

FrameWork

We are using POM based hybrid framework

POM stands for page object model

POM is a design pattern

In our framework we arrange our code in page wise class

In POM we create object repository to keep our WebElement type variable

In my framework we create four layer

1 Utility Layer(Generic layer)

It is also known as generic layer .In generic layer we have to create some customized method like as click(), clear(), switchtowindow(), popup handle(), sendkeys() etc.

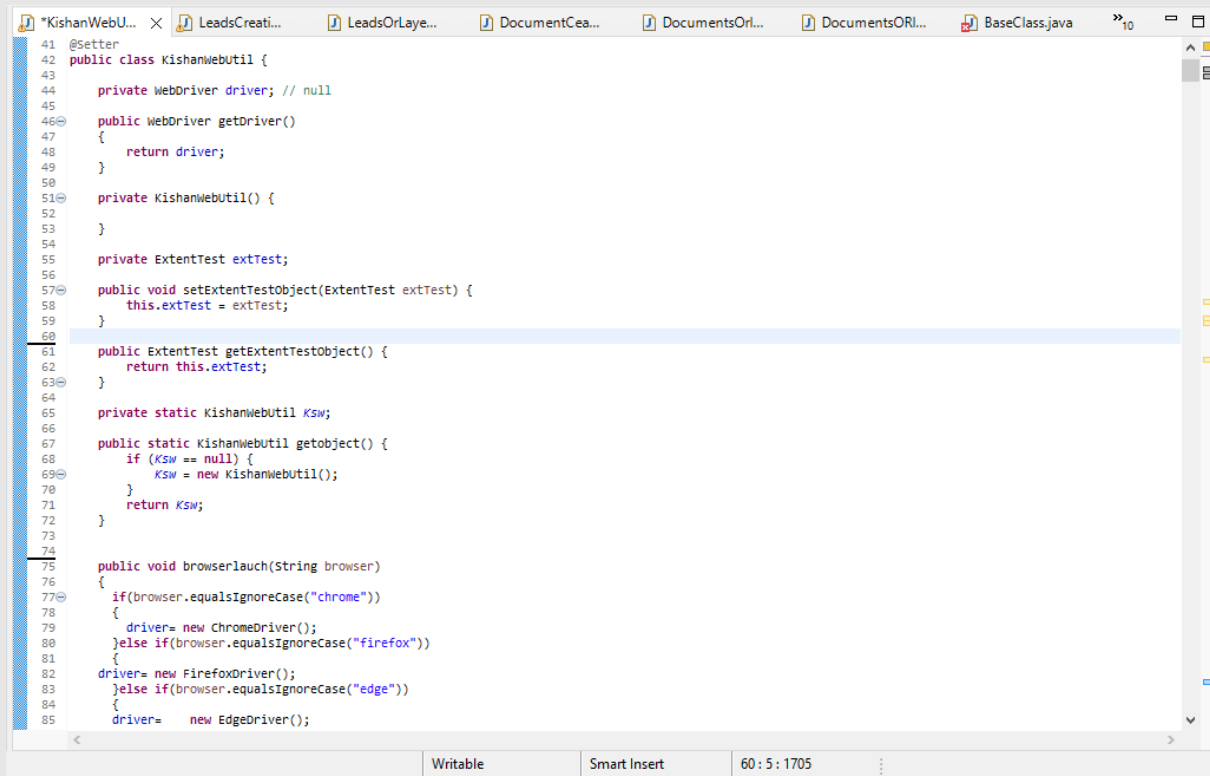
In our utility layer all method are independent

We have to use try and catch to handle the exception in generic layer

The main advantage of generic layer to achieve the reusability

In utility layer we use method overloading concept

Example ;-- with actions class method, Iframe, explicit wait,



```
41 @Setter
42 public class KishanWebutil {
43
44     private WebDriver driver; // null
45
46     public WebDriver getDriver()
47     {
48         return driver;
49     }
50
51     private KishanWebutil() {
52     }
53
54     private ExtentTest extTest;
55
56     public void setExtentTestObject(ExtentTest extTest) {
57         this.extTest = extTest;
58     }
59
60     public ExtentTest getExtentTestObject() {
61         return this.extTest;
62     }
63
64
65     private static KishanWebutil KSW;
66
67     public static KishanWebutil getObject() {
68         if (KSW == null) {
69             KSW = new KishanWebutil();
70         }
71         return KSW;
72     }
73
74
75     public void browserlauch(String browser)
76     {
77         if(browser.equalsIgnoreCase("chrome"))
78         {
79             driver= new ChromeDriver();
80         }else if(browser.equalsIgnoreCase("firefox"))
81         {
82             driver= new FirefoxDriver();
83         }else if(browser.equalsIgnoreCase("edge"))
84         {
85             driver= new EdgeDriver();
86         }
```

OR Layer(Object Repository);

