



Estácio

Nome do Campus: BANCÁRIOS.

Nome do Curso: Desenvolvimento Full Stack.

Nome da Disciplina: Nível 1: Iniciando o Caminho Pelo Java.

Número da Turma: 202403781506.

Semestre letivo: 3º período - semestre 2025.1.

Nome dos integrantes da Prática: Marcelo Picado Schulze.

Repositório no GIT: <https://github.com/darklaw53/AtividadeJava.git>

Missão Prática | Nível 1 | Mundo 3

<https://github.com/darklaw53/AtividadeJava.git>

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 usados:

```
package cadastropoo;

import java.io.IOException;
import java.util.Scanner;
import model.PessoaFisica;
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;
```

```

public class CadastroPOO
{
    public static void main(String[] args)
    {
        try (Scanner scanner = new Scanner(System.in))
        {
            PessoaFisicaRepo repoPF = new PessoaFisicaRepo();
            PessoaJuridicaRepo repoPJ = new PessoaJuridicaRepo();

            while (true)
            {
                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("=====");

                int opcao = scanner.nextInt();
                scanner.nextLine();

                if (opcao == 0) break;

                char tipo = 'x';
                if (opcao != 6 && opcao != 7)
                {
                    do
                    {
                        System.out.print("F - Pessoa Física | J - Pessoa Jurídica\n");
                        tipo = scanner.next().toUpperCase().charAt(0);
                        scanner.nextLine();
                    }
                    while (tipo != 'F' && tipo != 'J');
                }

                switch (opcao)
                {
                    case 1 ->
                    {
                        System.out.print("Digite o id da pessoa:\n");
                        int id = scanner.nextInt();
                        scanner.nextLine();
                        System.out.print("Insira os dados...\n");
                        System.out.print("Nome:\n");
                        String nome = scanner.nextLine();

                        if (tipo == 'F')
                        {

```

```

        System.out.print("CPF:\n");
        String cpf = scanner.nextLine();
        System.out.print("Idade:\n");
        int idade = scanner.nextInt();
        scanner.nextLine();
        repoPF.inserir(new PessoaFisica(id, nome, cpf, idade));
    }
    else
    {
        System.out.print("CNPJ:\n");
        String cnpj = scanner.nextLine();
        repoPJ.inserir(new PessoaJuridica(id, nome, cnpj));
    }
}
case 2 ->
{
    System.out.print("ID:\n");
    int id = scanner.nextInt();
    scanner.nextLine();
    if (tipo == 'F')
    {
        PessoaFisica pf = repoPF.obter(id);
        if (pf != null)
        {
            System.out.println("Dados atuais:");
            System.out.println("Nome: " + pf.getNome());
            System.out.println("CPF: " + pf.getCpf());
            System.out.println("Idade: " + pf.getIdade());
            System.out.print("Novo nome:\n");
            pf.setNome(scanner.nextLine());
            System.out.print("Novo CPF:\n");
            pf.setCpf(scanner.nextLine());
            System.out.print("Nova idade:\n");
            pf.setIdade(scanner.nextInt());
            scanner.nextLine();
        }
    }
    else
    {
        PessoaJuridica pj = repoPJ.obter(id);
        if (pj != null)
        {
            System.out.println("Dados atuais:");
            System.out.println("Nome: " + pj.getNome());
            System.out.println("CNPJ: " + pj.getCnpj());
            System.out.print("Novo nome:\n");
            pj.setNome(scanner.nextLine());
            System.out.print("Novo CNPJ:\n");
            pj.setCnpj(scanner.nextLine());
        }
    }
}
}

```

```

case 3 ->
{
    System.out.print("ID:\n");
    int id = scanner.nextInt();
    scanner.nextLine();
    if (tipo == 'F') repoPF.excluir(id);
    else repoPJ.excluir(id);
}
case 4 ->
{
    System.out.print("ID:\n");
    int id = scanner.nextInt();
    scanner.nextLine();
    if (tipo == 'F') {
        PessoaFisica pf = repoPF.obter(id);
        if (pf != null) {
            System.out.println("Nome: " + pf.getNome());
            System.out.println("CPF: " + pf.getCpf());
            System.out.println("Idade: " + pf.getIdade());
        }
    } else {
        PessoaJuridica pj = repoPJ.obter(id);
        if (pj != null) {
            System.out.println("Nome: " + pj.getNome());
            System.out.println("CNPJ: " + pj.getCnpj());
        }
    }
}
case 5 ->
{
    if (tipo == 'F') repoPF.obterTodos().forEach(pf -> {
        System.out.println("-----");
        System.out.println("Nome: " + pf.getNome());
        System.out.println("CPF: " + pf.getCpf());
        System.out.println("Idade: " + pf.getIdade());
    });
    else repoPJ.obterTodos().forEach(pj -> {
        System.out.println("-----");
        System.out.println("Nome: " + pj.getNome());
        System.out.println("CNPJ: " + pj.getCnpj());
    });
}
case 6 ->
{
    try
    {
        System.out.print("Prefixo do arquivo:\n");
        String prefixo = scanner.nextLine();
        repoPF.persistir(prefixo + ".fisica.bin");
        repoPJ.persistir(prefixo + ".juridica.bin");
    }
    catch (IOException e)

```



```

public String getNome()
{
    return nome;
}

public void setNome(String nome)
{
    this.nome = nome;
}
}

```

Package model;

```

public class PessoaFisica extends Pessoa
{
    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;
    }
}

```

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)
    {
        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)
    {
        return pessoas.stream().filter(p → p.getId() == id).findFirst().orElse(null);
    }

