Faculdade Estácio
Curso: **Desenvolvimento Full Stack**
Disciplina: **RPG0014 - Iniciando o caminho pelo Java**
Turma: **23.3**
Semestre letivo: **2024.2**
Luiz Antonio Soares Damasceno Neto

TÍTULO DA PRÁTICA:

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.

CÓDIGOS SOLICITADOS NESTE ROTEIRO DE AULA

Código Procedimento 1:

CadastroPOO.java [-/M]

Source | History

```java
package cadastropoo;

import model.PessoaFisica;
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;
import java.io.IOException;
import java.util.ArrayList;

public class CadastroPOO {

    public static void main(String[] args) {
        try {
            PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

            repo1.inserir(new PessoaFisica(1, "Ana", "11111111111", 25));
            repo1.inserir(new PessoaFisica(2, "Carlos", "22222222222", 52));

            String arquivoPessoaFisica = "pessoas_fisicas.bin";
            repo1.persistir(arquivoPessoaFisica);
            System.out.println("Dados de Pessoa Fisica Armazenados.");

            PessoaFisicaRepo repo2 = new PessoaFisicaRepo();

            repo2.recuperar(arquivoPessoaFisica);
            System.out.println("Dados de Pessoa Fisica Recuperados.");

            ArrayList<PessoaFisica> pessoasFisicas = repo2.obterTodos();
            for (PessoaFisica pf : pessoasFisicas) {
                System.out.println("Id: " + pf.getId());
                System.out.println("Nome: " + pf.getNome());
                System.out.println("CPF: " + pf.getCpf());
                System.out.println("Idade: " + pf.getIdade());
            }

            PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();

            repo3.inserir(new PessoaJuridica(3, "XPTO Sales", "33333333333333"));
            repo3.inserir(new PessoaJuridica(4, "XPTO Solutions", "44444444444444"));

            String arquivoPessoaJuridica = "pessoas_juridicas.bin";
            repo3.persistir(arquivoPessoaJuridica);
            System.out.println("Dados de Pessoa Juridica Armazenados.");

            PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();

            repo4.recuperar(arquivoPessoaJuridica);
            System.out.println("Dados de Pessoa Juridica Recuperados.");
```

