

CSE4006: Software Engineering

Lab 11. Testing (1)

Software Engineering Lab

Except where otherwise noted, the contents of this document are Copyright 2015 Gayeon Kim, Junghoon Lee, Scott Uk-Jin Lee. All rights reserved. Any redistribution, reproduction, transmission, or storage of part or all of the contents in any form is prohibited without the author's expressed written permission.

What is JUnit ?

JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development.

Features

- JUnit is an open source framework which is used for writing & running tests.
- Provides Annotation to identify the test methods.
- Provides Assertions for testing expected results.
- Provides Test runners for running tests.
- JUnit tests allow you to write code faster which increasing quality
- JUnit is elegantly simple. It is less complex & takes less time.
- JUnit tests can be run automatically and they check their own results and provide immediate feedback. There's no need to manually comb through a report of test results.
- JUnit tests can be organized into test suites containing test cases and even other test suites.
- Junit shows test progress in a bar that is green if test is going fine and it turns red when a test fails.

JUnit test fixture

```
import org.junit.*;

public class TestFoobar {
    @BeforeClass
    public static void setUpClass() throws Exception {
        // Code executed before the first test method
    }

    @Before
    public void setUp() throws Exception {
        // Code executed before each test
    }

    @Test
    public void testOneThing() {
        // Code that tests one thing
    }

    @Test
    public void testAnotherThing() {
        // Code that tests another thing
    }

    @Test
    public void testSomethingElse() {
        // Code that tests something else
    }

    @After
    public void tearDown() throws Exception {
        // Code executed after each test
    }

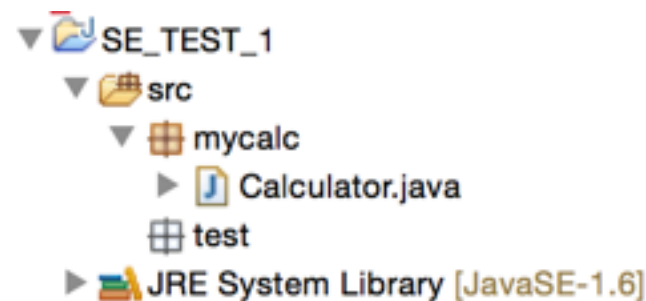
    @AfterClass
    public static void tearDownClass() throws Exception {
        // Code executed after the last test method
    }
}
```

