**CENTRO UNIVERSITÁRIO ESTÁCIO DE SANTA CATARINA**

**Missão | PráticaNível 1 | Mundo 3**

**Desenvolvimento FullStack**

**Jonison Rebelatto Moura Barbosa – 2023.03.93437-8**
https://github.com/rebelatto/estacio-fullstack-mp1-mundo3.git

# Relatório de Prática

## 1º Procedimento | Criação da Entidades e Sistema de Persistência

### Objetivo da Prática

O objetivo desta prática é estruturar e desenvolver um sistema de cadastro de pessoas físicas e jurídicas utilizando os princípios da Programação Orientada a Objetos (POO) em Java, com foco na aplicação dos conceitos da POO como herança, polimorfismo, encapsulamento e serialização de objetos. Além disso, busca-se implementar um mecanismo de persistência de dados em arquivos por meio de repositórios específicos para cada tipo de entidade, promovendo a organização do código, a reutilização de estruturas e o gerenciamento completo do ciclo de vida dos objetos (inserção, alteração, exclusão e recuperação).

## Estrutura de arquivos


## Códigos Solicitados no Roteiro de Aula

Classe Pessoa.java

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

public abstract class Pessoa implements Serializable {
    private static final long serialVersionUID = 1L;
    protected int id;
    protected String nome;

    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;
   }

    protected void exibir(int id, String nome){
        System.out.println("Id: " + id);
        System.out.println("Nome: " + nome);
    }

}
```

Classe PessoaFisica.java

```java
package model;


public class PessoaFisica extends Pessoa{
    private static final long serialVersionUID = 1L;

    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;
    }

    public void exibir(){
        super.exibir(id,nome);
        System.out.println("CPF: " + getCpf());
        System.out.println("Idade: " + getIdade());
    }
}
```

Estácio

Classe PessoaJuridica.java

```java
package model;


public class PessoaJuridica extends Pessoa{
    private static final long serialVersionUID = 1L;

    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;
    }

     public void exibir(){
        super.exibir(id, nome);
        System.out.println("CNPJ: " + getCnpj());
    }
}
```

Classe PessoaJuridicaRepo.java

```java
package model;
import java.io.*;
import java.util.ArrayList;

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

    public void inserir(PessoaJuridica empresa) {
        empresas.add(empresa);
    }

    public void alterar(PessoaJuridica empresa) {
        for (int i = 0; i < empresas.size(); i++) {
            if (empresas.get(i).getId() == empresa.getId()) {
                empresas.set(i, empresa);
                return;
            }
        }
    }

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

    public PessoaJuridica obter(int id) {
        for (PessoaJuridica e : empresas) {
            if (e.getId() == id) return e;
        }
        return null;
    }

    public ArrayList<PessoaJuridica> obterTodos() {
        return new ArrayList<>(empresas);
    }

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
            oos.writeObject(empresas);
        } catch (IOException  e) {
        e.printStackTrace();
    }
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            empresas = (ArrayList<PessoaJuridica>) ois.readObject();
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

