



# Estácio

## Estácio - Mundo 3 - Missão Nível 1

Faculdade Estácio - Polo Itaipava - Petrópolis/RJ.

Curso: Desenvolvimento Full Stack.

Disciplina: Nível 1: Iniciando o Caminho Pelo Java.  
RPG0014.

Semestre Letivo: 3.

Integrante: Jeferson Jones Smith da Rocha.

Repositório: <https://github.com/JefersonSmith/estacio-mundo3-nivel1>

IDE Utilizada: Apache NetBeans.

## **Título da Prática**

Iniciando o caminho pelo Java

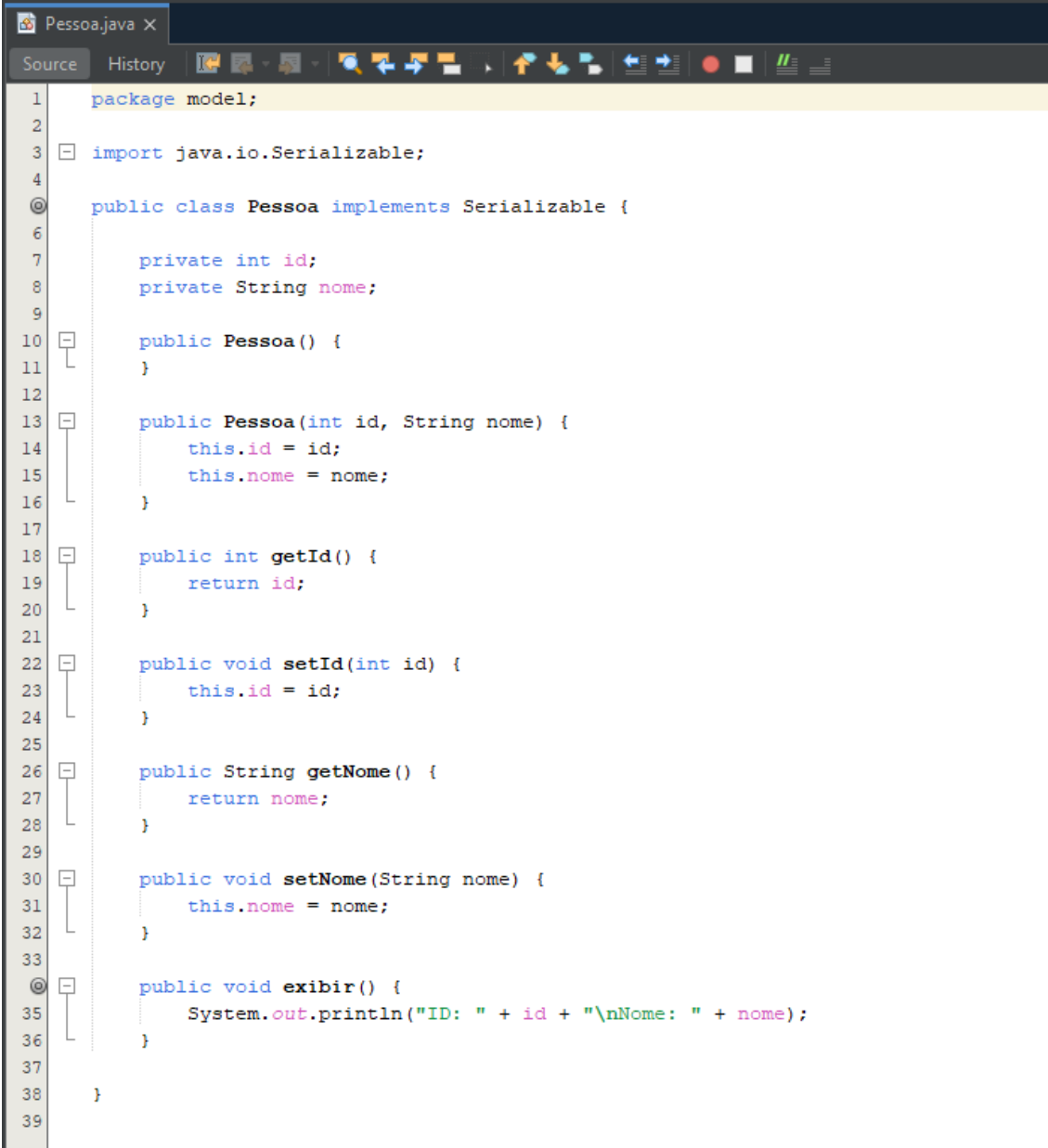
Implementação de um cadastro de clientes em modo texto, com persistência em arquivos, baseado na tecnologia Java.

