SWT301

Unit test
with
TestNG framework

Team 01
Member Vu Thien An
Nguyen Phat Dat
Nguyen Vi Khang
Nguyen Dang Loc
Nguyen Hong Minh

Agenda

01 **Basic concepts**

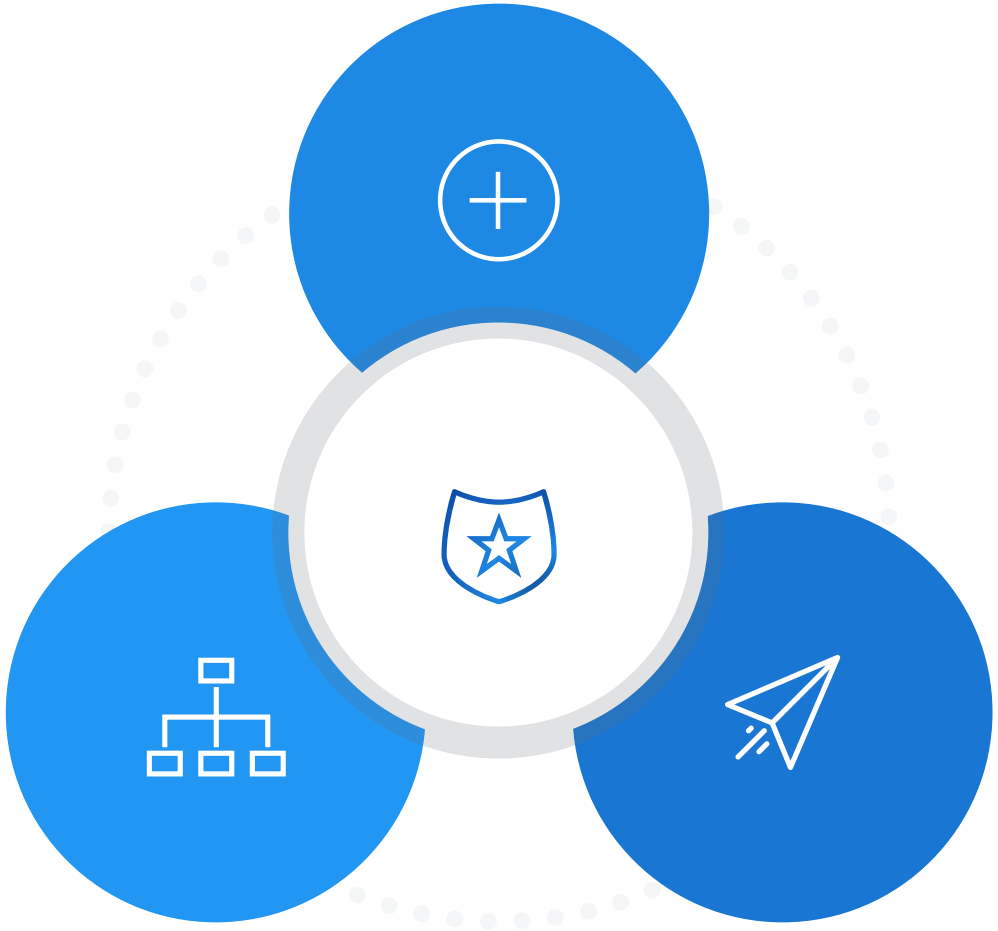
Definitions of unit test & some related concepts

02 **TestNG framework**

What is TestNG, how to use it?

