



Nome do Campus: Polo Castelo

Nome do Curso: Desenvolvimento FullStack

Nome da Disciplina: Iniciando o caminho pelo Java

Número da Turma: 2025.1

Semestre Letivo: Primeiro Semestre

Nome do Aluno: Roberto Birro Alves Filho

Matrícula: 202402849204

Título da Prática

Criar cadastro de clientes modo texto, persistência em arquivos, utilizando tecnologia Java.

Objetivo da Prática

Utilizar herança e polimorfismo na definição de entidades:

Criar uma hierarquia de classes (Pessoa, PessoaFisica, PessoaJuridica) para demonstrar herança e polimorfismo.

Utilizar persistência de objetos em arquivos binários:

Implementar métodos para salvar e recuperar dados em arquivos binários usando serialização.

Implementar uma interface cadastral em modo texto:

Desenvolver um menu interativo para interação com o usuário via linha de comando.

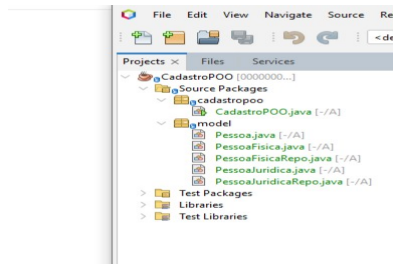
Utilizar o controle de exceções da plataforma Java:

Tratar possíveis erros durante operações como persistência e recuperação de dados.

Primeiro Procedimento:

Criação das entidades e sistema de Persistência


Códigos utilizados neste roteiro:



Classe Pessoa:

```
1 package model;
2
3 import java.io.Serializable;
4
5
6
7
8
9
10 public class Pessoa implements Serializable{
11     private int id;
12     private String nome;
13
14     // Construtor padrão
15     public Pessoa() {
16     }
17
18     // Construtor completo
19     public Pessoa(int id, String nome) {
20         this.id = id;
21         this.nome = nome;
22     }
23
24     // Método exibir
25     public void exibir() {
26         System.out.println("ID: " + id);
27         System.out.println("Nome: " + nome);
28     }
29
30     // Getters
31     public int getId() {
32         return id;
33     }
34
35     public String getNome() {
36         return nome;
37     }
38
39     // Setters
40     public void setId(int id) {
41         this.id = id;
42     }
43
44     public void setNome(String nome) {
45         this.nome = nome;
46     }
47 }
```

Classe Pessoa Fisica:



```
7  *
8  * @author Roberto Birro Alves Filho
9  */
10 public class PessoaFisica extends Pessoa implements Serializable{
11     private String cpf;
12     private int idade;
13
14     // Construtor padrão
15     public PessoaFisica() {
16         super(); // Chama o construtor padrão da classe Pessoa
17     }
18
19     // Construtor completo
20     public PessoaFisica(int id, String nome, String cpf, int idade) {
21         super(id, nome); // Chama o construtor da classe Pessoa
22         this.cpf = cpf;
23         this.idade = idade;
24     }
25
26     // Método exibir polimórfico (sobrescreve o método da classe Pessoa)
27     @Override
28     public void exibir() {
29         super.exibir(); // Chama o método exibir da classe Pessoa
30         System.out.println("CPF: " + cpf);
31         System.out.println("Idade: " + idade);
32     }
33
34     // Getters
35     public String getCpf() {
36         return cpf;
37     }
38
39     public int getIdade() {
40         return idade;
41     }
42
43     // Setters
44     public void setCpf(String cpf) {
45         this.cpf = cpf;
46     }
47
48     public void setIdade(int idade) {
49         this.idade = idade;
50     }
51
52 }
53 }
```

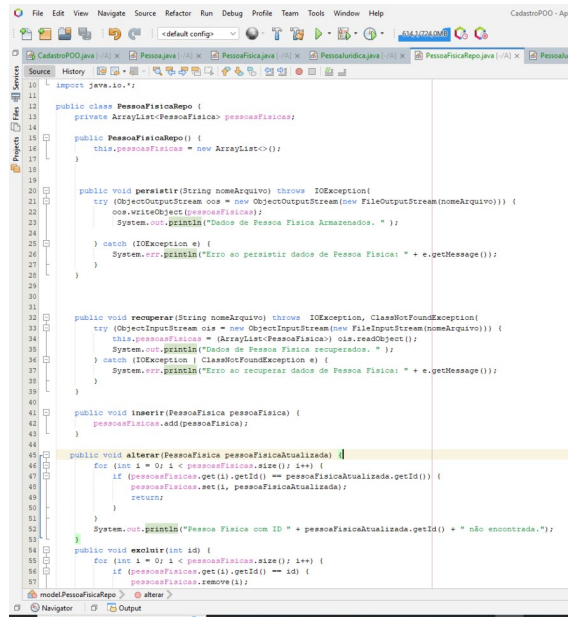
Classe Pessoa Juridica:



The screenshot shows an IDE window with the source code of the `PessoaJuridica` class. The code is in Java and is part of a package named `model`. It imports `java.io.Serializable` and extends the `Pessoa` class while implementing `Serializable`. The class has a private attribute `cnpj`. It includes two constructors: a default constructor that calls the superclass constructor, and a complete constructor that takes `id`, `nome`, and `cnpj` as parameters. There is also an `exibir` method that overrides the superclass method and prints the CNPJ value, and a `getCnpj` getter method. A `setCnpj` setter method is also present.

```
1 package model;
2
3 import java.io.Serializable;
4
5
6 /**
7  *
8  * @author Roberto Birro Alves Filho
9  */
10 public class PessoaJuridica extends Pessoa implements Serializable {
11     private String cnpj;
12
13
14     // Construtor padrão
15     public PessoaJuridica() {
16         super(); // Chama o construtor padrão da classe Pessoa
17     }
18
19     // Construtor completo
20     public PessoaJuridica(int id, String nome, String cnpj) {
21         super(id, nome); // Chama o construtor da classe Pessoa
22         this.cnpj = cnpj;
23     }
24
25     // Método exibir polimórfico (sobrescreve o método da classe Pes
26     @Override
27     public void exibir() {
28         super.exibir(); // Chama o método exibir da classe Pessoa
29         System.out.println("CNPJ: " + cnpj);
30     }
31
32     // Getter
33     public String getCnpj() {
34         return cnpj;
35     }
36
37     // Setter
38     public void setCnpj(String cnpj) {
39         this.cnpj = cnpj;
40     }
41
42
43
44 }
45
```

Classe Pessoa Fisica Repo:



```
10 import java.io.*;
11
12 public class PessoaFisicaRepo {
13     private ArrayList<PessoaFisica> pessoasFisicas;
14
15     public PessoaFisicaRepo() {
16         this.pessoasFisicas = new ArrayList<>();
17     }
18
19     public void persistir(String nomeArquivo) throws IOException {
20         try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
21             oos.writeObject(pessoasFisicas);
22             System.out.println("Dados de Pessoa Fisica Armazenados. ");
23         } catch (IOException e) {
24             System.err.println("Erro ao persistir dados de Pessoa Fisica: " + e.getMessage());
25         }
26     }
27
28     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
29         try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
30             this.pessoasFisicas = (ArrayList<PessoaFisica>) ois.readObject();
31             System.out.println("Dados de Pessoa Fisica recuperados. ");
32         } catch (IOException | ClassNotFoundException e) {
33             System.err.println("Erro ao recuperar dados de Pessoa Fisica: " + e.getMessage());
34         }
35     }
36
37     public void inserir(PessoaFisica pessoaFisica) {
38         pessoasFisicas.add(pessoaFisica);
39     }
40
41     public void alterar(PessoaFisica pessoaFisicaAtualizada) {
42         for (int i = 0; i < pessoasFisicas.size(); i++) {
43             if (pessoasFisicas.get(i).getId() == pessoaFisicaAtualizada.getId()) {
44                 pessoasFisicas.set(i, pessoaFisicaAtualizada);
45                 return;
46             }
47         }
48         System.out.println("Pessoa Fisica com ID " + pessoaFisicaAtualizada.getId() + " não encontrada.");
49     }
50
51     public void excluir(int id) {
52         for (int i = 0; i < pessoasFisicas.size(); i++) {
53             if (pessoasFisicas.get(i).getId() == id) {
54                 pessoasFisicas.remove(i);
55             }
56         }
57     }
58 }
```

