깨끗한 코드

상태	할 일
속성	
작성일시	@2021년 9월 27일 오전 12:03

여러분이 이 책을 읽고 있다면, 이유는 두가지다.

1. 프로그래머라서
2. **더 나은 프로그래머가 되려고.**

코드가 존재하리라

- 코드가 사라질 가망은 **전혀 없다!** → 지금은 바뀌지 않았을까?
- 프로그래밍? == 기계가 실행할 정도로 **상세하게** 요구사항을 명시하는 작업.
- 요구사항에 더욱 가까운 언어를 만들수도 있고, 정형 구조를 뽑아내는 도구를 만들어 낼 수도 있지만, 어느 순간에나 **정밀한** 표현이 필요하다!

나쁜 코드

- 회사가 망한 원인은 나쁜 코드 탓이었다.
- 나중에 더 좋은 코드로 바꿔야지~ → 나중은 결코 오지 않는다....ㅠ

나쁜 코드로 치르는 대가

- 나쁜 코드가 쌓일수록 팀 생산성은 떨어진다. → 코드를 '해독'하는데 걸리는 시간이 늘어난다는 것.
- 코드가 나쁘다 → 새로운 타이거 팀을 꾸려 재설계한다. → 코드가 나빠진다 → ...
- 깨끗한 코드를 만드는 노력 == 비용을 절감하는 방법 & 전문가로서 살아남는 길.

원초적 난제

- 기한을 맞추려면 나쁜 코드를 양산해야한다? → 틀렸음.
- 코드를 최대한 깨끗하게 유지하는 습관 → 빨리가는 유일한 방법.

깨끗한 코드를 작성하려면?

- 코드 감각 : 나쁜 코드 → 좋은 코드로 바꾸는 열쇠.
    - 이를 원래 선천적으로 가지고 있을 수도 있고, 아닐 수도 있다.

깨끗한 코드란?

비야네 스트롭스트룹 : 우아하고(보기에 즐거운) 효율적인 코드

- 철저한 오류 처리 : 세세한 사항까지 꼼꼼하게 처리하는 코드

그래디 부치 : 설계자의 의도를 숨기지 않는, 잘 쓴 문장처럼 읽히는 코드.

- 가독성을 강조.

큰 데이브 토마스 : 작성자가 아닌 사람도 읽기 쉽고 고치기 쉬운 것.

- 다른 사람이 고치기 쉬운 코드! → 인간이 읽기 좋은 코드를 작성해라.

마이클 페더스 : 모두를 아우르는 큰 특징, 누군가 주의 깊게 짰다는 느낌을 주는 코드

- 클린코드 : 코드를 주의깊게 짜는 법.

론 제프리스 : 중복을 피하라. 한 기능만 수행하라. 제대로 표현하라. 작게 추상화하라.

워드 커닝햄 : 짐작했던 기능이 수행된다면 깨끗한 코드, 그 문제를 풀기위한 언어처럼 보인다면? 아름다운 코드

- 코드를 독해하느라 머리를 쥐어짤 필요가 없는 코드여야한다.
- 언어를 단순하게 보이도록 만드는 책임은 프로그래머에게 있다.

캠프장은 처음 왔을 때보다 더 깨끗하게 해놓고 떠나라.

== 직장에 처음 들어갔을 때보다 더 깨끗한 코드를 남기고 떠나라.

감상평

뒤에 나올 깨끗하게 코드를 짜는 방법들이 왜 필요한지와 클린 코드란 무엇인지를 다양한 관점에서 볼 수 있었던 챕터이다. 프로그래머가 되기로 마음먹은 순간부터 어떻게하면 좋은 코드를 짤 수 있을지 고민하게 되었는데 그 고민들을 체험해본 사람들이 말 해주는 것들이라 더 마음에 와닿았던 것 같다.

하지만, 이렇게 좋은 글들을 읽는다고 해서 엄청나게 도움이 되는 것은 아니라는 것을 챕터 마지막에 깨달을 수 있었다. "연습해 연습!"