Campus: Polo Austin - Nova Iguaçu - RJ
Curso: Desenvolvimento Full Stack
Disciplina: Iniciando o Caminho pelo Java - **Turma:** 9001
Semestre letivo: 2024.2
Aluna: Vanessa Santana P de Souza

Título:
Criação do Cadastro em Modo Texto

Objetivo da prática:

O objetivo desta prática é desenvolver um sistema em Java que permita o cadastro, alteração, exclusão, busca e exibição de pessoas físicas e jurídicas, com a capacidade de persistir e recuperar dados de um arquivo binário. Além disso, o sistema deve ser capaz de operar em modo texto, interagindo com o usuário por meio do console.

Códigos Solicitados

Classe Pessoa

```
import java.io.Serializable;

public class Pessoa implements Serializable {
   private static final long serialVersionUID = 1L;

   private int id;
   private String nome;

   // Construtor padrão
   public Pessoa() {
   }

   // Construtor completo
   public Pessoa(int id, String nome) {
      this.id = id;
      this.nome = nome;
   }
```

```
    // Getters e Setters
    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;
    }

    // Método exibir para impressão dos dados
    public void exibir() {
        System.out.println("ID: " + id + ", Nome: " + nome);
    }

    @Override
    public String toString() {
        return "Pessoa{" +
                "id=" + id +
                ", nome='" + nome + '\'' +
                '}';
    }
}
```

Pessoa Física

```
package model;
import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {
    private static final long serialVersionUID = 1L;

    private String cpf;
    private int idade;


    // Construtor padrão
    public PessoaFisica() {
        super();
    }
```

```
    // Construtor completo
    public PessoaFisica(int id, String nome, String cpf, int idade) {
        super(id, nome); // Chama o construtor da classe base
        this.cpf = cpf;
        this.idade = idade;
    }

    // Getters e Setters
    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;
    }
```

Pessoa Jurídica

```
package model;


import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {
    private static final long serialVersionUID = 1L;

    private String cnpj;

    // Construtor padrão
    public PessoaJuridica() {
        super();
    }

    // Construtor completo
    public PessoaJuridica(int id, String nome, String cnpj) {
        super(id, nome); // Chama o construtor da classe base
        this.cnpj = cnpj;
    }
```

```
    // Getters e Setters
    public String getCnpj() {
        return cnpj;
    }

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

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

    @Override
    public String toString() {
        return "PessoaJuridica{" +
                "cnpj='" + cnpj + '\'' +
                '}';
    }
}
```

Pessoa Física Repo

```
package model;

import java.io.*;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

public class PessoaFisicaRepo implements Serializable {
    private static final long serialVersionUID = 1L;


    // ArrayList de PessoaFisica
    private List<PessoaFisica> listaPessoas;

    // Construtor
    public PessoaFisicaRepo() {
        this.listaPessoas = new ArrayList<>();
```

```
    }

    // Método para inserir uma nova PessoaFisica
    public void inserir(PessoaFisica pessoa) {
        listaPessoas.add(pessoa);
    }

    // Método para alterar uma PessoaFisica existente
    public void alterar(PessoaFisica pessoa) throws IllegalArgumentException {
        if (!listaPessoas.contains(pessoa)) {
            throw new IllegalArgumentException("Pessoa não encontrada para alteração.");
        }

        int index = listaPessoas.indexOf(pessoa);
        listaPessoas.set(index, pessoa);
    }

    // Método para excluir uma PessoaFisica pelo ID
    public void excluir(int id) {
        listaPessoas.removeIf(p -> p.getId() == id);
    }

    // Método para obter uma PessoaFisica pelo ID
    public PessoaFisica obter(int id) {
        return listaPessoas.stream()
                .filter(p -> p.getId() == id)
                .findFirst()
                .orElse(null);
    }

    // Método para obter todas as PessoasFisicas
    public List<PessoaFisica> obterTodos() {
        return new ArrayList<>(listaPessoas);
    }
    // Método para persistir os dados no disco
    public void persistir(String nomeArquivo) {
        try (FileOutputStream fileOut = new FileOutputStream(nomeArquivo);
             ObjectOutputStream out = new ObjectOutputStream(fileOut)) {
            out.writeObject(listaPessoas);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    @SuppressWarnings("unchecked")
    public void recuperar(String nomeArquivo) throws IOException,
ClassNotFoundException {
        try (FileInputStream fileIn = new FileInputStream(nomeArquivo);
```

```
            ObjectInputStream in = new ObjectInputStream(fileIn)) {
            listaPessoas = (ArrayList<PessoaFisica>) in.readObject();
        }
    }



}
```