Output

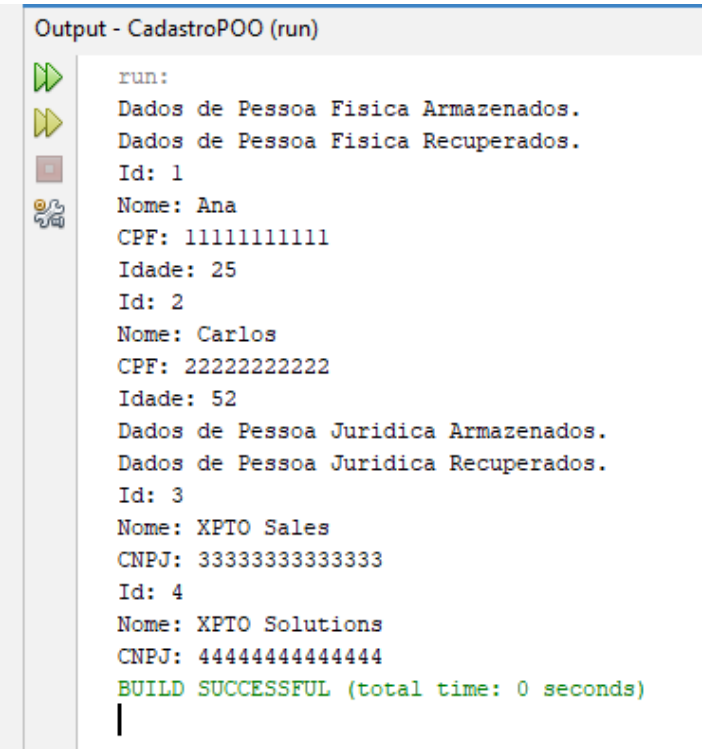
```java
            repo4.recuperar(arquivoPessoaJuridica);
            System.out.println("Dados de Pessoa Juridica Recuperados.");

            ArrayList<PessoaJuridica> pessoasJuridicas = repo4.obterTodos();
            for (PessoaJuridica pj : pessoasJuridicas) {
                System.out.println("Id: " + pj.getId());
                System.out.println("Nome: " + pj.getNome());
