



Centro Universitário Estácio

Curso: Desenvolvimento Full Stack

Disciplina: Iniciando o Caminho Pelo Java

Semestre: 3º

Aluno: Jonnatha Walben Saldanha da Silva

Criação do Cadastro em Modo Texto

O objetivo da prática com a criação do cadastro em modo texto é desenvolver um sistema simples de gerenciamento de dados de pessoas físicas e jurídicas utilizando Java. Este sistema permitirá que o usuário realize operações básicas, como incluir, alterar, excluir e visualizar informações de pessoas físicas e jurídicas por meio de uma interface de texto no console.

Os principais objetivos específicos dessa prática incluem:

- **Entendimento dos Conceitos Básicos:** Compreender os conceitos fundamentais de programação em Java, incluindo classes, métodos, entrada e saída de dados.
- **Manipulação de Dados:** Implementar lógicas para inserir, alterar, excluir e visualizar dados de pessoas físicas e jurídicas.
- **Uso de Classes e Objetos:** Utilizar classes e objetos para representar entidades do mundo real, como pessoas físicas e jurídicas, e manipular seus dados de forma eficiente.
- **Organização do Código:** Estruturar o código de forma organizada e modular, separando responsabilidades em diferentes classes e métodos para facilitar a manutenção e compreensão do código.
- **Persistência de Dados:** Implementar a persistência de dados por meio da leitura e gravação em arquivos, permitindo que as informações sejam armazenadas entre diferentes execuções do programa.
- **Interação com o Usuário:** Interagir com o usuário por meio de mensagens de texto no console, fornecendo instruções claras e solicitando entrada de dados de forma amigável.

Códigos Solicitados:

Implementação das entidades

- **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 void exibir() {
```

```
        System.out.println("ID: " + id);
```

```
        System.out.println("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;
    }
}

```

- **PessoaFisica.**

package model;

```

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;
}
```

@Override

```
public void exibir() {

    super.exibir();

    System.out.println("CPF: " + cpf);

    System.out.println("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;  
}  
}
```

- **PessoaJuridica**

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;  
    }  
}
```

@Override

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

```
}
```

```
// Getters e setters
```

```
public String getCnpj() {
```

```
    return cnpj;
```

```
}
```

```
public void setCnpj(String cnpj) {
```

```
    this.cnpj = cnpj;
```

```
}
```

```
}
```

Implementação dos gerenciadores:

- **PessoaFisicaRepo**

```
package model;
```

```
import java.io.*;
```

```
import java.util.ArrayList;
```

```
import java.util.Iterator;
```

```
import java.util.List;
```

```
public class PessoaFisicaRepo {
```

```
private List<PessoaFisica> pessoasFisicas;
```

```
public PessoaFisicaRepo() {
```

```
    this.pessoasFisicas = new ArrayList<>();
```

```
}
```

```
public void inserir(PessoaFisica pessoaFisica) {
```

```
    pessoasFisicas.add(pessoaFisica);
```

```
}
```

```
public void alterar(PessoaFisica pessoaFisica) {
```

```
    for (PessoaFisica pf : pessoasFisicas) {
```

```
        if (pf.getId() == pessoaFisica.getId()) {
```

```
            pf.setNome(pessoaFisica.getNome());
```

```
            pf.setCpf(pessoaFisica.getCpf());
```

```
            pf.setIdade(pessoaFisica.getIdade());
```

```
            return;
```

```
        }
```

```
    }
```

```
    throw new IllegalArgumentException("Pessoa Física com o ID " +  
pessoaFisica.getId() + " não encontrada.");
```

```
}
```

```
public void excluir(int id) {
```

```
Iterator<PessoaFisica> iterator = pessoasFisicas.iterator();

while (iterator.hasNext()) {

    PessoaFisica pf = iterator.next();

    if (pf.getId() == id) {

        iterator.remove();

        return;

    }

}

throw new IllegalArgumentException("Pessoa Física com o ID " + id + " não encontrada.");

}
```

```
public PessoaFisica obter(int id) {

    for (PessoaFisica pf : pessoasFisicas) {

        if (pf.getId() == id) {

            return pf;

        }

    }

    throw new IllegalArgumentException("Pessoa Física com o ID " + id + " não encontrada.");

}
```

```
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(pessoasFisicas);

    }

}

