Methods to Test and Test Reports:-

Writing a test in TestNG basically involves the following steps −

- Write the business logic of your test and insert TestNG annotations in your code.
- Add the information about your test (e.g. the class name, the groups you wish to run, etc.) in a testng.xml file or in build.xml.
- Run TestNG.

Here, one complete example of TestNG testing using POJO class, Business logic class and a test xml, which will be run by TestNG.

Create **EmployeeDetails.java** in **/work/testng/src**, which is a POJO class.

```
public class EmployeeDetails {

  private String name;
  private double monthlySalary;
  private int age;

  // @return the name

  public String getName() {
    return name;
  }

  // @param name the name to set

  public void setName(String name) {
    this.name = name;
  }

  // @return the monthlySalary

  public double getMonthlySalary() {
    return monthlySalary;
  }

  // @param monthlySalary the monthlySalary to set

  public void setMonthlySalary(double monthlySalary) {
    this.monthlySalary = monthlySalary;
  }

  // @return the age

  public int getAge() {
    return age;
  }

  // @param age the age to set

  public void setAge(int age) {
    this.age = age;
  }
}
```

EmployeeDetails class is used to −

- get/set the value of employee's name.
- get/set the value of employee's monthly salary.
- get/set the value of employee's age.

Create an **EmpBusinessLogic.java** in **/work/testng/src**, which contains business logic.

```
public class EmpBusinessLogic {

   // Calculate the yearly salary of employee
   public double calculateYearlySalary(EmployeeDetails employeeDetails) {
      double yearlySalary = 0;
      yearlySalary = employeeDetails.getMonthlySalary() * 12;
      return yearlySalary;
   }

   // Calculate the appraisal amount of employee
   public double calculateAppraisal(EmployeeDetails employeeDetails) {

      double appraisal = 0;

      if(employeeDetails.getMonthlySalary() < 10000) {
         appraisal = 500;

      } else {
         appraisal = 1000;
      }

      return appraisal;
   }
}
```

EmpBusinessLogic class is used for calculating −

- the yearly salary of employee.
- the appraisal amount of employee.

Now, let's create a TestNG class called **TestEmployeeDetails.java** in /work/testng/src. A TestNG class is a Java class that contains at least one TestNG annotation. This class contains test cases to be tested. A TestNG test can be configured by @BeforeXXX and @AfterXXX annotations (we will see this in the chapter TestNG - Execution Procedure), which allows to perform some Java logic before and after a certain point.

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

public class TestEmployeeDetails {
   EmpBusinessLogic empBusinessLogic = new EmpBusinessLogic();
   EmployeeDetails employee = new EmployeeDetails();
```

```
  @Test
  public void testCalculateAppriasal() {

    employee.setName("Rajeev");
    employee.setAge(25);
    employee.setMonthlySalary(8000);

    double appraisal = empBusinessLogic.calculateAppraisal(employee);
    Assert.assertEquals(500, appraisal, 0.0, "500");
  }

  // Test to check yearly salary
  @Test
  public void testCalculateYearlySalary() {

    employee.setName("Rajeev");
    employee.setAge(25);
    employee.setMonthlySalary(8000);

    double salary = empBusinessLogic.calculateYearlySalary(employee);
    Assert.assertEquals(96000, salary, 0.0, "8000");
  }
}
```

TestEmployeeDetails class is used for testing the methods of **EmpBusinessLogic** class. It does the following −

- Tests the yearly salary of the employee.
- Tests the appraisal amount of the employee.

Before you can run the tests, you must configure TestNG using a special XML file, conventionally named testng.xml. The syntax for this file is very simple, and its contents are as shown below. Create this file in **/work/testng/src**.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >

<suite name = "Suite1">
  <test name = "test1">
    <classes>
      <class name = "TestEmployeeDetails"/>
    </classes>
  </test>
</suite>
```

Details of the above file 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.
- Tag <test> represents one test and can contain one or more TestNG classes.
- <class> tag represents a TestNG class. It is a Java class that contains at least one TestNG annotation. It can contain one or more test methods.

