**Unit Testing Framework Tool**

**Available Unit testing framework tools in Current Market:**

TestNG ------------> Java, .Net [Eclipse / JDevelopers / Intellj]

Junit -------> Java [Eclipse / JDevelopers / Intellj]

Nunit ---------> .Net [Visual Studio]

Pydev --------> Python [PyChram]

Rspsc ----------> Ruby [Eclipse]

Jasmin 🡺 javascript [WebStrom]

All unit testing framework tool is implemented as plugin for eclipse IDE, but Junit is a default plugin for eclipse IDE.

**What is TestNG?**

🡺TestNG is a unit test **TDD** [test Driven Development] framework, which support java & .Net

**🡺**TestNG is an open source unit test framework, where NG stands for **Next Generation**.

🡺TestNg developed as addition plugin for Eclipse

🡺TestNG is inspired from JUNIT & NUNIT, it means it has all the feature of Nunit & Junit & also contains additional features that makes TestNg more powerful.

**Installation steps of TestNG:**

Go to Eclipse window

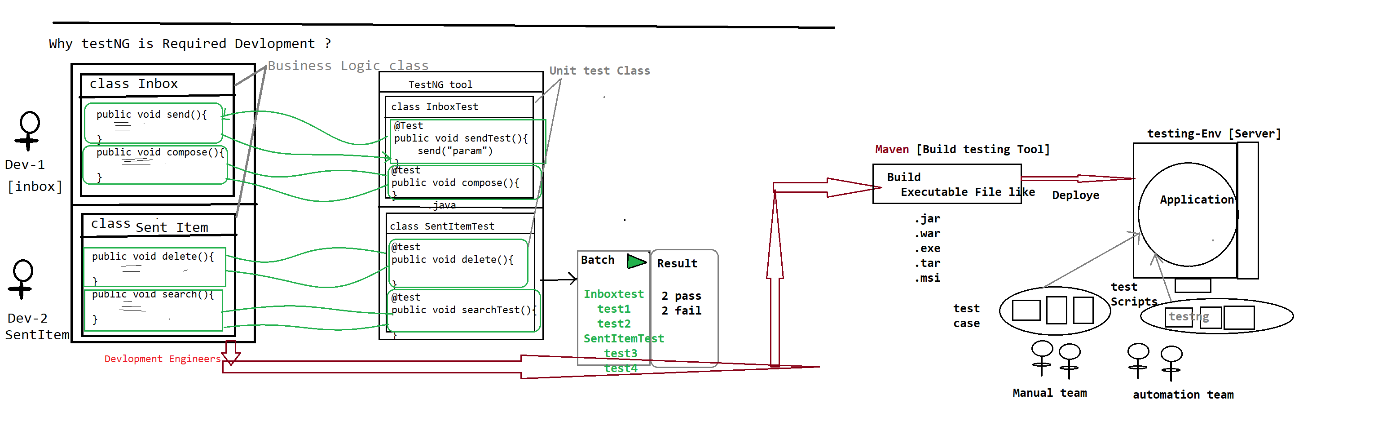
Click on help option-> Eclipse Marketplace

* Write TestNG in find edit box and click on go button.
* Find TestNG for eclipse division and click on **install** button
* Click on confirm button and I accept the terms and conditions and click on finish
* **In order to verify the** TestNG installation- 🡪 Go to windows

🡪 Show view 🡪 others 🡪expand java folder, TestNG symbol will be present

**TestNG usage in development:**

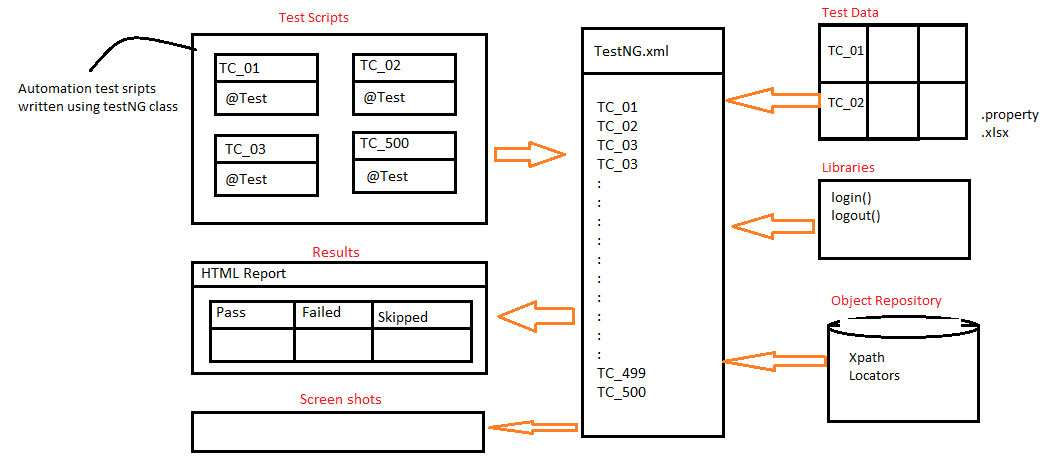
TestNG is used in development to write white box / unit test cases and each unit test case will be used to test the source code of the application



🡺In case of development, testNG will be used to develop unit test cases and each unit test case check the business logic of the source code.

🡺Used to achieve WBT

**TestNG usage in selenium automation**

****

🡺In case of automation, testNG will be used to develop all the scripts using testNG annotations and achieve batch execution without any manual interaction.

🡺TestNG will be used to handle all framework component & help us to run all the test scripts in batch / parallel / group without any manual intervention

🡺TestNG.xml is main controller of the selenium framework, where we start the execution

**Why TestNG? Why not Junit?**

* **Annotation**
* **Batch Execution**

**New functionalities are:**

->Html report

->Parallel execution

->Grouping execution

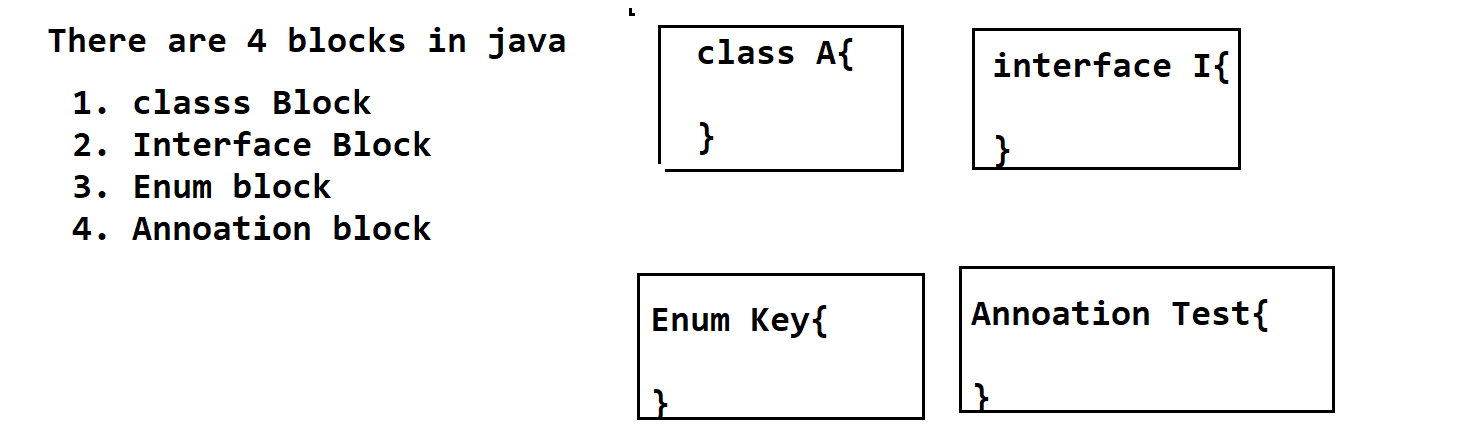
->Additional annotations

->Batch execution is easier

-> iTest Listeners [used to take Screenshot]

