 Copyright © tutorialspoint.com

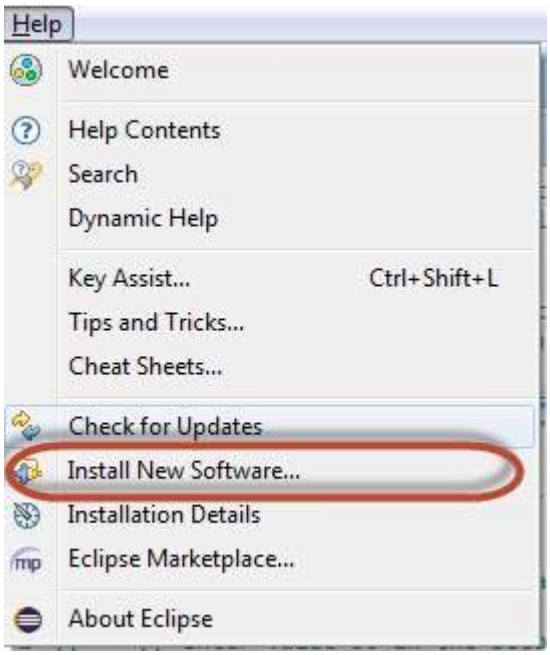
TestNG is a powerful testing framework, an enhanced version of JUnit which was in use for a long time before TestNG came into existence. NG stands for 'Next Generation'.

TestNG framework provides the following features:

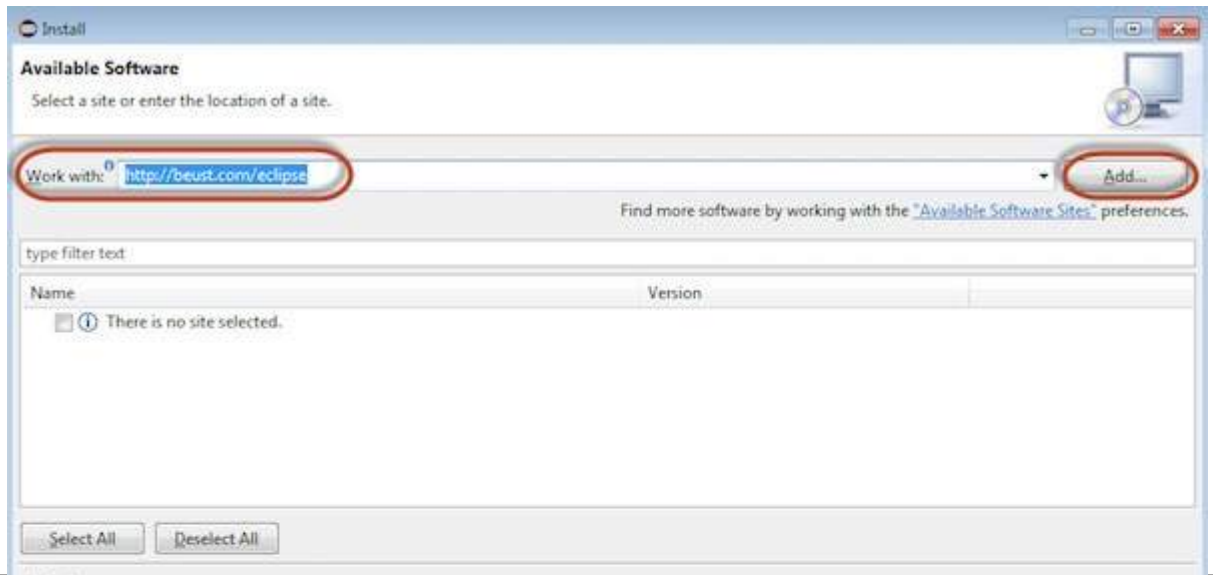
- Annotations help us organize the tests easily.
- Flexible test configuration.
- Test cases can be grouped more easily.
- Parallelization of tests can be achieved using TestNG.
- Support for data-driven testing.
- Inbuilt reporting.

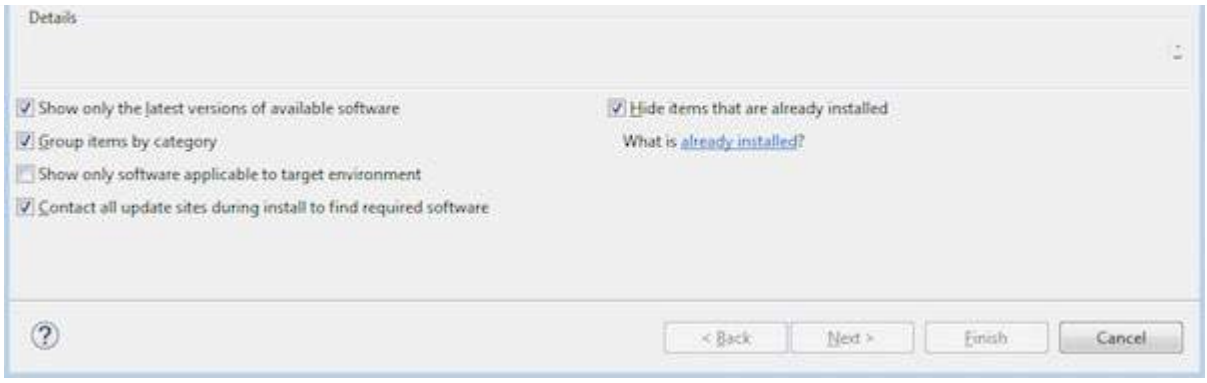
Installing TestNG for Eclipse

Step 1 : Launch Eclipse and select 'Install New Software'.

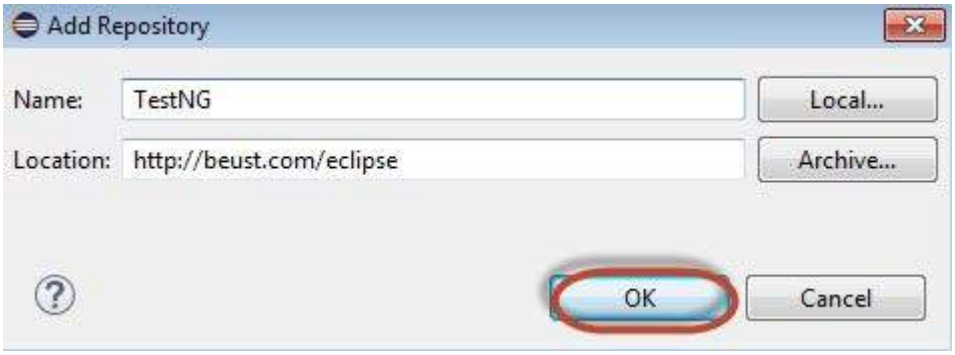


Step 2 : Enter the URL as 'http://beust.com/eclipse' and click 'Add'.

