Desenvolvimento FullStack - Missão Prática Nivel1 Mundo 3 – Fernanda Macagnan de França

- **GitHub:** https://github.com/FerMacagnan/Desenvolvimento-FullStack---Miss-o-Pr-tica-Nivel1-Mundo3---Fernanda-Macagnan.git

Projeto

Cadastro de Clientes em Java

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

- 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.

Aplicações

▼ **Procedimento 1** ✍ Criação das Entidades e Sistema de Persistência

Estrutura de arquivos

```
CadastroPOO/
├── bin/
├── lib/
└── src/
    ├── App.java
    └── model/
        ├── Pessoa.java
        ├── PessoaFisica.java
        ├── PessoaJuridica.Java
        ├── PessoaFisicaRepo.java
        └── PessoaJuridicaRepo.java
```

Códigos

▼ App.java

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

```
public class App {
    public static void main(String[] args) {
        try {
            PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
            repo1.inserir(new PessoaFisica(1, "Ana", "11111111111", 25));
            repo1.inserir(new PessoaFisica(2, "Carlos", "22222222222", 52));
            repo1.persistir("pessoasFisicas.dat");
            System.out.println("Dados de Pessoa Fisica Armazenados.");

            PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
            repo2.recuperar("pessoasFisicas.dat");
            System.out.println("Dados de Pessoa Fisica Recuperados.");
            for (PessoaFisica pf : repo2.obterTodos()) {
                pf.exibir();
            }

            PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
            repo3.inserir(new PessoaJuridica(3, "XPTO Sales", "33333333333333"));
            repo3.inserir(new PessoaJuridica(4, "XPTO Solutions", "44444444444444"));
            repo3.persistir("pessoasJuridicas.dat");
            System.out.println("Dados de Pessoa Juridica Armazenados.");

            PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
            repo4.recuperar("pessoasJuridicas.dat");
            System.out.println("Dados de Pessoa Juridica Recuperados.");
            for (PessoaJuridica pj : repo4.obterTodos()) {
                pj.exibir();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

▼ 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 void exibir() {
        System.out.println("ID: " + id + ", 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;
    }
}
```

▼ PessoaFisica.java

```
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 + ", 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;
    }
}
```

▼ PessoaJuridica.Java

```
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);
    }

    public String getCnpj() {
        return cnpj;
    }

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

▼ PessoaFisicaRepo.java

```
package model;
```

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

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

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

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

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

    public PessoaFisica obter(int id) {
        return pessoasFisicas.stream()
                .filter(p -> p.getId() == id)
                .findFirst()
                .orElse(null);
    }

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

▼ PessoaJuridicaRepo.java

```
package model;

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

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

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

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

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

```
    }

    public PessoaFisica obter(int id) {
        return pessoasFisicas.stream()
                .filter(p -> p.getId() == id)
                .findFirst()
                .orElse(null);
    }

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

Execuções

```
Dados de Pessoa Fisica Armazenados.
Dados de Pessoa Fisica Recuperados.
ID: 1, Nome: Ana
CPF: 11111111111, Idade: 25
ID: 2, Nome: Carlos
CPF: 22222222222, Idade: 52
Dados de Pessoa Juridica Armazenados.
Dados de Pessoa Juridica Recuperados.
ID: 3, Nome: XPTO Sales
CNPJ: 33333333333333
ID: 4, Nome: XPTO Solutions
CNPJ: 44444444444444
```

Análise e Conclusões

Quais as vantagens e desvantagens do uso de herança?

- Vantagens:
  - Permite reutilizar código fazendo o reaproveitamento de métodos e variáveis em diversos contextos, reduzindo a duplicação do códgo
  - Facilita a organização e manutenção do código
  - Permite que objetos de classes diferentes sejam tratados como o mesmo objetivo de uma superclasse, o que facilita a trabalhar com varios tipos de objetos ao mesmo tempo
- Desvantagens
  - Java não suporta herança multipla, o que limita o uso de uma unica superclasse por subclasse
  - Em Sistemas mais complexos é dificil de modificar a estrutura

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

A serialização é uma forma de avisar a JVM que aquela informações podem ser salvam em binário e assim convertendo o estado do objeto em um fluxo de bytes, que podem ser armazenados em um arquivo.
Essa prática é ideal por aumentar a performance de transferencia de dados e definir um padrao de persistencia que pode ser lidos por diferentes plataformas, versões e arquiteturas.
E todo arquivo serializado pode ser deserializado para leitura dinamica das informações

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

Essa fuga do paradigma do POO em Java na API Stream é utilizado para declarar de forma mais pratica toda manipulação de dados sequencial, como filter, reduce, map. Permitindo uma performance maior, sem complexidade e sem modificar os dados dos objetos originais, agindo em paralelo ao funcionamento padrão do programa.

No entanto ela ainda opera sobre objetos e classes, processando os dados de maneira funcional dentro da orientação por objetos de Java.

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

A persistencia de dados em arquivos no desenvolvimento com Java é o DAO. Que separa a lógica de acesso aos dados da lógica de negócios da aplicação.

▼ **Procedimento 2** ✍ Criação do Cadastro em Modo Texto

Estrutura de arquivos

