Centro Universitário Estácio do Ceará - Campus Centro

Curso: Desenvolvimento Full Stack

Disciplina: Nível 1: Iniciando o Caminho Pelo Java

Número da Turma: RPG0014

Semestre Letivo: 3

Integrantes: Fernando Rocha Fonteles Filho

Repositorio Git: https://github.com/fernandorff/EstacioFullStack_Mundo3_Nivel1_Procedimento1

Título da Prática: 1º Procedimento | Criação das Entidades e Sistema de Persistência

Objetivos 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.

Códigos solicitados neste roteiro de aula:

- Classe Pessoa (Entidade)

```
package EstacioFullStack_Mundo3_Nivel1_Procedimento1.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);
    }
}
```

- Classe PessoaFisica (Entidade)

```
package EstacioFullStack_Mundo3_Nivel1_Procedimento1.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);
    }
}
```

- Classe PessoaJuridica (Entidade)

```
package EstacioFullStack_Mundo3_Nivel1_Procedimento1.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);
    }
}
```

- Classe PessoaFisicaRepo (Gerenciador)

```
package EstacioFullStack_Mundo3_Nivel1_Procedimento1.model.gerenciadores;

import EstacioFullStack_Mundo3_Nivel1_Procedimento1.model.PessoaFisica;

import java.io.*;
import java.util.ArrayList;
import java.util.NoSuchElementException;
import java.util.Optional;

public class PessoaFisicaRepo {
    private ArrayList<PessoaFisica> pessoasFisicas;

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

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

    public void alterar(PessoaFisica pessoaFisica, String novoNome, String novoCpf, int novaIdade) {
        pessoaFisica.setNome(novoNome);
        pessoaFisica.setCpf(novoCpf);
        pessoaFisica.setIdade(novaIdade);
    }

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

    public PessoaFisica obter(int id) throws NoSuchElementException {
        Optional<PessoaFisica> pessoaFisicaEncontrada = pessoasFisicas.stream().
                filter(pessoaFisica -> pessoaFisica.getId() == id)
                .findFirst();

        if (pessoaFisicaEncontrada.isPresent()) {
            return pessoaFisicaEncontrada.get();
        } else {
            throw new NoSuchElementException("Pessoa física com ID " + id + " não encontrada.");
        }
    }

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

    public void persistir(String nomeArquivo) throws IOException {
        ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo));
        outputStream.writeObject(pessoasFisicas);
        outputStream.close();
        System.out.println("Dados da pessoa física armazenados.");
        System.out.println();
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo));
        pessoasFisicas = (ArrayList<PessoaFisica>) inputStream.readObject();
        inputStream.close();
        System.out.println("Dados da pessoa física recuperados.");
        System.out.println();
    }
}
```

- Classe PessoaJuridicaRepo (Gerenciador)

```
package EstacioFullStack_Mundo3_Nivel1_Procedimento1.model.gerenciadores;

import EstacioFullStack_Mundo3_Nivel1_Procedimento1.model.PessoaJuridica;

import java.io.*;
import java.util.ArrayList;
import java.util.NoSuchElementException;
import java.util.Optional;

public class PessoaJuridicaRepo {
    private ArrayList<PessoaJuridica> pessoasJuridicas;
```

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

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

    public void alterar(PessoaJuridica pessoaJuridica, String novoNome, String novoCpf) {
        pessoaJuridica.setNome(novoNome);
        pessoaJuridica.setCnpj(novoCpf);
    }

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

    public PessoaJuridica obter(int id) throws NoSuchElementException {
        Optional<PessoaJuridica> pessoaJuridicaEncontrada = pessoasJuridicas.stream().
                filter(pessoaJuridica -> pessoaJuridica.getId() == id)
                .findFirst();

        if (pessoaJuridicaEncontrada.isPresent()) {
            return pessoaJuridicaEncontrada.get();
        } else {
            throw new NoSuchElementException("Pessoa jurídica com ID " + id + " não encontrada.");
        }
    }

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

    public void persistir(String nomeArquivo) throws IOException {
        ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo));
        outputStream.writeObject(pessoasJuridicas);
        outputStream.close();
        System.out.println("Dados da pessoa jurídica armazenados.");
        System.out.println();
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo));
        pessoasJuridicas = (ArrayList<PessoaJuridica>) inputStream.readObject();
        inputStream.close();
        System.out.println("Dados da pessoa jurídica recuperados.");
        System.out.println();
    }
}
```

- Classe Application (Aplicação)

