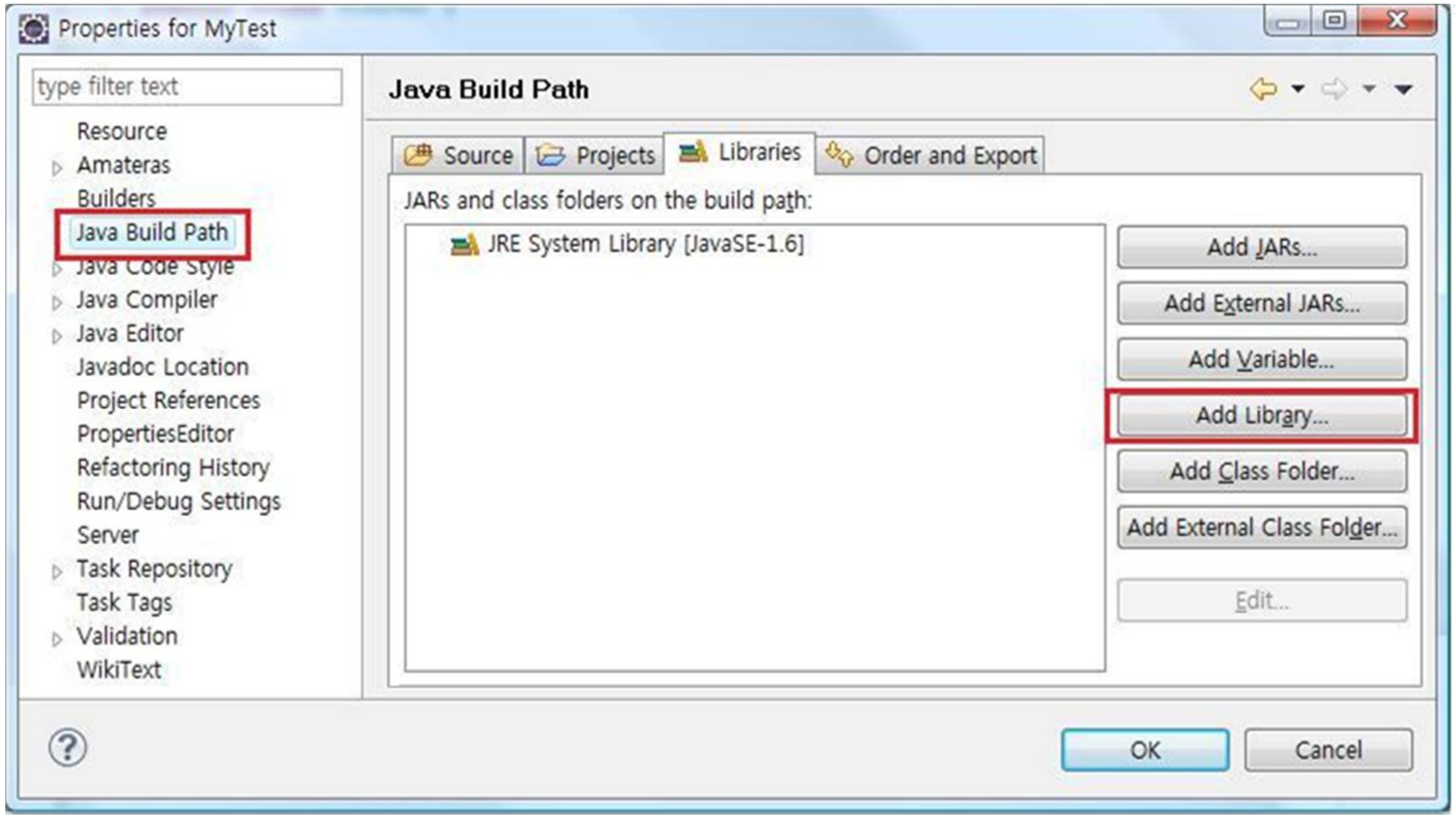
JUnit

- JUnit 프로젝트에 추가
- JUnit Test Case 작성
- JUnit annotations
- assert methods
  - assertEquals()
  - assertNull() + assertNotNull()
  - assertSame() and assertNotSame()
  - assertTrue() + assertFalse()
  - assertArrayEquals()
  - assertThat()
  - Exceptions Test

프로젝트에 JUnit 추가

- Java Build Path에서 JUnit을 추가하는 방법



JARs and class folders on the build path:

JRE System Library [JavaSE-1.6]

- build.gradle에 추가하는 방법

사칙 연산 클래스 만들기

두 개의 정수를 입력 받고, 두 수의덧셈, 뺄셈, 곱셈, 나눗셈 결과를 출력하는 프로그램을 작성하시오.

