit();
90        }   catch(SQLException exception) {
91                conn.rollback();
92                throw exception;
93        } finally { conn.setAutoCommit(true);}
94    }
95    public void excluir(Integer id) thorws SQLException{
96        String sqlPessoaJuridica = "Delete FROM PessoaJuridica WHERE idPessoaJuridica = ?";
97        String sqlPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";
98        try{
99            conn.setAutoCommit(false);
100           try(PreparedStatement stPessoaFisica = conn.prepareStatement(sqlPessoaJuridica)){
101               stPessoaJuridica.setInt(1, id);
102               stPessoaJuridica.executeUpdate();
103           }
104           try(PreparedStatement stPessoa = conn.prepareStatement(sqlPessoa)){
105               stPessoa.setInt(1, id);
106               stPessoa.executeUpdate();
107           }
108           conn.commit();
109       }    catch(SQLException exception) {
110               conn.rollback();
```

```java
public void excluir(Integer id) throws SQLException{
    String sqlPessoaJuridica = "Delete FROM PessoaJuridica WHERE idPessoaJuridica = ?";
    String sqlPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";
    try{
        conn.setAutoCommit(false);
        try(PreparedStatement stPessoaFisica = conn.prepareStatement(sqlPessoaJuridica)){
            stPessoaJuridica.setInt(1, id);
            stPessoaJuridica.executeUpdate();
        }
        try(PreparedStatement stPessoa = conn.prepareStatement(sqlPessoa)){
            stPessoa.setInt(1, id);
            stPessoa.executeUpdate();
        }
        conn.commit();
    }    catch(SQLException exception) {
            conn.rollback();
            throw exception;
        } finally { conn.setAutoCommit(true);}
}
```

## ConectoBD.java:

src > cadastro.model.util > ConectorBD.java > ConectorBD > loadProperties()

```java
package src.cadastro.model.util;

import java.io.*;
import java.util.Properties;

import javax.management.RuntimeErrorException;

import java.util.ArrayList;
import java.sql.SQLException;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.Connection;

public class ConectorBD {
    private static Connection conn = null;

    public static Connection gConnection() {
        if (conn == null) {
            try {
                Properties propt = loadProperties();
                String url = propt.getProperty("url");
                String login = propt.getProperty("login");
                String senha = propt.getProperty("url");
                conn = DriverManager.getConnection(url, login, senha);

            } catch (SQLException exception) {
                throw new RuntimeErrorException(
                        "Ops! erro ao obter conexão com o banco de dados: " + exception.getMessager(), except
            }
        }
        return conn;
    }
```

src > cadastro.model.util > ConectorBD.java > ConectorBD

```java
public static Connection gConnection() {
    if (conn == null) {
        try {
            Properties propt = loadProperties();
            String url = propt.getProperty("url");
            String login = propt.getProperty("login");
            String senha = propt.getProperty("url");
            conn = DriverManager.getConnection(url, login, senha);

        } catch (SQLException exception) {
            throw new RuntimeErrorException(
                    "Ops! erro ao obter conexão com o banco de dados: " + exception.getMessager(), exception)
        }
    }
    return conn;
}

public static void closeStatement(Statement sta){
    try{
        if(sta != null){ sta.close();}

        catch(SQLException exception){
            throw new RuntimeException("Ops! Erro ao fechar o Statement: " + exception.getMessage(),exceptior
        }
    }
}

public static void closeResultSet(ResultSet rset){
     try{
        if(rset != null){rset.close();}

        catch(SQLException exception){
            throw new RuntimeException("Ops! Erro ao fechar o ResultSet: " + exception.getMessage(),exceptior
        }
    }
}
```

src > cadastro.model.util > ConectorBD.java > ConectorBD

```java
> public static Connection gConnection() {···

> public static void closeStatement(Statement sta){···

> public static void closeResultSet(ResultSet rset){···

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

         catch(SQLException exception){
             throw new RuntimeException("Ops! Erro ao fechar a conexão com o Banco de Dados: " + exception.get
         }
     }
 }

 public static Properties loadProperties(){
     try(InputStream is = ConectorBD.class.getClassLoader().getResourceAsStream("properties.db")){
         Properties propt = new Properties();
         if(is != null){propt.load(is);}
         else{ throw new FileNotFoundException("Ops! Arquivo properties.db não encontrado no classpath.");}

         return propt;

         catch(SQLException exception){
             throw new RuntimeException("Ops! Erro ao carregar as propriedades a conexão do Banco de Dados: "
         }
     }
 }
```

## SequenceManager.java:

src > cadastro.modelUtil > SequenceManager.java > ...

