1. **What is TestNG?**

* TestNG is an automated open source testing framework.
* NG-Next Generation
* TestNG is a Testing Framework designed to simplify, a broad range of testing needs from Unit Testing to System Testing.
* Initially developed for Unit Testing, now used for all kinds of Testing
* TestNG inspired by JUnit (Java Platform), and NUnit (.NET platform) but introduced some new functionalities.

1. **What are the main features of the TestNG?**

* TestNG Annotations are easy to create Test Cases
* Test cases can be grouped and prioritized more easily.
* Supports Parameterization.
* Execute multiple programs / classes using XML. v) Generate HTML Reports. vi) Parallel Test Execution.

1. **What are the advantages of TestNG over JUnit?**

* In **TestNG**, there is no method name constraint as in JUnit. We can give any name to the test methods in**TestNG**.
* In JUnit, we have to declare @BeforeClass and @AfterClass but in TestNG this is not necessary.
* In TestNG we can tell the test that one method is dependent on another method but in JUnit it is not possible. In JUnit each test is independent of each other.
* We can group the test cases in TestNG which is not possible in JUnit.
* TestNG support various kind of annotations like @BeforeSuite, @AfterSuite, @BeforeTest, @AfterTest, @BeforeGroup, @AfterGroup etc

1. **What is TestNG in Selenium?**

* Using TestNG framework in Selenium we can,  
  > Generate Detailed (HTML) Test Reports.  
  > Group Test cases.  
  > Parallel Test execution.  
  > Parameterize Tests  
  > Execute multiple classes / programs using XML file etc.

1. **What are the important TestNG Annotations?**

@Test - Annotation for every Test (Method)

@BeforeMethod - pre-condition for every test case in a program.

@AfterMethod - Post condition for every test case in a program.

@BeforeClass - pre condition for all test cases in a program/class

@AfterClass - post condition for all test cases in a program/class

@BeforeTest - pre condition for all test cases in multiple classes/programs

@AfterTest - post condition for all test cases in multiple classes / programs

@BeforeGroups: The list of groups that this configuration method will run before.  
@AfterGroups: The list of groups that this configuration method will run after.

**Example Code :**  
import org.testng.annotations.AfterClass;  
import org.testng.annotations.AfterMethod;  
import org.testng.annotations.AfterSuite;  
import org.testng.annotations.AfterTest;  
import org.testng.annotations.BeforeClass;  
import org.testng.annotations.BeforeMethod;  
import org.testng.annotations.BeforeSuite;  
import org.testng.annotations.BeforeTest;  
import org.testng.annotations.Test;  
  
public class NewTest {  
  
 @Test  
 public void testCase1() {  
 System.out.println("This is the Test Case 1");  
 }  
  
 @Test  
 public void testCase2() {  
 System.out.println("This is the Test Case 2");  
 }  
  
 @BeforeMethod  
 public void beforeMethod() {  
 System.out.println("This will execute before every Method");  
 }  
  
 @AfterMethod  
 public void afterMethod() {  
 System.out.println("This will execute after every Method");  
 }  
  
 @BeforeClass  
 public void beforeClass() {  
 System.out.println("This will execute before the Class");  
 }  
  
 @AfterClass  
 public void afterClass() {  
 System.out.println("This will execute after the Class");  
 }  
  
 @BeforeTest  
 public void beforeTest() {  
 System.out.println("This will execute before the Test");  
 }  
  
 @AfterTest  
 public void afterTest() {  
 System.out.println("This will execute after the Test");  
 }  
  
 @BeforeSuite  
 public void beforeSuite() {  
 System.out.println("This will execute before the Test Suite");   
 }  
  
 @AfterSuite  
 public void afterSuite() {  
 System.out.println("This will execute after the Test Suite");  
 }

1. **Give an example for parallel Test execution?**

**XML File**  
  
<suite name = "Parallel Test suite" parallel = "classes" thread-count = "2">  
<test name = "Sanity Test">  
  
<classes>  
<class name = "seleniumTests.Class1"/>  
<class name = "seleniumTests.Class2"/>  
</classes>  
  
</test>  
</suite>  
------------------------  
parallel = "methods": TestNG will run all the methods in separate threads.  
parallel = "classes" : TestNG will run all the methods in the same class in the same thread.  
parallel = "tests" : TestNG will run all the methods in the same <test> tag in the same thread.

