



## **RPG0014 - Iniciando o caminho pelo Java**

**Jéssica Maria de Carvalho Matrícula: 202209187939**

**POLO JARDIM SÃO BERNARDO - SÃO PAULO - SP**

**Nível 1: Iniciando o Caminho Pelo Java – 9003 – 3º**

**Endereço do Repositório GIT: <https://github.com/Jessicac30/CadastroPOO1>**

### **Objetivo da Prática**

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

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.

## 1º Procedimento | Criação das Entidades e Sistema de Persistência

### Organização dos arquivos

CadastroPOO/

|— src/

| |— cadastro/

| | |— Main.java

| |— model/

| | |— Pessoa.java

| | |— PessoaFisica.java

| | |— PessoaFisicaRepo.java

| | |— PessoaJuridica.java

| | |— PessoaJuridicaRepo.java

```
Main.java X
Main.java
1 package cadastro;
2
3 import model.PessoaFisica;
4 import model.PessoaFisicaRepo;
5 import model.PessoaJuridica;
6 import model.PessoaJuridicaRepo;
7
8 public class Main {
9
10     public static void main(String[] args) {
11         PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
12
13         PessoaFisica pessoaFisica1 = new PessoaFisica(1, "Anma", "1111111111", 25);
14         PessoaFisica pessoaFisica2 = new PessoaFisica(2, "Carlos", "2222222222", 52);
15         repo1.inserir(pessoaFisica1);
16         repo1.inserir(pessoaFisica2);
17
18         String nomeArquivoPessoasFisicas = "pessoasfisicas.dat";
19         try {
20             repo1.persistir(nomeArquivoPessoasFisicas);
21             System.out.println("Dados de Pessoa Física Armazenados.");
22         } catch (Exception e) {
23             System.out.println("Erro ao persistir os dados de Pessoa Física: " + e.getMessage());
24         }
25
26         PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
27
28         try {
29             repo2.recuperar(nomeArquivoPessoasFisicas);
30             System.out.println("Dados de Pessoa Física Recuperados.");
31             for (PessoaFisica pessoa : repo2.obterTodos()) {
32                 System.out.println("Id: " + pessoa.getId());
33                 System.out.println("Nome: " + pessoa.getNome());
34                 System.out.println("CPF: " + pessoa.getCpf());
35                 System.out.println("Idade: " + pessoa.getIdade());
36             }
37         } catch (Exception e) {
38             System.out.println("Erro ao recuperar os dados de Pessoa Física: " + e.getMessage());
39         }
40
41
```

```
41         PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
42
43         PessoaJuridica pessoaJuridica1 = new PessoaJuridica(3, "xptro sales", "33333333333333");
44         PessoaJuridica pessoaJuridica2 = new PessoaJuridica(4, "xpto solutions", "44444444444444");
45         repo3.inserir(pessoaJuridica1);
46         repo3.inserir(pessoaJuridica2);
47
48         String nomeArquivoPessoasJuridicas = "pessoasjuridicas.dat";
49         try {
50             repo3.persistir(nomeArquivoPessoasJuridicas);
51             System.out.println("Dados de Pessoa Jurídica Armazenados.");
52         } catch (Exception e) {
53             System.out.println("Erro ao persistir os dados de Pessoa Jurídica: " + e.getMessage());
54         }
55
56         PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
57
58         try {
59             repo4.recuperar(nomeArquivoPessoasJuridicas);
60             System.out.println("Dados de Pessoa Jurídica Recuperados.");
61             for (PessoaJuridica pessoa : repo4.obterTodos()) {
62                 System.out.println("Id: " + pessoa.getId());
63                 System.out.println("Nome: " + pessoa.getNome());
64                 System.out.println("CNPJ: " + pessoa.getCnpj());
65             }
66         } catch (Exception e) {
67             System.out.println("Erro ao recuperar os dados de Pessoa Jurídica: " + e.getMessage());
68         }
69     }
70 }
71
```

