

TESTNG

INTRODUCTION:

- It is an open source automated testing framework; where NG of TestNG means 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.

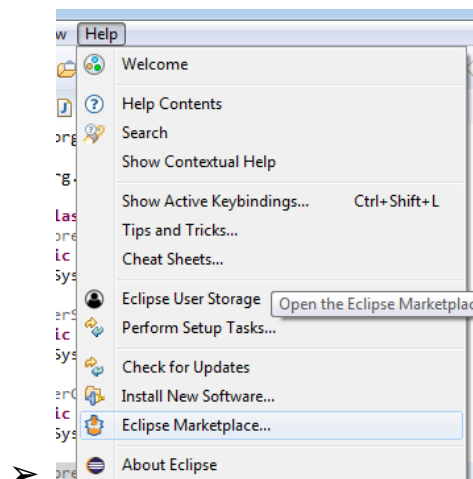
ADVANTAGES OF TESTNG:

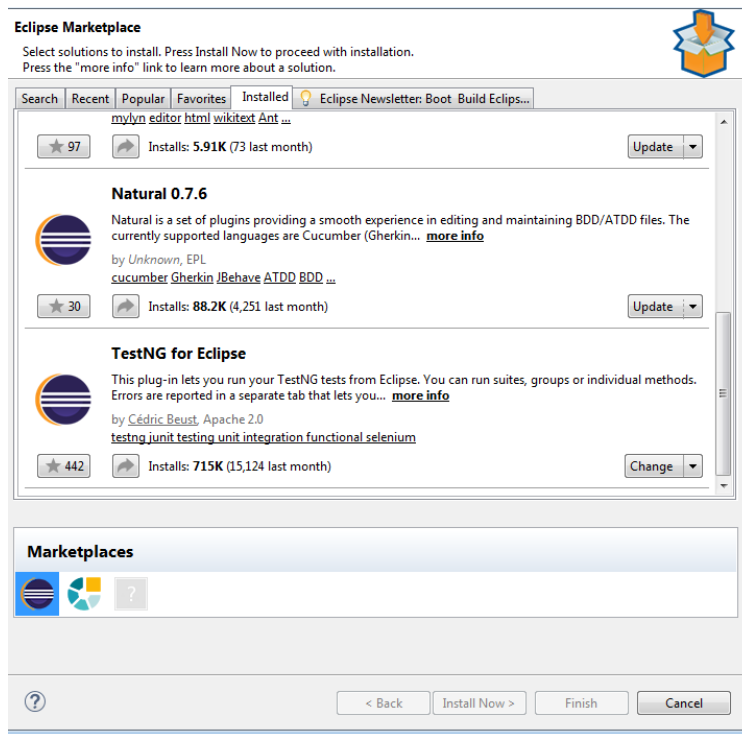
- It gives the ability to produce HTML Reports of execution
- Annotations made testers life easy
- Test cases can be Grouped & Prioritized more easily
- Parallel execution is possible
- Generates Logs
- Data Parameterization is possible
- Automatically return the failure test case

STEPS :

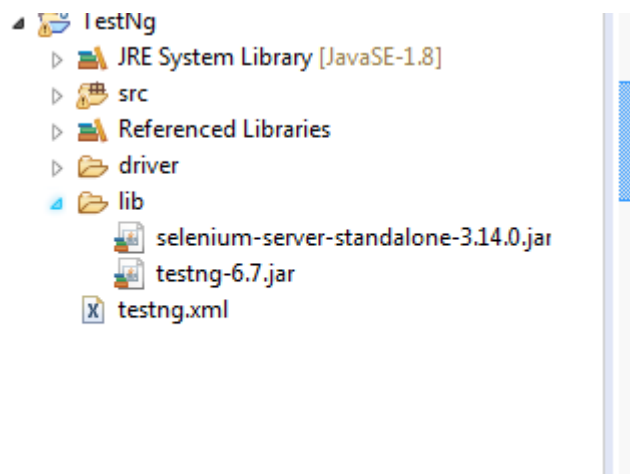
- Download the testng jar
- Add the testng jar in the eclipse buildpath
- Add the testng plugin in eclipse marketplace

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.

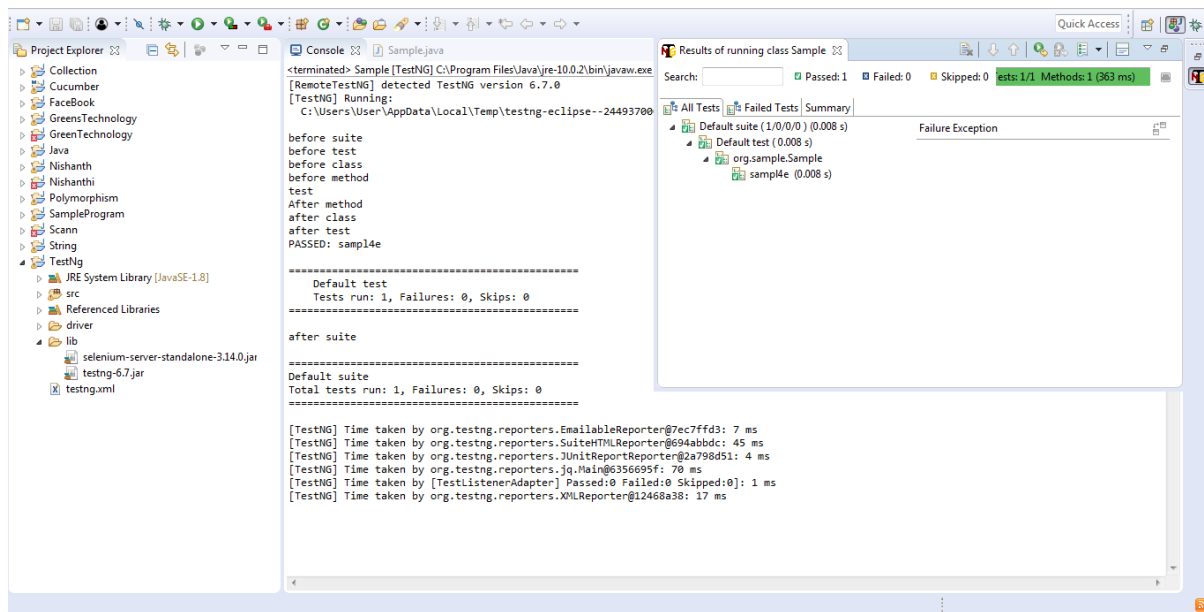
@AfterMethod: The annotated method will be run after each test method.