## **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.

## Arquivos do Projeto

### Pessoa.java

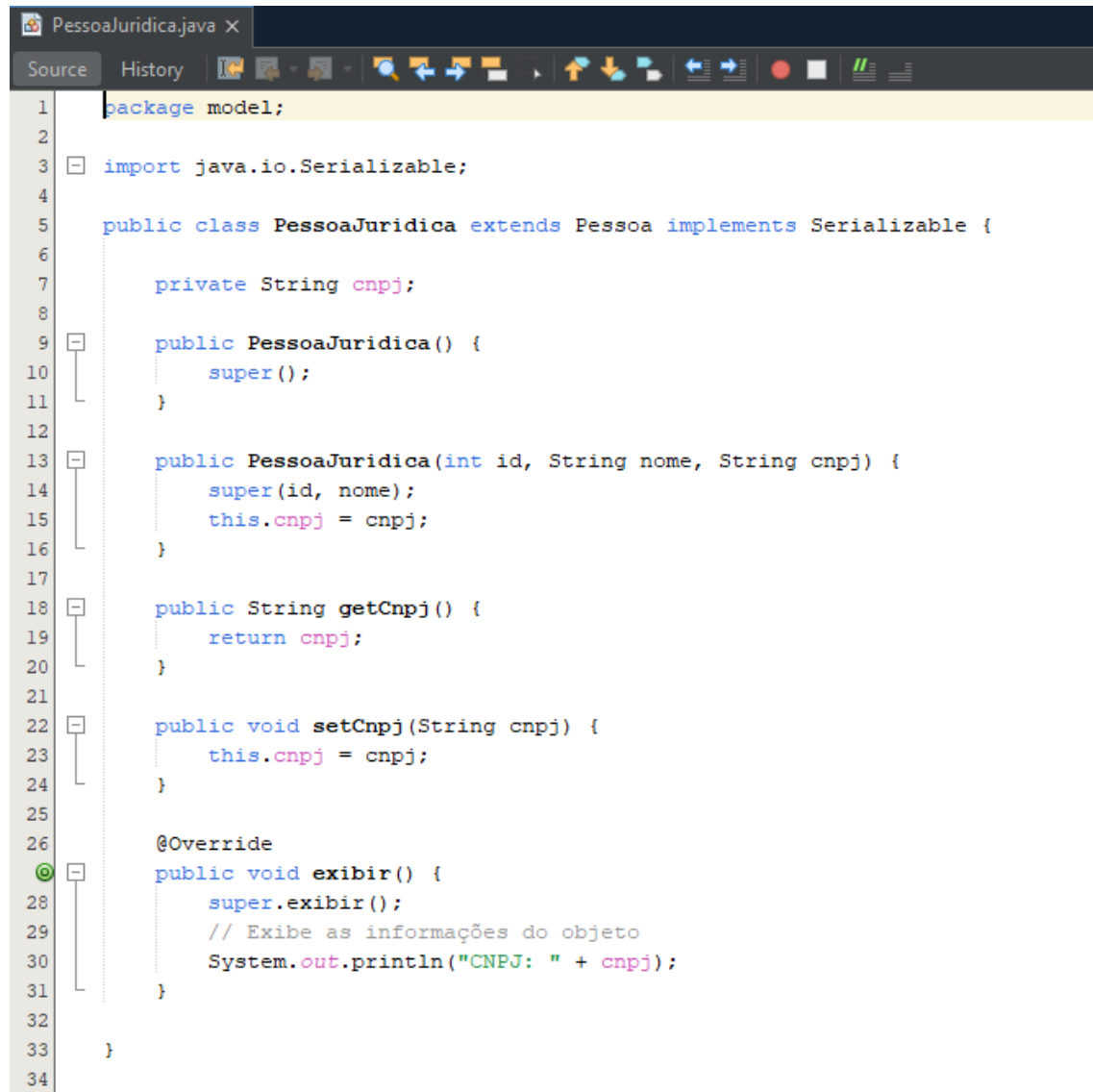


```
1 package model;
2
3 import java.io.Serializable;
4
5 public class Pessoa implements Serializable {
6     private int id;
7     private String nome;
8
9     public Pessoa() {
10     }
11
12     public Pessoa(int id, String nome) {
13         this.id = id;
14         this.nome = nome;
15     }
16
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 exibir() {
34         System.out.println("ID: " + id + "\nNome: " + nome);
35     }
36 }
37
38
39
```

