Structural pattern

1) Decorator
    a) Add new functionality dynamically to existing objects, or remove it
2) Proxy
    a) Control access to an object
    b) Create expensive objects on demand
3) .Bridge
    a) Enable development of the interface and implementation of a component to proceed independently.
    b) Select or switch implementations at runtime
4) Adapter
    a) Match otherwise incompatible interfaces
5) Flyweight
    a) Reduce the cost of working with large numbers of very small objects
6) Façade
    a) Reorganize a system with many subsystems into identifiable layers with single entry points
    b) Simplify the interface to a complex subsystem.
7) Composite
    a) Treat single objects and composite objects in the same way