```java
package src.cadastro.model.util;

public class BDException
extens RuntimeException{

    public ExceptionDB(String Mens){
        super(mens);
    }

}

```

## CadastroBDTeste.java:

src > cadastrobdteste.model > cadastroBDTeste.java > cadastroBDTeste > exibirPessoasFisicas(PessoaFisicaDAO)

```java
package src.cadastrobdteste.model;

import cadastrobd.modelbd.model.Pessoa;
import cadastrodb.modelbd.model.PessoaFisica;
import cadastrodb.modelbd.model.PessoaJuridica;
import cadastrodao.model.PessoaFisicaDAO;
import cadastrodao.model.PessoaJuridicaDAO;
import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;

import java.sql.Connection;
import java.sql.SQLException;
import java.util.list;
import java.util.Scanner;

public class cadastroBDTeste {
    Run | Debug
    public static void main(String[] args){
        Scanner scanner = new Scanner(System.in);
        Connection connection = ConectorBD.getConnection();
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(connection);
        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(connection);

        int opcao;
        do{
            System.out.println("================================");
            System.out.println("Selecione a opção desejada: ");
            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("6 - Exibir Pessoa Física");
            System.out.println("5 - Exibir Pessoa Jurídica");
            System.out.println("0 - Finalizar Programa");
            System.out.println("================================");
            opcao = scanner.nextInt();
```

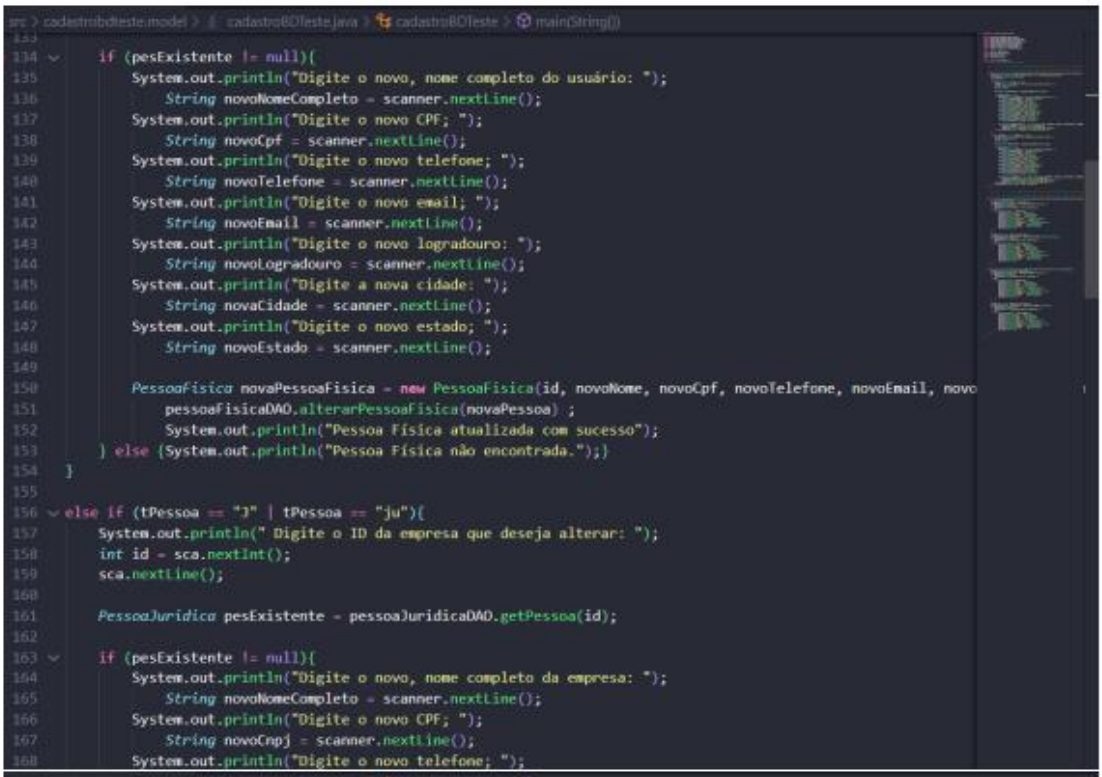
```
36      opcao = scanner.nextInt();
37      scanner.nextLine();
38      try{
39          switch(opcao){
40              case 1:
41                  system.out.println("F - Pessoa Física | J - Pessoa Jurídica");
42                  char tInclusao = scanner.next().charAt(opcao);
43                  scanner.nextLine();
44                  if (tInclusao == "F" | tInclusao == "fi"){cadastrarPessoaFisica(pessoaFisicaDAO, scanner);}
45                  else if (tInclusao == "J" | tInclusao == "ju"){cadastrarPessoaJuridica(pessoaFisicaJuridicaDAO,
46                  break;
47              case 2:
48                  alterarPessoa(pessoaFisicaDAO, pessoaJuridicaDAO, scanner);
49                  break;
50              case 3:
51                  excluirPessoa(pessoaFisicaDAO, pessoaJuridicaDAO, scanner);
52                  break;
53              case 4:
54                  buscarPeloId(pessoaFisicaDAO, pessoaJuridicaDAO, scanner);
55                  break;
56
57              case 5:
58                  exibirTodos(pessoaFisicaDAO, pessoaJuridicaDAO);
59                  break;
60              case 6:
61                  exibirPessoaFisica(pessoaFisicaDAO);
62                  break;
63              case 7:
64                   exibirPessoaJuridica(pessoaFisicaDAO, pessoaJuridicaDAO, scanner);
65                  break;
66              case 0:
67                  System.out.println("Encerrando o programa. Volte sempre que precisar!!!");
68                  break;
69              default: System.out.println("Ops... Opção inválida. Tente novamente!");
70          }  catch(SQLException exception){
```

