



# Estácio

**ALUNO: FILIPE MACIEL DE SOUZA ANDRADE**

**TURMA: 9001**

**CURSO: DESENVOLVIMENTO FULL STACK**

**DISCIPLINA: INICIANDO O CAMINHO PELO JAVA**

**MATRÍCULA: 2023.04.65842-1**

**POLO PORTO ALEGRE – RS**

**3º PERIODO**

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

### **Objetivos da Prática:**


1. Utilizar herança e polimorfismo na definição de entidades.
2. Utilizar persistência de objetos em arquivos binários.
3. Implementar uma interface cadastral em modo texto.
4. Utilizar o controle de exceções da plataforma Java.
5. 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.

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

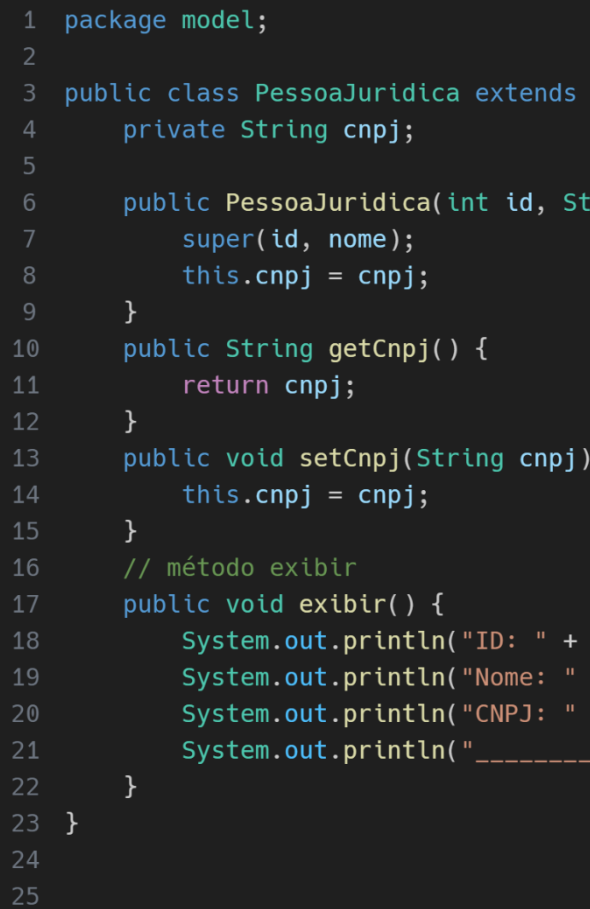
O primeiro procedimento do trabalho é a implementação das classes **Pessoa**, **PessoaFisica** e **PessoaJuridica**.



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



```
1 package model;
2
3 public class PessoaFisica extends Pessoa {
4     private String cpf;
5     private int idade;
6
7     public PessoaFisica(int id, String nome, String cpf, int idade) {
8         super(id, nome);
9         this.cpf = cpf;
10        this.idade = idade;
11    }
12    public String getCpf() {
13        return cpf;
14    }
15    public void setCpf(String cpf) {
16        this.cpf = cpf;
17    }
18    public int getIdade() {
19        return idade;
20    }
21    public void setIdade(int idade) {
22        this.idade = idade;
23    }
24    // método exibir
25    public void exibir() {
26        System.out.println("ID: " + getId());
27        System.out.println("Nome: " + getNome());
28        System.out.println("CPF: " + cpf);
29        System.out.println("_____" + "\n");
30    }
31 }
```



```
1 package model;
2
3 public class PessoaJuridica extends Pessoa {
4     private String cnpj;
5
6     public PessoaJuridica(int id, String nome, String cnpj) {
7         super(id, nome);
8         this.cnpj = cnpj;
9     }
10    public String getCnpj() {
11        return cnpj;
12    }
13    public void setCnpj(String cnpj) {
14        this.cnpj = cnpj;
15    }
16    // método exibir
17    public void exibir() {
18        System.out.println("ID: " + getId());
19        System.out.println("Nome: " + getNome());
20        System.out.println("CNPJ: " + cnpj);
21        System.out.println("_____ " + "\n");
22    }
23 }
24
25
```

