TESTNG

INTRODUCTION:

* It is an open source automated testing framework; where NG of TestNGmeans Next Generation.
* TestNG is similar to JUnit but it is much more powerful than JUnit but still it’s inspired by JUnit.
* It is designed to be better than JUnit, especially when testing integrated classes.

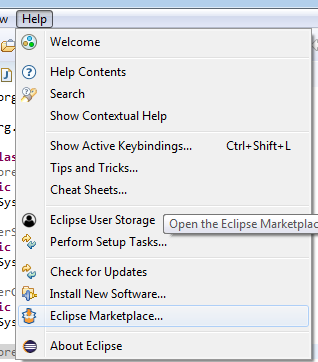
ADVANTAGES OF TESTNG:

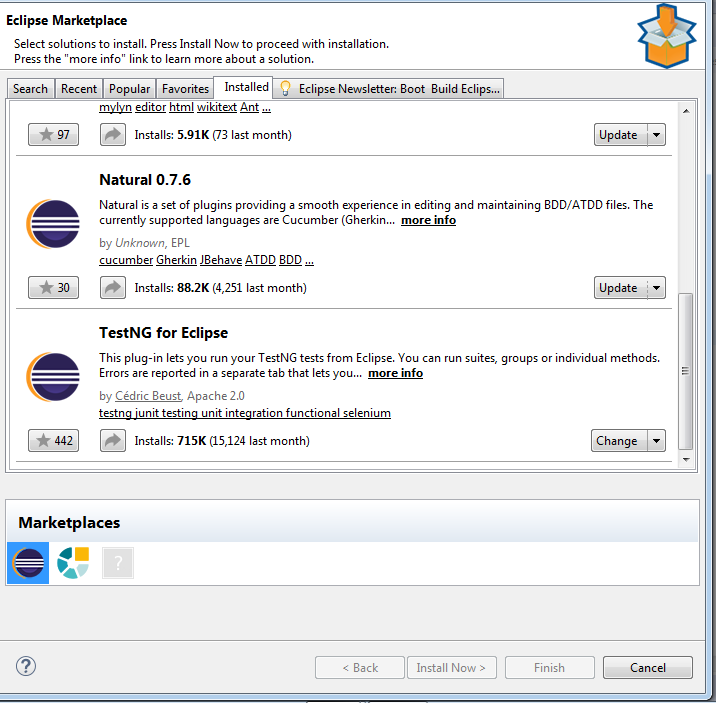
* It gives the ability to produce HTML Reports of execution
* Annotations made testers life easy
* Test cases can be Grouped & Prioritized more easily
* Parallel execution is possible
* Generates Logs
* Data Parameterization is possible
* Automatically return the failure test case

STEPS :

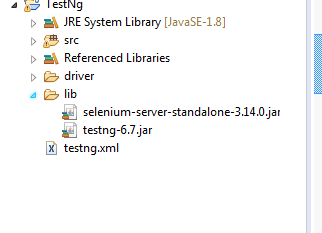
* Download the testng jar
* Add the testng jar in the eclipse buildpath
* Add the testng plugin in eclipse marketplace

Go to eclipse marketplace and install testng

* 



Add the testng jar file and configure



We have to click run as testng test

Annotations in TestNG:

@BeforeSuite: The annotated method will be run before all tests in this suite have run.

@AfterSuite: The annotated method will be run after all tests in this suite have run.

@BeforeTest: The annotated method will be run before any test method belonging to the classes inside the tag is run.

@AfterTest: The annotated method will be run after all the test methods belonging to the classes inside the tag have run.

@BeforeGroups: The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked.

@AfterGroups: The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked.

@BeforeClass: The annotated method will be run before the first test method in the current class is invoked.