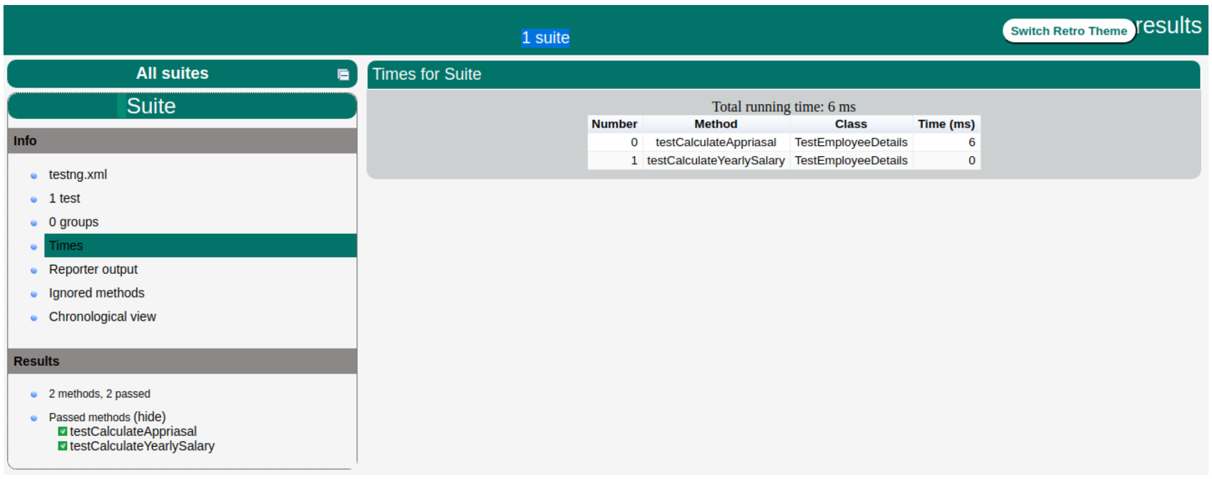
Compile the Test case classes using javac.

```
/work/testng/src$ javac EmployeeDetails.java EmpBusinessLogic.java
TestEmployeeDetails.java
```

Now TestNG with the following command −

```
/work/testng/src$ java org.testng.TestNG testng.xml
```

If all has been done correctly, you should see the results of your tests in the console. Furthermore, TestNG creates a very nice HTML report in a folder called **test-output** that is automatically created in the current directory. If you open it and load index.html, you will see a page similar to the one in the image below −



2 methods, 2 passed

Passed methods (hide)
- ✓ testCalculateAppriasal
- ✓ testCalculateYearlySalary

Total running time: 6 ms

Number	Method	Class	Time (ms)
0	testCalculateAppriasal	TestEmployeeDetails	6
1	testCalculateYearlySalary	TestEmployeeDetails	0

execution procedure of methods in TestNG. It explains the order of the methods called. Here is the execution procedure of the TestNG test API methods with an example.

Create a java class file name **TestngAnnotation.java** in in **/work/testng/src** to test annotations.

```
import org.testng.annotations.Test;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeSuite;
import org.testng.annotations.AfterSuite;

public class TestngAnnotation {
  // test case 1
  @Test
  public void testCase1() {
    System.out.println("in test case 1");
  }

  // test case 2
  @Test
  public void testCase2() {
    System.out.println("in test case 2");
  }

  @BeforeMethod
  public void beforeMethod() {
    System.out.println("in beforeMethod");
  }

  @AfterMethod
  public void afterMethod() {
    System.out.println("in afterMethod");
  }

  @BeforeClass
  public void beforeClass() {
    System.out.println("in beforeClass");
  }

  @AfterClass
  public void afterClass() {
    System.out.println("in afterClass");
  }

  @BeforeTest
  public void beforeTest() {
    System.out.println("in beforeTest");
  }

  @AfterTest
  public void afterTest() {
    System.out.println("in afterTest");
  }

  @BeforeSuite
```

```
   public void beforeSuite() {
      System.out.println("in beforeSuite");
   }

   @AfterSuite
   public void afterSuite() {
      System.out.println("in afterSuite");
   }

}
```

Next, let's create the file **testng.xml** in in **/work/testng/src** to execute annotations.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >

<suite name = "Suite1">
   <test name = "test1">
      <classes>
         <class name = "TestngAnnotation"/>
      </classes>
   </test>
</suite>
```

Compile the Test case class using javac.

```
/work/testng/src$ javac TestngAnnotation.java
```

Now, run the testng.xml, which will run the test case defined in the provided Test Case class.

```
/work/testng/src$ java org.testng.TestNG testng.xml
```

Verify the output.

```
in beforeSuite
in beforeTest
in beforeClass
in beforeMethod
in test case 1
in afterMethod
in beforeMethod
in test case 2
in afterMethod
in afterClass
in afterTest
in afterSuite
```

```
===============================================
Suite
Total tests run: 2, Failures: 0, Skips: 0
===============================================
```

Based on the above output, the execution procedure is as follows −

- First of all, beforeSuite() method is executed only once.
- Lastly, the afterSuite() method executes only once.
- Even the methods beforeTest(), beforeClass(), afterClass(), and afterTest() methods are executed only once.
- beforeMethod() method executes for each test case but before executing the test case.
- afterMethod() method executes for each test case but after executing the test case.
- In between beforeMethod() and afterMethod(), each test case executes.

TEST REPORTS:

TestNG, by default, generates multiple reports as part of its test execution. These reports mainly include TestNG HTML report, TestNG email-able report, TestNG report XML, and JUnit report XML files. These files can be found under the output report folder (in this case, test-output).

TestNG - CUSTOM REPORTER

1. Create Test Case Class. Create a java class, say, SampleTest. java in /work/testng/src. import org.

2. Create Custom Reporting Class. Create another new class named CustomReporter. java in /work/testng/src. import java.

3. Create testng. xml. Create testng.