

Aluno: Altair Silva Filho

Matrícula: 202403254409

Unidade: Palhoça – SC

Curso: Desenvolvimento Full Stack

Disciplina: Iniciando o caminho pelo Java

Período: 2025.1

Relatório da Missão Prática – Nível 1 – Mundo 3

Iniciando o caminho pelo Java

1º Procedimento | Criação das Entidades e Sistema de Persistência

Objetivo

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

Códigos do Projeto

Pasta Model

Pessoa.java

```
package model;
import java.io.Serializable;
/**
 *
 * @author Altair
 */
public class Pessoa implements Serializable {
    private int id;
    private String nome;

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

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

```

PessoaFisica.java

```

package model;
/**
 * @author Altair
 */
public class PessoaFisica extends Pessoa {
    private String CPF;
    private int idade;

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

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

PessoaJuridica.java

```
package model;
/**
 * @author Altair
 */
public class PessoaJuridica extends Pessoa {
    private String CNPJ;

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

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

    public String getCnpj() {
        return CNPJ;
    }

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

PessoaFisicaRepo.java

```
package model;

import java.io.*;
import java.util.ArrayList;
import java.util.List;

/**
 * @author Altair
 */
public class PessoaFisicaRepo {
    private ArrayList<PessoaFisica> lista = new ArrayList<>();

    public void inserir(PessoaFisica pessoa) {
        lista.add(pessoa);
    }
}
```

```

public void alterar(PessoaFisica pessoa) {
    lista.stream()
        .filter(p -> p.getID() == pessoa.getID())
        .findFirst()
        .ifPresent(original -> {
            lista.set(lista.indexOf(original), pessoa);
        });
}

public void excluir(int id) {
    lista.removeIf(p -> p.getID() == id);
}

public PessoaFisica obter(int id) {
    return lista.stream()
        .filter(p -> p.getID() == id)
        .findFirst()
        .orElse(null);
}

public List<PessoaFisica> obterTodos() {
    return lista.stream().toList();
}

public void persistir (String nomeArquivo) throws IOException {
    try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
        out.writeObject(lista);
    }
}

public void recuperar(String nomeArquivo) throws IOException,
    ClassNotFoundException {
    try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
        lista = (ArrayList<PessoaFisica>) in.readObject();
    }
}
}

```

PessoaJuridicaRepo.java

```
package model;
```

```
import java.util.ArrayList;
import java.io.*;
import java.util.List;
```

```
/**
```

```
* @author Altair
```

```
*/
```

```
public class PessoaJuridicaRepo {
    private ArrayList<PessoaJuridica> lista = new ArrayList<>();
```

```

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

public void alterar(PessoaJuridica pessoa) {
    lista.stream()
        .filter(p -> p.getID() == pessoa.getID())
        .findFirst()
        .ifPresent(original -> {
            lista.set(lista.indexOf(original), pessoa);
        });
}

public void excluir(int id) {
    lista.removeIf(p -> p.getID() == id);
}

public PessoaJuridica obter(int id) {
    return lista.stream()
        .filter(p -> p.getID() == id)
        .findFirst()
        .orElse(null);
}

public List<PessoaJuridica> obterTodos() {
    return lista.stream().toList();
}

public void persistir (String nomeArquivo) throws IOException {
    try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
        out.writeObject(lista);
    }
}

public void recuperar(String nomeArquivo) throws IOException,
    ClassNotFoundException {
    try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
        lista = (ArrayList<PessoaJuridica>) in.readObject();
    }
}
}