In the layer we store the data in variable of WebElement

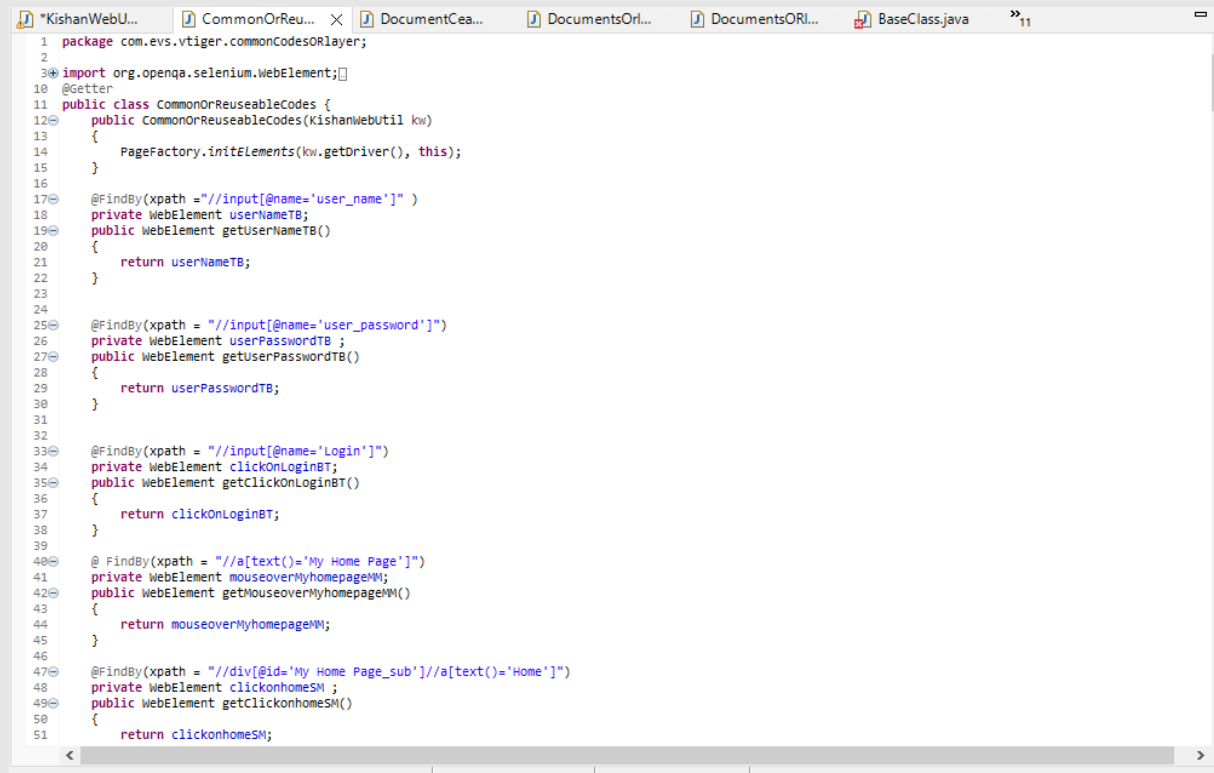
It is also known as collection of WebElement

In this we declare all the WebElement as private and for the each WebElement we create a getter method in getter method we return the WebElement type of variable to use in page wise classes.

We use annotation **@FindBy** an **@FindBy** to initialize the WebElement with the help of **PageFactory** class. We create constructor in OR layer

In constructor we create PageFactory .PageFactory is a class it has initElement static method it takes two parameter in first

parameter we pass **webUtil.driver** and second parameter we pass **this** keyword which will refers the current class object



```
1 package com.evs.vtiger.commonCodesORLayer;
2
3 import org.openqa.selenium.WebElement;
4
5 @Getter
6
7 public class CommonOrReusableCodes {
8     public CommonOrReusableCodes(KishanWebUtil kw)
9     {
10         PageFactory.initElements(kw.getDriver(), this);
11     }
12
13     @FindBy(xpath = "//input[@name='user_name']")
14     private WebElement userNameTB;
15     public WebElement getUserNameTB()
16     {
17         return userNameTB;
18     }
19
20     @FindBy(xpath = "//input[@name='user_password']")
21     private WebElement userPasswordTB;
22     public WebElement getUserPasswordTB()
23     {
24         return userPasswordTB;
25     }
26
27     @FindBy(xpath = "//input[@name='Login']")
28     private WebElement clickOnLoginBT;
29     public WebElement getClickOnLoginBT()
30     {
31         return clickOnLoginBT;
32     }
33
34     @FindBy(xpath = "//a[text()='My Home Page']")
35     private WebElement mouseoverMyhomepageMM;
36     public WebElement getMouseoverMyhomepageMM()
37     {
38         return mouseoverMyhomepageMM;
39     }
40
41     @FindBy(xpath = "//div[@id='My Home Page_sub']/a[text()='Home']")
42     private WebElement clickonhomeSM;
43     public WebElement getClickonhomeSM()
44     {
45         return clickonhomeSM;
46     }
47
48 }
49
50
51
```

