Refactoring 03 [가변 데이터]

코딩으로 학습하는 리팩토링

백기선

2022.11.25

Smell #6 가변 데이터 (Mutable Data)

- 데이터를 변경하다보면 예상치 못한 결과가 나올수도 있다.
- 함수형 프로그래밍 → 데이터를 전달시 복사해서 전달하므로 데이터 변경이 되지않음
  하지만 그밖의 프로그래밍 언어는 레퍼런스(참조형식)을 사용하여 데이터를 저장하기 때문에 변경해서 사이드 이펙트가 발생할 수있다.

리팩토링 - 변수 쪼개기

- Split Variable

변수가 여러번 재할당 될때

변수가 하나의 역할만 하도록 → 이름을 잘 지어서 하나의 책임을 표현하게 한다.

리팩토링 - 질의 함수와 변경 함수 분리하기

- Separate Query from Modifier

값을 조회 하는 메소드는 (눈에 띌 만한)사이드 이펙트 없이 값을 조회 할 수 있어야 한다.

명령 - 조회를 준비(Command - query separation) 규칙을 지키면 - 테스트도 용이해진다.

리팩토링 - 세터 제거하기

- Remove Setting Method

세터를 제공한다는 것은 필드가 변경 될 수 있다는 것이다.

객체가 생성 후 필드값이 변경하지 않길 바란다면 세터를 없애는 게 맞다.

리팩토링 - 파생 변수를 질의 함수로 바꾸기

- Replace Derived Variable with Query

변경할 수 있는 데이터를 최대한 줄이도록 노력해야 한다.

계산해서 알아낼 수 있는 변수를 대체 , 계산식을 함수로 표현해서 변수를 줄일 수 있다.

계산에 필요한 데이터가 변하지 않는 값이라면 계산의 결과에 해당하는 데이터 역시 불변 데이터기 때문에 해당 변수는 그대로 유지할 수 있다. (리팩토링을 적용할 필요 없다.)

리팩토링 - 여러 함수를 변환 함수로 묶기

- Combine Funtions into Transform

관련있는 여러 파생 변수를 만들어내는 함수가 여러곳에서 만들어지고 사용된다면 파생 변수를 통해서 한 곳으로 모아 둘 수 있다.

소스 데이터가 변경될 수 있는 경우에는 “여러 함수를 클래스로 묶기” 를 사용하는 것이 적절하다.

불변 데이터는 필드로 생성해두고 재사용 할 수도 있다.

리팩토링 - 참조를 값으로 바꾸기

- Change Reference to Value

레퍼런스 객체 를 값 객체로 바꾸는 것.

값 객체는 객체가 가진 필드의 값으로 동일성을 확인한다.

값 객체는 변하지 않는다.

어떤 객체의 변경 내역을 다른 곳으로 전파시키고 싶다면 레퍼런스 아니면 값 객체를 사용한다.