

소프트웨어 테스트

소프트웨어 테스트 중요성

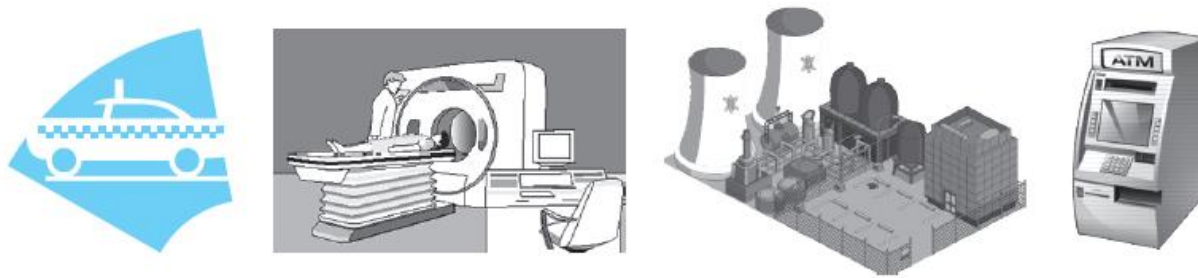


그림 4-1 일상생활을 편리하게 해주는 것들

일상생활을 편리하게 하나
문제가 발생하면 위험한
것들

소프트웨어 테스트 정의

- 프로그램의 오류를 발견하기 위해 프로그램을 실행시키는 과정
 - 품질 평가를 위해 프로그램을 실행하는 과정
 - 프로그램의 신뢰도를 높이기 위해 프로그램을 실행하는 과정
- 동적 테스트
- 오류를 발견하거나 오류의 발생을 미연에 방지할 목적으로 프로그램이나 문서들을 분석하는 과정
- 정적 분석(또는 테스트)

소프트웨어 테스트 용어 정의

- Mistake
- Fault
- Error
- Failure

Standard glossary of terms used in Software Testing

Version 2.1 (dd. April 1st, 2010)

**Produced by the ‘Glossary Working Party’
International Software Testing Qualifications Board**

Mistake/Human Error

- 잘못된 결과를 가져오는 인간의 행위를 의미하며 주어진 정보를 잘못 이해할 때 발생한다. 즉, 문제 영역, 해결 방안, 프로그래밍 언어의 문법이나 의미, 컴파일러, 운영체제 등의 환경에 대한 잘못된 이해로부터 비롯된다. 또한 철자 오류도 에러에 속한다



Fault(결함)

- 에러를 발생하게 하는 프로그램 부분
- 프로그램을 의도되지 않거나 예상치 못한 방식으로 수행하게 하는 잘못된 단계나 프로세스 및 데이터 정의를 의미한다.
 - ✓ 잘못된 정보를 반영하는 경우(commission)
 - ✓ 올바른 정보를 빠뜨리는 경우(omission)이다.
 - ✓ 알고리즘 결함. 알고리즘 결함은 잘못된 프로그램 경로를 수행하도록 하는 제어 흐름과 관련된 결함이나 대상이 되는 프로그램의 외부와 통신과 관련된 인터페이스 결함 및 잘못된 자료 구조의 사용으로 인한 결함 .

Error(에러)

- 프로그램의 올바르지 않은 내부 상태(예를 들면 선행 조건, 후행 조건, 프로그램의 무결성이 위배된 프로그램 상태)
- 계산되거나 측정된 값과 기대 값 사이의 차이

Failure(오작동)

- 프로그램이 명세와는 다르게 동작하는 것이 외부에서 관찰되는 상황이다.
- 오작동은 결함에 의해 발생하지만 결함이 있다고 해서 반드시 오작동이 발생하지는 않는다.
- 프로그램의 실행 결과와 기대 결과와의 (관찰가능한) 차이를 말한다.

예

예제 2 다음 코드는 한 개의 정수를 입력으로 받아 입력 값의 두 배를 계산하여 출력하는 프로그램이다. 2를 입력하면 오작동이 발생되지 않는다. 그 이유를 살펴보자.

Listing

```
int double (int param) {  
    1: int retVal=0;  
    2: retVal = param*param; // 프로그램 결함  
    3: return retVal  
}
```

소프트웨어 테스트 한계

- 완전한 테스트 Exhaustive Test
 - ✓ 완전하게 테스트한다는 의미는 모든 입력 조건 조합에 대하여 테스트한다는 의미이다

계산기 프로그램을 완전하게 테스트하라!!

$0+1, 0+2, \dots, 0+(\text{정수최대값}),$
 $1+0, 1+1, 1+0, 1+1, \dots,$
 $1+(\text{정수최대값}), \dots, 0+0+0,$
 $0+0+1, \dots$ 등.



