

# Iniciando o caminho pelo Java

Missão Prática | Nível 1 | mundo 3





POLO PRQ ARTUR ALVIM - SÃO PAULO – SP

Desenvolvimento Full Stack

Iniciando o caminho pelo Java

Turma 2025.1

Primeiro Semestre (2025)

Daniel Souza Pereira

Matrícula: 202401488135

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

---

## Objetivo da prática

Utilizar herança e polimorfismo na definição de entidades;

Criar uma hierarquia de classes (Pessoa, PessoaFisica, PessoaJuridica) para demonstrar herança e polimorfismo;

Utilizar persistência de objetos em arquivos binários;

Implementar métodos para salvar e recuperar dados em arquivos binários usando serialização.

Implementar uma interface cadastral em modo texto;

Desenvolver um menu interativo para interação com o usuário via linha de comando;

Utilizar o controle de exceções da plataforma Java;

Tratar possíveis erros durante operações como persistência e recuperação de dados.





## Códigos utilizados:

Criar um projeto do tipo Ant..Java Application no NetBeans, utilizando o nome CadastroPOO para o projeto.

Criar um pacote com o nome "model", para as entidades e gerenciadores.

No pacote model criar as entidades:

Classe Pessoa, com os campos id e nome (texto), método exibir, para impressão dos dados, construtor padrão e completo, além de getters e setters para todos os campos.

Classe PessoaFisica, herdando de Pessoa, com o acréscimo dos campos cpf e idade, método exibir polimórfico

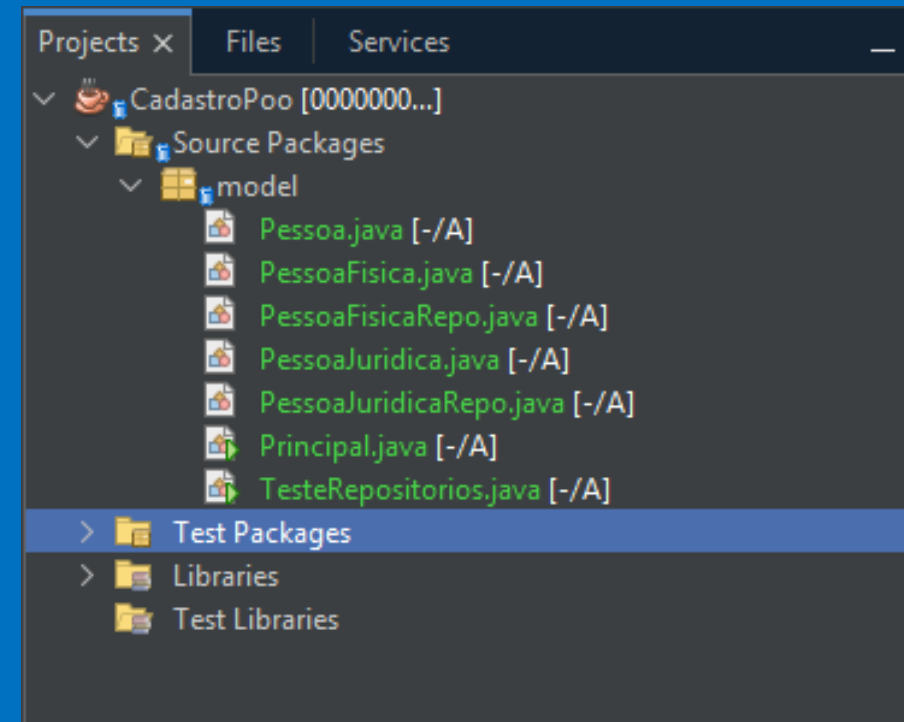
Classe PessoaJuridica, herdando de Pessoa, com o acréscimo do campo cnpj.