```
CadastroPOO/
├── bin/
├── lib/
└── src/
    ├── App.java
    └── model/
        ├── Pessoa.java
        ├── PessoaFisica.java
        ├── PessoaJuridica.Java
        ├── PessoaFisicaRepo.java
        └── PessoaJuridicaRepo.java
```

Códigos

▼ App.java

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

import java.io.IOException;
import java.util.Scanner;

public class App {
    private static PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
    private static PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();
    private static Scanner scanner = new Scanner(System.in);

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

        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 - Finalizar Programa");
            System.out.println("==================================");
            System.out.print("Escolha uma opção: ");
            opcao = scanner.nextInt();
            scanner.nextLine();

            switch (opcao) {
                case 1:
                    incluirPessoa();
                    break;
                case 2:
                    alterarPessoa();
                    break;
                case 3:
                    excluirPessoa();
                    break;
                case 4:
                    buscarPeloId();
                    break;
                case 5:
```

```
                    exibirTodos();
                    break;
                case 6:
                    persistirDados();
                    break;
                case 7:
                    recuperarDados();
                    break;
                case 0:
                    System.out.println("Finalizando programa...");
                    break;
                default:
                    System.out.println("Opção inválida. Tente novamente.");
            }
        } while (opcao != 0);

        scanner.close();
    }

    private static void excluirPessoa() {...}

    private static void buscarPeloId() {...}

    private static void exibirTodos() {...}

    private static void persistirDados() {...}

    private static void recuperarDados() {...}

    private static void incluirPessoa() {...}

    private static void alterarPessoa() {...}

}
```

▼ excluirPessoa()

```
private static void excluirPessoa() {
        System.out.println("Excluir Pessoa: [F] - Física | [J] - Jurídica");
        String tipo = scanner.nextLine().toUpperCase();
        System.out.print("Digite o id da pessoa: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        if (tipo.equals("F")) {
            repoFisica.excluir(id);
            System.out.println("Pessoa Física excluída.");
        } else if (tipo.equals("J")) {
            repoJuridica.excluir(id);
            System.out.println("Pessoa Jurídica excluída.");
        } else {
            System.out.println("Tipo inválido.");
        }
    }
```

▼ buscarPeloId()

```
private static void buscarPeloId() {
        System.out.println("Buscar Pessoa pelo ID: [F] - Física | [J] - Jurídica");
        String tipo = scanner.nextLine().toUpperCase();
        System.out.print("Digite o id da pessoa: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        if (tipo.equals("F")) {
            PessoaFisica pf = repoFisica.obter(id);
            if (pf != null) {
                pf.exibir();
            } else {
                System.out.println("Pessoa Física não encontrada.");
            }
        } else if (tipo.equals("J")) {
            PessoaJuridica pj = repoJuridica.obter(id);
            if (pj != null) {
```

```
                pj.exibir();
            } else {
                System.out.println("Pessoa Jurídica não encontrada.");
            }
        } else {
            System.out.println("Tipo inválido.");
        }
    }
```

▼ exibirTodos()

```
private static void exibirTodos() {
        System.out.println("Exibir Todos: [F] - Física | [J] - Jurídica");
        String tipo = scanner.nextLine().toUpperCase();

        if (tipo.equals("F")) {
            System.out.println("Pessoas Físicas:");
            for (PessoaFisica pf : repoFisica.obterTodos()) {
                pf.exibir();
            }
        } else if (tipo.equals("J")) {
            System.out.println("Pessoas Jurídicas:");
            for (PessoaJuridica pj : repoJuridica.obterTodos()) {
                pj.exibir();
            }
        } else {
            System.out.println("Tipo inválido.");
        }
    }
```

