



# Estácio

## Mundo 3 - Missão 1

Faculdade Estácio - Polo Centro - Canela - RS

Curso: Desenvolvimento Full Stack

Disciplina: RPG0014 - Iniciando o caminho pelo java.

Turma: 9001 - Semestre Letivo: 2025.1 - 3º semestre

Integrante: Pedro Henrique Marques Medeiros Pinho

Matrícula: 202402031831

Repositório Git: <https://github.com/PedroPinho23/Iniciando-o-caminho-pelo-Java.git>

## Iniciando o caminho pelo Java

Desenvolver uma aplicação em Java utilizando orientação a objetos com foco em herança, polimorfismo e persistência em arquivos binários. O objetivo é modelar um sistema de cadastro de pessoas físicas e jurídicas com repositórios separados e salvar os dados em arquivos locais.

### 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.
- 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.

## Códigos Utilizados:

### CadastroPOO.java:

```
package cadastropoo;

import model.*;

public class CadastroPOO {
    public static void main(String[] args) {
        try {
            // Pessoas Físicas
            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.\n");

            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();
            }

            // Pessoas Jurídicas
            PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
            repo3.inserir(new PessoaJuridica(3, "XPTO Sales", "3333333333333333"));
            repo3.inserir(new PessoaJuridica(4, "XPTO Solutions", "4444444444444444"));
            repo3.persistir("pessoas_juridicas.dat");
            System.out.println("Dados de Pessoa Juridica Armazenados.\n");

            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) {
        System.out.println("Erro: " + e.getMessage());
        e.printStackTrace();
    }
}

```

### Pessoa.java:

```

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 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:

```
package model;
```

```

public class PessoaFisica extends Pessoa {
    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() {
        super.exibir();
        System.out.println("CPF: " + cpf);
        System.out.println("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;
    }
}

```

### PessoaJuridica.java:

package model;

```

public class PessoaJuridica extends Pessoa {
    private String cnpj;

    public PessoaJuridica() {
    }

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

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

    public String getCnpj() {
        return cnpj;
    }
}

```

```

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

```

### PessoaFisicaRepo.java:

```

package model;

```

```

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

```

```

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

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

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

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

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

```

```

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

    public void persistir(String nomeArquivo) throws Exception {
        FileOutputStream fos = new FileOutputStream(nomeArquivo);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(pessoas);
        oos.close();
        fos.close();
    }

    public void recuperar(String nomeArquivo) throws Exception {
        FileInputStream fis = new FileInputStream(nomeArquivo);
        ObjectInputStream ois = new ObjectInputStream(fis);
        pessoas = (ArrayList<PessoaFisica>) ois.readObject();
        ois.close();
        fis.close();
    }
}

```

### PessoaJuridicaRepo.java:

```

package model;

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

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

    public void inserir(PessoaJuridica pessoa) {
        pessoas.add(pessoa);
    }

    public void alterar(PessoaJuridica pessoa) {
        for (int i = 0; i < pessoas.size(); i++) {
            if (pessoas.get(i).getId() == pessoa.getId()) {

```