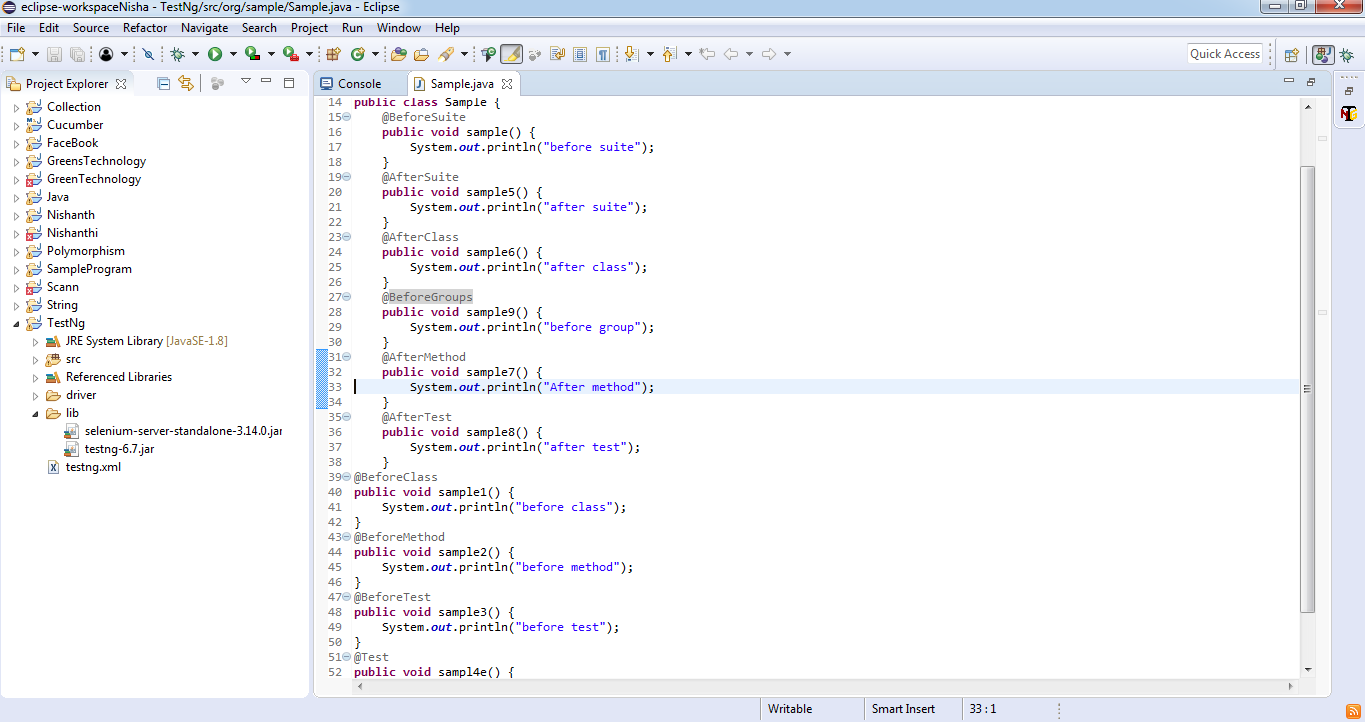
@AfterClass: The annotated method will be run after all the test methods in the current class have been run.

@BeforeMethod: The annotated method will be run before each test method.

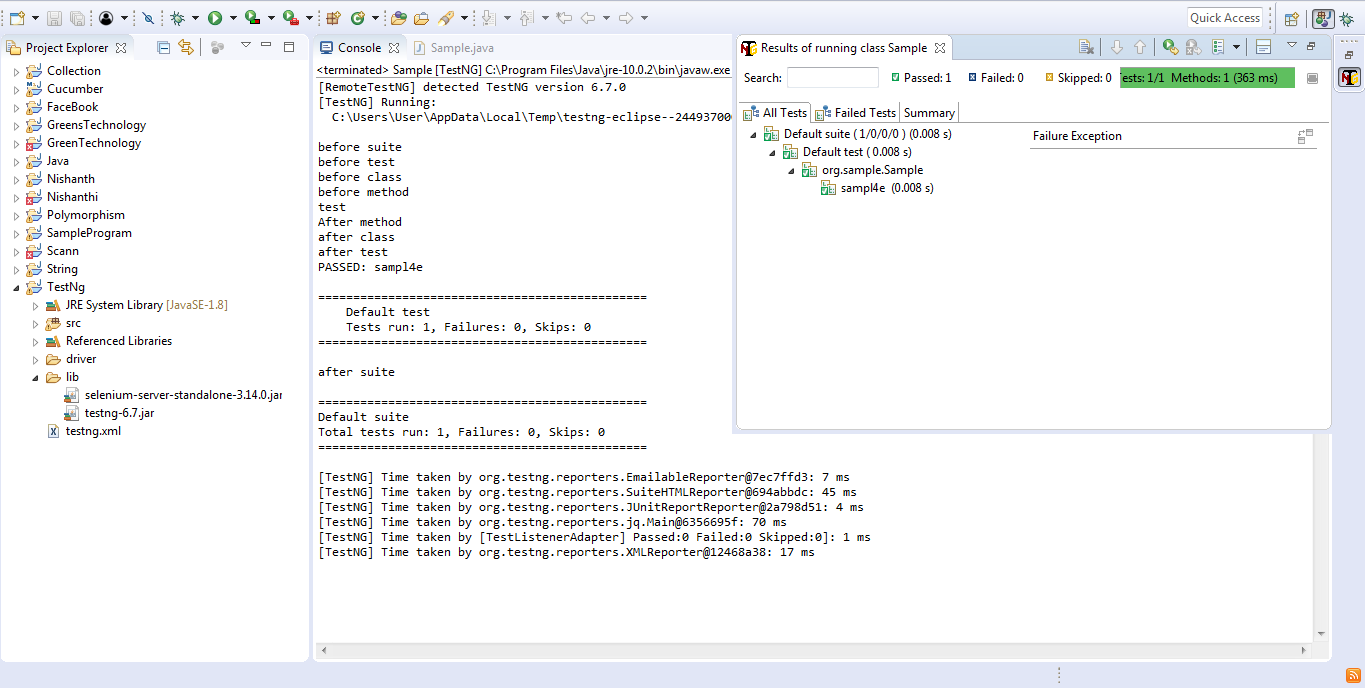
@AfterMethod: The annotated method will be run after each test method.