Classe Pessoa Juridica Repo:

```
PessoaJuridicaRepo.java
1 package model;
2
3 import java.io.*;
4
5 public class PessoaJuridicaRepo {
6     private ArrayList<PessoaJuridica> pessoas = new ArrayList<>();
7
8     // Método para inserir uma nova PessoaJuridica
9     public void inserir(PessoaJuridica pessoa) {
10         pessoas.add(pessoa);
11     }
12
13     // Método para alterar uma PessoaJuridica existente
14     public void alterar(PessoaJuridica pessoa) {
15         for (int i = 0; i < pessoas.size(); i++) {
16             if (pessoas.get(i).getId() == pessoa.getId()) {
17                 pessoas.set(i, pessoa);
18                 break;
19             }
20         }
21     }
22
23     // Método para excluir uma PessoaJuridica pelo ID
24     public void excluir(int id) {
25         pessoas.removeIf(p -> p.getId() == id);
26     }
27
28     // Método para obter uma PessoaJuridica pelo ID
29     public PessoaJuridica obter(int id) {
30         for (PessoaJuridica pessoa : pessoas) {
31             if (pessoa.getId() == id) {
32                 return pessoa;
33             }
34         }
35         return null; // Retorna null se não encontrar
36     }
37
38     // Método para obter todas as PessoaJuridica
39     public ArrayList<PessoaJuridica> obterTodos() {
40         return pessoas;
41     }
42
43     // Método para persistir os dados em um arquivo binário
44     public void persistir(String nomeArquivo) throws IOException {
45         try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
46             outputStream.writeObject(pessoas);
47         }
48     }
49
50     // Método para recuperar os dados de um arquivo binário
51     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
52         try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
53             pessoas = (ArrayList<PessoaJuridica>) inputStream.readObject();
54         }
55     }
56 }
57
58 }
```

