



- DESENVOLVIMENTO FULL STACK- TURMA 9001
- Disciplina: RPG0014 - Iniciando o caminho pelo Java
- Semestre Letivo: 2025.01
- Repositorio Git: [https://github.com/Elena-Gudimenko/Missao\\_01\\_Mundo\\_03.git](https://github.com/Elena-Gudimenko/Missao_01_Mundo_03.git)
- ELENA VICTOROVNA GUDIMENKO, MATRICULA: 2024.0277.9826

## **Missão Prática | Nível 1 | Mundo 3**

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

Procedimento 1: Criação das Entidades e Sistema de Persistência

Procedimento 2: Criação do Cadastro em Modo Texto

## **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:**

Procedimento 1: Criação das Entidades e Sistema de Persistência

Procedimento 2: Criação do Cadastro em Modo Texto

# Entidades:

## • Classe Pessoa

```
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;  
    }  
}
```

### • Classe PessoaFisica

```
package model;
```

```
import java.io.Serializable;
```

```
public class PessoaFisica extends Pessoa implements Serializable {  
    private String cpf;  
    private int idade;  
  
    public PessoaFisica() {  
        super();  
    }  
  
    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;
    }
}

```

### • **Classe PessoaJuridica**

```
package model;
```

```
import java.io.Serializable;
```

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

```

    public PessoaJuridica() {
        super();
    }

```

```

    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;
}
}

```

## Gerenciadores:

### • Classe PessoaFisicaRepo

```

package model;

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

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

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

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

```

```
    }  
    }  
}
```

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

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

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

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

```
@SuppressWarnings("unchecked")
```

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

```
}  
}
```

### • **Classe PessoaJuridicaRepo**

```
package model;
```

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

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

```
public class PessoaJuridicaRepo {  
    private ArrayList<PessoaJuridica> lista = new ArrayList<>();
```

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

```
    public void alterar(PessoaJuridica pessoa) {  
        for (int i = 0; i < lista.size(); i++) {  
            if (lista.get(i).getId() == pessoa.getId()) {  
                lista.set(i, pessoa);  
                return;  
            }  
        }  
    }
```

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

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

```

        }
    }
    return null;
}

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

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

@SuppressWarnings("unchecked")
public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
    FileInputStream fis = new FileInputStream(nomeArquivo);
    ObjectInputStream ois = new ObjectInputStream(fis);
    lista = (ArrayList<PessoaJuridica>) ois.readObject();
    ois.close();
    fis.close();
}
}

```

## Aplicação:

- **Classe CadastroPOO - Referente ao Procedimento 1**

```
import java.io.IOException;
```

```
import model.*;
```

```
public class CadastroPOO {
```



```

public static void main(String[] args) {
    try {
        // --- Pessoa Fisica ---

        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
        repo1.inserir(new PessoaFisica(1, "Ivan Silva", "123.456.789-00", 30));
        repo1.inserir(new PessoaFisica(2, "Maria Souza", "987.654.321-00", 25));
        repo1.persistir("pessoas_fisicas.dat");

        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
        repo2.recuperar("pessoas_fisicas.dat");

        System.out.println("PESSOAS FISICAS:");
        for (PessoaFisica pf : repo2.obterTodos()) {
            pf.exibir();
            System.out.println();
        }

        // --- Pessoa Jur?dica ---

        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
        repo3.inserir(new PessoaJuridica(1, "Empresa Alpha", "11.222.333/0001-44"));
        repo3.inserir(new PessoaJuridica(2, "Tech Solutions", "55.666.777/0001-99"));
        repo3.persistir("pessoas_juridicas.dat");

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

        System.out.println("PESSOAS JURIDICAS:");
        for (PessoaJuridica pj : repo4.obterTodos()) {
            pj.exibir();
            System.out.println();
        }
    }
}

```

```

    } catch (IOException | ClassNotFoundException e) {
        System.out.println("Erro: " + e.getMessage());
    }
}
}
}

```

- **Classe CadastroPOO2 - Referente ao Procedimento 2**

```

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

public class CadastroPOO2 {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        PessoaFisicaRepo pfRepo = new PessoaFisicaRepo();
        PessoaJuridicaRepo pjRepo = new PessoaJuridicaRepo();

        int opcao;
        do {
            System.out.println("\n===== MENU =====");
            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 opcao: ");
```

```
opcao = scanner.nextInt();
```

```
scanner.nextLine(); // Limpa o buffer
```

```
switch (opcao) {
```

```
    case 1 -> {
```

```
        System.out.print("Tipo (F para Fi sica / J para Juri dica): ");
```

```
        String tipo = scanner.nextLine().toUpperCase();
```

