



RELATÓRIO DISCENTE DE ACOMPANHAMENTO

Campus Polo Cavallhada – Porto Alegre

DESENVOLVIMENTO FULL STACK

Missão Prática | Nível 1 | Mundo 3

RPG0014 – Iniciando o caminho pelo Java

Turma: 9003

Semestre Letivo: 3º - 2024.1

GUILHERME BERNARDES BASTOS

Repositório:

CADASTROPOO

Objetivo da Prática

Criar um sistema cadastral em Java, utilizando a programação orientada a objetos, herança, polimorfismo, persistência de objetos em arquivos binários, utilizar o controle de exceções e implementar uma interface cadastral em modo texto.

1º Procedimento

CÓDIGOS

```
1 package casdastropoo;
2
3 import java.io.IOException;
4 import model.PessoaFisica;
5 import model.PessoaJuridica;
6 import model.gerenciadores.PessoaFisicaRepo;
7 import model.gerenciadores.PessoaJuridicaRepo;
8
9 /**
10  * @author guilhermebernardes
11  */
12
13 public class CasdastroP00 {
14     public static void main(String[] args) throws IOException, ClassNotFoundException {
15         PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
16
17         PessoaFisica pessoa1 = new PessoaFisica(1, "Ana", "1111111111", 25);
18         PessoaFisica pessoa2 = new PessoaFisica(2, "Carlos", "2222222222", 52);
19
20         repo1.inserir(pessoa1);
21         repo1.inserir(pessoa2);
22
23         try {
24             repo1.persistir("DocPessoaFisica.bin");
25             System.out.println("Dados de Pessoas Físicas armazenados.");
26         } catch (IOException e) {
27             System.out.println("ERRO");
28         }
29
30         PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
31     }
```

```
31
32
33     try {
34         System.out.println("Dados de Pessoa Física recuperados.");
35         repo2.recuperar("DocPessoaFisica.bin");
36         repo2.obterTodos()
37             .forEach(pessoaFisica -> {
38                 pessoaFisica.exibirInformacoes();
39                 System.out.println();
40             });
41     } catch (IOException | ClassNotFoundException e) {
42         System.out.println("ERRO");
43     }
44
45     PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
46
47     PessoaJuridica pessoaJuridica1 = new PessoaJuridica(3, "XPT0 Sales", "33333333333333");
48     PessoaJuridica pessoaJuridica2 = new PessoaJuridica(4, "XPT0 Solutions", "4444444444444444");
49
50     repo3.inserir(pessoaJuridica1);
51     repo3.inserir(pessoaJuridica2);
52
53     try {
54         repo3.persistir("DocPessoaJuridica.bin");
55         System.out.println("Dados de Pessoas Jurídicas armazenados.");
56     } catch (IOException e) {
57         System.out.println("ERRO");
58     }
59
60     PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
61
62     try {
63         System.out.println("Dados das Pessoas Jurídicas recuperados.");
64         repo4.recuperar("DocPessoaJuridica.bin");
65         repo4.obterTodos()
66             .forEach(pessoaJuridica -> {
67                 pessoaJuridica.exibirInformacoes();
68                 System.out.println();
69             });
70     } catch (IOException | ClassNotFoundException e) {
71         System.out.println("ERRO");
72     }
73 }
```

```
68         });
69     } catch (IOException | ClassNotFoundException e) {
70         System.out.println("ERRO");
71     }
72 }
73 }
```

```

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

```

```

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

```

```

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

```

```

1 package model.gerenciadores;
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 import java.util.List;
10 import model.PessoaFisica;
11 /**
12  * @author guilhermebernardes
13  */
14 public class PessoaFisicaRepo {
15     private List<PessoaFisica> pessoasFisicas;
16
17     public PessoaFisicaRepo() {
18         this.pessoasFisicas = new ArrayList<>();
19     }
20
21     public void inserir(PessoaFisica pessoaFisica) {
22         pessoasFisicas.add(pessoaFisica);
23     }
24
25     public void alterar(PessoaFisica pessoaFisica, String novoNome, String novoCpf, int novaIdade) {
26         pessoaFisica.setNome(novoNome);
27         pessoaFisica.setCpf(novoCpf);
28         pessoaFisica.setIdade(novaIdade);
29     }
30
31     public void excluir(int id) {
32         pessoasFisicas.removeIf(pessoaFisica -> pessoaFisica.getId() == id);
33     }
34
35     public PessoaFisica obter(int id) {
36         for (PessoaFisica pessoaFisica : pessoasFisicas) {
37             if (pessoaFisica.getId() == id) {
38                 return pessoaFisica;
39             }
40         }
41         return null;
42     }
43 }

