What are Hooks in Cucumber?

Cucumber supports ***hooks***, which are blocks of code that run ***before*** or ***after*** each scenario. You can define them anywhere in your project or step definition layers, using the methods ***@Before*** and ***@After***. ***Cucumber Hooks*** allows us to better manage the code workflow and helps us to reduce the code redundancy. We can say that it is an unseen step, which allows us to perform our scenarios or tests.

Why Cucumber Hooks?

In the world of testing, you must have encountered the situations where you need to perform the prerequisite steps before testing any test scenario. This prerequisite can be anything from:

Starting a webdriver
Setting up DB connections
Setting up test data
Setting up browser cookies
Navigating to certain page
or anything before the test

In the same way, there are always after steps as well of the tests like:

Killing the webdriver
Closing DB connections
Clearing the test data
Clearing browser cookies
Logging out from the application
Printing reports or logs
Taking screenshots on error
or anything after the test

To handle these kinds of situations, cucumber hooks are the best choice to use. Unlike ***TestNG Annotations***, cucumber supports only two hooks (*Before & After*) which works at the *start* and the *end* of the test scenario. As the name suggests, *@before* hook gets executed well before any other *test scenario*, and *@after* hook gets executed after executing the scenario.

How to implement Hooks in Cucumber Test

Let's do some easy and small examples of Cucumber Hooks just to understand the concept. I will bring the intelligent usage of Hooks in my later tutorial series of ***Designing Framework with***

Cucumber.

Test Hooks with Single Scenario

Feature File

```
Feature: Test Hooks

Scenario: This scenario is to test hooks functionality
        Given this is the first step
        When this is the second step
        Then this is the third step
```

Step Definitions

```
package stepDefinition;

import cucumber.api.java.en.Given;
import cucumber.api.java.en.Then;
import cucumber.api.java.en.When;

public class Hooks_Steps {

        @Given("^this is the first step$")
        public void This Is The First Step(){
                System.out.println("This is the first step");
        }

        @When("^this is the second step$")
        public void This Is The Second Step(){
                System.out.println("This is the second step");
        }

        @Then("^this is the third step$")
        public void This Is The Third Step(){
                System.out.println("This is the third step");
        }

}
```

Note: *There is no logic used in the step definitions. Just printing the step summary log.*

Hooks

```
package utilities;
import cucumber.api.java.After;
import cucumber.api.java.Before;
```

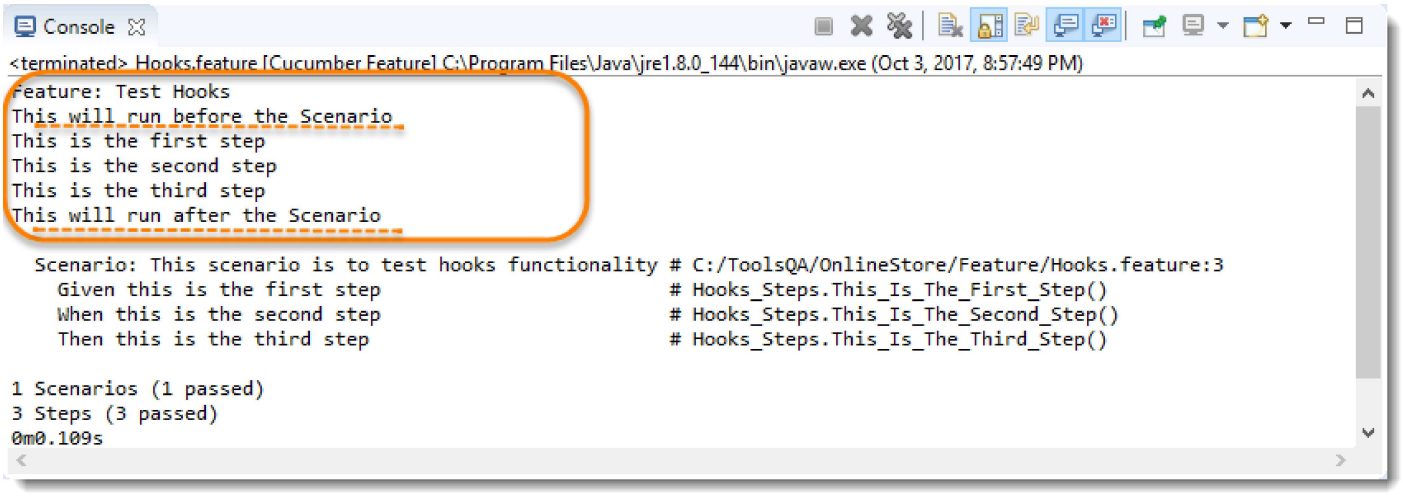
```
public class Hooks {

        @Before
    public void beforeScenario(){
        System.out.println("This will run before the Scenario");
    }

        @After
    public void afterScenario(){
        System.out.println("This will run after the Scenario");
    }
}
```