```
37              scanner.nextLine();
38              try{
39                  switch(opcao){
40  >                   case 1:···
47  >                   case 2:···
50  >                   case 3:···
53  >                   case 4:···
57  >                   case 5:···
60  >                   case 6:···
63  >                   case 7:···
66  >                   case 0:···
69                      default: System.out.println("Ops... Opção inválida. Tente novamente!");
70                  }  catch(SQLException exception){
71                      System.out.println("Ops... Erro no Banco de Dados: " + exception.getMessage());
72                      exception.printStackTrace();}
73                  while (opcao != 0);
74              }
75          }
76      }
```

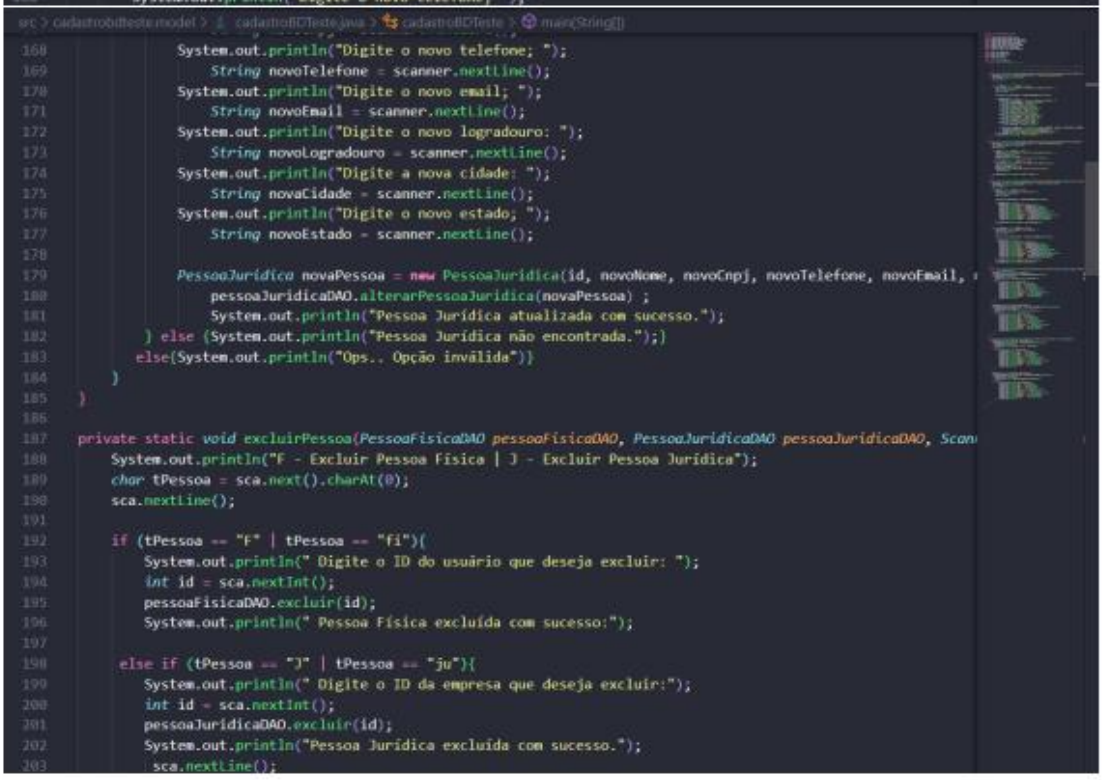
```java
Run | Debug
public static void main(String[] args){ ···