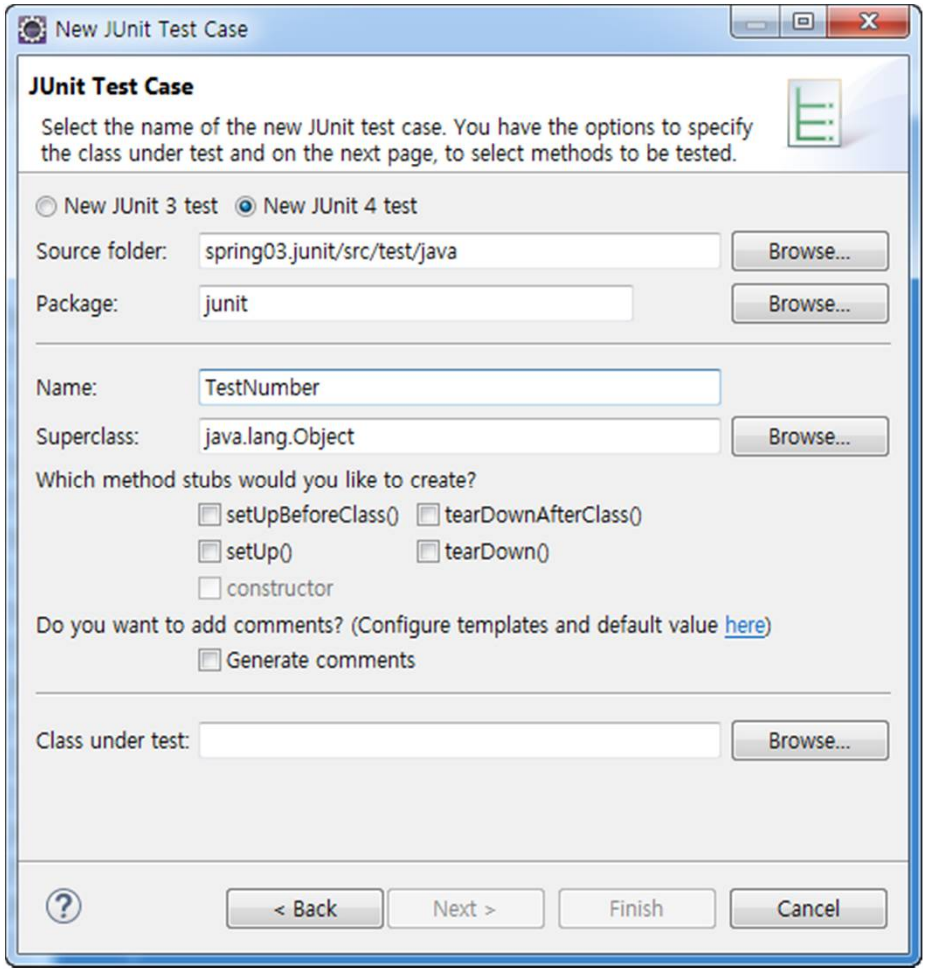
- Oper 클래스를 만들고 클래스 안에 Add(), Minus(), Mul(), Div() 메서드만을 만드시오.
- 테스트 클래스에서 각각의 연산 결과를 출력하시오.
- 나눗셈의 결과는 실수가 되도록 한다.
- 키보드 입력은 OperTest 클래스 에서 입력 받도록 하시오.

패키지명: java22.junit,   클래스명: Oper , OperTest

- 실행결과예시

```
First num : 2
Second num : 4
Add : 6          ---> Add() 메서드를 사용하시오
Minus : -2       ---> Minus() 메서드를 이용하시오
Mul : 8          ---> Mul() 메서드를 이용하시오
Div : 0.500000   ---> Div() 메서드를 이용하시오
```

JUnit Test Case 추가



JUnit Test Case

Select the name of the new JUnit test case. You have the options to specify the class under test and on the next page, to select methods to be tested.

New JUnit 3 test [✓ New JUnit 4 test]

