



# Estácio

**Campus:** Interlagos

**Curso:** Desenvolvimento Full Stack (2024.3)

**Disciplina:** RPG0014 - Iniciando o caminho pelo Java

**Turma:** EAD

**Integrantes:** 202303832338 - Rafael Rosendo Tagliaferro

github: <https://github.com/minoscorpion/est006/tree/main/proc1/CadastroPOO>

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

Implementação de um cadastro de clientes em modo texto, com persistência em arquivos, baseado na tecnologia Java.

### **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

## **Conclusão**

### **- Quais as vantagens e desvantagens do uso de herança?**

Uma das vantagens seria o reaproveitamento do código e a organização das classes nos projetos, baseando no principal da orientação ao objeto, de ter sempre características ligadas as classes criadas, evitando dessa forma, a criação de objetos duplicados de maneira desnecessária.

A Maior desvantagem que vejo, é a forte dependencia de suas subclasses, então, qualquer alteração feita na classe, precisa ser repassada as outras classes filhas.

### **- Por que a interface Serializable é necessária ao efetuar persistência em arquivos binários?**

Para converter a classe em um formato que o JVM possa grava-lo e recupera-lo novamente, no caso em bytes, também fica com uma marcação para que o objeto possa ser recuperado em qualquer ponto.

### **- Como o paradigma funcional é utilizado pela API stream no Java?**

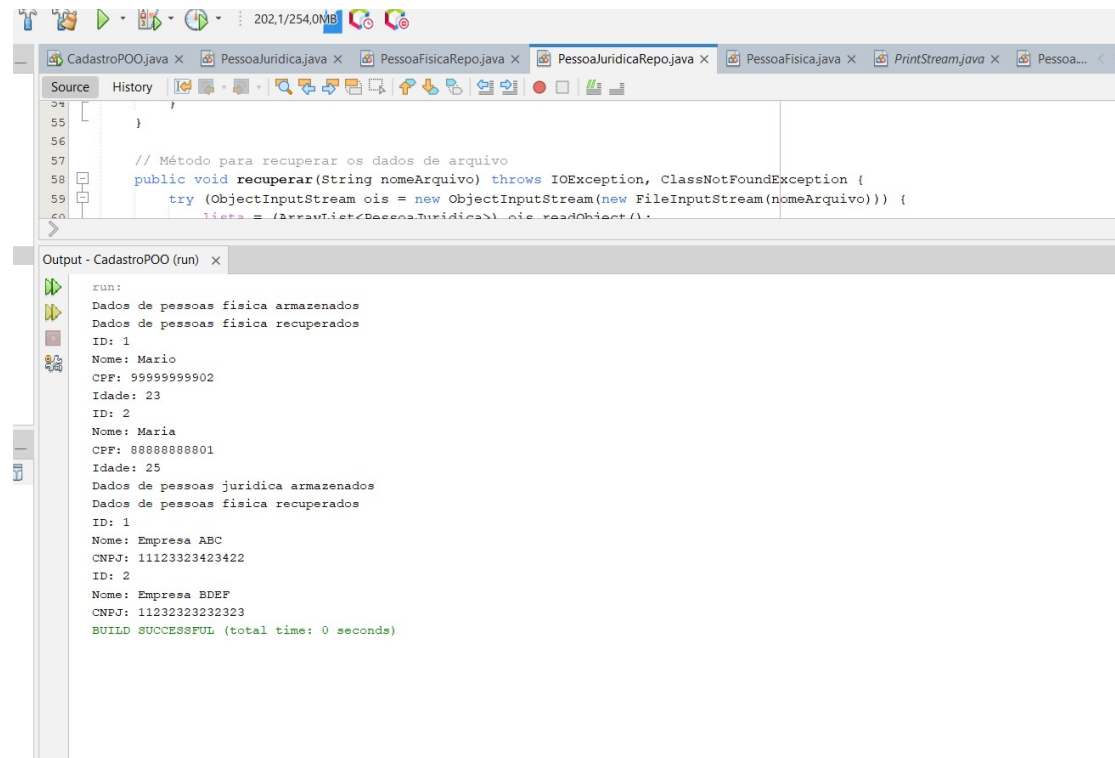
Basicamente torna muito mais facil a utilização para tratar os objetos, utilizando sintaxes mais simples como lambda ou linq, ou ate mesmo se utilizar de paralelismo.

