# Padrões de design estruturais

### *Composite*

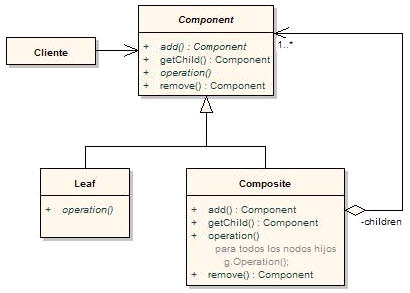
**Sobre Composite**

* O padrão Composite é usado para construir algoritmos ou objetos complexos de outros mais simples e similares, graças à composição recursiva e a uma estrutura de árvore.
* Em outras palavras, ele permite construir objetos complexos ao compor recursivamente objetos similares em uma estrutura de árvore.
* Simplifica o tratamento de objetos criados, já que todos eles têm uma interface comum, eles são tratados da mesma maneira.
* Procura representar uma hierarquia de objetos conhecida como "parte-tudo", onde a teoria de que as "partes" formam o "todo" é seguida, sempre levando em conta que cada "parte" pode ter outra "parte" interna.

**Aplicações de Composite**

* Este padrão deve ser usado quando:
  + Procura representar uma hierarquia de objetos como "parte-tudo".
  + Pretende-se que o cliente possa ignorar a diferença entre objetos primitivos e compostos (para que ele possa tratá-los da mesma maneira).

**Diagrama UML de Composite**

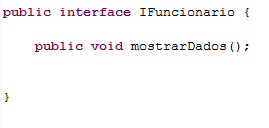
[](http://1.bp.blogspot.com/-LYqFnV_rA-o/TeZnLi9Ow7I/AAAAAAAAJMc/Esp_EEepGM8/s1600/image00.jpg)

* Component: implementa um comportamento comum entre as classes e declara uma interface de manipulação para os pais na estrutura recursiva.
* Leaf: representa os objetos "folha" (eles não têm filhos). Definir comportamentos para objetos primitivos.
* Composite: define um comportamento para objetos com filhos. Armazena componentes filhos. Implementa operações de relacionamento com crianças.
* Cliente: manipula objetos da composição através do Componente. Os clientes usam a interface Component para interagir com objetos na estrutura Composite. Se o receptor é uma folha, a interação é direta. Se for um Composite, ele deve alcançar os objetos "filhos" e levar a operações adicionais.

**Exemplo de Composite**

* Vamos fazer um exemplo de um banco. Um banco pode ter vários setores: Administrativo, Administrativo, Recursos Humanos, Bancos de Poupança, etc. Cada um desses setores terá funcionários que cobram um salário. Em nosso caso, usaremos o Composite para calcular a soma dos salários da empresa.

* + Para isso, definimos a interface e o composite.

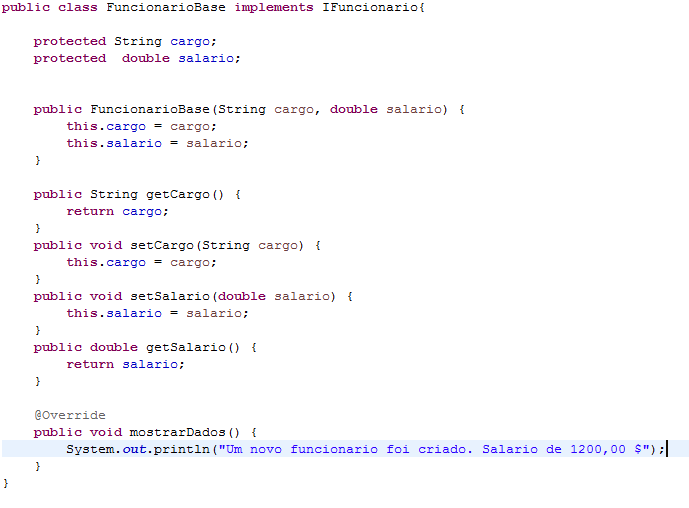


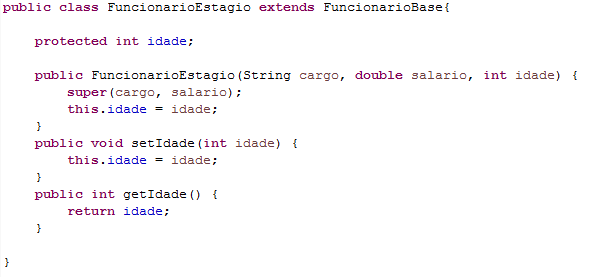


* Na classe Composite, há todo o segredo do padrão: ele contém uma coleção de filhos do tipo que é adicionada ao adicionar o método (IFuncionario) e quando o composite é executado, o salário () simplesmente percorre seus filhos e executa o método. Seus filhos podem ser de 2 tipos: Composite (que por sua vez fará a mesma jornada com seus próprios filhos) ou Folhas que retornará um valor.



* No nosso exemplo existem dois funcionários.





* Pronto, agora vamos ver como concatenar tudo. Fiz na classe Principal (que viria a representar o cliente), mas na realidade o armado do banco nem sempre tem que fazer o cliente. Na verdade, poderíamos usar um padrão de criação para nos ajudar na montagem. O banco é composto de vários setores, que podem ter pessoas ou mais setores dentro.



