# Refactoring with Clean Code

수원 개발자 스터디 - 최혁

왜 이 주제로
스터디하게 되었
나요?

# 경력 내내 궁금한 내용

- 대기업 전산실 4년 경력 내내
  - 남의 소스 보고, 이해하고, 기능추가, 버그 수정
- 프리랜서 경력
  - 신나게 신규 기능 추가 개발
  - 뒤돌아보고 수정하려니 Method가 100줄 200줄
  - Method를 나누는 기준이 있을까?에서 시작

# 커뮤니티에 질문

- Q:Method를 나누는 기준이 있나요?
- A: SRP(단일 책임 원칙)을 기억하세요.
- Q: 나누다보면 끝이 없어요 ㅠㅠ

- A: move() { copy(); delete(); }
  copy() { 블라블라 } delete() { 블라블라}
  Copy는 다시 스트림 open,write로 이뤄지고
  스트림 open은 이름 파싱과 iNode주소로 얻는 시스템 콜
  스트림 write는 open과 write 그리고 flush로 이뤄지겠죠
  하지만 하이레벨에서 보면 단지 move 하나일 뿐이구요
  이런식으로 계층별로 한가지 일만 하는 메소드를 많이 조합하는 코드가 유지보수하기 좋아요.

# 마냥 외우기만했던 SOLID

- 우리가 OOP를 배우며
  그냥 그러려니 외우기만 했던 SOLID의 지향점

- SOLID 원칙들은 소프트웨어 작업에서 프로그래머가 소스 코드가 읽기 쉽고 확장하기 쉽게 될 때까지 소프트웨어 소스 코드를 **리팩토링**하여 코드 냄새를 제거하기 위해 적용할 수 있는 지침이다. - 출처: 위키피디아

- SOLID는 결국 좋은 코드를 작성하기 위한 틀이고, 그것의 지향점을 향해 리팩토링을 실시할때, 좋은 코드에 다가갈 수 있다.

# 리팩토링은 무엇인가?

- 정의
  - 소프트웨어를 보다 쉽게 이해할 수 있고, 적은 비용으로 수정할 수 있도록 겉으로 보이는 동작의 변화없이 내부 구조를 변경하는 것.
- 목적
  - 소프트웨어를 보다 이해하기 쉽고, 수정하기 쉽도록 만든다.
  - 겉으로 보이는 소프트웨어의 기능을 변경하지 않는다.

# 왜 Refactoring을 해야하는가?

- 소프트웨어의 디자인을 개선시킨다.
- 소프트웨어를 더 이해하기 쉽게 만든다.
- 버그를 찾도록 도와준다.
- 프로그램을 빨리 작성하도록 도와준다. (중요!)

# 언제 Refactoring을 해야하는가?

- 같은 작업의 삼진 아웃 때
- 기능을 추가할 때
- 버그를 수정할 때
- 코드를 검수할 때

# 언제 하지 말아야 하는가?

- 코드를 처음부터 다시 작성해야 할 수준의 막장코드
  - 명확한 기준은 없다. ex) 안돌아가는 코드
- 마감일이 가까울때
  - 리팩토링의 생산성은 납기 후에 가시화 되니 쓸데 없다.

# 우리가 제일 궁금한 것

# 그럼 어떻게 적용 합니까?

이미 기존 소스가...

# 1. 테스트 코드 작성

- 하지만 테스트 코드 작성은 무척 어렵습니다.
- 미리 테스트 코드를 작성했다면, 테스트 하기 위해 Dependency가 느슨한 코드라 해당 클래스만 집중해서 테스트 할 수 있지만, 앞에서 주는 파라미터부터 온갖 데이터셋이 변한되어 들어옵니다.

# 새로운 개발이 떨어지면 테스트를 작성해라!

- 모든 코드를 테스트 = 초등학교 때 방학숙제 하는 기분
- 진이빠진다.
- 결과물이 엉망이다.
- 그냥 했다는데 의의를 둔다.

- 시작은 새로운 기능개발 or 버그 수정에 집중하면 됩니다
    - 기존프로젝트에 새로운 기능을 개발할 때, 보통 기존 소스의 수정에서 출발합니다.
        - 저는 이럴때 테스트 코드를 작성합니다.
- Q:이미 Dependency 개떡 치떡인데요?
    - A: 저의 Tip은 파라미터 객체를 Serialize 해버립니다.