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

Ordered in which the annotation execute: **Program**

```
14 public class Sample {
15     @BeforeSuite
16     public void sample() {
17         System.out.println("before suite");
18     }
19     @AfterSuite
20     public void sample5() {
21         System.out.println("after suite");
22     }
23     @AfterClass
24     public void sample6() {
25         System.out.println("after class");
26     }
27     @BeforeGroups
28     public void sample9() {
29         System.out.println("before group");
30     }
31     @AfterMethod
32     public void sample7() {
33         System.out.println("after method");
34     }
35     @AfterTest
36     public void sample8() {
37         System.out.println("after test");
38     }
39     @BeforeClass
40     public void sample1() {
41         System.out.println("before class");
42     }
43     @BeforeMethod
44     public void sample2() {
45         System.out.println("before method");
46     }
47     @BeforeTest
48     public void sample3() {
49         System.out.println("before test");
50     }
51     @Test
52     public void sample4() {
```

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.

```

1 package org.sample;
2
3 import org.testng.annotations.Test;
4
5 public class Priority {
6     @Test(priority=-8)
7     public void sample() {
8         System.out.println("Test with priority -8");
9     }
10    @Test(priority=10)
11    public void sample1() {
12        System.out.println("Test with priority 10 AAAA");
13    }
14    @Test(priority=10)
15    public void sample3() {
16        System.out.println("Test with priority 10 BBBB");
17    }
18    @Test
19    public void sample5() {
20        System.out.println("Test with default priority 0");
21    }
22    @Test(priority=100)
23    public void sample4() {
24        System.out.println("Test with priority 100");
25    }
26 }
27

```

Output:

The screenshot shows the results of running the `Priority` class. The console on the left displays the output of the tests, and the right pane shows a tree view of the test results.

Console Output:

```

<terminated> Priority [TestNG] C:\Program Files\Java\jre-10.0.2\bin\javaw.exe
[RemoteTestNG] detected TestNG version 6.7.0
[TestNG] Running:
  C:\Users\User\AppData\Local\Temp\testng-eclipse--39680995

Test with priority -8
Test with default priority 0
Test with priority 10 AAAA
Test with priority 10 BBBB
Test with priority 100
PASSED: sample
PASSED: sample5
PASSED: sample1
PASSED: sample3
PASSED: sample4

=====
Default test
Tests run: 5, Failures: 0, Skips: 0
=====

Default suite
Total tests run: 5, Failures: 0, Skips: 0
=====

[TestNG] Time taken by org.testng.reporters.EmailableReporter@7ec7ffd3: 0 ms
[TestNG] Time taken by org.testng.reporters.SuiteHTMLReporter@694abddc: 47 ms
[TestNG] Time taken by org.testng.reporters.JUnitReportReporter@2a798d51: 47 ms
[TestNG] Time taken by org.testng.reporters.jq.Main@6356695f: 93 ms
[TestNG] Time taken by [TestListenerAdapter] Passed:0 Failed:0 Skipped:0]: 0 ms
[TestNG] Time taken by org.testng.reporters.XMLReporter@12468a38: 16 ms

```

Test Results Summary:

- Search:
- Passed: 5, Failed: 0, Skipped: 0
- Tests: 1/1, Methods: 5 (344 ms)

Test Results Tree:

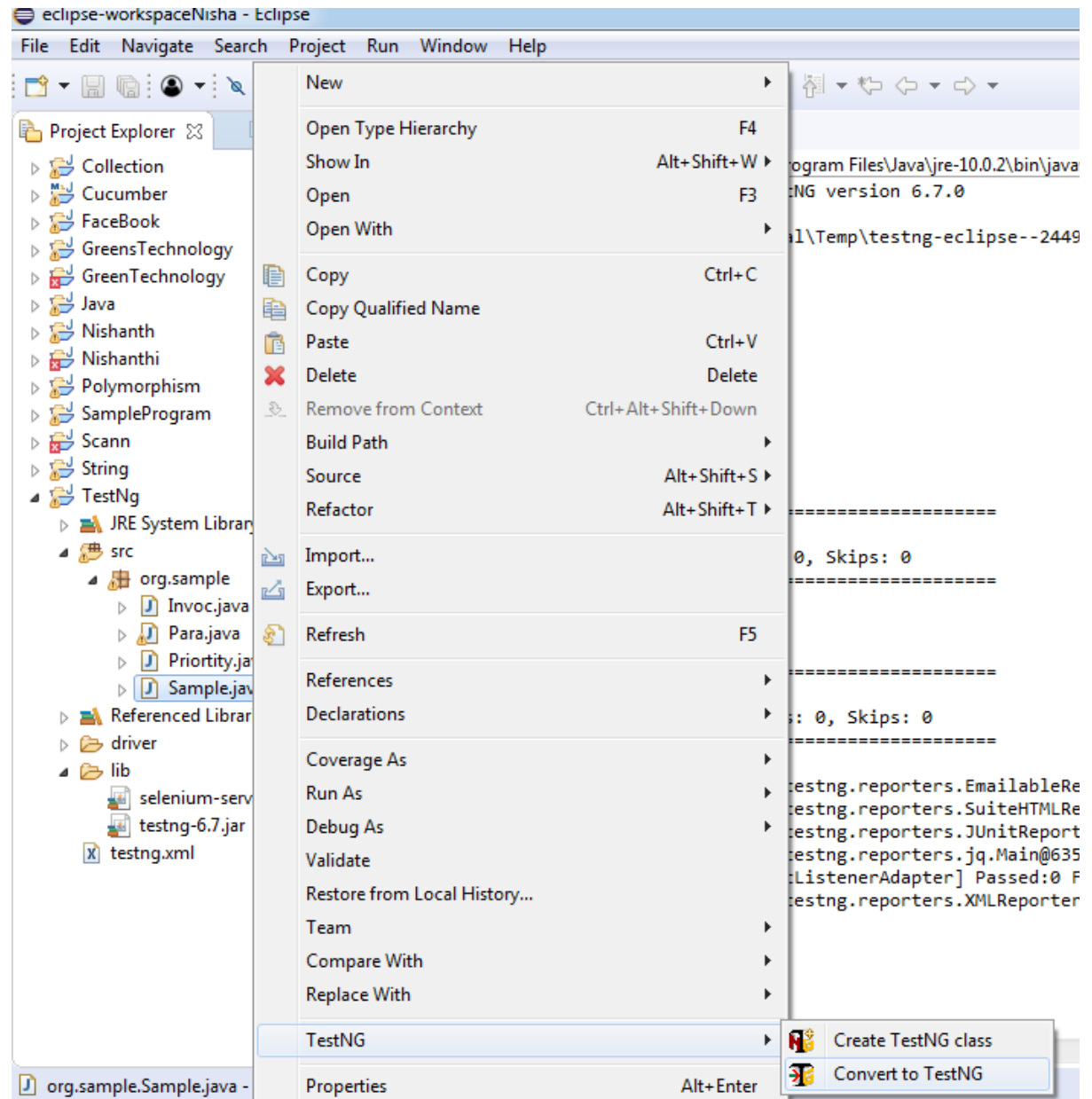
- Default suite (5/0/0/0) (0.015 s)
 - Default test (0.015 s)
 - org.sample.Priority
 - sample (0.015 s)
 - sample5 (0 s)
 - sample1 (0 s)
 - sample3 (0 s)
 - sample4 (0 s)