```

```

43
44 public List<PessoaFisica> obterTodos() {
45     return new ArrayList<>(pessoasFisicas);
46 }
47
48 public void persistir(String DocPessoaFisica) throws IOException {
49     try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(DocPessoaFisica))) {
50         outputStream.writeObject(pessoasFisicas);
51     }
52 }
53
54 public void recuperar(String DocPessoaFisica) throws IOException, ClassNotFoundException {
55     try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(DocPessoaFisica))) {
56         pessoasFisicas = (List<PessoaFisica>) inputStream.readObject();
57     }
58 }
59
60 public Iterable<PessoaFisica> exibirInformacoes() {
61     throw new UnsupportedOperationException("Not supported yet.");
62 }
63 }

```

```

1 package model.gerenciadores;
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 import java.util.List;
10 import model.PessoaFisica;
11 import model.PessoaJuridica;
12
13 /**
14  * @author guilhermebernardes
15  */
16 public class PessoaJuridicaRepo {
17     private List<PessoaJuridica> pessoasJuridicas;
18
19     public PessoaJuridicaRepo() {
20         this.pessoasJuridicas = new ArrayList<>();
21     }
22
23     public void inserir(PessoaJuridica pessoaJuridica) {
24         pessoasJuridicas.add(pessoaJuridica);
25     }
26
27     public void alterar(PessoaJuridica pessoaJuridica, String novoNome, String novoCnpj) {
28         pessoaJuridica.setNome(novoNome);
29         pessoaJuridica.setCnpj(novoCnpj);
30     }
31
32     public void excluir(int id) {
33         pessoasJuridicas.removeIf(pessoaJuridica -> pessoaJuridica.getId() == id);
34     }
35
36     public PessoaJuridica obter(int id) {
37         for (PessoaJuridica pessoaJuridica : pessoasJuridicas) {
38             if (pessoaJuridica.getId() == id) {
39                 return pessoaJuridica;
40             }
41         }
42         return null;
43     }
44
45     public List<PessoaJuridica> obterTodos() {
46         return new ArrayList<>(pessoasJuridicas);
47     }
48
49     public void persistir(String DocPessoaJuridica) throws IOException {
50         try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(DocPessoaJuridica))) {
51             outputStream.writeObject(pessoasJuridicas);
52         }
53     }
54
55     public void recuperar(String DocPessoaJuridica) throws IOException, ClassNotFoundException {
56         try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(DocPessoaJuridica))) {
57             pessoasJuridicas = (List<PessoaJuridica>) inputStream.readObject();
58         }
59     }

```

RESULTADO

```

Output - CasdastroPOO (run)

run:
Dados de Pessoas Físicas armazenados.
Dados de Pessoa Física recuperados.
ID: 1
Nome: Ana
CPF: 11111111111
Idade: 25

ID: 2
Nome: Carlos
CPF: 22222222222
Idade: 52

Dados de Pessoas Jurídicas armazenados.
Dados das Pessoas Jurídicas recuperados.
ID: 3
Nome: XPT0 Sales
CNPJ: 33333333333333

ID: 4
Nome: XPT0 Solutions
CNPJ: 4444444444444444

BUILD SUCCESSFUL (total time: 5 seconds)

