



**UNIVERSIDADE ESTÁCIO**  
**CURSO DE DESENVOLVIMENTO FULL STACK**

**RELATÓRIO DA MISSÃO PRÁTICA 1 - MUNDO 3**

**TURMA: 2023.1**

**DISCIPLINA: RPG0014 - INICIANDO CAMINHO PELO JAVA**

**CAMPUS: VIA CORPVS - FORTALEZA (CE)**

**MOISÉS EDUARDO GOMES DA COSTA**

## 1º Procedimento - Sistema Cadastro de Pessoas com P00

1. Implementar um sistema de cadastro e gerenciamento de pessoas físicas e jurídicas utilizando a Programação Orientada a Objetos (POO) em Java.
2. O sistema deve ser capaz de inserir, alterar, excluir obter dados específicos e todos os dados de pessoas físicas e(ou) jurídicas.
3. O sistema deve ser capaz de alterar os dados das pessoas físicas e jurídicas.
4. O sistema deve ser capaz de persistir os dados em um arquivo.
5. O sistema deve ser capaz de recuperar os dados de um arquivo.
6. Fazer uso de herança e polimorfismo

<https://github.com/BeaterOfWar/CadastroPoo>

```
package cadastropoo;

import cadastropoo.model.*;

import java.io.IOException;

public class CadastroPOO {

    public static void main(String[] args) {

        try {

            PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
            repo1.inserir(new PessoaFisica(1, "Ana", "11111111111", 25));
            repo1.inserir(new PessoaFisica(2, "Carlos", "22222222222", 52));
            repo1.persistir("pessoasFisicas.dat");
            System.out.println("Dados de Pessoa Fisica Armazenados.");
            PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
            repo2.recuperar("pessoasFisicas.dat");
            System.out.println("Dados de Pessoa Fisica Recuperados.");
            for (PessoaFisica p : repo2.obterTodos()) {
                p.exibir();
            }

            PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
            repo3.inserir(new PessoaJuridica(3, "XPTO Sales", "3333333333333"));
```

```

        repo3.inserir(new PessoaJuridica(4, "XPTO Solutions", "44444444444444"));
        repo3.persistir("pessoasJuridicas.dat");

        System.out.println("Dados de Pessoa Juridica Armazenados.");

        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
        repo4.recuperar("pessoasJuridicas.dat");

        System.out.println("Dados de Pessoa Juridica Recuperados.");

        for (PessoaJuridica p : repo4.obterTodos()) {
            p.exibir();
        }
    } catch (IOException | ClassNotFoundException e) {
        e.printStackTrace();
    }
}
}

```

## Pessoa.java

```

package cadastrapoo.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 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 + "\nNome: " + nome);  
}  
}
```

## PessoaFisica.java

```
package cadastradopoo.model;  
  
import java.io.Serializable;  
  
public class PessoaFisica extends Pessoa implements Serializable {  
    private String cpf;  
    private int idade;  
  
    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 + "\nIdade: " + idade);
    }
}

```

## PessoaFisicaRepo.java

```

package cadastrpoo.model;
import java.io.*;
import java.util.ArrayList;

public class PessoaFisicaRepo implements Serializable {
    private ArrayList<PessoaFisica> pessoasFisicas = new ArrayList<>();

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

    public void alterar(PessoaFisica pessoaFisica) {
        for (int i = 0; i < pessoasFisicas.size(); i++) {
            if (pessoasFisicas.get(i).getId() == pessoaFisica.getId()) {
                pessoasFisicas.set(i, pessoaFisica);
                break;
            }
        }
    }

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

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

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

    public void persistir(String nomeArquivo) throws IOException {
        FileOutputStream fos = new FileOutputStream(nomeArquivo);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(pessoasFisicas);
        oos.close();
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {

```

```

        FileInputStream fis = new FileInputStream(nomeArquivo);
        ObjectInputStream ois = new ObjectInputStream(fis);

        pessoasFisicas = (ArrayList<PessoaFisica>) ois.readObject();
        ois.close();
    }
}