### **- Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na**

**persistência de dados em arquivos?.**

utilizamos normalmente o DAO

## RESULTADO DO CÓDIGO



The screenshot shows an IDE with several tabs open: CadastroPOO.java, PessoaJuridica.java, PessoaFisicaRepo.java, PessoaJuridicaRepo.java, PessoaFisica.java, PrintStream.java, and Pessoa... The active tab is PessoaJuridicaRepo.java, showing a method `recuperar` that reads data from a file. Below the code editor, the 'Output - CadastroPOO (run)' window displays the execution results.

```
run:
Dados de pessoas fisica armazenados
Dados de pessoas fisica recuperados
ID: 1
Nome: Mario
CPF: 99999999902
Idade: 23
ID: 2
Nome: Maria
CPF: 88888888801
Idade: 25
Dados de pessoas juridica armazenados
Dados de pessoas fisica recuperados
ID: 1
Nome: Empresa ABC
CNPJ: 111232323423422
ID: 2
Nome: Empresa BDEF
CNPJ: 11232323232323
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Códigos do projeto:

### Main

```
package cadastrapoo;

import model.PessoaFisica;
import model.PessoaJuridica;
import model.PessoaFisicaRepo;
import model.PessoaJuridicaRepo;
import java.io.IOException;

public class CadastroPOO {

    public static void main(String[] args) {
```

```

try {

    /* PF */

    PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();

    repoFisica.inserir(new PessoaFisica(1, "Mario", "99999999902", 23));

    repoFisica.inserir(new PessoaFisica(2, "Maria", "88888888801", 25));

    System.out.println("Dados de pessoas fisica armazenados");

    repoFisica.persistir("pessoas_fisicas.dat");

    System.out.println("Dados de pessoas fisica recuperados");

    repoFisica.recuperar("pessoas_fisicas.dat");


    for (PessoaFisica pf : repoFisica.obterTodos()) pf.exibir();

    /* PJ */

    PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();

    repoJuridica.inserir(new PessoaJuridica(1, "Empresa ABC", "11123323423422"));

    repoJuridica.inserir(new PessoaJuridica(2, "Empresa BDEF", "11232323232323"));

    repoJuridica.persistir("pessoas_juridicas.dat");

    System.out.println("Dados de pessoas juridica armazenados");

    repoJuridica.recuperar("pessoas_juridicas.dat");

    System.out.println("Dados de pessoas fisica recuperados");

    for (PessoaJuridica pj : repoJuridica.obterTodos()) pj.exibir();

} catch (IOException | ClassNotFoundException e) {

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

}

}

}

```

## Pessoa.Java

```

/*
 * 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 rafaee
 */
public class Pessoa implements Serializable {

    private static final long serialVersionUID = 1L;

    private int id;

    public String nome;

    public Pessoa() {}

    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;

    }

    public void exibir() {

        System.out.println("ID: " + id);

        System.out.println("Nome: " + nome);

    }

}

```

### **PessoaFisica.Java**

```

/*

 * 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 rafaee

 */

```

```
// Classe PessoaFisica

public class PessoaFisica extends Pessoa implements Serializable {

    private static final long serialVersionUID = 1L;

    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;

    }

    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);

    }

}

```

## PessoaFisicaRepo.java

```

/*

 * 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;

import java.util.Optional;

/**

 *

```



```

* @author rafae
*/

// Classe PessoaFisicaRepo

public class PessoaFisicaRepo {

    private List<PessoaFisica> lista = new ArrayList<>();

    // Método para inserir uma nova pessoa

    public void inserir(PessoaFisica pessoa) {

        lista.add(pessoa);

