

**Informações:**

- Nome: Felipe de Souza Komatsu
- Semestre: 3°
- Turma: 9001
- Campus: 147 POLO ALCÂNTARA - SÃO GONÇALO - RJ
- Curso: Desenvolvimento Full Stack
- Github: <https://github.com/Felppss>

**1º procedimento: Criação das Entidades e Sistema de Persistência****Objetivos da prática:**

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

- Utilizar herança e polimorfismo na definição de entidades.
- Utilizar persistência de objetos em arquivos binários.
- Implementar uma interface cadastral em modo texto.
- Utilizar o controle de exceções da plataforma Java.
- 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:**Pessoa.java

```
/**
 * @author Felipe komatsu
 */
package model.entidades;

import java.io.Serializable;

public class Pessoa implements Serializable {
    protected int id;
    protected String nome;

    public void exibir(){
        System.out.println("ID: " + id);
        System.out.println("Nome: " + nome);
    }
}
```

```

    }

    public Pessoa(){
    }

    public Pessoa(int id, String nome) {
this.id = id;
        this.nome = nome;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    @Override    public
String toString() {
        StringBuilder sb = new StringBuilder();
sb.append("ID = ").append(id);
        sb.append(", Nome = ").append(nome);
        return sb.toString();
    }

}

```

### PessoaFisica.java

```

/**
 * @author Felipe komatsu
 */
package model.entidades;

import model.entidades.Pessoa;
import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {
protected String cpf;
protected int idade;

```

```

        @Override
        public void exibir(){
            super.exibir();
            System.out.println("CPF: " + cpf);
            System.out.println("Idade: " + idade);
        }

        public PessoaFisica(){

        }

        public PessoaFisica(int id, String nome, String cpf, int idade) {
            super(id, nome);    this.cpf = cpf;
            this.idade = idade;
        }
        public String getCpf() {
            return cpf;
        }

        public void setCpf(String cpf) {
            this.cpf = cpf;
        }

        public int getIdade() {
            return idade;
        }

        public void setIdade(int idade) {
            this.idade = idade;
        }

        @Override    public
        String toString() {
            StringBuilder sb = new StringBuilder();
            sb.append("ID = ").append(id);    sb.append(",
            Nome = ").append(nome);    sb.append(", CPF
            = ").append(cpf);    sb.append(", Idade =
            ").append(idade);
            return sb.toString();
        }

    }

```

## PessoaJuridica.java

```
/**
 * @author Felipe komatsu
 */
package model.entidades;

import model.entidades.Pessoa;
import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {
    protected String cnpj;

    @Override
    public void exibir(){
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
    }

    public PessoaJuridica(){

    }

    public PessoaJuridica(int id, String nome, String cnpj) {
        super(id, nome);
        this.cnpj = cnpj;
    }

    public String getCnpj() {
        return cnpj;
    }

    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("ID = ").append(id);    sb.append(",
Nome = ").append(nome);    sb.append(",
CNPJ = ").append(cnpj);
        return sb.toString();
    }
}
```

## PessoaFisicaRepo.java

```
/**
 * @author Felipe komatsu
 */
package model.gerenciadores;

import model.entidades.PessoaFisica;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException; import
java.io.ObjectInputStream; import
java.io.ObjectOutputStream;
import java.util.ArrayList;

public class PessoaFisicaRepo {

    private ArrayList<PessoaFisica> pessoasFisicas = new ArrayList<>();    public void inserir(PessoaFisica
    pessoa) {        pessoasFisicas.add(pessoa);
    }

    public ArrayList<PessoaFisica> obterTodos() {
        return pessoasFisicas;
    }

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
            out.writeObject(pessoasFisicas);
            System.out.println("Dados de Pessoa Fisica Armazenados.");
        }
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            pessoasFisicas = (ArrayList<PessoaFisica>) in.readObject();            System.out.println("Dados de
            Pessoa Fisica Recuperados:");
            for (PessoaFisica pessoa : pessoasFisicas) {
                pessoa.exibir();
            }
        }
    }

    public void alterar(PessoaFisica pessoa) {
        int index = pessoasFisicas.indexOf(pessoa);
        if (index != -1) {
            pessoasFisicas.set(index, pessoa);
            System.out.println("Pessoa Fisica alterada com sucesso!");
        }
    }
}
```

```

    } else {
        System.out.println("Pessoa Fisica não encontrada!");
    }
}

