**What is Unit testing**

* Unit testing means, test piece of small code. It is also called as unit.
* Not need to test whole application.
* Unit test cases are created by developer.

**Testing ways**

1. ***Manual testing :*** create a test cases manually & test it. More efforts and time consuming.
2. ***Automate testing :*** create test cases using Junit framework.

**There are the 3 thing in testing**

**1. Class/ model/ Service class -** Class to be tested (1 or more classes)

**2. Test case class**- This class contains test methods (1 or more classes)

**3. TestSuite -** Allows to combine multiple test cases classes

**Junit Framework:**

Java provides Junit framework/library based on Jdk, And Eclipse provide by default Junit library.

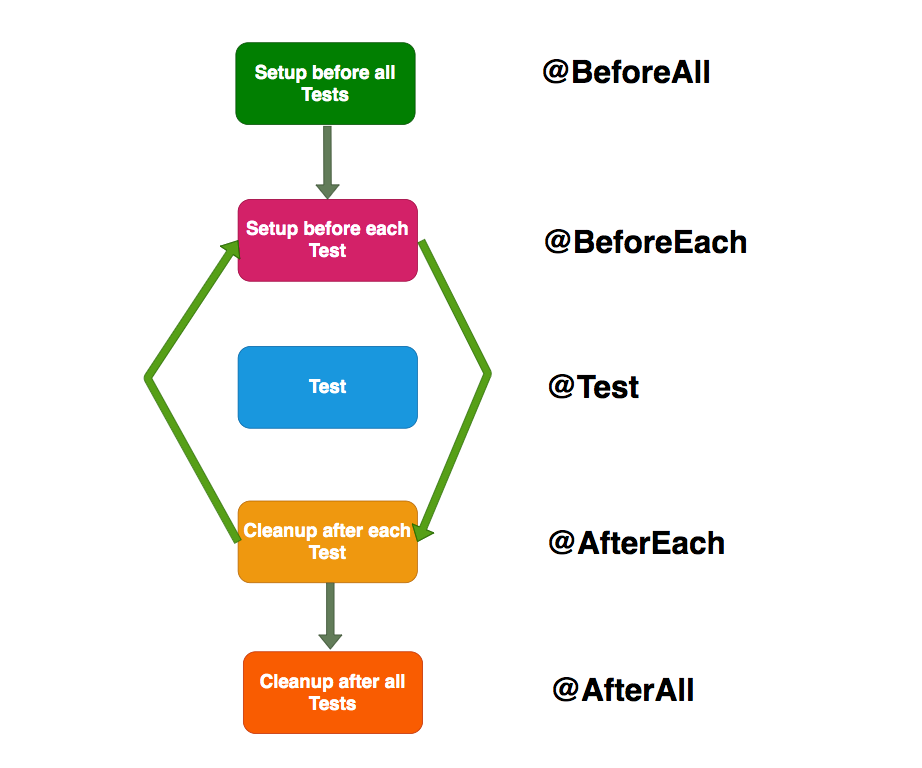
* For every java project, Junit lib by default available.
* You can create a number of test cases and test it.

**Architecture of Junit (Diagram)**

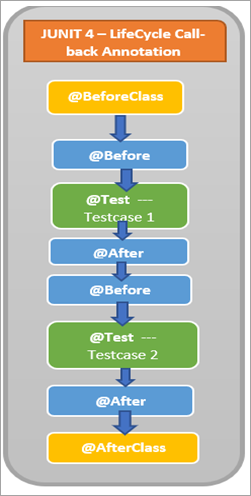
* **Junit Jupitor** - Junit 5 lib
* **Junit Vintage** - Junit 5/4 lib (for backword compability)
* **Junit Integration** - To allow Junit integration with TestNg, Mockito.

**JUnit 5 = JUnit Platform + JUnit Jupiter + JUnit Vintage**

**Life cycle of Junit 5**



**Life cycle Junit-4 Life cycle Junit-5**

* ***@BeforeClass*** / ***@BeforeAll*** is executed once at the start of the class.
* ***@Before*** / ***@BeforeEach*** executes before Testcase 1 begins.
* ***@Test*** is the Testcase in the class.
* ***@After/ @AfterEach*** runs after Testcase 1 completes execution.
* ***@Before*** / ***@BeforeEach*** executes before Testcase 2 begins.
* ***@Test*** is the Testcase in the class.
* ***@After*** / ***@AfterEach*** runs after Testcase 2 completes execution.
* ***@AfterClass*** ***/ @AfterAll*** is executed once at the end of the class after both Testcase 1 and 2 are executed.

**Annotations:**

| **JUNIT 4 ANNOTATION** | **JUNIT 5 ANNOTATION** | **Description in brief** |
| --- | --- | --- |





**Stub:**

**What is stub?**

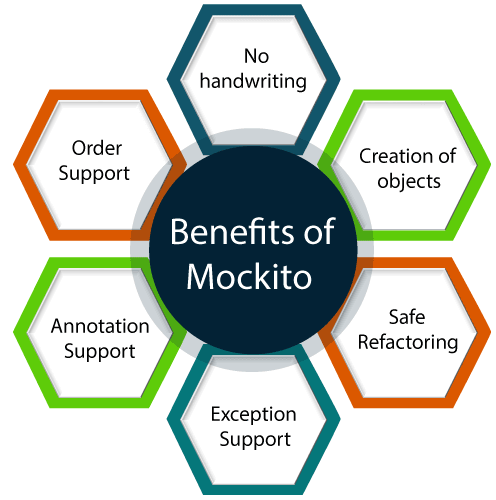
* Provide external dependencies

Mocking

* It does not required database connection, communicating to external servers or going over network to test a functionality.
* Mock objects simulate real service by taking by in actual input and returning back expected output.

Mock

* Mockito is an open source testing framework for java.
* The framework allows the creation of test doubles objects (mock objects) for unit tests. It uses Java Reflection in order to create mock objects.
* It is one of the top 10 Java libraries used by Java developers.



Benefits of Mockito

* It saves developers from writing test doubles or mock objects on their own.
* It has support for exception handling.
* It has support for stubbing methods. It helps us in returning configured response.
* It can check order of method calls and number of times method being called.
* Annotation support.

**Class Under Test** **External Dependencies**

EmailExist (CCS/ CMR)

EmailValidationCheck API

EmailService (CU team)

How to create test cases? Need to follow steps:

* @Runwith(MockitoJUnitRunner) : Use this annotation above the test class.
* @MockMvc : Create MockMvc object
* @InjectMock : Simple create EmailvalidationController Object using @InjectMock annotation. And this object inject into MockMvc object. Controller is a part of MVC.
* @Mock : Create mock objects using @Mock, Provide external dependencies so we need to create mock object for that.