BDD

Lesson – 5 : Cucumber

Lesson Objectives

In this lesson, you will learn:

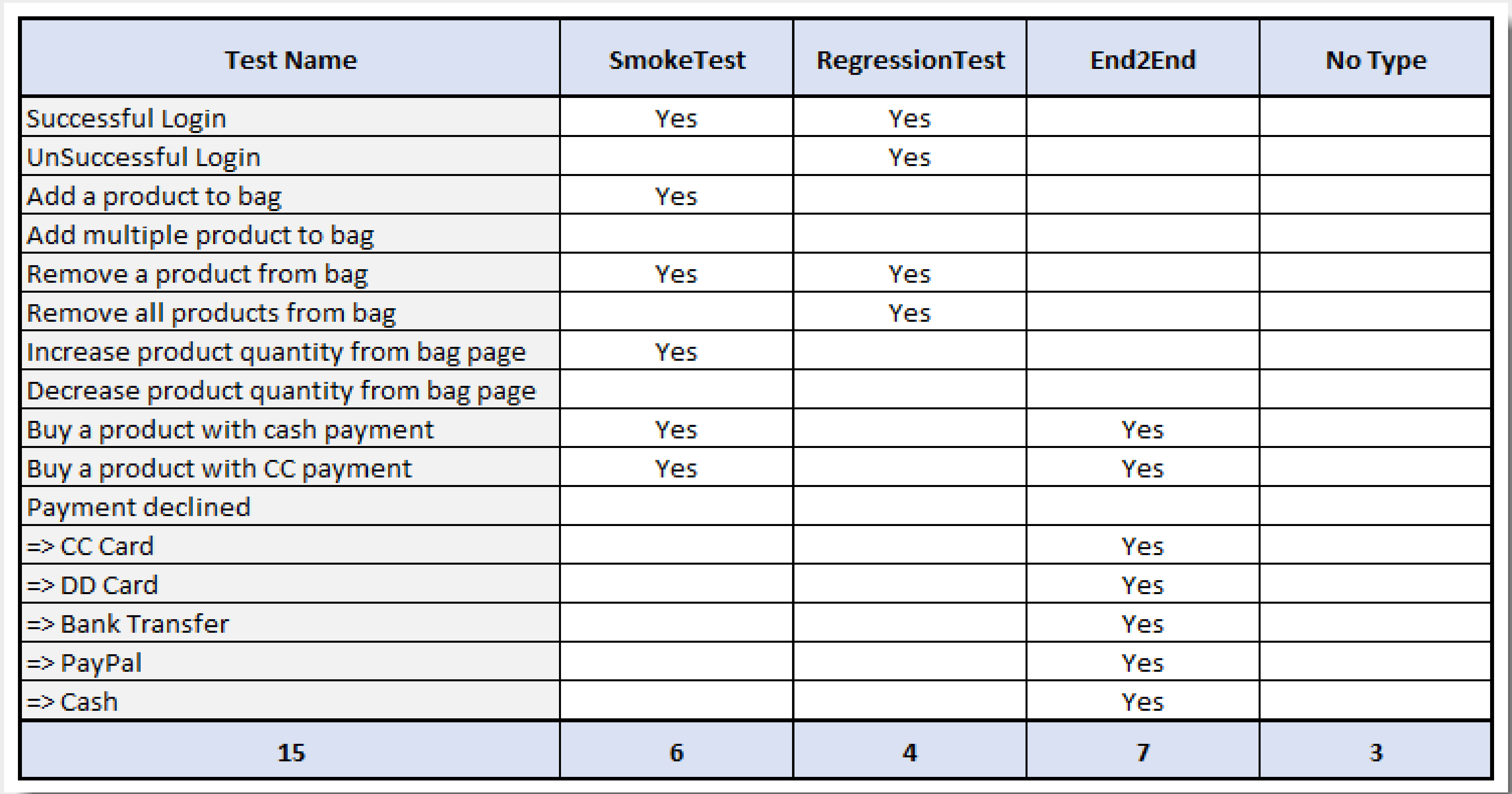
- Cucumber Tags
- Cucumber Hooks
- Background in Cucumber

