



Estácio

Campus: Maracanaú

Curso: Desenvolvimento Full Stack

Disciplina: Nível 1 - Iniciando o Caminho Pelo Java

Aluna: Luanna Barros Soares

Matrícula: 202209281315

Semestre: Terceiro

Endereço Git: <https://github.com/luannasoes02/Missao-Pratica-Nivel-1-Mundo-3>

MISSÃO PRÁTICA - NÍVEL 1 - MUNDO 3

MARACANAÚ - CE

2024

1º Procedimento | Criação das Entidades e Sistema de Persistência

1- PACOTE MODEL

1.1- Pessoa.java

```
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);
    }
}
```

1.2- PessoaFísica.java

```

package model;

import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {
    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);
    }
}

```

1.3- PessoaFísicaRepo.java

```

package model;

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

public class PessoaFisicaRepo {

```

```
private ArrayList<PessoaFisica> pessoasFisicas = new ArrayList<>();
```

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

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

```
public void excluir(int id) {  
    for (int i = 0; i < pessoasFisicas.size(); i++) {  
        if (pessoasFisicas.get(i).getId() == id) {  
            pessoasFisicas.remove(i);  
            break;  
        }  
    }  
}
```

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

```
public ArrayList<PessoaFisica> obterTodos() {  
    return pessoasFisicas;  
}
```

```
public void persistir(String nomeArquivo) throws IOException {  
    try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {  
        out.writeObject(pessoasFisicas);  
    }  
}
```

```
public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {  
    try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(nomeArquivo))) {  
        // ...  
    }  
}
```

```

        pessoasFisicas = (ArrayList<PessoaFisica>) in.readObject();
    }
}
}

```

1.4- PessoaJuridica.java

```

package model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {
    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);
    }
}

```

1.5- PessoaJuridicaRepo.java

```

package model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {
    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- Main.java

```

import java.io.IOException;
import model.PessoaFisica;
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;

public class Main {
    public static void main(String[] args) {
        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

        repo1.inserir(new PessoaFisica(1, "João", "123.456.789-00", 30));
        repo1.inserir(new PessoaFisica(2, "Maria", "987.654.321-00", 25));

        try {
            repo1.persistir("pessoas_fisicas.dat");
        } catch (IOException e) {
            System.out.println("Erro ao persistir os dados das pessoas físicas: " +
e.getMessage());
        }

        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
        try {
            repo2.recuperar("pessoas_fisicas.dat");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao recuperar os dados das pessoas físicas: " +
e.getMessage());
        }
    }
}

```

```

        System.out.println("Pessoas Físicas recuperadas:");
        for (PessoaFisica pessoa : repo2.obterTodos()) {
            pessoa.exibir();
        }

        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();

        repo3.inserir(new PessoaJuridica(1, "Empresa A", "12345678901234"));
        repo3.inserir(new PessoaJuridica(2, "Empresa B", "98765432109876"));

        try {
            repo3.persistir("pessoas_juridicas.dat");
        } catch (IOException e) {
            System.out.println("Erro ao persistir os dados das pessoas jurídicas: " + e.getMessage());
        }

        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
        try {
            repo4.recuperar("pessoas_juridicas.dat");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao recuperar os dados das pessoas jurídicas: " + e.getMessage());
        }

        System.out.println("Pessoas Jurídicas recuperadas:");
        for (PessoaJuridica pessoa : repo4.obterTodos()) {
            pessoa.exibir();
        }
    }
}

```

3- Main.java **atualizado***

```

public class Main {
    public static void main(String[] args) {
        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

        repo1.inserir(new PessoaFisica(1, "Ana", "11111111111", 25));
        repo1.inserir(new PessoaFisica(2, "Carlos", "22222222222", 52));

        try {
            repo1.persistir("pessoas_fisicas.dat");
            System.out.println("Dados de Pessoa Física Armazenados.");
        } catch (IOException e) {
            System.out.println("Erro ao persistir os dados das pessoas físicas: " + e.getMessage());
        }
    }
}

```

```

PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
try {
    repo2.recuperar("pessoas_fisicas.dat");
    System.out.println("Dados de Pessoa Física Recuperados.");
    for (PessoaFisica pessoa : repo2.obterTodos()) {
        System.out.println("Id: " + pessoa.getId());
        System.out.println("Nome: " + pessoa.getNome());
        System.out.println("CPF: " + pessoa.getCpf());
        System.out.println("Idade: " + pessoa.getIdade());
    }
} catch (IOException | ClassNotFoundException e) {
    System.out.println("Erro ao recuperar os dados das pessoas físicas: " + e.get-
Message());
}

```

