# 리팩토링

외부 동작을 바꾸지 않으면서 내부 구조를 개선하는 것

코드를 변경했을 때 외부 동작이 바뀌지 않았다는 것을 증명해야 함

# 마틴 파울러: 리팩토링

코드스멜 찾는 법

단위 테스트 작성

레거시 코드를 리팩토링 하는 방법

# 코드 스멜

# 레거시 코드

내가 작성하는 순간부터

테스트가 없는 코드

# Test Coverage

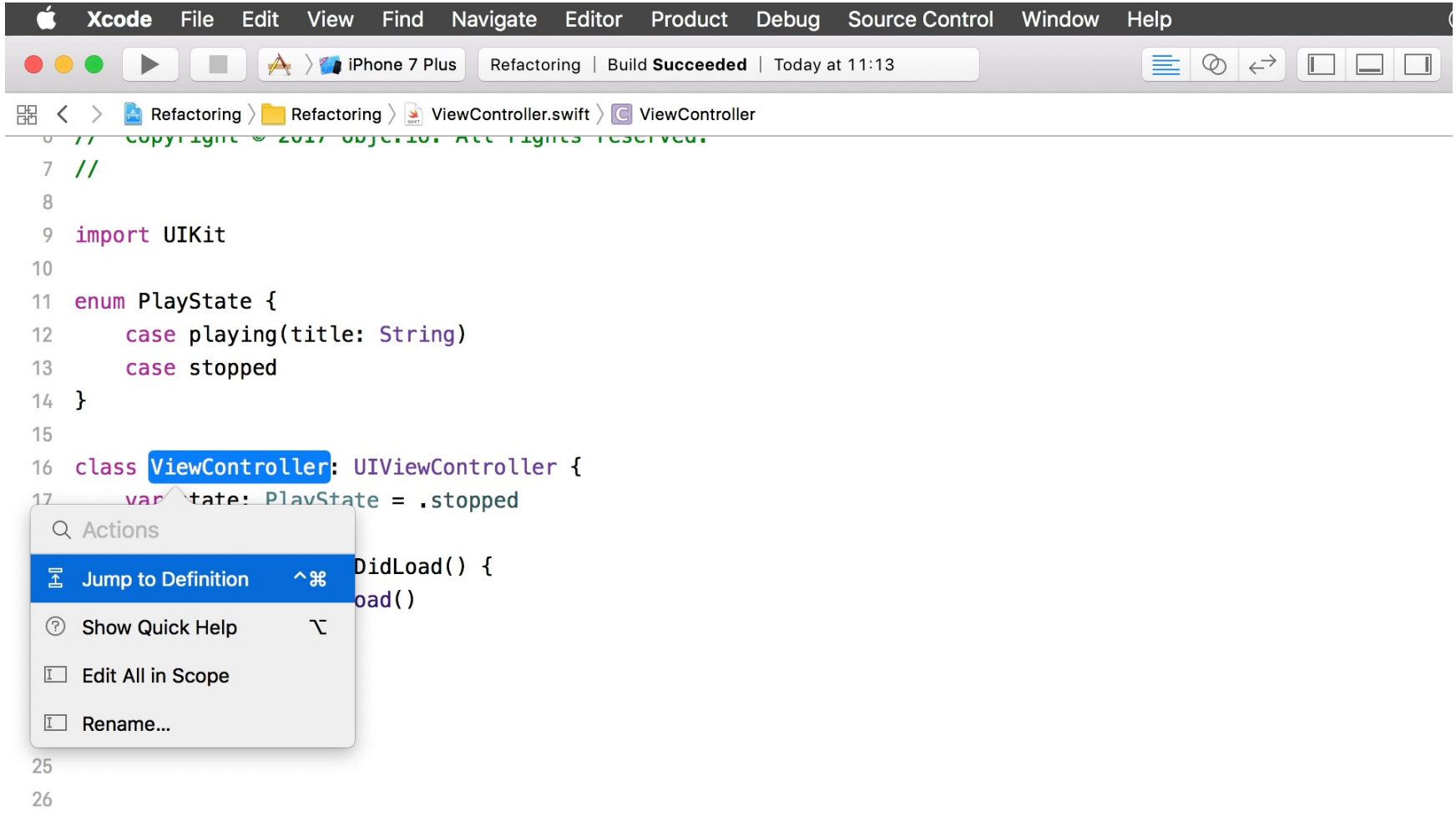
Code Coverage

Complexity Coverage

# XCode Refector

# Renaming

**command + shift + A**

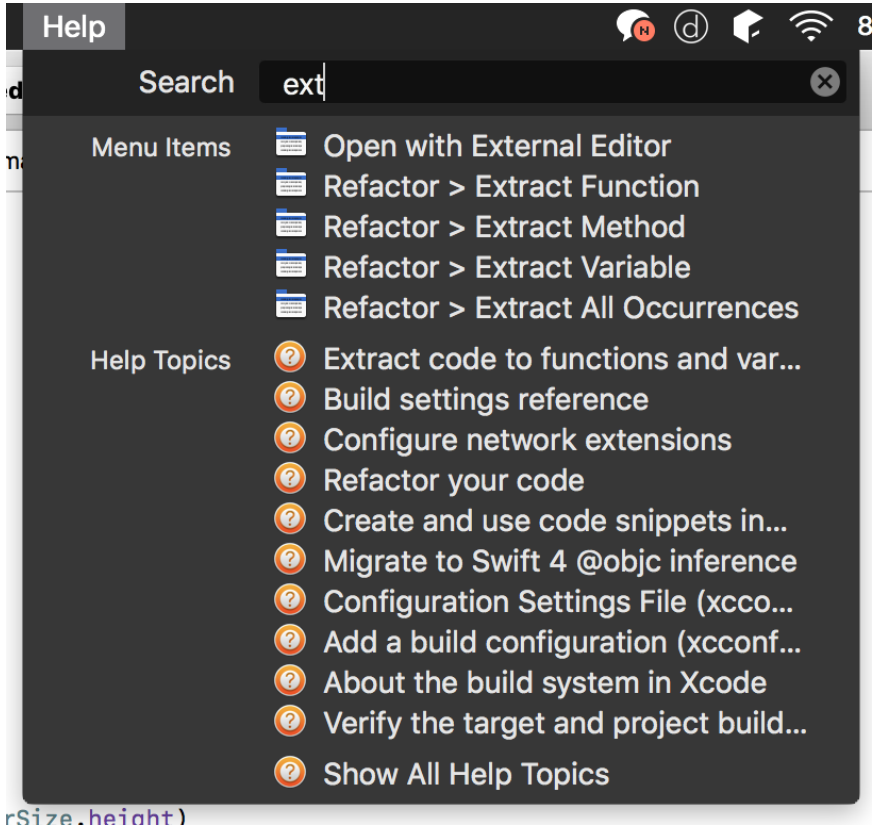


# Extractring Expression

**Editor > Refector > Extract Expression**

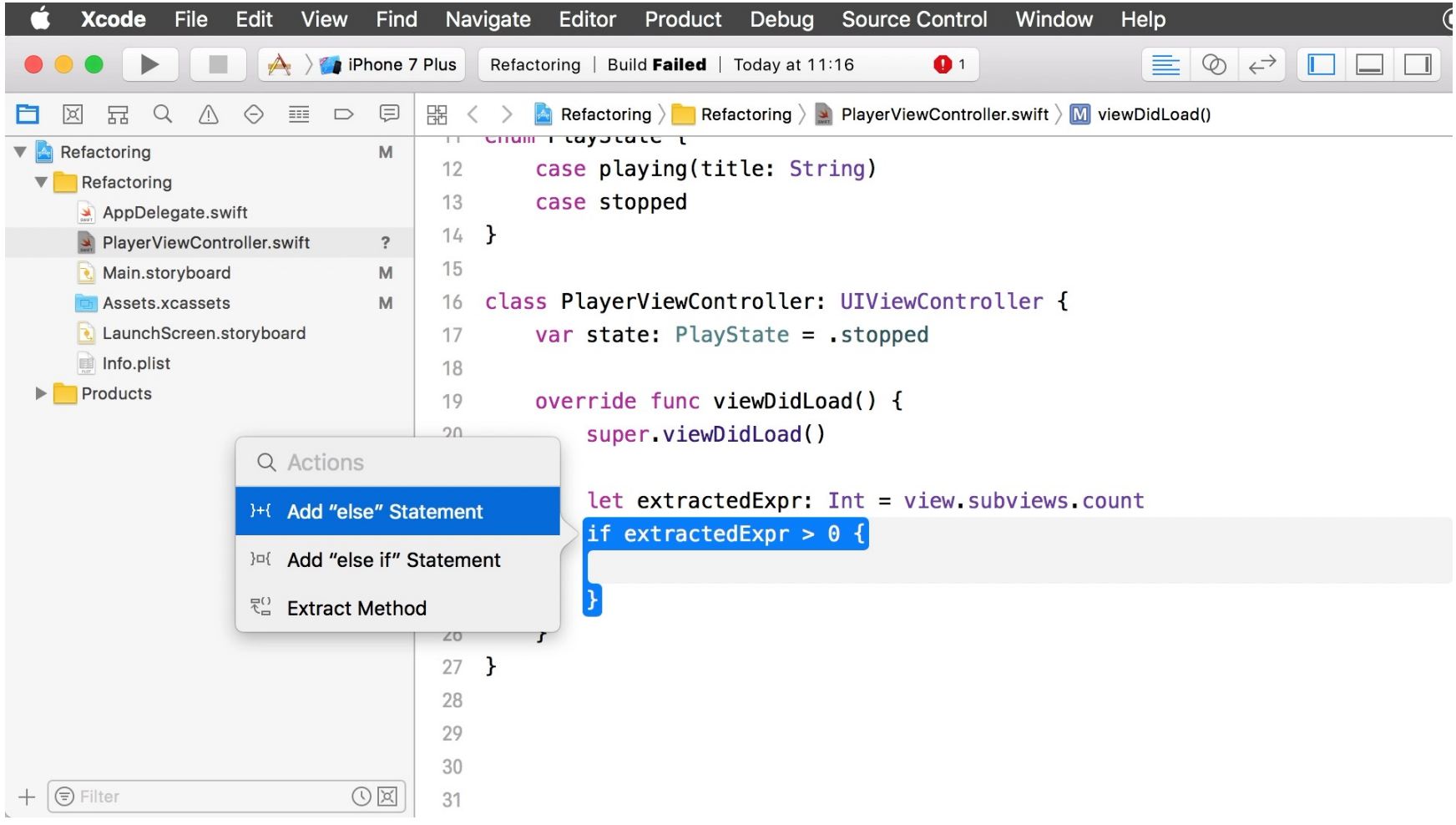
```
// Before
if view.subviews.count > 0 {

}

// After
let extractedExpr: Int = view.subviews.count
if extractedExpr > 0 {

}
```

