SOLID

객체지향 설계에서 지켜줘야 할 5개의 **소프트웨어 개발 원칙**

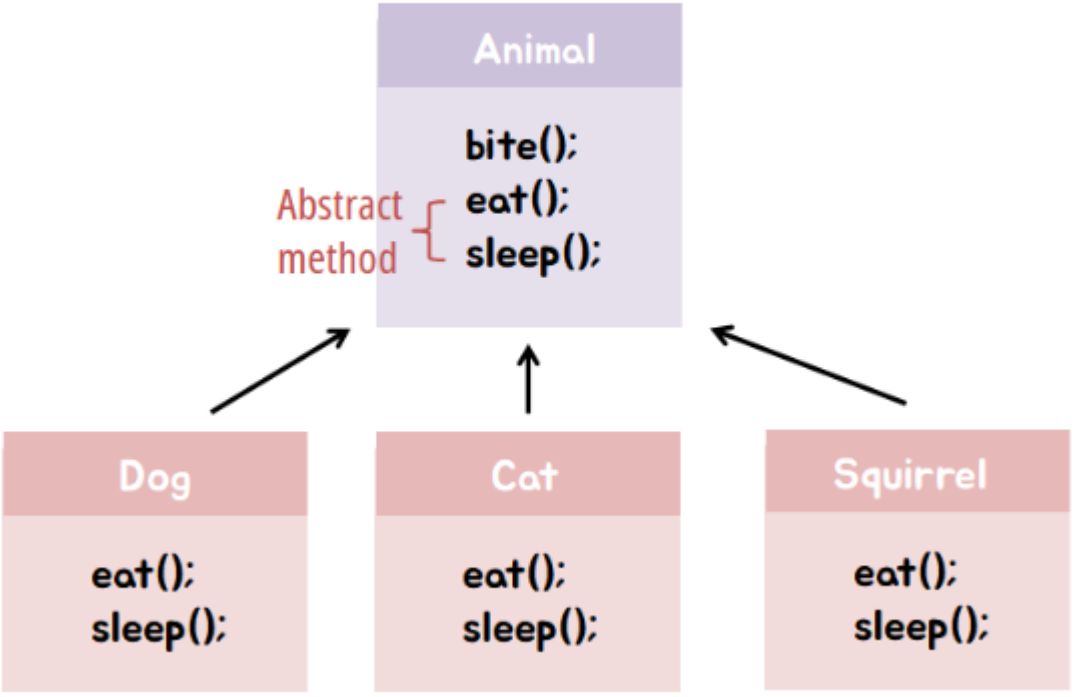
- SRP(Single Responsibility Principle): 단일 책임 원칙
- OCP(Open Closed Priciple): 개방 폐쇄 원칙
- LSP(Listov Substitution Priciple): 리스코프 치환 원칙
- ISP(Interface Segregation Principle): 인터페이스 분리 원칙
- DIP(Dependency Inversion Principle): 의존 역전 원칙

1. SRP(Single Responsibility Principle): 단일 책임 원칙

- 클래스(객체)는 단 하나의 책임만 가져야 한다는 원칙
- 여기서'책임' 이라는 의미는 하나의 '기능 담당'

