Junit:

Testing : process where we can figure out whether our code runs fine according to the expected result. It helps us discovering any bugs or defects before delivering the project to the client.

Unit testing:

A type of testing where we test individual units or small chunks of code. It is usually done during the development phase by the developers.

Importance:

1. Helps us identify bugs at a very early stage
2. Reduces time and efforts
3. Cost effective
4. Assures accuracy of codes.

What is Junit?

It is a unit testing framework in java which is used for writing and running tests.

It provides annotations to identify the test methods.

Mandatory annotation for a test case is ***“@Test”***

Steps of execution of unit testing:

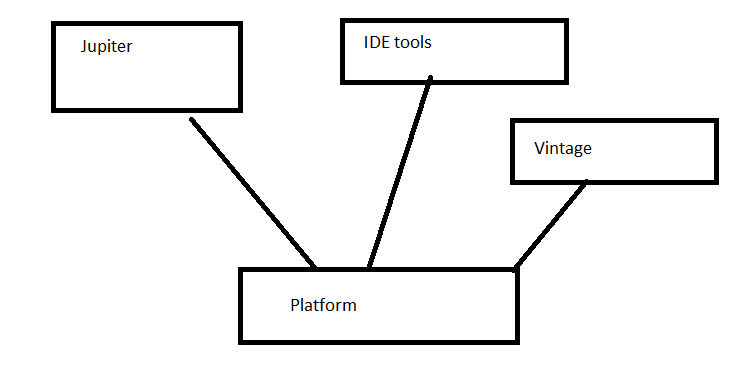
1. Preparation-- > we will write our test case
2. Provide the actual input
3. Provide the expected output
4. Run the test
5. Verify the test🡪 failure
6. If the test fail, We need to alert the developer.

We are going to use Junit-5

**Junit-4 disadvantages:**

1. More than 10 years old
2. Not compatible with new java features like lambda, functional interface, Stream API etc etc…
3. Monolithic architecture . i.e has only one jar file.

Junit API --🡪 Jupiter

Junit architecture: 

***The dependencies we need to add:***

***Junit engine, Jupiter api***

**Assertions**: it is a class. It is used to compare the expected and the actual (assertEquals(), assertAll())

SureFire plugin:

maven-surefire-plugin: is used to run the test in a maven way.

Steps🡪

Right click on the project-->Run as🡪Maven Test

Exception(assertThrows()): is used to test whether a method throws an exception as it is supposed to throw.

Junit life cycle hooks:

1. @BeforeAll: we mandatorily have to write static methods
2. @AfterAll: we mandatorily have to write static methods
3. @BeforeEach: it will run before each test cases
4. @AfterEach: it will run after each test cases

\*\*\*\*To use BeforeAll and AfterAll in ***non-static*** methods we have to use:

@TestInstance(TestInstance.Lifecycle.PER\_CLASS)

PER\_METHOD is the default value.

Some useful annotations:

1. @DisplayName(“the name”): to give a name for the test case
2. @Disabled

Conditional annotations:

1. @EnabledOnJre(JRE.***JAVA\_9***)
2. @EnabledOnOs(OS.***WINDOWS***)
3. EnabledIf
4. EnabledIfSystemProperty
5. @EnableIfEnvironmentVariable

@Nested: we an write a bunch of tests in a class and annotate is with @Nested

@RepeatedTest(int)

## @ParameterizedTest🡪 to use this annotation we need to add a dependency “[JUnit Jupiter Params](https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-params)

”

@ValueSource(ints={1,3,4,5})

@ValueSource(strings={“abc”,”mno”})