`command + shift + ? > Extract`



# Various Useful Actions

**command + shift + A**

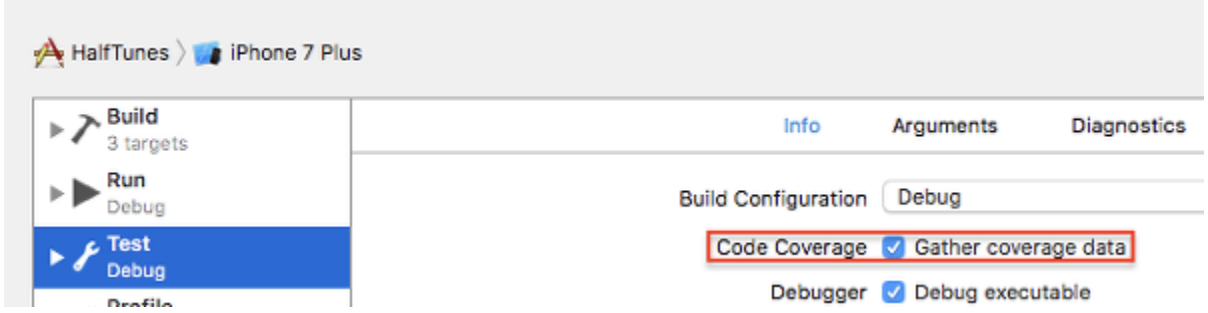


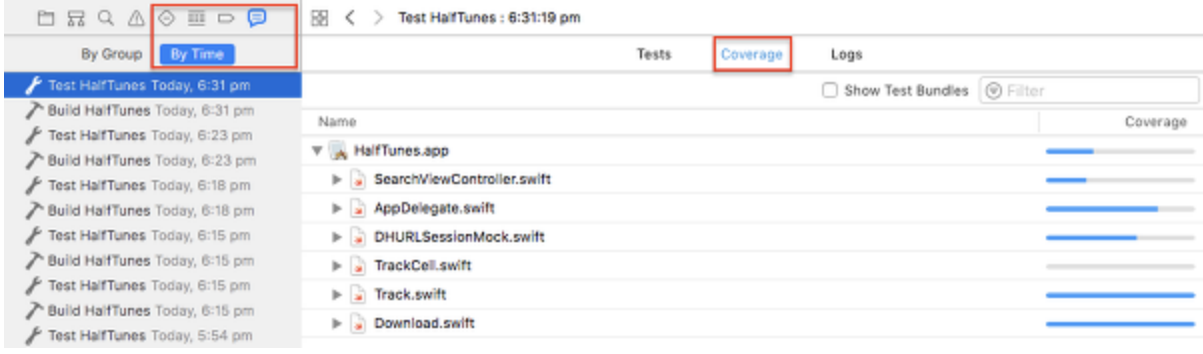
if-else **뿐만 아니라** enum, switch case**에서도 사용 가능**

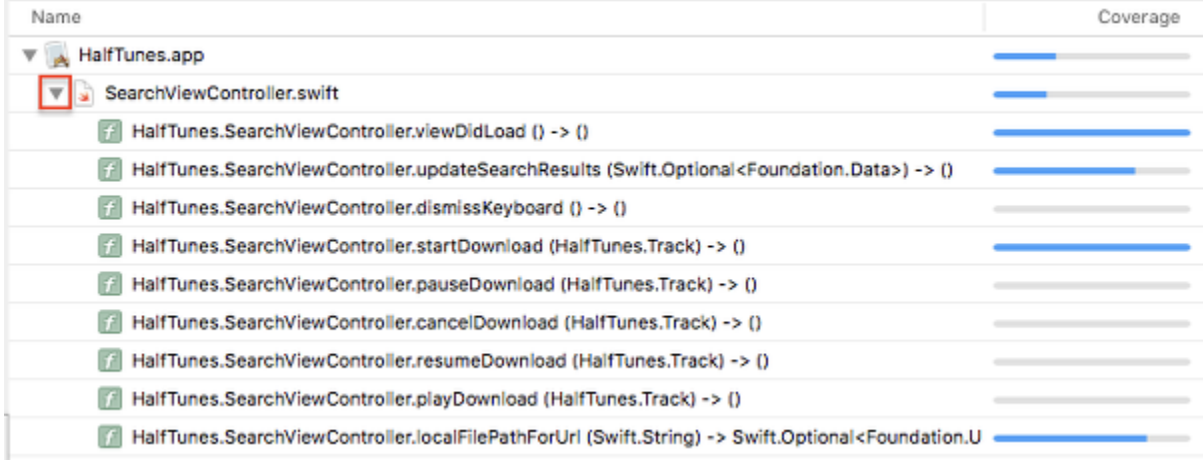
# Extracting ..

