Composite Pattern

> Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.

- The force behind the Composite pattern is LSP.
- Example That would benefit from Composite Pattern →
    - An application want to be able to use several types of gates. Gates like AND, OR are primitive, while gates like Flip-Flops are containers of other gates. Define the hierarchy of these classes such that the client can treat all the types of gate classes uniformly.

When to use Composite Pattern

- When you want to to represent part-whole hierarchies of objects.
- When you want the clients to be able to ignore the differences between the compositions of objects and individual objects. Clients will treat all objects in the composite structure uniformly.

Consequences of using Composite Pattern

- Primitive objects recursively composed into more complex objects.
- Wherever clients code expects primitive object, it can also take a composite.
- Simplified Client code — can treat composite & individual objects uniformly.
- Easier to add new kinds of Components.
- Makes design overly general.
- Hard to restrict the components of a composite.

Composite Pattern vs. Other Patterns

- Used for Chain of Responsibility
- Decorator often used with Composite
- Flyweight lets you share components but they no longer refer to their parents.
- Iterator can be used to traverse Composites.
- Visitor localises operations & behaviour that would otherwise be distributed across composite and leaf classes.