CadastroPOOParte1

CAMPUS DE ITABUNA

Curso: **Desenvolvedor Full Stack**

Disciplina: **Iniciando o Caminho Pelo Java Parte 1**

Turma: **2023.1**

Semestre Letivo: **3º**

Nome do Autor: **Pedro Carvalho Gama**

Link Repositório GitHub: https://github.com/PedroCGM/CadastroPooParte1

Objetivos:

1. Utilizar herança e polimorfismo na definição de entidades;

2. Utilizar persistência de objetos em arquivos binários;

3. Implementar uma interface cadastral em modo texto;

4. Utilizar o controle de exceções da plataforma Java;

5. No final do projeto terá implementado um sistema cadastral em Java, utilizando os recursos da programação orientada a objetos e a persistência em arquivos binários.

ANÁLISE E CONCLUSÃO

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

Vantagens: Reutilização de Código, Organização e hierarquia e o Polimorfismo.

Desvantagens: Herança Frágil, Herança Múltipla e o Acoplamento.

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

Ela é necessária pois permite que objetos Java sejam serializados, (convertidos em uma sequência de byte que representam o estado do objeto).

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

Ela oferece uma variedade de operações de alto nível, como collect, reduce, filter, map etc. elas permitem manipular e transformar elementos de um stream de modo declarativo.

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

Encapsulamento das operações de E/S, Flexibilidade na Implementação, Abstração da camada de dados, Separação de responsabilidades.

CÓDIGOS CADASTROPOO PARTE1

------CADASTROPOOPARTE1------

-----Main-----

```
package cadastropoo;

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

public class CadastroPOOParte1 {

    public static void main(String[] args) {
    // Criar um repositório de pessoas físicas (repo1)
    PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

    // Adicionar duas pessoas físicas usando o construtor completo
    repo1.inserir(new PessoaFisica(1, "Pedro", "55555555555", 30));
    repo1.inserir(new PessoaFisica(2, "Maria", "66666666666", 25));

    // Mensagem de confirmação
    System.out.println("Dados de Pessoa Fisica Armazenados.");

    // Invocar o método de persistência em repo1, fornecendo um nome de
arquivo fixo ("pessoaFisica.dat")
    try {
        repo1.persistir("pessoaFisica.dat");
    } catch (IOException e) {
        System.err.println("Erro ao persistir dados de pessoas físicas: " +
e.getMessage());
    }

    // Instanciar outro repositório de pessoas físicas (repo2)
    PessoaFisicaRepo repo2 = new PessoaFisicaRepo();

    // Invocar o método de recuperação em repo2, fornecendo o mesmo nome
de arquivo utilizado anteriormente
    try {
        repo2.recuperar("pessoaFisica.dat");
    } catch (IOException | ClassNotFoundException e) {
        System.err.println("Erro ao recuperar dados de pessoas físicas: " +
e.getMessage());
    }
```

```
    // Mensagem de confirmação
    System.out.println("Dados de Pessoa Fisica Recuperados.");

    // Exibir os dados de todas as pessoas físicas recuperadas
    for (PessoaFisica pessoaFisica : repo2.obterTodos()) {
        pessoaFisica.exibir();
    }

    // Criar um repositório de pessoas jurídicas (repo3)
    PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();

    // Adicionar duas pessoas jurídicas usando o construtor completo
    repo3.inserir(new PessoaJuridica(3, "FIT Fit", "77777777777777"));
    repo3.inserir(new PessoaJuridica(4, "FIT Project", "88888888888888"));

    // Mensagem de confirmação
    System.out.println("Dados de Pessoa Juridica Armazenados.");

    // Invocar o método de persistência em repo3, fornecendo um nome de
arquivo fixo ("pessoaJuridica.dat")
    try {
        repo3.persistir("pessoaJuridica.dat");
    } catch (IOException e) {
        System.err.println("Erro ao persistir dados de pessoas jurídicas: " +
e.getMessage());
    }

    // Instanciar outro repositório de pessoas jurídicas (repo4)
    PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();

    // Invocar o método de recuperação em repo4, fornecendo o mesmo nome
de arquivo utilizado anteriormente
    try {
        repo4.recuperar("pessoaJuridica.dat");
    } catch (IOException | ClassNotFoundException e) {
        System.err.println("Erro ao recuperar dados de pessoas jurídicas: " +
e.getMessage());
    }

    // Mensagem de confirmação
    System.out.println("Dados de Pessoa Juridica Recuperados.");

    // Exibir os dados de todas as pessoas jurídicas recuperadas
    for (PessoaJuridica pessoaJuridica : repo4.obterTodos()) {
        pessoaJuridica.exibir();
    }
    }
```

```
}

------Pessoa.Java------

/*
 * 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 Pessoa implements Serializable {
    private int id;
    private String nome;

    // Construtor padrão
    public Pessoa() {}

    // Construtor com todos os campos
    public 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 exibir
    public void exibir() {
        System.out.println("ID: " + id + ", Nome: " + nome);
```

```
    }
}


------PessoaFisica.Java------


/*
 * 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 PessoaFisica extends Pessoa implements Serializable {
    private String cpf;
    private int idade;

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

    // Construtor com todos os campos
    public PessoaFisica(int id, String nome, String cpf, int idade) {
        super(id, nome);
        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 exibir polimórfico
    @Override
    public void exibir() {
        System.out.println("ID: " + getId());
        System.out.println("Nome: " + getNome());
        System.out.println("CPF: " + getCpf());
        System.out.println("Idade: " + getIdade());
    }

}


------PessoaFisicaRepo.Java------


/*
 * 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.*;
import java.util.ArrayList;
import java.util.List;

public class PessoaFisicaRepo {
    private List<PessoaFisica> pessoasFisicas;

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

    // Método para inserir uma pessoa física
    public void inserir(PessoaFisica pessoaFisica) {
        pessoasFisicas.add(pessoaFisica);
    }

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

