



Estácio

Nome do campus: Polo Pirituba

Nome do curso: Desenvolvimento Full Stack

Nome da disciplina: Nível 1: Iniciando o caminho pelo java

Número da turma: EAD - 9001

Semestre letivo: 2025.1 3 semestre

Nome do integrante da prática: Guilherme Wissenbach Ferreira

Repositório Github:

Título da prática: Missão Prática | Nível 1 | Mundo 3

Objetivo da prática:

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

- 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.

No pacote model criar as entidades, com as seguintes características:

Classe Pessoa

```
Pessoa.java X
Source History
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5
6   package model;
7
8   /**
9    *
10   * @author Guilherme
11   */
12
13   import java.io.Serializable;
14
15   public class Pessoa implements Serializable{
16       private int id;
17       private String nome;
18
19       public Pessoa(int id, String nome) {
20           this.id = id;
21           this.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
40       public void exibir(){
41           System.out.print("Id: "+this.id + "\n" + "Nome: " + this.nome + "\n");
42       }
43   }
44
```



Estácio

Classe Pessoa Fisica

```
PessoaFisica.java [-A] x
Source History
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5
6   package model;
7
8   /**
9    *
10   * @author Guilherme
11   */
12
13   import java.io.Serializable;
14
15   public class PessoaFisica extends Pessoa implements Serializable {
16
17       private String cpf;
18       private int idade;
19
20       public PessoaFisica(int id, String nome, String cpf, int idade){
21           super(id, nome);
22           this.cpf = cpf;
23           this.idade = idade;
24       }
25
26       public String getCpf() {
27           return cpf;
28       }
29
30       public void setCpf(String cpf) {
31           this.cpf = cpf;
32       }
33
34       public int getIdade(){
35           return idade;
36       }
37
38       public void setIdade(int idade){
39           this.idade = idade;
40       }
41
42       public void exibir(){
43           System.out.print("\n" + "id: " + getId() + "\nNome: " + getNome() + "\nCPF: " + this.cpf + "\n" + "Idade: " + this.idade + "\n");
44       }
45   }
46
```

Classe Pessoa Juridica

```
PessoaJuridica.java [-A] x
Source History
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5
6   package model;
7
8   /**
9    *
10   * @author Guilherme
11   */
12
13   import java.io.Serializable;
14
15   public class PessoaJuridica extends Pessoa implements Serializable{
16
17       private String cnpj;
18
19       public PessoaJuridica(int id, String nome, String cnpj){
20           super(id, nome);
21           this.cnpj = cnpj;
22       }
23
24       public String getCnpj() {
25           return cnpj;
26       }
27
28       public void setCnpj(String cnpj) {
29           this.cnpj = cnpj;
30       }
31
32       public void exibir(){
33           System.out.print("\n" + "id: " + getId() + "\nNome: " + getNome() + "\nCNPJ: " + this.cnpj + "\n");
34       }
35   }
36
```



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

Classe PessoaFisicaRepo

```
PessoaFisicaRepo.java [-/A] x
Source History
1  /**
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5
6   package model;
7   import java.util.ArrayList;
8   import java.util.NoSuchElementException;
9   import java.util.Optional;
10  import java.io.*;
11
12  /**
13   *
14   * @author Guilherme
15   */
16  public class PessoaFisicaRepo {
17
18      private ArrayList<PessoaFisica> listaPessoasFisicas = new ArrayList<>();
19
20      public void inserir(PessoaFisica pessoaFisica){
21          listaPessoasFisicas.add(pessoaFisica);
22      }
23
24      public void alterar(PessoaFisica pessoaFisica, String novoNome, String novoCpf, int novaIdade) {
25          pessoaFisica.setNome(novoNome);
26          pessoaFisica.setCpf(novoCpf);
27          pessoaFisica.setIdade(novaIdade);
28      }
29
30      public void excluir(int id) {
31          try{
32              listaPessoasFisicas.remove(obter(id));
33          }catch(NoSuchElementException e){
34              System.out.println("erro");
35          }
36      }
37
38      public PessoaFisica obter(int id) {
39          Optional<PessoaFisica> pessoaFisicaLocalizada = listaPessoasFisicas.stream().
40              filter(pessoaFisica -> pessoaFisica.getId() == id).findFirst();
41          if (pessoaFisicaLocalizada.isPresent()) {
42              return pessoaFisicaLocalizada.get();
43          } else {
44              return null;
45          }
46      }
47
48      public ArrayList<PessoaFisica> obterTodos(){
49          return listaPessoasFisicas;
50      }
51
52      public void persistir(String arquivo) throws IOException {
53          ObjectOutputStream arquivoSaida = new ObjectOutputStream(new FileOutputStream(arquivo));
54          arquivoSaida.writeObject(listaPessoasFisicas);
55          arquivoSaida.close();
56          System.out.println("Dados de pessoas fisicas armazenados.");
57      }
58
59      public void recuperar(String arquivo) throws IOException, ClassNotFoundException {
60          ObjectInputStream arquivoEntrada = new ObjectInputStream(new FileInputStream(arquivo));
61          listaPessoasFisicas = (ArrayList<PessoaFisica>) arquivoEntrada.readObject();
62          arquivoEntrada.close();
63          System.out.println("Dados de pessoas fisicas recuperados.");
64      }
65  }
66
```



