JUNIT
JAVA

Unit Testing Framework - [TestNG
JAVA

https://testng.org/doc/download.html

Unit Testing

TestNG →

Next Generation

Features of TestNG · → Unit Testing

① → Annotations → Small piece of code, decides the execution Methods
↓
sequence of

↓
starts with → '@'

Execution of methods happens alphabetically order

(2) ✓ Generate the Test Results Report

(3) Parallel Execution on a Single M/C across diff browser

(4) ✓ Prioritisation → login ①, logout ④, create user ②, delete user ③

(5) Grouping → ✓ Smoke, functional, ✓ Regression, Sanity

group 1

(6) Assertions → for validations → if/else X

(7) dependsOn Methods – login(), Logout()

(8) Ignore the TestCase – enabled = false

Note:- In TestNG There is no need of psv main() method.

Annotations

@BeforeSuite

@BeforeTest →

@BeforeClass → (1 Time) N.M

x @BeforeMethod ✓ → N.M. (1 Time)

Method ← [@Test → for TestCase — Mandatory

@AfterMethod → N.M (1 Time)

@AfterClass → N.M

@AfterTest →

@AfterSuite →

[Test Suite
↓ collection of TCs]

Before Suite

Before Test

Before Class

Before Method

TC1 —

After Method

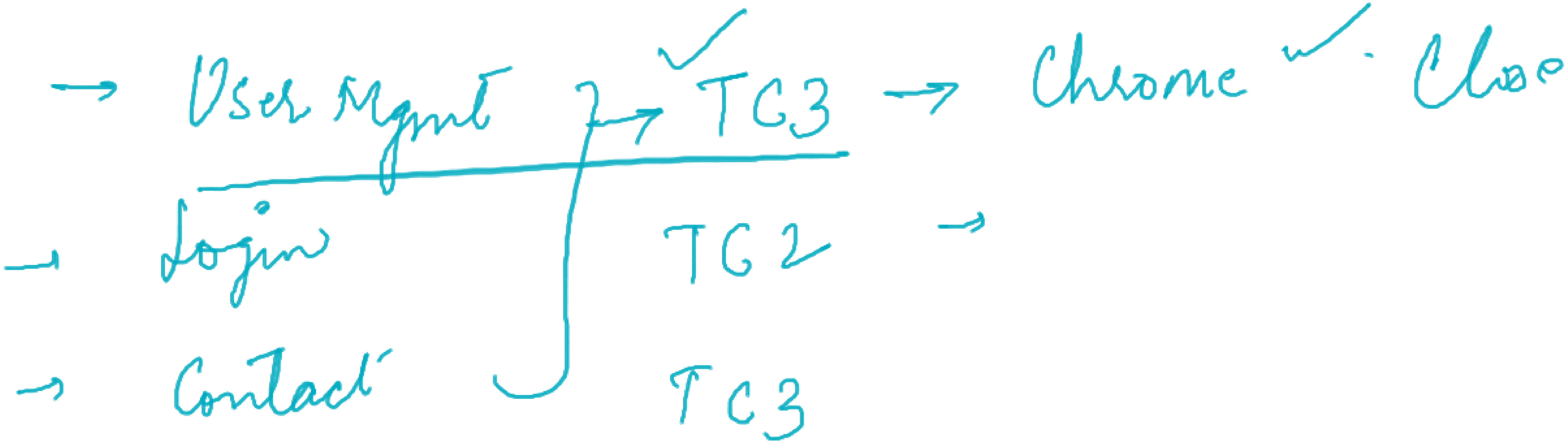
Before Method

TC2 —

After Method

After Test After Class

After Suite


→ User Mgmt → ✓ TC3 → Chrome ✓ Close
→ Login
TC2 →
→ Contact
TC3

@BeforeSuite - this method gets Executed at project level, before the BeforeTest annotation or other annotations

@BeforeTest - this method gets Executed at before the beforeclass annotation

@BeforeClass - this method gets Executed at class level, before the execution of @test or any other annotations