@Test: The annotated method is a part of a test case.

Ordered in which the annotation execute: Program

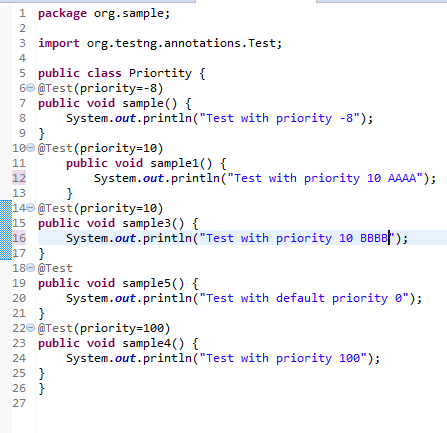


Output

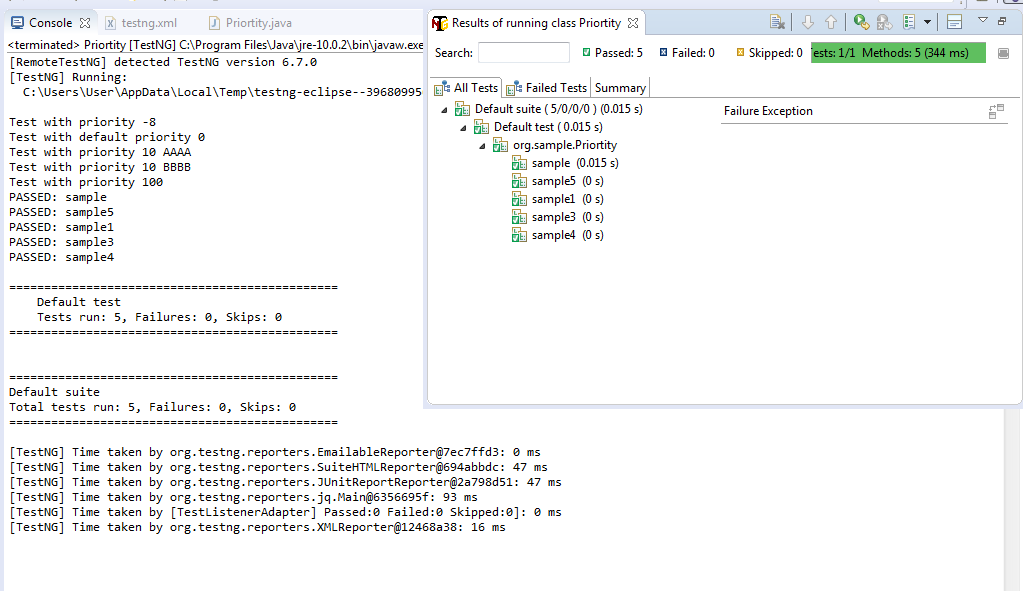


PRIORITY

* We can pass priority to the particular test case .
* We can pass both positive and negative value.
* It will execute based on ascending order.
* If we give Same priority then it will execute based on the alphabetic order.



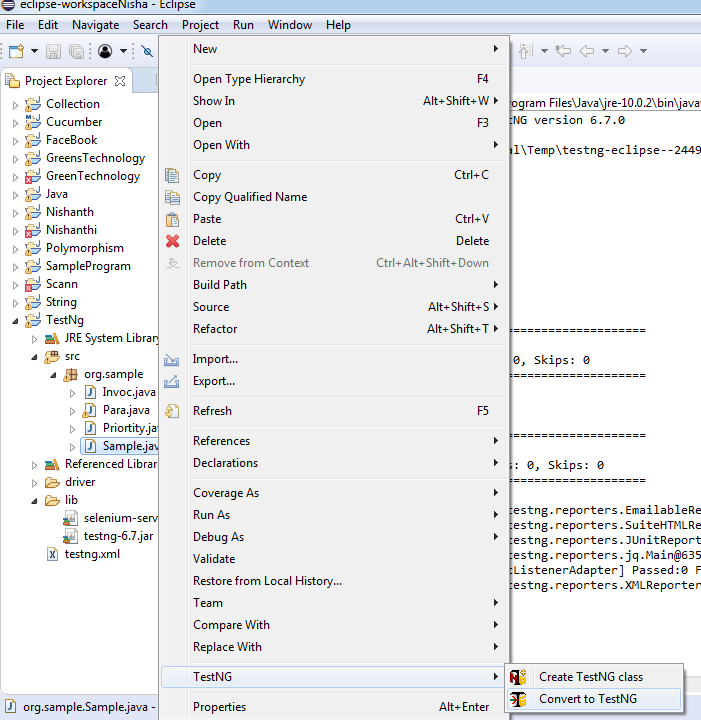
Ouput:

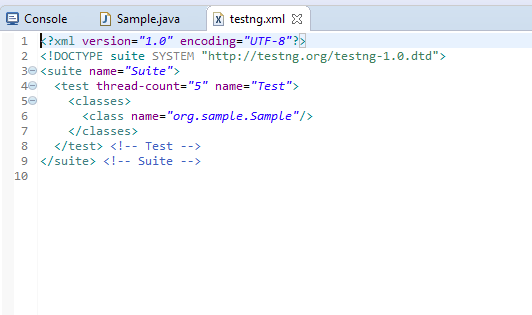


Suite-🡪Collection of test cases

TestCases🡪Collection of steps

We can also convert to xml by just right click the class and give

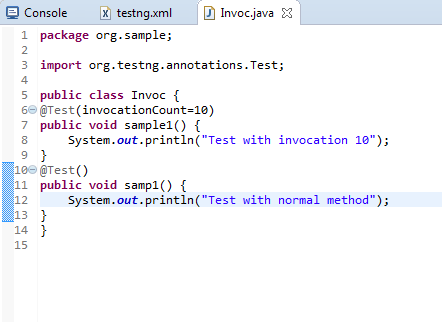
Testng🡪Convert to Testng

It will create a xml form

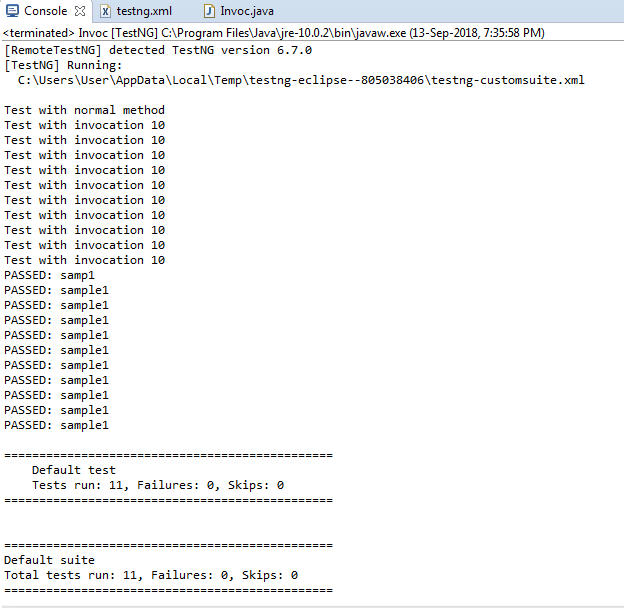
INVOCATION COUNT:

* If you want to run the particular test case to run for many times, We can use one method called invocation test case.
* It will run the test case for that particular times

Program:

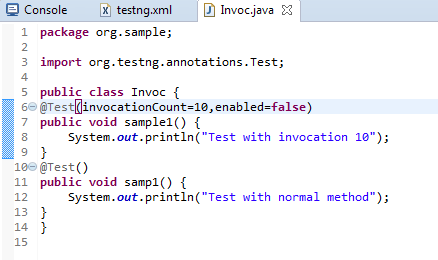


Output:

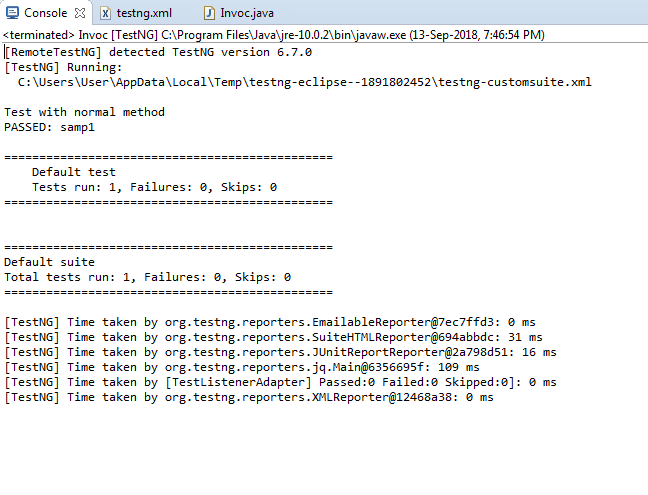


IGNORING THE TEST CASE:

* For ignoring the test case We can use one method called Enabled
* When we use enabled=false ,It will skip the particular test case

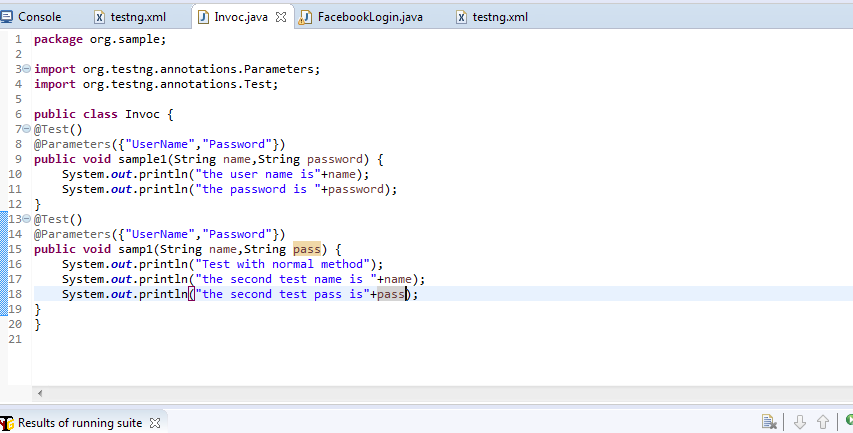
: 