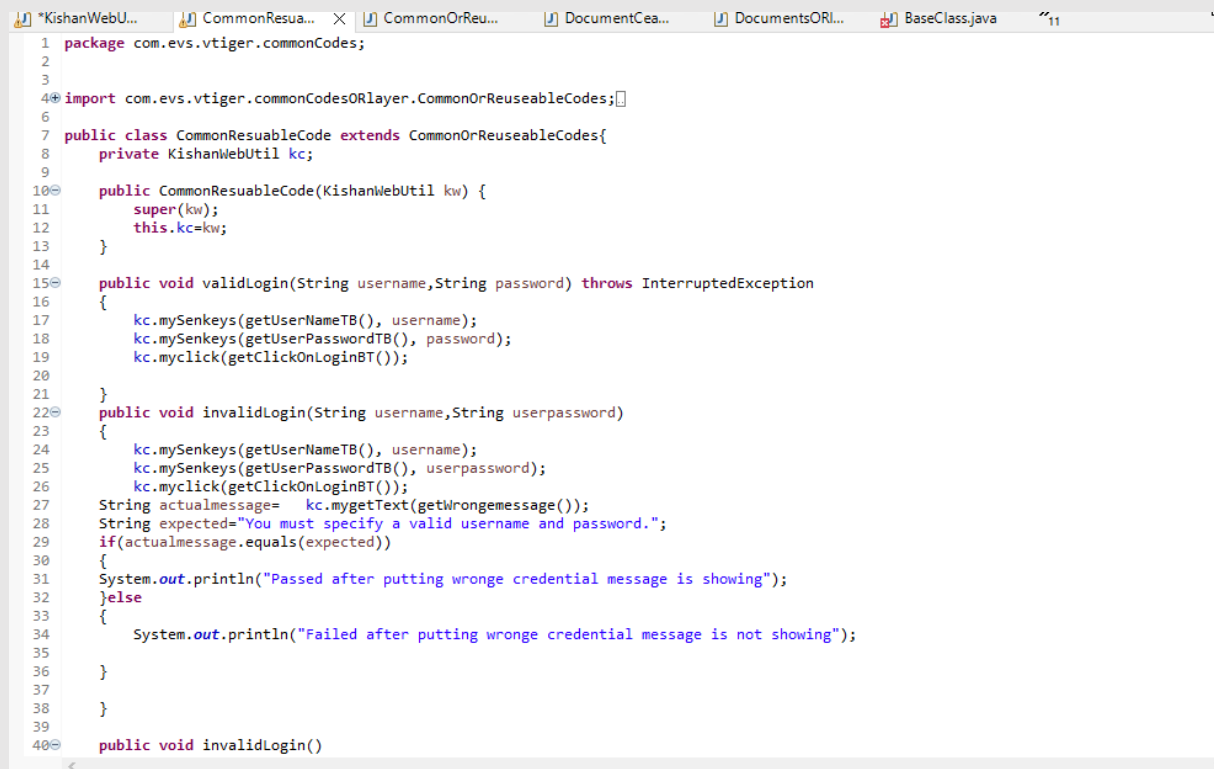
PageWise layer:-

If we talk about my third layer Page wise layer. In page wise layer we divide our project or task into page wise layer. For each page we create a class

Example: - If we talk about login page then we create a class for login page inside the class we create method for every functionality which is present in page wise class

In page wise class we call our generic method to automate our test case and we have to use extends keyword OR layer into page wise class

In page wise class we have to create parameter constructor to initialize our WebElement which is present in OR layer.



```
1 package com.evs.vtiger.commonCodes;
2
3
4 import com.evs.vtiger.commonCodesORLayer.CommonOrReuseableCodes;
5
6
7 public class CommonResuableCode extends CommonOrReuseableCodes{
8     private KishanWebUtil kc;
9
10    public CommonResuableCode(KishanWebUtil kw) {
11        super(kw);
12        this.kc=kw;
13    }
14
15    public void validLogin(String username,String password) throws InterruptedException
16    {
17        kc.mySenkeys(getUserNameTB(), username);
18        kc.mySenkeys(getUserPasswordTB(), password);
19        kc.myclick(getClickOnLoginBT());
20    }
21
22    public void invalidLogin(String username,String userpassword)
23    {
24        kc.mySenkeys(getUserNameTB(), username);
25        kc.mySenkeys(getUserPasswordTB(), userpassword);
26        kc.myclick(getClickOnLoginBT());
27        String actualmessage= kc.mygetText(getWrongemessage());
28        String expected="You must specify a valid username and password.";
29        if(actualmessage.equals(expected))
30        {
31            System.out.println("Passed after putting wronge credential message is showing");
32        }else
33        {
34            System.out.println("Failed after putting wronge credential message is not showing");
35        }
36    }
37
38    }
39
40    public void invalidLogin()
```

TestCase Layer:-

In test case layer we call our page wise layer to automate the test case

In test case layer we have created separate method for each and every running our test case we are using TestNG As we know that TestNG provide some method like as.

@before suite ,@after suite ,@before test ,@after test ,@before class,@after class, @before method ,@after method

@test

```

Run All
13 public class SuperAdmin extends BaseClass{
14     @Test( invocationCount = 10,priority = 1, enabled = true,retryAnalyzer = RetryAnalyzer.class)
Run | Debug
15     public void lm001employee() throws IOException
16     {
17         SuperAdminEmployeeCreation em= new SuperAdminEmployeeCreation(kw);
18         em.employeeCreation("Kishan@12345", "Kishan@12345");
19     }
20 }
21 @Test(invocationCount = 10,enabled = false,retryAnalyzer = RetryAnalyzer.class)
Run | Debug
22 public void uploadfile() throws InterruptedException
23 {
24     CommonResuableCode cd=new CommonResuableCode(kw);
25     cd.uploadfile("C:\\Users\\CBPC-09\\Pictures\\Saved Pictures\\logo2.png");
26 }
27 @Test(enabled = false,retryAnalyzer = RetryAnalyzer.class)
Run | Debug
28 public void uploadlogo()
29 {
30     CommonResuableCode cd= new CommonResuableCode(kw);
31     cd.uploadlogo("C:\\Users\\CBPC-09\\Pictures\\Saved Pictures\\download.jpg");
32 }
33 }
34 @Test(invocationCount = 1,enabled =false,retryAnalyzer = RetryAnalyzer.class)
Run | Debug
35 public void uploadprofile()
36 {
37     CommonResuableCode cd= new CommonResuableCode(kw);

```

Note:--

We have used OOPS concept also in our frame work like Data hiding,Polymorphism,Encapsulation,Inheritance

Inheritance--- :- We have used Inheritance in page wise classes. We extends our OR(Object Repository) layer into page wise layer.

Polymorphism:---:- We have use polymorphism concept in our util layer like we have used overloading.In Util layer we have to create some overloaded method like Handle frame ,with Actions class method, explicit wait etc.

Encapsulation:--

1 Data Hiding:--- In every classes we have declared our WebElement variable as private to achieve the Encapsulation concept and we create a getter method for each WebElement variable

Example:-- In Util layer my Webdriver variable is private and access that private variable outside of our class. We have created a getter method its name getdriver.

The beauty of my framework:---

Consume redundancy(duplicacy):--- If we are using same line of code again and again then we make a separate class and put the much line on code make it reusable.

Just for example:--- We have created a WebUtil class in that we kept our some util or generic method and we call those method in every page wise and test case classes. Instant of writing code again and again

Reusability---:- In our framework we are using reusability concept in our util class and page wise class.

Traceability---:- In any bug there update Accor(come) in our script then we make changes easily through going on that page because we have to used try catch exception in our util layer which is make it traceability

Understandability:--- It is easy to understand for new programmer.They look our code they can understand our code easily.

Maintainability--:- It is easy to maintain because we have created in page wise.