```

Main Class

CadastroPOO.java

```

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

```

```

/**
 * @author Altair
 */
public class CadastroPOO {
    public static void main(String[] args) throws IOException, ClassNotFoundException {
        // Pessoa Física
        //repo1
        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

        PessoaFisica PF1 = new PessoaFisica(1, "Ana", "111.111.111-11", 25);
        PessoaFisica PF2 = new PessoaFisica(2, "Carlos", "222.222.222-22", 52);
        repo1.inserir(PF1);
        repo1.inserir(PF2);

        repo1.persistir("D_PessoasFisicas.txt");
        System.out.println("Dados de Pessoa Física Armazenados");

        //repo2
        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
        repo2.recuperar("D_PessoasFisicas.txt");
        System.out.println("Dados de Pessoa Física Recuperados");
        for (PessoaFisica PF: repo2.obterTodos()) {
            PF.exibir();
            System.out.println("*****");
        }

        // Pessoa Jurídica
        //repo3
        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();

        PessoaJuridica PJ1 = new PessoaJuridica(3, "XPTO Sales", "33.333.333/333-33");
        PessoaJuridica PJ2 = new PessoaJuridica(4, "XPTO Solutions", "44.444.444/444-44");
        repo3.inserir(PJ1);
        repo3.inserir(PJ2);

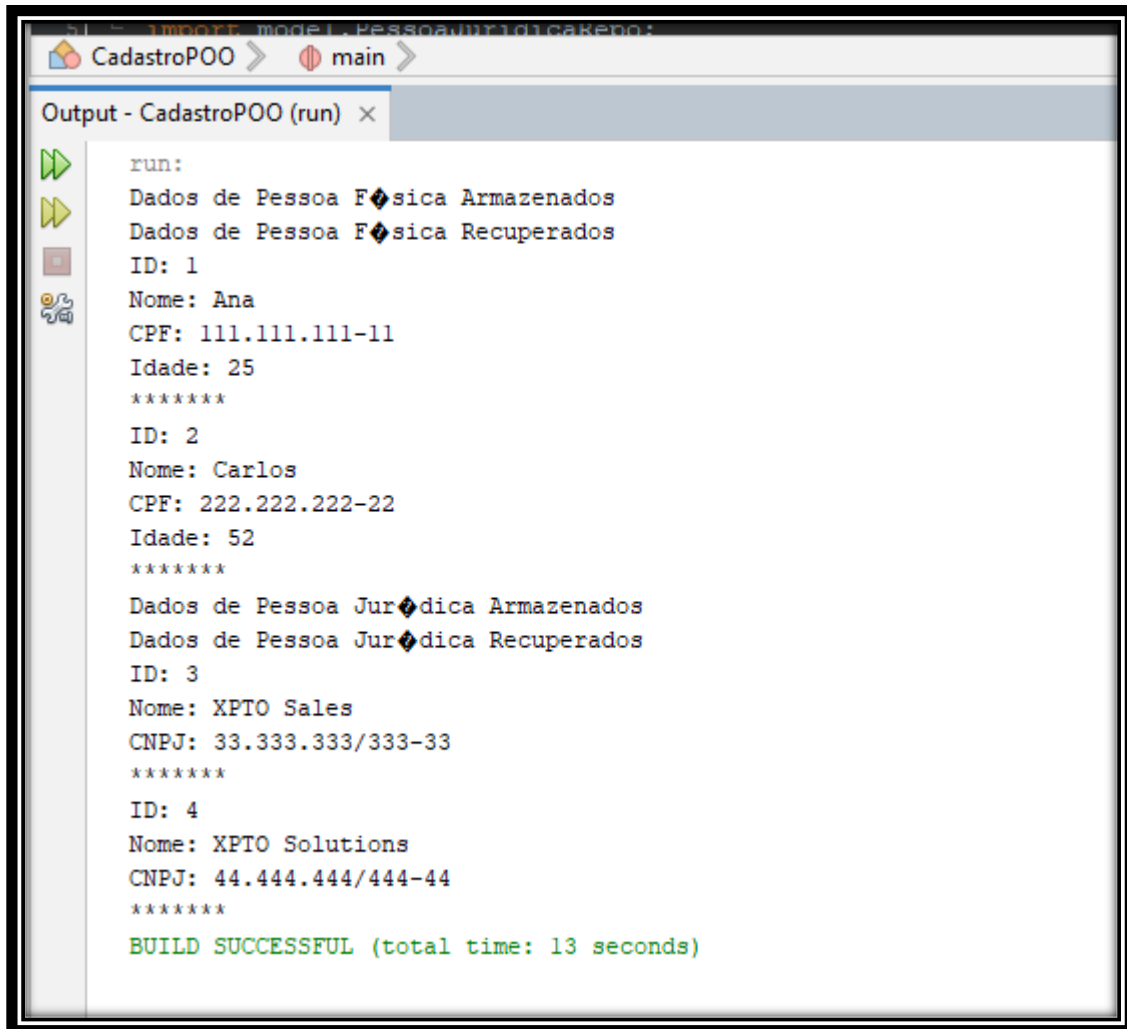
        repo3.persistir("D_PessoasJuridicas.txt");
        System.out.println("Dados de Pessoa Jurídica Armazenados");

        //repo4
        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
        repo4.recuperar("D_PessoasJuridicas.txt");
        System.out.println("Dados de Pessoa Jurídica Recuperados");
        for (PessoaJuridica pj: repo4.obterTodos()) {
            pj.exibir();
            System.out.println("*****");
        }
    }
}

