Gerenciamento de Pessoas com Persistência de Dados em Java

Curso:

Desenvolvimento Full Stack

Campus:

Polo São Cristóvão Porto Velho

Nome da Disciplina:

Iniciando o Caminho Pelo Java

Nome dos integrantes da Prática:

Ismael Lucena de Albuquerque

Objetivo da Prática:

O objetivo da Prática Implementar um sistema de gerenciamento de pessoas físicas e jurídicas em Java utilizando os conceitos de herança, polimorfismo e interfaces. Persistir os dados em arquivos binários com a interface Serializable e explorar a API Stream para manipulação de coleções.

Códigos Solicitados:

CadastroPOO:

```
/*
 * 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.util.Scanner;
import model.*;

public class CadastroPOO {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
        PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();

        while (true) {
            System.out.println("\n================================");
            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");
System.out.println("==============================");

int opcao = scanner.nextInt();
scanner.nextLine(); // Consumir quebra de linha

switch (opcao) {
    case 1: // Incluir Pessoa
        System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
        String tipoIncluir = scanner.nextLine().toUpperCase();

        if (tipoIncluir.equals("F")) {
            System.out.print("Digite o id da pessoa: ");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consumir quebra de linha
            System.out.print("Nome: ");
            String nome = scanner.nextLine();
            System.out.print("CPF: ");
            String cpf = scanner.nextLine();
            System.out.print("Idade: ");
            int idade = scanner.nextInt();
            scanner.nextLine();

            repoFisica.inserir(new PessoaFisica(id, nome, cpf, idade));
        } else if (tipoIncluir.equals("J")) {
            System.out.print("Digite o id da pessoa: ");
            int id = scanner.nextInt();
            scanner.nextLine();
            System.out.print("Nome: ");
            String nome = scanner.nextLine();
            System.out.print("CNPJ: ");
            String cnpj = scanner.nextLine();

            repoJuridica.inserir(new PessoaJuridica(id, nome, cnpj));
        }
        break;

    case 2: // Alterar Pessoa
        System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
        String tipoAlterar = scanner.nextLine().toUpperCase();

        if (tipoAlterar.equals("F")) {
```

```
            System.out.print("Digite o id da pessoa: ");
            int id = scanner.nextInt();
            scanner.nextLine();

            PessoaFisica pf = repoFisica.obter(id);
            if (pf != null) {
                System.out.println("Dados atuais: ");
                pf.exibir();

                System.out.print("Novo nome: ");
                String nome = scanner.nextLine();
                System.out.print("Novo CPF: ");
                String cpf = scanner.nextLine();
                System.out.print("Nova idade: ");
                int idade = scanner.nextInt();
                scanner.nextLine();

                repoFisica.alterar(id,new PessoaFisica(id, nome, cpf, idade));
            } else {
                System.out.println("Pessoa não encontrada.");
            }
        } else if (tipoAlterar.equals("J")) {
            System.out.print("Digite o id da pessoa: ");
            int id = scanner.nextInt();
            scanner.nextLine();

            PessoaJuridica pj = repoJuridica.obter(id);
            if (pj != null) {
                System.out.println("Dados atuais: ");
                pj.exibir();

                System.out.print("Novo nome: ");
                String nome = scanner.nextLine();
                System.out.print("Novo CNPJ: ");
                String cnpj = scanner.nextLine();

                repoJuridica.alterar(new PessoaJuridica(id, nome, cnpj));
            } else {
                System.out.println("Pessoa não encontrada.");
            }
        }
        break;

    case 3: // Excluir Pessoa
        System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
        String tipoExcluir = scanner.nextLine().toUpperCase();
```

```
            System.out.print("Digite o id da pessoa: ");
            int idExcluir = scanner.nextInt();
            scanner.nextLine();

            if (tipoExcluir.equals("F")) {
                repoFisica.excluir(idExcluir);
            } else if (tipoExcluir.equals("J")) {
                repoJuridica.excluir(idExcluir);
            }
            break;

        case 4: // Buscar pelo Id
            System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
            String tipoBuscar = scanner.nextLine().toUpperCase();

            System.out.print("Digite o id da pessoa: ");
            int idBuscar = scanner.nextInt();
            scanner.nextLine();

            if (tipoBuscar.equals("F")) {
                PessoaFisica pf = repoFisica.obter(idBuscar);
                if (pf != null) {
                    pf.exibir();
                } else {
                    System.out.println("Pessoa não encontrada.");
                }
            } else if (tipoBuscar.equals("J")) {
                PessoaJuridica pj = repoJuridica.obter(idBuscar);
                if (pj != null) {
                    pj.exibir();
                } else {
                    System.out.println("Pessoa não encontrada.");
                }
            }
            break;

        case 5: // Exibir Todos
            System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
            String tipoExibir = scanner.nextLine().toUpperCase();

            if (tipoExibir.equals("F")) {
                for (PessoaFisica pf : repoFisica.obterTodos()) {
                    pf.exibir();
                    System.out.println();
                }
```

```
                } else if (tipoExibir.equals("J")) {
                    for (PessoaJuridica pj : repoJuridica.obterTodos()) {
                        pj.exibir();
                        System.out.println();
                    }
                }
                break;

            case 6: // Persistir Dados
                System.out.print("Digite o prefixo do arquivo: ");
                String prefixoSalvar = scanner.nextLine();

                try {
                    repoFisica.persistir(prefixoSalvar + ".fisica.bin");
                    repoJuridica.persistir(prefixoSalvar + ".juridica.bin");
                } catch (Exception e) {
                    System.out.println("Erro ao salvar os dados: " + e.getMessage());
                }
                break;

            case 7: // Recuperar Dados
                System.out.print("Digite o prefixo do arquivo: ");
                String prefixoRecuperar = scanner.nextLine();

                try {
                    repoFisica.recuperar(prefixoRecuperar + ".fisica.bin");
                    repoJuridica.recuperar(prefixoRecuperar + ".juridica.bin");
                } catch (Exception e) {
                    System.out.println("Erro ao recuperar os dados: " + e.getMessage());
                }
                break;

            case 0: // Finalizar Programa
                System.out.println("Encerrando o programa...");
                scanner.close();
                return;

            default:
                System.out.println("Opção inválida!");
                break;
        }
    }
  }
}
```

