RPG0016 - BackEnd sem banco não tem

Kervini Ribeiro da Silva - 202301206073

Polo Centro – Araraquara – SP
Back-end Sem Banco Não Tem – 9001 – 3º Semestre

Objetivo da Prática

Implementar um sistema de persistência de dados, usando o middleware JDBC, criar um ORM conectando no banco de dados SQL Server usando padrão DAO. A partir disso criar uma interface em texto para alimentar o banco de dados e testar as funcionalidades.

GITHUB: https://github.com/Kervini/Mundo3-Nivel3.git

1º Procedimento | Mapeamento Objeto-Relacional e DAO

```java
package cadastrobd.model;
public abstract class Pessoa {
    private int id;
    private String nome;
    private String logradouro;
    private String cidade;
    private String estado;
    private String telefone;
    private String email;
    public Pessoa(){}
    public Pessoa(int id, String nome, String logradouro, String cidade, String estado, String telefone, String email){
        this.id = id;
        this.nome = nome;
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
        this.telefone = telefone;
        this.email = email;
    }
    public String exibir(){
        return "Id: " + id + "\tNome:" + nome + "\nLogradouro: " + logradouro + "\nCidade: " + cidade +
                ", estado: " + estado + ", telefone: " + telefone + "\nEmail: " + 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;
    }
}
```

```java
package cadastrobd.model;

public class PessoaFisica extends Pessoa{
    private String cpf;

    public PessoaFisica(){
        super();
    }

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

    @Override
    public String exibir() {
        return super.exibir() + "\nCPF: " + cpf;
    }

    public String getCpf() {
        return cpf;
    }
}
```

```java
package cadastrobd.model;

public class PessoaJuridica extends Pessoa{
    private String cnpj;

    public PessoaJuridica() {
        super();
    }

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

    @Override
    public String exibir() {
        return super.exibir() + "\nCNPJ: " + cnpj;
    }

    public String getCnpj() {
        return cnpj;
    }

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

```java
package cadastro.model;

import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;
import cadastrobd.model.PessoaFisica;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

public class PessoaFisicaDAO {
    private final ConectorBD connector;
    private final SequenceManager sm;

    public PessoaFisicaDAO() throws SQLException{
        connector = new ConectorBD();
        sm = new SequenceManager(connector.getConnection());
    }

    public PessoaFisica getPessoa(int id) throws SQLException{
        PessoaFisica pf = null;

        String sql = "select p.id, p.nome, p.logradouro, p.cidade, p.estado, p.telefone, p.email, pf.cpf "
                + "from pessoa p inner join pessoa_fisica pf on p.id = pf.idpessoa where p.id = ?";
        try(PreparedStatement ps = connector.getPrepared(sql)){
            ps.setInt(1, id);
            ResultSet rs = ps.executeQuery();

            if(rs.next()){
                pf = new PessoaFisica(rs.getInt("id"), rs.getString("nome"), rs.getString("logradouro"), rs.getString("cidade"),
                    rs.getString("estado"), rs.getString("telefone"), rs.getString("email"), rs.getString("cpf"));
            }
        }

        return pf;
    }
```

```
    public ArrayList<PessoaFisica> getPessoas() throws SQLException{
        ArrayList<PessoaFisica> pessoas = new ArrayList<>();

        String sql = "select p.id, p.nome, p.logradouro, p.cidade, p.estado, p.telefone, p.email, pf.cpf "
                + "from pessoa p inner join pessoa_fisica pf on p.id = pf.idpessoa";

        try(ResultSet rs = connector.getSelect(sql)){
            while(rs.next()){
                pessoas.add(new PessoaFisica(rs.getInt("id"), rs.getString("nome"), rs.getString("logradouro"),
                        rs.getString("cidade"), rs.getString("estado"), rs.getString("telefone"),
                        rs.getString("email"), rs.getString("cpf")));
            }
        }