```

PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();

```

```

repo3.inserir(new PessoaJuridica(3, "XPTO Sales", "333333333333"));
repo3.inserir(new PessoaJuridica(4, "XPTO Solutions", "4444444444444"));

```

```

try {
    repo3.persistir("pessoas_juridicas.dat");
    System.out.println("\nDados de Pessoa Jurídica Armazenados.");
} catch (IOException e) {
    System.out.println("Erro ao persistir os dados das pessoas jurídicas: " + e.get-
Message());
}

```

```

PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();

```

```

try {
    repo4.recuperar("pessoas_juridicas.dat");
    System.out.println("Dados de Pessoa Jurídica Recuperados.");
    for (PessoaJuridica pessoa : repo4.obterTodos()) {
        System.out.println("Id: " + pessoa.getId());
        System.out.println("Nome: " + pessoa.getNome());
        System.out.println("CNPJ: " + pessoa.getCnpj());
    }
} catch (IOException | ClassNotFoundException e) {
    System.out.println("Erro ao recuperar os dados das pessoas jurídicas: " +
e.getMessage());
}
}
}

```


4- RESULTADOS DA EXECUÇÃO:

```
Output - CadastroPOO (run) X
run:
Dados de Pessoa Física Armazenados.
Dados de Pessoa Física Recuperados.
Id: 1
Nome: Ana
CPF: 11111111111
Idade: 25
Id: 2
Nome: Carlos
CPF: 22222222222
Idade: 52

Dados de Pessoa Jur dica Armazenados.
Dados de Pessoa Jur dica Recuperados.
Id: 3
Nome: XPTO Sales
CNPJ: 333333333333
Id: 4
Nome: XPTO Solutions
CNPJ: 444444444444
BUILD SUCCESSFUL (total time: 1 second)
```

5- AN LISE E CONCLUS O

A. Quais as vantagens e desvantagens do uso de heran a?

R= Utilizar heran a ajuda a economizar tempo, bem como reduzir o c digo, especialmente quando v rias classe t m comportamentos semelhantes. Uma desvantagem   que tendo muitas classes que se relacionam entre si acaba deixando o c digo confuso, at  porque se alterar a classe base acaba afetando todas as que est o herdando e isso pode gerar problemas.

B. Por que a interface Serializable   necess ria ao efetuar persist ncia em arquivos bin rios?

R=   necess rio utiliz -la quando se quer guardar os dados em arquivos de forma que estes possam ser lidos depois.

C. Como o paradigma funcional   utilizado pela API stream no Java?

R=   como uma caixa de ferramentas, com ela se aplica filtros, transforma dados, entre outros.   como uma varinha m gica para transformar listas de n meros em outros n meros.

D. Quando trabalhamos com Java, qual padr o de desenvolvimento   adotado na persist ncia de dados em arquivos?

R= Repository. Ou seja, separar parte do c digo que lida com salvar e recuperar dados em classes pr prias, como se fossem caixas para guardar e buscar informa  es.

2º Procedimento | Criação do Cadastro em Modo Texto

1- Main.java **atualizado***

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;
import model.PessoaFisica;
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
        PessoaJuridicaRepo repo2 = new PessoaJuridicaRepo();

        while (true) {
            System.out.println("Selecione uma opção:");
            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. Finalizar");

            int opcao = scanner.nextInt();

            switch (opcao) {
                case 1:
                    System.out.println("Selecione o tipo (1 para Pessoa Física, 2 para Pessoa Jurídica):");
                    int tipo = scanner.nextInt();
                    if (tipo == 1) {
                        incluirPessoaFisica(scanner, repo1);
                    } else if (tipo == 2) {
                        incluirPessoaJuridica(scanner, repo2);
                    } else {
                        System.out.println("Opção inválida.");
                    }
                    break;
                case 2:
                    System.out.println("Selecione o tipo (1 para Pessoa Física, 2 para
```

```

Pessoa Jurídica:");
    int tipoAlterar = scanner.nextInt();
    if (tipoAlterar == 1) {
        alterarPessoaFisica(scanner, repo1);
    } else if (tipoAlterar == 2) {
        alterarPessoaJuridica(scanner, repo2);
    } else {
        System.out.println("Opção inválida.");
    }
    break;
case 3:
    System.out.println("Selecione o tipo (1 para Pessoa Física, 2 para
Pessoa Jurídica:");
    int tipoExcluir = scanner.nextInt();
    if (tipoExcluir == 1) {
        excluirPessoaFisica(scanner, repo1);
    } else if (tipoExcluir == 2) {
        excluirPessoaJuridica(scanner, repo2);
    } else {
        System.out.println("Opção inválida.");
    }
    break;
case 4:
    System.out.println("Selecione o tipo (1 para Pessoa Física, 2 para
Pessoa Jurídica:");
    int tipoExibirPorId = scanner.nextInt();
    if (tipoExibirPorId == 1) {
        exibirPessoaFisicaPorId(scanner, repo1);
    } else if (tipoExibirPorId == 2) {
        exibirPessoaJuridicaPorId(scanner, repo2);
    } else {
        System.out.println("Opção inválida.");
    }
    break;
case 5:
    System.out.println("Selecione o tipo (1 para Pessoa Física, 2 para
Pessoa Jurídica:");
    int tipoExibirTodos = scanner.nextInt();
    if (tipoExibirTodos == 1) {
        exibirTodasPessoasFisicas(repo1);
    } else if (tipoExibirTodos == 2) {
        exibirTodasPessoasJuridicas(repo2);
    } else {
        System.out.println("Opção inválida.");
    }
    break;
case 6:
    salvarDados(scanner, repo1, repo2);