Depois é criado os gerenciadores responsáveis por controlar a persistência dos dados.

```

1 package model;
2
3 import java.io.FileInputStream;
4 import java.io.FileOutputStream;
5 import java.io.IOException;
6 import java.io.ObjectInputStream;
7 import java.io.ObjectOutputStream;
8 import java.util.ArrayList;
9
10 public class PessoaFisicaRepo {
11     private ArrayList<PessoaFisica> pessoasFisicas = new ArrayList<>();
12     public void inserir(PessoaFisica pessoa) {
13         pessoasFisicas.add(pessoa);
14     }
15     public void alterar(PessoaFisica pessoa) {
16         for(int i = 0; i < pessoasFisicas.size(); i++) {
17             if(pessoasFisicas.get(i).getId() == pessoa.getId()) {
18                 pessoasFisicas.set(i, pessoa);
19                 return;
20             }
21         }
22     }
23     public boolean excluir(int id) {
24         for(int i = 0; i < pessoasFisicas.size(); i++) {
25             if(pessoasFisicas.get(i).getId() == id) {
26                 pessoasFisicas.remove(i);
27                 return true;
28             }
29         }
30         return false;
31     }
32     public PessoaFisica obter(int id) {
33         for(PessoaFisica pessoa : pessoasFisicas) {
34             if(pessoa.getId() == id) {
35                 return pessoa;
36             }
37         }
38         return null;
39     }
40     public ArrayList<PessoaFisica> obterTodos(){
41         return pessoasFisicas;
42     }
43     public void persistir(String nomeArquivo) throws IOException{
44         try(ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
45             out.writeObject(pessoasFisicas);
46         }
47     }
48     @SuppressWarnings("unchecked")
49     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
50         try(ObjectInputStream in = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
51             pessoasFisicas = (ArrayList<PessoaFisica>) in.readObject();
52         }
53     }
54 }
55

```

```

1 package model;
2
3 import java.io.FileInputStream;
4 import java.io.FileOutputStream;
5 import java.io.IOException;
6 import java.io.ObjectOutputStream;
7 import java.util.ArrayList;
8 import java.io.ObjectInputStream;
9
10 public class PessoaJuridicaRepo {
11     private ArrayList<PessoaJuridica> pessoasJuridicas = new ArrayList<>();
12
13     public void inserir(PessoaJuridica pessoa) {
14         pessoasJuridicas.add(pessoa);
15     }
16
17     public void alterar(PessoaJuridica pessoa) {
18         for(int i = 0; i < pessoasJuridicas.size(); i++) {
19             if(pessoasJuridicas.get(i).getId() == pessoa.getId()) {
20                 pessoasJuridicas.set(i, pessoa);
21                 return;
22             }
23         }
24     }
25
26     public boolean excluir(int id) {
27         for(int i = 0; i < pessoasJuridicas.size(); i++) {
28             if(pessoasJuridicas.get(i).getId() == id) {
29                 pessoasJuridicas.remove(i);
30                 return true;
31             }
32         }
33         return false;
34     }
35
36     public PessoaJuridica obter(int id) {
37         for(PessoaJuridica pessoa : pessoasJuridicas) {
38             if(pessoa.getId() == id) {
39                 return pessoa;
40             }
41         }
42         return null;
43     }
44
45     public ArrayList<PessoaJuridica> obterTodos(){
46         return pessoasJuridicas;
47     }
48
49     public void persistir(String nomeArquivo) throws IOException{
50         try (ObjectOutputStream output = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
51             output.writeObject(pessoasJuridicas);
52         }
53     }
54
55     @SuppressWarnings("unchecked")
56     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException{
57         try(ObjectInputStream input = new ObjectInputStream(new FileInputStream(nomeArquivo))){
58             pessoasJuridicas = (ArrayList<PessoaJuridica>) input.readObject();
59         }
60     }
61 }

