Iniciando o caminho pelo Java

Analécia Mariana Oliveira dos Santos - 202208453872

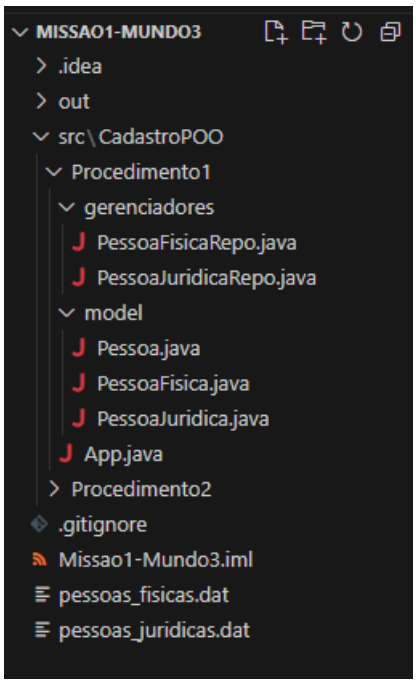
RPG0014 – Mundo 3

Link do projeto - https://github.com/Analecia/Missao1-Mundo3.git

Objetivo da Prática

1. Utilizar herança e polimorfismo na definição de entidades.
2. Utilizar persistência de objetos em arquivos binários.
3. Implementar uma interface cadastral em modo texto.
4. Utilizar o controle de exceções da plataforma Java.
7. arquivos binários.

1º Procedimento – Criação das Entidades e Sistema de Persistência



Pessoa.Java

```
package CadastroPOO.Procedimento1.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 this.id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNome() {
        return this.nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public void exibir() {
        System.out.println("ID: " + this.id);
        System.out.println("Nome: " + this.nome);
    }
}
```

PessoaFisica.Java

```
package CadastroPOO.Procedimento1.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 this.cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public int getIdade() {
        return this.idade;
    }

    public void setIdade(int idade) {
        this.idade = idade;
    }

    public void exibir() {
        super.exibir();
        System.out.println("CPF: " + this.cpf);
        System.out.println("Idade: " + this.idade);
    }
}
```

PessoaJuridica.Java

```
package CadastroPOO.Procedimento1.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 this.cnpj;
    }

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

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

```
    }
}
```

PessoaFisicaRepo.Java

```
package CadastroPOO.Procedimento1.gerenciadores;

import CadastroPOO.Procedimento1.model.PessoaFisica;
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.NoSuchElementException;
import java.util.Optional;

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

    public PessoaFisicaRepo() {
    }

    public void inserir(PessoaFisica pessoaFisica) {
        this.pessoasFisicas.add(pessoaFisica);
    }

    public void alterar(PessoaFisica pessoaFisica, String novoNome,
String novoCpf, int novaIdade) {
        pessoaFisica.setNome(novoNome);
        pessoaFisica.setCpf(novoCpf);
        pessoaFisica.setIdade(novaIdade);
    }

    public void excluir(int id) {
        this.pessoasFisicas.remove(this.obter(id));
    }

    public PessoaFisica obter(int id) throws NoSuchElementException {
        Optional<PessoaFisica> pessoaFisicaEncontrada =
this.pessoasFisicas.stream().filter((pessoaFisica) -> {
            return pessoaFisica.getId() == id;
        }).findFirst();
        if (pessoaFisicaEncontrada.isPresent()) {
            return (PessoaFisica)pessoaFisicaEncontrada.get();
        } else {
            throw new NoSuchElementException("Pessoa f\u00edsica com ID "
+ id + " n\u00e3o encontrada.");
        }
    }
```

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

    public void persistir(String nomeArquivo) throws IOException {
        ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(nomeArquivo));
        outputStream.writeObject(this.pessoasFisicas);
        outputStream.close();
        System.out.println("Dados da pessoa f\u00edsica armazenados.");
        System.out.println();
    }

    public void recuperar(String nomeArquivo) throws IOException,
ClassNotFoundException {
        ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(nomeArquivo));
        this.pessoasFisicas = (ArrayList)inputStream.readObject();
        inputStream.close();
        System.out.println("Dados da pessoa f\u00edsica recuperados.");
        System.out.println();
    }
}
```

