Desenvolvimento Full Stack

Felipe Marques de Almeida - 202208291929

Polo Rodovia Br 407 - Juazeiro - Ba
RPG 0014 - Iniciando o caminho pelo Java – Turma 22.3 – 3º Semestre
GitHub: Kifflom2108/Missao-Pratica-Nivel-1-Mundo-3 (github.com)

Objetivo da Prática

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.

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.

Codigos:

Main:

```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package com.mycompany.cadastropoo.model;

import java.io.IOException;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Main {

    public static void main(String[] args) {
        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
        PessoaFisica pessoa1 = new PessoaFisica(id: 1, nome: "lucreio", cpf: "35412665478", idade: 20);
        PessoaFisica pessoa2 = new PessoaFisica(id: 2, nome: "lucreia", cpf: "12345678900", idade: 19);
        repo1.inserir(pessoaFisica:pessoa1);
        repo1.inserir(pessoaFisica:pessoa2);
        try {
            repo1.persistir(nomeArquivo:"CPF.bin");
            System.out.println(x:"CPFs Salvo com sucesso!");

        } catch (IOException ex) {
            Logger.getLogger(name: Main.class.getName()).log(level:Level.SEVERE, msg: "Erro ao Salvar", thrown: ex);
        }

        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
        try {
            System.out.println(x:"CPFs Carregado com sucesso!");
            repo2.recuperar(nomeArquivo:"CPF.bin");
            List<PessoaFisica> obterTodosCpf = repo2.obterTodos();
            for (PessoaFisica pessoaFisica : obterTodosCpf) {
                pessoaFisica.exibir();
            }
        }
    } catch (IOException ex) {
        Logger.getLogger(name: Main.class.getName()).log(level:Level.SEVERE, msg: "Erro ao Carregar", thrown: ex);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(name: Main.class.getName()).log(level:Level.SEVERE, msg: "Erro ao Carregar", thrown: ex);
    }
    PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
    PessoaJuridica pessoa3 = new PessoaJuridica(id: 3, nome: "SilasTurbo", cnpj: "3214562312");
    PessoaJuridica pessoa4 = new PessoaJuridica(id: 4, nome: "AutoCores", cnpj: "2314567891");
    repo3.inserir(pessoaJuridica:pessoa3);
    repo3.inserir(pessoaJuridica:pessoa4);
    try {
        System.out.println(x:"CNPJs Salvo com sucesso!");
        repo3.persistir(nomeArquivo:"CNPJS.bin");
    } catch (IOException ex) {
        Logger.getLogger(name: Main.class.getName()).log(level:Level.SEVERE, msg: "Erro ao Salvar", thrown: ex);
    }
    PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
    try {
        repo4.recuperar(nomeArquivo:"CNPJS.bin");
        System.out.println(x:"CNPJs Carregado com sucesso!");
    } catch (IOException ex) {
        Logger.getLogger(name: Main.class.getName()).log(level:Level.SEVERE, msg: "Erro ao Carregar", thrown: ex);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(name: Main.class.getName()).log(level:Level.SEVERE, msg: "Erro ao Carregar", thrown: ex);
    }
    List<PessoaJuridica> obterTodosCnpj = repo4.obterTodos();
    for (PessoaJuridica pessoaJuridica : obterTodosCnpj) {
        pessoaJuridica.exibir();
    }

  }
}
```

Pessoa:

```java
package com.mycompany.cadastropoo.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:

```java
package com.mycompany.cadastropoo.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;
    }

    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:

```java
package com.mycompany.cadastropoo.model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {
    private String cnpj;

    public PessoaJuridica() {
        super();
        this.cnpj = "";
    }

    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);
    }
}
```

PessoaFisicaRepo:

```java
package com.mycompany.cadastropoo.model;

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

public class PessoaFisicaRepo {
    private List<PessoaFisica> listaPessoasFisicas;

    public PessoaFisicaRepo() {
        listaPessoasFisicas = new ArrayList<>();
    }

    public void inserir(PessoaFisica pessoaFisica) {
        listaPessoasFisicas.add(e: pessoaFisica);
    }

    public void alterar(PessoaFisica pessoaFisica) {
        for (int i = 0; i < listaPessoasFisicas.size(); i++) {
            PessoaFisica pessoaAtual = listaPessoasFisicas.get(index: i);
            if (pessoaAtual.getId() == pessoaFisica.getId()) {
                listaPessoasFisicas.set(index: i, element:pessoaFisica);
                return;
            }
        }
    }

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

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