```

Ao final é implementado os testes de persistência no método **Main**.

```
1 package model;
2
3 public class CadastroP00 {
4     public static void main(String[] args) throws Exception{
5         // pessoas fisicas
6         String pessoasDados = "pessoas.dat";
7
8         try {
9             PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
10            PessoaFisica pessoa1 = new PessoaFisica(1, "Ana", "11111111111", 25);
11            PessoaFisica pessoa2 = new PessoaFisica(2, "Carlos", "22222222222", 52);
12
13            repo1.inserir(pessoa1);
14            repo1.inserir(pessoa2);
15            repo1.persistir(pessoasDados);
16            System.out.println("Dados de Pessoa Fisica Armazenados.");
17
18            PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
19            repo2.recuperar(pessoasDados);
20            System.out.println("Dados de Pessoa Fisica Recuperados.");
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            }
28        } catch (Exception e) {
29            System.out.println("Erro: " + e.getMessage());
30        }
31        // pessoas juridicas
32        String empresasDados = "empresas.dat";
33
34        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
35
36        PessoaJuridica empresa1 = new PessoaJuridica(1, "XPT0 Sales", "33333333333333");
37        PessoaJuridica empresa2 = new PessoaJuridica(2, "XPT0 Solutions", "44444444444444");
38
39        repo3.inserir(empresa1);
40        repo3.inserir(empresa2);
41
42        try {
43            repo3.persistir(empresasDados);
44            System.out.println("Dados de Pessoa Juridica Armazenados");
45
46            PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
47            repo4.recuperar(empresasDados);
48            System.out.println("Dados de Pessoa Juridica Recuperados");
49
50            for (PessoaJuridica empresa : repo4.obterTodos()) {
51                System.out.println("Id: " + empresa.getId());
52                System.out.println("Nome: " + empresa.getNome());
53                System.out.println("CNPJ: " + empresa.getCnpj());
54            }
55        } catch (Exception e){
56            System.out.println("Erro: " + e.getMessage());
57        }
58    }
59 }
60
```

```
PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE  PORTS  SEARCH  GITLENS

Id: 1
Nome Ana
CPF: 111111111111
Idade: 25
Id: 2
Nome Carlos
CPF: 222222222222
Idade: 52
Dados de Pessoa Juridica Armazenados
Dados de Pessoa Juridica Recuperados
Id: 1
Nome: XPT0 Sales
CNPJ: 33333333333333
Id: 2
Nome: XPT0 Solutions
CNPJ: 44444444444444
PS C:\Users\filip\Documents\GitHub\CadastroP00> █
```

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

R: Vantagens:

- 1.1 Reutilização – Com a herança é possível que um classe herde atributos e métodos de outra classe, reduzindo duplicações e aumentando a legibilidade. Também deixa o código em uma estrutura hierárquica clara que facilita a manutenção do projeto.
- 1.2 Hierarquia – Cria relações naturais entre as classes, ajudando a organizar as classes em uma hierarquia.
- 1.3 Polimorfismo – Classes filhas podem sobrescrever métodos herdados da classe pai, facilitando a criação de um código mais flexível e adaptável a diferentes situações.

Desvantagens:

- 1.1 Rigidez – Mudar a classe pai pode afetar as classes filhas, o que faz com que a adaptação em diversas partes do código. Além disso as classes filhas acabam ficando muito vinculadas a classe pai, o que dificulta a reutilização em outros contextos.
- 1.2 Complexidade – As estruturas hierárquicas complexas com muitas classes filhas acabam tornando o código difícil de entender e de se manter. Classes filhas que dependem muito da classe pai acabam com conflitos de herança, o que dificulta a modificação independente.
- 1.3 Excesso – O uso da herança em excesso acaba por criar uma hierarquia complexa e difícil de gerenciar, o que pode ocasionar na dificuldade de reutilizar o código.

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



R: A interface Serializable é necessária para salvar objetos em binários e depois recuperá-los. Ela permite que a JVM converta os objetos em formato binário.

**3. Como o paradigma funcional é utilizado pela API Stream no Java?**

R: A API Stream do Java incorpora diversos princípios da programação funcional, como operações de filtragem, redução e mapeamento. Permitindo melhorar a legibilidade e a manutenção do código.

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

R: Dentre um dos mais populares se destaca a serialização de objetos, que foi usada no trabalho com a interface Serializable.

