

 <b>Estácio</b>	<p align="center"><b>UNIVERSIDADE ESTÁCIO DE SÁ</b></p> <p align="center">POLO DALPLAZA CENTER – SÃO LUÍS/MA</p> <p align="center">DESENVOLVIMENTO FULL STACK - 22.3</p> <p align="center">Relatório da Missão Prática   Nível 1   Mundo 3</p>
Aluno:	Lucas Silva Costa
Professor:	Rodrigo Dias
Repositório:	<a href="https://github.com/LutchasDev/Mundo-3---Nivel-1">https://github.com/LutchasDev/Mundo-3---Nivel-1</a>

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

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

**1. Objetivos da prática:**

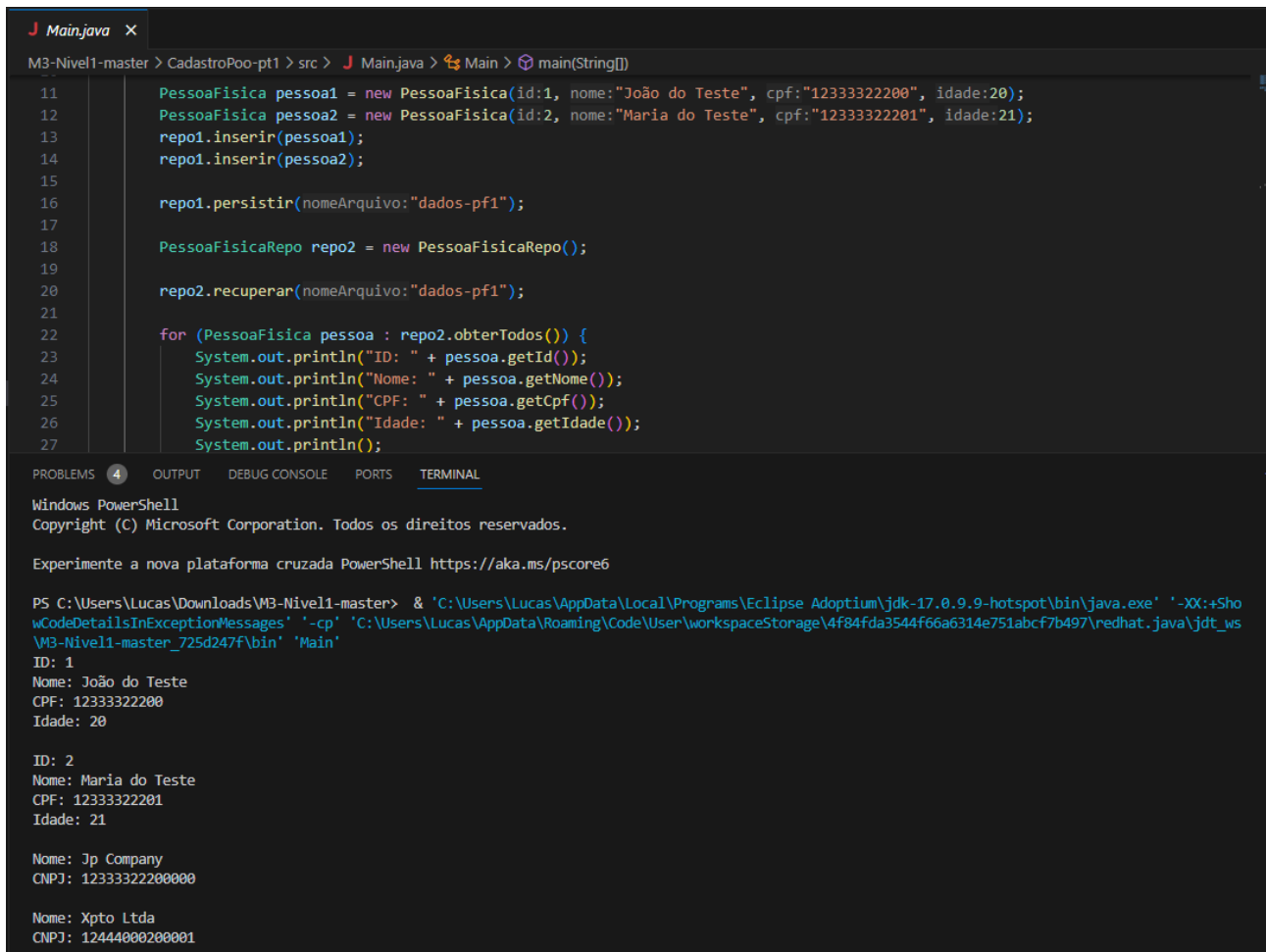
- Utilizar herança e polimorfismo na definição de entidades.
- Utilizar persistência de objetos em arquivos binários.
- Implementar uma interface cadastral em modo texto.
- Utilizar o controle de exceções da plataforma Java.
- No final do projeto, o aluno terá implementado um sistema cadastral em Java, utilizando os recursos da programação orientada a objetos e a persistência em arquivos binários.

