Trabalho Prático | DGT2821
Programação Back- end com Java

Curso: Desenvolvimento Full Stack

Polo: Polo - Vila Nova - Itu - Sp

Matrícula: 202404575268

Aluno: Rafael Lima de Medeiros

Turma: 2025.4 (5º Semestre)

Data: 10/02/2026

Github: https://github.com/Rafaldm/Programa-o-Back-end-com-Java/tree/main

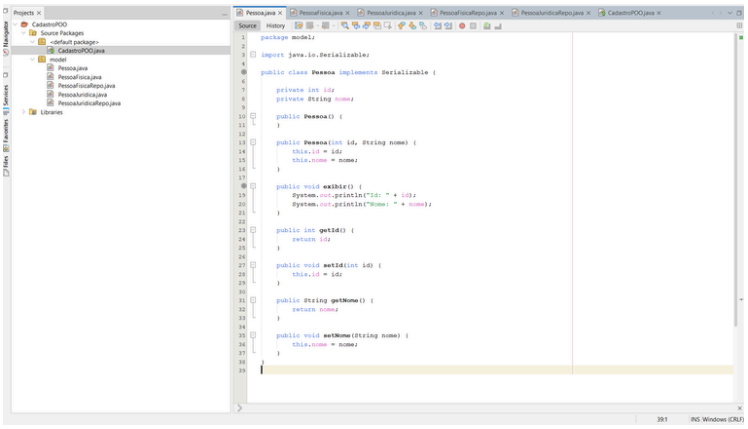
Objetivo: Demonstrar o uso de Programação Orientada a Objetos em Java, utilizando herança, polimorfismo e persistência de dados em arquivos binários.

1º Procedimento

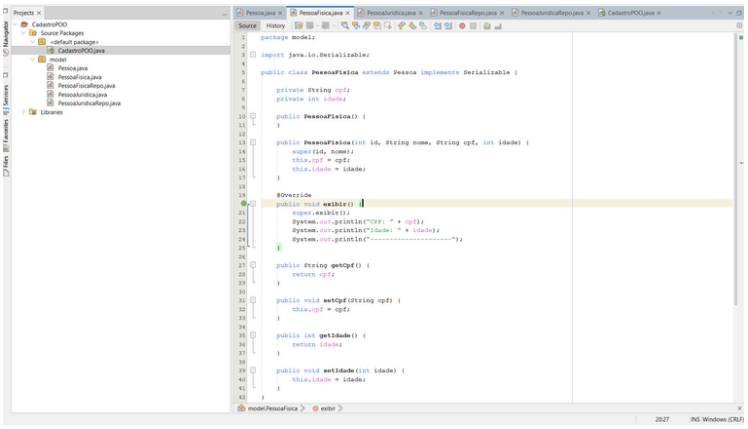
Para melhor visualização das imagens utilize Zoom 250% ou mais.