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

O segundo procedimento vai ser a criação de uma interface para o usuário executar as funcionalidades do programa.

```

1 package model;
2
3 import java.io.IOException;
4 import java.util.Scanner;
5 import java.util.logging.Level;
6 import java.util.logging.Logger;
7
8 public class CadastroPOODois {
9
10     public static void main(String[] args) {
11         PessoaFisicaRepo repoPessoaFisica = new PessoaFisicaRepo();
12         PessoaJuridicaRepo repoPessoaJuridica = new PessoaJuridicaRepo();
13
14         String tipoPessoa;
15         String prefixo;
16         int opcao;
17         Integer id;
18         do {
19             Scanner scanner = new Scanner(System.in);
20
21             System.out.println("=====");
22             System.out.println("1 - Incluir Pessoa");
23             System.out.println("2 - Alterar Pessoa");
24             System.out.println("3 - Excluir Pessoa");
25             System.out.println("4 - Buscar pelo Id");
26             System.out.println("5 - Exibir Todos");
27             System.out.println("6 - Persistir Dados");
28             System.out.println("7 - Recuperar Dados");
29             System.out.println("0 - Finalizar Programa");
30             System.out.println("=====");
31             opcao = scanner.nextInt();
32
33             switch (opcao) {
34                 case 1:
35                     tipoPessoa = fisicaOUjuridica(scanner);
36                     switch (tipoPessoa) {
37                         case "F":
38                             try {
39                                 PessoaFisica pessoaFisica = lerDadosFisica(scanner);
40                                 repoPessoaFisica.inserir(pessoaFisica);
41                             } catch (Exception e) {
42                                 System.out.println(e.getMessage());
43                             }
44                             break;
45                         case "J":
46                             try {
47                                 PessoaJuridica pessoaJuridica = lerDadosJuridica(scanner);
48                                 repoPessoaJuridica.inserir(pessoaJuridica);
49                             } catch (Exception e) {
50                                 System.out.println(e.getMessage());
51                             }
52                             break;
53                         default:
54                             alertaOpcaoInvalida();
55                             break;
56                     }
57                     break;
58
59                 case 2:
60                     tipoPessoa = fisicaOUjuridica(scanner);
61                     id = getId(scanner);
62                     if (id == null) {
63                         alertaOpcaoInvalida();
64                         break;
65                     }
66                     switch (tipoPessoa) {
67                         case "F":
68                             PessoaFisica pessoaFisica = repoPessoaFisica.obter(id);
69                             if (isPessoaValida(pessoaFisica)) {
70                                 pessoaFisica.exibir();
71
72                                 pessoaFisica = alterarDadosPessoaFisica(scanner, pessoaFisica);
73                                 repoPessoaFisica.alterar(pessoaFisica);
74                             } else alertaPessoaInvalida();
75                             break;
76
77                         case "J":
78                             PessoaJuridica pessoaJuridica = repoPessoaJuridica.obter(id);
79                             if (isPessoaValida(pessoaJuridica)) {
80                                 pessoaJuridica.exibir();
81
82                                 pessoaJuridica = alterarDadosPessoaJuridica(scanner, pessoaJuridica);
83                                 repoPessoaJuridica.alterar(pessoaJuridica);
84                             } else alertaPessoaInvalida();
85                             break;
86
87                         default:
88                             alertaOpcaoInvalida();
89                             break;
90                     }
91                     break;
92                 case 3:
93                     tipoPessoa = fisicaOUjuridica(scanner);
94                     id = getId(scanner);
95                     if (id == null) {
96                         alertaOpcaoInvalida();
97                         break;
98                     }
99
100                     switch (tipoPessoa) {
101                         case "F":
102                             if (repoPessoaFisica.excluir(id))
103                                 System.out.println("Pessoa excluida com sucesso.");
104                             else
105                                 System.out.println("Falha ao excluir.");
106                             break;

