Campus: Santa Luiza
Curso: Tecnologia em Desenvolvimento FullStack
Disciplina: RPG0014 - Iniciando o caminho pelo Java
Integrantes: Maykel Furtado Lacerda Batista
Git do projeto: https://github.com/maykellacerda/CadastroPOO

Objetivo

O Objetivo deste trabalho prático é testar os conhecimentos aprendidos em java para desenvolver um cadastro de pessoa física e pessoa jurídica.

Usando os conceitos de orientação a objeto criei as classes a fim de chegar ao resultado esperado.

Códigos propostos pela atividade

A classe “Pessoa”, tem por objetivo ser a classe genérica para servir de base para a criação das classes “PessoaFisica” e “PessoaJuridica”.

```
/*
 * 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;

/**
 *
 * @author maykel
 */
public class Pessoa implements Serializable {

    private int id;
    private String nome;

    public Pessoa() {

    }

    public Pessoa(int id, String nome) {
        setId(id);
        setNome(nome);
    }

    /**
     * @return the id
     */
    public int getId() {
        return id;
    }

    /**
     * @param id the id to set
     */
    public void setId(int id) {
        this.id = id;
```

```
    }

    /**
     * @return the nome
     */
    public String getNome() {
        return nome;
    }

    /**
     * @param nome the nome to set
     */
    public void setNome(String nome) {
        this.nome = nome;
    }

    public void exibir() {
        System.out.println("Id: " + getId());
        System.out.println("nome: " + getNome());
    }

}
```

A classe “PessoaFisica” herda os atributos e métodos de “Pessoa”, porém, com a adição dos métodos próprios chamados de “idade” e “CPF”.

```
/*
 * 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;

/**
 *
 * @author maykel
 */
public class PessoaFisica extends Pessoa implements Serializable {

    private String CPF;
    private int idade;

    PessoaFisica() {

    }

    PessoaFisica(String CPF, int idade, int id, String nome) {
        super(id, nome);
        setCPF(CPF);
        setIdade(idade);
    }

    /**
     * @return the CPF
     */
    public String getCPF() {
        return CPF;
    }

    /**
     * @param CPF the CPF to set
     */
    public void setCPF(String CPF) {
        this.CPF = CPF;
    }

    /**
     * @return the idade
```

```
     */
    public int getIdade() {
        return idade;
    }

    /**
     * @param idade the idade to set
     */
    public void setIdade(int idade) {
        this.idade = idade;
    }

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CPF: " + this.getCPF());
        System.out.println("Idade: " + this.getIdade());
    }

}
```

A classe “PessoaJuridica” herda os atributos e métodos da classe “Pessoa”, ela possui como atributo próprio o CNPJ da empresa.

```
/*
 * 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;

/**
 *
 * @author maykel
 */
public class PessoaJuridica extends Pessoa implements Serializable {

    private String CNPJ;

    public PessoaJuridica() {

    }

    public PessoaJuridica(String CNPJ, int id, String nome) {
        super(id, nome);
        setCNPJ(CNPJ);
    }

    /**
     * @return the CNPJ
     */
    public String getCNPJ() {
        return CNPJ;
    }

    /**
     * @param CNPJ the CNPJ to set
     */
    public void setCNPJ(String CNPJ) {
        this.CNPJ = CNPJ;
    }

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CNPJ: " + CNPJ);
```

```
    }

}
```

A classe “PessoaFisicaRepo” trabalha com uma lista para armazenar, adicionar listar e excluir pessoas através de seus métodos, bem como os métodos “Persistir” e “Recuperar” que trabalham com a conversão de objetos para bytes e recuperar o mesmo.

```
/*
 * 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;

/**
 *
 * @author maykel
 */
public class PessoaFisicaRepo {

    //Lista pessoa fisica
    private List<PessoaFisica> listaPessoasFisicas;

    //Construtor da classe PessoaFisicaRepo
    public PessoaFisicaRepo() {
        this.listaPessoasFisicas = new ArrayList<>();
    }

    //Metodo para inserir uma pessoa fisica
    public void inserirPessoaFisica(PessoaFisica pessoaFisica) {
        listaPessoasFisicas.add(pessoaFisica);
    }

    //Faz uma pesquisa em pessoa fisica para inserir a alteração
    public void alterarPessoaFisica(int index, PessoaFisica pessoaFisica) {
        if (index >= 0 && index < listaPessoasFisicas.size()) {
            listaPessoasFisicas.set(index, pessoaFisica);
        }
    }

    //Exclui pessoa fisica
    public void excluirPessoaFisica(int index) {
        if (index >= 0 && index < listaPessoasFisicas.size()) {
            listaPessoasFisicas.remove(index);
        }
```

```
    }

    //Obtem uma pessoa fisica específica
    public PessoaFisica obterPessoaFisica(int index) {
        if (index >= 0 && index < listaPessoasFisicas.size()) {
            return listaPessoasFisicas.get(index);
        }
        return null;
    }

    //Lista todas as pessoas fisicas
    public List<PessoaFisica> obterTodos() {
        return new ArrayList<>(listaPessoasFisicas);
    }

    //Serialização para converter os objetos em bytes antes de escrevê-los no arquivo.
    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
            outputStream.writeObject(listaPessoasFisicas);
        }
    }

    //Recuperar uma lista de objetos
    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException
{
        try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
            listaPessoasFisicas = (List<PessoaFisica>) inputStream.readObject();
        }
    }
}
```

