



# 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

Criar uma interface de cadastro em modo texto que permita ao usuário manipular cadastros de pessoas físicas e jurídicas, com opções de incluir, alterar, excluir, listar e salvar/recuperar dados utilizando arquivos binários.

### 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 cadastrapoo;

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

public class CadastroPOO {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
        PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();
        int opcao;

        do {
            System.out.println("\n==== MENU CADASTRO =====");
            System.out.println("1 - Incluir");
            System.out.println("2 - Alterar");
            System.out.println("3 - Excluir");
            System.out.println("4 - Exibir pelo ID");
            System.out.println("5 - Exibir todos");
            System.out.println("6 - Salvar dados");
            System.out.println("7 - Recuperar dados");
            System.out.println("0 - Sair");
            System.out.print("Escolha uma opção: ");
            opcao = scanner.nextInt();
            scanner.nextLine();

            switch (opcao) {
                case 1 -> {
                    System.out.print("Tipo (F = Física, J = Jurídica): ");
                    String tipo = scanner.nextLine();
                    if (tipo.equalsIgnoreCase("F")) {
                        System.out.print("ID: "); int id = scanner.nextInt(); scanner.nextLine();
                        System.out.print("Nome: "); String nome = scanner.nextLine();
                        System.out.print("CPF: "); String cpf = scanner.nextLine();
                    }
                }
            }
        } while (opcao != 0);
    }
}
```

```

        System.out.print("Idade: "); int idade = scanner.nextInt();
scanner.nextLine();
        repoFisica.inserir(new PessoaFisica(id, nome, cpf, idade));
    } else if (tipo.equalsIgnoreCase("J")) {
        System.out.print("ID: "); int id = scanner.nextInt(); scanner.nextLine();
        System.out.print("Nome: "); String nome = scanner.nextLine();
        System.out.print("CNPJ: "); String cnpj = scanner.nextLine();
        repoJuridica.inserir(new PessoaJuridica(id, nome, cnpj));
    }
}

case 2 -> {
    System.out.print("Tipo (F = Física, J = Jurídica): ");
    String tipo = scanner.nextLine();
    System.out.print("ID: "); int id = scanner.nextInt(); scanner.nextLine();
    if (tipo.equalsIgnoreCase("F")) {
        PessoaFisica pf = repoFisica.obter(id);
        if (pf != null) {
            pf.exibir();
            System.out.print("Novo nome: "); String nome = scanner.nextLine();
            System.out.print("Novo CPF: "); String cpf = scanner.nextLine();
            System.out.print("Nova idade: "); int idade = scanner.nextInt();
scanner.nextLine();
            repoFisica.alterar(new PessoaFisica(id, nome, cpf, idade));
        } else System.out.println("Pessoa Física não encontrada.");
    } else if (tipo.equalsIgnoreCase("J")) {
        PessoaJuridica pj = repoJuridica.obter(id);
        if (pj != null) {
            pj.exibir();
            System.out.print("Novo nome: "); String nome = scanner.nextLine();
            System.out.print("Novo CNPJ: "); String cnpj = scanner.nextLine();
            repoJuridica.alterar(new PessoaJuridica(id, nome, cnpj));
        } else System.out.println("Pessoa Jurídica não encontrada.");
    }
}

case 3 -> {
    System.out.print("Tipo (F = Física, J = Jurídica): ");
    String tipo = scanner.nextLine();
    System.out.print("ID a excluir: "); int id = scanner.nextInt();
scanner.nextLine();
    if (tipo.equalsIgnoreCase("F")) repoFisica.excluir(id);

```

