## **Interfaces, Agregação e Composição em Java**

### **Interfaces**

Uma interface em Java define um contrato que uma classe deve cumprir. Ela estabelece um conjunto de métodos que a classe deve implementar, mas não fornece a implementação em si. Essa característica promove o polimorfismo e o desacoplamento entre classes.

**Exemplo:**

Java

interface FormaGeometrica {

double calcularArea();

double calcularPerimetro();

}

### **Agregação**

A agregação representa um relacionamento "tem-um" entre classes. Um objeto de uma classe possui uma referência a um objeto de outra classe, mas a existência de um não depende da existência do outro.

**Exemplo:**

Java

class Carro {

private Motor motor;

}

### **Composição**

A composição é um tipo mais forte de agregação. Um objeto de uma classe "é composto por" um ou mais objetos de outra classe. A vida dos objetos está fortemente ligada: se o objeto composto for destruído, seus componentes também serão.

**Exemplo:**

Java

class Carro {

private Roda[] rodas;

}

### **Diferenças entre Agregação e Composição**

| **Característica** | **Agregação** | **Composição** |
| --- | --- | --- |
| Vida dos objetos | Objetos podem existir independentemente. | Objetos dependem do objeto composto. |
| Relação | "Tem-um" | "É composto por" |

### **Quando usar cada um?**

* **Interfaces:** Para definir contratos e promover polimorfismo.
* **Agregação:** Quando um objeto possui uma referência a outro objeto, mas eles podem existir independentemente.
* **Composição:** Quando um objeto é composto por outros objetos e a vida deles está fortemente ligada.

### **Exemplo Completo**

Java

interface Animal {

void fazerBarulho();

}

class Cachorro implements Animal {

public void fazerBarulho() {

System.out.println("Au au!");

}

}

class Pessoa {

private Animal animalDeEstimacao; // Agregação

}

class Casa {

private Quarto[] quartos; // Composição

}