    }

    // Método para alterar uma pessoa existente

    public void alterar(PessoaFisica pessoa) {

        Optional<PessoaFisica> existente = lista.stream()

            .filter(p -> p.getId() == pessoa.getId())

            .findFirst();

        existente.ifPresent(p -> p.setNome(pessoa.getNome()));

    }

    // Método para excluir uma pessoa por id

    public void excluir(int id) {

        lista.removeIf(p -> p.getId() == id);

    }

    // Método para obter uma pessoa por id

    public PessoaFisica obter(int id) {

```

```

        return lista.stream()

            .filter(p -> p.getId() == id)

            .findFirst()

            .orElse(null);
    }

    // Método para obter todas as pessoas
    public List<PessoaFisica> obterTodos() {

        return lista;
    }

    // Método para persistir os dados em arquivo
    public void persistir(String nomeArquivo) throws IOException {

        try (ObjectOutputStream oos = new ObjectOutputStream(new
        FileOutputStream(nomeArquivo))) {

            oos.writeObject(lista);

        }

    }

    // Método para recuperar os dados de arquivo
    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {

        try (ObjectInputStream ois = new ObjectInputStream(new
        FileInputStream(nomeArquivo))) {

            lista = (List<PessoaFisica>) ois.readObject();

        }

    }

}

```

## PessoaJuridica.Java

```
/*  
  
 * 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 rafae  
 */  
  
public class PessoaJuridica extends Pessoa implements Serializable {  
  
    private static final long serialVersionUID = 1L;  
  
    private String cnpj;  
  
  
    // Construtor padrão  
  
    public PessoaJuridica() {  
  
    }  
  
  
    // Construtor completo  
  
    public PessoaJuridica(int id, String nome, String cnpj) {  
  
        super(id, nome);  
  
        this.cnpj = cnpj;  
  
    }  
}
```

```

// Getters e Setters

public String getCnpj() {

    return cnpj;

}

public void setCnpj(String cnpj) {

    this.cnpj = cnpj;

}

// Método exibir polimórfico

@Override

public void exibir() {

    super.exibir(); // Chama o método da classe pai

    System.out.println("CNPJ: " + cnpj);

}

}

```

## PessoaJuridicaRepo.java

```

/*

* 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;

import java.util.Optional;

/**
 *
 * @author rafae
 */
// Classe PessoaJuridicaRepo
public class PessoaJuridicaRepo {

    private List<PessoaJuridica> lista = new ArrayList<>();

    // Método para inserir uma nova pessoa jurídica
    public void inserir(PessoaJuridica pessoa) {

        lista.add(pessoa);

    }

    // Método para alterar uma pessoa jurídica existente
    public void alterar(PessoaJuridica pessoa) {

        Optional<PessoaJuridica> existente = lista.stream()

            .filter(p -> p.getId() == pessoa.getId())

            .findFirst();

        existente.ifPresent(p -> p.setNome(pessoa.getNome()));

    }

    // Método para excluir uma pessoa jurídica por id
    public void excluir(int id) {

```

```
        lista.removeIf(p -> p.getId() == id);  
    }  
}
```

```
// Método para obter uma pessoa jurídica por id
```

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

```
// Método para obter todas as pessoas jurídicas
```

```
public List<PessoaJuridica> obterTodos() {  
    return lista;  
}
```

```
// Método para persistir os dados em arquivo
```

```
public void persistir(String nomeArquivo) throws IOException {  
    try (ObjectOutputStream oos = new ObjectOutputStream(new  
        FileOutputStream(nomeArquivo))) {  
        oos.writeObject(lista);  
    }  
}
```

```
// Método para recuperar os dados de arquivo
```

```
public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {  
    try (ObjectInputStream ois = new ObjectInputStream(new
```

```
FileInputStream(nomeArquivo))) {  
    lista = (ArrayList<PessoaJuridica>) ois.readObject();  
}  
}  
}
```