RELATÓRIO CADASTRO POO

UNIVERCIDADE: ESTÁCIO DE SÁ

CAMPUS: POLO CID UNUVERCITARIA - JUAZEIRO DO NORTE-CE

CURSO: DESENVOLVIMNTE FULL STACK

DISCIPLINA: RPG0014 - INICIANDO OS CAMINHOS PELO JAVA

TURMA: 9001 SEMESTRE LETIVO: 2024.4

NOME DO INTEGRANTE: EDNALDO RIBEIRO DE OLIVEIRA

CLASSE PESSOA

```
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.java

```
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);
    }
}
```

PessoaJuridica.java

```
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);
    }
}
```

PessoaFisicaRepo.java

```
package model;

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

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

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

    public void alterar(PessoaFisica pessoa) {
        PessoaFisica existente = obter(pessoa.getId());
        if (existente != null) {
            existente.setNome(pessoa.getNome());
            existente.setCpf(pessoa.getCpf());
            existente.setIdade(pessoa.getIdade());
        }
    }

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

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

```
    public List<PessoaFisica> obterTodos() {
        return pessoas;
    }

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

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

PessoaJuridicaRepo.java

```
package model;

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

public class PessoaJuridicaRepo {
    private List<PessoaJuridica> pessoas = new ArrayList<>();

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

    public void alterar(PessoaJuridica pessoa) {
        PessoaJuridica existente = obter(pessoa.getId());
        if (existente != null) {
            existente.setNome(pessoa.getNome());
            existente.setCnpj(pessoa.getCnpj());
        }
    }

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

    public PessoaJuridica obter(int id) {
        return pessoas.stream().filter(p -> p.getId() == id).findFirst().orElse(null);
    }

    public List<PessoaJuridica> obterTodos() {
        return pessoas;
    }
```

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

    public void recuperar(String arquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(arquivo))) {
            pessoas = (List<PessoaJuridica>) ois.readObject();
        } catch (FileNotFoundException e) {
            pessoas = new ArrayList<>();
        }
    }
}
```

CadastroPOO.java

```
import model.PessoaFisica;
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;

public class CadastroPOO {
    public static void main(String[] args) {
        try {
            // Repositório de Pessoas Físicas
            PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
            repoFisica.inserir(new PessoaFisica(1, "Ana", "12345678900", 25));
            repoFisica.inserir(new PessoaFisica(2, "Carlos", "98765432100", 32));
            repoFisica.persistir("pessoasFisicas.dat");

            PessoaFisicaRepo repoFisicaRec = new PessoaFisicaRepo();
            repoFisicaRec.recuperar("pessoasFisicas.dat");
            System.out.println("Pessoas Físicas:");
            repoFisicaRec.obterTodos().forEach(PessoaFisica::exibir);

            // Repositório de Pessoas Jurídicas
            PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();
            repoJuridica.inserir(new PessoaJuridica(3, "Empresa A", "11222333444455"));
            repoJuridica.inserir(new PessoaJuridica(4, "Empresa B", "55666777888999"));
            repoJuridica.persistir("pessoasJuridicas.dat");

            PessoaJuridicaRepo repoJuridicaRec = new PessoaJuridicaRepo();
            repoJuridicaRec.recuperar("pessoasJuridicas.dat");
            System.out.println("Pessoas Jurídicas:");
            repoJuridicaRec.obterTodos().forEach(PessoaJuridica::exibir);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

ANALISE E CONCLUSAO

1. Quais as vantagens e desvantagens do uso de herança?
- Vantagens:
  - Reutilização de código, permitindo uma melhor organização e redução de redundâncias.
  - Facilita a extensão de funcionalidades em projetos com hierarquias bem definidas.
  - Permite o uso de polimorfismo, otimizando a manutenção e evolução do sistema.

- Desvantagens:
  - Aumento do acoplamento entre classes, dificultando a alteração da estrutura.
  - Complexidade adicional em hierarquias profundas ou mal planejadas.

2. Por que a interface Serializable é necessária ao efetuar persistência em arquivos binários?
- A interface Serializable é essencial para transformar objetos em fluxos de bytes, permitindo sua escrita em arquivos ou envio através da rede. Sem ela, a persistência de objetos em arquivos binários não seria possível.

3. Como o paradigma funcional é utilizado pela API stream no Java?
- A API Stream em Java utiliza o paradigma funcional para processar coleções de dados de forma declarativa. Operações como filter, map e reduce permitem manipular dados imutavelmente, com expressões lambda que tornam o código mais conciso e legível.

4. Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na persistência de dados em arquivos?
- O padrão de desenvolvimento mais comum para persistência em arquivos é o DAO (Data Access Object). Esse padrão separa a lógica de acesso aos dados da lógica de negócios, promovendo modularidade, reutilização e organização do código.

RESULTADO DA EXECUSÃO