PessoaJuridicaRepo.Java

```
package CadastroPOO.Procedimento1.gerenciadores;

import CadastroPOO.Procedimento1.model.PessoaJuridica;
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.NoSuchElementException;
import java.util.Optional;

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

    public PessoaJuridicaRepo() {
    }

    public void inserir(PessoaJuridica pessoaJuridica) {
        this.pessoasJuridicas.add(pessoaJuridica);
    }
```

```
    public void alterar(PessoaJuridica pessoaJuridica, String novoNome,
String novoCpf) {
        pessoaJuridica.setNome(novoNome);
        pessoaJuridica.setCnpj(novoCpf);
    }

    public void excluir(int id) {
        this.pessoasJuridicas.remove(this.obter(id));
    }

    public PessoaJuridica obter(int id) throws NoSuchElementException {
        Optional<PessoaJuridica> pessoaJuridicaEncontrada =
this.pessoasJuridicas.stream().filter((pessoaJuridica) -> {
            return pessoaJuridica.getId() == id;
        }).findFirst();
        if (pessoaJuridicaEncontrada.isPresent()) {
            return (PessoaJuridica)pessoaJuridicaEncontrada.get();
        } else {
            throw new NoSuchElementException("Pessoa jur\u00eddica com ID
" + id + " n\u00e3o encontrada.");
        }
    }

    public ArrayList<PessoaJuridica> obterTodos() {
        return this.pessoasJuridicas;
    }

    public void persistir(String nomeArquivo) throws IOException {
        ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(nomeArquivo));
        outputStream.writeObject(this.pessoasJuridicas);
        outputStream.close();
        System.out.println("Dados da pessoa jur\u00eddica armazenados.");
        System.out.println();
    }

    public void recuperar(String nomeArquivo) throws IOException,
ClassNotFoundException {
        ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(nomeArquivo));
        this.pessoasJuridicas = (ArrayList)inputStream.readObject();
        inputStream.close();
        System.out.println("Dados da pessoa jur\u00eddica recuperados.");
        System.out.println();
    }
}
```

App.Java

```
package CadastroPOO.Procedimento1;
```

```java
import CadastroPOO.Procedimento1.model.PessoaFisica;
import CadastroPOO.Procedimento1.model.PessoaJuridica;
import CadastroPOO.Procedimento1.gerenciadores.PessoaFisicaRepo;
import CadastroPOO.Procedimento1.gerenciadores.PessoaJuridicaRepo;
import java.io.IOException;
public class App {
    public App() {
    }

    public static void main(String[] args) {
        PessoaFisicaRepo pessoaFisicaRepo1 = new PessoaFisicaRepo();
        PessoaFisica pessoaFisica1 = new PessoaFisica(1, "Maribel",
"12.123.123-12", 21);
        PessoaFisica pessoaFisica2 = new PessoaFisica(2, "Lucca",
"34.345.345-34",27);
        pessoaFisicaRepo1.inserir(pessoaFisica1);
        pessoaFisicaRepo1.inserir(pessoaFisica2);

        try {
            pessoaFisicaRepo1.persistir("pessoas_fisicas.dat");
            PessoaFisicaRepo pessoaFisicaRepo2 = new PessoaFisicaRepo();
            pessoaFisicaRepo2.recuperar("pessoas_fisicas.dat");
            pessoaFisicaRepo2.obterTodos().forEach((pessoaFisica) -> {
                pessoaFisica.exibir();
                System.out.println();
            });
        }

        catch (ClassNotFoundException | IOException var9) {
            System.out.println("Erro ao persistir ou recuperar os dados:
" + var9.getMessage());
        }

        PessoaJuridicaRepo pessoaJuridicaRepo1 = new
PessoaJuridicaRepo();
        PessoaJuridica pessoaJuridica1 = new PessoaJuridica(1, "Comercial
Casa da Cesta", "11.111.111/0000-00");
        PessoaJuridica pessoaJuridica2 = new PessoaJuridica(2,
"Supermercado Oliveira", "00.000.000/1111-11");
        pessoaJuridicaRepo1.inserir(pessoaJuridica1);
        pessoaJuridicaRepo1.inserir(pessoaJuridica2);

        try {
            pessoaJuridicaRepo1.persistir("pessoas_juridicas.dat");
            PessoaJuridicaRepo pessoaJuridicaRepo2 = new
PessoaJuridicaRepo();
            pessoaJuridicaRepo2.recuperar("pessoas_juridicas.dat");
```

```
pessoaJuridicaRepo2.obterTodos().stream().forEach((pessoaJuridica) -> {
                pessoaJuridica.exibir();
                System.out.println();
            });
        } catch (ClassNotFoundException | IOException var8) {
            System.out.println("Erro ao persistir ou recuperar os dados:
" + var8.getMessage());
        }

    }
}
```