private static void cadastroPessoaFisica(PessoaFisicaDAO pessoaFisicaDAO, Scanner sca) throws SQLException {
    System.out.println("Digite o nome completo do usuário: ");
    String nomeCompleto = scanner.nextLine();
    System.out.println("Digite o CPF; ");
    String cpf = scanner.nextLine();
    System.out.println("Digite o telefone; ");
    String telefone = scanner.nextLine();
    System.out.println("Digite o email; ");
    String email = scanner.nextLine();
    System.out.println("Digite o logradouro: ");
    String logradouro = scanner.nextLine();
    System.out.println("Digite a cidade: ");
    String cidade = scanner.nextLine();
    System.out.println("Digite o estado; ");
    String estado = scanner.nextLine();

    PessoaFisica newPessoaFisica = new PessoaFisica(0, nome, cpf, telefone, email, logradouro, cidade, estad
    pessoaFisicaDAO.inserirPessoaFisica(newPessoaFisica);
    System.out.println("Pessoa Física cadastrada com sucesso");
}

private static void cadastroPessoaJuridica(PessoaFisicaJuridicaDAO pessoaFisicaJuridicaDAO, Scanner sca)
        throws SQLException {
    System.out.println("Digite o nome completo da Empresa: ");
    String nomeCompleto = scanner.nextLine();
    System.out.println("Digite o CNPJ; ");
    String cpf = scanner.nextLine();
    System.out.println("Digite o telefone; ");
    String telefone = scanner.nextLine();
```

src > cadastrobdteste.model > cadastroBDteste.java > cadastroBDteste > main(String[])

```java
private static void cadastroPessoaJuridica(PessoaFisicaJuridicaDAO pessoaFisicaJuridicaDAO, Scanner sca)
        throws SQLException {
    System.out.println("Digite o nome completo da Empresa: ");
    String nomeCompleto = scanner.nextLine();
    System.out.println("Digite o CNPJ; ");
    String cpf = scanner.nextLine();
    System.out.println("Digite o telefone; ");
    String telefone = scanner.nextLine();
    System.out.println("Digite o email; ");
    String email = scanner.nextLine();
    System.out.println("Digite o logradouro: ");
    String logradouro = scanner.nextLine();
    System.out.println("Digite a cidade: ");
    String cidade = scanner.nextLine();
    System.out.println("Digite o estado; ");
    String estado = scanner.nextLine();

    PessoaFisica newPessoaFisicaJuridica = new PessoaFisica(0, nome, cnpj, telefone, email, logradouro, cida
            estado);
    pessoaFisicaDAO.inserirPessoaFisicaJuridica(newPessoaFisicaJuridica);
    System.out.println("Empresa cadastrada com sucesso");
}

private static void alterarPessoa(PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO, Scan
    System.out.println("F - Alterar Pessoa Física | J - Alterar Pessoa Jurídica");
    char tPessoa = sca.next().charAt(0);
    sca.nextLine();

    if (tPessoa == "F" | tPessoa == "fi"){
        System.out.println(" Digite o ID do usuário que deseja alterar: ");
        int id = sca.nextInt();
        sca.nextLine();