Methods, Inout **파라미터에서도 사용 가능함**

# XCode에서 Test Coverage



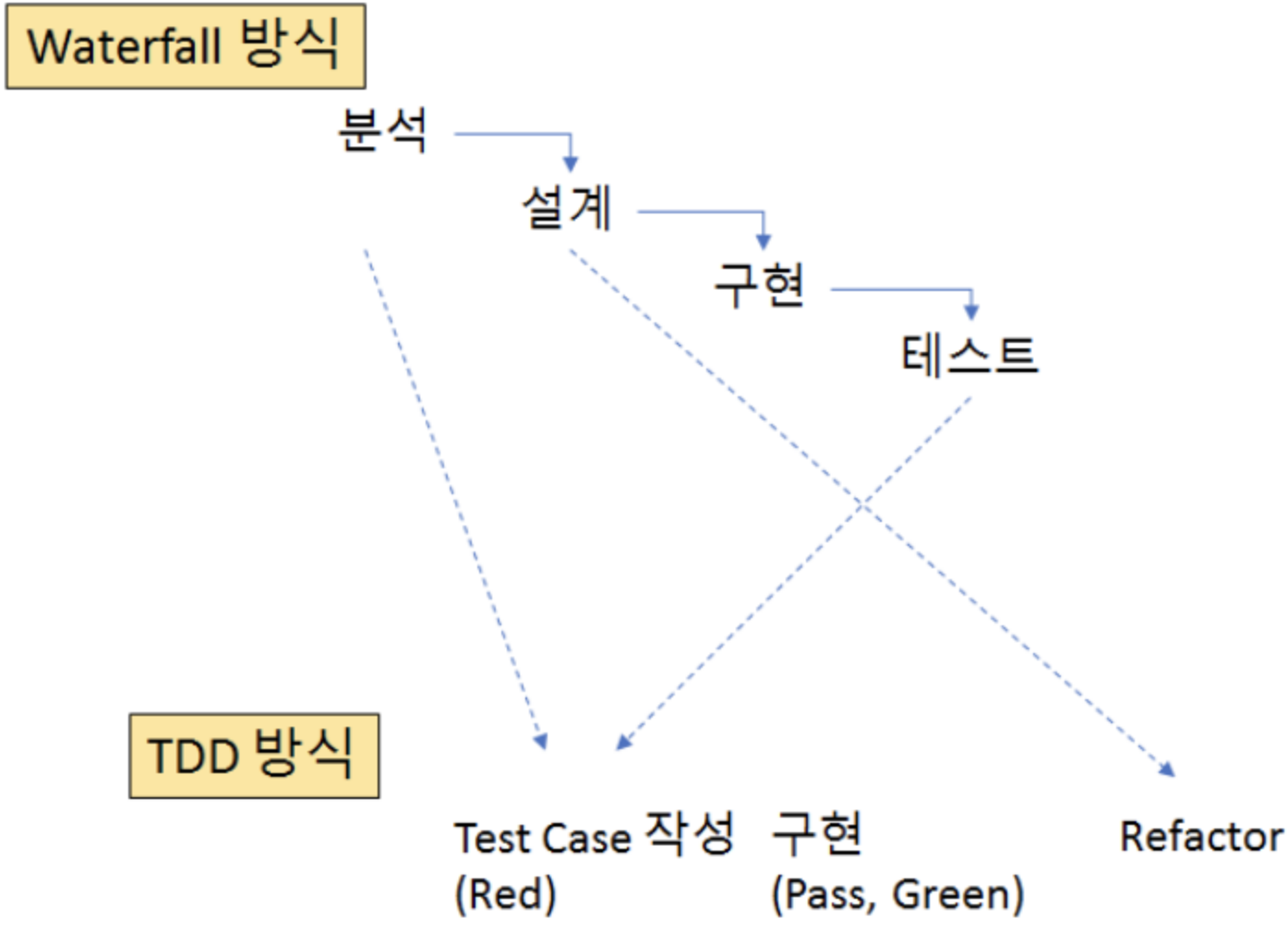



# 역할 나누는 것과 성능 중 우선 순위

# 참고

iOS Unit Testing and UI Testing Tutorial

# TDD

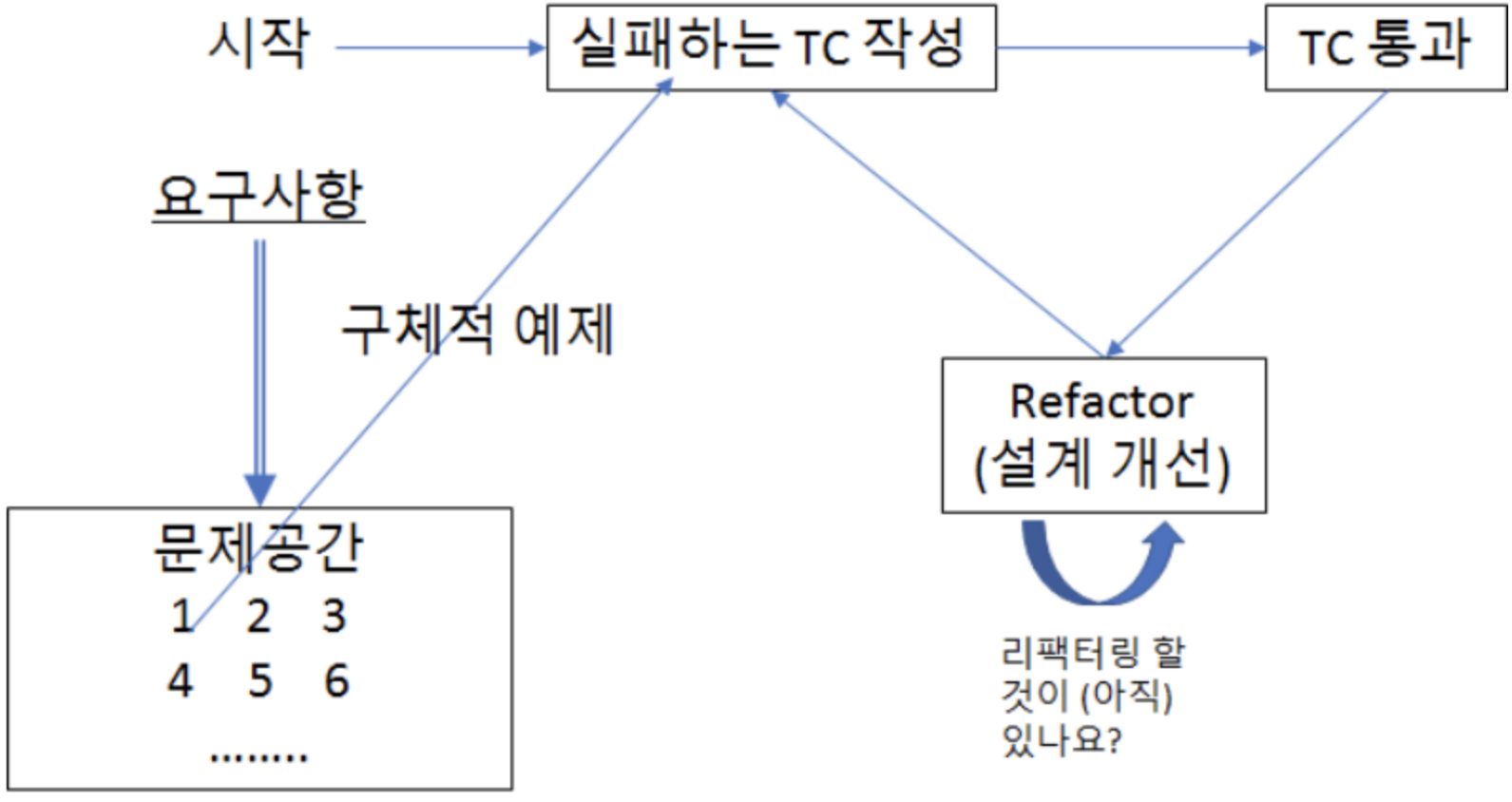

Waterfall 방식
분석
설계
구현
테스트
TDD 방식
Test Case 작성
(Red)
구현
(Pass, Green)
Refactor

# TDD에서 설계가 뒤로 가는 이유?

처음부터 요구사항을 완벽하게 이해하지 못함

설계는 지식의 양이 가장 많을 때 잘함

기존 프로세서는 지식의 양이 가장 적을 때 설계함


시작
실패하는 TC 작성
TC 통과
요구사항
구체적 예제
Refactor
(설계 개선)
문제공간
1 2 3
4 5 6
.....
리팩터링 할
것이 (아직)
있나요?

