

TestNG

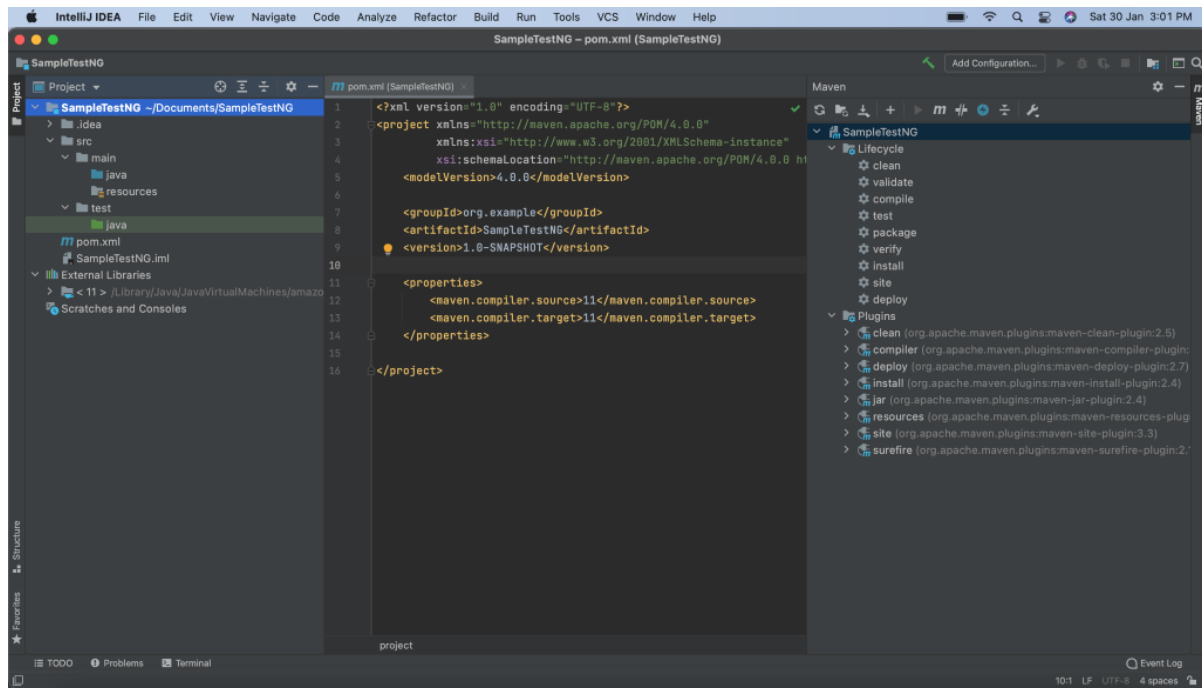
Introduction to TestNG - TestNG is a testing framework designed to simplify a broad range of testing needs, from unit testing (testing a class in isolation of the others) to integration testing (testing entire systems made of several classes, several packages and even several external frameworks, such as application servers).

Advantages of TestNG –

- 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.

Creating Maven Project with TestNG dependencies in IntelliJ IDEA-

Create a maven project



Open <https://mvnrepository.com/artifact/org.testng/testng/7.3.0>

The screenshot shows the Maven Repository website in a Chrome browser. The URL bar displays `mvnrepository.com/artifact/org.testng/testng/7.3.0`. The page header includes the Maven Repository logo and a search bar. On the left, there's a sidebar with 'Indexed Artifacts (18.8M)' and a line graph showing project growth over time. Below that, 'Popular Categories' are listed, including 'Aspect Oriented', 'Actor Frameworks', 'Application Metrics', 'Build Tools', 'Bytecode Libraries', 'Command Line Parsers', 'Cache Implementations', 'Cloud Computing', 'Code Analyzers', 'Collections', 'Configuration Libraries', 'Core Utilities', and 'Date and Time Utilities'. The main content area shows the 'TestNG > 7.3.0' page, which is a 'Testing framework for Java'. It lists the license as 'Apache 2.0', categories as 'Testing Frameworks', homepage as 'https://testng.org', date as '(Aug 02, 2020)', files as 'jar (900 KB)' with a 'View All' link, repositories as 'Central' and 'Liferay Public', and 'Used By' as '9,438 artifacts'. At the bottom, there are tabs for 'Maven', 'Gradle', 'SBT', 'Ivy', 'Grape', 'Leiningen', and 'Builder'. The 'Maven' tab is active, showing a dependency declaration snippet:

```
<dependency>
  <groupId>org.testng</groupId>
  <artifactId>testng</artifactId>
  <version>7.3.0</version>
  <scope>test</scope>
</dependency>
```

 There is also a checkbox for 'Include comment with link to declaration'.

