```
Q. What is mean by Assertion?
Ans: Asserstion is a way to verify that expected result and actual result
     matches or not.
```

```
Q. What is mean by hard assert?
Ans : When assert condition fails then AssertionError is thrown imdiately
      is nothing but hard assert.

      In hard assert rest of code won't be executed and move to next test case.

      Assert class talks about hard assert

      In TestNg hard assert is bydefault available
```

Example

```
@Test
public void m1Test()
{
        System.out.println("before assertEquals");
        int a=10;
        int b=20;
        Assert.assertEquals(a, b);
        System.out.println("after assertEquals");
        assertTrue(a==b);
        assertNotEquals(a, b);
}
```

```
---------------------------------------
Q. What is mean by soft assert?
Ans: When assert condition fails then AssertionError is not thrown immediately
      is nothing but soft assert.

Q. How to implement soft assert?

Ans: To implement soft asser we need to use SoftAssert class.
     SoftAssert class is similar to Assert class only difference is
     All methods of SoftAssert are non-static.

Note: If there is any exception and you want to throw that exception then
      we need to use assertAll() method as last statement in test method.

=============================================
```

```
Example
        @Test
        public void m1Test()
        {
                SoftAssert soft=new SoftAssert();
                System.out.println("before assertEquals");
                int a=10;
                int b=20;
                soft.assertEquals(a, b);
                System.out.println("after assertEquals");
                soft.assertTrue(a==b);
                soft.assertNotEquals(a, b);
                soft.assertAll();
        }
```

```
Exception in TestNg

-> If any method throws an exception and if we want to test that
   exception is getting or not.

-> We need to use expectedException attribute in @Test Annotation

Example 1

   @Test(expectedExceptions ={ArithmeticException.class})
   public void m2Test()
   {
                int a=10;
                int b=0;
                int c=a/b;

   }
```

```
Example 2

@Test(expectedExceptions ={ArithmeticException.class})
public void m2Test()
        {
                int a=10;
                int b=1;
                int c=a/b;
                throw new NullPointerException();

        }

----------------------------------------
```

Q. Tell some attribute of @Test annotation

Ans : testName,expectedExceptions, dataProvider,suiteName

TestNg Listeners

-> Listeners are interfaces in TestNg

-> These Listeners are used to record event generated while executing test case and by using that event we can modify the behaviour of test case.

-> TestNg provides below Listeners

1) ITestListener
2) IExecutionListner
3) IAnnotationTransformer
4) IAnnotationTransformer2
5) ISuiteListener
6) IConfigurationListener
7) IMethodInterpreter
8) IInvokedMethodListener
9) IHookable
10) IReporter

==========

==========

1) ITestListener

1) This Listener record the event for test running

2) ITestListener interface present in org.testng package

3) Parent interface of ITestListner is ITestNGListener

4) ITestNGListener is marker interface

Marker Interface - The interface which does not contains any methods but by implementing that interface our object get some ability, such types of interface are called as marker interface.

Note: In java 8 some features are added related to interface and those features are as below
1) inside interface we can take private methods
2) inside interface we can take static methods
3) inside interface we can take default methods

According to Java 8 features TestNG also added same features related to interface from 7.x version

So If we are implementing an interface then we no need to override all methods of interface in the implementation class

5) Methods of ITestListener interface.

default void onTestStart(ITestResult result)

This method get called each time before a test is invoked

To configure listener class we have two ways

a) Annotation base approach

Use @Listener annotation on the top of Test class and give fully qualified name of Listener class

Example

```
@Listeners(com.testng.demo.ITestListenerDemo.class)
Run All
public class Test101 {

    @Test(testName ="m1Test" )
    Run | Debug
    public void m1Test()
    {
        System.out.println(" i am inside m1Test");
    }

}
```

```
public class ITestListenerDemo implements ITestListener
{
    @Override
    public void onTestStart(ITestResult result) {
        System.out.println("I am inside onTestStart "+result.getEndMillis());
    }

}
```

b) tesng.xml file approach

To configure Listners in TestNG xml file we need to use

```
<listeners>
 <listener class-name="com.testng.demo.ITestListenerDemo"/>
</listeners>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="DemoSuite">
<listeners>
    <listener class-name="com.testng.demo.ITestListenerDemo"/>
</listeners>
   <test name="Test1">
      <classes>
      <class name="com.testng.demo.Test101">
      <methods>
      <include name="m1Test"/>
      </methods>
      </class>
      </classes>
   </test>
</suite>
```