```

## PessoaJuridica.java

```

package cadastradopoo.model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable
{
    private String cnpj;

    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);
    }
}

```

## PessoaJuridicaRepo.java

```
package cadastrpoo.model;

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

public class PessoaJuridicaRepo implements Serializable {
    private ArrayList<PessoaJuridica> pessoasJuridicas = new
    ArrayList<>();

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

    public void alterar(PessoaJuridica pessoaJuridica) {
        for (int i = 0; i < pessoasJuridicas.size(); i++) {
            if (pessoasJuridicas.get(i).getId() ==
pessoaJuridica.getId()) {
                pessoasJuridicas.set(i, pessoaJuridica);
                break;
            }
        }
    }

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

    public PessoaJuridica obter(int id) {
        for (PessoaJuridica p : pessoasJuridicas) {
            if (p.getId() == id) {
                return p;
            }
        }
    }
}
```

```
        return null;
    }

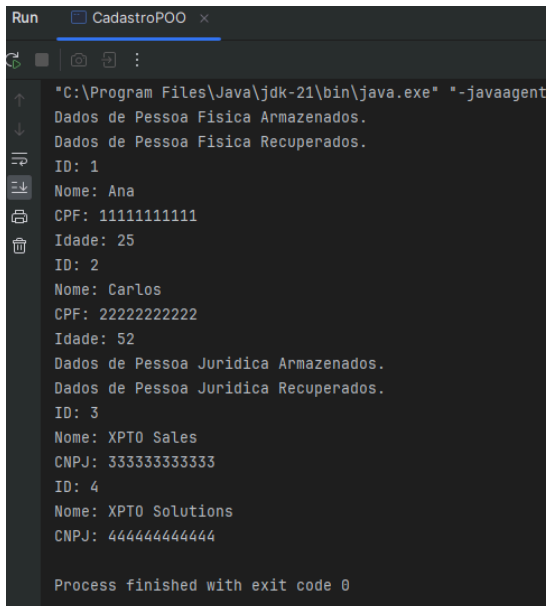
    public ArrayList<PessoaJuridica> obterTodos() {
        return pessoasJuridicas;
    }

    public void persistir(String nomeArquivo) throws IOException {
        FileOutputStream fos = new FileOutputStream(nomeArquivo);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(pessoasJuridicas);
        oos.close();
    }

    public void recuperar(String nomeArquivo) throws IOException,
        ClassNotFoundException {
        FileInputStream fis = new FileInputStream(nomeArquivo);
        ObjectInputStream ois = new ObjectInputStream(fis);
        pessoasJuridicas = (ArrayList<PessoaJuridica>)
ois.readObject();
        ois.close();
    }
}
```



## Resultados da Execução dos Códigos



```
Run CadastroPOO x
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent
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: XPT0 Sales
CNPJ: 33333333333
ID: 4
Nome: XPT0 Solutions
CNPJ: 44444444444
Process finished with exit code 0
```

**Figura 1:** Execução após finalizar código do primeiro procedimento

## Análise e Conclusão

(A) Quais as vantagens e desvantagens do uso de herança?

As vantagens são que a herança permite que as classes filhas herdem campos e métodos das classes pais, o que ajuda bastante na hora de reutilizar os códigos além de que o código fica mais limpo e organizado facilitando assim a leitura seja para quem está fazendo o código ou até mesmo para a equipe. Já as desvantagens são que o código acaba por ser mais difícil de modificar além de possuir um design ruim pois as vezes a classe filha vai obter comportamentos que não são muito necessários da classe pai e isso leva a um design de código ruim.

(B) Por que a interface `Serializable` é necessária ao efetuar persistência em arquivos binários?

A interface `Serializable` em Java é usada para indicar que uma classe pode ser serializada. A serialização é o processo de converter o estado de um objeto em um fluxo de bytes para armazenar o objeto em memória, um arquivo ou banco de dados, ou transmitir o objeto através de uma rede. Portanto, a interface `Serializable` é necessária ao efetuar persistência em

arquivos binários porque permite que os objetos sejam convertidos em um formato que pode ser armazenado ou transmitido e depois reconstruído em um objeto novamente.

