# Unit Testing

1. Unit may be an individual function, method, procedure, module or object.
2. It is done during development by the developer.
3. Validate each unit in isolated manner, checking it performs as expected.
4. Junit framework for automated testing in JAVA.
5. @Test from org.junit.Test;
6. assertEquals(expected,actual); expected should equal actual. Actual is the value after passing to the method.
7. Test method should return void. It should be public.
8. assertFalse(actual).
9. assertTrue(actual,”String shown”).
10. (String,other params). String is shown when test fails.
11. @Before
12. @After
13. @AfterClass runs once after the test.
14. @BeforeClass runs only once before the test.
15. assertArrayEquals() used to check two arrays. assertEquals() on two objects will compare the references.
16. @Test(expected=NullPointerException.class) if we expect an exception test will succeed. Otherwise try catch can be used.
17. After Junit 5, use assertThrows.
18. @Test(timeout=1000) test passes if test finished within 1000ms. In Junit5, it is done using assertTimeout(Duration.ofMilis(10),()->{}).
19. @RunWith(Parameterized.class) is used to have auto parameters in the test. Then a constructor is made for private variables input and expectedOutput. These variables are auto initialized with the parameters. The static method supplies test data. That method must return a Collection of objects, each object represent a set of parameters for test. @Parameters used.
20. @RunWith(Suite.class) @SuiteClasses({}) used to run all the test classes from one class.
21. @BeforeAll must be static method.

Mockito

1. If class depends on external sources like api, services, or databases. Then Mockito is used to create mock versions of these dependencies. The sole concern should be testing the logic of the class without relying on the external dependencies.
2. Stub is a class which returns a dummy data. Create a stub and a test for the main method. Because any changes in the main class and the stubs need to change. Stub is concrete implementation and no flexibility, focuses on behavior not on integration.
3. Mocking is creating objects that simulate the behavior of real objects. Mocks can be dynamically created from code at run time.
4. Obj ob=mock(ClassName.class); stub(ob.method(“”)).return()
5. Mockito mocks return default values when not stubbed. These are called nice mocks.
6. when().thenReturn(value); is similar to stub().return() but it is used for the newer versions.
7. lenient().when().thenReturn() useful in cases when a stub is created but not directly used in the testcase. Leads to less strict test. Avoiding warning or exceptions about unnecessary stubs.
8. when().thenReturn().thenReturn() multiple times. At first, first value is returned, then the next one.
9. Inside when() anyInt() can be written. There are many other any… present. These are called ArgumentMatchers.
10. BDD style- Given, When, Then.
    1. Given: given() and willReturn(). Is same as when().then().
    2. When: .get();
    3. Then: assertThat(method(),is()). then().should().method()
11. verify that a method was called on the mock, verify().method().
12. verify(,times(1)/atleast(5),atLeastOnce()/never())
13. ArgumentCaptor<String> stringArgumentCaptor=ArgumentCaptor.forClass(String.class);
14. assertThat(stringArgumentCaptor.getValue()/.getAllValues(),is())
15. Annotations
    1. @RunWith(MockitoJUnitRunner.class)
       1. @Rule is now preferred over it.
       2. Because 1 Junit class can have 1 runner. But it can have multiple rules.
       3. public MockitoRule mockitoRule=MockitoJUnit.rule();
    2. @Mock creates a mock of an object.
    3. @InjectMocks injects a mock into a class and gives that object. Suppose there is a class which uses an object of the mock type then this is used.
    4. @Captor creates a captor.
16. A mock does not care about the implementation, business logic. mock(ArrayList.class) does not create actual ArrayList, only creates a mock.
17. Spy creates an actual ArrayList. Used as spy(ArrayList.class). It retains all the functionalities of the class, although the functionalities can be overridden. Spy is also called a partial mock. stub(arrayListSpy.size()).toReturn(5) now the size will always be 5.
18. Mockito can mock both classes and interfaces.
19. Mockito can’t mock static and final methods.

Hamcrest

1. assertThat() from this library takes much more readable arguments. Takes two arguments the list and a method.
2. List
   1. hasSize()
   2. hasItems()
3. Arrays
   1. arrayWithSize()
   2. arrayContaining()
   3. arrayContainingInAnyOrder()