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:

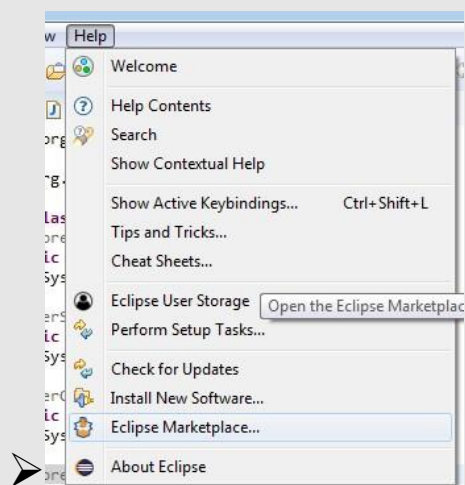
- It gives the ability to produce HTML Reports of execution
- Annotations made testers life easy
- Test cases can be Grouped & Prioritized more easily
- Parallel execution is possible
- Generates Logs

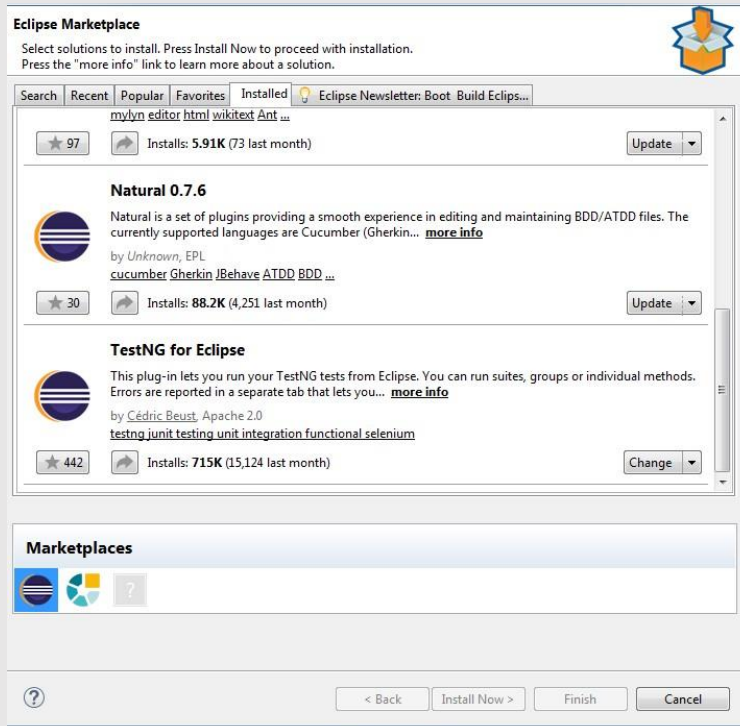
- Data Parameterization is possible
- Automatically return the failure test case

STEPS:

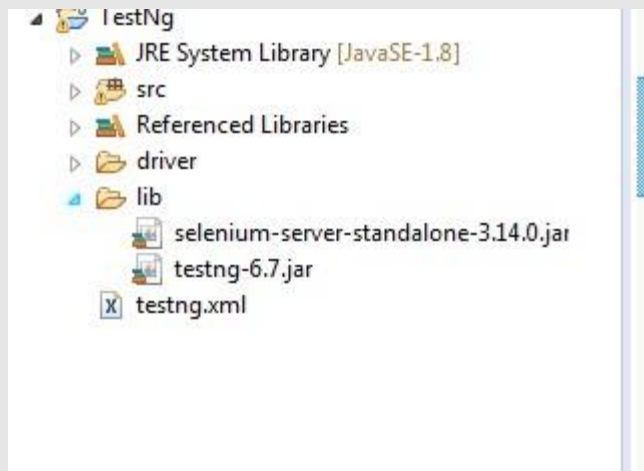
- Download the testing jar
- Add the testing jar in the eclipse build path
- Add the testing plugin in eclipse marketplace

Go to eclipse market place and install testing





Add the testing jar file and configure



We have to click run as testng test

Annotations in TestNG:

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

@After Suite: 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.

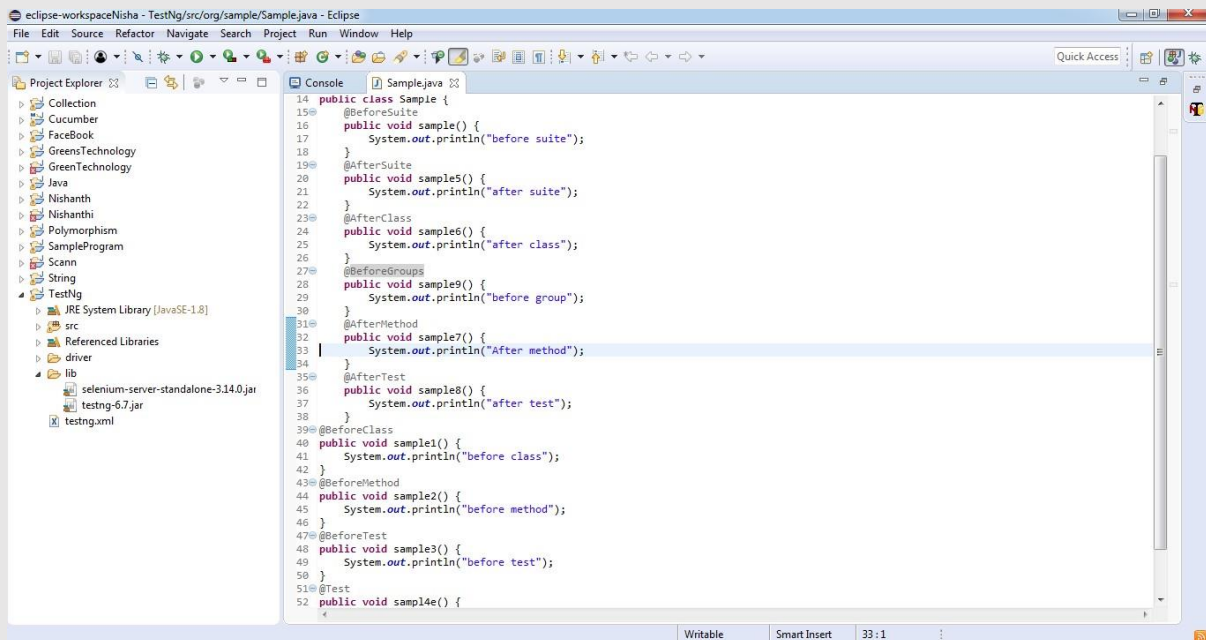
@BeforeMethod: The annotated method will be run before each test method.