@BeforeMethod - this method gets Executed before the each @Test method

@Test - aTestCase - TestCase1

@AfterMethod - this method gets Executed after the each @Test method

@BeforeMethod - this method gets Executed before the each @Test method

@Test - bTestCase - TestCase2

@AfterMethod - this method gets Executed after the each @Test method

@AfterClass - this method gets Executed at class level, after the execution of @test or any other annotations

@AfterTest - this method gets Executed after the afterclass or other annotation

@AfterSuite - this method gets Executed at project level, after the BeforeTest annotation or other annotations

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="All Test Suite">
  <test verbose="2" preserve-order="true" name="F:/Upgradation_Selenium/
LearnSeleniumTool">
    <classes>
      <class name="testNGDemos.TestNGAnnotations">
        <methods>
          <include name="aTestCase"/>
          <include name="bTestCase"/>
          <include name="afterMethodAnnot"/>
          <include name="afterClassAnnot"/>
        </methods>
      </class>
      <class name="testNGDemos.TestNGAnnotations_realEx">
        <methods>
          <include name="browserLaunch"/>
          <include name="launchApp"/>
          <include name="loginApp"/>
        </methods>
      </class>
    </classes>
  </test>
</suite>
```

①

(2)

(3)

Cl1

Cl2

methods

Before Suite

Test

Class

Method

— @Test

After Method

Class

Test

Suite

3/9

Prioritisation →

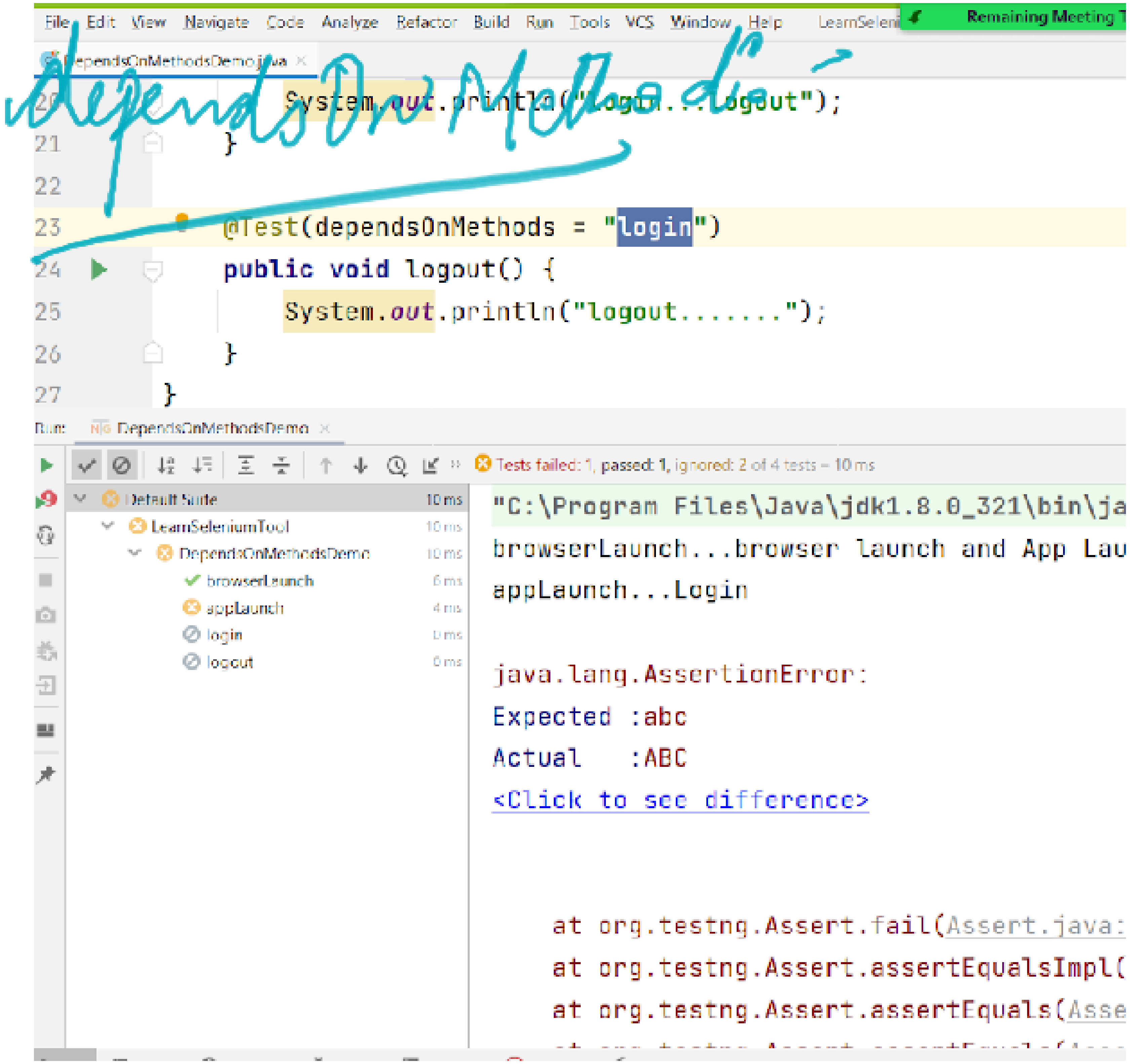
→ By default → alphabetical order

→ ~~order~~ Priority (High priority = 0)

or

If Not written

→ if two methods (@Test) having same Annotation will execute alphabetical order



dependsOnMethods