Classe PessoaJuridicaRepo

```
PessoaJuridicaRepo.java [-A] x
Source History
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5
6   package model;
7   import java.util.ArrayList;
8   import java.util.Optional;
9   import java.io.*;
10
11  /**
12   *
13   * @author Guilherme
14   */
15  public class PessoaJuridicaRepo {
16
17      private ArrayList<PessoaJuridica> listaPessoasJuridicas = new ArrayList<>();
18
19      public void inserir(PessoaJuridica pessoaJuridica) {
20          listaPessoasJuridicas.add(pessoaJuridica);
21      }
22
23      public void alterar(PessoaJuridica pessoaJuridica, String novoNome, String novoCnpj) {
24          pessoaJuridica.setNome(novoNome);
25          pessoaJuridica.setCnpj(novoCnpj);
26      }
27
28      public void excluir(int id) {
29          listaPessoasJuridicas.remove(obter(id));
30      }
31
32      public PessoaJuridica obter(int id) {
33          Optional<PessoaJuridica> pessoaJuridicaLocalizada = listaPessoasJuridicas.stream().
34              filter(pessoaJuridica -> pessoaJuridica.getId() == id).findFirst();
35          if (pessoaJuridicaLocalizada.isPresent()) {
36              return pessoaJuridicaLocalizada.get();
37          } else {
38              return null;
39          }
40      }
41
42      public ArrayList<PessoaJuridica> obterTodos() {
43          return listaPessoasJuridicas;
44      }
45
46      public void persistir(String arquivo) throws IOException {
47          ObjectOutputStream arquivoSaida = new ObjectOutputStream(new FileOutputStream(arquivo));
48          arquivoSaida.writeObject(listaPessoasJuridicas);
49          arquivoSaida.close();
50          System.out.println("\nDados das pessoas juridicas armazenados.");
51      }
52
53      public void recuperar(String arquivo) throws IOException, ClassNotFoundException {
54          ObjectInputStream arquivoEntrada = new ObjectInputStream(new FileInputStream(arquivo));
55          listaPessoasJuridicas = (ArrayList<PessoaJuridica>) arquivoEntrada.readObject();
56          arquivoEntrada.close();
57          System.out.println("Dados de pessoas juridicas recuperados.");
58      }
59  }
60
```



