CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

MISSÃO PRÁTICA

Nível 1

OBJETIVO

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

CÓDIGOS

A seguir, apresentamos os códigos dos dois procedimentos realizados neste trabalhos.

Códigos de Entidades

Códigos relativos às classes `Pessoa`, `PessoaFisica` e `PessoaJuridica`, responsáveis por estabelecer os estados e comportamentos dessas entidades.

Classe `Pessoa`

```
package model.entidades;

import java.io.Serializable;

public class Pessoa implements Serializable {
    // Atributos
    private int id;
    private String nome;

    // Construtores
    public Pessoa() {}

    public Pessoa(int id, String nome){
        this.id = id;
        this.nome = nome;
    }

    // Getters e setters
    public int getId() {
```

CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

```
        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;
    }

    // Métodos
    public void exibir() {
        System.out.println("ID: " + getId());
        System.out.println("NOME: " + getNome());
    }
}
```

Classe `PessoaFisica`

```
package model.entidades;

import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {
    // Atributos
    private String cpf;
    private int idade;

    // Construtores
    public PessoaFisica() {}

    public PessoaFisica(int id, String nome, String cpf, int idade) {
        super(id, nome);
        this.cpf = cpf;
        this.idade = idade;
    }

    // Getters e setters
    public String getCPF() {
        return this.cpf;
```

CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

```
    }

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

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

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

    @Override
    public void exibir() {
        System.out.println("ID: " + getId());
        System.out.println("NOME: " + getNome());
        System.out.println("CPF: " + getCPF());
        System.out.println("IDADE: " + getIdade());
    }
}
```

Classe `PessoaJuridica`

```
package model.entidades;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {
    // Atributos
    private String cnpj;

    // Construtores
    public PessoaJuridica() {}

    public PessoaJuridica(int id, String nome, String cnpj) {
        super(id, nome);
        this.cnpj = cnpj;
    }

    // Getters e setters
    public String getCNPJ(){
        return this.cnpj;
    }
```

CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

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

    // Métodos
    @Override
    public void exibir() {
        System.out.println("ID: " + getId());
        System.out.println("NOME: " + getNome());
        System.out.println("CNPJ: " + getCNPJ());
    }
}
```

Códigos de Gerenciadores

Códigos relativos às classes `PessoaFisicaRepo` e `PessoaJuridicaRepo`. Tais classes serão responsáveis por lidar com as operações de persistência de dados.

Classe `PessoaFisicaRepo`

```
package model.gerenciadores;


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.List;

import model.entidades.PessoaFisica;


public class PessoaFisicaRepo {

    // Atributos

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

CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

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

public void alterar(PessoaFisica pessoa) {
    pessoasFisicas.set(pessoasFisicas.indexOf(pessoa), pessoa);
}

public void excluir(int id) {
    pessoasFisicas.removeIf(pessoa -> pessoa.getId() == id);
}
public PessoaFisica obter(int id) {
    return pessoasFisicas.stream()
            .filter(pessoa -> pessoa.getId() == id)
            .findFirst()
            .orElse(null);
}
public List<PessoaFisica> obterTodos() {
    // Retorna uma cópia para evitar modificações externas da lista original.
    return new ArrayList<>(pessoasFisicas);
}

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

CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

```
            System.out.println("Dados de Pessoa Fisica Armazenados!");
        }
    }


                public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
                        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            pessoasFisicas = (List<PessoaFisica>) ois.readObject();
            System.out.println("Dados de Pessoa Fisica Recuperados!");
        }
    }
}
```

Classe `PessoaJuridicaRepo`

```
package model.gerenciadores;


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.List;
import model.entidades.PessoaJuridica;


public class PessoaJuridicaRepo {
    // Atributos
```

CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

```
private List<PessoaJuridica> pessoasJuridicas = new ArrayList<>();

// Métodos
public void inserir(PessoaJuridica pessoa) {
    pessoasJuridicas.add(pessoa);
}

public void alterar(PessoaJuridica pessoa) {
    pessoasJuridicas.set(pessoasJuridicas.indexOf(pessoa), pessoa);
}

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

public PessoaJuridica obter(int id) {
    return pessoasJuridicas.stream()
            .filter(pessoa -> pessoa.getId() == id)
            .findFirst()
            .orElse(null);
}

public List<PessoaJuridica> obterTodos() {
    // Retorna uma cópia para evitar modificações externas da lista original.
    return new ArrayList<>(pessoasJuridicas);
}
```

CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

```
    public void persistir(String nomeArquivo) throws IOException {
                        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
            oos.writeObject(pessoasJuridicas);
            System.out.println("Dados de Pessoa Juridica Armazenados!");
        }
    }

                public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
                        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            pessoasJuridicas = (List<PessoaJuridica>) ois.readObject();
            System.out.println("Dados de Pessoa Juridica Recuperados!");
        }
    }
}
```

Classe Principal

Códigos relativos à classe principal do programa: `CadastroP00`. Essa classe contém a função `main`, porta de entrada para a execução do programa.

