RELATÓRIO CADASTRO POO

UNIVERCIDADE: ESTÁCIO DE SÁ

CAMPUS: POLO CID UNUVERCITARIA - JUAZEIRO DO NORTE-CE

CURSO: DESENVOLVIMNTE FULL STACK

DISCIPLINA: RPG0014 - INICIANDO OS CAMINHOS PELO JAVA

TURMA: 9001 SEMESTRE LETIVO: 2024.4

NOME DO INTEGRANTE: EDNALDO RIBEIRO DE OLIVEIRA

REPOSITIO GIT: https://github.com/ednaldo1901/trabalho-mundo3-nivel-1

CLASSE PESSOA

```
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);
    }
}
```

PessoaFisica.java

```
public 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);
        System.out.println("Idade: " + idade);
    }
}
```

PessoaJuridica.java

```
package model;

public 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);
    }
}
```

PessoaFisicaRepo.java

```
package model;

import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class PessoaFisicaRepo {
    private List<PessoaFisica> pessoas = new ArrayList<>();

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

    public void alterar(PessoaFisica pessoa) {
        PessoaFisica existente = obter(pessoa.getId());
        if (existente != null) {
            existente.setNome(pessoa.getNome());
            existente.setCpf(pessoa.getCpf());
            existente.setIdade(pessoa.getIdade());
        }
    }

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

    public PessoaFisica obter(int id) {
        return pessoas.stream().filter(p -> p.getId() == id).findFirst().orElse(null);
    }
```

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

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

    public void recuperar(String arquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(arquivo))) {
            pessoas = (List<PessoaFisica>) ois.readObject();
        } catch (FileNotFoundException e) {
            pessoas = new ArrayList<>();
        }
    }
}
```

PessoaJuridicaRepo.java

```
package model;

import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class PessoaJuridicaRepo {
    private List<PessoaJuridica> pessoas = new ArrayList<>();

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

    public void alterar(PessoaJuridica pessoa) {
        PessoaJuridica existente = obter(pessoa.getId());
        if (existente != null) {
            existente.setNome(pessoa.getNome());
            existente.setCnpj(pessoa.getCnpj());
        }
    }

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

    public PessoaJuridica obter(int id) {
        return pessoas.stream().filter(p -> p.getId() == id).findFirst().orElse(null);
    }

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

```
    public void persistir(String arquivo) throws IOException {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(arquivo))) {
            oos.writeObject(pessoas);
        }
    }

    public void recuperar(String arquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(arquivo))) {
            pessoas = (List<PessoaJuridica>) ois.readObject();
        } catch (FileNotFoundException e) {
            pessoas = new ArrayList<>();
        }
    }
}
```