# 문제 공간에서 어떤 것을 테스트 케이스로 뽑을까?

무작위로 하면 됨. 하다보면 경험이 생겨서 어떻게 해야 할지 앎

일반적으로 잘 아는 것부터

간단하는 것부터

# 요구사항 분석

**추상적인 요구사항을 구체적인 테스트 케이스로 만드는 것**

# TDD 작성하기

## 첫 번째 테스트 케이스

- 첫 번째 테스트 케이스 만들기가 제일 어려움
- 하드 코딩으로 작성하기
- Refactor

# TDD가 익숙해지는 과정은 뇌 회로 생성되는 과정

# TDD Tip

- 익숙해지면 하드 코딩 과정을 거치지 않아도 됨
- 연습할 때 가급적 천천히
- 변화되지 않는 부분을 미리 예상하지 않는 것이 좋음

# 내가 느낀 TDD ? Divide and Conquer

# TDD를 혼자서 연습해볼 수 있는 곳

CyberDojo

Exercism

Coding Game

# 테스트 재정의하기

**시스템이 요구사항 대로 동작 여부에 대한 테스트가 테스팅인가?**

Specified

- 개발자

Unspecified

- QA, 메뉴얼 테스터

**체킹**

- Specified를 확인하는 것

**테스팅**

- Unspecified를 확인하는 것

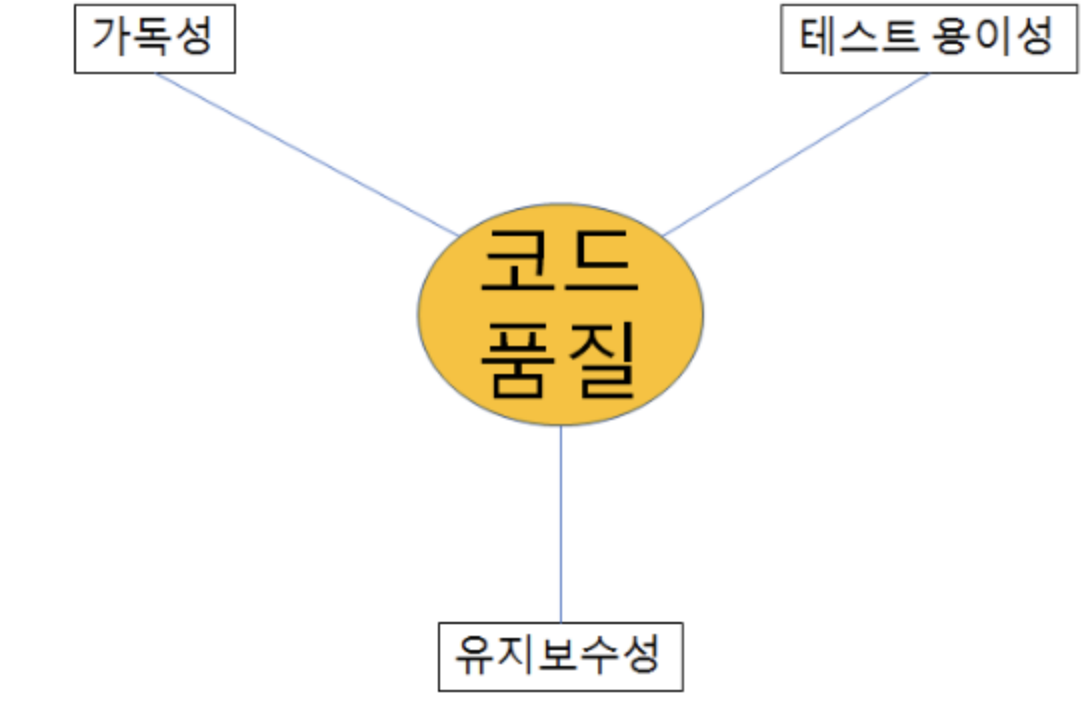
# 인수 테스트

Face to Face Commnucation

**구체적인 예로 커뮤니케이션 하고 요구사항을 이해해야 함**

# 코드 품질

이 읽어야 한다
쓰기 = 8 : 2



# 참고하면 좋은 것

## 인수테스트 주도 개발

## 레거시 코드 리팩토링은 무엇이며 왜 필요할까?