싱글톤 패턴 : 애플리케이션이 시작될 때 어떤 클래스가 치초 한번만 메모리를 할당하고 그 메모리에 인스턴스를 만들어 사용하는 디자인 패턴으로 생성자가 여러 차례 호출되더라도 객체는 하나만 생성된다. 즉 싱글톤 패턴은 단 하나의 인스턴스를 생성해 사용하는 디자인 패턴이다.

싱글톤으로 만들어진 클래스의 인스턴스는 전역 인스턴스이기 때문에 다른 클래스의 인스터스들이 공통된 객체를 여러개 생성해서 사용해야되는 경우에는 사용하기 좋지만 싱글톤 인스턴스가 너무 많은 일을 하거나 많은 데이터를 공유시킬 경우 다른 클래스의 인스턴스들 간에 결합도가 높아져 개방-폐쇄 원칙을 위반하게 된다.

커맨드 패턴 : 실행된 기능을 캡슐화함으로써 주어진 여러 기능을 실행할 수 있는 재사용성이 높은 클래스를 설계하는 패턴으로 이벤트가 발생했을 떄 실행될 기능이 다양하면서도 변경이 필요한 경우에 이벤트를 발생시키는 클래스를 변경하지 않고 재사용하고자 할 때 유용하다.

팩토리 패턴 : 객체를 생성하기 위한 인터페이스를 정의할 때 어떤 클래스의 인스턴스를 만들지는 서브클래스에서 결정하게 만드는 것으로 코드를 변경하거나 확장해야 할 때 코드를 다시 확인하고 추가 또는 제거해야 한다는 것을 의미한다.

템플릿 메서드 : 어떤 작업을 처리하는 특정부분을 서브 클래스로 캡슐화하여 전체 일을 수행하는 구조는 바꾸지 않으면서 특정단계에서 수행하는 내역을 바꾸는 패턴으로 전체적으로 동일하면서 부분적으로는 다른 구문으로 구성된 메서드의 코드 중복을 최소화 할 때 유용하다.

예시 : 엘리베이터 제어 시스템에서 모터를 구동시키는 것

문제점 : 다른 회사의 엘리베이터의 모터를 구동시켜야 할 때 제어가 불가능하다.

이터레이터 패턴 : 컬렉션 구현 방법을 노출시키지 않으면서도 그 집합체 안에 들어있는 모든 항목에 접근할 수 있는 방법을 제공한다. 이터레이터 패턴은 모든항목에 일일이 접근하는 작업을 반복자 객체에서 맡게 되므로 집합체의 인터페이스 및 구현이 간단해지고 집합체에서는 반복작업에서 손을 떼고 원래 자신이 할 일에 전념할 수 있다는 장점이 있다.