

## Faculdade Estácio - Polo Curitiba - Centro

**Curso:** Desenvolvimento Full Stack

**Disciplina:** Desenvolvimento Full Stack

**Número da Turma:** RPG0014

**Semestre Letivo:** 3

**Integrante:** Mariana Lucas Fernandes Onório

**Repositório:** <https://github.com/MariLF0/estacio-mundo3-missao-nivel-1>

### Sumário:

<b>Faculdade Estácio - Polo Curitiba - Centro</b>	<b>1</b>
Sumário:	1
1. Título da Prática:	2
2. Objetivos da Prática:	2
3. Arquivos do Projeto:	2
Arquivo: Pessoa.java	2
Arquivo: PessoaFisica.java	3
Arquivo: PessoaJuridica.java	3
Arquivo: PessoaFisicaRepo.java	4
Arquivo: PessoaJuridicaRepo.java	5
Arquivo: Main.java	6
4. Resultados da execução dos códigos:	7
5. Análise e Conclusão:	7
a. Quais as vantagens e desvantagens do uso de herança?	7
b. Porque a interface Serializable é necessária ao efetuar persistência em arquivos binários?.....	7
c. Como o paradigma funcional é utilizado pela API stream no Java?.....	8
d. Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na persistência de dados em arquivos?.....	8

## 1. Título da Prática:

Iniciando o caminho pelo Java

## 2. Objetivos da Prática:

1. Utilizar herança e polimorfismo na definição de entidades.
2. Utilizar persistência de objetos em arquivos binários.
3. Implementar uma interface cadastral em modo texto.
4. Utilizar o controle de exceções da plataforma Java.
5. No final do projeto, o aluno terá implementado um sistema cadastral em Java, utilizando os recursos da programação orientada a objetos e a persistência em arquivos binários.

## 3. Arquivos do Projeto:

### Arquivo: Pessoa.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: " + this.id + ", nome: " + this.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;
    }
}
```

## Arquivo: PessoaFisica.java

```
package model;

import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {
    private String cpf;
    private int idade;

    public PessoaFisica(int id, String nome, String cpf, int idade){
        this.id = id;
        this.nome = nome;
        this.cpf = cpf;
        this.idade = idade;
    }

    @Override
    public void exibir() {
        System.out.println("id: " + this.id + ", nome: " + this.nome + ", cpf: " + this.cpf + ",
idade: " + this.idade);
    }
}
```

## Arquivo: PessoaJuridica.java

```
package model;

import java.io.Serializable;

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

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

    @Override
    public void exibir() {
        System.out.println("id: " + this.id + ", nome: " + this.nome + ", cnpj: " + this.cnpj);
    }
}
```

## Arquivo: PessoaFisicaRepo.java

```
package model;

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

public class PessoaFisicaRepo {
    private ArrayList<PessoaFisica> pessoasFisicas;

    public PessoaFisicaRepo(){
        this.pessoasFisicas = new ArrayList<PessoaFisica>();
    }

    public void inserir(PessoaFisica entidade) {
        this.pessoasFisicas.add(entidade);
    }

    public void alterar(PessoaFisica entidade){
        this.excluir(entidade.id);
        this.inserir(entidade);
    }

    public void excluir(int id){
        this.pessoasFisicas.removeIf(item -> item.id == id);
    }

    public ArrayList<PessoaFisica> obterTodos(){
        return this.pessoasFisicas;
    }

    public void persistir(String nomeArquivo) throws IOException {
        FileOutputStream fos = new FileOutputStream(nomeArquivo);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(this.pessoasFisicas);
        System.out.println("Dados de Pessoa Física Armazenados.");
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        FileInputStream fis = new FileInputStream(nomeArquivo);
        ObjectInputStream ois = new ObjectInputStream(fis);
        this.pessoasFisicas = (ArrayList<PessoaFisica>) ois.readObject();
        System.out.println("Dados de Pessoa Física Recuperados.");
    }
}
```

## Arquivo: PessoaJuridicaRepo.java

```
package model;

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

public class PessoaJuridicaRepo {
    private ArrayList<PessoaJuridica> pessoasJuridicas;

    public PessoaJuridicaRepo(){
        this.pessoasJuridicas = new ArrayList<PessoaJuridica>();
    }

    public void inserir(PessoaJuridica entidade) {
        this.pessoasJuridicas.add(entidade);
    }

    public void alterar(PessoaJuridica entidade){
        this.excluir(entidade.id);
        this.inserir(entidade);
    }

    public void excluir(int id){
        this.pessoasJuridicas.removeIf(item -> item.id == id);
    }

    public ArrayList<PessoaJuridica> obterTodos(){
        return this.pessoasJuridicas;
    }

