9장. 단위 테스트

😋단위테스트 코드를 클린하게 작성하는 법

단위테스트 예시

```
def add_two_numbers(num1, num2):

    return num1 + num2
```

```
assert add_two_numbers(2, 3) == 5
assert add_two_numbers(-3, 3) == 0
```

TDD 법칙 세 가지

첫째 법칙: 실패하는 단위 테스트를 작성할 때까지 실제 코드를 작성하지 않는다.

둘째 법칙: 컴파일은 실패하지 않으면서 실행이 실패하는 정도로만 단위 테스트를 작성한다.

셋째 법칙: 현재 실패하는 테스트를 통과할 정도로만 실제 코드를 작성한다.

- 이렇게 테스트 주도 개발을 할 경우, 실제 코드와 맞먹는 방대한 양의 코드가 생산이 된다.

깨끗한 테스트 코드 유지하기

테스트 코드를 깨끗하게 작성하지 않았던 일화

1. 테스트 코드는 퀄리티가 떨어져도 된다는 요청을 받음
2. 시간이 지날수록 실제 코드도 변하고 그에 맞춰서 테스트 코드도 변함
3. 테스트 코드가 유지보수성이 떨어지니 폐기 됨
4. 테스트 코드가 없으니 시스템 수정 시에 검증 확실히 안되고 결함이 많이 발생

- 반대로 테스트코드가 잘 작성되어 있을 경우, 기존 코드 변경에 대한 공포가 덜함

깨끗한 테스트 코드

가장 중요한건 가독성

이중 표준: 단순하고 간결하고 표현력이 풍부해야 하지만, 실제 코드만큼 효율적인 필요는 없다.

가독성이 떨어지는 테스트 코드 예시

```
@Test
 public void turnOnLoTempAlarmAtThreashold() throws Exception {
 hw.setTemp(WAY_TOO_COLD);
 controller.tic();
```

```
assertTrue(hw.heaterState());
assertTrue(hw.blowerState());
assertFalse(hw.coolerState());
assertFalse(hw.hiTempAlarm());
assertTrue(hw.loTempAlarm());
}
```

개선된 예시

```
@Test
public void turnOnLoTempAlarmAtThreshold() throws Exception {
wayTooCold();
assertEquals("HBchL", hw.getState());
}

@Test
public void turnOnCoolerAndBlowerIfTooHot() throws Exception {
tooHot();
assertEquals("hBChl", hw.getState());
}

@Test
public void turnOnHeaterAndBlowerIfTooCold() throws Exception {
tooCold();
assertEquals("HBchl", hw.getState());
}

@Test
public void turnOnHiTempAlarmAtThreshold() throws Exception {
wayTooHot();
assertEquals("hBCHl", hw.getState());
}

@Test
public void turnOnLoTempAlarmAtThreshold() throws Exception {
wayTooCold();
assertEquals("HBchL", hw.getState());
}
```