Campus: Polo Ouro Fino
Curso: Tecnólogo em Desenvolvimento Full Stack
Disciplina: Iniciando o caminho pelo Java
Turma: RPG0014 / 9002
Semestre letivo: 1º semestre de 2025
Aluno: Luiz Guilherme de Souza
Repositório: https://github.com/luiz-guilherme-s/projeto-java-crud

---

Título da Prática

Criação de um sistema de cadastro de clientes em modo texto, com funcionalidades de CRUD, persistência em arquivos binários e recuperação de dados.

Objetivo da Prática

Desenvolver um sistema simples de cadastro de clientes (Pessoa Física e Pessoa Jurídica), utilizando programação orientada a objetos em Java, aplicando conceitos de repositório, persistência de dados em arquivos binários e menu em modo texto para interação com o usuário.

Código-Fonte

Os códigos completos estão organizados no repositório indicado. A estrutura principal está contida nas seguintes classes:

- `CadastrroPOO.java`
- `Pessoa.java`, `PessoaFisica.java`, `PessoaJuridica.java`
- `PessoaJuridicaRepo.java`,`PessoaFisicaRepo.java`

Análise e Conclusão

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 individuais. O método `main` precisa ser `static` para que o Java consiga executá-lo sem precisar criar uma instância da classe.

Para que serve a classe Scanner?

A classe `Scanner` é utilizada para capturar entradas do usuário, como números e textos, diretamente do console (teclado), permitindo a interação com o programa.

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

As classes de repositório permitiram uma separação clara entre a lógica de negócio (manipulação dos dados) e a lógica de apresentação (menu). Isso tornou o código mais limpo, reutilizável e de fácil manutenção.

Quais as vantagens e desvantagens do uso de herança?

Vantagens:

- Reutilização de código;
- Organização hierárquica;
- Facilita a manutenção.

Desvantagens:

- Pode aumentar o acoplamento entre classes;
- Dificuldade para entender hierarquias complexas.

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

A interface `Serializable` permite que objetos Java sejam convertidos em uma sequência de bytes para que possam ser armazenados em arquivos binários. Sem ela, o processo de persistência e recuperação de objetos não é possível.

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

A API `Stream` do Java permite o uso de funções como `map`, `filter`, `forEach`, entre outras, promovendo um estilo funcional de programação. Isso facilita o processamento de coleções de forma concisa, legível e eficiente.

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

Geralmente é usado o padrão DAO (Data Access Object), que encapsula o acesso aos dados, separando essa responsabilidade da lógica principal do programa. Neste projeto, isso foi feito pelas classes de repositório com métodos de persistência e recuperação.

Output - CadastroPOO (run) #5

```
run:

=== MENU PRINCIPAL ===
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Sair
Op��o: 1

=== INCLUIR PESSOA ===
1. Pessoa F�sica
2. Pessoa Jur�dica
Tipo: 1
ID: 10
Nome: Zidane
CPF: 510510510
Idade: 32
Pessoa f�sica cadastrada com sucesso!
```

```
=== MENU PRINCIPAL ===
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Sair
Op��o: 1

=== INCLUIR PESSOA ===
1. Pessoa F�sica
2. Pessoa Jur�dica
Tipo: 2
ID: 11
Nome: Ronaldinho Gaucho
CNPJ: 251251251
Pessoa jur�dica cadastrada com sucesso!
```

```
=== MENU PRINCIPAL ===
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Sair
Op��o: 2

=== ALTERAR PESSOA ===
1. Pessoa F�sica
2. Pessoa Jur�dica
Tipo: 1
ID da pessoa a alterar: 10
Dados atuais:
PF - ID: 10, Nome: Zidane, CPF: 510510510, Idade: 32
Novo nome: joao roberto
Novo CPF: 510510510510
Nova idade: 45
Pessoa f�sica alterada com sucesso!
```

```
=== MENU PRINCIPAL ===
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Sair
Op��o: 3

=== EXCLUIR PESSOA ===
1. Pessoa F�sica
2. Pessoa Jur�dica
Tipo: 1
ID da pessoa a excluir: 10
Pessoa f�sica exclu�da com sucesso!
```

```
=== MENU PRINCIPAL ===
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Sair
Op��o: 5

=== EXIBIR TODOS ===
1. Pessoa F�sica
2. Pessoa Jur�dica
Tipo: 1

Pessoas F�sicas:
PF - ID: 12, Nome: roberto carlos, CPF: 222222222, Idade: 28
PF - ID: 14, Nome: lucio, CPF: 88888888, Idade: 24

=== MENU PRINCIPAL ===
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Sair
Op��o: 5

=== EXIBIR TODOS ===
1. Pessoa F�sica
2. Pessoa Jur�dica
Tipo: 2

Pessoas Jur�dicas:
PJ - ID: 11, Nome: Ronaldinho Gaucho, CNPJ: 251251251
PJ - ID: 13, Nome: maradona, CNPJ: 999999999
PJ - ID: 15, Nome: dembel�, CNPJ: 94785462
```

```
=== MENU PRINCIPAL ===
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Sair
Op��o: 4

=== EXIBIR POR ID ===
1. Pessoa F�sica
2. Pessoa Jur�dica
Tipo: 1
ID da pessoa: 14
[DEBUG] Tentando buscar ID: 14
[DEBUG] IDs atualmente cadastrados:
-> 12 | roberto carlos
-> 14 | lucio
PF - ID: 14, Nome: lucio, CPF: 88888888, Idade: 24
```

```
=== MENU PRINCIPAL ===
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Sair
Op��o: 6

=== SALVAR DADOS ===
Prefixo para os arquivos: Dados salvos com sucesso!
```

```
run:

=== MENU PRINCIPAL ===
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Sair
Op��o: 7

=== RECUPERAR DADOS ===
Prefixo dos arquivos: Dados recuperados com sucesso!
```