**2. Todos os códigos solicitados neste roteiro de aula:**

<https://github.com/LutchasDev/Mundo-3---Nivel-1>

Obs.: Ao baixar o código, abra na IDE as pastas separadas, se não os arquivos vão ficar salvando na raiz da pasta M3-Nivel1.

### 3. Resultados da execução dos códigos PT1:



The screenshot shows an IDE with a Java file named `Main.java`. The code defines two classes, `PessoaFisica` and `PessoaFisicaRepo`, and a `main` method. The `main` method creates two `PessoaFisica` objects, inserts them into a repository, persists the data, and then retrieves and prints the details of all stored persons.

```
11 PessoaFisica pessoa1 = new PessoaFisica(id:1, nome:"João do Teste", cpf:"12333322200", idade:20);
12 PessoaFisica pessoa2 = new PessoaFisica(id:2, nome:"Maria do Teste", cpf:"12333322201", idade:21);
13 repo1.inserir(pessoa1);
14 repo1.inserir(pessoa2);
15
16 repo1.persistir(nomeArquivo:"dados-pf1");
17
18 PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
19
20 repo2.recuperar(nomeArquivo:"dados-pf1");
21
22 for (PessoaFisica pessoa : repo2.obterTodos()) {
23     System.out.println("ID: " + pessoa.getId());
24     System.out.println("Nome: " + pessoa.getNome());
25     System.out.println("CPF: " + pessoa.getCpf());
26     System.out.println("Idade: " + pessoa.getIdade());
27     System.out.println();
28 }
```

The terminal output shows the execution of the program, displaying the details of the two stored persons and the company information.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

PS C:\Users\Lucas\Downloads\M3-Nivel1-master> & 'C:\Users\Lucas\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.9-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Lucas\AppData\Roaming\Code\User\workspaceStorage\4f84fda3544f66a6314e751abcf7b497\redhat.java\jdt_ws\M3-Nivel1-master_725d247f\bin' 'Main'
ID: 1
Nome: João do Teste
CPF: 12333322200
Idade: 20