Pessoa:

```
package model;
```

```java
import java.io.Serializable;

public class Pessoa implements Serializable {
    private int id;
    private String nome;

    public Pessoa() {}

    public Pessoa(int id, String nome) {
        this.id = id;
        this.nome = nome;
    }

    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;
    }

    public void exibir() {
        System.out.println("Id: " + id);
        System.out.println("Nome: " + nome);
    }
```

```
}
```

PessoaFisica:

```
package model;

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

    public PessoaFisica() {}

    public PessoaFisica(int id, String nome, String cpf, int idade) {
        super(id, nome);
        this.cpf = cpf;
        this.idade = idade;
    }

    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;
    }

    @Override
    public void exibir() {
        super.exibir();
```

```
        System.out.println("CPF: " + cpf);
        System.out.println("Idade: " + idade);
    }
}
```

PessoaFisica:

```
package model;

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

    public PessoaFisica() {}

    public PessoaFisica(int id, String nome, String cpf, int idade) {
        super(id, nome);
        this.cpf = cpf;
        this.idade = idade;
    }

    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;
    }

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CPF: " + cpf);
        System.out.println("Idade: " + idade);
```

```
    }
}
```

PessoaFisicaRepo:

```
package model;

import java.io.*;
import java.util.ArrayList;

public class PessoaFisicaRepo {
    private ArrayList<PessoaFisica> lista = new ArrayList<>();

    public void inserir(PessoaFisica pessoa) {
        lista.add(pessoa);
    }

    public void alterar(int id, PessoaFisica novaPessoa) {
        for (int i = 0; i < lista.size(); i++) {
            if (lista.get(i).getId() == id) {
                lista.set(i, novaPessoa);
                break;
            }
        }
    }

    public void excluir(int id) {
        lista.removeIf(pessoa -> pessoa.getId() == id);
    }

    public PessoaFisica obter(int id) {
                        return lista.stream().filter(pessoa -> pessoa.getId() ==
id).findFirst().orElse(null);
    }

    public ArrayList<PessoaFisica> obterTodos() {
        return lista;
    }
```

```
    public void persistir(String arquivo) throws IOException {
            try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(arquivo))) {
        oos.writeObject(lista);
      }
    }

          public void recuperar(String arquivo) throws IOException,
ClassNotFoundException {
              try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(arquivo))) {
        lista = (ArrayList<PessoaFisica>) ois.readObject();
      }
    }
}
```

