Assertions



**Steps** : Explaining Assertion Methods:

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

assertTrue(condition)

assertFalse(condition)

Here, the condition is a boolean value.

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

assertNull(object)

assertNotNull(object)

Here, the object is a[Java](https://www.guru99.com/java-tutorial.html)object, for e.g**.** assertNull(actual);

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

assertSame(expected, actual), It will return true if expected == actual

assertNotSame(expected, actual)

### Assert Equals: If you want to test the 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 :

assertArrayEquals(expected, actual)

The above method must be used if the arrays have the same length for each valid value for **i** as shown 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 the assertion method with an additional stringparameter 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

 assertEquals(a , b)which relies on theequals()method of the Object class.

* 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 that the below-mentioned strings have the same values, let's test it using assertTrue.

String obj1="Junit";

String obj2="Junit";

assertEquals (obj1 , obj2);

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

## Floating point assertions

When you want to compare the 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 )

**Steps 4.2.2:** Writing code for Assertions

Let's use some of the above-mentioned methods in an example. Create a java class file named **TestAssertions.java** in /home/ubuntu/Desktop/JUNIT\_WORKSPACE/.

import org.junit.Test;

import static org.junit.Assert.\*;

public class TestAssertions {

@Test

public void testAssertions() {

//test data

String str1 = new String ("abc");

String str2 = new String ("abc");

String str3 = null;

String str4 = "abc";

String str5 = "abc";

int val1 = 5;

int val2 = 6;

String[] expectedArray = {"one", "two", "three"};

String[] resultArray = {"one", "two", "three"};

//Check that two objects are equal

assertEquals(str1, str2);

//Check that a condition is true

assertTrue (val1 < val2);

//Check that a condition is false

assertFalse(val1 > val2);

//Check that an object isn't null

assertNotNull(str1);

//Check that an object is null

assertNull(str3);

//Check if two object references point to the same object

assertSame(str4,str5);

//Check if two object references not point to the same object

assertNotSame(str1,str3);

//Check whether two arrays are equal to each other.

assertArrayEquals(expectedArray, resultArray);

}

}

Next, create a java class file named **TestRunner.java** in /home/ubuntu/Desktop/JUNIT\_WORKSPACE/ to execute the test case(s).

import org.junit.runner.JUnitCore;

import org.junit.runner.Result;

import org.junit.runner.notification.Failure;

public class TestRunner2 {

public static void main(String[] args) {

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

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

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

}

System.out.println(result.wasSuccessful());

}

}

Compile the Test Case and Test Runner classes using javac.

/home/ubuntu/Desktop/JUNIT\_WORKSPACE/>javac TestAssertions.java TestRunner.java

Now run the Runner, which will run the test case defined in the provided Test Case Class.

/home/ubuntu/Desktop/JUNIT\_WORKSPACE/>java TestRunner

Verify the output.

true

**Step 4.2.3:** Pushing the code to GitHub repositories:

Open your command prompt and navigate to the folder where you have created your files:

cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add .

Commit the changes using the following command:

git commit . -m “Changes have been committed.”

Push the files to the folder you initially created using the following command:

git push -u origin master