Missão Prática | Nível 1 | Mundo 3

Implementação de um cadastro de clientes em modo texto, com persistência em arquivos, baseado na tecnologia Java.

Objetivos:

- Utilizar herança e polimorfismo na definição de entidades.
- Utilizar persistência de objetos em arquivos binários.
- Implementar uma interface cadastral em modo texto.
- Utilizar o controle de exceções da plataforma Java.
- No final do projeto, o aluno terá implementado um sistema cadastral em Java, utilizando os recursos da programação orientada a objetos e a persistência em arquivos binários.

Procedimento 1

Criação das Entidades e Sistemas de Persistência

Os códigos desenvolvidos incluem:

1. Classes das entidades:
    a. 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 + "\nNome: " + nome);
    }
}
```

b. 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;
    }

    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 + "\nIdade: " + idade);
    }
}
```

c. 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;
    }

    public String getCnpj() {
        return cnpj;
    }

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

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

```
    }
}
```

2. Classes de repositórios:
    a. PessoaFisicaRepo

```
package model;

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

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

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

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

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

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

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

    public void persistir(String nomeArquivo) throws IOException {
```

```
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
            oos.writeObject(lista);
        }
    }

    public void recuperar(String nomeArquivo) throws IOException,
ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
            lista = (ArrayList<PessoaFisica>) ois.readObject();
        }
    }
}
```

b. PessoaJuridicaRepo.

```
package model;

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

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

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

    public void alterar(PessoaJuridica pessoa) {
        for (int i = 0; i < lista.size(); i++) {
            if (lista.get(i).getId() == pessoa.getId()) {
                lista.set(i, pessoa);
                return;
            }
        }
    }

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

    public PessoaJuridica obter(int id) {
        for (PessoaJuridica p : lista) {
```

```
            if (p.getId() == id) return p;
        }
        return null;
    }

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

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

    public void recuperar(String nomeArquivo) throws IOException,
ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
            lista = (ArrayList<PessoaJuridica>) ois.readObject();
        }
    }
}
```