->Retry Analyser [used to rerun the failed test script]

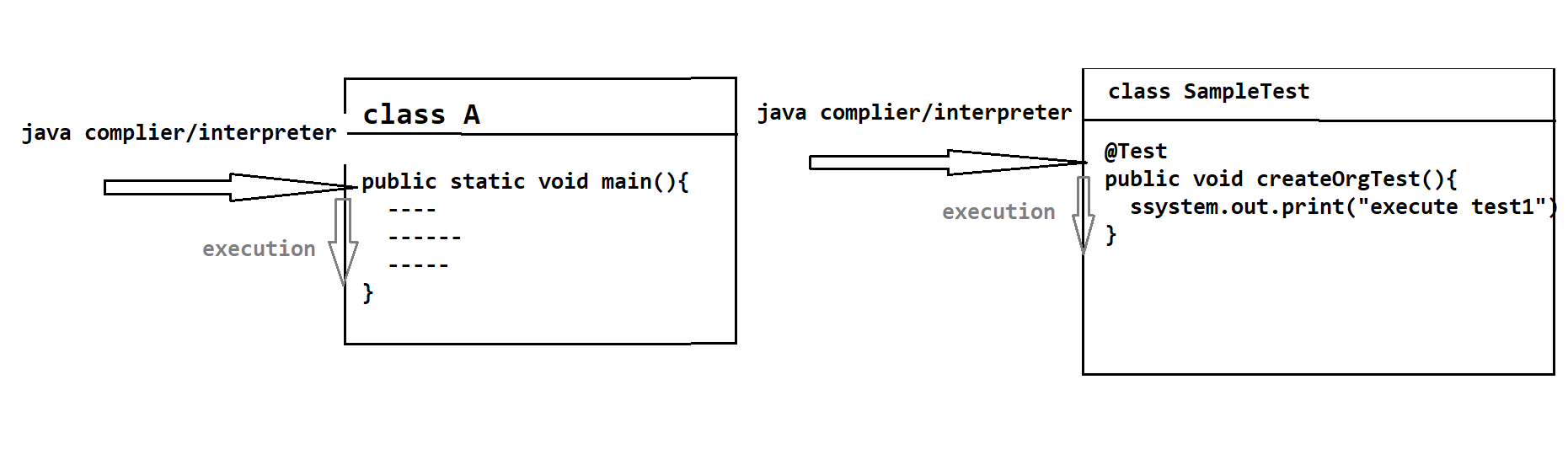
**Annotations:**

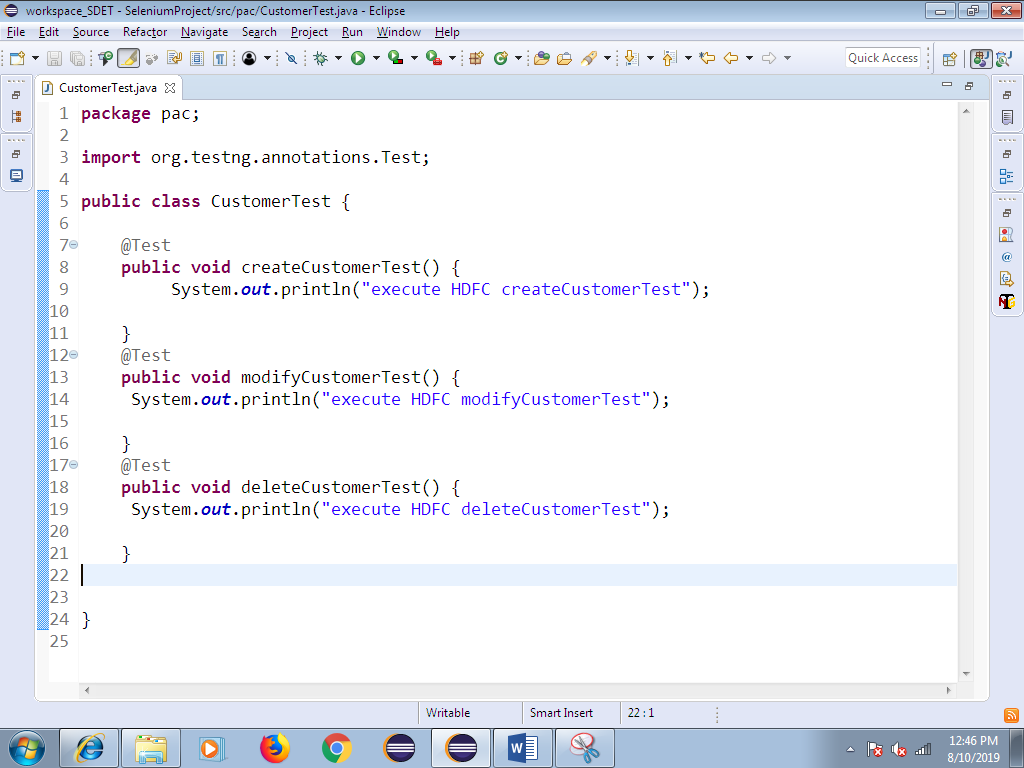
****

Its Java block, which is used to provide metadata (information/instruction) to the JVM, at the time of execution in Run-Time.

* Annotation always start with @symbol
* @Test
* @BeforeMethod
* @AfterMethod
* @BeforeClass
* @AfterClass
* @BeforeTest
* @AfterTest
* @paramaters
* @DataProvider
* @Listener
* ===================

**@Test**

****



1. Whenever we execute TestNG class, Java compiler / Interpreter always looks for @Test Annotation method to start the execution.
2. **Without @Test, TestNG class will not be executed, @Test annotation method act like main method in testNG**
3. In one TestNG class we can have multiple @Test methods, but each test method should have @Test annotation before method signature.
4. Annotation method return type should be “void” and access specifier should be public., but method name can be anything.
5. As per the Rule of the Automation, TestNG class Name should be Module Name, @Test method name should be manual test case name.
6. As per the Rule TestNG class Name & testNG method Name should end with “Test”
7. One Manual test case contains multiple steps all those steps should be automated using one @Test annotation & test name should be manual test-case Name

**How to Verify html report?**

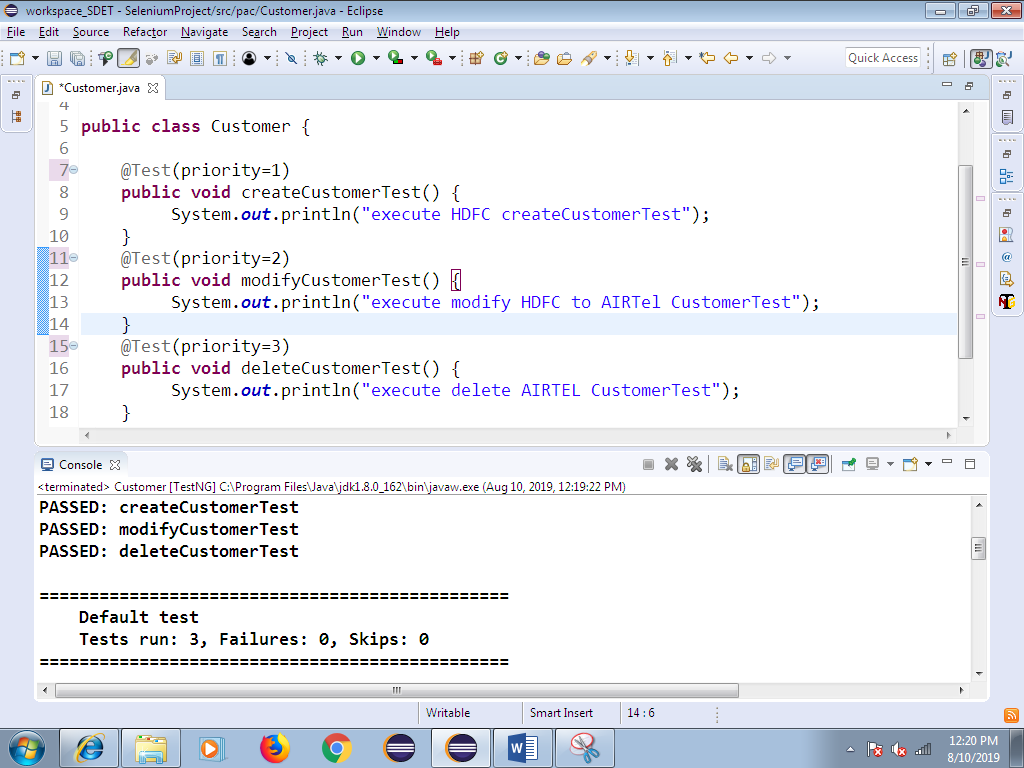
Refresh the project after execution—select project right click and click on refresh

Automatically we get **test-Output** folder within the same project.