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



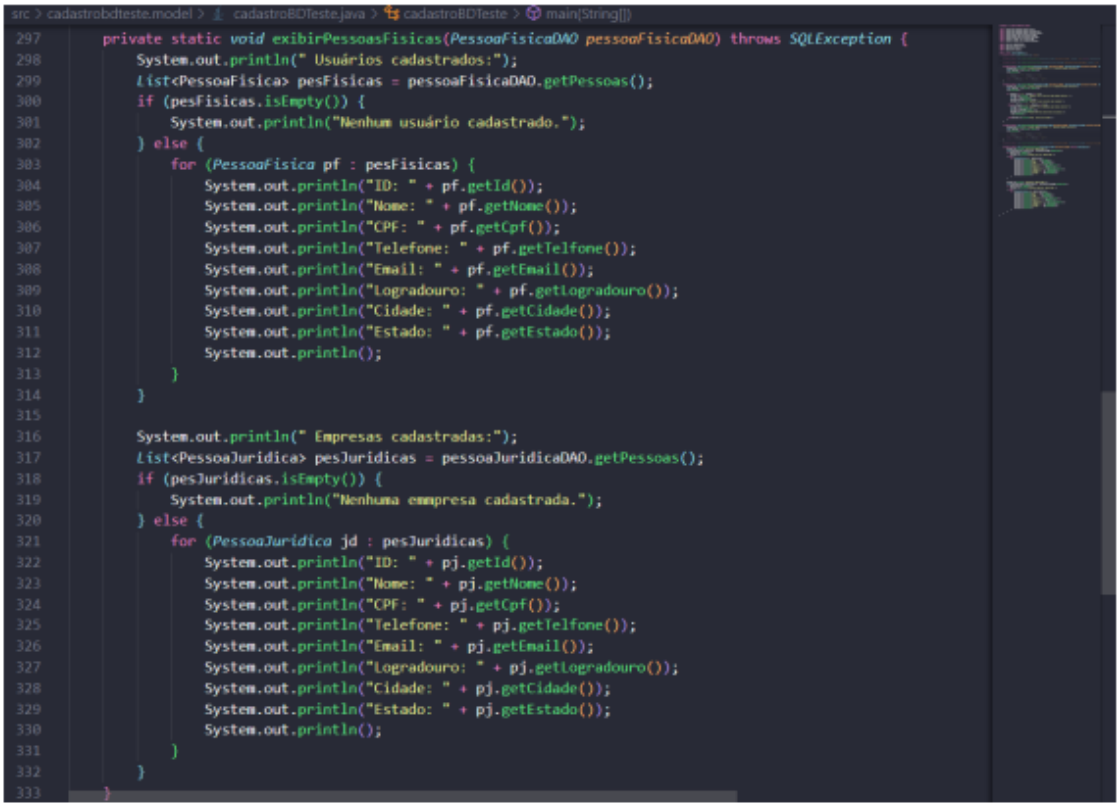
```
src > cadastrobdteste.model > cadastroBDTeste.java > cadastroBDTeste > main(String[])
134     if (pesExistente != null){
135         System.out.println("Digite o novo, nome completo do usuário: ");
136             String novoNomeCompleto = scanner.nextLine();
137         System.out.println("Digite o novo CPF; ");
138             String novoCpf = scanner.nextLine();
139         System.out.println("Digite o novo telefone; ");
140             String novoTelefone = scanner.nextLine();
141         System.out.println("Digite o novo email; ");
142             String novoEmail = scanner.nextLine();
143         System.out.println("Digite o novo logradouro: ");
144             String novoLogradouro = scanner.nextLine();
145         System.out.println("Digite a nova cidade: ");
146             String novaCidade = scanner.nextLine();
147         System.out.println("Digite o novo estado; ");
148             String novoEstado = scanner.nextLine();
149
150         PessoaFisica novaPessoaFisica = new PessoaFisica(id, novoNome, novoCpf, novoTelefone, novoEmail, novo
151             pessoaFisicaDAO.alterarPessoaFisica(novaPessoa) ;
152             System.out.println("Pessoa Física atualizada com sucesso");
153     } else {System.out.println("Pessoa Física não encontrada.");}
154 }
155
156 else if (tPessoa == "J" | tPessoa == "ju"){
157     System.out.println(" Digite o ID da empresa que deseja alterar: ");
158     int id = sca.nextInt();
159     sca.nextline();
160
161     PessoaJuridica pesExistente = pessoaJuridicaDAO.getPessoa(id);
162
163     if (pesExistente != null){
164         System.out.println("Digite o novo, nome completo da empresa: ");
165             String novoNomeCompleto = scanner.nextLine();
166         System.out.println("Digite o novo CPF; ");
167             String novoCnpj = scanner.nextline();
168         System.out.println("Digite o novo telefone; ");
```



```
src > cadastrobdteste.model > cadastroBDTeste.java > cadastroBDTeste > main(String[])
168             System.out.println("Digite o novo telefone; ");
169                 String novoTelefone = scanner.nextLine();
170             System.out.println("Digite o novo email; ");
171                 String novoEmail = scanner.nextLine();
172             System.out.println("Digite o novo logradouro: ");
173                 String novoLogradouro = scanner.nextLine();
174             System.out.println("Digite a nova cidade: ");
175                 String novaCidade = scanner.nextLine();
176             System.out.println("Digite o novo estado; ");
177                 String novoEstado = scanner.nextLine();
178
179             PessoaJuridica novaPessoa = new PessoaJuridica(id, novoNome, novoCnpj, novoTelefone, novoEmail,
180                 pessoaJuridicaDAO.alterarPessoaJuridica(novaPessoa) ;
181                 System.out.println("Pessoa Jurídica atualizada com sucesso.");
182         } else {System.out.println("Pessoa Jurídica não encontrada.");}
183        else{System.out.println("Ops.. Opção inválida")}
184     }
185 }
186
187 private static void excluirPessoa(PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO, Scan
188     System.out.println("F - Excluir Pessoa Física | J - Excluir Pessoa Jurídica");
189     char tPessoa = sca.next().charAt(0);
190     sca.nextline();
191
192     if (tPessoa == "f" | tPessoa == "fi"){
193         System.out.println(" Digite o ID do usuário que deseja excluir: ");
194         int id = sca.nextInt();
195         pessoaFisicaDAO.excluir(id);
196         System.out.println(" Pessoa Física excluída com sucesso:");
197
198      else if (tPessoa == "J" | tPessoa == "ju"){
199         System.out.println(" Digite o ID da empresa que deseja excluir:");
200         int id = sca.nextInt();