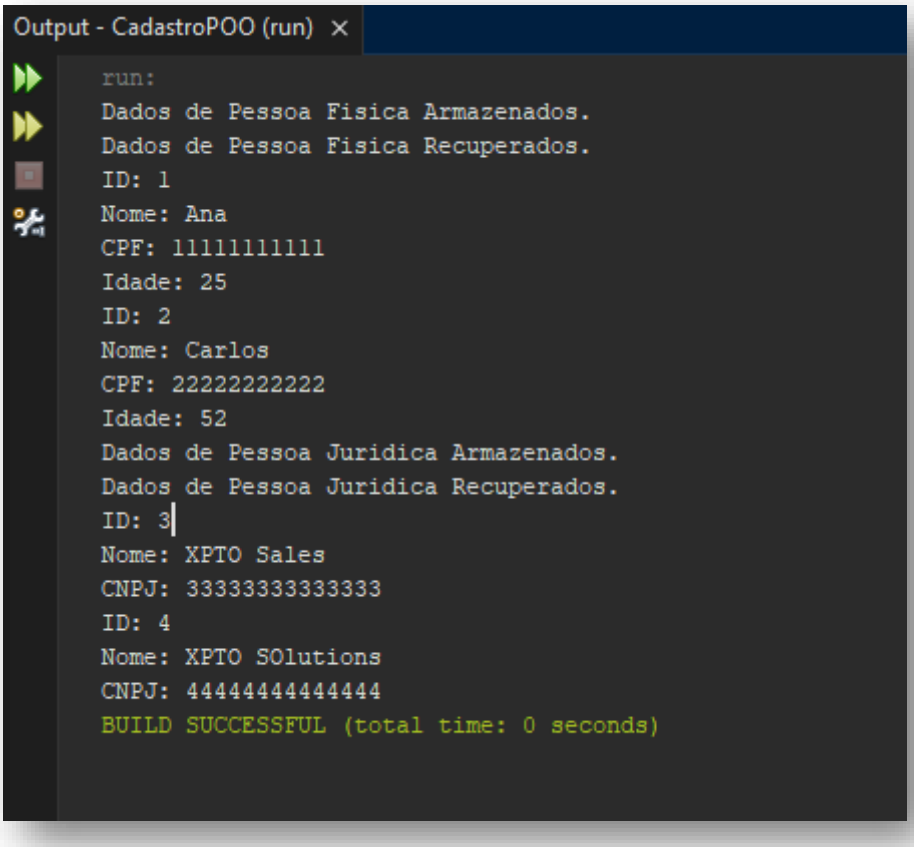
3. Método main:

```
package cadastropoo;

import model.*;

import java.io.IOException;



public class CadastroPOO {

    public static void main(String[] args) {

        PessoaFisicaRepo repoFisica1 = new PessoaFisicaRepo();

        repoFisica1.inserir(new PessoaFisica(1, "Ana", "11111111111", 25));

        repoFisica1.inserir(new PessoaFisica(2, "Carlos", "22222222222", 52));
```

```
        try {
            repoFisica1.persistir("pessoa_fisica.bin");
            System.out.println("Dados de Pessoa Fisica Armazenados.");
        } catch (IOException e) {
            System.err.println("Erro ao salvar dados de Pessoa Fisica. " + e.getMessage());
        }

        PessoaFisicaRepo repoFisica2 = new PessoaFisicaRepo();
        try {
            repoFisica2.recuperar("pessoa_fisica.bin");
            System.out.println("Dados de Pessoa Fisica Recuperados.");
            for (PessoaFisica pf : repoFisica2.obterTodos()) {
                pf.exibir();
            }
        } catch (IOException | ClassNotFoundException e) {
            System.err.println("Erro ao recuperar dados de Pessoa Fisica." + e.getMessage());
        }
        PessoaJuridicaRepo repoJuridica1 = new PessoaJuridicaRepo();
        repoJuridica1.inserir(new PessoaJuridica(3, "XPTO Sales", "33333333333333"));
        repoJuridica1.inserir(new PessoaJuridica(4, "XPTO SOlutions", "44444444444444"));

        try {
            repoJuridica1.persistir("pessoa_juridica.bin");
            System.out.println("Dados de Pessoa Juridica Armazenados.");
        } catch (IOException e) {
```

```
            System.err.println("Erro ao salvar dados de Pessoa Juridica." +
e.getMessage());
        }


        PessoaJuridicaRepo repoJuridica2 = new PessoaJuridicaRepo();
        try {
            repoJuridica2.recuperar("pessoa_juridica.bin");
            System.out.println("Dados de Pessoa Juridica Recuperados.");
            for (PessoaJuridica pj : repoJuridica2.obterTodos()) {
                pj.exibir();
            }
        } catch (IOException | ClassNotFoundException e) {
            System.err.println("Erro ao recuperar dados de Pessoa Juridica." +
e.getMessage());
        }
    }
}
```

Resultado da execução:



Análise e Conclusão:

Quais as vantagens e desvantagens do uso de herança?

A principal vantagem seria a reutilização de código, visto que a classe Pessoa contém atributos e métodos comuns, evitando que ocorra a duplicação em PessoaFisica e PessoaJuridica.

Uma desvantagem é o acoplamento onde as alterações na superclasse podem impactar todas as subclasses.

Por que a interface Serializable é necessária ao efetuar persistência em arquivos binários?

A interface Serializable é usada para que os objetos possam ser transformados em uma sequência de bytes. Depois, esses dados podem ser reconvertidos de volta ao formato original. Sem essa interface, o Java não permite esse processo para proteger a segurança e a integridade das informações.

Como o paradigma funcional é utilizado pela API stream no Java?

A API Stream no Java usa o paradigma funcional para trabalhar com dados de forma simples e direta. Em vez de usar loops, podemos usar métodos como forEach, filter e map para manipular os dados. Ela também permite que os dados sejam transformados sem alterar os originais, tornando o código mais claro e eficiente.

Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na persistência de dados em arquivos?