Cucumber Tags

- *We can define each scenario with a useful tag.*
- *In the runner file, we can decide which specific tag (and so as the scenario(s)) we want Cucumber to execute.*
- *Tag starts with "**@**". After "@" you can have any relevant text to define your tag like **@SmokeTests** just above the scenarios you like to mark.*
- *Then to target these tagged scenarios just specify the tags names in the **CucumberOptions** as **tags = {"@SmokeTests"}.***
- *Tagging not just specifically works with Scenarios, it also works with **Features**.*
- *Means you can also tag your features files.*
- ***Any tag that exists on a Feature will be inherited by Scenario, Scenario Outline or Examples.***

Cucumber Tags

- Let’s understand this with an example.
- Below is a excel sheet containing a list of scenarios of a single feature



Test Name	SmokeTest	RegressionTest	End2End	No Type
Successful Login	Yes	Yes		
UnSuccessful Login		Yes		
Add a product to bag	Yes			
Add multiple product to bag				
Remove a product from bag	Yes	Yes		
Remove all products from bag		Yes		
Increase product quantity from bag page	Yes			
Decrease product quantity from bag page				
Buy a product with cash payment	Yes		Yes	
Buy a product with CC payment	Yes		Yes	
Payment declined				
=> CC Card			Yes	
=> DD Card			Yes	
=> Bank Transfer			Yes	
=> PayPal			Yes	
=> Cash			Yes	
15	**6**	**4**	**7**	**3**

Cucumber Tags

In Excel file

- *Few scenarios are part of Smoke Test, Regression Test and End2End Test.*
- *Few scenarios are part of two or more Test Types. For example the first test is considered as Smoke as well as Regression.*
- *Few scenarios are not at all tagged*
- *Last scenario of Payment Declined, it is a single scenario but has five different test data. So this will be considered as five different scenarios.*

