**Clean Code**

20220019 안재영

**Chapter 2. Meaningful Names – 좋은 이름을 짓는 것이 중요하다**

**“이름”을 지을 때에는 의도를 분명히 밝혀야 한다.**

theList보다는 gameBoard가 더 좋다.

**의도를 분명히 하는 것에는 잘못된 이해를 막는 것도 중요하다.**

자료형이 List가 아닌 오브젝트를 accountList라고 명명하는 것을, 0과 O, 1과 l같이 혼동을 줄 수 있는 이름을 사용하는 것을 피해야 한다.

**발음하기 쉬운 이름을 사용하는 것도 좋다.**

말하기 쉬운 이름은 기억하기에도, 이해하기에도, 대화하기에도 더 적합하다.

**검색하기 쉬운 이름을 사용하라.**

짧은 이름이 좋은 이름은 아니다. WORK\_DAYS\_PER\_WEEK은 의미도 분명하며 검색하기도 쉬운 이름이다. E와 같은 한 글자 문자는 로컬 변수에만 사용한다.

**오브젝트의 종류에 따라 적합한 종류의 단어를 사용하라.**

클래스, 객체의 이름은 명사나 명사구가 적합하다. 동사를 사용하지 않고, data, info 등의 단어를 피하고 customer, account와 같은 단어가 좋다.

메소드 이름은 동사나 동사구가 적합하다. deletePage, save 등이 좋다.

**전체 코드에서 한가지 개념을 나타내는 단어를 통일하다.**

동일한 기능의 메소드를 클래스마다 fetch, retrieve, get으로 제각각 부르는 것을 피해라.

반대로, 한 단어를 두 가지 목적으로 사용해서도 안 된다.

**함수 이름에 맥락을 부여하라.**

firstName, lastName, houseNumber라는 이름으로는 맥락을 파악할 수 없지만, 그 앞에 addr라는 접두어를 추가하면 변수가 더 큰 구조인 “주소”에 속함이 분명해진다.

**다만, 불필요한 맥락은 제거해야 한다.**

accountAddress, customerAddress 클래스 인스턴스로 사용해서는 좋겠지만, 클래스 이름으로는 Adress가 더 좋다.

**Chapter 3. Functions – 짧고 명확한 함수를 만들어라**

**함수는 작게 만들어야 한다.**

중첩 구조가 없이 if, else, while 문 안에 함수를 호출하는 한 블록만이 들어가게 코드를 짜고 함수의 이름까지 적절하게 짓는다면 코드를 이해하기가 훨씬 편해진다.

**함수는 한 가지 일만을 해야 한다.**

함수가 이 조건을 만족하는지 의문스럽다면, 의미 있는 다른 함수를 추출해낼 수 있지 않은지 확인하고, 함수 내의 모든 문장이 동일한 추상화 수준에 있도록 한다.

이를 통해 함수가 확실히 한 가지 작업만 하도록 할 수 있고, 코드를 읽는 사람의 혼동을 막을 수 있다.

함수 내에 별도의 부수 효과를 구현하면 함수의 호출을 오히려 제한할 수 있다.

**함수의 기능을 잘 드러내는 서술적인 이름을 사용하라.**

다양한 함수들에 일관성 있게 명확한 이름을 붙인다면 그 길이는 중요하지 않다.

**함수의 인수는 적을수록 좋다.**

인수는 개념을 이해하기 어렵게 만든다. writeField(name)은 writeField(outputStream, name)보다 이해하기 쉽다.

인수가 여러 개라면, assertExpectedEqualsActual(expected, actual)과 같이 명확한 함수 이름으로 인수에 대한 이해를 도울 수 있다.

이상의 조건들을 갖춘 짧고 이름이 명확하며 체계적인 함수는 한번에 완성되지 않으므로, 여러 번의 수정을 거치는 것을 서슴지 않아야 한다.