Things to note

An important thing to note about the after hook is that even in case of test fail, after hook will execute for sure.
Method name can be anything, need not to be beforeScenario() or afterScenario(). can also be named as setUp() and tearDown().
*Make sure that the package import statement should be ***import cucumber.api.java.After;*** ***& import cucumber.api.java.Before;***

Often people mistaken and import Junit Annotations, so be careful with this.

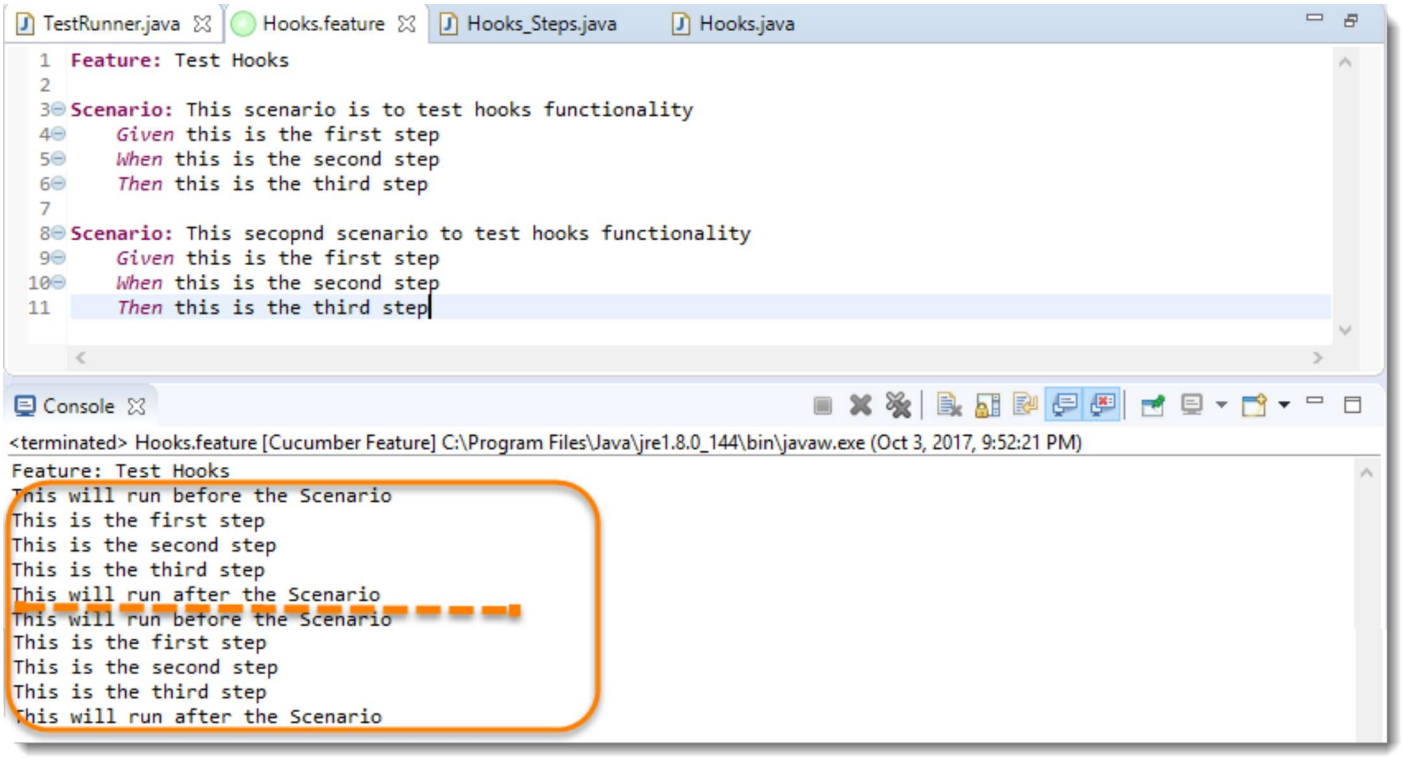
Output



No need for explanation, it is self-explanatory :)