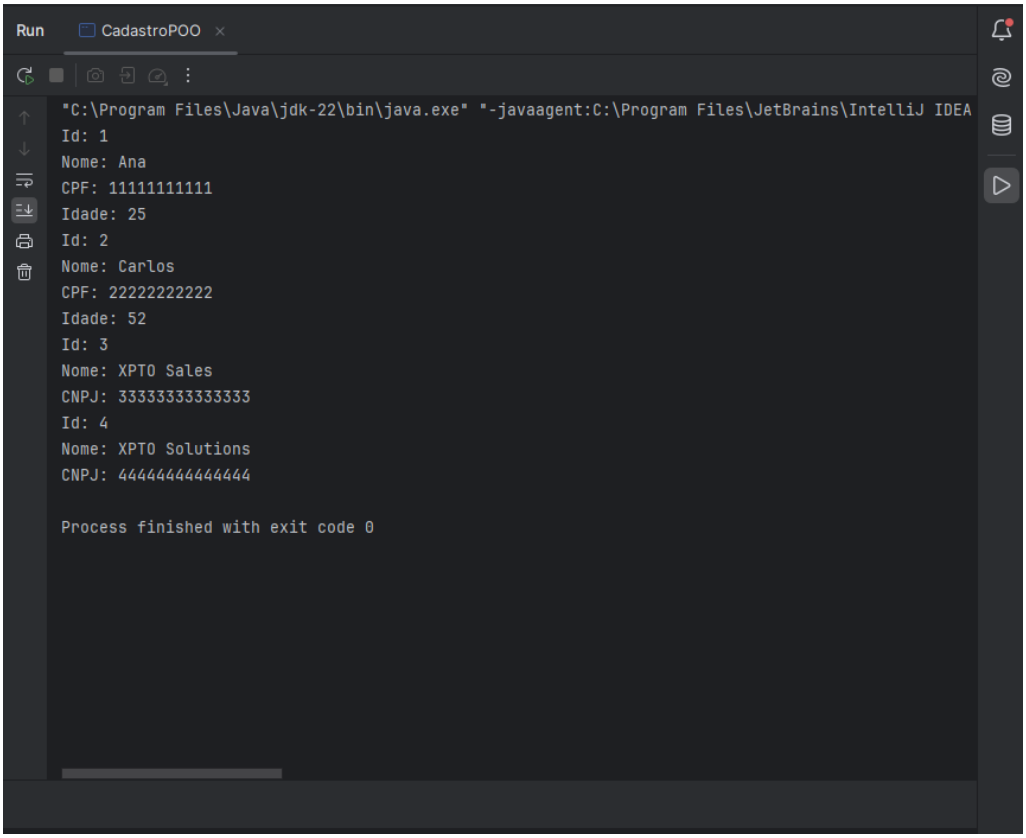
Classe PessoaFisicaRepo.java

```java
package model;
import java.io.*;
import java.util.*;

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

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

    public void alterar(PessoaFisica pessoa) {
        for (int i = 0; i < pf.size(); i++) {
            if (pf.get(i).getId() == pessoa.getId()) {
                pf.set(i, pessoa);
                return;
            }
        }
    }

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

    public PessoaFisica obter(int id) {
        for (PessoaFisica p : pf) {
            if (p.getId() == id) return p;
        }
        return null;
    }

    public ArrayList<PessoaFisica> obterTodos() {
        return new ArrayList<>(pf);
    }

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
            oos.writeObject(pf);
        } catch (IOException  e) {
            e.printStackTrace();
        }
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            pf = (ArrayList<PessoaFisica>) ois.readObject();
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

Classe 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("pf_repo.dat");

          PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
          repo2.recuperar("pf_repo.dat");
          for (PessoaFisica pf : repo2.obterTodos()) {
             pf.exibir();
          }

          PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
          repo3.inserir(new PessoaJuridica(3, "XPTO Sales", "33333333333333"));
          repo3.inserir(new PessoaJuridica(4, "XPTO Solutions", "44444444444444"));
          repo3.persistir("pj_repo.dat");

          PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
          repo4.recuperar("pj_repo.dat");
          for (PessoaJuridica pj : repo4.obterTodos()) {
             pj.exibir();
          }

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

    }
}
```

**Resultados da execução dos códigos**




## 1) Análise e Conclusão

a) Quais as vantagens de desvantagens do uso de herança?

- **Reutilização de código:** Permite que subclasses herdem atributos e métodos da superclasse, evitando duplicação.
- **Facilita a manutenção:** Alterações na superclasse são refletidas automaticamente nas subclasses.
- **Organização e clareza:** Favorece um modelo hierárquico que facilita a leitura e estrutura do sistema.

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

c) Como o paradigma funcional é utilizado pela API Stream no Java?

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

**2º Procedimento | Criação da Entidades e Sistema de Persistência**

1) Objetivo da Pratica

2) Códigos Solicitados no Roteiro de Aula

3) Resultados da execução dos códigos

4) Análise e Conclusão
   a) Quais as vantagens de desvantagens do uso de herança?

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

   c) Como o paradigma funcional é utilizado pela API Stream no Java?

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

**Referências**

Stackoverflow - https://pt.stackoverflow.com/questions/88270/qual-a-finalidade-da-interface-serializable. Acessado em março/2025

**Ferramentas**

IntelliJ Idea– disponível em https://www.jetbrains.com/pt-br/idea/.

Acessado em fev/2025