Missão prática | Nível 1 | Mundo 3

CAMPUS: POLO DISTRITO JK - ANÁPOLIS - GO

CURSO: DESENVOLVIMENTO FULL STACK

NÚMERO DA TURMA: 2024.3

SEMESTRE LETIVO: Segundo semestre de 2024

ALIUNO: Henrique Rodrigues Rabello Vieira

MATRÍCULA: 202301230527

RPG0014 – Iniciando o caminho pelo Java

Objetivos:

- 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.

Primeiro procedimento:

Códigos solicitados:

Criação da classe Pessoa:

```java
package model;

import java.io.Serializable;

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

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

    public int getId(){
        return this.id;
    }
    public String getNome(){
        return this.nome;
    }
    public void setId(int id){
        this.id = id;
    }
    public void setNome(String nome){
        this.nome = nome;
    }
    public void exibir(){
        System.out.printf("ID: %d \nNome: %s\n", getId(), getNome());
    }
}
```

Criação da classe PessoaFisica:

```java
package model;

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

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

    public String getCpf(){
        return this.cpf;
    }
    public int getIdade(){
        return this.idade;
    }
    public void setCpf(String cpf){
        this.cpf = cpf;
    }
    public void setIdade(int idade){
        this.idade = idade;
    }

    @Override
    public void exibir(){
        System.out.printf("ID: %d \nNome: %s\nCPF: %s\nIdade: %d\n", getId(), getNome(), getCpf(), getIdade());
    }
}
```

Criação da classe PessoaJuridica:

```java
package model;

public class PessoaJuridica extends Pessoa {
    private String cnpj;

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

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

    @Override
    public void exibir(){
        System.out.printf("ID: %d \nNome: %s\nCNPJ: %s\n", getId(), getNome(), getCnpj());
    }
}
```

Criação da classe PessoaFisicaRepo:

```java
public class PessoaFisicaRepo {
    private ArrayList<PessoaFisica> listaPessoaFisica = new ArrayList<>();

    public ArrayList<PessoaFisica> obterTodos(){
        return this.listaPessoaFisica;
    }
    public void inserir(PessoaFisica pessoa){
        obterTodos().add(pessoa);
    }
    public void alterar(PessoaFisica pessoa, PessoaFisica alteracao){
        int index = obterTodos().indexOf(pessoa);
        obterTodos().set(index, alteracao);
    }
    public void excluir(int Id){
        for(PessoaFisica pessoa: obterTodos()){
            if(pessoa.getId() == Id){
                obterTodos().remove(pessoa);
            }
        }
    }
    public PessoaFisica obter(int Id){
        for(PessoaFisica pessoa: obterTodos()){
            if(pessoa.getId() == Id){
                return pessoa;
            }
        }
        return null;
    }
    public void persistir(String nome_arquivo)throws FileNotFoundException, IOException{
            FileOutputStream arquivo = new FileOutputStream(nome_arquivo + ".ser");
            ObjectOutputStream output = new ObjectOutputStream(arquivo);
            output.writeObject(obterTodos());
            output.close();
            arquivo.close();
            System.out.println("Dados de Pessoa Fisica Armazenados");
    }
    public void recuperar(String nome_arquivo) throws FileNotFoundException, IOException, ClassNotFoundException{
        ArrayList<PessoaFisica> lista = null;
        String nome_arquivo_completo = nome_arquivo + ".ser";
        FileInputStream arquivo = new FileInputStream(nome_arquivo_completo);
        ObjectInputStream input = new ObjectInputStream(arquivo);
        lista = (ArrayList<PessoaFisica>) input.readObject();
        input.close();
        arquivo.close();
        System.out.println("Dados de Pessoa Fisica Recuperados");
        this.listaPessoaFisica = lista;
    }
}
```