Test Hooks with Multiple Scenarios

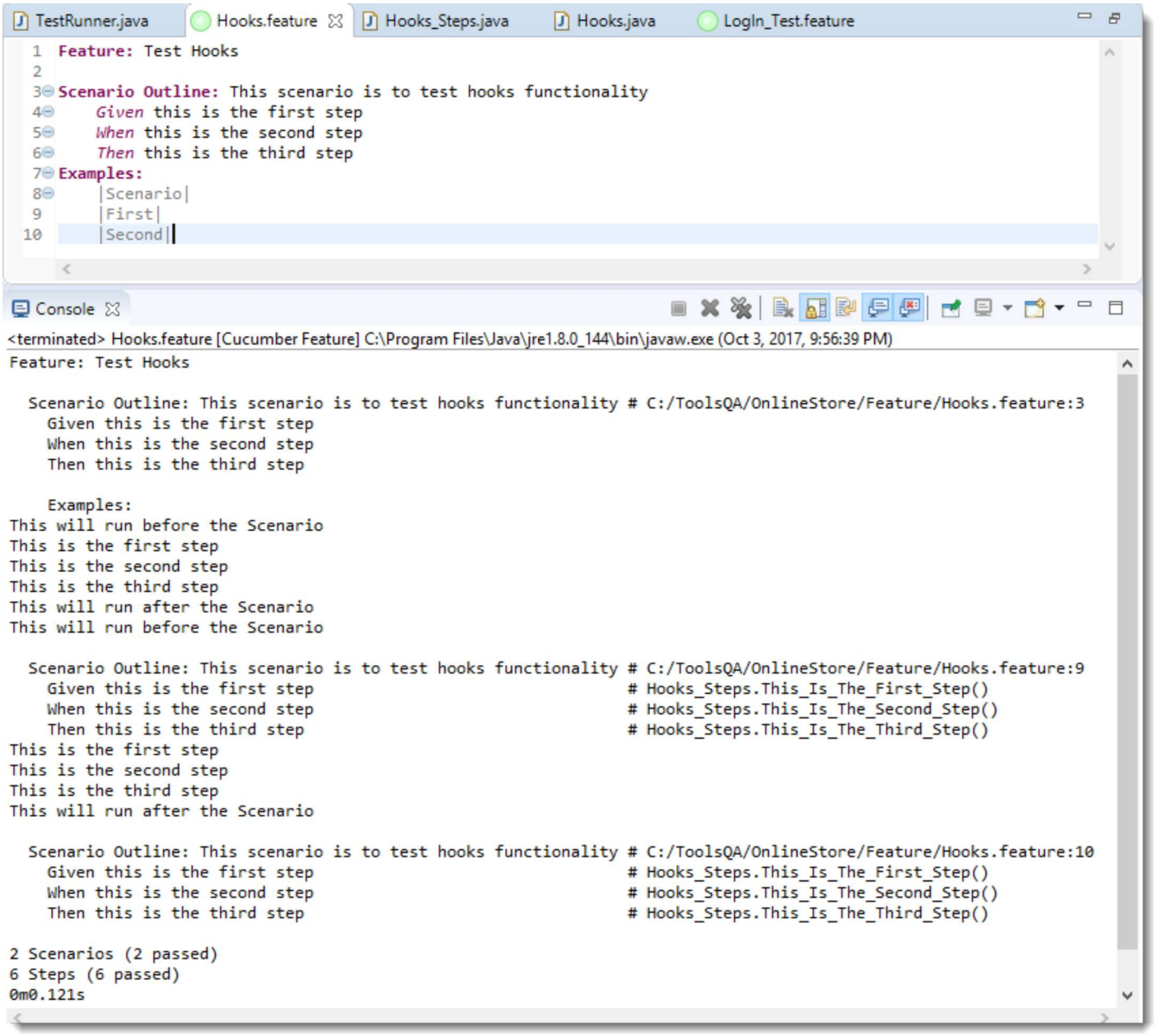
I just wanted to show you the reaction of Hooks with the multiple scenarios. Let's just add one more Test Scenario in the feature file and run the feature again.



Note: *Scenario Hooks execute before and after every scenario. In the above example, executed two times for two scenarios.*

Test Hooks with Example Scenarios

Lets take a look when we have Scenario Outline with Examples.



Note: *Again, in cucumber, every example is considered as a separate scenario. So the output is the same as the second example above.*