 Estácio	<p align="center"> UNIVERSIDADE ESTÁCIO DE SÁ POLO PARANGABA – FORTALEZA/CE DESENVOLVIMENTO FULL STACK - 22.3 Relatório da Missão Prática Nível 1 Mundo 3 </p>
Aluno:	Jefferson Ponte Pessoa
Professor:	Rodrigo Dias
Repositório:	https://github.com/jeffersonkako/M3-Nivel1

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:

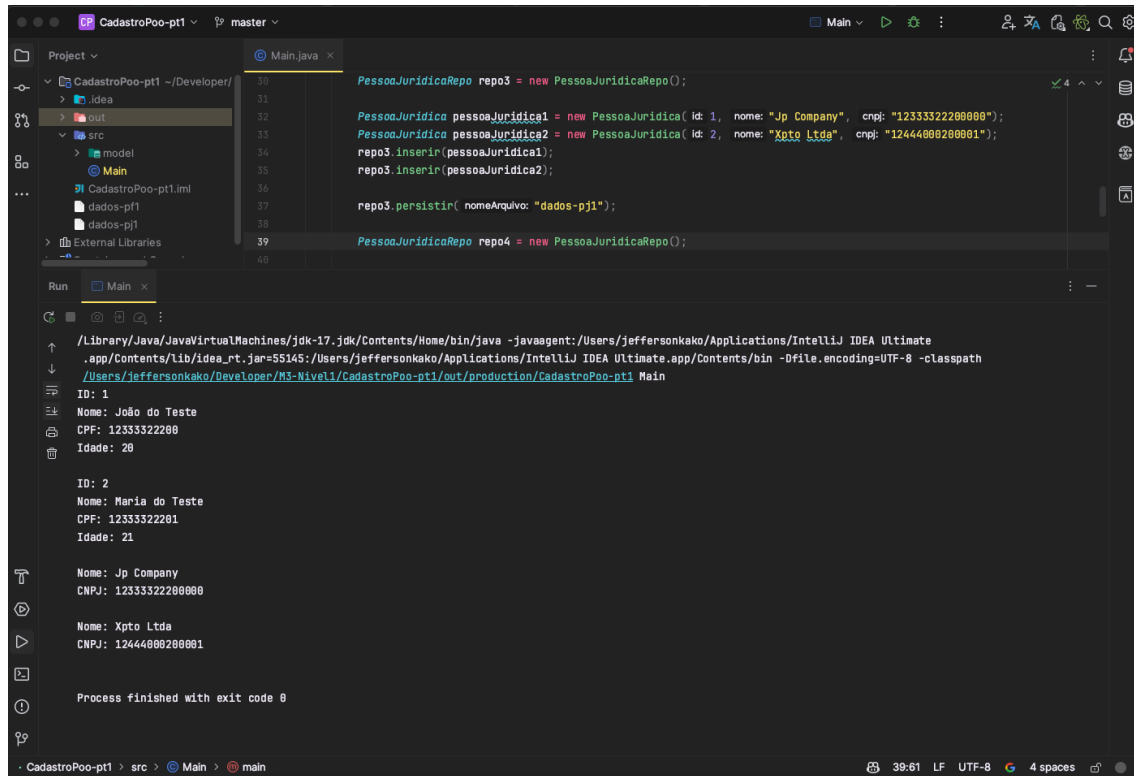
- ☐ Utilizar herança e polimorfismo na definição de entidades.
- ☐ Utilizar persistência de objetos em arquivos binários.
- ☐ Implementar uma interface cadastral em modo texto.
- ☐ Utilizar o controle de exceções da plataforma Java.
- ☐ No final do projeto, o aluno terá implementado um sistema cadastral 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/jeffersonkako/M3-Nivel1>

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:



```
30 PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
31
32 PessoaJuridica pessoaJuridica1 = new PessoaJuridica(id 1, nome: "Jp Company", cnpj: "12333322200000");
33 PessoaJuridica pessoaJuridica2 = new PessoaJuridica(id 2, nome: "Xpto Ltda", cnpj: "12444000200001");
34 repo3.inserir(pessoaJuridica1);
35 repo3.inserir(pessoaJuridica2);
36
37 repo3.persistir( nomeArquivo: "dados-pj1");
38
39 PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
40
```

Run: Main

/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Users/jeffersonkako/Applications/IntelliJ IDEA Ultimate.app/Contents/lib/idea_rt.jar=S5145:/Users/jeffersonkako/Applications/IntelliJ IDEA Ultimate.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/jeffersonkako/Developer/M3-Nivel1/CadastroPoo-pt1/out/production/CadastroPoo-pt1 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

Process finished with exit code 0

Análise e Conclusão:

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

Vantagens:

1. Reutilização de código: Permite herdar atributos e métodos de uma classe pai, economizando esforço na reescrita.
2. Extensibilidade: Facilita a criação de novas classes baseadas em classes existentes, 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?