```

2º Procedimento

CÓDIGOS

```
1 package casdastropoo;
2
3 import java.io.IOException;
4 import java.util.List;
5 import java.util.Scanner;
6 import model.PessoaFisica;
7 import model.PessoaJuridica;
8 import model.gerenciadores.PessoaFisicaRepo;
9 import model.gerenciadores.PessoaJuridicaRepo;
10 /**
11  * @author guilhermebernardes
12  */
13 public class CasdastroP00 {
14     public static void main(String[] args) throws IOException, ClassNotFoundException {
15         Scanner scanner = new Scanner(System.in);
16         PessoaFisicaRepo pessoaFisicaRepo = new PessoaFisicaRepo();
17         PessoaJuridicaRepo pessoaJuridicaRepo = new PessoaJuridicaRepo();
18
19         try {
20             pessoaFisicaRepo.recuperar("pessoaFisica.bin");
21             pessoaJuridicaRepo.recuperar("pessoaJuridica.bin");
22         } catch (IOException | ClassNotFoundException e) {
23             System.out.println();
24         }
25
26         while (true) {
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 Dados");
34             System.out.println("7 - Recuperar Dados");
35             System.out.println("0 - Finalizar Programa");
36             System.out.println("=====");
37
38             int opcao = scanner.nextInt();
39             scanner.nextLine();
40
41             String tipo = "";
42
```

```
44 switch (opcao) {
45     case 1:
46         while (!tipo.equals("F") && !tipo.equals("J")) {
47             System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
48             tipo = scanner.nextLine().toUpperCase();
49
50             if (!tipo.equals("F") && !tipo.equals("J")) {
51                 System.out.println("Opção inválida. Por favor, escolha F ou J.");
52             }
53
54             System.out.println("Digite o id da pessoa: ");
55             int id = scanner.nextInt();
56             scanner.nextLine();
57
58             System.out.println("Insira os dados...");
59             System.out.println("Nome: ");
60             String nome = scanner.nextLine();
61
62             if (tipo.equals("F")) {
63                 System.out.println("CPF: ");
64                 String cpf = scanner.nextLine();
65
66                 System.out.println("Idade: ");
67                 int idade = scanner.nextInt();
68
69                 pessoaFisicaRepo.inserir(new PessoaFisica(id, nome, cpf, idade));
70
71                 try {
72                     pessoaFisicaRepo.persistir("pessoaFisica.bin");
73                 } catch (IOException e) {
74                     System.out.println("ERRO: " + e.getMessage());
75                 }
76
77             } else if (tipo.equals("J")) {
78                 System.out.println("CNPJ: ");
79                 String cnpj = scanner.nextLine();
80
81                 pessoaJuridicaRepo.inserir(new PessoaJuridica(id, nome, cnpj));
82
83                 try {
84                     pessoaJuridicaRepo.persistir("pessoaJuridica.bin");
85
```

```

85         } catch (IOException e) {
86             System.out.println("ERRO: " + e.getMessage());
87         }
88     }
89     } else {
90         System.out.println("DADO INVÁLIDO");
91     }
92     break;
93
94     case 2:
95         while (!tipo.equals("F") && !tipo.equals("J")) {
96             System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
97             tipo = scanner.nextLine().toUpperCase();
98
99             if (!tipo.equals("F") && !tipo.equals("J")) {
100                 System.out.println("Opção inválida. Por favor, escolha F ou J.");
101             }
102         }
103
104         System.out.println("ID que será alterado: ");
105         int alterarId = scanner.nextInt();
106         scanner.nextLine();
107
108         if (tipo.equals("F")) {
109             var pessoaFisica = pessoaFisicaRepo.obter(alterarId);
110             while (pessoaFisica == null) {
111                 System.out.println("Nenhum dado encontrado com esse ID. Digite novamente: ");
112                 alterarId = scanner.nextInt();
113                 scanner.nextLine();
114                 pessoaFisica = pessoaFisicaRepo.obter(alterarId);
115             }
116             if (pessoaFisica != null) {
117                 System.out.println("Dados atuais: ");
118                 System.out.println("Nome: " + pessoaFisica.getNome());
119                 System.out.println("CPF: " + pessoaFisica.getCpf());
120                 System.out.println("Idade: " + pessoaFisica.getIdade());
121
122                 System.out.println("Novos dados: ");
123                 System.out.println("Nome: ");
124                 String novoNome = scanner.nextLine();
125                 System.out.println("CPF: ");
126                 String novoCpf = scanner.nextLine();

```

```

127                 System.out.println("Idade: ");
128                 int novaIdade = scanner.nextInt();
129                 scanner.nextLine();
130
131                 pessoaFisica.setNome(novoNome);
132                 pessoaFisica.setCpf(novoCpf);
133                 pessoaFisica.setIdade(novaIdade);
134
135                 System.out.println("Dados alterados com sucesso.");
136             } else {
137                 System.out.println("Nenhum dado encontrado com esse ID.");
138             }
139
140             try {
141                 pessoaFisicaRepo.persistir("pessoaFisica.bin");
142             } catch (IOException e) {
143                 System.out.println("ERRO: " + e.getMessage());
144             }
145
146         } else if (tipo.equals("J")) {
147             PessoaJuridica pessoaJuridica = pessoaJuridicaRepo.obter(alterarId);
148             while (pessoaJuridica == null) {
149                 System.out.println("Nenhum dado encontrado com esse ID. Digite novamente: ");
150                 alterarId = scanner.nextInt();
151                 scanner.nextLine();
152                 pessoaJuridica = pessoaJuridicaRepo.obter(alterarId);
153             }
154             if (pessoaJuridica != null) {
155                 System.out.println("Dados atuais: ");
156                 System.out.println("Nome: " + pessoaJuridica.getNome());
157                 System.out.println("CNPJ: " + pessoaJuridica.getCnpj());
158
159                 System.out.println("Novos dados: ");
160                 System.out.println("Nome: ");
161                 String novoNome = scanner.nextLine();
162                 System.out.println("CNPJ: ");
163                 String novoCnpj = scanner.nextLine();
164                 scanner.nextLine();
165
166                 pessoaJuridica.setNome(novoNome);
167                 pessoaJuridica.setCnpj(novoCnpj);
168