Cucumber Tags
Feature File

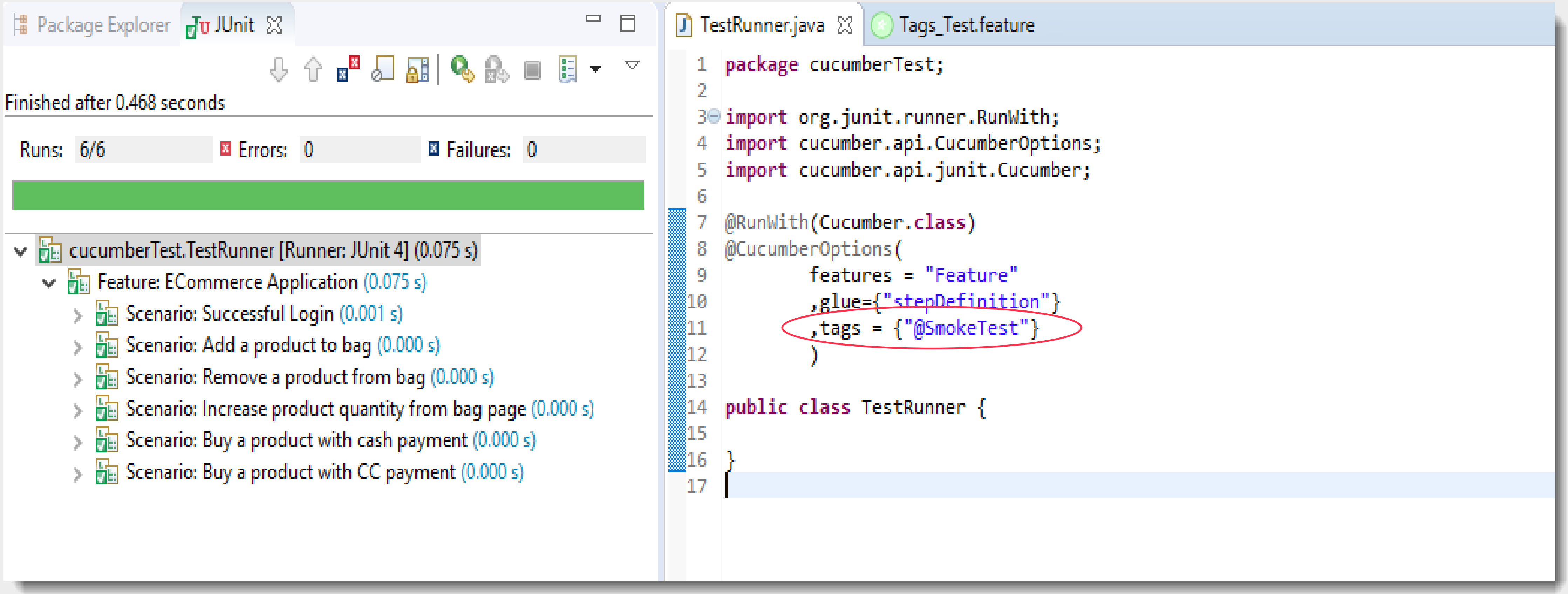
```
@FunctionalTest
Feature: ECommerce Application

@SmokeTest @RegressionTest
Scenario: Successful Login
Given This is a blank test

@RegressionTest
Scenario: UnSuccessful Login
Given This is a blank test

@SmokeTest
Scenario: Add a product to bag
Given This is a blank test

Scenario: Add multiple product to bag
Given This is a blank test

@SmokeTest @RegressionTest
Scenario: Remove a product from bag
Given This is a blank test

@RegressionTest
Scenario: Remove all products from bag
Given This is a blank test

@SmokeTest
Scenario: Increase product quantity from bag page
Given This is a blank test
```

Cucumber Tags
Feature File

```
Scenario: Decrease product quantity from bag page
Given This is a blank test

@SmokeTest @End2End
Scenario: Buy a product with cash payment
Given This is a blank test

@SmokeTest @End2End
Scenario: Buy a product with CC payment
Given This is a blank test

@End2End
Scenario Outline: Payment declined
Given This is a blank test
Examples:
|PaymentMethod|
|CC Card|
|DD Card|
|Bank Transfer|
|PayPal|
|Cash|
```

Cucumber Tags
Feature File

Running single Cucumber Feature file or single Cucumber Tag
Execute all tests tagged as @SmokeTests