Aqui temos primeiramente o código do 1º Procedimento e, em seguida, o código do 2º Procedimento.

Classe `CadastroP00`

```
package cadastropoo;


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

CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

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

public class CadastroPOO {
    // As variáveis aqui criadas estarão diposníveis tanto para a função "main" quanto para as demais.

    // Instanciação de objeto Scanner para receber as entradas de usuário.
    private static final Scanner scanner = new Scanner(System.in);

    // Repositórios
    private static final PessoaFisicaRepo repoPF = new PessoaFisicaRepo();
    private static final PessoaJuridicaRepo repoPJ = new PessoaJuridicaRepo();

    public static void main(String[] args) throws IOException, ClassNotFoundException {
        // 1º Procedimento | Criação das Entidades e Sistema de Persistência
        System.out.println("Procedimento 1 | Criacao das Entidades e Sistema de Persistencia\n");

        // Pessoas Físicas
        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
        repo1.inserir(new PessoaFisica(1, "Ana", "111.111.111-11", 25));
        repo1.inserir(new PessoaFisica(2, "Carlos", "222.222.222-22", 52));

        final String NOME_ARQUIVO_PF = "pessoas-fisicas.dat";
```

CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

```
try {
    repo1.persistir(NOME_ARQUIVO_PF);
}
catch (IOException e) {
    System.out.println("Erro ao gravar os dados: " + e.getMessage());
}

PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
try {
    repo2.recuperar(NOME_ARQUIVO_PF);
}
catch (Exception e) {
    System.out.println("Erro ao recuperar dados: " + e.getMessage());
}

List<PessoaFisica> pessoasFisicas = repo2.obterTodos();
for (PessoaFisica pessoa : pessoasFisicas) {
    pessoa.exibir();
}

// Pessoas Jurídicas
PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
repo3.inserir(new PessoaJuridica(3, "XPTO Sales", "33.333.333/3333-33"));
repo3.inserir(new PessoaJuridica(4, "XPTO Solutions", "44.444.444/4444-44"));

final String NOME_ARQUIVO_PJ = "pessoas-juridicas.dat";
```

```
        try {
            repo3.persistir(NOME_ARQUIVO_PJ);
        }
        catch (IOException e) {
            System.out.println("Erro ao gravar os dados: " + e.getMessage());
        }

        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
        try {
            repo4.recuperar(NOME_ARQUIVO_PJ);
        }
        catch (Exception e) {
            System.out.println("Erro ao recuperar dados: " + e.getMessage());
        }

        List<PessoaJuridica> pessoasJuridicas = repo4.obterTodos();
        for (PessoaJuridica pessoa : pessoasJuridicas) {
            pessoa.exibir();
        }

        /*****************************************************************/
        System.out.println("\
n================================================\n");

        // 2º Procedimento | Criação do Cadastro em Modo Texto
        System.out.println("Procedimento 2 | Criacao do Cadastro em Modo Texto\n");

        int opcao;
```

CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

```
    do {
        exibirMenu();
        opcao = scanner.nextInt();

        // Consumindo o caractere de quebra de linha.
        scanner.nextLine();

        processarOpcao(opcao);
    } while (opcao != 0);
    System.out.println("Programa encerrado.");

    scanner.close();
}

// Funções
private static void exibirMenu() {
    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("========================");
```

CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

```
    System.out.print("Escolha uma opcao: ");
}

private static void processarOpcao(int opcao) {
    switch (opcao) {
        case 0 -> {
        }
        case 1 -> inserirPessoa();
        case 2 -> alterarPessoa();
        case 3 -> excluirPessoa();
        case 4 -> buscarPessoa();
        case 5 -> exibirTodos();
        case 6 -> persistirDados();
        case 7 -> recuperarDados();
    }
}

private static void inserirPessoa() {
    System.out.print("Pessoa fisica (F) ou juridica (J)? ");
    char tipo = scanner.nextLine().toUpperCase().charAt(0);

    System.out.print("Id: ");
    int id = scanner.nextInt();
    scanner.nextLine();
    System.out.print("Nome: ");
    String nome = scanner.nextLine();
```

CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

```
    switch (tipo) {
        case 'F' -> {
            System.out.print("CPF: ");
            String cpf = scanner.nextLine();
            System.out.print("Idade: ");
            int idade = scanner.nextInt();
            scanner.nextLine();
            repoPF.inserir(new PessoaFisica(id, nome, cpf, idade));
        }
        case 'J' -> {
            System.out.print("CNPJ: ");
            String cnpj = scanner.nextLine();
            repoPJ.inserir(new PessoaJuridica(id, nome, cnpj));
        }
        default -> System.out.println("Tipo inválido!");
    }
}

private static void alterarPessoa() {
    System.out.print("Pessoa fisica (F) ou juridica (J)? ");
    char tipo = scanner.nextLine().toUpperCase().charAt(0);

    System.out.print("Id: ");
    int id = scanner.nextInt();
    scanner.nextLine();
    System.out.print("Nome: ");
    String nome = scanner.nextLine();
```

CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

```
    switch (tipo) {
        case 'F' -> {
            System.out.print("CPF: ");
            String cpf = scanner.nextLine();
            System.out.print("Idade: ");
            int idade = scanner.nextInt();
            scanner.nextLine();
            repoPF.alterar(new PessoaFisica(id, nome, cpf, idade));
        }
        case 'J' -> {
            System.out.print("CNPJ: ");
            String cnpj = scanner.nextLine();
            repoPJ.alterar(new PessoaJuridica(id, nome, cnpj));
        }
        default -> System.out.println("Tipo inválido!");
    }
}

private static void excluirPessoa() {
    System.out.print("Pessoa fisica (F) ou juridica (J)? ");
    char tipo = scanner.nextLine().toUpperCase().charAt(0);

    System.out.print("Id: ");
    int id = scanner.nextInt();

    switch (tipo) {
```

```
            case 'F' -> repoPF.excluir(id);
            case 'J' -> repoPJ.excluir(id);
            default -> System.out.println("Tipo inválido!");
        }
    }

    private static void buscarPessoa() {
        System.out.print("Pessoa fisica (F) ou juridica (J)? ");
        char tipo = scanner.nextLine().toUpperCase().charAt(0);

        System.out.print("Id: ");
        int id = scanner.nextInt();

        switch (tipo) {
            case 'F' -> repoPF.obter(id);
            case 'J' -> repoPJ.obter(id);
            default -> System.out.println("Tipo inválido!");
        }
    }

    private static void exibirTodos() {
        System.out.print("Pessoa fisica (F) ou juridica (J)? ");
        char tipo = scanner.nextLine().toUpperCase().charAt(0);

        switch (tipo) {
            case 'F' -> {
                List<PessoaFisica> pessoasFisicas = repoPF.obterTodos();
```

CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

```
            for (PessoaFisica pessoa : pessoasFisicas){
                pessoa.exibir();
            }
        }
        case 'J' -> {
            List<PessoaJuridica> pessoasJuridicas = repoPJ.obterTodos();
            for (PessoaJuridica pessoa : pessoasJuridicas) {
                pessoa.exibir();
            }
        }
        default -> System.out.println("Tipo inválido!");
    }
}

private static void persistirDados() {
    System.out.print("Pessoa fisica (F) ou juridica (J)? ");
    char tipo = scanner.nextLine().toUpperCase().charAt(0);

    System.out.print("Digite o prefixo do arquivo: ");
    String prefixo = scanner.nextLine();

    try {
        switch (tipo) {
            case 'F' -> repoPF.persistir(prefixo + ".fisica.bin");
            case 'J' -> repoPJ.persistir(prefixo + ".juridica.bin");
            default -> System.out.println("Tipo inválido!");
        }
```

CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

```
        }
        catch (IOException e) {
            System.out.println("Erro ao persistir os dados: " + e.getMessage());
        }
    }

    private static void recuperarDados() {
        System.out.print("Pessoa fisica (F) ou juridica (J)? ");
        char tipo = scanner.nextLine().toUpperCase().charAt(0);

        System.out.print("Digite o prefixo do arquivo: ");
        String prefixo = scanner.nextLine();

        try {
            switch (tipo) {
                case 'F' -> repoPF.recuperar(prefixo + ".fisica.bin");
                case 'J' -> repoPJ.recuperar(prefixo + ".juridica.bin");
                default -> System.out.println("Tipo inválido!");
            }
        }
        catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao recuperar os dados: " + e.getMessage());
        }
    }
}
```

CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

1º PROCEDIMENTO | CRIAÇÃO DAS ENTIDADES E SISTEMA DE PERSISTÊNCIA

ANÁLISE E CONCLUSÃO

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

Vantagens:

- Reutilização de código: as classes filhas herdam métodos e atributos das classes pai, evitando duplicação de código.
- Hierarquia de classes: organização das classes em uma estrutura hierárquica, facilitando a compreensão e a manutenção do código.
- Polimorfismo: as classes filhas podem sobrescrever métodos da classe pai, permitindo comportamentos específicos.