JUnit Tutorial

File > New Java Project

File > New Package

File > New Class

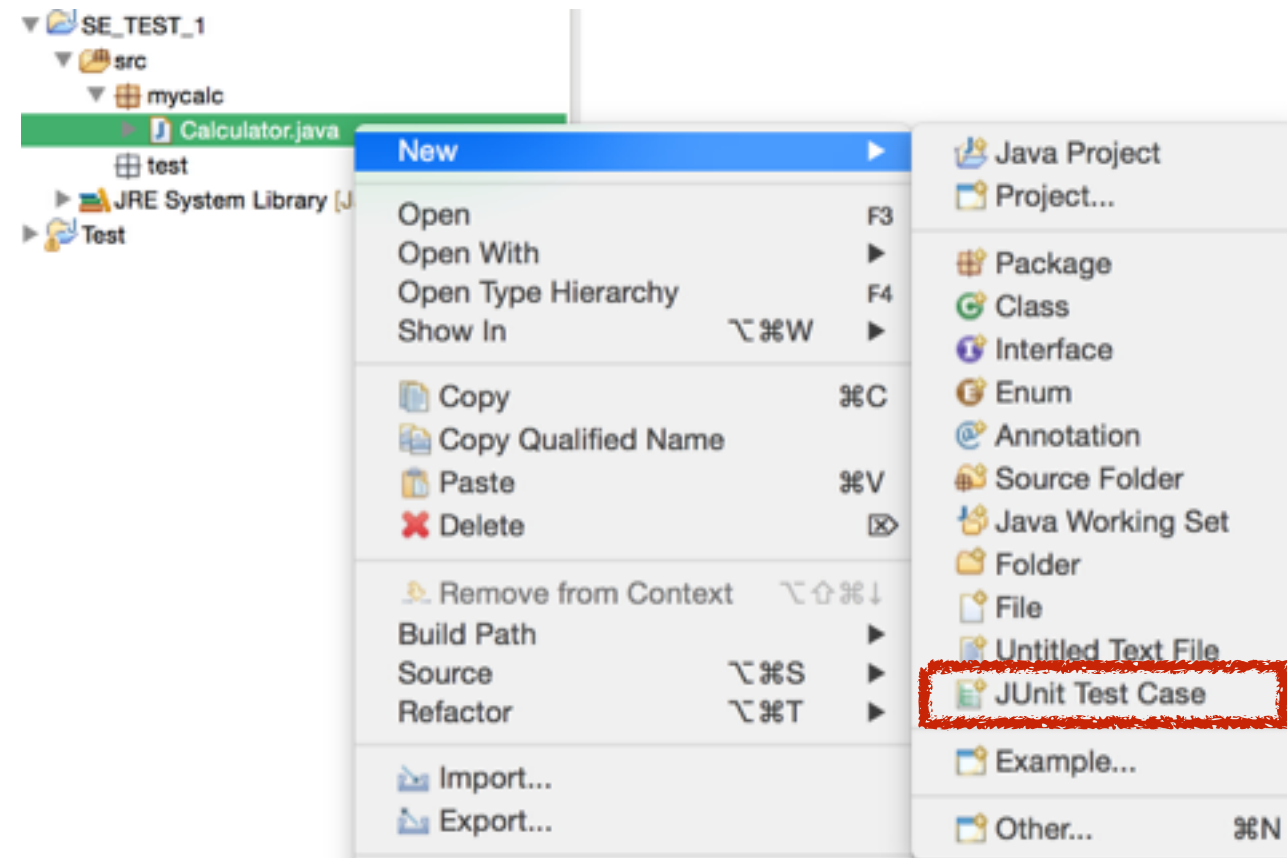


```
Calculator.java
1 package mycalc;
2
3 public class Calculator {
4     public Integer add(double a, double b) {
5         return (int)(a+b);
6     }
7
8     public Integer minus(int a, int b) {
9         return new Integer(String.format("%d", a - b));
10    }
11    public Integer mult(int a, int b) {
12        return new Integer(String.format("%d", a * b));
13    }
14
15    public int div(int a, int b) {
16        return new Integer(String.format("%d", a / b));
17    }
18
19 }
20
21
```

Simple Calculator JavaCode

download code : <http://pasted.co/b32693f3>

JUnit Tutorial (cont'd)



File > New > JUnit Test Case

JUnit Tutorial (cont'd)

here)' with an unchecked checkbox 'Generate comments'. At the very bottom are buttons: '?', '< Back', 'Next >', 'Cancel', and 'Finish'."/>

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:

Package:

Name:

Superclass:

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:

optional ->

New JUnit Test Case

! JUnit 4 is not on the build path. Do you want to add it?

☐ Not now
☐ Open the build path property page
☒ Perform the following action:

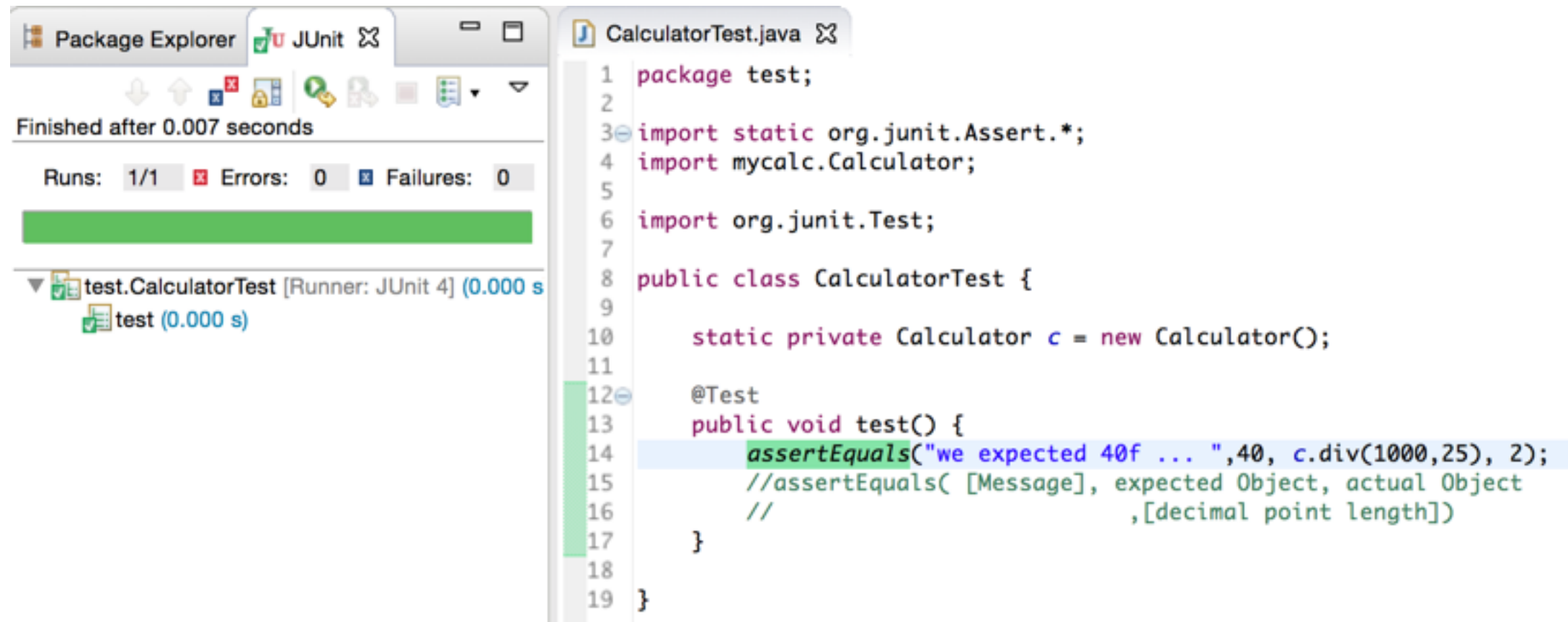
☒ Add JUnit 4 library to the build path

JUnit Tutorial (cont'd)

```
CalculatorTest.java
1 package test;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.After;
6 import org.junit.AfterClass;
7 import org.junit.Before;
8 import org.junit.BeforeClass;
9 import org.junit.Test;
10
11 public class CalculatorTest {
12
13     @BeforeClass
14     public static void setUpBeforeClass() throws Exception {
15     }
16
17     @AfterClass
18     public static void tearDownAfterClass() throws Exception {
19     }
20
21     @Before
22     public void setUp() throws Exception {
23     }
24
25     @After
26     public void tearDown() throws Exception {
27     }
28
29     @Test
30     public void test() {
31         fail("Not yet implemented");
32     }
33
34 }
```

Generated testing code, but not work.