Output:



PARAMETER:

You want to pass the input from xml sheet at that time parameters are used

You have to give @parameter annotation on the test case



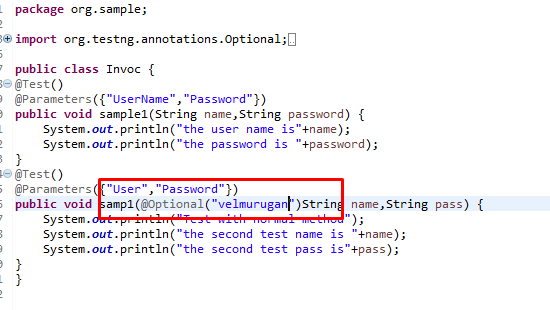
Output:



@Optional

In case of parameter is not exactly matched @optional is used

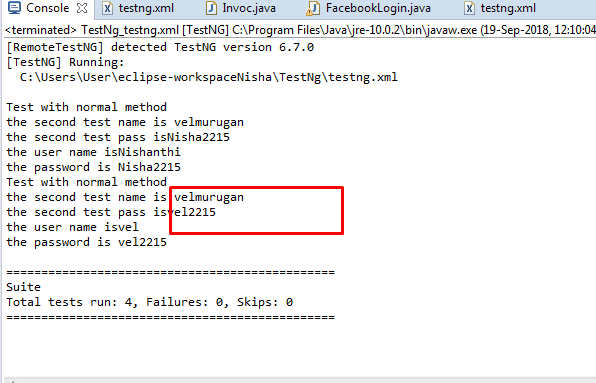
You have to pass the value at the time of initialization:



Here I am wrongly pass the parameter and I pass the optional value to.



Output:



Here it take the value from the optional not from the parameter.

Parallel Execution:

Thread-one person execute all the function in the program.

Multi Thread- more than one person will try to execute all the function in the program parallel.

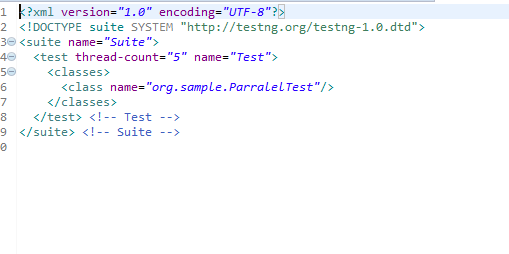
Default thread count is 5.

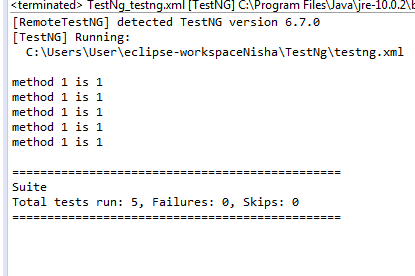
If you want to execute the 10 test case you have to set the test case as 10.

Multiposing test:

Run the test cases in many browser is called multiposing test.