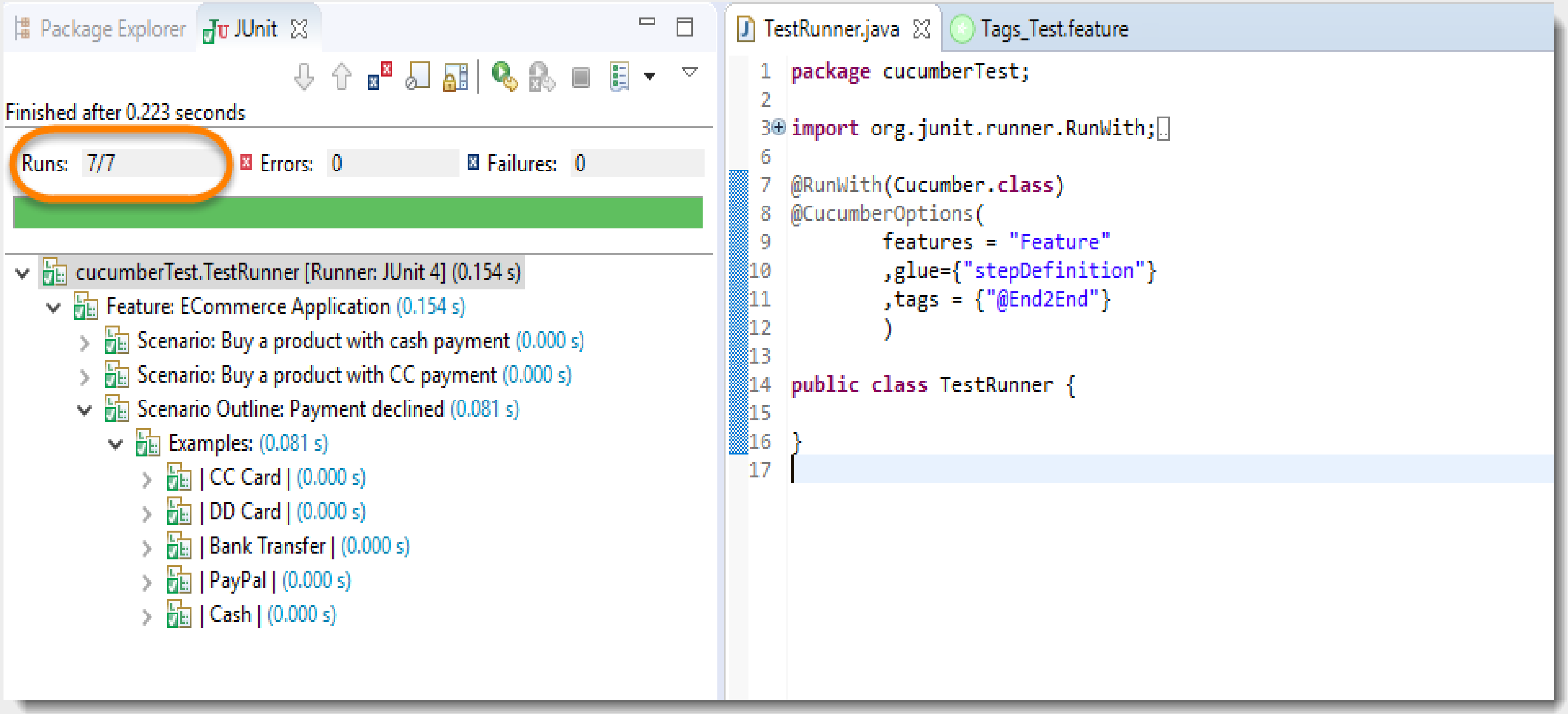


```
Finished after 0.468 seconds

Runs: 6/6    Errors: 0    Failures: 0

cucumberTest.TestRunner [Runner: JUnit 4] (0.075 s)
  Feature: ECommerce Application (0.075 s)
    Scenario: Successful Login (0.001 s)
    Scenario: Add a product to bag (0.000 s)
    Scenario: Remove a product from bag (0.000 s)
    Scenario: Increase product quantity from bag page (0.000 s)
    Scenario: Buy a product with cash payment (0.000 s)
    Scenario: Buy a product with CC payment (0.000 s)
```

```java
package cucumberTest;

import org.junit.runner.RunWith;
import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)
@CucumberOptions(
        features = "Feature"
        ,glue={"stepDefinition"}
        ,tags = {"@SmokeTest"}
        )

public class TestRunner {

}
```