```

```

169         System.out.println("Dados alterados com sucesso.");
170     } else {
171         System.out.println("Nenhum dado encontrado com esse ID.");
172     }
173
174     try {
175         pessoaJuridicaRepo.persistir("pessoaJuridica.bin");
176     } catch (IOException e) {
177         System.out.println("ERRO: " + e.getMessage());
178     }
179 } else {
180     System.out.println("DADO INVÁLIDO");
181 }
182 break;
183 case 3:
184     while (!tipo.equals("F") && !tipo.equals("J")) {
185         System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
186         tipo = scanner.nextLine().toUpperCase();
187
188         if (!tipo.equals("F") && !tipo.equals("J")) {
189             System.out.println("Opção inválida. Por favor, escolha F ou J.");
190         }
191     }
192
193     System.out.println("ID que será excluído: ");
194     int excluirId = scanner.nextInt();
195     scanner.nextLine();
196
197     if (tipo.equals("F")) {
198         if (pessoaFisicaRepo.obter(excluirId) != null) {
199             pessoaFisicaRepo.excluir(excluirId);
200             System.out.println("Pessoa Física excluída.");
201         } else {
202             System.out.println("Nenhum ID de Pessoa Física encontrado.");
203         }
204     } else if (tipo.equals("J")) {
205         if (pessoaJuridicaRepo.obter(excluirId) != null) {
206             pessoaJuridicaRepo.excluir(excluirId);
207             System.out.println("Pessoa Jurídica excluída.");
208         } else {
209             System.out.println("Nenhum ID de Pessoa Jurídica encontrado.2");
210         }
211     }

```

```

211     } else {
212         System.out.println("DADO INVÁLIDO");
213     }
214 }
215 break;
216 case 4:
217     while (!tipo.equals("F") && !tipo.equals("J")) {
218         System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
219         tipo = scanner.nextLine().toUpperCase();
220
221         if (!tipo.equals("F") && !tipo.equals("J")) {
222             System.out.println("Opção inválida. Por favor, escolha F ou J.");
223         }
224     }
225
226     System.out.println("ID que será visualizado: ");
227     int obterId = scanner.nextInt();
228     scanner.nextLine();
229
230     if (tipo.equals("F")) {
231         PessoaFisica pessoaFisica = pessoaFisicaRepo.obter(obterId);
232         if (pessoaFisica != null) {
233             System.out.println("Nome: " + pessoaFisica.getNome());
234             System.out.println("CPF: " + pessoaFisica.getCpf());
235             System.out.println("Idade: " + pessoaFisica.getIdade());
236         } else {
237             System.out.println("Nenhum dado encontrado por esse ID.");
238         }
239     } else if (tipo.equals("J")) {
240         PessoaJuridica pessoaJuridica = pessoaJuridicaRepo.obter(obterId);
241         if (pessoaJuridica != null) {
242             System.out.println("Nome: " + pessoaJuridica.getNome());
243             System.out.println("CNPJ: " + pessoaJuridica.getCnpj());
244         } else {
245             System.out.println("Nenhum dado encontrado por esse ID.");
246         }
247     } else {
248         System.out.println("DADO INVÁLIDO");
249     }
250 }
251 break;
252 case 5:
253     while (!tipo.equals("F") && !tipo.equals("J")) {
254         System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
255     }