Criação da classe PessoaJuridicaRepo:

```java
public class PessoaJuridicaRepo {
    private ArrayList<PessoaJuridica> listaPessoaJuridica = new ArrayList<>();

    public ArrayList<PessoaJuridica> obterTodos(){
        return this.listaPessoaJuridica;
    }
    public void inserir(PessoaJuridica pessoa){
        obterTodos().add(pessoa);
    }
    public void alterar(PessoaJuridica pessoa, PessoaJuridica alteracao){
        int index = obterTodos().indexOf(pessoa);
        obterTodos().set(index, alteracao);
    }
    public void excluir(int Id){
        for(PessoaJuridica pessoa: obterTodos()){
            if(pessoa.getId() == Id){
                obterTodos().remove(pessoa);
            }
        }
    }
    public PessoaJuridica obter(int Id){
        for(PessoaJuridica pessoa: obterTodos()){
            if(pessoa.getId() == Id){
                return pessoa;
            }
        }
        return null;
    }
    public void persistir(String nome_arquivo)throws FileNotFoundException, IOException{
            FileOutputStream arquivo = new FileOutputStream(nome_arquivo + ".ser");
            ObjectOutputStream output = new ObjectOutputStream(arquivo);
            output.writeObject(obterTodos());
            output.close();
            arquivo.close();
            System.out.println("Dados de Pessoa Jurídica Armazenados");
    }
    public void recuperar(String nome_arquivo) throws FileNotFoundException, IOException, ClassNotFoundException{
        ArrayList<PessoaJuridica> lista = null;
        String nome_arquivo_completo = nome_arquivo + ".ser";
        FileInputStream arquivo = new FileInputStream(nome_arquivo_completo);
        ObjectInputStream input = new ObjectInputStream(arquivo);
        lista = (ArrayList<PessoaJuridica>) input.readObject();
        input.close();
        arquivo.close();
        System.out.println("Dados de Pessoa Jurídica Recuperados");
        this.listaPessoaJuridica = lista;
    }
}
```

Alteração do método Main:

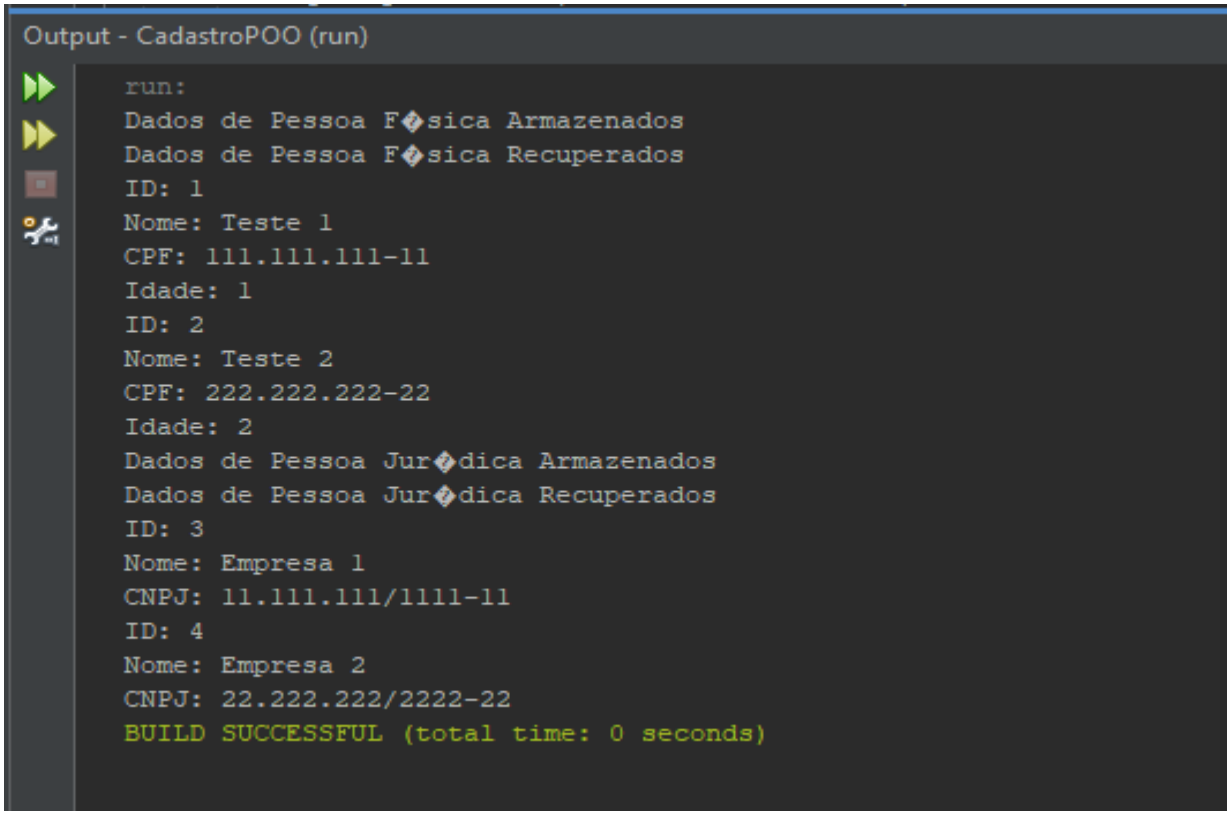
```java
package cadastropoo;

import java.io.*;
import model.*;

public class CadastroPOO {

    public static void main(String[] args) throws IOException, FileNotFoundException, ClassNotFoundException {
        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
        PessoaFisica fisica1 = new PessoaFisica(001, "Teste 1", "111.111.111-11", 01);
        repo1.inserir(fisica1);
        PessoaFisica fisica2 = new PessoaFisica(002, "Teste 2", "222.222.222-22", 02);
        repo1.inserir(fisica2);
        repo1.persistir("PessoaFisica01");
        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
        repo2.recuperar("PessoaFisica01");
        for (PessoaFisica pessoa: repo2.obterTodos()){
            pessoa.exibir();
        }
        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
        PessoaJuridica juridica1 = new PessoaJuridica(003, "Empresa 1", "11.111.111/1111-11");
        repo3.inserir(juridica1);
        PessoaJuridica juridica2 = new PessoaJuridica(004, "Empresa 2", "22.222.222/2222-22");
        repo3.inserir(juridica2);
        repo3.persistir("PessoaJuridica01");
        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
        repo4.recuperar("PessoaJuridica01");
        for (PessoaJuridica pessoa: repo4.obterTodos()){
            pessoa.exibir();
        }
    }
}
```

Resultado da execução do código:



Análise e conclusão:

Quais as vantagens e desvantagens do uso de herança?

Vantagens: Reutilização de código já escrito, estender o comportamento de uma classe já existente, adicionando métodos e propriedades e o polimorfismo facilitando a flexibilidade do código.

Desvantagens: Muita herança pode gerar o efeito contrário, deixando o código mais difícil de entender e de fazer a manutenção, aumentando também a complexidade do código de forma desnecessária e podendo levar a futuros problemas, e no caso específico do Java, ele não suporta herança múltipla, limitando seu uso.

Porque a interface Serializable é necessária ao efetuar persistência em arquivos binários?

Serializable quer dizer que que a classe pode ser transformada em bytes e remontada a partir desses bytes, possibilitando a persistência de objetos, porque os objetos podem ser salvos em arquivos binários, e por isso também facilitando o envio desses objetos pela rede, uma vez que podem ser recriados em outra máquina por serem arquivos binários.

Como o paradigma funcional é utilizado pela API stream do JAVA?

O paradigma adotado é o paradigma funcional para operações de processamento de dados. Utilizando funções lambda permitindo a definição de funções anônimas e contribuindo para um código mais legível. Operações de filtragem e mapeamento aplicam operações funcionalmente em elementos da coleção.

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

Padrão DAO (Data Acess Object). O padrão DAO abstrai e encapsula todos os acessos aos dados, permitindo uma separação de lógica de acesso aos dados e regras de negócio.