Nome do campus: 3366 Polo Centro – Garopaba - SC.

Nome do curso: Desenvolvimento Full Stack.

Nome da disciplina: Iniciando caminho pelo java.

Número da turma: 9001.

Nome: Daniel dos Santos Pereira.

Endereço: DaniielDev/Entidades_Sistema

Título da Prática: Criação das Entidades e Sistema de Prática.

Objetivo: Exibir dados de pessoas físicas, armazenados e recuperados pelo sistema.

Análise e conclusão:

Quais as vantagens e desvantagens do uso de herança?

Vantagens:

Reutilização de código: Evita repetição, pois classes filhas reutilizam atributos e métodos da classe pai.

Organização e estrutura: Facilita a modelagem hierárquica, tornando o código mais lógico e legível.

Polimorfismo: Permite usar objetos de subclasses como se fossem da superclasse, facilitando extensibilidade.

Desvantagens:

Alto acoplamento: Subclasses dependem fortemente da classe pai, dificultando mudanças isoladas.

Reuso limitado: Nem sempre uma subclasse herda apenas o que precisa (herança forçada).

Dificuldade em manutenção: Em hierarquias complexas, erros podem surgir de interdependências entre classes.

Por que a interface Serializable é necessária na persistência em arquivos binários?

A interface Serializable **indica que os objetos daquela classe podem ser convertidos para uma sequência de bytes** — processo chamado de *serialização*.

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

Imutabilidade: Os dados originais não são modificados.

Funções puras: Métodos como map, filter e reduce usam funções sem efeitos colaterais.

Expressividade: Permite trabalhar com coleções de forma concisa e declarativa.

Qual padrão de desenvolvimento é adotado na persistência de dados em arquivos no Java?

O padrão mais adotado é o DAO (Data Access Object), que separa a lógica de acesso a dados da lógica de negócio.

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 + ", 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;

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;
    }

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

PessoaFisicaRepo.java

```
package model;
```

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

public class PessoaFisicaRepo {
    private ArrayList<PessoaFisica> pessoas = new ArrayList<>();

    public void inserir(PessoaFisica p) {
        pessoas.add(p);
    }

    public void alterar(PessoaFisica p) {
        for (int i = 0; i < pessoas.size(); i++) {
            if (pessoas.get(i).getId() == p.getId()) {
                pessoas.set(i, p);
                break;
            }
        }
    }

    public void excluir(int id) {
        pessoas.removeIf(p -> p.getId() == id);
    }
```

```
public PessoaFisica obter(int id) {
    for (PessoaFisica p : pessoas) {
        if (p.getId() == id) return p;
    }
    return null;
}

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

public void persistir(String nomeArquivo) throws Exception {
    FileOutputStream fos = new FileOutputStream(nomeArquivo);
    ObjectOutputStream oos = new ObjectOutputStream(fos);
    oos.writeObject(pessoas);
    oos.close();
}

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

```
    }
```

PessoaJuridica.java

```
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;
   }

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

   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> pessoas = new ArrayList<>();

    public void inserir(PessoaJuridica p) {
        pessoas.add(p);
    }

    public void alterar(PessoaJuridica p) {
        for (int i = 0; i < pessoas.size(); i++) {
            if (pessoas.get(i).getId() == p.getId()) {
                pessoas.set(i, p);
                break;
            }
        }
    }

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

    public PessoaJuridica obter(int id) {
        for (PessoaJuridica p : pessoas) {
            if (p.getId() == id) return p;
        }
        return null;
    }
```

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

    public void persistir(String nomeArquivo) throws Exception {
        FileOutputStream fos = new FileOutputStream(nomeArquivo);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(pessoas);
        oos.close();
    }

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

Principal.java

```
import model.*;

public class Principal {
    public static void main(String[] args) {
        try {
            // Pessoas Físicas
            PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
            repo1.inserir(new PessoaFisica(1, "Ana", "123.456.789-00", 28));
            repo1.inserir(new PessoaFisica(2, "Carlos", "987.654.321-00", 35));
```

```
            repo1.persistir("pessoas_fisicas.dat");

            PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
            repo2.recuperar("pessoas_fisicas.dat");
            System.out.println("\n--- Pessoas Físicas Recuperadas ---");
            for (PessoaFisica pf : repo2.obterTodos()) {
                pf.exibir();
                System.out.println();
            }

            // Pessoas Jurídicas
            PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
            repo3.inserir(new PessoaJuridica(1, "Empresa Sales 1", "12.345.678/0001-00"));
            repo3.inserir(new PessoaJuridica(2, "Empresa Sales 2", "98.765.432/0001-00"));
            repo3.persistir("pessoas_juridicas.dat");

            PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
            repo4.recuperar("pessoas_juridicas.dat");
            System.out.println("\n--- Pessoas Jurídicas Recuperadas ---");
            for (PessoaJuridica pj : repo4.obterTodos()) {
                pj.exibir();
                System.out.println();
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```