                System.out.println("CNPJ: " + pj.getCnpj());
            }

        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao persistir ou recuperar dados: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Estácio

Resultado da execução Procedimento 1:



Estácio

Código Procedimento 2:

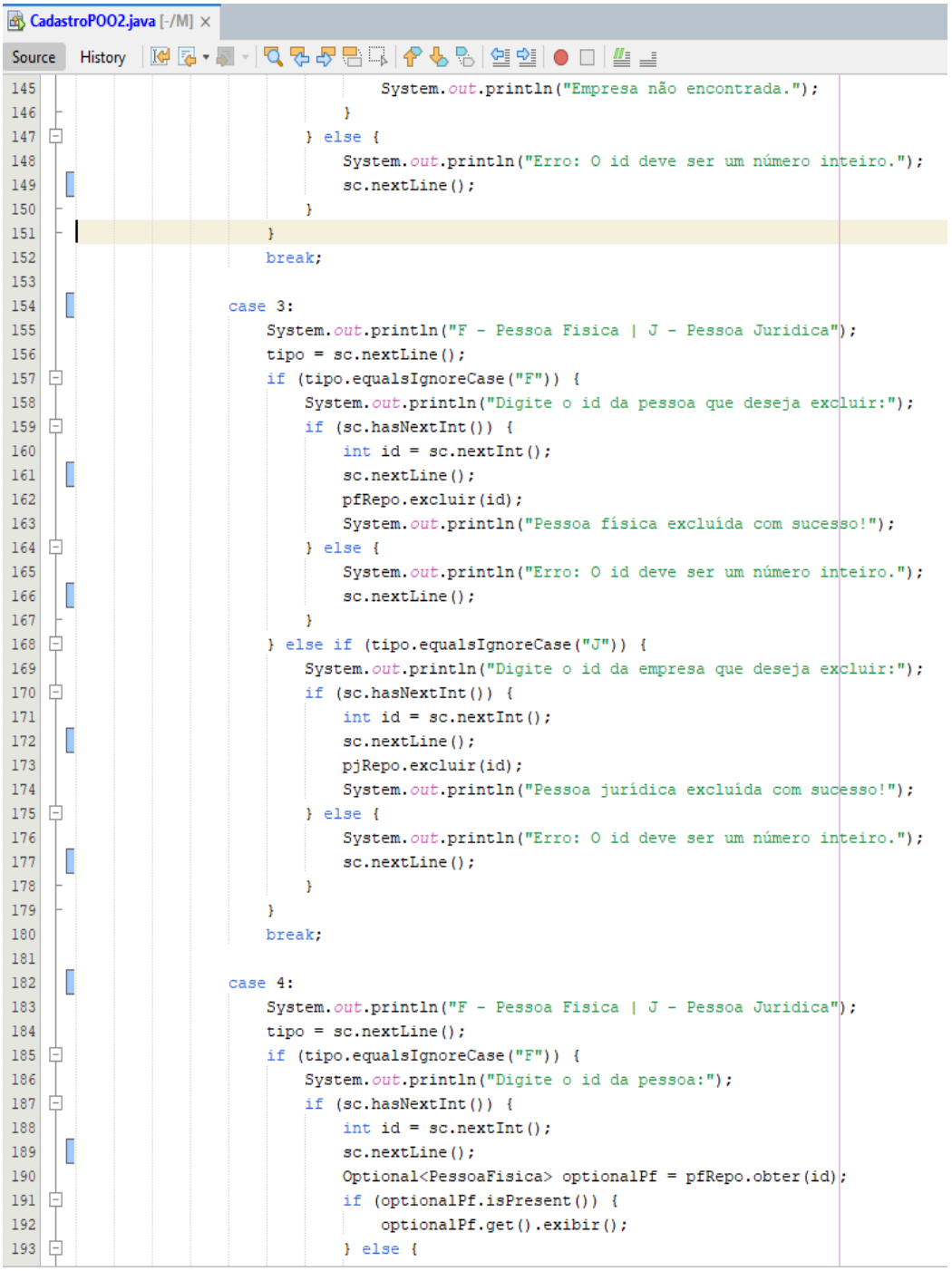
CadastroPOO2.java [-/M]

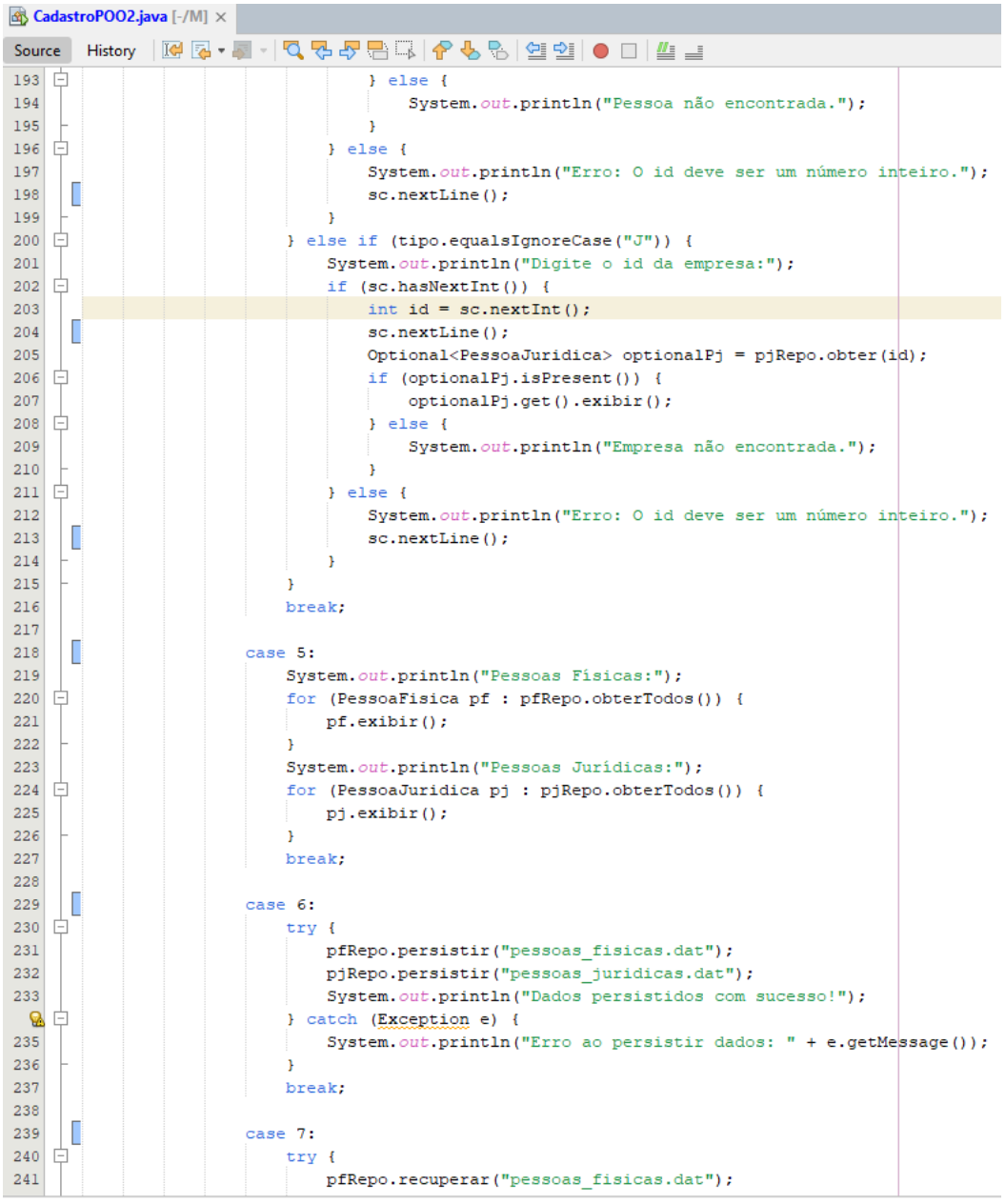
Source | History