        return pessoas;
    }

    public boolean incluir(PessoaFisica pf) throws SQLException{
        boolean retorno;
        int novoid = sm.getValue("sqc_pessoa_id");
        String sql = "insert into pessoa (id, nome, logradouro, cidade, estado, telefone, email) values (?, ?, ?, ?, ?, ? , ?);"
                + " insert into pessoa_fisica(idpessoa, cpf) values (?, ?)";

        try(PreparedStatement ps = connector.getConnection().prepareStatement(sql)){
            ps.setInt(1, novoid);
            ps.setString(2, pf.getNome());
            ps.setString(3, pf.getLogradouro());
            ps.setString(4, pf.getCidade());
            ps.setString(5, pf.getEstado());
            ps.setString(6, pf.getTelefone());
            ps.setString(7, pf.getEmail());
            ps.setInt(8, novoid);
            ps.setString(9, pf.getCpf());

            retorno = ps.executeUpdate() > 0;
        }

        return retorno;//Retorna true se o insert ocorreu com sucesso nas duas tabelas
    }
```

```
77
78      public boolean alterar(PessoaFisica pf) throws SQLException{
79          boolean retorno;
80
81          String sql = "update pessoa set nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ?"
82                  + " where id = ?; "
83                  + "update pessoa_fisica set cpf = ? where idpessoa = ?;";
84
85          try(PreparedStatement ps = connector.getPrepared(sql)){
86              ps.setString(1, pf.getNome());
87              ps.setString(2, pf.getLogradouro());
88              ps.setString(3, pf.getCidade());
89              ps.setString(4, pf.getEstado());
90              ps.setString(5, pf.getTelefone());
91              ps.setString(6, pf.getEmail());
92              ps.setInt(7, pf.getId());
93              ps.setString(8, pf.getCpf());
94              ps.setInt(9, pf.getId());
95
96              retorno = ps.executeUpdate() > 0;
97          }
98
99          return retorno;//Retorna true se o update ocorreu com sucesso nas duas tabelas
100     }
101
102     public boolean excluir(int id) throws SQLException{
103         boolean retorno;
104
105         String sql = "delete from pessoa_fisica where idpessoa = ?; delete from pessoa where id = ?";
106         try(PreparedStatement ps = connector.getPrepared(sql)){
107             ps.setInt(1, id);
108             ps.setInt(2, id);
109
110             retorno = ps.executeUpdate() > 0;
111         }
112
113         return retorno; //Retorna true se o delete ocorreu com sucesso nas duas tabelas
114     }
115 }
```

```java
package cadastro.model;

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

public class PessoaJuridicaDAO {
    private final ConectorBD conector;
    private final SequenceManager sm;

    public PessoaJuridicaDAO() throws SQLException{
        this.conector = new ConectorBD();
        this.sm = new SequenceManager(this.conector.getConnection());
    }

    public PessoaJuridica getPessoa(int id) throws SQLException{
        PessoaJuridica pj = null;

        String sql = "select p.id, p.nome, p.logradouro, p.cidade, p.estado, p.telefone, p.email, pj.cnpj " +
            "from pessoa p inner join pessoa_juridica pj on p.id = pj.idpessoa where p.id = ?";
        try(PreparedStatement ps = conector.getPrepared(sql)){
            ps.setInt(1, id);
            ResultSet rs = ps.executeQuery();

            if(rs.next()){
                pj = new PessoaJuridica(rs.getInt("id"), rs.getString("nome"), rs.getString("logradouro"), rs.getString("cidade"),
                    rs.getString("estado"), rs.getString("telefone"), rs.getString("email"), rs.getString("cnpj"));
            }
        }

        return pj;
    }
```

```java

    public ArrayList<PessoaJuridica> getPessoas() throws SQLException{
        ArrayList<PessoaJuridica> pessoas = new ArrayList<>();
        String sql = "select p.id, p.nome, p.logradouro, p.cidade, p.estado, p.telefone, p.email, pj.cnpj " +
            "from pessoa p inner join pessoa_juridica pj on p.id = pj.idpessoa";