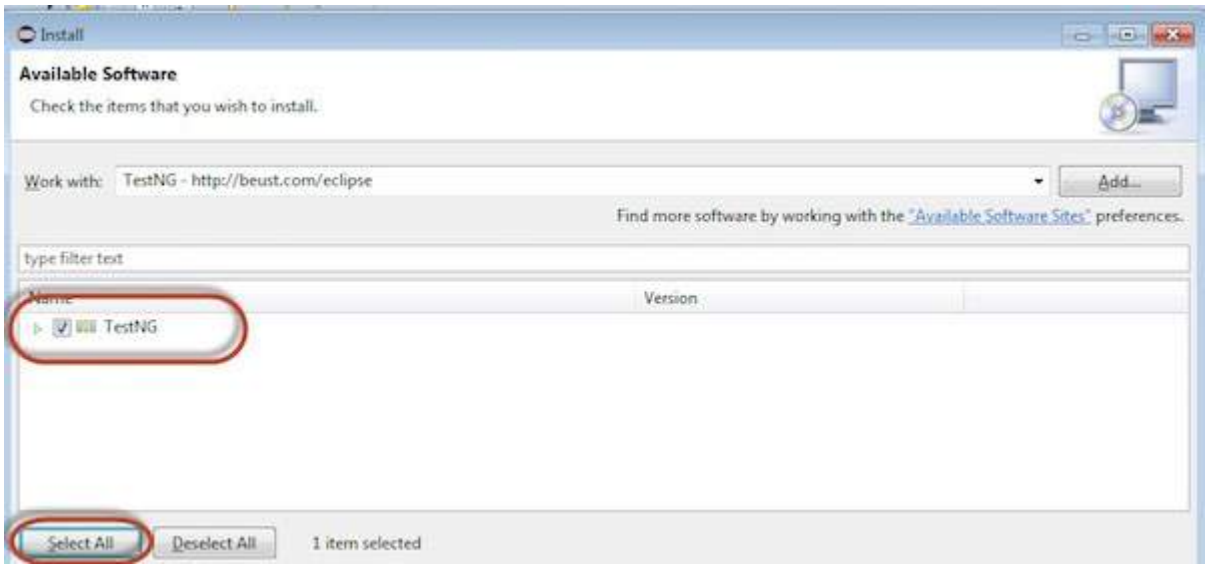




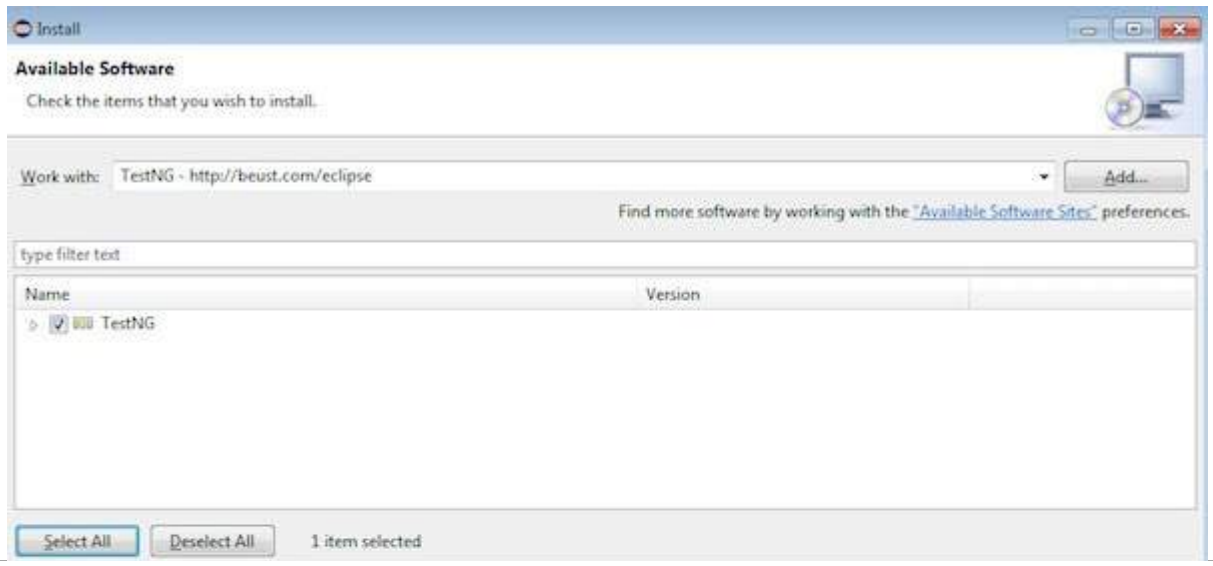
Step 3 : The dialog box 'Add Repository' opens. Enter the name as 'TestNG' and click 'OK'

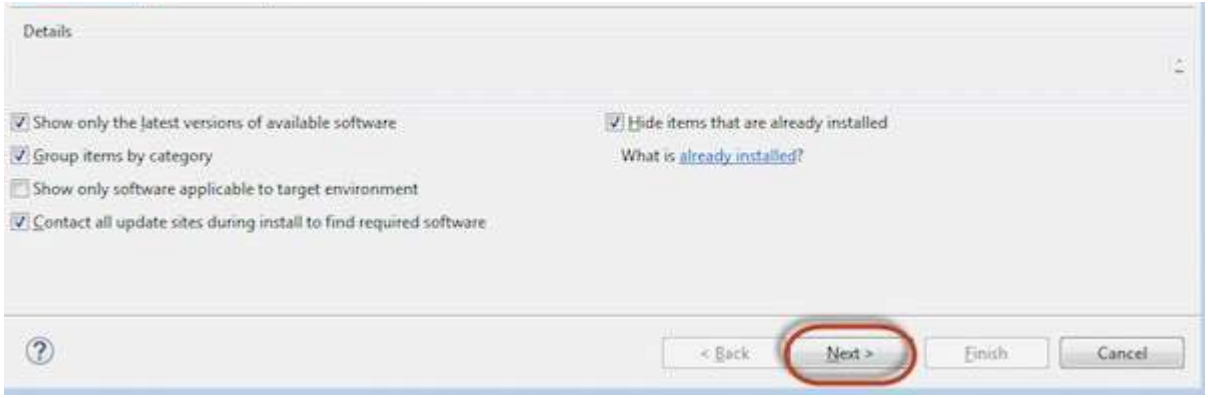


Step 4 : Click 'Select All' and 'TestNG' would be selected as shown in the figure.

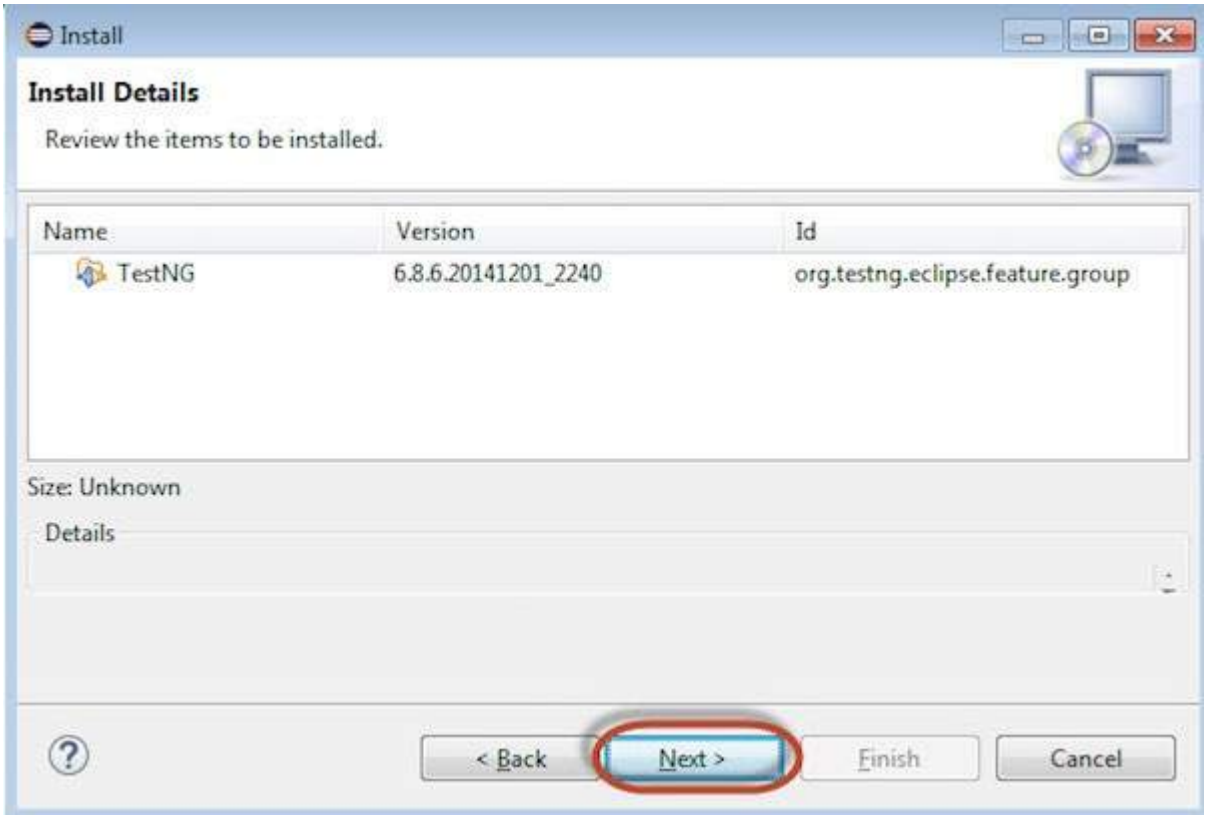


Step 5 : Click 'Next' to continue.

