Campus: POLO SAGUAÇU - JOINVILLE - SC

Curso: DESENVOLVIMENTO FULL STACK

Disciplina: Iniciando o Caminho Pelo Java

Turma: 9001

Semestre: 1° Semestre (2024)

Aluno: Jederson Borges de Oliveira

Link: https://github.com/JedersonBorges/CadastroPOO

Iniciando o caminho pelo Java

Objetivos da prática

1. **Utilizar herança e polimorfismo na definição de entidades.**
2. **Utilizar persistência de objetos em arquivos binários.**
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.**

Códigos utilizados

Classe 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 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 + ", Nome: " + nome);
    }
}
```

Classe PessoaFisica:

```java
package 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() {
        System.out.println("ID: " + getId());
        System.out.println("Nome: " + getNome());
        System.out.println("CPF: " + cpf);
        System.out.println("Idade: " + idade);
        System.out.println();
    }
}
```

Classe PessoaJuridica:

```
package model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {
    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);
        System.out.println(); // Linha em branco para separar as
pessoas
    }
}
```

Classe PessoaFisicaRepo:

```
package model;

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

public class PessoaFisicaRepo {
    private List<PessoaFisica> pessoasFisicas;

    public PessoaFisicaRepo() {
        this.pessoasFisicas = new ArrayList<>();
    }

    public void inserir(PessoaFisica pessoa) {
        pessoasFisicas.add(pessoa);
    }

    public void alterar(PessoaFisica pessoa) {
        for (int i = 0; i < pessoasFisicas.size(); i++) {
            if (pessoasFisicas.get(i).getId() == pessoa.getId()) {
                pessoasFisicas.set(i, pessoa);
                return;
            }
        }
    }

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

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

    public List<PessoaFisica> obterTodos() {
        return new ArrayList<>(pessoasFisicas);
    }

    public void persistir(String arquivo) throws IOException {
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(arquivo))) {
            oos.writeObject(pessoasFisicas);
        }
    }

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

```
    }
}
```

Classe PessoaFisicaRepo:

```
package model;

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

public class PessoaJuridicaRepo {
    private List<PessoaJuridica> pessoasJuridicas;

    public PessoaJuridicaRepo() {
        this.pessoasJuridicas = new ArrayList<>();
    }

    public void inserir(PessoaJuridica pessoa) {
        pessoasJuridicas.add(pessoa);
    }

    public void alterar(PessoaJuridica pessoa) {
        for (int i = 0; i < pessoasJuridicas.size(); i++) {
            if (pessoasJuridicas.get(i).getId() == pessoa.getId()) {
                pessoasJuridicas.set(i, pessoa);
                return;
            }
        }
    }

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

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

    public List<PessoaJuridica> obterTodos() {
        return new ArrayList<>(pessoasJuridicas);
    }

    public void persistir(String arquivo) throws IOException {
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(arquivo))) {
            oos.writeObject(pessoasJuridicas);
        }
    }

    public void recuperar(String arquivo) throws IOException,
ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(arquivo))) {
            pessoasJuridicas = (List<PessoaJuridica>)
```

```
ois.readObject();
        }
    }
}
```

Main:

```
package model;

import java.io.IOException;

public class Main {
    public static void main(String[] args) {
        try {
            PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

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

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

            PessoaFisicaRepo repo2 = new PessoaFisicaRepo();

            repo2.recuperar(arquivoFisica);
            System.out.println("Dados de Pessoa Fisica Recuperados.");

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

            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 arquivoJuridica = "pessoasJuridicas.dat";
            repo3.persistir(arquivoJuridica);
            System.out.println("Dados de Pessoa Juridica
Armazenados.");

            PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();

            repo4.recuperar(arquivoJuridica);
            System.out.println("Dados de Pessoa Juridica
Recuperados.");


            for (PessoaJuridica pj : repo4.obterTodos()) {
                pj.exibir();
            }
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
```

```
        }
    }
}
```

Resultado da execução dos códigos:

```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagen
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


Process finished with exit code 0
```

Análise e Conclusão

Quais as vantagens e desvantagens do uso de herança?

A herança permite a reutilização de código, facilitando a manutenção do sistema, em contrapartida pode deixar o código mais complexo e menos flexível

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

A interface Serializable é necessária para permitir que os objetos sejam convertidos em formatos binários e reconstruídos posteriormente

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

A API Stream no Java utiliza o paradigma funcional para manipular coleções de dados de forma mais eficiente e clara.

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

É adotado o uso de serialização, onde os objetos são convertidos em bytes e armazenados em arquivos binários