

# Composite :-

What is composite?

- We have a part-whole relationship or hierarchy of objects and we want to be able to treat all objects in this hierarchy uniformly.
- This is not a simple composition concept from object oriented programming but a further enhancement to that principle
- Think of composite pattern when dealing with tree structure of objects.

UML :-

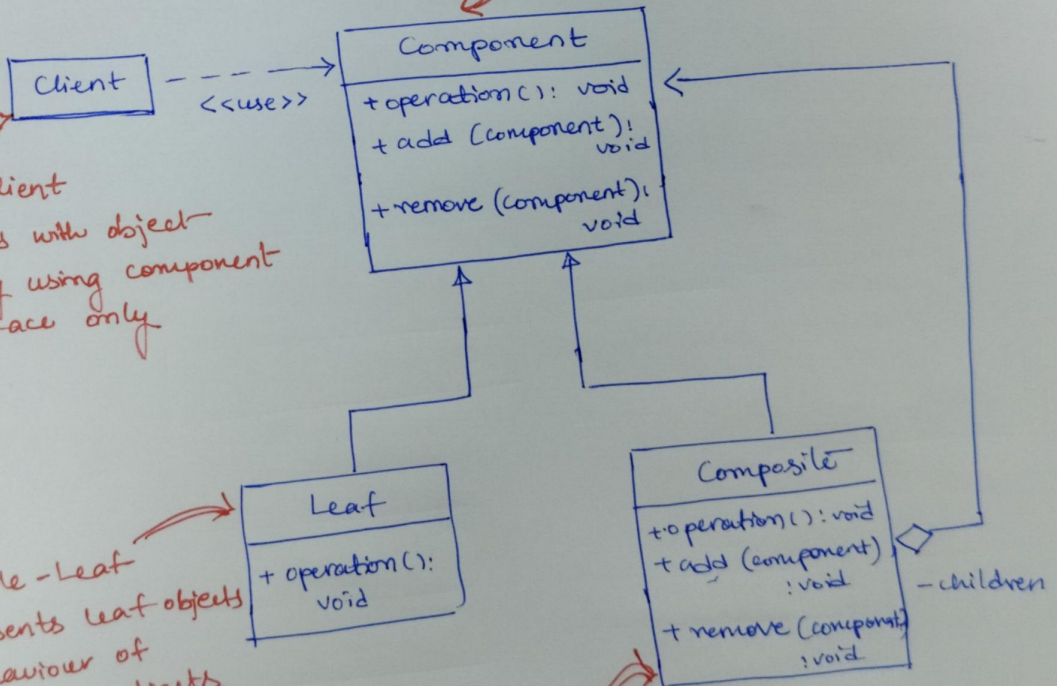
class Composite

Role - Component  
- defines behaviour common to all classes  
- including methods to access children

Role - Client  
- works with object hierarchy using component interface only

Role - Leaf  
- represents leaf objects w behaviour of primitive objects.

Role - Composite  
- stores child components



## Implement a Composite :-

- We start by creating an abstract class / interface for component.
- component must be declare all methods that are applicable to both leaf and composite.
  - We have to choose who defines the children management operations, component or composite.
  - Then we implement the composite. An operation invoked on composite is propagated to all it's children.
  - In leaf nodes we have to handle the non-applicable operations like add/remove a child if they are defined in component.
- In the end, a composite pattern implementation will allow you to write algorithms without worrying about whether node is leaf or composite.