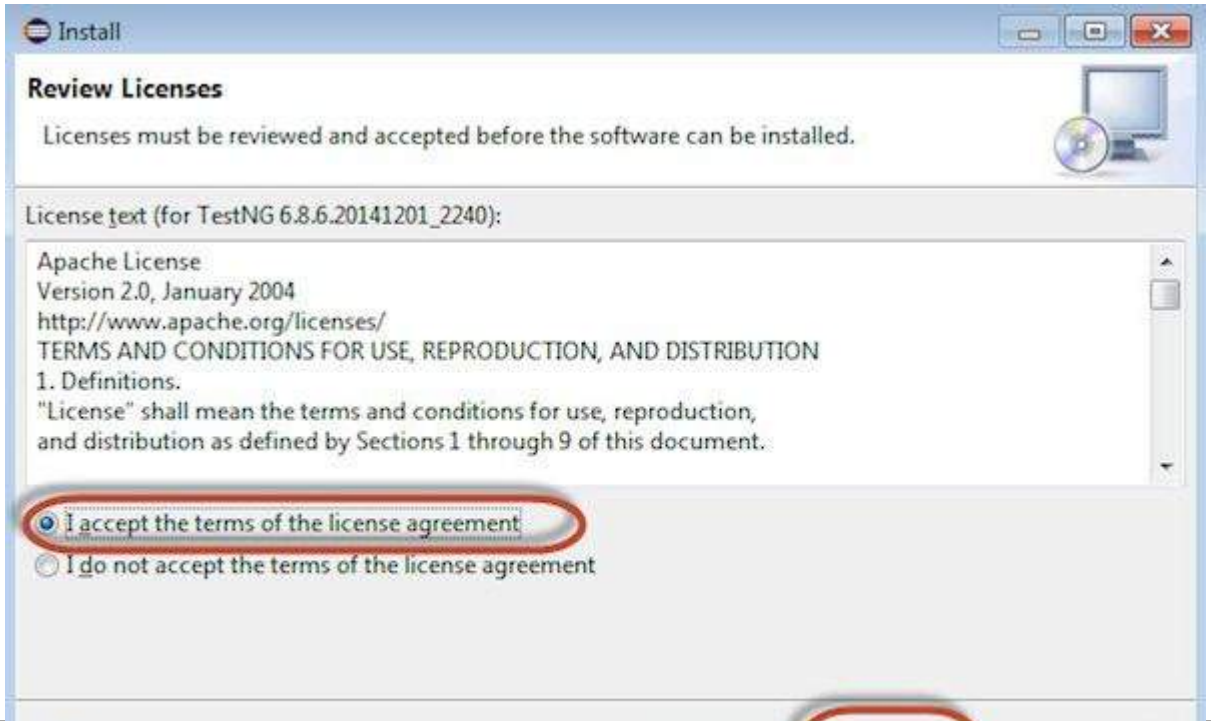




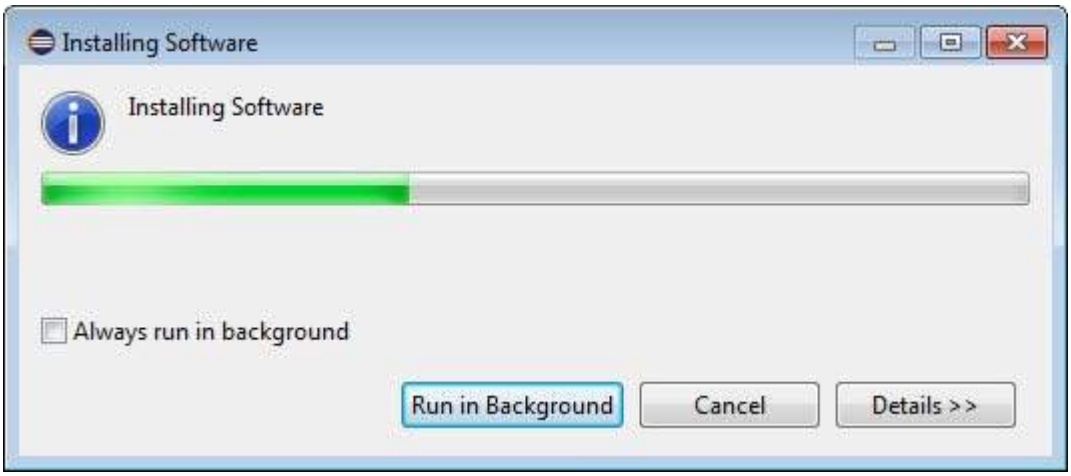
Step 6 : Review the items that are selected and click 'Next'.



Step 7 : "Accept the License Agreement" and click 'Finish'.

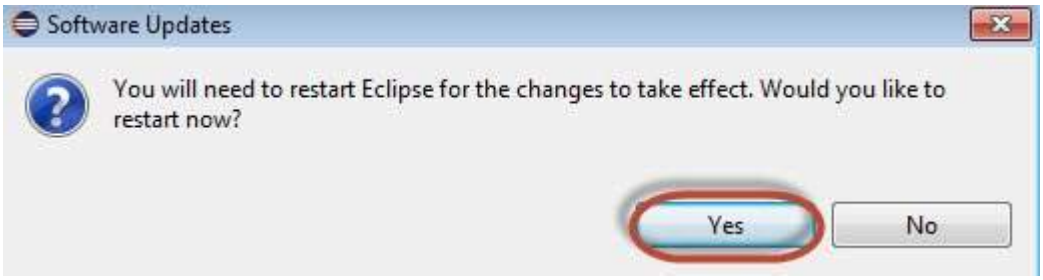


Step 8 : TestNG starts installing and the progress would be shown follows.



Step 9 : Security Warning pops up as the validity of the software cannot be established. Click 'Ok'.

Step 10 : The Installer prompts to restart Eclipse for the changes to take effect. Click 'Yes'.



Annotations in TestNG

Annotations were formally added to the Java language in JDK 5 and TestNG made the choice to use annotations to annotate test classes. Following are some of the benefits of using annotations. More about TestNG can be found **here**

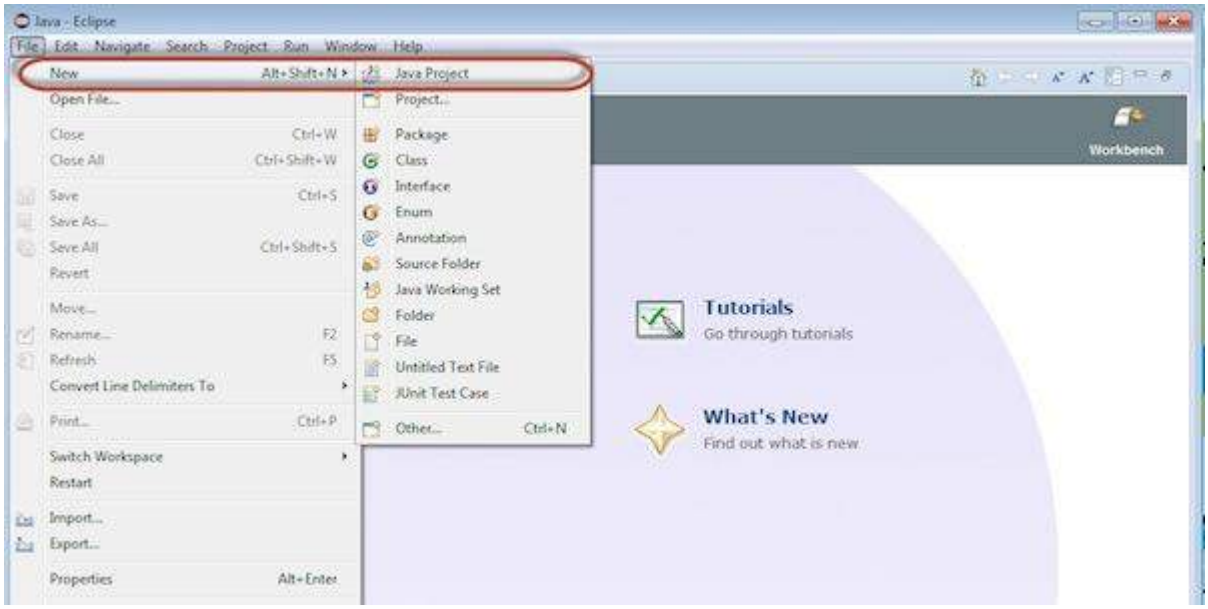
- TestNG identifies the methods it is interested in by looking up annotations. Hence, method names are not restricted to any pattern or format.
- We can pass additional parameters to annotations.
- Annotations are strongly typed, so the compiler will flag any mistakes right away.
- Test classes no longer need to extend anything *suchasTestCase*, *forJUnit*3.

Annotation	Description
@BeforeSuite	The annotated method will be run only once before all the tests in this suite have run.