Add TestNG dependency in pom.xml file

The screenshot shows the IntelliJ IDEA IDE with a project named 'SampleTestNG'. The 'pom.xml' file is open in the editor, showing the following XML content:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.example</groupId>
  <artifactId>SampleTestNG</artifactId>
  <version>1.0-SNAPSHOT</version>

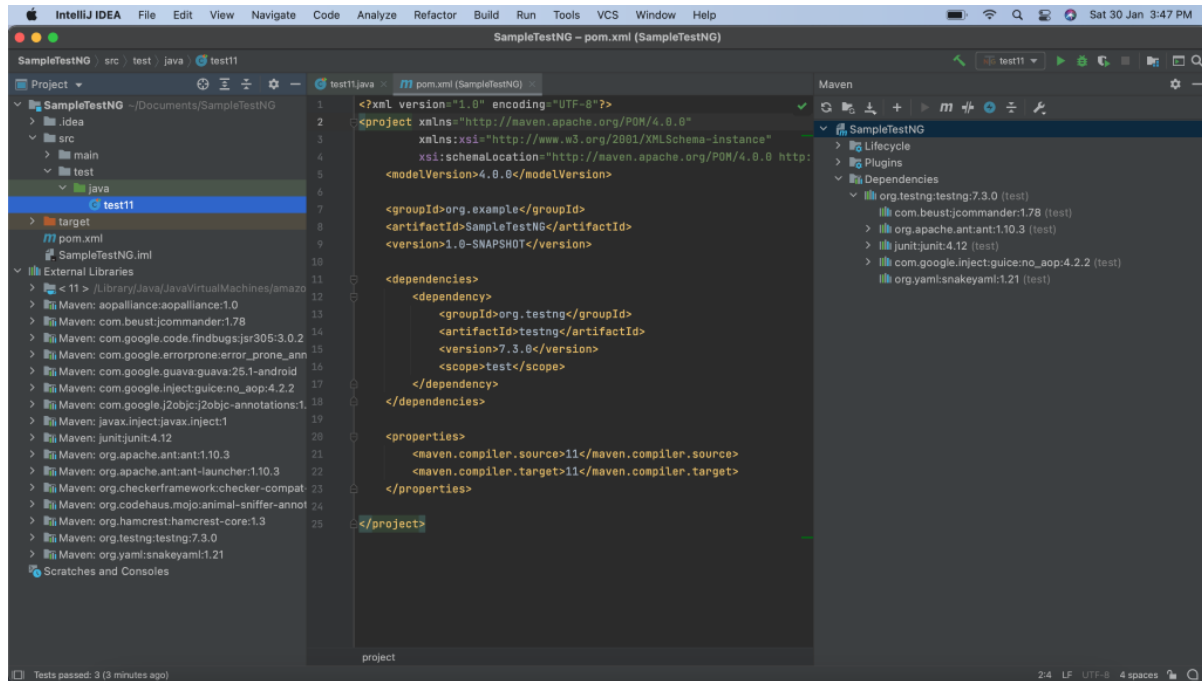
  <dependencies>
    <dependency>
      <groupId>org.testng</groupId>
      <artifactId>testng</artifactId>
      <version>7.3.0</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <properties>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
  </properties>

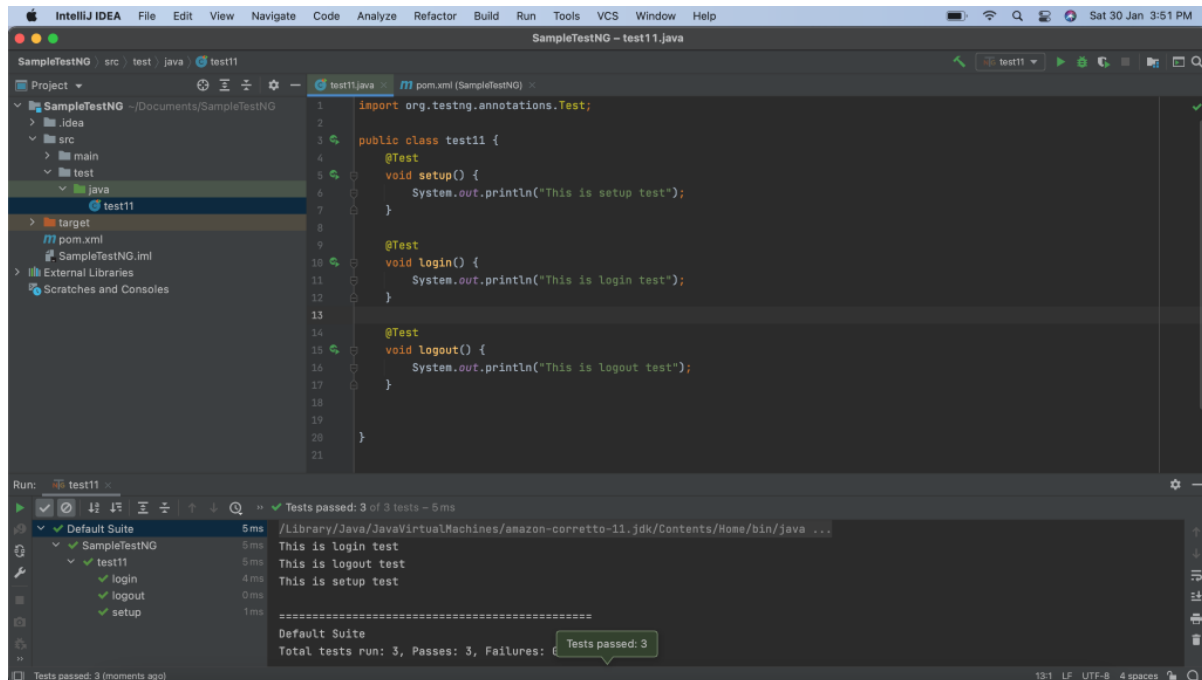
</project>
```