Expand test-output folder 🡪 select emailable.html and right click 🡪 open with🡪 Open with browser

**Priority**

Whenever we execute TestNG class, by default all the test method will be executed based on alphabetical order, in order the change the Order of Execution, we go for priority

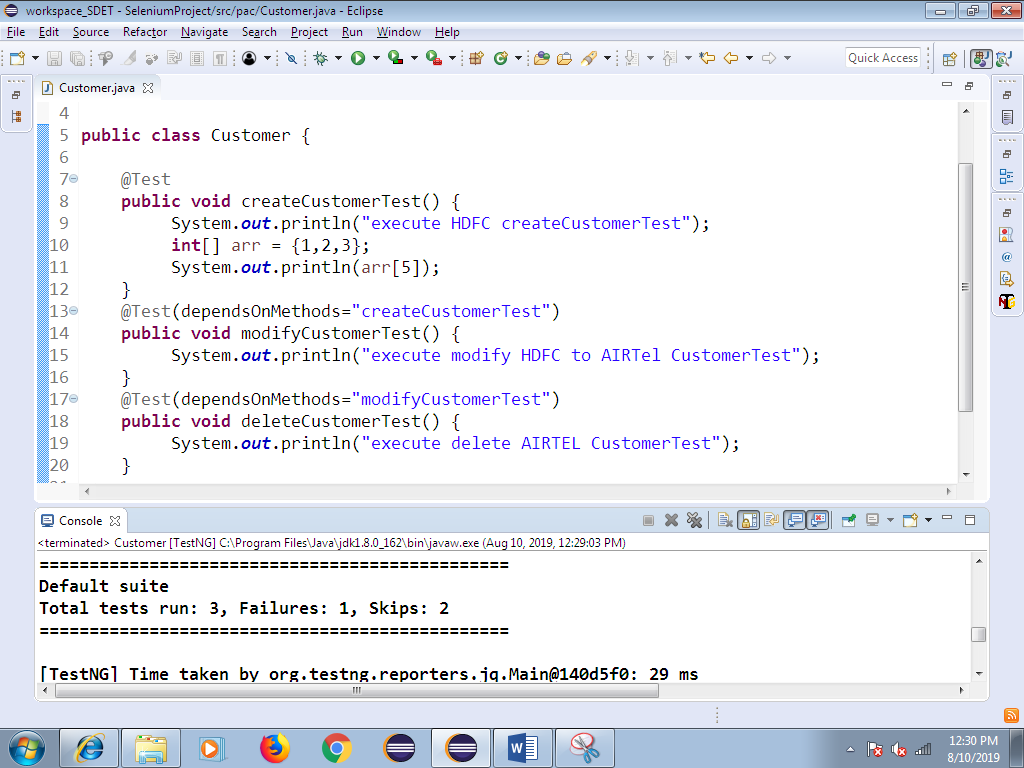


**DependsOnMethod:**

Its help us to check the dependent test case is pass or fail.

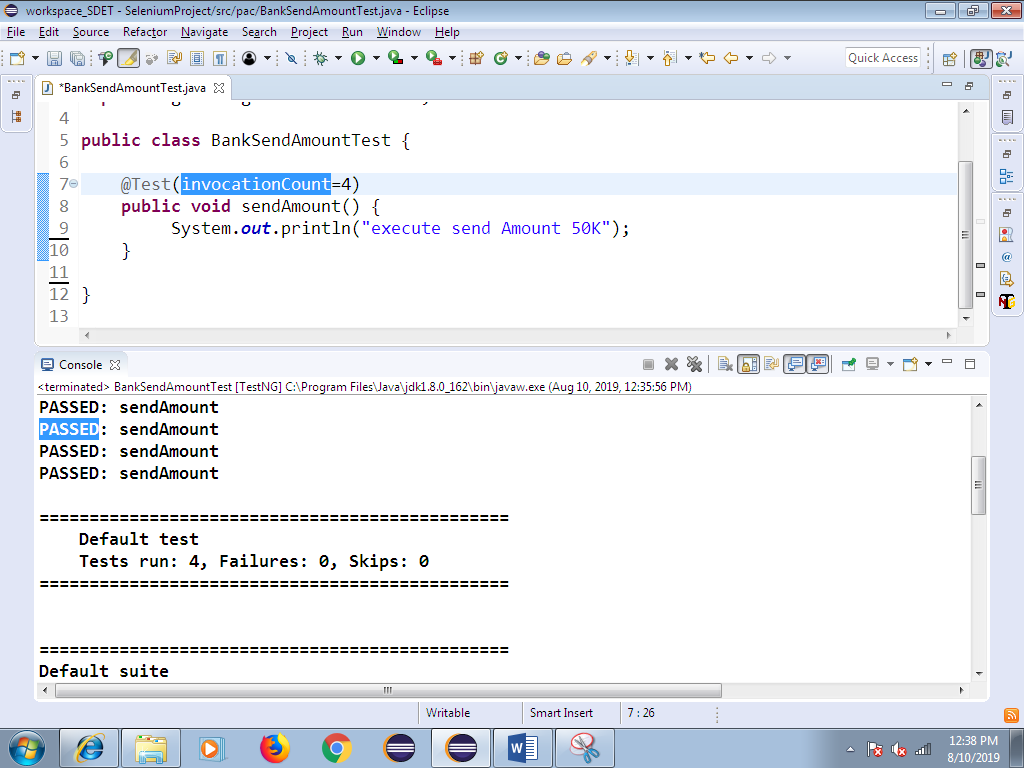
If dependent test-script gets pass, execution will continue.

If dependent test-script gets fail, it will skip the test execution.