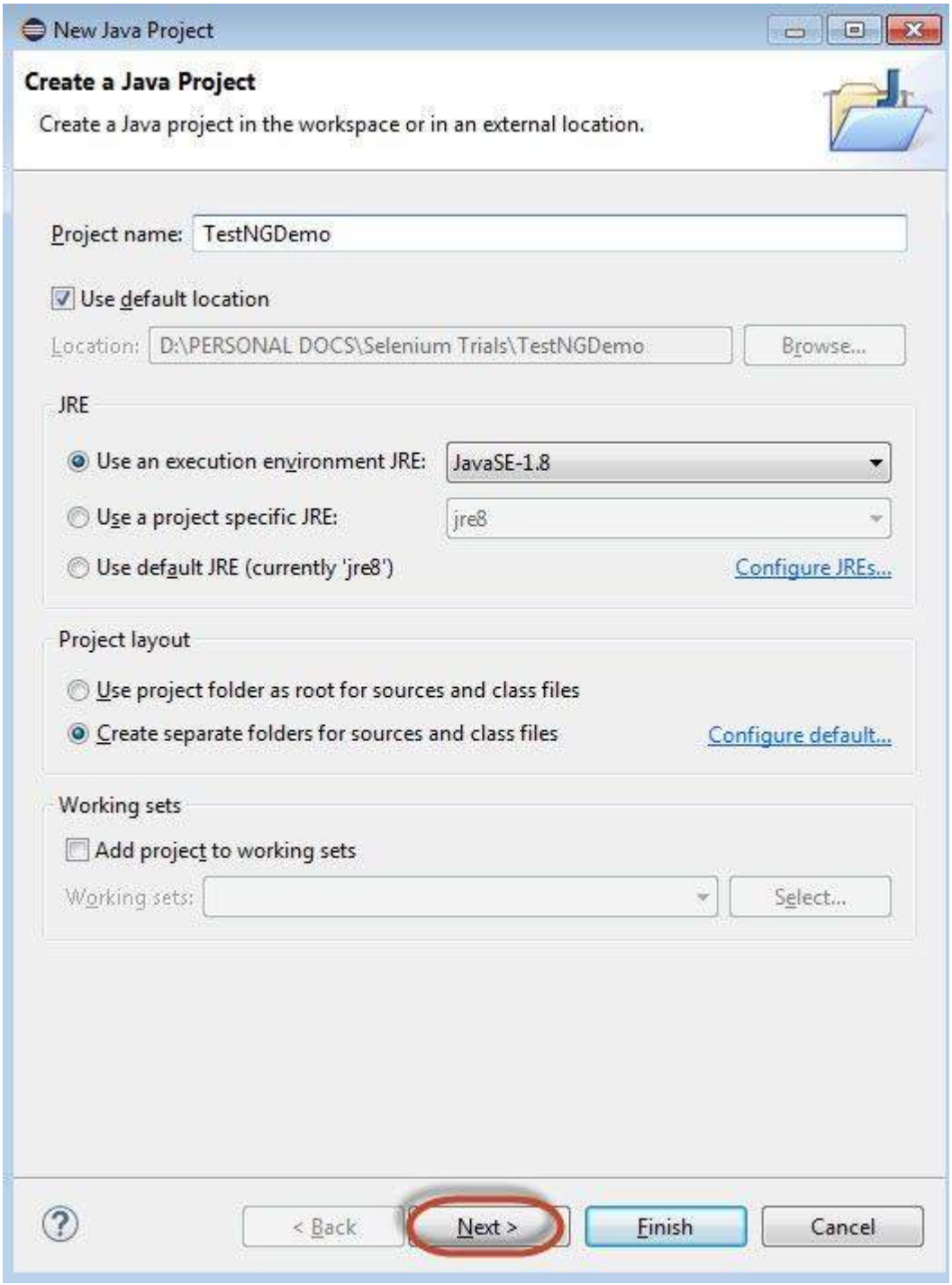
@AfterSuite	The annotated method will be run only once after all the tests in this suite have run.
@BeforeClass	The annotated method will be run only once before the first test method in the current class is invoked.
@AfterClass	The annotated method will be run only once after all the test methods in the current class have run.
@BeforeTest	The annotated method will be run before any test method belonging to the classes inside the <test> tag is run.
@AfterTest	The annotated method will be run after all the test methods belonging to the classes inside the <test> 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.
@BeforeMethod	The annotated method will be run before each test method.
@AfterMethod	The annotated method will be run after each test method.
@DataProvider	Marks a method as supplying data for a test method. The annotated method must return an Object[ ][ ] where each Object[ ] can be assigned the parameter list of the test method. The @Test method that wants to receive data from this DataProvider needs to use a dataProvider name equals to the name of this annotation.
@Factory	Marks a method as a factory that returns objects that will be used by TestNG as Test classes. The method must return Object[ ].
@Listeners	Defines listeners on a test class.
@Parameters	Describes how to pass parameters to a @Test method.
@Test	Marks a class or a method as part of the test.

TestNG-Eclipse Setup

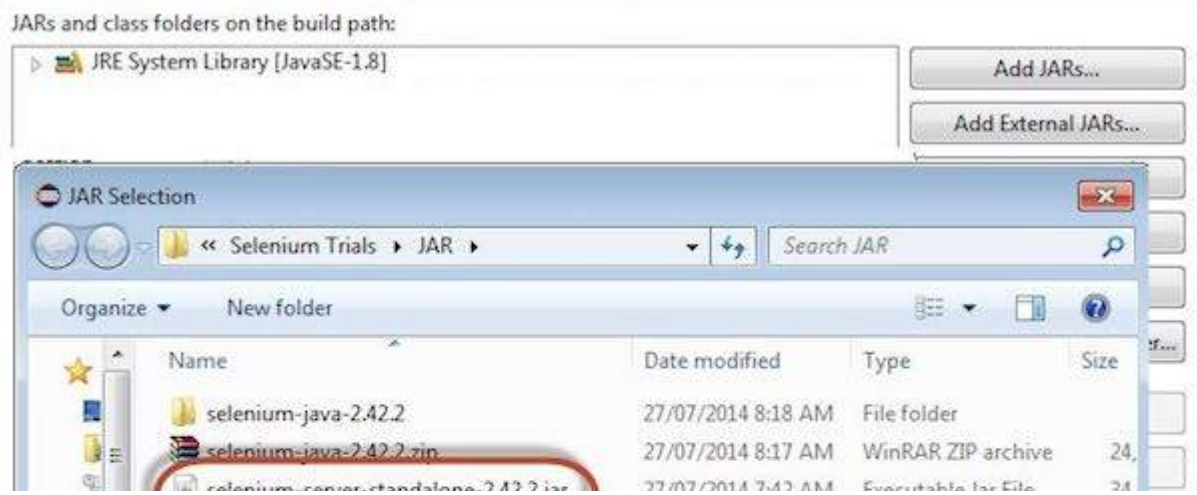
Step 1 : Launch Eclipse and create a 'New Java Project' as shown below.

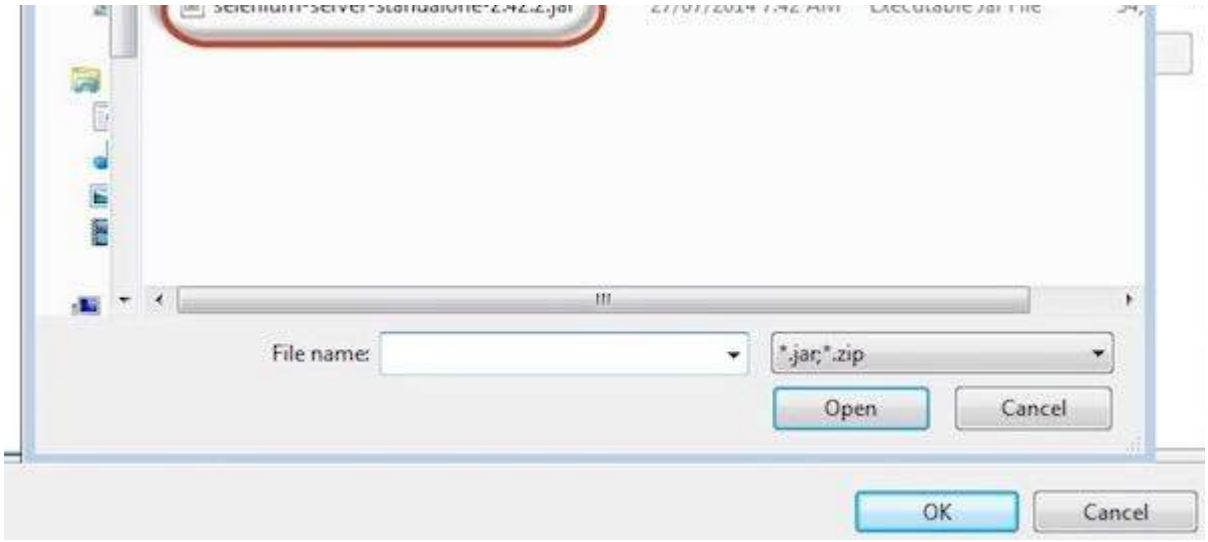


Step 2 : Enter the project name and click 'Next'.