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

```
PessoaFisica.java X
PessoaFisica.java
1 package model;
2 import java.io.Serializable;
3
4 public class PessoaFisica extends Pessoa implements Serializable {
5     private String cpf;
6     private int idade;
7
8     public PessoaFisica(int id, String nome, String cpf, int idade) {
9         super(id, nome);
10        this.cpf = cpf;
11        this.idade = idade;
12    }
13
14    public String getCpf() {
15        return cpf;
16    }
17
18    public void setCpf(String cpf) {
19        this.cpf = cpf;
20    }
21
22    public int getIdade() {
23        return idade;
24    }
25
26    public void setIdade(int idade) {
27        this.idade = idade;
28    }
29
30    @Override
31    public String toString() {
32        return super.toString() + ", CPF: " + cpf + ", Idade: " + idade;
33    }
34 }
35
```

```

PessoaFisicaRepo.java X
PessoaFisicaRepo.java
1  package model;
2
3  import java.io.IOException;
4  import java.io.ObjectInputStream;
5  import java.io.FileInputStream;
6  import java.io.ObjectOutputStream;
7  import java.io.FileOutputStream;
8  import java.util.ArrayList;
9
10 public class PessoaFisicaRepo {
11     private ArrayList<PessoaFisica> pessoasFisicas;
12
13     public PessoaFisicaRepo() {
14         pessoasFisicas = new ArrayList<>();
15     }
16
17     public void persistir(String nomeArquivo) throws IOException {
18         try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
19             outputStream.writeObject(pessoasFisicas);
20         }
21     }
22
23     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
24         try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
25             pessoasFisicas = (ArrayList<PessoaFisica>) inputStream.readObject();
26         }
27     }
28
29     public void inserir(PessoaFisica pessoaFisica) {
30         pessoasFisicas.add(pessoaFisica);
31     }
32
33     public void alterar(PessoaFisica pessoaFisica) {
34         for (int i = 0; i < pessoasFisicas.size(); i++) {
35             if (pessoasFisicas.get(i).getId() == pessoaFisica.getId()) {
36                 pessoasFisicas.set(i, pessoaFisica);
37                 return;
38             }
39         }
40     }

