Missão Prática | Nível 1 | Mundo 3

Sumário

Missão Prática | Nível 1 | Mundo 3

1. **Dados do Curso:**

- **Campus:** Madureira/RJ
- **Nome do Curso:** Desenvolvimento Full Stack
- **Nome da Disciplina:** RPG0014 - Iniciando o caminho pelo Java
- **Número da Turma:** 9001
- **Semestre letivo:** 2024.3
- **Nome dos integrantes da prática:** Joseane Leal Braz

2. Título da Prática:

- **Implementação de um cadastro de clientes em modo texto, com persistência em arquivos, baseado na tecnologia Java.**

3. Objetivo da Prática

- Utilizar herança e polimorfismo na definição de entidades.
- Utilizar persistência de objetos em arquivos binários.
- Implementar uma interface cadastral em modo texto.
- Utilizar o controle de exceções da plataforma Java.
- No final do projeto, o aluno terá implementado um sistema cadastral em Java, utilizando os recursos da programação orientada a objetos e a persistência em arquivos binários.

4. Análise e Conclusão:

1. **Quais as vantagens e desvantagens do uso de herança?**

A vantagem está no reaproveitamento do código, que pode ser construído de forma mais organizada, e sem precisar repetir várias vezes, dados que vamos usar em outras classes.
A desvantagem acredito ser, o fato que uma mudança na classe pai afeta as filhas

2. **Por que a interface Serializable é necessária ao efetuar persistência em arquivos binários?**

Porque permite que o objeto seja transformado em uma sequência de bytes e armazenado em um arquivo binário, ou transmitido pela rede.

3. **Como o paradigma funcional é utilizado pela API stream no Java?**

Missão Prática | Nível 1 | Mundo 3

As operações podem ser passadas como argumentos, tratadas como funções de primeira classe. Isso permite processar dos dados de maneira eficiente e imutável.

4. **Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na persistência de dados em arquivos?**

DAO (Dat Access Object). Organiza a maneira como os dados são manipulados e acessados, permitindo uma separação clara entre a lógica de negócios e a lógica de persistência de dados.

5. **O que são elementos estáticos e qual o motivo para o método main adotar esse modificador?**
   Variáveis e métodos que pertencem à classe, em vez de pertencerem a instâncias específicas da classe. Isso significa que eles podem ser acessados sem a necessidade de criar uma instância (ou objeto) da classe.
   O motivo para a adoção deste modificador está relacionada principalmente à eficiência (pois não será preciso gerenciar a criação de uma instância de classe) ,

6. **Para que serve a classe Scanner?**

   Para interação do usuário com o sistema, imprimindo mensagem na tela que permite que o usuário insira dados no programa,

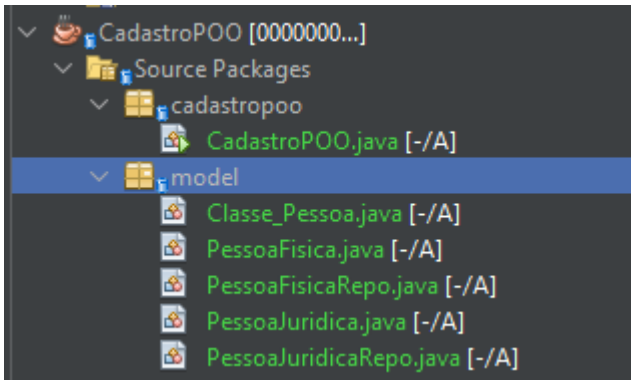
7. **Como o uso de classes de repositório impactou na organização do código?**

   Acredito que o código ficou mais organizado, mais fácil de testar, o fato de garantir consistência tb é muito importante.

Missão Prática | Nível 1 | Mundo 3

5. Códigos - Estrutura criada

A estrutura e classes foram criadas conforme as instruções do material



5.1 Códigos – Página Principal Cadastro POO

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package cadastropoo;



import java.io.IOException;
import java.util.Scanner;
import model.PessoaFisica;
import model.PessoaJuridica;
import model.PessoaFisicaRepo;
import model.PessoaJuridicaRepo;

public class CadastroPOO {
```

```
  private static final Scanner scanner = new Scanner(System.in);
  private static PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
  private static PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();

  public static void main(String[] args) {
    int opcao = -1;

    while (opcao != 0) {
      mostrarMenu();
      opcao = scanner.nextInt();
      scanner.nextLine(); // Limpar o buffer

      switch (opcao) {
        case 1:
          incluir();
          break;
        case 2:
          alterar();
          break;
        case 3:
          excluir();
          break;
        case 4:
          exibirPorId();
          break;
        case 5:
          exibirTodos();
          break;
        case 6:
          salvarDados();
          break;
        case 7:
          recuperarDados();
          break;
        case 0:
          System.out.println("Encerrando o programa.");
          break;
        default:
          System.out.println("Opção inválida.");
      }
    }
  }

  private static void mostrarMenu() {
    System.out.println("Selecione uma opção:");
    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 - Persistir dados");
    System.out.println("7 - Recuperar dados");
```