PRINTS DA TELA

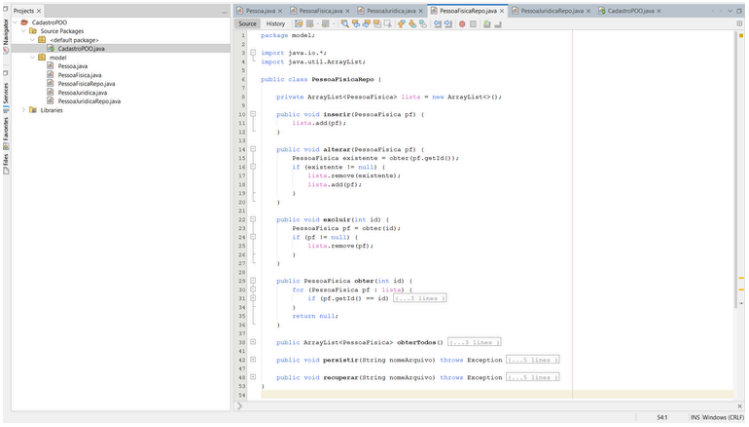
Pessoa.java



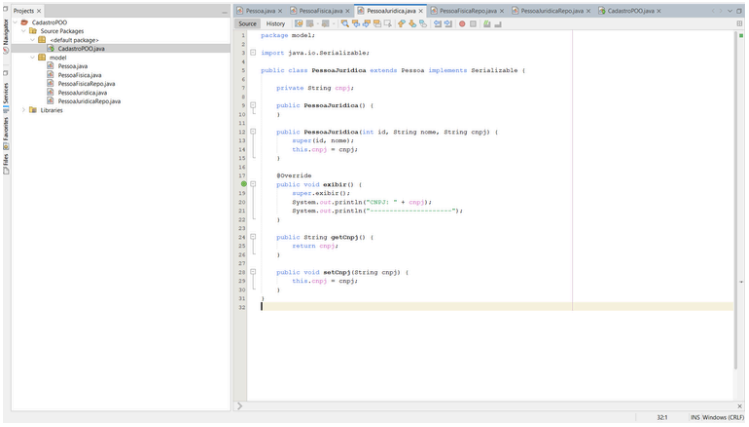
PessoaFisica.java



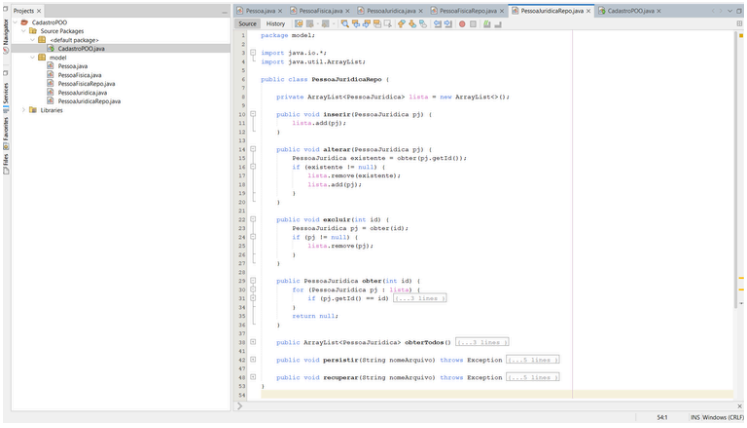
PessoaFisicaRepo.java



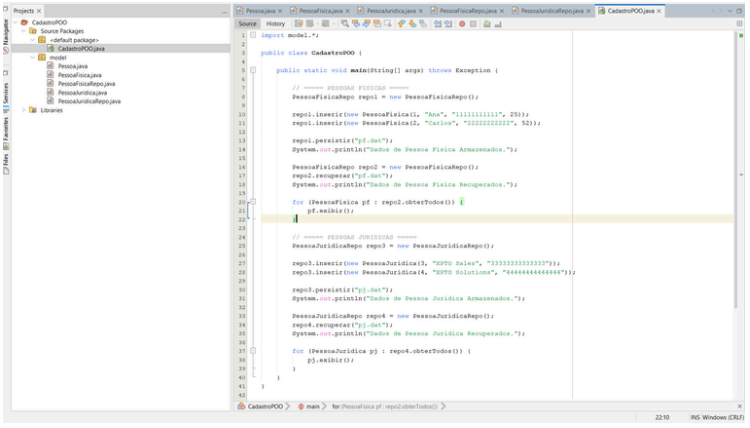
PessoaJuridica.java



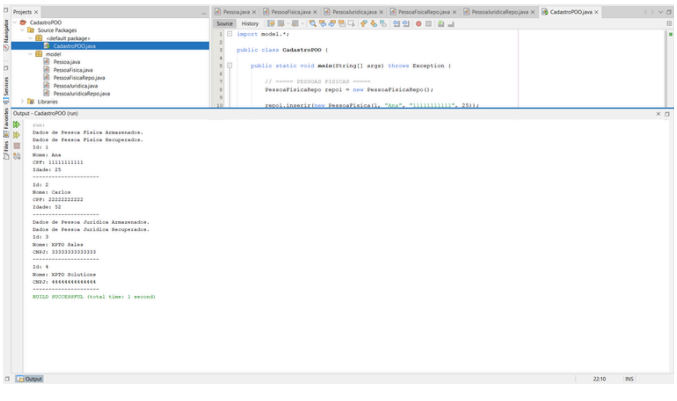
PessoaJuridicaRepo.java



CadastroPOO.java



RESULTADO



Ánalise e Conclusão

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

A herança traz várias vantagens, principalmente a reutilização de código. Com ela, é possível criar uma classe mais genérica e reaproveitar seus atributos e métodos em outras classes, evitando repetição. Além disso, a herança ajuda a organizar melhor o sistema e facilita o uso do polimorfismo, permitindo que métodos tenham comportamentos diferentes dependendo da classe.
Por outro lado, a herança também possui desvantagens. Uma delas é o alto acoplamento entre as classes, já que mudanças na classe pai podem afetar todas as classes filhas. Em projetos maiores, isso pode dificultar a manutenção e deixar o código mais complexo.