완전한 테스트(exhaustive test)

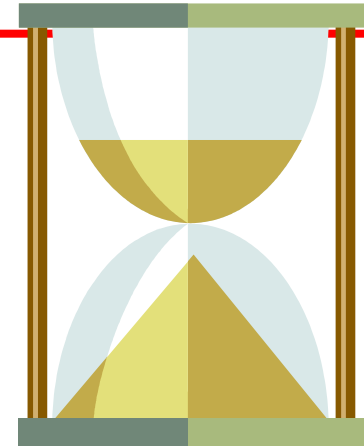
- 다음 프로그램을 완전하게 테스트 하려면 얼마나 걸릴까?
 - ✓ 10^{-9} sec/test
 - ✓ 10^{10} sec=600년

```
int foo(int a, int b) {  
    return a*b;  
}
```

$2^{32} \times 2^{32} = 2^{64} = 10^{19}$ 테스트

테스트 한계

- 테스트는 오류가 없음을 보여주지 않는다.
- 어떤 테스트를 선정할 것인가? 테스트 선정 기준
- 언제 테스트를 종료 할 것인가? 테스트 종료 조건

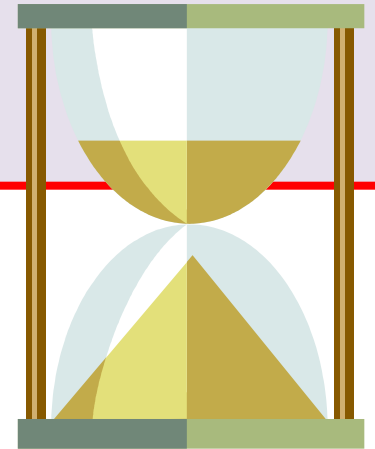


공리-테스트 한계

- 테스트는 오류가 없음을 보여주지 않는다.
- 언제 테스트를 종료 할 것인가?
- 테스트 종료 조건

종료조건

- 준비된 테스트 케이스들 중에서 95%가 통과되면 테스트를 종료한다.
- 준비된 테스트 케이스들 중에서 TC1, TC35, TC57은 반드시 통과하여야 한다.
- 새롭게 발생한 오류의 개수가 3개 이상 넘지 않으면 테스트를 종료한다.
- 주어진 기간 동안 치명적인 오류가 발견되지 않으면 테스트를 종료한다.



공리-결함 집중 원칙 등

- 프로그램의 어떤 부분에 오류가 남아 있을 확률은 이미 발견된 오류의 수에 직접적으로 비례한다.
- Bugs follow bugs
- 테스트 노력을 오류를 많이 발생시킨 부분에 집중

JUnit 4

- 단위 테스트 프레임워크
- 어노테이션을 제공하여 매우 쉽고 간결하게 테스트 코드를 작성 및 실행
- 더 이상 테스트를 하기 위해 코드 중간 중간에 출력문을 삽입할 필요가 없다