**Class 1**  
  
public class Class1 {  
    @BeforeTest  
    public void login(){  
    System.out.println("Login Successful");  
    }  
    @AfterTest  
    public void logout(){  
    System.out.println("Logout Successful");  
    }  
    @Test (priority = 1)  
    public void search(){  
    System.out.println("Search Successful");  
    }  
    @Test (priority = 2)  
    public void advancedSearch(){  
    System.out.println("Advanced Search Successful");  
    }  
    @Test (priority = 3)  
    public void buyProducts(){  
    System.out.println("Buying Products Successful");  
    }  
    @Test (priority = 4)  
    public void testCase(){  
    System.out.println("Test Case in "+getClass().getSimpleName()  
    + " With Thread id: " + Thread.currentThread().getId());  
    }  
    }  
-----------------------------------------  
**Class 2**  
  
public class Class2 {  
    /\*@BeforeClass  
    public void login(){  
    System.out.println("Login Successful");  
    }  
    @AfterClass  
    public void logout(){  
    System.out.println("Logout Successful");  
    }\*/  
    @Test (priority = 1)  
    public void accountSummary(){  
    System.out.println("Account Summary Successful");  
    }  
    @Test (priority = 2)  
    public void fundTransfer(){  
    System.out.println("Fund Transfer Successful");  
    }  
    @Test (priority = 3)  
    public void billPayment(){  
    System.out.println("Bill Payment Successful");  
    }  
    @Test (priority = 4)  
    public void testCase(){  
    System.out.println("Test Case in "+getClass().getSimpleName()  
    + " With Thread id: " + Thread.currentThread().getId());  
    }  
}

1. **What is grouping Test cases Give an Example for?**

* It is a new feature included in TestNG, it allows you to dispatch methods into **proper portions and perform grouping of test methods**.
* With group test, you can not only declare methods that belong to groups, but you can also specify groups that contain other groups.
* Groups are determined in your testing.xml file using the group test.

**XML File**  
<suite name ="suite">  
<test name ="test">  
  
<groups>  
<run>  
<include name = "sanity"/>  
</run>  
</groups>  
  
<classes>  
<class name = "seleniumTests.GroupTests"/>  
<class name = "seleniumTests.GroupTests2"/>  
</classes>  
  
</test>  
</suite>  
-------------------------------  
**Java Program/Class file**  
  
@BeforeTest (groups ={"sanity","regression"})  
    public void login(){  
    System.out.println("Login Successful");  
    }  
    @AfterTest (groups ={"sanity","regression"})  
    public void logout(){  
    System.out.println("Logout Successful");  
    }  
    @Test (groups ={"sanity"})  
    public void search(){  
    System.out.println("Search Successful");  
    }  
    @Test (groups ={"sanity","regression"})  
    public void advancedSearch(){  
    System.out.println("Advanced Search Successful");  
    }  
    @Test (groups ={"sanity", "regression"})  
    public void buyProducts(){  
    System.out.println("Buying Products Successful");  
    }  
    @Test (groups ={"regression"})  
    public void abcd(){  
    System.out.println("Abcd Successful");  
    }  
    @Test (groups ={"regression"})   
    public void xyza(){  
    System.out.println("Xyza Successful");  
    }  
    @Test (groups ={"regression"})  
    public void asdf(){  
    System.out.println("Asdf Successful");  
    }  
}

1. Give an example for Data Driven Testing?  
   If you want work with excel, then download third party jar files  
     
   **Ex: jxl**  
     
   public class DataDriven {  
       @Test (dataProvider="testdata")  
       public void Addition(String val1, String val2, String val3){  
           int a = Integer.parseInt(val1);  
           int b = Integer.parseInt(val2);  
           int c = Integer.parseInt(val3);  
           int result = a + b + c;  
           System.out.println(result);  
       }  
         
   @DataProvider(name="testdata")  
       public Object [] [] readExcel() throws BiffException, IOException {  
           File f = new File("C:/Users/gcreddy/Desktop/Input.xls");  
           Workbook w = Workbook.getWorkbook(f);  
           Sheet s = w.getSheet(0);  
           int rows = s.getRows();  
           int columns = s.getColumns();  
           //System.out.println(rows);  
           //System.out.println(columns);  
           String Inputdata [] [] = new String [rows] [columns];  
           for (int i = 0; i<rows; i++){  
               for (int j = 0; j<columns; j++){  
                   Cell c = s.getCell(j, i);  
                   Inputdata [i] [j] = c.getContents();  
                   //System.out.println(Inputdata[i][j]);  
               }  
           }  
           return Inputdata;  
           }  
   }

**9. How to create XML file?**  
 Create XML file  
 Select Java Project and Right click  
  
> New   
> Other  
> Enter XML and Select XML file  
> Enter File Name  
> Finish

**10. How to install TestNG in Eclipse?**  
 In Eclipse IDE,  
 Help menu -> Install New Software -> Click Add  
> Enter name as "TestNG"  
> Enter url as: "http://beust.com/eclipse/"  
> Select "TestNG"  
> Next > Next > accept the Agreement > finish

1. **What is TestNG in Selenium?**  
     
   Using TestNG framework in Selenium we can,  
     
   > Generate Detailed (HTML) Test Reports.  
     
   > Group Test cases.  
     
   > Parallel Test execution.  
     