```

        pessoas.set(i, pessoa);
        return;
    }
}

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

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

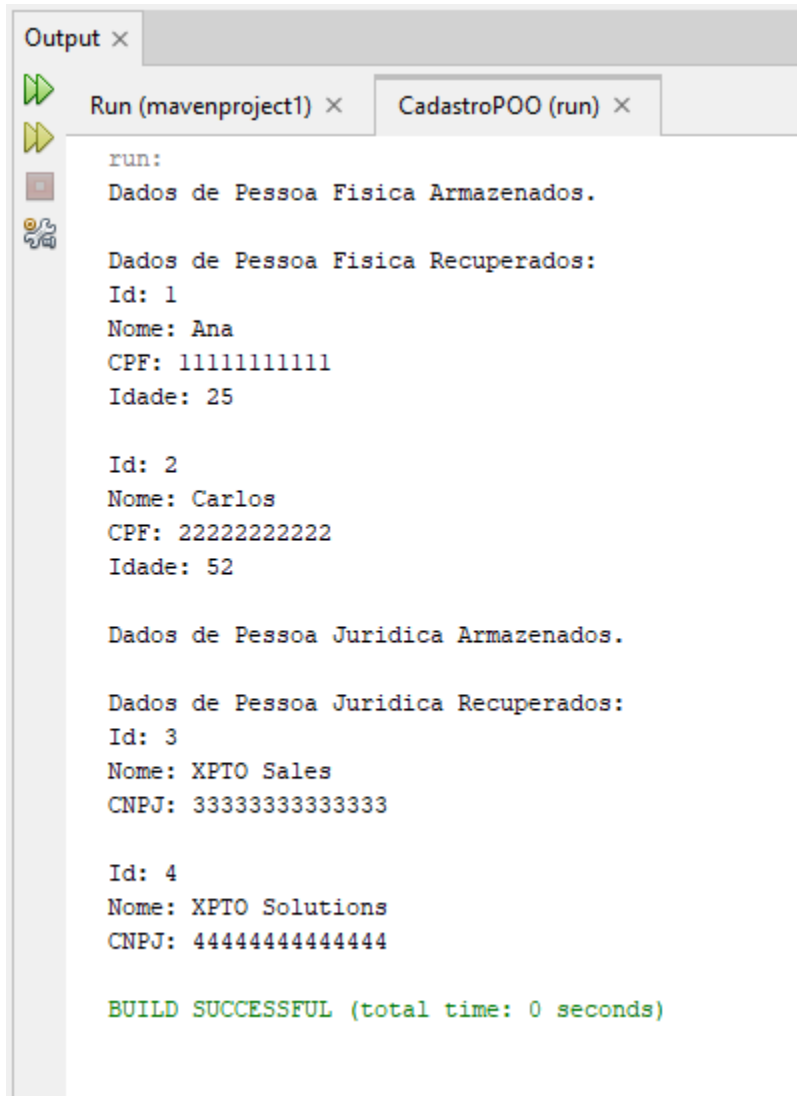
public ArrayList<PessoaJuridica> obterTodos() {
    return pessoas;
}

public void persistir(String nomeArquivo) throws Exception {
    FileOutputStream fos = new FileOutputStream(nomeArquivo);
    ObjectOutputStream oos = new ObjectOutputStream(fos);
    oos.writeObject(pessoas);
    oos.close();
    fos.close();
}

public void recuperar(String nomeArquivo) throws Exception {
    FileInputStream fis = new FileInputStream(nomeArquivo);
    ObjectInputStream ois = new ObjectInputStream(fis);
    pessoas = (ArrayList<PessoaJuridica>) ois.readObject();
    ois.close();
    fis.close();
}
}

```

## Resultados:



```
run:
Dados de Pessoa Fisica Armazenados.

Dados de Pessoa Fisica Recuperados:
Id: 1
Nome: Ana
CPF: 11111111111
Idade: 25

Id: 2
Nome: Carlos
CPF: 22222222222
Idade: 52

Dados de Pessoa Juridica Armazenados.

Dados de Pessoa Juridica Recuperados:
Id: 3
Nome: XPTO Sales
CNPJ: 33333333333333

Id: 4
Nome: XPTO Solutions
CNPJ: 4444444444444444

BUILD SUCCESSFUL (total time: 0 seconds)
```

Foram realizados testes diretos no método main com inserção de dados manual, salvamento e recuperação utilizando arquivos .bin, e exibição correta dos dados no terminal.

## Conclusão

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

Herança facilita a reutilização de código e deixa o sistema mais organizado, mas se usada sem planejamento pode causar acoplamento excessivo e dificultar alterações futuras.

### 2. Por que a interface Serializable é necessária ao efetuar persistência em arquivos binários?

Porque permite transformar objetos em uma sequência de bytes, possibilitando salvar e restaurar os dados da memória usando arquivos binários.

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

A API Stream usa funções como filter, map e forEach, permitindo que a gente processe listas com menos código e mais clareza, no estilo funcional.

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

O padrão mais comum é o DAO (Data Access Object), separando a lógica de acesso a dados do restante da aplicação.