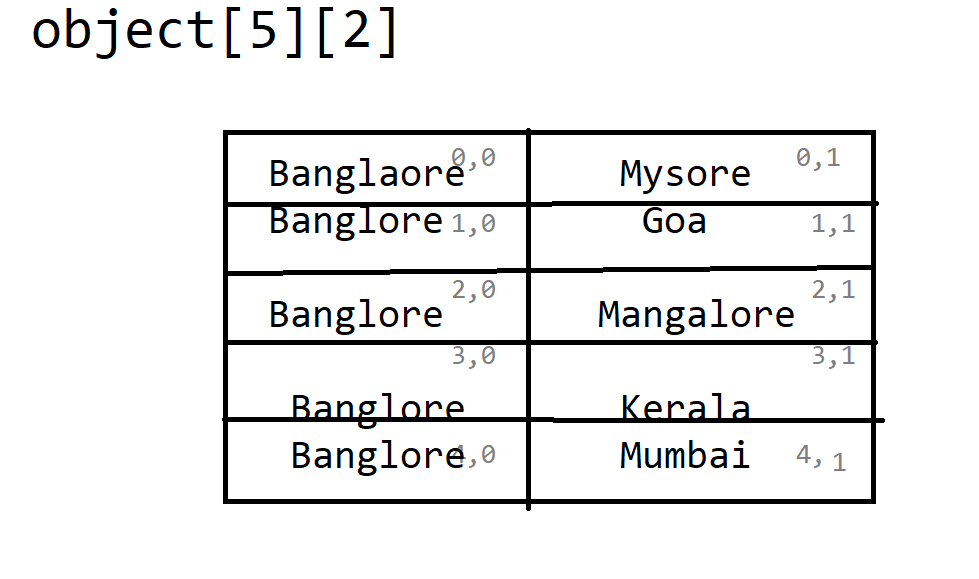
**10. Invocation Count:**

Same test-script executed with multiple Times with same test data

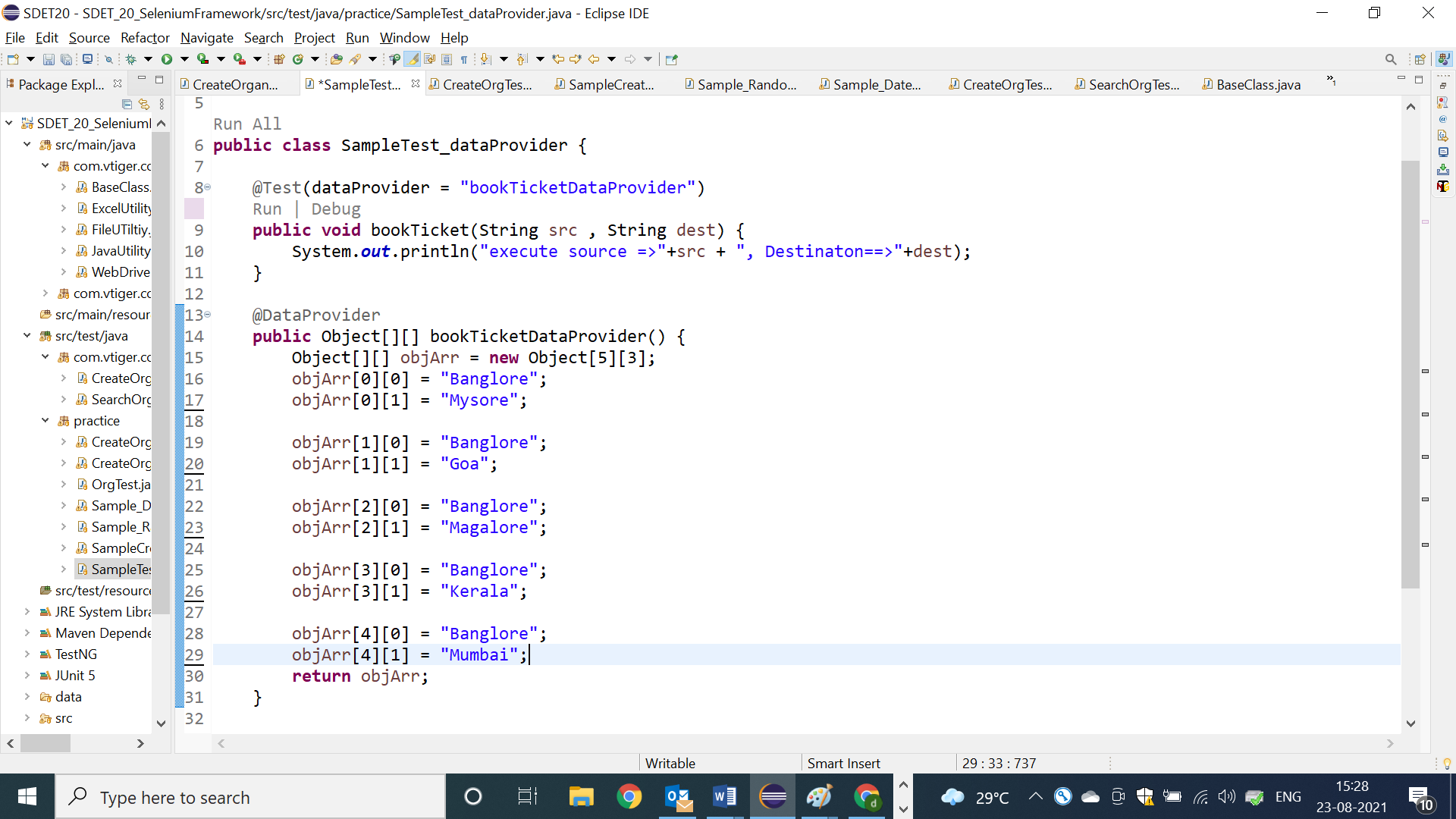


1. **Data Provider**
2. In Order to execute same test case multiple Times with different test Data, we go for @DataProvider annotation.
3. Data Provider annotation always return TWO –DIMENSTINAL Object array.
4. Data Provider annotation helps us to execute same test multiple times with the different set of data, each test-script should have dedicated @DataProvider annotation**.**
5. DataProvider annotation play major role in Data driven framework, where we need to test the application with huge amount of data like Ecommerce, banking application.
6. In Below example, row count is 5 🡺 it indicates test needs to be executed 5 times

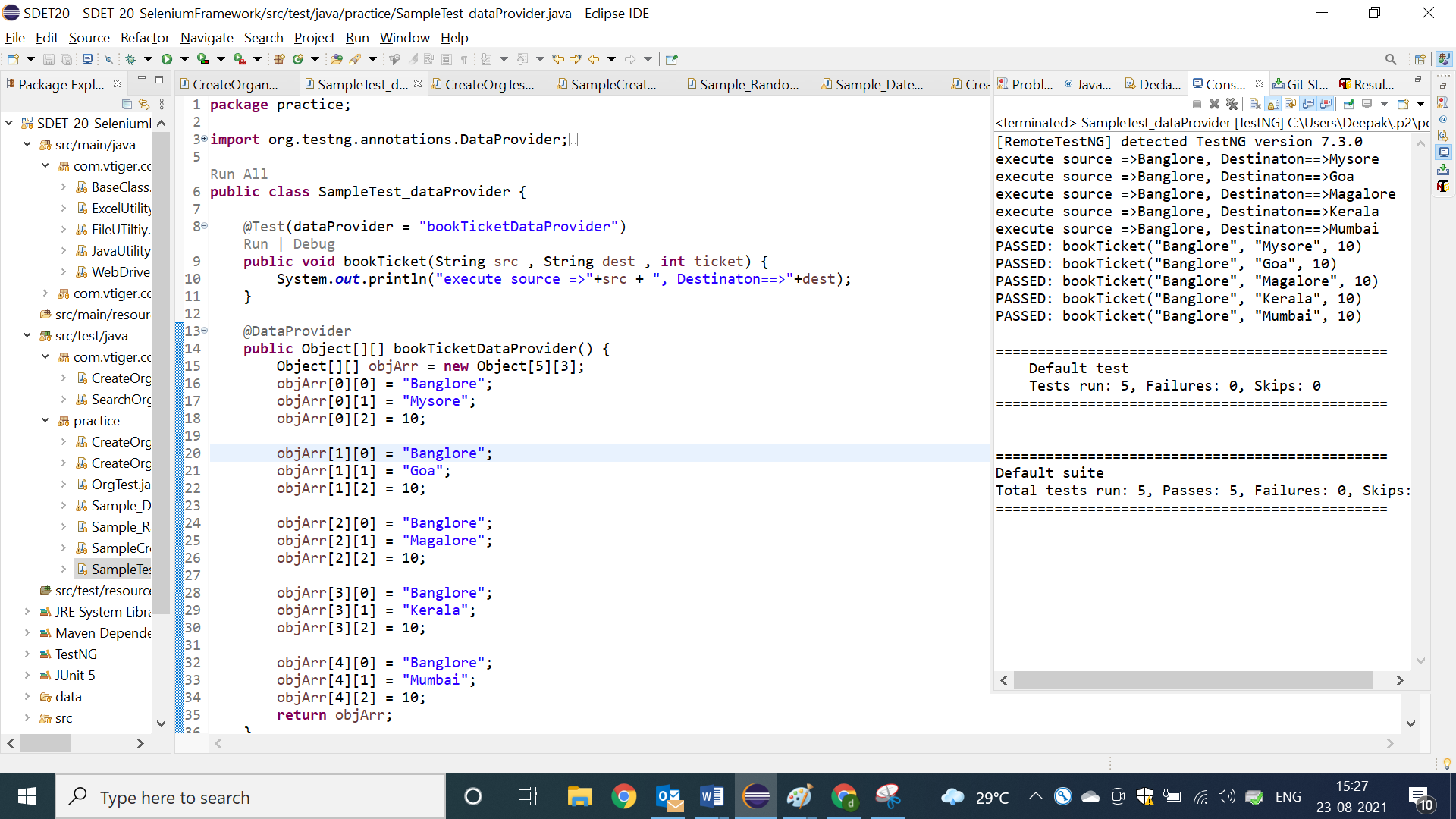
Column count is 2🡺 it indicates every iteration 2 arguments will be passed