Cucumber Tags
Feature File

Execute all tests tagged as @End2End

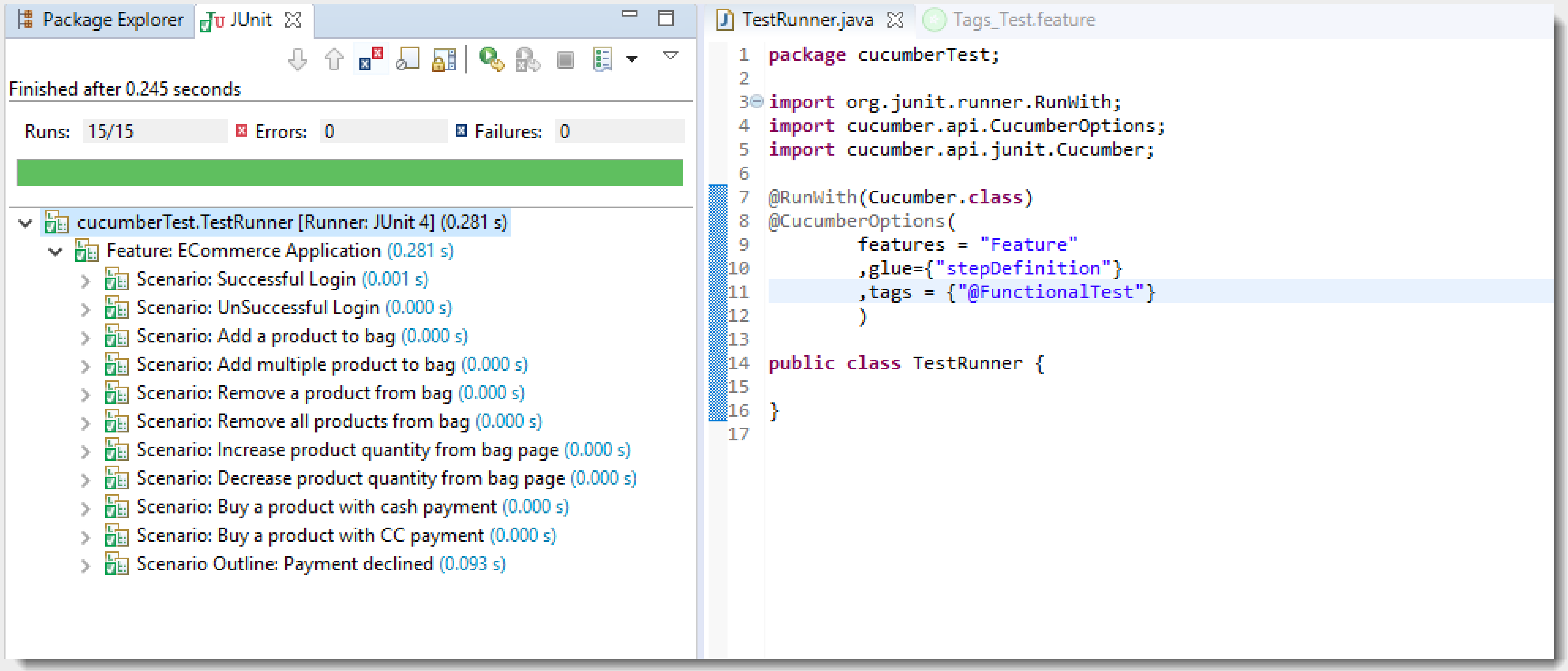

Package Explorer | JUnit | TestRunner.java | Tags_Test.feature

```
Finished after 0.223 seconds
Runs: 7/7    Errors: 0    Failures: 0

cucumberTest.TestRunner [Runner: JUnit 4] (0.154 s)
  Feature: ECommerce Application (0.154 s)
    Scenario: Buy a product with cash payment (0.000 s)
    Scenario: Buy a product with CC payment (0.000 s)
    Scenario Outline: Payment declined (0.081 s)
      Examples: (0.081 s)
        | CC Card | (0.000 s)
        | DD Card | (0.000 s)
        | Bank Transfer | (0.000 s)
        | PayPal | (0.000 s)
        | Cash | (0.000 s)
```

```java
package cucumberTest;

import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(
        features = "Feature"
        ,glue={"stepDefinition"}
        ,tags = {"@End2End"}
        )

public class TestRunner {

}
```

Cucumber Tags
Feature File

- ***Execute all tests of a Feature tagged as @FunctionalTest : Feature Tagging***
- Not only tags work with Scenario, tags work with Feature Files as well.
- Feature files pasted above is also tagged as ***@FunctionTests***.
- Let's just see how to executes all the tests in this feature.


