20180927 SD

오늘은 Unit test와 refactoring에 대해 조금 더 살펴볼 예정이다.

한번에 하나의 feature만 추가해서 당장의 현안만 해결하도록 해라?! 근데 이건 fool software development일 것이다. (초기단계에는) 이것은 미래의 확장성따위는 일단 고려하지 않는다!

Safe refactoring을 위해서는 unit test가 중요하다. 각각의 작은 method들이 잘 작동하는지 확인하면서 해야한다는 의미인 듯 하다.

refactoring이란 프로그램의 구조를 바꾸지만 그 동작 자체는 바꾸지 않으면서 바꾸는 것을 의미한다. 코드를 이해하기 조금 더 쉽게 만들고 수정하기 더욱 쉽게 만드는 것에 그 목적이 있는 경우가 많다.

리팩토링의 경우 일단 변수나 함수 이름을 renaming하는 경우도 많고 이런 저런 방법들이 많다!!

이클립스에서 자동 refactoring support를 지원한다고 한다!

리팩토링에서 구조만 바꾸고 기능을 그대로 놓은 경우 software에 기존 구현해두었던 unit test를 이용하여 test가 통과하는지 확인해 볼 수 있다.

작업을 할 때

최신 버전을 다운받는다-> 테스트가 통과하는지 확인한다 -> 추가 테스트를 작성한다-> 실패한다-> 추가 테스트를 통과하기 위한 코드를 작성한다 -> 결국 모든 테스트가 통과한다-> 리팩토링을 통해 코드를 간결화한다->코드를 제출한다(check in)

여기서 smoke testing도 마지막에 함께 진행하고 제출하는 것이 좋은데 smoke testing이란 소프트웨어 배포 단계 이전에 품질 검증을 거치기 전 불필요한 자원과 인력의 낭비를 막기 위해 가장 기본적인 기능들이 되는지 기초 검증을 하는 것이다.

일단 test를 먼저 작성하고 코드를 작성해라! 이것은 test-first programming이라 부르기도 하고 test-driven development라고도 한다.

이것이 버그를 찾아주진 못하지만 최소한 초기에 방지는 할 수 있다.

오픈소스를 다운받을 때 그것을 이해하는 가장 좋은 방법 중 하나는 먼저 테스트 폴더로 가서 테스트가 어떻게 진행되는지 확인하는 것이다.

Test coverage는 주기적으로 측정해주어야한다. 이것을 피드백으로 사용하여 더 많은 테스트를 작성하고 테스트 작성능력을 증진시킨다.

XP : extreme programming의 약자인듯??

Core XP practice는 아래와 같이 구성된다.

Pari programming, user stories, planning game, testing, refactoring

Lecture 4: refactoring

언제나 개발을 할 당시에는 그 코드가 최적이었을테지만 시간이 지나면서 변경이 필요하거나 버그 수정이 필요한 경우들이 있다. 따라서 이런 차후의 작업을 편안하게 하기 위하여 코드를 이해하기 쉽고 수정하기 쉽도록 바꾸는 리팩토링 작업은 굉장히 중요하다!

사실 renaming이 리팩토링에서 제일 중요함!

비슷한 코드가 반복되는 경우 그것을 method화 시키자!(inline method라고 하는듯?)

마찬가지로 비슷한 local variable들의 연산이 반복되는 경우에도 이것을 하나로 묶어서 새로운 변수로 만들어주자! 예를 들어 second\*60=1minute인데 이것을 매번 계산해주지 말고 그냥 변수로 만들어주는 느낌?

예를들어 이제 parameter가 없던 함수에 parameter를 추가하는 상황을 생각해보자

1. 먼저 superclass와 subclass를 확인해야 한다.
2. 그런 다음 함수의 백업을 만들어서 보존해 둔 뒤 parameter를 추가한다.
3. 그런 다음 함수 내부를 변경해서 parameter에 따라 동작하도록 한다
4. 그런 다음 그 함수를 이용하는 모든 곳에서 새로운 함수를 이용하도록 수정해준다.
5. 각각의 과정에서는 test가 진행되어야 한다.
6. 모든 것이 완료되면 백업해두었던 함수를 삭제한다.

Ch.4 11페이지부터 20페이지까지 보면 2개의 parameter를 1개로 합치는 과정에서 수정하는 좋은 과정이 잘 나와있다. 그냥 막 수정하지 말고 수정할때마다 정확한 과정을 거친다면 실수나 문제가 발생하지 않을 것이다!!

이런식으로 할 때 이클립스의 Add parameter 기능을 잘 활용한다면 caller들에도 전부 추가가 되도록 할 수 있는 듯 하다. IDE의 기능을 잘 활용해보자!!

리팩토링의 단위는 작아야 한다! 이런 작은 단위가 진행됨에 따라 계속 체크해주어야함! Major change의 경우 굉장히 많은 refactoring step이 필요할 것이다!!