```

```

public void recuperar(String nomeArquivo) throws IOException,
ClassNotFoundException {

    try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {

        pessoasFisicas = (List<PessoaFisica>) inputStream.readObject();

    }

}

}

```

- **PessoaJuridicaRepo**

package model;

import java.io.*;

import java.util.ArrayList;

import java.util.List;

```
public class PessoaJuridicaRepo {

    private List<PessoaJuridica> pessoasJuridicas;

    public PessoaJuridicaRepo() {

        this.pessoasJuridicas = new ArrayList<>();

    }

    public void inserir(PessoaJuridica pessoaJuridica) {

        pessoasJuridicas.add(pessoaJuridica);

    }

    public void alterar(int id, PessoaJuridica pessoaJuridica) {

        for (int i = 0; i < pessoasJuridicas.size(); i++) {

            PessoaJuridica pj = pessoasJuridicas.get(i);

            if (pj.getId() == id) {

                pessoasJuridicas.set(i, pessoaJuridica);

                return;

            }

        }

        throw new IllegalArgumentException("Pessoa Jurídica com o ID " + id + " não encontrada.");

    }

    public void excluir(int id) {
```

```
for (int i = 0; i < pessoasJuridicas.size(); i++) {  
  
    PessoaJuridica pj = pessoasJuridicas.get(i);  
  
    if (pj.getId() == id) {  
  
        pessoasJuridicas.remove(i);  
  
        return;  
  
    }  
  
}  
  
throw new IllegalArgumentException("Pessoa Jurídica com o ID " + id + " não encontrada.");  
  
}
```

```
public PessoaJuridica obter(int id) {  
  
    for (PessoaJuridica pj : pessoasJuridicas) {  
  
        if (pj.getId() == id) {  
  
            return pj;  
  
        }  
  
    }  
  
    throw new IllegalArgumentException("Pessoa Jurídica com o ID " + id + " não encontrada.");  
  
}
```

```
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(pessoasJuridicas);

        }

    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException
{

        try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {

            pessoasJuridicas = (List<PessoaJuridica>) inputStream.readObject();

        }

    }

    public void alterar(PessoaJuridica pessoa) {

        throw new UnsupportedOperationException("Not supported yet.");

    }

}

```

Classe principal

- **CadastroPOO**

```
import model.*;
```

```
import java.io.IOException;

import java.util.InputMismatchException;

import java.util.List;

import java.util.Scanner;


public class CadastroPOO {

    private static final Scanner scanner = new Scanner(System.in);

    private static final PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();

    private static final PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();


    public static void main(String[] args) {

        int opcao;

        do {

            exibirMenu();

            opcao = lerOpcao();

            switch (opcao) {

                case 1 -> incluirPessoa();

                case 2 -> alterarPessoa();

                case 3 -> excluirPessoa();

                case 4 -> exibirPessoaPorId();

                case 5 -> exibirTodasPessoas();

                case 6 -> salvarDados();

                case 7 -> recuperarDados();

                case 0 -> System.out.println("Encerrando o programa...");
```

```

        default -> System.out.println("Opção inválida! Tente novamente.");

    }

} while (opcao != 0);

}

```

```

private static void exibirMenu() {

    System.out.println("===== Menu =====");

    System.out.println("1 - Incluir Pessoa");

    System.out.println("2 - Alterar Pessoa");

    System.out.println("3 - Excluir Pessoa");

    System.out.println("4 - Exibir Pessoa 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("=====");

    System.out.print("Escolha uma opção:");

}

```

```

private static int lerOpcao() {

    int opcao;

    try {

        opcao = scanner.nextInt();

    } catch (InputMismatchException e) {

