Practical No. 11 TestNg Annotations

Date: _____

Aim:

To study TestNg Annotations.

Theory:

Annotation is a feature introduced in Java 5 and is used to add metadata (data about data) to Java source code.

This will allow you to add information to an existing data object in your source code.

It can be applied for classes, methods, variables, and parameters.

Annotations may affect the way different programs or tools use your source code.

There are certain predefined set of annotations defined in Java.

For example, @Override, @Deprecated, @SupressWarnings, and so on, but Java allows users to define their own annotations too.

TestNg makes use of the same feature provided by Java to define its own annotations and build an execution framework by using it.

The following is a table containing information about all the annotations provided by TestNG and a brief description of them:

Annotation	**Description**
@BeforeSuite or @AfterSuite	The annotated method will be executed before and after any tests declared inside a TestNG suite.
@BeforeTest or @AfterTest	The annotated methods will be executed before and after each test section declared inside a TestNG suite.
@BeforeGroups or @AfterGroups	These annotations are associated with the groups feature in TestNG. BeforeGroups annotated method will run before any of the test method of the specified group is executed.

	AfterGroups annotated method will run after any of the test method of the specified group gets executed. For this method to be executed, the user has to mention the list of groups this method belongs to using groups attribute with the said annotation. You can specify more than multiple groups if required.
@BeforeClass or @AfterClass	BeforeClass annotated method is executed before any of the test method of a test class. AfterClass annotated method is executed after the execution of every test methods of a test class are executed.
@BeforeMethod or @AfterMethod	These annotated methods are executed before/after the execution of each test method.
@DataProvider	Marks a method as a data providing method for a test method. The said method has to return an Object double array (Object[][ ]) as data.
@Factory	Marks a annotated method as a factory that returns an array of class objects (Object[ ]). These class objects will then be used as test classes by TestNG. This is used to run a set of test cases with different values.
@Listeners	Applied on a test class. Defines an array of test listeners classes extending org.testng.ITestNGListener. Helps in tracking the execution status and logging purpose.
@Parameters	This annotation is used to pass parameters to a test method. These parameter values are provided using the testng.xml configuration file at runtime.
@Test	Marks a class or a method as a test method. If used at class level, all the public methods of a class will be considered as a test method

Test annotation

One of the basic annotations of TestNG is the Test annotation.

This annotation marks a method or a class as part of the TestNG test.

If applied at class level this annotation will mark all the public methods present inside the class as test methods for TestNG test.

It supports lot of attributes which you can use along with the annotation, which will enable you to use the different features provided by TestNG.

The following is a list of attributes supported by the Test annotation:

Supported attributes	**Description**
alwaysRun	Takes a true or false value. If set to true this method will always run even if its depending method fails.
dataProvider	The name of the data provider, which will provide data for data-driven testing to this method.
dataProviderClass	The class where TestNG should look for the dataprovider method mentioned in the dataProvider attribute. By default its the current class or its base classes.
dependsOnGroups	Specifies the list of groups this method depends on.
dependsOnMethods	Specifies the list of methods this method depends on.
description	The description of this method
enabled	Sets whether the said method or the methods inside the said class should be enabled for execution or not. By default, its value is true.
expectedExceptions	This attribute is used for exception testing. This attribute specifies the list of exceptions this method is expected to throw. In case a different exception is thrown.
groups	List of groups the said method or class belongs to.

timeOut	This attribute is used for a time out test and specifies the time (in millisecs) this method should take to execute.

Parameterization of test

One of the important features of TestNG is parameterization.

This feature allows user to pass parameter values to test methods as arguments. This is supported by using the Parameters and DataProvider annotations.

There are mainly two ways through which we can provide parameter values to test-methods:

Through testng XML configuration file

Through DataProviders

Parameterization through testng.xml

If you need to pass some simple values such as String types to the test methods at runtime, you can use this approach of sending parameter values through TestNG XML configuration files.

You have to use the Parameters annotation for passing parameter values to the test method.

DataProvider

One of the important features provided by TestNG is the DataProvider feature.

It helps the user to write data-driven tests, that means same test method can be run multiple times with different datasets.