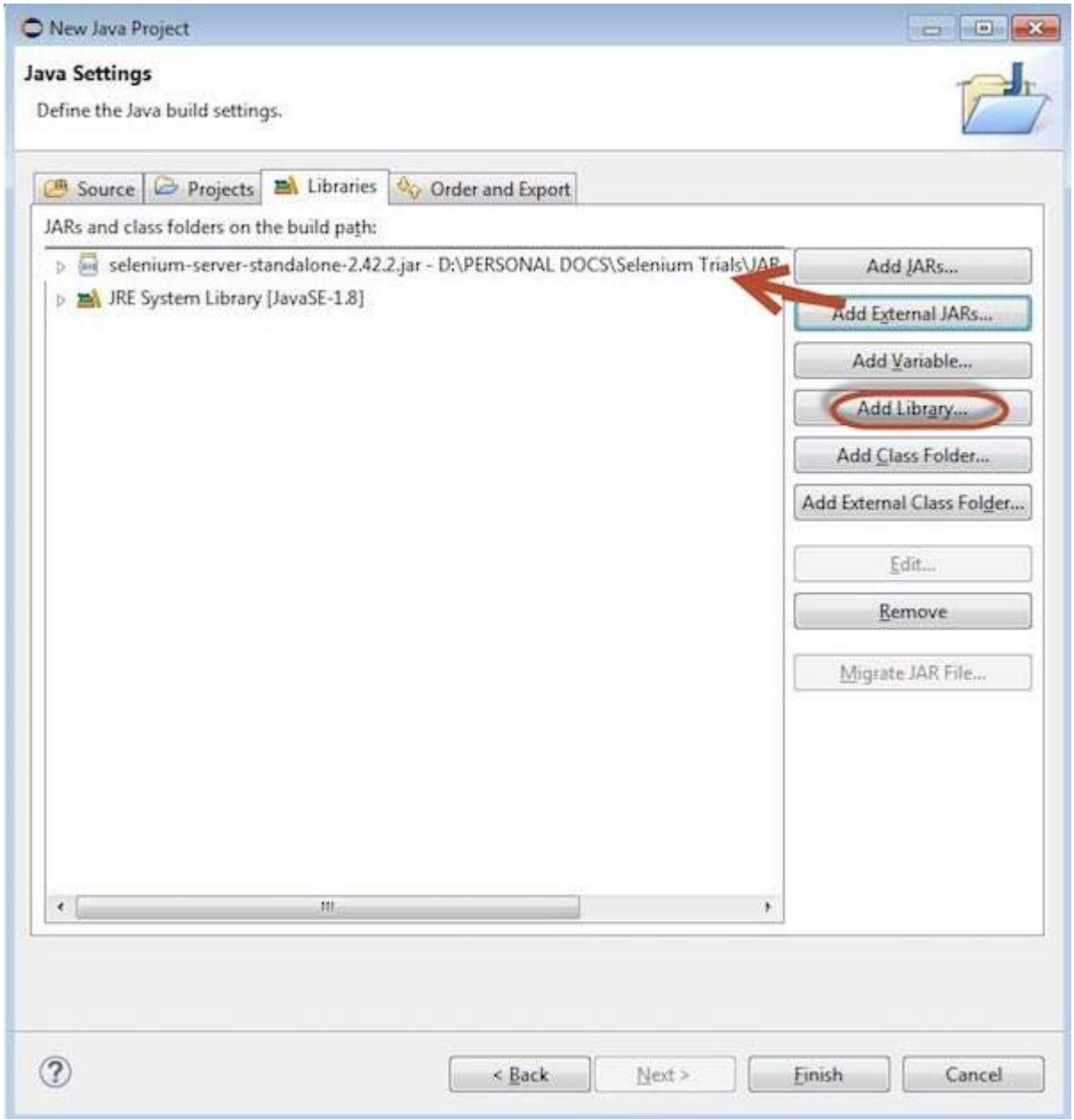


Step 3 : Navigate to "Libraries" Tab and Add the Selenium Remote Control Server JAR file by clicking on "Add External JAR's" as shown below.

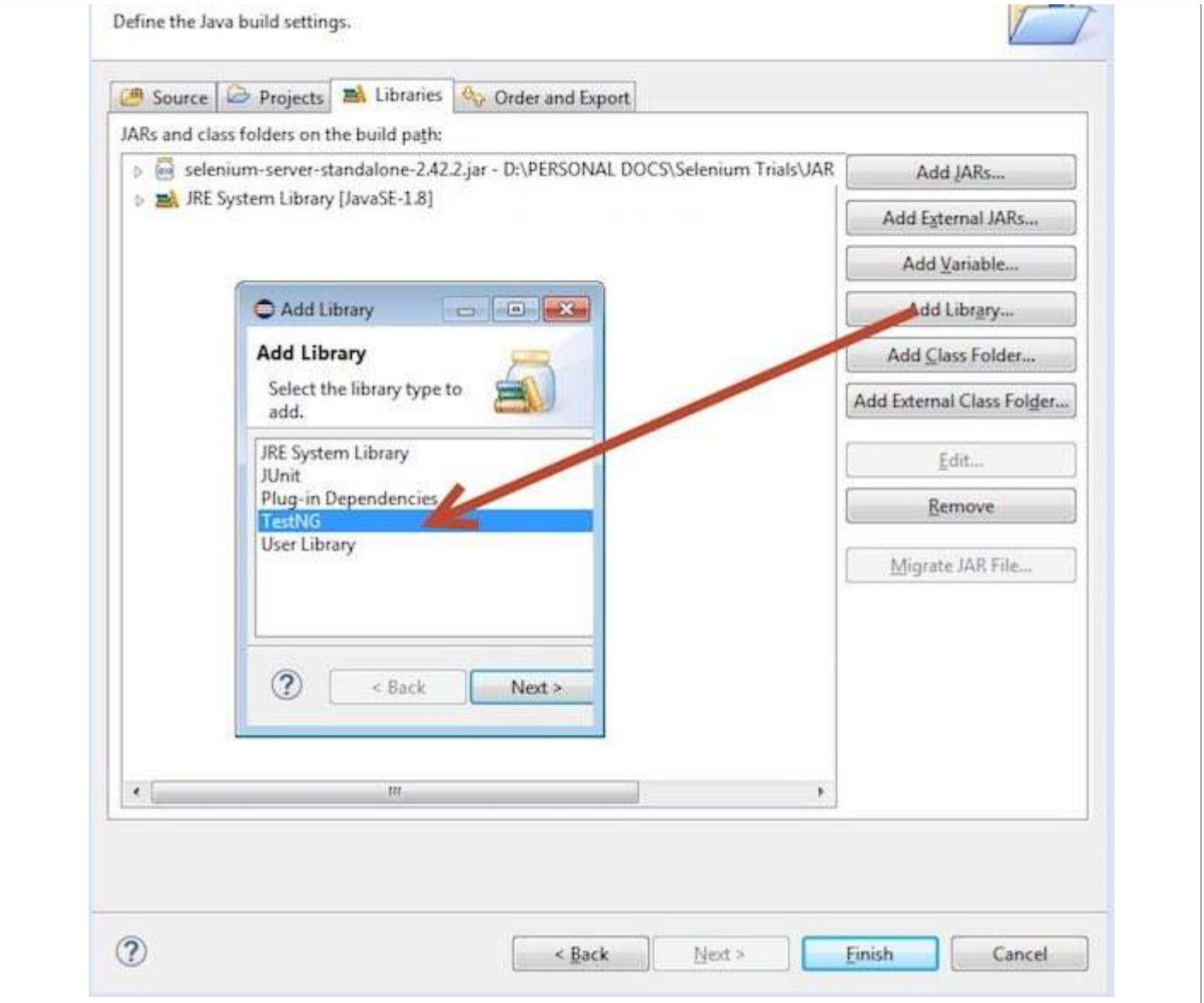




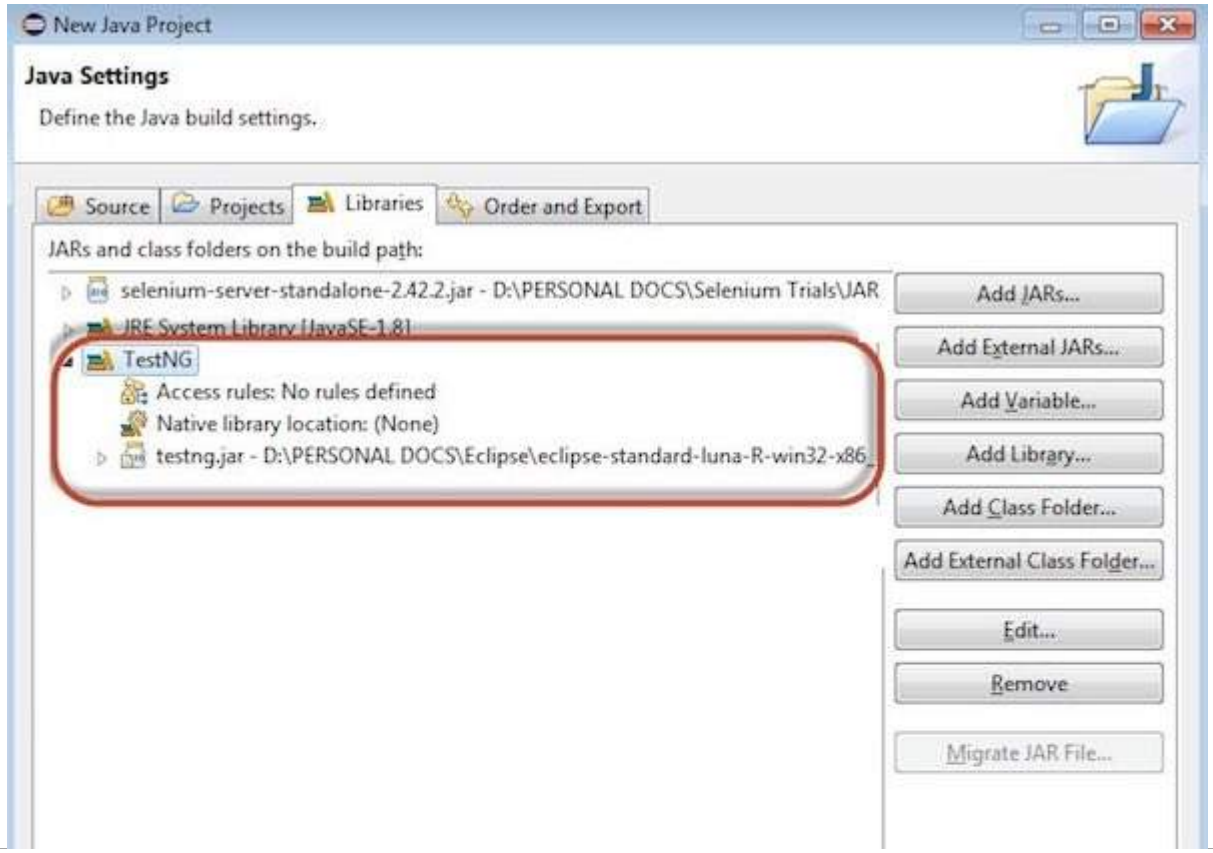
Step 4 : The added JAR file is shown here. Click 'Add Library'.

