Now that we have defined the test its time to run our test. But before we do that we have to add a class for running our tests. *Cucumber* uses ***Junit framework*** to run. If you are not familiar with *JUnit* read our tutorials ***here***. As *Cucumber* uses *Junit* we need to have a ***Test Runner class***. This class will use the *Junit annotation* **@RunWith()**, which tells *JUnit* what is the *test runner class*. It more like a starting point for Junit to start executing your tests. In the src folder create a class called ***TestRunner***.

## JUnit Test Runner Class

Create a new ***Class*** file in the '*cucumberTest*' package and name it as '*TestRunner*', by right click on the *Package* and select ***New > Class***. This class just needs *annotations* to understand that *cucumber features* would be run through it and you can specify feature files to be picked up plus the steps package location. There are bunch of other parameters that it can take, to be discussed later in ***Cucumber Options***.

***Test Runner Class****

```
package cucumberTest;

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

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

public class TestRunner {

}
```

For curious minds, I will explain this code. Note that it is covered in detail in coming tutorials. Consider this as a limited description.

### *Import Statements*

First import statement '*org.junit.runner.RunWith*' imports *@RunWith annotation* from the Junit class. *@RunWith annotation* tells *JUnit* that tests should run using ***Cucumber class*** present in

'*Cucumber.api.junit*' package.

Second import statement '*cucumber.api.CucumberOptions*' imports the ***@CucumberOptions*** annotation. This annotation tells Cucumber a lot of things like where to look for feature files, what reporting system to use and some other things also. But as of now in the above test, we have just told it for the Feature file folder.

