Estácio

Missão Prática - Nível 1 – Mundo 3

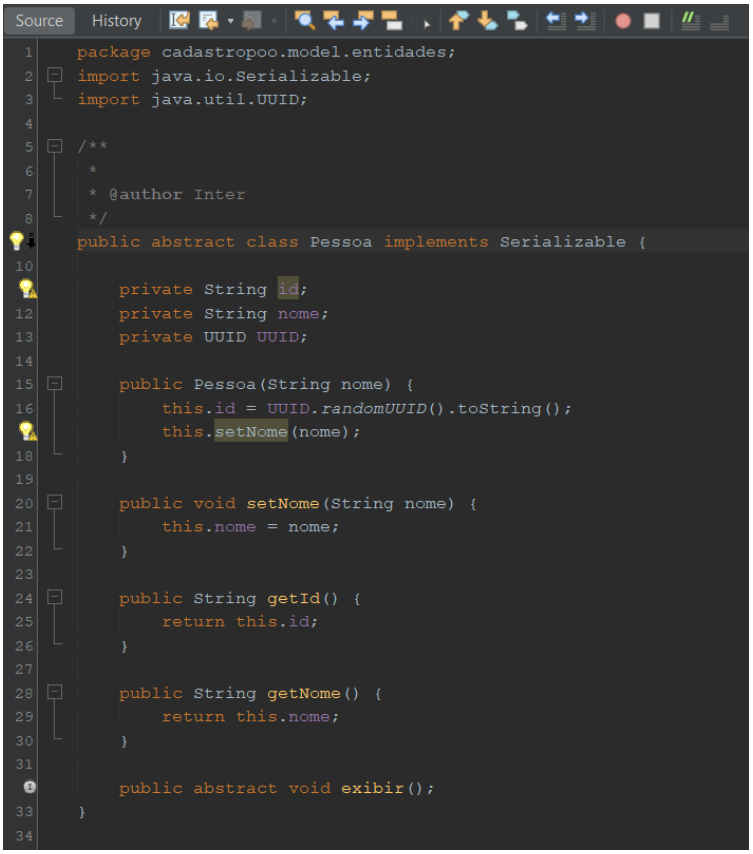
João Guilherme Fernandes Borba - 2022 0851 5835

POLO IPIRANGA - PORTO ALEGRE
Desenvolvimento Full Stack
RPG0014 - Iniciando o caminho pelo Java! – Turma 9001 – 3º Semestre

1. Objetivos da Prática

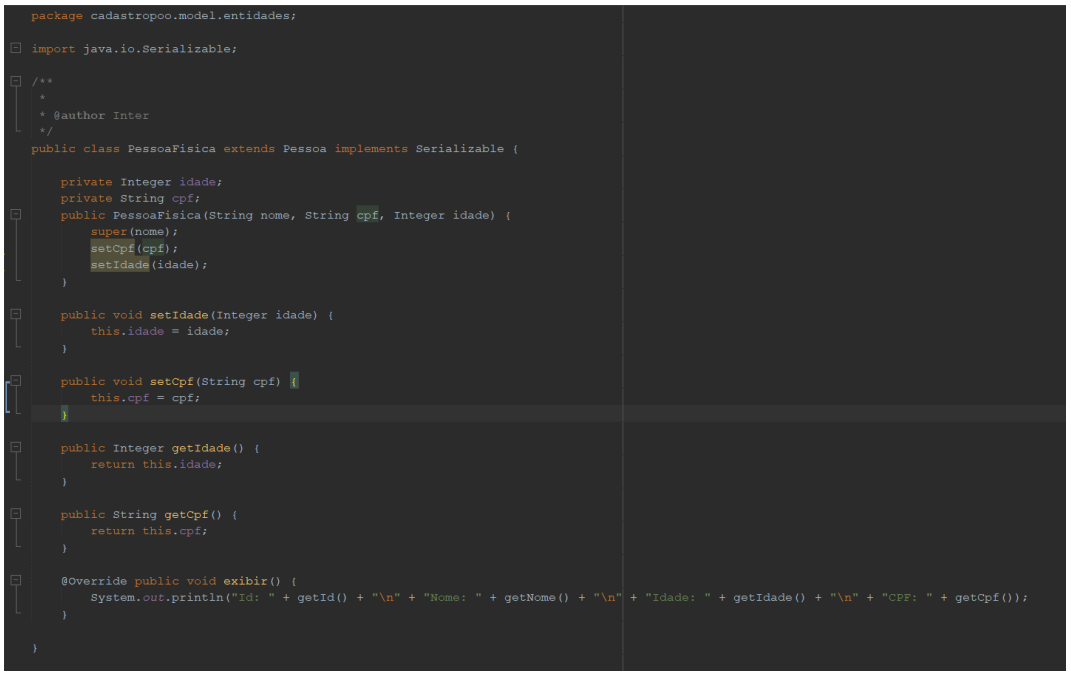
1. Utilizar herança e polimorfismo na definição de entidades.
2. Utilizar persistência de objetos em arquivos binários.
3. Implementar uma interface cadastral em modo texto.
4. Utilizar o controle de exceções da plataforma Java.
5. No final do projeto, o aluno terá implementado um sistema cadastral em Java,
6. utilizando os recursos da programação orientada a objetos e a persistência em
7. arquivos binários.