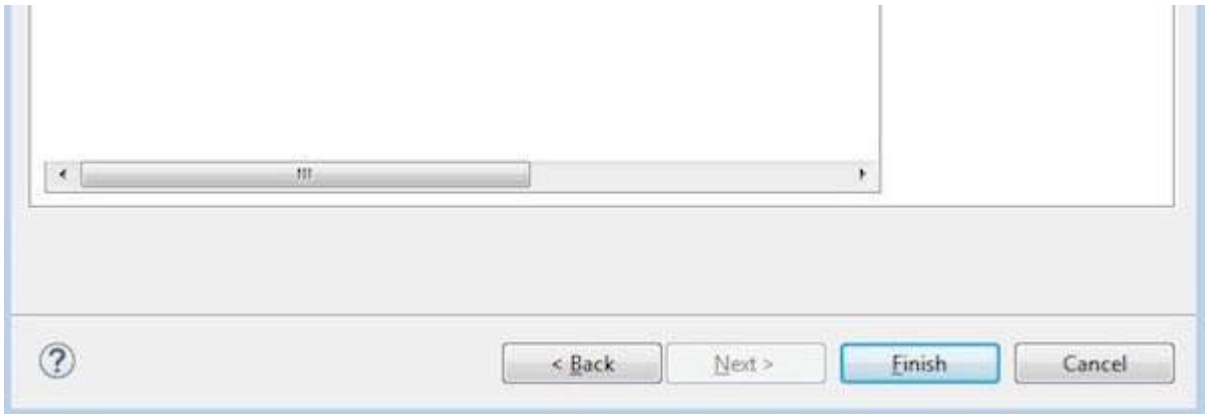


Step 5 : The 'Add Library' dialog opens. Select 'TestNG' and click 'Next' in the 'Add Library' dialog box.

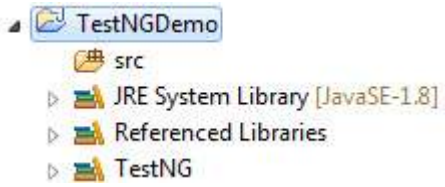


Step 6 : The added 'TestNG' Library is added and it is displayed as shown below.