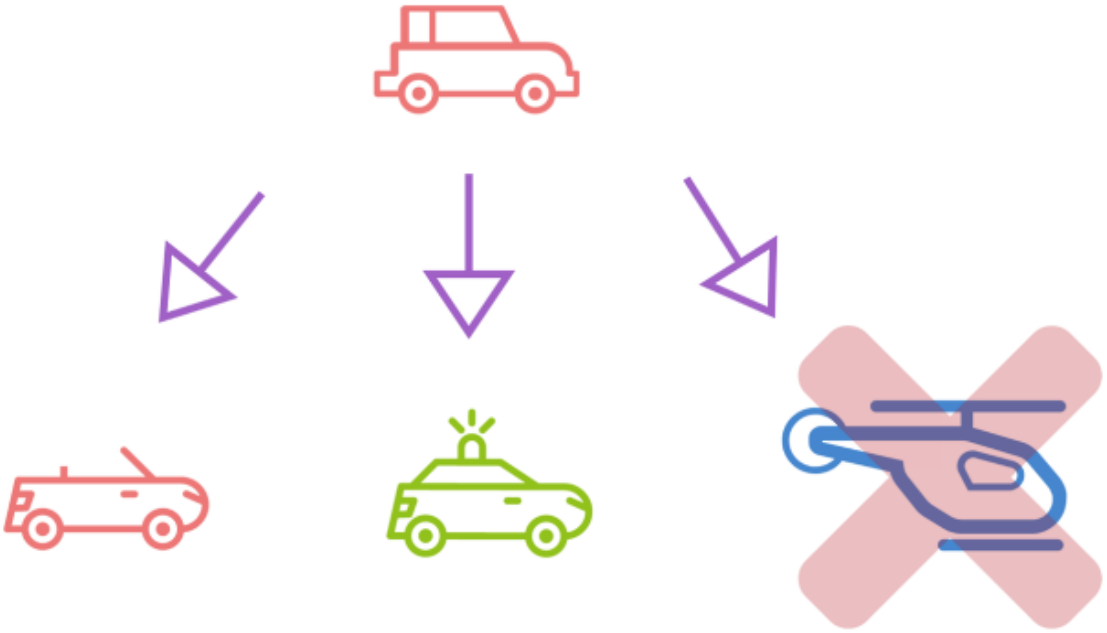
- 즉, 하나의 클래스는 하나의 기능 담당하여 하나의 책임을 수행하는데 집중되도록 클래스를 따로따로 여러개 설계하라는 원칙
    - 만일 하나의 클래스에 기능(책임)이 여러개 있다면 기능 변경(수정) 이 일어났을때 수정해야할 코드가 많아지기 때문

2. OCP(Open Closed Priciple): 개방 폐쇄 원칙



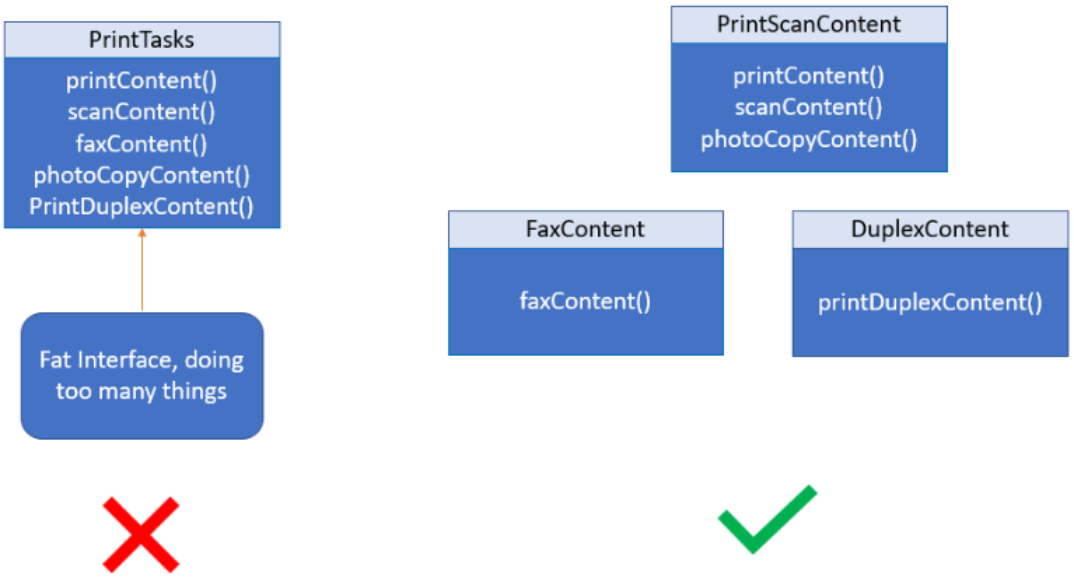
- 클래스는 '확장에 열려있어야 하며, 수정에는 닫혀있어야 한다' 를 뜻함
- 기능 추가 요청이 오면 클래스를 확장을 통해 손쉽게 구현하면서,확장에 따른 클래스 수정은 최소화 하도록 프로그램을 작성해야 하는 설계 기법
    - [확장에 열려있다] - 새로운 변경 사항이 발생했을 때 유연하게 코드를 추가함으로써 큰 힘을 들이지 않고 애플리케이션의 기능을 확장할 수 있음
    - [변경에 닫혀있다] - 새로운 변경 사항이 발생했을 때 객체를 직접적으로 수정을 제한함.
- 즉, 다형성과 확장을 가능하게 하는 객체지향의 장점을 극대화하는 기본적인 설계 원칙
- 이를 통해 코드의 안정성과 재사용성이 향상됨