We can see what happen if we don’t give parallel execution

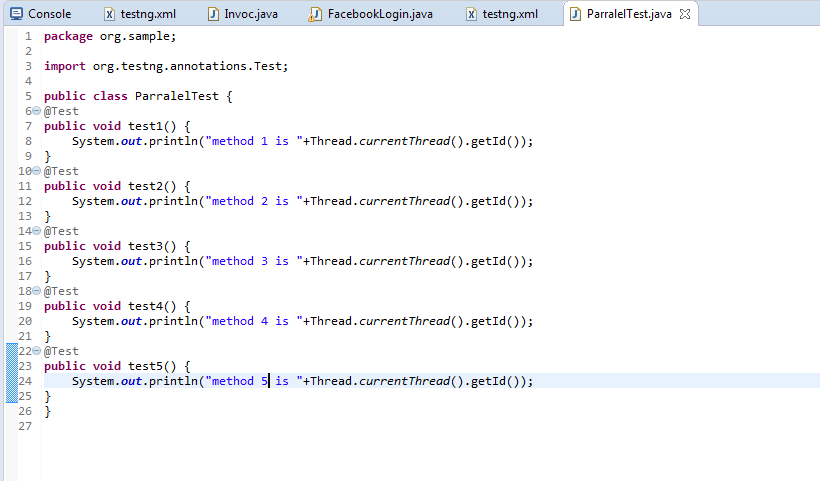


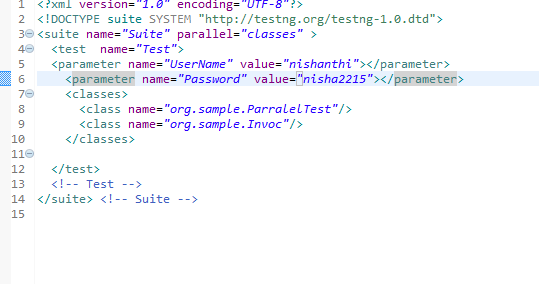
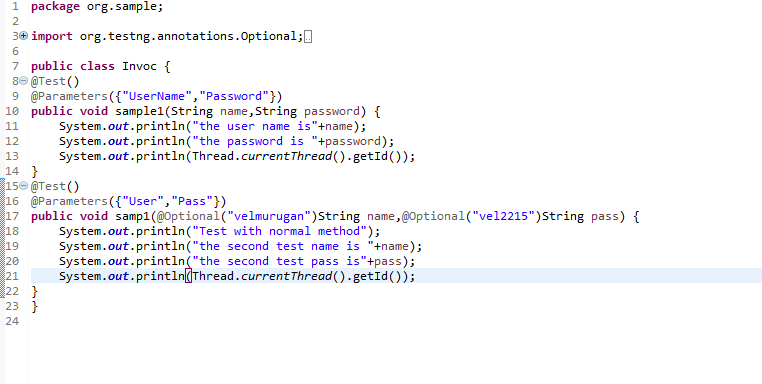


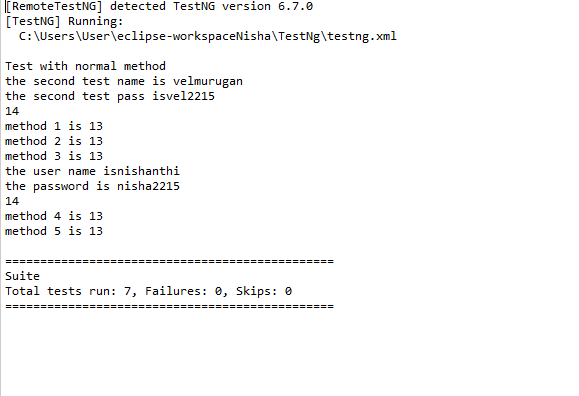
Here all the 5 tests will be run by only one test

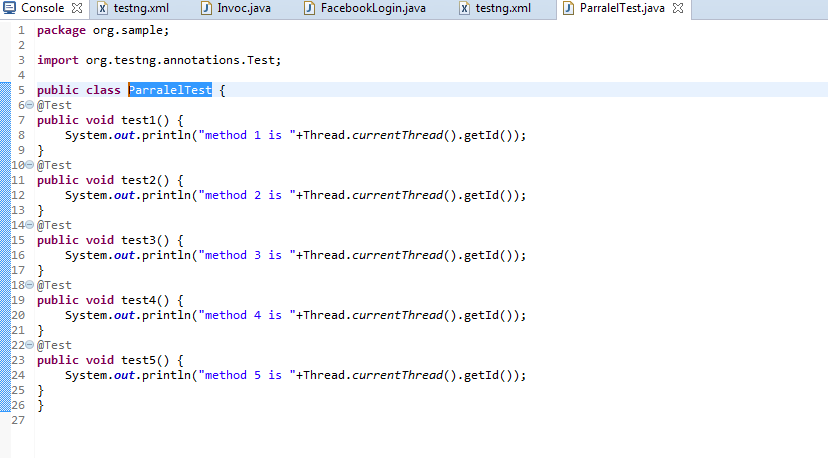
Ways to parallel execution:

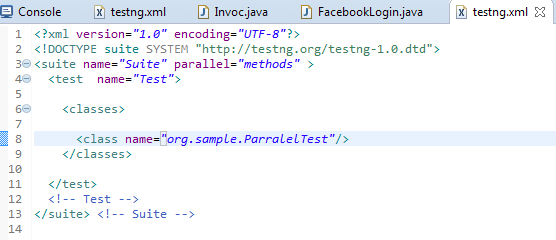
* Test
* Methods
* Classes

Classes: 

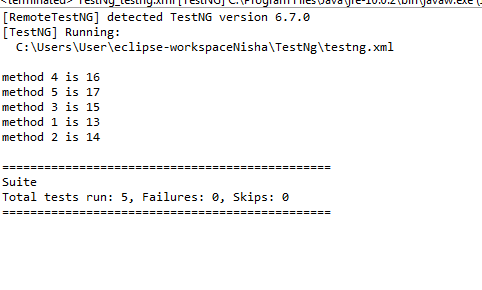


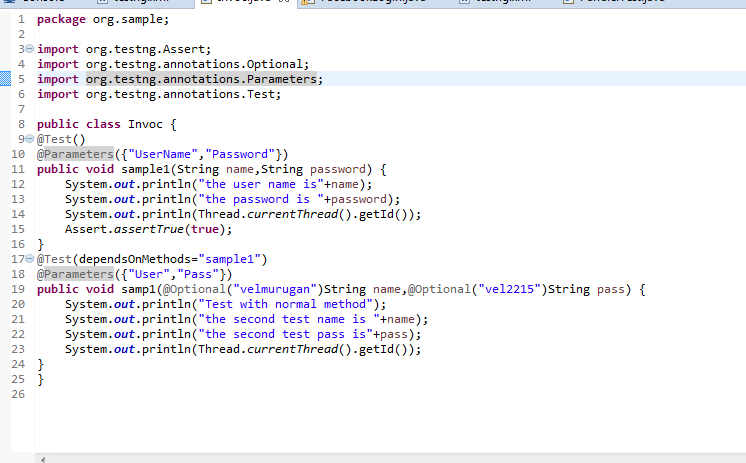
Output: 