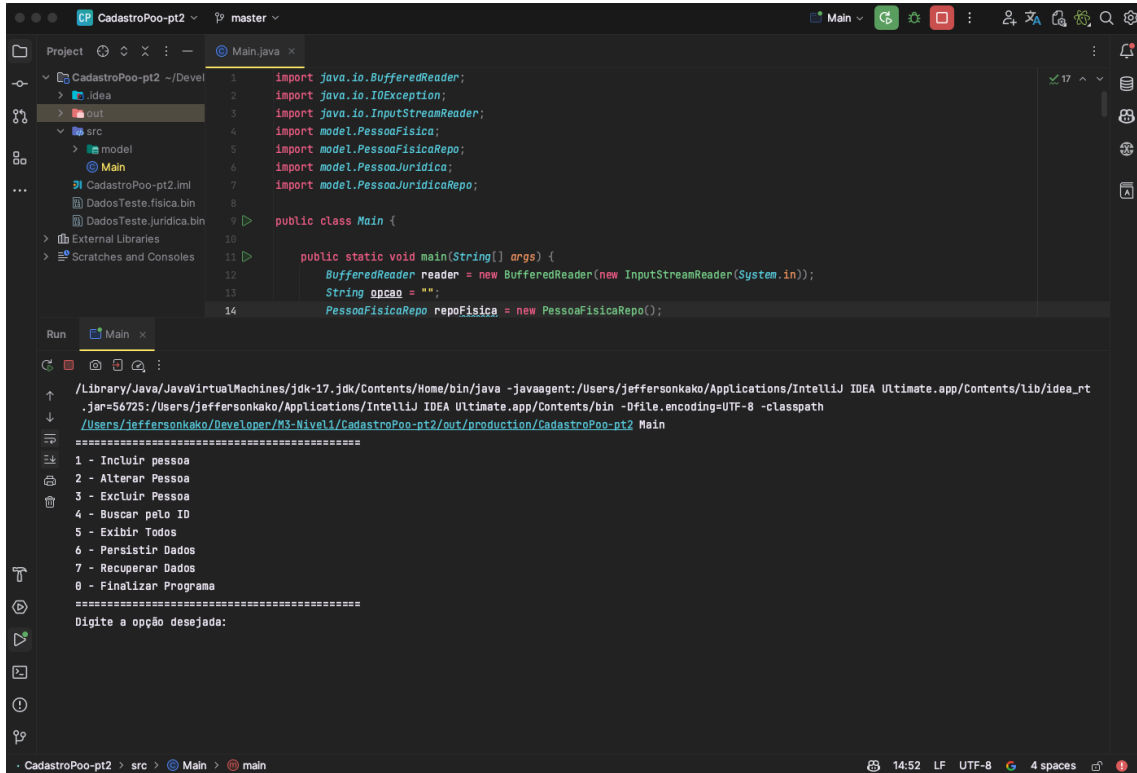
A API Stream no Java utiliza o paradigma funcional ao permitir operações de transformação e filtragem de dados em coleções de forma declarativa, usando funções lambda e expressões funcionais. Isso promove código mais conciso e legível, seguindo os princípios do paradigma funcional, como imutabilidade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

