

**Campus:** 202301037751 - POLO CENTRO - PORTO REAL - RJ

**Curso:** Desenvolvimento Full Stack

**Disciplina:** Nível 1: Iniciando o Caminho Pelo Java

**Turma:** 9001

**Semestre letivo:** 3º Semestre

**Nome do estudante:** Leonardo Naves de Lima Araujo

Missão Prática | Nível 1 | Mundo 3

RPG0014 - Iniciando o caminho pelo Java

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

Implementação de Cadastro de Clientes em Java com Persistência em Arquivos

### 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,
6. utilizando os recursos da programação orientada a objetos e a persistência em
7. arquivos binários.

### Códigos da Prática

#### Main.java

---

```
package model;
```

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

```
public class Main {
```

```
    public static void main(String[] args) {  
        String arquivoPessoasFisicas = "pessoasFisicas.dat";  
        String arquivoPessoasJuridicas = "pessoasJuridicas.dat";
```

```
        PessoaFisicaRepo repoPessoaFisica = new PessoaFisicaRepo();  
        repoPessoaFisica.inserir(new PessoaFisica(1, "Ana", "1111111111", 25));  
        repoPessoaFisica.inserir(new PessoaFisica(2, "Carlos", "2222222222", 52));
```

```

try {
    repoPessoaFisica.persistir(arquivoPessoasFisicas);
    System.out.println("Dados de Pessoa Fisica Armazenados.");

    PessoaFisicaRepo repoPessoaFisicaRecuperado = new PessoaFisicaRepo();
    repoPessoaFisicaRecuperado.recuperar(arquivoPessoasFisicas);
    System.out.println("Dados de Pessoa Fisica Recuperados.");
    repoPessoaFisicaRecuperado.obterTodos().forEach(PessoaFisica::exibir);
} catch (Exception e) {
    e.printStackTrace();
}

PessoaJuridicaRepo repoPessoaJuridica = new PessoaJuridicaRepo();
repoPessoaJuridica.inserir(new PessoaJuridica(3, "XPTO Sales", "3333333333333333"));
repoPessoaJuridica.inserir(new PessoaJuridica(4, "XPTO Solutions",
"4444444444444444"));

try {
    repoPessoaJuridica.persistir(arquivoPessoasJuridicas);
    System.out.println("Dados de Pessoa Juridica Armazenados.");

    PessoaJuridicaRepo repoPessoaJuridicaRecuperado = new PessoaJuridicaRepo();
    repoPessoaJuridicaRecuperado.recuperar(arquivoPessoasJuridicas);
    System.out.println("Dados de Pessoa Juridica Recuperados.");
    repoPessoaJuridicaRecuperado.obterTodos().forEach(PessoaJuridica::exibir);
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

## Pessoa.java

---

```

package model;

import java.io.Serializable;

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

    // Construtor padrão
    public Pessoa() {
    }
}

```

```

// Construtor completo
public Pessoa(int id, String nome) {
    this.id = id;
    this.nome = nome;
}

// Getters e Setters
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;
}

// Método exibir
public void exibir() {
    System.out.println("Id: " + id);
    System.out.println("Nome: " + nome);
}

// Sobrescrita do método toString
@Override
public String toString() {
    return "Pessoa{" + "id=" + id + ", nome=" + nome + "}";
}
}

```

## **PessoaFisica.java**

---

```

package model;

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

    // Construtor padrão
    public PessoaFisica() {
        super();
    }
}

```

```

    }

    // Construtor completo
    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);
    }

    @Override
    public String toString() {
        return super.toString() + ", CPF: " + cpf + ", Idade: " + idade;
    }
}

```

## PessoaFisicaRepo.java

---

```
package model;
```

```

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

```

```

public class PessoaFisicaRepo {
    private ArrayList<PessoaFisica> pessoasFisicas;

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

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

    public void alterar(PessoaFisica pessoaFisica) {
        int index = pessoasFisicas.indexOf(pessoaFisica);
        if (index != -1) {
            pessoasFisicas.set(index, pessoaFisica);
        }
    }

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

    public PessoaFisica obter(int id) {
        return pessoasFisicas.stream()
            .filter(p -> p.getId() == id)
            .findFirst()
            .orElse(null);
    }

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

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream out = new ObjectOutputStream(new
        FileOutputStream(nomeArquivo))) {
            out.writeObject(pessoasFisicas);
        }
    }

    @SuppressWarnings("unchecked")
    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException
    {
        try (ObjectInputStream in = new ObjectInputStream(new
        FileInputStream(nomeArquivo))) {
            pessoasFisicas = (ArrayList<PessoaFisica>) in.readObject();
        }
    }
}

```

## PessoaJuridica.java

---

```
package model;

public class PessoaJuridica extends Pessoa {
    private String cnpj;

    public PessoaJuridica() {
        super();
    }

    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);
    }

    @Override
    public String toString() {
        return super.toString() + ", CNPJ: " + cnpj;
    }
}
```

## PessoaJuridicaRepo.java

---

```
package model;

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

public class PessoaJuridicaRepo {
```

```

private ArrayList<PessoaJuridica> pessoasJuridicas;

