Junit and assertJ – 실용적인 테스트 가이드

1. Junit and assertJ

둘 다 테스트 용도로 사용하는 라이브러리. Junit은 JAVA 프로젝트 테스트에 필요한 기본 문법을 제공하고, assertJ는 Junit을 좀 더 편리하게 사용하기 위한 다양한 API를 제공한다.

2. 기본 문법

(1) assertThat

특정 객체의 값이 서로 일치하는지 비교할 수 있다.

```
assertThat(cafeKiosk.getBeverages().get(0)).isEqualTo(americano);
assertThat(cafeKiosk.getBeverages().get(1)).isEqualTo(americano);
```

(2) assertThatThrownBy

예외가 던져졌는지 검사할 수 있다.

```
assertThatThrownBy(()-> cafeKiosk.add(americano, 0))
        .isInstanceOf(IllegalArgumentException.class)
        .hasMessage("음료는 1잔 이상 주문할 수 있습니다.");
```

isInstanceOf 를 사용하면 어떤 종류의 예외가 던져졌는지 검사할 수 잇다.

hasMessage를 사용하면 예외에 어떤 문구가 포함되어 있는지 검사할 수 있다.

3. 테스트 기본

(1) 해피 케이스와 예외 케이스에 관해 각각 테스트를 작성한다.

(2) 경계값을 테스트하는 것이 중요하다.

(3) 테스트하기 어려운 영역도 존재한다. (날짜, 시간 등 가변적인 input, DB에 데이터 넣기 등 직접적으로 외부에 output 등)

4. 테스트 주도 개발(TDD)

- 프로덕션 코드보다 테스트 코드를 먼저 작성해서 테스트 코드가 프로덕션 코드를 이끄는 방식

(1) 실패하는 테스트 작성 -> (2) 테스트를 통과하기 위한 최소한의 코딩 -> (3) 리팩토링을 하면서 구현코드를 개선함.

5. 테스트 이름 정하기

- @DisplayName을 사용하여 테스트 이름을 지정할 수 있다. 테스트 이름을 섬세하게 정하자. 명사의 나열보다는 문장으로 표현하는 것이 좋다.

- 도메인 용어를 사용하여 좀 더 specific한 표현을 문장으로 쓰도록 한다. 메서드 자체 관점 보다는 도메인(비즈니스적) 용어를 사용하여 문장을 쓴다.

6. BDD 스타일로 작성하기

- TDD에서 파생된 개발 방법
- 함수 단위로 테스트하지 않고, 시나리오 기반한 테스트케이스에 집중하여 테스트한다.
- 개발자가 아닌 사람이 봐도 이해할 수 있을 정도의 추상화 수준을 권장함.

(1) Given: 시나리오 진행에 필요한 모든 준비 과정 (환경)

(2) When: 시나리오 행동 진행 (행동)

(3) Then: 시나리오 진행에 대한 결과 검증 (결과)