DataProvider is the second way of passing parameters to test methods. It helps in providing complex parameters to the test methods as it is not possible to do this from XML.

To use the DataProvider feature in your tests you have to declare a method annotated by DataProvider and then use the said method in the test method using the dataProvider attribute in the Test annotation.

Groups

Grouping test methods is one of the most important features of TestNG.

In TestNG users can group multiple test methods into a named group.

You can also execute a particular set of test methods belonging to a group or multiple groups.

This feature allows the test methods to be segregated into different sections or modules.

For example, you can have a set of tests that belong to sanity test where as others may belong to regression tests.

You can also segregate the tests based on the functionalities/features that the test method verifies. This helps in executing only a particular set of tests as and when required

Implementation

1. Create a test class with @BeforeClass/@AfterClass, @BeforeMethod/@AfterMethod annotations and execute it using a testng.xml
2. Create and execute a TestNG class using test annotation on class. The class contains two public methods and one private method.
3. Create TestNG class containing three test methods using test annotation out of which any two methods are enabled and remaining method is disabled. Use appropriate attributes of test annotation.
4. Create a test class with @BeforeSuite/@AfterSuite, @BeforeTest/@AfterTest annotations and execute it using a testng.xml.
5. Create a test class that contains four test methods. Two of which should belong to one group and the remaining two to another group. Create testing.xml file to execute tests in a particular group.
6. Write a test class containing test method that calculates the average marks that awarded by two reviewers prints whether writer is shortlisted if average is >4. The marks are passed as parameters whose values are passed from testing.xml at test level.
7. Write a test class containing test method that prints the value of parameters which are passed from data provider.

Conclusion: Understood how to use TestNG Annotations.

After performing this Practical/lab, students are expected to answer following questions

Q.1 What are TestNG Annotations?

Q.2 What is data provider?

Practical No. 11

TestNg Annotations

1.Create a test class with@BeforeClass/@AfterClass@BeforeMethod/@AfterMethod annotations and execute it using a testng.xml

BeforeAfter.java

```
package beforeafter;

import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

public class BeforeAfter {

	@BeforeMethod
	public void beforeMethod1() {
		System.out.println("Before method");
	}

	@AfterMethod
	public void afterMethod1() {
		System.out.println("After method");
	}

	@AfterClass
	public void afterClass2() {
		System.out.println("After Class method");
	}

	@BeforeClass
	public void beforeClass2() {
		System.out.println("Before Class method");

	}

	@Test
	public void FirstTestMethods() {
		System.out.println("First test method");
	}

	@Test
	public void SecondTestMethods()
	{
		System.out.println("Second test method");
	}
	@Test
	public void ThirdTestMethods()
	{
```

Practical No. 11

TestNg Annotations

```
		System.out.println("Third test method");
	}
	@Test
	public void FourthTestMethods()
	{
		System.out.println("Fourth test method");
	}

}
```

Beforeaftertesting.xml

```
<suite name="Testing Annotation" verbose="1">

		<test name="Test one">
			<classes>
				<class name="beforeafter.BeforeAfter"></class>
			</classes>
		</test>
</suite>
```

Output:

```
Javadoc  Console  Results of running suite
<terminated> AnnotationsDemo_beforeaftertesting.xml [TestNG] C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1.v20211116-1657\jre\bin\j
[RemoteTestNG] detected TestNG version 7.4.0
[TestNGContentHandler] [WARN] It is strongly recommended to add "<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >" a
Before Class method
Before method
First test method
After method
Before method
Fourth test method
After method
Before method
Second test method
After method
Before method
Third test method
After method
After Class method

===============================================
Testing Annotation
Total tests run: 4, Passes: 4, Failures: 0, Skips: 0
===============================================
```

2. Create and execute a TestNG class using test annotation on class. The class contains two public methods and one private method.

