UNIVERSIDADE ESTÁCIO DE SÁ

DESENVOLVIMENTO FULLSTACK

RPG0014

INICIANDO O CAMINHO PELO JAVA

202205003922

JARDS DE OLIVEIRA GUIMARÃES

MACEIÓ - AL

2023

SUMÁRIO

1. INTRODUÇÃO

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

1.1 OBJETIVO GERAL

- ✓ Utilizar herança e polimorfismo na definição de entidades.
- ✓ Utilizar persistência de objetos em arquivos binários.
- ✓ Implementar uma interface cadastral em modo texto.
- ✓ Utilizar o controle de exceções da plataforma Java.
- ✓ persistência em arquivos binários.

2. CÓDIGOS SOLICITADOS NESTE ROTEIRO DE AULA

2.1 CLASSE 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.printf("   ID: %d%n Nome: %s%n", this.getId(), this.getNome());
    }

}
```

2.2 CLASSE PESSOA FÍSICA

```
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() {
```

```
        System.out.printf("   ID: %d%n Nome: %s%n  CPF: %s%nIdade: %d%n", this.getId(),
                this.getNome(), this.getCpf(), this.getIdade());
    }

}
```

2.3CLASSE PESSOA JURÍDICA

```
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() {
        System.out.printf("   ID: %d%n Nome: %s%n CNPJ: %s%n", this.getId(), this.getNome(),
                this.getCnpj());
    }

}
```

2.4CLASSE PESSOAFISICAREPO

```
package model;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
```

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

public class PessoaFisicaRepo {

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

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

   public void alterar(int indice, PessoaFisica pf) {
      listaDePessoasFisica.set(indice, pf);
   }

   public void excluir(int id) {
      listaDePessoasFisica.remove(id);
   }

   public PessoaFisica obter(int id) {
      return listaDePessoasFisica.get(id);
   }

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

   public void persistir(String nomeArquivo) {
      try (FileOutputStream fout = new FileOutputStream("src\\files\\" + nomeArquivo + ".txt");
            ObjectOutputStream oos = new ObjectOutputStream(fout)) {
         oos.writeObject(this.listaDePessoasFisica);
         System.out.println("Dados de Pessoa Física Armazenados.");
      } catch (IOException e) {
         e.printStackTrace();
      }
   }

   public ArrayList<PessoaFisica> recuperar(String nomeArquivo) {
      try (FileInputStream fin = new FileInputStream("src\\files\\" + nomeArquivo + ".txt");
            ObjectInputStream ois = new ObjectInputStream(fin)) {
         listaDePessoasFisica = (ArrayList<PessoaFisica>) ois.readObject();
         System.out.println("Dados de Pessoas Física Recuperados.");
      } catch (IOException e) {
         e.printStackTrace();
      } catch (ClassNotFoundException e) {
```

```
            e.printStackTrace();
        }

        return listaDePessoasFisica;

    }
}
```

2.5CLASSE PESSOAJURIDICAREPO

```
package model;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;

public class PessoaJuridicaRepo {

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

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

    public void alterar(int indice, PessoaJuridica pj) {
        listaDePessoasJuridica.set(indice, pj);
    }

    public void excluir(int id) {
        listaDePessoasJuridica.remove(id);
    }

    public PessoaJuridica obter(int id) {
        return listaDePessoasJuridica.get(id);
    }

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

    public void persistir(String nomeArquivo) {
        try (FileOutputStream fout = new FileOutputStream("src\\files\\" + nomeArquivo + ".txt");
```

```
            ObjectOutputStream oos = new ObjectOutputStream(fout)) {
          oos.writeObject(this.listaDePessoasJuridica);
          System.out.println("Dados de Pessoa Juridica Armazenados.");
        } catch (IOException e) {
          e.printStackTrace();
        }
    }

    public ArrayList<PessoaJuridica> recuperar(String nomeArquivo) {
        try (FileInputStream fin = new FileInputStream("src\\files\\" + nomeArquivo + ".txt");
             ObjectInputStream ois = new ObjectInputStream(fin)) {
          listaDePessoasJuridica = (ArrayList<PessoaJuridica>) ois.readObject();
          System.out.println("Dados de Pessoas Juridica Recuperados.");
        } catch (IOException e) {
          e.printStackTrace();
        } catch (ClassNotFoundException e) {
          e.printStackTrace();
        }

        return listaDePessoasJuridica;

    }

}
```

2.6 CLASSE APP(MAIN)

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

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

        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

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

        repo1.persistir("PessoaFisica");

        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
        repo2.recuperar("PessoaFisica");
```

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

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

        repo3.persistir("PessoaJuridica");

        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
        repo4.recuperar("PessoaJuridica");

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

3. RESULTADOS DA EXECUÇÃO DOS CÓDIGOS

```
Dados de Pessoa Física Armazenados.
Dados de Pessoas Física Recuperados.
  ID: 1
 Nome: Ana
  CPF: 11111111111
Idade: 25
  ID: 2
 Nome: Carlos
  CPF: 22222222222
Idade: 52
Dados de Pessoa Juridica Armazenados.
Dados de Pessoas Juridica Recuperados.
  ID: 3
 Nome: XPTO Sales
 CNPJ: 33333333333333
  ID: 1
 Nome: Ana
  CPF: 11111111111
```

Idade: 25

ID: 2

Nome: Carlos

CPF: 22222222222

Idade: 52

Dados de Pessoa Juridica Armazenados.

Dados de Pessoas Juridica Recuperados.

ID: 3

Nome: XPTO Sales

CNPJ: 33333333333333

ID: 4

Nome: XPTO Solutions

CNPJ: 44444444444444

4. ANÁLISE E CONCLUSÃO

4.1 Quais as vantagens e desvantagens do uso de herança?

A herança tem a vantagem de capturar o que é comum e o isolar daquilo que é diferente, além de ser vista diretamente no código OO. Porém, ela gera um forte acoplamento, pois uma pequena mudança numa superclasse afeta todas as suas subclasses. Essa característica da herança viola um dos princípios da Orientação a Objetos que é manter o fraco acoplamento. Além disso, a herança é um relacionamento estático. Algumas vezes, os objetos do mundo OO precisam sofrer mutações, e algumas delas não são possíveis quando utilizamos herança (por exemplo, se um funcionário mudou de cargo dentro de uma empresa, seu objeto deveria mudar de classe, o que não ocorre).

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

A serialização em Java é o processo no qual a instância de um objeto é transformada em uma sequência de bytes e é útil quando precisamos enviar objetos pela rede, salvar no disco, ou comunicar de uma JVM com outra. Isso porque o estado atual do objeto é “congelado” e na outra “ponta” nós podemos “descongelar” este objeto sem perder nenhuma informação.

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

De modo geral, a API stream é uma ferramenta que possibilita o gerenciamento de coleções JAVA com maior praticidade e eficiência, levando em consideração os princípios aplicados na programação funcional. Com isso, todo o controle de fluxo e loop será executado pela API, fazendo com que o programador atente apenas para a regra do negócio.

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

O MVC, que funciona como um padrão de arquitetura de software que melhora a conexão entre as camadas de dados, lógica de negócio e interação com usuário. Através da sua divisão em três componentes, o processo de programação se torna algo mais simples e dinâmico.