Package Explorer | JUnit | TestRunner.java | Tags_Test.feature

```
Finished after 0.245 seconds

Runs: 15/15    Errors: 0    Failures: 0

cucumberTest.TestRunner [Runner: JUnit 4] (0.281 s)
  Feature: ECommerce Application (0.281 s)
    Scenario: Successful Login (0.001 s)
    Scenario: UnSuccessful Login (0.000 s)
    Scenario: Add a product to bag (0.000 s)
    Scenario: Add multiple product to bag (0.000 s)
    Scenario: Remove a product from bag (0.000 s)
    Scenario: Remove all products from bag (0.000 s)
    Scenario: Increase product quantity from bag page (0.000 s)
    Scenario: Decrease product quantity from bag page (0.000 s)
    Scenario: Buy a product with cash payment (0.000 s)
    Scenario: Buy a product with CC payment (0.000 s)
    Scenario Outline: Payment declined (0.093 s)
```

```java
package cucumberTest;

import org.junit.runner.RunWith;
import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)
@CucumberOptions(
        features = "Feature"
        ,glue={"stepDefinition"}
        ,tags = {"@FunctionalTest"}
        )

public class TestRunner {

}
```

Cucumber Tags
Feature File

Logically ANDing and ORing Tags

Execute all tests tagged as @SmokeTest OR @RegressionTest

Tags which are ***comma*** separated are ORed.

Example : tags = “@SmokeTest, @RegressionTest”

Execute all tests tagged as @SmokeTest AND @RegressionTest

Tags which are passed in separate ***quotes*** are ANDed

Example : tags = “@SmokeTest” , ”@RegressionTest”

Ignoring Cucumber Tests

- This is again a good feature of Cucumber Tags that you can even skip tests in the group execution.
- Special Character **~** is used to skip the tags. This also works both for *Scenarios* and *Features*.
- And this can also works in conjunction with AND or OR.
- Example :tags = “@SmokeTest” , ”~@RegressionTest”

  Will execute all tests of the feature tagged as @FunctionalTests but skip scenarios tagged as @SmokeTest

Cucumber Hooks

- 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.
- These can be used to perform the prerequisite steps before testing any test scenario.
- In the same way there are always after steps as well of the tests