CadastroPOO.java

```
package model;

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

public class MenuPricipal {
    private static final PessoaFisicaRepo pessoaFisicaRepo = new PessoaFisicaRepo();
    private static final PessoaJuridicaRepo pessoaJuridicaRepo = new PessoaJuridicaRepo();
    private static final Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        int opcao;

        do {
            exibirMenu();
            opcao = scanner.nextInt();
            scanner.nextLine(); // Consumir a quebra de linha

            switch (opcao) {
                case 1 -> gerenciarPessoaFisica();
                case 2 -> gerenciarPessoaJuridica();
                case 0 -> System.out.println("Saindo do programa...");
                default -> System.out.println("Opção inválida!");
            }
        } while (opcao != 0);
    }

    private static void exibirMenu() {
        System.out.println("=== MENU PRINCIPAL ===");
        System.out.println("1. Gerenciar Pessoa Física");
        System.out.println("2. Gerenciar Pessoa Jurídica");
        System.out.println("0. Sair");
        System.out.print("Escolha uma opção: ");
    }
```

```
private static void gerenciarPessoaFisica() {
    int opcao;
    do {
        System.out.println("--- Gerenciar Pessoa Física ---");
        System.out.println("1. Inserir");
        System.out.println("2. Alterar");
        System.out.println("3. Excluir");
        System.out.println("4. Listar todas");
        System.out.println("5. Salvar em arquivo");
        System.out.println("6. Recuperar de arquivo");
        System.out.println("0. Voltar");
        System.out.print("Escolha uma opção: ");
        opcao = scanner.nextInt();
        scanner.nextLine();

        switch (opcao) {
            case 1 -> inserirPessoaFisica();
            case 2 -> alterarPessoaFisica();
            case 3 -> excluirPessoaFisica();
            case 4 -> listarPessoaFisica();
            case 5 -> salvarPessoaFisica();
            case 6 -> recuperarPessoaFisica();
            case 0 -> System.out.println("Voltando ao menu principal...");
            default -> System.out.println("Opção inválida!");
        }
    } while (opcao != 0);
}

private static void gerenciarPessoaJuridica() {
    int opcao;
    do {
        System.out.println("--- Gerenciar Pessoa Jurídica ---");
        System.out.println("1. Inserir");
        System.out.println("2. Alterar");
        System.out.println("3. Excluir");
        System.out.println("4. Listar todas");
        System.out.println("5. Salvar em arquivo");
        System.out.println("6. Recuperar de arquivo");
        System.out.println("0. Voltar");
        System.out.print("Escolha uma opção: ");
        opcao = scanner.nextInt();
        scanner.nextLine();

        switch (opcao) {
            case 1 -> inserirPessoaJuridica();
            case 2 -> alterarPessoaJuridica();
            case 3 -> excluirPessoaJuridica();
            case 4 -> listarPessoaJuridica();
            case 5 -> salvarPessoaJuridica();
            case 6 -> recuperarPessoaJuridica();
            case 0 -> System.out.println("Voltando ao menu principal...");
            default -> System.out.println("Opção inválida!");
        }
    } while (opcao != 0);
```

```
}

private static void inserirPessoaFisica() {
    System.out.print("ID: ");
    int id = scanner.nextInt();
    scanner.nextLine();
    System.out.print("Nome: ");
    String nome = scanner.nextLine();
    System.out.print("CPF: ");
    String cpf = scanner.nextLine();
    System.out.print("Idade: ");
    int idade = scanner.nextInt();

    PessoaFisica pessoa = new PessoaFisica(id, nome, cpf, idade);
    pessoaFisicaRepo.inserir(pessoa);
}

private static void alterarPessoaFisica() {
    System.out.print("ID da Pessoa Física a alterar: ");
    int id = scanner.nextInt();
    scanner.nextLine();
    System.out.print("Novo Nome: ");
    String nome = scanner.nextLine();
    System.out.print("Novo CPF: ");
    String cpf = scanner.nextLine();
    System.out.print("Nova Idade: ");
    int idade = scanner.nextInt();

    PessoaFisica pessoa = new PessoaFisica(id, nome, cpf, idade);
    pessoaFisicaRepo.alterar(pessoa);
}

private static void excluirPessoaFisica() {
    System.out.print("ID da Pessoa Física a excluir: ");
    int id = scanner.nextInt();
    pessoaFisicaRepo.excluir(id);
}

private static void listarPessoaFisica() {
    System.out.println("--- Lista de Pessoas Físicas ---");
    pessoaFisicaRepo.obterTodos().forEach(PessoaFisica::exibir);
}

private static void salvarPessoaFisica() {
    System.out.print("Nome do arquivo para salvar: ");
    String arquivo = scanner.nextLine();
    try {
        pessoaFisicaRepo.persistir(arquivo);
    } catch (IOException e) {
        System.out.println("Erro ao salvar o arquivo: " + e.getMessage());
    }
}

private static void recuperarPessoaFisica() {
```

```
        System.out.print("Nome do arquivo para recuperar: ");
        String arquivo = scanner.nextLine();
        try {
            pessoaFisicaRepo.recuperar(arquivo);
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao recuperar o arquivo: " + e.getMessage());
        }
    }

    private static void inserirPessoaJuridica() {
        System.out.print("ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Nome: ");
        String nome = scanner.nextLine();
        System.out.print("CNPJ: ");
        String cnpj = scanner.nextLine();

        PessoaJuridica pessoa = new PessoaJuridica(id, nome, cnpj);
        pessoaJuridicaRepo.inserir(pessoa);
    }

    private static void alterarPessoaJuridica() {
        System.out.print("ID da Pessoa Jurídica a alterar: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Novo Nome: ");
        String nome = scanner.nextLine();
        System.out.print("Novo CNPJ: ");
        String cnpj = scanner.nextLine();

        PessoaJuridica pessoa = new PessoaJuridica(id, nome, cnpj);
        pessoaJuridicaRepo.alterar(pessoa);
    }

    private static void excluirPessoaJuridica() {
        System.out.print("ID da Pessoa Jurídica a excluir: ");
        int id = scanner.nextInt();
        pessoaJuridicaRepo.excluir(id);
    }

    private static void listarPessoaJuridica() {
        System.out.println("--- Lista de Pessoas Jurídicas ---");
        pessoaJuridicaRepo.obterTodos().forEach(PessoaJuridica::exibir);
    }

    private static void salvarPessoaJuridica() {
        System.out.print("Nome do arquivo para salvar: ");
        String arquivo = scanner.nextLine();
        try {
            pessoaJuridicaRepo.persistir(arquivo);
        } catch (IOException e) {
            System.out.println("Erro ao salvar o arquivo: " + e.getMessage());
        }
```

```
    }

    private static void recuperarPessoaJuridica() {
        System.out.print("Nome do arquivo para recuperar: ");
        String arquivo = scanner.nextLine();
        try {
            pessoaJuridicaRepo.recuperar(arquivo);
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao recuperar o arquivo: " + e.getMessage());
        }
    }
}
```