ID: 2
Nome: Maria do Teste
CPF: 12333322201
Idade: 21

Nome: Jp Company
CNPJ: 12333322200000

Nome: Xpto Ltda
CNPJ: 12444000200001
```

### Análise e Conclusão:

#### a) Quais as vantagens e desvantagens do uso de herança?

Vantagens:

1. Reutilização de código: Possibilita herdar atributos e métodos de uma classe pai, sem exigir muito esforço na escrita.
2. Extensibilidade: Permite a criação de novas classes baseadas em classes existentes, de forma mais ágil, adicionando ou modificando comportamentos.

Desvantagens:

1. Acoplamento forte: Pode levar a um alto acoplamento entre classes, tornando o código menos flexível e mais difícil de manter;
2. Herança múltipla não suportada: Java não permite herança múltipla de classes, o que limita a flexibilidade em certos cenários;

3. Fragilidade da hierarquia: Mudanças na classe pai podem afetar inadvertidamente todas as classes filhas, causando problemas de manutenção;
4. Complexidade: Hierarquias profundas de herança podem tornar o código complexo e difícil de compreender.

### **b) Por que a interface `Serializable` é necessária ao efetuar persistência em arquivos binários?**

A interface `Serializable` é necessária ao efetuar persistência em arquivos binários em Java porque permite que os objetos de uma classe sejam convertidos em uma sequência de bytes, tornando-os serializáveis, o que facilita sua gravação em um arquivo binário. Isso é importante para salvar e carregar objetos em sistemas de armazenamento permanente, como arquivos, bancos de dados ou redes.

### **c) Como o paradigma funcional é utilizado pela API `stream` no Java?**

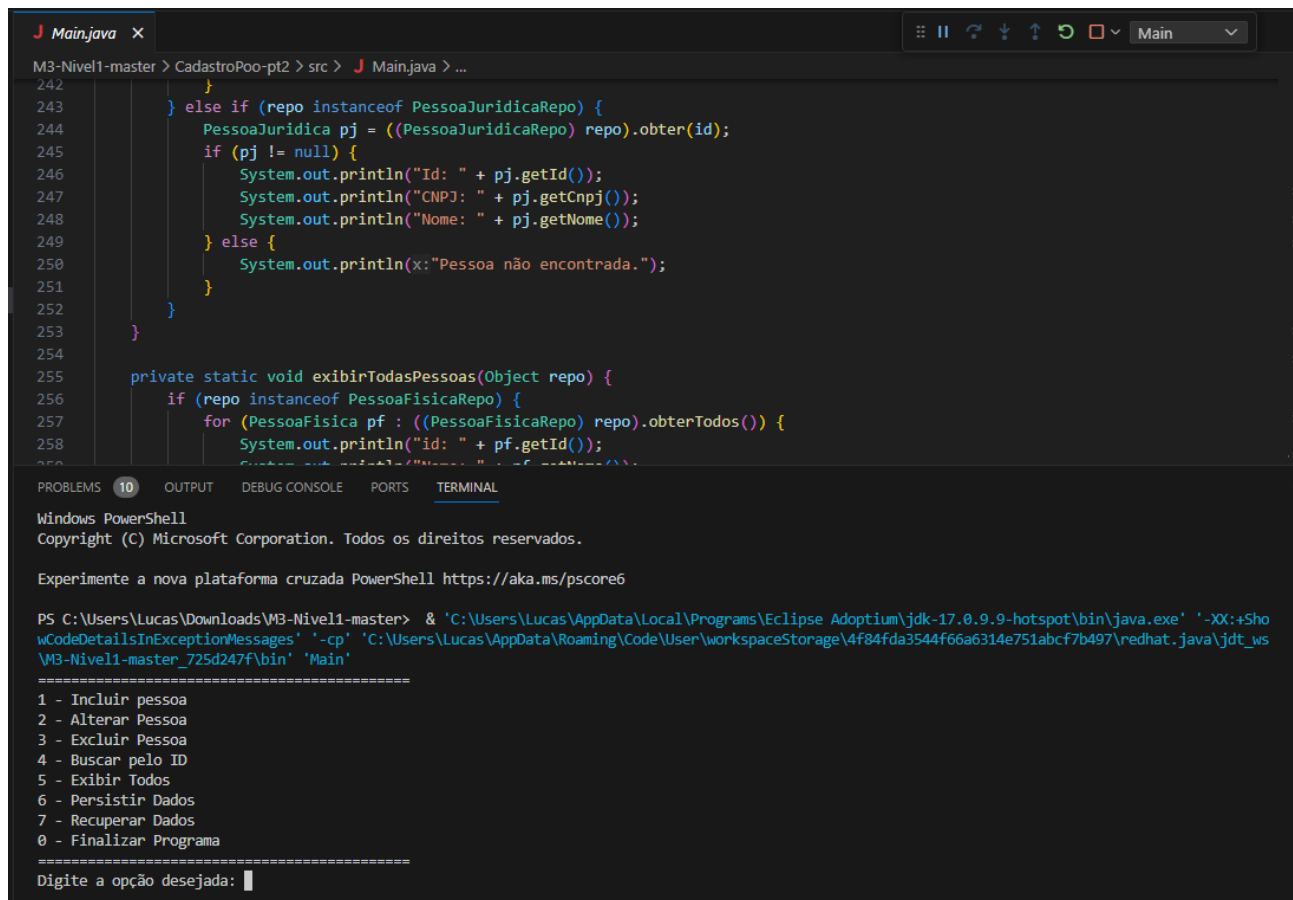
A API `Stream` no Java utiliza o paradigma funcional ao permitir operações de transformação e filtragem de dados em coleções de forma declarativa, usando funções `lambda` e expressões funcionais. Isso promove código mais conciso e legível, seguindo os princípios do paradigma funcional, como imutabilidade e composição de funções.

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

Em Java, um padrão comum de desenvolvimento para a persistência de dados em arquivos é o uso do padrão de projeto "Serialização". A serialização permite que objetos Java sejam convertidos em uma sequência de bytes e, em seguida, gravados em arquivos binários. Isso facilita a persistência e a recuperação de objetos e seus dados em arquivos, tornando-os portáteis e eficientes para armazenamento e transporte. Para implementar a serialização, é comum utilizar as interfaces `'Serializable'` e `'ObjectInputStream/ObjectOutputStream'`.

## 2º Procedimento | Criação do Cadastro em Modo Texto

### 1. Resultados da execução dos códigos PT2:



The screenshot displays the Eclipse IDE interface. The top editor shows a Java file named `Main.java` with the following code:

```
242     }
243     } else if (repo instanceof PessoaJuridicaRepo) {
244         PessoaJuridica pj = ((PessoaJuridicaRepo) repo).obter(id);
245         if (pj != null) {
246             System.out.println("Id: " + pj.getId());
247             System.out.println("CNPJ: " + pj.getCnpj());
248             System.out.println("Nome: " + pj.getNome());
249         } else {
250             System.out.println(x:"Pessoa não encontrada.");
251         }
252     }
253 }
254
255 private static void exibirTodasPessoas(Object repo) {
256     if (repo instanceof PessoaFisicaRepo) {
257         for (PessoaFisica pf : ((PessoaFisicaRepo) repo).obterTodos()) {
258             System.out.println("id: " + pf.getId());
259             System.out.println("Nome: " + pf.getNome());
260         }
261     }
262 }
```

The bottom panel shows the **TERMINAL** tab with the following output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

PS C:\Users\Lucas\Downloads\M3-Nivel1-master> & 'C:\Users\Lucas\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.9-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Lucas\AppData\Roaming\Code\User\workspaceStorage\4f84fda3544f66a6314e751abcf7b497\redhat.java\jdt_ws\M3-Nivel1-master_725d247f\bin' 'Main'
=====
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
=====
Digite a opção desejada: 
```

```
J Main.java x
M3-Nivel1-master > CadastroPoo-pt2 > src > J Main.java > ...
242 }
243 } else if (repo instanceof PessoaJuridicaRepo) {
244     PessoaJuridica pj = ((PessoaJuridicaRepo) repo).obter(id);
245     if (pj != null) {
246         System.out.println("Id: " + pj.getId());
247         System.out.println("CNPJ: " + pj.getCnpj());
248         System.out.println("Nome: " + pj.getNome());
249     } else {
250         System.out.println(x: "Pessoa não encontrada.");
251     }
252 }
253 }
254
255 private static void exibirTodasPessoas(Object repo) {
256     if (repo instanceof PessoaFisicaRepo) {
257         for (PessoaFisica pf : ((PessoaFisicaRepo) repo).obterTodos()) {
258             System.out.println("Id: " + pf.getId());
259             System.out.println("Nome: " + pf.getNome());
260         }
261     }
262 }
263 }

PROBLEMS 10 OUTPUT DEBUG CONSOLE PORTS TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

PS C:\Users\Lucas\Downloads\M3-Nivel1-master> & 'C:\Users\Lucas\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.9.9-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Lucas\AppData\Roaming\Code\User\workspaceStorage\4f84fda3544f66a6314e751abcf7b497\redhat.java\jdt_ws\M3-Nivel1-master_725d247f\bin' 'Main'
=====
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
=====
Digite a opção desejada: 1
1 - Pessoa Física | 2 - Pessoa Jurídica
1
Digite o Nome do usuário: Lucas Sobrenome
Digite o CPF do usuário: 12345678910
Digite a Idade do usuário: 24
Digite o Id do usuário: 123321
((( Pessoa inserida com sucesso. )))
=====
```

```
J Main.java x
M3-Nivel1-master > CadastroPoo-pt2 > src > J Main.java > ...
242
243     } else if (repo instanceof PessoaJuridicaRepo) {
244         PessoaJuridica pj = ((PessoaJuridicaRepo) repo).obter(id);
245         if (pj != null) {
246             System.out.println("Id: " + pj.getId());
247             System.out.println("CNPJ: " + pj.getCnpj());
248             System.out.println("Nome: " + pj.getNome());
249         } else {
250             System.out.println(x:"Pessoa não encontrada.");
251         }
252     }
253 }
254
255 private static void exibirTodasPessoas(Object repo) {
256     if (repo instanceof PessoaFisicaRepo) {
257         for (PessoaFisica pf : ((PessoaFisicaRepo) repo).obterTodos()) {
258             System.out.println("id: " + pf.getId());
259             System.out.println("Nome: " + pf.getNome());
260             System.out.println("CPF: " + pf.getCpf());
261         }
262     } else if (repo instanceof PessoaJuridicaRepo) {