Na persistência de dados em arquivos, seguimos o Data Access Object (DAO), um padrão que separa a lógica de acesso aos dados (salvar, recuperar) da lógica de negócio. Em nosso caso, isso foi implementado nas classes PessoaFisicaRepo e PessoaJuridicaRepo. Esse padrão promove modularidade, facilita testes unitários e possibilita a substituição da camada de persistência sem impacto no restante do sistema.

Procedimento 2

Criação do Cadastro em Modo Texto

Os códigos desenvolvidos incluem:

1. A alteração do método main:

```
package cadastropoo;

import model.*;

import java.io.IOException;

import java.util.Scanner;


public class CadastroPOO {

    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("\n========================");
    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.print("========================\n");
    int opcao = scanner.nextInt();
    scanner.nextLine();

    if (opcao == 0) {
        System.out.println("Encerrando o programa...");
        break;
    }

    switch (opcao) {
        case 1 -> incluir(scanner, repoFisica, repoJuridica);
        case 2 -> alterar(scanner, repoFisica, repoJuridica);
        case 3 -> excluir(scanner, repoFisica, repoJuridica);
        case 4 -> exibirPorId(scanner, repoFisica, repoJuridica);
        case 5 -> exibirTodos(scanner, repoFisica, repoJuridica);
        case 6 -> salvarDados(scanner, repoFisica, repoJuridica);
        case 7 -> recuperarDados(scanner, repoFisica, repoJuridica);
        default -> System.out.println("Opção invalida. Tente novamente.");
    }
```

```
        }

        scanner.close();
    }

    private static void incluir(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
        System.out.print("Incluir (1 - Fisica, 2 - Juridica): ");
        int tipo = scanner.nextInt();
        scanner.nextLine();

        if (tipo == 1) {
            System.out.print("Digite o ID: ");
            int id = scanner.nextInt();
            scanner.nextLine();
            System.out.print("Insira os dados...\n");
            System.out.print("Nome: ");
            String nome = scanner.nextLine();
            System.out.print("CPF: ");
            String cpf = scanner.nextLine();
            System.out.print("Idade: ");
            int idade = scanner.nextInt();
            repoFisica.inserir(new PessoaFisica(id, nome, cpf, idade));
        } else if (tipo == 2) {
            System.out.print("Digite o ID: ");
            int id = scanner.nextInt();
            scanner.nextLine();
            System.out.print("Insira os dados...\n");
            System.out.print("Nome: ");
            String nome = scanner.nextLine();
```

```
            System.out.print("CNPJ: ");
            String cnpj = scanner.nextLine();
            repoJuridica.inserir(new PessoaJuridica(id, nome, cnpj));
        } else {
            System.out.println("Tipo invalido.");
        }
    }

    private static void alterar(Scanner scanner, PessoaFisicaRepo repoFisica,
PessoaJuridicaRepo repoJuridica) {
        System.out.print("Alterar (1 - Fisica, 2 - Juridica): ");
        int tipo = scanner.nextInt();
        scanner.nextLine();

        if (tipo == 1) {
            System.out.print("ID: ");
            int id = scanner.nextInt();
            scanner.nextLine();
            PessoaFisica pf = repoFisica.obter(id);
            if (pf != null) {
                System.out.println("Dados atuais:");
                pf.exibir();
                System.out.print("Novo nome: ");
                pf.setNome(scanner.nextLine());
                System.out.print("Novo CPF: ");
                pf.setCpf(scanner.nextLine());
                System.out.print("Nova idade: ");
                pf.setIdade(scanner.nextInt());
                repoFisica.alterar(pf);
            } else {
```

```
            System.out.println("Pessoa Fisica nao encontrada.");
        }
    } else if (tipo == 2) {
        System.out.print("ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        PessoaJuridica pj = repoJuridica.obter(id);
        if (pj != null) {
            System.out.println("Dados atuais:");
            pj.exibir();
            System.out.print("Novo nome: ");
            pj.setNome(scanner.nextLine());
            System.out.print("Novo CNPJ: ");
            pj.setCnpj(scanner.nextLine());
            repoJuridica.alterar(pj);
        } else {
            System.out.println("Pessoa Juridica nao encontrada.");
        }
    } else {
        System.out.println("Tipo invalido.");
    }
}

private static void excluir(Scanner scanner, PessoaFisicaRepo repoFisica,
PessoaJuridicaRepo repoJuridica) {
    System.out.print("Excluir (1 - Fisica, 2 - Juridica): ");
    int tipo = scanner.nextInt();
    System.out.print("ID: ");
    int id = scanner.nextInt();
```