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

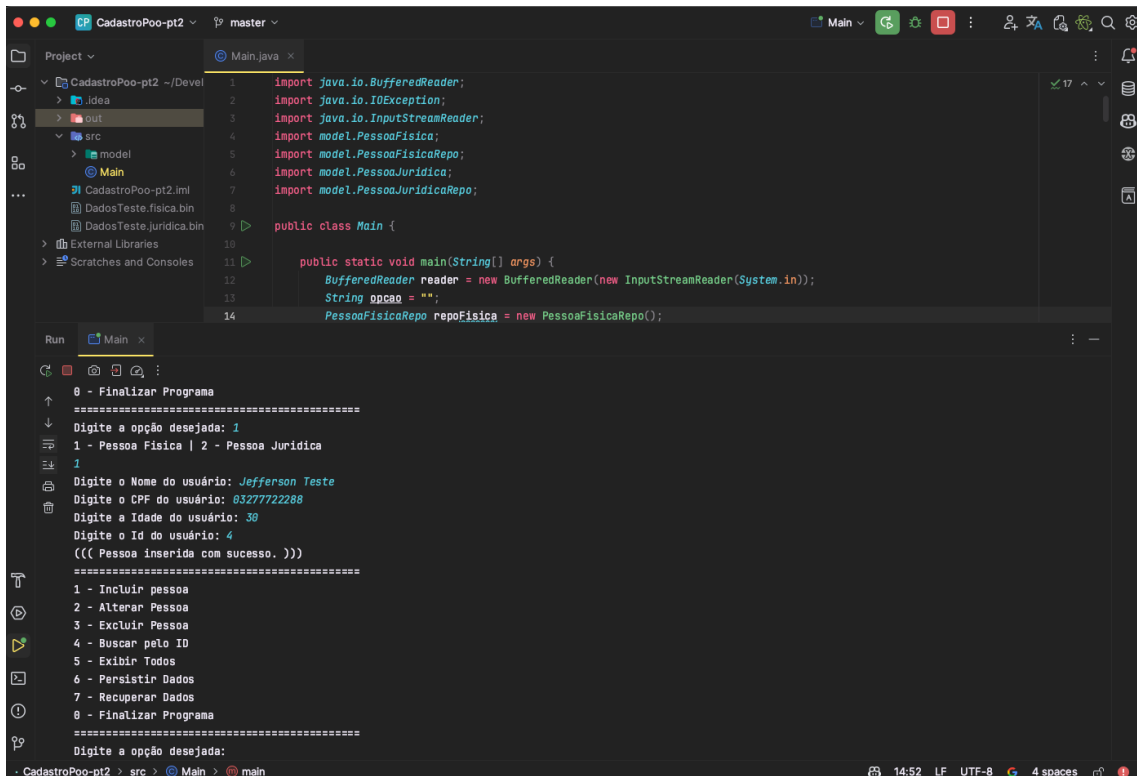
Em Java, um padrão comum de desenvolvimento para a persistência de dados em arquivos é o uso do padrão de projeto "Serialização". A serialização permite que objetos Java sejam convertidos em uma sequência de bytes e, em seguida, gravados em arquivos binários. Isso facilita a persistência e a recuperação 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 ObjectOutputStream/ObjectInputStream.