RESULTADO DA PRÁTICA / PROCEDIMENTO 1

```
Dados da pessoa física recuperados.

ID: 1
Nome: Maribel
CPF: 12.123.123-12
Idade: 21

ID: 2
Nome: Lucca
CPF: 34.345.345-34
Idade: 27

Dados da pessoa jurídica armazenados.

Dados da pessoa jurídica recuperados.

ID: 1
Nome: Comercial Casa da Cesta
CNPJ: 11.111.111/0000-00

ID: 2
Nome: Supermercado Oliveira
CNPJ: 00.000.000/1111-11

PS C:\mundo3-estacio\Missao1-Mundo3>
```

Análise e Conclusão:

a. Quais as vantagens e desvantagens do uso de herança?

As principais vantagens que podem ser citadas são: a reutilização de código, o polimorfismo e a facilidade em utilizar a hierarquia de classes.

E as principais desvantagens que podem ser citadas são: o acoplamento, a herança múltipla não suportada, complexidade e legibilidade, também a quebra do encapsulamento.

De maneira simplificada, a herança em Java oferece benefícios significativos de reutilização e organização de código, mas seu uso excessivo ou inadequado pode levar a problemas de manutenção, legibilidade e complexidade. É importante equilibrar o uso da herança com outras técnicas de design para criar um código robusto e flexível.

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

Ela marca uma classe como sendo capaz de ser convertida em uma sequência de bytes. Essa sequência pode ser utilizada para converter para um formato que pode ser armazenado ou transmitido e reconstruir o objeto a partir da sequência de bytes.

Em resumo é necessária ao efetuar persistência em arquivos binários para indicar que uma classe pode ser convertida em bytes e restaurada posteriormente.

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

É utilizado principalmente para realizar operações de maneira mais declarativa e flexível sobre coleções de dados. Ela introduz operações de alto nível que permitem manipular e processar dados de forma concisa e eficiente, aplicando algumas funcionalidades.

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

Não existe um padrão fixo, mas os padrões mais comuns incluem: Leitura e gravação de arquivos de texto e/ou binários; formatos de dados específicos; serialização e bibliotecas de persistência de terceiros.

2º Procedimento – Criação do Cadastro e Modo Texto

PessoaFisicaRepo.java

```
package CadastroPOO.Procedimento2.gerenciadores;

import CadastroPOO.Procedimento2.model.PessoaFisica;

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

public class PessoaFisicaRepo {
    private ArrayList<PessoaFisica> pessoasFisicas;

    public PessoaFisicaRepo() {
        pessoasFisicas = new ArrayList<>();
    }

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

    public void alterar(PessoaFisica pessoaFisica, String novoNome,
String novoCpf, int novaIdade) {
        pessoaFisica.setNome(novoNome);
        pessoaFisica.setCpf(novoCpf);
        pessoaFisica.setIdade(novaIdade);
    }

    public void excluir(int id) {
        pessoasFisicas.remove(obter(id));
    }

    public PessoaFisica obter(int id) throws NoSuchElementException {
        Optional<PessoaFisica> pessoaFisicaEncontrada =
pessoasFisicas.stream().
                filter(pessoaFisica -> pessoaFisica.getId() == id)
                .findFirst();
```

```
        if (pessoaFisicaEncontrada.isPresent()) {
            return pessoaFisicaEncontrada.get();
        } else {
            throw new NoSuchElementException("Pessoa física com ID " + id
+ " não encontrada.");
        }
    }

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

    public void persistir(String nomeArquivo) throws IOException {
        ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(nomeArquivo));
        outputStream.writeObject(pessoasFisicas);
        outputStream.close();
        System.out.println("Dados da pessoa física armazenados.");
        System.out.println();
    }

    public void recuperar(String nomeArquivo) throws IOException,
ClassNotFoundException {
        ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(nomeArquivo));
        pessoasFisicas = (ArrayList<PessoaFisica>)
inputStream.readObject();
        inputStream.close();
        System.out.println("Dados da pessoa física recuperados.");
        System.out.println();
    }
}
```