(C) Como o paradigma funcional é utilizado pela API stream no Java?

A API Java Stream é uma das maneiras pelas quais o paradigma funcional é usado em Java. A API Stream permite lidar com coleções de objetos de maneira concisa e fácil de entender. Ele fornece funções como filtragem, mapeamento, redução, agregação, etc. que pode ser encadeado em um único pipeline de processamento de dados. Essas funções são funções avançadas, o que significa que aceitam outras funções como argumentos. Este é um exemplo de paradigma funcional em ação.

(D) Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na persistência de dados em arquivos?

Ao trabalhar com Java, o padrão de desenvolvimento mais comum para persistência de dados em arquivos é o uso de frameworks ORM, como o Hibernate ou o Spring Data JPA. Esses frameworks automatizam a serialização e desserialização de objetos, tornando o processo mais eficiente e produtivo.

## 2º Procedimento - Sistema Cadastro de Pessoas em Modo Texto

### Objetivo da Prática

- Implementar um sistema de cadastro e gerenciamento de pessoas físicas e jurídicas utilizando a Programação Orientada a Objetos (POO) em Java.
- O sistema deve ser capaz de inserir, alterar, excluir obter dados específicos e todos os dados de pessoas físicas e(ou) jurídicas.
- O sistema deve ser capaz de alterar os dados das pessoas físicas e jurídicas.
- O sistema deve ser capaz de persistir os dados em um arquivo.
- O sistema deve ser capaz de recuperar os dados de um arquivo.
- Implementar o modo texto ao código
- Fazer uso de herança e polimorfismo

### Códigos

Link do Github com os códigos da Prática:

<https://github.com/BeaterOfWar/CadastroPoo>

### CadastroPOO2.java

```
package cadastrapoo;

import java.util.ArrayList;

import java.util.Scanner;

import cadastrapoo.model.*;

import java.io.IOException;

public class CadastroPOO2 {

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

        Scanner scanner = new Scanner(System.in);

        PessoaFisicaRepo repoPessoaFisica = new PessoaFisicaRepo();

        PessoaJuridicaRepo repoPessoaJuridica = new PessoaJuridicaRepo();

        int opcao;
```

```
do {

    System.out.println("\n=====");

    System.out.println("Opções:");

    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. Finalizar");

    System.out.println("=====");

    System.out.print("Escolha uma opção: ");

    opcao = scanner.nextInt();

    scanner.nextLine();

    switch (opcao) {

        case 1:

            System.out.print("Tipo Fisica (F) ou Juridica (J): ");

            String tipoCadastro = scanner.nextLine();

            if (tipoCadastro.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();

        System.out.print("Idade: ");

        int idade = scanner.nextInt();

        scanner.nextLine();

        PessoaFisica novaPessoaFisica = new PessoaFisica(id, nome, cpf,
idade);

        repoPessoaFisica.inserir(novaPessoaFisica);

        System.out.println("Pessoa fisica adicionada com sucesso!");

    } else if (tipoCadastro.equalsIgnoreCase("J")) {

        System.out.print("ID: ");

        int id = scanner.nextInt();

        scanner.nextLine();

        System.out.print("Nome da empresa: ");

        String nomeEmpresa = scanner.nextLine();

        System.out.print("CNPJ: ");

        String cnpj = scanner.nextLine();

        PessoaJuridica novaEmpresa = new PessoaJuridica(id, nomeEmpresa,
cnpj);

        repoPessoaJuridica.inserir(novaEmpresa);

        System.out.println("Empresa adicionada com sucesso!");

    } else {

        System.out.println("Tipo inválido.");

    }

    break;

case 2:

```

```

        System.out.print("Tipo Fisica (F) ou Juridica (J): ");

        String tipoAlteracao = scanner.nextLine();

        System.out.print("ID da pessoa a ser alterada: ");

        int idAlteracao = scanner.nextInt();

        scanner.nextLine();

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

            PessoaFisica pessoaAlteracao = repoPessoaFisica.obter(idAlteracao);

            if (pessoaAlteracao != null) {

                pessoaAlteracao.exibir();

                System.out.print("Novo nome: ");

                String novoNome = scanner.nextLine();

                pessoaAlteracao.setNome(novoNome);

                System.out.println("Dados alterados com sucesso!");

            } else {

                System.out.println("Pessoa física não encontrada.");

            }

        } else if (tipoAlteracao.equalsIgnoreCase("J")) {

            PessoaJuridica empresaAlteracao =
repoPessoaJuridica.obter(idAlteracao);

            if (empresaAlteracao != null) {

                empresaAlteracao.exibir();

                System.out.print("Novo nome da empresa: ");

                String novoNomeEmpresa = scanner.nextLine();

                empresaAlteracao.setNome(novoNomeEmpresa);

```

```

        System.out.println("Dados da empresa alterados com sucesso!");

    } else {

        System.out.println("Empresa não encontrada.");

    }

} else {

    System.out.println("Tipo inválido.");

}

break;

case 3:

    System.out.print("Tipo Fisica (F) ou Juridica (J): ");

    String tipoExclusao = scanner.nextLine();

    System.out.print("ID da pessoa a ser excluída: ");

    int idExclusao = scanner.nextInt();

    scanner.nextLine();

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

        boolean excluiu = repoPessoaFisica.excluir(idExclusao);

        if (excluiu) {

            System.out.println("Pessoa excluída com sucesso!");

        } else {

            System.out.println("Pessoa não encontrada.");

        }

    } else if (tipoExclusao.equalsIgnoreCase("J")) {

    } else {

        System.out.println("Tipo inválido.");

    }

}

```

```

        break;

    case 4:

        System.out.print("Tipo Fisica (F) ou Juridica (J): ");

        String tipoExibicao = scanner.nextLine();

        System.out.print("ID da pessoa a ser exibida: ");

        int idExibicao = scanner.nextInt();

        scanner.nextLine();

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

            PessoaFisica pessoaExibicao = repoPessoaFisica.obter(idExibicao);

            if (pessoaExibicao != null) {

                pessoaExibicao.exibir();

            } else {

                System.out.println("Pessoa física não encontrada.");

            }

        } else if (tipoExibicao.equalsIgnoreCase("J")) {

            PessoaJuridica empresaExibicao =
repoPessoaJuridica.obter(idExibicao);

            if (empresaExibicao != null) {

                empresaExibicao.exibir();

            } else {

                System.out.println("Empresa não encontrada.");

            }

        } else {

            System.out.println("Tipo inválido.");

        }

        break;

    case 5:

```



```

        System.out.print("Tipo Fisica (F) ou Juridica (J): ");

        String tipoExibicaoTodos = scanner.nextLine();

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

            ArrayList<PessoaFisica> todasPessoasFisicas =
repoPessoaFisica.obterTodos();

            for (PessoaFisica pessoa : todasPessoasFisicas) {

                pessoa.exibir();

            }

        } else if (tipoExibicaoTodos.equalsIgnoreCase("J")) {

            ArrayList<PessoaJuridica> todasEmpresas =
repoPessoaJuridica.obterTodos();

            for (PessoaJuridica empresa : todasEmpresas) {

                empresa.exibir();

            }

        } else {

            System.out.println("Tipo inválido.");

        }

        break;

    case 6:

        System.out.print("Prefixo dos arquivos: ");

        String prefixo = scanner.nextLine();

        repoPessoaFisica.persistir(prefixo + ".fisica.bin");

        repoPessoaJuridica.persistir(prefixo + ".juridica.bin");

        System.out.println("Dados salvos com sucesso!");

        break;

    case 7:

        System.out.print("Prefixo dos arquivos: ");