## PessoaFisica.java

```
PessoaFisica.java X
Source History
1 package model;
2
3 import java.io.Serializable;
4 import model.Pessoa;
5
6 public class PessoaFisica extends Pessoa implements Serializable {
7
8     private String cpf;
9     private int idade;
10
11     public PessoaFisica() {
12         super();
13     }
14
15     public PessoaFisica(int id, String nome, String cpf, int idade) {
16         super(id, nome);
17         this.cpf = cpf;
18         this.idade = idade;
19     }
20
21     public String getCpf() {
22         return cpf;
23     }
24
25     public void setCpf(String cpf) {
26         this.cpf = cpf;
27     }
28
29     public int getIdade() {
30         return idade;
31     }
32
33     public void setIdade(int idade) {
34         this.idade = idade;
35     }
36
37     @Override
38     public void exibir() {
39         // Chama o método exibir da classe pai
40         super.exibir();
41         // Mostra as informações do objeto
42         System.out.println("CPF: " + cpf + "\nIdade: " + idade);
43     }
44 }
45
```

## PessoaJuridica.java



```
1 package model;
2
3 import java.io.Serializable;
4
5 public class PessoaJuridica extends Pessoa implements Serializable {
6
7     private String cnpj;
8
9     public PessoaJuridica() {
10         super();
11     }
12
13     public PessoaJuridica(int id, String nome, String cnpj) {
14         super(id, nome);
15         this.cnpj = cnpj;
16     }
17
18     public String getCnpj() {
19         return cnpj;
20     }
21
22     public void setCnpj(String cnpj) {
23         this.cnpj = cnpj;
24     }
25
26     @Override
27     public void exibir() {
28         super.exibir();
29         // Exibe as informações do objeto
30         System.out.println("CNPJ: " + cnpj);
31     }
32
33 }
34
```

## PessoaFisicaRepo.java

```
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 import java.util.List;
10
11 public class PessoaFisicaRepo {
12
13     private List<PessoaFisica> pessoasFisicas;
14
15     public PessoaFisicaRepo() {
16         this.pessoasFisicas = new ArrayList<>();
17     }
18
19     public void inserir(PessoaFisica pessoaFisica) {
20         pessoasFisicas.add(pessoaFisica);
21     }
22
23     public void alterar(PessoaFisica pessoaFisica) {
24         for (int i = 0; i < pessoasFisicas.size(); i++) {
25             if (pessoasFisicas.get(i).getId() == pessoaFisica.getId()) {
26                 pessoasFisicas.set(i, pessoaFisica);
27                 return;
28             }
29         }
30     }
31
32     public void excluir(int id) {
33         pessoasFisicas.removeIf(pessoaFisica -> pessoaFisica.getId() == id);
34     }
35
36     public PessoaFisica obter(int id) {
37         for (PessoaFisica pf : pessoasFisicas) {
38             if (pf.getId() == id) {
39                 return pf;
40             }
41         }
42         return null;
43     }
44
45     public List<PessoaFisica> obterTodos() {
46         return new ArrayList<>(pessoasFisicas);
47     }
48
49     public void persistir(String nomeArquivo) throws IOException {
50         try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
51             oos.writeObject(pessoasFisicas);
52         }
53     }
54
55     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
56         try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
57             pessoasFisicas = (ArrayList<PessoaFisica>) ois.readObject();
58         }
59     }
60
61     public void setLista(List<PessoaFisica> listaFisica) {
62         this.pessoasFisicas = listaFisica;
63     }
64
65 }
66
```

