POLO OLARIA - RIO DE JANEIRO - RJ

DESENVOLVIMENTO FULL STACK

Nivel 1: Iniciando o Caminho Pelo Java | Turma 9001

Erik Bastos de Moraes

Missão Prática | Nível 1 | Mundo 3

1º Procedimento | Criação das Entidades e Sistema de Persistência

Objetivos da prática

1- Utilizar herança e polimorfismo na definição de entidades.

2- Utilizar persistência de objetos em arquivos binários.r

3- Implementar uma interface cadastral em modo texto.

4- Utilizar o controle de exceções da plataforma Java.

5- 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.

Pessoa.java

```
package model;


import java.io.Serializable;


public class Pessoa implements Serializable {
    protected int id;
    protected 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; }

}
```

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; }
}
```

PessoaFisicaRepo.java

```
package model;


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


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);
                return;
            }
        }
    }


    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 List<PessoaFisica> obterTodos() {
        return pessoas;
    }

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

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

PessoaJuridicaRepo.java

```
package model;


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


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);
                return;
            }
        }
    }


    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 List<PessoaJuridica> obterTodos() {
        return pessoas;
    }

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

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

main.java

```
package cadastropoo;


import model.*;


public class main {
    public static void main(String[] args) {
        try {
            PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
            repo1.inserir(new PessoaFisica(1, "Roland", "123.456.789-00", 30));
            repo1.inserir(new PessoaFisica(2, "Angelica", "987.654.321-00", 25));
            repo1.persistir("pessoasFisicas.dat");

            PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
            repo2.recuperar("pessoasFisicas.dat");
            for (PessoaFisica pf : repo2.obterTodos()) {
                pf.exibir();
                System.out.println("---");
            }

            PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
            repo3.inserir(new PessoaJuridica(1, "Limbus Company", "12.345.678/0001-99"));
            repo3.inserir(new PessoaJuridica(2, "W Corp", "98.765.432/0001-11"));
            repo3.persistir("pessoasJuridicas.dat");
```

```
            PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
            repo4.recuperar("pessoasJuridicas.dat");
            for (PessoaJuridica pj : repo4.obterTodos()) {
                pj.exibir();
                System.out.println("---");
            }

        } catch (Exception e) {
            System.out.println("Erro: " + e.getMessage());
        }
    }
}
```

Resultado

```
run:
ID: 1, Nome: Roland
CPF: 123.456.789-00, Idade: 30
---
ID: 2, Nome: Angelica
CPF: 987.654.321-00, Idade: 25
---
ID: 1, Nome: Limbus Company
CNPJ: 12.345.678/0001-99
---
ID: 2, Nome: W Corp
CNPJ: 98.765.432/0001-11
---
BUILD SUCCESSFUL (total time: 0 seconds)
```

Análise e Conclusão

A. Quais as vantagens e desvantagens do uso de herança?

R: Como vantagem, usar heranças evita a repetição do código, facilita o uso de métodos genéricos que funcionam com diferentes subclasses, e melhora a organização do código. Como desvantagem, mudanças na classe pai acabam por afetar todas as subclasses, dificultando a manutenção do código.

B. Por que a interface Serializable é necessária ao efetuar persistência em arquivos binários?

R: Sem essa inferface, o Java não saberia como quebrar o objeto em que possam ser gravados.

C. Como o paradigma funcional é utilizado pela API stream no Java?

R: A API Stream do Java traz conceitos de programação funcional para trabalhar com coleções de forma mais declarativa e concisa, isso melhora a clareza, legibilidade e reduz bugs ao evitar manipulação direta das coleções.

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

R: O Java Persistence API (JPA).

---

https://github.com/ErikBM2661/Cadastro.git