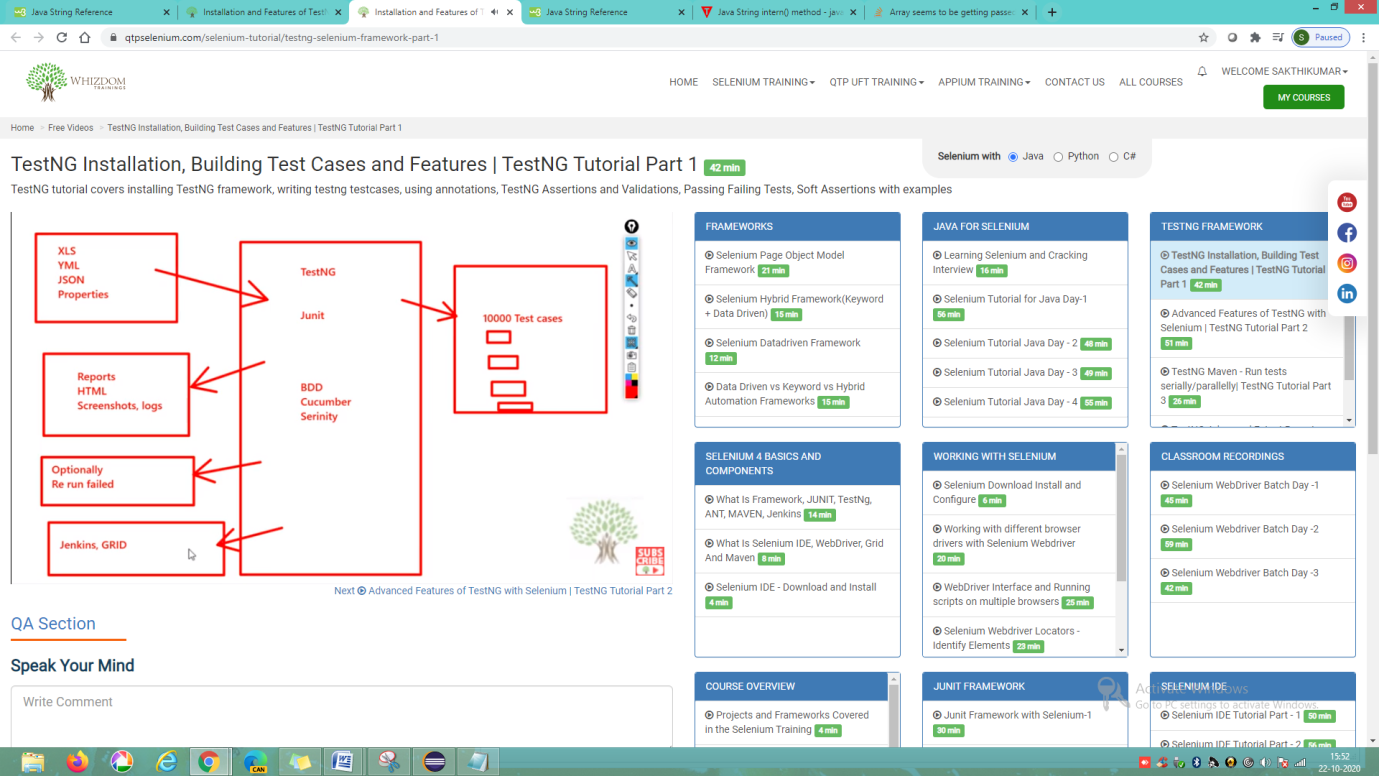
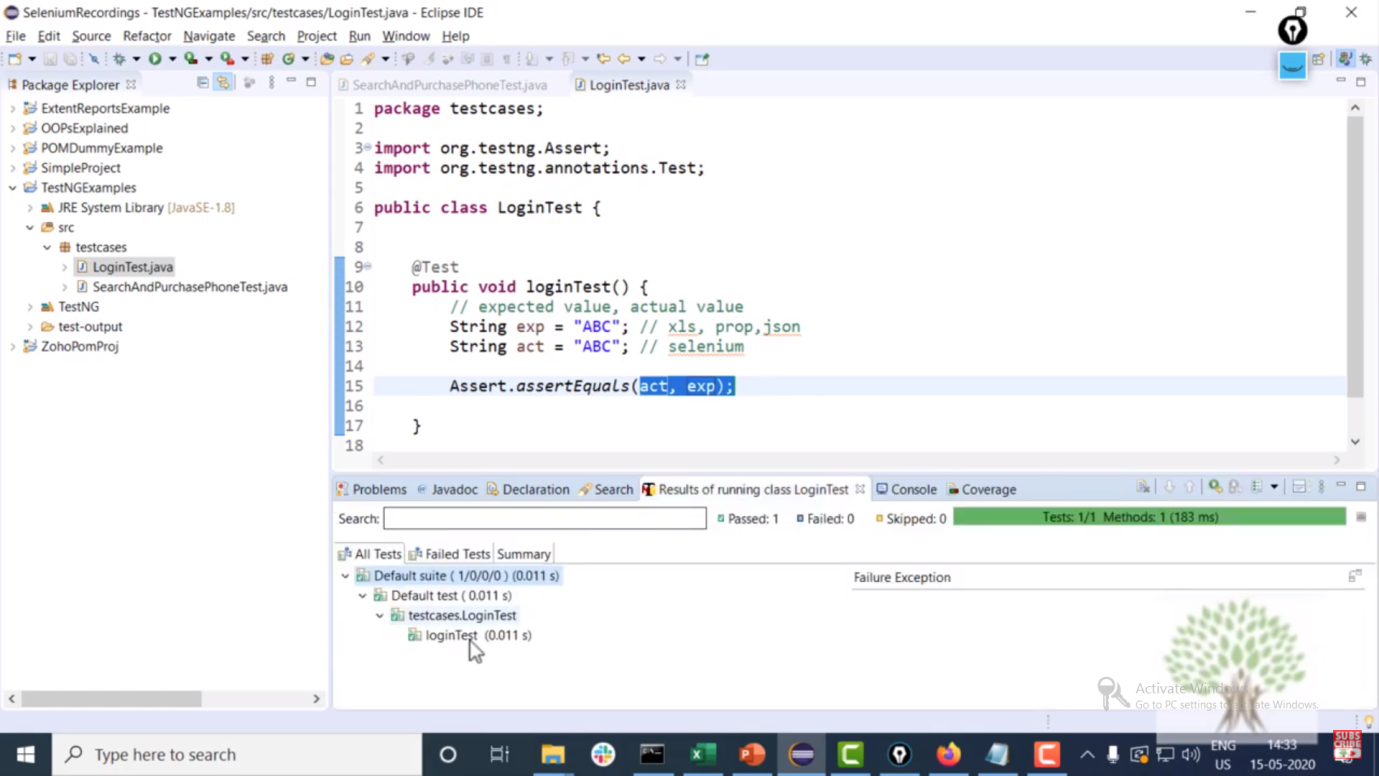
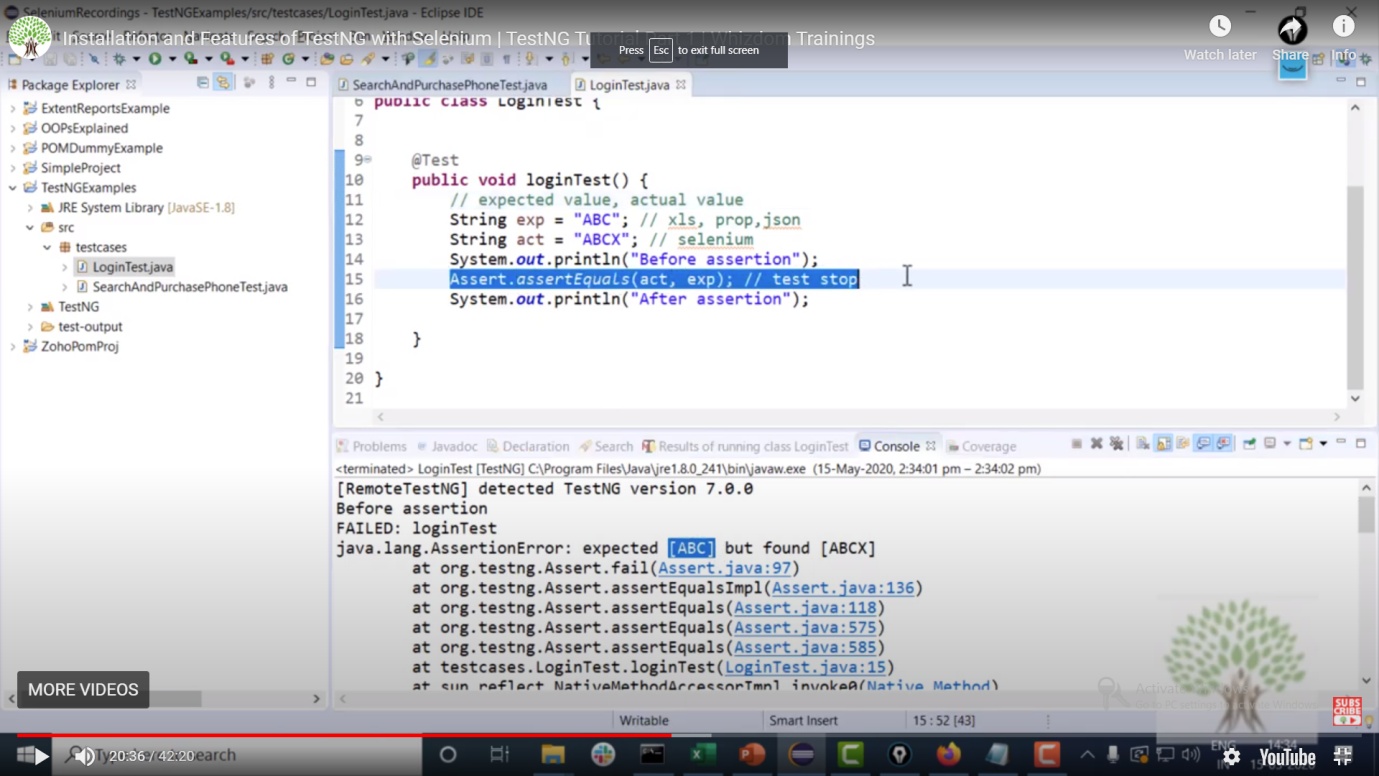
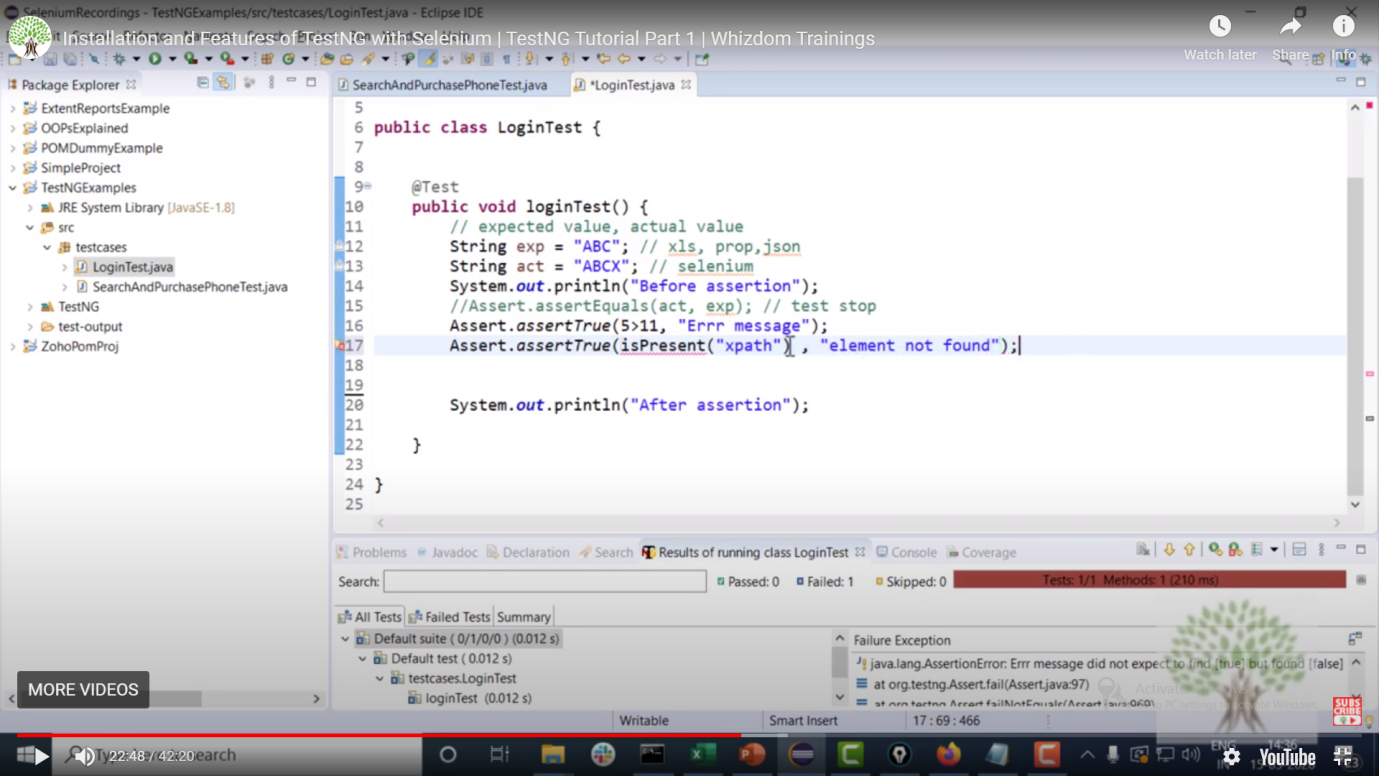
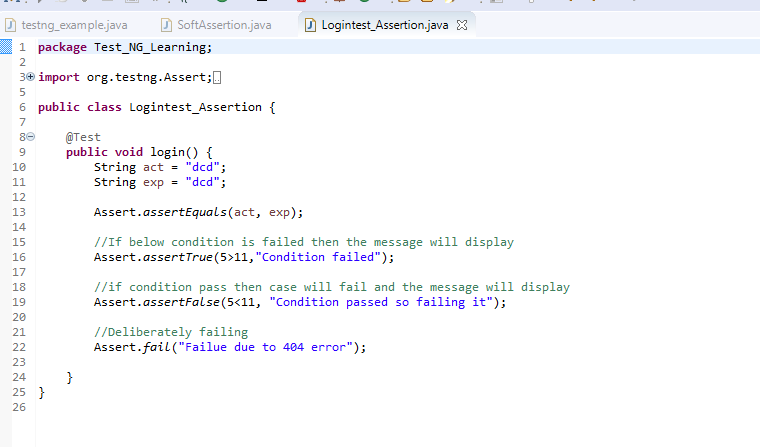
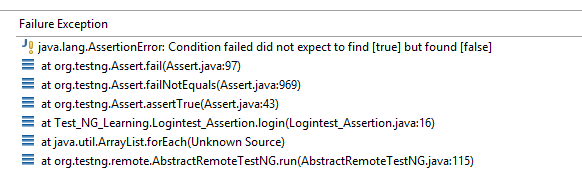
**TestNG points**