main class CadastroPOO

```
CadastroPOO.java [-A] x
Source History
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
4  */
5
6  package model;
7  import java.io.IOException;
8
9  /**
10   *
11   * @author Guilherme
12   */
13  public class CadastroPOO {
14      public static void main(String[] args) {
15
16          PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
17          PessoaFisica pessoaFisica1 = new PessoaFisica(1, "Ana", "1111111111", 25);
18          PessoaFisica pessoaFisica2 = new PessoaFisica(2, "Carlos Jose", "2222222222", 52);
19          repo1.inserir(pessoaFisica1);
20          repo1.inserir(pessoaFisica2);
21          try {
22              repo1.persistir("listaPessoasFisicas.bin");
23          } catch (IOException erro) {
24              System.out.println("Erro ao persistir os dados: " + erro.getMessage());
25          }
26
27          PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
28
29          try {
30              repo2.recuperar("listaPessoasFisicas.bin");
31              repo2.obterTodos()
32                  .forEach(pessoaFisica -> {
33                      pessoaFisica.exibir();
34                  });
35          } catch (IOException | ClassNotFoundException erro) {
36              System.out.println("Erro ao recuperar os dados: " + erro.getMessage());
37          }
38
39
40
41          PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
42          PessoaJuridica pessoaJuridica1 = new PessoaJuridica(3, "XPTO Sales", "33333333333333");
43          PessoaJuridica pessoaJuridica2 = new PessoaJuridica(4, "XPTO Solutions", "44444444444444");
44          repo3.inserir(pessoaJuridica1);
45          repo3.inserir(pessoaJuridica2);
46          try {
47              repo3.persistir("listaPessoasJuridicas.bin");
48          } catch (IOException erro) {
49              System.out.println("Erro ao persistir os dados: " + erro.getMessage());
50          }
51
52          PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
53          try {
54              repo4.recuperar("listaPessoasJuridicas.bin");
55              repo4.obterTodos()
56                  .forEach(pessoaJuridica -> {
57                      pessoaJuridica.exibir();
58                  });
59          } catch (IOException | ClassNotFoundException erro) {
60              System.out.println("Erro ao recuperar os dados: " + erro.getMessage());
61          }
62      }
63  }
64  }
```



Resultado

```
Output - CadastroPOO (run) x
run:
Dados de pessoas fisicas armazenados.
Dados de pessoas fisicas recuperados.

id: 1
Nome: Ana
CPF: 11111111111
Idade: 25

id: 2
Nome: Carlos Jose
CPF: 22222222222
Idade: 52

Dados das pessoas juridicas armazenados.
Dados de pessoas juridicas recuperados.

id: 3
Nome: XPT0 Sales
CNPJ: 333333333333333

id: 4
Nome: XPT0 Solutions
CNPJ: 444444444444444
BUILD SUCCESSFUL (total time: 0 seconds)
```

Análise e Conclusão:

Quais as vantagens e desvantagens do uso de herança?

Vantagens

Reutilização de Código, pois evita duplicação de código;

Facilidade de Manutenção, é fácil as alterações na superclasse que são refletidas automaticamente nas subclasses;

Organização Estrutural que proporciona uma estrutura hierárquica lógica, tornando o código mais compreensível e modular.

Polimorfismo que permite que objetos da subclasse sejam tratados como objetos da superclasse, facilitando o uso de conceitos como sobreposição de métodos e métodos abstratos;

Classes podem herdar características(métodos e atributos) de outras classes situadas acima ou transmitir suas características às classes abaixo. Caso uma alteração seja necessária, ela só precisará ser feita na classe pai, e será automaticamente propagada para as subclasses.

Desvantagens

É fraco o encapsulamento entre classes e subclasses, onde ao mudar uma superclasse pode afetar todas as subclasses;

Quando um objeto precisa ser de uma classe diferente em momentos diferentes e não é possível com a herança.



Estácio

Análise e Conclusão:

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

Essa interface permite que os objetos sejam serializados(convertidos em uma sequência de bytes) e desserializados com a conversão de volta à um objeto. Esse processo é essencial para armazenar objetos em arquivos, transmiti-los via rede ou gravá-los em banco de dados de forma eficiente.

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

O paradigma funcional é amplamente utilizado na API Stream do Java que é usada para manipular coleções (Collections) de uma maneira mais eficiente, utilizando funções. Ela possibilita uma interação sobre essas coleções de objetos e, a cada elemento, realizar alguma ação, seja ela de filtragem, mapeamento, transformação, etc.

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

Na persistência de dados em Java, o padrão de desenvolvimento mais adotado é o Data Access Object (DAO) que é o padrão que separa a lógica de acesso a dados da lógica de negócios, tornando o código mais organizado e fácil de mante-lo. Nesse nosso projeto foram utilizadas a classe ObjectOutputStream para escrever objetos em um arquivo "[prefixo].fisica.bin e [prefixo].juridica.bin" e a classe ObjectInputStream para ler os objetos dos mesmos arquivos.