```java
        System.out.println("Login...logout");
    }

    @Test(dependsOnMethods = "login")
    public void logout() {
        System.out.println("logout.......");
    }
}
```

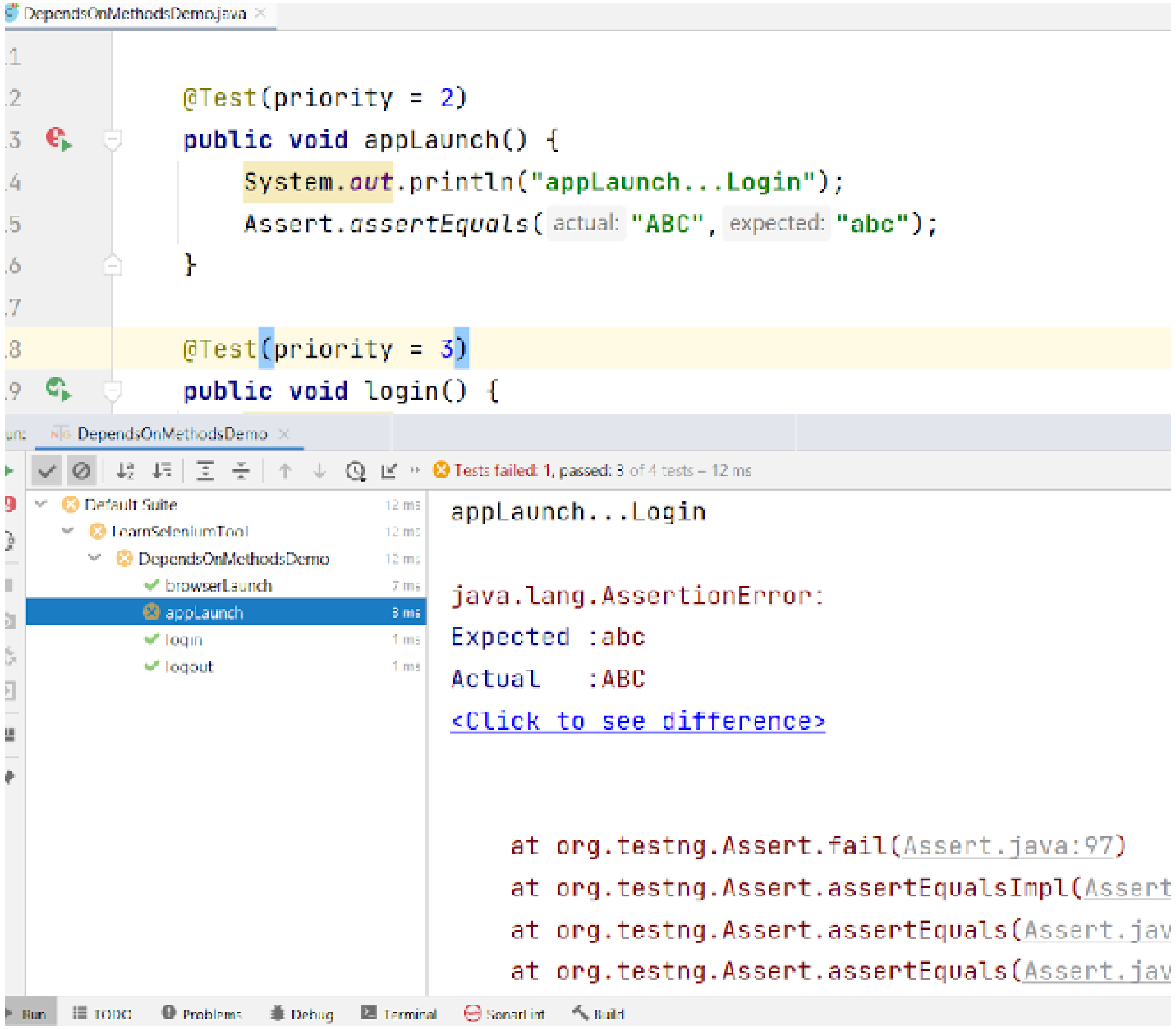
Tests failed: 1, passed: 1, ignored: 2 of 4 tests – 10 ms