```

```

    public void excluir(int id) {
        PessoaFisica pessoa = obter(id);
        if (pessoa != null) {
            pessoasFisicas.remove(pessoa);
        }
    }
}

```

```

    public PessoaFisica obter(int id) {
        for (PessoaFisica pessoa : pessoasFisicas) {
            if (pessoa.getId() == id) {
                return pessoa;
            }
        }
        return null;
    }
}

```

```

}

```

### PessoaJuridicaRepo.java

```

/**
 * @author Felipe komatsu
 */
package model.gerenciadores;

import model.entidades.PessoaJuridica;
import java.io.FileInputStream; import
java.io.FileOutputStream; import
java.io.IOException; import
java.io.ObjectInputStream; import
java.io.ObjectOutputStream;
import java.util.ArrayList;

public class PessoaJuridicaRepo {

    private ArrayList<PessoaJuridica> pessoasJuridicas = new ArrayList<>();

    public void inserir(PessoaJuridica pessoa) {
        pessoasJuridicas.add(pessoa);
    }

    public ArrayList<PessoaJuridica> obterTodos() {
        return pessoasJuridicas;
    }
}

```

```

    }

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
            out.writeObject(pessoasJuridicas);
            System.out.println("Dados de Pessoa Juridica Armazenados.");
        }
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            pessoasJuridicas = (ArrayList<PessoaJuridica>) in.readObject();
            System.out.println("Dados de Pessoa Juridica Recuperados:");
            for (PessoaJuridica pessoa : pessoasJuridicas) {
                pessoa.exibir();
            }
        }
    }

    public void alterar(PessoaJuridica pessoa) {
        int index = pessoasJuridicas.indexOf(pessoa);
        if (index != -1) {
            pessoasJuridicas.set(index, pessoa);
            System.out.println("Pessoa Juridica alterada com sucesso!");
        } else {
            System.out.println("Pessoa Juridica não encontrada!");
        }
    }

    public void excluir(int id) {
        PessoaJuridica pessoa = obter(id);
        if (pessoa != null) {
            pessoasJuridicas.remove(pessoa);
        }
    }

    public PessoaJuridica obter(int id) {
        for (PessoaJuridica pessoa : pessoasJuridicas) {
            if (pessoa.getId() == id) {
                return pessoa;
            }
        }
        return null;
    }
}

