JUnit

Using JUNIT from Eclipse

Agenda (Contd.).

JUnit with Eclipse

Objectives

At the end of this module, you will be able to:

Use JUNIT from Eclipse..

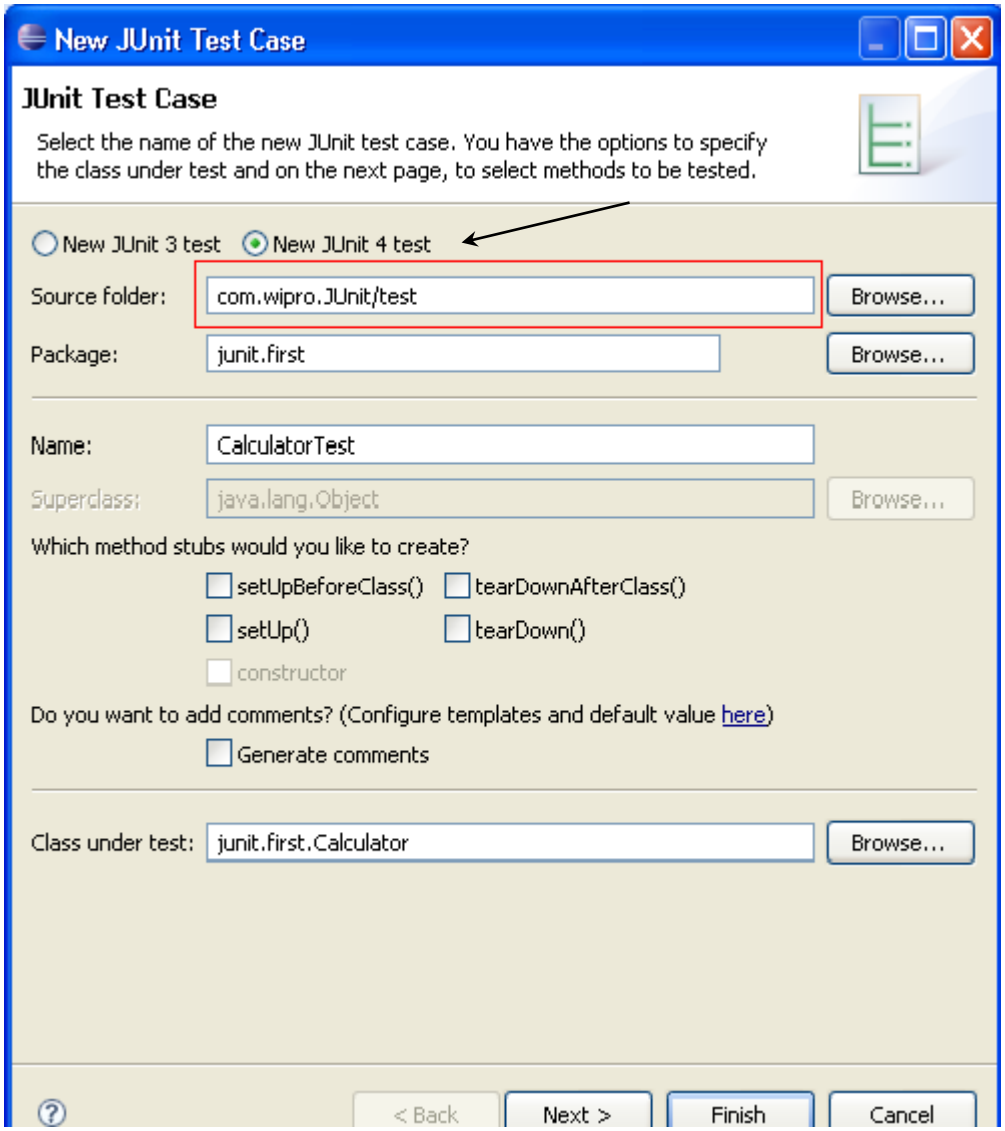
JUnit with Eclipse

JUnit with Eclipse

1. Create a new Project com.wipro.JUnit
2. Add JUnit.jar to the Classpath
3. Right click the Project and create a new Source folder called ‘test’
4. Create a new Java class called Calculator in a package junit.first
5. Add 2 methods add and sub to the Calculator class which does addition and subtraction of 2 numbers respectively

