UNIVERSIDADE ESTÁCIO DE SÁ - POLO SÃO MATEUS DO SUL

RELATÓRIO DE ACOMPANHAMENTO

CRIAÇÃO DE ENTIDADES E PERSISTÊNCIA COM JAVA

GUSTAVO ALMEIDA RAMOS

Relatório referente à aula prática da matéria

Iniciando o Caminho Pelo Java do

Curso Desenvolvimento Full Stack

2025

Objetivo

O objetivo desta tarefa foi criar um sistema simples de cadastro de pessoas físicas e pessoas jurídicas utilizando de Programação Orientada a Objetos em Java. A tarefa fez que fizéssemos o uso de herança, polimorfismo, interface Serializable para fazer o uso dos dados em arquivos, e manipulação de dados com listas.

Códigos

Pessoa

```
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 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; }

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

PessoaFisica

```
package model;

public class PessoaFisica extends Pessoa {
  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;
  }

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

PessoaJuridica

```
package model;

public class PessoaJuridica extends Pessoa {
  private String 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 void exibir() {
    super.exibir();
    System.out.println("CNPJ: " + cnpj);
  }
}
```

Main

```
import model.PessoaFisica;
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;

public class CadastroPOO {
  public static void main(String[] args) {
    try {
      // Repositório de Pessoas Físicas (repo1)
      PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

      // Adicionando duas pessoas físicas
      PessoaFisica pf1 = new PessoaFisica(1, "Ana", "11111111111", 25);
      PessoaFisica pf2 = new PessoaFisica(2, "Carlos", "22222222222", 52);
      repo1.inserir(pf1);
      repo1.inserir(pf2);


      String arquivoPF = "pessoas_fisicas.dat";
      repo1.persistir(arquivoPF);
      System.out.println("Dados de Pessoa Fisica Armazenados.");

      // arrumar que está dando erro
      PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
```

```
        repo2.recuperar(arquivoPF);
        System.out.println("Dados de Pessoa Fisica Recuperados.");
        for (PessoaFisica pf : repo2.obterTodos()) {
          System.out.println("Id: " + pf.getId());
          System.out.println("Nome: " + pf.getNome());
          System.out.println("CPF: " + pf.getCpf());
          System.out.println("Idade: " + pf.getIdade());
        }


        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();


        PessoaJuridica pj1 = new PessoaJuridica(3, "XPTO Sales", "33333333333333");
        PessoaJuridica pj2 = new PessoaJuridica(4, "XPTO Solutions", "44444444444444");
        repo3.inserir(pj1);
        repo3.inserir(pj2);


        String arquivoPJ = "pessoas_juridicas.dat";
        repo3.persistir(arquivoPJ);
        System.out.println("Dados de Pessoa Juridica Armazenados.");
        // arrumar dando erro no repo
        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
        repo4.recuperar(arquivoPJ);
        System.out.println("Dados de Pessoa Juridica Recuperados.");
        for (PessoaJuridica pj : repo4.obterTodos()) {
          System.out.println("Id: " + pj.getId());
          System.out.println("Nome: " + pj.getNome());
          System.out.println("CNPJ: " + pj.getCnpj());
        }
        //arrumar class errada
      } catch (Exception e) {
        System.out.println("Erro: " + e.getMessage());
        e.printStackTrace();
      }
    }
}
```

Resultados

Em seguida ficou assim o console de saída gerada:

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

Análise e Conclusão

Quais as vantagens e desvantagens do uso de herança?

Alguns prós da atividade e no código bens foram:

- O link de código entre classes relacionadas onde consegui fazer o aport entre as classes.
- Facilidade de identificar o erro e executar a correção e extensão do sistema
- Mais um pró foi o uso do polimorfismo para o comportamentos do código.

Agora algumas desvantagens da atividade e no código bens foram:

- Entender o funcionamento me bati muito no começo e tive que ver alguns vídeos falando sobre e tive que até pesquisar uma explicação mais detalhada no chat gpt.
- Alguns problemas nas heranças causadas pelo forte aport entre as classes.
- Complexa para iniciante várias vezes deu o erro de java.io.NotSerializableException e no começo eu não entendia o porquê, foi aí que eu percebi que deveria fazer o link de todos.

Por que a interface Serializable é necessária?

A interface Serializable é obrigatória nesse caso porque a gente quer que seja verificado o objeto em arquivos binário constantemente. A interface informa à "JVM" que o objeto pode ser transformado em uma sequência de byte e recuperada depois, mantendo seu estado natural sem modificação onde é de extremamente importante.

Sem ela, ocorre o erro que citei no começo do "java.io.NotSerializableException".

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

A API Stream em Java faz que a programação funcional funcione com métodos como map, filter, reduce e forEach. Esses "parâmetros" operam sobre arquivos de forma uniforme e sem a necessidade de loops extensivos. Por exemplo o uso inclui a filtragem de dados gerais, transformações de dados e a soma.

Padrão de desenvolvimento na persistência de dados em arquivos no Java

O padrão que foi colocado é conhecido como DAO “Data Access Object”, na nossa atividade os repositórios de “Pessoa Física Repo” e “Pessoa Jurídica Repo” são feitas como DAOs, prendendo o acesso a dados (como a inserção, alteração, exclusão, serializá-lo e recuperação) de forma separada e eficiente.