        try(ResultSet rs = conector.getSelect(sql)){
            while(rs.next()){
                pessoas.add(new PessoaJuridica(rs.getInt("id"), rs.getString("nome"), rs.getString("logradouro"),
                        rs.getString("cidade"), rs.getString("estado"), rs.getString("telefone"),
                        rs.getString("email"), rs.getString("cnpj")));
            }
        }

        return pessoas;
    }

    public boolean incluir(PessoaJuridica pj) throws SQLException{
        boolean retorno;
        int novoid = sm.getValue("sqc_pessoa_id");
        String sql = "insert into pessoa (id, nome, logradouro, cidade, estado, telefone, email)"
                + " values (?, ?, ?, ?, ?, ?, ?); insert into pessoa_juridica (idpessoa, cnpj) values(?, ?);";

        try(PreparedStatement ps = conector.getPrepared(sql)){
            ps.setInt(1, novoid);
            ps.setString(2, pj.getNome());
            ps.setString(3, pj.getLogradouro());
            ps.setString(4, pj.getCidade());
            ps.setString(5, pj.getEstado());
            ps.setString(6, pj.getTelefone());
            ps.setString(7, pj.getEmail());
            ps.setInt(8, novoid);
            ps.setString(9, pj.getCnpj());

            retorno = ps.executeUpdate() > 0;
        }

        return retorno;//Retorna true se o insert ocorreu com sucesso nas duas tabelas
    }

```

```java
    public boolean alterar(PessoaJuridica pj) throws SQLException{
        boolean retorno;
        String sql = "update pessoa set nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ?"
                + " where id = ?; "
                + "update pessoa_juridica set cnpj = ? where idpessoa = ?;";

        try(PreparedStatement ps = conector.getPrepared(sql)){
            ps.setString(1, pj.getNome());
            ps.setString(2, pj.getLogradouro());
            ps.setString(3, pj.getCidade());
            ps.setString(4, pj.getEstado());
            ps.setString(5, pj.getTelefone());
            ps.setString(6, pj.getEmail());
            ps.setInt(7, pj.getId());
            ps.setString(8, pj.getCnpj());
            ps.setInt(9, pj.getId());

            retorno = ps.executeUpdate() > 0;
        }
        return retorno; //Retorna true se o update ocorreu com sucesso nas duas tabelas
    }

    public boolean excluir(int id) throws SQLException{
        boolean retorno;
        String sql = "delete from pessoa_juridica where idpessoa = ?;"
                + " delete from pessoa where id = ?;";

        try(PreparedStatement ps = conector.getPrepared(sql)){
            ps.setInt(1, id);
            ps.setInt(2, id);

            retorno = ps.executeUpdate() > 0;
        }

        return retorno; //Retorna true se o delete ocorreu com sucesso nas duas tabelas
    }
}
```

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
public final class ConectorBD {
    private final Connection conn;
    public ConectorBD() throws SQLException {
        this.conn = startConnection();
    }
    private Connection startConnection() throws SQLException {
        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            return DriverManager.getConnection("jdbc:sqlserver://localhost\\SQLEXPRESS:1433;"
                    + "databaseName=loja;encrypt=true;user=loja;password=loja;trustServerCertificate=true;");
        } catch (ClassNotFoundException e) {
            throw new SQLException("Erro ao iniciar o driver de conexao.", e);
        } catch (SQLException e) {
            throw new SQLException("Verifique a URL de conexao.", e);
        }
    }
    public Connection getConnection() {
        return this.conn;
    }
    public PreparedStatement getPrepared(String sql) throws SQLException {
        return this.conn.prepareStatement(sql);
    }
    private Statement getStatement() throws SQLException {
        return this.conn.createStatement();
    }
    public ResultSet getSelect(String sql) throws SQLException {
        return getStatement().executeQuery(sql);
    }
    public void close() throws SQLException {
        this.conn.close();
    }
    public void close(Statement st) throws SQLException {
        st.close();
    }
    public void close(ResultSet rs) throws SQLException {
        rs.close();
    }
```