Desvantagens:

- Acoplamento: a classe filha é fortemente acoplada (depende diretamente) da classe pai, dificultando alterações na hierarquia sem impactos em todo o sistema.
- Complexidade: hierarquias de herança muito profundas podem tornar o código mais complexo e difícil de manter.

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

A interface *Serializable* é necessária para persistir objetos em arquivos binários porque ela permite que o estado do objeto seja transformado em uma sequência de bytes (serialização), que pode ser posteriormente reconstruída (desserializada) a partir do arquivo. Isso possibilita salvar e recuperar objetos Java de forma eficiente.

Sem a implementação da interface *Serializable*, o Java não teria como saber como transformar o estado de um objeto em bytes e vice-versa, impossibilitando a persistência em arquivos binários.

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

CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

O paradigma funcional utilizado pela API Stream no Java permite processar coleções de dados de forma declarativa (expressando o que se deseja fazer, e não como fazer) e concisa (escrevendo menos código).

As principais características do paradigma funcional exploradas pela API Stream são:

- Imutabilidade: os dados originais não são alterados, novos resultados são gerados a partir de suas transformações.
- Funções de alta ordem: alguns métodos (como *map*, *filter* e *reduce*) permitem que funções sejam passadas como argumentos para transformar dados.
- *Laziness*: algumas operações intermediárias são avaliadas apenas quando a operação terminal é executada, otimizando o processamento.
- Paralelismo: muitas operações em *streams* podem ser paralelizadas automaticamente, explorando o poder de processamento multicores.

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

É adotado o *Repository Pattern* (Padrão de Repositório).

Neste padrão, as classes responsáveis pela persistência e recuperação de dados encapsulam a lógica de acesso aos dados, abstraindo a complexidade do armazenamento e fornecendo uma interface simples para as classes de domínio.

No nosso código, as classes *PessoaFisicaRepo* e *PessoaJuridicaRepo* encapsulam a manipulação do *ArrayList* e a persistência dos dados em arquivos binários. Métodos como “inserir”, “alterar”, “excluir” e “obter” implementam as operações CRUD (criar, ler, atualizar e excluir), enquanto “persistir” e “recuperar” gerenciam a persistência em disco.

2º PROCEDIMENTO | CRIAÇÃO DO CADASTRO EM MODO TEXTO

ANÁLISE E CONCLUSÃO

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

CAMPUS: Polo Bezerra de Menezes – Fortaleza/CE
CURSO: Desenvolvimento Full Stack
DISCIPLINA: Iniciando o Caminho pelo Java
TURMA: 9001
SEMESTRE: 2024.4
ALUNO: Daniel Kilzer Brasil Dias

Um elemento estático (métodos ou atributo) pertence à classe e não a uma instância da classe (objeto). Dessa forma, para acessar um elemento estático, não é preciso criar um objeto; esse elemento pode ser acessado diretamente pela classe.

Com relação ao método *main*, este é o ponto de partida de um programa em Java, sendo chamado antes de qualquer instanciação. Assim, sendo estático, o método *main* pode ser chamado diretamente pela classe principal do programa, sem depender da criação de um objeto para isso.

b) Para que serve a classe *Scanner*?

A classe *Scanner* serve para ler uma entrada de dados, seja a entrada padrão (teclado) ou arquivos.

No código apresentado, a classe *Scanner* serve para ler as entradas do usuário através do teclado.

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

O uso das classes de repositório impactou positivamente na organização e manutenibilidade do código. Os benefícios são:

- Responsabilidade única: as classes de repositório encapsulam todas as operações relacionadas ao armazenamento e manipulação de dados (inserir, alterar, excluir, recuperar, etc.), reduzindo a complexidade na lógica principal.

- Separação de responsabilidades: as classes de negócio (como *PessoaFisica* e *PessoaJuridica*) se concentram em representar os dados, enquanto as classes de repositório se concentram em persistir e recuperar esses dados.

- Reutilização: as classes de repositório podem ser reutilizadas em outras partes do programa ou projetos semelhantes, promovendo reaproveitamento de código.

- Escalabilidade: se fosse necessário alterar o tipo de armazenamento (exemplo: de um arquivo binário para um banco de dados), as mudanças seriam limitadas às classes de repositório, sem impactar diretamente a lógica principal.

Destaque-se que essa abordagem segue o padrão de projeto *Repository*, que é uma prática comum em projetos Java para separar a lógica de acesso a dados da lógica de negócios.