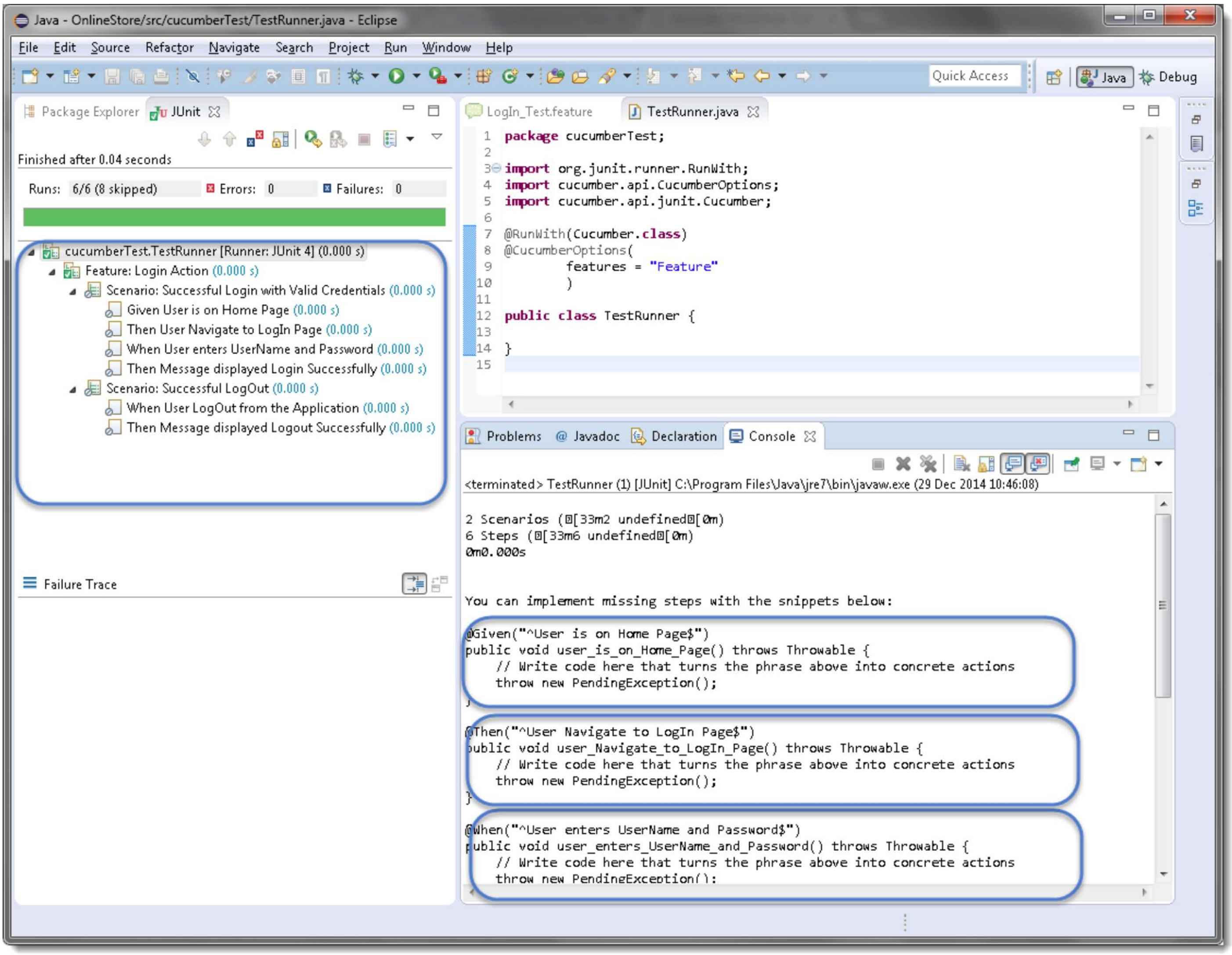
## *Run the Cucumber Test*

Now we are all set to run the first Cucumber test. There are multiple ways and runners to use when it comes to cucumber feature files. We would try to understand how to run it from the IDE first and then from a command line at a later point.

Even from the IDE, there are a couple of ways to run these feature files.

*Click on the **Run** button on eclipse and you have your test run*
*Right Click on **TestRunner** class and Click **Run As  > JUnit Test Application***

You will think where is the java code that will execute for these tests? Well, don't worry about that at this moment. Let's just see what we have on the console window. Here is the text that I got on my console. Look how Cucumber has suggested that you should implement these methods so that the Steps mentioned in the Feature file can be traced to Java methods, which can be executed while executing the feature file.



Java - OnlineStore/src/cucumberTest/TestRunner.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Explorer
JUnit
LogIn_Test.feature
TestRunner.java
Failure Trace
Problems
Javadoc
Declaration
Console

Finished after 0.04 seconds

Runs: 6/6 (8 skipped) Errors: 0 Failures: 0

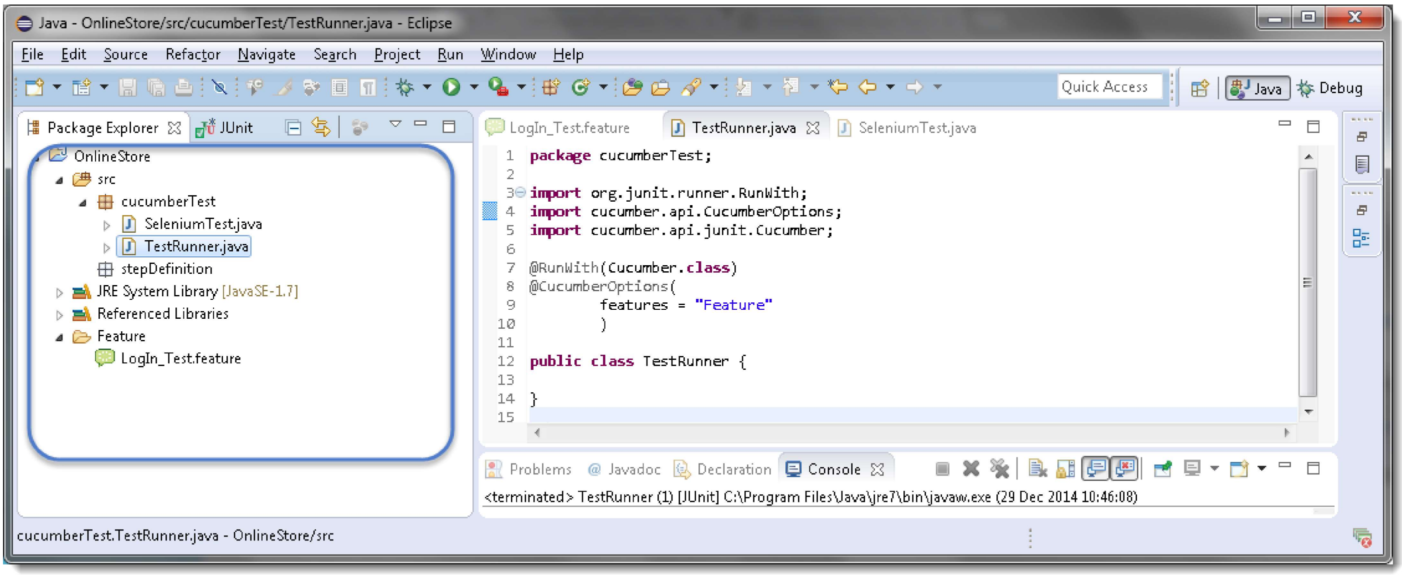
- cucumberTest.TestRunner [Runner: JUnit 4] (0.000 s)
  - Feature: Login Action (0.000 s)
    - Scenario: Successful Login with Valid Credentials (0.000 s)
      - Given User is on Home Page (0.000 s)
      - Then User Navigate to LogIn Page (0.000 s)
      - When User enters UserName and Password (0.000 s)
      - Then Message displayed Login Successfully (0.000 s)
    - Scenario: Successful LogOut (0.000 s)
      - When User LogOut from the Application (0.000 s)
      - Then Message displayed Logout Successfully (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"
        )

