UNIVERSIDADE ESTÁCIO DE SÁ

POLO CENTRO – IGREJINHA – RS

Curso: Desenvolvimento Fulll Stack

Disciplina: Iniciando o caminho pelo Java (RPG0014)

Turma: 2024.1 | 3º semestre

Nome: Emily Duarte Watthier

**Relatório de Prática: Cadastro com POO e Persistência Binária**

***Objetivos da Prática:***

- 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.

***Códigos Implementados:***

## 1. Entidades (pacote `model`)

- Pessoa.java

```java
package model;
import java.io.Serializable;

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

    public Pessoa() {}

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

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

- PessoaFisica.java

```java
package model;
import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {
    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;
    }

    @Override
    public void exibir() {
        System.out.println("Pessoa Física - ID: " + id + ", Nome: " + nome + ", CPF: " + c
    }

    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; }
}
```

- PessoaJuridica.java

```java
package model;
import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {
    private String cnpj;

    public PessoaJuridica() {}

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

    @Override
    public void exibir() {
        System.out.println("Pessoa Jurídica - ID: " + id + ", Nome: " + nome + ", CNPJ: "
    }

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

## 2. Repositórios (pacote `model`)

- PessoaFisicaRepo.java

```java
package model;

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

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

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

    public void alterar(PessoaFisica pf) {
        excluir(pf.getId());
        inserir(pf);
    }

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

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

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

    public void persistir(String nomeArquivo) throws Exception {
        try (FileOutputStream fos = new FileOutputStream(nomeArquivo);
             ObjectOutputStream oos = new ObjectOutputStream(fos)) {
            oos.writeObject(lista);
            System.out.println("Arquivo salvo em: " + new File(nomeArquivo).getAbsolutePat
        }
    }

    public void recuperar(String nomeArquivo) throws Exception {
        try (FileInputStream fis = new FileInputStream(nomeArquivo);
             ObjectInputStream ois = new ObjectInputStream(fis)) {
            lista = (ArrayList<PessoaFisica>) ois.readObject();
            System.out.println("Dados carregados de: " + new File(nomeArquivo).getAbsolute
        }
    }
}
```

