POLO SETOR O – BRASÍLIA – DF
DESENVOLVIMENTO FULL STACK
NIVEL 1: INICIANDO O CAMINHO PELO JAVA
2024.2 FULL STACK
MAYCON MOURA

Título da Prática:

Criação do Cadastro em Modo Texto

Objetivo da Prática:

Utilização de conceitos básicos de programação em Java, incluindo elementos estáticos, métodos principais, a classe Scanner para entrada de dados e o impacto das classes de repositório na organização do código.

Códigos

CadastroPOO

```
package cadastropoo;

import java.io.*;
import java.util.*;
import model.Pessoa;

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

public class CadastroPOO {

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

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

        do {
```

```
        exibirMenu();
        opcao = scanner.nextInt();
        scanner.nextLine(); // Limpar o buffer do scanner

        switch (opcao) {
            case 1:
                incluirOpcao();
                break;
            case 2:
                alterarOpcao();
                break;
            case 3:
                excluirOpcao();
                break;
            case 4:
                exibirPorIdOpcao();
                break;
            case 5:
                exibirTodosOpcao();
                break;
            case 6:
                salvarDadosOpcao();
                break;
            case 7:
                recuperarDadosOpcao();
                break;
            case 0:
                System.out.println("Encerrando o programa...");
                break;
            default:
                System.out.println("Opcao invalida! Digite novamente.");
        }

        System.out.println();

    } while (opcao != 0);

    scanner.close();
}

private static void exibirMenu() {
    System.out.println("### Menu ###");
    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. Sair");
    System.out.print("Escolha uma opcao: ");
}

private static void incluirOpcao() {
    System.out.println("### Incluir ###");
```

```
        System.out.println("Escolha o tipo:");
        System.out.println("F. Pessoa Fisica");
        System.out.println("J. Pessoa Juridica");
        char tipo = scanner.next().charAt(0);
        scanner.nextLine(); // Limpar o buffer do scanner

        switch (Character.toUpperCase(tipo)) {
            case 'F':
                incluirPessoaFisica();
                break;
            case 'J':
                incluirPessoaJuridica();
                break;
            default:
                System.out.println("Opcao invalida!");
        }
    }

    private static void incluirPessoaFisica() {
        System.out.print("Digite o ID da Pessoa Fisica: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Digite o nome da Pessoa Fisica: ");
        String nome = scanner.nextLine();
        System.out.print("Digite o CPF da Pessoa Fisica: ");
        String cpf = scanner.nextLine();
        System.out.print("Digite a idade da Pessoa Fisica: ");
        int idade = scanner.nextInt();
        scanner.nextLine();

        PessoaFisica pessoaFisica = new PessoaFisica(id, nome, cpf, idade);
        pessoaFisicaRepo.inserir(pessoaFisica);
        System.out.println("Pessoa Fisica incluida com sucesso!");
    }

    private static void incluirPessoaJuridica() {
        System.out.print("Digite o ID da Pessoa Juridica: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Digite o nome da Pessoa Juridica: ");
        String nome = scanner.nextLine();
        System.out.print("Digite o CNPJ da Pessoa Juridica: ");
        String cnpj = scanner.nextLine();

        PessoaJuridica pessoaJuridica = new PessoaJuridica(id, nome, cnpj);
        pessoaJuridicaRepo.inserir(pessoaJuridica);
        System.out.println("Pessoa Juridica incluida com sucesso!");
    }

    private static void alterarOpcao() {
        System.out.println("### Alterar ###");
        System.out.println("Escolha o tipo:");
        System.out.println("1. Pessoa Fisica");
        System.out.println("2. Pessoa Juridica");
        int tipo = scanner.nextInt();
```

```
        scanner.nextLine();

        switch (tipo) {
            case 1:
                alterarPessoaFisica();
                break;
            case 2:
                alterarPessoaJuridica();
                break;
            default:
                System.out.println("Opcao invalida!");
                break;
        }
    }

    private static void alterarPessoaFisica() {
        System.out.print("Digite o ID da Pessoa Fisica para alterar: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        PessoaFisica pessoaExistente = pessoaFisicaRepo.obter(id);
        if (pessoaExistente != null) {
            System.out.println("Dados atuais da Pessoa Fisica:");
            exibirPessoa(pessoaExistente);

            System.out.print("Digite o novo nome da Pessoa Fisica: ");
            String nome = scanner.nextLine();
            System.out.print("Digite o novo CPF da Pessoa Fisica: ");
            String cpf = scanner.nextLine();
            System.out.print("Digite a nova idade da Pessoa Fisica: ");
            int idade = scanner.nextInt();
            scanner.nextLine();

            PessoaFisica pessoaAtualizada = new PessoaFisica(id, nome, cpf, idade);
            pessoaFisicaRepo.alterar(pessoaAtualizada);
            System.out.println("Pessoa Fisica alterada com sucesso!");
        } else {
            System.out.println("Pessoa Fisica nao encontrada para o ID informado.");
        }
    }

    private static void alterarPessoaJuridica() {
        System.out.print("Digite o ID da Pessoa Juridica para alterar: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        PessoaJuridica pessoaExistente = pessoaJuridicaRepo.obter(id);