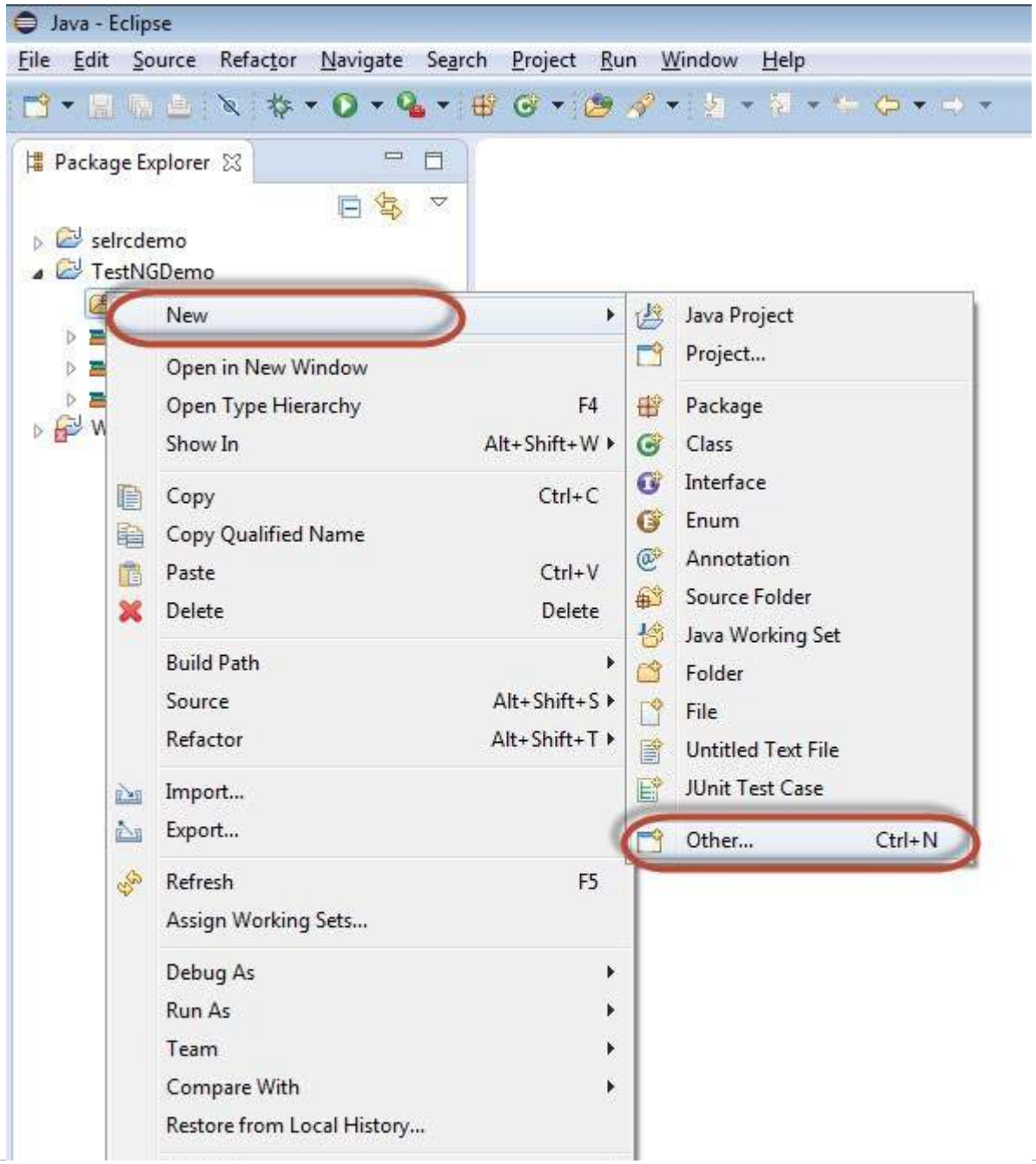




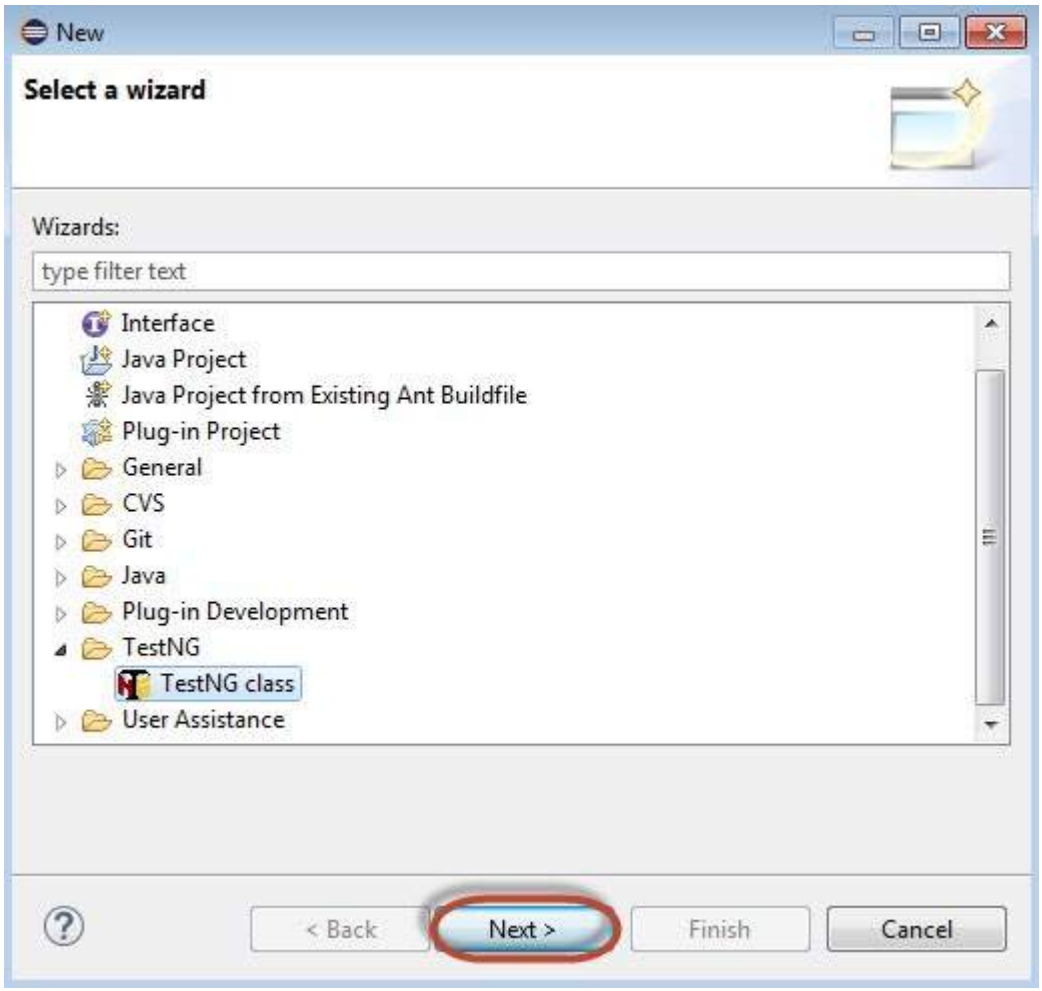
Step 7 : Upon creating the project, the structure of the project would be as shown below.



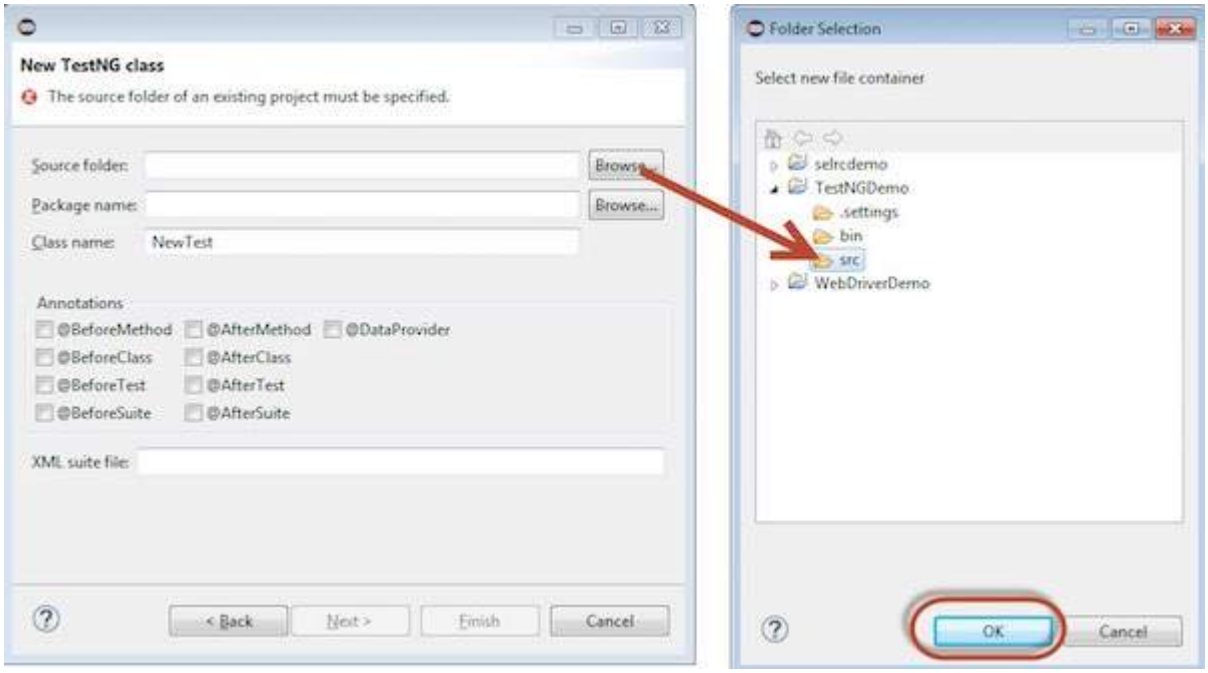
Step 8 : Right-click on 'src' folder and select New >> Other.



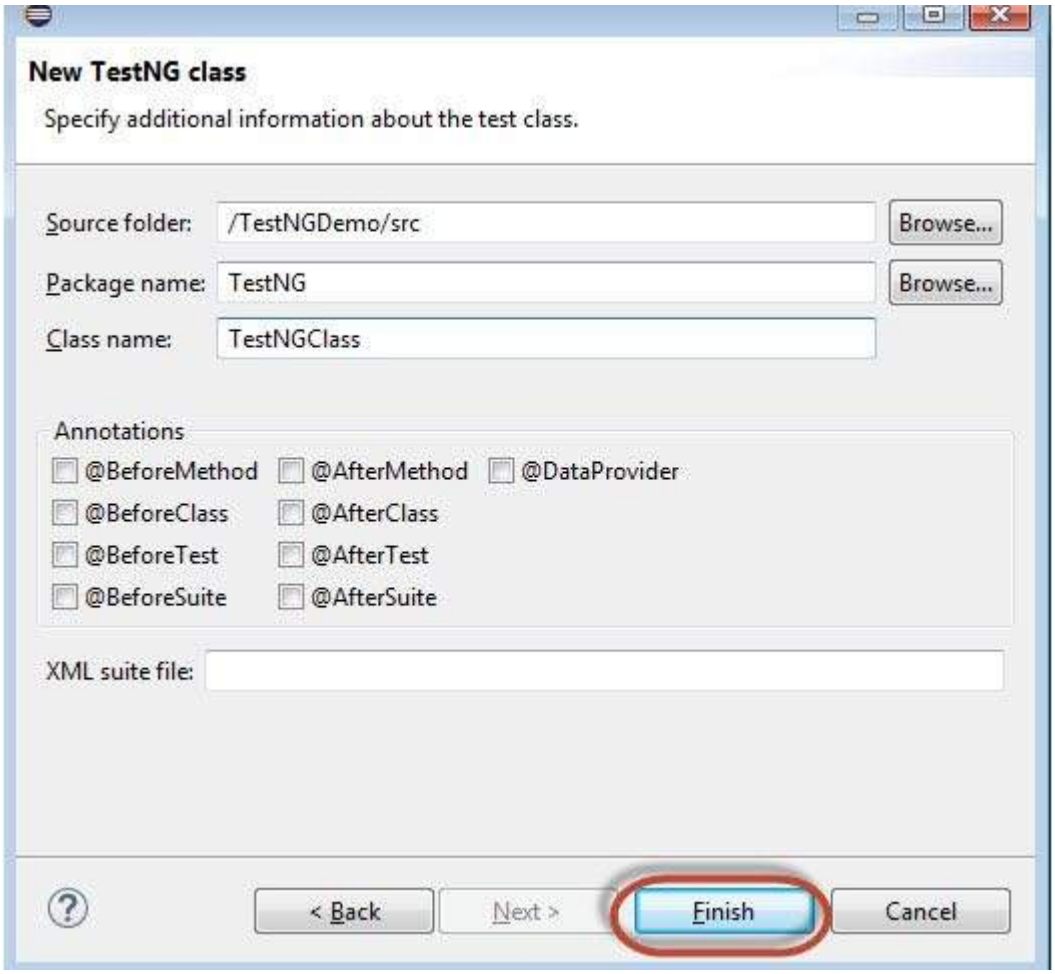
Step 9 : Select 'TestNG' and click 'Next'.