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

public class SequenceManager {
    private final Connection conn;

    public SequenceManager(Connection conn){
        this.conn = conn;
    }

    public int getValue(String sequenceNome) throws SQLException{
        String sql = "select next value for "+ sequenceNome + " as id";

        try(Statement st = conn.createStatement()){
            ResultSet rs = st.executeQuery(sql);

            if(rs.next())
                return rs.getInt(1);
        }

        return 0;
    }
}
```

```java
package cadastrobd.control;

import cadastro.model.PessoaFisicaDAO;
import cadastro.model.PessoaJuridicaDAO;
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaJuridica;
import java.sql.SQLException;
import java.util.ArrayList;

public class CadastroBDTeste {

    public static void main(String[] args) {
        try {
            //instância os repositórios para persistência de dados
            PessoaFisicaDAO repoFisica =  new PessoaFisicaDAO();
            PessoaJuridicaDAO repoJuridica = new PessoaJuridicaDAO();

            //Instânciar um objeto de Pessoa Fisíca e incluir no banco
            PessoaFisica pf = new PessoaFisica(0, "Kervini", "07 de Setembro", "Rincão",
                    "SP", "996045313", "202301206073@alunos.estacio.br", "12345678910");

            if (repoFisica.incluir(pf)) {
                System.out.println("Cadastrado com sucesso!");
            } else {
                System.out.println("Nao foi possivel realizar o cadastro!");
            }

            //alterando um registro existente do banco
            PessoaFisica pf2 = new PessoaFisica(20, "teste", "Rua do meio", "Aquela",
                    "SP", "987654312", "teste@email.com", "12345678910");

            if (repoFisica.alterar(pf2)) {
                System.out.println("Alterado com sucesso!");
            } else {
                System.out.println("Pessoa nao encontrada!");
            }

            //Listar todas as pessoas fisicas do banco
            ArrayList<PessoaFisica> pessoasfisicas = repoFisica.getPessoas();

            for (PessoaFisica pessoaf : pessoasfisicas) {
                System.out.println(pessoaf.exibir() + "\n");
            }
```

```
            //excluir uma pessoa fisica
            if (repoFisica.excluir(11)) {
                System.out.println("Excluido com sucesso!");
            } else {
                System.out.println("Pessoa não encontrada!");
            }

            //Instânciar uma pessoa juridica e persistir no banco
            PessoaJuridica pj = new PessoaJuridica(0, "Codificando trabalhos",
                    "Rua da minha empresa", "String", "BD", "0101010101", "cript@email.com", "010101010101");

            if (repoJuridica.incluir(pj)) {
                System.out.println("Cadastrado com sucesso!");
            } else {
                System.out.println("Não foi possível cadastrar a empresa!");
            }

            //alterando os dados de uma pessoa juridica
            PessoaJuridica pj2 = new PessoaJuridica(12, "Novo nome fantasia",
                    "Rua bela", "Varchar", "MG", "0202020202", "criptomoedas@email.com", "0202020203");

            if (repoJuridica.alterar(pj2)) {
                System.out.println("Dados da empresa alterados com sucesso!");
            } else {
                System.out.println("Nao foi possivel alterar os dados!");
            }

            //Listar todas as pessoas juridicas do banco
            ArrayList<PessoaJuridica> pessoasjuridicas = repoJuridica.getPessoas();

            for (PessoaJuridica pessoaj : pessoasjuridicas) {
                System.out.println(pessoaj.exibir() + "\n");
            }
```

```
            //excluindo uma pessoa juridica do banco
            if (repoJuridica.excluir(12)) {
                System.out.println("Excluido com sucesso!");
            } else {
                System.out.println("Pessoa não encontrada!");
            }

