**JUNIT**

Document of Understanding

Unit Testing: Unit Testing can be described as Testing of individual classes to check whether it meet the expectation. It helps to find the bugs at early stage.

Junit:

It is an open source unit testing framework used for java developers. Unit Testing is used to identify defects early in software development cycle.

To get the features of Junit initially need to place the below mentioned dependency in pom.xml if it is a maven project.

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

</dependency>

Else need to install below mentioned jar in the project.

Junit.jar and hamcrest-core.jar

Below is the sample code to create a unit test case file.

Consider a controller containing two methods.

1. FetchProgramDetails
2. SaveProgramDetails.

Requirement is to check whether these two methods provides expected result. Generate a UnitTest Class file with the help of below mentioned steps:

In Eclipse select the controller class and right click on the class file.

Select New -> Other -> search with Junit and Select Junit Test Case file, which will create the file under src/test/java with same controller name appended with test as suffix and the methods will be generated inside the class same as methods present in controller class with suffix test at the end of each method.

To make the method work as Test Case it needs to be annotated with @Test Annotation. All the methods that are annotated with @Test will be executed once the class file is executed. In result it will display the total count of test methods executed and out of which how many are success and how many are failure.

There is also a possibility to skip the test case if not required to execute. @Ignore annotation will help the test case not to execute during execution.

If multiple test cases are present there is a possibility to skip all the test cases not to execute during project building by providing the below mentioned command.

**Command:** clean install -DskipTests=true -Dmaven.test.failure.ignore=true

Test Order in Junit 4:

@FixMethodOrder needs to be added for the class which accepts following classes as input.

1. MethodSorters.DEFAULT
2. MethodSorters.JVM
3. MethodSorters.NAME\_ASCENDING

Default: It in general uses methods hashcode to sort the order.

JVM : It utilises JVM Ordering which can be different for each run

NAME\_ASCENDING : It uses lexicographic order.

Test Order in Junit 5:

@TestMethodOrder needs to be added for the class which accepts following data as input.

1. OrderAnnotation.class : Need to mention @Order annotation for each method and mention order number like 1 or 2.
2. Alphanumeric.class : Follows alphanumeric . It is case sensitive so Capital letters comes first followed by small letter.
3. CustomOrder

Asserts :

Assert is a method useful for in determining pass or fail of a testcase. Below are various assert methods:

1. Boolean
   1. assertTrue
   2. assertFalse
2. Null
   1. assertNull
   2. assertNotNull
3. Identical
   1. assertSame
   2. assertNotSame
4. assertEquals
5. assertArrayEquals

Junit Test Suite: A Test Suite contains many numbers of tests which once executed tell the status of each test. @RunWith and @Suite annotations are used to run the suite tests.

Below is the example:

@RunWith(Suite.**class**)

@Suite.SuiteClasses({SampleControllerTest.**class**,SampleServiceTest.**class**})

Junit 4 Annotations:

@Before: The methods that are with @before will be executed before each test method.

@After: The methods that are with @After will be executed after each test method.

@BeforeClass: The methods that are with @BeforeClass will be executed once at the start of test execution. It is a static import statement .

@AfterClass: The methods that are with @AfterClass will be executed at last of the all test execution. It is a static import statement.

Below are the annotations that are changed from Junit 4 to Junit 5

|  |  |
| --- | --- |
| JUNIT 4 | JUNIT 5 |
| @Before | @BeforeEach |
| @After | @AfterEach |
| @BeforeClass | @BeforeAll |
| @AfterClass | @AfterAll |
| @Ignore | @Disabled |

If the test case is to check whether null data is passed then nullpointer exception is expected then we can give @Test(expected = NullPointerException.class). If on execution of the testcase the result is a null pointer exception then the testcase is true else it is failure.