실습

pom.xml

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Archetype을 simpleproject로

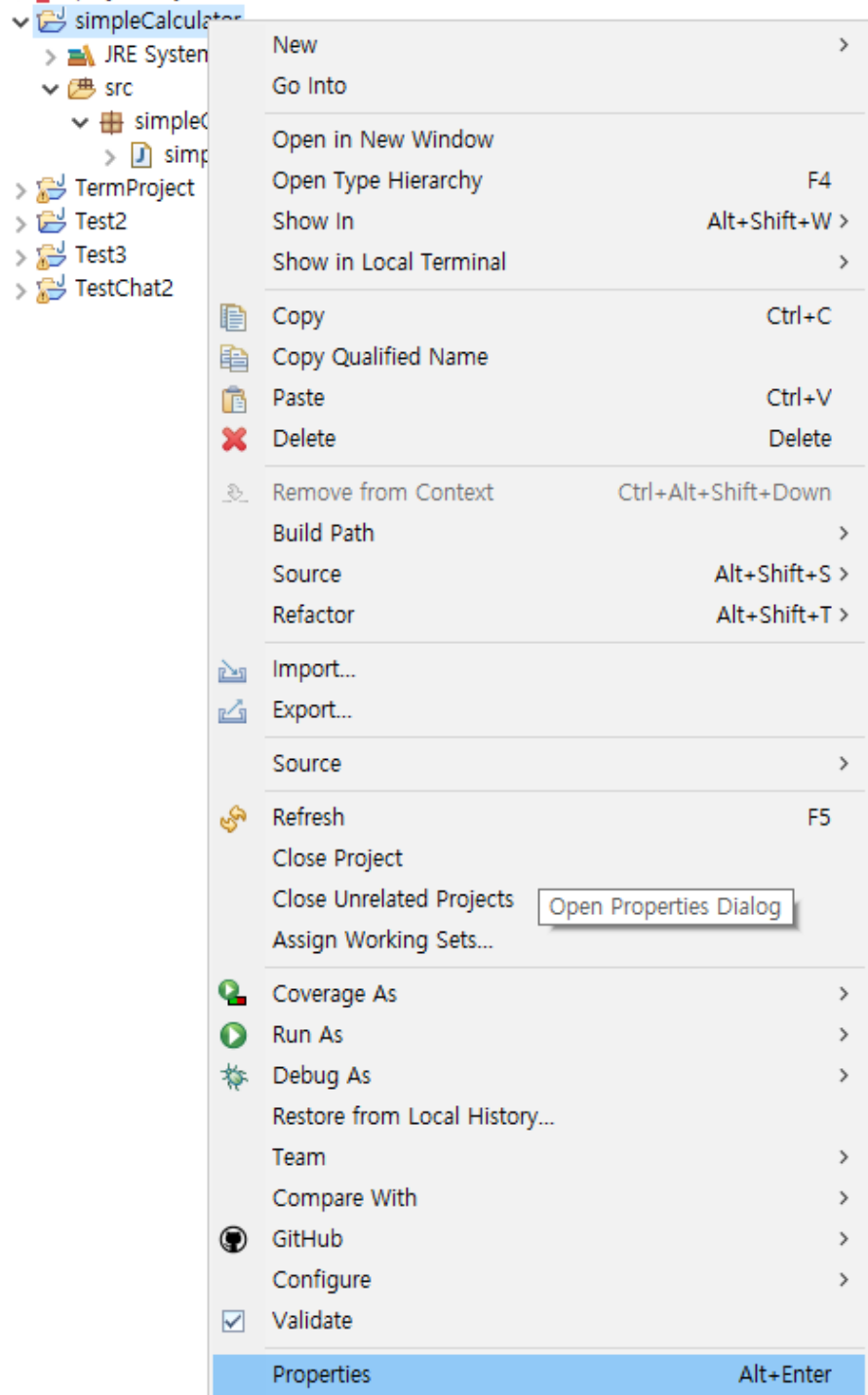
예제 코드

```
package simpleCalculator;

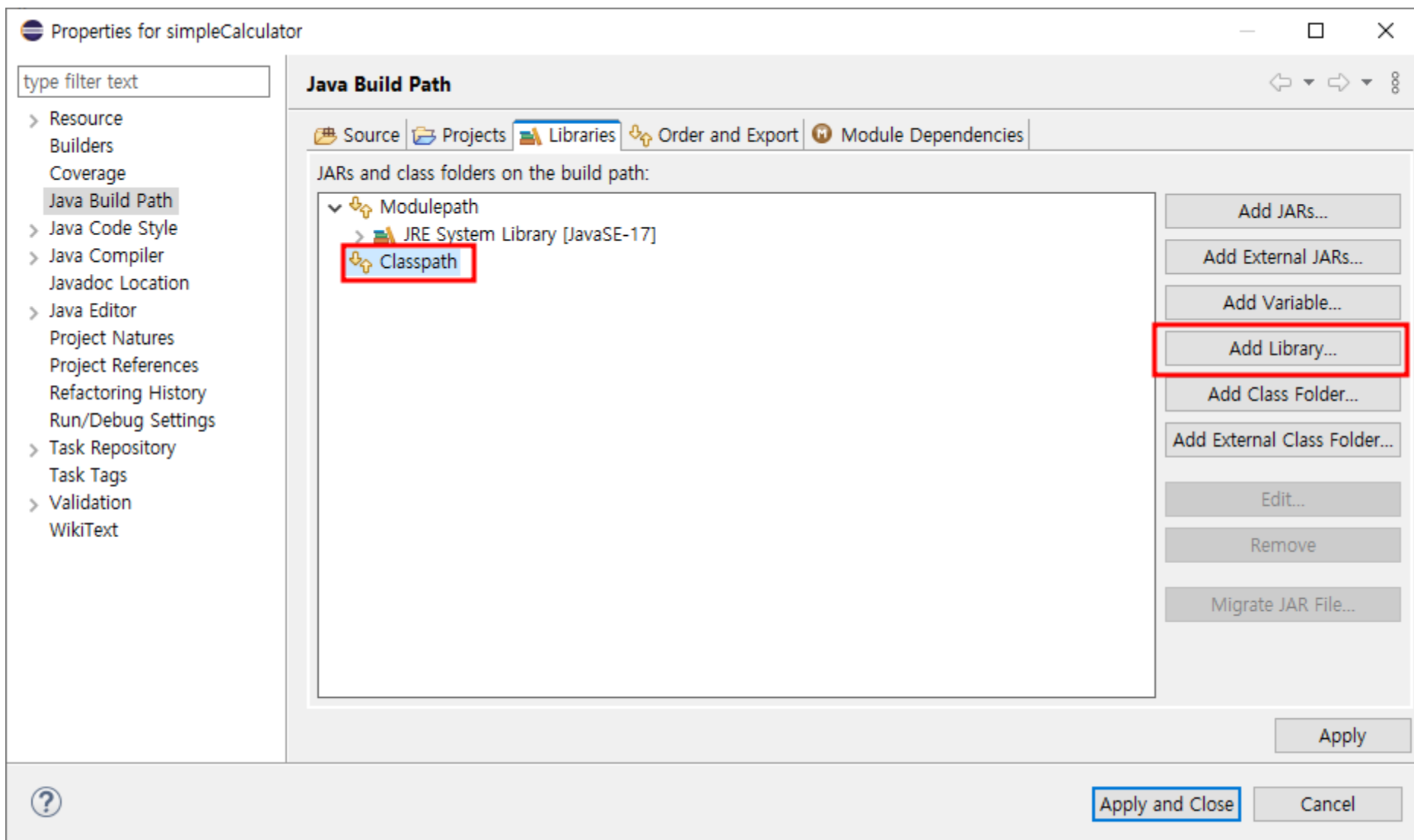
public class SimpleCalculator {
    private int res=0;

    public void add(int x, int y) {
        res=x+y;
    }
    public void sub(int x, int y) {
        res=x-y;
    }
    public void inc(int d) {
        res+=d;
    }
    public void dec(int d) {
        res-=d;
    }
    public int getResult() {
        return res;
    }
}
```