Cucumber Hooks

Test Hooks with Single Scenario

Step Definitions

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
```

```
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");
    }
}
```

Cucumber Hooks

Test Hooks with Single Scenario

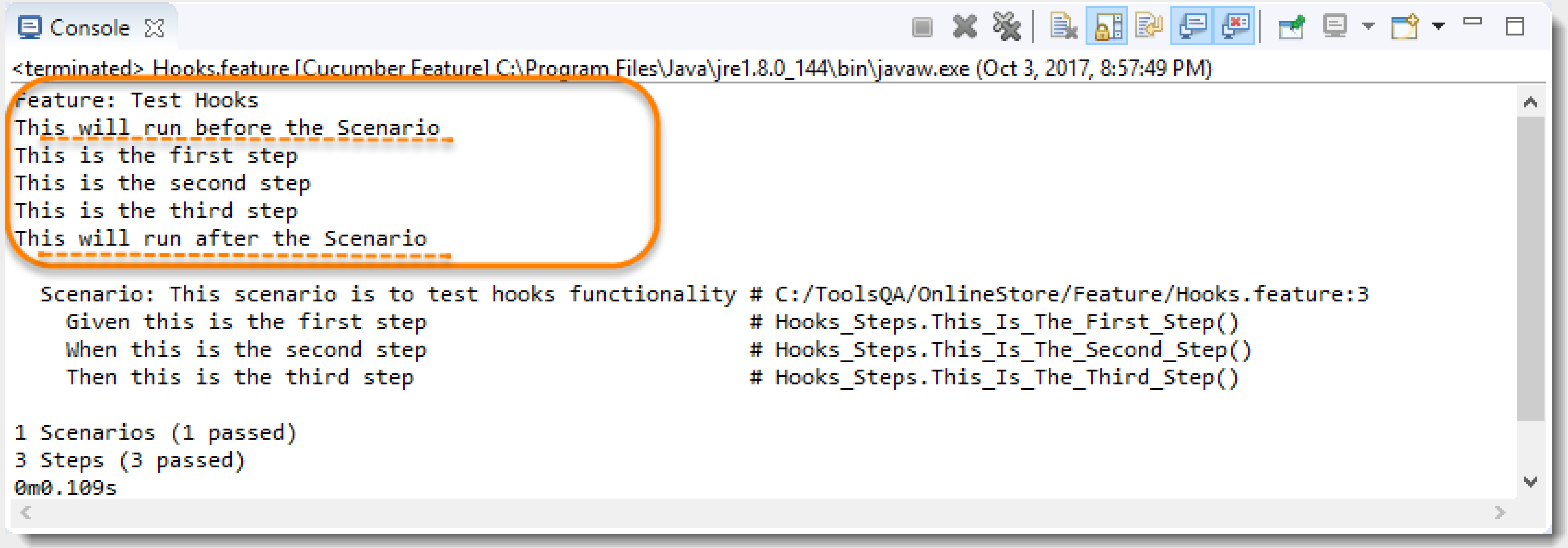
Hooks

```
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");
    }
}
```

Cucumber Hooks

Test Hooks with Single Scenario

Output



```
<terminated> Hooks.feature [Cucumber Feature] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (Oct 3, 2017, 8:57:49 PM)
Feature: Test Hooks
This will run before the Scenario
This is the first step
This is the second step
This is the third step
This will run after the Scenario

  Scenario: This scenario is to test hooks functionality # C:/ToolsQA/OnlineStore/Feature/Hooks.feature:3
    Given this is the first step                         # Hooks_Steps.This_Is_The_First_Step()
    When this is the second step                         # Hooks_Steps.This_Is_The_Second_Step()
    Then this is the third step                          # Hooks_Steps.This_Is_The_Third_Step()

1 Scenarios (1 passed)
3 Steps (3 passed)
0m0.109s
```

Background in Cucumber

- ***Background in Cucumber*** is used to define a step or series of steps which are common to all the tests in the feature file.
- It allows you to add some context to the scenarios for a feature where it is defined.
- A Background is much like a scenario containing a number of steps. But it runs before each and every scenario where for a feature in which it is defined.

- *For example to purchase a product on any E-Commerce website, you need to do following steps:*
  - *Navigate to Login Page*
  - *Submit UserName and Password*

After these steps only you will be able to add a product to your *cart/basket* and able to perform the payment. Now as we are in a feature file where we will be testing only the *Add to Cart* or *Add to Bag* functionality, these tests become common for all tests.

So instead of writing them again and again for all tests we can move it under the *Background* keyword.

Background in Cucumber

- If we create a feature file of the scenario we explained above, this is how it will look like:
- ***Feature File***

```
Feature: Test Background Feature
Description: The purpose of this feature is to test the Background keyword

Background: User is Logged In
Given I navigate to the login page
When I submit username and password
Then I should be logged in

Scenario: Search a product and add the first product to the User basket
Given User search for Lenovo Laptop
When Add the first laptop that appears in the search result to the basket
Then User basket should display with added item

Scenario: Navigate to a product and add the same to the User basket
Given User navigate for Lenovo Laptop
When Add the laptop to the basket
Then User basket should display with added item
```

Background in Cucumber

- In the this example, we have two different scenarios where user is adding a product from search and directly from product page.
- But the common step is to log In to website for both the scenario.
- *This is why we creates another Scenario for Log In but named it as Background rather then a Scenario.* So that it executes for both the Scenarios***Feature File***

Background in Cucumber

Step Definitions