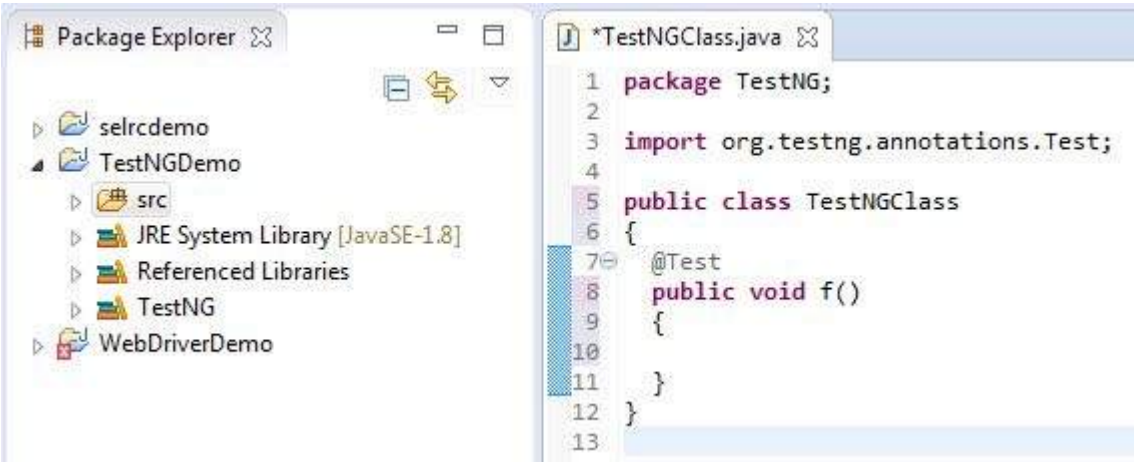
Step 10 : Select the 'Source Folder' name and click 'Ok'.



Step 11 : Select the 'Package name', the 'class name', and click 'Finish'.



Step 12 : The Package explorer and the created class would be displayed.



First Test in TestNG

Now let us start scripting using TestNG. Let us script for the same example that we used for understanding the WebDriver. We will use the demo application, www.calculator.net, and perform percent calculator.

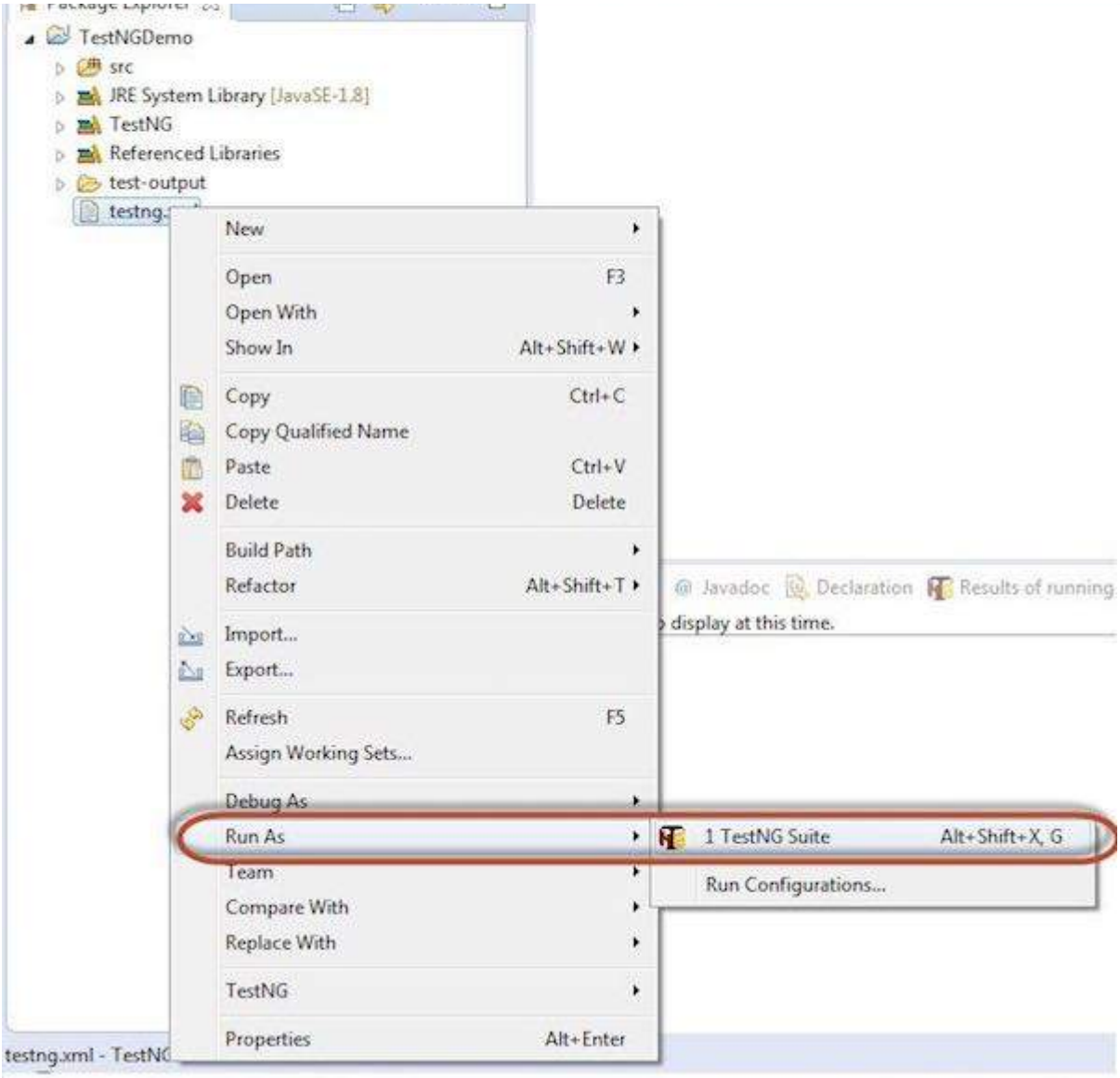
In the following test, you will notice that there is NO main method, as testNG will drive the program execution flow. After initializing the driver, it will execute the '@BeforeTest' method followed by '@Test' and then '@AfterTest'. Please note that there can be any number of '@Test' annotation in a class but '@BeforeTest' and '@AfterTest' can appear only once.

```
package TestNG;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.*;
import org.openqa.selenium.firefox.FirefoxDriver;
```

```
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;

public class TestNGClass
{
   WebDriver driver = new FirefoxDriver();

   @BeforeTest
   public void launchapp()
   {
      // Puts an Implicit wait, Will wait for 10 seconds before throwing exception
      driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
      // Launch website
      driver.navigate().to("http://www.calculator.net");
      driver.manage().window().maximize();
   }
   @Test
   public void calculatepercent()
   {
      // Click on Math Calculators
      driver.findElement(By.xpath(".//*[@id='menu']/div[3]/a")).click();

      // Click on Percent Calculators
      driver.findElement(By.xpath(".//*[@id='menu']/div[4]/div[3]/a")).click();

      // Enter value 10 in the first number of the percent Calculator
      driver.findElement(By.id("cpar1")).sendKeys("10");

      // Enter value 50 in the second number of the percent Calculator
      driver.findElement(By.id("cpar2")).sendKeys("50");

      // Click Calculate Button

driver.findElement(By.xpath(".//*[@id='content']/table/tbody/tr/td[2]/input")).click();

      // Get the Result Text based on its xpath
      String result =
driver.findElement(By.xpath(".//*[@id='content']/p[2]/span/font/b")).getText();

      // Print a Log In message to the screen
      System.out.println(" The Result is " + result);

      if(result.equals("5"))
      {
         System.out.println(" The Result is Pass");
      }
      else
      {
         System.out.println(" The Result is Fail");
      }
   }
   @AfterTest
   public void terminatetest()
   {
      driver.close();
   }
}
```

Execution

To execute, right click on the created XML and select "Run As" >> "TestNG Suite"

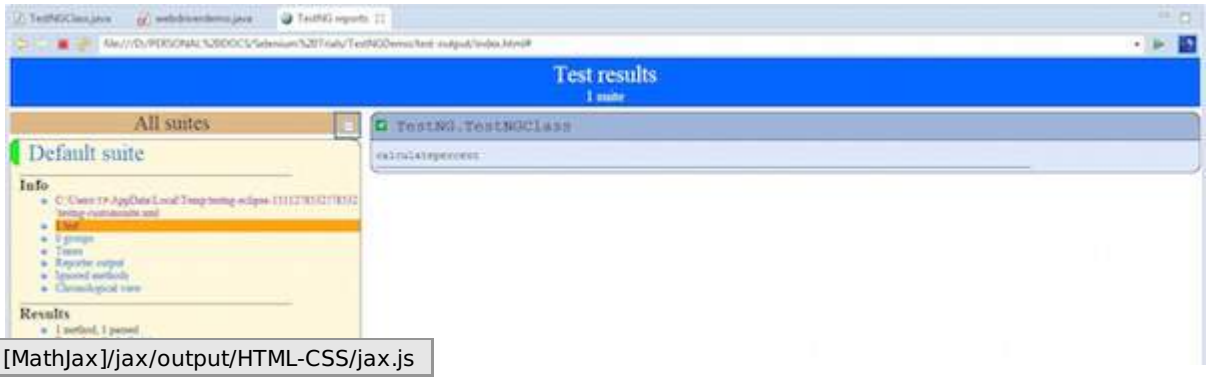


Result Analysis

The output is thrown to the console and it would appear as shown below. The console output also has an execution summary.

The result of TestNG can also be seen in a different tab. Click on 'HTML Report View' button as shown below.

The HTML result would be displayed as shown below.



Loading [MathJax]/jax/output/HTML-CSS/jax.js