The IDE interface includes a project explorer on the left showing the file structure: 'SampleTestNG' (root), '.idea', 'src' (main, test), 'target', 'pom.xml', 'SampleTestNG.iml', 'External Libraries', and 'Scratches and Consoles'. The bottom status bar shows 'Tests passed: 3 (moments ago)' and '2/4 LF UTF-8 4 spaces'.

We can see reflection of dependencies here



Create a test file and test it. If there are multiple tests in a test file by default all tests follow alphabetical order



Annotations in TestNG:

Sr.No.	Annotation & Description
1	<p data-bbox="350 449 552 485">@BeforeSuite</p> <p data-bbox="350 562 1409 646">The annotated method will be run only once before all tests in this suite have run.</p>
2	<p data-bbox="350 772 524 808">@AfterSuite</p> <p data-bbox="350 886 1409 970">The annotated method will be run only once after all tests in this suite have run.</p>
3	<p data-bbox="350 1098 558 1134">@BeforeClass</p> <p data-bbox="350 1211 1409 1295">The annotated method will be run only once before the first test method in the current class is invoked.</p>
4	<p data-bbox="350 1423 532 1459">@AfterClass</p> <p data-bbox="350 1537 1409 1621">The annotated method will be run only once after all the test methods in the current class have run.</p>

5	<p>@BeforeTest</p> <p>The annotated method will be run before any test method belonging to the classes inside the <test> tag is run.</p>