Estácio

main class CadastroPOO2

```
CadastroPOO2.java [-/A] x
Source History
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
4   */
5
6
7  package model;
8  import java.io.IOException;
9  import java.util.Scanner;
10
11
12  /**
13   *
14   * @author Guilherme
15   */
16  public class CadastroPOO2 {
17
18      public static void main(String[] args) {
19
20          PessoaFisicaRepo pfRepo = new PessoaFisicaRepo();
21          PessoaJuridicaRepo pjRepo = new PessoaJuridicaRepo();
22
23          Scanner scan = new Scanner(System.in);
24          String escolha;
25
26          do {
27              System.out.println("=====");
28              System.out.println("1 - Incluir Pessoa");
29              System.out.println("2 - Alterar Pessoa");
30              System.out.println("3 - Excluir Pessoa");
31              System.out.println("4 - Buscar pelo Id");
32              System.out.println("5 - Exibir Todos");
33              System.out.println("6 - Persistir/Salvar Dados");
34              System.out.println("7 - Recuperar/Carregar Dados");
35              System.out.println("0 - Finalizar Programa");
36              System.out.println("=====");
37
38              escolha = scan.next();
39
40              switch (escolha) {
41                  // Incluir
42                  case "1":
43                      do {
44                          System.out.println("=====");
45                          System.out.println("F - Pessoa Fisica | J - Pessoa Juridica | M - Menu");
46
47                          escolha = scan.next();
48                          scan.nextLine();
49
50                          switch (escolha.toUpperCase()) {
51                              case "F":
52                                  System.out.print("Digite o id da pessoa: ");
53                                  int idInformado = scan.nextInt();
54                                  System.out.println("Insira os dados... ");
55                                  scan.nextLine();
56                                  System.out.print("Nome: ");
57                                  String nome = scan.nextLine();
58                                  System.out.print("CPF: ");
59                                  String cpf = scan.nextLine();
60                                  System.out.print("Idade: ");
61                                  int idade = scan.nextInt();
62
63                                  int pfRepoSize = pfRepo.obterTodos().size();
64
65                                  PessoaFisica pessoaFisica = new PessoaFisica(idInformado, nome, cpf, idade);
66                                  pfRepo.inserir(pessoaFisica);
67
68                                  System.out.println("Inclusao realizada com sucesso!");
69                                  pessoaFisica.exibir();
70                                  break;
71
72                              case "J":
73                                  System.out.print("Digite o id da pessoa: ");
74                                  int idInformado2 = scan.nextInt();
75                                  scan.nextLine();
76                                  System.out.print("Nome: ");
77                                  nome = scan.nextLine();
78                                  System.out.print("CNPJ: ");
79                                  String cnpj = scan.nextLine();
80
81                                  // ... (rest of the code for case J)
82
83                              // ... (rest of the code for other cases)
84
85                          }
86                      } while (true);
87
88                  // ... (rest of the code for other cases)
89
90              }
91
92              System.out.println("=====");
93          } while (true);
94      }
95  }
```




