Campus: Polo Azenha | Curso: Desenvolvimento Full Stack

Disciplina: Iniciando o Caminho Pelo Java | Número da Turma: 9003

Semestre Letivo: 3º | Integrantes da Prática: João Pedro da Silva Zanirati Nunes

Repositório no GIT: https://github.com/JPZanirati/MP_m3_n1

Título da Prática: Criação das Entidades e Sistema de Persistência

Objetivo da Prática: Criar um código Java para gerenciar informações de Pessoas Físicas e Jurídicas. Foram utilizados conceitos de herança de classes, serialização de objetos para salvar em arquivos e a implementação de uma interface para persistência.

Classe Pessoa:

```
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 void exibir() {
        System.out.println("ID: " + id);
        System.out.println("Nome: " + nome);
    }

    // Getters e setters para id e 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;
    }
}
```

Classe PessoaFisica:

```
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;
    }

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CPF: " + cpf);
        System.out.println("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;
    }
}
```

Classe PessoaJuridico:

```
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;
    }

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
    }

    public String getCnpj() {
        return cnpj;
    }

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

Classe PessoaFisicaRepo:

```
package model;

public class PessoaFisicaRepo implements Serializable {
    private static final long serialVersionUID = 1L;
    private final List<PessoaFisica> pessoasFisicas = new ArrayList<>();

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

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

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

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

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

    public void persistir(String nomeArquivo) throws IOException {
         try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo)))
{
            outputStream.writeObject(this);
        }
    }

     public static PessoaFisicaRepo recuperar(String nomeArquivo) throws IOException, ClassNotFoundException
{
        try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            return (PessoaFisicaRepo) inputStream.readObject();
        }
    }
}
```

Classe PessoaJuridicaRepo:

```
public class PessoaJuridicaRepo implements Serializable {
    private static final long serialVersionUID = 1L;
    private final List<PessoaJuridica> pessoasJuridicas = new ArrayList<>();

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

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

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

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

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

    public void persistir(String nomeArquivo) throws IOException {
         try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo)))
{
            outputStream.writeObject(this);
        }
    }

                public static PessoaJuridicaRepo recuperar(String nomeArquivo) throws IOException,
ClassNotFoundException {
        try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            return (PessoaJuridicaRepo) inputStream.readObject();
        }
    }
}
```

Classe Main:

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

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

        try {
            repo1.persistir("pessoas_fisicas.txt");
            System.out.println("Dados de Pessoa Fisica Armazenados.");
        } catch (IOException e) {
            System.err.println("Erro ao persistir dados de Pessoa Fisica: " + e.getMessage());
        }

        PessoaFisicaRepo repo2;

        try {
            repo2 = PessoaFisicaRepo.recuperar("pessoas_fisicas.txt");
            System.out.println("Dados de Pessoa Fisica Recuperados.");

            for (PessoaFisica pessoa : repo2.obterTodos()) {
                System.out.println("Id: " + pessoa.getId());
                System.out.println("Nome: " + pessoa.getNome());
                System.out.println("CPF: " + pessoa.getCpf());
                System.out.println("Idade: " + pessoa.getIdade());
                System.out.println();
            }
        } catch (IOException | ClassNotFoundException e) {
            System.err.println("Erro ao recuperar dados de Pessoa Fisica: " + e.getMessage());
        }

        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();

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

        try {
            repo3.persistir("pessoas_juridicas.txt");
            System.out.println("Dados de Pessoa Juridica Armazenados.");
        } catch (IOException e) {
            System.err.println("Erro ao persistir dados de Pessoa Juridica: " + e.getMessage());
        }

        PessoaJuridicaRepo repo4;

        try {
            repo4 = PessoaJuridicaRepo.recuperar("pessoas_juridicas.txt");
            System.out.println("Dados de Pessoa Juridica Recuperados.");

            for (PessoaJuridica pessoa : repo4.obterTodos()) {
                System.out.println("Id: " + pessoa.getId());
                System.out.println("Nome: " + pessoa.getNome());
                System.out.println("CNPJ: " + pessoa.getCnpj());
                System.out.println();
```

```
            }
        } catch (IOException | ClassNotFoundException e) {
            System.err.println("Erro ao recuperar dados de Pessoa Juridica: " + e.getMessage());
        }
    }
}
```

```
run:
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

BUILD SUCCESSFUL (total time: 7 seconds)
```

Vantagens da herança:

Ela permite reutilizar código e criar hierarquias de classes, tornando o código mais organizado e promovendo o polimorfismo. Isso facilita a manutenção e economiza tempo.

Desvantagem da herança:

Herdar pode causar problemas, pois as mudanças na classe principal afetam as classes filhas, tornando o código mais complicado e difícil de consertar.

Serializable:

É uma ferramenta que permite que objetos sejam transformados em uma forma que pode ser salva em arquivos ou transmitida pela rede, e depois restaurados para sua forma original.

API Stream no Java:

Ajuda a lidar com listas de dados de forma mais fácil e rápida. Ela utiliza algumas técnicas especiais que tornam a programação mais simples e eficiente. Por exemplo, em vez de escrever muitas linhas de código, você pode fazer tarefas complexas em apenas algumas linhas com a ajuda da API Stream.