PessoaJuridica:

```
package model;

public class PessoaJuridica extends Pessoa {
    private String cnpj;

    public PessoaJuridica() {}

    public PessoaJuridica(int id, String nome, String cnpj) {
        super(id, nome);
        this.cnpj = cnpj;
    }

    public String getCnpj() {
        return cnpj;
    }

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

```
    }

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
    }
}
```

PessoaJuridicaRepo:

```
package model;

import java.io.*;
import java.util.ArrayList;

public class PessoaJuridicaRepo {
    // Lista privada de pessoas jurídicas
    private ArrayList<PessoaJuridica> lista = new ArrayList<>();

    // Método para inserir uma nova pessoa jurídica
    public void inserir(PessoaJuridica pj) {
        lista.add(pj);
    }

    // Método para alterar uma pessoa jurídica existente (baseado no ID)
    public void alterar(PessoaJuridica pj) {
        for (int i = 0; i < lista.size(); i++) {
            if (lista.get(i).getId() == pj.getId()) {
                lista.set(i, pj);
                return;
            }
        }
    }

    // Método para excluir uma pessoa jurídica pelo ID
    public void excluir(int id) {
        lista.removeIf(pj -> pj.getId() == id);
    }

    // Método para obter uma pessoa jurídica pelo ID
    public PessoaJuridica obter(int id) {
        for (PessoaJuridica pj : lista) {
            if (pj.getId() == id) {
                return pj;
```

```
            }
        }
        return null;
    }

    // Método para obter todas as pessoas jurídicas
    public ArrayList<PessoaJuridica> obterTodos() {
        return new ArrayList<>(lista);
    }

    // Método para persistir os dados no disco
    public void persistir(String arquivo) throws IOException {
                try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(arquivo))) {
            oos.writeObject(lista);
        }
    }

    // Método para recuperar os dados do disco
                public void recuperar(String arquivo) throws IOException,
ClassNotFoundException {
                    try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(arquivo))) {
            lista = (ArrayList<PessoaJuridica>) ois.readObject();
        }
    }
}
```

Resultados da Execução

Pessoas Físicas:

Id: 1
Nome: Ana
CPF: 11111111111
Idade: 25

Id: 2

Nome: Carlos
CPF: 22222222222
Idade: 52

Pessoas Jurídicas:

Id: 3
Nome: XPTO Sales
CNPJ: 33333333333333

Id: 4
Nome: XPTO Solutions
CNPJ: 44444444444444

Vantagens e Desvantagens do Uso de Herança

Vantagens:

- Reutilização de código e redução de redundância.
- Organização clara com estrutura hierárquica.

Desvantagens:

- Acoplamento excessivo pode dificultar manutenção.
- Nem sempre as relações entre classes são hierárquicas.

Por que a Interface Serializable é Necessária?

Ela permite a conversão de objetos em fluxos de bytes para armazenagem ou transmissão. Sem Serializable, o Java lança exceções quando tentamos serializar objetos.

Como o Paradigma Funcional é Utilizado pela API Stream?

A API Stream utiliza operações funcionais como filter, map, e forEach para manipular coleções de forma declarativa

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

Quando trabalhamos com Java na persistência de dados em arquivos, o padrão de desenvolvimento adotado é DAO (Data Access Object), juntamente com a técnica de Serialização de Objetos.

O que são elementos estáticos e qual o motivo para o método main adotar esse modificador?

Elementos estáticos em Java pertencem à classe e não a instâncias dela. Isso significa que eles podem ser acessados diretamente usando o nome da classe, sem a necessidade de criar um objeto. Métodos ou variáveis marcados com o modificador static são úteis quando o comportamento ou dado não depende de estados específicos das instâncias.

O método main é estático porque ele é o ponto de entrada do programa e precisa ser executado antes que qualquer instância da classe seja criada. Como a JVM (Java Virtual Machine) chama diretamente o método main para iniciar a execução do programa, ele precisa ser static.

Para que serve a classe Scanner?

A classe Scanner em Java é utilizada para ler dados de entrada, como texto ou números, a partir de diferentes fontes, como o teclado (System.in), arquivos, ou strings. No contexto deste programa, ela permite ao usuário interagir com o sistema digitando comandos ou informações necessárias para operações como inclusão ou alteração de dados.

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

O uso de classes de repositório, como PessoaFisicaRepo e PessoaJuridicaRepo, melhorou a modularidade e a organização do código, separando as responsabilidades relacionadas à manipulação de dados das operações da interface do programa.