```java
    public List<PessoaFisica> obterTodos() {
        return listaPessoasFisicas;
    }

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(name: nomeArquivo))) {
            oos.writeObject(obj: listaPessoasFisicas);
        }
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(name: nomeArquivo))) {
            listaPessoasFisicas = (List<PessoaFisica>) ois.readObject();
        }
    }
}
```

PessoaJuridicaRepo:

```java
package com.mycompany.cadastropoo.model;

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

public class PessoaJuridicaRepo {
    private List<PessoaJuridica> listaPessoasJuridicas;

    public PessoaJuridicaRepo() {
        listaPessoasJuridicas = new ArrayList<>();
    }

    public void inserir(PessoaJuridica pessoaJuridica) {
        listaPessoasJuridicas.add(e: pessoaJuridica);
    }

    public void alterar(PessoaJuridica pessoaJuridica) {
        for (int i = 0; i < listaPessoasJuridicas.size(); i++) {
            PessoaJuridica pessoaAtual = listaPessoasJuridicas.get(index: i);
            if (pessoaAtual.getId() == pessoaJuridica.getId()) {
                listaPessoasJuridicas.set(index: i, element: pessoaJuridica);
                return;
            }
        }
    }

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

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

    public List<PessoaJuridica> obterTodos() {
        return listaPessoasJuridicas;
    }

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(name: nomeArquivo))) {
            oos.writeObject(obj: listaPessoasJuridicas);
        }
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(name: nomeArquivo))) {
            listaPessoasJuridicas = (List<PessoaJuridica>) ois.readObject();
        }
    }
}
```

Resultado:

```
] --- exec:3.1.0:exec (default-cli) @ CadastroPOO ---
  CPFs Salvo com sucesso!
  CPFs Carregado com sucesso!
  ID: 1
  Nome: lucreio
  CPF: 35412665478
  Idade: 20
  ID: 2
  Nome: lucreia
  CPF: 12345678900
  Idade: 19
  CNPJs Salvo com sucesso!
  CNPJs Carregado com sucesso!
  ID: 3
  Nome: SilasTurbo
  CNPJ: 3214562312
  ID: 4
  Nome: AutoCores
- CNPJ: 2314567891
  ------------------------------------------------------------------------
  BUILD SUCCESS
  ------------------------------------------------------------------------
  Total time:  2.943 s
  Finished at: 2023-09-23T13:03:29-03:00
  ------------------------------------------------------------------------
```

Análise e Conclusão:

A herança na programação orientada a objetos oferece a vantagem de reutilização de código, estrutura hierárquica organizada e polimorfismo. No entanto, ela também apresenta desvantagens, como acoplamento forte, complexidade e risco de fragilidade do contrato. A decisão de usar herança deve ser equilibrada com base nas necessidades específicas do projeto e no cuidado com a manutenção futura do código.

A interface Serializable é necessária ao efetuar persistência em arquivos binários porque ela indica que uma classe pode ser serializada, ou seja, seus objetos podem ser convertidos em uma sequência de bytes para serem armazenados ou transmitidos. Ao implementar Serializable, a classe fornece informações ao sistema de que seus objetos podem ser gravados e lidos em arquivos binários, garantindo a integridade dos dados ao realizar a persistência. Isso é fundamental para que a leitura e gravação de objetos em formato binário ocorram de forma correta e segura.

O paradigma funcional é amplamente utilizado pela API Stream no Java para operações de processamento de dados de maneira mais concisa e declarativa. Por meio de operações como map, filter, reduce e forEach, a API Stream permite que os programadores apliquem funções lambda a coleções de dados de forma funcional, evitando loops explícitos. Isso resulta em um código mais legível, modular e paralelizável, aumentando a eficiência e expressividade na manipulação de dados em Java.

Em Java, um padrão comum de desenvolvimento para a persistência de dados em arquivos é o uso da serialização e desserialização de objetos. Isso envolve a implementação da interface Serializable nas classes cujos objetos serão armazenados em arquivos binários, permitindo que sejam convertidos em bytes e gravados em disco. Isso é frequentemente utilizado para salvar e recuperar objetos em aplicativos que não requerem um banco de dados completo.