```

```
        prefixo = scanner.nextLine();

        try {

            repoPessoaFisica.recuperar(prefixo + ".fisica.bin");

        } catch (ClassNotFoundException e) {

            throw new RuntimeException(e);

        }

        try {

            repoPessoaJuridica.recuperar(prefixo + ".juridica.bin");

        } catch (ClassNotFoundException e) {

            throw new RuntimeException(e);

        }

        System.out.println("Dados recuperados com sucesso!");

        break;

    case 0:

        System.out.println("Finalizando o programa.");

        break;

    default:

        System.out.println("Opção inválida.");

    }

} while (opcao != 0);

scanner.close();

}

}
```

## Resultados da Execução dos Códigos

```
=====
Opções:
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Finalizar
=====
Escolha uma opção: 1
Tipo Física (F) ou Juridica (J): f
ID: 111
Nome: MOISES EDUARDO
CPF: 312451278821
Idade: 20
Pessoa física adicionada com sucesso!
=====
```

**Figura 2:** Resultado da execução da opção 1

```
=====
Opções:
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Finalizar
=====
Escolha uma opção: 2
Tipo Física (F) ou Juridica (J): f
ID da pessoa a ser alterada: 111
ID: 111
Nome: MOISES EDUARDO
CPF: 312451278821
Idade: 20
Novo nome: Joaozinho
Dados alterados com sucesso!
=====
```

**Figura 3:** Resultado da execução da opção 2

```
=====
Escolha uma opção: 3
Tipo Fisica (F) ou Juridica (J): f
ID da pessoa a ser excluída: 111
Pessoa física excluída com sucesso!
=====
```

**Figura 4:** Resultado da execução da opção 3

```
=====
Opções:
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Finalizar
=====
Escolha uma opção: 4
Tipo Fisica (F) ou Juridica (J): f
ID da pessoa a ser exibida: 111
ID: 111
Nome: Moises
CPF: 183467245
Idade: 20
=====
```

**Figura 5:** Resultado da execução da opção 4

```
=====
Escolha uma opção: 5
Tipo Fisica (F) ou Juridica (J): f
ID: 111
Nome: Moises
CPF: 183467245
Idade: 20
ID: 132
Nome: Moises
CPF: 136751211
Idade: 19
```

**Figura 6:** Resultado da execução da opção 5

```
=====
Escolha uma opção: 6
Prefixo dos arquivos: dados 1
Dados salvos com sucesso!
=====
```

**Figura 7:** Resultado da execução da opção 6

```
=====
Opções:
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Finalizar
=====
Escolha uma opção: 7
Prefixo dos arquivos: dados 1
Dados recuperados com sucesso!
=====
```

**Figura 8:** Resultado da execução da opção 7

```
=====
Opções:
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Finalizar
=====
Escolha uma opção: 0
Finalizando o programa.

Process finished with exit code 0
```

**Figura 9:** Resultado da Execução da opção 0

### **Análise e Conclusão:**

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

Em Java, elementos estáticos são aqueles que pertencem à classe em vez de uma instância específica da classe. e o método main estático é uma convenção que garante que a JVM saiba onde começar a executar o programa.

- (B) Para que serve a classe Scanner?

A classe scanner serve para que o java leia os dados do teclado é basicamente um input do java

- (C) Como o uso de classes de repositório impactou na organização do código?

Impactou positivamente pois trouxe uma melhor organização ao código melhorando principalmente o gerenciamento dele e facilitou a rastreabilidade das alterações.

## Referências Bibliográficas

### [1] “Qual a finalidade da interface Serializable?”

Disponível em:

<https://pt.stackoverflow.com/questions/88270/qual-a-finalidade-da-interface-serializable>

Acesso em 19 de Abril de 2024.

### [2] “Java 8: Iniciando o desenvolvimento com a Streams API”

Disponível em

<https://www.oracle.com/br/technical-resources/articles/java-stream-api.html>Acesso em 19 de Abril de 2024.

### [3] “Persistindo dados em Java com JPA”

Disponível em

<https://www.linhadecodigo.com.br/artigo/2525/persistencia-em-java-com-api-jpa.aspx>

Acesso em 19 de Abril de 2024.