```

### CadastroPOO.java

```
package cadastropoo;
```

```

/**
 * @author Felipe komatsu
 */
import java.io.IOException; import
model.entidades.PessoaFisica; import
model.gerenciadores.PessoaFisicaRepo; import
model.entidades.PessoaJuridica;
import model.gerenciadores.PessoaJuridicaRepo;

public class CadastroPOO {    public
static void main(String[] args) {    try
{
    PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
    PessoaFisica pf1 = new PessoaFisica(1, "Ana", "1111111111", 25);
    PessoaFisica pf2 = new PessoaFisica(2, "Carlos", "2222222222", 52);
    repo1.inserir(pf1);    repo1.inserir(pf2);

    repo1.persistir("pessoasFisicas.dat");

    PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
    repo2.recuperar("pessoasFisicas.dat");

    PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
    PessoaJuridica pj1 = new PessoaJuridica(3, "XPTO Sales", "333333333333");
    PessoaJuridica pj2 = new PessoaJuridica(4, "XPTO Solutions", "444444444444");
    repo3.inserir(pj1);    repo3.inserir(pj2);

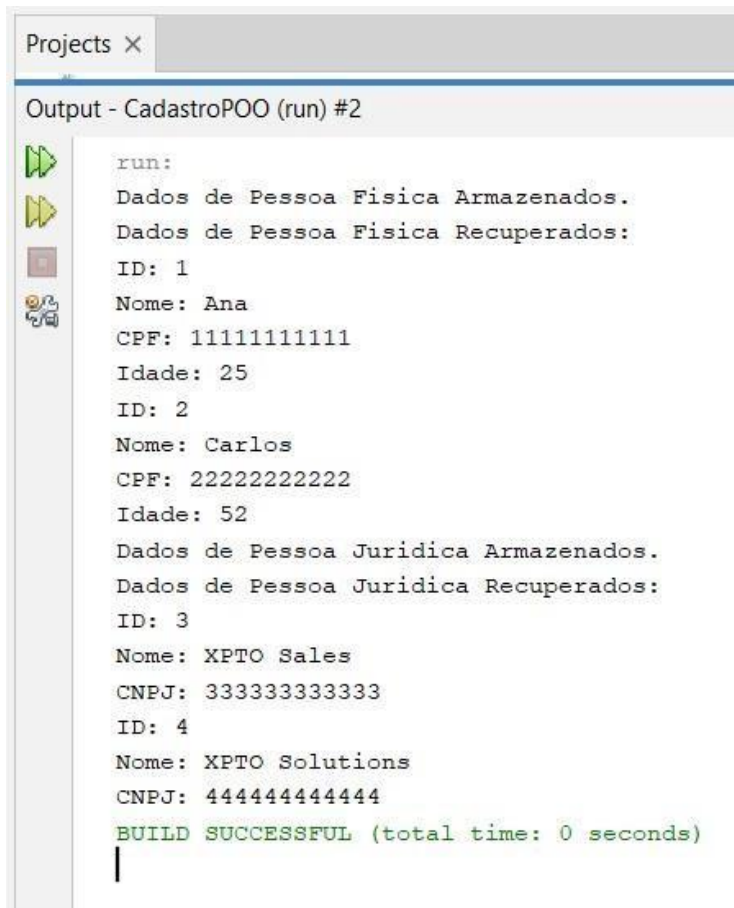
    repo3.persistir("pessoasJuridicas.dat");

    PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
    repo4.recuperar("pessoasJuridicas.dat");
    } catch (IOException e) {
        System.err.println("Erro ao acessar o arquivo: " + e.getMessage());
    } catch (ClassNotFoundException e) {
        System.err.println("Erro ao carregar os dados: " + e.getMessage());
    }
}
}

```

## Resultado da Execução:





```
run:
Dados de Pessoa Fisica Armazenados.
Dados de Pessoa Fisica Recuperados:
ID: 1
Nome: Ana
CPF: 11111111111
Idade: 25
ID: 2
Nome: Carlos
CPF: 22222222222
Idade: 52
Dados de Pessoa Juridica Armazenados.
Dados de Pessoa Juridica Recuperados:
ID: 3
Nome: XPTO Sales
CNPJ: 333333333333
ID: 4
Nome: XPTO Solutions
CNPJ: 444444444444
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Análise e Conclusão:

- **Quais as vantagens e desvantagens do uso de herança?**

### Vantagens:

Reutilização de código e facilidade de manutenção e extensão do sistema.

### Desvantagens:

Pode levar a uma hierarquia complexa e difícil de gerenciar e o aumento de dependência entre classes pode aumentar.

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

A interface `Serializable` é necessária para permitir que objetos Java sejam convertidos em uma sequência de bytes, o que possibilita a gravação e leitura dos objetos de forma mais eficiente.

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

A API Stream do Java utiliza conceitos funcionais como operações de alta ordem, imutabilidade e expressões lambda para realizar operações em coleções de maneira mais concisa e expressiva.

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

O padrão de desenvolvimento adotado na persistência de dados em arquivos em Java é a serialização de objetos. As classes comumente usadas para serialização de objetos em Java são `ObjectOutputStream` e `ObjectInputStream`.