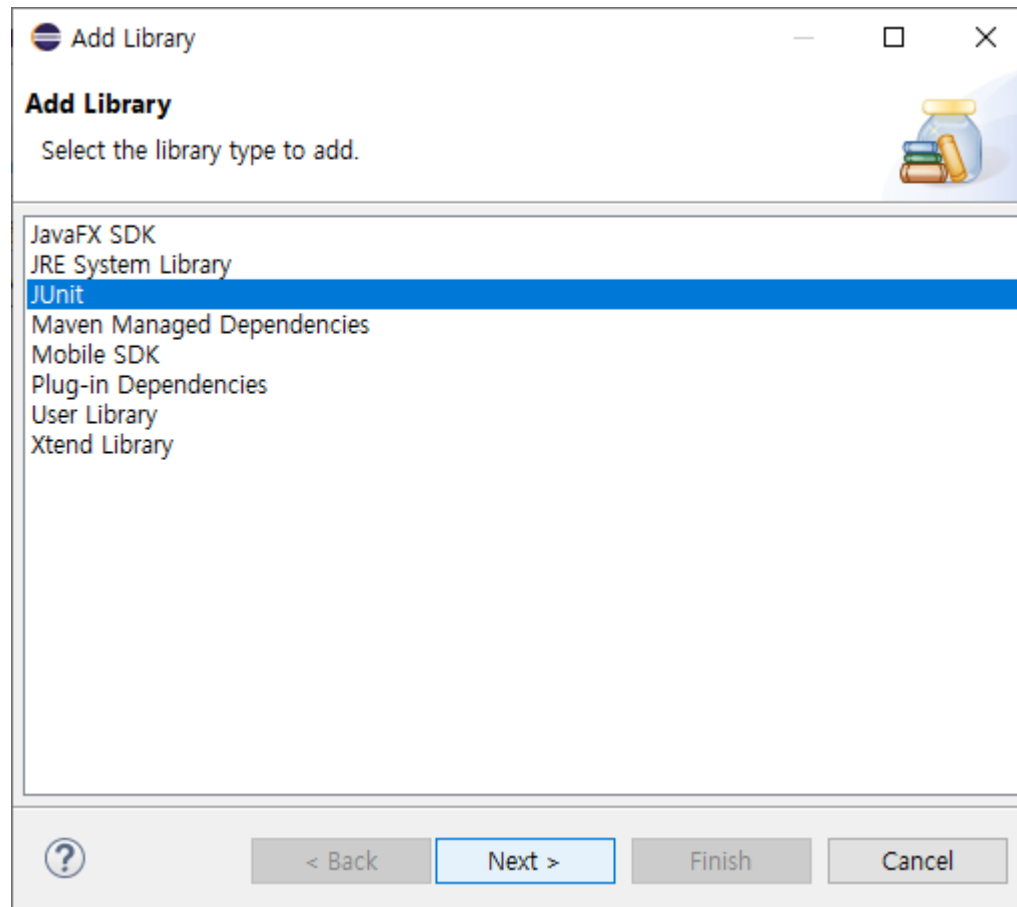
프로젝트를 선택하고 우클릭
후 Properties를 선택



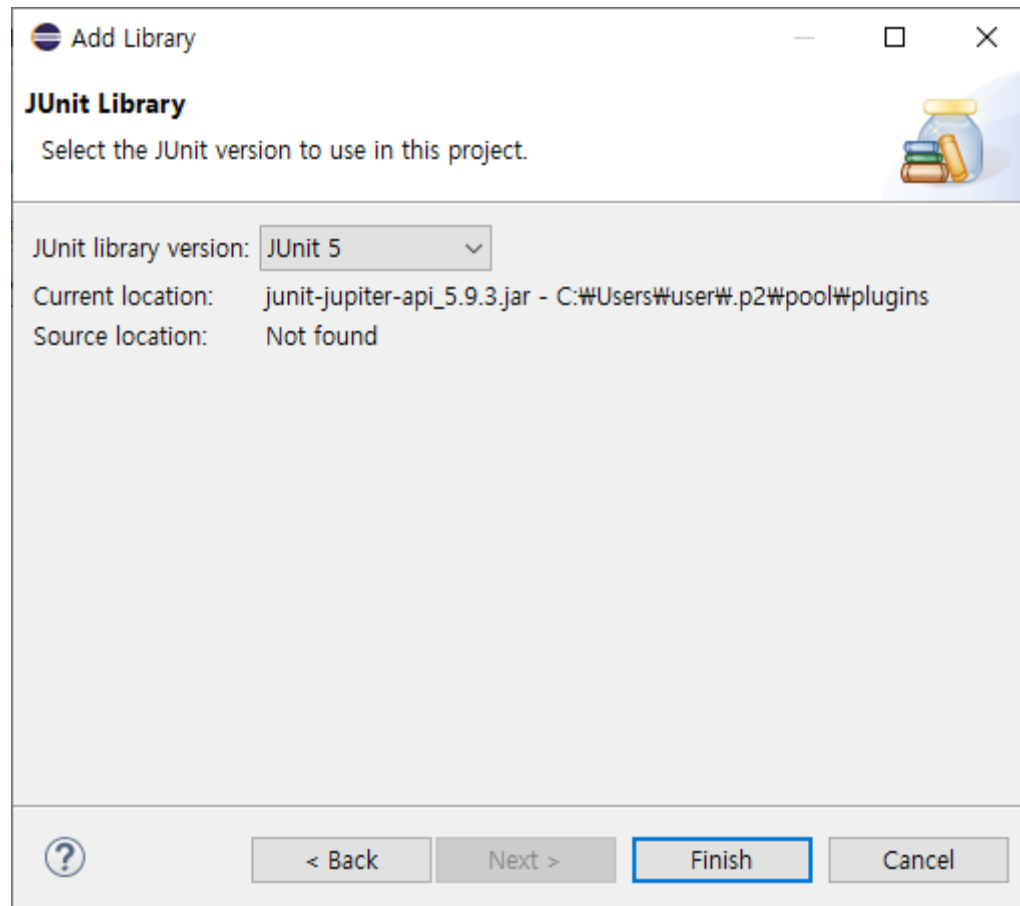
Java Build Path를 선택하고 Libraries 탭에서 Add Library를 선택