👉 2º Procedimento | Criação do Cadastro em Modo Texto

1. Resultados da execução dos códigos PT2:



```
Project: CadastroPoo-pt2 ~/Devel
Main.java
1 import java.io.BufferedReader;
2 import java.io.IOException;
3 import java.io.InputStreamReader;
4 import model.PessoaFisica;
5 import model.PessoaFisicaRepo;
6 import model.PessoaJuridica;
7 import model.PessoaJuridicaRepo;
8
9 public class Main {
10
11     public static void main(String[] args) {
12         BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
13         String opcao = "";
14         PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
15
16     }
17 }
18
19 Run: Main
20
21 /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Users/jeffersonkako/Applications/IntelliJ IDEA Ultimate.app/Contents/lib/idea_rt.jar=56725:/Users/jeffersonkako/Applications/IntelliJ IDEA Ultimate.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/jeffersonkako/Developer/MS-Nivel1/CadastroPoo-pt2/out/production/CadastroPoo-pt2 Main
22
23 =====
24 1 - Incluir pessoa
25 2 - Alterar Pessoa
26 3 - Excluir Pessoa
27 4 - Buscar pelo ID
28 5 - Exibir Todos
29 6 - Persistir Dados
30 7 - Recuperar Dados
31 8 - Finalizar Programa
32 =====
33 Digite a opção desejada:
```



```
Project: CadastroPoo-pt2 ~/Devel
Main.java
1 import java.io.BufferedReader;
2 import java.io.IOException;
3 import java.io.InputStreamReader;
4 import model.PessoaFisica;
5 import model.PessoaFisicaRepo;
6 import model.PessoaJuridica;
7 import model.PessoaJuridicaRepo;
8
9 public class Main {
10
11     public static void main(String[] args) {
12         BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
13         String opcao = "";
14         PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
15
16     }
17 }
18
19 Run: Main
20
21 8 - Finalizar Programa
22 =====
23 Digite a opção desejada: 1
24 1 - Pessoa Fisica | 2 - Pessoa Juridica
25 1
26 Digite o Nome do usuário: Jefferson Teste
27 Digite o CPF do usuário: 0327772288
28 Digite a Idade do usuário: 30
29 Digite o Id do usuário: 4
30 ((( Pessoa inserida com sucesso. )))
31 =====
32 1 - Incluir pessoa
33 2 - Alterar Pessoa
34 3 - Excluir Pessoa
35 4 - Buscar pelo ID
36 5 - Exibir Todos
37 6 - Persistir Dados
38 7 - Recuperar Dados
39 8 - Finalizar Programa
40 =====
41 Digite a opção desejada:
```

```
Project: CadastroPoo-pt2 - master
Main.java
1 import java.io.BufferedReader;
2 import java.io.IOException;
3 import java.io.InputStreamReader;
4 import model.PessoaFisica;
5 import model.PessoaFisicaRepo;
6 import model.PessoaJuridica;
7 import model.PessoaJuridicaRepo;
8
9 public class Main {
10
11     public static void main(String[] args) {
12         BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
13         String opcao = "";
14         PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
15
16         while (true) {
17             System.out.println("1 - Incluir pessoa");
18             System.out.println("2 - Alterar Pessoa");
19             System.out.println("3 - Excluir Pessoa");
20             System.out.println("4 - Buscar pelo ID");
21             System.out.println("5 - Exibir Todos");
22             System.out.println("6 - Persistir Dados");
23             System.out.println("7 - Recuperar Dados");
24             System.out.println("8 - Finalizar Programa");
25             System.out.println("Digite a opção desejada: ");
26
27             int op = Integer.parseInt(reader.readLine());
28
29             switch (op) {
30                 case 1:
31                     System.out.println("Digite o Id que deseja alterar: ");
32                     String id = reader.readLine();
33                     System.out.println("CPF Antigo: 03277722288");
34                     System.out.println("Digite o novo CPF do usuário: ");
35                     String cpf = reader.readLine();
36                     System.out.println("Idade Antiga: 30");
37                     System.out.println("Digite a nova Idade do usuário: ");
38                     String idade = reader.readLine();
39                     System.out.println("Nome Antigo: Jefferson Teste");
40                     System.out.println("Digite o novo Nome do usuário: ");
41                     String nome = reader.readLine();