Suite→Collection of test cases

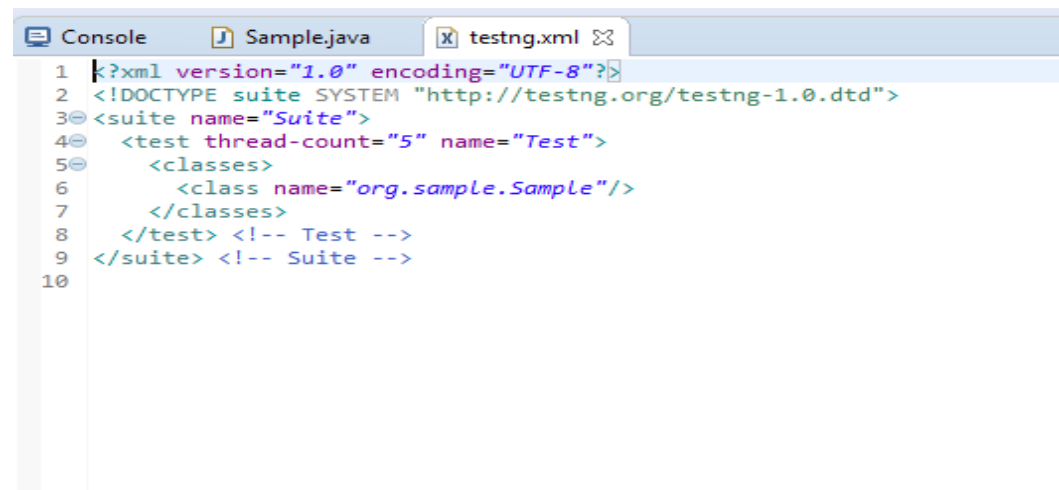
TestCases→Collection of steps

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

Testng→Convert to Testng



It will create a xml form

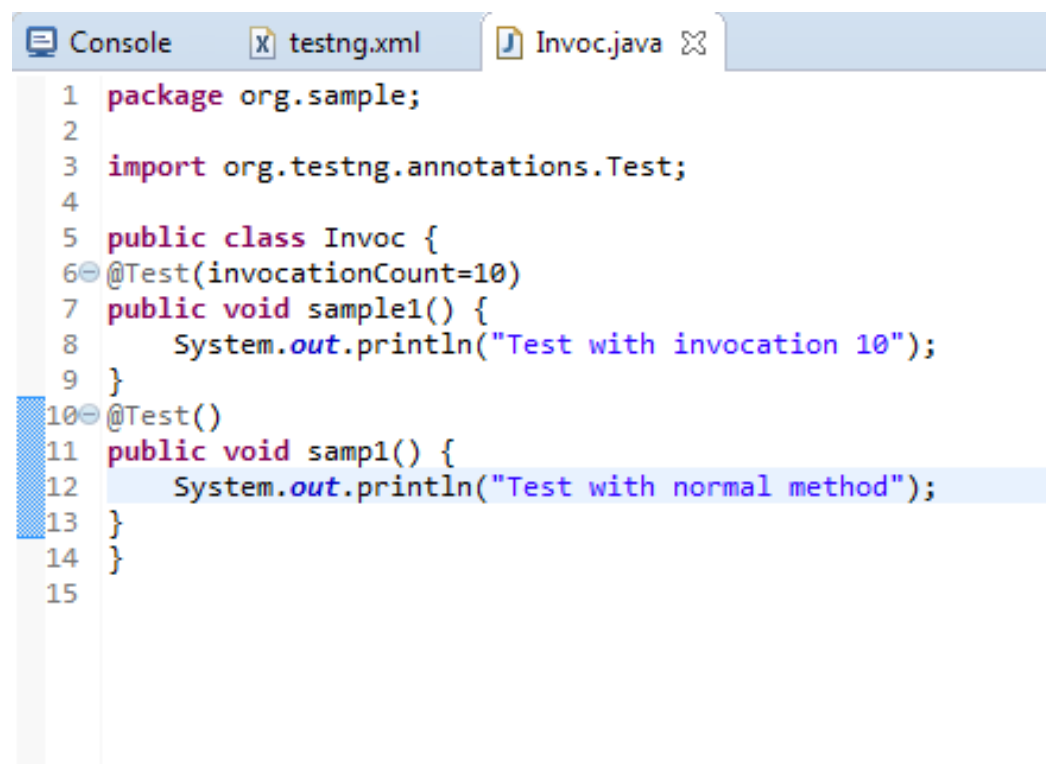


```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
3 <suite name="Suite">
4   <test thread-count="5" name="Test">
5     <classes>
6       <class name="org.sample.Sample"/>
7     </classes>
8   </test> <!-- Test -->
9 </suite> <!-- Suite -->
10
```

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:



```
1 package org.sample;
2
3 import org.testng.annotations.Test;
4
5 public class Invoc {
6   @Test(invocationCount=10)
7   public void sample1() {
8     System.out.println("Test with invocation 10");
9   }
10  @Test()
11  public void samp1() {
12    System.out.println("Test with normal method");
13  }
14 }
15
```

Output:

```

Console testng.xml Invoc.java
<terminated> Invoc [TestNG] C:\Program Files\Java\jre-10.0.2\bin\javaw.exe (13-Sep-2018, 7:35:58 PM)
[RemoteTestNG] detected TestNG version 6.7.0
[TestNG] Running:
  C:\Users\User\AppData\Local\Temp\testng-eclipse--805038406\testng-customsuite.xml

Test with normal method
Test with invocation 10
Test with invocation 10
Test with invocation 10
Test with invocation 10
Test with invocation 10
Test with invocation 10
Test with invocation 10
Test with invocation 10
Test with invocation 10
Test with invocation 10
PASSED: samp1
PASSED: sample1
PASSED: sample1
PASSED: sample1
PASSED: sample1
PASSED: sample1
PASSED: sample1
PASSED: sample1
PASSED: sample1
PASSED: sample1
PASSED: sample1
=====
    Default test
      Tests run: 11, Failures: 0, Skips: 0
=====

=====
Default suite
Total tests run: 11, Failures: 0, Skips: 0
=====

```

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


```

1 package org.sample;
2
3 import org.testng.annotations.Test;
4
5 public class Invoc {
6     @Test(invocationCount=10,enabled=false)
7     public void sample1() {
8         System.out.println("Test with invocation 10");
9     }
10    @Test()
11    public void samp1() {
12        System.out.println("Test with normal method");
13    }
14 }
15
:

```

Output:

```

<terminated> Invoc [TestNG] C:\Program Files\Java\jre-10.0.2\bin\javaw.exe (13-Sep-2018, 7:46:54 PM)
[[RemoteTestNG] detected TestNG version 6.7.0
[TestNG] Running:
  C:\Users\User\AppData\Local\Temp\testng-eclipse--1891802452\testng-customsuite.xml

Test with normal method
PASSED: samp1

=====
      Default test
      Tests run: 1, Failures: 0, Skips: 0
=====

=====
Default suite
Total tests run: 1, Failures: 0, Skips: 0
=====

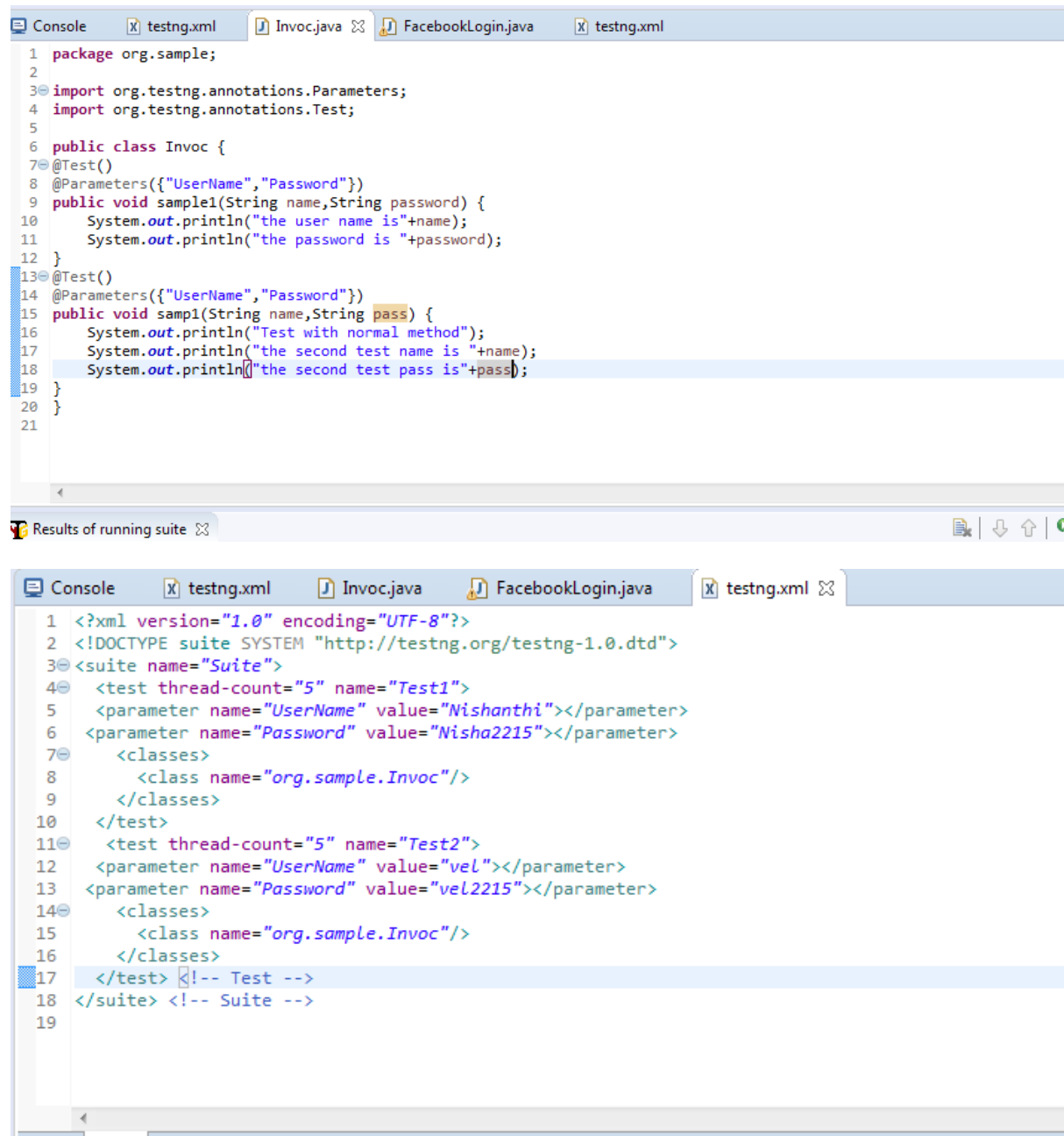
[TestNG] Time taken by org.testng.reporters.EmailableReporter@7ec7ffd3: 0 ms
[TestNG] Time taken by org.testng.reporters.SuiteHTMLReporter@694abbd3: 31 ms
[TestNG] Time taken by org.testng.reporters.JUnitReportReporter@2a798d51: 16 ms
[TestNG] Time taken by org.testng.reporters.jq.Main@6356695f: 109 ms
[TestNG] Time taken by [TestListenerAdapter] Passed:0 Failed:0 Skipped:0: 0 ms
[TestNG] Time taken by org.testng.reporters.XMLReporter@12468a38: 0 ms

```

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



The screenshot displays an IDE with two tabs: 'testng.xml' and 'Invoc.java'. The 'Invoc.java' tab is active, showing a Java class with two test methods. The first method, 'sample1', uses '@Parameters' and '@Test' annotations. The second method, 'sampl', also uses '@Parameters' and '@Test' annotations, but includes a '@parameter' annotation on the 'pass' parameter. The 'testng.xml' tab is also visible, showing the XML output of the test suite. The XML output includes two test cases, 'Test1' and 'Test2', each with parameters for 'UserName' and 'Password'. The 'Test1' test case has parameters 'Nishanthi' and 'Nisha2215', while 'Test2' has 'vel' and 'vel2215'. The XML output is as follows:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
3 <suite name="Suite">
4   <test thread-count="5" name="Test1">
5     <parameter name="UserName" value="Nishanthi"></parameter>
6     <parameter name="Password" value="Nisha2215"></parameter>
7     <classes>
8       <class name="org.sample.Invoc"/>
9     </classes>
10  </test>
11  <test thread-count="5" name="Test2">
12    <parameter name="UserName" value="vel"></parameter>
13    <parameter name="Password" value="vel2215"></parameter>
14    <classes>
15      <class name="org.sample.Invoc"/>
16    </classes>
17  </test> <!-- Test -->
18 </suite> <!-- Suite -->
19
```

Output:

```

Console  X testng.xml  J Invoc.java  J FacebookLogin.java  X testng.xml
<terminated> TestNg_testng.xml [TestNG] C:\Program Files\Java\jre-10.0.2\bin\javaw.exe (19-Sep-2018, 7:40:12 AM)
[RemoteTestNG] detected TestNG version 6.7.0
[TestNG] Running:
  C:\Users\User\eclipse-workspaceNisha\TestNg\testng.xml

Test with normal method
the second test name is Nishanthi
the second test pass isNisha2215
the user name isNishanthi
the password is Nisha2215
Test with normal method
the second test name is vel
the second test pass isvel2215
the user name isvel
the password is vel2215

=====
Suite
Total tests run: 4, Failures: 0, Skips: 0
=====

```

@Optional

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

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