        } catch (SQLException ex) {
            //Captura todas as exceções do tipo SQLExcpetion, que normalmente são por falha de conexão
            System.out.println("Falha de conexão com o banco de dados.\n" + ex.getMessage());
        }
    }

}
```

Analise e Conclusão

a) Qual a importância dos componentes de middleware, como o JDBC? **R:** Fazer a comunicação entre a aplicação e o servidor de banco de dados. Servindo como camada intermediaria de troca de dados de forma prática e de fácil uso.

b) Qual a diferença no uso de *Statement* ou *PreparedStatement* para a manipulação de dados? **R:** Statement excuta instruções que não tem para a serem definidos antes da execução da instrução. Já o PreparedStatement serve para adicionar paramêtros que são entendidos pelo SGBD, são normalmente usados para evitar ataques de SQL Injection.

c) Como o padrão DAO melhora a manutenibilidade do software? **R:** O padrão DAO encapsula as tarefas relacionadas a persistência de dados e com ajuda na divisão de responsabilidades dentro do projeto e facilitando a compreensão de cada funcionalidade já que estão dívidas por contexto.

d) Como a herança é refletida no banco de dados, quando lidamos com um modelo estritamente relacional? **R:** Exatamente como no projeto, mas no banco precisamos relacionar as tabelas por meio de FK's. E temos que ter o cuidado para manter a integridade do banco ao inserir dados respeitando a ordem de inserção dos registros com dependência.

2º Procedimento | Alimentando a Base

```
1   package cadastrobd.control;
2
3   import cadastro.model.PessoaFisicaDAO;
4   import cadastro.model.PessoaJuridicaDAO;
5   import cadastrobd.model.Pessoa;
6   import cadastrobd.model.PessoaFisica;
7   import cadastrobd.model.PessoaJuridica;
8   import java.sql.SQLException;
9   import java.util.ArrayList;
10  import java.util.InputMismatchException;
11  import java.util.Scanner;
12
13  public class CadastroBDTeste {
14
15      private static final Scanner entrada = new Scanner(System.in);
16
17      public static void main(String[] args) {
18          PessoaFisicaDAO repoFisica = null;
19          PessoaJuridicaDAO repoJuridica = null;
20
21          boolean conectado = false;
22
23          do {
24              try {
25                  repoFisica = new PessoaFisicaDAO();
26                  repoJuridica = new PessoaJuridicaDAO();
27                  conectado = true;
28              } catch (SQLException ex) {
29                  System.out.println("Erro de conexao: " + ex.getMessage());
30                  System.out.println("Pressione enter para tentar conectar novamente.");
31                  entrada.nextLine();
32              }
33          } while (!conectado);
34
```

```
34
35          boolean continuar = true;
36          String opcao;
37          do {
38              System.out.println("\n============================");
39              System.out.println(" --- MENU --- ");
40              System.out.println("1 - para incluir");
41              System.out.println("2 - para alterar");
42              System.out.println("3 - para excluir");
43              System.out.println("4 - buscar pessoa");
44              System.out.println("5 - exibir todos");
45              System.out.println("0 - para finalizar");
46              opcao = entrada.nextLine().trim();
47
48              OUTER:
49              switch (opcao) {
50                  case "1" -> {
51                      System.out.println("\nIncluindo...");
52                      System.out.println("F - para pessoa fisica. J - para pessoa juridica.");
53                      String tipo = entrada.nextLine().toUpperCase().trim();
54                      switch (tipo) {
55                          case "F" -> {
56                              PessoaFisica pf = new PessoaFisica();
57                              setarPessoa(pf);
58
59                              System.out.println("Digite o CPF: ");
60                              String valor = entrada.nextLine();
61                              pf.setCpf(valor.substring(0, Math.min(valor.length(), 11)));//Truncan
62                                                                                          //o valor
63
64                              try {
66                                  if (repoFisica.incluir(pf)) {
67                                      System.out.println("Cadastrado com sucesso!");
68                                  } else {
69                                      System.out.println("Nao foi possível realizar o cadastro!");
70                                  }
71                              } catch (SQLException ex) {
72                                  System.out.println("Falha de conexao.\n" + ex.getMessage());
73                                  ///testar o fluxo
74                                  break OUTER;
75                              }
```

