Textbook Summary

Clean Code by Robert C. Martin

Chapter 2 & Chapter 3

20180916 임경빈 컴퓨터공학과, kbini@postech.ac.kr

Chapter 2 : Meaningful Names

Use Intention-Revealing Names

변수, 함수 또는 클래스의 이름을 지정할 때 기능과 사용법 등이 직관적으로 드러나야 한다. 예를 들어 int m; // money taken 과 같은 변수는 옆의 주석을 확인해야 그 의미를 알 수 있지만 int TakenMoney 와 같이 지을 경우 그 의미를 알기 쉽다.

Avoid Disinformation

여러 의미로 해석될 수 있거나 특별한 의미를 가지는 단어를 이용한 이름은 가급적 지양해야 한다. memberList 보다는 memberGroup 으로 짓는 것이 더 낫다.

Make Meaningful Distinctions

비슷한 이름을 가진 클래스나 변수가 있을 경우 혼동되기 쉬우므로 가급적 차별적인 이름을 사용해야 한다.

Use Pronounceable Names

Data Record를 줄여 DtRcd와 같은 변수명을 사용하면 읽기 어렵기 때문에 의사소통에 방해가 될 수 있으므로 DaRe와 같이 직관적으로 읽기 쉬운 이름을 사용하는 것이 좋다.

Use Searchable Names

하나의 알파벳이나 숫자 하나로만 이루어진 이름은 전체 텍스트에서 검색할 때 위치를 찾기 어렵다는 단점이 있다. 따라서 검색되기 쉬운 변수명을 사용하는 것이 좋다.

Class Names

클래스 이름을 지을 때는 명사 또는 명사구를 사용하는 것이 좋으며 Data, Info와 같은 혼동되기 쉬운 단어를 사용하는 것은 좋지 않다.

Method Names

클래스 네임과 달리 동사 또는 동사구를 사용해서 메소드임을 표현하는 것이 좋다. 접근자는 get, 변경자는 set, 조건자는 is로 시작하도록 통일하는 것이 좋다. 생성자를 오버로드할 경우에는 정적 메소드를 사용하고 private 선언을 한다.

Don’t Be Cute + Don’t Pun

특정 지역이나 문화권에서만 통용되는 기발한 이름은 혼동을 줄 수 있으므로 보편적이고 분명한 의미를 전달할 수 있는 단어를 사용해야 한다.

Pick One Word per Concept

추상적 컨셉 하나 당 단어 하나를 사용하는 것이 좋다.

Use Solution Domain Names + Use Problem Domain Names

만약 개발자와 함께 협업을 한다면 JobQueue와 같이 전문적인 용어를 사용하는 것은 바람직하다. 하지만 개발자가 없다면 문제 도메인과 관련이 있는 용어를 사용해야 한다.

Add Meaningful Context + Don’t Add Gratuitous Context

연결된 클래스와 함수를 같은 맥락으로 표현하면 의미가 묶여 직관적이다. 클래스의 이름 앞에 불필요한 맥락에서 같은 단어를 붙인다면 클래스를 검색할 때 모두 검색되는 등 불편함을 초래할 수 있으므로 간결한 단어로 이름 붙여야 한다.

Chapter 3 : Functions

Small!

함수를 작성할 때는 한 함수당 3~5줄 이내로 간결하게 작성해야 한다. 각 함수 별 들여쓰기를 줄여야 어떤 기능을 하는 메소드인지 알기 쉽다.

Do One Thing + Have No Side Effects + Command Query Seperation

하나의 함수는 한가지만의 기능을 가져야 한다. 함수가 여러 파트로 나뉜다면 그 함수는 여러 기능을 하는 셈이므로 함수를 나누어 따로 작성해야 한다. 또한 함수가 사이드 이펙트를 가진다면 그것 또한 하나의 역할을 하는 것이 아니므로 주의를 기울여야 한다.

또한 어떠한 함수는 객체 상태를 변경하거나 정보를 반환하거나 둘 중 하나의 역할 만을 해야 하므로 이 둘은 분리되어야 한다.

One Level of Abstraction per Function

함수 코드는 위에서 아래로 내려가면서 기능해야 한다. 만약 여러 번 다시 위로 올라간다면 그 기능을 이해하기 힘들어진다.

Use Descriptive Names

함수의 기능이 한가지만 존재한다면 그 기능을 사용하여 서술적인 이름을 지어야 한다. 그 기능을 미리 예상할 수 있어 이해가 쉬워진다.

Function Arguments

함수의 인수는 적을수록 좋다. 또한 함수의 입력 인수로 결과를 받는 경우는 이해하기 어려우므로 사용을 지양해야 한다. 또한 함수의 인수를 줄이기 위해 여러 인수를 하나의 클래스 변수로 줄일 수 있는지 고려해야 한다.

Prefer Exceptions to Returning Error Codes

Try/Catch 블록을 사용하면 오류 처리 코드가 따로 나오게 되므로 원래의 코드가 읽기 쉬워진다.

Don't Repeat Yourself

항상 중복을 하지 않도록 주의해야 한다. 중복은 코드를 팽창시키고 알고리즘을 변경해야 할 경우 몇배의 수정을 필요로 하게 된다. 중복을 피하기 위해 구조화된 프로그래밍, 구성 요소 지향 프로그래밍 등의 전략을 사용해야 한다.

How Do You Write Functions Like This?

소프트웨어를 작성하는 것은 글쓰기와 같이 처음에는 길고 복잡할 수 있으나 여러 번 수정을 거듭하여 간결하고 깔끔하게 작성하는 것이다.