

🏆 미션 완료

👤 나의 팀

📄 슬라이드

main method vs JUnit

main method

main method의 용도

- 프로그램을 시작
- 구현한 프로그램을 테스트

이 과정은 main method를 테스트 용도로 사용하는 경우에 대해 다루고 있다.

맛보기 프로그램 구현

- 사칙연산이 가능한 계산기
 - 덧셈(add)
 - 뺄셈(subtract)
 - 곱셈(multiply)
 - 나눗셈(divide)
- 위 4개의 기능을 구현하고 main method를 활용해 테스트한다.

Production Code

```
public class Calculator {  
    int add(int i, int j) {  
        return i + j;  
    }  
  
    int subtract(int i, int j) {  
        return i - j;  
    }  
  
    int multiply(int i, int j) {  
        return i * j;  
    }  
  
    int divide(int i, int j) {  
        return i / j;  
    }  
}
```

Test Code

```
public static void main(String[] args) {  
    Calculator cal = new Calculator();  
    System.out.println(cal.add(3, 4));  
    System.out.println(cal.subtract(5, 4));  
    System.out.println(cal.multiply(2, 6));  
    System.out.println(cal.divide(8, 4));  
}
```

- 위 그림을 통해 확인할 수 있듯이 **프로덕션 코드(Production Code)** 프로그램 구현을 담당하는 부분으로 사용자가 실제로 사용하는 소스 코드를 의미한다.
- **테스트 코드(test code)** 는 프로덕션 코드가 정상적으로 동작하는지를 확인하는 코드이다.

main method 테스트 문제점

- Production code와 Test Code가 클래스 하나에 존재한다. 클래스 크기가 커짐. 복잡도 증가함.
- Test Code가 실 서비스에 같이 배포됨.
- **main method** 하나에서 여러 개의 기능을 테스트 함. 복잡도 증가.
- method 이름을 통해 어떤 부분을 테스트하는지에 대한 의도를 드러내기 힘들.
- 테스트 결과를 사람이 수동으로 확인

JUnit

- main method를 활용해 테스트할 때 발생하는 문제점을 해결하기 위해 등장한 도구가 [JUnit](#)이다.



- @Test
- @BeforeEach, @AfterEach
- @BeforeAll, @AfterAll
- Assert 클래스의 static assert method를 활용해 테스트 결과 검증



미션 완료



나의 팀



슬라이드

```
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

public class LifecycleTest {
    @BeforeAll // JUnit 4의 @BeforeClass
    static void initAll() {
        System.out.println("initAll");
    }

    @BeforeEach // JUnit 4의 @Before
    void init() {
        System.out.println("init");
    }

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

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

    @AfterEach // JUnit 4의 @After
    void tearDown() {
        System.out.println("tearDown");
    }

    @AfterAll // JUnit 4의 @AfterClass
    static void tearDownAll() {
        System.out.println("tearDownAll");
    }
}
```

JUnit 4.x 버전

- 애노테이션(Annotation)을 활용해 테스트 코드 구현
 - @Test
 - @Before, @After
 - @BeforeClass, @AfterClass
- Assert 클래스의 static assert method를 활용해 테스트 결과 검증



```
import org.junit.Before;
import org.junit.Before;
import org.junit.Test;

public class CalculatorTest {
    Calculator cal;

    @Before
    public void setUp() {
        cal = new Calculator();
    }

    @Test
    public void 덧셈() {
        assertEquals(7, cal.add(3, 4));
    }

    @Test
    public void 뺄셈() {
        assertEquals(1, cal.subtract(5, 4));
    }

    @Test
    public void 곱셈() {
        assertEquals(6, cal.multiply(2, 3));
    }

    @Test
    public void 나눗셈() {
        assertEquals(2, cal.divide(8, 4));
    }

    @After
    public void tearDown() {
        cal = null;
    }
}
```

method 실행 순서

- 다음과 같은 JUnit 테스트를 실행할 경우 실행 순서는 어떻게 될까?

```
public class SlippTest {
    @Before
    public void setup() {
        System.out.println("setup");
    }

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

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

    @After
    public void teardown() {
        System.out.println("teardown");
    }
}
```

assertion

- [assertj](#) : 메소드 체이닝을 통해 좀 더 깔끔하고 읽기 쉬운 assert 코드 구현 가능



```
// assertEquals
assertThat(frodo.getName()).isEqualTo("Frodo");
assertThat(frodo).isNotEqualTo(sauron);

// chaining string specific assertions
assertThat(frodo.getName()).startsWith("Fro")
    .endsWith("do")
    .isEqualToIgnoringCase("frodo");

// collection specific assertions (there are plenty more)
// in the examples below fellowshipOfTheRing is a List<TolkienCharacter>
assertThat(fellowshipOfTheRing).hasSize(9)
    .contains(frodo, sam)
    .doesNotContain(sauron);
```



미션 완료



나의 팀



슬라이드