```
76                  }
77                  case "J" -> {
78                      PessoaJuridica pj = new PessoaJuridica();
79                      setarPessoa(pj);
80
81                      System.out.println("Digite o CNPJ: ");
82                      String valor = entrada.nextLine();
83                      pj.setCnpj(valor.substring(0, Math.min(valor.length(), 14)));
84
85                      try {
87                          if (repoJuridica.incluir(pj)) {
88                              System.out.println("Cadastrado com sucesso!");
89                          } else {
90                              System.out.println("Nao foi possível realizar o cadastro!");
91                          }
92
93                      } catch (SQLException ex) {
94                          System.out.println("Falha de conexao.\n" + ex.getMessage());
95                          ///testar o fluxo
96                          break OUTER;
97                      }
98                  }
99                  default -> {
100                     System.out.println("Tipo invalido! Tente novamente.");
101                     break OUTER;
102                 }
103             }
104             case "2" -> {
105                 System.out.println("\nAlterando...");
106                 System.out.println("F - para pessoa fisica. J - para pessoa juridica.");
107                 String tipo = entrada.nextLine().toUpperCase().trim();
```

```
                switch (tipo) {
                    case "F" -> {
                        System.out.println("Digite o ID: ");
                        try {
                            int id = entrada.nextInt();
                            entrada.nextLine();
                            PessoaFisica pf = repoFisica.getPessoa(id);
                            if (pf == null) {
                                System.out.println("ID nao encontrado!");
                                break OUTER;
                            }
                            System.out.println(pf.exibir()+"\n");

                            setarPessoa(pf);

                            System.out.println("Digite o CPF: ");
                            String valor = entrada.nextLine();
                            pf.setCpf(valor.substring(0, Math.min(valor.length(), 11)));

                            if (repoFisica.alterar(pf)) {
                                System.out.println("Alterado com sucesso!");
                            } else {
                                System.out.println("Registro nao encontrado! Nao foi possivel alterar.");
                            }

                        } catch (InputMismatchException e) {
                            entrada.nextLine();
                            System.out.println("ID invalido, digite apenas numeros.");
                        } catch (SQLException e) {
                            System.out.println("Erro de conexão..." + e.getMessage());
                        }

                    }
```

```
142                     case "J" -> {
143                         System.out.println("Digite o ID: ");
144                         try {
145                             int id = entrada.nextInt();
146                             entrada.nextLine();
                            PessoaJuridica pj = repoJuridica.getPessoa(id);
148                             if (pj == null) {
149                                 System.out.println("ID nao encontrado!");
150                                 break OUTER;
151                             }
152                             System.out.println(pj.exibir()+"\n");
153 
154                             setarPessoa(pj);
155                             System.out.println("Digite o CNPJ: ");
156                             String valor = entrada.nextLine();
157                             pj.setCnpj(valor.substring(0, Math.min(valor.length(), 14)));
158 
159                             if (repoJuridica.alterar(pj)) {
160                                 System.out.println("Alterado com sucesso!");
161                             } else {
162                                 System.out.println("Registro nao encontrado! Não foi possivel alterar.");
163                             }
164 
165                         } catch (InputMismatchException e) {
166                             entrada.nextLine();
167                             System.out.println("ID invalido, digite apenas numeros.");
168                         } catch (SQLException e) {
169                             System.out.println("Erro de conexao..." + e.getMessage());
170                         }
171                     }
172                     default -> {
173                         System.out.println("Tipo invalido! Tente novamente.");
174                         break OUTER;
175                     }
176                 }
177             }
```

