03. 설계 원칙

- SOLID 원칙은 함수와 데이터 구조를 클래스로 배치하는 방법, 그리고 이들 클래스를 서로 결합하는 방법을 설명한다.
- SOLID 원칙의 목적
  - 변경에 유연하다
  - 이해하기 쉽다
  - 많은 소프트웨어 시스템에 사용될 수 있는 컴포넌트의 기반이 된다.
- 코드 수준보다는 조금 상위에서 적용되며 모듈과 컴포넌트 내부에서 사용되는 소프트웨어 구조를 정의하는 데 도움을 준다.
- SRP(single responsibility principle): 단일 책임 원칙
  - 각 소프트웨어 모듈은 변경의 이유가 단 하나여야 한다.
- OCP(open-closed principle): 개방-폐쇄 원칙
  - 기존 코드를 수정하기보다는 반드시 새로운 코드를 추가하는 방식으로 시스템의 행위를 변경할 수 있도록 설계해야지 소프트 웨어 시스템을 쉽게 변경할 수 있따.
- LSP(liskov substitution principle): 리스코프 치환 원칙
  - 상호 대체 가능한 구성요소를 이용해 소프트웨어 시스템을 만들 수 있으려면, 이들 구성요소는 반드시 서로 치환 가능해야한다.
- ISP(interface segregation principle): 인터페이스 분리 원칙
  - 소프트 웨어 설계자는 사용하지 않은 것에 의존하지 않아야한다.
- DIP(dependency inversion principle): 의존성 역전 원칙
  - 고수준 정책을 구현하는 코드는 저수준 세부사항을 구현하는 코드에 절대로 의존해서는 안된다. 대신 세부사항이 정책에 의존해야한다.