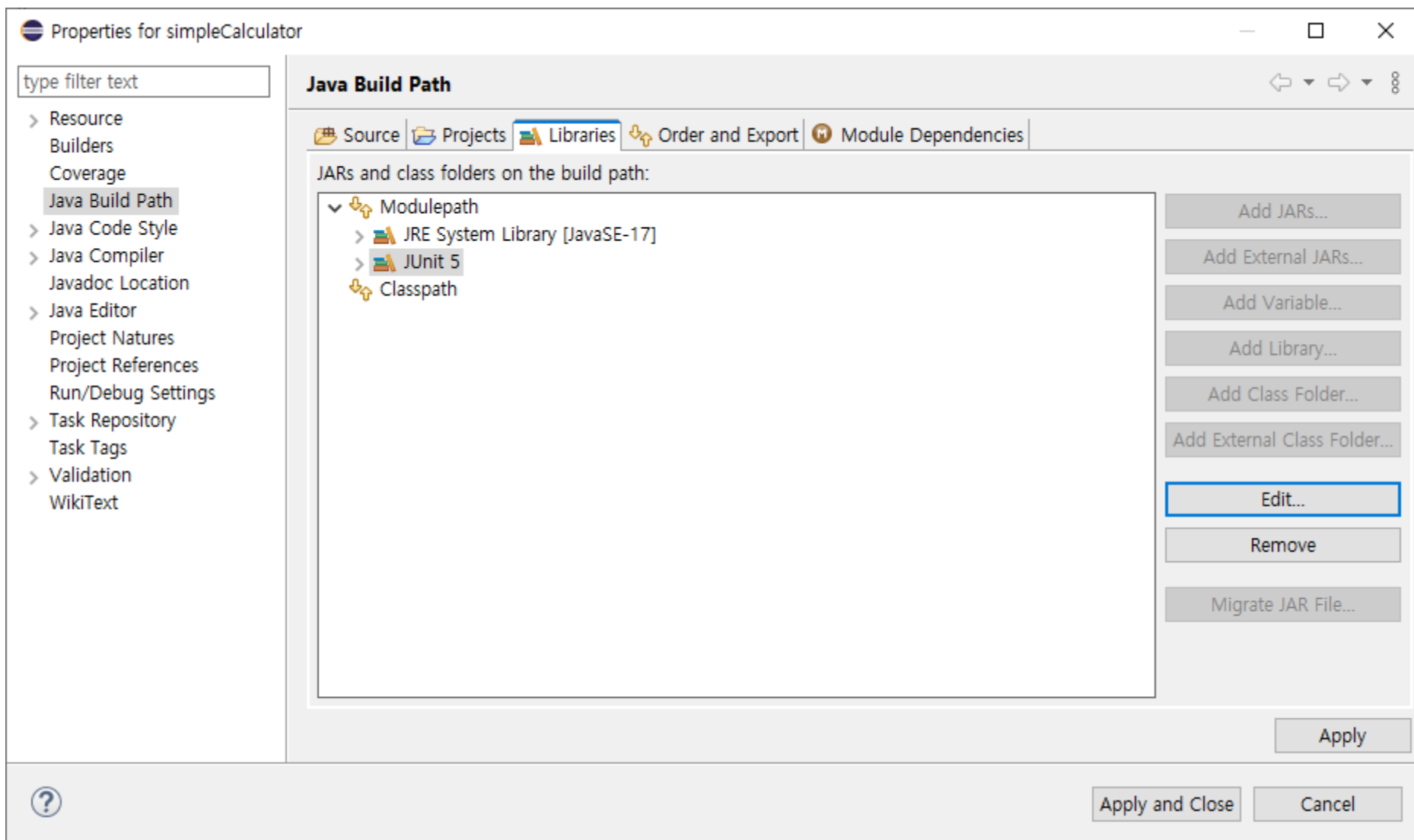
JUnit을 선택하고, Next를 클릭



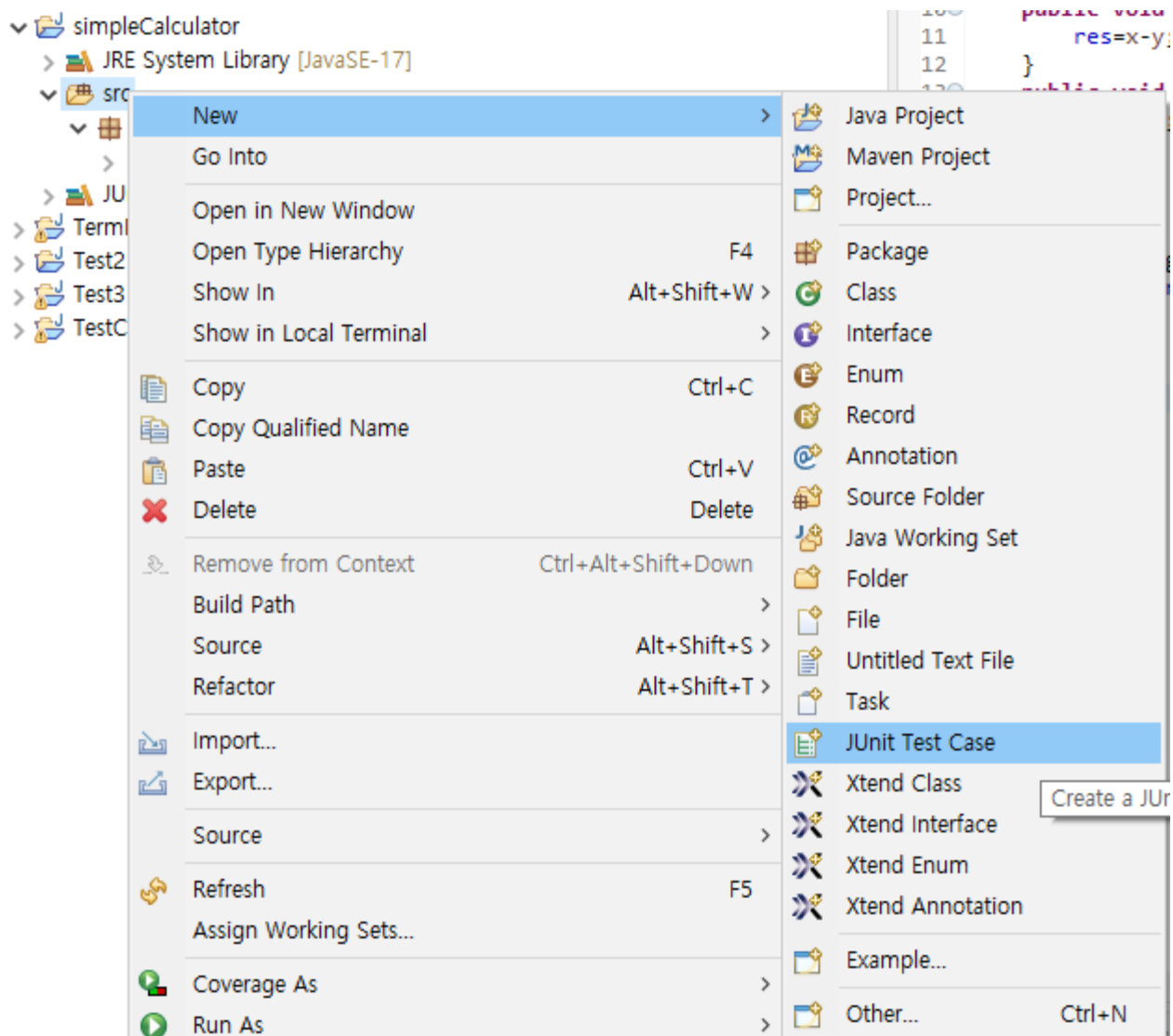
JUnit의 버전 선택 후 Finish 버튼



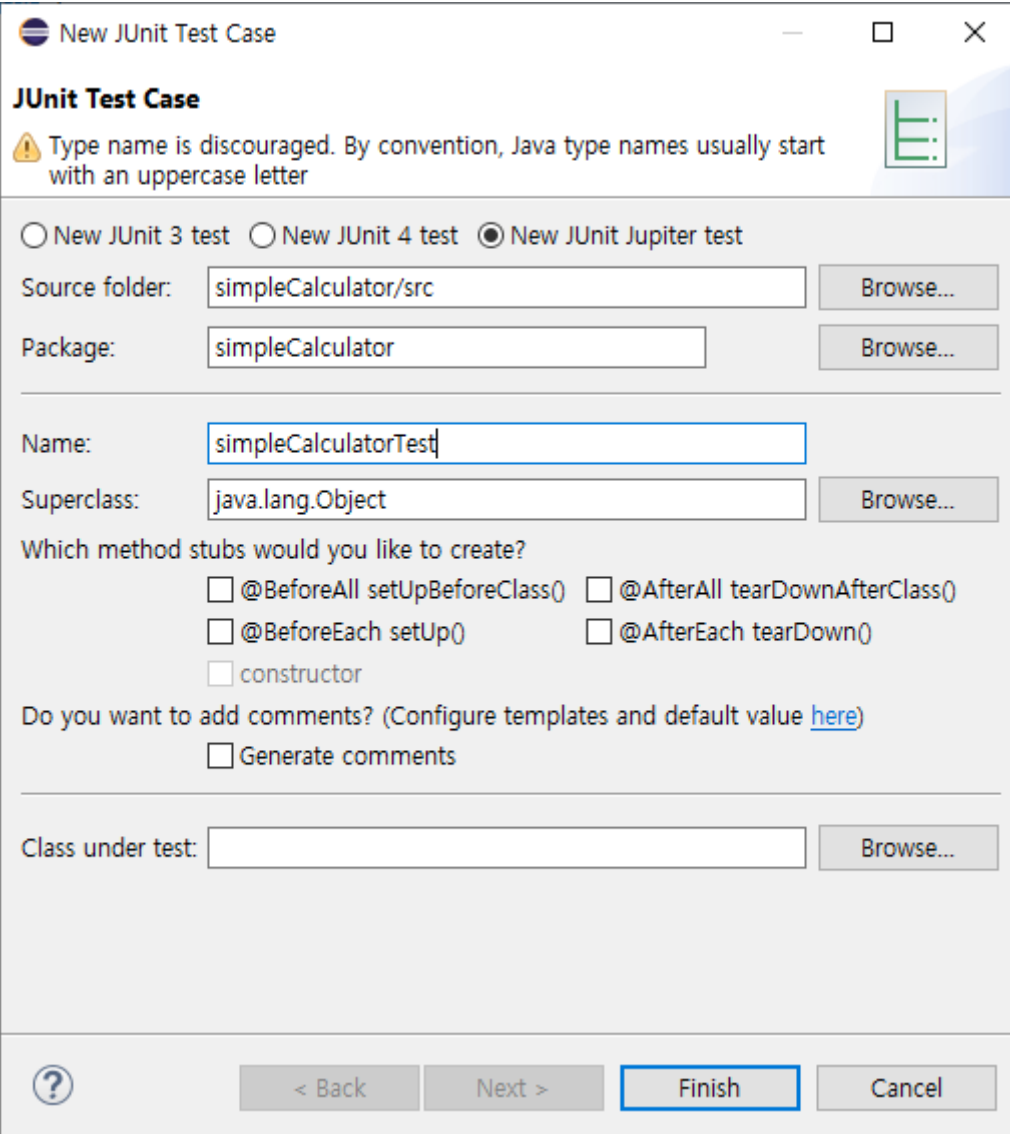
Properties 창에 JUnit 라이브러리가 추가된 것을 확인



JUnit 사용방법 : JUnit 테스트 클래스 생성



테스트 클래스 이름 입력 simpleCalculatorTest



New JUnit Test Case

JUnit Test Case

Type name is discouraged. By convention, Java type names usually start with an uppercase letter

☐ New JUnit 3 test ☐ New JUnit 4 test ☒ New JUnit Jupiter test

Source folder: simpleCalculator/src Browse...

Package: simpleCalculator Browse...

Name: simpleCalculatorTest

Superclass: java.lang.Object Browse...

Which method stubs would you like to create?

☐ @BeforeAll setUpBeforeClass() ☐ @AfterAll tearDownAfterClass()
☐ @BeforeEach setUp() ☐ @AfterEach tearDown()
☐ constructor

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

Class under test: Browse...

? < Back Next > Finish Cancel

다음과 같이 테스트 코드 작성, @Test 어노테이션
assertEquals를 이용하여 리턴 값이 지정한 값과 같은지를 확인하는 메서드

```
SimpleCalculator.java  SimpleCalculatorTest.java ×
1 package simpleCalculator;
2
3+ import static org.junit.jupiter.api.Assertions.assertEquals;
6
7 class SimpleCalculatorTest {
8
9-     @Test
10     public void testAdd() {
11         SimpleCalculator calc = new SimpleCalculator();
12         calc.add(10,20);
13         assertEquals(30, calc.getResult());
14     }
15
16 }
17
```

테스트 결과

Finished after 0.063 seconds

Runs: 1/1

Errors: 0

Failures: 0

SimpleCalculatorTest [Runner: JUnit 5] (0.000 s)

testAdd() (0.000 s)

