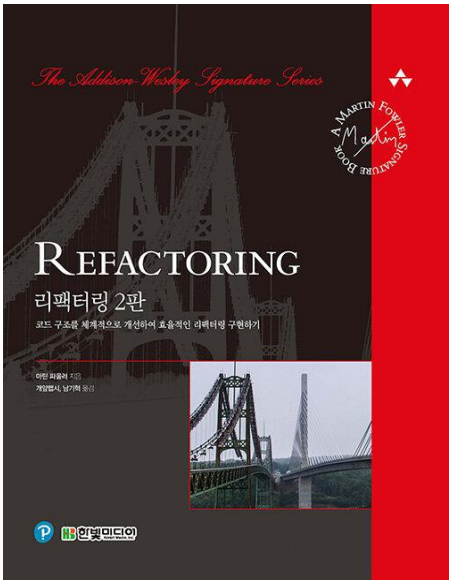
Refactoring 01 [이름 변경, 중복 코드 제거]

코딩으로 학습하는 리팩토링

백기선

2022.11.16



Refactoring (리팩터링 2판)

읽어보기!

리팩토링(Refactoring)

구조 변경으로 에러를 줄이고 과거에 최선이였던 코드를 현재의 최선의 코드로 변환 시키는 것.

→ Code smell 을 위주로 생각해서 리팩토링하는 방식을 익혀봐야한다.

냄새 **1.** 이해하기 힘든 이름 **(Myterius Name)**

변수의 이름, 클래스 이름, 모듈의 이름, 함수의 이름등 여러가지 이름을 이해하기 쉽게 좋은 이름으로 바꿔줘야 한다.

1) 함수 선언 변경하기 **(Change Function Declaration)**

함수의 선언 부분 변경 (이름, 매개변수 추가, 매개변수 제거, 시그니처 변경)

함수의 매개변수는 내부의 문맥을 결정 , 의존성을 결정한다.

2) 변수 이름 바꾸기 **(Rename Variable)**

변수 이름은 중요! (scope이 넓어질수록 중요해 진다.)

다이나믹 타입을 지원하는 언어에서는 타입을 이름에 넣기도 한다.

3) 필드 이름 바꾸기 **(Rename Field)**

필드는 클래스 범위 밖에서도 사용될 수 있기 때문에 매우 중요하다.

Record 자료 구조를 사용하게 되면 필드를 묶어놓은 자료구조

Record는 관련있는 클래스를 한번에 모아둔 자료형 Immutable

Record : 파이썬의 **dictionary , C++ record** 와 유사 → **DTO** 에 사용하면 편할듯

냄새 **2.** 중복 코드 **Duplicated Code**

1) 중복 코드가 동일하게 반복 → 함수 추출 **(Extract Function)**

'의도' 와 '구현' 분리

의도를 잘 판단하도록 함수를 추출했을 때 한줄짜리 메소드여도 괜찮다.

2) 비슷한 코드의 반복 → 코드 분리 **(Slide Statements)**

코드를 위 아래로 위치를 변경해준다. 관련있는 코드끼리 가까이 붙이는 refactoring

3) 여러 하위 클래스에 동일한 코드 → 메소드 올리기 **(Pull Up Method)**

상속 구조에서 중복되는 메소드를 상위 클래스로 올리는 작업

비슷하지만 일부 값만 다른 경우라면 " 함수 매개변수화 하기" 리팩토링 후에 방법을 사용하거나 함수를 동일하게 맞춘후 상위 클래스로 올릴 수 있다.