        if (pessoaExistente != null) {
            System.out.println("Dados atuais da Pessoa Juridica:");
            exibirPessoa(pessoaExistente);

            System.out.print("Digite o novo nome da Pessoa Juridica: ");
            String nome = scanner.nextLine();
            System.out.print("Digite o novo CNPJ da Pessoa Juridica: ");
            String cnpj = scanner.nextLine();
```

```
            PessoaJuridica pessoaAtualizada = new PessoaJuridica(id, nome, cnpj);
            pessoaJuridicaRepo.alterar(pessoaAtualizada);
            System.out.println("Pessoa Juridica alterada com sucesso!");
        } else {
            System.out.println("Pessoa Juridica nao encontrada para o ID informado.");
        }
    }

    private static void excluirOpcao() {
        System.out.println("### Excluir ###");
        System.out.println("Escolha o tipo:");
        System.out.println("1. Pessoa Fisica");
        System.out.println("2. Pessoa Juridica");
        int tipo = scanner.nextInt();
        scanner.nextLine();

        System.out.print("Digite o ID da entidade para excluir: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        switch (tipo) {
            case 1:
                pessoaFisicaRepo.excluir(id);
                System.out.println("Pessoa Fisica excluida com sucesso!");
                break;
            case 2:
                pessoaJuridicaRepo.excluir(id);
                System.out.println("Pessoa Juridica excluida com sucesso!");
                break;
            default:
                System.out.println("Opcao invalida!");
                break;
        }
    }

    private static void exibirPorIdOpcao() {
        System.out.println("### Exibir por ID ###");
        System.out.println("Escolha o tipo:");
        System.out.println("1. Pessoa Fisica");
        System.out.println("2. Pessoa Juridica");
        int tipo = scanner.nextInt();
        scanner.nextLine();

        System.out.print("Digite o ID da entidade para exibir: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        switch (tipo) {
            case 1:
                {
                    PessoaFisica pessoa = pessoaFisicaRepo.obter(id);
                    if (pessoa != null) {
                        System.out.println("Dados da Pessoa Fisica:");
                        exibirPessoa(pessoa);
```

```
            } else {
                System.out.println("Pessoa Fisica nao encontrada para o ID informado.");
            }       break;
        }
    case 2:
        {
            PessoaJuridica pessoa = pessoaJuridicaRepo.obter(id);
            if (pessoa != null) {
                System.out.println("Dados da Pessoa Juridica:");
                exibirPessoa(pessoa);
            } else {
                System.out.println("Pessoa Juridica nao encontrada para o ID informado.");
            }       break;
        }
    default:
        System.out.println("Opcao invalida!");
        break;
  }
}

private static void exibirTodosOpcao() {
    System.out.println("### Exibir todos ###");
    System.out.println("Escolha o tipo:");
    System.out.println("1. Pessoa Fisica");
    System.out.println("2. Pessoa Juridica");
    int tipo = scanner.nextInt();
    scanner.nextLine();

    switch (tipo) {
        case 1:
            {
                List<PessoaFisica> pessoas = pessoaFisicaRepo.obterTodos();
                System.out.println("Lista de Pessoas Fisicas:");
                for (PessoaFisica pessoa : pessoas) {
                    exibirPessoa(pessoa);
                }       break;
            }
        case 2:
            {
                List<PessoaJuridica> pessoas = pessoaJuridicaRepo.obterTodos();
                System.out.println("Lista de Pessoas Juridicas:");
                for (PessoaJuridica pessoa : pessoas) {
                    exibirPessoa(pessoa);
                }       break;
            }
        default:
            System.out.println("Opcao invalida!");
            break;
    }
}

private static void salvarDadosOpcao() {
    System.out.println("### Salvar dados ###");
    System.out.print("Digite o prefixo dos arquivos: ");
    String prefixo = scanner.nextLine();
```

```
        try {
            pessoaFisicaRepo.persistir(prefixo + ".fisica.bin");
            pessoaJuridicaRepo.persistir(prefixo + ".juridica.bin");
            System.out.println("Dados salvos com sucesso nos arquivos " + prefixo + ".fisica.bin
e " + prefixo + ".juridica.bin");
        } catch (IOException e) {
            System.out.println("Erro ao salvar os dados: " + e.getMessage());
        }
    }

    private static void recuperarDadosOpcao() {
        System.out.println("### Recuperar dados ###");
        System.out.print("Digite o prefixo dos arquivos: ");
        String prefixo = scanner.nextLine();

        try {
            pessoaFisicaRepo.recuperar(prefixo + ".fisica.bin");
            pessoaJuridicaRepo.recuperar(prefixo + ".juridica.bin");
            System.out.println("Dados recuperados com sucesso dos arquivos " + prefixo +
".fisica.bin e " + prefixo + ".juridica.bin");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao recuperar os dados: " + e.getMessage());
        }
    }

    private static void exibirPessoa(Pessoa pessoa) {
        System.out.println("ID: " + pessoa.getId());
        System.out.println("Nome: " + pessoa.getNome());
        if (pessoa instanceof PessoaFisica pf) {
            System.out.println("CPF: " + pf.getCpf());
            System.out.println("Idade: " + pf.getIdade());
        } else if (pessoa instanceof PessoaJuridica pj) {
            System.out.println("CNPJ: " + pj.getCnpj());
        }
        System.out.println("--------------------");
    }
}
```