```

```

        break;
    case 7:
        recuperarDados(scanner, repo1, repo2);
        break;
    case 0:
        System.out.println("Finalizando...");
        return;
    default:
        System.out.println("Opção inválida.");
    }
}
}

```

```

public static void incluirPessoaFisica(Scanner scanner, PessoaFisicaRepo repo) {
    System.out.println("Digite o ID:");
    int id = scanner.nextInt();
    System.out.println("Digite o nome:");
    String nome = scanner.next();
    System.out.println("Digite o CPF:");
    String cpf = scanner.next();
    System.out.println("Digite a idade:");
    int idade = scanner.nextInt();

    PessoaFisica pessoa = new PessoaFisica(id, nome, cpf, idade);
    repo.inserir(pessoa);
    System.out.println("Pessoa física adicionada com sucesso.");
}

```

```

public static void incluirPessoaJuridica(Scanner scanner, PessoaJuridicaRepo repo) {
    System.out.println("Digite o ID:");
    int id = scanner.nextInt();
    System.out.println("Digite o nome:");
    String nome = scanner.next();
    System.out.println("Digite o CNPJ:");
    String cnpj = scanner.next();

    PessoaJuridica pessoa = new PessoaJuridica(id, nome, cnpj);
    repo.inserir(pessoa);
    System.out.println("Pessoa jurídica adicionada com sucesso.");
}

```

```

public static void alterarPessoaFisica(Scanner scanner, PessoaFisicaRepo repo) {
    System.out.println("Digite o ID da pessoa física a ser alterada:");
    int id = scanner.nextInt();
    PessoaFisica pessoa = repo.obter(id);
}

```

```

    if (pessoa != null) {
        System.out.println("Dados atuais da pessoa física:");
        pessoa.exibir();
        System.out.println("Digite o novo nome:");
        String nome = scanner.next();
        System.out.println("Digite o novo CPF:");
        String cpf = scanner.next();
        System.out.println("Digite a nova idade:");
        int idade = scanner.nextInt();
        pessoa.setNome(nome);
        pessoa.setCpf(cpf);
        pessoa.setIdade(idade);
        System.out.println("Pessoa física alterada com sucesso.");
    } else {
        System.out.println("Pessoa física não encontrada.");
    }
}

```

```

public static void alterarPessoaJuridica(Scanner scanner, PessoaJuridicaRepo
repo) {
    System.out.println("Digite o ID da pessoa jurídica a ser alterada:");
    int id = scanner.nextInt();
    PessoaJuridica pessoa = repo.obter(id);
    if (pessoa != null) {
        System.out.println("Dados atuais da pessoa jurídica:");
        pessoa.exibir();
        System.out.println("Digite o novo nome:");
        String nome = scanner.next();
        System.out.println("Digite o novo CNPJ:");
        String cnpj = scanner.next();
        pessoa.setNome(nome);
        pessoa.setCnpj(cnpj);
        System.out.println("Pessoa jurídica alterada com sucesso.");
    } else {
        System.out.println("Pessoa jurídica não encontrada.");
    }
}

