@Test Annotation

In this lesson, we'll look into a quick review of JUnit's 5 @Test annotation. This annotation provides a powerful tool for performing unit testing.

We'll cover the following

Class Under Test

Let's first create a class with some functionality to test. This class will have the method for which we will write our test scenarios to demonstrate the @Test annotation's usability. The class name in our example would be - `OddEven`, as shown in the code below.

Method Under Test

Inside `OddEven` class, let's create a method by name `isEvenNumber()`. This method will take in an integer value and return a boolean value that whether the given number is even number or not. If the number is even it will return true and if it is odd then it will return false. Let's have a look into the code below.

OddEven.java

```
1  package com.hubberspot.junit5;
2
3  public class OddEven {
4
5      public boolean isNumberEven(int number) {
6          return number % 2 == 0;
7      }
```

```
}
```

Testing the Method

In order to test `isEvenNumber()` method, we will write use cases that it suppose to fulfill. The test scenarios would be:-

1. Given an even number, when isEvenNumber() method is called, then it should return true.
2. Given an odd number, when isEvenNumber() method is called, then it should return false.

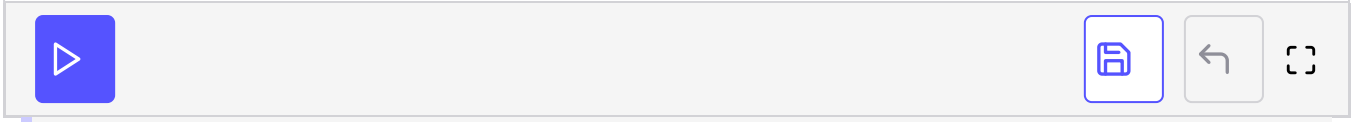
In order to test `isEvenNumber()` method, we will write a test class which will have two methods to test the above two scenarios. The test class will be created in the test folder as discussed in the previous lesson.

In order for the methods created in the test class to be recognized as test methods, we mark it with `@Test` annotation. Let's have a look at the test class and test methods:-

OddEvenTest.java

OddEven.java

```
package io.educative.junit5;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

class OddEvenTest {

	@Test
	void givenEvenNumber_whenIsEvenIsCalled_thenTrueIsReturned() {
		OddEven oddEven = new OddEven();
		assertTrue(oddEven.isNumberEven(10));
	}

	@Test
	void givenOddNumber_whenIsEvenIsCalled_thenFalseIsReturned() {
		OddEven oddEven = new OddEven();
		assertFalse(oddEven.isNumberEven(11));
	}

}
```



> You can perform code changes to above code widget, run and practice different outcomes.

Explanation

Junit 5 `@Test` annotation has following characteristics:-

1. It is applied over methods to mark them as test methods.
2. It is present in `org.junit.jupiter.api` package.
3. Its visibility can be made public, default and protected.

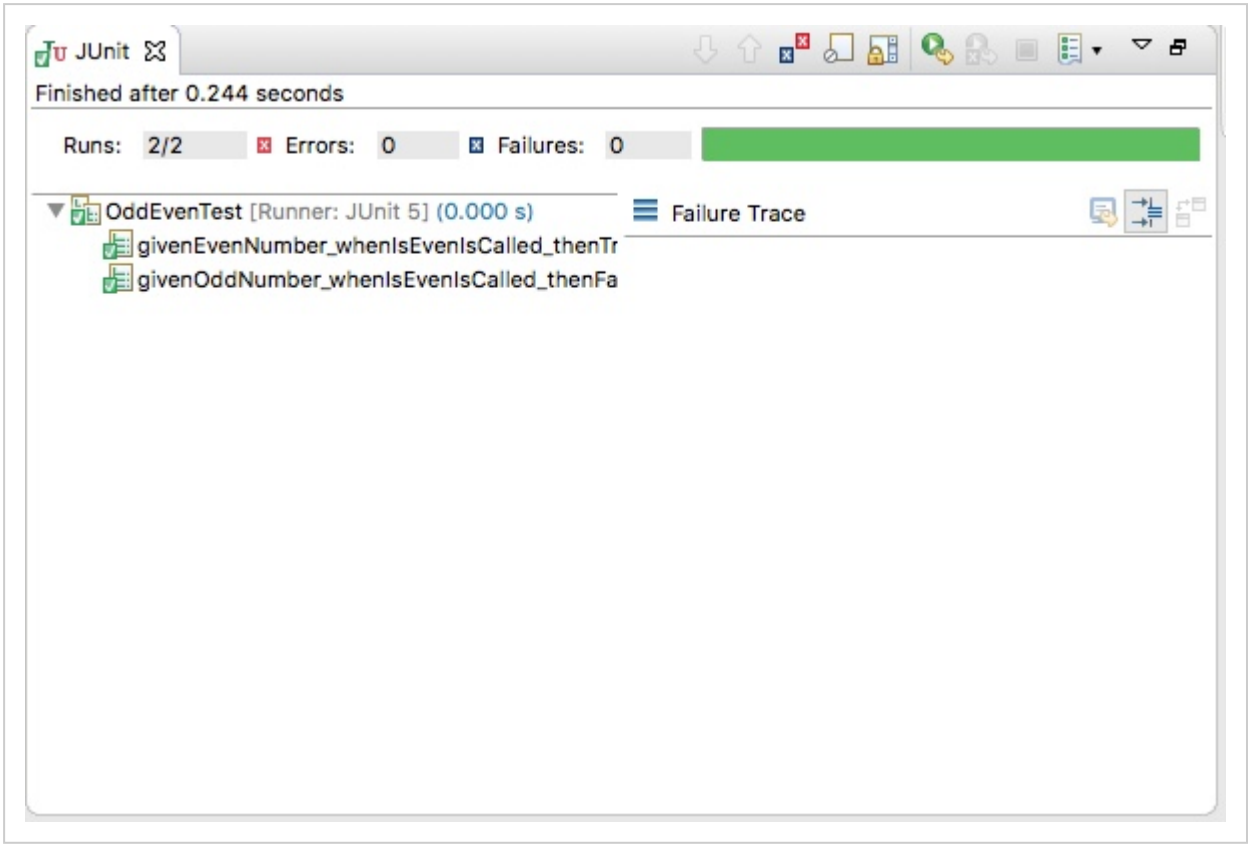
`assertTrue()` methods take in a boolean value and ensure that value is true. If a false value is passed, it will fail the test case.

`assertFalse()` methods take in a boolean value and ensure that value is false. If a true value is passed, it will fail the test case.

We will discuss more about `assertTrue()` and `assertFalse()` methods in upcoming lessons.

Running Junit 5 Test

As shown in the figure below, upon running the two test methods, it gives success that both the test cases are passed.



@Test annotation

In the next chapter, we will look into various `Assertions` JUnit 5 supports.