Adicionar interface Serializable em todas as Classes

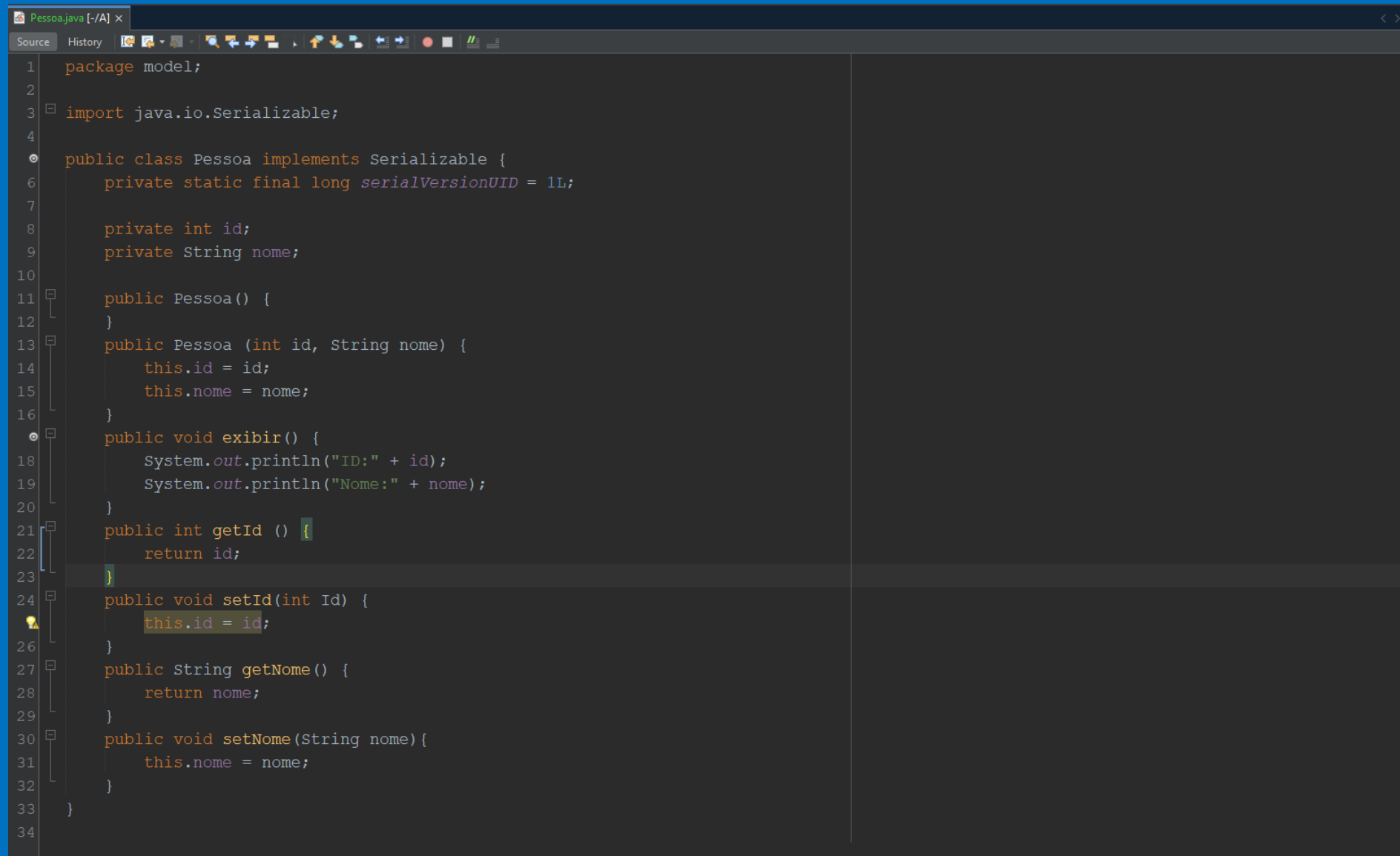
No pacote model criar os gerenciadores, com as seguintes características:

Classe PessoaFisicaRepo e Classe PessoaFisicaRepo, contendo um ArrayList de PessoaFisica, nível de acesso privado, e métodos

públicos inserir, alterar, excluir, obter e obterTodos, para gerenciamento das entidades contidas no ArrayList.



# Classe Pessoa



```
1 package model;
2
3 import java.io.Serializable;
4
5 public class Pessoa implements Serializable {
6     private static final long serialVersionUID = 1L;
7
8     private int id;
9     private String nome;
10
11     public Pessoa() {
12     }
13
14     public Pessoa (int id, String nome) {
15         this.id = id;
16         this.nome = nome;
17     }
18
19     public void exibir() {
20         System.out.println("ID:" + id);
21         System.out.println("Nome:" + nome);
22     }
23
24     public int getId () {
25         return id;
26     }
27
28     public void setId(int Id) {
29         this.id = id;
30     }
31
32     public String getNome() {
33         return nome;
34     }
35
36     public void setNome(String nome){
37         this.nome = nome;
38     }
39 }
```