A classe “PessoaJuridicaRepo” trabalha com uma lista para armazenar, adicionar listar e excluir pessoas através de seus métodos, bem como os métodos “Persistir” e “Recuperar” que trabalham com a conversão de objetos para bytes e recuperar o mesmo.

```
/*
 * 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;

/**
 *
 * @author maykel
 */
public class PessoaJuridicaRepo {

    //Lista para pessoa juridica
    private List<PessoaJuridica> listaPessoasJuridicas;

    //Construtos pessoaJuridicaRepo
    public PessoaJuridicaRepo() {
        this.listaPessoasJuridicas = new ArrayList<>(); //Lista pessoas juridicas
    }

    //Metodo para inserir Pessoa Juridica
    public void inserirPessoaJuridica(PessoaJuridica pessoaJuridica) {
        listaPessoasJuridicas.add(pessoaJuridica);
    }

    //Função que altera pessoa jurifica, faz a pesquisa, se achar altera o valor
    public void alterarPessoaJuridica(int index, PessoaJuridica pessoaJuridica) {
        if (index >= 0 && index < listaPessoasJuridicas.size()) {
            listaPessoasJuridicas.set(index, pessoaJuridica);
        }
    }

    //Função para excluir pessoa juridica.
    public void excluirPessoaJuridica(int index) {
        if (index >= 0 && index < listaPessoasJuridicas.size()) {
            listaPessoasJuridicas.remove(index);
        }
```

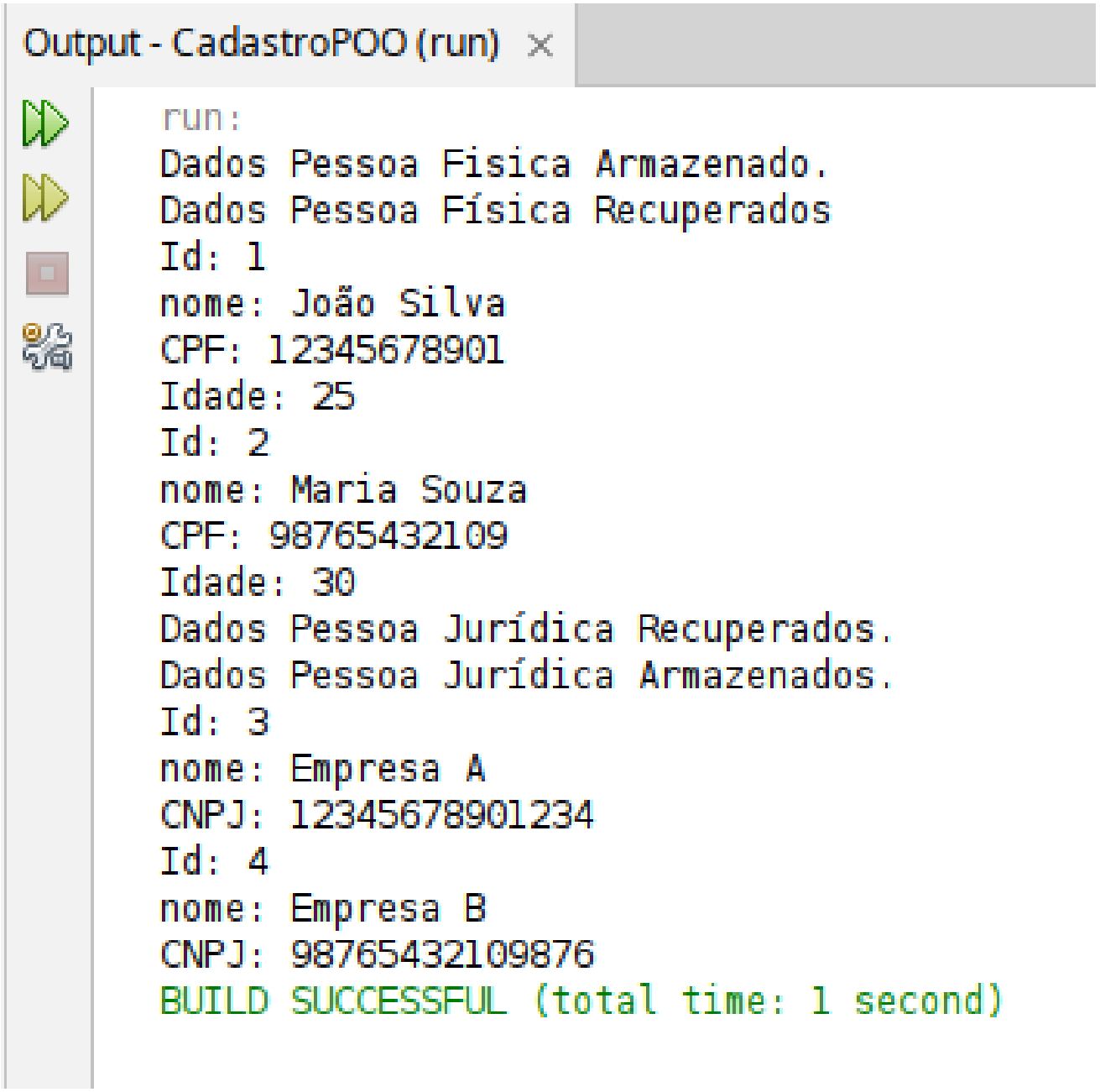
```
    }

    //Função para obter uma pessoa juridica.
    public PessoaJuridica obterPessoaJuridica(int index) {
        if (index >= 0 && index < listaPessoasJuridicas.size()) {
            return listaPessoasJuridicas.get(index);
        }
        return null;
    }

    //Lista todas as pessoas juridicas
    public List<PessoaJuridica> obterTodos() {
        return new ArrayList<>(listaPessoasJuridicas);
    }

    //Serialização para converter os objetos em bytes antes de escrevê-los no arquivo.
    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
            outputStream.writeObject(listaPessoasJuridicas);
        }
    }

    //Recuperar uma lista de objetos
    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException
{
        try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
            listaPessoasJuridicas = (List<PessoaJuridica>) inputStream.readObject();
        }
    }
}
```