```

```
        opcao = -1;
    }

    return opcao;
}
```

```
private static void incluirPessoa() {

    System.out.println("Incluir Pessoa (1 - Física, 2 - Jurídica): ");

    int tipo = lerOpcao();

    switch (tipo) {

        case 1 -> incluirPessoaFisica();

        case 2 -> incluirPessoaJuridica();

        default -> System.out.println("Opção inválida!");

    }

}
```

```
private static void incluirPessoaFisica() {

    System.out.print("Digite o ID da pessoa física: ");

    int id = scanner.nextInt();

    scanner.nextLine();

    System.out.print("Digite o nome da pessoa física: ");

    String nome = scanner.nextLine();

    System.out.print("Digite o CPF da pessoa física: ");

    String cpf = scanner.nextLine();

    System.out.print("Digite a idade da pessoa física: ");
```

```

        int idade = scanner.nextInt();

        scanner.nextLine();

        repoFisica.inserir(new PessoaFisica(id, nome, cpf, idade));

        System.out.println("Pessoa física adicionada com sucesso!");
    }

    private static void incluirPessoaJuridica() {

        System.out.print("Digite o ID da pessoa jurídica: ");

        int id = scanner.nextInt();

        scanner.nextLine(); // Consumir a quebra de linha

        System.out.print("Digite o nome da pessoa jurídica: ");

        String nome = scanner.nextLine();

        System.out.print("Digite o CNPJ da pessoa jurídica: ");

        String cnpj = scanner.nextLine();

        repoJuridica.inserir(new PessoaJuridica(id, nome, cnpj));

        System.out.println("Pessoa jurídica adicionada com sucesso!");
    }

    private static void alterarPessoaFisica() {

        System.out.println("Digite o ID da pessoa física a ser alterada: ");

        int id = scanner.nextInt();

        scanner.nextLine(); // Limpar o buffer do scanner

        PessoaFisica pessoa = repoFisica.obter(id);
    }

```



```

if (pessoa != null) {

    System.out.println("Digite o novo nome da pessoa física: ");

    String nome = scanner.nextLine();

    System.out.println("Digite o novo CPF da pessoa física: ");

    String cpf = scanner.nextLine();

    System.out.println("Digite a nova idade da pessoa física: ");

    int idade = scanner.nextInt();

    scanner.nextLine(); // Limpar o buffer do scanner

    pessoa.setNome(nome);

    pessoa.setCpf(cpf);

    pessoa.setIdade(idade);

    repoFisica.alterar(pessoa);

    System.out.println("Pessoa física alterada com sucesso!");

} else {

    System.out.println("Pessoa física não encontrada!");

}

}

```

```

private static void alterarPessoaJuridica() {

    System.out.println("Digite o ID da pessoa jurídica a ser alterada: ");

    int id = scanner.nextInt();

    scanner.nextLine();

    PessoaJuridica pessoa = repoJuridica.obter(id);

    if (pessoa != null) {

```

```
        System.out.println("Digite o novo nome da pessoa jurídica: ");

        String nome = scanner.nextLine();

        System.out.println("Digite o novo CNPJ da pessoa jurídica: ");

        String cnpj = scanner.nextLine();

        pessoa.setNome(nome);

        pessoa.setCnpj(cnpj);

        repoJuridica.alterar(id, pessoa);

        System.out.println("Pessoa jurídica alterada com sucesso!");

    } else {

        System.out.println("Pessoa jurídica não encontrada!");

    }

}
```

```
private static void excluirPessoa() {

    System.out.println("Excluir pessoa (1 - Física, 2 - Jurídica): ");

    int tipo = lerOpcao();

    switch (tipo) {

        case 1 -> excluirPessoaFisica();

        case 2 -> excluirPessoaJuridica();

        default -> System.out.println("Opção inválida!");

    }

}
```

```
private static void excluirPessoaFisica() {
```

```
System.out.println("Digite o ID da pessoa fisica a ser excluída: ");

int id = scanner.nextInt();

scanner.nextLine();

if (repoFisica.obter(id) != null) {

    repoFisica.excluir(id);

    System.out.println("Pessoa física excluída com sucesso!");

} else {

    System.out.println("Pessoa física não encontrada!");

}

}
```

```
private static void excluirPessoaJuridica() {

    System.out.println("Digite o ID da pessoa jurídica a ser excluída: ");

    int id = scanner.nextInt();

    scanner.nextLine();

    if (repoJuridica.obter(id) != null) {

        repoJuridica.excluir(id);

        System.out.println("Pessoa jurídica excluída com sucesso!");

    } else {

        System.out.println("Pessoa jurídica não encontrada!");

    }

}
```

```
private static void exibirPessoaPorId() {
```

```

System.out.println("Exibir pessoa (1 - Física, 2 - Jurídica): ");

int tipo = lerOpcao();

switch (tipo) {

    case 1 -> exibirPessoaFisicaPorId();

    case 2 -> exibirPessoaJuridicaPorId();

    default -> System.out.println("Opção inválida!");

}

}

private static void exibirPessoaFisicaPorId() {

    System.out.println("Digite o ID da pessoa física a ser exibida: ");

    int id = scanner.nextInt();

    scanner.nextLine();

    PessoaFisica pessoa = repoFisica.obter(id);

    if (pessoa != null) {

        System.out.println("Pessoa física encontrada:");

        System.out.println(pessoa);

    } else {

        System.out.println("Pessoa física não encontrada!");

    }

}

private static void exibirPessoaJuridicaPorId() {

    System.out.println("Digite o ID da pessoa jurídica a ser exibida: ");