# Classe PessoaFisica

```
PessoaFisica.java [-/A] x
Source History
1 package model;
2
3 import java.io.Serializable;
4
5 public class PessoaFisica extends Pessoa implements Serializable {
6     private static final long serialVersionUID = 1L;
7
8     private String cpf;
9     private int idade;
10
11     public PessoaFisica() {
12     }
13
14     public PessoaFisica (int id, String nome, String cpf, int idade) {
15         super(id, nome);
16         this.cpf = cpf;
17         this.idade = idade;
18     }
19
20     @Override
21     public void exibir() {
22         super.exibir();
23         System.out.println("CPF:" + cpf);
24         System.out.println("Idade:" + idade);
25     }
26
27     public String getCpf() {
28         return cpf;
29     }
30     public void setCpf (String cpf) {
31         this.cpf = cpf;
32     }
33     public int getIdade() {
34         return idade;
35     }
36     public void setIdade(int idade) {
37         this.idade = idade;
38     }
39 }
40
```

# Classe PessoaJuridica

```
PessoaJuridica.java [-/A] x
Source History
1 package model;
2
3 import java.io.Serializable;
4
5 public class PessoaJuridica extends Pessoa implements Serializable {
6     private static final long serialVersionUID = 1L;
7
8     private String cnpj;
9
10    public PessoaJuridica() {
11    }
12
13    public PessoaJuridica (int id, String nome, String cnpj) {
14        super(id, nome);
15        this.cnpj = cnpj;
16    }
17
18    @Override
19    public void exhibir() {
20        super.exibir();
21        System.out.println("CNPJ:" + cnpj);
22    }
23    public String getCnpj() {
24        return cnpj;
25    }
26    public void setCnpj(String cnpj) {
27        this.cnpj = cnpj;
28    }
29 }
30
```



# Classe PessoaFisicaRepo

```
PessoaFisicaRepo.java [-/A] x
Source History
1 package model;
2
3 import java.io.*;
4 import java.util.ArrayList;
5
6 public class PessoaFisicaRepo {
7     private ArrayList<PessoaFisica> pessoas = new ArrayList<>();
8
9     public void inserir (PessoaFisica pessoa){
10         pessoas.add(pessoa);
11     }
12
13     public void alterar (PessoaFisica pessoa){
14         for (int i = 0; i < pessoas.size(); i++) {
15             if (pessoas.get(i).getId() == pessoa.getId()){
16                 pessoas.set(i, pessoa);
17                 break;
18             }
19         }
20     }
21
22     public void excluir(int id){
23         pessoas.removeIf(p -> p.getId() == id);
24     }
25
26     public PessoaFisica obter(int id) {
27         for (PessoaFisica pessoa : pessoas){
28             if (pessoa.getId() == id){
29                 return pessoa;
30             }
31         }
32         return null;
33     }
34     public ArrayList<PessoaFisica> obterTodos(){
35         return pessoas;
36     }
37     public void Persistir (String nomeArquivo) throws IOException {
38         try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
39             outputStream.writeObject(pessoas);
40         }
41     }
42     public void recuperar (String nomeArquivo) throws IOException, ClassNotFoundException {
43         try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
44             pessoas = (ArrayList<PessoaFisica>) inputStream.readObject();
45         }
46     }
47 }
48
```

# Classe PessoaJuridicaRepo

```
PessoaJuridicaRepo.java [-/A] x
Source History
C:\Users\rockn\OneDrive\Área de Trabalho\JavaProjects\Aulas\CadastroPoo\src\model\PessoaJuridicaRepo.java

1 package model;
2
3 import java.io.*;
4 import java.util.ArrayList;
5
6 public class PessoaJuridicaRepo {
7     private ArrayList<PessoaJuridica> pessoas = new ArrayList<>();
8
9     public void inserir(PessoaJuridica pessoa) {
10         pessoas.add(pessoa);
11     }
12
13     public void alterar (PessoaJuridica pessoa) {
14         for (int i = 0; i < pessoas.size(); i++) {
15             if (pessoas.get(i).getId() == pessoa.getId()) {
16                 pessoas.set(i, pessoa);
17                 break;
18             }
19         }
20     }
21
22     public void excluir(int id){
23         pessoas.removeIf(p -> p.getId() == id);
24     }
25
26     public PessoaJuridica obter(int id) {
27         for (PessoaJuridica pessoa : pessoas) {
28             if (pessoa.getId() == id) {
29                 return pessoa;
30             }
31         }
32         return null;
33     }
34
35     public ArrayList<PessoaJuridica> obterTodos() {
36         return pessoas;
37     }
38
39     public void persistir(String nomeArquivo) throws IOException {
40         try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
41             outputStream.writeObject(pessoas);
42         }
43     }
44
45     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
46         try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
47             pessoas = (ArrayList<PessoaJuridica>) inputStream.readObject();
48         }
49     }
50 }
```