263         for (PessoaJuridica pj : ((PessoaJuridicaRepo) repo).obterTodos()) {
264             System.out.println("id: " + pj.getId());
265             System.out.println("CNPJ: " + pj.getCnpj());
266             System.out.println("Nome: " + pj.getNome());
267         }
268     }
269 }
270 }

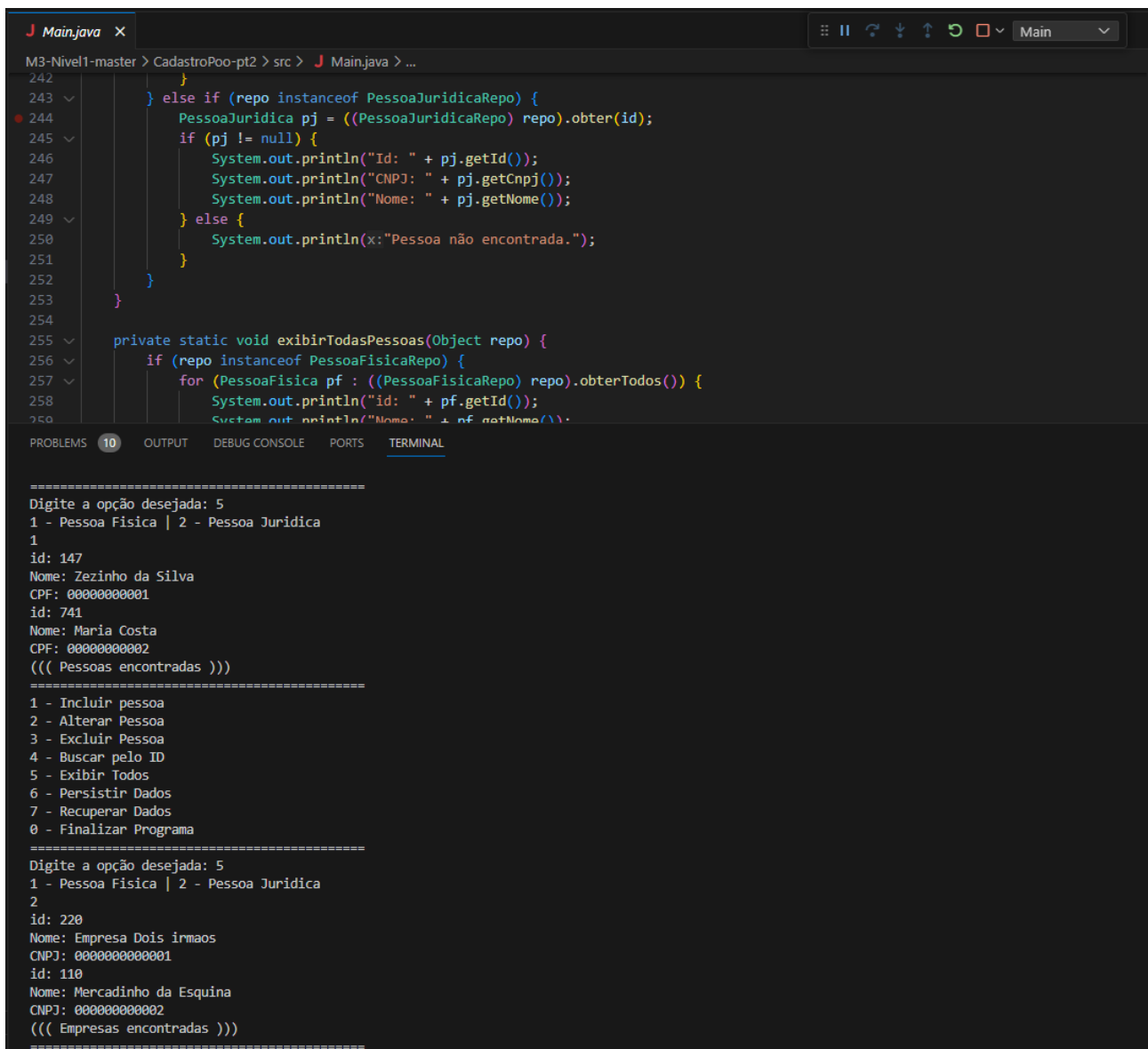
PROBLEMS 10 OUTPUT DEBUG CONSOLE PORTS TERMINAL
=====
1 - Incluir pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
Digite a opção desejada: 2
1 - Pessoa Fisica | 2 - Pessoa Juridica
1
Digite o Id que deseja alterar: 123321
CPF Antigo: 12345678910
Digite o novo CPF do usuário: 10987654321
Idade Antiga: 24
Digite a nova Idade do usuário: 28
Nome Antigo: Lucas Sobrenome
Digite o novo Nome do usuário: Joao Sobrenome
Pessoa alterada com sucesso.
((( Pessoa alterada com sucesso. )))
=====
```

```
J Main.java x
M3-Nivel1-master > CadastroPoo-pt2 > src > J Main.java > ...
242
243     } else if (repo instanceof PessoaJuridicaRepo) {
244         PessoaJuridica pj = ((PessoaJuridicaRepo) repo).obter(id);
245         if (pj != null) {
246             System.out.println("Id: " + pj.getId());
247             System.out.println("CNPJ: " + pj.getCnpj());
248             System.out.println("Nome: " + pj.getNome());
249         } else {
250             System.out.println(x:"Pessoa não encontrada.");
251         }
252     }
253 }
254
255 private static void exibirTodasPessoas(Object repo) {
256     if (repo instanceof PessoaFisicaRepo) {
257         for (PessoaFisica pf : ((PessoaFisicaRepo) repo).obterTodos()) {
258             System.out.println("id: " + pf.getId());
259             System.out.println("Nome: " + pf.getNome());
260             System.out.println("CPF: " + pf.getCpf());
261         }
262     } else if (repo instanceof PessoaJuridicaRepo) {
263         for (PessoaJuridica pj : ((PessoaJuridicaRepo) repo).obterTodos()) {
264             System.out.println("id: " + pj.getId());
265             System.out.println("CNPJ: " + pj.getCnpj());
266             System.out.println("Nome: " + pj.getNome());
267         }
268     }
269 }
270 }

PROBLEMS 10 OUTPUT DEBUG CONSOLE PORTS TERMINAL
=====
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
=====
Digite a opção desejada: 3
1 - Pessoa Fisica | 2 - Pessoa Juridica
1
Digite o Id do usuário: 123321
Pessoa excluída com sucesso.
((( Pessoa excluída com sucesso. )))
=====
```

```
J Main.java X
M3-Nivel1-master > CadastroPoo-pt2 > src > J Main.java > ...
242 }
243 } else if (repo instanceof PessoaJuridicaRepo) {
244     PessoaJuridica pj = ((PessoaJuridicaRepo) repo).obter(id);
245     if (pj != null) {
246         System.out.println("Id: " + pj.getId());
247         System.out.println("CNPJ: " + pj.getCnpj());
248         System.out.println("Nome: " + pj.getNome());
249     } else {
250         System.out.println("Pessoa não encontrada.");
251     }
252 }
253 }
254
255 private static void exibirTodasPessoas(Object repo) {
256     if (repo instanceof PessoaFisicaRepo) {
257         for (PessoaFisica pf : ((PessoaFisicaRepo) repo).obterTodos()) {
258             System.out.println("id: " + pf.getId());
259             System.out.println("Nome: " + pf.getNome());
260             System.out.println("CPF: " + pf.getCpf());
261         }
262     } else if (repo instanceof PessoaJuridicaRepo) {
263         for (PessoaJuridica pj : ((PessoaJuridicaRepo) repo).obterTodos()) {
264             System.out.println("id: " + pj.getId());
265             System.out.println("CNPJ: " + pj.getCnpj());
266             System.out.println("Nome: " + pj.getNome());
267         }
268     }
269 }
270 }
271 }

PROBLEMS 10 OUTPUT DEBUG CONSOLE PORTS TERMINAL
=====
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
=====
Digite a opção desejada: 4
1 - Pessoa Fisica | 2 - Pessoa Juridica
1
Digite o Id do usuário: 147
Id: 147
CPF: 00000000001
Idade: 18
Nome: Zezinho da Silva
((( Pessoa encontrada com sucesso. )))
=====
```