PessoaJuridicaRepo.java

```
package CadastroPOO.Procedimento2.gerenciadores;

import CadastroPOO.Procedimento2.model.PessoaJuridica;

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

public class PessoaJuridicaRepo {
    private ArrayList<PessoaJuridica> pessoasJuridicas;

    public PessoaJuridicaRepo() {
        pessoasJuridicas = new ArrayList<>();
```

```
    }

    public void inserir(PessoaJuridica pessoaJuridica) {
        pessoasJuridicas.add(pessoaJuridica);
    }

    public void alterar(PessoaJuridica pessoaJuridica, String novoNome,
String novoCpf) {
        pessoaJuridica.setNome(novoNome);
        pessoaJuridica.setCnpj(novoCpf);
    }

    public void excluir(int id) {
        pessoasJuridicas.remove(obter(id));
    }

    public PessoaJuridica obter(int id) throws NoSuchElementException {
        Optional<PessoaJuridica> pessoaJuridicaEncontrada =
pessoasJuridicas.stream().
                filter(pessoaJuridica -> pessoaJuridica.getId() == id)
                .findFirst();

        if (pessoaJuridicaEncontrada.isPresent()) {
            return pessoaJuridicaEncontrada.get();
        } else {
            throw new NoSuchElementException("Pessoa jurídica com ID " +
id + " não encontrada.");
        }
    }

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

    public void persistir(String nomeArquivo) throws IOException {
        ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(nomeArquivo));
        outputStream.writeObject(pessoasJuridicas);
        outputStream.close();
        System.out.println("Dados da pessoa jurídica armazenados.");
        System.out.println();
    }

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

```
        inputStream.close();
        System.out.println("Dados da pessoa jurídica recuperados.");
        System.out.println();
    }
}
```

Pessoa.java

```
package CadastroPOO.Procedimento2.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.java

```
package CadastroPOO.Procedimento2.model;
```

```java
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);
    }
}
```

PessoaJuridica.java

```java
package CadastroPOO.Procedimento2.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);
    }
}
```