# Classe Principal

```
package model;

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

public class Principal {
    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 ===");
            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.println("Escolha uma Opção:");
            opcao = scanner.nextInt();
            scanner.nextLine();
        } while (opcao != 0);
    }
}
```

```
try {
    switch (opcao) {
        case 1:
            incluir(scanner, repoFisica, repoJuridica);
            break;
        case 2:
            alterar(scanner, repoFisica, repoJuridica);
            break;
        case 3:
            excluir(scanner, repoFisica, repoJuridica);
            break;
        case 4:
            exibirPorId(scanner, repoFisica, repoJuridica);
            break;
        case 5:
            exibirTodos(scanner, repoFisica, repoJuridica);
            break;
        case 6:
            salvarDados(scanner, repoFisica, repoJuridica);
            break;
        case 7:
            recuperarDados(scanner, repoFisica, repoJuridica);
            break;
        case 0:
            System.out.println("Finalizando Sistema...");
            break;
        default:
            System.out.println("Opção Inválida. Por favor tente novamente");
    }
} catch (Exception e) {
    System.out.println("Erro: " + e.getMessage());
}
```

# Classe Principal

```
    } while (opcao != 0);
    scanner.close();
}

private static void incluir(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
    System.out.println("Escolha o tipo (1 - fisica - 2 Juridica):");
    int tipo = scanner.nextInt();
    scanner.nextLine();

    if (tipo == 1) {
        System.out.println("ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.println("Nome: ");
        String nome = scanner.nextLine();
        System.out.println("CPF: ");
        String cpf = scanner.nextLine();
        System.out.println("Idade: ");
        int idade = scanner.nextInt();
        repoFisica.inserir(new PessoaFisica(id, nome, cpf, idade));
    } else if (tipo == 2) {
        System.out.println("ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.println("Nome: ");
        String nome = scanner.nextLine();
        System.out.println("CNPJ: ");
        String cnpj = scanner.nextLine();
        repoJuridica.inserir(new PessoaJuridica(id, nome, cnpj));
    } else {
        System.out.println("Tipo Invalido!");
    }
}

public static void alterar(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
    System.out.println("Escolha o tipo (1 - fisica - 2 Juridica):");
    int tipo = scanner.nextInt();
    scanner.nextLine();
}
```

```
System.out.println("ID: ");
int id = scanner.nextInt();
scanner.nextLine();

if (tipo == 1) {
    PessoaFisica pessoa = repoFisica.obter(id);
    if (pessoa != null) {
        System.out.println("Dados Atuais:");
        pessoa.exibir();
        System.out.println("Novo Nome: ");
        String nome = scanner.nextLine();
        System.out.println("CPF: ");
        String cpf = scanner.nextLine();
        System.out.println("Nova Idade: ");
        int idade = scanner.nextInt();
        pessoa.setNome(nome);
        pessoa.setCpf(cpf);
        pessoa.setIdade(idade);
        repoFisica.alterar(pessoa);
        System.out.println("Pessoa Fisica alterada com sucesso!");
    } else {
        System.out.println("Pessoa Fisica Não encontrada!");
    }
} else if (tipo == 2) {
    PessoaJuridica pessoa = repoJuridica.obter(id);
    if (pessoa != null) {
        System.out.println("Dados Atuais:");
        pessoa.exibir();
        System.out.println("Novo Nome: ");
        String nome = scanner.nextLine();
        System.out.println("Novo CNPJ: ");
        String cnpj = scanner.nextLine();
        pessoa.setNome(nome);
        pessoa.setCnpj(cnpj);
        repoJuridica.alterar(pessoa);
        System.out.println("Pessoa Juridica alterada com sucesso!");
    } else {
        System.out.println("Pessoa Juridica Não encontrada!");
    }
} else {
    System.out.println("Tipo Inválido!");
}
}
```

```
private static void excluir(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
    System.out.println("Escolha o tipo (1 - fisica - 2 Juridica):");
    int tipo = scanner.nextInt();
    scanner.nextLine();

    System.out.println("ID: ");
    int id = scanner.nextInt();

    if (tipo == 1) {
        repoFisica.excluir(id);
        System.out.println("Pessoa Fisica excluida com sucesso!");
    } else if (tipo == 2) {
        repoJuridica.excluir(id);
        System.out.println("Pessoa Juridica excluida com sucesso!");
    } else {
        System.out.println("Tipo Invalido!");
    }
}

private static void exibirPorId(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
    System.out.println("Escolha o tipo (1 - fisica - 2 Juridica):");
    int tipo = scanner.nextInt();
    scanner.nextLine();

    System.out.println("ID: ");
    int id = scanner.nextInt();

    if (tipo == 1) {
        PessoaFisica pessoa = repoFisica.obter(id);
        if (pessoa != null) {
            pessoa.exibir();
        } else {
            System.out.println("Pessoa Fisica não encontrada!");
        }
    } else if (tipo == 2) {
        PessoaJuridica pessoa = repoJuridica.obter(id);
        if (pessoa != null) {
            pessoa.exibir();
        } else {
            System.out.println("Pessoa Juridica não encontrada!");
        }
    } else {
        System.out.println("Tipo Inválido!");
    }
}
}
```