Classe Principal:

```
CadastroPOO.java
13 *
14 * @author Roberto Birro Alves Filho
15 */
16 public class CadastroPOO {
17
18     /**
19      * @param args the command line arguments
20      */
21     public static void main(String[] args) {
22         Scanner scanner = new Scanner(System.in);
23         PessoaFisicaRepo pessoaFisicaRepo = new PessoaFisicaRepo();
24         PessoaJuridicaRepo pessoaJuridicaRepo = new PessoaJuridicaRepo();
25
26         int opcao;
27         do {
28             System.out.println("\n--- Menu ---");
29             System.out.println("1 - Incluir");
30             System.out.println("2 - Alterar");
31             System.out.println("3 - Excluir");
32             System.out.println("4 - Obter por ID");
33             System.out.println("5 - Obter todos");
34             System.out.println("6 - Salvar dados");
35             System.out.println("7 - Recuperar dados");
36             System.out.println("0 - Sair");
37             System.out.print("Escolha uma opção: ");
38             opcao = scanner.nextInt();
39             scanner.nextLine(); // Consumir a quebra de linha
40
41             switch (opcao) {
42                 case 1:
43                     incluir(scanner, pessoaFisicaRepo, pessoaJuridicaRepo);
44                     break;
45                 case 2:
46                     alterar(scanner, pessoaFisicaRepo, pessoaJuridicaRepo);
47                     break;
48                 case 3:
49                     excluir(scanner, pessoaFisicaRepo, pessoaJuridicaRepo);
50                     break;
51                 case 4:
52                     obterPorId(scanner, pessoaFisicaRepo, pessoaJuridicaRepo);
53                     break;
54                 case 5:
55                     obterTodos(scanner, pessoaFisicaRepo, pessoaJuridicaRepo);
56                     break;
57                 case 6:
58                     salvarDados(scanner, pessoaFisicaRepo, pessoaJuridicaRepo);
59                     break;
60                 case 0:
61                     sair();
62                     break;
63                 default:
64                     System.out.println("Opção inválida. Tente novamente.");
65             }
66         } while (opcao != 0);
67     }
68
69     private static void incluir(Scanner scanner, PessoaFisicaRepo pessoaFisicaRepo, PessoaJuridicaRepo pessoaJuridicaRepo) {
70         System.out.println("Incluir Pessoa");
71         System.out.println("1 - Pessoa Física");
72         System.out.println("2 - Pessoa Jurídica");
73         int tipo = scanner.nextInt();
74         scanner.nextLine();
75
76         if (tipo == 1) {
77             PessoaFisica pessoaFisica = new PessoaFisica();
78             pessoaFisica.setNome(scanner.nextLine());
79             pessoaFisica.setCPF(scanner.nextLong());
80             pessoaFisica.setEndereco(scanner.nextLine());
81             pessoaFisicaRepo.incluir(pessoaFisica);
82         } else if (tipo == 2) {
83             PessoaJuridica pessoaJuridica = new PessoaJuridica();
84             pessoaJuridica.setNome(scanner.nextLine());
85             pessoaJuridica.setCNPJ(scanner.nextLong());
86             pessoaJuridica.setEndereco(scanner.nextLine());
87             pessoaJuridicaRepo.incluir(pessoaJuridica);
88         }
89     }
90
91     private static void alterar(Scanner scanner, PessoaFisicaRepo pessoaFisicaRepo, PessoaJuridicaRepo pessoaJuridicaRepo) {
92         System.out.println("Alterar Pessoa");
93         System.out.println("1 - Pessoa Física");
94         System.out.println("2 - Pessoa Jurídica");
95         int tipo = scanner.nextInt();
96         scanner.nextLine();
97
98         if (tipo == 1) {
99             PessoaFisica pessoaFisica = new PessoaFisica();
100             pessoaFisica.setNome(scanner.nextLine());
101             pessoaFisica.setCPF(scanner.nextLong());
102             pessoaFisica.setEndereco(scanner.nextLine());
103             pessoaFisicaRepo.alterar(pessoaFisica);
104         } else if (tipo == 2) {
105             PessoaJuridica pessoaJuridica = new PessoaJuridica();
106             pessoaJuridica.setNome(scanner.nextLine());
107             pessoaJuridica.setCNPJ(scanner.nextLong());
108             pessoaJuridica.setEndereco(scanner.nextLine());
109             pessoaJuridicaRepo.alterar(pessoaJuridica);
110         }
111     }
112
113     private static void excluir(Scanner scanner, PessoaFisicaRepo pessoaFisicaRepo, PessoaJuridicaRepo pessoaJuridicaRepo) {
114         System.out.println("Excluir Pessoa");
115         System.out.println("1 - Pessoa Física");
116         System.out.println("2 - Pessoa Jurídica");
117         int tipo = scanner.nextInt();
118         scanner.nextLine();
119
120         if (tipo == 1) {
121             PessoaFisica pessoaFisica = new PessoaFisica();
122             pessoaFisica.setNome(scanner.nextLine());
123             pessoaFisica.setCPF(scanner.nextLong());
124             pessoaFisicaRepo.excluir(pessoaFisica.getId());
125         } else if (tipo == 2) {
126             PessoaJuridica pessoaJuridica = new PessoaJuridica();
127             pessoaJuridica.setNome(scanner.nextLine());
128             pessoaJuridica.setCNPJ(scanner.nextLong());
129             pessoaJuridicaRepo.excluir(pessoaJuridica.getId());
130         }
131     }
132
133     private static void obterPorId(Scanner scanner, PessoaFisicaRepo pessoaFisicaRepo, PessoaJuridicaRepo pessoaJuridicaRepo) {
134         System.out.println("Obter Pessoa por ID");
135         System.out.println("1 - Pessoa Física");
136         System.out.println("2 - Pessoa Jurídica");
137         int tipo = scanner.nextInt();
138         scanner.nextLine();
139
140         if (tipo == 1) {
141             PessoaFisica pessoaFisica = pessoaFisicaRepo.obter(scanner.nextLong());
142             if (pessoaFisica != null) {
143                 System.out.println("Pessoa Física encontrada:");
144                 System.out.println("Nome: " + pessoaFisica.getNome());
145                 System.out.println("CPF: " + pessoaFisica.getCPF());
146                 System.out.println("Endereço: " + pessoaFisica.getEndereco());
147             } else {
148                 System.out.println("Pessoa Física não encontrada.");
149             }
150         } else if (tipo == 2) {
151             PessoaJuridica pessoaJuridica = pessoaJuridicaRepo.obter(scanner.nextLong());
152             if (pessoaJuridica != null) {
153                 System.out.println("Pessoa Jurídica encontrada:");
154                 System.out.println("Nome: " + pessoaJuridica.getNome());
155                 System.out.println("CNPJ: " + pessoaJuridica.getCNPJ());
156                 System.out.println("Endereço: " + pessoaJuridica.getEndereco());
157             } else {
158                 System.out.println("Pessoa Jurídica não encontrada.");
159             }
160         }
161     }
162
163     private static void obterTodos(Scanner scanner, PessoaFisicaRepo pessoaFisicaRepo, PessoaJuridicaRepo pessoaJuridicaRepo) {
164         System.out.println("Obter todas as Pessoas");
165         System.out.println("1 - Pessoa Física");
166         System.out.println("2 - Pessoa Jurídica");
167         int tipo = scanner.nextInt();
168         scanner.nextLine();
169
170         if (tipo == 1) {
171             ArrayList<PessoaFisica> pessoasFisicas = pessoaFisicaRepo.obterTodos();
172             if (pessoasFisicas != null) {
173                 System.out.println("Pessoas Físicas encontradas:");
174                 for (PessoaFisica pessoaFisica : pessoasFisicas) {
175                     System.out.println("Nome: " + pessoaFisica.getNome());
176                     System.out.println("CPF: " + pessoaFisica.getCPF());
177                     System.out.println("Endereço: " + pessoaFisica.getEndereco());
178                 }
179             } else {
180                 System.out.println("Nenhuma Pessoa Física encontrada.");
181             }
182         } else if (tipo == 2) {
183             ArrayList<PessoaJuridica> pessoasJuridicas = pessoaJuridicaRepo.obterTodos();
184             if (pessoasJuridicas != null) {
185                 System.out.println("Pessoas Jurídicas encontradas:");
186                 for (PessoaJuridica pessoaJuridica : pessoasJuridicas) {
187                     System.out.println("Nome: " + pessoaJuridica.getNome());
188                     System.out.println("CNPJ: " + pessoaJuridica.getCNPJ());
189                     System.out.println("Endereço: " + pessoaJuridica.getEndereco());
190                 }
191             } else {
192                 System.out.println("Nenhuma Pessoa Jurídica encontrada.");
193             }
194         }
195     }
196
197     private static void salvarDados(Scanner scanner, PessoaFisicaRepo pessoaFisicaRepo, PessoaJuridicaRepo pessoaJuridicaRepo) {
198         System.out.println("Salvar dados");
199         System.out.println("1 - Pessoa Física");
200         System.out.println("2 - Pessoa Jurídica");
201         int tipo = scanner.nextInt();
202         scanner.nextLine();
203
204         if (tipo == 1) {
205             PessoaFisica pessoaFisica = new PessoaFisica();
206             pessoaFisica.setNome(scanner.nextLine());
207             pessoaFisica.setCPF(scanner.nextLong());
208             pessoaFisica.setEndereco(scanner.nextLine());
209             pessoaFisicaRepo.persistir("dados_fisicas.dat");
210         } else if (tipo == 2) {
211             PessoaJuridica pessoaJuridica = new PessoaJuridica();
212             pessoaJuridica.setNome(scanner.nextLine());
213             pessoaJuridica.setCNPJ(scanner.nextLong());
214             pessoaJuridica.setEndereco(scanner.nextLine());
215             pessoaJuridicaRepo.persistir("dados_juridicas.dat");
216         }
217     }
218
219     private static void sair() {
220         System.out.println("Saindo do programa.");
221     }
222 }
```