**🡺Sample program for data provider**

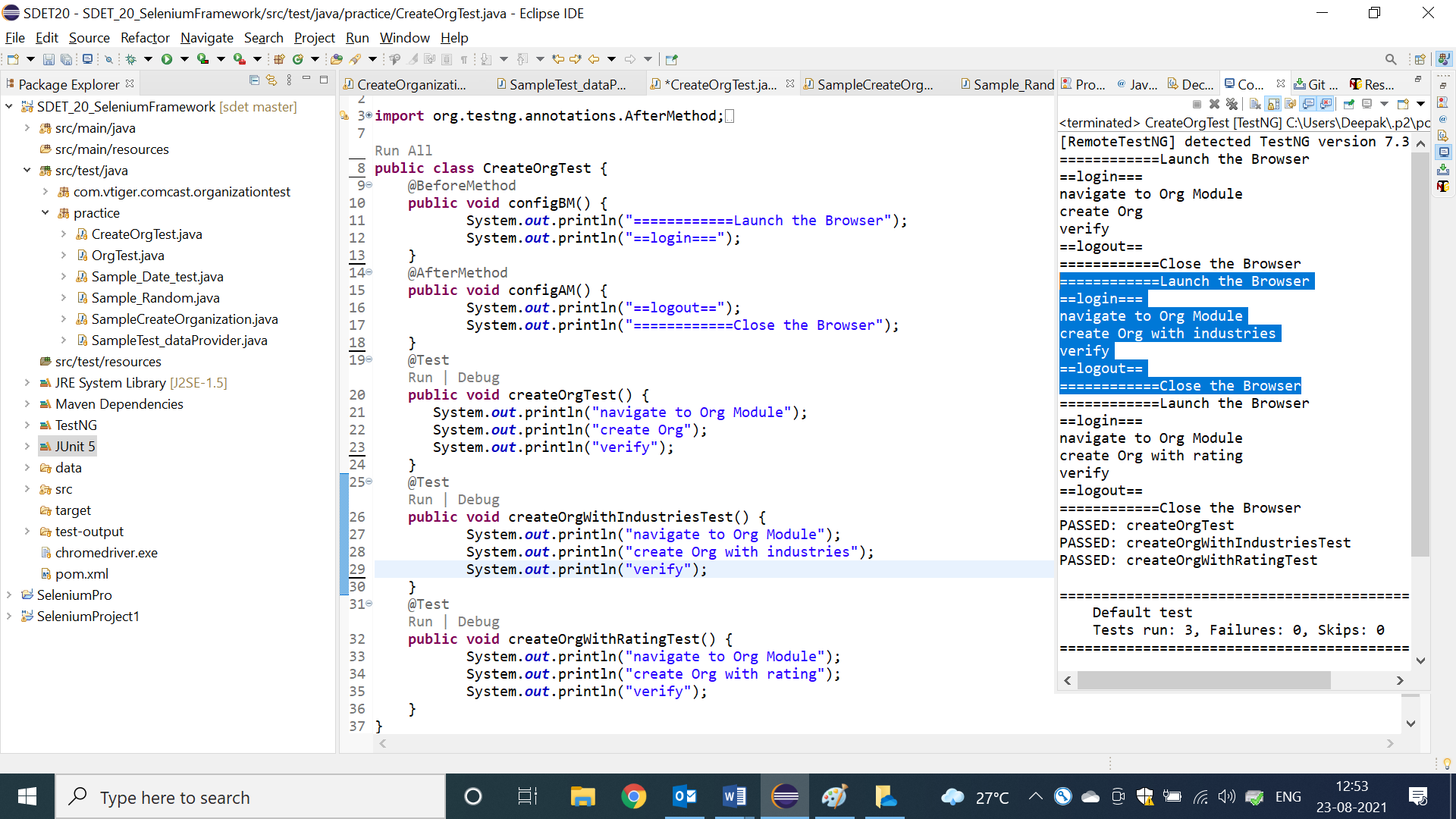


**🡺Data provider example with 3 arguments**

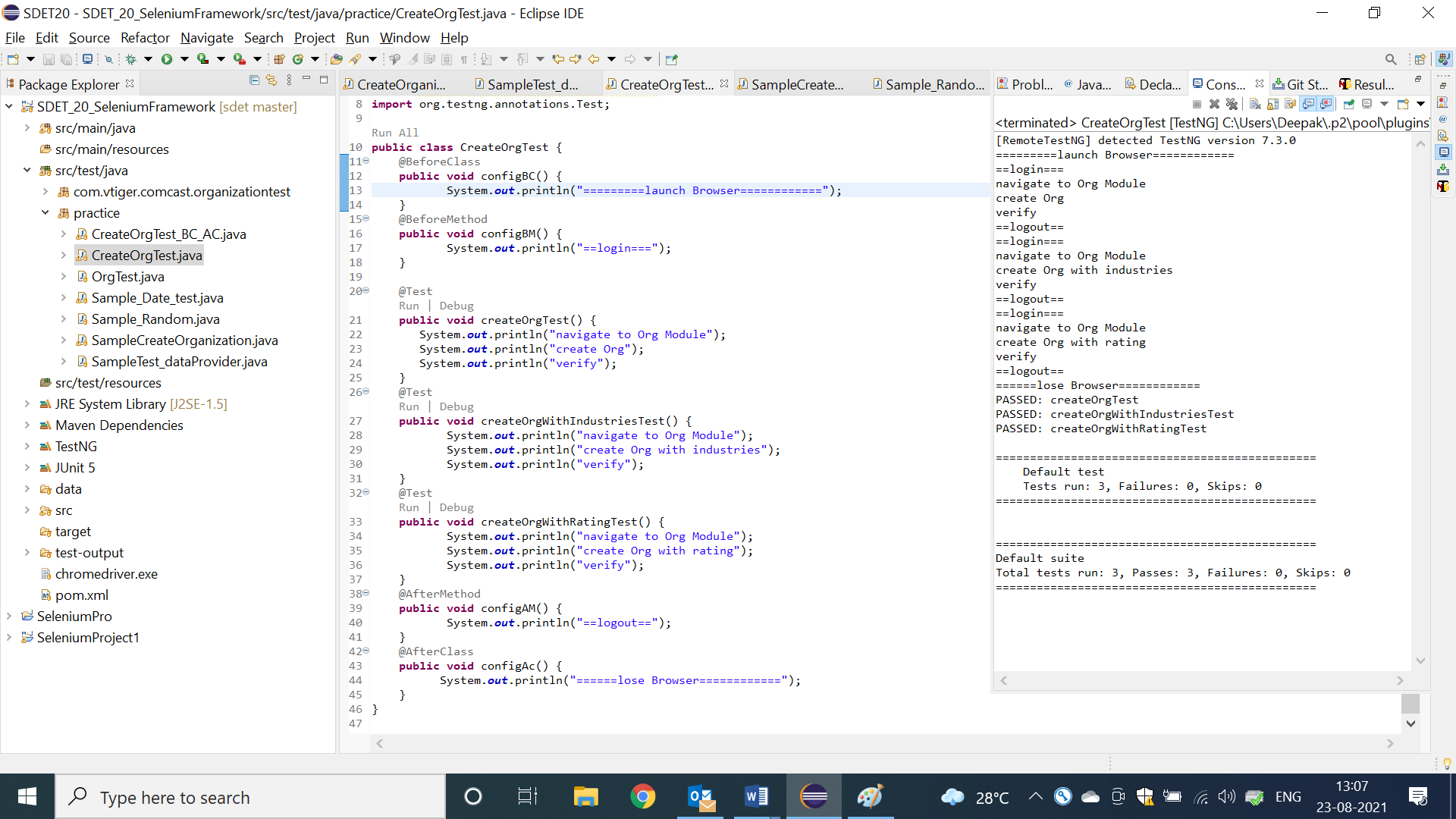


**@BeforeMethod @AfterMethod**

1. Before method annotations will be executed, before executing each @Test method in a class
2. After method annotation will be executed, after executing each @Test in a class
3. BeforeMethod & AfterMethod will not be executed, without @Test annotation method, because they are configuration method.
4. In order to implement similar pre-condition for all the test case like “LOGIN code” we go for @BM
5. In order to implement similar post-condition for all the test case like “LOGOUT code” we go for @AM



**@BeforeClass & @AfterClass**



1. BeforeClass Annotation method will be executed, before Executing first @Test in a class.
