좋은 객체 지향 설계의 5가지 원칙 (SOLID)

2022.03.06

SOLID

클린코드로 유명한 로버트 마틴이 좋은 객체 지향 설계의 5가지 원칙을 정리

- SRP: 단일 책임 원칙(Single responsibility principle)
- OCP: 개방-폐쇄 원칙(Open/closed principle)
- LSP: 리스코프 치환 원칙(Liskov substitution principle)
- ISP: 인터페이스 분리 원칙(Interface segregation principle)
- DIP: 의존관계 역전 원칙(Dependency inversion principle)

SRP 단일 책임 원칙
Single responsibility principle

- 한 클래스는 하나의 책임만 가져야 한다.
- 하나의 책임이라는 것은 모호하다.
- **중요한 기준은 변경**이다.
  - 변경이 있을 때 파급 효과가 적으면 단일 책임 원칙을 잘 따른 것
  - Ex) UI 변경, 객체의 생성과 사용을 분리
- 객체 간의 응집도는 높고 결합도가 낮은 프로그램

OCP 개방-폐쇄 원칙
Open/closed principle

- 소프트웨어 요소는 **확장에는 열려** 있으나 **변경에는 닫혀** 있어야 한다.
- 다형성을 활용
- 인터페이스를 구현한 새로운 클래스를 하나 만들어서 새로운 기능을 구현
- 문제점
- 클라이언트가 구현 클래스를 직접 선택
  - 구현 객체를 변경하려면 클라이언트 코드를 변경해야 한다.
- 분명 다형성을 사용했지만 OCP 원칙을 지킬 수 없다.
  - 객체를 생성하고, 연관관계를 맺어주는 별도의 조립, 설정자가 필요하다.
  - Spring Framework는 Spring Container가 이 역할을 담당한다.

LSP 리스코프 치환 원칙

Liskov substitution principle

- 프로그램의 객체는 프로그램의 정확성을 깨뜨리지 않으면서 하위 타입의 인스턴스로 바꿀 수 있어야 한다

- 다형성에서 하위 클래스는 인터페이스 규약을 다 지켜야 한다는 것, 다형성을 지원하기 위한 원칙, 인터페이스를 구현한 구현체를 믿고 사용하려면 이 원칙이 필요하다.

- 단순히 컴파일에 성공하는 것을 넘어서는 이야기
  - 부모클래스를 상속하는 자식클래스는 부모 클래스의 규약을 무시하거나 오버라이딩을 자제해야 한다.

ISP 인터페이스 분리 원칙

Interface segregation principle

- 특정 클라이언트를 위한 인터페이스 여러 개가 범용 인터페이스 하나보다 낫다
  - 자동차 인터페이스 -> 운전 인터페이스, 정비 인터페이스로 분리
  - 사용자 클라이언트 -> 운전자 클라이언트, 정비사 클라이언트로 분리
- 분리하면 정비 인터페이스 자체가 변해도 운전자 클라이언트에 영향을 주지 않음
- 인터페이스가 명확해지고, 대체 가능성이 높아진다.

DIP 의존관계 역전 원칙

Dependency inversion principle

- 추상화에 의존 해야지, 구체화에 의존하면 안된다.
  - 의존성 주입은 이 원칙을 따르는 방법 중 하나다.

- 구현 클래스에 의존하지 말고, 인터페이스에 의존하라는 뜻
- 클라이언트가 구현체에 의존하게 되면 변경이 아주 어려워진다.

- OCP는 인터페이스에 의존하지만, 구현 클래스도 동시에 의존한다.
  - 이는 DIP를 위반하는 것
- 다형성만으로는 OCP, DIP를 지킬 수 없다.
  - Spring Framework는 스프링이 지원하는 의존성 주입(DI)과 DI컨테이너를 이용해 자바 객체의 의존성을 연동해준다.