Associação, Agregação, Composição - Diferenças

Mário Leite

Na Programação Orientada a Objetos, classes raramente existem isoladas. Na prática, sistemas reais são formados por objetos que relacionam entre si, cooperando para resolver problemas do mundo real. Por isto, entender **como as classes se relacionam** é tão importante quanto saber criar métodos e atributos; então três conceitos fundamentais entram em cena: **Associação, Agregação e Composição**. E, embora muitas vezes confundidos, eles representam níveis diferentes de vínculo, dependência e acoplamento - e impactam diretamente a qualidade do projeto, a manutenção do código e a robustez da aplicação.

1 - Associação: apenas uso

A Associação indica que uma classe **usa** a outra; não há posse, nem dependência estrutural.
Exemplo**: Professor** e **Disciplina**
Uma usa o outra, mas ambos existem separadamente.
É uma relação de **uso**.
Vantagem**:** baixo acoplamento.
Desvantagem**:** não expressa estrutura.

2 - Agregação: "tem um", mas não depende

É uma relação **todo-parte fraca**.
O TODO contém a parte, mas a parte **existe sem o todo**.
Exemplo**: Departamento** e **Funcionário**
O departamento tem funcionários, mas o funcionário existe fora dele.
É uma relação de **posse leve**.
Vantagem**:** boa para estruturas organizacionais.
Desvantagem**:** mais acoplamento que associação.

3 - Composição: dependência total

A Composição é relação **todo-parte forte**. A parte **não existe sem o todo**.
Exemplo**: Pedido** e **Item do Pedido**
Sem pedido, não há item.
É uma relação de **dependência estrutural**.
Vantagem**:** garante integridade.
Desvantagem**:** alto acoplamento.

- **Associação** ==> usa
- **Agregação** ==> tem
- **Composição** ==> é parte de

Assim, podemos concluir:
Associação é convivência.
Agregação é companhia.
Composição é dependência.

Veja os exemplos em pseudocódigo...

Associação ==> “tem um”, mas não depende

```
Tipo Professor
   nome: cadeia;
FimTipo

Tipo Disciplina
   nome: cadeia;
FimTipo

Func Principal()
   Var p: Professor;
   Var d: Disciplina;

   p.nome <- "Carlos";
   d.nome <- "Algoritmos";

   EscrevaLn(p.nome, " ministra ", d.nome);
   Retorne vazio;
FimFunc
```

Professor usa Disciplina; nenhum depende do outro.

==========

Agregação ==> “tem um”, mas não depende

```
Tipo Funcionario
    nome: cadeia;
FimTipo

Tipo Departamento
    nome: cadeia;
    funcionarios: Vetor[10] de Funcionario;
FimTipo

Func Principal()
    Var dep: Departamento;
    dep.nome <- "TI";
    dep.funcionarios[1].nome <- "Ana";
    dep.funcionarios[2].nome <- "João";
    Retorne vazio;
FimFunc
```

Departamento tem Funcionários, mas eles existem fora dele.

==========

Composição ==> dependência total

```
Tipo ItemPedido
    descricao: cadeia;
    valor: real;
FimTipo

Tipo Pedido
    itens: Vetor[10] de ItemPedido;
FimTipo

Func Principal()
    Var p: Pedido;

    p.itens[1].descricao <- "Teclado";
    p.itens[1].valor <- 150.00;
    Retorne vazio;
FimFunc
```

ItemPedido só existe dentro de Pedido.