2. AfterClass Annotation method will be executed, after executing all/last test-case with in a class.
3. BeforeClass & AfterClass annotations will be executed only once in an entire class execution.
4. It will be used to develop global configuration like launch browser, object initialization

NOTE: As per the Rule of the Automation, test case should not have dependency between one to another test, every test case should be unique (it means every test should have fresh login & logout code)

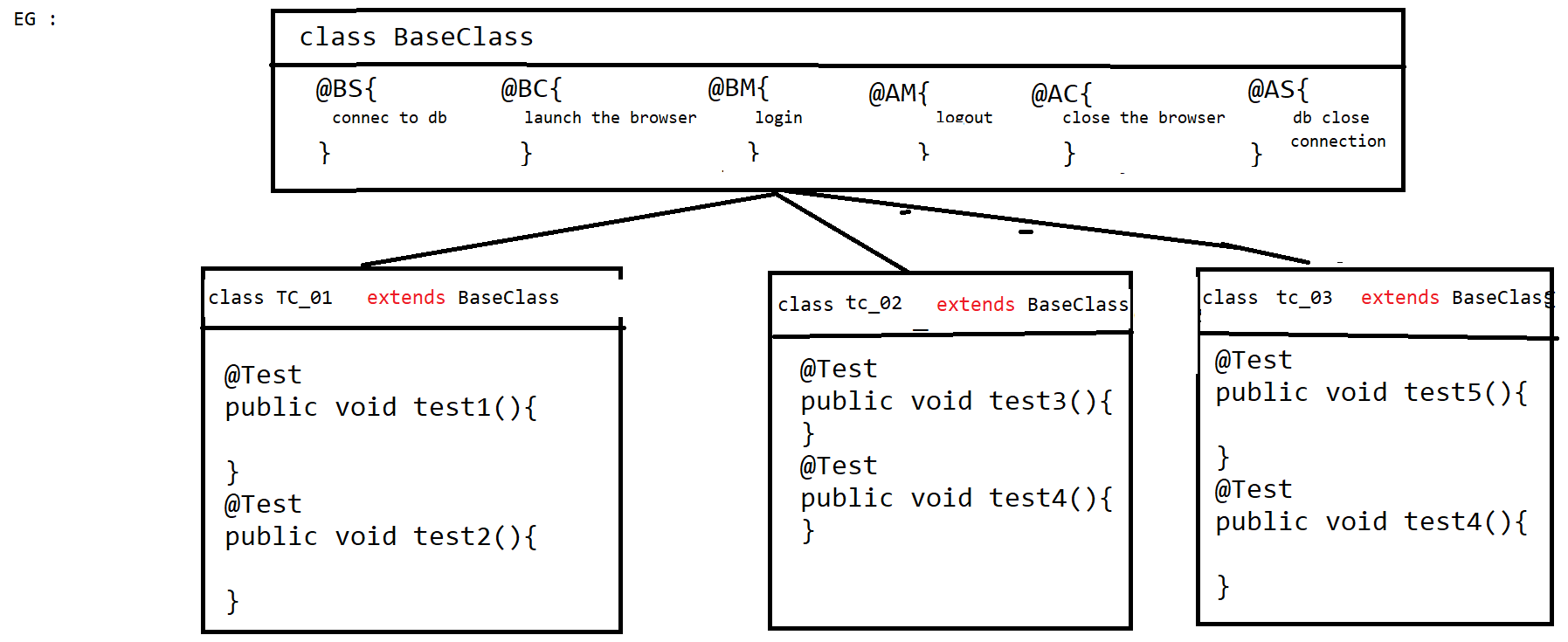
**Config Annotation Methods Usage in REAL Selenium Framework**

1. In Real Selenium Framework, all the configure annotation will be implemented

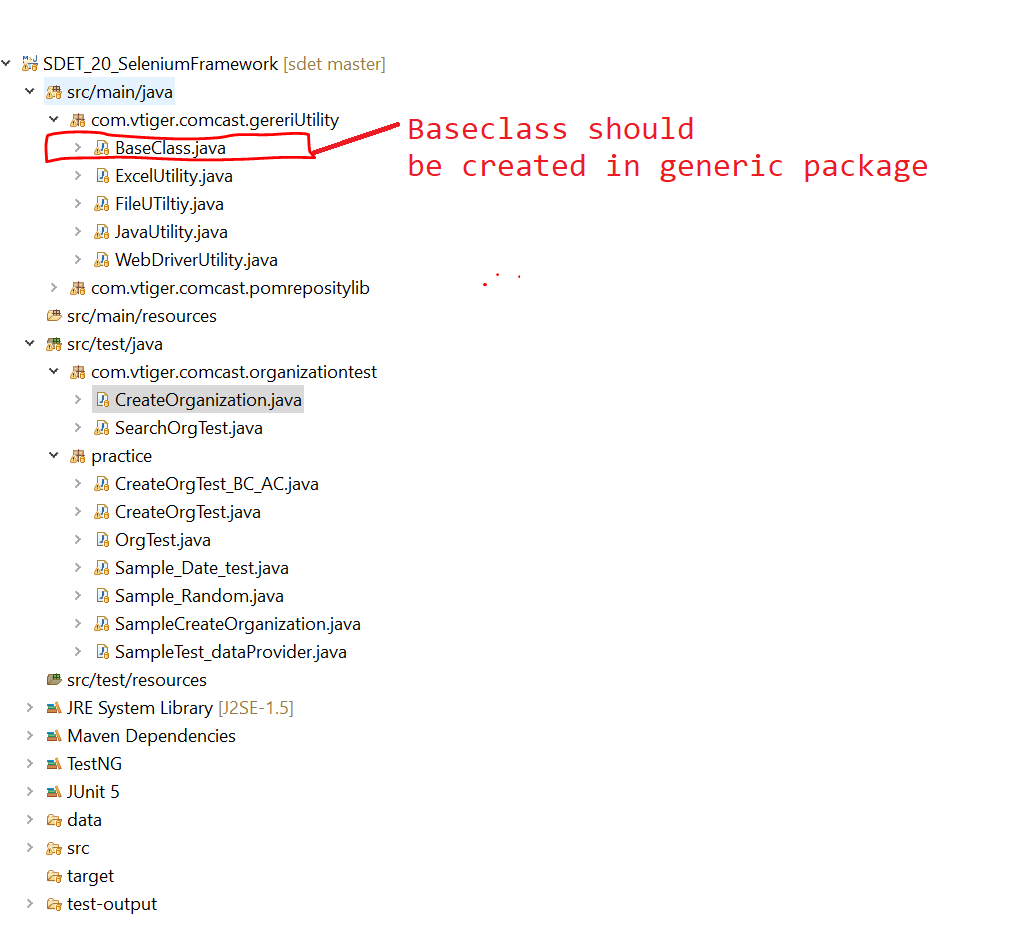
Inside the BaseClass, that is being shared to all the Automation engineers via GITGUB