```
public class BackGround_Steps {
@Given("^I navigate to the login page$")
public void i_navigate_to_the_login_page() throws Throwable {
System.out.println("I am at the LogIn Page");
}
@When("^I submit username and password$")
public void i_submit_username_and_password() throws Throwable {
System.out.println("I Submit my Username and Password");
}
@Then("^I should be logged in$")
public void i_should_be_logged_in() throws Throwable {
System.out.println("I am logged on to the website");
}
@Given("^User search for Lenovo Laptop$")
public void user_searched_for_Lenovo_Laptop() throws Throwable {
System.out.println("User searched for Lenovo Laptop");
}
@When("^Add the first laptop that appears in the search result to the basket$")
public void add_the_first_laptop_that_appears_in_the_search_result_to_the_basket() throws Throwable {
System.out.println("First search result added to bag");
}
@Then("^User basket should display with added item$")
public void user_basket_should_display_with_item() throws Throwable {
System.out.println("Bag is now contains the added product");
}
@Given("^User navigate for Lenovo Laptop$")
public void user_navigate_for_Lenovo_Laptop() throws Throwable {
System.out.println("User navigated for Lenovo Laptop");
}
@When("^Add the laptop to the basket$")
public void add_the_laptop_to_the_basket() throws Throwable {
System.out.println("Laptop added to the basket");
}
}
```

Background in Cucumber

Output

```
Feature: Test Background Feature
  Description: The purpose of this feature is to test the Background keyword

I am at the LogIn Page
I Submit my Username and Password
I am logged on to the website
User searched for Lenovo Laptop
First search result added to bag
Bag is now contains the added product

I am at the LogIn Page
I Submit my Username and Password
I am logged on to the website
User navigated for Lenovo Laptop
Laptop added to the basket
Bag is now contains the added product
```

The background ran two times in the feature before each scenario.

Background in Cucumber

Background with Hooks

- This is so interesting to see the working of *Background with Hooks*. *The background is run before each of your scenarios but after any of your* ***@Before hook***.
- To get it straight, let's assign a task to the *Before & After Hook*in the same test.
- *@Before: Print the starting logs*
- *@Before: Start browser and Clear the cookies*
- *@After: Close the browser*
- *@After: Print the closing logs*

Background in Cucumber

Hooks File

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

public class Hooks {
@Before(order=1)
    public void beforeScenario(){
        System.out.println("Start the browser and Clear the cookies");
    }
@Before(order=0)
    public void beforeScenarioStart(){
        System.out.println("-----------------Start of Scenario-----------------");
    }
@After(order=0)
    public void afterScenarioFinish(){
        System.out.println("-----------------End of Scenario-----------------");
    }
@After(order=1)
    public void afterScenario(){
        System.out.println("Log out the user and close the browser");
    }
}
```

Background in Cucumber

Output

```
Feature: Test Background Feature
  Description: The purpose of this feature is to test the Background keyword
  -----------------Start of Scenario-----------------
Start the browser and Clear the cookies
I am at the LogIn Page
I Submit my Username and Password
I am logged on to the website
User searched for Lenovo Laptop
First search result added to bag
Bag is now contains the added product
Log out the user and close the browser
-----------------End of Scenario-----------------
-----------------Start of Scenario-----------------
Start the browser and Clear the cookies
I am at the LogIn Page
I Submit my Username and Password
I am logged on to the website
User navigated for Lenovo Laptop
Laptop added to the basket
Bag is now contains the added product
Log out the user and close the browser
-----------------End of Scenario-----------------
```

Summary

In this lesson, you have learnt :

- Cucumber Tags
- Cucumber Hooks
- Background in Cucumber

Review Question

Question 1: Which of the below is used as a hook in Cucumber?

a. When
b. Then
c. After
d. Result