Pessoa Jurídica Repo

```
package model;

import java.io.*;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

public class PessoaJuridicaRepo implements Serializable {
    private static final long serialVersionUID = 1L;
    private List<PessoaJuridica> listaPessoas;

    // Construtor
    public PessoaJuridicaRepo() {
        this.listaPessoas = new ArrayList<>();
    }

    // Método para inserir uma nova PessoaJuridica
    public void inserir(PessoaJuridica pessoa) {
        listaPessoas.add(pessoa);
    }

    // Método para alterar uma PessoaJuridica existente
    public boolean alterar(PessoaJuridica pessoa) {
        Optional<PessoaJuridica> pessoaExistente = listaPessoas.stream()
                .filter(p -> p.getId() == pessoa.getId())
                .findFirst();

        if (pessoaExistente.isPresent()) {
            int index = listaPessoas.indexOf(pessoaExistente.get());
            listaPessoas.set(index, pessoa);
            return true;
        }
```

```
        return false;
    }

    // Método para excluir uma PessoaJuridica pelo ID
    public void excluir(int id) {
        listaPessoas.removeIf(p -> p.getId() == id);
    }

    // Método para obter uma PessoaJuridica pelo ID
    public PessoaJuridica obter(int id) {
        return listaPessoas.stream()
                .filter(p -> p.getId() == id)
                .findFirst()
                .orElse(null);
    }

    // Método para obter todas as PessoasJuridicas
    public List<PessoaJuridica> obterTodos() {
        return new ArrayList<>(listaPessoas);
    }
    // Método para persistir os dados no disco
    public void persistir(String nomeArquivo) {
        try (FileOutputStream fileOut = new FileOutputStream(nomeArquivo);
             ObjectOutputStream out = new ObjectOutputStream(fileOut)) {
            out.writeObject(listaPessoas);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    // Método para recuperar os dados do disco
    @SuppressWarnings("unchecked")
    public void recuperar(String nomeArquivo) {
        try (FileInputStream fileIn = new FileInputStream(nomeArquivo);
             ObjectInputStream in = new ObjectInputStream(fileIn)) {
            listaPessoas = (ArrayList<PessoaJuridica>) in.readObject();
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

Main

```
import java.io.IOException;
import java.util.Scanner;

