프로그래밍 역량 강화 전문기관, 민코딩

클린코드

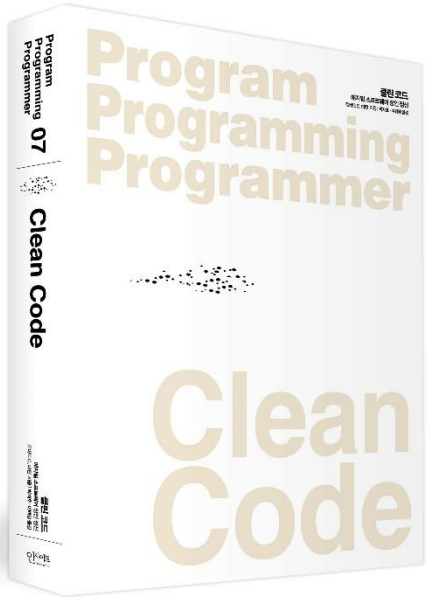
목차

Clean Code 소개

클린 코드

로버트 C 마틴의 지침

- 절대적으로 지켜야 하는 내용이 아니다.
- 팀이나 공동체에서 서로 동의하는 합리적인 원칙을 세우기 위한 소통
- Clean Code는 소통을 위한 기초지식을 제공하고 생각할 거리를 던져주는 책



대단한 비밀은 없다.

깨끗한 코드가 술술 나오는 비밀은 없다.

오랫동안 일했던 개발자라면
한번쯤 고민해보고 생각한 이야기를 체계적으로 정리한 것

개발자마다 생각이 다를 수 있다.

전문가는 세세한 디테일에 신경을 많이 쓴다.
다른 말로는 사소한 것을 중요하게 여긴다고 할 수 있다.

설계가 중요한 것에 대해 누구나 동의한다.
그 세세함에 대해서는 생각이 다를 수 있다.

- 한 엔지니어의 생각을 적은 것이기에,
  이 내용을 받아들이지 않으면 잘못된 것으로 생각하면 안된다.

- 동의하는 부분은 내 지식으로 받아들이자.

클린 코드는 빠른 유지보수가 가능하다.

SW에서는 제조 보다는 유지보수 업무가 훨씬 많다.

- 따라서 유지보수 하기 쉽게 만들어야 한다.

클린 코드는 유지보수가 쉽도록 만들기 위한 노력이다.

- 클린 코드는 생산보다는 유지보수에 초점을 맞춘다.

클린 코드가 되지 않는 이유

Clean Code 상태에서 유지보수는 빠르게 된다는 것을 알고 있다.
그럼에도 불구하고, 점차 Clean하지 않는 코드를 유지하는 이유

1. 기한을 맞추기 위해, 나쁜 코드를 양산한다.
2. 나쁜 코드 위에서 유지보수를 지속적으로 하게 된다.
3. 점점 더 나쁜 코드 위에서 유지보수를 하므로, 속도가 점차 느려진다.

코드를 깨끗하게 유지하려는 노력이
오히려 기한을 맞출 수 있는 좋은 방법이다.

클린 코드는 습관화가 되어야한다.

클린 코드는 청소로 비유할 수 있다.

- 정리 : 적절한 명명법으로 정리
- 정돈 : 코드는 누구나 예상하는 위치에 있어야 한다.
- 청소 : 쓰레기는 치운다. 필요 없는 주석은 지운다.
- 규율화 : 청소 방식, 그룹 내 구현 스타일 기법 통일화
- 생활화 : 자기 작품을 자주 돌아본다.

✓깔끔한 방 상태를 유지하려는 노력이, 항시 깨끗한 방을 만드는 것 처럼 클린 코드를 유지하려는 노력은 습관화가 되어야 한다.

Clean Code의 다양한 정의

클린 코드에 대한 다양한 정의

“

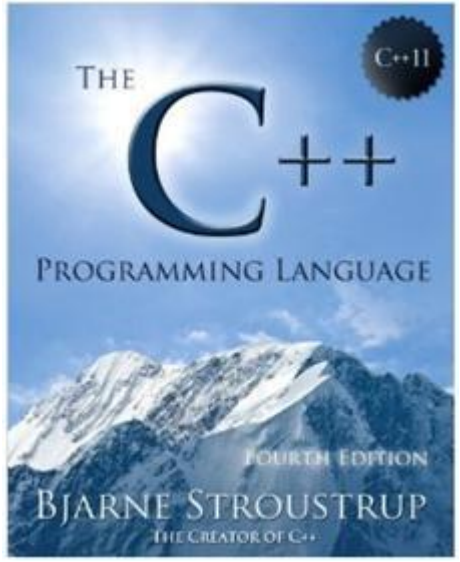
✓ 나는 우아하고 효율적인 코드를 좋아한다. 논리가 간단해야 버그가 숨어들지 못한다.

의존성을 최대한 줄여야 유지보수가 쉬워진다. 오류는 명백한 전략에 의거해 철저히 관리한다.