   > Parameterize Tests  
     
   > Execute multiple classes / programs using XML file etc...
2. **List out various ways in which TestNG can be invoked?**

TestNG can be invoked in different ways like

* Using Eclipse
* With ant
* From the command line

1. **Explain what is testing.xml file used for?**

* File testing.xml captures your entire testing in XML.
* This file makes it easy to define all your test suites and their parameters in one file, which you can verify in your code repository or e-mail to coworkers.
* It also makes easy to pull out subsets of your tests or split several runtime configurations

1. **In TestNG how can you disable a test?**

* To disable the test case you don’t want, you can use annotations

Ex: **@Test(enabled = false).**

1. **Explain what is Time-Out test in TestNG?**

* The Time-Out test in TestNG is nothing but time allotted to perform unit testing.
* If the unit test fails to finish in that specific time limit, TestNG will abandon further testing and mark it as a failed.

**Ex: @Test(threadPoolSize = 3, invocationCount = 10,  timeOut = 10000)**

**public void testCase1(){**

* In this example, the function testCase1 will be invoked ten times from three different threads. Additionally, a time-out of ten seconds guarantees that none of the threads will block on this thread forever.

1. **Mention what does the “suite test” does in TestNG?**

* “Suite Test” is done when you have to run few unit test together, “ Suite Test” bundle this unit test together.
* XML file is used to run the suite test.

1. **Explain what is parametric testing?**

* Parameterized testing allows developers to execute the same test over and over again using different values.  In two different ways TestNG allows you to pass parameters directly to your test methods.
* With testing.xml
* With Data Providers

1. **Explain what does @Test(invocationCount=?) and (threadPoolSize=?) indicates?**

* **@Test (threadPoolSize=?):** The threadPoolSize attributes tell TestNG to form a thread pool to run the test method through multiple threads. With threadpool, the running time of the test method reduces greatly.

**Note:** This attribute is ignored if invocationCount is not specified

Ex: @Test(threadPoolSize = 3, <code class="plain">invocationCount = </code><code class="value">10</code>) public void testCase1(){

In this example, the method *testCase1* will be invoked from three different threads

* **@Test(invocationCount=?):** The invocationcount tells how many times TestNG should run this test method.

**Ex: @Test(invocationCount = 10)**

**public void testCase1(){**

In this example, the method *testCase1* will be invoked ten times

1. **Mention different ways in which you can produce reports for TestNG results?**

There are two ways to produce a report with Test NG, they are

* **Listeners:** For a listener class to implement, the class has to implement the org.testng./TestListener interface. These classes are informed at runtime by TestNG when the test begins, finsishes, skips, passes or fails.
* **Reporters:** For a reporting class to implement, the class has to implement an org.testng/Reporter interface. When the whole suite run ends, these classes are called. When called, the object consisting the information of the whole test run is delivered to this class.

1. Can you arrange the below <testng.xml> tags from parent to child?

**<test>  
<suite>  
<class>  
<methods>  
<classes>**

**Answer:** The <testng.xml> file will have the following structure.

* The parent tag in the <testng.xml> file is the <suite> tag.
* <suite> tag can include one or more <test> tags.
* <test> tag can include the <classes> tag.
* <classes> tag can include one or more <class> tags.
* <class> tag wraps the <methods> tag where we define the test methods to include or exclude.

Hence, the correct order of the TestNG tags would be.

**<suite>  
<test>  
<classes>  
<class>  
<methods>**

1. **How to define the priority of @Test method? Also, mention its usage?**

* **@Test(priority=0)**

Using priority, you can manage @Test method execution sequence as per your requirement. That means @Test method with priority = 0 will run 1st and @Test method with priority = 1 will execute 2nd and so on.

1. **How to apply regular expression in <testng.xml> file to find @Test methods containing “product” keyword?**

<methods>

<include name=".\*product.\*"/>

</methods>

1. **Benefits of TestNG**

There are number of benefits but from Selenium perspective, major advantages of TestNG are :

* It gives the ability to produce ***HTML Reports*** of execution
* ***Annotations***made testers life easy
* Test cases can be ***Grouped & Prioritized*** more easily
* ***Parallel***testing is possible
* Generates ***Logs***
* Data ***Parameterization***is possible

1. **Create dependency**

A test can dependent on a single method or more than a single method

@Test(dependsOnMethods={"testTwo"})

1. **AlwaysRun**

Eventhough the test case fail then it will force to Run as

@Test(alwaysRun=true)

1. **How to exclude a particular test method from a test case execution?**

By adding the exclude tag in the testng.xml

**<classes>**

**<class name="TestCaseName">**

**<methods>**

**<exclude name="TestMethodNameToExclude"/>**

**</methods>**

**</class>**

**</classes>**

1. **How to exclude a particular test group from a test case execution?**

By adding the exclude tag in the testng.xml

**<groups>**

**<run>**

**<exclude name="TestGroupNameToExclude"/>**

**</run>**

**</groups>**