42                     PessoaFisica pessoa = new PessoaFisica(id, cpf, idade, nome);
43                     repoFisica.alterar(pessoa);
44                     System.out.println("Pessoa alterada com sucesso.");
45                     System.out.println("((( Pessoa alterada com sucesso. )))");
46                     break;
47                 case 2:
48                     System.out.println("Digite o Id que deseja alterar: ");
49                     String id2 = reader.readLine();
50                     System.out.println("CPF Antigo: 03277722288");
51                     System.out.println("Digite o novo CPF do usuário: ");
52                     String cpf2 = reader.readLine();
53                     System.out.println("Idade Antiga: 30");
54                     System.out.println("Digite a nova Idade do usuário: ");
55                     String idade2 = reader.readLine();
56                     System.out.println("Nome Antigo: Jefferson Teste");
57                     System.out.println("Digite o novo Nome do usuário: ");
58                     String nome2 = reader.readLine();
59                     PessoaFisica pessoa2 = new PessoaFisica(id2, cpf2, idade2, nome2);
60                     repoFisica.alterar(pessoa2);
61                     System.out.println("Pessoa alterada com sucesso.");
62                     System.out.println("((( Pessoa alterada com sucesso. )))");
63                     break;
64                 case 3:
65                     System.out.println("Digite o Id que deseja alterar: ");
66                     String id3 = reader.readLine();
67                     System.out.println("CPF Antigo: 03277722288");
68                     System.out.println("Digite o novo CPF do usuário: ");
69                     String cpf3 = reader.readLine();
70                     System.out.println("Idade Antiga: 30");
71                     System.out.println("Digite a nova Idade do usuário: ");
72                     String idade3 = reader.readLine();
73                     System.out.println("Nome Antigo: Jefferson Teste");
74                     System.out.println("Digite o novo Nome do usuário: ");
75                     String nome3 = reader.readLine();
76                     PessoaFisica pessoa3 = new PessoaFisica(id3, cpf3, idade3, nome3);
77                     repoFisica.alterar(pessoa3);
78                     System.out.println("Pessoa alterada com sucesso.");
79                     System.out.println("((( Pessoa alterada com sucesso. )))");
80                     break;
81                 case 4:
82                     System.out.println("Digite o Id que deseja alterar: ");
83                     String id4 = reader.readLine();
84                     System.out.println("CPF Antigo: 03277722288");
85                     System.out.println("Digite o novo CPF do usuário: ");
86                     String cpf4 = reader.readLine();
87                     System.out.println("Idade Antiga: 30");
88                     System.out.println("Digite a nova Idade do usuário: ");
89                     String idade4 = reader.readLine();
90                     System.out.println("Nome Antigo: Jefferson Teste");
91                     System.out.println("Digite o novo Nome do usuário: ");
92                     String nome4 = reader.readLine();
93                     PessoaFisica pessoa4 = new PessoaFisica(id4, cpf4, idade4, nome4);
94                     repoFisica.alterar(pessoa4);
95                     System.out.println("Pessoa alterada com sucesso.");
96                     System.out.println("((( Pessoa alterada com sucesso. )))");
97                     break;
98                 case 5:
99                     System.out.println("Exibindo todos os dados:");
100                    repoFisica.exibirTodos();
101                    break;
102                 case 6:
103                    repoFisica.persistirDados();
104                    break;
105                 case 7:
106                    repoFisica.recuperarDados();
107                    break;
108                 case 8:
109                    System.out.println("Finalizando programa...");
110                    break;
111            }
112        }
113    }
114}
```