03 **Practical demo**

Design, implement unit test with TestNG framework



Definition

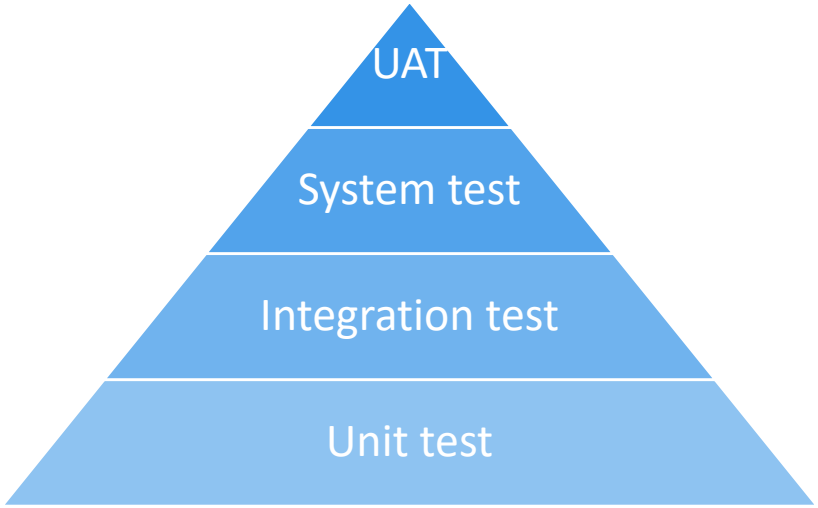
Unit testing

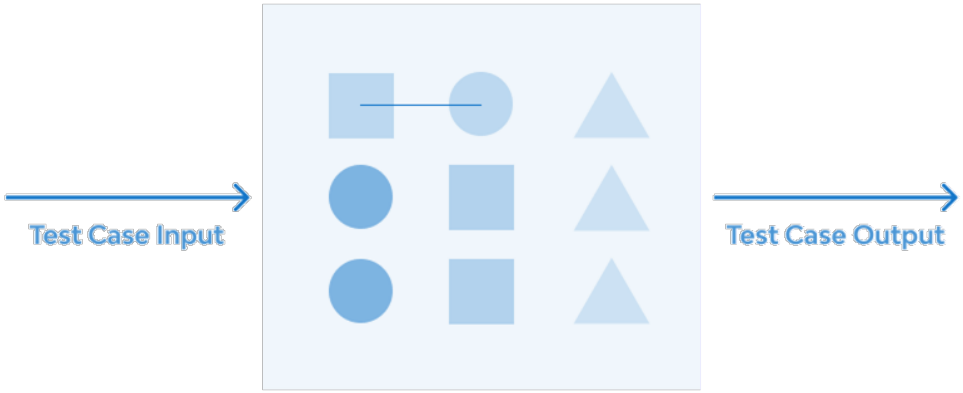
individual units or components of a software are tested

is done before integration testing

is white box testing done during the development (coding phase) of an application by the developers

by developer





Unit ← Function | Method | Procedure | Module | Object | .....

Other concepts

Assertion

a statement asserts the expected behavior of the test, fails if the result is different than what expected

Test case

specification of the inputs, execution conditions, testing procedure, and expected results

Testsuite

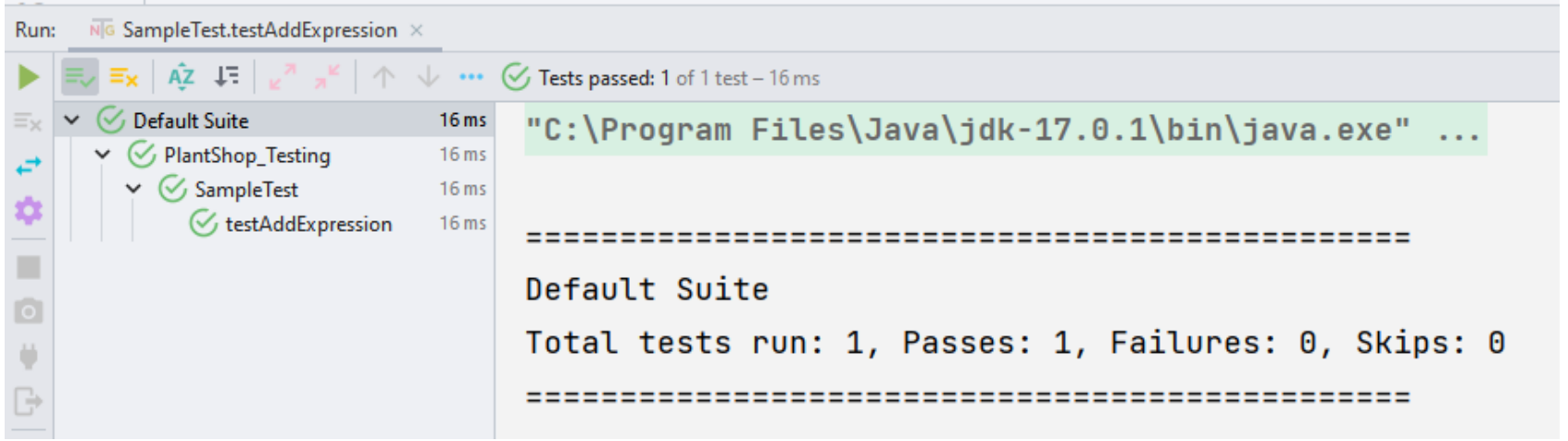
a collection of test cases

Basic example

```java
public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }
}
```

```java
import DTO.Calculator;
import org.testng.Assert;
import org.testng.annotations.Test;

public class SampleTest {
    @Test
    public void testAddExpression() {
        Calculator cal = new Calculator();
        Assert.assertEquals(cal.add( a: 2,  b: 2),  expected: 4);
    }
}
```



Test coverage

Statement coverage

ⓘ The percentage of statements that have been tested

$$\text{Statement coverage} = \frac{\text{Number of excecuted statements}}{\text{Total number of statements}} \times 100$$

Test case 1: Input: a = 3, b = 9

Coverage 71%

```
2   void Prints(int a, int b) {
3       int res = a + b;
4       if (res > 0)
5           Print("Positive", res);
6       else
7           Print("Negative", res);
8   }
```