```

```
int id = scanner.nextInt();

scanner.nextLine();

PessoaJuridica pessoa = repoJuridica.obter(id);

if (pessoa != null) {

    System.out.println("Pessoa jurídica encontrada:");

    System.out.println(pessoa);

} else {

    System.out.println("Pessoa jurídica não encontrada!");

}

}
```

```
private static void exibirTodasPessoas() {

    System.out.println("Exibir todas pessoas (1 - Física, 2 - Jurídica): ");

    int tipo = lerOpcao();

    switch (tipo) {

        case 1 -> {

            List<PessoaFisica> pessoas = repoFisica.obterTodos();

            if (!pessoas.isEmpty()) {

                System.out.println("Pessoas físicas encontradas:");

                for (PessoaFisica pessoa : pessoas) {

                    System.out.println(pessoa);

                }

            } else {

                System.out.println("Nenhuma pessoa física cadastrada!");

            }

        }

    }

}
```

```

        }
    }

    case 2 -> {

        List<PessoaJuridica> pessoas = repoJuridica.obterTodos();

        if (!pessoas.isEmpty()) {

            System.out.println("Pessoas jurídicas encontradas:");

            for (PessoaJuridica pessoa : pessoas) {

                System.out.println(pessoa);

            }

        } else {

            System.out.println("Nenhuma pessoa jurídica cadastrada!");

        }

    }

    default -> System.out.println("Opção inválida!");

}

}

```

```

private static void salvarDados() {

    System.out.println("Salvando dados...");

    try {

        repoFisica.persistir("pessoas_fisicas.dat");

        repoJuridica.persistir("pessoas_juridicas.dat");

        System.out.println("Dados salvos com sucesso!");

    } catch (IOException e) {

        System.out.println("Erro ao salvar os dados: " + e.getMessage());

    }

}

```

```
}
```

```
private static void recuperarDados() {
```

```
    System.out.println("Recuperando dados...");
```

```
    try {
```

```
        repoFisica.recuperar("pessoas_fisicas.dat");
```

```
        repoJuridica.recuperar("pessoas_juridicas.dat");
```

```
        System.out.println("Dados recuperados com sucesso!");
```

```
    } catch (IOException | ClassNotFoundException e) {
```

```
        System.out.println("Erro ao recuperar os dados: " + e.getMessage());
```

```
    }
```

```
}
```

```
private static void alterarPessoa() {
```

```
    System.out.println("Alterar pessoa (1 - Física, 2 - Jurídica): ");
```

```
    int tipo = lerOpcao();
```

```
    switch (tipo) {
```

```
        case 1 -> alterarPessoaFisica();
```

```
        case 2 -> alterarPessoaJuridica();
```

```
        default -> System.out.println("Opção inválida!");
```