Resultado da execução:

```
Principal [Java Application] C:\Users\MAURICIO\Desktop\sts-4.27.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.5.v20241023-1957\jre\bin\javaw.exe (11 de mar. de 2025 17:03:13 elapsed: 0:00:21) [pid: 12864]
|
|== Menu ==
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Sair
Escolha uma opção:
```

Análise e Conclusão

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

Vantagens:

Reutilização de código: Classes derivadas herdaram comportamentos e atributos da superclasse.

Organização: Facilita a estruturação do código em hierarquias lógicas.

Polimorfismo: Permite tratar objetos de diferentes tipos de forma uniforme.

Desvantagens:

Complexidade: Hierarquias profundas podem dificultar a manutenção.

Rigidez: Alterações na superclasse podem impactar todas as subclasses.

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

A interface **Serializable** é um marcador que indica que uma classe pode ser convertida em bytes para ser armazenada ou transmitida. Sem ela, o Java não sabe como serializar os objetos, resultando em erros durante a persistência.

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

A API Stream permite processar coleções de forma declarativa, utilizando conceitos como:

Funções lambda: Para definir operações concisas.

Operações intermediárias e terminais: Ex.: filter, map, collect.

Imutabilidade: Os dados originais não são alterados.

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

O padrão mais comum é o DAO (Data Access Object) , que separa a lógica de acesso aos dados da lógica de negócios. No projeto, isso foi implementado nas classes **PessoaFisicaRepo** e **PessoaJuridicaRepo**.