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

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

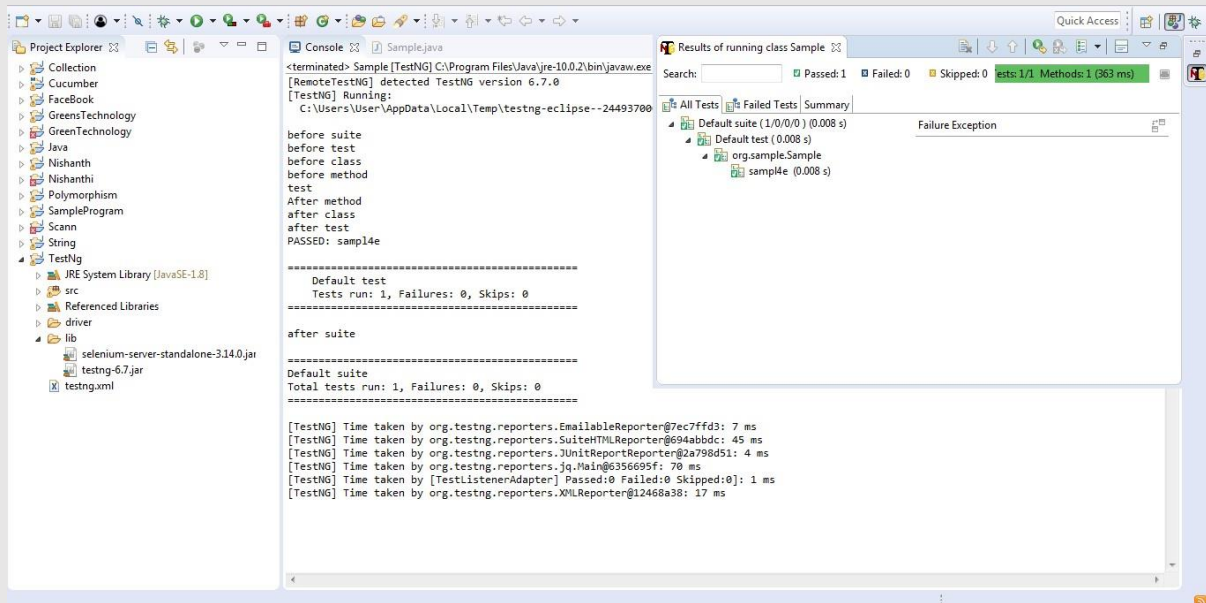
Ordered in which the annotation execute:

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() {
53     }
54 }
```

Output



```
<terminated> Sample [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--24493700

before suite
before test
before class
before method
test
After method
after method
after class
after test
PASSED: sample4