## PessoaJuridicaRepo.java

```
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 import java.util.List;
10
11 /**
12  *
13  * @author Smith
14  */
15 public class PessoaJuridicaRepo {
16
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) {
28         for (int i = 0; i < pessoasJuridicas.size(); i++) {
29             if (pessoasJuridicas.get(i).getId() == pessoaJuridica.getId()) {
30                 pessoasJuridicas.set(i, pessoaJuridica);
31                 return;
32             }
33         }
34     }
35
36     public void excluir(int id) {
37         pessoasJuridicas.removeIf(pessoaJuridica -> pessoaJuridica.getId() == id);
38     }
39
40     public PessoaJuridica obter(int id) {
41         for (PessoaJuridica pj : pessoasJuridicas) {
42             if (pj.getId() == id) {
43                 return pj;
44             }
45         }
46         return null;
47     }
48
49     public List<PessoaJuridica> obterTodos() {
50         return new ArrayList<>(pessoasJuridicas);
51     }
52
53     public void persistir(String nomeArquivo) throws IOException {
54         try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
55             oos.writeObject(pessoasJuridicas);
56         }
57     }
58
59     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
60         try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
61             pessoasJuridicas = (ArrayList<PessoaJuridica>) ois.readObject();
62         }
63     }
64
65     public void setLista(List<PessoaJuridica> listaJuridica) {
66         this.pessoasJuridicas = listaJuridica;
67     }
68
69 }
70
```