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

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

public void alterar(PessoaJuridica pessoaJuridica) {
    int index = pessoasJuridicas.indexOf(pessoaJuridica);
    if (index != -1) {
        pessoasJuridicas.set(index, pessoaJuridica);
    }
}

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

public PessoaJuridica obter(int id) {
    return pessoasJuridicas.stream()
        .filter(p -> p.getId() == id)
        .findFirst()
        .orElse(null);
}

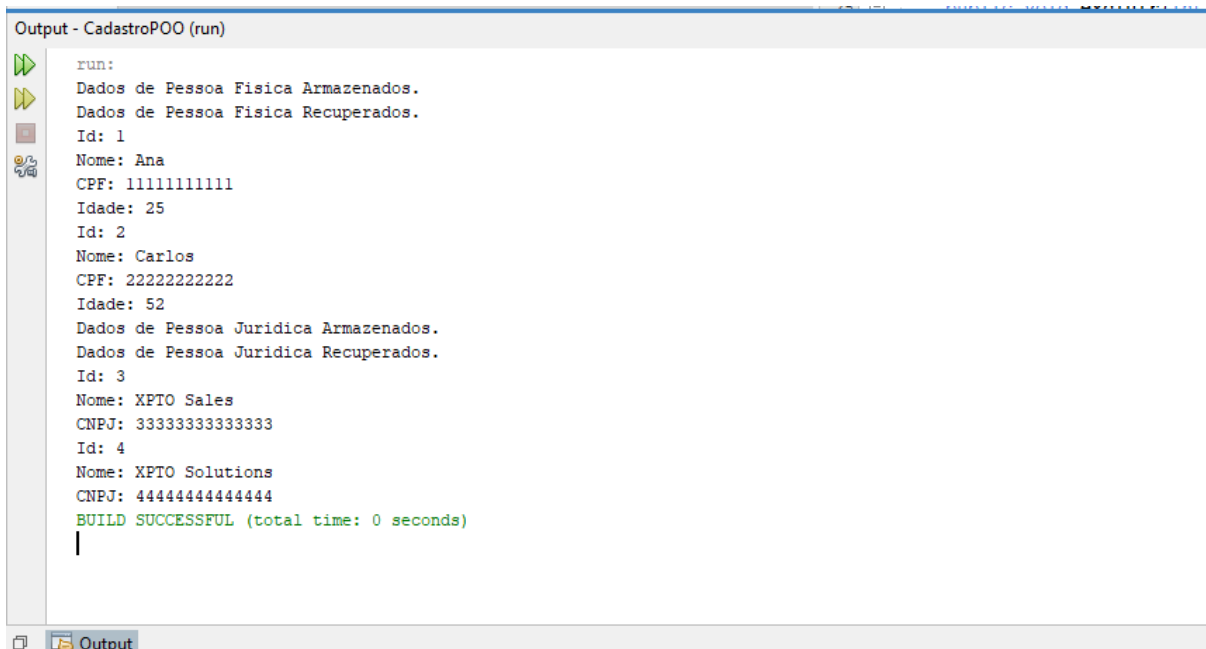
public List<PessoaJuridica> obterTodos() {
    return new ArrayList<>(pessoasJuridicas);
}

public void persistir(String nomeArquivo) throws IOException {
    try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
        out.writeObject(pessoasJuridicas);
    }
}

@SuppressWarnings("unchecked")
public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException
{
    try (ObjectInputStream in = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
        pessoasJuridicas = (ArrayList<PessoaJuridica>) in.readObject();
    }
}
}

```

## Resultados da Execução



```
Output - CadastroPOO (run)
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: 44444444444444
BUILD SUCCESSFUL (total time: 0 seconds)
|
```

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: 44444444444444  
BUILD SUCCESSFUL (total time: 0 seconds)



## Análise e Conclusão

- **Vantagens e Desvantagens do Uso de Herança:** A herança facilita a reutilização de código e melhora a organização do mesmo ao estabelecer uma hierarquia clara de classes. Contudo, pode levar a uma estrutura rígida e frágil, dificultando alterações futuras e a compreensão do fluxo de execução em sistemas complexos.
- **Interface Serializable:** Esta interface é necessária para a persistência em arquivos binários, pois habilita a capacidade de um objeto ser transformado em uma sequência de bytes e ser posteriormente reconstruído sem perder as informações de estado.
- **Paradigma Funcional na API Stream:** O paradigma funcional é utilizado na API Stream do Java para permitir operações de processamento de coleções de forma concisa e expressiva, apoiando-se em expressões lambda, operações de alto nível e processamento paralelo.
- **Padrão de Desenvolvimento em Persistência de Dados:** Na comunidade Java é comum o uso da serialização como meio de armazenamento de dados para recuperação futura, o que envolve a conversão de objetos em um formato que possa ser facilmente gerenciado e recuperado quando necessário, possibilitando assim o armazenamento até mesmo de estruturas de dados complexas.

## Github do projeto:

<https://github.com/Navesz/Iniciando-o-caminho-pelo-Java>