App.java

```
package CadastroPOO.Procedimento2;



import CadastroPOO.Procedimento2.model.PessoaFisica;
import CadastroPOO.Procedimento2.model.PessoaJuridica;
import CadastroPOO.Procedimento2.gerenciadores.PessoaFisicaRepo;
import CadastroPOO.Procedimento2.gerenciadores.PessoaJuridicaRepo;
import java.io.IOException;
import java.util.Scanner;

public class App {

    public static void clearConsole() {
        try {
            if (System.getProperty("os.name").contains("Windows")) {
                new ProcessBuilder("cmd", "/c",
"cls").inheritIO().start().waitFor();
            } else {
                System.out.print("\033[H\033[2J");
                System.out.flush();
            }
        } catch (Exception e) {
        }
    }
```

```
    public static void main(String[] args) {

        PessoaFisicaRepo pessoaFisicaRepo = new PessoaFisicaRepo();
        PessoaJuridicaRepo pessoaJuridicaRepo = new PessoaJuridicaRepo();

        Scanner scanner = new Scanner(System.in);
        String opcaoSelecionada;

        do {
            System.out.println("=============================");
            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 - Sair");
            System.out.println("=============================");

            opcaoSelecionada = scanner.next();

            switch (opcaoSelecionada) {

                // Incluir Pessoa
                case "1":
                    do {

System.out.println("=============================");
                        System.out.println("F - Incluir Pessoa Física");
                        System.out.println("J - Incluir Pessoa
Jurídica");
                        System.out.println("V - Voltar ao Menu
Principal");

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

                        opcaoSelecionada = scanner.next();
                        scanner.nextLine();

                        switch (opcaoSelecionada.toUpperCase()) {

                            // Inserir pessoa fisica
                            case "F":
                                System.out.println("Insira o nome da
pessoa física: ");
                                String nome = scanner.nextLine();
```

```
                                System.out.println("Insira o cpf da
pessoa física: ");
                                String cpf = scanner.nextLine();
                                System.out.println("Insira a idade da
pessoa física: ");
                                int idade = scanner.nextInt();

                                int pessoaFisicaTamanhoRepo =
pessoaFisicaRepo.obterTodos().size();
                                int pessoaFisicaIdSerie =
pessoaFisicaTamanhoRepo > 0 ? pessoaFisicaRepo.obterTodos()
                                        .get(pessoaFisicaTamanhoRepo -
1).getId() + 1 : 1;

                                PessoaFisica pessoaFisica = new
PessoaFisica(pessoaFisicaIdSerie, nome, cpf, idade);

                                pessoaFisicaRepo.inserir(pessoaFisica);

                                System.out.println("Pessoa física
inserida com sucesso!");
                                pessoaFisica.exibir();
                                break;

                            // Incluir Pessoa Juridica
                            case "J":
                                System.out.println("Insira o nome da
pessoa juridica: ");
                                nome = scanner.nextLine();
                                System.out.println("Insira o cnpj da
pessoa juridica: ");
                                String cnpj = scanner.nextLine();

                                int pessoaJuridicaTamanhoRepo =
pessoaJuridicaRepo.obterTodos().size();
                                int pessoaJuridicaIdSerie =
pessoaJuridicaTamanhoRepo > 0 ? pessoaJuridicaRepo.obterTodos()
                                        .get(pessoaJuridicaTamanhoRepo -
1).getId() + 1 : 1;

                                PessoaJuridica pessoaJuridica = new
PessoaJuridica(pessoaJuridicaIdSerie, nome, cnpj);


pessoaJuridicaRepo.inserir(pessoaJuridica);

                                System.out.println("Pessoa juridica
inserida com sucesso!");
                                pessoaJuridica.exibir();
```

```
                                break;

                            // Opcao invalida
                            default:
                                System.out.println("Opção inválida. Por
favor, selecione uma opção válida.");
                                break;
                        }

                        // Voltar para o menu principal
                    } while (!opcaoSelecionada.equalsIgnoreCase("V"));
                    break;

                // Alterar Pessoa
                case "2":
                    do {

System.out.println("============================");
                        System.out.println("F - Alterar pessoa física por
ID");
                        System.out.println("J - Alterar pessoa juridica
por ID");
                        System.out.println("V - Voltar ao Menu
Principal");

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

                        opcaoSelecionada = scanner.next();
                        scanner.nextLine();

                        switch (opcaoSelecionada.toUpperCase()) {

                            // Alterar Pessoa Fisica
                            case "F":
                                System.out.println("Insira o ID da pessoa
física: ");
                                int idPessoaFisica = scanner.nextInt();
                                scanner.nextLine();

                                PessoaFisica pessoaFisicaObtida =
pessoaFisicaRepo.obter(idPessoaFisica);

                                if (pessoaFisicaObtida != null) {
                                    pessoaFisicaObtida.exibir();

                                    System.out.println("Nome atual da
pessoa física: " + pessoaFisicaObtida.getNome());
                                    System.out.println("Insira um novo
nome: ");
```

```
                                    String novoNome = scanner.nextLine();

                                    System.out.println("CPF atual da
pessoa física: " + pessoaFisicaObtida.getCpf());
                                    System.out.println("Insira um novo
CPF: ");
                                    String novoCPF = scanner.nextLine();

                                    System.out.println("Idade atual da
pessoa física: " + pessoaFisicaObtida.getCpf());
                                    System.out.println("Insira uma nova
Idade: ");
                                    int novaIdade = scanner.nextInt();


pessoaFisicaRepo.alterar(pessoaFisicaObtida, novoNome, novoCPF,
novaIdade);

                                    System.out.println("Pessoa fìsica
alterada com sucesso!");
                                } else
                                    System.out.println("Pessoa física com
este ID não foi encontrada! Tente novamente.");
                                break;

                            // Alterar Pessoa Juridica
                            case "J":
                                System.out.println("Insira o ID da pessoa
juridica: ");
                                int idPessoaJuridica = scanner.nextInt();
                                scanner.nextLine();

                                PessoaJuridica pessoaJuridicaObtida =
pessoaJuridicaRepo.obter(idPessoaJuridica);

                                if (pessoaJuridicaObtida != null) {
                                    pessoaJuridicaObtida.exibir();

                                    System.out.println("Nome atual da
pessoa juridica: " + pessoaJuridicaObtida.getNome());
                                    System.out.println("Insira um novo
nome: ");
                                    String novoNome = scanner.nextLine();

                                    System.out.println("CNPJ atual da
pessoa juridica: " + pessoaJuridicaObtida.getCnpj());
                                    System.out.println("Insira um novo
CNPJ: ");
                                    String novoCNPJ = scanner.nextLine();
```

```
pessoaJuridicaRepo.alterar(pessoaJuridicaObtida, novoNome, novoCNPJ);

                                    System.out.println("Pessoa juridica
alterada com sucesso!");
                                } else
                                    System.out.println("Pessoa juridica
com este ID não foi encontrada! Tente novamente.");
                                break;

                            // Opcao invalida
                            default:
                                System.out.println("Opção inválida. Por
favor, selecione uma opção válida.");
                                break;
                        }

                        // Voltar para o menu principal
                    } while (!opcaoSelecionada.equalsIgnoreCase("V"));