1. BaseClass should be available in generic libraries package.
2. As per the Rule, every test scripts class should extend BaseClass, so that all the configure annotation will be inherited to the test scripts automatically

**Advantages:**

1. Code Reusability
2. Code Optimization
3. Modification is easy
4. Maintenance is easy
5. Test Development is faster 

**Project Structure in Eclipse**



**=========Base class Program=============**

**package** com.vtiger.comcast.gereriUtility;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.testng.annotations.AfterClass;

**import** org.testng.annotations.AfterMethod;

**import** org.testng.annotations.AfterSuite;

**import** org.testng.annotations.BeforeClass;

**import** org.testng.annotations.BeforeMethod;

**import** org.testng.annotations.BeforeSuite;

**import** com.vtiger.comcast.pomrepositylib.Home;

**import** com.vtiger.comcast.pomrepositylib.Login;

**public** **class** BaseClass {

**public** WebDriver driver;

/\*Object Creation for Lib\*/

**public** JavaUtility jLib = **new** JavaUtility();

**public** WebDriverUtility wLib = **new** WebDriverUtility();

**public** FileUTiltiy fLib = **new** FileUTiltiy();

**public** ExcelUtility eLib = **new** ExcelUtility();

@BeforeSuite

**public** **void** configBS() {

System.***out***.println("========================connect to DB========================");

}

@BeforeClass

**public** **void** configBC() {

System.***out***.println("=============Launch the Browser=======");

driver = **new** ChromeDriver();

wLib.waitUntilPageLoad(driver);

driver.manage().window().maximize();

}

@BeforeMethod

**public** **void** configBM() **throws** Throwable {

/\*common Data\*/

String USERNAME = fLib.getPropertyKeyValue("username");

String PASSWORD = fLib.getPropertyKeyValue("password");

String URL = fLib.getPropertyKeyValue("url");

String BROWSER = fLib.getPropertyKeyValue("browser");

/\* Navigate to app\*/

driver.get(URL);

/\* step 1: login \*/

Login loginPage = **new** Login(driver);

loginPage.loginToApp(USERNAME, PASSWORD);

}

@AfterMethod

**public** **void** configAM() {

/\*step 6: logout\*/

Home homePage = **new** Home(driver);

homePage.logout();

}

@AfterClass

**public** **void** configAC() {

System.***out***.println("=============Close the Browser=======");

driver.quit();

}

@AfterSuite

**public** **void** configAS() {

System.***out***.println("========================close DB========================");

}

}

=============Sample Test Script code using base class==============

**package** com.vtiger.comcast.organizationtest;

**import** org.testng.annotations.Test;

**import** com.vtiger.comcast.gereriUtility.BaseClass;

**import** com.vtiger.comcast.pomrepositylib.CreateNewOrganization;

**import** com.vtiger.comcast.pomrepositylib.Home;

**import** com.vtiger.comcast.pomrepositylib.OrganizationInfo;

**import** com.vtiger.comcast.pomrepositylib.Organizations;

**public** **class** CreateOrganization **extends** BaseClass{

@Test

**public** **void** createOrgTest() **throws** Throwable {

**int** randomInt = jLib.getRanDomNumber();

/\*test script Data\*/

String orgName = eLib.getDataFromExcel("Sheet1", 1, 2) + randomInt;

/\*step 2: navigate to organization\*/

Home homePage = **new** Home(driver);

homePage.getOrganizationLnk().click();

/\*step 3: navigate to "create new organization"page by click on "+" image \*/

Organizations orgPage = **new** Organizations(driver);

orgPage.getCreateOrgImg().click();

/\*step 4: create organization\*/

CreateNewOrganization cno = **new** CreateNewOrganization(driver);

cno.createOrg(orgName);

/\*step 5: verify the successful msg with org name\*/

OrganizationInfo orginfoPage = **new** OrganizationInfo(driver);

String actSuccesfullMg = orginfoPage.getSuccesfullMsg().getText();

**if**(actSuccesfullMg.contains(orgName)) {

System.***out***.println(orgName + "==>created successfully");

}**else** {

System.***out***.println(orgName + "==> not created successfully");

}

}

@Test

**public** **void** createOrgWithIndutriesTest() **throws** Throwable {

/\*test script Data\*/

**int** randomInt = jLib.getRanDomNumber();

String orgName = eLib.getDataFromExcel("Sheet1", 4, 2) + randomInt;

String industriesType = eLib.getDataFromExcel("Sheet1", 4, 3);

/\*step 2: navigate to organization\*/

Home homePage = **new** Home(driver);

homePage.getOrganizationLnk().click();

/\*step 3: navigate to "create new organization"page by click on "+" image \*/

Organizations orgPage = **new** Organizations(driver);

orgPage.getCreateOrgImg().click();

/\*step 4: create organization\*/

CreateNewOrganization cno = **new** CreateNewOrganization(driver);

cno.createOrg(orgName, industriesType);

/\*verify orgname & industry \*/

OrganizationInfo orginfoPage = **new** OrganizationInfo(driver);

String actSuccesfullMg = orginfoPage.getSuccesfullMsg().getText();

**if**(actSuccesfullMg.contains(orgName)) {

System.***out***.println(orgName + "==>created successfully");

}**else** {

System.***out***.println(orgName + "==> not created successfully");

}

String actIndustryType = orginfoPage.getIndutryTypeInfo().getText();

**if**(actIndustryType.equals(industriesType)) {

System.***out***.println(industriesType + "==>industry is verified successfully");

}**else** {

System.***out***.println(industriesType + "==>industry is not verified successfully");

}

}

@Test

**public** **void** createOrgWithRatingTest() **throws** Throwable {

/\*test script Data\*/

**int** randomInt = jLib.getRanDomNumber();

String orgName = eLib.getDataFromExcel("Sheet1", 7, 2) + randomInt;

String rating = eLib.getDataFromExcel("Sheet1", 7, 3);

/\*step 2: navigate to organization\*/

Home homePage = **new** Home(driver);

homePage.getOrganizationLnk().click();

/\*step 3: navigate to "create new organization"page by click on "+" image \*/

Organizations orgPage = **new** Organizations(driver);

orgPage.getCreateOrgImg().click();

/\*step 4: create organization\*/

CreateNewOrganization cno = **new** CreateNewOrganization(driver);

cno.createOrg(orgName, rating, **true**);

/\*verify orgname & industry \*/

OrganizationInfo orginfoPage = **new** OrganizationInfo(driver);

String actSuccesfullMg = orginfoPage.getSuccesfullMsg().getText();

**if**(actSuccesfullMg.contains(orgName)) {

System.***out***.println(orgName + "==>created successfully");

}**else** {

System.***out***.println(orgName + "==> not created successfully");

}

String actRatingType = orginfoPage.getRatingTypeInfo().getText();

**if**(actRatingType.equals(rating)) {

System.***out***.println(rating + "==>industry is verified successfully");

}**else** {

System.***out***.println(rating + "==>industry is not verified successfully");

}

}

}

**Batch Execution**

* Collection of multiple test script is called batch, execute multiple test script through xml in a single click is called batch execution.
* In order to achieve batch execution, we go for Testng.xml configuration file.
* TestNG xml file always start with suite xml tag followed by <test> and <classes>.
* In one xml file we can invoke N-number of TestNG classes, but all the classes should be present within a project.
* All the class name should be followed by the package name.

