KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

JUnit5

[KB] IT's Your Life

자바 테스트 프레임워크

검증된 코드만 사용하는 것이 현재 트렌드

검증되지 않ㄴ은 코드를 함부로 써서 서비스에 영향을 주면 안된다.

이때 테스트 중에 단위 테스트라는 기법이 있다.

제일 기본이 되는 테스트. 단위는 메소드를 의미.

메소드 단위로 검증이 되어야 한다.

이 메소드 단위의 검증을 자동화 시키는
테스트 프레임워크.

자바에서는 JUNIT이라는 프레임워크 버전은 5를 이용할 예정.

우리가 만든 메소드들을 단위 테스트 해보자.

백엔드 갓을 때 스프링에서도 늘상 단위 테스트를 해야하는 기술이다.

기초다 기초

KB국민은행

JUnit 5

- **JUnit**

  자바진영에서 대표적인 단위 테스트 라이브러리
  . 반들시 해야하는 단위 테스트
  메소드 단위

  - 단위 테스트 라이브러리
  - 빌드 시스템을 gradle로 지정한 경우 JUnit5가 디폴트 단위 테스트 라이브러리로 설정됨
  - test 폴더 자동 생성됨
    - 실제 코드와 테스트 코드를 분리해서 관리

JUnit 5

- **JUnit**
  - @DisplayName("테스트이름")
    - 테스트 구분 제목이 됨
  - @Test
    - 테스트 메서드를 지정
- **Assertions**
  - 여러가지 단정문 static 메서드를 제공
  - assertEquals(실제값, 기대값)
    - 실제값과 기대값이 다르면 테스트 실패 → 예외 발생

매번 if문써서 예외 던지기 번거루움
->단조로움을 해결할 라이브러리 assertions

검사 결과가 true면 통과
false면 예외 발생

테스트할때는 기본적으로 scanner안됨.
즉 사용자 입력안됨. 키보드 입려도 안되지

테스트 원칙. 사람 개입X.

원칙에 맞으려면 테스트는 자동화 되어야함.
지금은 배우는시점이라 아니지만

JUnit으로 테스트 클래스 만들기

- src → test → java/JUnitTest.java

```
public class JUnitTest {

    @DisplayName("1+2는 3이다")
    @Test
    public void junitTest() {
        int a = 1;
        int b = 2;
        int sum = 3;

        Assertions.assertEquals(a+b, sum);
    }
}
```

