HONG KONG INSTITUTE OF VOCATIONAL EDUCATION

**Laboratory 8: Introduction to JUnit**

**Module Intended Learning Outcome:**

On completion of the module, students are expected to be able to:

* Apply software testing techniques in various software development stages.

**ENVIRONMENT:**

JDK 8

NetBeans IDE 8.2

**TASK:**

1. A simple example **SimpleTest** in lab08 directory will illustrate the basic concepts involved in testing with JUnit.

**import junit.framework.\*;**

Since we are using some constructs created by the makers of JUnit we must import any of the classes we desire to use, most of these reside in the framework subdirectory, hence the import statement.

**public class SimpleTest extends TestCase {**

**protected int fValue1;**

**protected int fValue2;**

Our simple class needs to define its own test method(s) to actually be of any use so it extends TestCase which provides us with the ability to define our own test methods.

**protected void setUp() {**

**fValue1= 3;**

**fValue2= 3;**

**}**

TestCase allows us to use the method ***setUp***to set up any necessary variables or objects. ***setUp***is called before the evaluation of each test.

**public static Test suite() {**

**return new TestSuite(SimpleTest.class);**

**}**

**public static void main (String[] args) {**

**junit.textui.TestRunner.run(suite());**

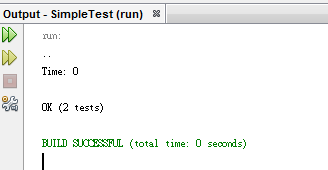
**}**

The main and suite methods help to construct the testing class.

|  |
| --- |
| **public void testAdd() {**  **int expectedResult = 5;**    **int v = add(2, 3);**  **Assert.assertTrue(v == expectedResult); // should be true**  **}**  **public void testEquals() {**  **Assert.assertEquals(fValue1, fValue2);**  **Assert.assertEquals(12, 12);**  **Assert.assertEquals(add(3,7), 10);**  **}** |

**}**

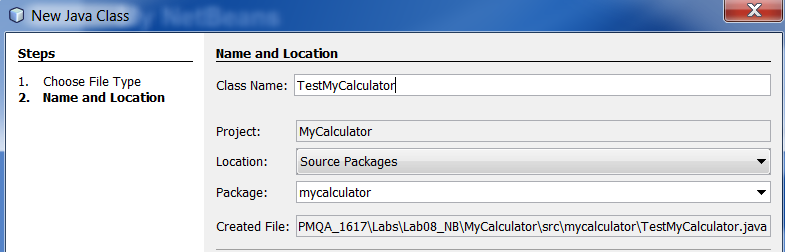
2. Run and you get:



3. Modify the test cases and see the differences.

***Part 2: JUnit Using NetBeans:***

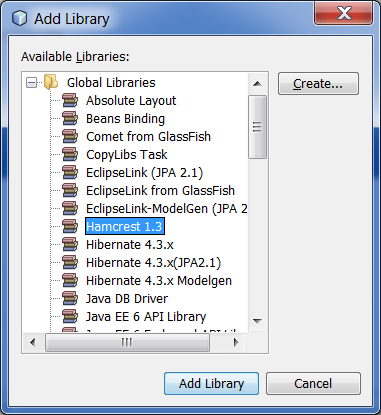
1. Open NetBeans;
2. Open the Project called **MyCalculator**, try run it, observe the output (there is a problem).
3. Create a new Java Class **TestMyCalculator**



1. Add **JUnit** library to project

|  |  |
| --- | --- |
|  |  |

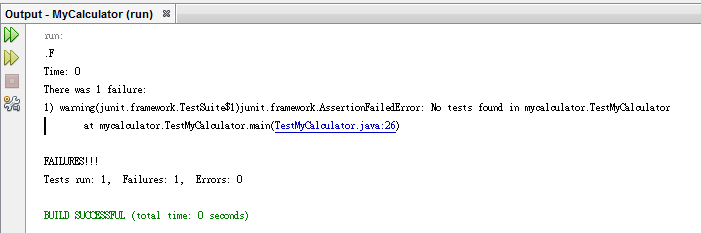
Add one more library: **Hamcrest**



1. Modify the TestMyCalculator.java as below:

|  |
| --- |
| package mycalculator;  // JUnit packages  import org.junit.Assert;  import junit.framework.Test;  import junit.framework.TestCase;  import junit.framework.TestSuite;  public class TestMyCalculator extends TestCase{  MyCalculator myCalc;    // setUp() will be called when EACH test starts  protected void setUp() {  myCalc = new MyCalculator();  }  public static Test suite() {  return new TestSuite(TestMyCalculator.class);  }  public static void main(String[] args) {  junit.textui.TestRunner.run(suite());  }  } |

Try run it (Right click TestMyCalculator.java, Run File)



Hints: It’s normal, as you have not defined any test case yet.

1. Add one test case to test the sub() method, run it

|  |
| --- |
| public void testSub1() {  int expectedResult = 5;    int v = myCalc.sub(8, 3);  Assert.assertTrue(v == expectedResult);  } |

Hints: You should get an assertion error, why? Your sub() has a bug!

1. Fix the **sub()** method and rerun the test, you should pass the test.

Graphical user interface, text, email

Description automatically generated

1. Add a test case **testCalculate1()**: verify 10 + 3 - 2 - 11 = 0, use assertTrue().

Hints: You must call add() and sub() !!!

public void testCalulate1() {

int expectedResult = 0;

int v = myCalc.add(10, 3);

v = myCalc.sub(v, 2);

v = myCalc.sub(v, 11);

Assert.assertFalse(v == expectedResult);

}

1. Add a test case **testCalculate2()**: verify 10 + 3 - 2 - 11 is not equal to 2, use assertFalse().

public void testCalulate2() {

int expectedResult = 2;

int v = myCalc.add(10, 3);

v = myCalc.sub(v, 2);

v = myCalc.sub(v, 11);

Assert.assertFalse(v == expectedResult);

}

1. Add a test case **testCalculate3()**: Assume that buying a lunch box costs $5, you have $37. Verify that you can buy 7 lunch boxes, use assertTrue(). Hints: Not just divide 37 by 5!!!

public void testCalulate3() {

int expectedResult = 7;

int money = 37;

int v = 0;

while (money >= 5) {

money -= 5;

v++;

}

Assert.assertTrue(v == expectedResult);

}

1. Add the following in your TestMyCalculator class

|  |
| --- |
| static int m = 0;  private static void buyCoffee() {  m -= 10;  }  private static void buyLemonTea() {  m -= 15;  } |

Add the following line in the setUp() method

|  |
| --- |
| m = 100; |

1. Add a test case **testCalculate4()**: Verify that after buying 1 coffee and 1 lemon tea, you have have $75 left. Use assertTrue().

public void testCalulate4() {

int expectedResult = 75;

buyCoffee();

buyLemonTea();

Assert.assertTrue(m == expectedResult);

}

1. Add a test case **testCalculate5()**: Verify that you cannot buy 11 coffee. Use assertFalse().

public void testCalulate5() {

for (int i=0; i< 11; i++) {

buyCoffee();

}

Assert.assertFalse(m >= 0);

}