```
178                 case "3" -> {
179                     System.out.println("\nExcluindo...");
180                     System.out.println("F - para pessoa fisica. J - para pessoa juridica.");
181                     String tipo = entrada.nextLine().toUpperCase().trim();
182 
183                     switch (tipo) {
184                         case "F" -> {
185                             System.out.println("Digite o ID: ");
186                             try {
187                                 int id = entrada.nextInt();
188                                 entrada.nextLine();
189 
190                                 if (repoFisica.excluir(id)) {
191                                     System.out.println("Excluido com sucesso!");
192                                 } else {
193                                     System.out.println("Registro nao encontrado! Nao foi possivel excluir.");
194                                 }
195 
196                             } catch (InputMismatchException e) {
197                                 entrada.nextLine();
198                                 System.out.println("ID invalido, digite apenas numeros.");
199                             } catch (SQLException e) {
200                                 System.out.println("Erro de conexao..." + e.getMessage());
201                             }
202                         }
203                         case "J" -> {
204                             System.out.println("Digite o ID: ");
205                             try {
206                                 int id = entrada.nextInt();
207                                 entrada.nextLine();
208 
209                                 if (repoJuridica.excluir(id)) {
210                                     System.out.println("Excluido com sucesso!");
211                                 } else {
212                                     System.out.println("Registro nao encontrado! Nao foi possivel excluir.");
213                                 }
```

```
                        } catch (InputMismatchException e) {
                            entrada.nextLine();
                            System.out.println("ID invalido, digite apenas numeros.");
                        } catch (SQLException e) {
                            System.out.println("Erro de conexao..." + e.getMessage());
                        }
                    }
                    default -> {
                        System.out.println("Tipo invalido! Tente novamente.");
                        break OUTER;
                    }
                }
            }
            case "4" -> {
                System.out.println("\nBuscando...");
                System.out.println("F - para pessoa fisica. J - para pessoa juridica.");
                String tipo = entrada.nextLine().toUpperCase().trim();

                switch (tipo) {
                    case "F" -> {
                        System.out.println("Digite o ID: ");
                        try {
                            int id = entrada.nextInt();
                            entrada.nextLine();
                            PessoaFisica pf = repoFisica.getPessoa(id);
                            if (pf == null) {
                                System.out.println("ID nao encontrado!");
                                break OUTER;
                            }
                            System.out.println(pf.exibir());
                        } catch (InputMismatchException e) {
                            entrada.nextLine();
                            System.out.println("ID invalido, digite apenas números.");
                        } catch (SQLException e) {
                            System.out.println("Erro de conexao..." + e.getMessage());
                        }
                    }
```

```
251                     }
252                     case "J" -> {
253                         System.out.println("Digite o ID: ");
254                         try {
255                             int id = entrada.nextInt();
256                             entrada.nextLine();
                            PessoaJuridica pj = repoJuridica.getPessoa(id);
258                         if (pj == null) {
259                             System.out.println("ID nao encontrado!");
260                             break OUTER;
261                         }
262                         System.out.println(pj.exibir());
263                     } catch (InputMismatchException e) {
264                         entrada.nextLine();
265                         System.out.println("ID invalido, digite apenas numeros.");
266                     } catch (SQLException e) {
267                         System.out.println("Erro de conexao..." + e.getMessage());
268                     }
269                 }
270                 default -> {
271                     System.out.println("Tipo invalido! Tente novamente.");
272                     break OUTER;
273                 }
274             }
275         }
```