## Main2.java

```
1 package cadastrapoo;
2
3 /**
4  *
5  * @author Smith
6  */
7 import java.io.FileInputStream;
8 import java.io.FileOutputStream;
9 import java.io.IOException;
10 import java.io.ObjectInputStream;
11 import java.io.ObjectOutputStream;
12 import static java.lang.Integer.parseInt;
13 import java.util.List;
14 import java.util.Scanner;
15 import model.PessoaFisica;
16 import model.PessoaFisicaRepo;
17 import model.PessoaJuridica;
18 import model.PessoaJuridicaRepo;
19
20 public class Main2 {
21
22     private static Scanner sc = new Scanner(System.in);
23     private static PessoaFisicaRepo rspoFisica = new PessoaFisicaRepo();
24     private static PessoaJuridicaRepo rspoJuridica = new PessoaJuridicaRepo();
25
26     public static void main(String[] args) {
27         int opcao;
28         do {
29             mostrarMenu();
30             opcao = sc.nextInt();
31             sc.nextLine();
32             mostrarEscolha(opcao);
33         } while (opcao != 0);
34     }
35
36     private static void mostrarMenu() {
37         System.out.println("==== CADASTRO POO =====");
38         System.out.println("Selecione uma opcao:");
39         System.out.println("=====");
40         System.out.println("1 - Incluir Pessoa");
41         System.out.println("2 - Alterar Pessoa");
42         System.out.println("3 - Excluir Pessoa");
43         System.out.println("4 - Buscar pelo ID");
44         System.out.println("5 - Exibir Todos");
45         System.out.println("6 - Persistir Dados");
46         System.out.println("7 - Recuperar Dados");
47         System.out.println("0 - Finalizar Programa");
48         System.out.println("=====");
49         System.out.print("Digite uma opcao: ");
```



```

50     }
51
52     private static void mostrarEscolha(int opcao) {
53         switch (opcao) {
54             case 1 ->
55                 incluirPessoa();
56             case 2 ->
57                 alterarPessoa();
58             case 3 ->
59                 excluirPessoa();
60             case 4 ->
61                 exibirPorId();
62             case 5 ->
63                 exibirTodos();
64             case 6 ->
65                 salvarDados();
66             case 7 ->
67                 recuperarDados();
68             case 0 ->
69                 System.out.println("Encerrando...");
70             default ->
71                 System.out.println("\n Opcao invalida! ");
72         }
73     }
74
75     private static void incluirPessoa() {
76         System.out.println("Incluir Pessoa (1 - Fisica, 2 - Juridica): ");
77         int tipo = sc.nextInt();
78         sc.nextLine();
79
80         switch (tipo) {
81             case 1 -> {
82                 PessoaFisica pf = new PessoaFisica();
83                 System.out.println("Digite o ID: ");
84                 pf.setId(parseInt(sc.nextLine()));
85                 System.out.println("Digite o nome: ");
86                 pf.setNome(sc.nextLine());
87                 System.out.println("Digite o CPF: ");
88                 pf.setCpf(sc.nextLine());
89                 System.out.println("Digite a idade: ");
90                 int idade = sc.nextInt();
91                 sc.nextLine();
92                 pf.setIdade(idade);
93                 repoFisica.inserir(pf);
94                 System.out.println("\n Pessoa Fisica adicionada com sucesso! ");
95             }
96             case 2 -> {
97                 PessoaJuridica pj = new PessoaJuridica();
98                 System.out.println("Digite o ID: ");
99                 pj.setId(parseInt(sc.nextLine()));
100                 System.out.println("Digite o nome:");
101                 pj.setNome(sc.nextLine());
102                 System.out.println("Digite o CNPJ:");
103                 pj.setCnpj(sc.nextLine());
104                 repoJuridica.inserir(pj);
105                 System.out.println("\n Pessoa Juridica adicionada com sucesso! ");
106             }
107             default -> System.out.println("\n Opcao invalida. ");
108         }
109     }
110
111     private static void alterarPessoa() {
112         System.out.println("Alterar Pessoa (1 - Fisica, 2 - Juridica): ");
113         int tipo = sc.nextInt();
114         sc.nextLine();
115
116         System.out.println("Digite o ID da pessoa:");
117         int id = sc.nextInt();
118         sc.nextLine();
119
120         switch (tipo) {
121             case 1 -> {
122                 PessoaFisica pf = repoFisica.obter(id);
123                 if (pf != null) {
124                     System.out.println("Dados atuais: ");
125                     pf.exibir();
126
127                     System.out.println("Digite o nome:");
128                     String nome = sc.nextLine();
129                     if (!nome.isEmpty()) {
130                         pf.setNome(nome);
131                     }
132
133                     System.out.println("Digite o CPF:");
134                     String cpf = sc.nextLine();
135                     if (!cpf.isEmpty()) {
136                         pf.setCpf(cpf);
137                     }
138
139                     System.out.println("Digite a idade:");

```

```

140         int idade = sc.nextInt();
141         sc.nextLine();
142         if (idade != 0) {
143             pf.setIdade(idade);
144         }
145
146         repoFisica.alterar(pf);
147         System.out.println("\n Pessoa Fisica atualizada com sucesso!");
148     } else {
149         System.out.println("\n Pessoa Fisica não encontrada.");
150     }
151 }
152 case 2 -> {
153     PessoaJuridica pj = repoJuridica.obter(id);
154     if (pj != null) {
155         System.out.println("Dados atuais: ");
156         pj.exibir();
157
158         System.out.println("Digite o nome:");
159         String nome = sc.nextLine();
160         if (!nome.isEmpty()) {
161             pj.setNome(nome);
162         }
163
164         System.out.println("Digite o CNPJ:");
165         String cnpj = sc.nextLine();
166         if (!cnpj.isEmpty()) {
167             pj.setCnpj(cnpj);
168         }
169
170         repoJuridica.alterar(pj);
171         System.out.println("\n Pessoa Juridica atualizada com sucesso! ");
172     } else {
173         System.out.println("\n Pessoa Juridica não encontrada. ");
174     }
175 }
176 default -> System.out.println("\n Opcao invalida. ");
177 }
178 }
179
180 private static void excluirPessoa() {
181     System.out.println("Excluir Pessoa (1 - Fisica, 2 - Juridica): ");
182     int tipo = sc.nextInt();
183     sc.nextLine();
184