main class CadastroPOO2

```
82      int pjRepoSize = pjRepo.obterTodos().size();
83
84      PessoaJuridica pessoaJuridica = new PessoaJuridica(idInformado2, nome, cnpj);
85      pjRepo.inserir(pessoaJuridica);
86
87      System.out.println("Inclusao realizada com sucesso!");
88      pessoaJuridica.exibir();
89      break;
90
91
92      case "M":
93          break;
94
95      default:
96          System.out.println("Opcao invalida.");
97          break;
98
99      } while (!escolha.equalsIgnoreCase("M"));
100      break;
101
102      // Alterar
103      case "2":
104          do {
105              System.out.println("=====");
106              System.out.println("F - Pessoa Fisica | J - Pessoa Juridica | M - Menu");
107
108              escolha = scan.next();
109              scan.nextLine();
110
111              switch (escolha.toUpperCase()) {
112
113                  case "F":
114                      System.out.println("Digite o ID da pessoa: ");
115                      int idPessoaFisica = scan.nextInt();
116                      scan.nextLine();
117
118                      PessoaFisica pessoaFisicaLocalizada = pfRepo.obter(idPessoaFisica);
119
120                      if (pessoaFisicaLocalizada != null) {
121                          pessoaFisicaLocalizada.exibir();
122
123                          System.out.println("Nome atual: " + pessoaFisicaLocalizada.getNome());
124                          System.out.print("Novo nome: ");
125                          String novoNome = scan.nextLine();
126
127                          System.out.println("CPF atual: " + pessoaFisicaLocalizada.getCpf());
128                          System.out.print("Novo CPF: ");
129                          String novoCPF = scan.nextLine();
130
131                          System.out.println("Idade atual: " + pessoaFisicaLocalizada.getIdade());
132                          System.out.print("Nova Idade: ");
133                          int novaIdade = scan.nextInt();
134
135                          pfRepo.alterar(pessoaFisicaLocalizada, novoNome, novoCPF, novaIdade);
136
137                          System.out.println("Pessoa alterada com sucesso!");
138                      } else
139                          System.out.println("Pessoa nao localizada!");
140                      break;
141
142                  case "J":
143                      System.out.println("Digite o ID da pessoa: ");
144                      int idPessoaJuridica = scan.nextInt();
145                      scan.nextLine();
146
147                      PessoaJuridica pessoaJuridicaLocalizada = pjRepo.obter(idPessoaJuridica);
148
149                      if (pessoaJuridicaLocalizada != null) {
150                          pessoaJuridicaLocalizada.exibir();
151
152                          System.out.println("Nome atual: " + pessoaJuridicaLocalizada.getNome());
153                          System.out.print("Novo nome: ");
154                          String novoNome = scan.nextLine();
155
156                          System.out.println("CNPJ atual: " + pessoaJuridicaLocalizada.getCnpj());
157                          System.out.print("Novo CNPJ: ");
158                          String novoCNPJ = scan.nextLine();
159
160                          pjRepo.alterar(pessoaJuridicaLocalizada, novoNome, novoCNPJ);
```



Estácio

main class CadastroPOO2

```
161
162         System.out.println("Pessoa alterada com sucesso!");
163     } else
164         System.out.println("Pessoa nao localizada!");
165     break;
166
167     case "M":
168         break;
169
170     default:
171         System.out.println("Opcao invalida.");
172         break;
173 }
174 } while (!escolha.equalsIgnoreCase("M"));
175 break;
176
177 // EXCLUIR
178 case "3":
179     do {
180         System.out.println("=====");
181         System.out.println("F - Pessoa Fisica | J - Pessoa Juridica | M - Menu");
182
183         escolha = scan.next();
184         scan.nextLine();
185
186         switch (escolha.toUpperCase()) {
187
188             case "F":
189                 System.out.println("Digite o ID da pessoa: ");
190                 int idPessoaFisica = scan.nextInt();
191
192                 PessoaFisica pessoaFisicaLocalizada = pfRepo.obter(idPessoaFisica);
193
194                 if (pessoaFisicaLocalizada != null) {
195                     pessoaFisicaLocalizada.exibir();
196                     pfRepo.excluir(idPessoaFisica);
197
198                     System.out.println("Pessoa excluida com sucesso!");
199                 } else
200                     System.out.println("Pessoa nao localizada!");
201                 break;
202
203             case "J":
204                 System.out.println("Digite o ID da pessoa: ");
205                 int idPessoaJuridica = scan.nextInt();
206
207                 PessoaJuridica pessoaJuridicaLocalizada = pjRepo.obter(idPessoaJuridica);
208
209                 if (pessoaJuridicaLocalizada != null) {
210                     pessoaJuridicaLocalizada.exibir();
211                     pjRepo.excluir(idPessoaJuridica);
212
213                     System.out.println("Pessoa excluida com sucesso!");
214                 } else
215                     System.out.println("Pessoa nao localizada!");
216                 break;
217
218             case "M":
219                 break;
220
221             default:
222                 System.out.println("Opcao invalida.");
223                 break;
224         }
225     } while (!escolha.equalsIgnoreCase("M"));
226     break;
227
228 // obterId
229 case "4":
230     do {
231         System.out.println("=====");
232         System.out.println("F - Pessoa Fisica | J - Pessoa Juridica | M - Menu");
233
234         escolha = scan.next();
235         scan.nextLine();
236
237         switch (escolha.toUpperCase()) {
```