```
Default Suite                        10 ms
  LearnSeleniumTool                  10 ms
    DependsOnMethodsDemo             10 ms
      ✓ browserLaunch                 6 ms
      ☒ appLaunch                     4 ms
      ⊘ login                         0 ms
      ⊘ logout                        0 ms
```

```
"C:\Program Files\Java\jdk1.8.0_321\bin\ja
browserLaunch...browser launch and App Lau
appLaunch...Login

java.lang.AssertionError:
Expected :abc
Actual   :ABC
<Click to see difference>


    at org.testng.Assert.fail(Assert.java:
    at org.testng.Assert.assertEqualsImpl(
    at org.testng.Assert.assertEquals(Asse
```



```java
    @Test(priority = 2)
    public void appLaunch() {
        System.out.println("appLaunch...Login");
        Assert.assertEquals( actual: "ABC", expected: "abc");
    }

    @Test(priority = 3)
    public void login() {
```

Tests failed: 1, passed: 3 of 4 tests – 12 ms

```
Default Suite                        12 ms
  LearnSeleniumTool                  12 ms
    DependsOnMethodsDemo             12 ms
      ✓ browserLaunch                 7 ms
      ☒ appLaunch                     3 ms
      ✓ login                         1 ms
      ✓ logout                        1 ms
```

