클린코드 정리

이책을 읽는 이유는

1. 프로그래머라서
2. 더나은 프로그래머가 되려고

이 책은 사방에 코드가 많다. 그래서 어렵다. 그 정도를 감안하고 읽고 있었을 것.

* 코드가 존재하리라

1. 코드는 요구사항을 표현
2. 나중에 코드는 자동으로 나올 시대가 올 것

* 나쁜 코드

1. 나쁜 코드는 대충 짠 코드가 쌓이고 쌓이면서 나쁜 영향을 끼키게 되는 코드가 나쁜코드이다.
2. 나쁜 코드로 인해 회사가 망하는 경우도 존재한다.
3. 르블랑의 법칙을 알아야 할 것

* 나쁜코드로 치르는 대가

1. 얽히고 설킨 코드를 해독하여 또 얽히고 설킨 코드로 쌓여가고 청소할 시간도 없다. 불가항력이다.
2. 나쁜 코드가 쌓일수록 팀 생산성도 떨어진다. 그러다 0에 근접

* 원대한 재설계의 꿈
  + 기존 시스템이 너무 개쓰레기이기 때문에 기존시스템을 재설계하는 시간도 상당히 걸리게 된다.
* 태도
  + 상사가 하라는대로 ㅈ같이 짠다면 그 상사는 전문가 답지 못하다.
  + 좋은 코드를 사수하는 것 자체가 프로그래머의 책임이다.
* 원초적 난제
  + 나쁜 코드를 양산하지 않기 위해 코드를 항상 깨끗하게 유지하는 방법을 써야한다.
* 깨끗한 코드란?
  + 노련한 프로그래머들의 대답
* 비야네 스트롭스트룹
  + C++창시자
  + 우아하고 효율적인 코드, 논리가 간단, 의존성을 최대한으로 줄임
  + 오류는 명백한 전략에 의거하여 철저히 처리, 한가지를 제대로 함
* 그래디 부치
  + 단순하고 직접적
  + 깨끗한 코드는 잘 쓴 문장처럼 읽힘, 결코 설계자의 의도를 숨기지 않음
  + 오히려 명쾌한 추상화와 단순한 제어문으로 가득
* 빅 데이브 토마스
  + 작성자가 아닌 사람도 읽기 쉽고 고치기 쉬워야 한다
  + 단위테스트 케이스와 인수테스트 케이스가 존재
  + 의미있는 이름을 붙일 것
  + 특정 목적을 달성하는방법은 하나만 제공
* 마이클 페더스
  + 깨끗한 코드의 특징은 많지만 그 중에서도 누군가 주의 깊게 짰다는 느낌을 줌
  + 고치려고 살펴봐도 딱히 손댈 곳이 없음
  + 작성자가 이미 모든 사항을 고려했으므로 고칠 궁리를 하다보면 언제나 제자리로 돌아옴
* 론 제프리스
  + 모든 테스트를 통과한다
  + 중복이 없다.
  + 시스템 내 모든 설계 아이디어를 표현한다
  + 클래스 메서드 함수 등 최대한 줄인다
* 워드 커닝햄
  + 짐작했던 기능이 각 루틴이 그대로 수행한다면 깨끗한 코드라고 불러도 된다.
  + 코드가 그 문제를 풀기 위한 언어로 보인다면 아름다운 코드라고 해도 됨
* 보이 스카우트 원칙
  + 체크아웃 할때보다 좀 더 깨끗한 코드를 체크인해라
* 결론
  + 예술에 대한 책을 읽는다고 예술가가 된다는 보장은 없다. 책은 단지 다른 예술가가 사용하던 기법 그리고 생각하는 방식만 소개할 뿐이다.
  + 무수히 많은 연습이 필요하다.