 Assertions 

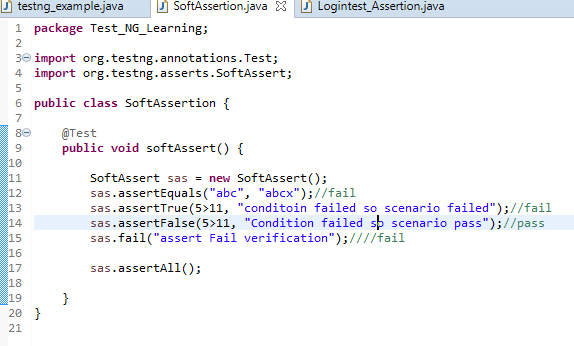
After assertion next line of code is not executed due to assertion failure. To avoid this we are going for soft assertion. 

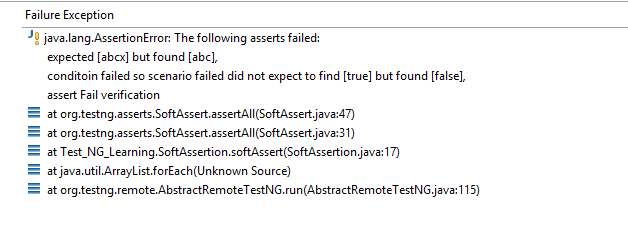
Assert True usage 

**Normal Assert**

In Normal assert approach we will verify “assertEquals”,”assertTrue”,”assertFalse” and “fail” methods. Here the drawback is if any of the assert is failed then the next lines will not execute. So to avoid the situation we are going for Soft Asserts. 

**SoftAssert**

using this we can report multiple failures (Step wise failures) in a single testcase 



**Depends on methods, Skipping a testcase deliberately and ITestContext**

dependsOnMethods – is an array will accept values like an array. This will consist dependent testcases names in it

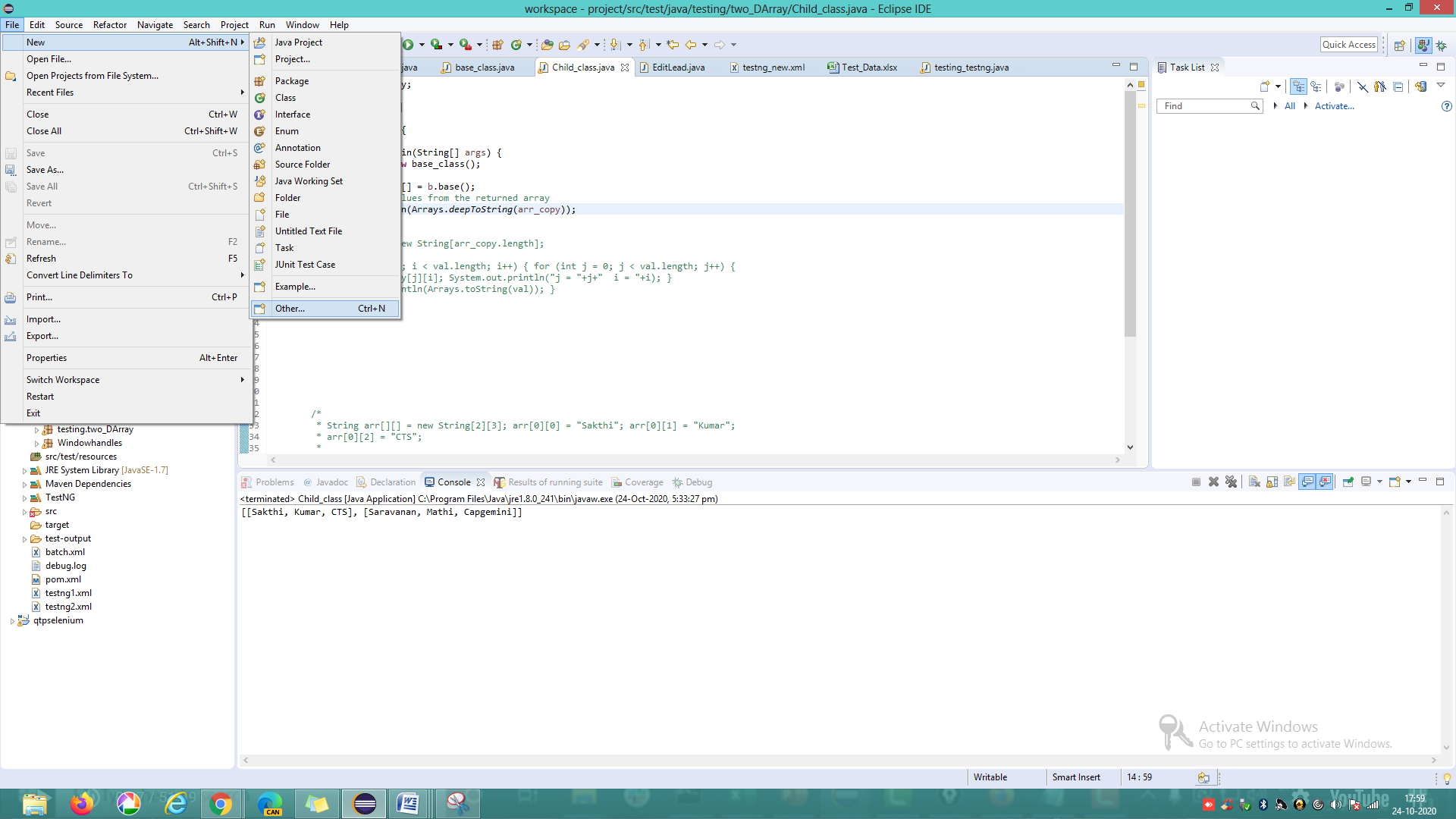
Skipping testcase – “throw New SkipException()” will help us to skip a testcase