Estácio

main class CadastroPOO2

```
242         case "F":
243             System.out.println("Digite o ID da pessoa: ");
244             int idPessoaFisica = scan.nextInt();
245
246             PessoaFisica pessoaFisicaLocalizada = pfRepo.obter(idPessoaFisica);
247
248             if (pessoaFisicaLocalizada != null) {
249                 System.out.println("Pessoa localizada!");
250                 pessoaFisicaLocalizada.exibir();
251             } else
252                 System.out.println("Pessoa nao localizada!");
253             break;
254
255         case "J":
256             System.out.println("Digite o ID da pessoa: ");
257             int idPessoaJuridica = scan.nextInt();
258
259             PessoaJuridica pessoaJuridicaLocalizada = pjRepo.obter(idPessoaJuridica);
260
261             if (pessoaJuridicaLocalizada != null) {
262                 System.out.println("Pessoa localizada!");
263
264                 pessoaJuridicaLocalizada.exibir();
265             } else
266                 System.out.println("Pessoa nao localizada!");
267             break;
268
269         case "M":
270             break;
271
272         default:
273             System.out.println("Opcao invalida.");
274             break;
275     }
276
277     } while (!escolha.equalsIgnoreCase("M"));
278     break;
279
280     //obterTodos
281     case "5":
282         do {
283             System.out.println("=====");
284             System.out.println("F - Pessoa Fisica | J - Pessoa Juridica | M - Menu");
285
286             escolha = scan.next();
287             scan.nextLine();
288
289             switch (escolha.toUpperCase()) {
290
291                 case "F":
292                     System.out.println("Lista de pessoas Fisicas: ");
293                     pfRepo.obterTodos()
294                         .forEach(pessoaFisica -> {
295                             pessoaFisica.exibir();
296                             System.out.println();
297                         });
298                     break;
299
300                 case "J":
301                     System.out.println("Lista de pessoas juridicas: ");
302                     pjRepo.obterTodos()
303                         .forEach(pessoaJuridica -> {
304                             pessoaJuridica.exibir();
305                             System.out.println();
306                         });
307                     break;
308
309                 case "M":
310                     break;
311
312                 default:
313                     System.out.println("Opcao invalida");
314                     break;
315             }
316
317         } while (!escolha.equalsIgnoreCase("M"));
318     break;
319
```



main class CadastroPOO2

```
320 // Persistir/Salvar
321 case "6":
322     System.out.println("Escolha o nome do arquivo");
323     escolha = scan.next();
324     scan.nextLine();
325     try {
326         pfRepo.persistir(escolha+".fisica.bin");
327         pjRepo.persistir(escolha+".juridica.bin");
328     } catch (IOException erro) {
329         System.out.println("Erro ao persistir/salvar os dados: " + erro.getMessage());
330     }
331     break;
332
333 //Recuperar/Carregar
334 case "7":
335     System.out.println("Informe o nome do arquivo salvo");
336     escolha = scan.next();
337     scan.nextLine();
338     try {
339         pfRepo.recuperar(escolha+".fisica.bin");
340         pjRepo.recuperar(escolha+".juridica.bin");
341     } catch (ClassNotFoundException | IOException erro) {
342         System.out.println("Erro ao recuperar os dados: " + erro.getMessage());
343     }
344     break;
345
346 case "0":
347     System.out.println("Sistema Finalizado com sucesso.");
348     break;
349
350 default:
351     System.out.println("Opcao invalida");
352     break;
353 }
354 } while (!escolha.equals("0"));
355 scan.close();
356 }
357 }
358 }
```



Estácio

Resultado

```
Output - CadastroPOO2 (run) X
run:
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir/Salvar Dados
7 - Recuperar/Carregar Dados
0 - Finalizar Programa
=====
1
=====
F - Pessoa Fisica | J - Pessoa Juridica | M - Menu
f
Digite o id da pessoa: 1
Insira os dados...
Nome: jorge
CPF: 424
Idade: 24
Inclusao realizada com sucesso!

id: 1
Nome: jorge
CPF: 424
Idade: 24
=====
F - Pessoa Fisica | J - Pessoa Juridica | M - Menu
f
Digite o id da pessoa: 2
Insira os dados...
Nome: joão
CPF: 242
Idade: 42
Inclusao realizada com sucesso!

id: 2
Nome: joão
CPF: 242
Idade: 42
=====
```

```
Output - CadastroPOO2 (run) X
Idade: 42
=====
F - Pessoa Fisica | J - Pessoa Juridica | M - Menu
j
Digite o id da pessoa: 3
Nome: jose
CNPJ: 444
Inclusao realizada com sucesso!

id: 3
Nome: jose
CNPJ: 444
=====
F - Pessoa Fisica | J - Pessoa Juridica | M - Menu
m
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir/Salvar Dados
7 - Recuperar/Carregar Dados
0 - Finalizar Programa
=====
2
=====
F - Pessoa Fisica | J - Pessoa Juridica | M - Menu
f
Digite o ID da pessoa:
2

id: 2
Nome: joão
CPF: 242
Idade: 42
Nome atual: joão
Novo nome: reinaldo
CPF atual: 242
Novo CPF: 222
Idade atual: 242
Nova Idade: 22
Pessoa alterada com sucesso!
=====
F - Pessoa Fisica | J - Pessoa Juridica | M - Menu
m
=====
```