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

    public void persistir(String arquivo) throws IOException
    {
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(arquivo)))
        {
            oos.writeObject(pessoas);
        }
    }

    @SuppressWarnings("unchecked")
    public void recuperar(String arquivo) throws IOException, ClassNotFoundException
    {

```

```
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(arquivo)))
        {
            pessoas = (List<PessoaFisica>) ois.readObject();
        }
    }
}
```

Package model;

```
public class PessoaJuridica extends Pessoa
{
    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;
    }

    public String getCnpj ()
    {
        return cnpj;
    }

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

Package model;

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

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

    public void inserir( PessoaJuridica empresa)
    {
        empresas.add(empresa);
    }

    public void alterar(PessoaJuridica empresa)
    {
        for(int i = 0; i < empresas.size(); i++)
        {
```



```

        if (empresas.get(i).getId() == empresa.getId())
        {
            empresas.set(i, empresa);
            return;
        }
    }
}

public void excluir (int id)
{
    empresas.removeIf(p → p.getId() == id);
}

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

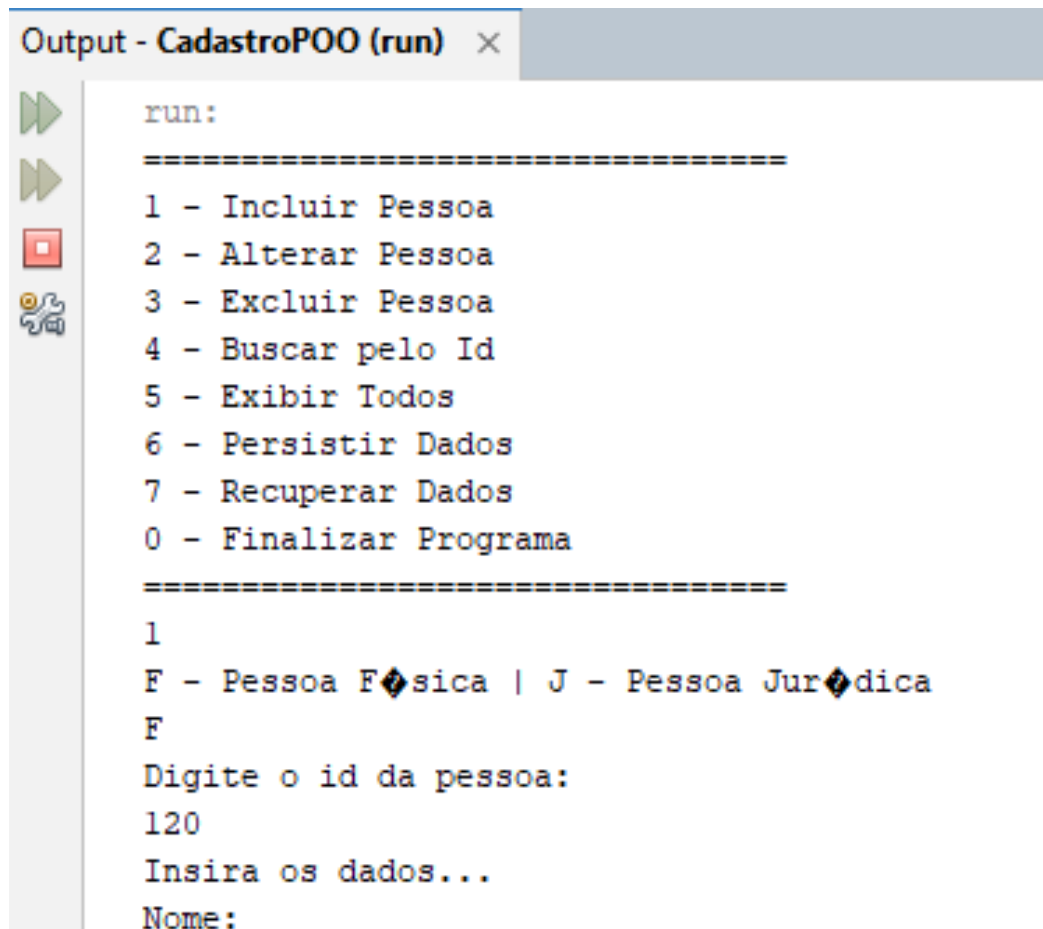
public List<PessoaJuridica> obterTodos()
{
    return new ArrayList<>(empresas);
}

public void persistir(String arquivo) throws IOException
{
    try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(arquivo)))
    {
        oos.writeObject(empresas);
    }
}

@SuppressWarnings("unchecked")
public void recuperar(String arquivo) throws IOException, ClassNotFoundException
{
    try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(arquivo)))
    {
        empresas = (List<PessoaJuridica>) ois.readObject();
    }
}
}

```

Output da execução:



```
Output - CadastroPOO (run) x
run:
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
1
F - Pessoa Física | J - Pessoa Jurídica
F
Digite o id da pessoa:
120
Insira os dados...
Nome:
```

Análise e Conclusão:

O que são elementos estáticos e qual o motivo para o método main adotar esse modificador?

- Os elementos estáticos são aqueles que são relacionados a classe e não uma instância da classe e são carregados apenas uma vez, a Java necessita que main seja estático por precisar de uma entrada fixa e acessível independente de instância.

Para que serve a classe Scanner?

- Ler a entrada de dados como a do teclado para poder digitar e preencher campos que o programa peça do usuário.

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

- Permitiu a **separação da lógica de dados** da lógica da aplicação para deixar o código limpo e estruturado.