**Clean Code Summary Chap 2-3**

컴퓨터공학과 20200102 송현지

**Chap 2 의미 있는 이름**

* 의도를 분명히 밝혀라
* 그릇된 정보를 피하라

유사한 개념은 유사한 표기법을 사용한다. 일관성이 떨어지는 표기법은 그릇된 정보이다.

소문자 L이나 O를 피하라.

* 의미 있게 구분하라

연속적인 숫자를 덧붙인 이름(a1, a2, …)를 피하기, 불용어는 중복이다.

* 발음하기 쉬운 이름을 사용하라
* 검색하기 쉬운 이름을 사용하라

문자 하나를 사용하는 이름과 상수는 텍스트 코드에서 쉽게 눈에 띄지 않는다.

* 자신의 기억력을 자랑하지 마라 (명료함이 최고)
* 기발한 이름은 피해라
* 클래스 이름&객체 이름: 명사구
* 메서드 이름: 동사나 동사구
* 한 개념에 한 단어를 사용하라
* 해법 영역/문제 영역에서 가져온 이름을 사용하라
* 의미 있는 맥락을 추가하라
* 불필요한 맥락을 없애라
* 인코딩을 피하라

인코딩한 이름은 발음하기 어렵고, 오타가 생기기도 쉽다.

* 헝가리식 표기법
* 멤버 변수 접두어
* 인터페이스 클래스와 구현 클래스

인터페이스 클래스의 이름과 구현 클래스 이름 중 하나를 인코딩해야 한다면 구현 클래스 이름을 택하겠다.

**Chap 3 함수**

* 작게 만들어라

중첩 구조가 생길만큼 함수가 커져서는 안 된다.

* + 블록과 들여쓰기

If 문/else 문/while 문 등에 들어가는 블록은 한 줄이어야 한다.

* 한 가지만 해라
  + 함수 내 섹션

한 가지 작업만 하는 함수는 자연스럽게 섹션으로 나누기 어렵다.

* 함수 당 추상화 수준은 하나로

위에서 아래로 코드 읽기: 내려가기 규칙

* Switch문

일반적으로 저자는 switch문을 한 번만 참아준다. (다형적 객체를 생성하는 코드)

* 서술적인 이름을 사용하라
* 함수 인수
  + 많이 쓰는 단항 형식
  + 플래그 이수
  + 이항 함수
  + 삼항 함수
  + 인수 객체
  + 인수 목록
  + 동사와 키워드
* 부수 효과를 일으키지 마라

명시하라

* + 출력 인수
* 명령과 조회를 분리하라
* 오류 코드보다 예외를 사용하라
  + Try/Catch 블럭 뽑아내기
  + 오류 처리도 한 가지 작업이다
  + Error.java 의존성 자석
* 반복하지 마라
* 구조적 프로그래밍
* 함수를 짜는 방법