```

```

41
42     public void excluir(int id) {
43         for (int i = 0; i < pessoasFisicas.size(); i++) {
44             if (pessoasFisicas.get(i).getId() == id) {
45                 pessoasFisicas.remove(i);
46                 return;
47             }
48         }
49     }
50
51     public PessoaFisica obter(int id) {
52         for (PessoaFisica pessoaFisica : pessoasFisicas) {
53             if (pessoaFisica.getId() == id) {
54                 return pessoaFisica;
55             }
56         }
57         return null;
58     }
59
60     public ArrayList<PessoaFisica> obterTodos() {
61         return pessoasFisicas;
62     }
63 }
64

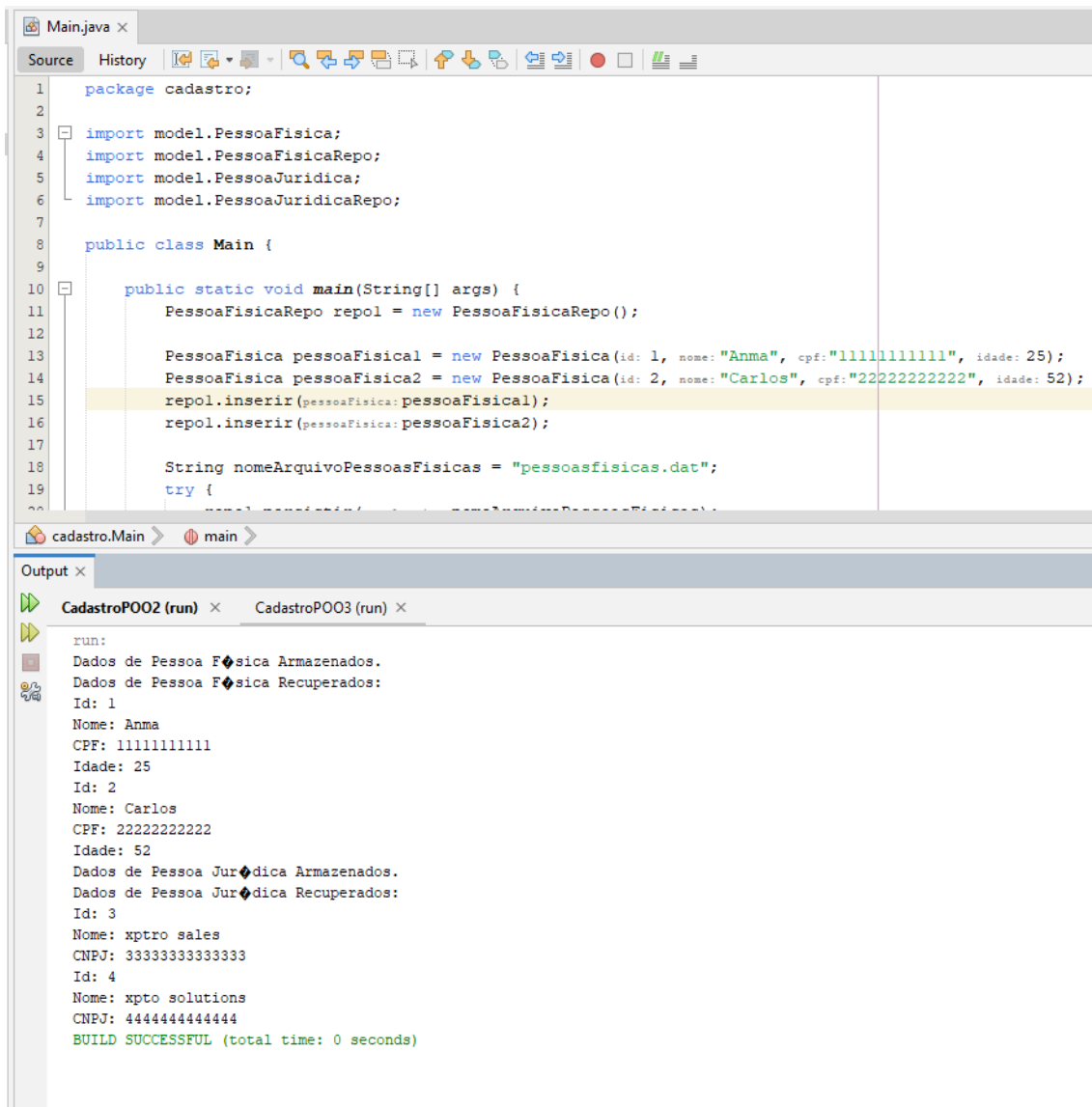
```

```
PessoaJuridica.java X
PessoaJuridica.java
1  package model;
2
3  import java.io.Serializable;
4
5  public class PessoaJuridica extends Pessoa implements Serializable {
6      private String cnpj;
7
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 cnpj;
15     }
16
17     public void setCnpj(String cnpj) {
18         this.cnpj = cnpj;
19     }
20
21     @Override
22     public String toString() {
23         return super.toString() + ", CNPJ: " + cnpj;
24     }
25
26 }
27
```

```
PessoaJuridicaRepo.java X
PessoaJuridicaRepo.java
1  package model;
2
3  import java.io.IOException;
4  import java.io.ObjectInputStream;
5  import java.io.FileInputStream;
6  import java.io.ObjectOutputStream;
7  import java.io.FileOutputStream;
8  import java.util.ArrayList;
9
10 public class PessoaJuridicaRepo {
11     private ArrayList<PessoaJuridica> pessoasJuridicas;
12
13     public PessoaJuridicaRepo() {
14         pessoasJuridicas = new ArrayList<>();
15     }
16
17     public void persistir(String nomeArquivo) throws IOException {
18         try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
19             outputStream.writeObject(pessoasJuridicas);
20         }
21     }
22
23     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
24         try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
25             pessoasJuridicas = (ArrayList<PessoaJuridica>) inputStream.readObject();
26         }
27     }
28
29     public void inserir(PessoaJuridica pessoaJuridica) {
30         pessoasJuridicas.add(pessoaJuridica);
31     }
32
33     public void alterar(PessoaJuridica pessoaJuridica) {
34         for (int i = 0; i < pessoasJuridicas.size(); i++) {
35             if (pessoasJuridicas.get(i).getId() == pessoaJuridica.getId()) {
36                 pessoasJuridicas.set(i, pessoaJuridica);
37                 return;
38             }
39         }
40     }
41 }
```

```
41
42     public void excluir(int id) {
43         for (int i = 0; i < pessoasJuridicas.size(); i++) {
44             if (pessoasJuridicas.get(i).getId() == id) {
45                 pessoasJuridicas.remove(i);
46                 return;
47             }
48         }
49     }
50
51     public PessoaJuridica obter(int id) {
52         for (PessoaJuridica pessoaJuridica : pessoasJuridicas) {
53             if (pessoaJuridica.getId() == id) {
54                 return pessoaJuridica;
55             }
56         }
57         return null;
58     }
59
60     public ArrayList<PessoaJuridica> obterTodos() {
61         return pessoasJuridicas;
62     }
63 }
64
```