6	<p>@AfterTest</p> <p>The annotated method will be run after all the test methods belonging to the classes inside the <test> tag have run.</p>
7	<p>@BeforeGroups</p> <p>The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked.</p>
8	<p>@AfterGroups</p> <p>The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked.</p>
9	<p>@BeforeMethod</p> <p>The annotated method will be run before each test method.</p>

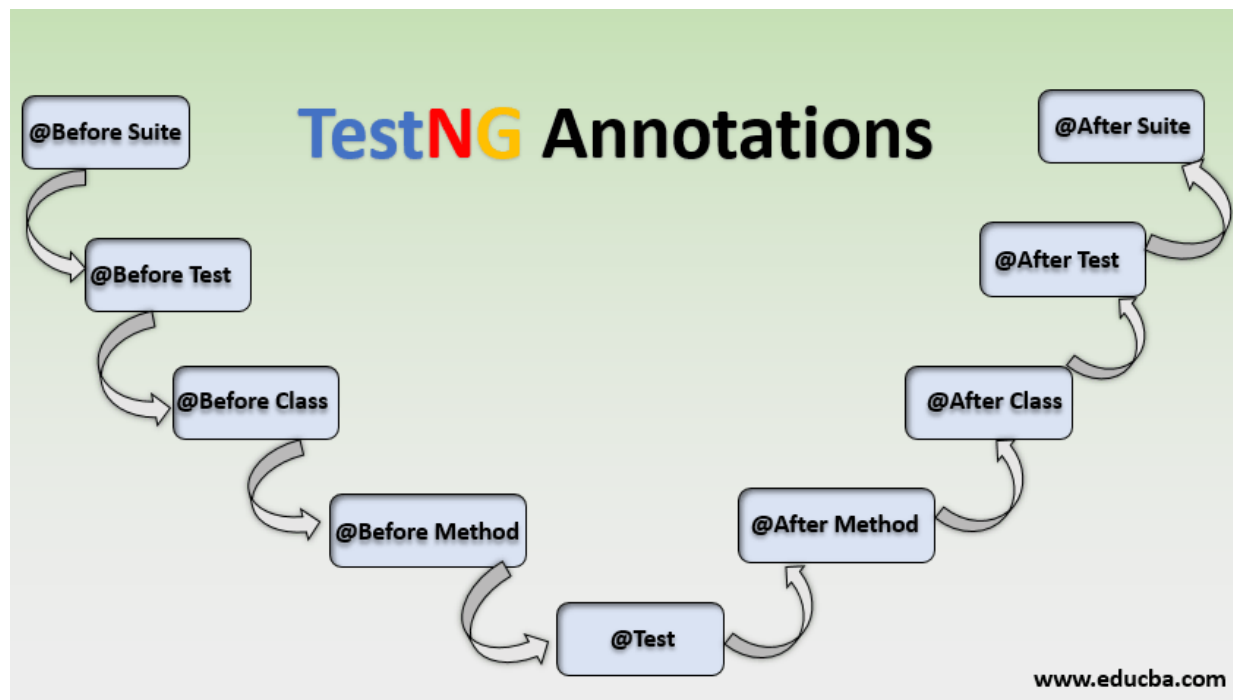
10	<p>@AfterMethod</p> <p>The annotated method will be run after each test method.</p>
11	<p>@DataProvider</p> <p>Marks a method as supplying data for a test method. The annotated method must return an Object[][], where each Object[] can be assigned the parameter list of the test method. The @Test method