```

```

185     System.out.println("Digite o ID da pessoa a ser excluida:");
186     int id = sc.nextInt();
187     sc.nextLine();
188
189     switch (tipo) {
190         case 1 -> {
191             PessoaFisica pf = repoFisica.obter(id);
192             if (pf != null) {
193                 repoFisica.excluir(id);
194                 System.out.println("\n Pessoa Fisica removida com sucesso! ");
195             } else {
196                 System.out.println("\n Pessoa Fisica não encontrada. ");
197             }
198         }
199         case 2 -> {
200             PessoaJuridica pj = repoJuridica.obter(id);
201             if (pj != null) {
202                 repoJuridica.excluir(id);
203                 System.out.println("\n Pessoa Juridica removida com sucesso! ");
204             } else {
205                 System.out.println("\n Pessoa Juridica nao encontrada. ");
206             }
207         }
208         default -> System.out.println("\n Opcao invalida. ");
209     }
210 }
211
212 private static void exibirPorId() {
213     System.out.println("Exibir dados de Pessoa (1 - Fisica, 2 - Juridica): ");
214     int tipo = sc.nextInt();
215     sc.nextLine();
216
217     System.out.println("Digite o ID da pessoa:");
218     int id = sc.nextInt();
219     sc.nextLine();
220
221     switch (tipo) {
222         case 1 -> {
223             PessoaFisica pf = repoFisica.obter(id);
224             if (pf != null) {
225                 System.out.println("Dados da Pessoa Fisica:");
226                 pf.exibir();
227             } else {
228                 System.out.println("\n Pessoa Fisica nao encontrada. ");
229             }
230         }
231         case 2 -> {
232             PessoaJuridica pj = repoJuridica.obter(id);

