



Faculdade Estácio - POLO FREGUESIA - RIO DE JANEIRO - RJ

Curso: Desenvolvimento Full Stack

Disciplina: Iniciando O Caminho Pelo Java

Número da Turma: 9001

Semestre Letivo: 2025.1

Integrante: Gabriel Galvão Reis

Repositório: <https://github.com/galvao-reis/estacio-missao-pratica-3-1>

Iniciando o Caminho pelo Java

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

Objetivo 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.

Sumário

Objetivo da Prática.....	1
Sumário.....	2
Arquivos desenvolvidos pela Prática.....	3
Resultado da execução do código.....	9
Análise e Conclusão.....	10

Arquivos desenvolvidos pela Prática

1. cadastroPOO.java

```
1 package cadastropoo;
2
3 import model.*;
4 import java.io.*;
5
6 public class CadastroPOO {
7
8     /**
9      * @param args the command line arguments
10     */
11     public static void main(String[] args) {
12
13         PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
14
15         repo1.inserir(new PessoaFisica(1, "Ana", "1111111111", 25));
16         repo1.inserir(new PessoaFisica(2, "Carlos", "2222222222", 52));
17         try{
18             repo1.persistir( "RepositorioPessoasFisicas" );
19         }
20         catch(IOException exception){
21             System.out.println(exception.toString());
22         }
23
24         PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
25         try{
26             repo2.recuperar( "RepositorioPessoasFisicas" );
27         }
28         catch(IOException | ClassNotFoundException exception){
29             System.out.println(exception.toString());
30         }
31
32         for (PessoaFisica pessoa : repo2.obterTodos()){
33             pessoa.exibir();
34         }
35
36         PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
37         repo3.inserir(new PessoaJuridica(3, "XPTO Sales", "3333333333333333"));
38         repo3.inserir(new PessoaJuridica(4, "XPTO Solutions", "444444444444444444"));
39
40         try{
41             repo3.persistir( "RepositorioPessoasJuridicas" );
42         }
43         catch(IOException exception){
44             System.out.println(exception.toString());
45         }
46
47         PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
48         try{
49             repo4.recuperar( "RepositorioPessoasJuridicas" );
50         }
51         catch( IOException | ClassNotFoundException exception){
52             System.out.println(exception.toString());
53         }
54
55         for (PessoaJuridica pessoa : repo4.obterTodos()){
56             pessoa.exibir();
57         }
58     }
59 }
```

2. Pessoa.java

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

3. PessoaFisica.java

```
1  package model;
2
3  import java.io.Serializable;
4
5  public class PessoaFisica extends Pessoa implements Serializable {
6
7      private String cpf;
8      private int idade;
9
10     public PessoaFisica(int id,String nome, String cpf, int idade){
11         super(id,nome);
12         this.cpf = cpf;
13         this.idade = idade;
14     }
15
16     public String getCpf(){
17         return this.cpf;
18     }
19
20     public void setCpf(String cpf) {
21         this.cpf = cpf;
22     }
23
24     public int getIdade(){
25         return this.idade;
26     }
27     public void setIdade(int idade){
28         this.idade = idade;
29     }
30
31     @Override
32     public void exibir(){
33         System.out.println(
34             "Id: " + this.getId() +
35             "\nNome: " + this.getNome() +
36             "\nCPF: " + this.getCpf() +
37             ",\nIdade: " + this.getIdade());
38     }
39 }
40
41
```

4. 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      public PessoaJuridica(int id, String nome, String cnpj){
9          super(id, nome);
10         this.cnpj = cnpj;
11     }
12
13     public String getCnpj(){
14         return this.cnpj;
15     }
16
17     public void setCnpj(String cnpj){
18         this.cnpj = cnpj;
19     }
20
21     @Override
22     public void exibir(){
23         System.out.println(
24             "Id: " + this.getId() +
25             "\nNome: " + this.getNome() +
26             "\nCNPJ: " + this.getCnpj());
27     }
28
29 }
30
```

5. PessoaFisicaRepo.java