                    break;

                // Excluir Pessoa
                case "3":
                    do {

System.out.println("============================");
                        System.out.println("F - Excluir pessoa física por
ID");
                        System.out.println("J - Excluir pessoa juridica
por ID");
                        System.out.println("V - Voltar ao Menu
Principal");

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

                        opcaoSelecionada = scanner.next();
                        scanner.nextLine();

                        switch (opcaoSelecionada.toUpperCase()) {

                            // Excluindo pessoa fisica
                            case "F":
                                System.out.println("Insira o ID da pessoa
física: ");
                                int idPessoaFisica = scanner.nextInt();

                                PessoaFisica pessoaFisicaObtida =
pessoaFisicaRepo.obter(idPessoaFisica);
```

```
                                if (pessoaFisicaObtida != null) {
                                    pessoaFisicaObtida.exibir();

pessoaFisicaRepo.excluir(idPessoaFisica);

                                    System.out.println("Pessoa fìsica
excluida com sucesso!");
                                } else
                                    System.out.println("Pessoa física com
este ID não foi encontrada! Tente novamente.");
                                break;

                            // Excluindo pessoa juridica
                            case "J":
                                System.out.println("Insira o ID da pessoa
juridica: ");
                                int idPessoaJuridica = scanner.nextInt();

                                PessoaJuridica pessoaJuridicaObtida =
pessoaJuridicaRepo.obter(idPessoaJuridica);

                                if (pessoaJuridicaObtida != null) {
                                    pessoaJuridicaObtida.exibir();

pessoaJuridicaRepo.excluir(idPessoaJuridica);

                                    System.out.println("Pessoa fìsica
excluida com sucesso!");
                                } else
                                    System.out.println("Pessoa juridica
com este ID não foi encontrada! Tente novamente.");
                                break;

                            // Opcao invalida
                            default:
                                System.out.println("Opção inválida. Por
favor, selecione uma opção válida.");
                                break;
                        }

                        // Voltar para o menu principal
                    } while (!opcaoSelecionada.equalsIgnoreCase("V"));
                    break;

                // BUSCAR PESSOA PELO ID
                case "4":
```

```
                    do {

System.out.println("============================");
                        System.out.println("F - Obter dados da Pessoa
física por ID");
                        System.out.println("J - Obter dados da pessoa
juridica por ID");
                        System.out.println("V - Voltar ao Menu
Principal");

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

                        opcaoSelecionada = scanner.next();
                        scanner.nextLine();

                        switch (opcaoSelecionada.toUpperCase()) {

                            // Buscando pessoa fisica pelo id
                            case "F":
                                System.out.println("Insira o ID da pessoa
física: ");
                                int idPessoaFisica = scanner.nextInt();

                                PessoaFisica pessoaFisicaObtida =
pessoaFisicaRepo.obter(idPessoaFisica);

                                if (pessoaFisicaObtida != null) {
                                    System.out.println("Pessoa fisica
encontrada!");


                                    pessoaFisicaObtida.exibir();
                                } else
                                    System.out.println("Pessoa física com
este ID não foi encontrada! Tente novamente.");
                                break;

                            // Buscandi pessoa juridica pelo id
                            case "J":
                                System.out.println("Insira o ID da pessoa
juridica: ");
                                int idPessoaJuridica = scanner.nextInt();

                                PessoaJuridica pessoaJuridicaObtida =
pessoaJuridicaRepo.obter(idPessoaJuridica);

                                if (pessoaJuridicaObtida != null) {
                                    System.out.println("Pessoa juridica
encontrada!");
```

```
                                    pessoaJuridicaObtida.exibir();
                                } else
                                    System.out.println("Pessoa juridica
com este ID não foi encontrada! Tente novamente.");
                                break;

                            // Opcao invalida
                            default:
                                System.out.println("Opção inválida. Por
favor, selecione uma opção válida.");
                                break;
                        }

                        // Voltar para o menu principal
                    } while (!opcaoSelecionada.equalsIgnoreCase("V"));
                    break;

                //BUSCAR TODAS AS PESSOAS
                case "5":
                    do {

System.out.println("==============================");
                        System.out.println("F - Obter lista de Pessoas
Físicas");
                        System.out.println("J - Obter lista de Pessoas
Jurídicas");
                        System.out.println("V - Voltar ao Menu
Principal");

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

                        opcaoSelecionada = scanner.next();
                        scanner.nextLine();

                        switch (opcaoSelecionada.toUpperCase()) {

                            // Buscar todas as pessoa fisicas
                            case "F":
                                System.out.println("Lista de pessoas
fìsicas: ");
                                pessoaFisicaRepo.obterTodos()
                                        .forEach(pessoaFisica -> {
                                            pessoaFisica.exibir();
                                            System.out.println();
                                        });
                                break;

                            // Buscar todas as pessoas juridicas
                            case "J":
```

```
                                System.out.println("Lista de pessoas
juridicas: ");
                                pessoaJuridicaRepo.obterTodos()
                                        .forEach(pessoaJuridica -> {
                                            pessoaJuridica.exibir();
                                            System.out.println();
                                        });
                                break;

                            // Opcao invalida
                            default:
                                System.out.println("Opção inválida. Por
favor, selecione uma opção válida.");
                                break;
                        }

                        // Voltar para o menu principal
                    } while (!opcaoSelecionada.equalsIgnoreCase("V"));
                    break;

                // PERSISTIR OS DADOS
                case "6":
                    try {

pessoaFisicaRepo.persistir("pessoas_fisicas.dat");

pessoaJuridicaRepo.persistir("pessoas_juridicas.dat");
                    } catch (IOException erro) {
                        System.out.println("Erro ao persistir ou
recuperar os dados: " + erro.getMessage());
                    }
                    break;

                //RECUPERAR OS DADOS
                case "7":
                    try {

pessoaFisicaRepo.recuperar("pessoas_fisicas.dat");

pessoaJuridicaRepo.recuperar("pessoas_juridicas.dat");
                    } catch (ClassNotFoundException | IOException erro) {
                        System.out.println("Erro ao persistir ou
recuperar os dados: " + erro.getMessage());
                    }
                    break;

                // Fechar a aplicacao
                case "0":
```

```
                    System.out.println("Aplicação finalizada com
sucesso!");
                    System.exit(0);

                    scanner.close();

                // Opcao invalida
                default:
                    System.out.println("Opção inválida. Por favor,
selecione uma opção válida.");
                    break;

            }

        } while (!opcaoSelecionada.equals("0"));

        scanner.close();
    }
}
```