```

Resultados da execução dos códigos

A janela de resultados dos testes é mostrada na figura abaixo.



```
run:
Dados de Pessoa Física Armazenados
Dados de Pessoa Física Recuperados
ID: 1
Nome: Ana
CPF: 111.111.111-11
Idade: 25
*****
ID: 2
Nome: Carlos
CPF: 222.222.222-22
Idade: 52
*****
Dados de Pessoa Jur dica Armazenados
Dados de Pessoa Jur dica Recuperados
ID: 3
Nome: XPTO Sales
CNPJ: 33.333.333/333-33
*****
ID: 4
Nome: XPTO Solutions
CNPJ: 44.444.444/444-44
*****
BUILD SUCCESSFUL (total time: 13 seconds)
```

Resultado da execu  o dos c digos

An lise e Conclus o

A miss o mostrou o uso da t cnica de heran a em Java, ou seja, a capacidade que uma classe tem de transmitir comportamento e caracter sticas  s classes filhas. No projeto, as classes “PessoaF sica” e “PessoaJur dica” herdaram da classe “Pessoa” os atributos id e nome. Como esses atributos s o comuns  s duas classes herdeiras (pessoas f sicas e jur dicas precisam de um identificador e um nome), elas n o precisam, cada uma, criar estes atributos, mas herdaram da classe pai “Pessoa”.

Isto exemplifica algumas vantagens da heran a: a economia de tempo; a reutiliza  o de c digo uma vez que as similaridades s o compartilhadas; e facilita  o de manuten  o no sistema, pois a t cnica traz maior legibilidade do c digo.

Apesar disto, a heran a tem suas desvantagens. Entre elas: pode violar o princ pio do encapsulamento uma vez que os atributos podem fazer parte de v rias classes e forte acoplamento entre elas; e mudan as na classe pai afetam as classes herdeiras.

Outra caracter stica do projeto foi o uso da t cnica chamada *Persist ncia de Dados*, que   o meio usado para salvar e recuperar dados de um sistema de armazenamento, em nosso caso, arquivos gravados no disco do computador. Em Java, o padr o de desenvolvimento adotado em

persistência de dados em arquivos é o *Java Persistence API*, que simplificou o modelo de programação de persistência.

Neste processo de persistência, algo essencial é a classe *Serializable*, que permitiu a transformação dos dados das pessoas físicas e jurídicas em uma sequência de bytes, que uma vez salvos em um arquivo, podem ser recuperados no objeto Java.

E finalmente, temos o uso do paradigma funcional, que é o uso de funções para criar programas, oferecendo ao desenvolvedor a possibilidade de trabalhar com conjuntos de elementos de forma mais simples e com menos linhas de código principalmente quando usamos abstrações. Neste aspecto, entre diversas funcionalidades está a *API Stream* cuja proposta é reduzir a preocupação do desenvolvedor com a forma implementar controle de fluxos ao lidar com coleções. Isto foi usado nos códigos das classes gerenciadores.

Armazenado no Github: <https://github.com/altairsf/CadastroPOO.git>