Códigos

Main

```
import model.*;
import java.util.*;
import java.io.*;

public class CadastroPOO {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
    PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();

    int opcao;
    do {
      System.out.println("\n--- Menu ---");
      System.out.println("1 - Incluir");
      System.out.println("2 - Alterar");
      System.out.println("3 - Excluir");
      System.out.println("4 - Exibir pelo ID");
      System.out.println("5 - Exibir todos");
      System.out.println("6 - Salvar dados");
      System.out.println("7 - Recuperar dados");
      System.out.println("0 - Sair");
      System.out.print("Escolha uma opcao: ");
      opcao = Integer.parseInt(sc.nextLine());

      switch (opcao) {
        case 1 -> { // Incluir
          System.out.print("Tipo (F - Fisica, J - Juridica): ");
          String tipo = sc.nextLine();
          if (tipo.equalsIgnoreCase("F")) {
            System.out.print("ID: "); int id = Integer.parseInt(sc.nextLine());
            System.out.print("Nome: "); String nome = sc.nextLine();
            System.out.print("CPF: "); String cpf = sc.nextLine();
            System.out.print("Idade: "); int idade = Integer.parseInt(sc.nextLine());
            repoFisica.inserir(new PessoaFisica(id, nome, cpf, idade));
          } else {
            System.out.print("ID: "); int id = Integer.parseInt(sc.nextLine());
            System.out.print("Nome: "); String nome = sc.nextLine();
            System.out.print("CNPJ: "); String cnpj = sc.nextLine();
            repoJuridica.inserir(new PessoaJuridica(id, nome, cnpj));
          }
        }
        case 2 -> { // Alterar
          System.out.print("Tipo (F - Fisica, J - Juridica): ");
          String tipo = sc.nextLine();
          System.out.print("ID: "); int id = Integer.parseInt(sc.nextLine());
          if (tipo.equalsIgnoreCase("F")) {
            PessoaFisica pf = repoFisica.obter(id);
            if (pf != null) {
              pf.exibir();
              System.out.print("Novo nome: "); pf.setNome(sc.nextLine());
```

```
            System.out.print("Novo CPF: "); pf.setCpf(sc.nextLine());
            System.out.print("Nova idade: "); pf.setIdade(Integer.parseInt(sc.nextLine()));
            repoFisica.alterar(pf);
        }
    } else {
        PessoaJuridica pj = repoJuridica.obter(id);
        if (pj != null) {
            pj.exibir();
            System.out.print("Novo nome: "); pj.setNome(sc.nextLine());
            System.out.print("Novo CNPJ: "); pj.setCnpj(sc.nextLine());
            repoJuridica.alterar(pj);
        }
    }
}
case 3 -> { // Excluir
    System.out.print("Tipo (F - Fisica, J - Juridica): ");
    String tipo = sc.nextLine();
    System.out.print("ID: "); int id = Integer.parseInt(sc.nextLine());
    if (tipo.equalsIgnoreCase("F")) repoFisica.excluir(id);
    else repoJuridica.excluir(id);
}
case 4 -> { // Exibir pelo ID
    System.out.print("Tipo (F - Fisica, J - Juridica): ");
    String tipo = sc.nextLine();
    System.out.print("ID: "); int id = Integer.parseInt(sc.nextLine());
    if (tipo.equalsIgnoreCase("F")) {
        PessoaFisica pf = repoFisica.obter(id);
        if (pf != null) pf.exibir();
    } else {
        PessoaJuridica pj = repoJuridica.obter(id);
        if (pj != null) pj.exibir();
    }
}
case 5 -> { // Exibir todos
    System.out.print("Tipo (F - Fisica, J - Juridica): ");
    String tipo = sc.nextLine();
    if (tipo.equalsIgnoreCase("F")) {
        for (PessoaFisica pf : repoFisica.obterTodos()) pf.exibir();
    } else {
        for (PessoaJuridica pj : repoJuridica.obterTodos()) pj.exibir();
    }
}
case 6 -> { // Salvar
    System.out.print("Prefixo do arquivo: ");
    String prefixo = sc.nextLine();
    try {
        repoFisica.persistir(prefixo + ".fisica.bin");
        repoJuridica.persistir(prefixo + ".juridica.bin");
        System.out.println("Dados salvos.");
    } catch (Exception e) {
        System.out.println("Erro ao salvar: " + e.getMessage());
    }
}
case 7 -> { // Recuperar
    System.out.print("Prefixo do arquivo: ");
    String prefixo = sc.nextLine();
    try {
        repoFisica.recuperar(prefixo + ".fisica.bin");
        repoJuridica.recuperar(prefixo + ".juridica.bin");
        System.out.println("Dados recuperados.");
    } catch (Exception e) {
```

```
                System.out.println("Erro ao recuperar: " + e.getMessage());
            }
          }
          case 0 -> System.out.println("Finalizando...");
          default -> System.out.println("Opcao invalida!");
        }
      } while (opcao != 0);

      sc.close();
    }
}
```

Análise e Conclusão

O que são elementos estáticos e qual o motivo para o método main adotar esse modificador?

Elementos estáticos são das classes e não a uma condição de classe como um item do código e sim dele como todo, significa que o método “Main” é obrigatório um elemento estático porque ele é o ponto inicial do código, já que ainda não existe nenhum objeto sendo declarado então automaticamente precisa de um jeito para declarar uma classe.

Para que serve a classe Scanner?

A classe Scanner é usado para ler dados da entrada somente.

Como o uso de classes de repositório impactou na organização do código?

As classes de repositório (como “Pessoa Física Repo” e “Pessoa Jurídica Repo” que utilizei no projeto) são feitas para centralizar os dados do projeto de cada classe, sem essas classes os dados ficariam bagunçados.