TestNgPrivate.java

```
package beforeafter;

import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
```

Practical No. 11

TestNg Annotations

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

public class TestNgPrivate {
        @Test
        public void testMethodOne() {
                System.out.println("First Test Method");
        }
        @Test
        public void testMethodTwo()
        {
                System.out.println("Second Test Method");
        }

        @Test
        private void testMethodThree() {
                System.out.println("Third test method");
        }

}
```

Output:

```
[RemoteTestNG] detected TestNG version 7.4.0
First Test Method
Second Test Method
PASSED: testMethodOne
PASSED: testMethodTwo

===============================================
    Default test
    Tests run: 2, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 2, Passes: 2, Failures: 0, Skips: 0
===============================================
```

---

3. Create TestNG class containing three test methods using test annotation out of which any two methods are enabled and remaining method is disabled. Use appropriate attributes of test annotation.

TestClasses.java

```
package beforeafter;

import org.testng.annotations.Test;

public class TestClasses {
  @Test
```

Practical No. 11

TestNg Annotations

```
  public void testMethodOne() {
          System.out.println("First Test Method");
  }
  @Test(enabled=false)
  public void testMethodTwo()
  {
          System.out.println("Second Test Method");
  }

  @Test(enabled=false)
  public void testMethodThree() {
          System.out.println("Third test method");
  }

}
```

Test-testing.xml

```
<suite name="Test Suite" verbose="1">

<test name="First Test">

        <classes>
                <class name="beforeafter.TestClasses">
                </class>
        </classes>
</test>
</suite>
```

Output:

```
Console  Results of running suite
<terminated> AnnotationsDemo_test-testng.xml [TestNG] C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64
[RemoteTestNG] detected TestNG version 7.4.0
[TestNGContentHandler] [WARN] It is strongly recommended to add "<!DOCTYPE suite SYSTEM "https://t
First Test Method

===============================================
Test Suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
===============================================
```

4. Create a test class with @BeforeSuite/@AfterSuite, @BeforeTest/@AfterTest annotations and execute it using a testng.xml.

Beforesuite_aftersuite.java

Practical No. 11

TestNg Annotations

```
package beforeafter;

import org.testng.annotations.AfterSuite;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeSuite;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;

public class beforesuite_aftersuite {
	@BeforeSuite
	public void beforeSuite() {
		System.out.println("Before Suite method");
	}

	@AfterSuite
	public void afterSuite() {
		System.out.println("After suite method");
	}

	@AfterTest
	public void afterTest() {
		System.out.println("After Test Method");
	}

	@BeforeTest
	public void beforeTest() {
		System.out.println("Before Test Method");
	}
	@Test
	public void FirstTestMethods() {
		System.out.println("First test method");
	}

	@Test
	public void SecondTestMethods()
	{
		System.out.println("Second test method");
	}
	@Test
	public void ThirdTestMethods()
	{
		System.out.println("Third test method");
	}
	@Test
	public void FourthTestMethods()
	{
		System.out.println("Fourth test method");
	}

}
```

Practical No. 11

TestNg Annotations

Beforesuite.xml

```
<suite name="Testing Annotation" verbose="1">

        <test name="Test one">
            <classes>
                <class name="beforeafter.beforesuite_aftersuite"></class>
            </classes>
        </test>

</suite>
```

Output:

```
Console  Results of running suite
<terminated> AnnotationsDemo_beforesuite.xml [TestNG] C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1.v20211116-165
[RemoteTestNG] detected TestNG version 7.4.0
[TestNGContentHandler] [WARN] It is strongly recommended to add "<!DOCTYPE suite SYSTEM "https://testng.org/testng-
Before Suite method
Before Test Method
First test method
Fourth test method
Second test method
Third test method
After Test Method
After suite method

===============================================
Testing Annotation
Total tests run: 4, Passes: 4, Failures: 0, Skips: 0
===============================================
```

---

5. Create a test class that contains four test methods. Two of which should belong to one group and the remaining two to another group. Create testing.xml file to execute tests in a particular group.

GroupsTestMethod.java

```
package beforeafter;

import org.testng.annotations.Test;

public class GroupsTestMethod {
    @Test(groups= {"GroupOne"})
    public void FirstTestMethod() {
        System.out.println("First test method");
    }

    @Test(groups= {"GroupOne"})
    public void SecondTestMethod()
```

Practical No. 11

TestNg Annotations