public class TestRunner {

}
```

<terminated> TestRunner (1) [JUnit] C:\Program Files\Java\jre7\bin\javaw.exe (29 Dec 2014 10:46:08)

```
2 Scenarios (□[33m2 undefined□[0m)
6 Steps (□[33m6 undefined□[0m)
0m0.000s


You can implement missing steps with the snippets below:

@Given("^User is on Home Page$")
public void user_is_on_Home_Page() throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@Then("^User Navigate to LogIn Page$")
public void user_Navigate_to_LogIn_Page() throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@When("^User enters UserName and Password$")
public void user_enters_UserName_and_Password() throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
```

Now your project should look like this in Eclipse IDE:



Java - OnlineStore/src/cucumberTest/TestRunner.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Explorer
JUnit
LogIn_Test.feature
TestRunner.java
SeleniumTest.java
Problems
Javadoc
Declaration
Console

- OnlineStore
  - src
    - cucumberTest
      - SeleniumTest.java
      - TestRunner.java
    - stepDefinition
  - JRE System Library [JavaSE-1.7]
  - Referenced Libraries
  - Feature
    - LogIn_Test.feature

```java
package cucumberTest;

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

@RunWith(Cucumber.class)
@CucumberOptions(
        features = "Feature"
        )

public class TestRunner {

}
```

<terminated> TestRunner (1) [JUnit] C:\Program Files\Java\jre7\bin\javaw.exe (29 Dec 2014 10:46:08)

cucumberTest.TestRunner.java - OnlineStore/src

## *Errors on running Cucumber Feature*

*Exception in thread "**main**" cucumber.runtime.CucumberException: No backends were found. Please make sure you have a backend module on your CLASSPATH.*

***Solution***

*Most probably this means that your **cucumber-java** version and **java** version on your machine is not compatible with each other.  First, check Java Version on your machine by going through this article **How to check Java/JDK Version Installed on your Machine**.*

*On my machine, I have Java 1.8.0 with **cucumber-Java8-1.2.5** and it did not work. When I degraded my cucumber java version to **cucumber-Java-1.2.5**, it worked fine for me. Just make sure that first, you remove the cucumber-java which did not work for you from **Project build path >> Libraries** and then add new. Keeping both may create further issues for you.*

*Exception in thread "**main**" java.lang.NoClassDefFoundError: gherkin/formatter/Formatter*

***Solution***

*This means that the Gherkin version you are using is not compatible with other Cucumber libraries. I tried using the latest **gherkin3-3.0.0**  but it did not work for me, so I degraded it to ***gherkin-2.12.2***

***I got below versions on Oct'17 for Cucumber***

*cobertura-2.1.1*<br>
*cucumber-core-1.2.5*<br>
*cucumber-java-1.2.5*<br>
*cucumber-junit-1.2.5*<br>
*cucumber-jvm-deps-1.0.5*<br>
*cucumber-reporting-3.10.0*<br>
*gherkin-2.12.2*<br>
*junit-4.12*<br>
*mockito-all-2.0.2-beta*

With this understanding let's move on to the next topic where we will talk about ***Gherkin Keywords*** and the syntax it provided to write application tests/behavior.