**Assertions**

**Assert is a method useful in determining Pass or Fail status of a test case, The assert methods are provided by the class org.junit.Assert which extends java.lang.Object class.**

There are various types of assertions like Boolean, Null, Identical etc.

Junit provides a class named Assert, which provides a bunch of assertion methods useful in writing test cases and to detect test failure

The assert methods are provided by the class org.junit.Assert which extends java.lang.Object class.

In this tutorial, you will learn-

## JUnit Assert methods

### Boolean

If you want to test the boolean conditions (true or false), you can use following assert methods

1. assertTrue(condition)
2. assertFalse(condition)

Here the condition is a boolean value.

### Null object

If you want to check the initial value of an object/variable, you have the following methods:

1. assertNull(object)
2. assertNotNull(object)

Here object java object e.g. assertNull(actual);

### Identical

If you want to check whether the objects are identical (i.e. comparing two references to the same java object), or different.

1. assertSame(expected, actual), It will return true if expected == actual
2. assertNotSame(expected, actual)

### Assert Equals

If you want to test equality of two objects, you have the following methods

* assertEquals(expected, actual)

It will return true if: expected.equals( actual ) returns true.

### Assert Array Equals

If you want to test equality of arrays, you have the following methods as given below:

* assertArrayEquals(expected, actual)

Above method must be used if arrays have the same length, for each valid value for i, you can check it as given below:

* assertEquals(expected[i],actual[i])
* assertArrayEquals(expected[i],actual[i])

### Fail Message

If you want to throw any assertion error, you have fail() that always results in a fail verdict.

* Fail(message);

You can have assertion method with an additional String parameter as the first parameter. This string will be appended in the failure message if the assertion fails. E.g. fail( message ) can be written as

* assertEquals( message, expected, actual)

## JUnit assertEquals

You have assertEquals(a,b) which relies on the equals() method of the Object class.

* Here it will be evaluated as a.equals( b ).
* Here the class under test is used to determine a suitable equality relation.
* If a class does not override the equals() method of Object class, it will get the default behaviour of equals() method, i.e. object identity.

If a and b are primitives such as byte, int, boolean, etc. then the following will be done for assertEquals(a,b) :

a and b will be converted to their equivalent wrapper object type (Byte,Integer, Boolean, etc.), and then a.equals( b ) will be evaluated.

For Example: Consider below-mentioned strings having same values, let's test it using assertTrue

String obj1="Junit";

String obj2="Junit";

assertEquals(obj1,obj2);

Above assert statement will return true as obj1.equals(obj2) returns true.

## Floating point assertions

When you want to compare floating point types (e.g. double or float), you need an additional required parameter delta to avoid problems with round-off errors while doing floating point comparisons.

The assertion evaluates as given below:

* Math.abs( expected – actual ) <= delta

For example:

assertEquals( aDoubleValue, anotherDoubleValue, 0.001 )

## JUnit Assert Example

Below example demonstrates how to assert a condition using JUnit assert methods.

Let's create a simple test class named Junit4AssertionTest.java and a test runner class TestRunner.java.

You will create few variables and important assert statements in JUnit.

In this example, you will execute our test class using TestRunner.java

Step 1) Lets create a class covering all important assert statement methods in junit:

Junit4AssertionTest.java

package guru99.junit;

import static org.junit.Assert.\*;

import org.junit.Test;

public class Junit4AssertionTest {

@Test

public void testAssert(){

//Variable declaration

String string1="Junit";

String string2="Junit";

String string3="test";

String string4="test";

String string5=null;

int variable1=1;

int variable2=2;

int[] airethematicArrary1 = { 1, 2, 3 };

int[] airethematicArrary2 = { 1, 2, 3 };

//Assert statements

assertEquals(string1,string2);

assertSame(string3, string4);

assertNotSame(string1, string3);

assertNotNull(string1);

assertNull(string5);

assertTrue(variable1<variable2);

assertArrayEquals(airethematicArrary1, airethematicArrary2);

}

}

Step 2) You need to create a test runner class to execute above class:

TestRunner.java

package guru99.junit;

import org.junit.runner.JUnitCore;

import org.junit.runner.Result;

import org.junit.runner.notification.Failure;

public class TestRunner {

public static void main(String[] args) {

Result result = JUnitCore.runClasses(Junit4AssertionTest.class);

for (Failure failure : result.getFailures()) {

System.out.println(failure.toString());

}

System.out.println("Result=="+result.wasSuccessful());

}

}

Step 3) Lets analyse expected output step by step:

Consider all assert statements one by one:

1. assertEquals(string1,string2);

Now compare string1=" Junit" with string2=" Junit" with equals method of object class. Replacing assertEquals method from java.lang.Object.equals() method :

string1.equals(string2)=> returns true

So assertEquals(string1,string2) will return true.

1. assertSame(string3, string4);

"assertSame()" functionality is to check that the two objects refer to the same object.

Since string3="test" and string4="test" means both string3 and string4 are of the same type so assertSame(string3, string4) will return true.

1. assertNotSame(string1, string3);

"assertNotSame()" functionality is to check that the two objects do not refer to the same object.

Since string1="Junit" and string3="test" means both string1 and string3 are of different types, so assertNotSame(string1, string3) will return true.

1. assertNotNull(string1);

"assertNotNull()" functionality is to check that an object is not null.

Since string1= "Junit" which is a non-null value so assertNotNull(string1) will return true.

1. assertNull(string5);

"assertNull()" functionality is to check that an object is null.

Since string5= null which is a null value so assertNull(string5) will return true.

1. assertTrue(variable1<variable2);

"assertTrue()" functionality is to check that a condition is true.

Since variable1=1 and variable2=2, which shows that variable1<variable2 condition is true so assertTrue(variable1<variable2) will return true.

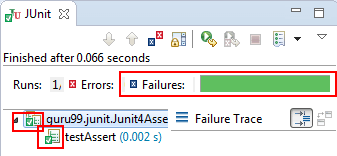
1. assertArrayEquals(airethematicArrary1, airethematicArrary2);

"assertArrayEquals()" functionality is to check that the expected array and the resulted array are equal. The type of Array might be int, long, short, char, byte or java.lang.Object.

Since airethematicArrary1 = { 1, 2, 3 } and airethematicArrary2 = { 1, 2, 3 } which shows both the arrays are equal so assertArrayEquals(airethematicArrary1, airethematicArrary2) will return true

Since all seven assert statements of Junit4AssertionTest.java class returns true, therefore when you execute the test assert class, it will return a successful test. (see the output below)

Step 4) Right click on Junit4AssertionTest.java and click on runAs->JUnit. You will see the output as given below:



Above output shows a successful test result as expected.