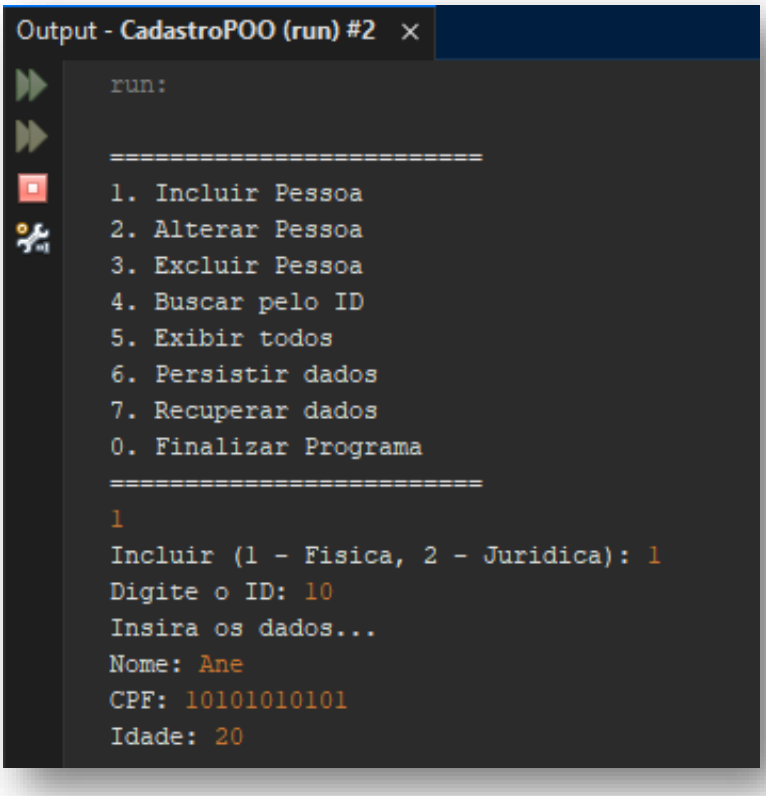
```
        if (tipo == 1) {
            repoFisica.excluir(id);
        } else if (tipo == 2) {
            repoJuridica.excluir(id);
        } else {
            System.out.println("Tipo invalido.");
        }
    }

    private static void exibirPorId(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
        System.out.print("Exibir pelo ID (1 - Fisica, 2 - Juridica): ");
        int tipo = scanner.nextInt();
        System.out.print("ID: ");
        int id = scanner.nextInt();

        if (tipo == 1) {
            PessoaFisica pf = repoFisica.obter(id);
            if (pf != null) pf.exibir();
            else System.out.println("Pessoa Fisica nao encontrada.");
        } else if (tipo == 2) {
            PessoaJuridica pj = repoJuridica.obter(id);
            if (pj != null) pj.exibir();
            else System.out.println("Pessoa Juridica nao encontrada.");
        } else {
            System.out.println("Tipo invalido.");
        }
    }
```