```

1 package org.sample;
2
3 import org.testng.annotations.Optional;
4
5
6 public class Invoc {
7     @Test()
8     @Parameters({"UserName","Password"})
9     public void sample1(String name,String password) {
10         System.out.println("the user name is"+name);
11         System.out.println("the password is "+password);
12     }
13
14     @Test()
15     @Parameters({"User","Password"})
16     public void samp1(@Optional("velmurugan")String name,String pass) {
17         System.out.println("Test with normal method");
18         System.out.println("the second test name is "+name);
19         System.out.println("the second test pass is"+pass);
20     }
21 }
2

```

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

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
3 <suite name="Suite">
4   <test thread-count="5" name="Test1">
5     <parameter name="UserName" value="Nishanthi"></parameter>
6     <parameter name="Password" value="Nisha2215"></parameter>
7     <classes>
8       <class name="org.sample.Invoc"/>
9     </classes>
10  </test>
11  <test thread-count="5" name="Test2">
12    <parameter name="UserName" value="vel"></parameter>
13    <parameter name="Password" value="vel2215"></parameter>
14    <classes>
15      <class name="org.sample.Invoc"/>
16    </classes>
17  </test> <!-- Test -->
18 </suite> <!-- Suite -->
19
```

Output:

```
<terminated> TestNg_testng.xml [TestNG] C:\Program Files\Java\jre-10.0.2\bin\javaw.exe (19-Sep-2018, 12:10:04)
[RemoteTestNG] detected TestNG version 6.7.0
[TestNG] Running:
  C:\Users\User\eclipse-workspaceNisha\TestNg\testng.xml

Test with normal method
the second test name is velmurugan
the second test pass isNisha2215
the user name isNishanthi
the password is Nisha2215
Test with normal method
the second test name is velmurugan
the second test pass isvel2215
the user name isvel
the password is vel2215

=====
Suite
Total tests run: 4, Failures: 0, Skips: 0
=====
```

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

```
Console testng.xml Invoc.java FacebookLogin.java testng.xml ParralelTest.java
1 package org.sample;
2
3 import org.testng.annotations.Test;
4
5 public class ParralelTest {
6     @Test
7     public void test1() {
8         System.out.println("method 1 is "+Thread.currentThread().getId());
9     }
10    @Test
11    public void test2() {
12        System.out.println("method 1 is "+Thread.currentThread().getId());
13    }
14    @Test
15    public void test3() {
16        System.out.println("method 1 is "+Thread.currentThread().getId());
17    }
18    @Test
19    public void test4() {
20        System.out.println("method 1 is "+Thread.currentThread().getId());
21    }
22    @Test
23    public void test5() {
24        System.out.println("method 1 is "+Thread.currentThread().getId());
25    }
26 }
27
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
3 <suite name="Suite">
4     <test thread-count="5" name="Test">
5         <classes>
6             <class name="org.sample.ParralelTest"/>
7         </classes>
8     </test> <!-- Test -->
9 </suite> <!-- Suite -->
0
```

```

<terminated> TestNg_testng.xml [TestNG] C:\Program Files\Java\jre-10.0.2\k
[RemoteTestNG] detected TestNG version 6.7.0
[TestNG] Running:
  C:\Users\User\workspaceNisha\TestNg\testng.xml

method 1 is 1
method 1 is 1
method 1 is 1
method 1 is 1
method 1 is 1

=====
Suite
Total tests run: 5, Failures: 0, Skips: 0
=====

```

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

Ways to parallel execution:

- Test
- Methods
- Classes

Classes:



```

1 package org.sample;
2
3 import org.testng.annotations.Test;
4
5 public class ParralelTest {
6     @Test
7     public void test1() {
8         System.out.println("method 1 is "+Thread.currentThread().getId());
9     }
10    @Test
11    public void test2() {
12        System.out.println("method 2 is "+Thread.currentThread().getId());
13    }
14    @Test
15    public void test3() {
16        System.out.println("method 3 is "+Thread.currentThread().getId());
17    }
18    @Test
19    public void test4() {
20        System.out.println("method 4 is "+Thread.currentThread().getId());
21    }
22    @Test
23    public void test5() {
24        System.out.println("method 5 is "+Thread.currentThread().getId());
25    }
26 }
27

```

```

1 package org.sample;
2
3 import org.testng.annotations.Optional;
4
5
6
7 public class Invoc {
8     @Test()
9     @Parameters({"UserName", "Password"})
10    public void sample1(String name, String password) {
11        System.out.println("the user name is "+name);
12        System.out.println("the password is "+password);
13        System.out.println(Thread.currentThread().getId());
14    }
15    @Test()
16    @Parameters({"User", "Pass"})
17    public void sampl(@Optional("velmurugan")String name, @Optional("vel2215")String pass) {
18        System.out.println("Test with normal method");
19        System.out.println("the second test name is "+name);
20        System.out.println("the second test pass is "+pass);
21        System.out.println(Thread.currentThread().getId());
22    }
23 }
24

```

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
3 <suite name="Suite" parallel="classes" >
4     <test name="Test">
5         <parameter name="UserName" value="nishanthi"></parameter>
6         <parameter name="Password" value="nisha2215"></parameter>
7     <classes>
8         <class name="org.sample.ParralelTest"/>
9         <class name="org.sample.Invoc"/>
10    </classes>
11 </test>
12 <!-- Test -->
13 </suite> <!-- Suite -->
14
15

```


Output:

```
[[RemoteTestNG] detected TestNG version 6.7.0
[TestNG] Running:
  C:\Users\User\eclipse-workspaceNisha\TestNg\testng.xml

