Clean Code Review

20160690 Junho Bing

Chapter 2. Meaningful Names

Use Intention-Revealing Names

변수, 함수, 클래스 이름 등 이름을 지을 때, 부차적인 설명 없이도 의도와 기능을 파악할 수 있도록 만들어라. Constant 들도 숫자 그대로 쓸 것이 아니라, 그 숫자를 변수에 담아 무슨 역할을 하는 수인지 알게 하자. 알아보기도 쉽고, 후에 변경하기도 용이하도록.

Avoid Disinformation

잘못된 정보를 주는 이름을 사용하지 마라. 예컨대, ~~list 라고 변수명을 붙였는데 실제로 해당 변수가 list가 아니라면, 잘못된 정보를 주는 이름인 것이다. 소문자 L이나 대문자 o도 숫자 0, 1이랑 헷갈리니 쓰지 말 것.

Make Meaningful Distinctions

변수명에 숫자를 뒤에 붙여서 구분하기 보다는 의미 있는 구분을 하자.

Use Pronounceable Names

발음할 수 있는 이름을 짓자. 발음하기기 힘들면 부르기가 힘들다.

Use Searchable Names

Avoid Encodings

One word per concept

Chapter 3. Functions

Small! 함수는 굉장히 짧아야 한다. 20줄 이내로 될 만큼.

Do One Thing

함수는 너무 많은 기능을 하기보다는, 하나의 기능만을 잘 해야 한다. 함수가 하나의 기능만 하는지 확인하는 방법 중의 하나는, 한 함수에서 다른 함수를 뽑아낼 수 있는가를 체크해보는 것이다.

One level of abstraction per function.

코드가 위에서부터 아래로 읽히도록 짜자.

Switch statement

Switch statement를 쓸 때는 여러 일을 하는 경우이므로 하나의 일만 하기는 힘들다. 하지만 polymorphism을 이용해 low-level class를 유지하고 중복되지 않도록 해야 한다. 예를 들어, type에 따라 switch문을 만든 경우, type이 추가되면 코드가 길어지고 새로 작업을 해야 한다. 이를 해결하기 위해 switch문을 ABSTRACT FACTORY의 basement에 넣어 겉으로 드러나지 않게 할 수 있다.

함수의 이름도 기능을 잘 설명하는 이름으로 만들자. 길게 이름을 만드는 것에 두려움이 있을 수 있지만, 길고 descriptive한 이름이 짧고 알아보기 힘든 이름보다 낫다. 한편, 서로 다른 함수의 이름 간에도 사용하는 명사나 동사 등에 있어 일관성이 있어야 한다. Add, get 등.

Function Arguments

Argument도 많으면 힘들다. Testcase로 주어야하는 인자도 많아지고, 한 눈에 알아보기도 어려워진다. 인자가 아예 없으면 가장 좋고, 한 개, 두 개 까지는 괜찮지만, 3개 이상은 지양하는 것이 좋다. 인자가 많아질 것 같은 경우에는, class를 만들어 객체를 호출함으로써 인자를 줄이는 방법을 쓸 수 있다.

Flag argument는 함수를 복잡하게 하고, 하나 이상의 기능을 하게 한다.

Error code를 return 하기 보다는, exception을 쓰면 코드를 잘 분리해서 쓸 수 있으니 유용하다.

마지막으로, 코드를 짜는 것은 시스템의 이야기를 풀어내는 것이라고 생각하고, 이 목적에 따라 함수들은 깔끔하고 정확한 언어로 이해할 수 있게 쓰여야 한다.