Campus: POLO VILA DOS REMÉDIOS – OSASCO – SP
Curso: Desenvolvimento Full Stack
Nome Disciplina: Nível 1: Iniciando o caminho pelo Java
Matrícula: 2023 0397 9797
Semestre Letivo: 3º semestre
Nome: Anderson Barbosa Almeida

Repositório no GIT: https://github.com/andydevbarbosa/RPG0014---Iniciando-o-caminho-pelo-Java/commit/66e1c7d62ea69b263d015d04ec952b54e1f8667a

Título da Prática: Criação do Cadastro em Modo Texto

Objetivo da Prática: O objetivo desta prática foi implementar um sistema de gerenciamento de entidades em Java, utilizando herança, persistência em arquivos e conceitos básicos de orientação a objetos.

Códigos:
Aqui estão os códigos desenvolvidos durante a prática:

```
package cadastropoo;
import java.io.*;
import java.util.ArrayList;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

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);
    }
}

class PessoaFisica extends Pessoa {
    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 + ", Idade: " + idade);
    }
}

class PessoaJuridica extends Pessoa {
    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);
    }
}

class PessoaFisicaRepo {
    private ArrayList<PessoaFisica> pessoasFisicas = new ArrayList<>();

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

Estácio

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

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

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

    public ArrayList<PessoaFisica> obterTodos() {
        return 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 = (ArrayList<PessoaFisica>) ois.readObject();
        }
    }
}
```

Estácio

```
class PessoaJuridicaRepo {
    private ArrayList<PessoaJuridica> 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);
                break;
            }
        }
    }

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

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

    public ArrayList<PessoaJuridica> obterTodos() {
        return 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 {
```

Estácio

```
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(arquivo))) {
            pessoasJuridicas = (ArrayList<PessoaJuridica>) ois.readObject();
        }
    }
}

public class CadastroPOO {
    public static void main(String[] args) {
        BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));

        PessoaFisicaRepo repoPessoaFisica = new PessoaFisicaRepo();
        PessoaJuridicaRepo repoPessoaJuridica = new PessoaJuridicaRepo();

        try {
            int opcao;
            do {
                System.out.println("Selecione uma opção:");
                System.out.println("1 - Incluir");
                System.out.println("2 - Alterar");
                System.out.println("3 - Excluir");
                System.out.println("4 - Exibir pelo ID");
                System.out.println("5 - Exibir todos");
                System.out.println("6 - Salvar dados");
                System.out.println("7 - Recuperar dados");
                System.out.println("0 - Finalizar a execução");
                System.out.print("Opção: ");
                opcao = Integer.parseInt(reader.readLine());

                switch (opcao) {
                    case 1:
                        System.out.println("Escolha o tipo (1 - Pessoa
Física, 2 - Pessoa Jurídica): ");
                        int tipo = Integer.parseInt(reader.readLine());
                        if (tipo == 1) {
                            // Incluir Pessoa Física
                            System.out.println("Digite o ID:");
                            int id = Integer.parseInt(reader.readLine());
                            System.out.println("Digite o nome:");
                            String nome = reader.readLine();
                            System.out.println("Digite o CPF:");
                            String cpf = reader.readLine();
```

```
                            System.out.println("Digite a idade:");
                            int idade = Integer.parseInt(reader.readLine());
                            PessoaFisica pessoaFisica = new PessoaFisica(id,
nome, cpf, idade);
                            repoPessoaFisica.inserir(pessoaFisica);
                            System.out.println("Pessoa Física incluída com
sucesso.");
                        } else if (tipo == 2) {
                            // Incluir Pessoa Jurídica
                            System.out.println("Digite o ID:");
                            int id = Integer.parseInt(reader.readLine());
                            System.out.println("Digite o nome:");
                            String nome = reader.readLine();
                            System.out.println("Digite o CNPJ:");
                            String cnpj = reader.readLine();
                            PessoaJuridica pessoaJuridica = new
PessoaJuridica(id, nome, cnpj);
                            repoPessoaJuridica.inserir(pessoaJuridica);
                            System.out.println("Pessoa Jurídica incluída com
sucesso.");
                        } else {
                            System.out.println("Opção inválida.");
                        }
                        break;
                    case 2:
                        // Implementar a opção Alterar
                        break;
                    case 3:
                        // Implementar a opção Excluir
                        break;
                    case 4:
                        // Implementar a opção Exibir pelo ID
                        break;
                    case 5:
                        // Implementar a opção Exibir todos
                        break;
                    case 6:
                        // Implementar a opção Salvar dados
                        break;
                    case 7:
                        // Implementar a opção Recuperar dados
                        break;
                    case 0:
```

```
                        System.out.println("Finalizando a execução.");
                        break;
                    default:
                        System.out.println("Opção inválida.");
                }
            } while (opcao != 0);
        } catch (IOException e) {
            System.err.println("Erro de entrada/saída: " + e.getMessage());
    }
    }
}
```

Análise e Conclusão:

- **- O que são elementos estáticos e qual o motivo para o método main adotar esse modificador?**
  Elementos estáticos em Java referem-se a membros de uma classe que pertencem à classe em si, em vez de pertencerem a instâncias individuais dessa classe. Isso significa que eles são compartilhados por todas as instâncias da classe. Um método estático é um método que pertence à classe em vez de pertencer a uma instância específica da classe. O método `main` é frequentemente marcado como estático porque é o ponto de entrada para o programa Java e precisa ser chamado pelo sistema Java antes que qualquer instância da classe seja criada. Marcar o método `main` como estático permite que ele seja chamado sem criar uma instância da classe, o que é necessário para iniciar a execução do programa.

- **Para que serve a classe Scanner?**
  A classe Scanner em Java é usada para obter entrada do usuário a partir do console ou de outros fluxos de entrada, como arquivos. Ela fornece métodos para analisar e processar os dados de entrada em diferentes tipos de dados, como inteiros, ponto flutuantes, strings, etc. Isso permite que os programas Java interajam com os usuários, solicitando entrada e respondendo com base nessa entrada.

- **Como o uso de classes de repositório impactou na organização do código?** O uso de classes de repositório tem um impacto significativo na organização do código, pois ajuda a separar a lógica de negócios da lógica de persistência de dados. Ao encapsular a lógica de acesso a dados em classes de repositório dedicadas, o código torna-se mais modular e fácil de entender, pois as operações relacionadas ao armazenamento e recuperação de dados estão centralizadas em um único lugar. Isso facilita a manutenção e a extensão do código, pois qualquer alteração na lógica de acesso a dados só precisa ser feita em um lugar, em vez de espalhada por todo o código. Além disso, isso promove um melhor design orientado a objetos, pois cada classe é responsável por uma única responsabilidade, seguindo o princípio de responsabilidade única.