```
        }
      }
   }

   // Método para excluir uma pessoa física
   public void excluir(int id) {
      pessoasFisicas.removeIf(p -> p.getId() == id);
   }

   // Método para obter uma pessoa física por id
   public PessoaFisica obter(int id) {
      for (PessoaFisica pessoaFisica : pessoasFisicas) {
         if (pessoaFisica.getId() == id) {
            return pessoaFisica;
         }
      }
      return null;
   }

   // Método para obter todas as pessoas físicas
   public List<PessoaFisica> obterTodos() {
      return new ArrayList<>(pessoasFisicas);
   }

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

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


------PessoaJuridica.Java------


/*
```

```
 * 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 PessoaJuridica extends Pessoa implements Serializable {
    private String cnpj;

    // Construtor padrão
    public PessoaJuridica() {
        super();
    }

    // Construtor com todos os campos
    public PessoaJuridica(int id, String nome, String cnpj) {
        super(id, nome);
        this.cnpj = cnpj;
    }

    // Getters e setters
    public String getCnpj() {
        return cnpj;
    }

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

    // Método exibir polimórfico
    @Override
    public void exibir() {
        System.out.println("ID: " + getId());
        System.out.println("Nome: " + getNome());
        System.out.println("CNPJ: " + getCnpj());
    }

}


-------PessoaJuridicaRepo.Java-------


/*
```

```java
 * 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.*;
import java.util.ArrayList;
import java.util.List;

public class PessoaJuridicaRepo {
    private List<PessoaJuridica> pessoasJuridicas;

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

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

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

    // Método para excluir uma pessoa jurídica
    public void excluir(int id) {
        pessoasJuridicas.removeIf(p -> p.getId() == id);
    }

    // Método para obter uma pessoa jurídica por id
    public PessoaJuridica obter(int id) {
        for (PessoaJuridica pessoaJuridica : pessoasJuridicas) {
            if (pessoaJuridica.getId() == id) {
                return pessoaJuridica;
            }
        }
        return null;
    }
```

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

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

    // Método para recuperar os dados de um arquivo
    public void recuperar(String nomeArquivo) throws IOException,
ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
            pessoasJuridicas = (List<PessoaJuridica>) ois.readObject();
        }
    }
}
```

RESULTADO:
run:
Dados de Pessoa Fisica Armazenados.
Dados de Pessoa Fisica Recuperados.
ID: 1
Nome: Pedro
CPF: 55555555555
Idade: 30
ID: 2
Nome: Maria
CPF: 66666666666
Idade: 25
Dados de Pessoa Juridica Armazenados.
Dados de Pessoa Juridica Recuperados.
ID: 3
Nome: FIT Fit
CNPJ: 77777777777777
ID: 4
Nome: FIT Project
CNPJ: 88888888888888
BUILD SUCCESSFUL (total time: 0 seconds)

Print do Resultado: