# Trabalho Prático | DGT2821 Programação Back- end com Java

**Curso:** Desenvolvimento Full Stack

**Polo:** Polo - Vila Nova - Itu - Sp

**Matrícula:** 202404575268

**Aluno:** Rafael Lima de Medeiros

**Turma:** 2025.4 (5º Semestre)

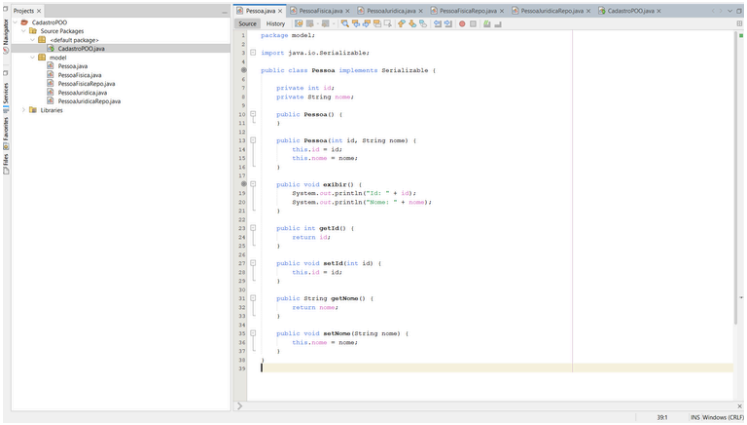
**Data:** 10/02/2026

**Github:** https://github.com/Rafaldm/Programa-o-Back-end-com-Java/tree/main

**Objetivo:** Demonstrar o uso de Programação Orientada a Objetos em Java, utilizando herança, polimorfismo e persistência de dados em arquivos binários.

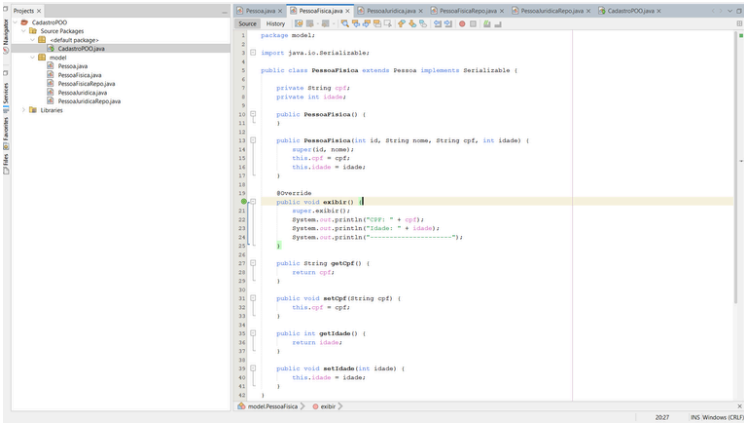
# 1º Procedimento | Criação das Entidades e Sistema de Persistência

## Pessoa.java



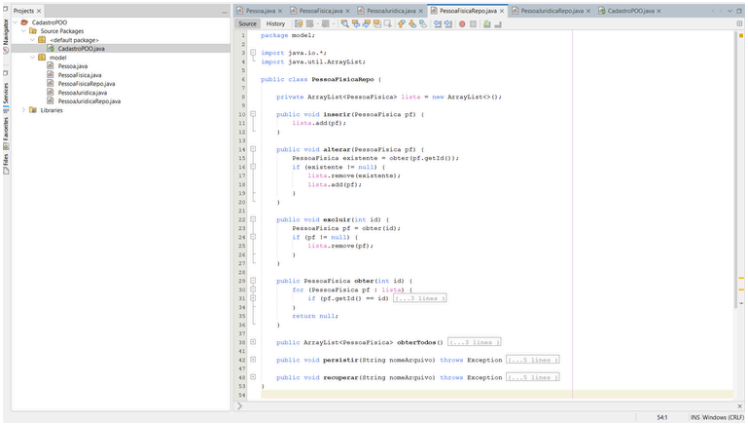
```
package model;

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



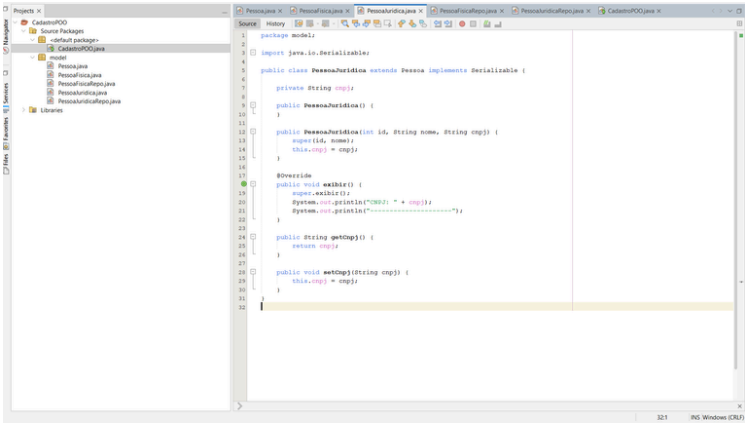
```
package model;

import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {

    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);
        System.out.println("--------------------");
    }

    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;
    }
}
```

## PessoaFisicaRepo.java



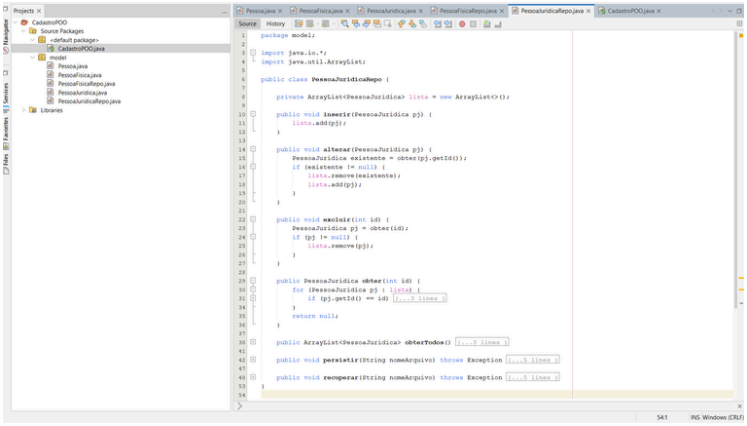
```
package model;

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

public class PessoaFisicaRepo {

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

    public void inserir(PessoaFisica pf) {
        lista.add(pf);
    }

    public void alterar(PessoaFisica pf) {
        PessoaFisica existente = obter(pf.getId());
        if (existente != null) {
            lista.remove(existente);
            lista.add(pf);
        }
    }

    public void excluir(int id) {
        PessoaFisica pf = obter(id);
        if (pf != null) {
            lista.remove(pf);
        }
    }

    public PessoaFisica obter(int id) {
        for (PessoaFisica pf : lista) {
            if (pf.getId() == id) {...3 lines }
        }
        return null;
    }

    public ArrayList<PessoaFisica> obterTodos() {...3 lines }

    public void persistir(String nomeArquivo) throws Exception {...5 lines }

    public void recuperar(String nomeArquivo) throws Exception {...5 lines }
}
```

## PessoaJuridica.java



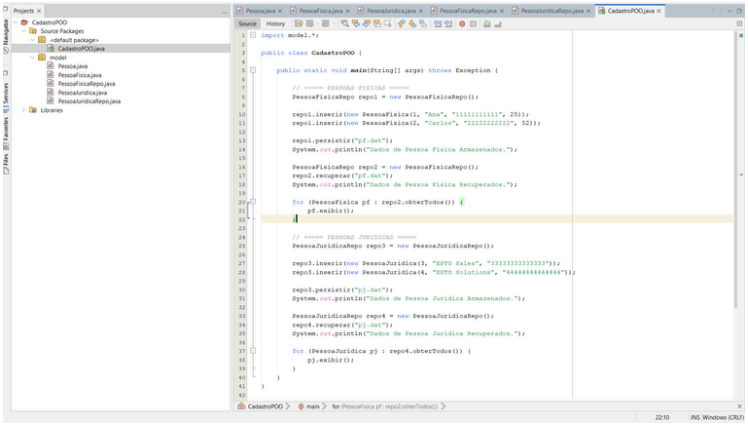
```
package model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {

    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);
        System.out.println("--------------------");
    }

    public String getCnpj() {
        return cnpj;
    }

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

## PessoaJuridicaRepo.java



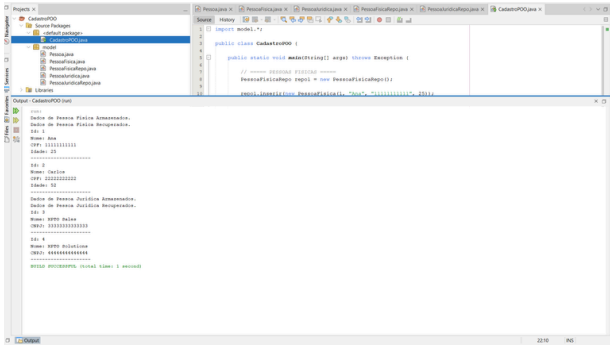
```
package model;

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

public class PessoaJuridicaRepo {

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

    public void inserir(PessoaJuridica pj) {
        lista.add(pj);
    }

    public void alterar(PessoaJuridica pj) {
        PessoaJuridica existente = obter(pj.getId());
        if (existente != null) {
            lista.remove(existente);
            lista.add(pj);
        }
    }

    public void excluir(int id) {
        PessoaJuridica pj = obter(id);
        if (pj != null) {
            lista.remove(pj);
        }
    }

    public PessoaJuridica obter(int id) {
        for (PessoaJuridica pj : lista) {
            if (pj.getId() == id) {...3 lines }
        }
        return null;
    }

    public ArrayList<PessoaJuridica> obterTodos() {...3 lines }

    public void persistir(String nomeArquivo) throws Exception {...5 lines }

    public void recuperar(String nomeArquivo) throws Exception {...5 lines }
}
```

## CadastroPOO.java



```
import model.*;

public class CadastroPOO {

    public static void main(String[] args) throws Exception {

        // ===== PESSOAS FISICAS =====
        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

        repo1.inserir(new PessoaFisica(1, "Ana", "11111111111", 25));
        repo1.inserir(new PessoaFisica(2, "Carlos", "22222222222", 52));

        repo1.persistir("pf.dat");
        System.out.println("Dados de Pessoa Fisica Armazenados.");

        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
        repo2.recuperar("pf.dat");
        System.out.println("Dados de Pessoa Fisica Recuperados.");

        for (PessoaFisica pf : repo2.obterTodos()) {
            pf.exibir();
        }

        // ===== PESSOAS JURIDICAS =====
        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();

        repo3.inserir(new PessoaJuridica(3, "XPTO Sales", "33333333333333"));
        repo3.inserir(new PessoaJuridica(4, "XPTO Solutions", "44444444444444"));

        repo3.persistir("pj.dat");
        System.out.println("Dados de Pessoa Juridica Armazenados.");

        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
        repo4.recuperar("pj.dat");
        System.out.println("Dados de Pessoa Juridica Recuperados.");

        for (PessoaJuridica pj : repo4.obterTodos()) {
            pj.exibir();
        }
    }
}
```

# RESULTADO



```
====
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: 1 second)
```

# Ánalise e Conclusão

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

A herança traz várias vantagens, principalmente a reutilização de código. Com ela, é possível criar uma classe mais genérica e reaproveitar seus atributos e métodos em outras classes, evitando repetição. Além disso, a herança ajuda a organizar melhor o sistema e facilita o uso do polimorfismo, permitindo que métodos tenham comportamentos diferentes dependendo da classe.
Por outro lado, a herança também possui desvantagens. Uma delas é o alto acoplamento entre as classes, já que mudanças na classe pai podem afetar todas as classes filhas. Em projetos maiores, isso pode dificultar a manutenção e deixar o código mais complexo.

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

A interface Serializable é necessária porque ela permite que os objetos sejam transformados em uma sequência de bytes. Essa transformação é essencial para que os dados possam ser gravados em arquivos binários no disco.
Sem a implementação da interface Serializable, o Java não permite salvar ou recuperar objetos diretamente de arquivos, pois não sabe como converter esses objetos para um formato que possa ser armazenado.

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

O paradigma funcional é utilizado na API Stream principalmente por meio do uso de expressões lambda. Com a Stream API, é possível trabalhar com coleções de dados de forma mais simples e direta, utilizando operações como filter, map e forEach.
Esse tipo de abordagem torna o código mais limpo, fácil de ler e reduz a necessidade de laços repetitivos, como o uso excessivo de for ou while.

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

Na persistência de dados em arquivos, geralmente é adotado o padrão DAO (Data Access Object). Esse padrão separa a lógica de acesso aos dados da lógica principal do sistema.
No projeto desenvolvido, as classes de repositório são responsáveis por inserir, alterar, excluir, salvar e recuperar os dados (Crud), deixando as classes de entidade focadas apenas na representação das informações.

# Códigos

- **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;

import java.io.Serializable;

public class PessoaFisica extends Pessoa implements
Serializable {

    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);
        System.out.println("----------------------");
    }

    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;
    }
}
```

- **PessoaFisicaRepo.java**

```
package model;

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

public class PessoaFisicaRepo {

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

   public void inserir(PessoaFisica pf) {
      lista.add(pf);
   }

   public void alterar(PessoaFisica pf) {
      PessoaFisica existente = obter(pf.getId());
      if (existente != null) {
         lista.remove(existente);
         lista.add(pf);
      }
   }

   public void excluir(int id) {
      PessoaFisica pf = obter(id);
      if (pf != null) {
         lista.remove(pf);
      }
   }

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

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

   public void persistir(String nomeArquivo) throws Exception {
      ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(nomeArquivo));
      out.writeObject(lista);
      out.close();
   }

   public void recuperar(String nomeArquivo) throws Exception
{
      ObjectInputStream in = new ObjectInputStream(new
FileInputStream(nomeArquivo));
      lista = (ArrayList<PessoaFisica>) in.readObject();
      in.close();
   }
}
```

- **PessoaJuridica.java**

```
package model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements
Serializable {

   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);
      System.out.println("----------------------");
   }

   public String getCnpj() {
      return cnpj;
   }

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

- **PessoaJuridicaRepo.java**

```
package model;

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

public class PessoaJuridicaRepo {

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

   public void inserir(PessoaJuridica pj) {
      lista.add(pj);
   }

   public void alterar(PessoaJuridica pj) {
      PessoaJuridica existente = obter(pj.getId());
      if (existente != null) {
         lista.remove(existente);
         lista.add(pj);
      }
   }

   public void excluir(int id) {
      PessoaJuridica pj = obter(id);
      if (pj != null) {
         lista.remove(pj);
      }
   }

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

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

   public void persistir(String nomeArquivo) throws Exception {
      ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(nomeArquivo));
      out.writeObject(lista);
      out.close();
   }

   public void recuperar(String nomeArquivo) throws Exception
{
      ObjectInputStream in = new ObjectInputStream(new
FileInputStream(nomeArquivo));
      lista = (ArrayList<PessoaJuridica>) in.readObject();
      in.close();
   }
}
```

- **CadastroPOO.java**

```
import model.*;

public class CadastroPOO {

   public static void main(String[] args) throws Exception
{

      // ===== PESSOAS FISICAS =====
      PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

      repo1.inserir(new PessoaFisica(1, "Ana",
"11111111111", 25));
      repo1.inserir(new PessoaFisica(2, "Carlos",
"22222222222", 52));

      repo1.persistir("pf.dat");
      System.out.println("Dados de Pessoa Fisica
Armazenados.");

      PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
      repo2.recuperar("pf.dat");
      System.out.println("Dados de Pessoa Fisica
Recuperados.");

      for (PessoaFisica pf : repo2.obterTodos()) {
         pf.exibir();
      }

      // ===== PESSOAS JURIDICAS =====
      PessoaJuridicaRepo repo3 = new
PessoaJuridicaRepo();

      repo3.inserir(new PessoaJuridica(3, "XPTO Sales",
"33333333333333"));
      repo3.inserir(new PessoaJuridica(4, "XPTO
Solutions", "44444444444444"));

      repo3.persistir("pj.dat");
      System.out.println("Dados de Pessoa Juridica
Armazenados.");

      PessoaJuridicaRepo repo4 = new
PessoaJuridicaRepo();
      repo4.recuperar("pj.dat");
      System.out.println("Dados de Pessoa Juridica
Recuperados.");

      for (PessoaJuridica pj : repo4.obterTodos()) {
         pj.exibir();
      }
   }
}
```

# Resultado

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)

# 2º Procedimento | Criação do Cadastro em Modo Texto

## Parte 1

```
import java.util.Scanner;
import model.*;

public class CadastroPOO {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();

int opcao = -1;

while (opcao != 0) {

System.out.println("====================================");
System.out.println("1 - Incluir Pessoa");
System.out.println("2 - Alterar Pessoa");
System.out.println("3 - Excluir Pessoa");
System.out.println("4 - Buscar pelo Id");
System.out.println("5 - Exibir Todos");
System.out.println("6 - Persistir Dados");
System.out.println("7 - Recuperar Dados");
System.out.println("0 - Finalizar Programa");
System.out.println("====================================");

opcao = sc.nextInt();
sc.nextLine();

switch (opcao) {

case 1: // INCLUIR
System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
String tipo = sc.nextLine();

System.out.println("Digite o id da pessoa:");
int id = sc.nextInt();
sc.nextLine();

System.out.println("Insira os dados...");

System.out.print("Nome: ");
String nome = sc.nextLine();

if (tipo.equalsIgnoreCase("F")) {

System.out.print("CPF: ");
String cpf = sc.nextLine();

System.out.print("Idade: ");
int idade = sc.nextInt();
sc.nextLine();

repoFisica.inserir(new PessoaFisica(id, nome, cpf, idade));

} else {

System.out.print("CNPJ: ");
String cnpj = sc.nextLine();

repoJuridica.inserir(new PessoaJuridica(id, nome, cnpj));
}

break;
```

## Parte 2

```
case 2: // ALTERAR
System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
tipo = sc.nextLine();

System.out.println("Digite o id da pessoa:");
int idAlt = sc.nextInt();
sc.nextLine();

if (tipo.equalsIgnoreCase("F")) {

PessoaFisica pf = repoFisica.obter(idAlt);

if (pf != null) {
pf.exibir();

System.out.println("Digite os novos dados...");

System.out.print("Nome: ");
nome = sc.nextLine();

System.out.print("CPF: ");
String cpf = sc.nextLine();

System.out.print("Idade: ");
int idade = sc.nextInt();
sc.nextLine();

repoFisica.alterar(new PessoaFisica(idAlt, nome, cpf, idade));
} else {
System.out.println("Pessoa nao encontrada.");
}

} else {

PessoaJuridica pj = repoJuridica.obter(idAlt);

if (pj != null) {
pj.exibir();

System.out.println("Digite os novos dados...");

System.out.print("Nome: ");
nome = sc.nextLine();

System.out.print("CNPJ: ");
String cnpj = sc.nextLine();

repoJuridica.alterar(new PessoaJuridica(idAlt, nome, cnpj));
} else {
System.out.println("Pessoa nao encontrada.");
}
}

break;

case 3: // EXCLUIR
System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
tipo = sc.nextLine();

System.out.println("Digite o id da pessoa:");
int idExc = sc.nextInt();
sc.nextLine();
```

# Parte 3

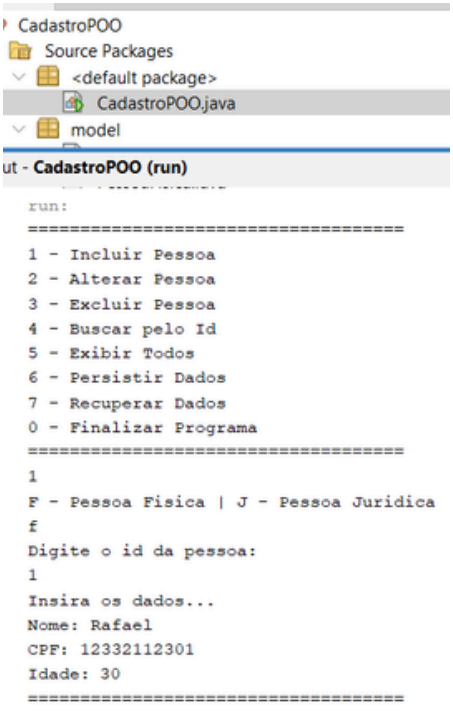
```
if (tipo.equalsIgnoreCase("F"))
repoFisica.excluir(idExc);
else
repoJuridica.excluir(idExc);

break;

case 4: // BUSCAR
System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
tipo = sc.nextLine();

System.out.println("Digite o id da pessoa:");
int idBus = sc.nextInt();
sc.nextLine();

if (tipo.equalsIgnoreCase("F")) {
PessoaFisica pf = repoFisica.obter(idBus);
if (pf != null)
pf.exibir();
else
System.out.println("Pessoa nao encontrada.");
} else {
PessoaJuridica pj = repoJuridica.obter(idBus);
if (pj != null)
pj.exibir();
else
System.out.println("Pessoa nao encontrada.");
}

break;

case 5: // EXIBIR TODOS
System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
tipo = sc.nextLine();

if (tipo.equalsIgnoreCase("F")) {
for (PessoaFisica pf : repoFisica.obterTodos())
pf.exibir();
} else {
for (PessoaJuridica pj : repoJuridica.obterTodos())
pj.exibir();
}

break;

case 6: // PERSISTIR
try {
System.out.println("Digite o prefixo do arquivo:");
String prefixo = sc.nextLine();

repoFisica.persistir(prefixo + ".fisica.bin");
repoJuridica.persistir(prefixo + ".juridica.bin");

System.out.println("Dados persistidos com sucesso.");

} catch (Exception e) {
System.out.println("Erro ao persistir dados.");
}
break;
```

# Parte 4

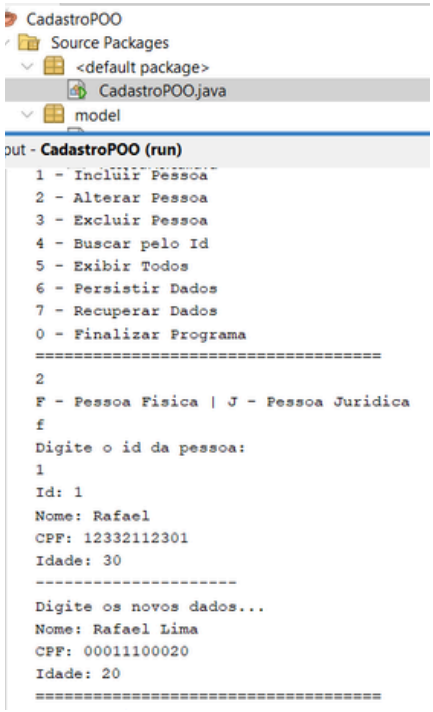
```
case 7: // RECUPERAR
try {
System.out.println("Digite o prefixo do arquivo:");
String prefixo = sc.nextLine();

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 dados.");
}
break;

case 0:
System.out.println("Programa finalizado.");
break;

default:
System.out.println("Opcao invalida.");
}
}

sc.close();
}
}
```