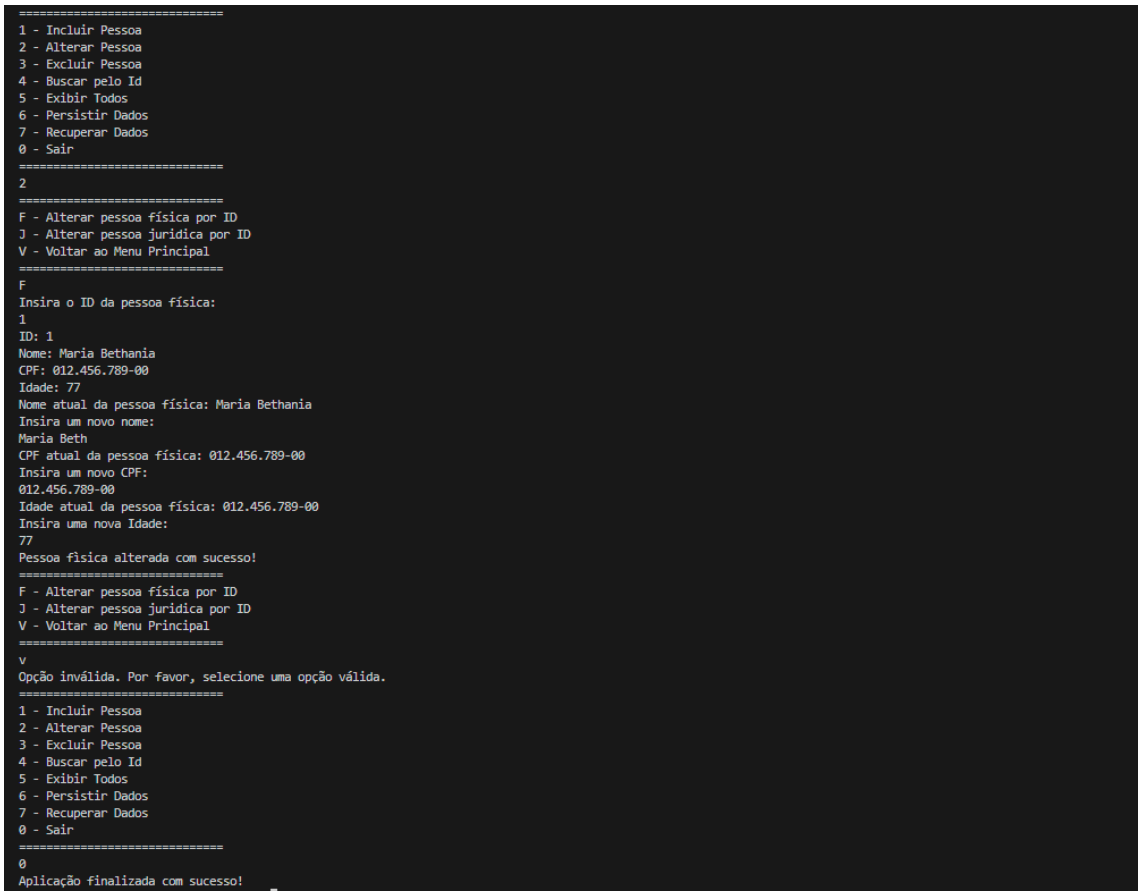
RESULTADO DÁ PRÁTICA / PROCEDIMENTO 2

```
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Sair
============================
1
============================
F - Incluir Pessoa Física
J - Incluir Pessoa Jurídica
V - Voltar ao Menu Principal
============================
f
Insira o nome da pessoa física:
Maria Bethania
Insira o cpf da pessoa física:
012.456.789-00
Insira a idade da pessoa física:
77
Pessoa física inserida com sucesso!
ID: 1
Nome: Maria Bethania
CPF: 012.456.789-00
Idade: 77
============================
F - Incluir Pessoa Física
J - Incluir Pessoa Jurídica
V - Voltar ao Menu Principal
============================
j
Insira o nome da pessoa juridica:
GOOGLE LLC
Insira o cnpj da pessoa juridica:
00.000.000/0001-00
Pessoa juridica inserida com sucesso!
ID: 1
Nome: GOOGLE LLC
CNPJ: 00.000.000/0001-00
============================
F - Incluir Pessoa Física
J - Incluir Pessoa Jurídica
V - Voltar ao Menu Principal
============================
```

```
============================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Sair
============================
5
============================
F - Obter lista de Pessoas Físicas
J - Obter lista de Pessoas Jurídicas
V - Voltar ao Menu Principal
============================
F
Lista de pessoas físicas:
ID: 1
Nome: Maria Bethania
CPF: 012.456.789-00
Idade: 77

============================
F - Obter lista de Pessoas Físicas
J - Obter lista de Pessoas Jurídicas
V - Voltar ao Menu Principal
============================
J
Lista de pessoas juridicas:
ID: 1
Nome: GOOGLE LLC
CNPJ: 00.000.000/0001-00

============================
F - Obter lista de Pessoas Físicas
J - Obter lista de Pessoas Jurídicas
V - Voltar ao Menu Principal
============================
```