201         pessoaJuridicaDAO.excluir(id);
202         System.out.println("Pessoa Jurídica excluída com sucesso.");
203          sca.nextline();
```

```
src > cadastrobdteste.model > J cadastroBDTeste.java > cadastroBDTeste > main(String[])
205             else{System.out.println("Ops.. Opção inválida")} }
206         }
207     }
208
209     private static void buscarPessoaPeloId(PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaD
210         System.out.println("F - buscar Pessoa Física | J - Buscar Pessoa Jurídica");
211         char tPessoa = sca.next().charAt(0);
212         sca.nextLine();
213
214 >       if (tPessoa == "F" | tPessoa == "fi"){ ···
234
235         else if (tPessoa == "J" | tPessoa == "ju"){
236             System.out.println(" Digite o ID da empresa: ");
237             int id = sca.nextInt();
238             sca.nextLine();
239
240             PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);
241
242             if(pessoaFisica != null){
243                 System.out.println("Detalhes do Empresa:");
244                 System.out.println("ID: " + pessoaJuridica.getId());
245                 System.out.println("Nome: " + pessoaJuridica.getNome());
246                 System.out.println("CNPJ: " + pessoaJuridica.getCpf());
247                 System.out.println("Telefone: " + pessoaJuridica.getTelfone());
248                 System.out.println("Email: " + pessoaJuridica.getEmail());
249                 System.out.println("Logradouro: " + pessoaJuridica.getLogradouro());
250                 System.out.println("Cidade: " + pessoaJuridica.getCidade());
251                 System.out.println("Estado: " + pessoaJuridica.getEstado());
252             }
253             else{System.out.println("Empresa não encontrada.");}
254             else{System.out.println("Opção inválida.");}
255         }
256     }
```

```
src > cadastrobdteste.model > J cadastroBDTeste.java > cadastroBDTeste > main(String[])
260         System.out.println(" Usuário cadastrado:");
261         List<PessoaFisica> pesFisicas = pessoaFisicaDAO.getPessoas();
262         if (pesFisicas.isEmpty()) {
263             System.out.println("Nenhum usuário cadastrado.");
264         } else {
265             for (PessoaFisica pf : pesFisicas) {
266                 System.out.println("ID: " + pf.getId());
267                 System.out.println("Nome: " + pf.getNome());
268                 System.out.println("CPF: " + pf.getCpf());
269                 System.out.println("Telefone: " + pf.getTelfone());
270                 System.out.println("Email: " + pf.getEmail());
271                 System.out.println("Logradouro: " + pf.getLogradouro());
272                 System.out.println("Cidade: " + pf.getCidade());
273                 System.out.println("Estado: " + pf.getEstado());
274                 System.out.println();
275             }
276         }
277
278         System.out.println(" Empresa cadastrada:");
279         List<PessoaJuridica> pesJuridicas = pessoaJuridicaDAO.getPessoasJuridicas();
280         if (pesJuridicas.isEmpty()) {
281             System.out.println("Nenhuma empresa cadastrada.");
282         } else {
283             for (PessoaJuridica pj : pesJuridicas) {
284                 System.out.println("ID: " + pj.getId());
285                 System.out.println("Nome: " + pj.getNome());
286                 System.out.println("CNPJ: " + pj.getCnpj());
287                 System.out.println("Telefone: " + pj.getTelfone());
288                 System.out.println("Email: " + pj.getEmail());
289                 System.out.println("Logradouro: " + pj.getLogradouro());
290                 System.out.println("Cidade: " + pj.getCidade());
291                 System.out.println("Estado: " + pj.getEstado());
292                 System.out.println();
293             }
294         }
295     }
296
```


