**Q. Purposes and Features:**

* + Open source, freely available in the market.
  + Its available in the form of JAR.
  + Design test cases in a systematic way.
  + **Generate very good html report**
  + Gives lots of different annotation for executing methods in test cases.
  + Manage test suite and test cases.
  + **Have priorities/sequence** (Which should be execute first or which should be execute last).

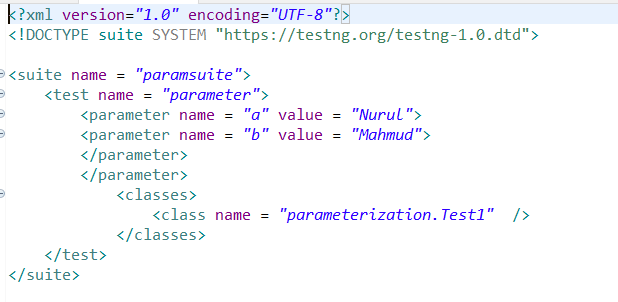
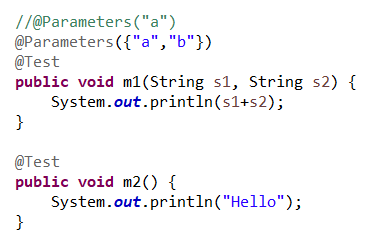
@Test(priority=1)

* + **Have dependency means** one test case is depend on another test case. Suppose login method is depend of OpenUrl. If url is open successfully then login will be passed. What happen here if openUrl is failed then login will be skipped.
  + @Test (priority = 2, dependsOnMethods = {"openUrl"})
  + dependsOnMethod==”loginTest”. Next test is depended on previous test. If previous test getting failed then next test will not be executed.
  + **Grouping (suppose you have 100 test case; you can group the test cases).**

Suppose 10 test cases for sanity test, 10 test cases for regression test and make them group.

* + Disabling Test method: suppose we have 4 test method. Out of 4 we want to run only 2. So, how I will do this? @Test(priority = 3, enabled = **false**) //This method will not run
  + **Most important feature is data provider feature @DataProvider(name =”users”)**
* Let’s see you want to execute your test case multiple times with different set of data.
* Suppose a login feature is there, you want to pass user name and password again and again.
* You have 10 user name and password and you want to execute same test cases again and again with 10 set of data.
* That case data provider is the amazing feature available in TestNG.
* If you using TestNG with Selenium, then data provider you have to use.
* Life will be very easy with TestNG, you just need to concentrate with your logic and the entire thing will be taken care by TestNG.
* That’s why when we designing any automation framework, we always use TestNG to write the test cases.
* **We can define groups also. Suppose we have 100 test cases. 10 test cases are related to search page.**
* @Test(priority-1, groups-title)
* @Test(priority-2, groups-logo)
* @Test(priority-3, groups-link test)
* **InvocationCountTest---If we want to execute same test case 10 times/multiple times then no need to write different method 10 times. Then we have @Test(invocationCount=10).**
* ExceptionTimeOutTest: expectedExceptions = NumberFormatException.class instead of try-catch.

**Q. How to pass parameter in TestNG?** Through XML file

**Q. What is the way of parameterization? Parallel testing is achieved through parameterization. Execute test case in multiple browsers.** Through xml file and Through method---@Data Provider

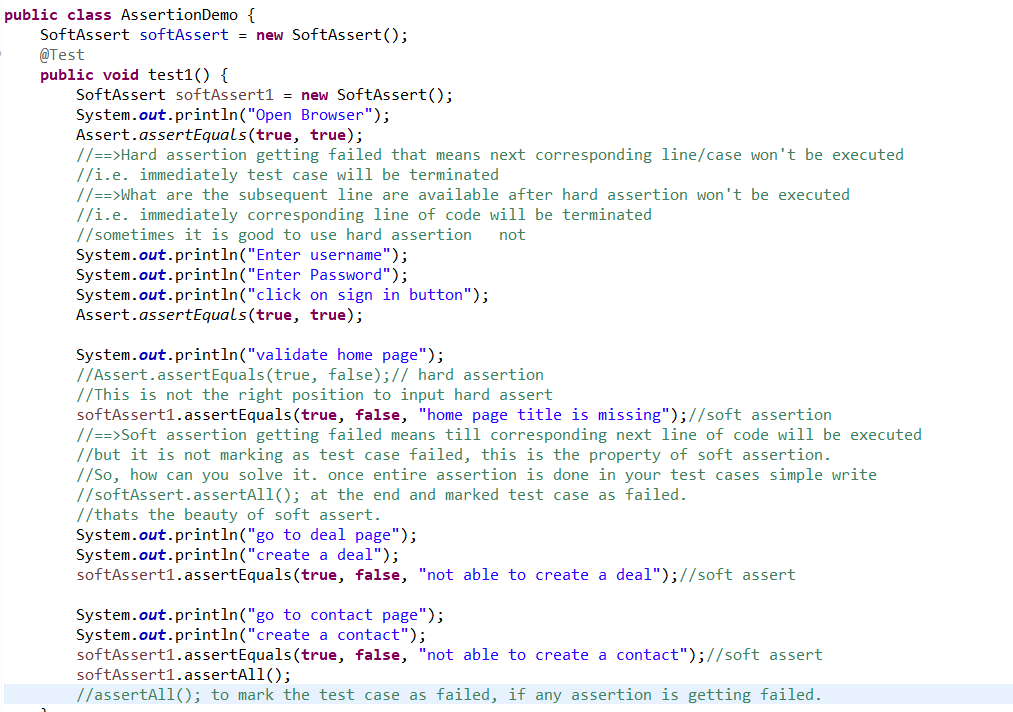
****

**Q. Why TestNG doesn’t need main () method?**

**TestNG.class**contains **main ()**actually which calls suite internally to execute. So, you do not need to write **main ()**method in a TestNG class to run it as TestNG takes care of that by defining annotations. You just need to provide proper annotations to methods and rest TestNG will execute them in a manner by implicit call to main method of TestNG class.

**Q. The concepts used in this documentation are as follows:**

* A suite is represented by one XML file. It can contain one or more tests and is defined by the <suite> tag.
* A test is represented by <test> and can contain one or more TestNG classes.
* A TestNG class is a Java class that contains at least one TestNG annotation. It is represented by the <class> tag and can contain one or more test methods.
* A test method is a Java method annotated by @Test in your source.
* **Validation is the most important think in any test cases. We have to put validation in the form of Assertion.**
* **What is Assertion?**
* Assertion is kind of validation.
* We have one class—Assert.
* We have method-assertEquals ().
* Assert.assertEquals(actual, expected);
* Assert.assertEquals(title, “Google”);
* In this case, no need to write if else condition.
* If the result not matched, we can write one message (title, “Google”, “title is not found”);
* **Assert.assertTrue(b);**
* **Assert.assertEquals(b, “true”);**
* **Assert.fail();**



**TestNG provides a bunch of listeners as a part of its testing environment. These listeners are as follows:**

ITestListener-interface

IReporter

ISuiteListener

IInvokedMethod

IHookable

IConfigurationListener

IConfigurableListener

IAnnotationTransformer

IExecution

IMethodIntercep