```
276                 case "5" -> {
277                     System.out.println("\nListando...");
278                     System.out.println("F - para pessoa fisica. J - para pessoa juridica.");
279                     String tipo = entrada.nextLine().toUpperCase().trim();
280 
281                     switch (tipo) {
282                         case "F" -> {
283                             try {
                                 ArrayList<PessoaFisica> pessoas = repoFisica.getPessoas();
285                                 System.out.println("============================");
286                                 System.out.println("Lista de pessoas fisicas\n");
287                                 for (PessoaFisica pf : pessoas) {
288                                     System.out.println(pf.exibir() + "\n");
289                                 }
290                             } catch (SQLException e) {
291                                 System.out.println("Erro de conexao..." + e.getMessage());
292                             }
293                         }
294                         case "J" -> {
295                             try {
                                 ArrayList<PessoaJuridica> pessoas = repoJuridica.getPessoas();
297                                 System.out.println("============================");
298                                 System.out.println("Lista de pessoas juridicas\n");
299                                 for (PessoaJuridica pj : pessoas) {
300                                     System.out.println(pj.exibir() + "\n");
301                                 }
302                             } catch (SQLException e) {
303                                 System.out.println("Erro de conexao..." + e.getMessage());
304                             }
305                         }
306                         default -> {
307                             System.out.println("Tipo invalido! Tente novamente.");
308                             break OUTER;
309                         }
310                     }
311                 }
```

```
312                 case "0" -> {
313                     System.out.println("\nFinalizando....");
314                     try {
                            repoFisica.fecharConexao();
                            repoJuridica.fecharConexao();
317                     } catch (SQLException e) {
318                         System.out.println("Erro de conexao..." + e.getMessage());
319                     } finally {
320                         continuar = false;
321                     }
322                 }
323                 default -> {
324                     System.out.println("Opcao invalida! Tente novamente");
325                     break;
326                 }
327             }
328         } while (continuar);
329     }
330
331     public static void setarPessoa(Pessoa p) {
332         String valor;
333
334         System.out.println("Digite o nome: ");
335         valor = entrada.nextLine();
336         p.setNome(valor.substring(0, Math.min(valor.length(), 150)));//Limita os ca
337         System.out.println("Digite o logradouro: ");
338         valor = entrada.nextLine();
339         p.setLogradouro(valor.substring(0, Math.min(valor.length(), 100)));
340         System.out.println("Digite a cidade: ");
341         valor = entrada.nextLine();
342         p.setCidade(valor.substring(0, Math.min(valor.length(), 100)));
343         System.out.println("Digite o estado: ");
344         valor = entrada.nextLine();
345         p.setEstado(valor.substring(0, Math.min(valor.length(), 2)));
346         System.out.println("Digite o telefone: ");
347         valor = entrada.nextLine();
348         p.setTelefone(valor.substring(0, Math.min(valor.length(), 11)));
349         System.out.println("Digite o e-mail: ");
350         valor = entrada.nextLine();
351         p.setEmail(valor.substring(0, Math.min(valor.length(), 150)));
352     }
```

Analise e Conclusão

a) Quais as diferenças entre a persistência em arquivo e a persistência em banco de dados? **R:** A persistência em um banco de dados é muito flexível pois permite a manipulação das informações com o uso da linguaguem SQL, a persistência em arquivo requer o uso da aplicação para qualquer tipo de manipulação que ainda assim é bastante limitada.

b) Como o uso de operador *lambda* simplificou a impressão dos valores contidos nas entidades, nas versões mais recentes do Java? **R:** Podemos definir funções simples em tempo real que nos economizam um tempo considerado e permitiu ir além dos usos de funções normais.

c) Por que métodos acionados diretamente pelo método main, sem o uso de um objeto, precisam ser marcados como *static*? **R:** Uma classe estática é uma classe que nunca vai ser instanciada e não teremos um objeto na HEAP para trabalhar com seus métodos, então por padrão todos seus métodos também devem ser *static*. Ou seja, toda classe não estática deve ser instanciada em um objeto para ser usada. Só temos acesso a seus métodos através de uma instancia ou que ela seja estática.

Conclusão

A missão serviu para introduzir a persistência de dados em um banco de dados relacional e configurar esse ambiente. Como usar os mecanismos disponíveis na linguagem java para persistir meus dados e a importância de delegar reponsabilidades únicas para as funcionalidades do meu sistema, facilitando todo o processo de codificar, manutenção e entendimento.