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

A interface Serializable é necessária porque ela permite que os objetos sejam transformados em uma sequência de bytes. Essa transformação é essencial para que os dados possam ser gravados em arquivos binários no disco.
Sem a implementação da interface Serializable, o Java não permite salvar ou recuperar objetos diretamente de arquivos, pois não sabe como converter esses objetos para um formato que possa ser armazenado.

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

O paradigma funcional é utilizado na API Stream principalmente por meio do uso de expressões lambda. Com a Stream API, é possível trabalhar com coleções de dados de forma mais simples e direta, utilizando operações como filter, map e forEach.
Esse tipo de abordagem torna o código mais limpo, fácil de ler e reduz a necessidade de laços repetitivos, como o uso excessivo de for ou while.

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

Na persistência de dados em arquivos, geralmente é adotado o padrão DAO (Data Access Object). Esse padrão separa a lógica de acesso aos dados da lógica principal do sistema.
No projeto desenvolvido, as classes de repositório são responsáveis por inserir, alterar, excluir, salvar e recuperar os dados (Crud), deixando as classes de entidade focadas apenas na representação das informações.

Códigos

- **Pessoa.java**

```
package model;

import java.io.Serializable;

public class Pessoa implements Serializable {

    private int id;
    private String nome;

    public Pessoa() {
    }

    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;

import java.io.Serializable;

public class PessoaFisica extends Pessoa implements
Serializable {

    private String cpf;
    private int idade;

    public PessoaFisica() {
    }

    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);
        System.out.println("----------------------");
    }

    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;
    }
}
```

- **PessoaFisicaRepo.java**

```
package model;

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

public class PessoaFisicaRepo {

   private ArrayList<PessoaFisica> lista = new ArrayList<>();

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

   public void alterar(PessoaFisica pf) {
      PessoaFisica existente = obter(pf.getId());
      if (existente != null) {
         lista.remove(existente);
         lista.add(pf);
      }
   }

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

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

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

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

   public void recuperar(String nomeArquivo) throws Exception
{
      ObjectInputStream in = new ObjectInputStream(new
FileInputStream(nomeArquivo));
      lista = (ArrayList<PessoaFisica>) in.readObject();
      in.close();
   }
}
```

- **PessoaJuridica.java**

```
package model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements
Serializable {

   private String cnpj;

   public PessoaJuridica() {
   }

   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);
      System.out.println("----------------------");
   }

   public String getCnpj() {
      return cnpj;
   }

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

- **PessoaJuridicaRepo.java**

```
package model;

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

public class PessoaJuridicaRepo {

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

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

   public void alterar(PessoaJuridica pj) {
      PessoaJuridica existente = obter(pj.getId());
      if (existente != null) {
         lista.remove(existente);
         lista.add(pj);
      }
   }

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

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

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

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

   public void recuperar(String nomeArquivo) throws Exception
{
      ObjectInputStream in = new ObjectInputStream(new
FileInputStream(nomeArquivo));
      lista = (ArrayList<PessoaJuridica>) in.readObject();
      in.close();
   }
}
```

- **CadastroPOO.java**

```
import model.*;

public class CadastroPOO {

   public static void main(String[] args) throws Exception
{

      // ===== PESSOAS FISICAS =====
      PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

      repo1.inserir(new PessoaFisica(1, "Ana",
"11111111111", 25));
      repo1.inserir(new PessoaFisica(2, "Carlos",
"22222222222", 52));

      repo1.persistir("pf.dat");
      System.out.println("Dados de Pessoa Fisica
Armazenados.");

      PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
      repo2.recuperar("pf.dat");
      System.out.println("Dados de Pessoa Fisica
Recuperados.");

      for (PessoaFisica pf : repo2.obterTodos()) {
         pf.exibir();
      }

      // ===== PESSOAS JURIDICAS =====
      PessoaJuridicaRepo repo3 = new
PessoaJuridicaRepo();

      repo3.inserir(new PessoaJuridica(3, "XPTO Sales",
"33333333333333"));
      repo3.inserir(new PessoaJuridica(4, "XPTO
Solutions", "44444444444444"));

      repo3.persistir("pj.dat");
      System.out.println("Dados de Pessoa Juridica
Armazenados.");

      PessoaJuridicaRepo repo4 = new
PessoaJuridicaRepo();
      repo4.recuperar("pj.dat");
      System.out.println("Dados de Pessoa Juridica
Recuperados.");

      for (PessoaJuridica pj : repo4.obterTodos()) {
         pj.exibir();
      }
   }
}
```

Resultado

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: 33333333333333

Id: 4
Nome: XPTO Solutions
CNPJ: 44444444444444

BUILD SUCCESSFUL (total time: 0 seconds)