```
        System.out.println("0 - Finalizar Programa");
    }

    private static void incluir() {
        System.out.println("Incluir pessoa Física ou Jurídica (F/J)?");
        String tipo = scanner.nextLine().toUpperCase();

        if (tipo.equals("F")) {
            PessoaFisica pf = new PessoaFisica();
            System.out.println("Informe o ID:");
            pf.setId(scanner.nextInt());
            scanner.nextLine();
            System.out.println("Informe o nome:");
            pf.setNome(scanner.nextLine());
            System.out.println("Informe o CPF:");
            pf.setCpf(scanner.nextLine());
            System.out.println("Informe a idade:");
            pf.setIdade(scanner.nextInt());
            scanner.nextLine();
            repoFisica.inserir_pf(pf);
        } else if (tipo.equals("J")) {
            PessoaJuridica pj = new PessoaJuridica();
            System.out.println("Informe o ID:");
            pj.setId(scanner.nextInt());
            scanner.nextLine();
            System.out.println("Informe a razão social:");
            pj.setNome(scanner.nextLine());
            System.out.println("Informe o CNPJ:");
            pj.setCnpj(scanner.nextLine());
            repoJuridica.inserir_pj(pj);
        } else {
            System.out.println("Tipo inválido.");
        }
    }

    private static void alterar() {
        System.out.println("Alterar pessoa Física ou Jurídica (F/J)?");
        String tipo = scanner.nextLine().toUpperCase();

        if (tipo.equals("F")) {
            System.out.println("Informe o ID:");
            int id = scanner.nextInt();
            scanner.nextLine();
            PessoaFisica pf = repoFisica.obter_pf(id);
            if (pf != null) {
                System.out.println("Dados atuais: " + pf);
                System.out.println("Informe o novo nome:");
                pf.setNome(scanner.nextLine());
                System.out.println("Informe o novo CPF:");
                pf.setCpf(scanner.nextLine());
                System.out.println("Informe a nova idade:");
                pf.setIdade(scanner.nextInt());
```

```
            scanner.nextLine();
            repoFisica.alterar_pf(pf);
        } else {
            System.out.println("Pessoa Física não encontrada.");
        }
    } else if (tipo.equals("J")) {
        System.out.println("Informe o ID:");
        int id = scanner.nextInt();
        scanner.nextLine();
        PessoaJuridica pj = repoJuridica.obter_pj(id);
        if (pj != null) {
            System.out.println("Dados atuais: " + pj);
            System.out.println("Informe a nova razão social:");
            pj.setNome(scanner.nextLine());
            System.out.println("Informe o novo CNPJ:");
            pj.setCnpj(scanner.nextLine());
            repoJuridica.alterar_pj(pj);
        } else {
            System.out.println("Pessoa Jurídica não encontrada.");
        }
    } else {
        System.out.println("Tipo inválido.");
    }
}

private static void excluir() {
    System.out.println("Excluir pessoa Física ou Jurídica (F/J)?");
    String tipo = scanner.nextLine().toUpperCase();

    if (tipo.equals("F")) {
        System.out.println("Informe o ID:");
        int id = scanner.nextInt();
        scanner.nextLine();
        repoFisica.excluir_pf(id);
    } else if (tipo.equals("J")) {
        System.out.println("Informe o ID:");
        int id = scanner.nextInt();
        scanner.nextLine();
        repoJuridica.excluir_pj(id);
    } else {
        System.out.println("Tipo inválido.");
    }
}

private static void exibirPorId() {
    System.out.println("Exibir pessoa Física ou Jurídica (F/J)?");
    String tipo = scanner.nextLine().toUpperCase();

    if (tipo.equals("F")) {
        System.out.println("Informe o ID:");
        int id = scanner.nextInt();
        scanner.nextLine();
```