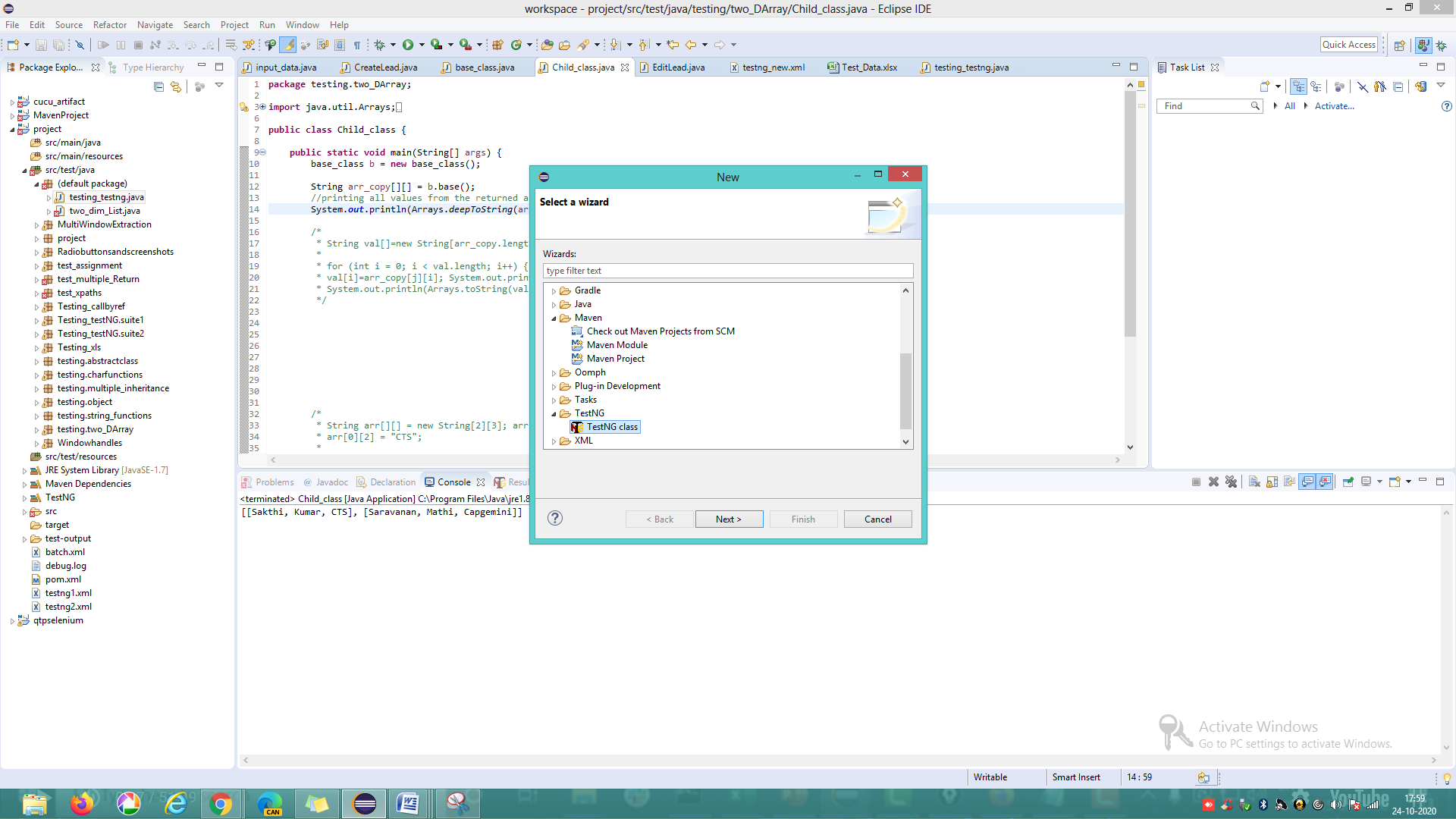
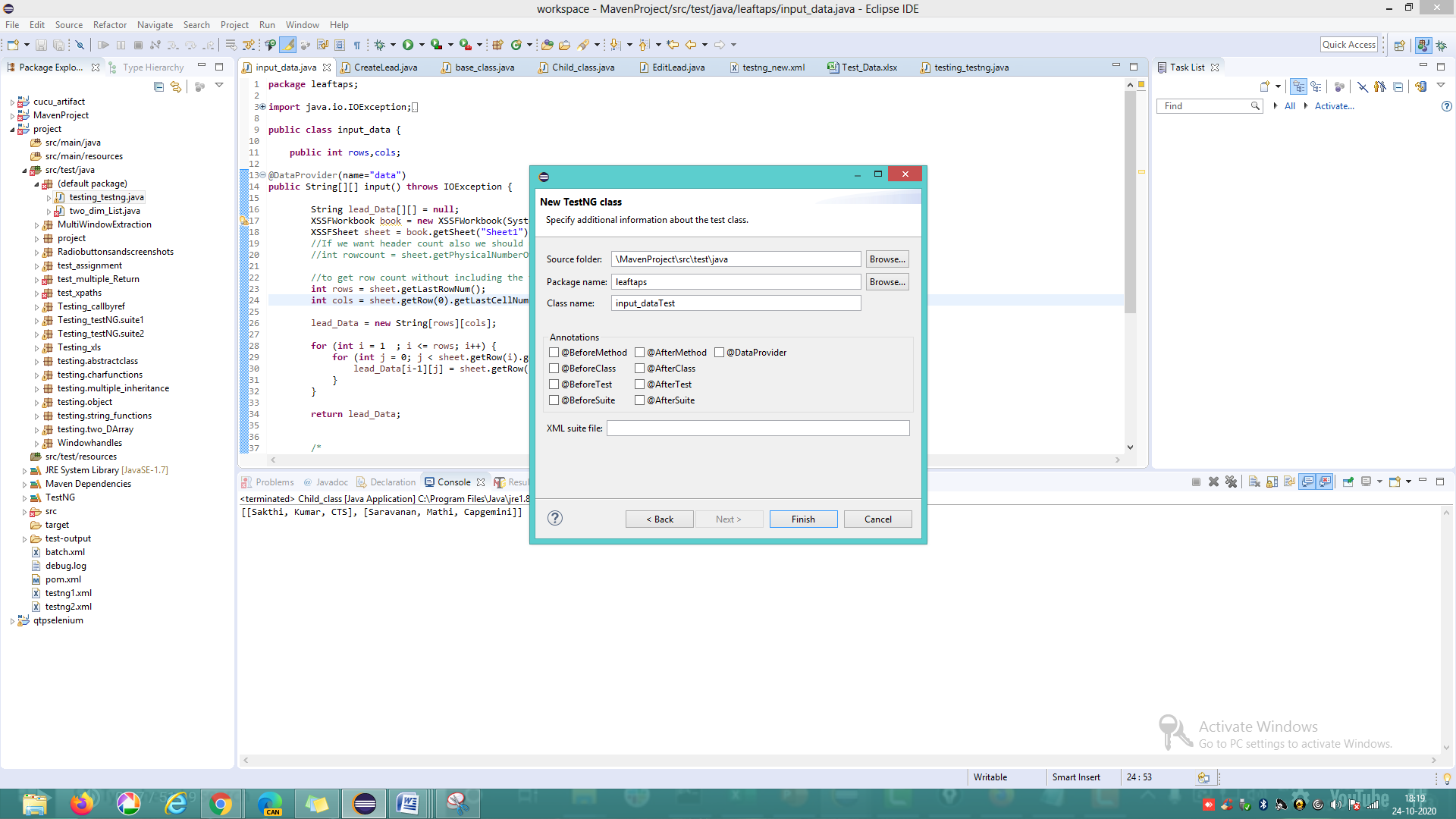
ITestContext – used to pass values betweek one case to another case. This is an interface used to pass the value in the form of (Key,Value) like map.