src > cadastrobdteste.model > cadastroBDTeste.java > cadastroBDTeste > main(String[])
297 private static void exibirPessoasFisicas(PessoaFisicaDAO pessoaFisicaDAO) throws SQLException {
298 System.out.println(" Usuários cadastrados:");
299 List<PessoaFisica> pesFisicas = pessoaFisicaDAO.getPessoas();
300 if (pesFisicas.isEmpty()) {
301 System.out.println("Nenhum usuário cadastrado.");
302 } else {
303 for (PessoaFisica pf : pesFisicas) {
304 System.out.println("ID: " + pf.getId());
305 System.out.println("Nome: " + pf.getNome());
306 System.out.println("CPF: " + pf.getCpf());
307 System.out.println("Telefone: " + pf.getTelfone());
308 System.out.println("Email: " + pf.getEmail());
309 System.out.println("Logradouro: " + pf.getLogradouro());
310 System.out.println("Cidade: " + pf.getCidade());
311 System.out.println("Estado: " + pf.getEstado());
312 System.out.println();
313 }
314 }
315
316 System.out.println(" Empresas cadastradas:");
317 List<PessoaJuridica> pesJuridicas = pessoaJuridicaDAO.getPessoas();
318 if (pesJuridicas.isEmpty()) {
319 System.out.println("Nenhuma emmpresa cadastrada.");
320 } else {
321 for (PessoaJuridica jd : pesJuridicas) {
322 System.out.println("ID: " + pj.getId());
323 System.out.println("Nome: " + pj.getNome());
324 System.out.println("CPF: " + pj.getCpf());
325 System.out.println("Telefone: " + pj.getTelfone());
326 System.out.println("Email: " + pj.getEmail());
327 System.out.println("Logradouro: " + pj.getLogradouro());
328 System.out.println("Cidade: " + pj.getCidade());
329 System.out.println("Estado: " + pj.getEstado());
330 System.out.println();
331 }
332 }
333 }