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

after suite

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

[TestNG] Time taken by org.testng.reporters.EmailableReporter@7ec7ffd3: 7 ms
[TestNG] Time taken by org.testng.reporters.SuiteHTMLReporter@694abdc: 45 ms
[TestNG] Time taken by org.testng.reporters.JUnitReportReporter@2a798d51: 4 ms
[TestNG] Time taken by org.testng.reporters.JqMain@6356695f: 70 ms
[TestNG] Time taken by [TestListenerAdapter] Passed:0 Failed:0 Skipped:0: 1 ms
[TestNG] Time taken by org.testng.reporters.XMLReporter@12468a38: 17 ms
```

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:

Consoletestng.xmlPriority.java

<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
=====

Results of running class Priority

Search: Passed: 5 Failed: 0 Skipped: 0ests: 1/1 Methods: 5 (344 ms)

All TestsFailed TestsSummary

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)

Failure Exception

=====

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

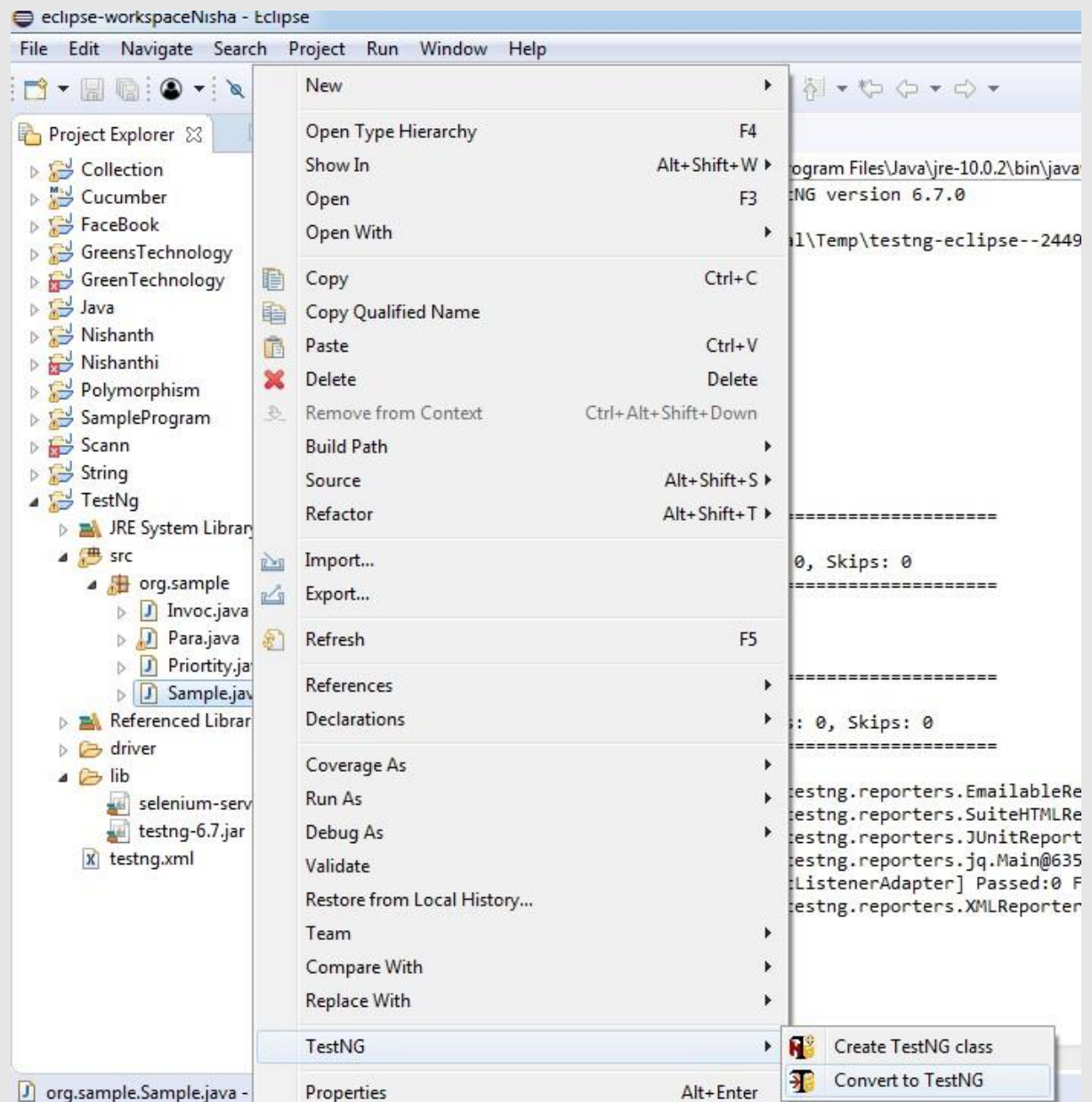
[TestNG] Time taken by org.testng.reporters.EmailableReporter@7ec7ffd3: 0 ms
[TestNG] Time taken by org.testng.reporters.SuiteHTMLReporter@694abddc: 47 ms
[TestNG] Time taken by org.testng.reporters.JUnitReportReporter@2a798d51: 47 ms
[TestNG] Time taken by org.testng.reporters.jq.Main@6356695f: 93 ms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

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

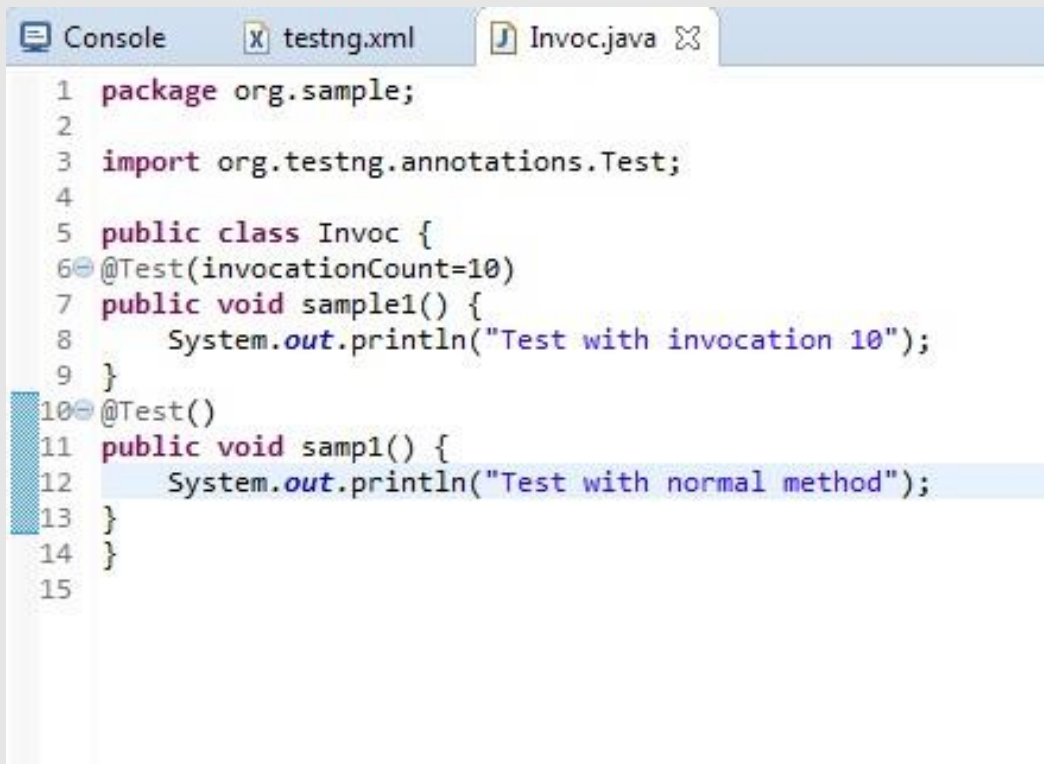
A screenshot of an IDE window with three tabs: 'Console', 'Sample.java', and 'testng.xml'. The 'testng.xml' tab is active, showing the following XML code:

```
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
```

INVOCATIONCOUNT:

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

Program:

A screenshot of an IDE window with three tabs: 'Console', 'testng.xml', and 'Invoc.java'. The 'Invoc.java' tab is active, showing a Java class named 'Invoc' in the 'org.sample' package. The class has two methods: 'sample1()' annotated with '@Test(invocationCount=10)' and 'samp1()' annotated with '@Test()'. The 'sample1()' method prints 'Test with invocation 10' and 'samp1()' prints 'Test with normal method'.