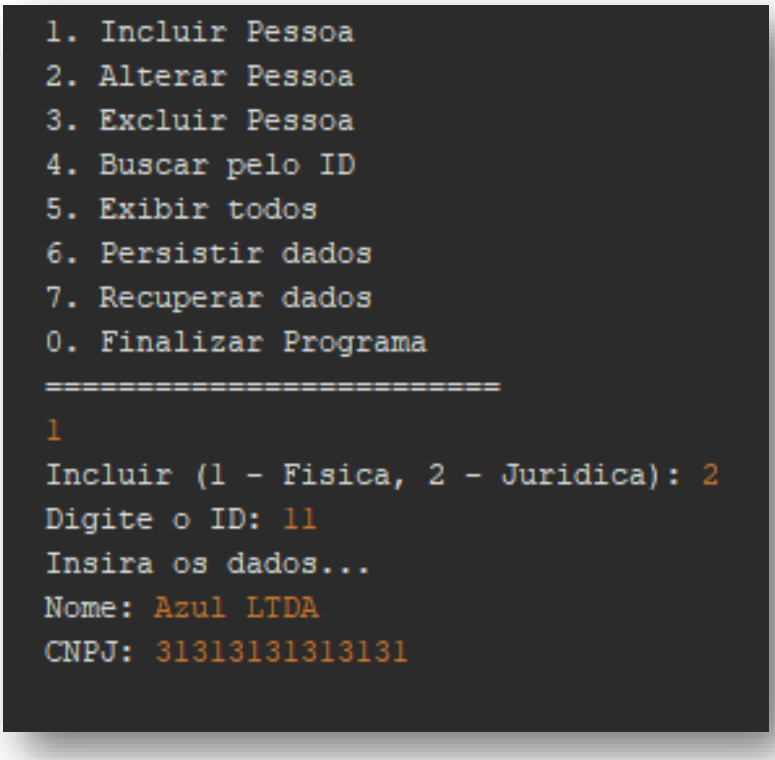
```
    private static void exibirTodos(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
        System.out.print("Exibir todos (1 - Fisica, 2 - Juridica): ");
        int tipo = scanner.nextInt();

        if (tipo == 1) {
            for (PessoaFisica pf : repoFisica.obterTodos()) {
                pf.exibir();
            }
        } else if (tipo == 2) {
            for (PessoaJuridica pj : repoJuridica.obterTodos()) {
                pj.exibir();
            }
        } else {
            System.out.println("Tipo invalido.");
        }
    }

    private static void salvarDados(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
        System.out.print("Prefixo do arquivo: ");
        String prefixo = scanner.next();
        try {
            repoFisica.persistir(prefixo + ".fisica.bin");
            repoJuridica.persistir(prefixo + ".juridica.bin");
            System.out.println("Dados salvos com sucesso.");
        } catch (IOException e) {
            System.err.println("Erro ao salvar os dados: " + e.getMessage());
        }
    }
```

```
    private static void recuperarDados(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
        System.out.print("Prefixo do arquivo: ");
        String prefixo = scanner.next();
        try {
            repoFisica.recuperar(prefixo + ".fisica.bin");
            repoJuridica.recuperar(prefixo + ".juridica.bin");
            System.out.println("Dados recuperados com sucesso.");
        } catch (IOException | ClassNotFoundException e) {
            System.err.println("Erro ao recuperar os dados: " + e.getMessage());
        }
    }
}
```

Resultados da execução:

Incluir Pessoa:





Exibir Todos:

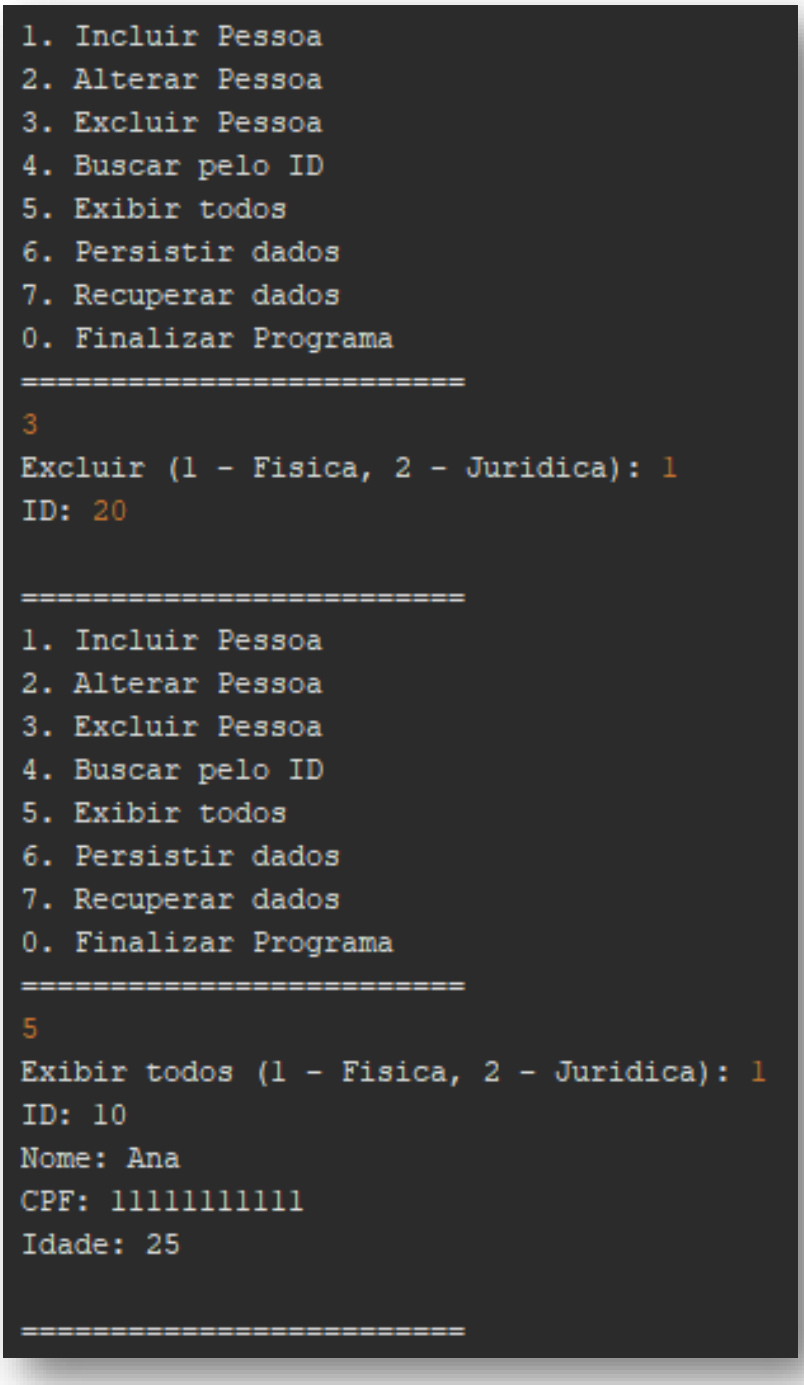
```
1. Incluir Pessoa
2. Alterar Pessoa
3. Excluir Pessoa
4. Buscar pelo ID
5. Exibir todos
6. Persistir dados
7. Recuperar dados
0. Finalizar Programa
==========================
5
Exibir todos (1 - Fisica, 2 - Juridica): 1
ID: 10
Nome: Ane
CPF: 10101010101
Idade: 20
ID: 20
Nome: Alex
CPF: 20202020202
Idade: 30
```

```
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
==========================
5
Exibir todos (1 - Fisica, 2 - Juridica): 2
ID: 11
Nome: Azul LTDA
CNPJ: 31313131313131
ID: 21
Nome: Vermelho LTDA
CNPJ: 41414141414141
```

Alterar Pessoa e Buscar pelo ID:

```
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
==========================
2
Alterar (1 - Fisica, 2 - Juridica): 1
ID: 10
Dados atuais:
ID: 10
Nome: Ane
CPF: 10101010101
Idade: 20
Novo nome: Ana
Novo CPF: 11111111111
Nova idade: 25

==========================
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
==========================
4
Exibir pelo ID (1 - Fisica, 2 - Juridica): 1
ID: 10
ID: 10
Nome: Ana
CPF: 11111111111
Idade: 25
```

Excluir Pessoa e Exibir todos:



Persistir dados, Recuperar dados e Exibir todos:

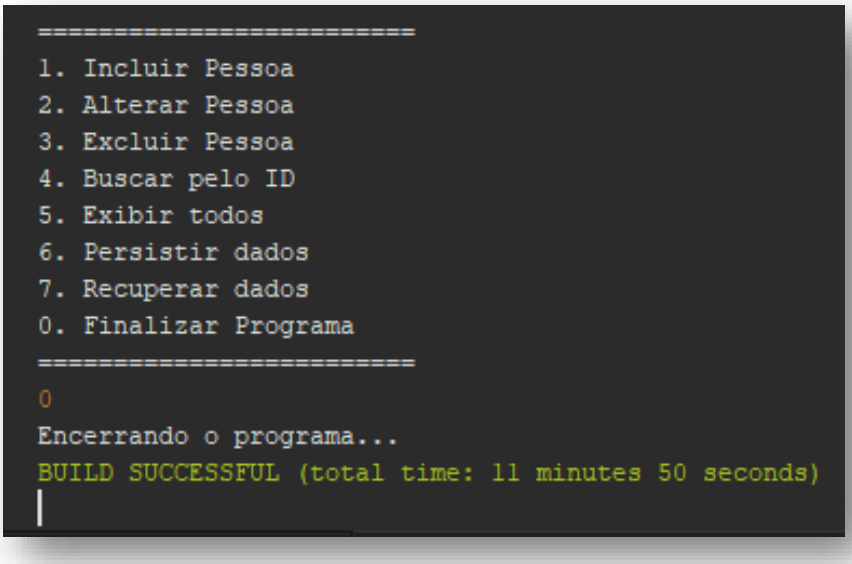
```
==========================
1. Incluir Pessoa
2. Alterar Pessoa
3. Excluir Pessoa
4. Buscar pelo ID
5. Exibir todos
6. Persistir dados
7. Recuperar dados
0. Finalizar Programa
==========================
6
Prefixo do arquivo: teste
Dados salvos com sucesso.

==========================
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
==========================
7
Prefixo do arquivo: teste
Dados recuperados com sucesso.

==========================
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
==========================
5
Exibir todos (1 - Fisica, 2 - Juridica): 2
ID: 11
Nome: Azul LTDA
CNPJ: 31313131313131
ID: 21
Nome: Vermelho LTDA
CNPJ: 41414141414141
```

Excluir Pessoa, Exibir todos, Recuperar dados e Exibir todos:

```
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
========================
3
Excluir (1 - Fisica, 2 - Juridica): 2
ID: 11

========================
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
========================
3
Excluir (1 - Fisica, 2 - Juridica): 2
ID: 21

========================
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. Incluir Pessoa
2. Alterar Pessoa
3. Excluir Pessoa
4. Buscar pelo ID
5. Exibir todos
6. Persistir dados
7. Recuperar dados
0. Finalizar Programa
==========================
5
Exibir todos (1 - Fisica, 2 - Juridica): 2

==========================
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
==========================
7
Prefixo do arquivo: teste
Dados recuperados com sucesso.

==========================
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
==========================
5
Exibir todos (1 - Fisica, 2 - Juridica): 2
ID: 11
Nome: Azul LTDA
CNPJ: 31313131313131
ID: 21
Nome: Vermelho LTDA
CNPJ: 41414141414141
```

Finalizar Programa:



Análise e Conclusão:

O que são elementos estáticos e por que o método main usa esse modificador?

Elementos estáticos são aqueles que pertencem à classe, não a objetos específicos. O método main é estático porque a JVM precisa chamá-lo sem criar um objeto da classe. Assim, ele pode ser o ponto de entrada do programa sem depender de uma instância.

Para que serve a classe Scanner?

A classe Scanner serve para ler dados do usuário, como texto e números, a partir do teclado ou de outros lugares. No programa, usamos ela para capturar entradas do usuário durante a execução.

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

As classes de repositório ajudam a organizar o código, separando a parte que lida com dados (como salvar, alterar ou excluir) do restante do programa. Isso torna o código mais claro, fácil de manter e permite que possamos mudar a forma de armazenar os dados sem complicar o resto do programa.

Repositório GitHub:

IMCS01/Miss-o-Pr-tica-N-vel-1-Mundo-3