```
1  package cadastropoo2;
2
3  import java.util.Optional;
4  import java.util.Scanner;
5  import model.PessoaFisica;
6  import model.PessoaJuridica;
7  import model.PessoaFisicaRepo;
8  import model.PessoaJuridicaRepo;
9
10 public class CadastroPOO2 {
11
12     public static void main(String[] args) {
13         Scanner sc = new Scanner(System.in);
14         PessoaFisicaRepo pfRepo = new PessoaFisicaRepo();
15         PessoaJuridicaRepo pjRepo = new PessoaJuridicaRepo();
16         String tipo;
17
18         int opcao = -1;
19         do {
20             System.out.println("====================");
21             System.out.println("1 - Incluir Pessoa");
22             System.out.println("2 - Alterar Pessoa");
23             System.out.println("3 - Excluir Pessoa");
24             System.out.println("4 - Buscar pelo Id");
25             System.out.println("5 - Exibir Todos");
26             System.out.println("6 - Persistir Dados");
27             System.out.println("7 - Recuperar Dados");
28             System.out.println("0 - Finalizar Programa");
29             System.out.println("====================");
30
31             while (!sc.hasNextInt()) {
32                 System.out.println("Entrada inválida. Por favor, insira um número.");
33                 sc.next();
34             }
35             opcao = sc.nextInt();
36             sc.nextLine();
37
38             switch (opcao) {
39                 case 1:
40                     System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
41                     tipo = sc.nextLine();
42                     if (tipo.equalsIgnoreCase("F")) {
43                         System.out.println("Digite o id da pessoa:");
44                         if (sc.hasNextInt()) {
45                             int id = sc.nextInt();
46                             sc.nextLine();
47                             System.out.println("Digite o nome da pessoa:");
48                             String nome = sc.nextLine();
49                             System.out.println("Digite o CPF da pessoa:");
```


CadastroPOO2.java [-/M]
Source
History
System.out.println("Digite o CPF da pessoa:");
String cpf = sc.nextLine();
System.out.println("Digite a idade da pessoa:");
if (sc.hasNextInt()) {
int idade = sc.nextInt();
sc.nextLine();
PessoaFisica pf = new PessoaFisica(id, nome, cpf, idade);
pfRepo.inserir(pf);
System.out.println("Pessoa física incluída com sucesso!");
} else {
System.out.println("Erro: A idade deve ser um número inteiro.");
sc.nextLine();
}
} else {
System.out.println("Erro: O id deve ser um número inteiro.");
sc.nextLine();
}
} else if (tipo.equalsIgnoreCase("J")) {
System.out.println("Digite o id da empresa:");
if (sc.hasNextInt()) {
int id = sc.nextInt();
sc.nextLine();
System.out.println("Digite o nome da empresa:");
String nome = sc.nextLine();
System.out.println("Digite o CNPJ da empresa:");
String cnpj = sc.nextLine();
PessoaJuridica pj = new PessoaJuridica(id, nome, cnpj);
pjRepo.inserir(pj);
System.out.println("Pessoa jurídica incluída com sucesso!");
} else {
System.out.println("Erro: O id deve ser um número inteiro.");
sc.nextLine();
}
}
break;
case 2:
System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
tipo = sc.nextLine();
if (tipo.equalsIgnoreCase("F")) {
System.out.println("Digite o id da pessoa que deseja alterar:");
if (sc.hasNextInt()) {
int id = sc.nextInt();
sc.nextLine();
Optional<PessoaFisica> optionalPf = pfRepo.obter(id);
if (optionalPf.isPresent()) {
PessoaFisica pf = optionalPf.get();


CadastroPOO2.java [-/M]
Source
History
PessoaFisica pf = optionalPf.get();
System.out.println("Dados atuais: ");
pf.exibir();
System.out.println("Digite o novo nome da pessoa:");
String nome = sc.nextLine();