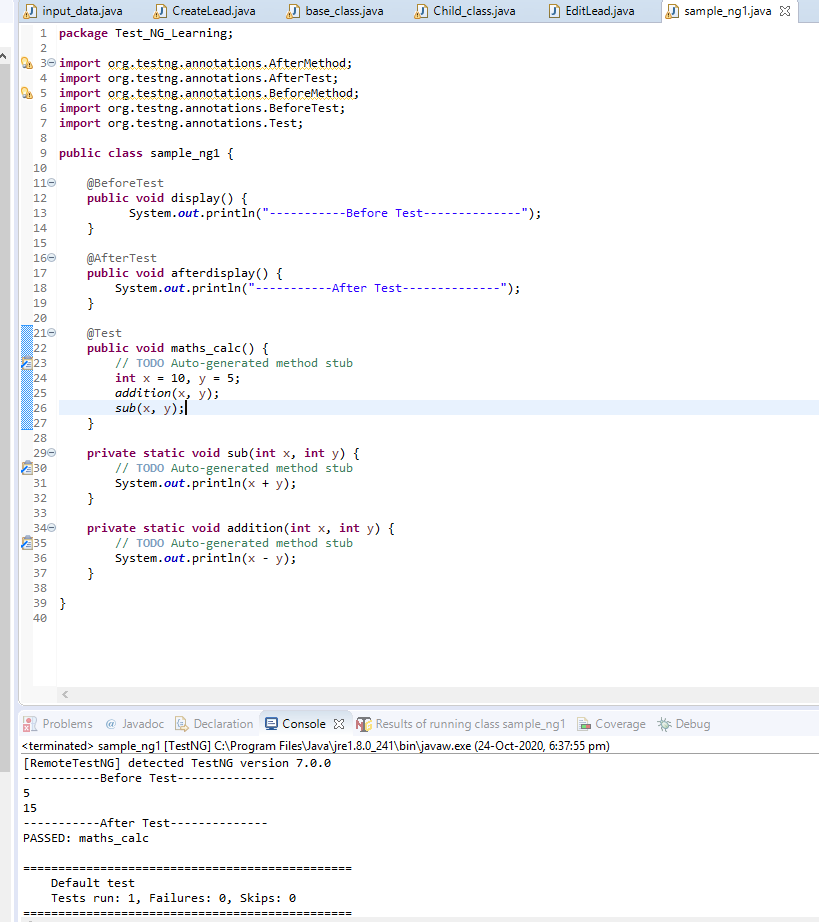
setAttribute - method will send the value in the form of (Key,Value)

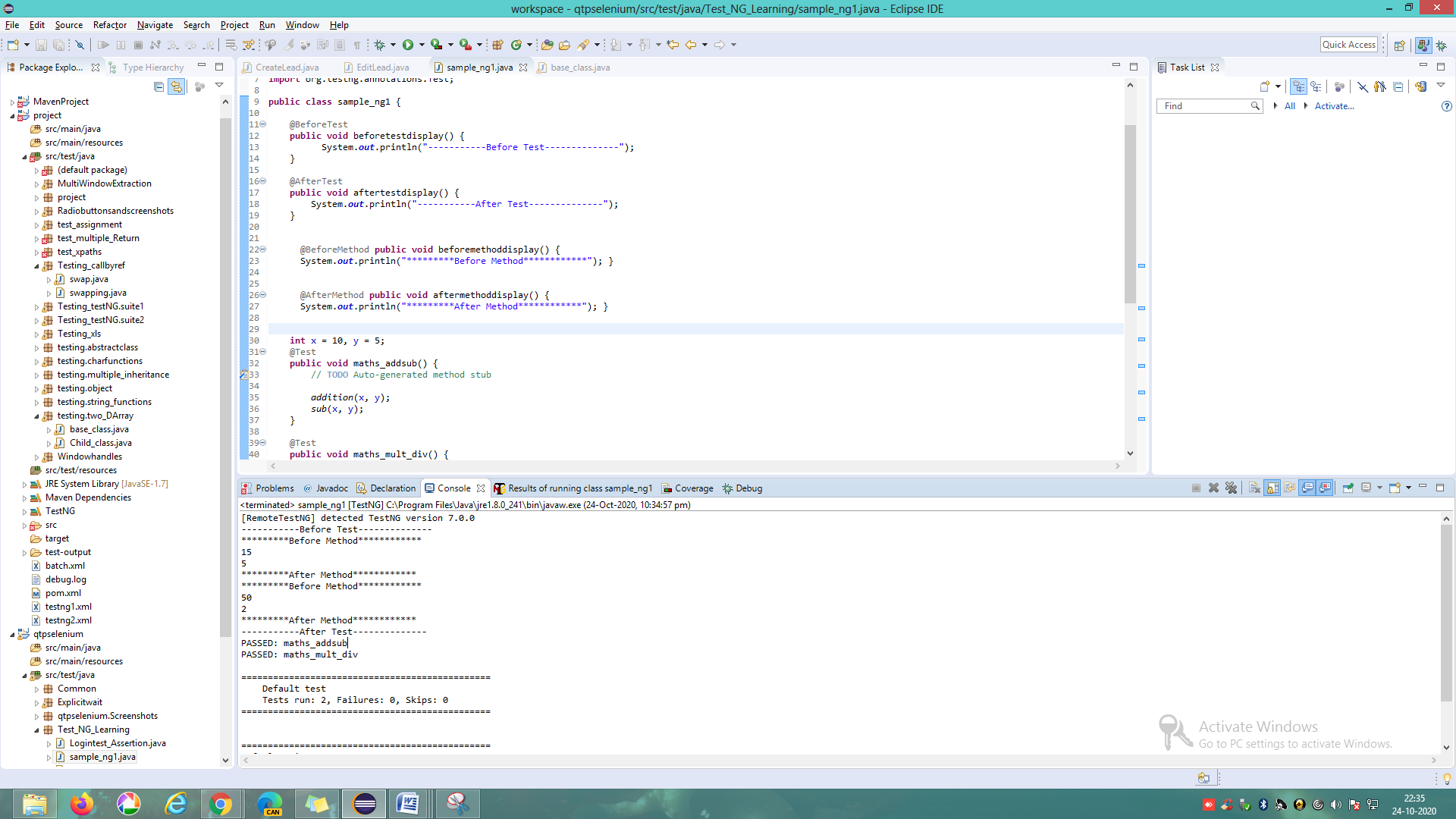
getAttribute – method will receive the value in Object datatype format so we need to typecast the values.

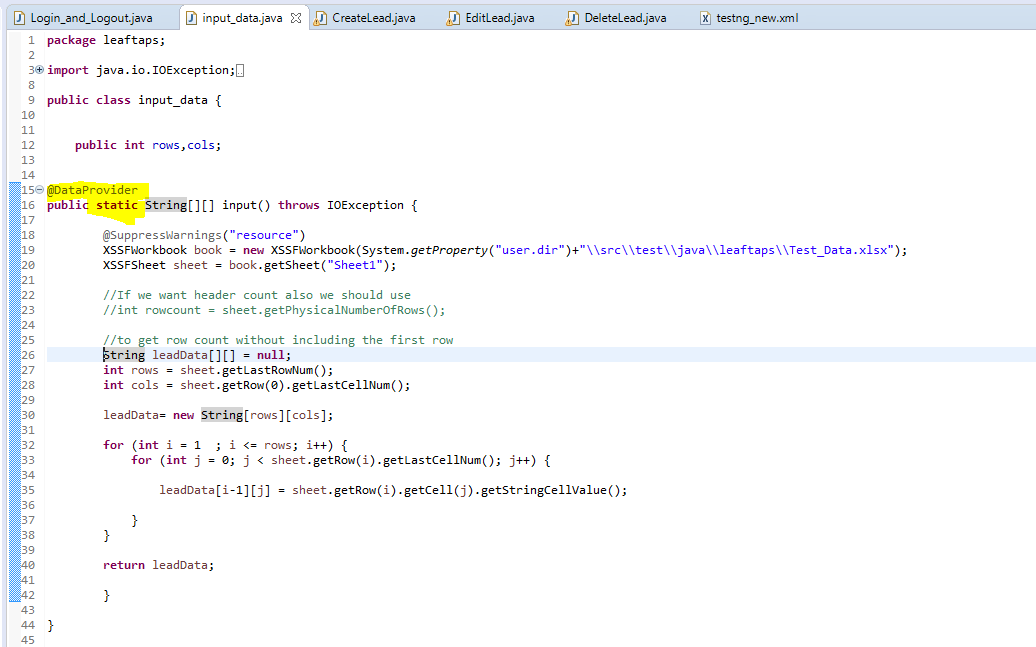


**Annotations **

@BeforeTest and @AfterTest are executed before and after the testcases. This will execute even the case is pass or fail 

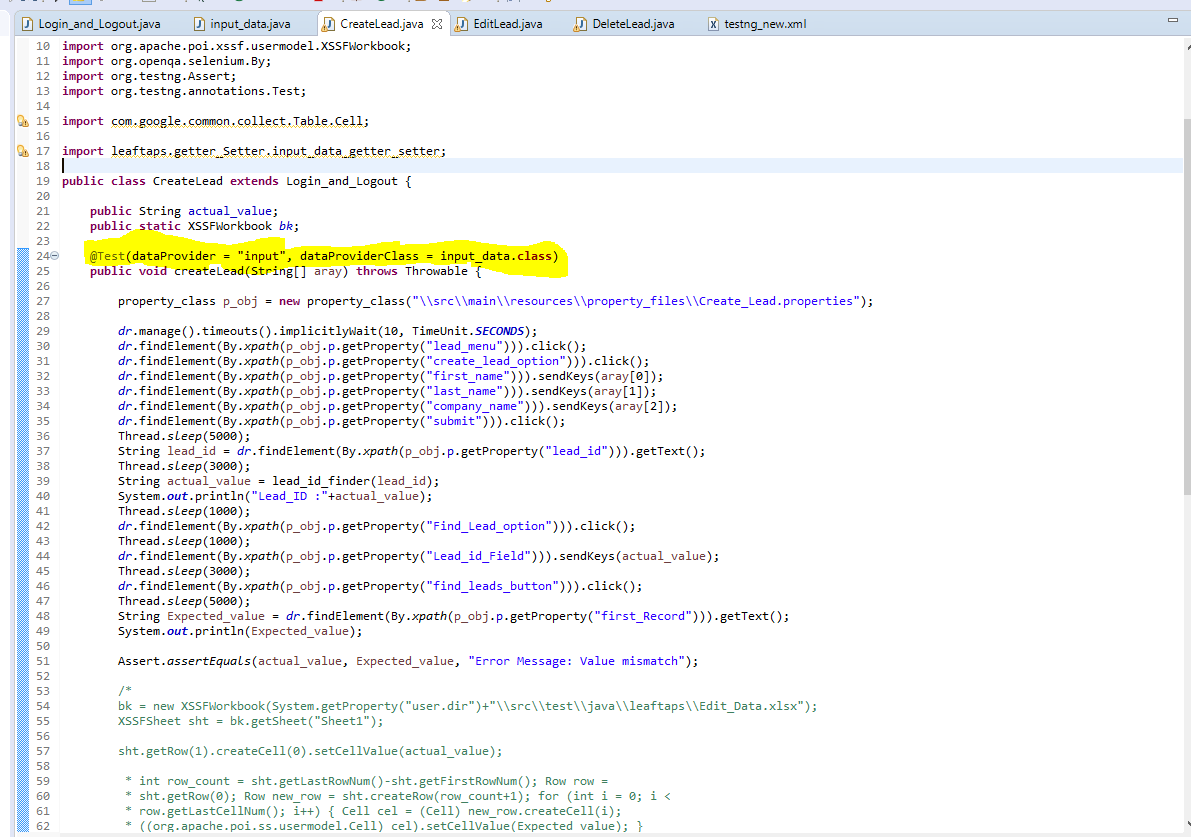
@BeforeMethod and @AfterMethod will be used to execute before and after each test specified in a single class file 