성능은 최적으로 유지해야 사람들이 원칙 없는 최적화로 코드를 망치려는 유혹에 빠지지 않는다.

✓ 깨끗한 코드는 한 가지를 제대로 한다.

”



Bjarne Stroustrup, inventor of C++:

클린 코드에 대한 다양한 정의

“

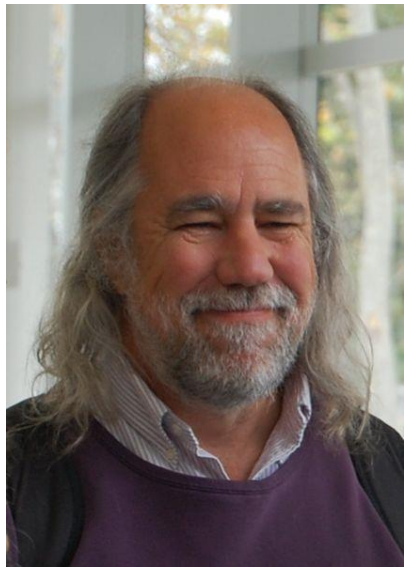
깨끗한 코드는 단순하고 직접적이다. 깨끗한 코드는 잘 쓴 문장처럼 읽힌다.

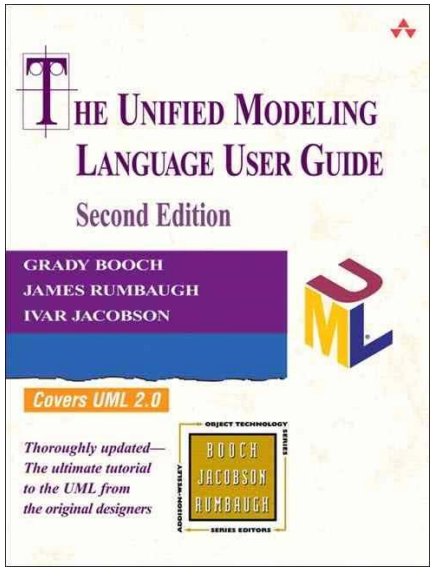
깨끗한 코드는 결코 설계자의 의도를 숨기지 않는다.

오히려 명쾌한 추상화와 단순한 제어문으로 가득하다.

”

Grady Booch,
author of Object-Oriented Analysis
and Design with Applications





클린 코드에 대한 다양한 정의

“

깨끗한 코드의 특징은 많지만 그 중에서도 모두를 아우르는 특징이 하나 있다.

깨끗한 코드는 언제나 누군가 주의 깊게 짰다는 느낌을 준다.

고치려고 살펴봐도 딱히 손 댈 곳이 없다.

작성자가 이미 모든 사항을 고려했으므로,

고칠 궁리를 하다 보면 언제나 제자리로 돌아온다.

”

마이클 페더스

Working Effectively with Legacy Code 저자

클린 코드에 대한 다양한 정의

“

코드를 읽으면서 짐작했던 기능을 각 루틴이 그대로 수행한다면

깨끗한 코드라 불러도 되겠다.

깨끗한 코드는 읽으면서 놀랄 일이 없어야 한다.

”

워드 커닝햄
Wiki 창시자, XP 공동 창시자

Clean Code를 만드는 마인드

클린 코드 마인드 1 : 저자 마인드

우리는 저자다.

책 저자는 글을 쓰고, 본인이 쓴 글을 읽고, 읽고, 또 읽고, 다시 읽는다.
독자 입장에서 다시 읽고 고친다. 독자와 잘 소통이 되는지 확인한다.

코드를 짤 때는 자신이 저자임을 기억하고
끊임없이 기존 코드를 읽어야한다.

소스코드는 이야기이다.
우리는 이야기를 작성하는 작가이다.

클린 코드 마인드 2 : 보이스카우트 규칙

시간이 지나도 언제나 깨끗하게 유지해야 한다.
*** 지속적인 개선**

캠프장은 처음 왔을 때 보다 더 깨끗하게 해놓고 떠나야한다.
Pull 할 때보다, 더 깨끗한 코드를 Push 한다면
코드는 나빠질 일이 없다.

이는 시간이 지날수록 코드가 좋아지는 프로젝트가 된다.

[참고] 클린코드 주요 영역

✓01. 깨끗한 코드란?
✓02. 의미 있는 이름
✓03. 함수
✓04. 주석
✓05. 형식 맞추기
✓06. 객체와 자료구조
✓07. 에러 핸들링
✓08. 경계
✓09. 단위 테스트
✓10. 클래스
✓11. 시스템
✓12. 창발성
✓13. 동시성
✓14. 점진적인 개선
✓15. JUnit 들여다보기
✓16. SerialData 리팩터링
✓17. 냄새와 휴리스틱

다루는 내용

다루지 않지만,
자가 학습 권장

학습 목표와
맞지 않음