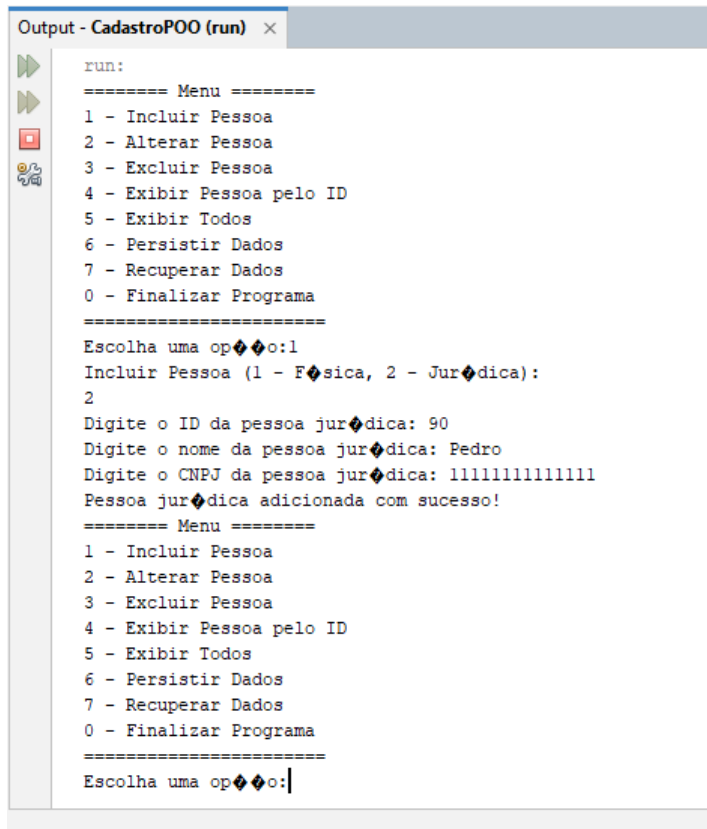
```
    }
```

```
}
```

```
}
```

Ao final desta prática, foi apresentado os conhecimentos básicos em Java, incluindo entrada e saída de dados, manipulação de arquivos, estruturação de classes e métodos, com a capacidade de desenvolver e manter sistemas simples de gerenciamento de dados em modo texto.

Resultado da execução.



```
Output - CadastroPOO (run) x
run:
===== Menu =====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Exibir Pessoa pelo ID
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
Escolha uma opção:1
Incluir Pessoa (1 - Física, 2 - Jurídica):
2
Digite o ID da pessoa jurídica: 90
Digite o nome da pessoa jurídica: Pedro
Digite o CNPJ da pessoa jurídica: 111111111111111111
Pessoa jurídica adicionada com sucesso!
===== Menu =====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Exibir Pessoa pelo ID
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
Escolha uma opção:|
```

Elementos estáticos:

Elementos estáticos em Java são aqueles que pertencem à classe em vez de pertencerem a uma instância específica da classe. Isso significa que eles são compartilhados por todas as instâncias da classe e podem ser acessados sem a necessidade de criar uma instância da classe.

Método main e modificador estático:

O método **main** é o ponto de entrada para um programa Java. Ele é o método que o ambiente de execução Java (JVM) procura quando inicia a execução do programa. O modificador **static** é usado para indicar que o método **main** pertence à classe em vez de pertencer a uma instância específica da classe. Isso permite que o método **main** seja chamado diretamente na classe, sem a necessidade de criar uma instância dela. É uma

convenção do Java que o método **main** seja definido como estático para permitir que ele seja o ponto de entrada do programa.

Classe Scanner:

A classe **Scanner** em Java é usada para obter entrada do usuário em programas Java. Ela fornece métodos para ler diferentes tipos de dados, como inteiros, números de ponto flutuante, strings, etc., do teclado ou de qualquer outra fonte de entrada. O **Scanner** divide a entrada em tokens usando um padrão especificado, que por padrão é o espaço em branco.

Impacto do uso de classes de repositório na organização do código:

O uso de classes de repositório, como **PessoaFisicaRepo** e **PessoaJuridicaRepo**, tem um impacto significativo na organização do código. Essas classes são responsáveis por armazenar e manipular os dados das pessoas físicas e jurídicas, respectivamente. Isso ajuda a separar as preocupações do domínio de negócios (manipulação de dados) da lógica de apresentação (interface do usuário). Além disso, as classes de repositório facilitam a manutenção do código, permitindo que as operações de manipulação de dados sejam centralizadas em um único local. Isso torna o código mais modular, reutilizável e fácil de entender e manter.