### 3. Classe Principal - CadastroPOO.java

```java
package cadastropoo;

import model.*;

public class CadastroPOO {
    public static void main(String[] args) {
        try {
            PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
            repo1.inserir(new PessoaFisica(1, "Ana", "11111111111", 25));
            repo1.inserir(new PessoaFisica(2, "Carlos", "22222222222", 52));
            repo1.persistir("pessoas_fisicas.dat");
            System.out.println("Dados de Pessoa Fisica Armazenados.");

            PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
            repo2.recuperar("pessoas_fisicas.dat");
            System.out.println("Dados de Pessoa Fisica Recuperados.");
            for (PessoaFisica pf : repo2.obterTodos()) {
                pf.exibir();
                System.out.println();
            }

            PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
            repo3.inserir(new PessoaJuridica(3, "XPTO Sales", "33333333333333"));
            repo3.inserir(new PessoaJuridica(4, "XPTO Solutions", "44444444444444"));
            repo3.persistir("pessoas_juridicas.dat");
            System.out.println("Dados de Pessoa Juridica Armazenados.");

            PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
            repo4.recuperar("pessoas_juridicas.dat");
            System.out.println("Dados de Pessoa Juridica Recuperados.");
            for (PessoaJuridica pj : repo4.obterTodos()) {
                pj.exibir();
                System.out.println();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## Resultados da Execução:

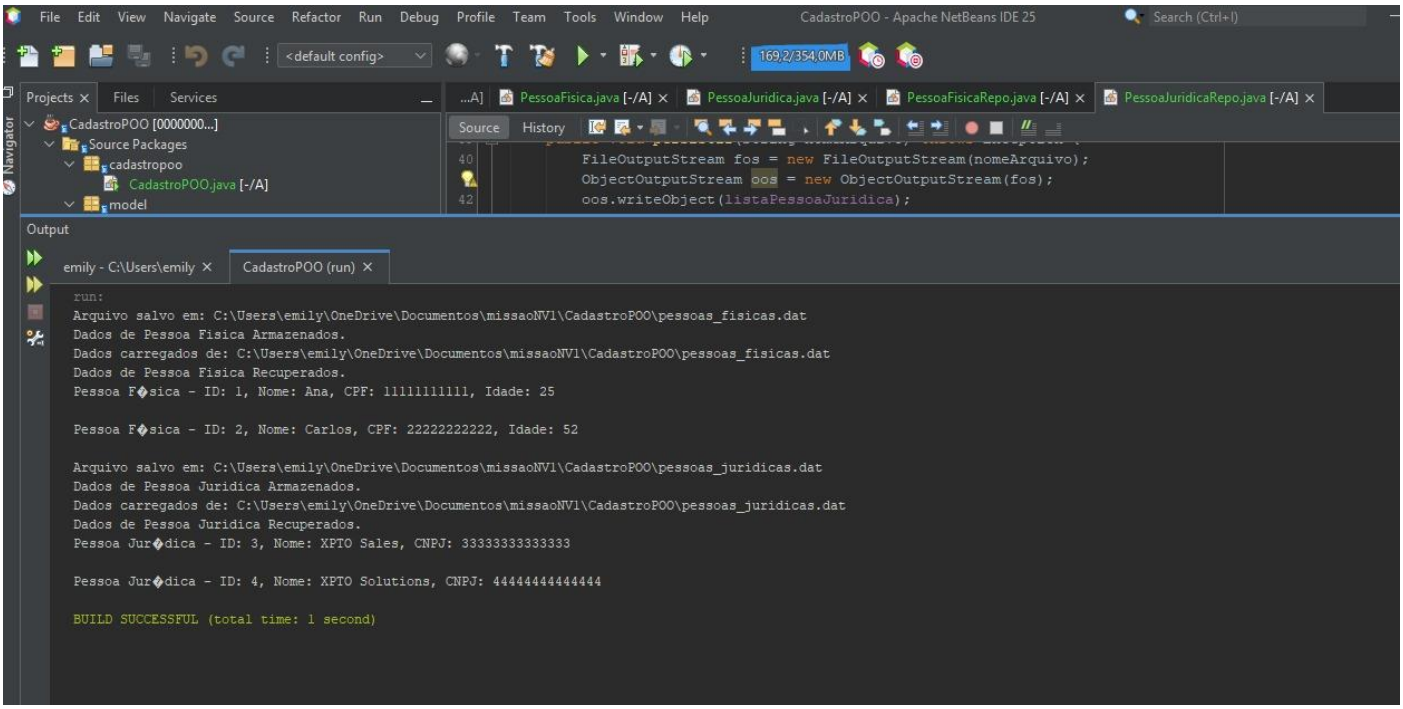
As execuções foram realizadas com sucesso, gerando os seguintes arquivos:

Pessoas_fisicas.dat

Pessoas_juridicas.dat

A saída do console apresentou a leitura correta dos dados armazenados e recuperados. Foram exibidas as informações de duas pessoas físicas e duas jurídicas.

Print da Execução:




***Repositório do Projeto:***

O projeto completo está hospedado no GitHub, no seguinte endereço:

https://github.com/Watthier09/missaoNV1JAVA/tree/main/missaoNV1/CadastroPOO

## Análise e Conclusão:

a. *Quais as vantagens e desvantagens do uso de herança?*

Vantagens:

- Reutilização de código entre classes.
- Estruturação hierárquica mais clara.
- Facilita extensões de funcionalidades.

Desvantagens:

- Forte acoplamento entre super e subclasses.
- Pode haver uso indevido em lugar de composição.
- Rastreio de comportamentos herdados pode ficar complexo.

b. *Por que a interface Serializable é necessária ao efetuar persistência em arquivos binários?*

A interface Serializable permite transformar objetos em uma sequência de bytes, essencial para que possam ser salvos e recuperados de arquivos binários. Sem ela, a JVM não permite a serialização.

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

   A API Stream permite operar sobre coleções de forma declarativa, utilizando funções como filter, map, reduce. Isso melhora a legibilidade e permite paralelismo. Embora não tenha sido utilizado nesta prática, é ideal para manipular listas.

d. *Qual padrão de desenvolvimento é adotado na persistência de dados em arquivos?*

O padrão DAO (Data Access Object) é o mais comum. Ele separa a lógica de acesso aos dados da lógica de negócios, como feito nas classes PessoaFisicaRepo e PessoaJuridicaRepo.