that wants to receive data from this DataProvider needs to use a dataProvider name equal to the name of this annotation.</p>
12	<p>@Factory</p> <p>Marks a method as a factory that returns objects that will be used by TestNG as Test classes. The method must return Object[].</p>
13	<p>@Listeners</p> <p>Defines listeners on a test class.</p>
14	<p>@Parameters</p> <p>Describes how to pass parameters to a @Test method.</p>

15

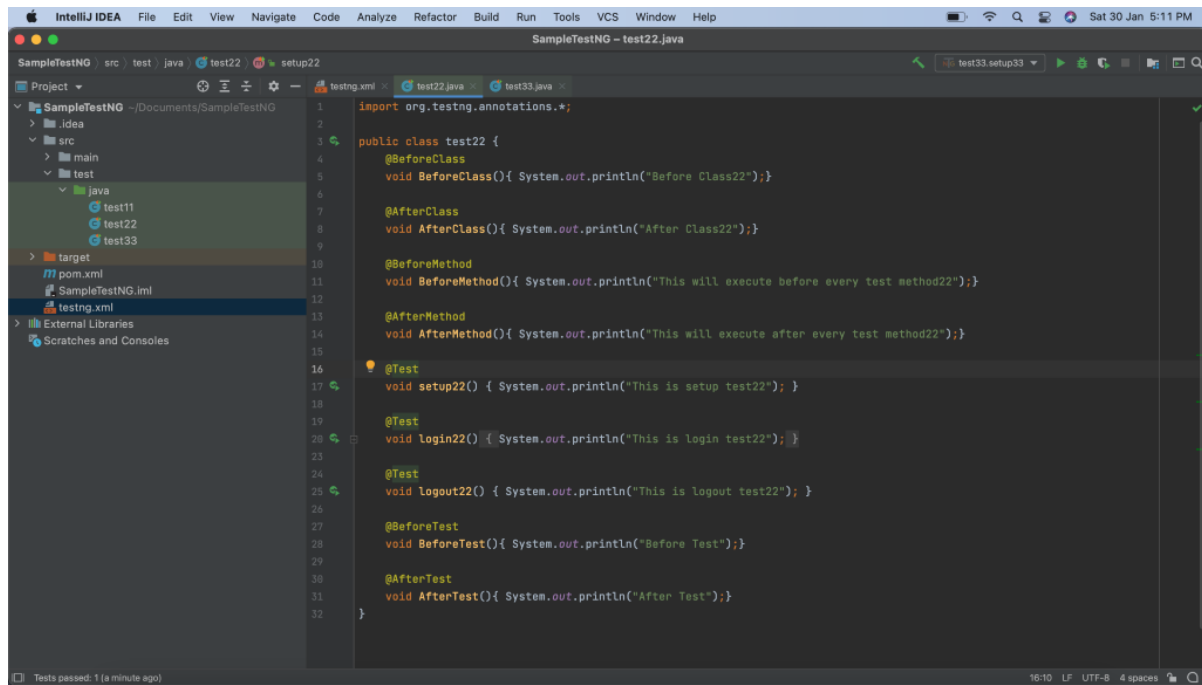
@Test

Marks a class or a method as a part of the test.

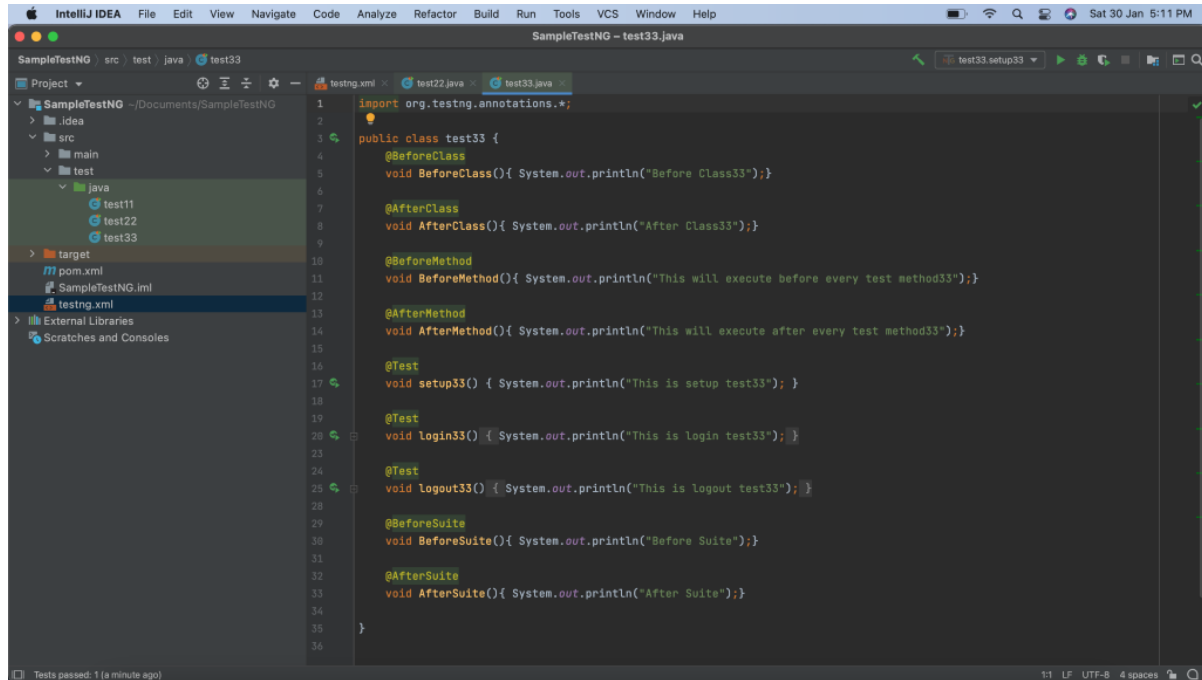
Hierarchy in TestNG annotations



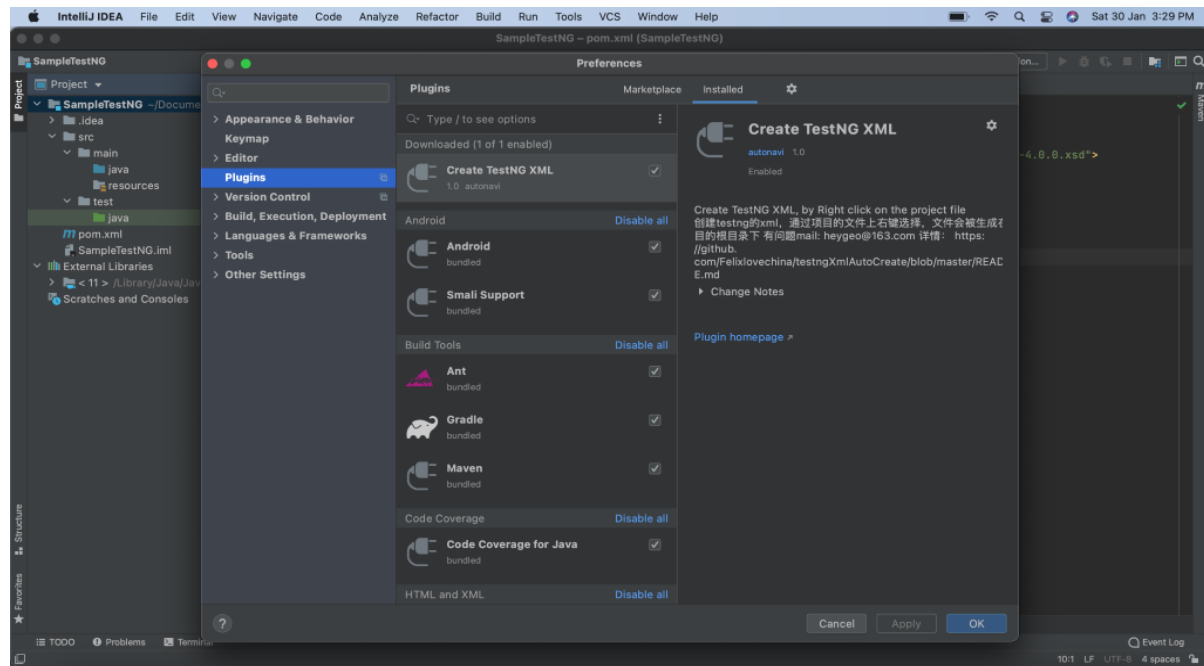
Create a test



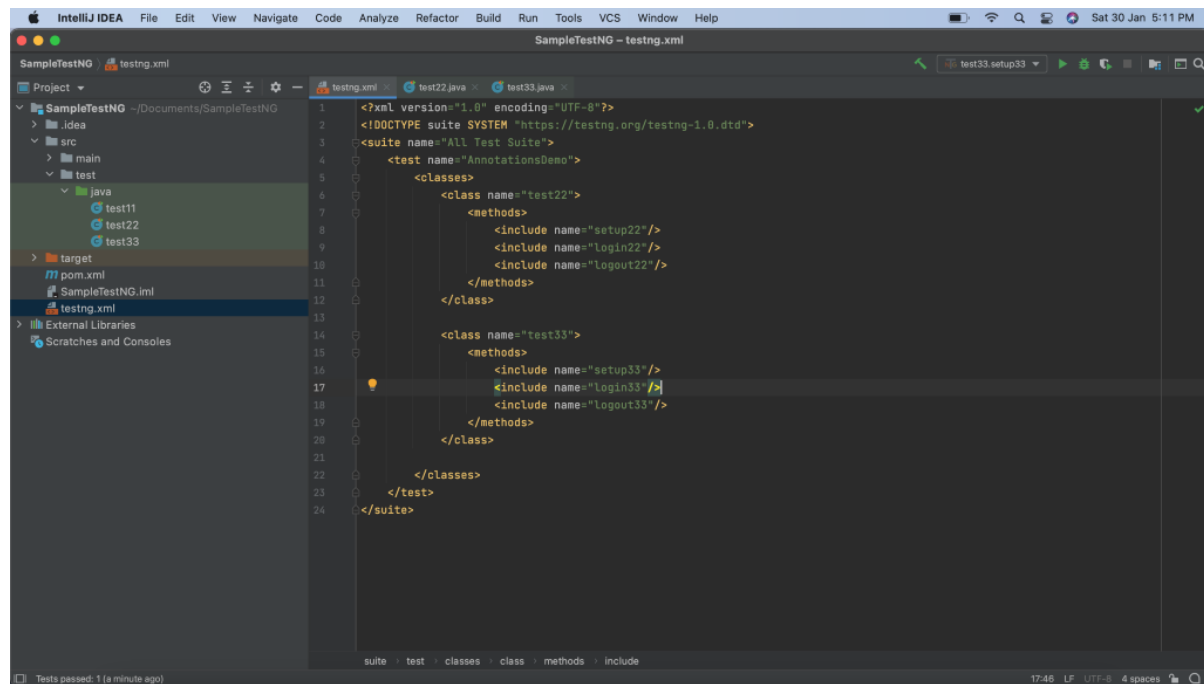
Create a one more test