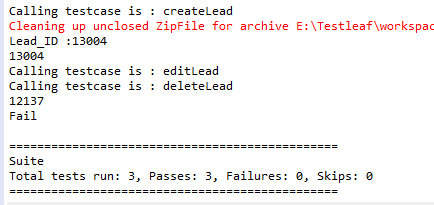
@DataProvide – used to give testdata for testcases mention the data provider class seperately. To access this class anywhere we need to declare the class as a static 

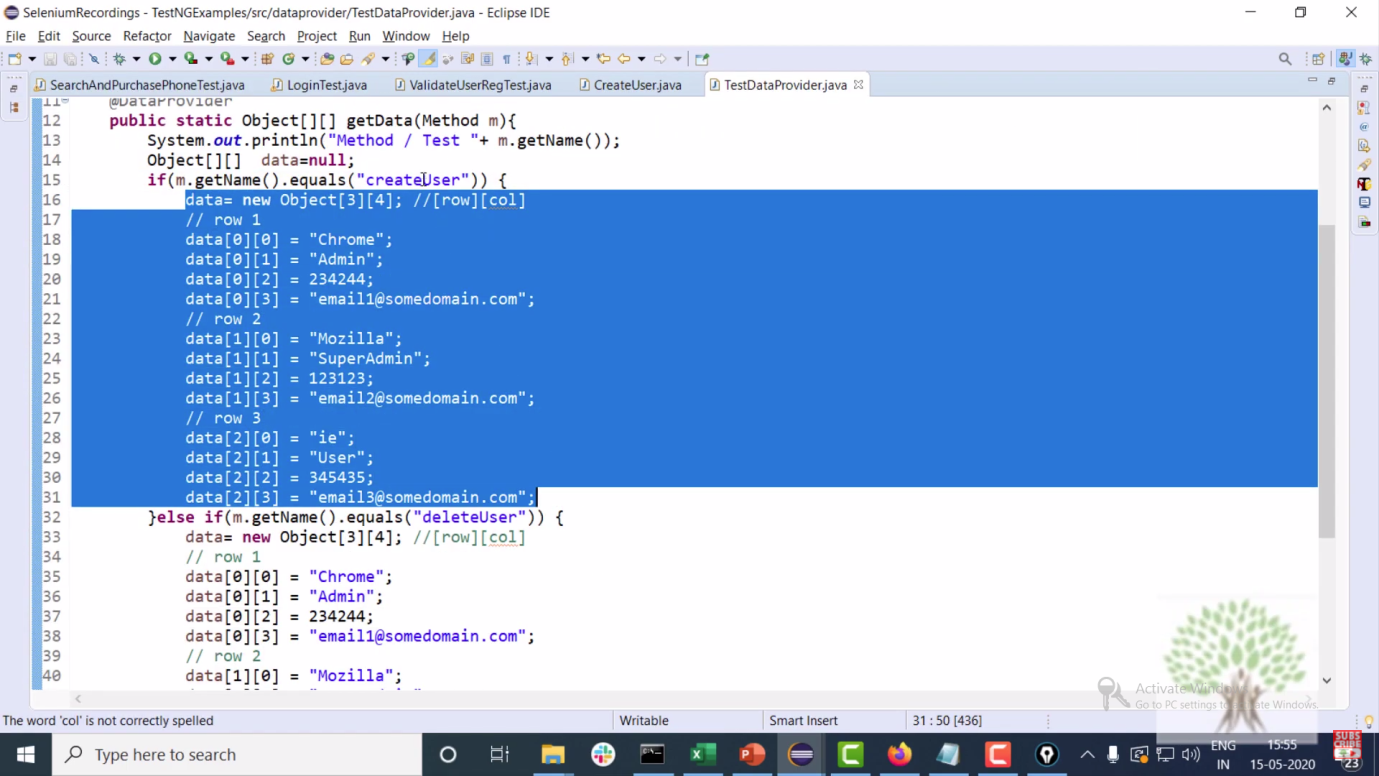
Data provider should be declared as below in receiving class

Syntax:

@Test(dataProviderClass = dataprovider classname.class,dataProvider = “methodname”)

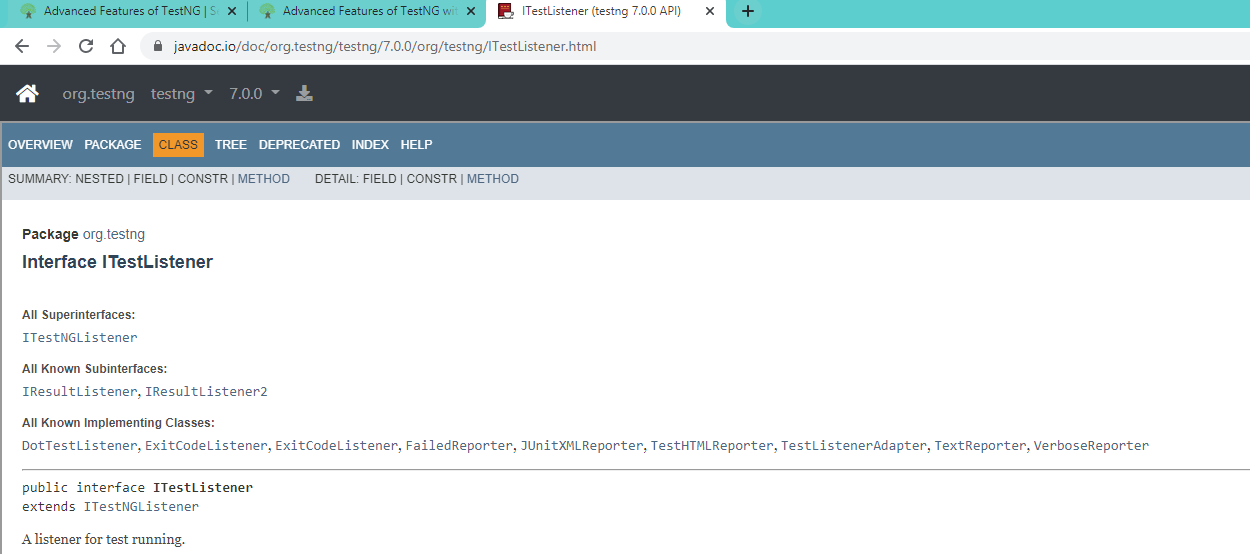


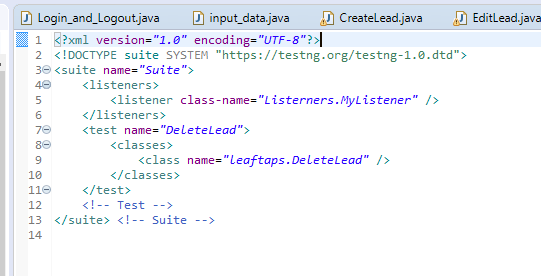
Using “Method” class we can identify the calling testcase name inside the dataprovider  

Based on testcase name we can change the test data conditions as below 

We can seperately mention the testcases in xml file to get the results specifically to each case

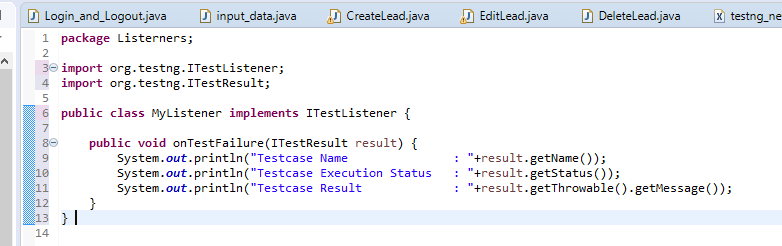
**Listener interface**

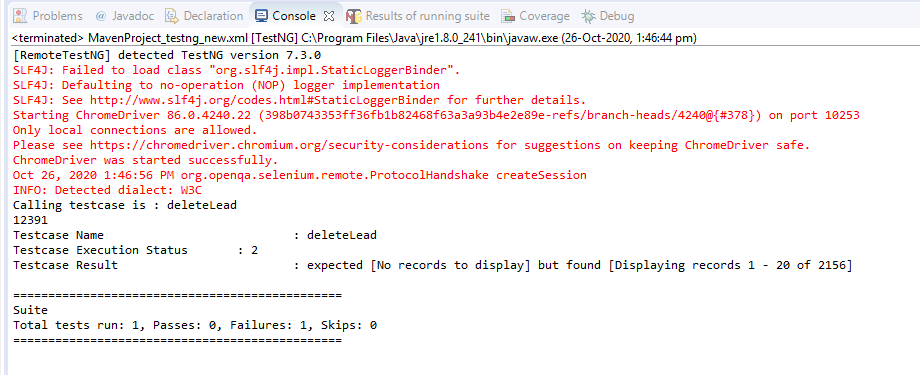
Listeners are used to check which testcase is pass or fail. Listeners use below sub classes for it  