Análise e Conclusão

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

Eles referem-se a membros de uma classe que pertencem à própria classe em vez de pertencerem a instâncias individuais dessa classe. Isso inclui variáveis estáticas (ou campos estáticos) e métodos estáticos.

O método main é estático porque é o ponto de entrada para um programa Java. Quando um programa é executado, ele procura o método main com a assinatura padrão para começar a execução. A declaração do método main como estático permite que ele seja invocado sem a necessidade de criar uma instância da classe, pois ele pertence à classe em si e não a um objeto específico.

Por ser estático, o método main pode ser chamado para iniciar a execução do programa, sem a necessidade de criar uma instância da classe que o contém. Isso é crucial, pois permite que o Java inicie a execução sem criar objetos e sem a necessidade de um contexto de instância.

b. Para que serve a classe Scanner?

é uma ferramenta útil para ler dados de entrada de diferentes fontes, como entrada do teclado, arquivos ou outras fontes de entrada. Ela oferece métodos para analisar e extrair diferentes tipos de dados, como inteiros, doubles, strings e outros, facilitando a leitura de informações fornecidas pelo usuário ou por algum arquivo.

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

Elas permitem isolar a lógica de acesso a dados seguindo o princípio de responsabilidade única, onde cada classe tem uma única responsabilidade bem definida. Além disso, a separação da lógica de acesso a dados também permite uma melhor reutilização de código, pois outras partes do programa podem utilizar a mesma classe de repositório para acessar os dados de forma consistente. Dessa forma, o uso de classe auxilia diretamente em toda a organização, execução, reutilização e até mesmo, se necessário, uma posterior alteração ou adaptação no código.