```
appLaunch...Login

java.lang.AssertionError:
Expected :abc
Actual   :ABC
<Click to see difference>


    at org.testng.Assert.fail(Assert.java:97)
    at org.testng.Assert.assertEqualsImpl(Assert
    at org.testng.Assert.assertEquals(Assert.jav
    at org.testng.Assert.assertEquals(Assert.jav
```

Team - 5 Members

Depends On Method

M1 - ✓

M2 - → X

M3 -

M4 -

M5 -

skip

Skip.

Assertion → used for Validation purpose

can't use if else condition

- → Hard Assertion → TC1 → 10 lines → At @5 ↓ error ↓ 5 lines X

  ↓ it will halt/stop the execution if any error encountered

- → Soft Assertion

Grouping → Group of TCs →

Functional → QA →

- Smoke
- Sanity
- Functional | GUI
- Retesting

1000 TCs
↓
100 TCs

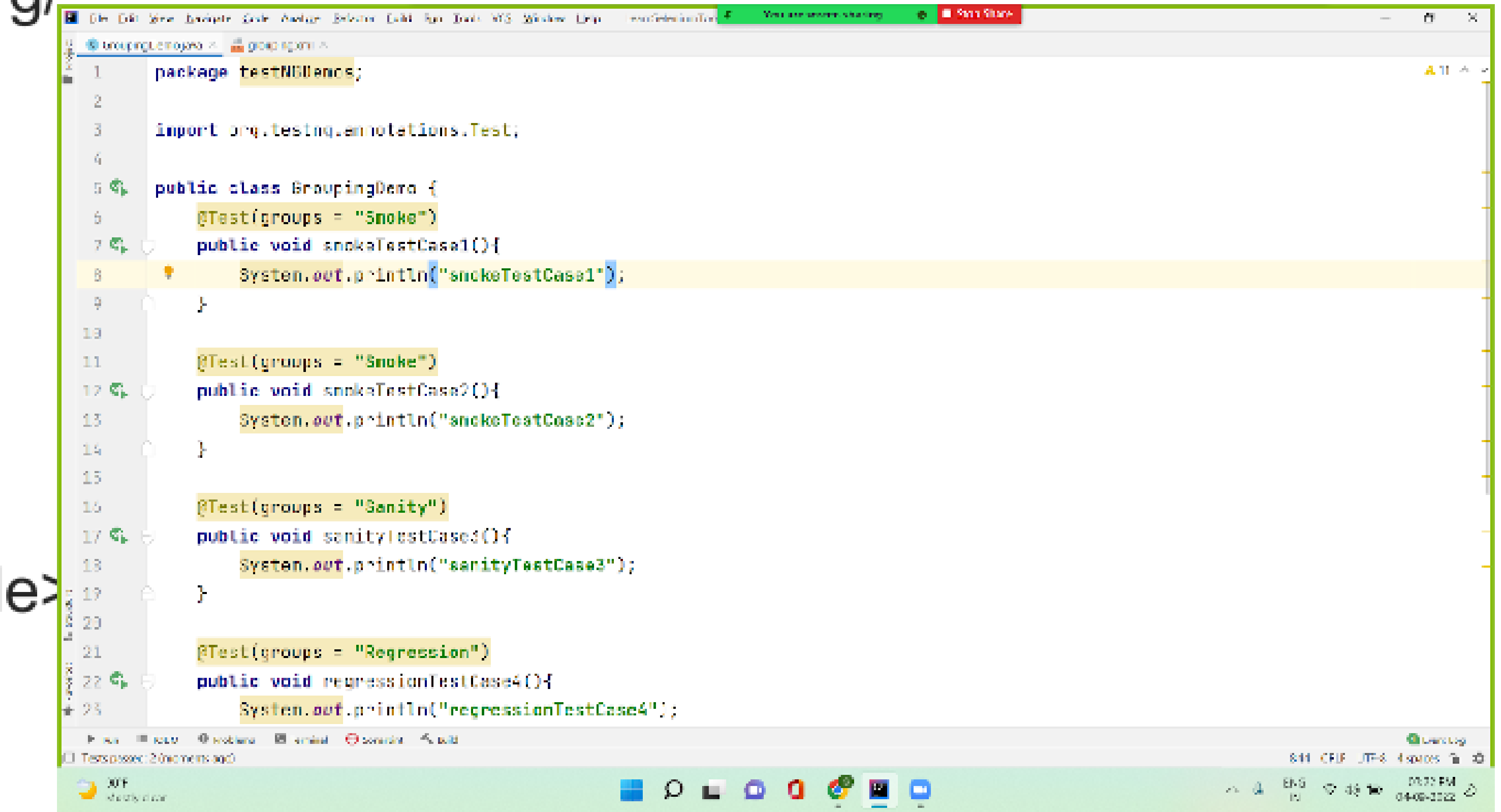
QA Env.

Prod Env. → Smoke | Sanity

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

        <groups>
            <run>
                <include name="Smoke"></include>
            </run>
        </groups>

        <classes>
            <class name="testNGDemos.GroupingDemo">
            </class>
        </classes>
    </test>
</suite>
```



```
package testNGDemos;

import org.testng.annotations.Test;

public class GroupingDemo {
    @Test(groups = "Smoke")
    public void smokeTestCase1(){
        System.out.println("smokeTestCase1");
    }

    @Test(groups = "Smoke")
    public void smokeTestCase2(){
        System.out.println("smokeTestCase2");
    }

    @Test(groups = "Sanity")
    public void sanityTestCase3(){
        System.out.println("sanityTestCase3");
    }

    @Test(groups = "Regression")
    public void regressionTestCase4(){
        System.out.println("regressionTestCase4");
```