Run: Main

1 - Incluir pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
8 - Finalizar Programa
Digite a opção desejada: 4
CPF Antigo: 03277722288
Digite o novo CPF do usuário: 0333441100
Idade Antiga: 30
Digite a nova Idade do usuário: 32
Nome Antigo: Jefferson Teste
Digite o novo Nome do usuário: Jefferson teste 2
Pessoa alterada com sucesso.
(((Pessoa alterada com sucesso.)))
=====

```
Project: CadastroPoo-pt2 - master
Main.java
1 import java.io.BufferedReader;
2 import java.io.IOException;
3 import java.io.InputStreamReader;
4 import model.PessoaFisica;
5 import model.PessoaFisicaRepo;
6 import model.PessoaJuridica;
7 import model.PessoaJuridicaRepo;
8
9 public class Main {
10
11     public static void main(String[] args) {
12         BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
13         String opcao = "";
14         PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
15
16         while (true) {
17             System.out.println("1 - Incluir pessoa");
18             System.out.println("2 - Alterar Pessoa");
19             System.out.println("3 - Excluir Pessoa");
20             System.out.println("4 - Buscar pelo ID");
21             System.out.println("5 - Exibir Todos");
22             System.out.println("6 - Persistir Dados");
23             System.out.println("7 - Recuperar Dados");
24             System.out.println("8 - Finalizar Programa");
25             System.out.println("Digite a opção desejada: ");
26
27             int op = Integer.parseInt(reader.readLine());
28
29             switch (op) {
30                 case 1:
31                     System.out.println("Digite o Id que deseja alterar: ");
32                     String id = reader.readLine();
33                     System.out.println("CPF Antigo: 03277722288");
34                     System.out.println("Digite o novo CPF do usuário: ");
35                     String cpf = reader.readLine();
36                     System.out.println("Idade Antiga: 30");
37                     System.out.println("Digite a nova Idade do usuário: ");
38                     String idade = reader.readLine();
39                     System.out.println("Nome Antigo: Jefferson Teste");
40                     System.out.println("Digite o novo Nome do usuário: ");
41                     String nome = reader.readLine();
42                     PessoaFisica pessoa = new PessoaFisica(id, cpf, idade, nome);
43                     repoFisica.alterar(pessoa);
44                     System.out.println("Pessoa alterada com sucesso.");
45                     System.out.println("((( Pessoa alterada com sucesso. )))");
46                     break;
47                 case 2:
48                     System.out.println("Digite o Id que deseja alterar: ");
49                     String id2 = reader.readLine();
50                     System.out.println("CPF Antigo: 03277722288");
51                     System.out.println("Digite o novo CPF do usuário: ");
52                     String cpf2 = reader.readLine();
53                     System.out.println("Idade Antiga: 30");
54                     System.out.println("Digite a nova Idade do usuário: ");
55                     String idade2 = reader.readLine();
56                     System.out.println("Nome Antigo: Jefferson Teste");
57                     System.out.println("Digite o novo Nome do usuário: ");
58                     String nome2 = reader.readLine();
59                     PessoaFisica pessoa2 = new PessoaFisica(id2, cpf2, idade2, nome2);
60                     repoFisica.alterar(pessoa2);
61                     System.out.println("Pessoa alterada com sucesso.");
62                     System.out.println("((( Pessoa alterada com sucesso. )))");
63                     break;
64                 case 3:
65                     System.out.println("Digite o Id que deseja alterar: ");
66                     String id3 = reader.readLine();
67                     System.out.println("CPF Antigo: 03277722288");
68                     System.out.println("Digite o novo CPF do usuário: ");
69                     String cpf3 = reader.readLine();
70                     System.out.println("Idade Antiga: 30");
71                     System.out.println("Digite a nova Idade do usuário: ");
72                     String idade3 = reader.readLine();
73                     System.out.println("Nome Antigo: Jefferson Teste");
74                     System.out.println("Digite o novo Nome do usuário: ");
75                     String nome3 = reader.readLine();
76                     PessoaFisica pessoa3 = new PessoaFisica(id3, cpf3, idade3, nome3);
77                     repoFisica.alterar(pessoa3);
78                     System.out.println("Pessoa alterada com sucesso.");
79                     System.out.println("((( Pessoa alterada com sucesso. )))");
80                     break;
81                 case 4:
82                     System.out.println("Digite o Id que deseja alterar: ");
83                     String id4 = reader.readLine();
84                     System.out.println("CPF Antigo: 03277722288");
85                     System.out.println("Digite o novo CPF do usuário: ");
86                     String cpf4 = reader.readLine();
87                     System.out.println("Idade Antiga: 30");
88                     System.out.println("Digite a nova Idade do usuário: ");
89                     String idade4 = reader.readLine();
90                     System.out.println("Nome Antigo: Jefferson Teste");
91                     System.out.println("Digite o novo Nome do usuário: ");
92                     String nome4 = reader.readLine();
93                     PessoaFisica pessoa4 = new PessoaFisica(id4, cpf4, idade4, nome4);
94                     repoFisica.alterar(pessoa4);
95                     System.out.println("Pessoa alterada com sucesso.");
96                     System.out.println("((( Pessoa alterada com sucesso. )))");
97                     break;
98                 case 5:
99                     System.out.println("Exibindo todos os dados:");
100                    repoFisica.exibirTodos();
101                    break;
102                 case 6:
103                    repoFisica.persistirDados();
104                    break;
105                 case 7:
106                    repoFisica.recuperarDados();
107                    break;
108                 case 8:
109                    System.out.println("Finalizando programa...");
110                    break;
111            }
112        }
113    }
114}
```