System.out.println("Digite o novo CPF da pessoa:");
String cpf = sc.nextLine();
System.out.println("Digite a nova idade da pessoa:");
if (sc.hasNextInt()) {
int idade = sc.nextInt();
sc.nextLine();
pf.setNome(nome);
pf.setCpf(cpf);
pf.setIdade(idade);
pfRepo.alterar(pf);
System.out.println("Pessoa física alterada com sucesso!");
} else {
System.out.println("Erro: A idade deve ser um número inteiro.");
sc.nextLine();
}
} else {
System.out.println("Pessoa não encontrada.");
}
} else {
System.out.println("Erro: O id deve ser um número inteiro.");
sc.nextLine();
}
} else if (tipo.equalsIgnoreCase("J")) {
System.out.println("Digite o id da empresa que deseja alterar:");
if (sc.hasNextInt()) {
int id = sc.nextInt();
sc.nextLine();
Optional<PessoaJuridica> optionalPj = pjRepo.obter(id);
if (optionalPj.isPresent()) {
PessoaJuridica pj = optionalPj.get();
System.out.println("Dados atuais: ");
pj.exibir();
System.out.println("Digite o novo nome da empresa:");
String nome = sc.nextLine();
System.out.println("Digite o novo CNPJ da empresa:");
String cnpj = sc.nextLine();
pj.setNome(nome);
pj.setCnpj(cnpj);
pjRepo.alterar(pj);
System.out.println("Pessoa jurídica alterada com sucesso!");
} else {
System.out.println("Empresa não encontrada.");


CadastroPOO2.java [-/M]
Source
History
System.out.println("Empresa não encontrada.");
} else {
System.out.println("Erro: O id deve ser um número inteiro.");
sc.nextLine();
}
}
break;
case 3:
System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
tipo = sc.nextLine();
if (tipo.equalsIgnoreCase("F")) {
System.out.println("Digite o id da pessoa que deseja excluir:");
if (sc.hasNextInt()) {
int id = sc.nextInt();
sc.nextLine();
pfRepo.excluir(id);
System.out.println("Pessoa física excluída com sucesso!");
} else {
System.out.println("Erro: O id deve ser um número inteiro.");
sc.nextLine();
}
} else if (tipo.equalsIgnoreCase("J")) {
System.out.println("Digite o id da empresa que deseja excluir:");
if (sc.hasNextInt()) {
int id = sc.nextInt();
sc.nextLine();
pjRepo.excluir(id);
System.out.println("Pessoa jurídica excluída com sucesso!");
} else {
System.out.println("Erro: O id deve ser um número inteiro.");
sc.nextLine();
}
}
break;
case 4:
System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
tipo = sc.nextLine();
if (tipo.equalsIgnoreCase("F")) {
System.out.println("Digite o id da pessoa:");
if (sc.hasNextInt()) {
int id = sc.nextInt();
sc.nextLine();
Optional<PessoaFisica> optionalPf = pfRepo.obter(id);
if (optionalPf.isPresent()) {
optionalPf.get().exibir();
} else {


CadastroPOO2.java [-/M]
Source
History
} else {
System.out.println("Pessoa não encontrada.");
}
} else {
System.out.println("Erro: O id deve ser um número inteiro.");
sc.nextLine();
}
} else if (tipo.equalsIgnoreCase("J")) {
System.out.println("Digite o id da empresa:");
if (sc.hasNextInt()) {
int id = sc.nextInt();
sc.nextLine();
Optional<PessoaJuridica> optionalPj = pjRepo.obter(id);
if (optionalPj.isPresent()) {
optionalPj.get().exibir();
} else {
System.out.println("Empresa não encontrada.");
}
} else {
System.out.println("Erro: O id deve ser um número inteiro.");
sc.nextLine();
}
}
break;
case 5:
System.out.println("Pessoas Fisicas:");
for (PessoaFisica pf : pfRepo.obterTodos()) {
pf.exibir();
}
System.out.println("Pessoas Jurídicas:");
for (PessoaJuridica pj : pjRepo.obterTodos()) {
pj.exibir();
}
break;
case 6:
try {
pfRepo.persistir("pessoas_fisicas.dat");
pjRepo.persistir("pessoas_juridicas.dat");
System.out.println("Dados persistidos com sucesso!");
} catch (Exception e) {
System.out.println("Erro ao persistir dados: " + e.getMessage());
}
break;
case 7:
try {
pfRepo.recuperar("pessoas_fisicas.dat");

Estácio

CadastroPOO2.java [-/M]

Source | History