2. Classe Abstrata Pessoa



```java
package cadastropoo.model.entidades;
import java.io.Serializable;
import java.util.UUID;

/**
 *
 * @author Inter
 */
public abstract class Pessoa implements Serializable {

    private String id;
    private String nome;
    private UUID UUID;

    public Pessoa(String nome) {
        this.id = UUID.randomUUID().toString();
        this.setNome(nome);
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getId() {
        return this.id;
    }

    public String getNome() {
        return this.nome;
    }

    public abstract void exibir();
}
```

2.1. Classe PessoaFisica



```java
package cadastropoo.model.entidades;

import java.io.Serializable;

/**
 *
 * @author Inter
 */
public class PessoaFisica extends Pessoa implements Serializable {

    private Integer idade;
    private String cpf;
    public PessoaFisica(String nome, String cpf, Integer idade) {
        super(nome);
        setCpf(cpf);
        setIdade(idade);
    }

    public void setIdade(Integer idade) {
        this.idade = idade;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public Integer getIdade() {
        return this.idade;
    }

    public String getCpf() {
        return this.cpf;
    }

    @Override public void exibir() {
        System.out.println("Id: " + getId() + "\n" + "Nome: " + getNome() + "\n" + "Idade: " + getIdade() + "\n" + "CPF: " + getCpf());
    }

}
```

2.2. Classe PessoaJuridica

```java
package cadastropoo.model.entidades;

import java.io.Serializable;

/**
 *
 * @author Inter
 */
public class PessoaJuridica extends Pessoa implements Serializable {

    private String cnpj;

    public PessoaJuridica(String nome, String cnpj) {
        super(nome);
        setCnpj(cnpj);
    }

    private void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }

    public String getCnpj() {
        return this.cnpj;
    }

    @Override
    public void exibir() {
        System.out.println("Id: " + getId() + "\n" + "Nome: " + getNome() + "\n" + "CNPJ: " + getCnpj());
    }
}
```

2.3. Classe PessoaFisicaRepo

```java
package cadastropoo.model.gerenciadores;

import cadastropoo.model.entidades.PessoaFisica;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.List;

/**...4 lines */
public class PessoaFisicaRepo {

    private List<PessoaFisica> pessoas = new ArrayList<PessoaFisica>();

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

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

    public void exlcuir(String id) {
        pessoas.removeIf(pessoa -> pessoa.getId().equals(id));
    }

    public PessoaFisica obter(String id) {
        return pessoas.stream()
                .filter(e -> e.getId().equals(id))
                .findFirst()
                .orElse(null);
    }
```

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

    public void persistir(String arqNome) throws IOException {
        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(arqNome))) {
            out.writeObject(pessoas);
        }
    }

