230829 전략패턴

알고리즘군을 정의하고 캡슐화해서 각각의 알고리즘군을 수정해서 쓸 수 있게 해준다. 전략 패턴을 사용하면 클라이언트로부터 알고리즘을 분리해서 독립적으로 변경할 수 있다.

디자인 원칙

- 애플리케이션에서 달라지는 부분을 찾아내고, 달라지지않는 부분과 분리한다.
  - 달라지는 부분은 따로 캡슐화한다.
- 구현보다는 인터페이스에 맞춰서 프로그래밍한다.
  - 즉 인터페이스라는 큰 틀에 맞추어 개발할 것
- 상속보다는 **구성**을 활용할것
  - 고릴라 바나나 문제
  - https://www.johndcook.com/blog/2011/07/19/you-wanted-banana/

OCP와 전략패턴

OCP

> Open-Closed Principle

- 개방 폐쇄 원칙
- 소프트웨어 구성 요소(컴포넌트, 클래스, 모듈, 함수 등)는 확장에 대해서는 개방되어 있어야 하며 변경에 대해서는 폐쇄되어야 한다.
- 기존의 코드는 변경하지 않으면서 기능을 추가할 수 있도록 설계되어야 한다는 뜻
- **오리의 나는 법이 추가되어도 기존의 코드가 바뀌지 않아야함.**

극복방법 : Is-a 와 has-a

상속은 그것이 무엇 인지(is-a)를 디자인 하는 것이고 구성은 무엇을 가지는(has-a)를 디자인하는 것

Is-a

- 상위클래스가 바뀌면 하위클래스에 끼치는 영향이 매우 크다는 단점

Has-a

- 변경 될 것과 변하지 않을것을 엄격히 구분한다.
- 이 두 모듈이 만나는 지점에 인터페이스를 정의
- 구현에 의존하기보다 정의한 인터페이스에 의존하도록 코드를 작성

전략 패턴

- 전략을 쉽게 변경할 수 있도록 해주는 디자인 패턴으로, 행위를 클래스를 캡슐화해 동적으로 행위를 자유롭게 바꿀 수 있게 해주는 패턴
- 새로운 기능의 추가가 기존의 코드에 영향을 미치지 못하게 하므로 OCP를 만족