▼ persistirDados()

```
private static void persistirDados() {
        System.out.print("Digite o prefixo para os arquivos de persistência: ");
        String prefixo = scanner.nextLine();
        String arquivoFisica = prefixo + ".fisica.bin";
        String arquivoJuridica = prefixo + ".juridica.bin";

        try {
            repoFisica.persistir(arquivoFisica);
            repoJuridica.persistir(arquivoJuridica);
            System.out.println("Dados persistidos com sucesso.");
        } catch (IOException e) {
            System.err.println("Erro ao persistir os dados: " + e.getMessage());
        }
    }
```

▼ recuperarDados()

```
private static void recuperarDados() {
        System.out.print("Digite o prefixo para os arquivos de recuperação: ");
        String prefixo = scanner.nextLine();
        String arquivoFisica = prefixo + ".fisica.bin";
        String arquivoJuridica = prefixo + ".juridica.bin";

        try {
            repoFisica.recuperar(arquivoFisica);
            repoJuridica.recuperar(arquivoJuridica);
            System.out.println("Dados recuperados com sucesso.");
        } catch (IOException | ClassNotFoundException e) {
            System.err.println("Erro ao recuperar os dados: " + e.getMessage());
        }
    }
```

▼ incluirPessoa()

```
private static void incluirPessoa() {
        System.out.println("Incluir Pessoa: [F] - Física | [J] - Jurídica");
        String tipo = scanner.nextLine().toUpperCase();

        while (!tipo.equals("F") && !tipo.equals("J")) {
            System.out.println("Tipo inválido. Por favor, digite F para Pessoa Física ou J para Pess
oa Jurídica.");
            System.out.println("Incluir Pessoa: [F] - Física | [J] - Jurídica");
```

```
            tipo = scanner.nextLine().toUpperCase();

        }

        System.out.print("Digite o id da pessoa: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        System.out.print("Digite o nome da pessoa: ");
        String nome = scanner.nextLine();

        if (tipo.equals("F")) {
            System.out.print("Digite o CPF da pessoa: ");
            String cpf = scanner.nextLine();

            System.out.print("Digite a idade da pessoa: ");
            int idade = scanner.nextInt();
            scanner.nextLine();

            PessoaFisica pf = new PessoaFisica(id, nome, cpf, idade);
            repoFisica.inserir(pf);
            System.out.println("Pessoa Física incluída com sucesso.");
        } else {
            System.out.print("Digite o CNPJ da pessoa: ");
            String cnpj = scanner.nextLine();

            PessoaJuridica pj = new PessoaJuridica(id, nome, cnpj);
            repoJuridica.inserir(pj);
            System.out.println("Pessoa Jurídica incluída com sucesso.");
        }
    }
```

▼ alterarPessoa()

```
private static void alterarPessoa() {
        System.out.println("Alterar Pessoa: [F] - Física | [J] - Jurídica");
        String tipo = scanner.nextLine().toUpperCase();

        System.out.print("Digite o id da pessoa para alteração: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        if (tipo.equals("F")) {
            PessoaFisica pf = repoFisica.obter(id);
            if (pf != null) {
                System.out.print("Digite o novo nome da pessoa Física (ou deixe em branco para não a
lterar): ");
                String nome = scanner.nextLine();
                if (!nome.isEmpty())
                    pf.setNome(nome);

                System.out.print("Digite o novo CPF da pessoa Física (ou deixe em branco para não al
terar): ");
                String cpf = scanner.nextLine();
                if (!cpf.isEmpty())
                    pf.setCpf(cpf);

                System.out.print("Digite a nova idade da pessoa Física (ou 0 para não alterar): ");
                int idade = scanner.nextInt();
                if (idade != 0)
                    pf.setIdade(idade);

                repoFisica.alterar(pf);
                System.out.println("Pessoa Física alterada com sucesso.");
            } else {
                System.out.println("Pessoa Física não encontrada.");
            }
        } else if (tipo.equals("J")) {
            PessoaJuridica pj = repoJuridica.obter(id);
            if (pj != null) {
                System.out.print("Digite o novo nome da pessoa Jurídica (ou deixe em branco para não
alterar): ");
                String nome = scanner.nextLine();
                if (!nome.isEmpty())
```

```
                    pj.setNome(nome);

                System.out.print("Digite o novo CNPJ da pessoa Jurídica (ou deixe em branco para não
alterar): ");
                String cnpj = scanner.nextLine();
                if (!cnpj.isEmpty())
                    pj.setCnpj(cnpj);

                repoJuridica.alterar(pj);
                System.out.println("Pessoa Jurídica alterada com sucesso.");
            } else {
                System.out.println("Pessoa Jurídica não encontrada.");
            }
        } else {
            System.out.println("Tipo inválido.");
        }
    }
```