```

```

233         if (pj != null) {
234             System.out.println("Dados da Pessoa Juridica:");
235             pj.exibir();
236         } else {
237             System.out.println("\n Pessoa Juridica nao encontrada. ");
238         }
239     }
240     default -> System.out.println("\n Opcao invalida. ");
241 }
242 }
243
244 private static void exibirTodos() {
245     System.out.println("Exibir todos (1 - Fisica, 2 - Juridica): ");
246     int tipo = sc.nextInt();
247     sc.nextLine();
248
249     switch (tipo) {
250         case 1 -> {
251             System.out.println("Lista de Todas as Pessoas Fisicas:");
252             for (PessoaFisica pf : repoFisica.obterTodos()) {
253                 pf.exibir();
254                 System.out.println("-----");
255             }
256         }
257         case 2 -> {
258             System.out.println("Lista de Todas as Pessoas Juridicas:");
259             for (PessoaJuridica pj : repoJuridica.obterTodos()) {
260                 pj.exibir();
261                 System.out.println("-----");
262             }
263         }
264         default -> System.out.println("\n Opcao invalida. ");
265     }
266 }
267
268 private static void recuperarDados() {
269     System.out.println("Digite o nome dos arquivos para recuperacao:");
270     String nomeArquivo = sc.nextLine();
271
272     // Recuperando dados de Pessoa Fisica
273     try (ObjectInputStream oisPF = new ObjectInputStream(new FileInputStream(nomeArquivo + ".fisica.bin"))) {
274         List<PessoaFisica> listaFisica = (List<PessoaFisica>) oisPF.readObject();
275         repoFisica.setLista(listaFisica);
276         System.out.println("\nDados de Pessoas Fisicas recuperados com sucesso. ");
277     } catch (IOException | ClassNotFoundException e) {
278         System.err.println("\n Erro ao recuperar dados de Pessoas Fisicas: " + e.getMessage());
279     }
280
281     // Recuperando dados de Pessoa Juridica
282     try (ObjectInputStream oisPJ = new ObjectInputStream(new FileInputStream(nomeArquivo + ".juridica.bin"))) {
283         List<PessoaJuridica> listaJuridica = (List<PessoaJuridica>) oisPJ.readObject();
284         repoJuridica.setLista(listaJuridica);
285         System.out.println("\n Dados de Pessoas Juridicas recuperados com sucesso. ");
286     } catch (IOException | ClassNotFoundException e) {
287         System.err.println("\n Erro ao recuperar dados de Pessoas Juridicas: " + e.getMessage());
288     }
289 }
290
291 private static void salvarDados() {
292     System.out.println("Digite o nome para salvar os arquivos:");
293     String nomeArquivo = sc.nextLine();
294
295     // Salvando dados de Pessoa Fisica
296     try (ObjectOutputStream oosPF = new ObjectOutputStream(new FileOutputStream(nomeArquivo + ".fisica.bin"))) {
297         oosPF.writeObject(repoFisica.obterTodos());
298         System.out.println("\n Dados de Pessoas Fisicas salvos com sucesso.");
299     } catch (IOException e) {
300         System.err.println("\n Erro ao salvar dados de Pessoas Fisicas: " + e.getMessage());
301     }
302
303     // Salvando dados de Pessoa Juridica
304     try (ObjectOutputStream oosPJ = new ObjectOutputStream(new FileOutputStream(nomeArquivo + ".juridica.bin"))) {
305         oosPJ.writeObject(repoJuridica.obterTodos());
306         System.out.println("\n Dados de Pessoas Juridicas salvos com sucesso. ");
307     } catch (IOException e) {
308         System.err.println("\n Erro ao salvar dados de Pessoas Juridicas: " + e.getMessage());
309     }
310 }
311 }
312
313

```

## Resultados da execução dos códigos:

```
Output X
CadastroPOO (run) X CadastroPOO - C:\Users\Smith\Repositório\NetBeansProjects\CadastroPOO X

run:
==== CADASTRO POO ====
Selecione uma opcao:
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
Digite uma opcao: 1
Incluir Pessoa (1 - Fisica, 2 - Juridica):
1
Digite o ID:
1
Digite o nome:
Jeferson
Digite o CPF:
11445577889
Digite a idade:
36

Pessoa Fisica adicionada com sucesso!
==== CADASTRO POO ====
Selecione uma opcao:
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
Digite uma opcao:
```

## **Análise e Conclusão:**

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

Elementos estáticos em Java são membros de uma classe que pertencem à própria classe, em vez de pertencerem a instâncias individuais dessa classe. Isso significa que eles são compartilhados por todas as instâncias da classe e podem ser acessados diretamente através do nome da classe, sem a necessidade de criar uma instância da classe.

### **2) Para que serve a classe Scanner?**

A classe Scanner em Java é usada para ler dados de entrada a partir de várias fontes, como o teclado (entrada padrão), arquivos ou strings. Ela oferece métodos para analisar diferentes tipos de dados, como inteiros, ponto flutuante, caracteres, entre outros, a partir da entrada fornecida.

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

O uso de classes de repositório pode ter um impacto significativo na organização do código em um projeto, especialmente em aplicativos que implementam o padrão arquitetural Repository.

O padrão Repository separa a lógica de persistência de dados do restante da aplicação, fornecendo uma interface comum para acessar e manipular dados em um sistema. As classes de repositório são responsáveis por encapsular a lógica de acesso aos dados, fornecendo métodos para recuperar, armazenar, atualizar e excluir entidades de um determinado tipo.