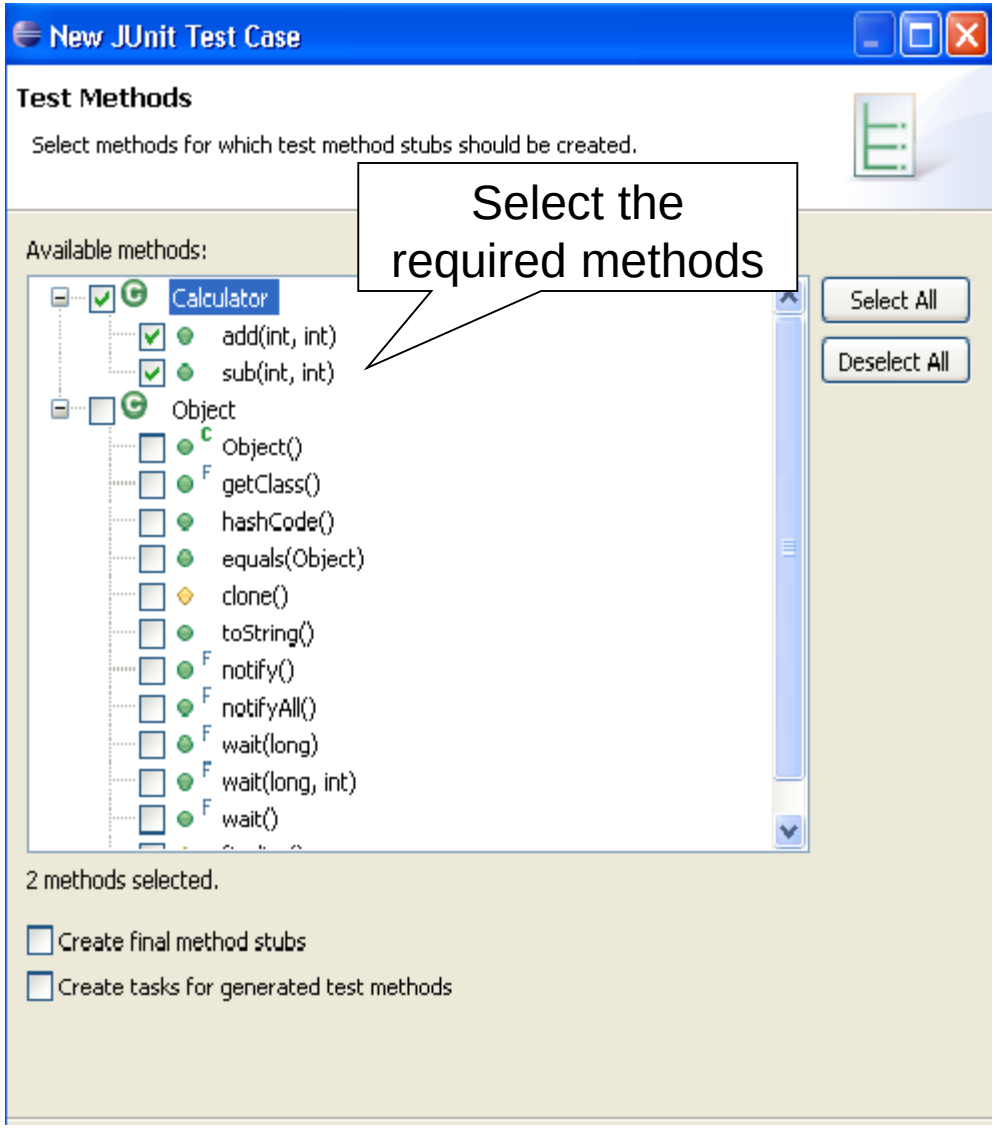
```
package junit.first;
public class Calculator {
public int add(int x,int y)
{ return x+y; }
public int sub(int x,int y)
{ return x-y; } }
```

JUnit with Eclipse (Contd.).



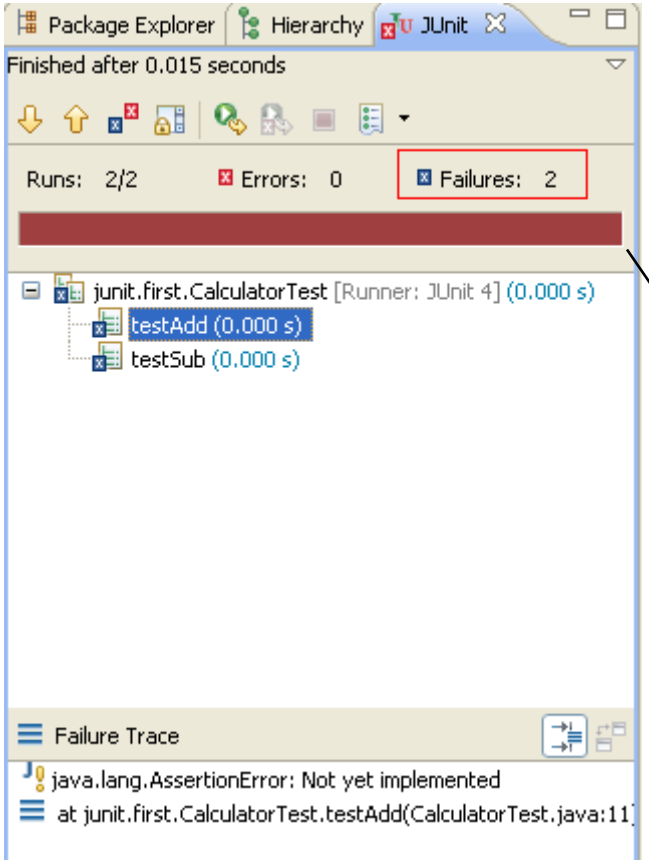
- Right click on the Calculator class in the Package Explorer and select New->JUnitTestCase
- select "New JUnit4 test“
- set the source folder to "test“ – the test class gets created here

JUnit with Eclipse (Contd.).



- Press "Next" and select the methods you want to test

JUnit with Eclipse (Contd.).



- Right click on CalculatorTest class and select

  Run-As → JUnit Test
- The results of the test will be displayed in JUnit view
- This is because the testAdd and testSub are not implemented correctly

Brown color indicates failure

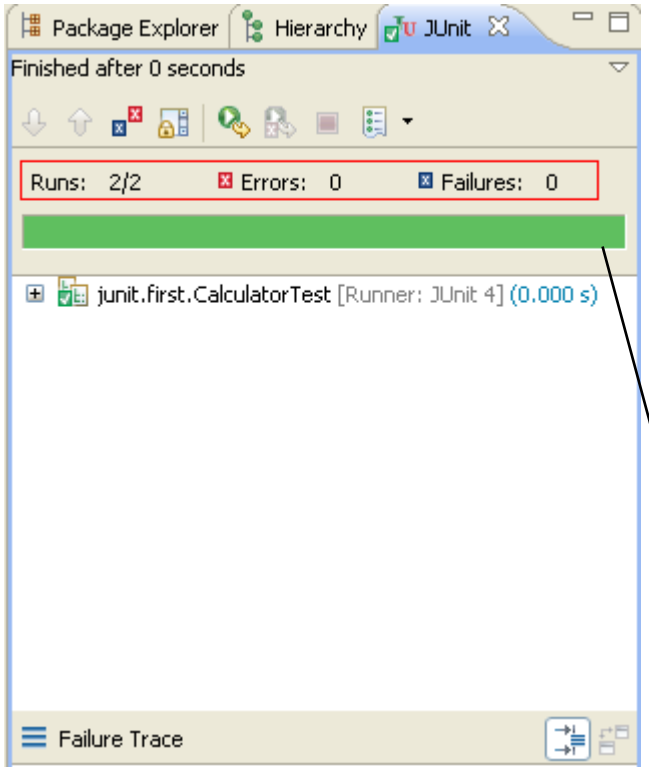
How to write a JUnit test method

- All the test methods should be marked with the JUnit annotation - @org.junit.Test
- All the JUnit test methods should be "public" methods
- The return type of the JUnit test method must be “void"
- The test method need not start with the test keyword
- Here is a simple JUnit test method:

```
@Test
public void testAdd()
{
   Calculator c=new Calculator();
   assertEquals("Result",5,c.add(2,3));
}
```

JUnit with Eclipse

- Now let's provide implementation to the code and run the test again




```
package junit.first;
import static org.junit.Assert.*;
import org.junit.Test;
public class CalculatorTest {
@Test
public void testAdd() {
Calculator c=new Calculator();
assertEquals(5,c.add(2,3));
}
@Test
public void testSub() {
Calculator c=new Calculator();
assertEquals(20,c.sub(100,80));
} }
```

Green color indicates pass

JUnit with Eclipse

- Unit tests are implemented as classes with test methods. Each test method usually tests a single method of the target class. Sometimes, a test method can test more than one method in the target class, and sometimes, if the method to test is big, you split the test into multiple test methods.
- The unit test class is an ordinary class, with two methods, tesAdd() and testSub. Notice how this method is annotated with the JUnit annotation @Test. This is done to signal to the unit test runner, that this is method represents a unit test, that should be executed. Methods that are not annotated with @Test are not executed by the test runner.

JUnit with Eclipse

- Inside the testAdd() method an instance of Calculator is created. Then it's add() method is called with two integer values.
- Finally, the assertEquals() method is called. It is this method that does the actual testing. In this method we compare the output of the called method (add()) with the expected output.
- If the two values are equal, nothing happens. The assertEquals() method returns normally. If the two values are not equal, an exception is thrown, and the test method stops executing here.
- The assertEquals() method is a statically imported method, which normally resides in the org.junit.Assert class. Notice the static import of this class at the top of MyUnitTest. Using the static import of the method is shorter than writing Assert.assertEquals().

Quiz

- What is meant by import static org.junit.Assert.*;
- What is the content of the JAR file? How you will add a Jar file to your project?
- What is meant by SDLC? What are the phases of SDLC?
- Testing is done by Testing team; Then, why a programmer like you should worry about testing?

Summary

- JUnit with Eclipse

Thank You