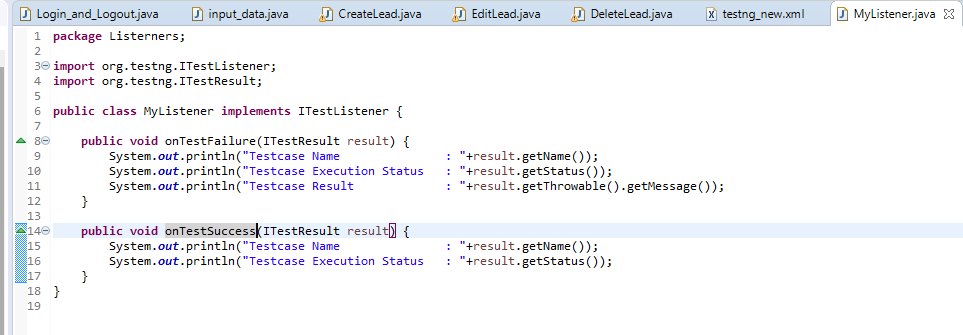
Listener Declaration in xml file (Mention the Package name .class name)

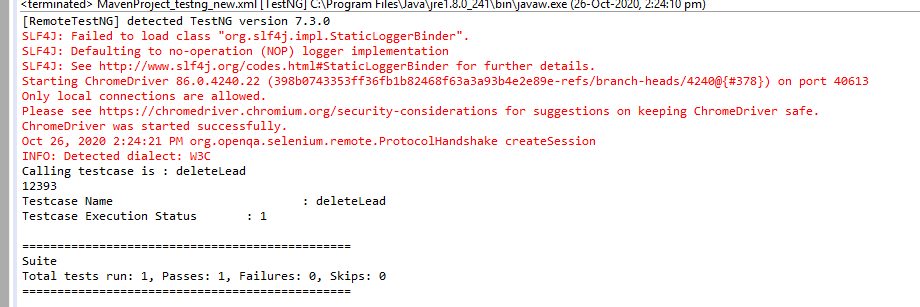
Listener class (if the error “Save could not be completed. Reason: Some characters cannot be mapped using "Cp1252" is displayed then rename the “onTestFailure” as “onTestFailures” then again rename it as “onTestFailure”)

**onTestfailure Method**

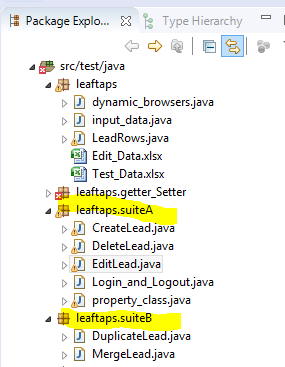


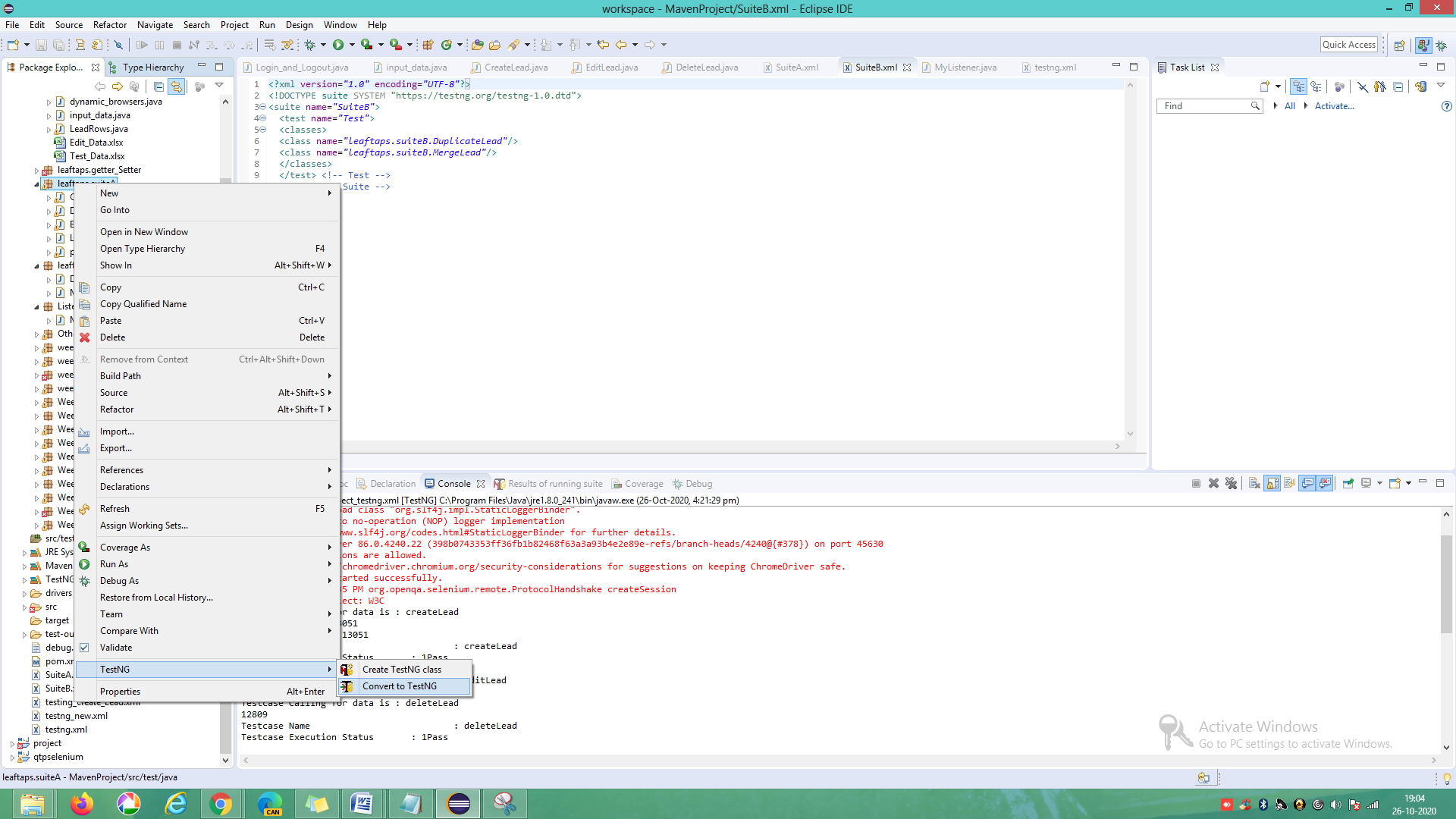
Result (If case failed the status will display the value 2)

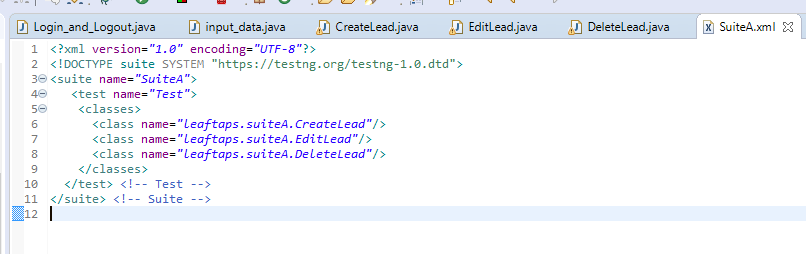
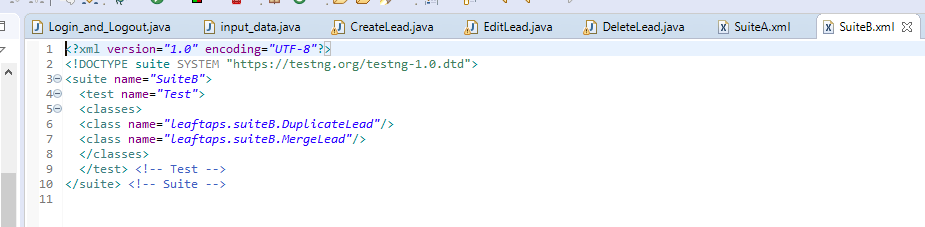
**onTestSuccess Method**

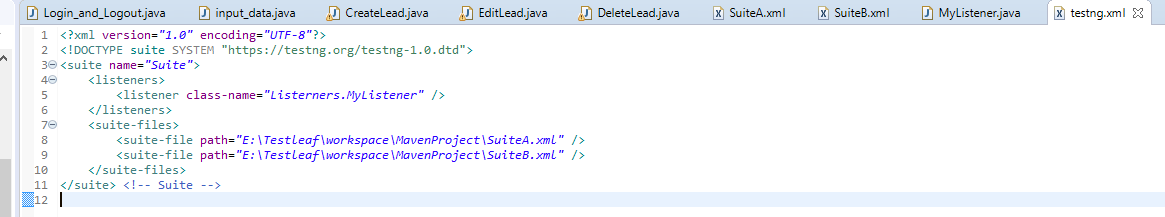
Result (If case passed the status will display the value 1)****

**Test Suite creation**

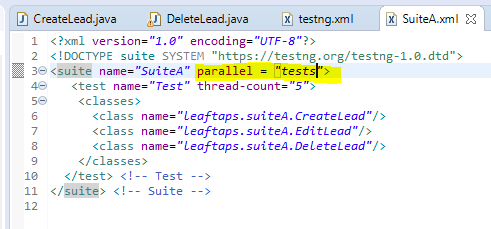
Step1 : Create 2 packages SuiteA and SuiteB. Add testcases into it. 