public class Main {
```

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
    PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();

    while (true) {
        System.out.println("================================");
        System.out.println("1 - Incluir Pessoa");
        System.out.println("2 - Alterar Pessoa");
        System.out.println("3 - Excluir Pessoa");
        System.out.println("4 - Buscar pelo Id");
        System.out.println("5 - Exibir Todos");
        System.out.println("6 - Persistir Dados");
        System.out.println("7 - Recuperar Dados");
        System.out.println("0 - Finalizar Programa");
        System.out.println("================================");
        int opcao = scanner.nextInt();
        scanner.nextLine(); // Consumir a nova linha

        if (opcao == 0) {
            break;
        }

        switch (opcao) {
            case 1:
                System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
                char tipo = scanner.nextLine().charAt(0);
                if (tipo == 'F') {
                    System.out.println("Digite o id da pessoa:");
                    int id = scanner.nextInt();
                    scanner.nextLine(); // Consumir a nova linha
                    System.out.println("Insira os dados...");
                    System.out.print("Nome: ");
                    String nome = scanner.nextLine();
                    System.out.print("CPF: ");
                    String cpf = scanner.nextLine();
                    System.out.print("Idade: ");
                    int idade = scanner.nextInt();
                    scanner.nextLine(); // Consumir a nova linha

                    PessoaFisica pf = new PessoaFisica(id, nome, cpf, idade);
                    repoFisica.inserir(pf);
                } else if (tipo == 'J') {
                    System.out.println("Digite o id da pessoa:");
                    int id = scanner.nextInt();
                    scanner.nextLine(); // Consumir a nova linha
                    System.out.println("Insira os dados...");
```

```
            System.out.print("Nome: ");
            String nome = scanner.nextLine();
            System.out.print("CNPJ: ");
            String cnpj = scanner.nextLine();

            PessoaJuridica pj = new PessoaJuridica(id, nome, cnpj);
            repoJuridica.inserir(pj);
        }
        break;

    case 2:
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        tipo = scanner.nextLine().charAt(0);
        if (tipo == 'F') {
            System.out.print("Digite o id da pessoa fisica: ");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consumir a nova linha
            PessoaFisica pf = repoFisica.obter(id);
            if (pf != null) {
                System.out.println("Dados atuais:");
                pf.exibir();
                System.out.println("Insira os novos dados...");
                System.out.print("Nome: ");
                pf.setNome(scanner.nextLine());
                System.out.print("CPF: ");
                pf.setCpf(scanner.nextLine());
                System.out.print("Idade: ");
                pf.setIdade(scanner.nextInt());
                scanner.nextLine(); // Consumir a nova linha
                repoFisica.alterar(pf);
            } else {
                System.out.println("Pessoa Fisica não encontrada.");
            }
        } else if (tipo == 'J') {
            System.out.print("Digite o id da pessoa juridica: ");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consumir a nova linha
            PessoaJuridica pj = repoJuridica.obter(id);
            if (pj != null) {
                System.out.println("Dados atuais:");
                pj.exibir();
                System.out.println("Insira os novos dados...");
                System.out.print("Nome: ");
                pj.setNome(scanner.nextLine());
                System.out.print("CNPJ: ");
                pj.setCnpj(scanner.nextLine());
                repoJuridica.alterar(pj);
```

```
            } else {
                System.out.println("Pessoa Juridica não encontrada.");
            }
        }
        break;

    case 3:
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        tipo = scanner.nextLine().charAt(0);
        if (tipo == 'F') {
            System.out.print("Digite o id da pessoa fisica: ");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consumir a nova linha
            repoFisica.excluir(id);
        } else if (tipo == 'J') {
            System.out.print("Digite o id da pessoa juridica: ");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consumir a nova linha
            repoJuridica.excluir(id);
        }
        break;

    case 4:
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        tipo = scanner.nextLine().charAt(0);
        if (tipo == 'F') {
            System.out.print("Digite o id da pessoa fisica: ");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consumir a nova linha
            PessoaFisica pf = repoFisica.obter(id);
            if (pf != null) {
                pf.exibir();
            } else {
                System.out.println("Pessoa Fisica não encontrada.");
            }
        } else if (tipo == 'J') {
            System.out.print("Digite o id da pessoa juridica: ");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consumir a nova linha
            PessoaJuridica pj = repoJuridica.obter(id);
            if (pj != null) {
                pj.exibir();
            } else {
                System.out.println("Pessoa Juridica não encontrada.");
            }
        }
        break;
```

```
            case 5:
                System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
                tipo = scanner.nextLine().charAt(0);
                if (tipo == 'F') {
                    for (PessoaFisica pf : repoFisica.obterTodos()) {
                        pf.exibir();
                    }
                } else if (tipo == 'J') {
                    for (PessoaJuridica pj : repoJuridica.obterTodos()) {
                        pj.exibir();
                    }
                }
                break;

            case 6:
                System.out.print("Digite o prefixo do arquivo: ");
                String prefixo = scanner.nextLine();
                repoFisica.persistir(prefixo + ".fisica.bin");
                repoJuridica.persistir(prefixo + ".juridica.bin");
                break;

            case 7:
                System.out.print("Digite o prefixo do arquivo: ");
                prefixo = scanner.nextLine();
                try {
                    repoFisica.recuperar(prefixo + ".fisica.bin");
                    repoJuridica.recuperar(prefixo + ".juridica.bin");
                } catch (IOException | ClassNotFoundException e) {
                    System.out.println("Erro ao recuperar dados: " + e.getMessage());
                }
                break;

            default:
                System.out.println("Opção inválida.");
                break;
        }
    }

    scanner.close();
  }
}
```

Resultado da execução dos códigos.

"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Educational Edition

```
2022.2.2\lib\idea_rt.jar=64141:C:\Program Files\JetBrains\IntelliJ IDEA Educational Edition 2022.2.2\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath "C:\Programas Vanessa Estudo\CadastroPOO2\out\production\CadastroPOO2" Main
================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
================================
1
F - Pessoa Fisica | J - Pessoa Juridica
F
Digite o id da pessoa:
6
Insira os dados...
Nome: Vanessa
CPF: 000000000000
Idade: 37
================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
================================
```

Análise e conclusão:

O que são elementos estáticos e qual o motivo para o método main adotar esse modificador?

Por definição, um atributo ou método estático é aquele que não requer uma instância para ser utilizado. Isso quer dizer que um método estático, como no seu exemplo, pode ser executado livremente sem a necessidade de instanciação de um objeto

Em resumo, elementos estáticos em Java pertencem á classe e não a uma instância específica, o que permite acessá-los diretamente pela classe. O método é estático para fornecer um ponto de entrada claro e simples para a execução do programa, permitindo que a JVM inicie a

execução do programa, permitindo que a JMV inicie a execução sem a necessidade de criar instâncias da classe.

Para que serve a classe Scanner?

A classe Scanner faz parte do pacote java. util e fornece métodos para analisar dados de entrada em diferentes tipos primitivos, como inteiros, números de ponto flutuante, strings e muito mais.

Como o uso de classes de repositório impactou na organização do código?

O uso de classes de repositório melhora a organização do código ao promover a separação de responsabilidades, facilitando a manutenção, os testes e a leitura do código, além de promover o reuso de operações de dados.