```
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 x testng.xml J 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

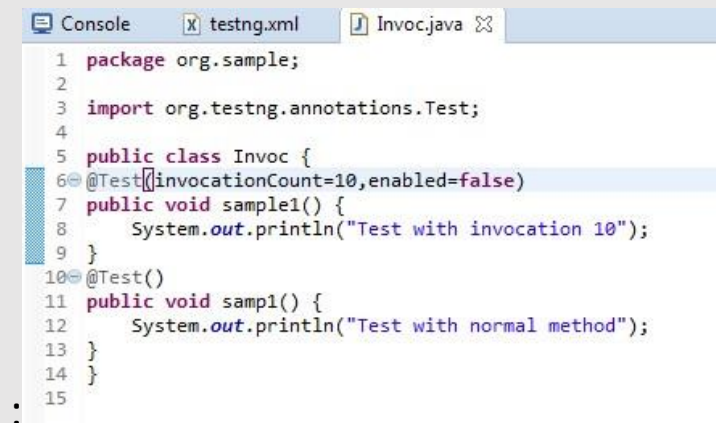
Test with normal method
Test with invocation 10
Test with invocation 10
Test with invocation 10
Test with invocation 10
Test with invocation 10
Test with invocation 10
Test with invocation 10
Test with invocation 10
Test with invocation 10
Test with invocation 10
Test with invocation 10
PASSED: sample1
PASSED: sample1
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
=====
```

IGNORINGTHETESTCASE:

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



The screenshot shows a Java IDE with three tabs: 'Console', 'testng.xml', and 'Invoc.java'. The 'Invoc.java' tab is active, displaying the following code:

```
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:

```
Console testng.xml Invoc.java
<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

```
Console  testng.xml  Invoc.java  FacebookLogin.java  testng.xml
1 package org.sample;
2
3 import org.testng.annotations.Parameters;
4 import org.testng.annotations.Test;
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     @Test()
14     @Parameters({"UserName","Password"})
15     public void sampl(String name,String pass) {
16         System.out.println("Test with normal method");
17         System.out.println("the second test name is "+name);
18         System.out.println("the second test pass is "+pass);
19     }
20 }
21
```

Results of running suite

```
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">
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  testng.xml  Invoc.java  FacebookLogin.java  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 }
22 }
```

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.

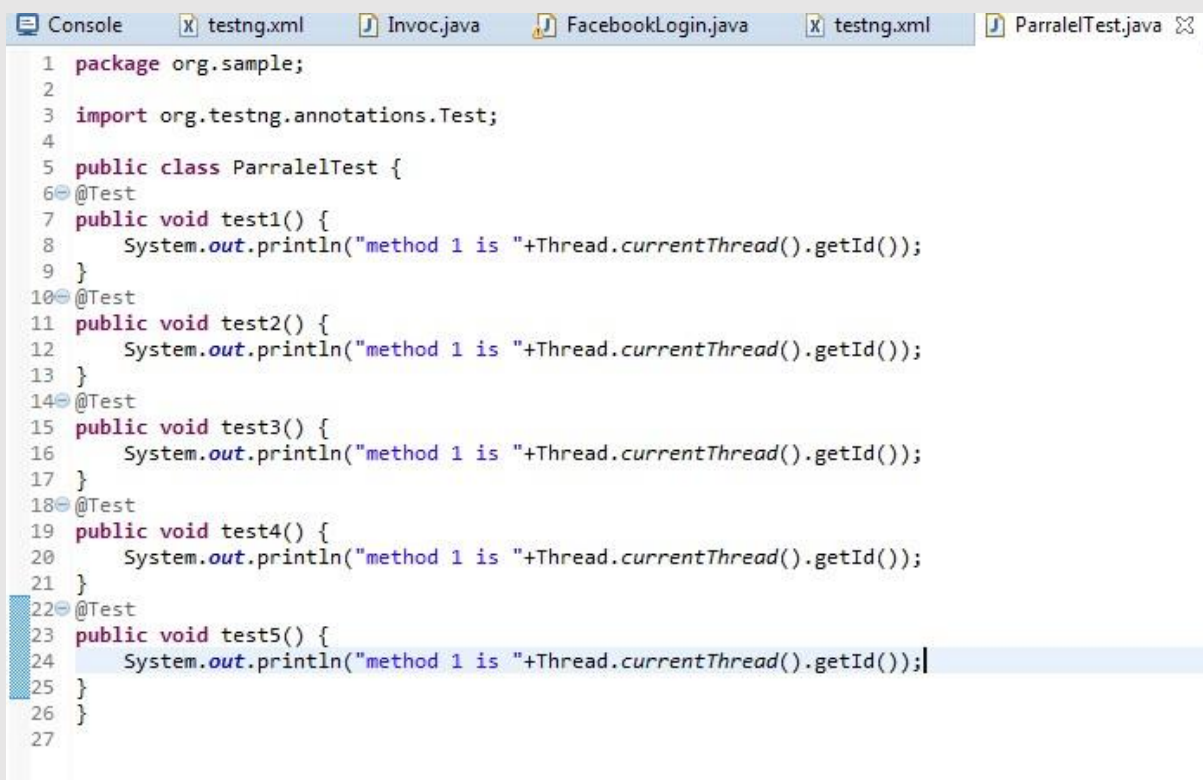
MultiThread-more than one person will try to execute all the function in the program parallel. Default thread count is 5.

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

Multi posing test:

Run the test cases in many browser is called multi posing test.

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



The screenshot shows an IDE window with the following tabs: Console, testng.xml, Invoc.java, FacebookLogin.java, testng.xml, and ParallelTest.java. The code in ParallelTest.java is as follows:

```
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 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 -->

```