```
1  package model;
2
3  import java.io.*;
4  import java.util.ArrayList;
5
6  public class PessoaFisicaRepo{
7      private ArrayList<PessoaFisica> pessoasFisicas;
8
9      public PessoaFisicaRepo(){
10         this.pessoasFisicas = new ArrayList<PessoaFisica>();
11     }
12
13     public void inserir( PessoaFisica pessoa ){
14         this.pessoasFisicas.add(pessoa);
15     }
16
17     public void alterar( PessoaFisica pessoa ){
18         this.excluir(pessoa.getId());
19         this.inserir(pessoa);
20     }
21
22     public void excluir( int id ){
23         this.pessoasFisicas.removeIf( pessoa -> pessoa.getId() == id);
24     }
25
26     public PessoaFisica obter( int id){
27         for (PessoaFisica pessoa : this.pessoasFisicas){
28             if (pessoa.getId() == id){
29                 return pessoa;
30             }
31         }
32         return null;
33     }
34
35     public ArrayList<PessoaFisica> obterTodos(){
36         return this.pessoasFisicas;
37     }
38
39     public void persistir(String nomeArquivo)throws IOException{
40         FileOutputStream fos = new FileOutputStream(nomeArquivo);
41         ObjectOutputStream oos = new ObjectOutputStream(fos);
42         oos.writeObject(this.pessoasFisicas);
43     }
44
45     public void recuperar ( String nomeArquivo) throws IOException, ClassNotFoundException{
46         FileInputStream fis = new FileInputStream(nomeArquivo);
47         ObjectInputStream ois = new ObjectInputStream(fis);
48         this.pessoasFisicas = (ArrayList<PessoaFisica>) ois.readObject();
49     }
50 }
51
```

6. PessoaJuridicaRepo.java

```
1  package model;
2
3  import java.io.*;
4  import java.util.ArrayList;
5
6  public class PessoaJuridicaRepo{
7
8      private ArrayList<PessoaJuridica> pessoasJuridicas;
9
10     public PessoaJuridicaRepo(){
11         this.pessoasJuridicas = new ArrayList<PessoaJuridica>();
12     }
13
14     public void inserir(PessoaJuridica pessoa){
15         this.pessoasJuridicas.add(pessoa);
16     }
17
18     public void alterar(PessoaJuridica pessoa){
19         this.excluir(pessoa.getId());
20         this.inserir(pessoa);
21     }
22
23     public void excluir( int id ){
24         this.pessoasJuridicas.removeIf( pessoa -> pessoa.getId() == id );
25     }
26
27     public PessoaJuridica obter( int id ){
28         for (PessoaJuridica pessoa : pessoasJuridicas){
29             if (pessoa.getId() == id){
30                 return pessoa;
31             }
32         }
33         return null;
34     }
35
36     public ArrayList<PessoaJuridica> obterTodos(){
37         return this.pessoasJuridicas;
38     }
39
40     public void persistir( String nomeArquivo ) throws IOException{
41         FileOutputStream fos = new FileOutputStream(nomeArquivo);
42         ObjectOutputStream oos = new ObjectOutputStream(fos);
43         oos.writeObject(this.pessoasJuridicas);
44     }
45
46     public void recuperar( String nomeArquivo) throws IOException, ClassNotFoundException {
47         FileInputStream fis = new FileInputStream(nomeArquivo);
48         ObjectInputStream ois = new ObjectInputStream(fis);
49         this.pessoasJuridicas = (ArrayList<PessoaJuridica>) ois.readObject();
50     }
51 }
52
53
54
```


Resultado da execução do código

Output - CadastroPOO (run)



Id: 2



Nome: Carlos

CPF: 22222222222,



Idade: 52



Id: 3

Nome: XPTO Sales

CNPJ: 33333333333333

Id: 4

Nome: XPTO Solutions

CNPJ: 4444444444444444

BUILD SUCCESSFUL (total time: 0 seconds)

Análise e Conclusão

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

A herança permite a reutilização de código. Por permitir que as classes filhas tenham as mesmas propriedades e atributos da classe mãe, ela evita que haja duplicação de código, facilitando a manutenção e implementação de eventuais modificações.

No entanto, com o aumento da complexidade do projeto, a herança pode dificultar a modificação e manutenção do código, uma vez que é possível se criar hierarquias complexas. Outro problema da herança é que ela quebra o princípio de encapsulamento, já que a classe filha sempre depende da classe mãe, tornando o sistema mais frágil.

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

A interface Serializable é necessária pois ela sinaliza que os objetos da classe que a implementa são capazes de serem serializados. Caso não implementada, a tentativa de serializar um objeto irá causar uma exceção do tipo "NotSerializableException"

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

O API Stream do Java é utilizado para processar coleções de objetos. Uma Stream é uma sequência de objetos que suportam diversos métodos que podem ser encadeados para gerar o resultado desejado sem que ocorram modificações nos dados originais.

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

Para tratar da persistência de dados em arquivos, utiliza-se o JPA - Java Persistence API. No entanto, ele requer que se utilize um provedor, tal quais Hibernate, Apache OpenJPA e ObjectDB dentre outros.