```
        {
                System.out.println("Second test method");
        }
        @Test(groups= {"GroupTwo"})
        public void ThirdTestMethod()
        {
                System.out.println("Third test method");
        }
        @Test(groups= {"GroupTwo"})
        public void FouthTestMethod()
        {
                System.out.println("Fourth test method");
        }
}
```

Grouptest_methods.xml

```
<suite name="GroupsTest" verbose="1">
        <test name="groups testing">
                <groups>
                        <run>
                                <include name="GroupOne">
                                </include>
                                <include name="GroupTwo">
                                </include>
                        </run>
                </groups>
        <classes>
                <class name="beforeafter.GroupsTestMethod"/>
        </classes>
        </test>
</suite>
```

Output:

```
Console   Results of running suite
<terminated> AnnotationsDemo_grouptest_methods.xml [TestNG] C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.ful
[RemoteTestNG] detected TestNG version 7.4.0
[TestNGContentHandler] [WARN] It is strongly recommended to add "<!DOCTYPE suite SYSTEM "https:
First test method
Fourth test method
Second test method
Third test method

===============================================
GroupsTest
Total tests run: 4, Passes: 4, Failures: 0, Skips: 0
===============================================
```

Practical No. 11

TestNg Annotations

6. Write a test class containing test method that calculates the average marks that awarded by two reviewers prints whether writer is shortlisted if average is >4. The marks are passed as parameters whose values are passed from testing.xml at test level.

ParameterizationDemo.java

```
package beforeafter;

import org.testng.annotations.Parameters;
import org.testng.annotations.Test;

public class ParameterizationDemo {
  @Test
  @Parameters({"reviwer1_marks","reviwer2_marks"})
  public void check_if_shortlisted(int marks1,int marks2) {
            //calculate average of marks given by reviewer 1 and reviewer 2
            float average=(marks1 + marks2)/2;

            System.out.println("The average achieved by the writer is" +average);
            if(average>=4)
            {
                        System.out.println("The writer is shortlisted");
            }
            else
            {
                        System.out.println("The writer is not shortlisted");
            }
  }
}
```

Parametrization-testng.xml

```
<suite name="parameterization " verbose="1">
<parameter name="reviwer1_marks" value="6"/>
          <parameter name="reviwer2_marks" value="4"/>
          <test name="parameterized test">
          <parameter name="reviwer1_marks" value="10"/>
          <parameter name="reviwer2_marks" value="2"/>

                    <classes>
                              <class name="beforeafter.ParameterizationDemo"/>
                    </classes>
          </test>
</suite>
```

Output:

Practical No. 11

TestNg Annotations

```
Console ×  Results of running suite
<terminated> AnnotationsDemo_parameterization-testng.xml [TestNG] C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.fu
[RemoteTestNG] detected TestNG version 7.4.0
[TestNGContentHandler] [WARN] It is strongly recommended to add "<!DOCTYPE suite SYSTEM "https://
The average achieved by the writer is6.0
The writer is shortlisted
[TestHTMLReporter] The system cannot find the path specified

===============================================
parameterization
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
===============================================
```

7. Write a test class containing test method that prints the value of parameters which are passed from data provider.

DataProviderDemo.java

```
package beforeafter;

import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

public class DataProviderDemo {
  @DataProvider(name="dataprovider")
  public Object[][] DataProviderMethod() {
          return new Object[][] {{"data one"},{"data two"}};
  }

  @Test(dataProvider="dataprovider")
  public void testMethod(String data) {
          System.out.println("Data is: " +data);
  }
}
```

Dataprovider-testng.xml

```
<suite name="data provider testing" verbose="1">
<test name="testing data provider">
        <classes>
                <class name="beforeafter.DataProviderDemo"></class>
        </classes>
</test>
</suite>
```

Output:

Practical No. 11

TestNg Annotations

```
Console  Results of running suite
<terminated> AnnotationsDemo_dataprovider-testng.xml [TestNG] C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64
[RemoteTestNG] detected TestNG version 7.4.0
[TestNGContentHandler] [WARN] It is strongly recommended to add "<!DOCTYPE suite SYSTEM "https://testng.o
Data is: data one
Data is: data two

===============================================
data provider testing
Total tests run: 2, Passes: 2, Failures: 0, Skips: 0
===============================================
```