Universidade Estácio

Campus POLO FSP-RO

Curso: Desenvolvimento Full Stack

Disciplina: Nível 1: Iniciando o Caminho Pelo Java

Turma: 2025

Semestre Letivo: 1

Integrantes: Eliton Rodriguês de Oliveira

Relatório Discente de Acompanhamento

1. Título da Prática

Cadastro de Clientes em Java com Persistência em Arquivos Binários

2. Objetivo da Prática

O objetivo desta prática é implementar um sistema de cadastro de clientes utilizando a linguagem Java. O projeto deve utilizar os conceitos de herança e polimorfismo, além de persistência em arquivos binários com a interface Serializable.

3. Códigos Implementados

3.1. Classe Pessoa

```
package model;

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);
    }
}
```

3.2. Classe PessoaFisica

```
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 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);
    }
}
```

3.3. Classe PessoaJuridica

```
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 cnpj; }
    public void setCnpj(String cnpj) { this.cnpj = cnpj; }
    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
    }
}
```

3.4. Classe Main

```
package main;

import model.*;

import java.io.*;

import java.util.*;

public class Main {

   public static void main(String[] args) {

      PessoaFisica p1 = new PessoaFisica(1, "Ana", "11111111111", 25);

      PessoaFisica p2 = new PessoaFisica(2, "Carlos", "22222222222", 52);

      PessoaJuridica p3 = new PessoaJuridica(3, "XPTO Sales", "33333333333333");

      PessoaJuridica p4 = new PessoaJuridica(4, "XPTO Solutions", "44444444444444");

      List<Pessoa> pessoas = Arrays.asList(p1, p2, p3, p4);

      for (Pessoa p : pessoas) {

         p.exibir();

         System.out.println();

      }

   }

}
```

4. Resultados da Execução

A execução do programa gera a seguinte saída:

```
Id: 1

Nome: Ana

CPF: 11111111111

Idade: 25


Id: 2
```

Nome: Carlos

CPF: 22222222222

Idade: 52

Id: 3

Nome: XPTO Sales

CNPJ: 33333333333333

Id: 4

Nome: XPTO Solutions

CNPJ: 44444444444444

BUILD SUCCESSFUL (total time: 1 second)

5. Análise e Conclusão

5.1. Vantagens e Desvantagens do Uso de Herança

Vantagens:

- **Reutilização de código.**
- **Redução de redundância.**
- **Facilita a manutenção do sistema.**

Desvantagens:

- **Pode levar ao acoplamento excessivo.**
- **Flexibilidade reduzida se houver alterações na superclasse.**

5.2. Importância da Interface Serializable

A interface Serializable é necessária para gravar objetos em arquivos binários. Sem ela, o Java não consegue transformar os objetos em bytes para armazená-los e recuperá-los posteriormente.

5.3. Uso do Paradigma Funcional com a API Stream no Java

A API Stream do Java permite operar coleções de forma declarativa e imutável. Exemplos:

pessoas.stream().filter(p -> p instanceof PessoaFisica).forEach(Pessoa::exibir);

Aqui, usamos filter para filtrar PessoaFisica e forEach para exibir os dados.

5.4. Padrões de Persistência em Arquivos no Java

O Java geralmente usa o padrão DAO (Data Access Object) ou Repository para persistência de dados. Neste projeto, utilizamos repositórios (PessoaFisicaRepo e PessoaJuridicaRepo) para gerenciar a persistência dos objetos.

6. Repositório no Git

O projeto está disponível no GitHub: [URL DO REPOSITORIO]

Este relatório documenta toda a implementação, execução e análise do projeto de Cadastro de Clientes em Java, atendendo às diretrizes exigidas pela disciplina.