A classe “Main” tem a finalidade de testar o código para obter os resultados desejados conforme o solicitado.

```
/*
 * 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;

public class Main {

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

        // Adicionar duas pessoas físicas, utilizando o construtor completo.
        PessoaFisica pessoaFisica1 = new PessoaFisica("12345678901", 25, 1, "João Silva");
        PessoaFisica pessoaFisica2 = new PessoaFisica("98765432109", 30, 2, "Maria
Souza");
        repo1.inserirPessoaFisica(pessoaFisica1);
        repo1.inserirPessoaFisica(pessoaFisica2);

        // Invocar o método de persistência em repo1, fornecendo um nome de arquivo fixo.
        String nomeArquivoPessoaFisica = "pessoasFisicas.dat";
        try {
            repo1.persistir(nomeArquivoPessoaFisica);
        } catch (Exception e) {
            System.out.println("Erro ao persistir 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.
        try {
            repo2.recuperar(nomeArquivoPessoaFisica);
        } catch (Exception e) {
            System.out.println("Erro ao recuperar pessoas físicas: " + e.getMessage());
        }

        // Exibir os dados de todas as pessoas físicas recuperadas.
        System.out.println("Dados Pessoa Fisica Armazenado.");
        System.out.println("Dados Pessoa Física Recuperados");
        for (PessoaFisica pessoa : repo2.obterTodos()) {
```

```
            pessoa.exibir();
        }

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

        // Adicionar duas pessoas jurídicas, utilizando o construtor completo.
        PessoaJuridica pessoaJuridica1 = new PessoaJuridica("12345678901234", 3,
"Empresa A");
        PessoaJuridica pessoaJuridica2 = new PessoaJuridica("98765432109876", 4,
"Empresa B");
        repo3.inserirPessoaJuridica(pessoaJuridica1);
        repo3.inserirPessoaJuridica(pessoaJuridica2);

        // Invocar o método de persistência em repo3, fornecendo um nome de arquivo fixo.
        String nomeArquivoPessoaJuridica = "pessoasJuridicas.dat";
        try {
            repo3.persistir(nomeArquivoPessoaJuridica);
        } catch (Exception e) {
            System.out.println("Erro ao persistir 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.
        try {
            repo4.recuperar(nomeArquivoPessoaJuridica);
        } catch (Exception e) {
            System.out.println("Erro ao recuperar pessoas jurídicas: " + e.getMessage());
        }

        // Exibir os dados de todas as pessoas jurídicas recuperadas.
         System.out.println("Dados Pessoa Jurídica Recuperados.");
        System.out.println("Dados Pessoa Jurídica Armazenados.");
        for (PessoaJuridica pessoa : repo4.obterTodos()) {
            pessoa.exibir();
        }
    }
}
```

Tela do resultado



Análise e Conclusão

1) **Vantagens e desvantagens da herança**:

a) **Vantagem**: Reutilização de Código, Extensibilidade, Polimorfismo, Organização e Hierarquização

a.2) **Desvantagem**: Acoplamento Forte, Herança Multipla Compleza, se muito aprofundada pode ser confusa, Rigidez na hierarquia de classes