Source folder: spring03.junit/src/test/java

Package: junit

Name: TestNumber

Superclass: java.lang.Object

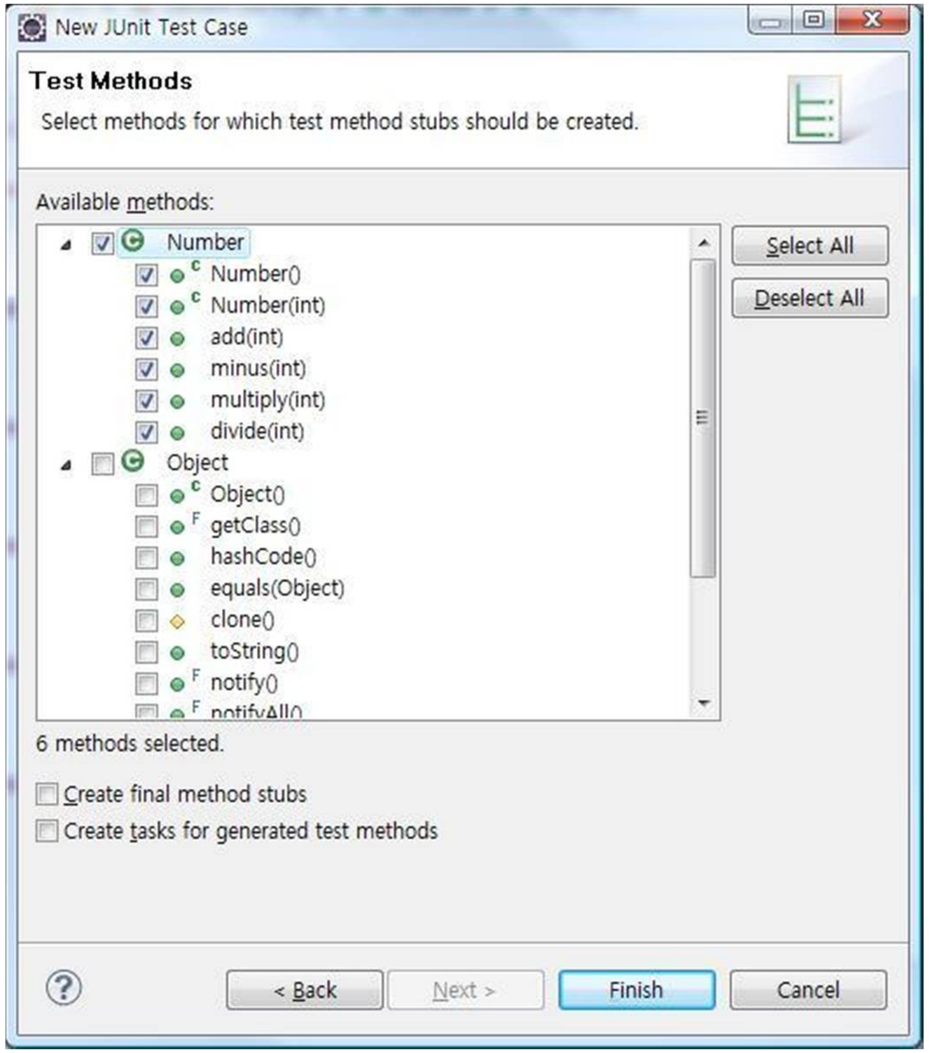
Which method stubs would you like to create?

- [ ] setUpBeforeClass()
- [ ] tearDownAfterClass()
- [ ] setUp()
- [ ] tearDown()
- [ ] constructor

Do you want to add comments? (Configure templates and default value here)

- [ ] Generate comments

Class under test:



Test Methods

Select methods for which test method stubs should be created.

Available methods:

- [✓] Number
  - [✓] Number()
  - [✓] Number(int)
  - [✓] add(int)
  - [✓] minus(int)
  - [✓] multiply(int)
  - [✓] divide(int)
- [ ] Object
  - [ ] Object()
  - [ ] getClass()
  - [ ] hashCode()
  - [ ] equals(Object)
  - [ ] clone()
  - [ ] toString()
  - [ ] notify()
  - [ ] notifyAll()

6 methods selected.

- [ ] Create final method stubs
- [ ] Create tasks for generated test methods

assertEquals()

```
import org.junit.Test;
import static org.junit.Assert.*;

public class MyUnitTest {

    @Test
    public void testConcatenate() {
        MyUnit myUnit = new MyUnit();

        String result = myUnit.concatenate("one", "two");

        assertEquals("onetwo", result);
    }
}
```

assertNull() + assertNotNull()

```
import org.junit.Test;
import static org.junit.Assert.*;

public class MyUnitTest {

    @Test
    public void testGetTheObject() {
        MyUnit myUnit = new MyUnit();

        assertNull(myUnit.getTheObject());

        assertNotNull(myUnit.getTheObject());
    }
}
```

assertSame() and assertNotSame()

```
import org.junit.Test;
import static org.junit.Assert.*;

public class MyUnitTest {

    @Test
    public void testGetTheSameObject() {
        MyUnit myUnit = new MyUnit();

        assertSame   (myUnit.getTheSameObject(),  myUnit.getTheSameObject());

        assertNotSame(myUnit.getTheSameObject(),  myUnit.getTheSameObject());
    }
}
```

assertTrue() + assertFalse()

```
import static org.junit.Assert.*;

public class MyUnitTest {

    @Test
    public void testGetTheBoolean() {
        MyUnit myUnit = new MyUnit();

        assertTrue (myUnit.getTheBoolean());

        assertFalse(myUnit.getTheBoolean());
    }
}
```