Estácio

Resultado

```
Output - CadastroPOO2 (run) x
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir/Salvar Dados
7 - Recuperar/Carregar Dados
0 - Finalizar Programa
=====
3
=====
F - Pessoa Fisica | J - Pessoa Juridica | M - Menu
f
Digite o ID da pessoa:
2

id: 2
Nome: reinaldo
CPF: 222
Idade: 22
Pessoa excluida com sucesso!
=====
F - Pessoa Fisica | J - Pessoa Juridica | M - Menu
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir/Salvar Dados
7 - Recuperar/Carregar Dados
0 - Finalizar Programa
=====
4
=====
F - Pessoa Fisica | J - Pessoa Juridica | M - Menu
j
Digite o ID da pessoa:
3
Pessoa localizada!

id: 3
Nome: jose
CPF: 444
=====
F - Pessoa Fisica | J - Pessoa Juridica | M - Menu
=====
```

```
Output - CadastroPOO2 (run) x
F - Pessoa Fisica | J - Pessoa Juridica | M - Menu
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir/Salvar Dados
7 - Recuperar/Carregar Dados
0 - Finalizar Programa
=====
5
=====
F - Pessoa Fisica | J - Pessoa Juridica | M - Menu
f
Lista de pessoas Fisicas:

id: 1
Nome: jorge
CPF: 424
Idade: 24

=====
F - Pessoa Fisica | J - Pessoa Juridica | M - Menu
j
Lista de pessoas juridicas:

id: 3
Nome: jose
CPF: 444

=====
F - Pessoa Fisica | J - Pessoa Juridica | M - Menu
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir/Salvar Dados
7 - Recuperar/Carregar Dados
0 - Finalizar Programa
=====
7
Informe o nome do arquivo salvo
jose
Erro ao recuperar os dados: jose.fisica.bin (O sistema n o pode encontrar o arquivo especificado)
=====
```



Estácio

Resultado

```
Output - CadastroPOO2 (run) X
0 - Finalizar Programa
=====
5
=====
F - Pessoa Fisica | J - Pessoa Juridica | M - Menu
f
Lista de pessoas Fisicas:

id: 1
Nome: jorge
CPF: 424
Idade: 24

=====
F - Pessoa Fisica | J - Pessoa Juridica | M - Menu
j
Lista de pessoas juridicas:

id: 3
Nome: jose
CPFJ: 444

=====
F - Pessoa Fisica | J - Pessoa Juridica | M - Menu
m
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir/Salvar Dados
7 - Recuperar/Carregar Dados
0 - Finalizar Programa
=====
7
Informe o nome do arquivo salvo
jose
Erro ao recuperar os dados: jose.fisica.bin (O sistema não pode encontrar o arquivo especificado)
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir/Salvar Dados
7 - Recuperar/Carregar Dados
0 - Finalizar Programa
=====
0
Sistema Finalizado com sucesso.
BUILD SUCCESSFUL (total time: 2 minutes 24 seconds)
```

Análise e Conclusão:

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

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

Para que serve a classe Scanner?

A classe Scanner em Java é usada para ler entradas do usuário ou de outras fontes de dados, como arquivos e fluxos de entrada inseridos pelo usuário através do teclado.

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

As classes de repositório serviram para organizar as atividades de excluir, inserir, alterar, localizar e recuperar e também serviram para salvar os dados de pessoas físicas e jurídicas.