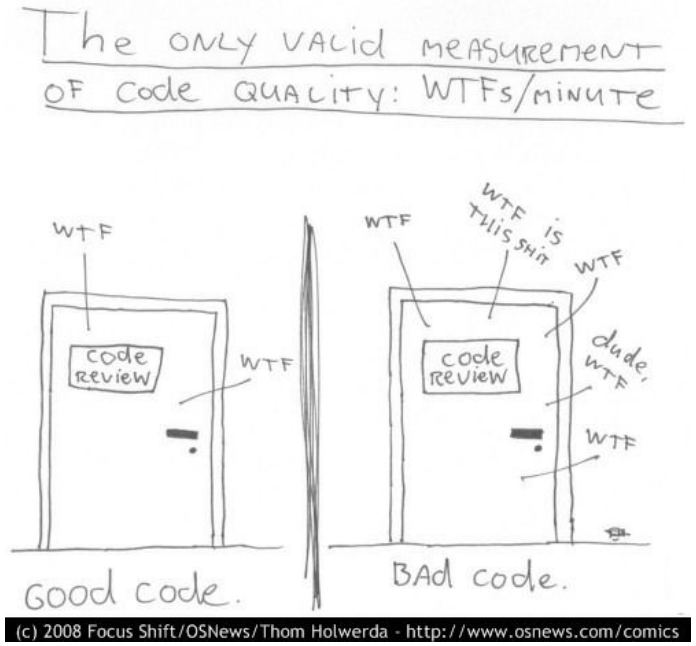
클린코드 스터디 1주차. 복간에 부쳐

“출판사 요청에 따라 2년이 훨씬 넘어 다시 한번 본문을 검토해보니 역시 좋은 책은 시간이 흘러도 유효한 내용을 담고 있따는 사실이 느껴졌다. 자바 버전은 높아졌고, 프레임워크와..... 등 주변 환경이 많이 변했지만, ‘클린 코드’ 에서 설명하는 원칙과 기본은 변하지 않았따고 보면 틀림없다.”

“프로그래밍도 유행이나 스타일이 조금씩 바뀌기에 본문에서 소개하는 원칙이 현실과 잘 맞지 않는 부분 역시 존재하기 마련이다. 예를 들어, 로버트 마틴이 조잡하기에 피해야 한다고 충고한 기차 충돌 형태의 코드는 요즘 들어 읽기 쉬운 코드를 지향하는 **fluent interface** 에 등장하고 있다. 따라서 이 책에 나오는 모든 지침은 로버트 C. 마틴이 이미 밝혔듯이 절대적이라 생각하면 안 되며, 언제든지 개선의 여지가 있다고 생각하는 편이 바람직하다.”



* fluent interface ?

객체를 생성할 때, 혹은 어떤 로직을 처리할 때 chaining 을 지원하는 패턴
Builder Pattern 과 같은 종류인 것 같음

클린코드 스터디 1주차. 1장 〈깨끗한 코드〉

코드가 존재하리라

“**기계가 실행할 정도로 상세하게 요구사항을 명시하는 작업**, 바로 이것이 프로그래밍이다. 이렇게 명시한 결과가 바로 코드다.”

“요구사항을 모호하게 줘도 우리 의도를 정확히 꿰뚫어 프로그램을 완벽하게 실행하는 그런 기계 말이다. 절대로 불가능한 기대다. 창의력과 직관을 보유한 우리 인간조차도 고객의 막연한 감정만 갖고는 성공적인 시스템을 구현하지 못한다.”

- 최근에는 상황이 조금 바뀐 것 같기도

“궁극적으로 **코드는 요구사항을 표현하는 언어**라는 사실을 명심한다. 요구사항에 더욱 가까운 언어를 만들 수도 있고, 요구사항에서 정형 구조를 뽑아내는 도구를 만들 수도 있다. 하지만 **어느 순간에는 정밀한 표현이 필요**하다. 그 필요성을 없앨 방법은 없다. **그러므로 코드도 항상 존재**하리라.”

클린코드 스터디 1주차. 1장 〈깨끗한 코드〉

나쁜 코드

“그들은 출시에 바빠 코드를 마구 짰다. **기능을 추가할수록 코드는 엉망**이 되어갔고, **결국은 감당이 불가능한 수준**에 이르렀다. 회사가 망한 원인은 바로 나쁜 코드 탓이었다.”

“**어째서 나쁜 코드를 짰는가?** 급해서? 서두르느라? 아마 그랬으리라. 제대로 짤 시간이 없다고 생각해서, 코드를 다듬느라 시간을 보냈다가 상사한테 욕 먹을까봐, 지겨워서 빨리 끝내려고, 다른 업무가 너무 밀려 후딱 해치우고 밀린 업무로 넘어가려고... **모두가 겪어본 상황**이다.”

“우리 모두는 자신이 짠 쓰레기 코드를 쳐다보며 나중에 손보겠다고 생각한 경험이 있다. 우리 모두는 대충 짠 프로그램이 돌아간다는 사실에 안도감을 느끼며 **그래도 안 돌아가는 프로그램보다 돌아가는 쓰레기가 좋다고 스스로를 위로한 경험**이 있다. 다시 돌아와 나중에 정리하겠다고 다짐했었다. 물론 그때 그 시절 우리는 르블랑의 법칙을 몰랐다. **나중은 결코 오지 않는다**.”