The screenshot shows an IDE with a Java file named `Main.java`. The code defines a `main` method that uses a `Scanner` to read user input. It has two main loops. The first loop handles options 1-7, which involve searching for people by ID or displaying all people. The second loop handles options 1-8, which involve adding, modifying, or deleting people. The code uses `PessoaJuridicaRepo` and `PessoaFisicaRepo` interfaces and their corresponding `Impl` classes. The output window shows the program's execution, including prompts for options and the display of person details.

```
242 }
243 } else if (repo instanceof PessoaJuridicaRepo) {
244     PessoaJuridica pj = ((PessoaJuridicaRepo) repo).obter(id);
245     if (pj != null) {
246         System.out.println("Id: " + pj.getId());
247         System.out.println("CNPJ: " + pj.getCnpj());
248         System.out.println("Nome: " + pj.getNome());
249     } else {
250         System.out.println("Pessoa não encontrada.");
251     }
252 }
253 }
254
255 private static void exibirTodasPessoas(Object repo) {
256     if (repo instanceof PessoaFisicaRepo) {
257         for (PessoaFisica pf : ((PessoaFisicaRepo) repo).obterTodos()) {
258             System.out.println("id: " + pf.getId());
259             System.out.println("Nome: " + pf.getNome());
260         }
261     }
262 }
```