```
package EstacioFullStack_Mundo3_Nivel1_Procedimento1;

import EstacioFullStack_Mundo3_Nivel1_Procedimento1.model.PessoaFisica;
import EstacioFullStack_Mundo3_Nivel1_Procedimento1.model.PessoaJuridica;
import EstacioFullStack_Mundo3_Nivel1_Procedimento1.model.gerenciadores.PessoaFisicaRepo;
import EstacioFullStack_Mundo3_Nivel1_Procedimento1.model.gerenciadores.PessoaJuridicaRepo;

import java.io.IOException;

public class Application {

    public static void main(String[] args) {

        PessoaFisicaRepo pessoaFisicaRepo1 = new PessoaFisicaRepo();

        PessoaFisica pessoaFisica1 = new PessoaFisica(1, "John", "11111111111", 30);
        PessoaFisica pessoaFisica2 = new PessoaFisica(2, "Mary", "22222222222", 25);
        pessoaFisicaRepo1.inserir(pessoaFisica1);
        pessoaFisicaRepo1.inserir(pessoaFisica2);

        try {
            pessoaFisicaRepo1.persistir("pessoas_fisicas.dat");

            PessoaFisicaRepo pessoaFisicaRepo2 = new PessoaFisicaRepo();
            pessoaFisicaRepo2.recuperar("pessoas_fisicas.dat");
```

```
            pessoaFisicaRepo2.obterTodos()
                    .forEach(pessoaFisica -> {
                        pessoaFisica.exibir();
                        System.out.println();
                    });

        } catch (IOException | ClassNotFoundException erro) {
            System.out.println("Erro ao persistir ou recuperar os dados: " + erro.getMessage());
        }

        PessoaJuridicaRepo pessoaJuridicaRepo1 = new PessoaJuridicaRepo();

        PessoaJuridica pessoaJuridica1 = new PessoaJuridica(1, "X Corp", "1231231230");
        PessoaJuridica pessoaJuridica2 = new PessoaJuridica(2, "Acme LTDA", "4564564560");
        pessoaJuridicaRepo1.inserir(pessoaJuridica1);
        pessoaJuridicaRepo1.inserir(pessoaJuridica2);

        try {
            pessoaJuridicaRepo1.persistir("pessoas_juridicas.dat");

            PessoaJuridicaRepo pessoaJuridicaRepo2 = new PessoaJuridicaRepo();
            pessoaJuridicaRepo2.recuperar("pessoas_juridicas.dat");

            pessoaJuridicaRepo2.obterTodos().stream()
                    .forEach(pessoaJuridica -> {
                        pessoaJuridica.exibir();
                        System.out.println();
                    });

        } catch (IOException | ClassNotFoundException erro) {
            System.out.println("Erro ao persistir ou recuperar os dados: " + erro.getMessage());
        }
    }
}
```

- Resultado da execução do código.

```
Dados da pessoa física armazenados.

Dados da pessoa física recuperados.

ID: 1
Nome: John
CPF: 11111111111
Idade: 30

ID: 2
Nome: Mary
CPF: 22222222222
Idade: 25

Dados da pessoa jurídica armazenados.

Dados da pessoa jurídica recuperados.

ID: 1
Nome: X Corp
CNPJ: 1231231230

ID: 2
Nome: Acme LTDA
CNPJ: 4564564560


Process finished with exit code 0
```

Análise e Conclusão

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

A herança tem como principal objetivo reutilizar funcionalidades de outros elementos da aplicação por meio da extensão de atributos e métodos de uma classe.

A vantagem da herança é a possibilidade de se definir novos atributos e métodos para a classe, além dos atributos e métodos que essa classe está herdando.

A desvantagem de se utilizar a herança é enfraquecer o conceito de encapsulamento que é importante no paradigma de orientação a objeto. Além disso, torna a subclasse dependente da implementação da classe superior, tornando o código mais difícil de se manter, pois mudanças na classe superior pode afetar todas as classes que herdam dela.

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

Essa interface permite que os objetos sejam serializados, ou seja, convertidos em uma sequência de bytes e desserializados com a conversão de volta à um objeto.

A vantagem de se usar arquivos binários é que eles ocupam menos espaço e podem ser escritos e lidos mais rapidamente que um arquivo de texto normal. Além disso, podem preservar o tipo e a estrutura dos dados.

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

A API Stream do Java permite a execução de operações em sequência de forma declarativa usando o paradigma funcional. É possível especificar o que fazer com os elementos da sequência e criar um encadeamento de processamento de dados não modificam a fonte dos dados original.

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

O padrão adotado nesse projeto foi com o uso da classe ObjectOutputStream para escrever objetos em um arquivo binário e a classe ObjectInputStream para ler os objetos de um arquivo binário.

Além disso é possível persistir dados por meio de arquivos de texto simples, CSV, XML, JSON e outros.