Test with normal method
the second test name is velmurugan
the second test pass isvel2215
14
method 1 is 13
method 2 is 13
method 3 is 13
the user name isnishanthi
the password is nisha2215
14
method 4 is 13
method 5 is 13

=====
Suite
Total tests run: 7, Failures: 0, Skips: 0
=====
```

Methods:



```
1 package org.sample;
2
3 import org.testng.annotations.Test;
4
5 public class ParallelTest {
6     @Test
7     public void test1() {
8         System.out.println("method 1 is "+Thread.currentThread().getId());
9     }
10    @Test
11    public void test2() {
12        System.out.println("method 2 is "+Thread.currentThread().getId());
13    }
14    @Test
15    public void test3() {
16        System.out.println("method 3 is "+Thread.currentThread().getId());
17    }
18    @Test
19    public void test4() {
20        System.out.println("method 4 is "+Thread.currentThread().getId());
21    }
22    @Test
23    public void test5() {
24        System.out.println("method 5 is "+Thread.currentThread().getId());
25    }
26 }
27
```

```
Console  testng.xml  Invoc.java  FacebookLogin.java  testng.xml  ⌵
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
3  <suite name="Suite" parallel="methods" >
4    <test name="Test">
5
6    <classes>
7
8    <class name="org.sample.ParralelTest"/>
9    </classes>
10
11  </test>
12  <!-- Test -->
13 </suite> <!-- Suite -->
14
```

Output:

```
<terminated> testng_testng.xml [testng] C:\Program Files\Java\jre-10.0.2\bin\javaw.exe (-
[RemoteTestNG] detected TestNG version 6.7.0
[TestNG] Running:
  C:\Users\User\eclipse-workspaceNisha\TestNg\testng.xml

method 4 is 16
method 5 is 17
method 3 is 15
method 1 is 13
method 2 is 14

=====
Suite
Total tests run: 5, Failures: 0, Skips: 0
=====
```

Depends on methods

```
1 package org.sample;
2
3 import org.testng.Assert;
4 import org.testng.annotations.Optional;
5 import org.testng.annotations.Parameters;
6 import org.testng.annotations.Test;
7
8 public class Invoc {
9     @Test()
10     @Parameters({"UserName", "Password"})
11     public void sample1(String name, String password) {
12         System.out.println("the user name is "+name);
13         System.out.println("the password is "+password);
14         System.out.println(Thread.currentThread().getId());
15         Assert.assertTrue(true);
16     }
17     @Test(dependsOnMethods="sample1")
18     @Parameters({"User", "Pass"})
19     public void sampl(@Optional("velmurugan")String name, @Optional("vel2215")String pass) {
20         System.out.println("Test with normal method");
21         System.out.println("the second test name is "+name);
22         System.out.println("the second test pass is "+pass);
23         System.out.println(Thread.currentThread().getId());
24     }
25 }
26
```

Output:

```
[RemoteTestNG] detected TestNG version 6.7.0
[TestNG] Running:
  C:\Users\User\eclipse-workspaceNisha\TestNg\testng.xml
```

```
the user name isnishanthi
the password is nisha2215
13
Test with normal method
the second test name is velmurugan
the second test pass is vel2215
14
```

```
=====
Suite
Total tests run: 2, Failures: 0, Skips: 0
=====
```

In case the depended method is false it skip the method

```
1 package org.sample;
2
3 import org.testng.Assert;
4 import org.testng.annotations.Optional;
5 import org.testng.annotations.Parameters;
6 import org.testng.annotations.Test;
7
8 public class Invoc {
9     @Test()
10     @Parameters({"UserName", "Password"})
11     public void sample1(String name, String password) {
12         System.out.println("the user name is" + name);
13         System.out.println("the password is " + password);
14         System.out.println(Thread.currentThread().getId());
15         Assert.assertTrue(false);
16     }
17     @Test(dependsOnMethods = "sample1")
18     @Parameters({"User", "Pass"})
19     public void samp1(@Optional("velmurugan") String name, @Optional("vel2215") String pass) {
20         System.out.println("Test with normal method");
21         System.out.println("the second test name is " + name);
22         System.out.println("the second test pass is " + pass);
23         System.out.println(Thread.currentThread().getId());
24     }
25 }
26
```

Output:

```
[[RemoteTestNG] detected TestNG version 6.7.0
[TestNG] Running:
  C:\Users\User\eclipse-workspaceNisha\TestNg\testng.xml

the user name isnishanthi
the password is nisha2215
13

=====
Suite
Total tests run: 2, Failures: 1, Skips: 1
=====
```

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")

```
Console testng.xml Invoc.java FacebookLogin.java testng.xml ParralelTest.
1 package org.sample;
2
3 import org.testng.annotations.Test;
4
5 public class ParralelTest {
6 @Test(groups="car")
7 public void test1() {
8     System.out.println("method 1 is "+Thread.currentThread().getId());
9 }
10 @Test
11 public void test2() {
12     System.out.println("method 2 is "+Thread.currentThread().getId());
13 }
14 @Test
15 public void test3() {
16     System.out.println("method 3 is "+Thread.currentThread().getId());
17 }
18 @Test(groups="car")
19 public void test4() {
20     System.out.println("method 4 is "+Thread.currentThread().getId());
21 }
22 @Test
23 public void test5() {
24     System.out.println("method 5 is "+Thread.currentThread().getId());
25 }
26 }
27
```

```
Console testng.xml Invoc.java FacebookLogin.java testng.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
3 <suite name="Suite" parallel="methods" >
4 <test name="Test">
5 <groups>
6 <define name="one">
7 <include name="cars"></include></define><run>
8 <include name="car"></include>
9 </run></groups>
10 <classes>
11
12 <class name="org.sample.ParralelTest"/>
13 </classes>
14
15 </test>
16 <!-- Test -->
17 </suite> <!-- Suite -->
18
```

Output:

```
Console  testng.xml  Invoc.java  FacebookLogin.java
<terminated> TestNg_testng.xml [TestNG] C:\Program Files\Java\jre-10.0.2\bin\java
[RemoteTestNG] detected TestNG version 6.7.0
[TestNG] Running:
  C:\Users\User\eclipse-workspaceNisha\TestNg\testng.xml

method 1 is 13
method 4 is 14

=====
Suite
Total tests run: 2, Failures: 0, Skips: 0
=====
```

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:

```
1 package org.sample;
2
3 import org.testng.IRetryAnalyzer;
4
5
6 public class RetryFailed implements IRetryAnalyzer {
7     private int retrycount=0;
8     private int retrymaxcount=3;
9
10
11     @Override
12     public boolean retry(ITestResult arg0) {
13         // TODO Auto-generated method stub
14         if(retrycount<retrymaxcount) {
15             retrycount++;
16             return true;
17         }
18         return false;
19     }
20 }
21
22
23 }
```

Program:

```
4
5 import org.testng.Assert;
6 import org.testng.annotations.Test;
7
8 public class ParralelTest {
9     @Test(retryAnalyzer=RetryFailed.class)
10    public void test1() {
11        System.out.println("method 1 is "+Thread.currentThread().getId());
12        Assert.assertTrue(false);
13    }
14    @Test
15    public void test2() {
16        System.out.println("method 2 is "+Thread.currentThread().getId());
17        Assert.assertTrue(false);
18    }
19    @Test()
20    public void test3() {
21        System.out.println("method 3 is "+Thread.currentThread().getId());
22    }
23    @Test()
24    public void test4() {
25        System.out.println("method 4 is "+Thread.currentThread().getId());
26    }
27    }
28    @Test
29    public void test5() {
30        System.out.println("method 5 is "+Thread.currentThread().getId());
31    }
32    }
33    }
```

Output:

```
<terminated> TestNg_testng.xml [TestNG] C:\Program Files\Java\jre-10.0.2\bin\javaw.exe (19-Sep-
[RemoteTestNG] detected TestNG version 6.7.0
[TestNG] Running:
  C:\Users\User\eclipse-workspaceNisha\TestNg\testng.xml

method 1 is 13
method 3 is 15
method 4 is 16
method 5 is 17
method 2 is 14
method 1 is 13
method 1 is 13
method 1 is 13
=====
Suite
Total tests run: 8, Failures: 5, Skips: 0
=====
```

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:

```
1 package org.sample;
2
3 import java.lang.reflect.Constructor;
4 import java.lang.reflect.Method;
5 import java.util.Set;
6
7 import org.testng.IAnnotationTransformer;
8 import org.testng.IRetryAnalyzer;
9 import org.testng.annotations.ITestAnnotation;
10
11 public class IAnnot implements IAnnotationTransformer {
12
13     @Override
14     public void transform(ITestAnnotation arg0, Class arg1, Constructor arg2, Method arg3) {
15         // TODO Auto-generated method stub
16         IRetryAnalyzer analyzer = arg0.getRetryAnalyzer();
17         if(analyzer==null) {
18             arg0.setRetryAnalyzer(RetryFailed.class);
19         }
20     }
21 }
22
23
```

```
Console | testng.xml | ParralelTest.java | IAnnot.java
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
3 <suite name="Suite" parallel="methods" >
4 <listeners>
5 <listener class-name="org.sample.IAnnot"></listener></listeners>
6 <test name="Test">
7
8 <classes>
9
10 <class name="org.sample.ParralelTest"/>
11 </classes>
12
13 </test>
14 <!-- Test -->
15 </suite> <!-- Suite -->
16
```


Program:

```
1 package org.sample;
2
3 import static org.testng.Assert.assertTrue;
4
5
6
7
8 public class ParralelTest {
9     @Test
10    public void test1() {
11        System.out.println("method 1 is "+Thread.currentThread().getId());
12        Assert.assertTrue(false);
13    }
14    @Test
15    public void test2() {
16        System.out.println("method 2 is "+Thread.currentThread().getId());
17        Assert.assertTrue(false);
18    }
19    @Test
20    public void test3() {
21        System.out.println("method 3 is "+Thread.currentThread().getId());
22    }
23    @Test
24    public void test4() {
25        System.out.println("method 4 is "+Thread.currentThread().getId());
26    }
27 }
28 @Test
29 public void test5() {
30     System.out.println("method 5 is "+Thread.currentThread().getId());
31 }
32 }
33 }
```

Output:

```
<terminated> TestNg_testng.xml [TestNG] C:\Program Files\Java\jre-10.0.2\bin\javaw.exe (19-Sep-2018, 4:53:19 PM)
```

```
[[RemoteTestNG] detected TestNG version 6.7.0
```

```
[TestNG] Running:
```

```
C:\Users\User\eclipse-workspaceNisha\TestNg\testng.xml
```

```
method 5 is 17
```

```
method 1 is 13
```

```
method 4 is 16
```

```
method 3 is 15
```

```
method 2 is 14
```

```
method 1 is 13
```

```
method 1 is 13
```

```
method 1 is 13
```

```
method 2 is 14
```

```
method 2 is 14
```

```
method 2 is 14
```

```
=====
```

```
Suite
```

```
Total tests run: 11, Failures: 8, Skips: 0
```

```
=====
```

Here All the failed test are executed 3 times.