# Clean Code Chapter 2 & 3 요약

**20200120 차선호[carprefer]**

## Chapter 2 Meaningful Names

코드를 작성할 때 변수, 함수, 클래스 들의 이름을 잘 짓는 것이 중요하다. 잘 지어두면 다른 사람들, 심지어 자신이 다시 코드를 읽을 때 그 의미를 쉽고 빠르게 이해할 수 있다.

좋은 이름을 짜는 방법들은 다음과 같다.

1. 의도가 드러나는 작명
2. 의미 있게 구별되는 작명
   a1, a2 -> source, destination
3. 발음하기 쉬운 이름
4. 찾기 쉬운 이름
   "The length of a name should correspond to the size of its scope"
   IDE의 자동완성 key의 도움을 효과적으로 받을 수 있도록 작명
5. Class 이름은 명사나 명사구로 지을 것
6. Method 이름은 동사나 동사구로 지을 것
7. 하나의 단어가 하나의 개념에 대응되도록 지을 것
8. CS의 배경지식을 적절히 활용하여 작명
9. 적절한 class 활용으로 의미 있는 context를 제공할 것

피해야 하는 작명법들은 다음과 같다.

1. 잘못된 정보를 주는 작명
2. 추가적인 인코딩이 필요한 작명
3. Hungarian Notation 사용
4. Member 접두사(m_) 사용
5. 재치 있는 작명
6. 같은 단어를 다른 의미로 사용

## Chapter 3 Functions

요즘 프로그램들에 있어서 함수는 가장 기초적이고 많은 비중을 차지하는 문법이다. 저자는 이 함수를 짜는 방식에 있어서 다음을 강조하고 있다.

1. 함수는 작으면 작을수록 좋다. (20줄 이하로)
   화면 안에 함수 전체가 보일 수 있도록 하자. (많아도 100줄 이하)
2. "Functions should do one thing. They should do it well. They should do it only."
   여기서 one thing이란 함수 내 statement들이 같은 추상화 레벨에 놓여있어야 함을 말한다.
3. 함수 이름을 의미 있게 지어야 한다.
   함수 이름은 동사이고, 인자의 이름은 명사이다. 이들의 조합으로 이 함수가 어떤 일을 하는지 알 수 있도록 작명하자. 또한, Side effect를 고려하여 작명하자.
4. 중복된 코드가 없도록 작성한다.
5. 함수 인자는 적을수록 좋다. (3개 이하)
   함수 인자가 적으면 Test의 측면에서 case를 줄이는 장점이 있다. 인자 3개보다는 2개, 2개보다는 1개, 1개보다는 0개가 낫다. 인자 사이에 ordering이 존재하는 경우, 2개나 3개도 괜찮다. 인자를 줄이는 방법으로 argument object, argument list를 활용하자.
6. 인자로 flag를 넘겨주는 것은 피해야 한다.
7. 인자로 output을 받지 말아야 한다.
8. If 문과 while문의 block은 한 줄이어야 한다.
9. Command와 query를 분리하여 함수를 작성해야 한다.
10. 에러를 처리할 때 return 대신 exception을 사용해야 한다.
    Try/catch 문법을 사용할 때도 one thing이 적용됨을 기억하자.
11. Single-entry, single-exit rule을 기억하자.
    짧은 함수에서는 여러 개의 return, break, continue의 사용도 괜찮지만, 긴 함수에서는 위의 룰을 지키자. Goto의 경우, 긴 함수에서 필요할 경우에만 쓰도록 하자.