클린코드 스터디 1주차. 1장 〈깨끗한 코드〉

나쁜 코드로 치르는 대가

“나쁜 코드는 개발 속도를 크게 떨어뜨린다. 프로젝트 초반에는 번개처럼 나가다가 1-2년만에 굼뱅이처럼 기어가는 팀도 많다. **코드를 고칠 때 마다 엉뚱한 곳에서 문제가 생긴다**. 간단한 변경은 없다. **매번 얽히고설킨 코드를 ‘해독’해서 얽히고설킨 코드를 더한다**. 시간이 지나면서 쓰레기 더미는 점점 높아지고 깊어지고 커진다. 청소할 방법이 없다. **불가항력**이다.”

“... 생상선을 증가시키려는 희망을 품고 프로젝트에 인력을 추가로 투입한다. 하지만 새 인력은 시스템 설계에 대한 조예가 깊지 않다. 설계 의도에 맞는 변경과 설계 의도에 반하는 변경을 구분하지 못한다. 게다가 새 인력과 팀은 생산성을 높여야 한다는 극심한 압력에 시달린다. 그래서 **결국은 나쁜 코드를 더 많이 양산**한다. 덕택에 생산성은 더더욱 떨어져 거의 0이 된다.”

클린코드 스터디 1주차. 1장 〈깨끗한 코드〉

원대한 재설계의 꿈

“방금 한 이야기를 일부라도 겪었다면 **시간을 들여 깨끗한 코드를 만드는 노력이 비용을 절감하는 방법일 뿐만 아니라 전문가로서 살아남는 길**이라는 사실을 인정하리라.”

태도

“좋은 코드를 사수하는 일은 바로 우리 **프로그래머들의 책임**이다.”

원초적 난제

“나쁜 코드를 양산하면 기한을 맞추지 못한다. 오히려 엉망진창인 상태로 인해 속도가 곧바로 늦어지고, 결국 기한을 놓친다. 기한을 맞추는 유일한 방법은, 그러니까 빨리 가는 유일한 방법은, **언제나 코드를 최대한 깨끗하게 유지하는 습관**이다.”

클린코드 스터디 1주차. 1장 〈깨끗한 코드〉

깨끗한 코드라는 예술 ?

“깨끗한 코드와 나쁜 코드를 구분할 줄 안다고 **깨끗한 코드를 작성할 줄 안다는 뜻은 아니다**.”

“‘코드 감각’ 이 없는 프로그래머도 때로는 나쁜 모듈을 알아본다. 하지만 그것으로 끝이다. ‘코드 감각’ 이 있는 프로그래머는 **나쁜 모듈을 보면 좋은 모듈로 개선할 방안을 떠올린다**. ‘코드 감각’ 으로 최고 방안을 선택한 후 여기서 거기까지 이동하는 경로를 계획한다.”

깨끗한 코드란 ?

“나는 우아하고 효율적인 코드를 좋아한다. 논리가 간단해야 버그가 숨어들지 못한다. 의존성을 최대한 줄여야 유지보수가 쉬워진다. 오류는 명백한 전략에 의거해 철저리 처리한다. 성능을 최적으로 유지해야 사람들이 원칙 없는 최적화로 코드를 망치려는 유혹에 빠지지 않는다. **깨끗한 코드는 한 가지를 제대로 한다**.”

- *C++ 창시자이자 The C++ Programming Language 저자*

“**깨끗한 코드는 단순하고 직접적**이다. **깨끗한 코드는 잘 쓴 문장처럼 읽힌다**. 깨끗한 코드는 결코 설계자의 의도를 숨기지 않는다. 오히려 명쾌한 추상화와 단순한 제어문으로 가득하다.”

- *그래디 부치*

클린코드 스터디 1주차. 1장 〈깨끗한 코드〉

우리는 저자다

“코드를 읽는 시간 대 코드를 짜는 시간 비율이 10 대 1을 훌쩍 넘는다. 새 코드를 짜면서 우리는 끊임없이 기존 코드를 읽는다. 비율이 이렇게 높으므로 **읽기 쉬운 코드가 매우 중요**하다. 비록 읽기 쉬운 코드를 짜기가 쉽지는 않더라도 말이다. 하지만 **기존 코드를 읽어야 새 코드를 짜므로 읽기 쉽게 만들면 사실은 짜기도 쉬워진다**.”

보이스카우트 규칙

“**캠프장은 처음 왔을 때보다 더 깨끗하게 해놓고 떠나라**.”

“체크아웃할 때보다 좀 더 깨끗한 코드를 체크인한다면 코드는 절대 나빠지지 않는다. **한꺼번에 많은 시간과 노력을 투자해 코드를 정리할 필요가 없다**. 변수 이름 하나를 개선하고, 조금 긴 함수 하나를 분할하고, 약간의 중복을 제거하고, 복잡한 if 문 하나를 정리하면 충분하다.”

결론

“**연습해, 연습!**”