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

```
        int id = scanner.nextInt();
```

```
        scanner.nextLine();
```

```
        System.out.print("Nome: ");
```

```
        String nome = scanner.nextLine();
```

```
        if (tipo.equals("F")) {
```

```
            System.out.print("CPF: ");
```

```
            String cpf = scanner.nextLine();
```

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

```
            int idade = scanner.nextInt();
```

```
            scanner.nextLine();
```

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

```
        } else if (tipo.equals("J")) {
```

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

```
            String cnpj = scanner.nextLine();
```

```
            pjRepo.inserir(new PessoaJuridica(id, nome, cnpj));
```

```
        }
```

```
    }
```

```
    case 2 -> {
```

```
        System.out.print("Tipo (F para Fi sica / J para Juri dica): ");
```

```
        String tipo = scanner.nextLine().toUpperCase();
```

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

```
int id = scanner.nextInt();
scanner.nextLine();

if (tipo.equals("F")) {
    PessoaFisica pf = pfRepo.obter(id);
    if (pf != null) {
        System.out.println("Dados atuais:");
        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();

        pfRepo.alterar(new PessoaFisica(id, nome, cpf, idade));
    }
} else if (tipo.equals("J")) {
    PessoaJuridica pj = pjRepo.obter(id);
    if (pj != null) {
        System.out.println("Dados atuais:");
        pj.exibir();

        System.out.print("Novo nome: ");
        String nome = scanner.nextLine();
        System.out.print("Novo CNPJ: ");
        String cnpj = scanner.nextLine();

        pjRepo.alterar(new PessoaJuridica(id, nome, cnpj));
    }
}
```

```

    }
}
case 3 -> {
    System.out.print("Tipo (F para Fi sica / J para Juri dica): ");
    String tipo = scanner.nextLine().toUpperCase();
    System.out.print("ID: ");
    int id = scanner.nextInt();
    scanner.nextLine();

    if (tipo.equals("F")) {
        pfRepo.excluir(id);
    } else if (tipo.equals("J")) {
        pjRepo.excluir(id);
    }
}

case 4 -> {
    System.out.print("Tipo (F para Fi sica / J para Juri dica): ");
    String tipo = scanner.nextLine().toUpperCase();
    System.out.print("ID: ");
    int id = scanner.nextInt();
    scanner.nextLine();

    if (tipo.equals("F")) {
        PessoaFisica pf = pfRepo.obter(id);
        if (pf != null) pf.exibir();
    } else if (tipo.equals("J")) {
        PessoaJuridica pj = pjRepo.obter(id);
        if (pj != null) pj.exibir();
    }
}

case 5 -> {
    System.out.print("Tipo (F para Fi sica / J para Juri dica): ");

```

```

String tipo = scanner.nextLine().toUpperCase();

if (tipo.equals("F")) {
    for (PessoaFisica pf : pfRepo.obterTodos()) {
        pf.exibir();
        System.out.println();
    }
} else if (tipo.equals("J")) {
    for (PessoaJuridica pj : pjRepo.obterTodos()) {
        pj.exibir();
        System.out.println();
    }
}

case 6 -> {
    System.out.print("Prefixo do arquivo: ");
    String prefixo = scanner.nextLine();
    try {
        pfRepo.persistir(prefixo + ".fisica.bin");
        pjRepo.persistir(prefixo + ".juridica.bin");
        System.out.println("Dados salvos com sucesso!");
    } catch (IOException e) {
        System.out.println("Erro ao salvar: " + e.getMessage());
    }
}

case 7 -> {
    System.out.print("Prefixo do arquivo: ");
    String prefixo = scanner.nextLine();
    try {
        pfRepo.recuperar(prefixo + ".fisica.bin");
        pjRepo.recuperar(prefixo + ".juridica.bin");
        System.out.println("Dados recuperados com sucesso!");
    }
}