Step2: Select the testcases and convert them to TestNG xml files. Do it seperately for each suite

Step3: Check the xmls having the testcases  

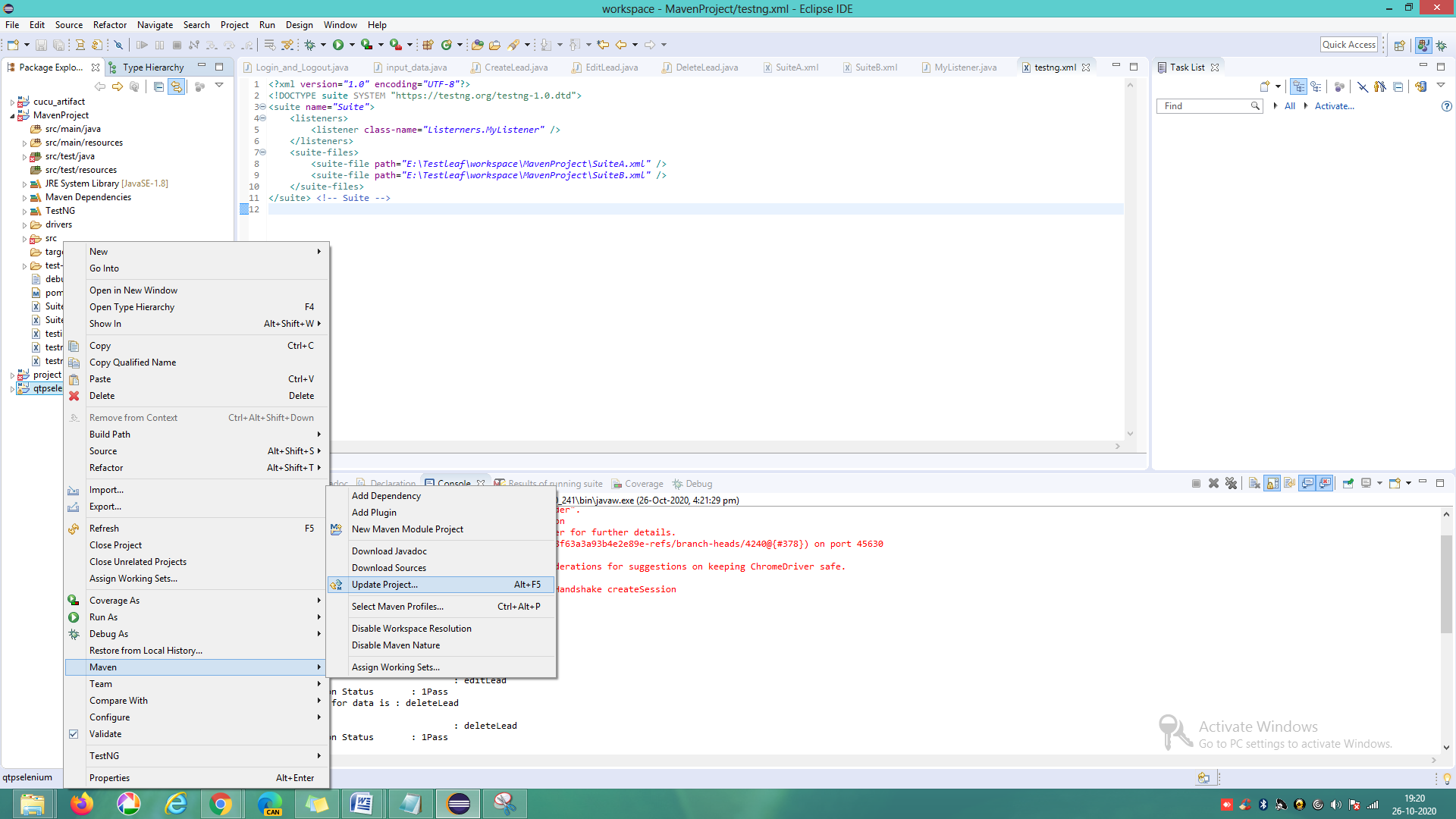
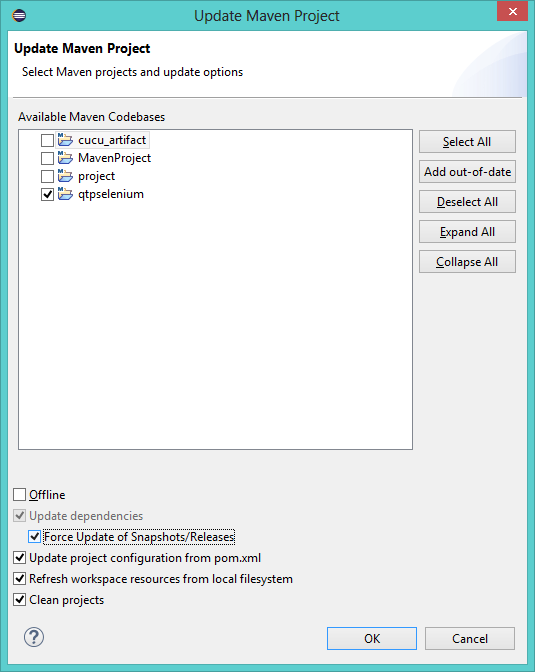
Step4: Call the suite xml files (suite A and B) in common xml file with listener 

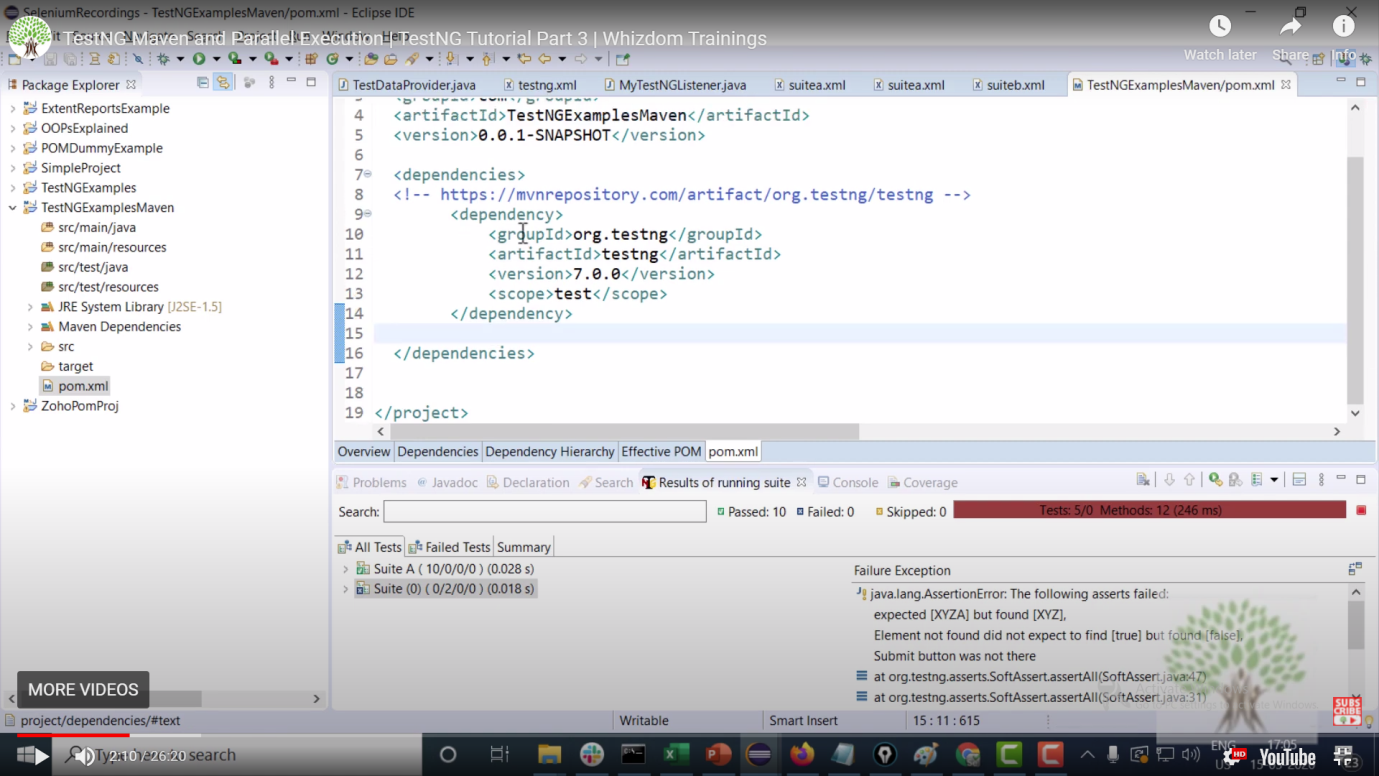
Parallel Execution of testcases



**Maven Configuration**

First create a maven project. If Maven project shows any error then right click on project->Maven->UpdateProject->Forceupdate

add the TestNG dependency file given below by searching in google “Maven testNG depedency” version 7.0.0****

Add the build code after the dependecies tag and within the project tag. To run the cases in unix or shell scripts we need this approach.