```

```

107
108         case "J":
109             if (repoPessoaJuridica.excluir(id))
110                 System.out.println("Empresa excluida com sucesso.");
111             else
112                 System.out.println("Falha ao excluir.");
113             break;
114
115         default:
116             alertaOpcaoInvalida();
117             break;
118     }
119     break;
120
121     case 4:
122         tipoPessoa = fisicaOUJuridica(scanner);
123         id = getId(scanner);
124         if (id == null){
125             alertaOpcaoInvalida();
126             break;
127         }
128
129         switch (tipoPessoa) {
130             case "F":
131                 PessoaFisica pessoaFisica = repoPessoaFisica.obter(id);
132                 if (isPessoaValida(pessoaFisica))
133                     pessoaFisica.exibir();
134                 else alertaPessoaInvalida();
135                 break;
136
137             case "J":
138                 PessoaJuridica pessoaJuridica = repoPessoaJuridica.obter(id);
139                 if (isPessoaValida(pessoaJuridica))
140                     pessoaJuridica.exibir();
141                 else alertaPessoaInvalida();
142                 break;
143
144             default:
145                 alertaOpcaoInvalida();
146                 break;
147         }
148         break;
149
150     case 5:
151         tipoPessoa = fisicaOUJuridica(scanner);
152         switch (tipoPessoa) {
153             case "F":
154                 for (PessoaFisica pessoa : repoPessoaFisica.obterTodos()) {
155                     pessoa.exibir();
156                 }
157                 break;
158
159             case "J":
160                 for (PessoaJuridica empresa : repoPessoaJuridica.obterTodos()) {
161                     empresa.exibir();
162                 }
163                 break;
164
165             default:
166                 alertaOpcaoInvalida();
167                 break;
168         }
169         break;
170
171     case 6:
172         prefixo = obterPrefixo(scanner);
173
174         try {
175             repoPessoaFisica.persistir(prefixo + ".fisica.bin");
176             System.out.println("Dados de Pessoas Fisicas Armazenados");
177
178             repoPessoaJuridica.persistir(prefixo + ".juridica.bin");
179             System.out.println("Dados de Pessoas Juridica Armazenados");
180         } catch (IOException ex) {
181             Logger.getLogger(CadastroPOODois.class.getName()).log(Level.SEVERE, null, ex);
182         }
183         break;
184
185     case 7:
186         prefixo = obterPrefixo(scanner);
187
188         try {
189             repoPessoaFisica.recuperar(prefixo + ".fisica.bin");
190             System.out.println("Dados de Pessoas Fisicas Recuperados");
191
192             repoPessoaJuridica.recuperar(prefixo + ".juridica.bin");
193             System.out.println("Dados de Pessoas Juridica Armazenados");
194         } catch (IOException | ClassNotFoundException ex) {
195             Logger.getLogger(CadastroPOODois.class.getName()).log(Level.SEVERE, null, ex);
196         }
197         break;
198
199     case 0:
200         System.out.println("Finalizando...");
201         break;
202     default:
203         alertaOpcaoInvalida();
204         break;
205     }
206 } while (opcao != 0);
207 }

```

```

208
209     private static void alertaOpcaoInvalida(){
210         System.out.println("Opção inválida. Tente novamente.");
211     }
212
213     private static void alertaPessoaInvalida(){
214         System.out.println("Pessoa/Empresa não encontrada.");
215     }
216
217     private static boolean isPessoaValida(Object obj){
218         return obj != null;
219     }
220
221     private static String fisicaOUjuridica(Scanner scanner){
222         System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
223         return scanner.next();
224     }
225
226     private static String obterPrefixo(Scanner scanner){
227         System.out.println("Informe o prefixo do arquivo a ser salvo");
228         return scanner.next();
229     }
230
231     private static Integer getId(Scanner scanner){
232         System.out.println("Digite o id da pessoa:");
233         try{
234             return scanner.nextInt();
235         } catch (Exception e) {
236             return null;
237         }
238     }
239
240     private static PessoaFisica lerDadosFisica(Scanner scanner) throws Exception{
241         try{
242             System.out.println("Insira os dados...");
243             System.out.println("Digite o id da pessoa");
244             int id = scanner.nextInt();
245
246             System.out.println("Digite o nome da pessoa");
247             String nome = scanner.next();
248
249             System.out.println("Digite o cpf da pessoa");
250             String cpf = scanner.next();
251
252             System.out.println("Digite a idade da pessoa");
253             int idade = scanner.nextInt();
254
255             return new PessoaFisica(id, nome, cpf, idade);
256         } catch (Exception e){
257             throw new Exception("Dado digitado está incorreto. Tente novamente.");
258         }
259     }
260
261     private static PessoaJuridica lerDadosJuridica(Scanner scanner) throws Exception{
262         try{
263             System.out.println("Digite o id da empresa");
264             int id = scanner.nextInt();
265
266             System.out.println("Digite o nome da empresa");
267             String nome = scanner.next();
268
269             System.out.println("Digite o cnpj da empresa");
270             String cnpj = scanner.next();
271
272             return new PessoaJuridica(id, nome, cnpj);
273         } catch (Exception e){
274             throw new Exception("Dado incorreto. Tente novamente.");
275         }
276     }
277
278     private static PessoaFisica alterarDadosPessoaFisica(Scanner scanner, PessoaFisica pessoa){
279         boolean continuar = true;
280
281         while(continuar){
282             System.out.println("Selecione uma opção:");
283             System.out.println("N - Nome");
284             System.out.println("C - CPF");
285             System.out.println("I - Idade");
286             System.out.println("F - Finalizar");
287             String opcao = scanner.next();
288

