**Listeners:**

It is a part of testNG. So it doesn’t require any external jar files

Listeners gives us the ability to act before and after of every **Suite, Test and Methods** (All the methods available in the class file)

Logging of information will be done in **testNG report and in console o/p**.

Listeners are similar to that of interface.

* **IsuiteListener** 🡪 OnStart() and OnFinish() 🡪(<Suite>,</Suite>)
* **ITestListener** 🡪

1. OnStart(), 🡪 <Test>
2. OnFinish(),🡪 </Test>
3. Onteststart(), 🡪 @Test 🡪 Tell whether the @Test has started
4. onTestSuccess(),🡪 @Test 🡪Validated success
5. OnTestFailure(), 🡪 @Test🡪Validated failure
6. OnTestSkipped(), 🡪 @Test🡪Validated skipped
7. onTestFailedButWithinSuccessPercentage()🡪 @Test 🡪 partial pass status

* **IInvokedMethodListener** 🡪 BeforeInvocation() and AfterInvocation() 🡪 Applied to all the methods available in the class(independent of annotations)

**Syntax:**

The below statement should be mentioned with in the methods of the above listeners.

**Reporter.log(“message”, true/false);**

**True🡪 displayed in the console and in testNG report**

**False 🡪 it will not be displayed any where**

Inside the testng xml we need to mention listeners commonly in the <Suite> tag level. So that whatever the class file we try to run the listeners will be applied.

**Syntax:**

**<listeners>**

**<listener class-name="packagename.classname"></listener>**

**</listeners>**

**Types of Listeners:**

**ISuiteListener**:

It has two method in it **onStart()** & **onFinish()**. Whenever a class implements this listener, TestNG guarantees the end-user that it will invoke the methods onStart() and onFinish() before and after running a TestNG Suite. So before TestNG picks up your suite for execution, it first makes a call to onStart() method and runs whatever has been scripted in this method. In a similar way, it again makes a call to onFinish() method after a suite has been run.

**Example:** with in a testng.xml <suite>

**ITestListener**:

The working of this listener is also exactly the same as ISuiteListerner but the only difference is that it makes the call before and after the Test not the Suite.

* **onFinish()**: Invoked after all the tests have run and all their Configuration methods have been called.
* **onStart():**Invoked after the test class is instantiated and before any configuration method is called.
* **onTestFailedButWithinSuccessPercentage(ITestResult result):**Invoked each time a method fails but has been annotated with successPercentage and this failure still keeps it within the success percentage requested.
* **onTestFailure(ITestResult result):** Invoked each time a test fails.
* **onTestSkipped(ITestResult result):**Invoked each time a test is skipped
* **onTestStart(ITestResult result):**Invoked each time before a test will be invoked.
* **onTestSuccess(ITestResult result):**Invoked each time a test succeeds.

**Example:** what it will do if @Test fails/pass

**IInvokedMethodListener**:

The working of this listener is also exactly the same as ISuiteListerner & ITestListerner and the only difference is that it makes the call before and after every Method. It has only two methods in it.

* **afterInvocattion():** Invoke after each method
* **beforeInvocation():** Invoke before each method

**Example:**

@Test/@BeforeMethod/@AfterMethod/Class/Suite

Public void method1(){

Afterinvocation/Beforeinvocation

}