EG:

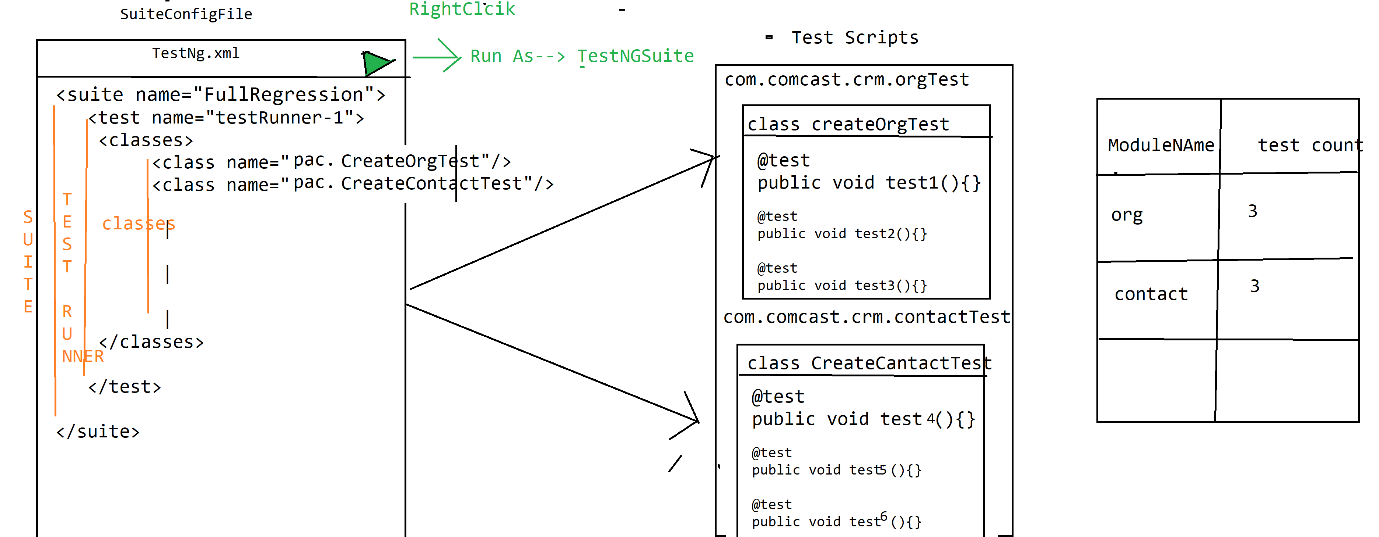
<class name=”com.comcast.orgtest.CreateOrgTest”></class>

**How to create TestNG.xml file automatically through eclipse?**

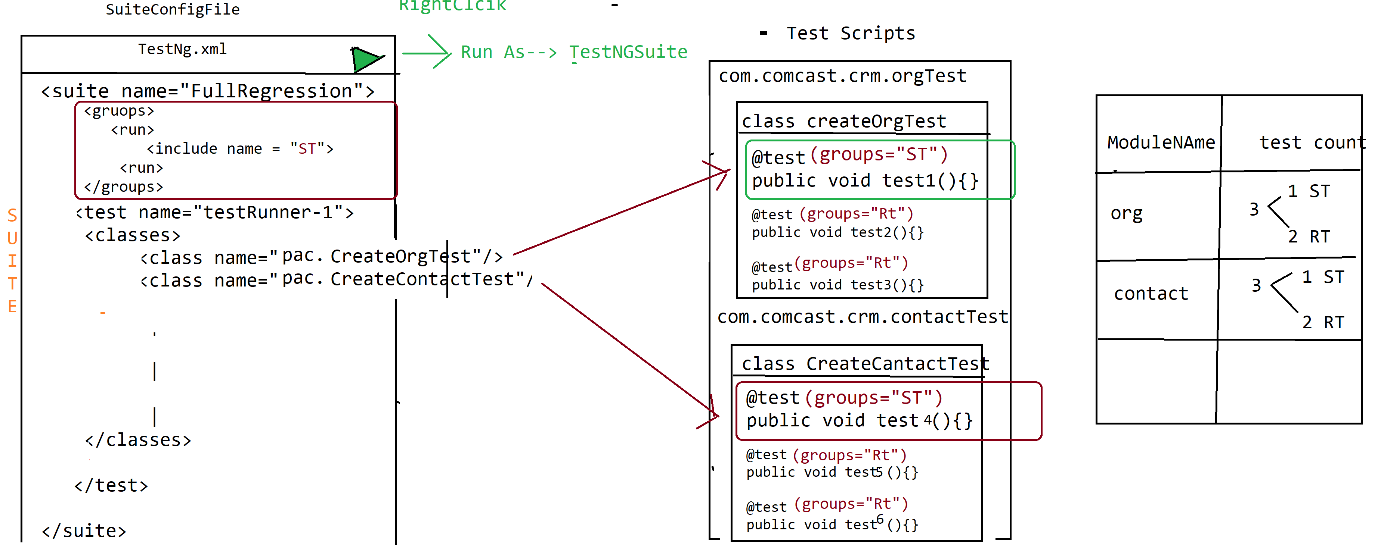
Select all the TestNG classes or packages -> right click-> select🡺 testNG 🡺and click on “Convert to TestNG” and🡺 click on finish.

* Automatically you will get the testng.xml file with in the project
* In order to edit xml File🡺 double click on TestNG.xml🡺 click on “Source”

**Batch Execution: [Full regression testing]**

****

**Grouping Execution:**



* Collection of similar test scripts across the testing classes is called grouping Execution
* In order to achieve grouping execution, each & every test script should have group name, group name will be written along with annotation.
* In group execution, all configure annotations should have group name, otherwise those annotations will not participate in grouping execution like

@BeforeSuite @BeforeClass, @BeforeMethod etc.

EG:

@BeforeSuite(groups = {"smokeTest","regressionTest"})

**public** **void** configBS() {

System.***out***.println("======Execute BeforeSuite=============");

}

**Smoke Test:**

@Test(groups={"smokeTest"})

**public** **void** createCustomerTest()

{

System.***out***.println("execute createCustomerTest");

}

@Test(groups={"regressionTest"})

**public** **void** modifyCustomerTest()

{

System.***out***.println("execute modifyCustomerTest");

}

* In order to invoke grouping execution should declare Group Key in testing.xml file

& group key should be declared before <test>, after <suite> tag

**Smoke Test:**

<suite name=*"Suite"*>

<groups>

<run>

<include name=*"smokeTest"*/>

</run>

</groups>

<test name=*"Test"*>

<classes>

<class name=*"pac1.ProjectAndCustomerTest"*/>

<class name=*"pac2.ReportTest"*/>

</classes>

</test>

</suite>

🡺We can invoke multiple group keys in one XML File

<suite name=*"Suite"*>

<groups>

<run>

<include name=*"smokeTest"*/>

<include name=*"regressionTest"*/>

</run>

</groups>

<test name=*"Test"*>

<classes>

<class name=*"pac1.ProjectAndCustomerTest"*/>

<class name=*"pac2.ReportTest"*/>

</classes>

</test>

🡺One test can have multiple Group name

@Test(groups={"regressionTest","smokeTest"})

**public** **void** modifyCustomerTest()

{

System.***out***.println("execute modifyCustomerTest");

}

**Regional Regression Test:**

🡺To execute particular test cases across the Suite is called regional regression testing

🡺In real time, impact area is given by developer / test lead, based on that idea XML will be created.

🡺 Whenever we want execute particular @Test method inside class, we go for <method> & <include>

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Suite"*>

<test thread-count=*"5"* name=*"Test"*>

<classes>

<class name=*"com.comcast.crm.contacttest.CreateContactTest"*>

<methods>

<include name=*"createdContactTest"*/>

</methods>

</class>

<class name=*"com.comcast.crm.orgtest.CreateOrgTest"*>

<methods>

<include name=*"createdOrgWithRatingTest"*/>

</methods>

</class>

</classes>

</test> <!-- Test -->

</suite> <!-- Suite -->