```

else if (tipo.equalsIgnoreCase("J")) repoJuridica.excluir(id);
}
case 4 -> {
System.out.print("Tipo (F = Física, J = Jurídica): ");
String tipo = scanner.nextLine();
System.out.print("ID: "); int id = scanner.nextInt(); scanner.nextLine();
if (tipo.equalsIgnoreCase("F")) {
PessoaFisica pf = repoFisica.obter(id);
if (pf != null) pf.exibir();
else System.out.println("Pessoa Física não encontrada.");
} else if (tipo.equalsIgnoreCase("J")) {
PessoaJuridica pj = repoJuridica.obter(id);
if (pj != null) pj.exibir();
else System.out.println("Pessoa Jurídica não encontrada.");
}
}
case 5 -> {
System.out.print("Tipo (F = Física, J = Jurídica): ");
String tipo = scanner.nextLine();
if (tipo.equalsIgnoreCase("F")) {
for (PessoaFisica pf : repoFisica.obterTodos()) pf.exibir();
} else if (tipo.equalsIgnoreCase("J")) {
for (PessoaJuridica pj : repoJuridica.obterTodos()) pj.exibir();
}
}
case 6 -> {
System.out.print("Prefixo do arquivo: ");
String prefixo = scanner.nextLine();
try {
repoFisica.persistir(prefixo + ".fisica.bin");
repoJuridica.persistir(prefixo + ".juridica.bin");
System.out.println("Dados salvos com sucesso.");
} catch (Exception e) {
System.out.println("Erro ao salvar os dados: " + e.getMessage());
}
}
case 7 -> {
System.out.print("Prefixo do arquivo: ");
String prefixo = scanner.nextLine();
try {

```

```

        repoFisica.recuperar(prefixo + ".fisica.bin");
        repoJuridica.recuperar(prefixo + ".juridica.bin");
        System.out.println("Dados recuperados com sucesso.");
    } catch (Exception e) {
        System.out.println("Erro ao recuperar os dados: " + e.getMessage());
    }
}
case 0 -> System.out.println("Finalizando...");
default -> System.out.println("Opção inválida!");
}
} while (opcao != 0);

scanner.close();
}
}

```

## 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;

    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;

import java.io.Serializable;

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;
    }
}

```

```

@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;

import java.io.Serializable;

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

    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;
import java.util.List;

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

    public void inserir(PessoaFisica pessoa) {
        if (obter(pessoa.getId()) != null) {
            System.out.println("Pessoa Física com esse ID já existe.");
            return;
        }
        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;
            }
        }
        System.out.println("Pessoa Física não encontrada para alteração.");
    }

    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 List<PessoaFisica> obterTodos() {
        return pessoas;
    }

    public void persistir(String nomeArquivo) {
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
            oos.writeObject(pessoas);
        } catch (IOException e) {
            System.out.println("Erro ao salvar Pessoa Física: " + e.getMessage());
        }
    }

    public void recuperar(String nomeArquivo) {
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
            pessoas = (List<PessoaFisica>) ois.readObject();
        } catch (IOException | ClassNotFoundException e) {

```

```

        System.out.println("Erro ao recuperar Pessoa Física: " + e.getMessage());
    }
}
}