# Classe Principal

```
public static void exibirTodos(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
    System.out.println("Escolha o tipo (1 - fisica - 2 Juridica):");
    int tipo = scanner.nextInt();
    scanner.nextLine();

    if (tipo == 1) {
        for (PessoaFisica pessoa : repoFisica.obterTodos()) {
            pessoa.exibir();
            System.out.println("-----");
        }
    } else if (tipo == 2) {
        for (PessoaJuridica pessoa : repoJuridica.obterTodos()) {
            pessoa.exibir();
            System.out.println("-----");
        }
    } else {
        System.out.println("Tipo Inválido!");
    }
}

public static void salvarDados(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) throws IOException {
    System.out.println("Informe o prefixo dos arquivos: ");
    String prefixo = scanner.nextLine();

    repoFisica.Persistir(prefixo + ".fisica.bin");
    repoJuridica.persistir(prefixo + ".juridica.bin");
    System.out.println("Dados Salvos com sucesso!");
}

private static void recuperarDados (Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) throws IOException, ClassNotFoundException {
    System.out.println("informe o prefixo dos arquivos ");
    String prefixo = scanner.nextLine();

    repoFisica.recuperar(prefixo + ".fisica.bin");
    repoJuridica.recuperar(prefixo + ".juridica.bin");
    System.out.println("Dados recuperados com Sucesso!");
}
}
```

# Classe TesteRepositorios

```
package model;

import java.io.IOException;

public class TesteRepositorios {
    public static void main(String[] args) {
        try {
            PessoaFisicaRepo repol = new PessoaFisicaRepo();
            repol.inserir(new PessoaFisica(3421, "Daniel Souza Pereira", "222.333.111-10", 27));
            repol.inserir(new PessoaFisica(4511, "Karen Sateles", "111.111.444.555-10", 27));

            String arquivoPessoasFisicas = "pessoas_fisicas.dat";
            repol.Persistir(arquivoPessoasFisicas);

            PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
            repo2.recuperar(arquivoPessoasFisicas);

            System.out.println("Pessoas Fisicas Recuperadas!");
            for (PessoaFisica pessoa : repo2.obterTodos()) {
                pessoa.exibir();
                System.out.println("-----");
            }
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao manipular o repositória de Pessoas Fisicas: " + e.getMessage());
        }

        try {
            PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
            repo3.inserir(new PessoaJuridica(1997, "Dann Tech", "10.123.456/0001-00"));
            repo3.inserir(new PessoaJuridica(2025, "Karen Nutri", "20.321.456/0001-00"));

            String arquivoPessoasJuridicas = "pessoas_juridicas.dat";
            repo3.persistir(arquivoPessoasJuridicas);

            PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
            repo4.recuperar(arquivoPessoasJuridicas);

            System.out.println("\nPessoas Jurídicas Recuperadas!");
            for (PessoaJuridica pessoa : repo4.obterTodos()) {
                pessoa.exibir();
                System.out.println("-----");
            }
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao manipular o repositória de Pessoas Jurídicas: " + e.getMessage());
        }
    }
}
```