JUnit Tutorial (cont'd)



The screenshot displays an IDE interface with two main panels. The left panel, titled 'JUnit', shows the test execution results. It indicates that the test 'test' in the package 'test' completed successfully after 0.007 seconds. The summary shows 1/1 runs, 0 errors, and 0 failures. A green progress bar is visible. Below the summary, a tree view shows the test hierarchy: 'test.CalculatorTest [Runner: JUnit 4] (0.000 s)' and 'test (0.000 s)'. The right panel shows the source code for 'CalculatorTest.java'. The code is as follows:

```
1 package test;
2
3 import static org.junit.Assert.*;
4 import mycalc.Calculator;
5
6 import org.junit.Test;
7
8 public class CalculatorTest {
9
10     static private Calculator c = new Calculator();
11
12     @Test
13     public void test() {
14         assertEquals("we expected 40f ... ", 40, c.div(1000, 25), 2);
15         //assertEquals( [Message], expected Object, actual Object
16         //                , [decimal point length])
17     }
18
19 }
```

Simplest unit testing code_(right) and result_(left)

JUnit Tutorial (cont'd)

CalculatorTest.java

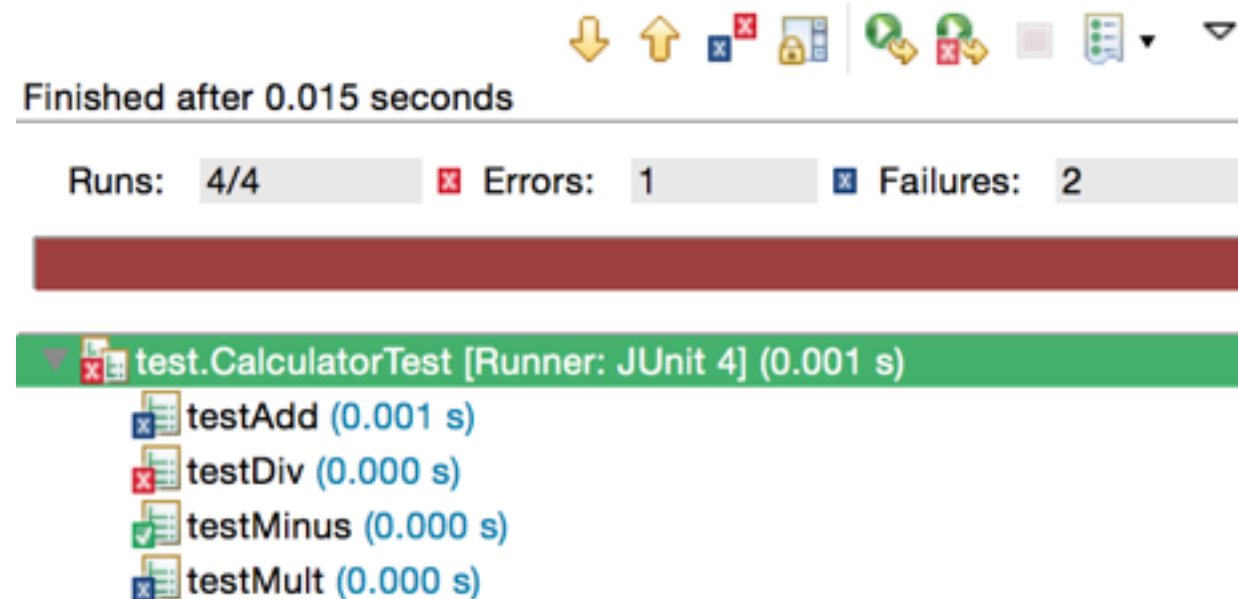
```
1 package test;
2
3 import static org.junit.Assert.*;
4 import mycalc.Calculator;
5
6 import org.junit.After;
7 import org.junit.AfterClass;
8 import org.junit.Before;
9 import org.junit.BeforeClass;
10 import org.junit.Test;
11
12 public class CalculatorTest {
13
14     static private Calculator c;
15     static private int i = 1;
16
17     @BeforeClass
18     public static void setUpBeforeClass() throws Exception {
19         c = new Calculator();
20         System.out.println("-- Start Test");
21     }
22
23     @AfterClass
24     public static void tearDownAfterClass() throws Exception {
25         System.out.println("-- Finish Test");
26     }
27
28     @Before
29     public void setUp() throws Exception {
30         System.out.println("Unit Test "+i+" Start");
31     }
32
33     @After
34     public void tearDown() throws Exception {
35         System.out.println("Unit Test "+i+" End");
36         i++;
37     }
38 }
```

```
39
40 @Test
41 public void testAdd() {
42     Object r = c.add(1,1);
43     assertEquals("ADD test..1", 2, r);
44     // 2^27 + 2^31
45     double r2 = c.add(1, 2147483647);
46     assertEquals("ADD test..2", 2147483647d + 1d, r2, 0);
47 }
48 @Test
49 public void testMinus() {
50     Object r = c.minus(10,100);
51     assertEquals("MINUT test..1", -90, r);
52
53     r = c.minus(-2147483647,-1);
54     assertEquals("MINUT test..2", -2147483647 - (-1), r);
55 }
56 @Test
57 public void testMult() {
58     Object r = c.mult(1,0);
59     assertEquals("MULT test..1", 0, r);
60
61     //2^16 * 2^16 = 4294967296
62     double r2 = c.mult(65536,65536);
63     assertEquals("MULT test..2", 65536d * 65536d, r2, 0);
64 }
65 @Test
66 public void testDiv() {
67     double r = c.div(10,100);
68     assertEquals("DIV test..1", 0.01, r, 4);
69
70     r = c.div(1,0);
71     assertEquals("DIV test..2", 0, r, 4);
72 }
73 }
```

