1. Mediator Pattern

* Realationship이 굉장히 많이 존재하여, communication이나 control의 common point가 필요할 때 사용.
* EX> 공항 control tower : 굉장히 많은 비행기들이 오고가는데, 비행기들끼리 서로 통신을 하면 굉장히 많은 통신을 해야함. 따라서 Control Tower를 이용하여 tower가 중계자(Mediator) 역할을 함.

1. Visitor Pattern

* 알고리즘을 객체 구조에서 분리시킴.
* 구조는 변하지 않으면서 기능만 따로 추가되거나 확장되어야 할 경우에 사용.
* Visit 객체를 통해 알고리즘이 객체를 방문.
* 객체에서 operation decoupling.

1. Proxy Pattern

* 객체에 접근을 제어할 때, 객체에 접근 시에 기능을 추가할 때 사용.
* 객체(original)에 접근을 제어하기 위한 새로운 객체(proxy)를 제공한다.

1. Remote proxy

* 객체가 다른 주소공간에 존재하거나, 원격 주소에 있는 어떤 객체를 제어할 일이 생겼을때.
* Remote proxy는 다른 주소 공간에 존재하는 개체를 가리키며, request 전달.

1. Virtual proxy

* 접근하는데 시간이 걸릴 때, 접근을 지연 시키기 위해 virtual proxy에서 객체에 대한 정보를 가지고 있음.
* 생성하기에 비용이 많이 드는 객체가 준비완료 될 때까지 대신한다.
* EX> 용량이 큰 이미지 표시를 요청할 때, 로딩에 시간이 걸리므로 로딩하는 동안 로딩중이라는 것을 표시할 새로운 객체(Virtual proxy)를 생성.

1. Protection proxy

* 요청을 수행할 때, caller 가 액세스 권한이 있는지 확인.