```



```

253         tipo = scanner.nextLine().toUpperCase();
254
255         if (!tipo.equals("F") && !tipo.equals("J")) {
256             System.out.println("Opção inválida. Por favor, escolha F ou J.");
257         }
258     }
259
260     if (tipo.equals("F")) {
261         List<PessoaFisica> pessoasFisicas = pessoaFisicaRepo.obterTodos();
262         if (!pessoasFisicas.isEmpty()) {
263             System.out.println("PESSOAS FÍSICAS: ");
264             for (PessoaFisica pessoaFisica : pessoaFisicaRepo.obterTodos()) {
265                 System.out.println("ID: " + pessoaFisica.getId());
266                 System.out.println("Nome: " + pessoaFisica.getNome());
267                 System.out.println("CPF: " + pessoaFisica.getCpf());
268                 System.out.println("Idade: " + pessoaFisica.getIdade());
269                 System.out.println();
270             }
271         } else {
272             System.out.println("Nenhum dado encontrado de pessoas físicas.");
273         }
274     } else if (tipo.equals("J")) {
275         List<PessoaJuridica> pessoasJuridicas = pessoaJuridicaRepo.obterTodos();
276         if (!pessoasJuridicas.isEmpty()) {
277             System.out.println("PESSOAS JURÍDICAS: ");
278             for (PessoaJuridica pessoaJuridica : pessoaJuridicaRepo.obterTodos()) {
279                 System.out.println("ID: " + pessoaJuridica.getId());
280                 System.out.println("Nome: " + pessoaJuridica.getNome());
281                 System.out.println("CNPJ: " + pessoaJuridica.getCnpj());
282                 System.out.println();
283             }
284         } else {
285             System.out.println("Nenhum dado encontrado de pessoas jurídicas.");
286         }
287     } else {
288         System.out.println("DADO INVÁLIDO");
289     }
290     break;
291 case 6:
292     System.out.println("Digite o nome do arquivo a ser salvo: ");
293     String prefixo = scanner.nextLine();
294
295     try {
296         pessoaFisicaRepo.persistir(prefixo + "fisica.bin");
297         pessoaJuridicaRepo.persistir(prefixo + "juridica.bin");
298         System.out.println("DADOS SALVOS COM SUCESSO");
299     } catch (IOException e) {
300         System.out.println("ERRO: " + e.getMessage());
301     }
302     break;
303 case 7:
304     System.out.println("Digite o nome do arquivo a ser recuperado: ");
305     String prefixoRecuperado = scanner.nextLine();
306
307     try {
308         pessoaFisicaRepo.recuperar(prefixoRecuperado + "fisica.bin");
309         pessoaJuridicaRepo.recuperar(prefixoRecuperado + "juridica.bin");
310         System.out.println("DADOS RECUPERADOS COM SUCESSO");
311     } catch (IOException | ClassNotFoundException e) {
312         System.out.println("ERRO: " + e.getMessage());
313     }
314     break;
315 case 0:
316     System.out.println("Programa finalizado");
317     return;
318 default:
319     System.out.println("Opção Inválida");
320 }
321 }
322 }
323 }
324 }
325 }

```

RESULTADO

Output - CadastroPOO (run)

run:

```
=====
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 Fisica | J - Pessoa Juridica
F
Digite o id da pessoa:
1
Insira os dados...
Nome:
Ana
CPF:
11111111111
Idade:
25
=====
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 Fisica | J - Pessoa Juridica
J
Digite o id da pessoa:
3
Insira os dados...
Nome:
XPTO Sales
CNPJ:
33333333333333
=====
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
=====

0
Programa finalizado
BUILD SUCCESSFUL (total time: 1 minute 23 seconds)
```

ANÁLISE E CONCLUSÃO

Procedimento 1

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

Uma das vantagens mais significativas vejo que seria a reutilização do código em si, com a herança podemos criar classes que herdam métodos e atributos de outras, não precisando reescrever tudo novamente. Também permite alterar o comportamento de classes para adicionar novos métodos em uma subclasse. A manutenção do código, quando temos que alterar algo, com a herança, alteramos na classe base e assim todas as outras subclasses herdarão essas mudanças.

A principal desvantagem do uso de herança se dá na rigidez do código, no qual onde queremos alterar apenas um detalhe na classe, todas as outras subclasses serão alteradas e passíveis de erros. A complexidade de hierarquia pode se tornar difícil de entender e resultar em possíveis problemas.

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

Ela é necessária porque converte o código em uma sequência de bites em um arquivo para ser transmitido em rede.

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

Ela uso o paradigma funcional que permite operações de transformação e agregação de dados em coleções de uma forma mais declarativa.

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

O padrão mais comum usado é o DAO (Data Access Object).

Procedimento 2

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

São métodos ou variáveis que pertencem a própria classe ao invés de instâncias. O motivo do método main adotar esse modificador é que assim ele pode ser chamado diretamente para execução do programa sem a necessidade de depender de uma instância.

b. Para que serve a classe Scanner?

Serve para ler entradas no programa.

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

Ela impactou diretamente na manutenção do código, separando cada responsabilidade, reutilização de códigos e acesso a dados.