```
        PessoaFisica pf = repoFisica.obter_pf(id);
        //if (pf != null) {Id(id);
        if (pf != null) {
            System.out.println("Dados: " + pf);
        } else {
            System.out.println("Pessoa Física não encontrada.");
        }
    } else if (tipo.equals("J")) {
        System.out.println("Informe o ID:");
        int id = scanner.nextInt();
        scanner.nextLine();
        PessoaJuridica pj = repoJuridica.obter_pj(id);
        if (pj != null) {
            System.out.println("Dados: " + pj);
        } else {
            System.out.println("Pessoa Jurídica não encontrada.");
        }
    } else {
        System.out.println("Tipo inválido.");
    }
}

private static void exibirTodos() {
    System.out.println("Exibir todas as pessoas Físicas ou Jurídicas (F/J)?");
    String tipo = scanner.nextLine().toUpperCase();

    if (tipo.equals("F")) {
        for (PessoaFisica pf : repoFisica.obterTodos_pf()) {
            System.out.println(pf);
        }
    } else if (tipo.equals("J")) {
        for (PessoaJuridica pj : repoJuridica.obterTodos_pj()) {
            System.out.println(pj);
        }
    } else {
        System.out.println("Tipo inválido.");
    }
}

private static void salvarDados() {
    try {
        System.out.println("Informe o prefixo dos arquivos:");
        String prefixo = scanner.nextLine();
        repoFisica.persistir_pf(prefixo + "_fisica.dat");
        repoJuridica.persistir_pj(prefixo + "_juridica.dat");
        System.out.println("Dados salvos com sucesso.");
    } catch (IOException e) {
        System.err.println("Erro ao salvar os dados: " + e.getMessage());
    }
}

private static void recuperarDados() {
```

```
        try {
            System.out.println("Informe o prefixo dos arquivos:");
            String prefixo = scanner.nextLine();
            repoFisica.recuperar_pf(prefixo + "_fisica.dat");
            repoJuridica.recuperar_pj(prefixo + "_juridica.dat");
            System.out.println("Dados recuperados com sucesso.");
        } catch (IOException | ClassNotFoundException e) {
            System.err.println("Erro ao recuperar os dados: " + e.getMessage());
        }
    }
}
```

5.2 Códigos – Classe Pessoa

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package model;

import java.io.Serializable;
public class Classe_Pessoa implements Serializable {
    private int id;
    private String nome;

//Contrutor padrão
public Classe_Pessoa (){
}

//Construtor completo
public Classe_Pessoa(int id, String nome){
    this.id = id;
    this.nome = nome;
}

//Getters e Setters
public int getId(){
    return id;
}

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

public String getNome(){
     return nome;
    }

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

```
//Método para imprimir os dados
public void exibir(){
   System.out.println("ID: " + id);

   System.out.println("Nome: " + nome);

}

   }

/**

 *

 * @author Josy Leal

 */
```

5.3 Códigos – Classe Pessoa Física

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package model;

public class PessoaFisica extends Classe_Pessoa {
   private String cpf;
   private int idade;

//Contrutor padrão
public PessoaFisica(){
   super();
}

//Contrutor completo
public PessoaFisica (int id, String nome, String cpf, int idade){
   super(id, nome); //Chama o contrutor da classe Pessoa
   this.cpf = cpf;
   this.idade = idade;
}
 //Getters e Setters
public String getCpf(){
   return cpf;
}
public void setCpf(String cpf) {
  this.cpf = cpf;
}
public int getIdade(){
   return idade;
}
```

```
public void setIdade(int idade){
   this.idade = idade;
}

//Método para exibir polimórfico
public void exibir (){
   super.exibir(); //Chama o método exibir da classe Pessoa
   System.out.println("CPF: "+ cpf);
   System.out.println("Idade: "+idade);
}
 // Sobrescrevendo o método toString
   @Override
   public String toString() {
      return "Pessoa Fisica [ID=" + getId() + ", Nome=" + getNome() + ", CPF=" + cpf + ", Idade=" + idade + "]";
   }
}



/**
 *
 * @author Josy Leal
 */
```