ANALISE E CONCLUSAO

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

Elementos estáticos: Elementos marcados com o modificador static pertencem à classe e não a uma instância específica dela. Isso significa que podem ser acessados sem a necessidade de criar um objeto da classe.

Motivo do método main ser estático: O método main é o ponto de entrada de execução de um programa Java. Ele é declarado como estático porque a JVM (Java Virtual Machine) precisa chamá-lo sem instanciar a classe. Isso permite que o programa inicie sua execução sem depender de objetos específicos. Além disso, elementos estáticos como os repositórios (pessoaFisicaRepo e pessoaJuridicaRepo) e o Scanner são usados dentro do método main e em outros métodos estáticos. Essa abordagem evita a necessidade de criar instâncias para acessar tais recursos.

Para que serve a classe Scanner?

A classe Scanner é usada para ler entradas do usuário a partir de diferentes fontes como o console (System.in), arquivos, ou strings.No código fornecido, o Scanner lê dados inseridos pelo usuário no console. Ele é utilizado para capturar diferentes tipos de dados (como int, String) durante a interação com o programa, permitindo:
Seleção de opções no menu (scanner.nextInt()).
Captura de informações como nome, CPF, e ID ao inserir ou alterar registros (scanner.nextLine()).

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

Impacto positivo:

Separação de responsabilidades: A lógica de gerenciamento e armazenamento de dados de Pessoa
Física e Jurídica foi encapsulada em classes repositórios (PessoaFisicaRepo e PessoaJuridicaRepo),
mantendo o código do menu mais limpo e focado apenas em gerenciar o fluxo de interação do usuário.
Reutilização de código: O repositório centraliza operações como inserir, alterar, excluir, listar, salvar e recuperar dados. Isso evita duplicação e facilita a manutenção.

Testabilidade: A separação da lógica de persistência em repositórios torna mais fácil realizar testes isolados, garantindo maior confiabilidade ao sistema.

Impacto na organização:

O código segue o princípio SOLID de separação de responsabilidades, onde cada classe tem uma única
responsabilidade: o menu gerencia a interface com o usuário e os repositórios cuidam das operações com os dados.

Facilita futuras expansões:

novos métodos ou funcionalidades relacionadas ao gerenciamento de dados podem ser adicionados diretamente
às classes de repositório, sem impactar o restante do programa.

Conclusão:

O método main é estático para permitir que a JVM o execute sem instanciar a classe. A classe Scanner
facilita a interação com o usuário, capturando entradas no console. A adoção de repositórios no código
promove uma organização modular, separando responsabilidades e tornando o código mais limpo, reutilizável,
e fácil de manter.

RESULTADO DA EXECUSÃO

```
run:

=== MENU PRINCIPAL ===
1. Gerenciar Pessoa F�sica
2. Gerenciar Pessoa Jur�dica
0. Sair
Escolha uma op��o: 1

--- Gerenciar Pessoa F�sica ---
1. Inserir
2. Alterar
3. Excluir
4. Listar todas
5. Salvar em arquivo
6. Recuperar de arquivo
0. Voltar
Escolha uma op��o: 1
ID: 12
Nome: ednaldo Ribeiro de Oliveira
CPF: 0128736981
Idade: 38

--- Gerenciar Pessoa F�sica ---
1. Inserir
2. Alterar
3. Excluir
4. Listar todas
5. Salvar em arquivo
6. Recuperar de arquivo
0. Voltar
Escolha uma op��o:
```