## Resultado



The screenshot displays an IDE window with a Java source file named `Main.java`. The code defines a package `cadastro` and imports classes from the `model` package. It includes a `Main` class with a `main` method that creates two `PessoaFisica` objects, inserts them into a repository, and attempts to save them to a file named `pepsoasfisicas.dat`. Below the source editor, the `Output` window shows the execution results for `CadastroPOO2 (run)` and `CadastroPOO3 (run)`. The output indicates that data for physical persons was successfully stored and retrieved, while data for legal persons was not. The build process completed successfully in 0 seconds.

```
1 package cadastro;
2
3 import model.PessoaFisica;
4 import model.PessoaFisicaRepo;
5 import model.PessoaJuridica;
6 import model.PessoaJuridicaRepo;
7
8 public class Main {
9
10     public static void main(String[] args) {
11         PessoaFisicaRepo repol = new PessoaFisicaRepo();
12
13         PessoaFisica pessoaFisical = new PessoaFisica(id: 1, nome: "Anma", cpf: "1111111111", idade: 25);
14         PessoaFisica pessoaFisica2 = new PessoaFisica(id: 2, nome: "Carlos", cpf: "2222222222", idade: 52);
15         repol.inserir(pessoaFisica: pessoaFisical);
16         repol.inserir(pessoaFisica: pessoaFisica2);
17
18         String nomeArquivoPessoasFisicas = "pepsoasfisicas.dat";
19         try {
20             repol.salvarArquivo(pessoaFisica: pessoaFisical, nomeArquivoPessoasFisicas);
21         } catch (IOException e) {
22             e.printStackTrace();
23         }
24     }
25 }
```

run:  
Dados de Pessoa Física Armazenados.  
Dados de Pessoa Física Recuperados:  
Id: 1  
Nome: Anma  
CPF: 1111111111  
Idade: 25  
Id: 2  
Nome: Carlos  
CPF: 2222222222  
Idade: 52  
Dados de Pessoa Jurídica Armazenados.  
Dados de Pessoa Jurídica Recuperados:  
Id: 3  
Nome: xptro sales  
CNPJ: 33333333333333  
Id: 4  
Nome: xpto solutions  
CNPJ: 44444444444444  
BUILD SUCCESSFUL (total time: 0 seconds)



## **Análise e Conclusão**

### **Vantagens do Uso de Herança:**

**Reutilização de Código:** Herança permite a reutilização de código ao herdar atributos e métodos de uma classe pai, evitando a repetição de implementações similares em várias classes.

**Extensibilidade:** Novas classes podem ser criadas com base em classes existentes, permitindo a extensão e personalização das funcionalidades.

**Polimorfismo:** Herança possibilita o polimorfismo, onde objetos de classes derivadas podem ser tratados como objetos da classe base, facilitando a manipulação de objetos de diferentes tipos de maneira uniforme.

### **Desvantagens do Uso de Herança:**

**Acoplamento:** Herança pode criar um alto grau de acoplamento entre classes, tornando o código mais difícil de manter e entender.

**Fragilidade:** Modificações na classe base podem afetar todas as classes derivadas, o que pode ser problemático em sistemas complexos.

**Hierarquias Complexas:** Hierarquias de herança profundas e complexas podem tornar o código difícil de gerenciar e entender.

### **A Interface Serializable na Persistência em Arquivos Binários:**

A interface `Serializable` é necessária ao efetuar persistência em arquivos binários porque ela permite que objetos sejam serializados, ou seja, transformados em uma sequência de bytes que podem ser gravados em um arquivo e posteriormente desserializados para recriar o objeto. Isso é essencial para salvar e carregar objetos em sistemas, como o Java, que precisam armazenar dados em arquivos binários. A interface `Serializable`

fornece uma marcação para o sistema de serialização do Java, indicando quais objetos podem ser serializados de forma segura.

#### Uso do Paradigma Funcional pela API Stream no Java:

A API Stream no Java utiliza o paradigma funcional para operações de processamento de coleções de dados. Ela permite que você execute operações como mapeamento, filtragem, redução e ordenação em coleções de maneira mais declarativa e funcional. Isso resulta em código mais conciso e legível, facilitando a manipulação de grandes conjuntos de dados de forma eficiente. A API Stream utiliza conceitos como funções lambda e expressões funcionais para alcançar esses objetivos.

#### Padrão de Desenvolvimento na Persistência de Dados em Arquivos em Java:

No desenvolvimento em Java, um padrão comum na persistência de dados em arquivos é o uso de serialização de objetos. A serialização permite que objetos sejam gravados em arquivos binários de maneira simples e eficaz, preservando a estrutura dos objetos. Além disso, o uso de classes de entrada e saída, como `ObjectInputStream` e `ObjectOutputStream`, é comum para manipular a leitura e escrita de objetos serializados. Esse padrão é amplamente utilizado para salvar e carregar dados em aplicativos Java que requerem persistência de dados.