01

Junit Unit Testing

WHAT WILL YOU LEARN?

- What is Unit Testing and Why?
- What is Junit
- Junit Annotations
- Junit Assertions
- Junit Architecture
- Junit Suit
- Assumptions

What is Unit testing and why?

- UNIT TESTING is a type of software testing where individuals units or component of a software
- The purpose is to validate that each unit of the software code performs as expected
- Unit testing done in development phase of an application by developer
- Unit test Is isolate section of code and verify it correctness
- A unit may be an individual function , method ,procedure, module , or object.

What is Junit?

- Junit Is a free and open source Unit Testing framework for java application
- Junit developed by Kent Beck and Erich Gamma.
- Its first version was released in 1997.
- It become one of the most popular framework in the java community ease of use.
- It is lightweight testing framework which allowed java developers to write unit test cases in java language.
- The current version is unit 5

Junit Annotations

- @Test
    - It is used to mark a method as a junit test.
- @BeforeAll (it required static method)
    - The annotated method will be run before all test methods in the test class. This method must be static.
- @AfterAll (it required static method)
    - The annotated method will be run after all test methods in the test class. This method must be static.
- @ParametrizedTest
- @
- @CSVSource
- @BeforeEach
    - The annotated method will be run before each test method in the test class.
- @AfterEach
    - The annotated method will be run after each test method in the test class.
- @DisplayName
    - Used to provide any custom display name for a test class or test method
- @Disable
    - It is used to disable or ignore a test class or test method from the test suite.
- @Nested
    - Used to create nested test classes

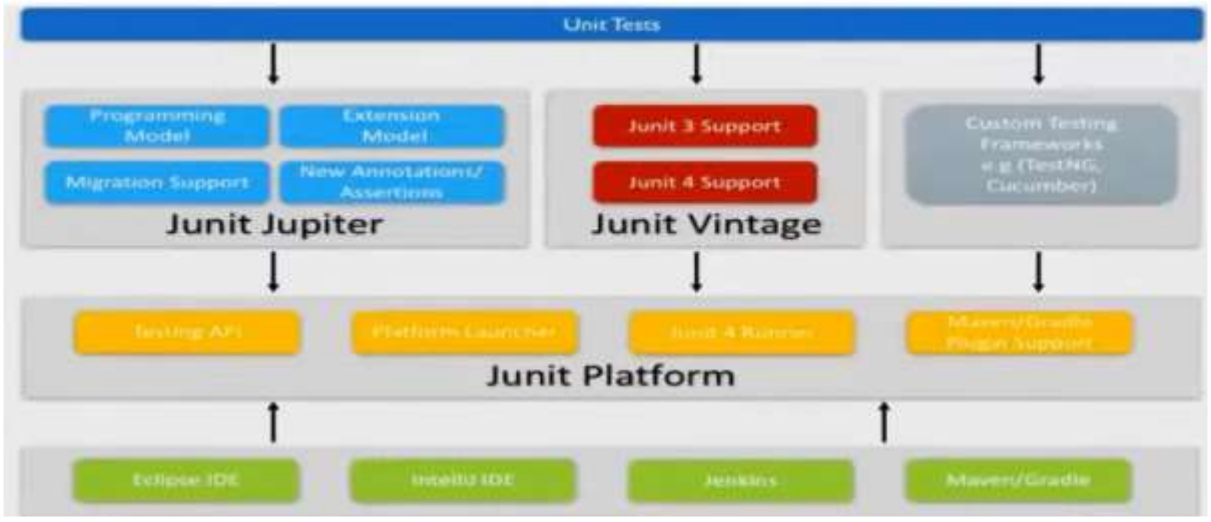
Assertions

- Assertions help in validating the expected output with the actual output of a test.
- Assertions are nothing but static methods that we call in our tests to verify expected behavior
- All Junit Jupiter assertions are present in
    - org.junit.jupiter.Assertions class

Assert Methods

- assertEqual and assertNotEquals
- assertTrue and assertFalse
- assertNull and assertNotNull
- assertSame and assertNotSame
- assertThrow

JUNIT Architecture



JUnit 5 test suites are written with @Suite annotation. Suites help us run the tests spread into multiple classes and packages.

- @Suite –
  Just add the @Suite annotation of a class and start including or excluding the test classes and methods into the suite.

- @SuiteDisplayName - Use this annotation to give a display name for the annotated test class that is executed as a test suite on the JUnit Platform.
- @SelectPackages –

  @SelectPackages specifies the names of packages to select when running a test suite via @RunWith(JUnitPlatform.class).
- @SelectClasses - @SelectClasses specifies the classes to select when running a test suite via @RunWith(JUnitPlatform.class).
- @IncludePackages and @ExcludePackages-
  If you want to exclude any specific package or include any package then you may use @IncludePackages and @ExcludePackages annotations.
- @IncludeClassNamePatterns and @ExcludeClassNamePatterns
- @IncludeTags and @ExcludeTags

Assumptions

JUnit 5 assumptions class provides static methods to support conditional test execution based on assumptions. A failed assumption results in a test being aborted.

JUnit Jupiter Assumptions class has the following methods:

- assumeFalse()
- assumeTrue()
- assumingThat()