실제값

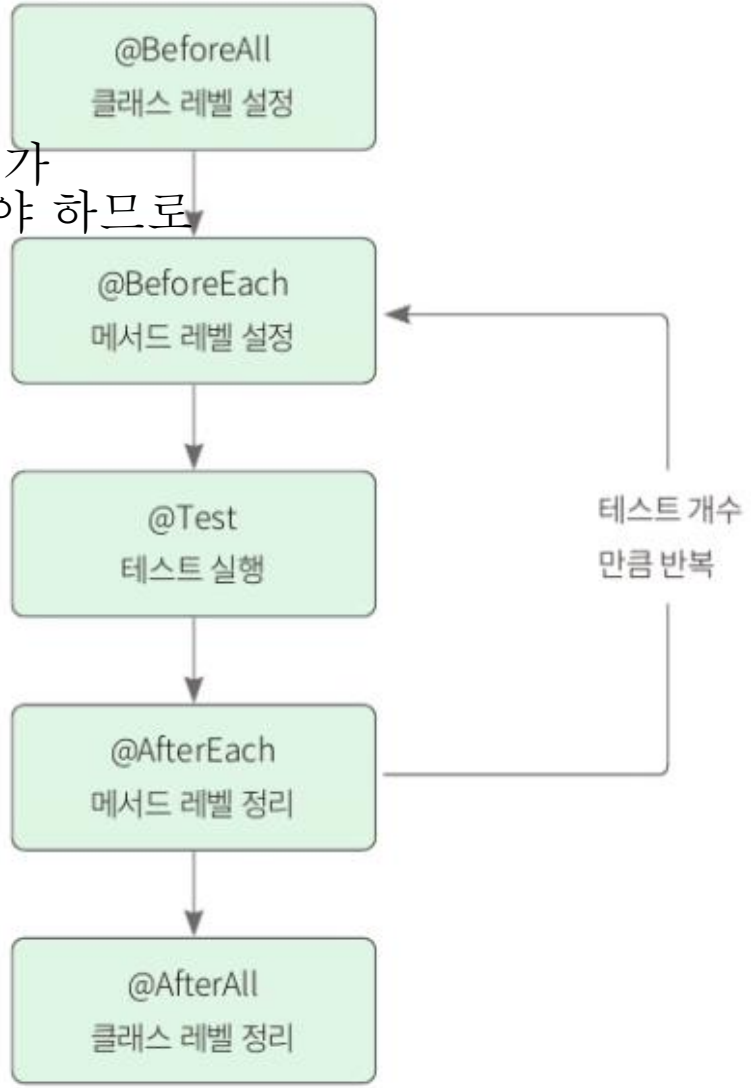
기대값

어떤 테스트는 초기화 과정을 거쳐야한다. 예를 들어 CRUD.
CRUD를 하려면 연결이되어야 할거 아니야 Connect객체 획득해야될거 아니야
또 테스트 끝나면 메모리해제 close같은 클린업도 해야되잖아

JUnit 주요 어노테이션

- @BeforeAll
  - 전체 테스트 실행 전 1회 호출
  - static 메서드로 정의
- @BeforeEach
  - 각 테스트 케이스마다 실행 전 호출
- @AfterEach
  - 각 테스트 케이스마다 실행 후 호출
- @AfterAll
  - 전체 테스트 실행 후 1회 호출
  - static 메서드로 정의

테스트 인스턴스가
만들어지전이어야 하므로



단위테스트 원칙
하나의 테스트가
다른 테스트에
영향을 미쳐선 안되요

가각 독립적이어야함.

테스트할때마다
인스턴스를
새로 만들어서
테스트함.

JUnitCycleTest .java

```
public class JUnitCycleTest {
    @BeforeAll  // 전체 테스트 시작전 1회 실행, static 선언
    static void beforeAll() {
        System.out.println("@BeforeAll");
    }

    @BeforeEach // 텟트 케이스를 시작하기 전마다 실행
    public void beforeEach() {
        System.out.println("@BeforeEach");
    }

    @Test
    public void test1() {
        System.out.println("test1");
    }

    @Test
    public void test2() {
        System.out.println("test2");
    }

    @Test
    public void test3() {
        System.out.println("test3");
    }
```

뭐가 먼저 테스트 실행할지는 몰라. 랜덤임

하지만 순서가 필요할때가 있음
이럴때는 수동적으로 순서를 지정해야함
@TestMethodOrder 어노테이션을 사용

어노테이션 인자값으로 기준을 정할 수 있고
메서드명, 랜덤, 직접지정 순서가 있다.

JUnitCycleTest .java

```
    @AfterEach  // 테스트 케이스를 종료하기 전마다 실행
    public void afterEach() {
        System.out.println("@AfterEach");
    }

    @AfterAll   // 전체 테스트를 마지고 종료하기 전 1회. static 선언
    static void afterAll() {
        System.out.println("@AfterAll");
    }

}
```

```
@BeforeAll
@BeforeEach
test1
@AfterEach
@BeforeEach
test2
@AfterEach
@BeforeEach
test3
@AfterEach
@AfterAll
```

AssertJ

앞서 본 단정문

메서드 이름	설명
isEqualTo(A)	A값과 같은지 검증
isNotEqualTo(A)	A값과 다른지 검증
contains(A)	A값을 포함하는지 검증
doesNotcontains(A)	A값을 포함하는지 않는지 검증
startsWith(A)	접두사가 A인지 검증
endsWith(A)	접미사가 A인지 검증
isEmpty()	비어있는 값인지 검증
isNotEmpty()	비어있는 않은 값인지 검증
isPositive()	양수인지 검증
isNegative()	음수인지 검증
isGreaterThan(n)	n보다 큰 값인지 검증
isGreaterThan(n)	n보다 작은 값인지 검증

테스트 클래스 만들기

- 테스트 대상 클래스 >> Generate... > Test...

아까 JDBC프로그래밍에서 봤던 화면