    public void persistir(String nomeArquivo) throws IOException {
        FileOutputStream fos = new FileOutputStream(nomeArquivo);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(this.pessoasJuridicas);
        System.out.println("Dados de Pessoa Jurídica Armazenados.");
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        FileInputStream fis = new FileInputStream(nomeArquivo);
        ObjectInputStream ois = new ObjectInputStream(fis);
        this.pessoasJuridicas = (ArrayList<PessoaJuridica>) ois.readObject();
        System.out.println("Dados de Pessoa Jurídica Recuperados.");
    }
}
```

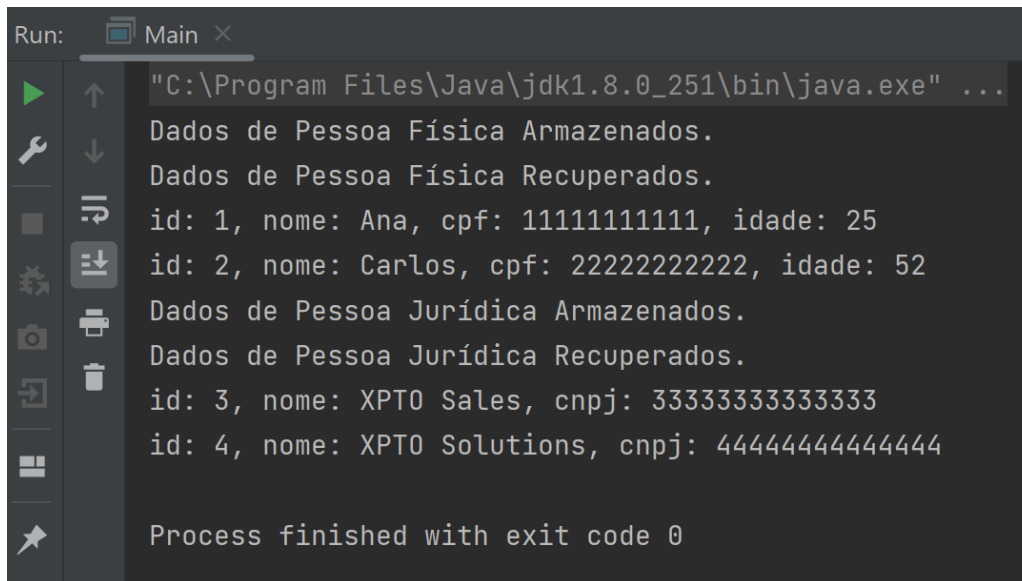
## Arquivo: Main.java

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

import java.io.IOException;

public class Main {
    public static void main(String[] args) {
        // Pessoa Física
        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
        repo1.inserir(new PessoaFisica(1, "Ana", "11111111111", 25));
        repo1.inserir(new PessoaFisica(2, "Carlos", "22222222222", 52));
        try{
            repo1.persistir("pessoasFisicas.dat");
        }
        catch(IOException exception) {
            exception.printStackTrace();
        }
        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
        try {
            repo2.recuperar("pessoasFisicas.dat");
        } catch (Exception exception) {
            exception.printStackTrace();
        }
        for (PessoaFisica pessoaFisica : repo2.obterTodos()) {
            pessoaFisica.exibir();
        }
        // Pessoa Jurídica
        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
        repo3.inserir(new PessoaJuridica(3, "XPTO Sales", "3333333333333333"));
        repo3.inserir(new PessoaJuridica(4, "XPTO Solutions", "4444444444444444"));
        try{
            repo3.persistir("pessoasJuridicas.dat");
        }
        catch(Exception exception) {
            exception.printStackTrace();
        }
        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
        try {
            repo4.recuperar("pessoasJuridicas.dat");
        } catch (Exception exception) {
            exception.printStackTrace();
        }
        for (PessoaJuridica pessoaJuridica : repo4.obterTodos()) {
            pessoaJuridica.exibir();
        }
    }
}
```

## 4. Resultados da execução dos códigos:



```
Run: Main X
"C:\Program Files\Java\jdk1.8.0_251\bin\java.exe" ...
Dados de Pessoa Física Armazenados.
Dados de Pessoa Física Recuperados.
id: 1, nome: Ana, cpf: 11111111111, idade: 25
id: 2, nome: Carlos, cpf: 22222222222, idade: 52
Dados de Pessoa Jurídica Armazenados.
Dados de Pessoa Jurídica Recuperados.
id: 3, nome: XPT0 Sales, cnpj: 33333333333333
id: 4, nome: XPT0 Solutions, cnpj: 44444444444444
Process finished with exit code 0
```

## 5. Análise e Conclusão:

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

**Resposta:** Reutilização de código: A herança permite que as classes filhas herdem atributos e métodos da classe mãe, o que evita a duplicação de código.

Além disso, isso permite que as classes filhas possam ser tratadas como objetos da classe mãe, o que facilita a criação de código mais genérico através do polimorfismo e aumenta a flexibilidade do sistema.

Outra vantagem é a facilidade de manutenção, através da herança, as mudanças feitas na classe mãe são refletidas diretamente nas classes filhas, sem precisar alterar o código em diversos lugares.

Uma desvantagem de se utilizar a herança é que ela pode gerar hierarquias complexas, por vezes dificultando a escalabilidade e flexibilidade de um projeto mais extenso, além disso, a herança quebra o encapsulamento já que uma classe filha depende diretamente da classe mãe, tornando-se mais frágil.

### b. Porque a interface Serializable é necessária ao efetuar persistência em arquivos binários?

**Resposta:** A interface Serializable é necessária ao efetuar persistência em arquivos binários porque ela sinaliza que estes objetos estão aptos a serem serializados. Não implementar essa interface resulta na exceção "java.io.NotSerializableException".

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

**Resposta:** A API Streams do Java é usada para processar coleções de objetos. Uma stream é uma sequência de objetos que suporta vários métodos que podem ser encadeados para produzir o resultado desejado sem alterar os dados originais.

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

**Resposta:** Quando se trata de persistência de dados o JPA (Java Persistence API), Hibernate e Spring JDBC, são alguns dentre os diversos frameworks utilizados para este fim.