```

```

public static void excluirPessoaFisica(Scanner scanner, PessoaFisicaRepo re-
po) {
    System.out.println("Digite o ID da pessoa física a ser excluída:");
    int id = scanner.nextInt();
    repo.excluir(id);
    System.out.println("Pessoa física excluída com sucesso.");
}

```

```

public static void excluirPessoaJuridica(Scanner scanner, PessoaJuridicaRepo
repo) {

```

```

        System.out.println("Digite o ID da pessoa jurídica a ser excluída:");
        int id = scanner.nextInt();
        repo.excluir(id);
        System.out.println("Pessoa jurídica excluída com sucesso.");
    }

```

```

    public static void exibirPessoaFisicaPorId(Scanner scanner, PessoaFisicaRe-
    po repo) {
        System.out.println("Digite o ID da pessoa física:");
        int id = scanner.nextInt();
        PessoaFisica pessoa = repo.obter(id);
        if (pessoa != null) {
            pessoa.exibir();
        } else {
            System.out.println("Pessoa física não encontrada.");
        }
    }
}

```

```

    public static void exibirPessoaJuridicaPorId(Scanner scanner, PessoaJuridica-
    Repo repo) {
        System.out.println("Digite o ID da pessoa jurídica:");
        int id = scanner.nextInt();
        PessoaJuridica pessoa = repo.obter(id);
        if (pessoa != null) {
            pessoa.exibir();
        } else {
            System.out.println("Pessoa jurídica não encontrada.");
        }
    }
}

```

```

    public static void exibirTodasPessoasFisicas(PessoaFisicaRepo repo) {
        ArrayList<PessoaFisica> pessoas = repo.obterTodos();
        if (!pessoas.isEmpty()) {
            for (PessoaFisica pessoa : pessoas) {
                pessoa.exibir();
            }
        } else {
            System.out.println("Não há pessoas físicas cadastradas.");
        }
    }
}

