Aula 6 mais detalhada

📦 Aula 06 – Conceitos de Orientação a Objetos (OO)

🎯 O que é a Programação Orientada a Objetos (POO)?

POO é um **paradigma de programação** que modela o mundo real usando **objetos**, que possuem **estado** (atributos) e **comportamentos** (métodos).

O foco está em **organizar o código em torno de entidades lógicas**, promovendo reuso, clareza e manutenção.

🧩 Conceitos Fundamentais

🔷 Classe

- É uma **estrutura (molde)** que define atributos (variáveis) e métodos (funções) de um objeto.

```
public class Pessoa {
    String nome;
    int idade;

    void apresentar() {
        System.out.println("Olá, meu nome é " + nome);
    }
}
```

🔶 Objeto

- Uma **instância** de uma classe, ou seja, a classe em uso real.
- Cada objeto tem seu **próprio estado (atributos com valores diferentes)**.

```
Pessoa p1 = new Pessoa();
p1.nome = "Felipe";
p1.idade = 21;
p1.apresentar();
```

📦 Estado da Classe

- O **conjunto de valores atuais dos atributos** de um objeto representa seu **estado**.
- Esse estado pode **mudar** com o tempo, à medida que métodos são executados.

⚙️ Instanciação de Objetos

- Feita usando a palavra-chave `new`.
- Chama o **construtor** da classe para criar o objeto.

```
Pessoa p2 = new Pessoa(); // instanciando a classe Pessoa
```

✍️ Assinatura de um Método

- Conjunto que define **nome, parâmetros e tipo de retorno** do método.

Exemplo de assinatura:

```
public int somar(int a, int b)
```

- Nome: `somar`
- Parâmetros: `(int a, int b)`
- Tipo de retorno: `int`

🔧 Tipos de Métodos

✅ 1. Métodos de Instância

- Chamados em um objeto da classe.
- Podem acessar os atributos e outros métodos da **mesma instância**.

```
public void apresentar() {
    System.out.println("Olá, eu sou " + nome);
}
```

🔁 2. Métodos Estáticos (`static`)

- Pertencem à **classe**, não ao objeto.
- Usados sem precisar instanciar a classe.

```
public static double calcularPI() {
    return 3.1415;
}
```

```
double pi = MinhaClasse.calcularPI();
```

🏗️ 3. Construtores

- São métodos especiais para **criar e inicializar objetos**.
- Têm o **mesmo nome da classe** e **não têm tipo de retorno**.

```
public Pessoa(String nome) {
    this.nome = nome;
}
```

🔐 4. Getters e Setters (Encapsulamento)

```
private int idade;
```

```
public int getIdade() {
    return idade;
}

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

💬 5. Métodos com ou sem retorno

- `void` → não retorna nada
- `int`, `double`, `String` → retornam algum valor

```
public void imprimir() {
    System.out.println("Texto");
}

public int soma(int a, int b) {
    return a + b;
}
```

🧠 Princípios da POO

🧬 1. Abstração

- Foca nos **detalhes importantes** e esconde a complexidade.

🔐 2. Encapsulamento

- **Restringe o acesso direto** aos dados internos e os protege com `getters` e `setters`.

🧬 3. Herança

- Uma classe pode **herdar** de outra para reutilizar comportamentos.

```
class Animal {
    void dormir() {
        System.out.println("Dormindo...");
    }
}

class Cachorro extends Animal {
    void latir() {
        System.out.println("Au au!");
    }
}
```

🔁 4. Polimorfismo

- Permite que um **mesmo método** tenha **diferentes comportamentos**, dependendo da classe que o implementa.

```
class Animal {
    void emitirSom() {
        System.out.println("Som genérico");
    }
}

class Gato extends Animal {
    void emitirSom() {
        System.out.println("Miau");
    }
}
```

📌 Exemplo Completo

```
public class Pessoa {
    private String nome;
```

```
    public Pessoa(String nome) {
        this.nome = nome;
    }

    public void apresentar() {
        System.out.println("Olá, meu nome é " + nome);
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Pessoa p = new Pessoa("Felipe");
        p.apresentar();
    }
}
```