```

```

        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao recuperar: " + e.getMessage());
        }
    }

    case 0 -> System.out.println("Encerrando...");
    default -> System.out.println("Opcao invalida.");
}

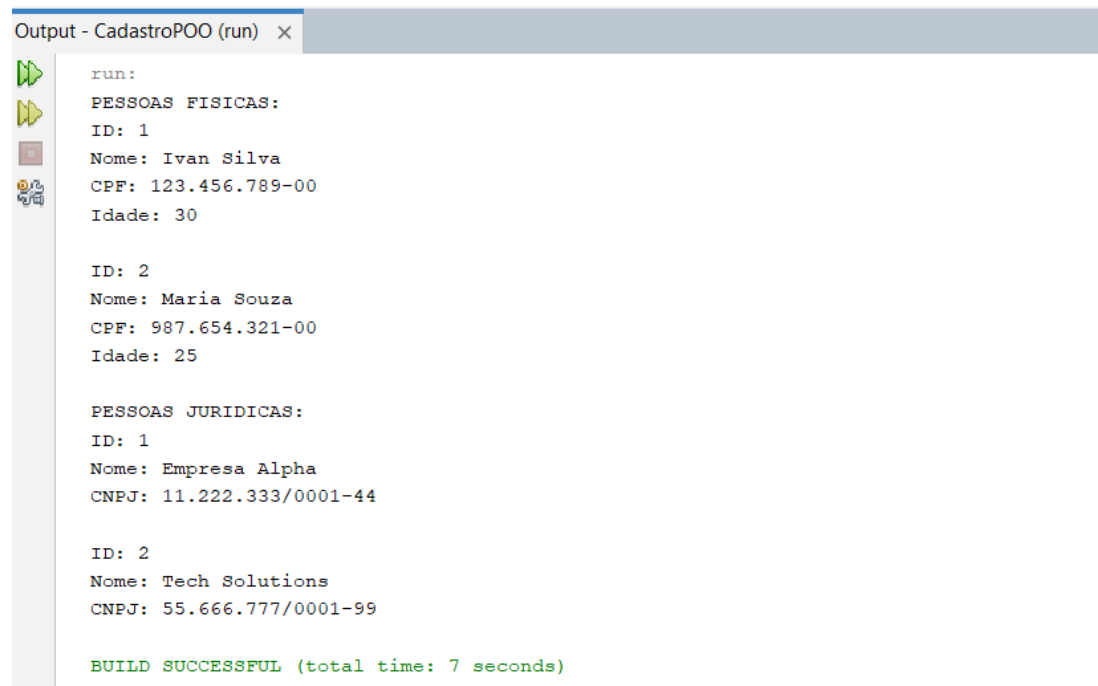
} while (opcao != 0);

scanner.close();
}
}

```

## Resultados:

### Procedimento 1:



```

Output - CadastroPOO (run) x
run:
PESSOAS FISICAS:
ID: 1
Nome: Ivan Silva
CPF: 123.456.789-00
Idade: 30

ID: 2
Nome: Maria Souza
CPF: 987.654.321-00
Idade: 25

PESSOAS JURIDICAS:
ID: 1
Nome: Empresa Alpha
CNPJ: 11.222.333/0001-44

ID: 2
Nome: Tech Solutions
CNPJ: 55.666.777/0001-99

BUILD SUCCESSFUL (total time: 7 seconds)

```

## Análise e Conclusão

- Quais as vantagens e desvantagens do uso de herança?

### Vantagens:

1. Classes podem herdar características(métodos e atributos) de outras classes situadas acima ou transmitir suas características às classes abaixo; 2. Evita repetir o mesmo código várias vezes; 3. Caso uma alteração seja necessária, ela só precisará ser feita na classe

pai, e será automaticamente propagada para as subclasses.

### Desvantagens:

1. Fraco encapsulamento entre classes e subclasses e o forte acoplamento entre elas onde ao mudar uma superclasse pode afetar todas as subclasses; 2. Quando um objeto precisa ser de uma classe diferente em momentos diferentes e não é possível com a herança.

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

Essa interface permite que os objetos sejam serializados(convertidos em uma sequência de bytes) e desserializados com a conversão de volta à um objeto.

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

A API stream é usada para manipular coleções (Collections) de uma maneira mais eficiente, utilizando funções.Ela possibilita uma iteração sobre essas coleções de objetos e, a cada elemento, realizar alguma ação, seja ela de filtragem, mapeamento, transformação, etc.

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

Nesse projeto foram utilizadas a classe ObjectOutputStream para escrever objetos em um arquivo "[prefixo].fisica.bin e [prefixo].juridica.bin" e a classe ObjectInputStream para ler os objetos dos mesmos arquivos.

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

Os elementos estáticos são elementos que "existem", ou seja, estão disponíveis para uso, sem a necessidade de serem instanciados. Podem ser utilizados em código sem a necessidade de existirem objetos produzidos (sem a necessidade de um comando "new Classe(")).

- Para que serve a classe Scanner ?

Para leitura de dados de entrada (inteiros, boolean, string, etc) inseridos pelo usuario atraves do teclado.

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



As classes de repositórios serviram para gerenciar, centralizar e organizar as atividades de inserir, excluir, alterar, localizar, recuperar e salvar os dados de pessoa física e jurídica.