3. LSP(Listov Substitution Priciple): 리스코프 치환 원칙



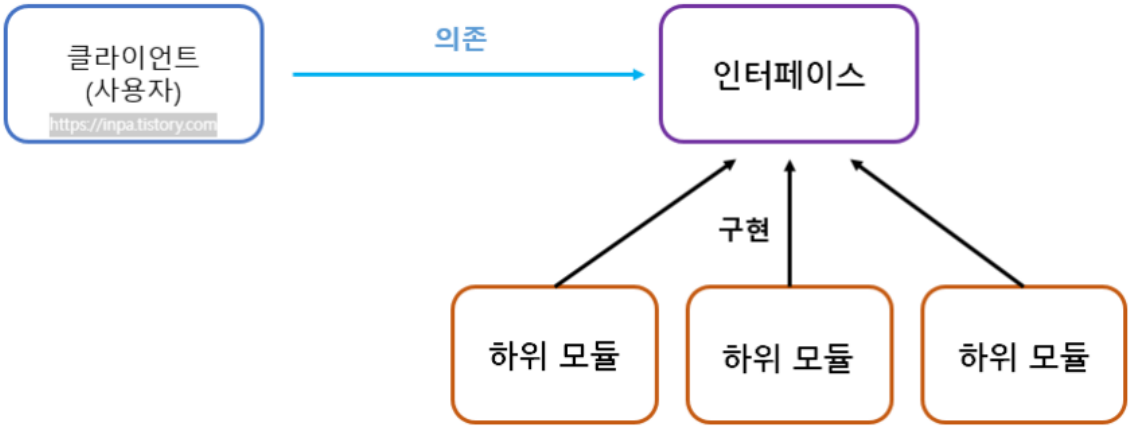
- 서브 타입은 언제나 기반(부모) 타입으로 교체할 수 있어야 한다는 원칙
- 즉, 부모 클래스의 인스턴스를 자식 클래스의 인스턴스로 대체해도 프로그램의 의미가 변하지 않아야 함
- 다형성 원리를 이용하기 위한 원칙 개념

4. ISP(Interface Segregation Principle): 인터페이스 분리 원칙



- 인터페이스를 각각 사용에 맞게 끔 잘게 분리해야한다는 설계 원칙
- SRP 원칙이 클래스의 단일 책임을 강조한다면, ISP는인터페이스의 단일 책임을 강조하는 것
- ISP 원칙은 인터페이스를 사용하는 클라이언트를 기준으로 분리함으로써, 클라이언트의 목적과 용도에 적합한 인터페이스 만을 제공하는 것이 목표.
- 주의해야 할점 : 한번 인터페이스를 분리하여 구성해놓고 나중에 무언가 수정사항이 생겨서 또 인터페이스들을 분리하는 행위를 가하지 말아야 함
- 인터페이스는 클라이언트가 필요로 하는 기능에 대해서만 제공해야 합니다. 이를 통해 인터페이스를 작고 응집력 있는 단위로 유지할 수 있습니다.

5. DIP(Dependency Inversion Principle): 의존 역전 원칙



- 어떤 Class를 참조해서 사용해야하는 상황이 생긴다면, 그 Class를 직접 참조하는 것이 아니라 그대상의 상위 요소(추상 클래스 or 인터페이스)로 참조하라는 원칙
- 구현 클래스에 의존하지 말고,인터페이스에 의존하라는 뜻
- 의존 관계를 맺을 때 변화하기 쉬운 것 또는 자주 변화하는 것보다는, 변화하기 어려운 것 거의 변화가 없는 것에 의존하라는 것

출처 : https://inpa.tistory.com/entry/OOP-💠-객체-지향-설계의-5가지-원칙-SOLID