# Resultado

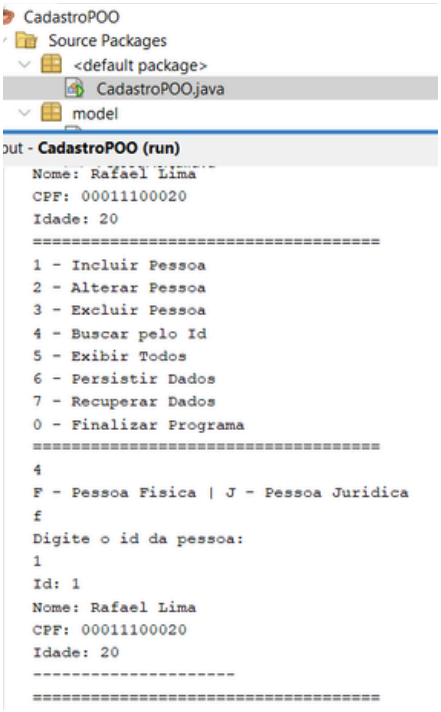
## Incluir PF



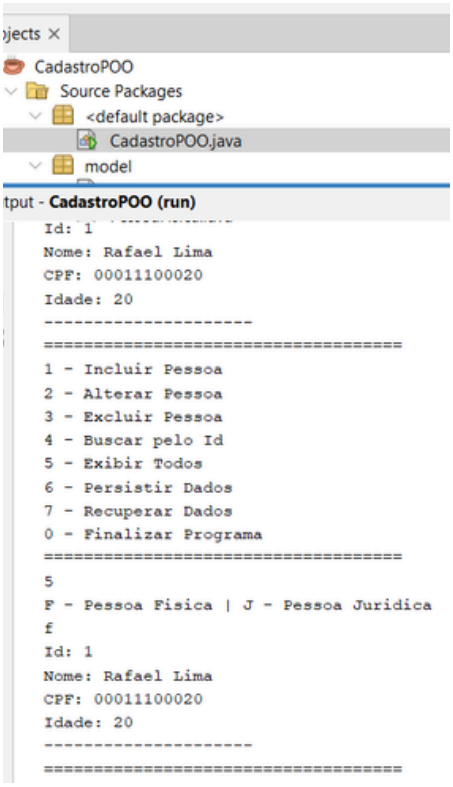

## Alterar




## Buscar por ID

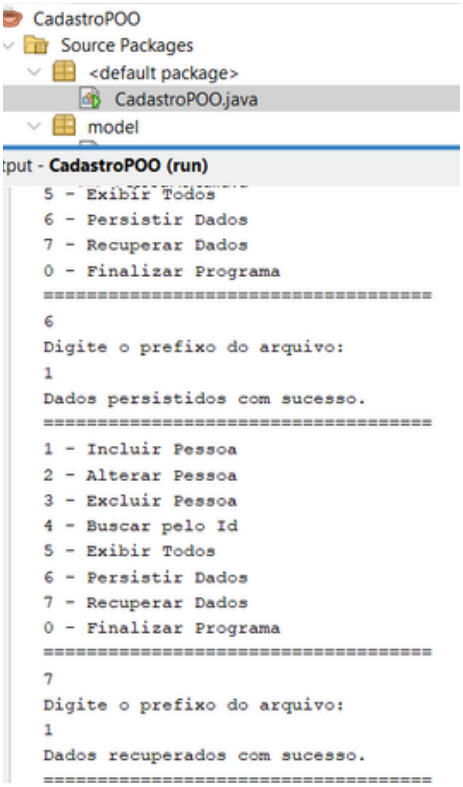



## Exibir todos




## Incluir PJ


## Persistir e Recuperar



# Ánalise e Conclusão

- **O que são elementos estáticos e por que o main é static?**

Elementos estáticos pertencem à classe e não a um objeto específico.
O método main é static porque o Java precisa executá-lo sem precisar criar um objeto da classe primeiro.

- **Para que serve a classe Scanner?**

A classe Scanner serve para ler dados digitados pelo usuário no teclado.
Ela permite capturar números, textos e outras informações durante a execução do programa.

- **Como o uso de classes de repositório impactou na organização do código?**

O uso das classes de repositório deixou o código mais organizado, separando a parte de manipulação de dados da parte principal do sistema.
Isso facilita manutenção, leitura e futuras melhorias no projeto.

# Ánalise Crítica da Missão Prática

Esta prática foi muito importante porque trabalhou vários conceitos fundamentais da Programação Orientada a Objetos ao mesmo tempo.
Primeiramente, foi possível aplicar herança e polimorfismo na criação das classes Pessoa, PessoaFisica e PessoaJuridica, o que reforça a organização e reutilização de código. Isso mostra na prática como estruturar um sistema de forma mais profissional.
Outro ponto relevante foi o uso da persistência em arquivos binários com Serializable. Isso aproximou o exercício de uma aplicação real, pois armazenar e recuperar dados é algo essencial em qualquer sistema.
A criação do menu em modo texto também ajudou a entender melhor o funcionamento do método main, da classe Scanner e do controle de fluxo com switch e loops. Foi necessário organizar bem o código para evitar confusão, principalmente separando a lógica nas classes de repositório, o que melhorou bastante a estrutura do projeto.
Como ponto de melhoria, o sistema ainda é simples e não possui validações mais avançadas ou interface gráfica. Porém, para fins de aprendizado, ele cumpre muito bem o objetivo de consolidar os conceitos básicos de orientação a objetos, persistência e organização de código.