5.4 Códigos – Classe Repositório de Pessoas Físicas

```
package model;

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */

/**
 *
 * @author Josy Leal
 */

import java.nio.charset.StandardCharsets;
import java.io.*;
import java.util.ArrayList;

public class PessoaFisicaRepo extends Classe_Pessoa{
   private ArrayList<PessoaFisica> pessoasFisicas;

   public PessoaFisicaRepo() {
      pessoasFisicas = new ArrayList<>();
   }

   // Método para inserir uma nova PessoaFisica
   public void inserir_pf (PessoaFisica pessoa) {
```

```
        pessoasFisicas.add(pessoa);
    }

    // Método para alterar uma PessoaFisica existente
    public void alterar_pf(PessoaFisica pessoa) {
        for (int i = 0; i < pessoasFisicas.size(); i++) {
            if (pessoasFisicas.get(i).getId() ==pessoa.getId()){
            pessoasFisicas.set(i, pessoa);
                return;
            }
        }
    }

    // Método para excluir uma PessoaFisica pelo ID
    public void excluir_pf (int id) {
        pessoasFisicas.removeIf(p -> p.getId() ==id);
    }

    // Método para obter uma PessoaFisica pelo ID
    public PessoaFisica obter_pf (int id) {
        for (PessoaFisica p : pessoasFisicas) {
            if (p.getId() == p.getId()) {
                return p;
            }
        }
        return null;
    }

    // Método para obter todas as PessoaFisica
    public ArrayList<PessoaFisica> obterTodos_pf() {
        return new ArrayList<>(pessoasFisicas);
    }

    // Método para persistir os dados em um arquivo
    public void persistir_pf (String nomeArquivo_pf) throws IOException {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(nomeArquivo_pf))){
            oos.writeObject(pessoasFisicas) ;
        }
    }

    // Método para recuperar os dados de um arquivo
    public void recuperar_pf(String nomeArquivo_pf) throws IOException, ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo_pf))) {
            pessoasFisicas = (ArrayList<PessoaFisica>) ois.readObject();
        }
    }
}
```

5.5 Códigos – Classe Pessoa Jurídica

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
```

```
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package model;

/**
 *
 * @author Josy Leal
 */
public class PessoaJuridica extends Classe_Pessoa {
   private String cnpj;

//Contrutor padrão
public PessoaJuridica(){
   super();
}
//Construtor completo
public PessoaJuridica(int id,  String nome, String cnpj){
   super(id,nome); //Chama o contrutotr da classe Pessoa
   this.cnpj = cnpj;
}
//Getters e Setters
public String getCnpj(){
   return cnpj;
}
public void setCnpj (String cnpj){
   this.cnpj = cnpj;
}
//Método para exibir polimórfico
public void exibir(){
   super.exibir();//Chama o método exibir da classe Pessoa
   System.out.println("CNPJ: "+cnpj);
}
// Sobrescrevendo o método toString
   @Override
   public String toString() {
      return "Pessoa Juricica [ID=" + getId() + ", Nome=" + getNome() + ", CNPJ=" + cnpj+"]";
   }
}
```

5.6 Códigos – Classe Repositório de Pessoas Jurídicas

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package model;

/**
 *
 * @author Josy Leal
 */
import java.io.*;
```

Missão Prática | Nível 1 | Mundo 3

```
import java.util.ArrayList;

public class PessoaJuridicaRepo extends Classe_Pessoa {
   private ArrayList<PessoaJuridica> pessoasJuridicas;

   public PessoaJuridicaRepo() {
      pessoasJuridicas = new ArrayList<>();
   }

   // Método para inserir uma nova PessoaJuridica
   public void inserir_pj(PessoaJuridica pessoa) {
      pessoasJuridicas.add(pessoa);
   }

   // Método para alterar uma PessoaJuridica existente
   public void alterar_pj(PessoaJuridica pessoa) {
      for (int i = 0; i < pessoasJuridicas.size(); i++) {
         if (pessoasJuridicas.get(i).getId() == pessoa.getId()) {
            pessoasJuridicas.set(i, pessoa);
            return;
         }
      }
   }

  // Método para excluir uma PessoaJuridica pelo ID
public void excluir_pj(int id) {
   pessoasJuridicas.removeIf(p -> p.getId()== id);
}

   // Método para obter uma PessoaJuridica pelo ID
   public PessoaJuridica obter_pj(int id) {
      for (PessoaJuridica p : pessoasJuridicas) {
         if (p.getId() == id) {
            return p;
         }
      }
      return null;
   }

   // Método para obter todas as PessoaJuridica
   public ArrayList<PessoaJuridica> obterTodos_pj() {
      return new ArrayList<>(pessoasJuridicas);
   }

   // Método para persistir os dados em um arquivo
   public void persistir_pj(String nomeArquivo_pj) throws IOException {
      try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(nomeArquivo_pj))) {
         oos.writeObject(pessoasJuridicas);
      }
   }

   // Método para recuperar os dados de um arquivo
```

```
    public void recuperar_pj(String nomeArquivo_pj) throws IOException, ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo_pj))) {
            pessoasJuridicas = (ArrayList<PessoaJuridica>) ois.readObject();
        }
    }
}
```