Test case 2: Input: a = -3, b = -9

Coverage 86%

```
2   void Prints(int a, int b) {
3       int res = a + b;
4       if (res > 0)
5           Print("Positive", res);
6       else
7           Print("Negative", res);
8   }
```

→ **Test suite**

Coverage 100%

Test coverage

Path coverage

ⓘ The percentage of paths that have been tested.

$$\text{Path coverage} = \frac{\text{Number of executed branches}}{\text{Total number of branches}} \times 100$$

Test case 1: Input: a = 2

Coverage 50%

```
2    void sample(int a) {
3        if (a > 5)
4            a *= 3;
5        Print(a);
6    }
```

Test case 2: Input: a = 6

Coverage 50%

```
2    void sample(int a) {
3        if (a > 5)
4            a *= 3;
5        Print(a);
6    }
```

→ **Test suite**

Coverage 100%

Design unit test

1. **ARRANGE** Initialize objects, resources, the environment, and other conditions.

2. **ACT** Call the method/function with inputs, get actual results

3. **ASSERT** Compare expected values and actual values received

   PASSED / FAILED

→ Clean up resources, get the test report

Good unit test

Automated & repeatable

Easy to read & understand

Run quickly

Have full control of the unit under test

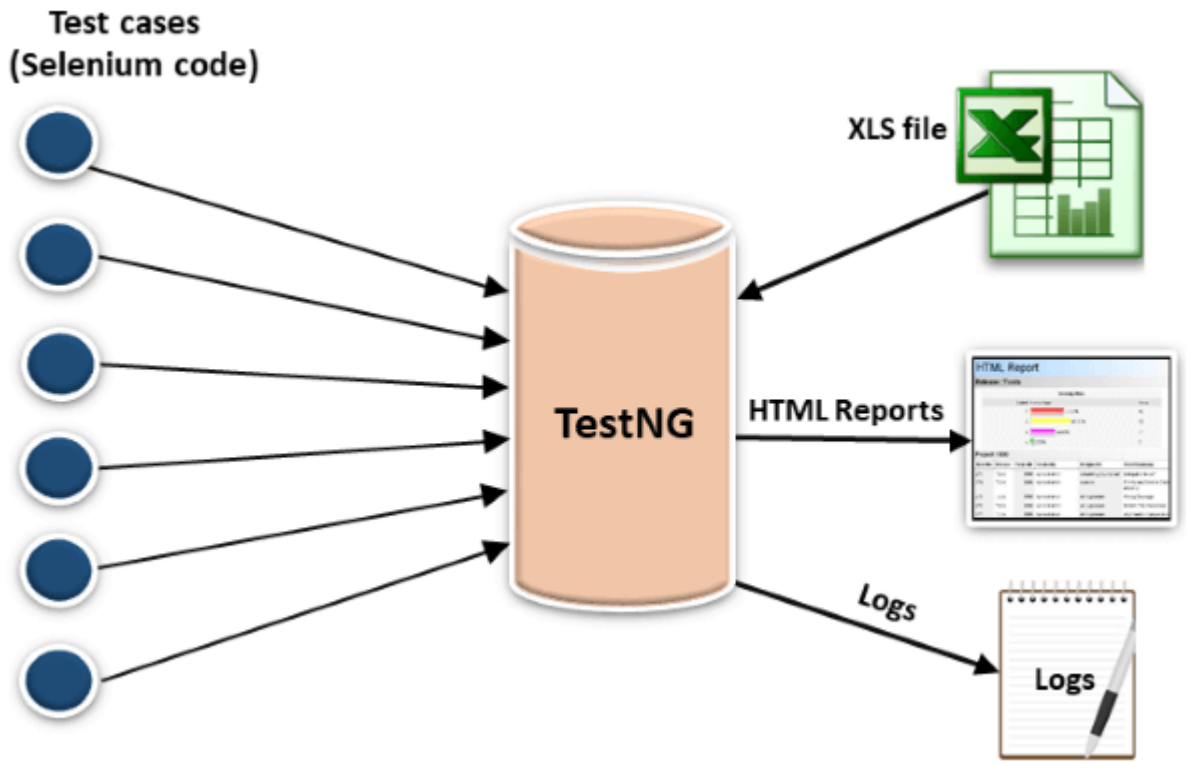
Be fully isolated

TestNG

An open-source test automation framework for Java

Developed on the same lines of JUnit and NUnit

NG in TestNG stands for ‘Next Generation’



Comparative advantages

Not require declare *@BeforeClass* and *@AfterClass*

Group and prioritize the tests easier

Allow dependency on multiple methods

Produces the HTML reports

Parametrization is more convenient and easier by *@DataProvider*

BASIS	JUNIT	TESTNG
Annotation support	✓	✓
Suite test	✓	✓
Ignore test	✓	✓
Exception test	✓	✓
Timeout	✓	✓
Parameterized test	✓	✓
Dependency test	✗	✓

Sources: *https://www.guru99.com/junit-vs-testng.html*

Demo TestNG

Q & A

Thank you!

Goodbye!