# Saída de Execução

```
run:
Pessoas Físicas Recuperadas!
ID:3421
Nome:Daniel Souza Pereira
CPF:222.333.111-10
Idade:27
-----
ID:4511
Nome:Karen Sateles
CPF:111.111.444.555-10
Idade:27
-----

Pessoas Jurídicas Recuperadas!
ID:1997
Nome:Dann Tech
CNPJ:10.123.456/0001-00
-----
ID:2025
Nome:Karen Nutri
CNPJ:20.321.456/0001-00
-----
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Análise e Conclusão:

### 1. Quais as vantagens e desvantagens do uso de herança?

#### Vantagens:

- 1. Reutilização de Código:** Com herança, você pode criar uma classe base (como Pessoa) que contém atributos e comportamentos comuns, como nome, idade, ou métodos como `exibirDados()`. Outras classes, como Cliente ou Funcionário, podem herdar dessa classe base, evitando duplicação de código.
- 2. Organização e Estrutura:** A herança ajuda a estruturar o sistema de forma hierárquica, tornando o código mais legível e organizado.
- 3. Facilidade de Extensão:** Caso precise adicionar novas categorias de pessoas (como Aluno ou Professor), é possível estender a classe base com facilidade.



## Análise e Conclusão:

### 1. Quais as vantagens e desvantagens do uso de herança?

#### Desvantagens:

1. Acoplamento Forte: Sistemas baseados em herança podem se tornar rigidamente acoplados, dificultando alterações nas classes base sem impactar todas as classes derivadas.
2. Sobrecarga de Complexidade: O uso excessivo de herança pode tornar o sistema mais complexo do que necessário, especialmente se a hierarquia for muito profunda.
3. Dificuldade de Manutenção: Alterar a classe base pode gerar consequências inesperadas em todas as subclasses, aumentando a dificuldade de manutenção.
4. Risco de Generalização Excessiva: Nem sempre todos os atributos e comportamentos da classe base são necessários para todas as subclasses, o que pode levar a uma modelagem inadequada.

## Análise e Conclusão:

### 2. Por que a interface Serializable é necessária ao efetuar persistência em arquivos binários?

A interface Serializable permite que objetos sejam transformados em bytes para serem salvos ou transmitidos, e posteriormente recriados.

Ao realizar a persistência de objetos em arquivos binários ela indica que um objeto pode ser convertido em uma sequência de bytes (processo conhecido como serialização). Isso é necessário para salvar o estado do objeto em um arquivo ou transmiti-lo através de uma rede, e posteriormente recriá-lo (deserialização).

Sem esta interface, o Java não consegue realizar a serialização, resultando em erro. Basicamente, a interface é como um marcador que comunica ao Java que o objeto pode ser transformado em bytes de maneira segura.

## Análise e Conclusão:

### 3. Como o paradigma funcional é utilizado pela API stream no Java?

O paradigma funcional é utilizado pela API Stream no Java por meio de operações que permitem trabalhar com dados de forma declarativa. Essa abordagem enfatiza o "o quê" deve ser feito, em vez do "como". Alguns exemplos de como isso é aplicado:

- **Imutabilidade:** As Streams não alteram as coleções originais; em vez disso, criam novas coleções ou resultados.
- **Funções como parâmetros:** Métodos como filter, map e reduce aceitam expressões lambda ou referências de métodos, permitindo manipulações concisas e expressivas.
- **Operações encadeadas:** Permite combinar várias etapas em uma única pipeline de operações, como filtrar, transformar e reduzir os dados.
- **Lazy evaluation:** As operações só são executadas quando necessário, otimizando o processamento.

## Análise e Conclusão:

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

Ao trabalhar com persistência de dados em arquivos no Java, o padrão de desenvolvimento frequentemente adotado é o Data Access Object (DAO). Esse padrão organiza e abstrai o acesso aos dados, permitindo que a lógica de persistência fique separada da lógica de negócios. Ele fornece métodos específicos para operações como salvar, atualizar, buscar e deletar dados, facilitando a manutenção e a reutilização do código.

Link para o Repositório no GitHub:

[DannPereira10/Iniciando-o-caminho-pelo-Java](#)