SENG 301 - Software Analysis and Design

Lab 6- Automated Testing /Junit Tutorial

May Mahmoud

Automated Testing

- Automated software testing is a process in which software tools execute pre-scripted tests on a software application before it is released into production.

- Automated testing tools are capable of executing tests, reporting outcomes and comparing results with earlier test runs. Tests carried out with these tools can be run repeatedly, at any time of day.

What is JUnit?

- *JUnit* in version 4.x is a test framework which uses annotations to identify methods that specify a test

- Available in latest Eclipse version

- A JUnit *test* is a method contained in a class which is only used for testing. This is called a *Test class*. To write a JUnit 4 test you annotate a method with the @org.junit.Test annotation.

Sample JUnit test

```
@Test
public void testAdd() {
int total = 4;
int sum = Calculation.add(value1, value2);
assertEquals(sum, total);
                    }
```

Some Annotations used in JUnit

Annotation	Description
@Test public void method()	The `@Test` annotation identifies a method as a test method.
@Test (expected = Exception.class)	Fails if the method does not throw the named exception.
@Test(timeout=100)	Fails if the method takes longer than 100 milliseconds.
@Before public void method()	This method is executed before each test. It is used to prepare the test environment (e.g., read input data, initialize the class).
@After public void method()	This method is executed after each test. It is used to cleanup the test environment (e.g., delete temporary data, restore defaults). It can also save memory by cleaning up expensive memory structures.
@BeforeClass public static void method()	This method is executed once, before the start of all tests. It is used to perform time intensive activities, for example, to connect to a database. Methods marked with this annotation need to be defined as `static` to work with JUnit.

Assert Statement

- JUnit provides static methods to test for certain conditions via the Assert class. These assert statements typically start with assert. They allow you to specify the error message, the expected and the actual result.

- An *assertion method* compares the actual value returned by a test to the expected value. It throws an AssertionException if the comparison fails

Some Assert statements in JUnit

assertTrue([message,] boolean condition)	Checks that the boolean condition is true.
assertFalse([message,] boolean condition)	Checks that the boolean condition is false.
assertEquals([message,] expected, actual)	Tests that two values are the same. Note: for arrays the reference is checked not the content of the arrays.
assertEquals([message,] expected, actual, tolerance)	Test that float or double values match. The tolerance is the number of decimals which must be the same.
assertNull([message,] object)	Checks that the object is null.
assertNotNull([message,] object)	Checks that the object is not null.
assertSame([message,] expected, actual)	Checks that both variables refer to the same object.
assertNotSame([message,] expected, actual)	Checks that both variables refer to different objects.

Tutorial