## Resultado da execução dos códigos:

```
 ===================
 1 - Incluir Pessoa
 2 - Alterar Pessoa
 3 - Excluir Pessoa
 4 - Buscar pelo Id
 5 - Exibir Todos
 6 - Exibir Somente Pessoas Físicas
 7 - Exibir Somente Pessoas Jurídicas
 0 - Finalizar Programa
 ===================
 0
 Encerrando o programa.
```

```
=====================================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Exibir Somente Pessoas Físicas
7 - Exibir Somente Pessoas Jurídicas
0 - Finalizar Programa
=====================================================
4
F - Buscar Pessoa Física | J - Buscar Pessoa Jurídica
J
Digite o ID da Pessoa Jurídica:
21
Detalhes da Pessoa Jurídica:
ID: 21
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1111-1111
Email: email@example.com
CNPJ: 1111111111111
=====================================================
```

```
======================================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Exibir Somente Pessoas Fisicas
7 - Exibir Somente Pessoas Jurídicas
0 - Finalizar Programa
======================================================
3
F - Excluir Pessoa Física | J - Excluir Pessoa Jurídica
F
Digite o ID da Pessoa Física a ser excluída:
35
Pessoa Física excluída com sucesso.
======================================================
```

```
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Exibir Somente Pessoas Físicas
7 - Exibir Somente Pessoas Jurídicas
0 - Finalizar Programa
==================
5
PessoasJuridicasCadastradas:

ID: 33
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 2222-2222
Email: email@exemple.com
CNPJ: 11111111111111

ID: 7
Nome: Joao
Logradouro: Rua 12, casa 3, Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: email@example.com
CPF: 11111111111
```

Importância dos componentes de middleware, como o JDBC:

- O JDBC (Java Database Connectivity) é um middleware crucial para a comunicação entre aplicativos Java e bancos de dados. Ele fornece uma interface padronizada para acesso a dados, independentemente do banco de dados subjacente, aumentando a portabilidade e facilitando a manutenção.

Diferença no uso de Statement ou PreparedStatement para a manipulação de dados:

- **Statement** é usado para executar consultas SQL simples.
- **PreparedStatement** é preferido para consultas parametrizadas. A principal diferença está na prevenção de injeção de SQL e no ganho de desempenho e pode ser pré-compilado.

Como o padrão DAO melhora a manutenibilidade do software:

- O padrão DAO (Data Access Object) separa a lógica de acesso a dados do restante do código, facilitando a manutenção. Isso permite alterações na camada de persistência sem afetar outras partes do sistema, promovendo uma melhor organização e modularidade.

Reflexo da herança no banco de dados em um modelo estritamente relacional:

- Em um modelo estritamente relacional, a herança é frequentemente representada usando tabelas separadas para cada classe concreta. Isso resulta em uma estrutura de várias tabelas relacionadas, onde cada tabela representa uma classe específica ou uma subclasse.

Diferenças entre persistência em arquivo e persistência em banco de dados:

- A persistência em arquivo armazena dados em arquivos no sistema de arquivos, sendo mais simples, mas menos eficiente para consultas complexas. A persistência em banco de dados utiliza sistemas gerenciadores de banco de dados, proporcionando maior segurança, concorrência e capacidade de consultas avançadas.

Simplificação da impressão de valores usando operador lambda no Java mais recente:

- O uso de operadores lambda simplifica a expressão de operações em coleções, como listas. Com a introdução de métodos como forEach em coleções, a impressão de valores contidos nas entidades pode ser realizada de forma mais concisa e legível.

Razão pela qual métodos acionados diretamente pelo método main precisam ser marcados como static:

- Métodos acionados diretamente pelo método main precisam ser static porque o método main é estático. Métodos estáticos pertencem à classe, não a instâncias específicas, e podem ser chamados sem criar uma instância da classe. O método main é o ponto de entrada do programa e precisa ser estático para ser chamado sem criar um objeto da classe principal.

**Conclusão:**

Esses conceitos fundamentais, como acesso a dados, design de padrões, herança em bancos de dados, persistência e novos recursos da linguagem, são cruciais para o desenvolvimento eficiente e sustentável de software em Java. Compreender esses elementos contribui para a construção de sistemas robustos, modulares e de fácil manutenção.