**SOLID 원칙**

객체지향 프로그래밍에서 유지보수가 쉽고, 확장 가능하며, 유연한 코드를 작성하기 위해 제안된 5가지 설계 원칙이다. 각 원칙의 앞글자를 따서 S.O.L.I.D라고 부른다.

**단일 책임 원칙 (Single Responsibility Principle)** :

클래스는 하나의 책임만 가져야 함. (여러가지 일을 하지 않기 때문에 유지보수가 쉬워짐)

**개방-폐쇄 원칙 (Open/Closed Principle):**

확장에는 열려 있고, 수정에는 닫혀 있어야 함. 기존 코드를 수정하지 않은 채로 기능을 확장 할 수 있다는 뜻. 보통 추상 클래스나 인터페이스를 사용.

**리스코프 치환 원칙 (Liskov Substitution Principle)**:

자식 클래스는 부모 클래스 대신 사용 가능해야 함. 자식이 부모보다 행동을 좁히면 안 됨. 함수가 부모 클래스를 사용중이라면, 그걸 자식 클래스로 바꿔도 문제가 없어야 함.

**인터페이스 분리 원칙 (Interface Segregation Principle):**

클라이언트는 사용하지 않는 인터페이스에 의존하면 안 됨. 큰 인터페이스를 여러 개의 작은 인터페이스로 나눠줘야 함.

**의존 역전 원칙 (Dependency Inversion Principle):**

추상화에 의존해야지, 구체적인 구현에 의존하면 안 됨. 즉 구체적인 클래스가 아닌, 인터페이스(추상화)에 의존해야 함. 이렇게 하면 장점으로 기능을 바꿔도 상위 모듈을 수정하지 않아도 됨.