# 테스트 하면서 어려운 점

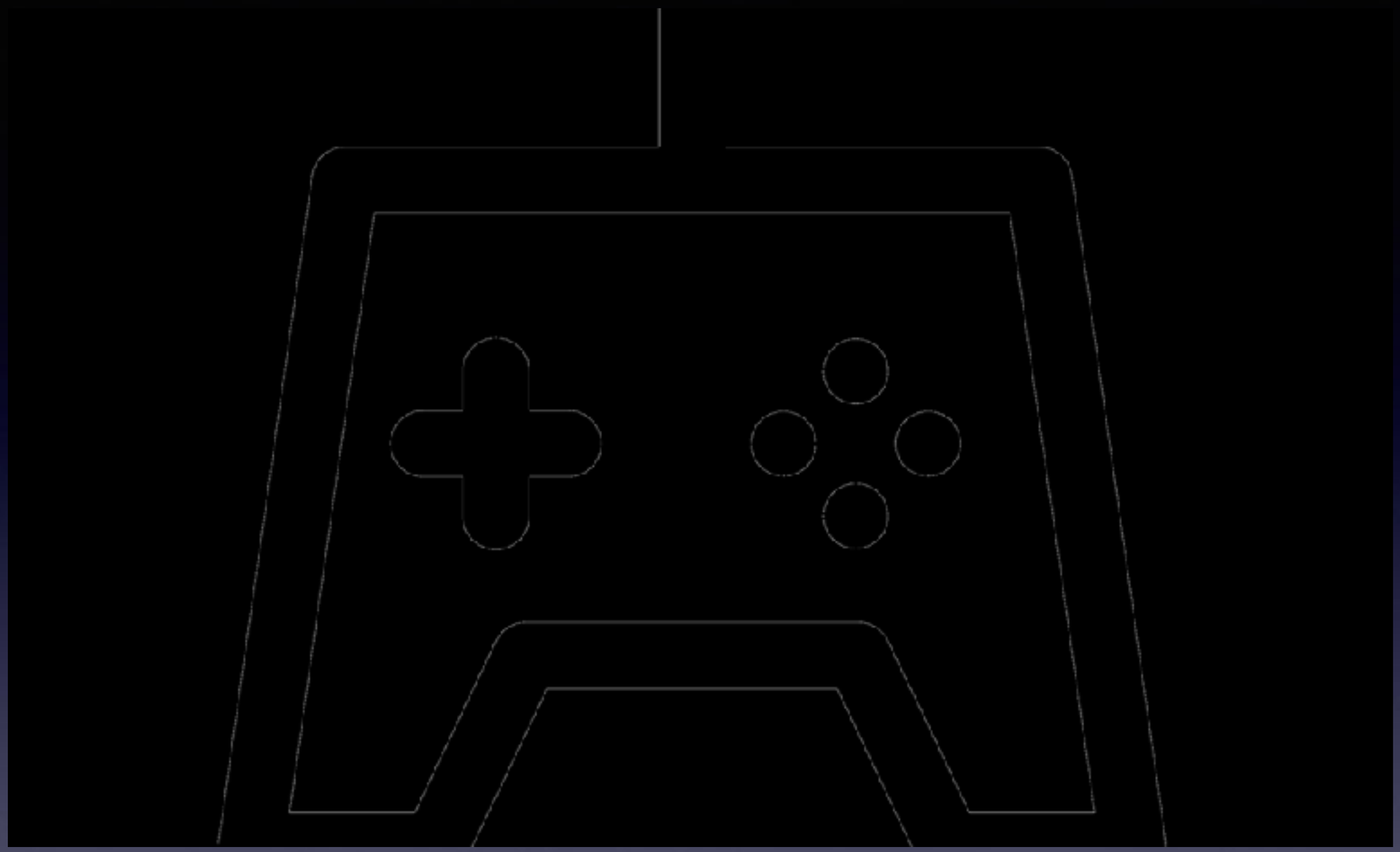
- 가장 어려운점은 경계 설정입니다.
(테스트 바운더리)

  - 테스트는 모든 케이스를 커버하는 게 아닙니다.

    - ex) 2 ~ 8까지 print함수.

      - 1,2,7,8만 테스트 해보면, 경계를 설정해 테스트

      - 시간이 지나 5에서 에러가 났다? 그때 테스트를 추가하면 됩니다. (회귀 테스트)



# 결론

리팩토링은 내일의 나를 위한 생산성 향상이 목적!
ex) 빨리 퇴근하고 게임하기