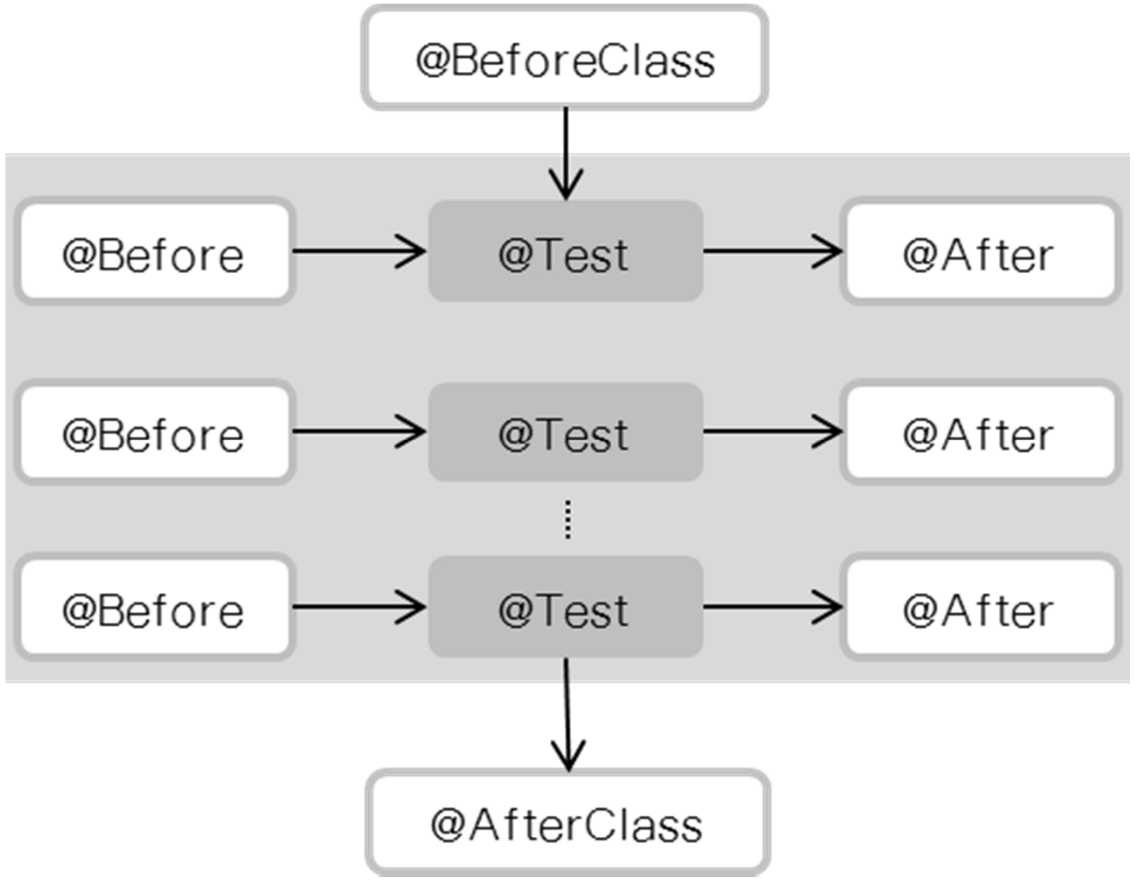
assertArrayEquals

```
import org.junit.Test;
import static org.junit.Assert.*;

public class MyUnitTest {

    @Test
    public void testGetTheStringArray() {
        MyUnit myUnit = new MyUnit();

        String[] expectedArray = {"one", "two", "three"};

        String[] resultArray =  myUnit.getTheStringArray();

        assertArrayEquals(expectedArray, resultArray);
    }
}
```

exceptions Test

```
package com.mkyong;

import org.junit.Test;
import java.util.ArrayList;

public class Exception1Test {

    @Test(expected = ArithmeticException.class)
    public void testDivisionWithException() {
        int i = 1 / 0;
    }

    @Test(expected = IndexOutOfBoundsException.class)
    public void testEmptyList() {
        new ArrayList<>().get(0);
    }

}
```

JUnit annotations



Introduction

No.	Annotation	Description
1	@Test	The Test annotation tells JUnit that the public void method to whic h it is attached can be run as a test case.
2	@Before	Several tests need similar objects created before they can run. Ann otating a public void method with @Before causes that method to be run before each Test method.
3	@After	If you allocate external resources in a Before method, you need to r elease them after the test runs. Annotating a public void method w ith @After causes that method to be run after the Test method.
4	@BeforeClas s	Annotating a public static void method with @BeforeClass causes it to be run once before any of the test methods in the class.
5	@AfterClass	This will perform the method after all tests have finished. This can b e used to perform clean-up activities.
6	@Ignore	The Ignore annotation is used to ignore the test and that test will n ot be executed.
7	@Rule	