```

```

289         try{
290             switch (opcao) {
291                 case "N":
292                     System.out.println("Digite o novo nome:");
293                     String nome = scanner.next();
294                     pessoa.setNome(nome);
295                     break;
296                 case "C":
297                     System.out.println("Digite o novo CPF:");
298                     String cpf = scanner.next();
299                     pessoa.setCpf(cpf);
300                     break;
301                 case "I":
302                     System.out.println("Digite a nova idade:");
303                     int idade = scanner.nextInt();
304                     pessoa.setIdade(idade);
305                     break;
306                 case "F":
307                     continuar = false;
308                     break;
309                 default:
310                     alertaOpcaoInvalida();
311             }
312         } catch(Exception e){
313             alertaOpcaoInvalida();
314         }
315     }
316     return pessoa;
317 }
318
319 private static PessoaJuridica alterarDadosPessoaJuridica(Scanner scanner, PessoaJuridica empresa){
320     boolean continuar = true;
321
322     while(continuar){
323         System.out.println("Selecione uma opção:");
324         System.out.println("N - Nome");
325         System.out.println("C - CNPJ");
326         System.out.println("F - Finalizar");
327         String opcao = scanner.next();
328
329         try{
330             switch (opcao) {
331                 case "N":
332                     System.out.println("Digite o novo nome:");
333                     String nome = scanner.next();
334                     empresa.setNome(nome);
335                     break;
336                 case "C":
337                     System.out.println("Digite o novo CNPJ:");
338                     String cnpj = scanner.next();
339                     empresa.setCnpj(cnpj);
340                     break;
341                 case "F":
342                     continuar = false;
343                     break;
344                 default:
345                     alertaOpcaoInvalida();
346             }
347         } catch(Exception e){
348             alertaOpcaoInvalida();
349         }
350     }
351     return empresa;
352 }
353 }

```

```
PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE  PORTS  SEARCH  GITLENS

=====
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
=====
1
F - Pessoa Física | J - Pessoa Jurídica
F
Insira os dados...
Digite o id da pessoa
01
Digite o nome da pessoa
Teste
Digite o cpf da pessoa
111.111.111-11
Digite a idade da pessoa
27
=====
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
=====
5
F - Pessoa Física | J - Pessoa Jurídica
F
ID: 1
Nome: Teste
CPF: 111.111.111-11
=====
```

## Análise e Conclusão:

### 1.1 O que são elementos estáticos e qual o motivo para o método Main adotar esse modificador?

R: Os elementos estáticos são membros de classe que não pertencem a nenhuma instancia específica de classe, mas sim à própria classe em si. Isso significa que eles existem e podem ser acessados sem a necessidade de criar um objeto.

### 1.2 Para que serve a classe Scanner?

R: A classe Scanner fornece métodos para ler entradas de dados do usuário através do console ou arquivo.

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

R: O uso de classes de repositório tem impacto positivo na organização de diversas maneiras. As classes de repositório encapsulam lógica de acesso a dados, o que torna o código mais modular e fácil de entender. As classes de repositório podem ser reutilizadas em diferentes partes da aplicação, o que evita duplicação. O uso de classes de repositório torna o código mais fácil de manter, já que as alterações as alterações de acesso a dados

podem ser feita em um único local. E por fim, as classes de repositório são facilmente substituídas por outras implementações, o que torna o código flexível e adaptável.