more detail ...

lab(se);

JUnit Tutorial (cont'd)



Finished after 0.015 seconds

Runs: 4/4 Errors: 1 Failures: 2

test.CalculatorTest [Runner: JUnit 4] (0.001 s)

- testAdd (0.001 s)
- testDiv (0.000 s)
- testMinus (0.000 s)
- testMult (0.000 s)

The screenshot shows the JUnit test results in an IDE. At the top, it says 'Finished after 0.015 seconds'. Below that, a summary bar shows 'Runs: 4/4', 'Errors: 1', and 'Failures: 2'. A red progress bar is visible. The test suite 'test.CalculatorTest [Runner: JUnit 4] (0.001 s)' is expanded, showing four test methods: 'testAdd (0.001 s)', 'testDiv (0.000 s)', 'testMinus (0.000 s)', and 'testMult (0.000 s)'. 'testAdd' and 'testDiv' are marked with red 'X' icons, indicating failures, while 'testMinus' and 'testMult' are marked with green checkmarks, indicating success.

Result

Failure Trace

java.lang.AssertionError: ADD test..2 expected:<2.147483648E9> but was:<2.147483647E9>
at test.CalculatorTest.testAdd(CalculatorTest.java:45)

testAdd() Failure Trace

Failure Trace

java.lang.ArithmeticException: / by zero
at mycalc.Calculator.div(Calculator.java:16)
at test.CalculatorTest.testDiv(CalculatorTest.java:69)

testDiv() Failure Trace

Exercise.

- Write test code for each method. (makeText(), reverseText(), halfText())
- Check **halfTest()** is equal to **halfText2()**.

```
1 package code;
2
3
4 public class TextGenarator {
5
6     public TextGenarator() {
7
8     }
9
10    public String makeText(String origin, int mult) {
11        String r = "";
12        for (int i = 0; i < mult; i++) {
13            r += origin;
14        }
15        return r;
16    }
17
18    public String reverseText(String origin) {
19        return new StringBuffer(origin).reverse().toString();
20    }
21
22    public String halfText(String origin) {
23        return origin.substring(0, origin.length()/2);
24    }
25
26    public String halfText2(String origin) {
27        String r = "";
28        for (int i = 0; i <= origin.length()/2; i++) {
29            r += origin.charAt(i);
30        }
31        return r;
32    }
33 }
```

download code : <http://pasted.co/82ced221>

Submission

1. Take screenshots your code and results
2. Compress your screenshots to 20xx03xxxx_YourName.zip
3. Send mail to ng0301@gmail.com