Groups - 10

↳

9 run

1 don't run

```
<groups>
      <define name="all">
        <include name="Smoke"></include>
        <include name="Sanity"></include>
        <include name="Regression"></include>
      </define>

      <run>
        <include name="all"></include>
      </run>

    </groups>
```

"1", "G2"

groups={"Smoke","Regression"} → OR → either one of groups

groups= {"Smoke,Regression"} → AND

both

Login

Logout

AddFriend

comment

post

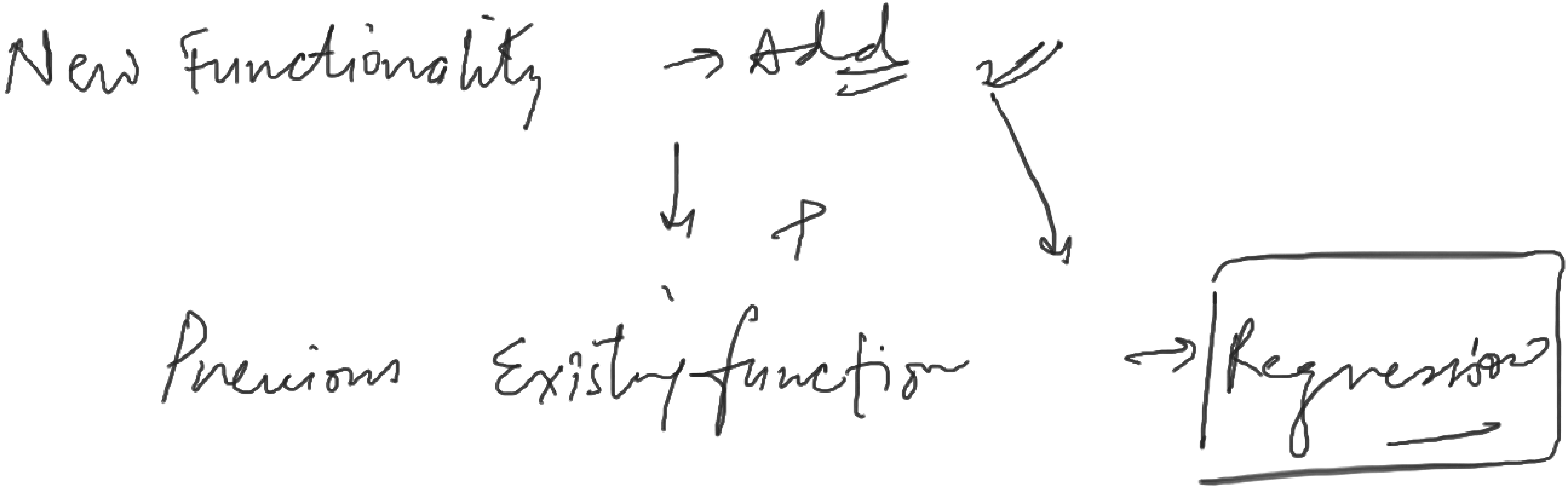
chat

Smoke TestCases

Group - Smoke

Prod Env.

Smoke



New Functionality → Add

Previous Existing function → Regression