Run: Main

6 - Persistir Dados
7 - Recuperar Dados
8 - Finalizar Programa
Digite a opção desejada: 3
1 - Pessoa Fisica | 2 - Pessoa Juridica
1
Digite o Id do usuário: 4
Pessoa excluída com sucesso.
(((Pessoa excluída com sucesso.)))
=====

```
Project: CadastroPoo-pt2 ~/Devel
Main.java
151
152
153
154
155
156
157
158
159
160
161
162
163
164

    }
    break;
} catch (IOException e) {
    System.out.println(" [X] Erro de entrada/saída: " + e.getMessage());
}

private static PessoaFisica lerDadosPessoaFisica(BufferedReader reader) throws IOException { 1 usage
    System.out.print("Digite o Nome do usuário: ");
    String nome = reader.readLine();
    System.out.print("Digite o CPF do usuário: ");
    String cpf = reader.readLine();
    System.out.print("Digite a Idade do usuário: ");
}

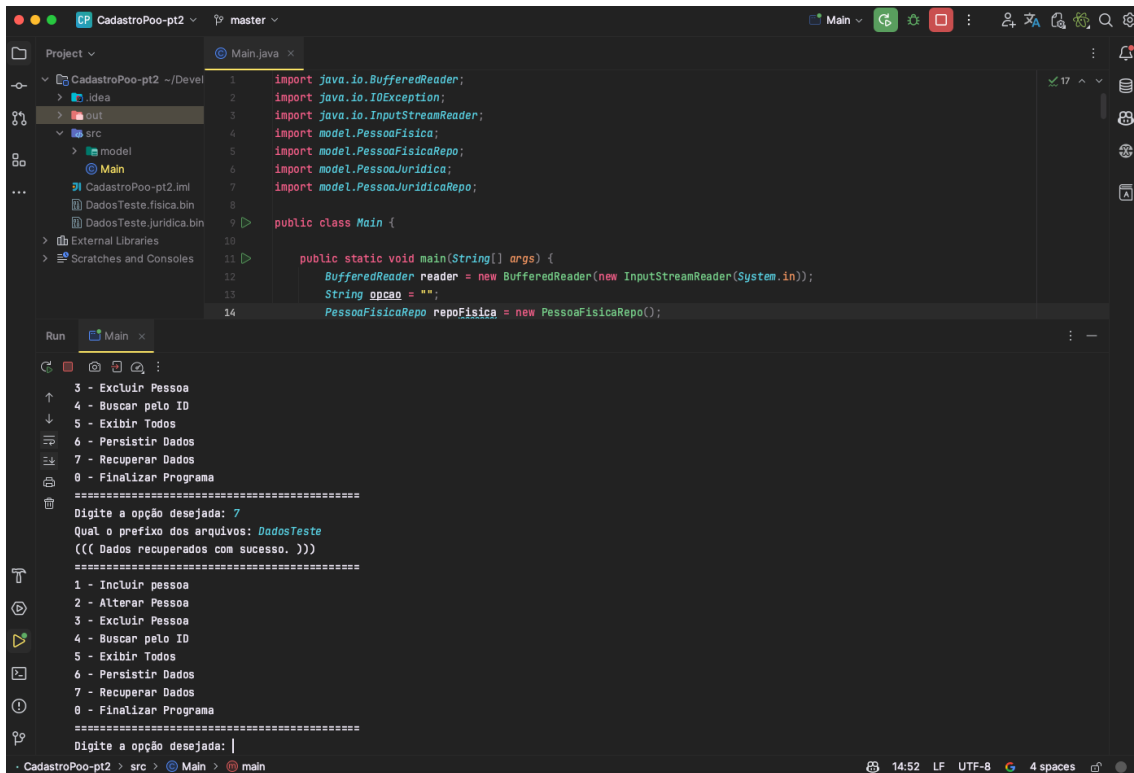
Run: Main
=====
Digite a opção desejada: 4
1 - Pessoa Física | 2 - Pessoa Jurídica
1
Digite o Id do usuário: 2
Id: 2
CPF: 22233322211
Idade: 2
Nome: Toim da Costa
((( Pessoa encontrada com sucesso. )))
=====
1 - Incluir pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
8 - Finalizar Programa
=====
Digite a opção desejada: |
```

```
Project: CadastroPoo-pt2 ~/Devel
Main.java
151
152
153
154
155
156
157
158
159
160
161
162
163
164

    }
    break;
} catch (IOException e) {
    System.out.println(" [X] Erro de entrada/saída: " + e.getMessage());
}

private static PessoaFisica lerDadosPessoaFisica(BufferedReader reader) throws IOException { 1 usage
    System.out.print("Digite o Nome do usuário: ");
    String nome = reader.readLine();
    System.out.print("Digite o CPF do usuário: ");
    String cpf = reader.readLine();
    System.out.print("Digite a Idade do usuário: ");
}

Run: Main
0 - Finalizar Programa
=====
Digite a opção desejada: 5
1 - Pessoa Física | 2 - Pessoa Jurídica
1
id: 1
Nome: Joao da silva
CPF: 11122233344
id: 2
Nome: Toim da Costa
CPF: 22233322211
id: 3
Nome: Tereza do Teste
CPF: 33322244411
((( Pessoas encontradas )))
=====
1 - Incluir pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
```



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.