<build>

<plugins>

<!-- Compiler plug-in -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>${jdk.level}</source>

<target>${jdk.level}</target>

</configuration>

</plugin>

<!-- Below plug-in is used to execute tests -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.0.0-M4</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>E:\Testleaf\workspace\qtpselenium\src\test\resources\testng.xml</suiteXmlFile>

<!-- TestNG suite XML files

<suiteXmlFile>src/test/resources/suiteA.xml</suiteXmlFile>

<suiteXmlFile>src/test/resources/suiteB.xml</suiteXmlFile>

<suiteXmlFile>src/test/resources/suiteC.xml</suiteXmlFile

<suiteXmlFile>target/surefire-reports/testng-failed.xml</suiteXmlFile-->

</suiteXmlFiles>

<properties>

<property>

<name>suitethreadpoolsize</name>

<value>3</value>

</property>

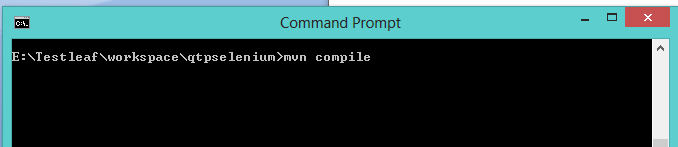
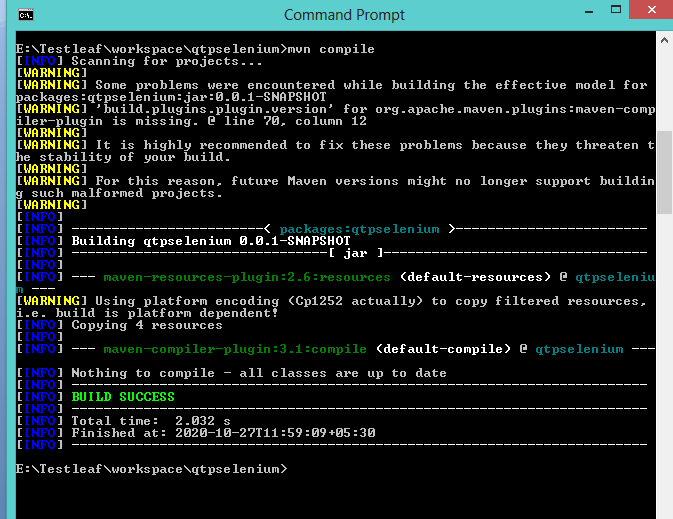
</properties>

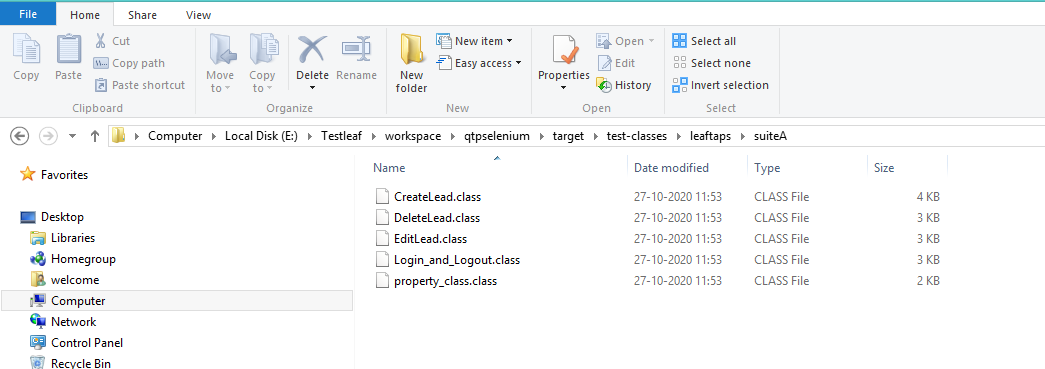
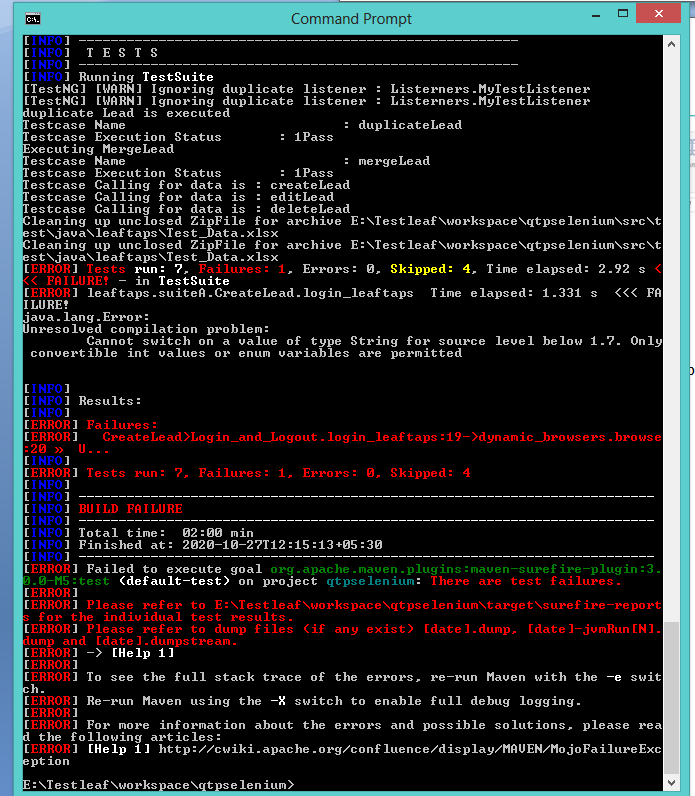
</configuration>

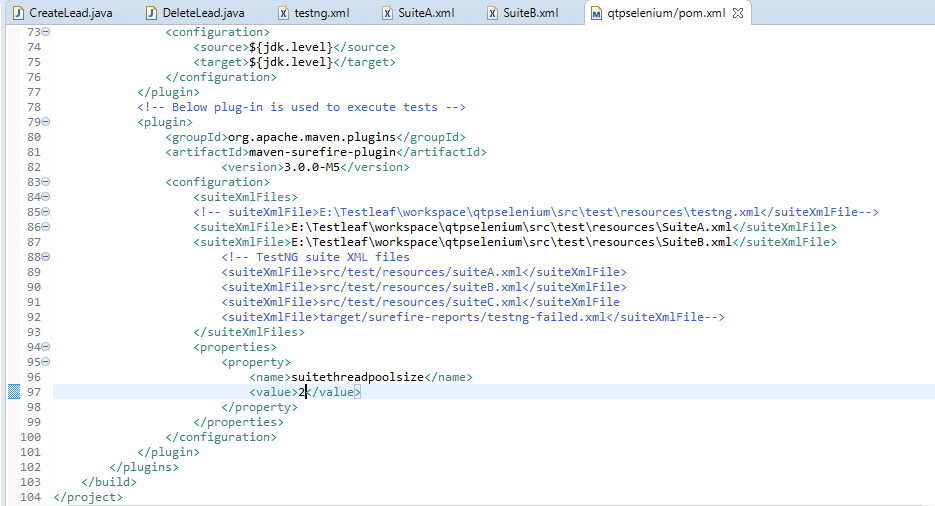
</plugin>

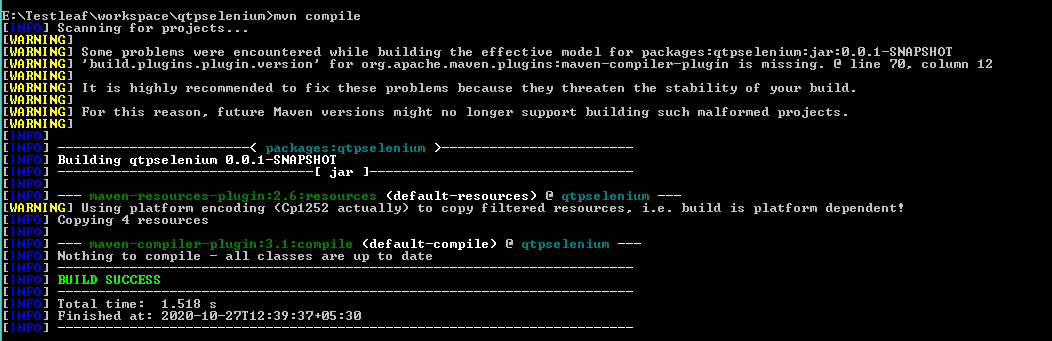
</plugins>

</build>

After adding above build in POM file. Go to command prompt.navigate to the project path and execute the command “mvn compile”  

After running go to Target folder in the project there navigate to the project folder. Here we could see the class files After these steps run the program by giving “mvn test”. If any failures occured go and install the latest version of maven surefire plugin in dependency. 

Parallel test execution of test suites can be achieved through maven easily. Specify the suite details as below

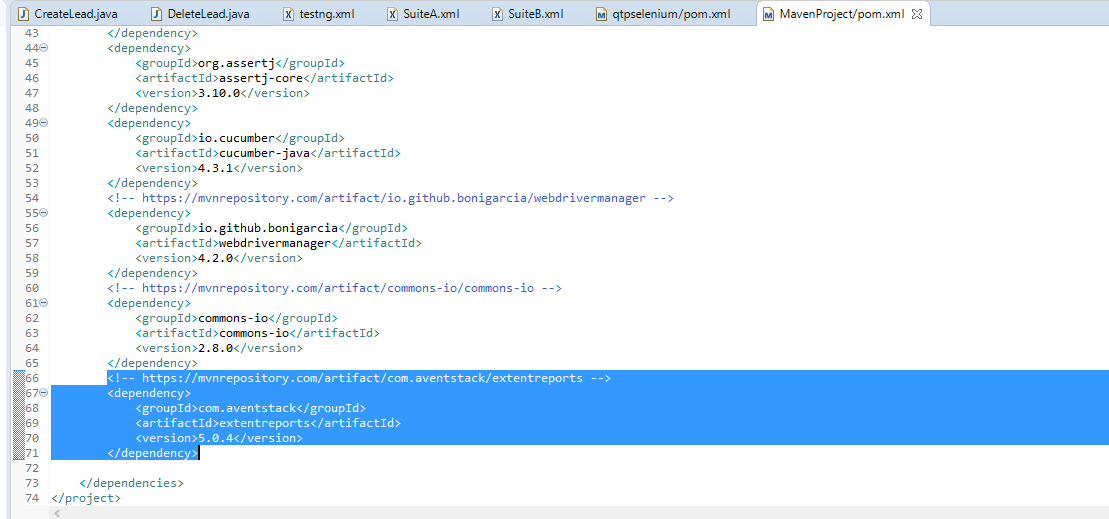
Compile the program by navigating to the project path 

Run the program by specifying “mvn test” 

To Run failed testcases change the dependency file as below and run in maven

**Extent reports**

Copy and paste the below dependency for extent report from the path “<https://mvnrepository.com/artifact/com.aventstack/extentreports>”



Create a new package then inside it create a class and write the below code