=====

Digite a opção desejada: 5  
1 - Pessoa Física | 2 - Pessoa Jurídica  
1  
id: 147  
Nome: Zezinho da Silva  
CPF: 00000000001  
id: 741  
Nome: Maria Costa  
CPF: 00000000002  
((( Pessoas encontradas )))

=====

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

=====

Digite a opção desejada: 5  
1 - Pessoa Física | 2 - Pessoa Jurídica  
2  
id: 220  
Nome: Empresa Dois irmãos  
CNPJ: 000000000001  
id: 110  
Nome: Mercadinho da Esquina  
CNPJ: 000000000002  
((( Empresas encontradas )))

=====

## Análise e Conclusão:

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

Elementos estáticos em Java são associados à classe em vez de instâncias individuais dessa classe. O método "main" é declarado como estático para que possa ser chamado sem a necessidade de criar um objeto da classe. Isso permite que seja o ponto de entrada do programa, sendo invocado diretamente pela JVM (Java Virtual Machine) durante a execução, sem a necessidade de criar uma instância da classe que contém o método.

**b) Para que serve a classe Scanner?**



A classe "Scanner" em Java é usada para ler entrada de dados do usuário ou de outros fluxos, como arquivos, de maneira simples e conveniente. Ela fornece métodos para ler diferentes tipos de dados primitivos, como inteiros, números de ponto flutuante, caracteres e strings, a partir de fontes de entrada, como o teclado (System.in) ou arquivos. A classe "Scanner" é muito útil para interagir com o usuário e processar informações inseridas por ele no programa Java.

### **c) Como o uso de classes de repositório impactou na organização do código?**

O uso de classes de repositório melhora a organização do código, separando a lógica de acesso a dados da lógica de negócios, promovendo maior reutilização, testabilidade e clareza no código.