▼ 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 void exibir() {
        System.out.println("ID: " + id + ", 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;
    }
}
```

▼ PessoaFisica.java

```
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 + ", 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;
    }
}
```

▼ PessoaJuridica.Java

```
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);
    }

    public String getCnpj() {
        return cnpj;
    }

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

▼ PessoaFisicaRepo.java

```
package model;

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

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

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

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

```
        }
    }

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

    public PessoaFisica obter(int id) {
        return pessoasFisicas.stream()
                .filter(p -> p.getId() == id)
                .findFirst()
                .orElse(null);
    }

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

▼ PessoaJuridicaRepo.java

```
package model;

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

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

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

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

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

    public PessoaFisica obter(int id) {
        return pessoasFisicas.stream()
                .filter(p -> p.getId() == id)
                .findFirst()
                .orElse(null);
    }

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

Execuções

▼ Todas as funcionalidades

```
==================================
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
==================================
Escolha uma opção: 1
Incluir Pessoa: [F] - Física | [J] - Jurídica
f
Digite o id da pessoa: 01
Digite o nome da pessoa: Maria Silva
Digite o CPF da pessoa: 99999999999
Digite a idade da pessoa: 20
Pessoa Física incluída 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
==================================
Escolha uma opção: 5
Exibir Todos: [F] - Física | [J] - Jurídica
f
Pessoas Físicas:
ID: 1, Nome: Maria Silva
CPF: 99999999999, Idade: 20
==================================
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
==================================
Escolha uma opção: 4
Buscar Pessoa pelo ID: [F] - Física | [J] - Jurídica
f
Digite o id da pessoa: 01
ID: 1, Nome: Maria Silva
CPF: 99999999999, Idade: 20
==================================
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
==================================
Escolha uma opção: 3
Excluir Pessoa: [F] - Física | [J] - Jurídica
f
Digite o id da pessoa: 01
Pessoa Física excluída.
==================================
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
==================================
Escolha uma opção: 5
Exibir Todos: [F] - Física | [J] - Jurídica
f
Pessoas Físicas:
==================================
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
==================================
Escolha uma opção: 1
Incluir Pessoa: [F] - Física | [J] - Jurídica
j
Digite o id da pessoa: 02
Digite o nome da pessoa: Maria Pessoa
JuridicaDigite o CNPJ da pessoa:
00000000000000 Pessoa Jurídica incluída 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
==================================
Escolha uma opção: 6
Digite o prefixo para os arquivos de persistência:
MariaDados persistidos 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
==================================
Escolha uma opção: 5
Exibir Todos: [F] - Física | [J] - Jurídica
j
Pessoas Jurídicas:
ID: 2, Nome: Maria Pessoa Juridica
CNPJ: 00000000000000
==================================
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
==================================
Escolha uma opção: 3
Excluir Pessoa: [F] - Física | [J] - Jurídica
j
Digite o id da pessoa: 02
Pessoa Jurídica excluída.
==================================
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
==================================
Escolha uma opção: 5
Exibir Todos: [F] - Física | [J] - Jurídica
j
Pessoas Jurídicas:
==================================
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
==================================
Escolha uma opção: 7
Digite o prefixo para os arquivos de recuperação: Maria
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
==================================
Escolha uma opção: 5
Exibir Todos: [F] - Física | [J] - Jurídica
j
Pessoas Jurídicas:
ID: 2, Nome: Maria Pessoa Juridica
CNPJ: 00000000000000
==================================
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
==================================
Escolha uma opção: 0
Finalizando programa...
```

Análise e Conclusões

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

Os elementos estáticos pertencem a classe e não a uma instancia especifica da classe. E por isso ele pode ser acessado sem a necessidade de criar uma instancia, por isso o metodo mais é `static`.

Para que serve a classe Scanner?

A classe `Scanner` serve pra ler dados. Independente da fonte (entrada do usuários, arquivos, strings).

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

- Isola a lógica de acesso aos dados da lógica de construção deles.
- Permite facilitar testes, testando de forma independente