```

<terminated> TestNg_testng.xml [TestNG] C:\Program Files\Java\jre-10.0.2\bin\java.exe
[RemoteTestNG] detected TestNG version 6.7.0
[TestNG] Running:
  C:\Users\User\eclipse-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:](#)

```
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 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 samp1(@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 -->
14 </suite> <!-- Suite -->
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 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 <?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_testing.xml [TestNG] C:\Program Files\Java\jre-10.0.2\bin\java.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 sampl1(@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 isvel2215  
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 test 2 is skipped.

Groups:

We can group the multiple test cases by using groups concept

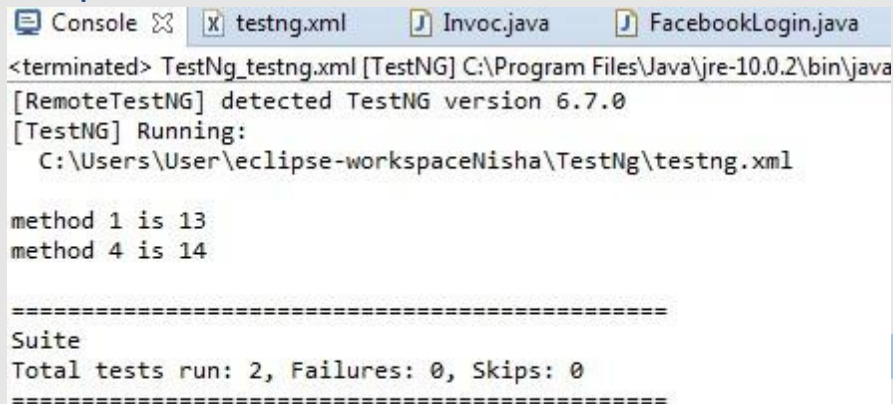
We have to give groups name in the @test annotation

@Test(groups="group name")

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

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 }
```

```
Console  RetryFailed.java  testng.xml  ParralelTest.java
<terminated> TestNg_testng.xml [TestNG] C:\Program Files\Java\jre-10.0.2\bin
[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
=====
```

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 I Annotation

Transformer Program for I Annotation Transformer:

```
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
```

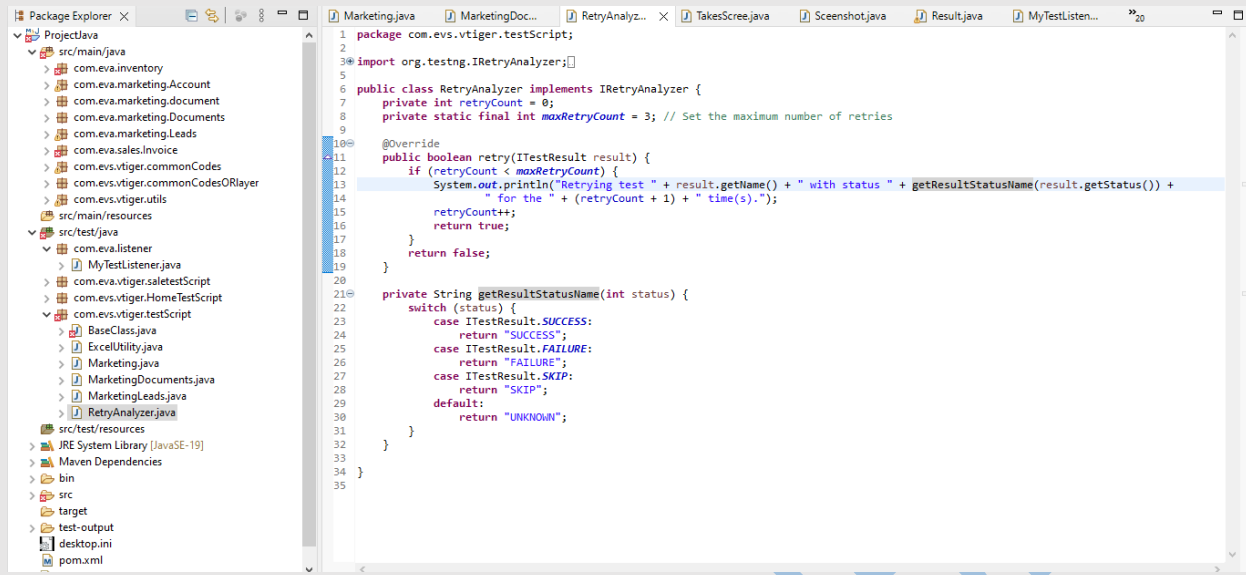
Output:

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

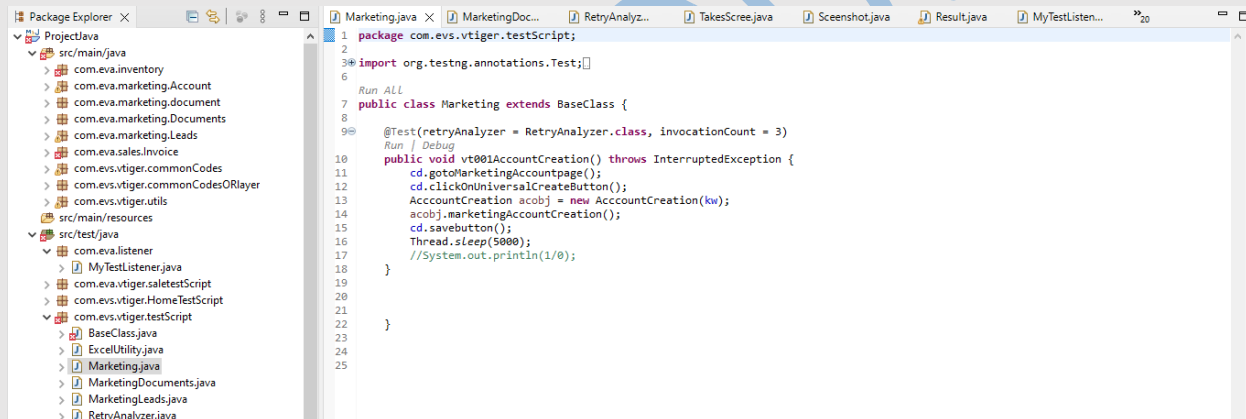
Listener

The screenshot shows an IDE with the Project Explorer on the left and the Source Editor on the right. The Project Explorer displays a project named 'ProjectJava' with a package structure including 'com.eva.listener'. The Source Editor displays the code for 'MyTestListener.java', which implements the 'IEventListener' interface. The code includes imports for 'org.testng.ITestContext' and 'org.testng.ITestResult', and implements methods like 'onTestStart', 'onTestSuccess', 'onTestFailure', 'onStart', and 'onFinish'.

2 step



3 step



4 step



Excel Data Read