```

## PessoaJuridicaRepo.java:

```
package model;
```

```
import java.io.*;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```

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

    public void inserir(PessoaJuridica pessoa) {
        if (obter(pessoa.getId()) != null) {
            System.out.println("Pessoa Jurídica com esse ID já existe.");
            return;
        }
        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;
            }
        }
        System.out.println("Pessoa Jurídica não encontrada para alteração.");
    }

    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;
    }

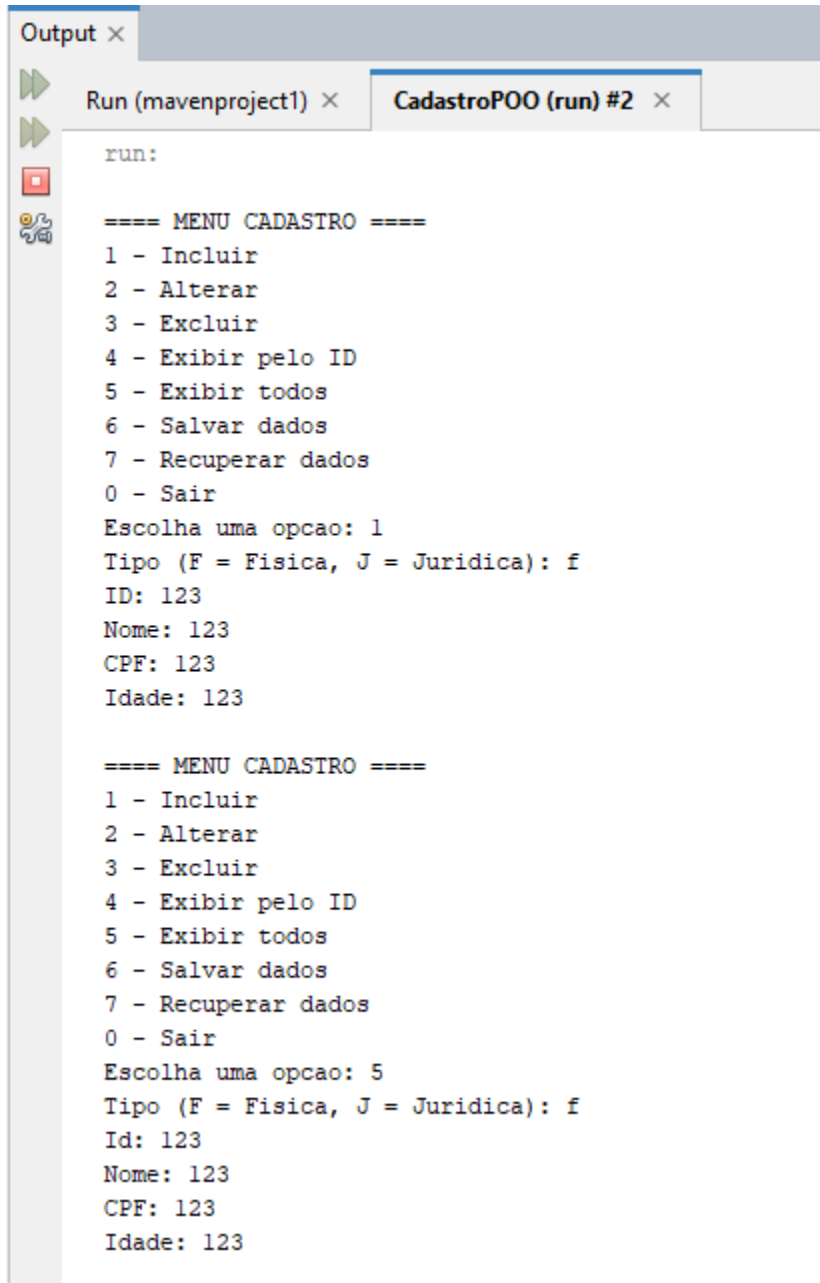
    public List<PessoaJuridica> obterTodos() {
        return pessoas;
    }

    public void persistir(String nomeArquivo) {
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
            oos.writeObject(pessoas);
        } catch (IOException e) {
            System.out.println("Erro ao salvar Pessoa Jurídica: " + e.getMessage());
        }
    }

    public void recuperar(String nomeArquivo) {
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
            pessoas = (List<PessoaJuridica>) ois.readObject();
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao recuperar Pessoa Jurídica: " + e.getMessage());
        }
    }
}

```

## Resultados:



```
Output x
Run (mavenproject1) x  CadastroPOO (run) #2 x
run:
==== MENU CADASTRO ====
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Sair
Escolha uma opcao: 1
Tipo (F = Fisica, J = Juridica): f
ID: 123
Nome: 123
CPF: 123
Idade: 123

==== MENU CADASTRO ====
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Sair
Escolha uma opcao: 5
Tipo (F = Fisica, J = Juridica): f
Id: 123
Nome: 123
CPF: 123
Idade: 123
```

O menu foi exibido corretamente no terminal, as opções de cadastro e manipulação funcionaram conforme esperado, e os arquivos .bin foram gerados e lidos com sucesso.

## Conclusão

1. O que são elementos estáticos e por que o método main adota esse modificador?

Elementos estáticos pertencem à classe e não a uma instância. O método main é static para poder ser chamado diretamente pela JVM sem precisar criar um objeto da classe.

2. Para que serve a classe Scanner?

A classe Scanner serve para ler dados digitados pelo usuário no console, como texto e números.

3. Como o uso de classes de repositório impactou na organização do código?

As classes de repositório ajudaram a separar a lógica de acesso e manipulação de dados, deixando o código mais limpo, modular e fácil de manter.