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

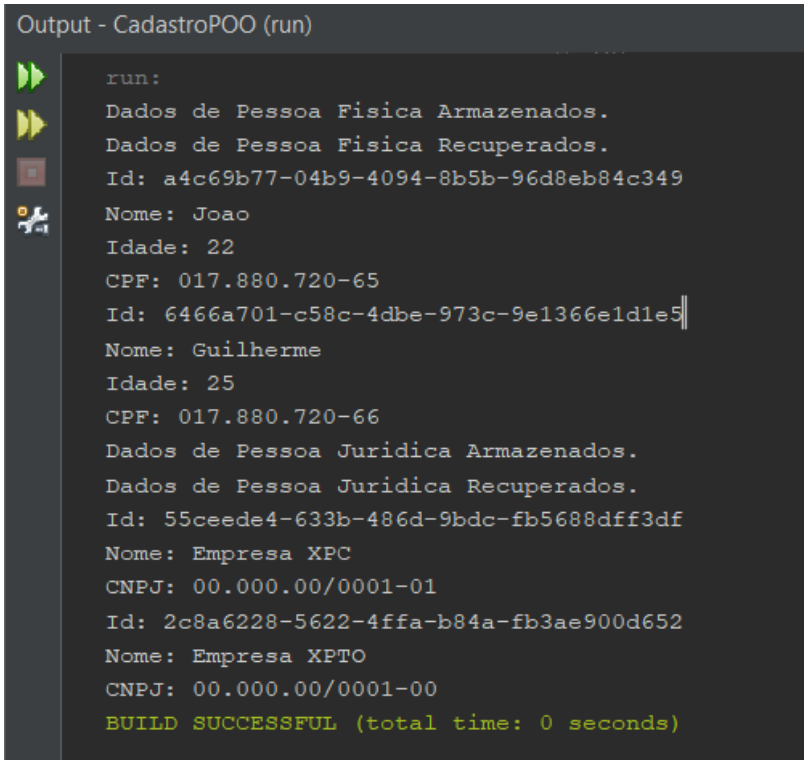
2.4. Classe PessoaJuridicaRepo

```
package cadastropoo.model.gerenciadores;

import cadastropoo.model.entidades.PessoaJuridica;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Inter
 */
public class PessoaJuridicaRepo {

    private List<PessoaJuridica> pessoas = new ArrayList<PessoaJuridica>();

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

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

    public void exlcuir(String id) {
        pessoas.removeIf(pessoa -> pessoa.getId().equals(id));
    }

    public PessoaJuridica obter(String id) {
        return pessoas.stream()
                .filter(e -> e.getId().equals(id))
                .findFirst()
                .orElse(null);
    }

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

     public void persistir(String arqNome) throws IOException {
         try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(arqNome))) {
             out.writeObject(pessoas);
         }
     }

     public void recuperar(String arqNome) throws IOException, ClassNotFoundException {
         try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(arqNome))) {
             pessoas = (List<PessoaJuridica>) in.readObject();
         }
     }
}
```

2.5. Class CadastroPOO

```java
package cadastropoo;

import cadastropoo.model.entidades.PessoaJuridica;
import cadastropoo.model.entidades.PessoaFisica;
import cadastropoo.model.gerenciadores.PessoaFisicaRepo;
import cadastropoo.model.gerenciadores.PessoaJuridicaRepo;
import java.io.IOException;

/**
 *
 * @author Inter
 */
public class CadastroPOO {

    public static void main(String[] args) {
        String arquivoPF = "pessoas_fisicas.dat";
        String arquivoPJ = "pessoas_juridicas.dat";

        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
        repo1.inserir(new PessoaFisica("Joao", "017.880.720-65", 22));
        repo1.inserir(new PessoaFisica("Guilherme", "017.880.720-66", 25));

        try {
            repo1.persistir(arquivoPF);
            System.out.println("Dados de Pessoa Fisica Armazenados.");
        } catch (IOException e) {
            System.out.println("Error ao armazenar dados de Pessoa Fisica: " + e.getMessage());
        }

        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
        try {
            repo2.recuperar(arquivoPF);
            System.out.println("Dados de Pessoa Fisica Recuperados.");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao recuperar dados de Pessoa Fisica: " + e.getMessage());
        }

        repo2.obterTodos().forEach(PessoaFisica::exibir);

        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();

        repo3.inserir(new PessoaJuridica("Empresa XPC", "00.000.00/0001-01"));
        repo3.inserir(new PessoaJuridica("Empresa XPTO", "00.000.00/0001-00"));

        try {
            repo3.persistir(arquivoPJ);
            System.out.println("Dados de Pessoa Juridica Armazenados.");
        } catch (IOException e) {
            System.out.println("Error ao armazenar dados de Pessoa Juridica: " + e.getMessage());
        }

        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
        try {
            repo4.recuperar(arquivoPJ);
            System.out.println("Dados de Pessoa Juridica Recuperados.");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Error ao recuperar dados de Pessoa Juridica: " + e.getMessage());
        }

        repo4.obterTodos().forEach(PessoaJuridica::exibir);

    }

}
```

3. Resultados da Execução



4. Análise e Conclusão

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

- **Vantagens:** Reutilização de código, estrutura hierárquica clara e maior organização do código.
- **Desvantagens:** Pode levar a um forte acoplamento entre classes e tornar o código mais difícil de modificar.

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

A interface Serializable é essencial porque permite que os objetos sejam convertidos em bytes e armazenados em arquivos ou transmitidos pela rede. Sem essa interface, o Java não permitiria a conversão automática para um formato serializável.

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

A API Stream permite operar coleções de forma funcional, utilizando métodos como filter, map, forEach, reduzindo a necessidade de loops explícitos e aumentando a legibilidade do código.

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

O padrão mais utilizado é o **DAO (Data Access Object)**, que separa a lógica de negócios da manipulação de dados, garantindo um código mais modular e organizado.

---

5. Repositório Git

O código-fonte está disponível em: GitHub

---

6. Conclusão Final

O experimento demonstrou a importância da serialização e do uso de Stream para manipulação de dados.