Getting Started with Junit 5:

Whereas the Junit User guide is an amazing place to refer how to write a Junit test Case, sometime a novice gets overwhelmed by all the user manuals and Javadoc. Well, for the starters, here is my cheat doc(not cheat sheet though). I will explain in brief, how to kickstart writing your Junit tests.
Well, So, if we are going to write a Junit test Case, I believe we have a test piece of software which we want to test. To start with, let us take a very simple java class (ClassToTest.java) as below:

```
public class ClassToTest {

    public int addition(int a,int b){
        return a+b;
    }

}
```

Well, the class above has a single method, addition, which simply adds two numbers and returns the result. In real world, the methods we are going to test will be far complex. Let us say, we have used Maven as our choice of Build tool. now we need to inform Maven to include Junit libraries for use. We do that by adding a few lines in pom.xml of the project. We ask maven to add the dependency as highlighted code as below:

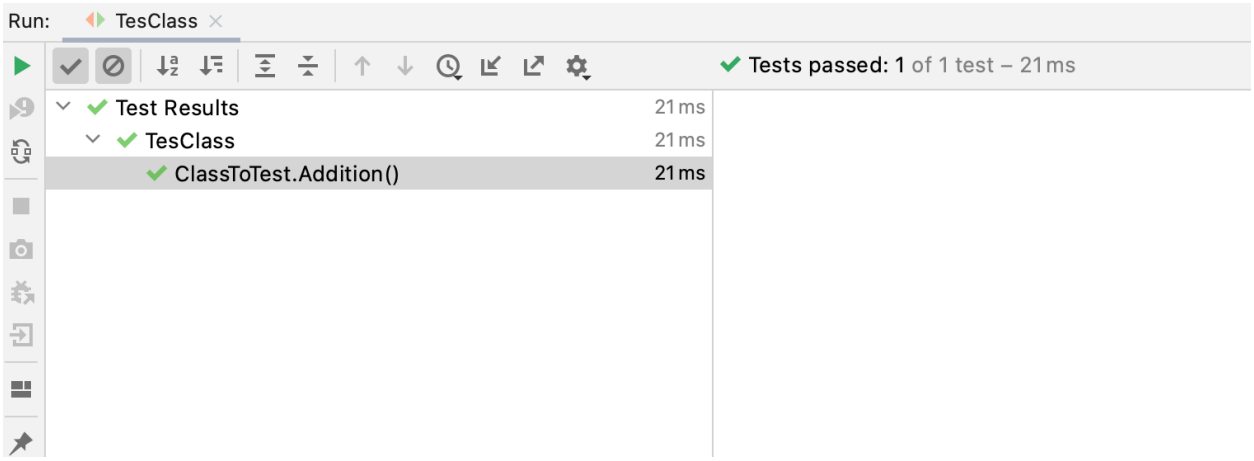
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.example</groupId>
    <artifactId>JUnitCheat</artifactId>
    <version>1.0-SNAPSHOT</version>
    <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>
    <dependencies>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.5.2</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>5.5.2</version>
        <scope>test</scope>
    </dependency>
    </dependencies>
</project>
```

Okay, halfway down. Now it's time to write tests. In the test folder of our project, we create another class TestClass.java

```java
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Order;
import org.junit.jupiter.api.Test;

public class TesClass {

    @Test
    @DisplayName("ClassToTest.Addition()")
    @Order(1)   // will gurantee the order of execution
    public void TestAddition(){
        ClassToTest cl=new ClassToTest();
        Assertions.assertEquals( expected: 5, cl.addition( a: 2, b: 3));
        Assertions.assertNotEquals( unexpected: 3, cl.addition( a: 2, b: 3));
        Assertions.assertTrue( condition: 5==cl.addition( a: 2, b: 3),() -> "true" );
        Assertions.assertFalse( condition: 3==cl.addition( a: 2, b: 3),() -> "false" );
    }

}
```

For now, we want to test if the addition method is giving us correct result. We work on two things here, the expected value and the actual value. We are validating if the expected result is same as the result returned by the method. As seen above, we have done it in four ways using assertEquals, assertNotEquals, assertTrue and assertFalse.

Easy right! Now when we run the TestClass.java we get output like below:



Find more about other methods provided by Junit here :
https://junit.org/junit5/docs/current/user-guide/#writing-tests