Methods: 

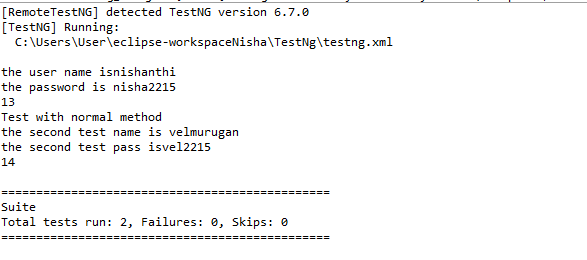


Output:

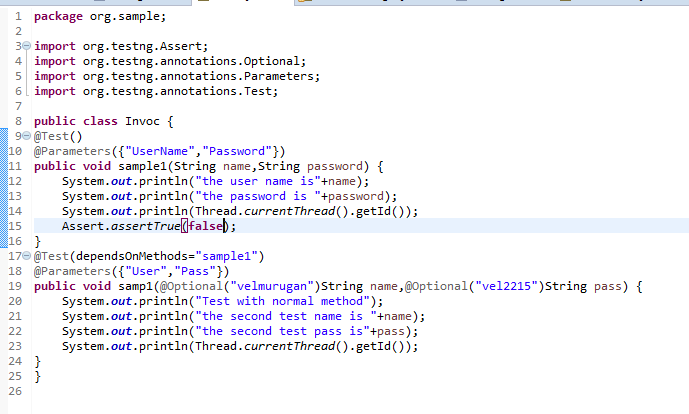


Depends on methods

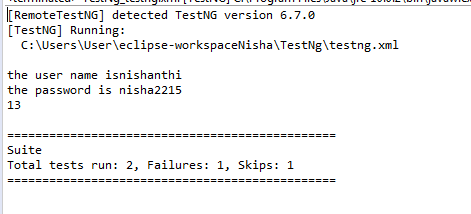
Output:



In case the depended method is false it skip the method



Ouput:



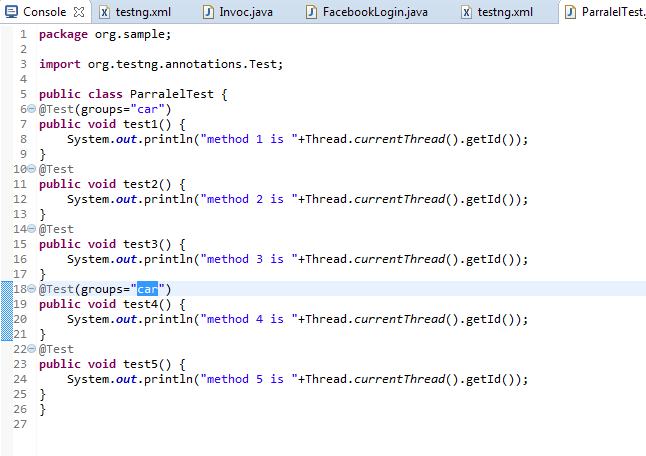
Here the test 1 is fail therefore test2 is skipped.

Groups:

We can group the multiple test cases by using groups concept

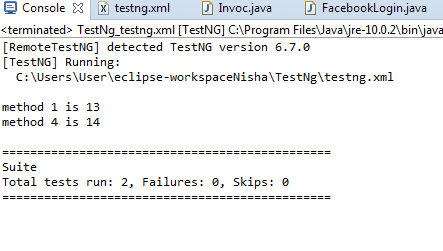
We have to give groups name in the @test annotation

@Test(groups=”grpname”)





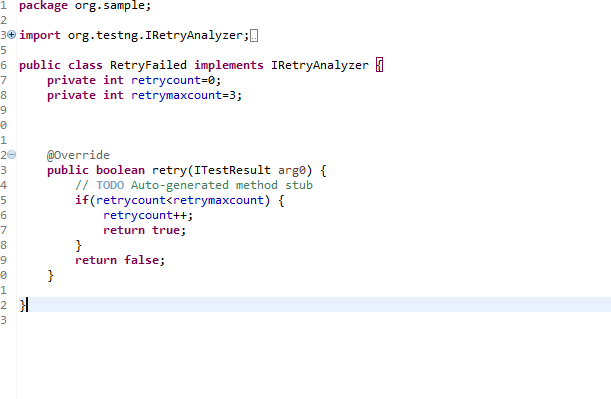
Output:



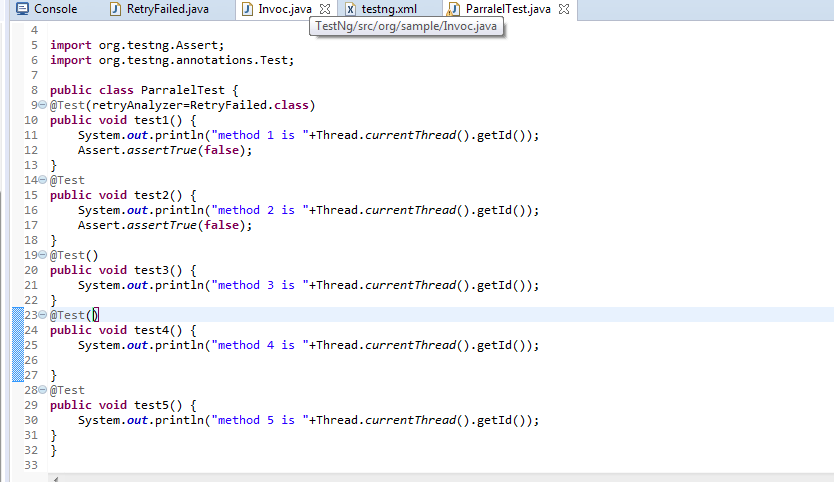
Re-execute the failed test

When we know the particular test case is failed. We have to use RetryAnalyzer interface is used

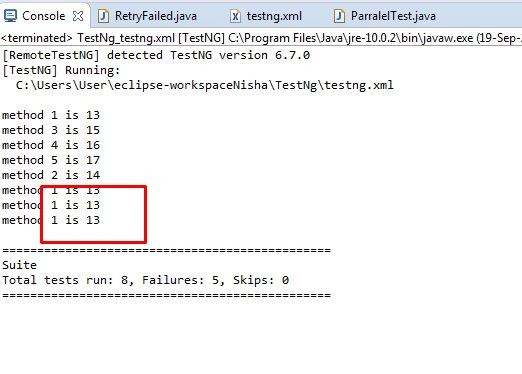
Program for retry class:



Program:



Output:



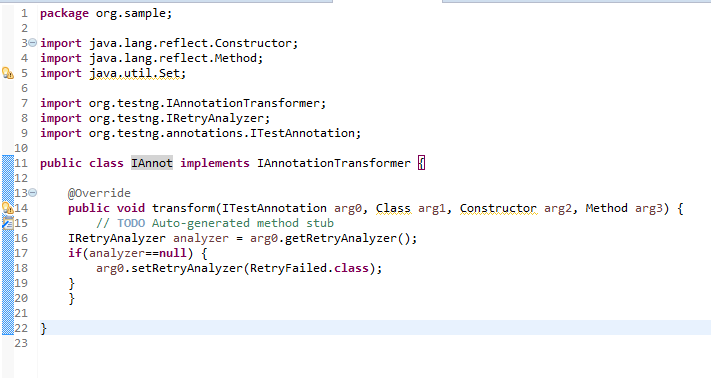
Here method 1 is executed 3 times because in that test I mention retryanalyzer=RetryFailed.class

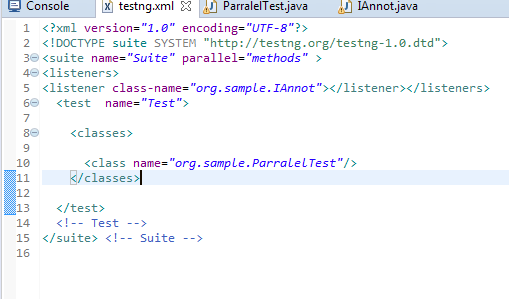
Test 2 is also failed but I don’t execute 4 times reason is I don’t mention retryanalyzer=RetryFailed.class

Re-Execute the test case we don’t know:

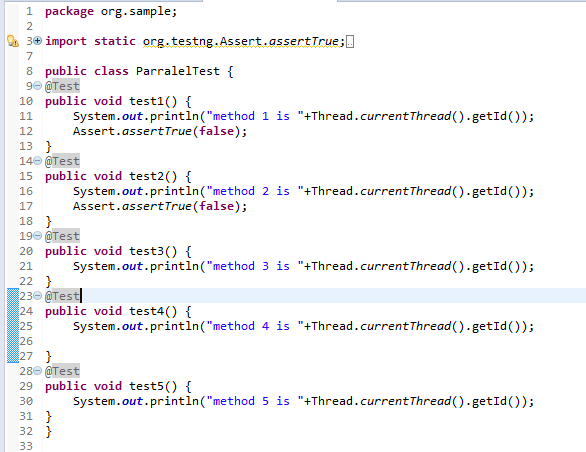
For re-execute the all test case we have to use IAnnotationTransformer

Program for IAnnotationTransformer:





Program:



Output:



Here All the failed test are executed 3 times.