```
228
229                     case 6:
230                         try {
231                             pfRepo.persistir("pessoas_fisicas.dat");
232                             pjRepo.persistir("pessoas_juridicas.dat");
233                             System.out.println("Dados persistidos com sucesso!");
235                         } catch (Exception e) {
236                             System.out.println("Erro ao persistir dados: " + e.getMessage());
237                         }
238                         break;
239
240                     case 7:
241                         try {
242                             pfRepo.recuperar("pessoas_fisicas.dat");
243                             pjRepo.recuperar("pessoas_juridicas.dat");
245                             System.out.println("Dados recuperados com sucesso!");
246                         } catch (Exception e) {
247                             System.out.println("Erro ao recuperar dados: " + e.getMessage());
248                         }
249                         break;
250
251                     case 0:
252                         System.out.println("Programa finalizado.");
253                         break;
254
255                     default:
256                         System.out.println("Opção inválida.");
257                         break;
258                 }
259             } while (opcao != 0);
260             sc.close();
        }
    }
```

Resultado da execução Procedimento 2:

Output - **CadastroPOO2 (run) #2**

```
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:
120
Digite o nome da pessoa:
```

ANÁLISE E CONCLUSÃO PROCEDIMENTOS 1 E 2

- ❖ PROCEDIMENTO 1

Quais as vantagens e desvantagens do uso de herança?

Vantagens:

- **Reuso de código:** permite aproveitar funcionalidades já implementadas em classes base.
- **Manutenção facilitada:** mudanças na classe base são automaticamente aplicadas às classes derivadas.
- **Organização hierárquica:** representa relações "é um", criando uma estrutura lógica e clara.

Desvantagens:

- **Aumento da dependência:** alterações na classe base podem impactar classes derivadas.
- **Dificuldade de compreensão:** heranças complexas podem tornar o código difícil de entender.
- **Rigidez:** limita flexibilidade, pois o comportamento das subclasses é fortemente acoplado à superclasse.

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

A interface **Serializable** é necessária porque indica ao Java que um objeto pode ser convertido em uma sequência de bytes, permitindo que seja gravado e lido de um arquivo binário. Sem essa interface, a JVM não saberia como transformar o objeto para persistência.

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

O paradigma funcional é usado na API Stream através de operações como **map, filter e reduce**, que permitem processar coleções de dados de forma

declarativa, sem modificar os objetos originais. Essas operações são compostas por expressões lambda e executadas de forma eficiente, promovendo imutabilidade e melhor desempenho.

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

No Java, a persistência de dados em arquivos costuma seguir o padrão **Data Access Object (DAO)** ou **Repositório**. Essas classes isolam a lógica de persistência dos dados, facilitando a manutenção e o reaproveitamento de código.

- PROCEDIMENTO 2

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

Elementos estáticos pertencem à classe e não a instâncias de objetos. O método **`main`** é estático porque ele deve ser acessível sem a criação de uma instância da classe, sendo o ponto de entrada do programa.

Para que serve a classe Scanner?

A classe **Scanner** é usada para ler entradas do usuário (por exemplo, através do teclado) ou de arquivos, facilitando a manipulação de dados de entrada em diferentes tipos primitivos (int, String, etc.).

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

O uso de classes de repositório melhorou a organização, separando a lógica de manipulação de dados (como inserção, alteração e exclusão) da lógica de interface ou regras de negócio. Isso promoveu maior coesão e facilitou a manutenção e evolução do código.

Estácio

ENDEREÇOS

Endereço Procedimento 1:
https://github.com/NetoDamasceno/cadastropoo-procedimento1.git

Endereço Procedimento 2:
https://github.com/NetoDamasceno/cadastropoo-procedimento2.git