```

```

    public static void exibirTodasPessoasJuridicas(PessoaJuridicaRepo repo) {
        ArrayList<PessoaJuridica> pessoas = repo.obterTodos();
        if (!pessoas.isEmpty()) {
            for (PessoaJuridica pessoa : pessoas) {
                pessoa.exibir();
            }
        } else {

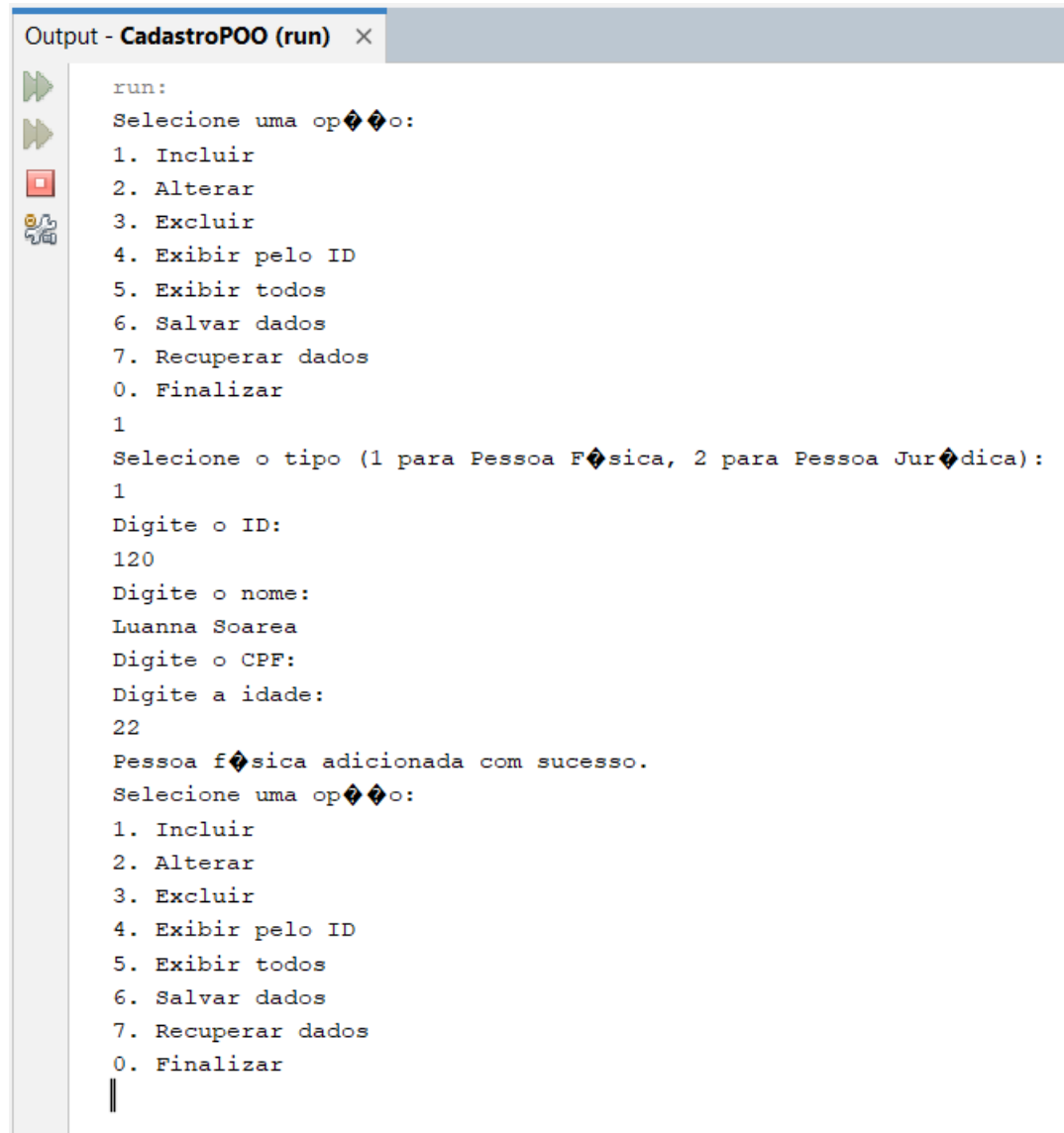
```

```
        System.out.println("Não há pessoas jurídicas cadastradas.");
    }
}
```

```
public static void salvarDados(Scanner scanner, PessoaFisicaRepo repo1,
PessoaJuridicaRepo repo2) {
    try {
        System.out.println("Digite o prefixo dos arquivos:");
        String prefixo = scanner.next();
        repo1.persistir(prefixo + ".fisica.bin");
        repo2.persistir(prefixo + ".juridica.bin");
        System.out.println("Dados salvos com sucesso.");
    } catch (IOException e) {
        System.out.println("Erro ao salvar os dados: " + e.getMessage());
    }
}
```

```
public static void recuperarDados(Scanner scanner, PessoaFisicaRepo repo1,
PessoaJuridicaRepo repo2) {
    try {
        System.out.println("Digite o prefixo dos arquivos:");
        String prefixo = scanner.next();
        repo1.recuperar(prefixo + ".fisica.bin");
        repo2.recuperar(prefixo + ".juridica.bin");
        System.out.println("Dados recuperados com sucesso.");
    } catch (IOException | ClassNotFoundException e) {
        System.out.println("Erro ao recuperar os dados: " + e.getMessage());
    }
}
}
```

2- RESULTADOS DA EXECUÇÃO:



```
run:
Selecione uma opção:
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Finalizar
1
Selecione o tipo (1 para Pessoa Física, 2 para Pessoa Jurídica):
1
Digite o ID:
120
Digite o nome:
Luanna Soarea
Digite o CPF:
Digite a idade:
22
Pessoa física adicionada com sucesso.
Selecione uma opção:
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Finalizar
||
```

3- ANÁLISE E CONCLUSÃO:

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

R= Elementos estáticos em Java são membros de uma classe que pertencem à própria classe em vez de pertencerem a instâncias individuais dessa classe. Isso significa que eles podem ser acessados sem criar uma instância da classe. O método main é geralmente declarado como estático para permitir que ele seja chamado sem instanciar a classe.

B. Para que serve a classe Scanner?

R= É usada para obter entrada do usuário a partir do console. Ela fornece métodos para ler diferentes tipos de dados, como inteiros, decimais e strings, do console. Isso permite que os programas Java interajam com o usuário, aceitando entrada de teclado e respondendo com saída adequada.

C. Como o uso de classes de repositório impactou a organização do código?

R= O uso de classes de repositório, como PessoaFisicaRepo e PessoaJuridicaRepo,

impactou positivamente na organização do código, separando a lógica de acesso a dados e manipulação de objetos em classes específicas. Isso promove uma separação clara de responsabilidades, facilitando a manutenção e extensão do código. Além disso, permite o encapsulamento das operações relacionadas ao armazenamento e recuperação de dados, tornando o código mais modular e reutilizável.