Campus: POLO VILA DOS REMÉDIOS – OSASCO – SP

Curso: Desenvolvimento Full Stack

Nome Disciplina: Nível 1: Iniciando o caminho pelo Java

Matrícula: 2023 0397 9797

Semestre Letivo: 3º semestre

Nome: Anderson Barbosa Almeida

Repositório no GIT: https://github.com/andydevbarbosa/RPG0014---Iniciando-o-caminho-pelo-Java/blob/main/CadastroPOO.java

Título da Prática: Criação das Entidades e Sistema de Persistência

Objetivo da Prática: O objetivo desta prática foi implementar um sistema de gerenciamento de entidades em Java, utilizando herança, persistência em arquivos e conceitos básicos de orientação a objetos.

Códigos:

Aqui estão os códigos desenvolvidos durante a prática:

```
package cadastropoo;

import java.io.*;

import java.util.ArrayList;


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 + ", Nome: " + nome);
    }
}

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 + ", Idade: " + idade);
    }
}
```

```
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);
    }
}

class PessoaFisicaRepo {
    private ArrayList<PessoaFisica> pessoasFisicas = new ArrayList<>();
```

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

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

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

public PessoaFisica obter(int id) {
    for (PessoaFisica pessoa : pessoasFisicas) {
        if (pessoa.getId() == id) {
            return pessoa;
        }
    }
    return null;
}

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

```
    }

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

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

class PessoaJuridicaRepo {
    private ArrayList<PessoaJuridica> pessoasJuridicas = new ArrayList<>();

    public void inserir(PessoaJuridica pessoa) {
        pessoasJuridicas.add(pessoa);
    }

    public void alterar(PessoaJuridica pessoa) {
        for (int i = 0; i < pessoasJuridicas.size(); i++) {
            if (pessoasJuridicas.get(i).getId() == pessoa.getId()) {
                pessoasJuridicas.set(i, pessoa);
                break;
```

```
        }
    }
}

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

public PessoaJuridica obter(int id) {
    for (PessoaJuridica pessoa : pessoasJuridicas) {
        if (pessoa.getId() == id) {
            return pessoa;
        }
    }
    return null;
}

public ArrayList<PessoaJuridica> obterTodos() {
    return pessoasJuridicas;
}

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

public void recuperar(String arquivo) throws IOException,
ClassNotFoundException {
```

```
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(arquivo))) {
            pessoasJuridicas = (ArrayList<PessoaJuridica>) ois.readObject();
        }
    }
}

public class CadastroPOO {
    public static void main(String[] args) {
        try {
            PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
            repo1.inserir(new PessoaFisica(1, "Joao", "123.456.789-00", 30));
            repo1.inserir(new PessoaFisica(2, "Maria", "987.654.321-00", 25));
            repo1.persistir("pessoas_fisicas.txt");

            PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
            repo2.recuperar("pessoas_fisicas.txt");
            System.out.println("Pessoas Fisicas:");
            for (PessoaFisica pessoa : repo2.obterTodos()) {
                pessoa.exibir();
            }

            PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
            repo3.inserir(new PessoaJuridica(1, "Joao S.A.", "12.345.678/0001-
50"));
            repo3.inserir(new PessoaJuridica(2, "Maria S.A.", "97.654.321/0001-
95"));
            repo3.persistir("pessoas_juridicas.txt");

            PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
```

```
            repo4.recuperar("pessoas_juridicas.txt");
            System.out.println("\nPessoas Juridicas:");
            for (PessoaJuridica pessoa : repo4.obterTodos()) {
                pessoa.exibir();
            }
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

Análise e Conclusão:

- Vantagens e Desvantagens do Uso de Herança:

A herança em Java permite a reutilização de código e a criação de uma hierarquia de classes, o que pode tornar o código mais organizado e facilitar a manutenção. No entanto, o uso excessivo de herança pode levar a um acoplamento forte entre as classes e dificultar a compreensão do código.

- Necessidade da Interface Serializable na Persistência em Arquivos Binários:

A interface Serializable é necessária ao efetuar persistência em arquivos binários porque permite que os objetos sejam convertidos em uma sequência de bytes que podem ser armazenados ou transmitidos e posteriormente reconstruídos. Isso é importante para salvar objetos em arquivos de forma que possam ser recuperados posteriormente.

- **Utilização do Paradigma Funcional pela API Stream no Java:**

A API Stream no Java utiliza o paradigma funcional para operar em sequências de elementos de forma concisa e eficiente. Isso permite realizar operações como filtragem, mapeamento e redução de forma mais expressiva e idiomática.

- Padrão de Desenvolvimento na Persistência de Dados em Arquivos em Java:

No Java, o padrão de desenvolvimento comumente adotado na persistência de dados em arquivos é usar as classes ObjectOutputStream e ObjectInputStream para serializar e desserializar objetos, respectivamente. Isso permite salvar e recuperar objetos em arquivos de forma eficiente e confiável.