Disable Test Method and Class - @Disabled

This lesson demonstrates how to disable the test method or a complete test class.

WE'LL COVER THE FOLLOWING

- @Disabled annotation
- @Disabled annotation on Test class
- @Disabled annotation on @Test method

@Disabled annotation

There are many ways to enable or disable test methods or test class in Junit 5. `@Disabled` annotation if applied to the test class, then all the tests present in the class gets disabled and if applied on the test method, then that test method is not executed. `@Disabled` annotation also takes one optional parameter, which provides a reason for disabling test case for documentation.

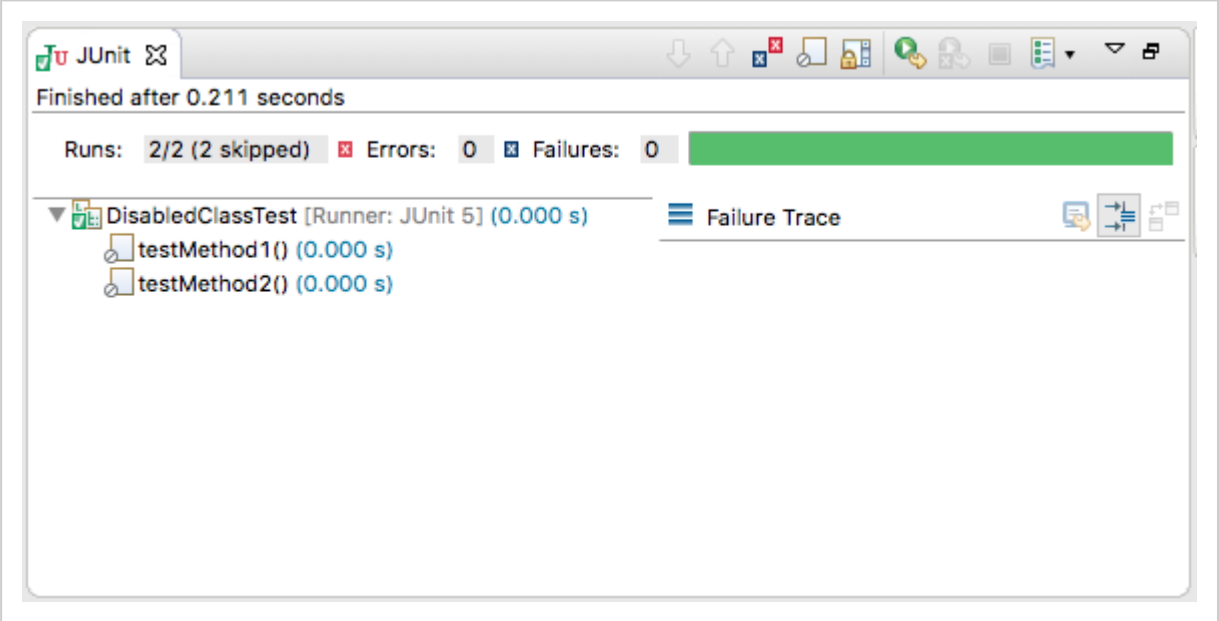
@Disabled annotation on Test class

Applying `@Disabled` annotation on test class will make test class and all test methods within that class disabled.

Let's look at a demo.

```
package io.educative.junit5;

import static org.junit.jupiter.api.Assertions.assertFalse;
import static org.junit.jupiter.api.Assertions.assertTrue;

import org.junit.jupiter.api.Disabled;
import org.junit.jupiter.api.Test;

@Disabled
public class DisabledClassTest {

        @Test
        void testMethod1() {
                assertTrue(4 > 0);
```

```
        }

        @Test

        void testMethod2() {
                assertFalse(4 < 0);
        }
}
```



Here, `@Disabled` annotation is placed on test class `DisabledClassTest`. On running above test class it shows that test case is passed, but it also shows that 2/2 (2 skipped). Thus, providing `@Disabled` annotation on test class will make all of its test methods as disabled.

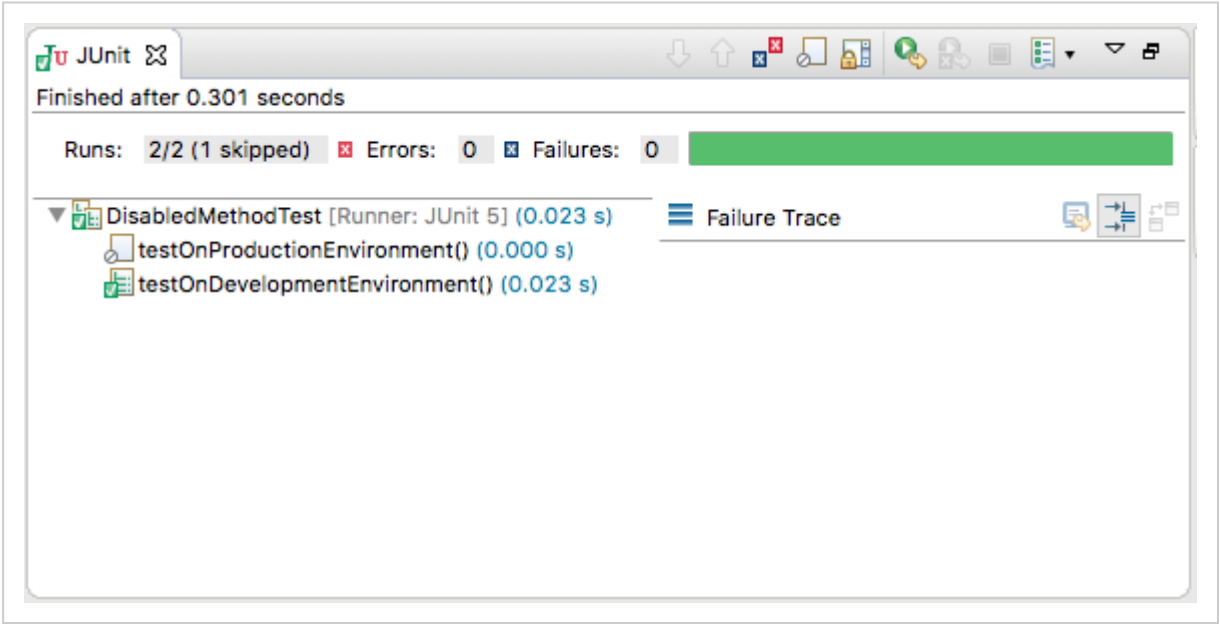
@Disabled annotation on @Test method

Applying `@Disabled` annotation on the test method will make the test method disabled. It will be skipped from execution.

Let's look at a demo.

```
package io.educative.junit5;

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

import org.junit.jupiter.api.Disabled;
import org.junit.jupiter.api.Test;

public class DisabledMethodTest {

    @Test
    void testOnDevelopmentEnvironment() {
```

```
        System.setProperty("ENV", "DEV");
        assertTrue("DEV".equals(System.getProperty("ENV")));
    }


    @Disabled("Skip in upper environments")
    @Test
    void testOnProductionEnvironment() {
        System.setProperty("ENV", "PROD");
        assertTrue("PROD".equals(System.getProperty("ENV")));
    }
}
```



Here, `@Disabled` annotation is placed on the `@Test` method `testOnDevelopmentEnvironment()`. On running above test class it shows that test cases are passed, but it also shows that 2/2 (1 skipped). Thus, providing `@Disabled` annotation on the test method will make it disabled i.e. it will be skipped from getting executed.

In the next lesson we will learn about `DisabledOnOs` and `EnableOnOs`.