Finished after 0.078 seconds

Runs: 1/1

Errors: 0

Failures: 1

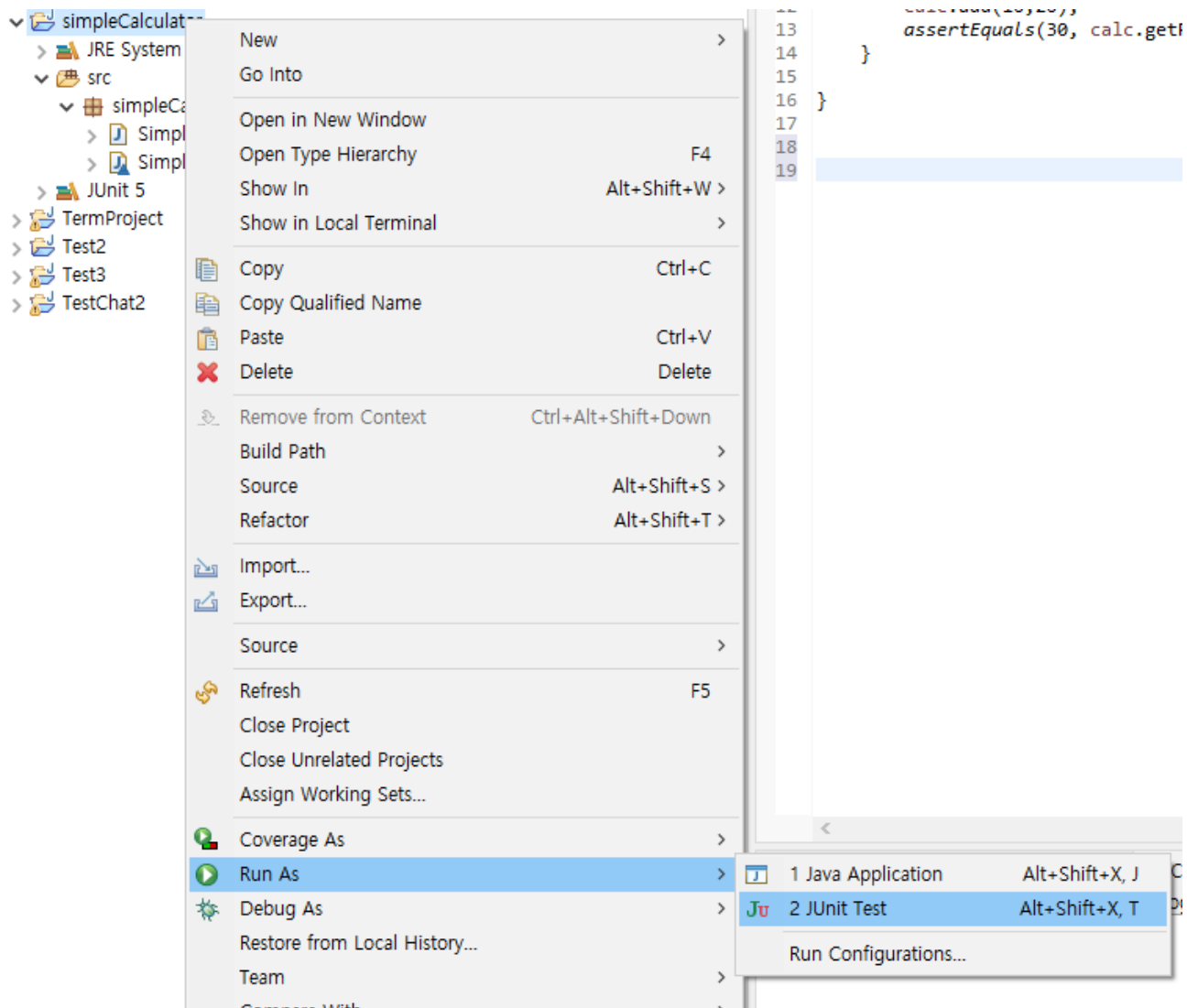
SimpleCalculatorTest [Runner: JUnit 5] (0.006 s)

testAdd() (0.006 s)

```
@Test
```

```
public void testAdd() {  
    SimpleCalculator calc = new SimpleCalculator();  
    calc.add(10,20);  
    assertEquals(40, calc.getResult());  
}
```

Run As > JUnit Test를 선택하여 테스트 클래스 실행



테스트 코드 추가

```
SimpleCalculator.java *SimpleCalculatorTest.java ×
1 package simpleCalculator;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4
5 import org.junit.jupiter.api.Test;
6
7 class SimpleCalculatorTest {
8
9     @Test
10    public void testAdd() {
11        SimpleCalculator calc = new SimpleCalculator();
12        calc.add(10,20);
13        assertEquals(30, calc.getResult());
14    }
15
16    @Test
17    public void testAdd0() {
18        SimpleCalculator calc = new SimpleCalculator();
19        calc.add(0,0);
20        assertEquals(0, calc.getResult());
21    }
22
23    @Test
24    public void testSub() {
25        SimpleCalculator calc = new SimpleCalculator();
26        calc.sub(10,20);
27        assertEquals(-10, calc.getResult());
28    }
29
30 }
```

실습

- 다음은 소프트웨어공학 과목에 학점을 주는 기준이다. 프로그램 코드와 Junit 테스트를 작성하시오.
 - ✓ 학점은 중간시험, 기말각각 35점이 만점이며 과제점수는 30점이 만점이다. 총 점수가 80점 이상이면 'A' 70점 이상이면 'B', 60점 이상이면 'C' 그 이하는 모두 'F'로 처리된다. 모든 입력은 정수 값으로 가정한다.
 - ✓ 테스트 통과 기준
 - 모든 학점이 반드시 한 번은 테스트 되어야 한다.