Curso: Sistemas de Informação	
Disciplina: Padrões de projeto	**Período:** 8
Professor: Luiz Gustavo Dias	**Tipo:** Design Patterns
Objetivo: Composite	

DESIGN PATTERNS: COMPOSITE

Compor objetos em estruturas de árvore para representarem hierarquias partes-todo. Composite permite aos clientes tratarem de maneira uniforme objetos individuais e composições de objeto.

Basicamente, diz que através deste padrão podemos utilizar a composição de objetos simples para tratar a composição de objetos mais complexos. Podemos tratar tanto a estrutura quanto um único objeto da estrutura de maneira mais uniforme.

Aplicações gráficas, tais como editores de desenhos e sistemas de captura esquemática, permitem aos usuários construir diagramas complexos a partir de componentes simples. O usuário pode agrupar componentes para formar componentes maiores, os quais, por sua vez, podem ser agrupados para formar componentes ainda maiores. Uma implementação simples poderia definir classes para primitivas gráficas, tais como Texto e Linhas, além de outras classes que funcionam como recipientes (containers) para essas primitivas.

Porém, há um problema com essa abordagem: o código que usa essas classes deve tratar objetos primitivos e objetos recipientes de modo diferente, mesmo se na maior parte do tempo o usuário os trata de forma idêntica. Ter que distinguir entre esses objetos torna a aplicação mais complexa. O padrão Composite descreve como usar a composição recursiva de maneira que os clientes não tenham que fazer essa distinção.

A chave para o padrão Composite é uma classe abstrata que representa tanto as primitivas como os seus recipientes.

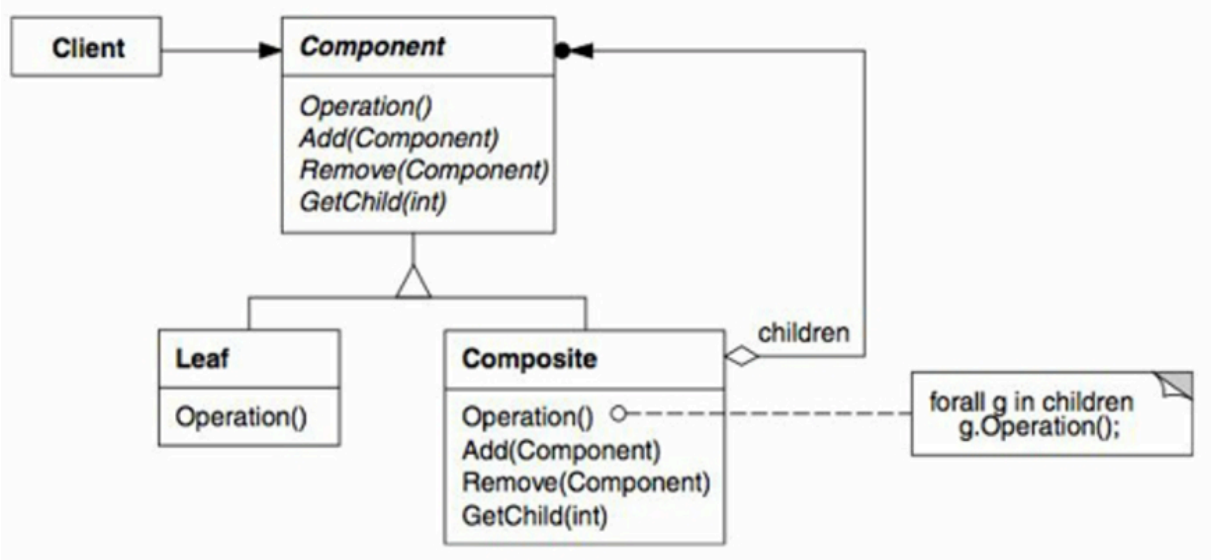
Sobre o padrão

- É um padrão da categoria estrutural;
- Faz mais sentido em estruturas que podem ser tratadas hierarquicamente (como árvores);
- Pode ser uma solução para estruturas complexas que podem ser tratadas de maneira uniforme – basicamente podemos chamar o mesmo método na estrutura ou em um objeto apenas;
- Prioriza composição ao invés de herança;
    - Ex.: produto solto com preço e caixa com vários do mesmo produto também com preço.

Aplicabilidade

- Estrutura de objetos possa ser representada hierarquicamente, como por exemplo, estrutura do tipo árvore;
- Você quiser que o código cliente trate objetos compostos e objetos simples da mesma maneira.

Estrutura



Component: declara a interface para os objetos da composição; implementa comportamento padrão para a interface comum a todas as classes, conforme

aproriado; declara uma interface para acessar e gerenciar os seus componentes filhos.

Composite: define comportamento para componentes que têm filhos; armazena os componentes filhos; implementa as operações relacionadas com os filhos presentes na interface Component.

Leaf: representa objetos-folha na composição – uma folha não tem filhos; define comportamento para objetos primitivos na composição.

Client: manipula objetos na composição através da interface de Component.

Consequências

Positivas:

1. É muito fácil criar objetos complexos por composição;
2. Assim como gerar hierarquia de objetos;
3. Assim como usar polimorfismo por recursão;
4. Assim como adicionar novos elementos na estrutura (OCP).

Negativas:

1. Dependendo da estrutura pode quebra o princípio de segregação da interface (ISP).