Parameterized Test with @ValueSource

This lesson demonstrates use of @ValueSource to pass different arguments to @ParameterizedTest.

WE'LL COVER THE FOLLOWING

- @ValueSource

@ValueSource

`@ValueSource` is one of the simplest ways to pass arguments array to `@ParameterizedTest` method. This array can be of following types -

1. short
2. byte
3. int
4. long
5. float
6. double
7. char
8. java.lang.String
9. java.lang.Class

Let's look into a demo.

Step 1 - Let's create a class `OddEven.java`, it is our class under test.

Step 2 - To this class we provide a method by name `isNumberEven()`. This method takes in an integer value and returns true if the number is even or false if the number is odd.

OddEven.java

```
package com.hubberspot.junit5.parameterized;
```

```
public class OddEven {

	public boolean isNumberEven(int number) {
		return number % 2 == 0;
	}

}
```

Step 3 - We create a test class by name, `OddEvenTest.java`.

Step 4 - It contains a test method by name, `givenANumber_whenIsEvenIsCalled_thenTrueIsReturnedForEvenNumbers`. In order to provide different parameters/values to the same test method, this method is marked as `@ParameterizedTest` instead of `@Test`. `@ParameterizedTest` annotation makes this test method eligible to take multiple values from different sources.

Step 5 - In order to provide different and multiple values through `@ValueSource` we mark this test method with `@ValueSource` annotation. This annotation takes arguments in the form of an array.

Step 6 - Let's pass integer array with different values such as 2,4,6,8, Integer.MAX_VALUE. There are 5 integer values so `@ParameterizedTest` will execute 5 times. In each iteration, it will assert one integer value to check whether it is even or not. Thus, saving a lot of time spent writing the same tests for different values again and again.

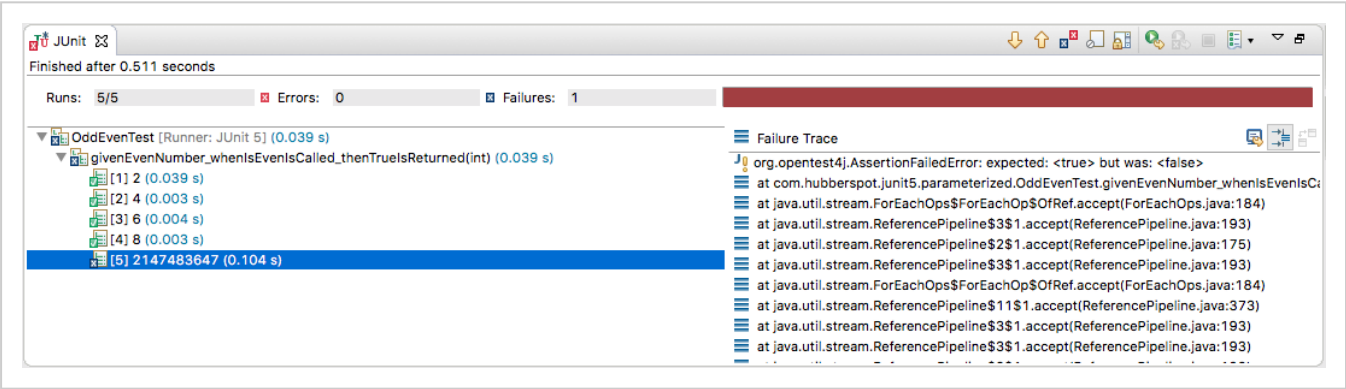
Step 7 - Run it as, JUnit Test Case.

OddEven.java

OddEvenTest.java

```
package io.educative.junit5;

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

import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.ValueSource;

class OddEvenTest {

	@ParameterizedTest
	@ValueSource(ints = {2,4,6,8,Integer.MAX_VALUE})
	void givenANumber_whenIsEvenIsCalled_thenTrueIsReturnedForEvenNumbers(int number) {
		OddEven oddEven = new OddEven();
```

```
                assertTrue(oddEven.isNumberEven(number));
        }


}
```



Output of @ParameterizedTest demo

Above image demonstrates the working of `@ParameterizedTest`. As we have provided 5 different values, the test case ran 5 times. As 2,4,6,8 are even numbered so respective test cases pass, but Integer.MAX_VALUE(2147483647) is odd thus, last test case fail.

In the next lesson we will be studying parameterized tests with `@Enumsource`