Pessoa

```
package 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);
    }
}
```

PessoaFisica

```
package model;

import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {
    private String cpf;
    private int idade;

    public PessoaFisica() {
        super();
    }

    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

```
package model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {
    private String cnpj;

    public PessoaJuridica() {
        super();
    }

    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

```
package model;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.List;

public class PessoaFisicaRepo {

    private List<PessoaFisica> listaPessoasFisicas;

    public PessoaFisicaRepo() {
        this.listaPessoasFisicas = new ArrayList<>();
    }

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

    public void alterar(PessoaFisica pessoa) {
        for (int i = 0; i < listaPessoasFisicas.size(); i++) {
            if (listaPessoasFisicas.get(i).getId() == pessoa.getId()) {
                listaPessoasFisicas.set(i, pessoa);
                return;
            }
        }
        throw new IllegalArgumentException("Pessoa não encontrada para alteração");
    }

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

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

```
        }
        return null;
    }

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

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

    @SuppressWarnings("unchecked")
    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException
{
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
            listaPessoasFisicas = (List<PessoaFisica>) ois.readObject();
        }
    }
}
```

PessoaJuridicaRepo

```
package model;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.List;

public class PessoaJuridicaRepo {
    private List<PessoaJuridica> listaPessoasJuridicas;

    public PessoaJuridicaRepo() {
        this.listaPessoasJuridicas = new ArrayList<>();
    }

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

```
    public void alterar(PessoaJuridica pessoa) {
        for (int i = 0; i < listaPessoasJuridicas.size(); i++) {
            if (listaPessoasJuridicas.get(i).getId() == pessoa.getId()) {
                listaPessoasJuridicas.set(i, pessoa);
                return;
            }
        }
        throw new IllegalArgumentException("Pessoa não encontrada para alteração");
    }

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

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

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

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

    @SuppressWarnings("unchecked")
    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException
{
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
            listaPessoasJuridicas = (List<PessoaJuridica>) ois.readObject();
        }
    }
}
```

Resultado da Execução

```
### Menu ###
1. Incluir
```

2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Sair
Escolha uma opcao: 1
Incluir
Escolha o tipo:
F. Pessoa Fisica
J. Pessoa Juridica
F
Digite o ID da Pessoa Fisica: 1
Digite o nome da Pessoa Fisica: Maria
Digite o CPF da Pessoa Fisica: 8888
Digite a idade da Pessoa Fisica: 87
Pessoa Fisica incluida com sucesso!

Menu
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Sair
Escolha uma opcao: 6
Salvar dados
Digite o prefixo dos arquivos: teste
Dados salvos com sucesso nos arquivos teste.fisica.bin e teste.juridica.bin

Menu
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Sair
Escolha uma opcao:

Análise e Conclusão:

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

Elementos estáticos em Java são aqueles que pertencem à classe em si, e não a instâncias individuais da classe. Isso significa que podem ser acessados diretamente pela classe, sem precisar criar objetos. O método `main` adota o modificador `static` porque ele precisa ser executado pela JVM sem a necessidade de criar uma instância da classe. Como ponto de entrada do programa, ele deve estar disponível para execução imediata quando a classe é carregada.

Para que serve a classe Scanner?

A classe `Scanner` em Java é usada para ler a entrada de dados do usuário. Ela permite que o programa aceite vários tipos de dados, como strings, inteiros e floats, digitados pelo usuário através do console. Isso é fundamental para tornar os programas interativos, permitindo que os usuários forneçam informações necessárias durante a execução do programa.

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

O uso de classes de repositório melhorou significativamente a organização do código ao separar a lógica de persistência de dados da lógica de negócios. Com as classes de repositório, todas as operações relacionadas ao armazenamento, recuperação e gerenciamento de dados são encapsuladas em uma camada distinta. Isso torna o código mais modular, fácil de manter e de entender.