Create TestNG XML Plugin and Restart IntelliJ IDEA



Edit TestNG.xml file



Result

IntelliJ IDEA File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

SampleTestNG - testng.xml

Run: /Users/gyanpratapsingh/Documents/SampleTestNG/testng.xml

Tests passed: 6 of 6 tests - 21ms

- ✓ All Test Suite 21ms
 - ✓ AnnotationsDemo 21ms
 - ✓ test22 6ms
 - Before Suite
 - Before Test
 - Before Class22
 - This will execute before every test method22
 - This is setup test22
 - This will execute after every test method22
 - This will execute before every test method22
 - This is login test22
 - This will execute after every test method22
 - This will execute before every test method22
 - This is logout test22
 - This will execute after every test method22
 - After Class22
 - After Test
 - After Suite
 - ✓ test33 7ms
 - Before Suite
 - Before Test
 - Before Class33
 - This will execute before every test method33
 - This is setup test33
 - This will execute after every test method33
 - This will execute before every test method33
 - This is login test33
 - This will execute after every test method33
 - This will execute before every test method33
 - This is logout test33
 - This will execute after every test method33
 - After Class33
 - After Test
 - After Suite

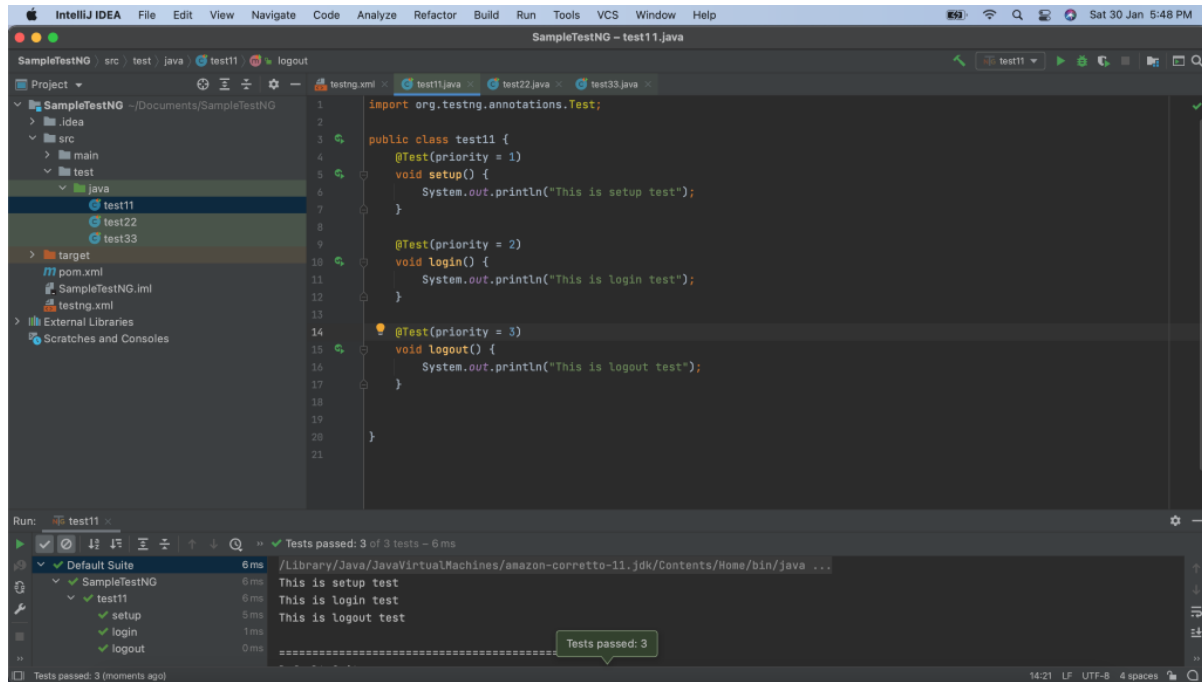
=====

All Test Suite

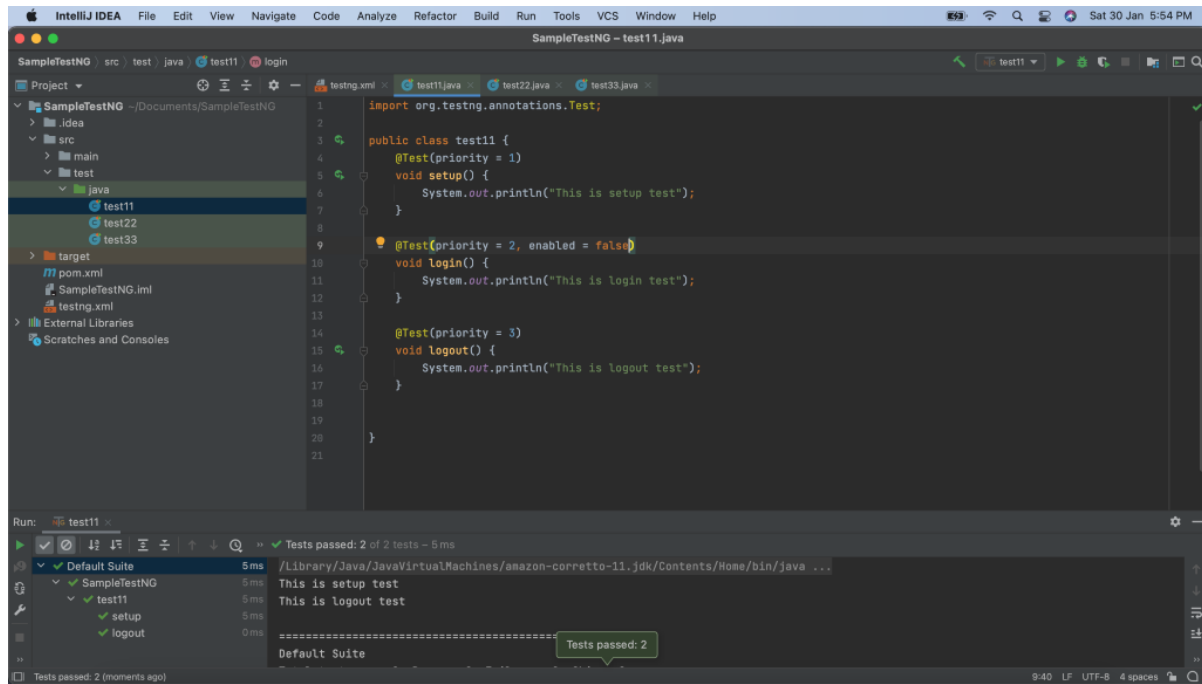
Tests passed: 6 (moments ago)

LF UTF-8 4 spaces

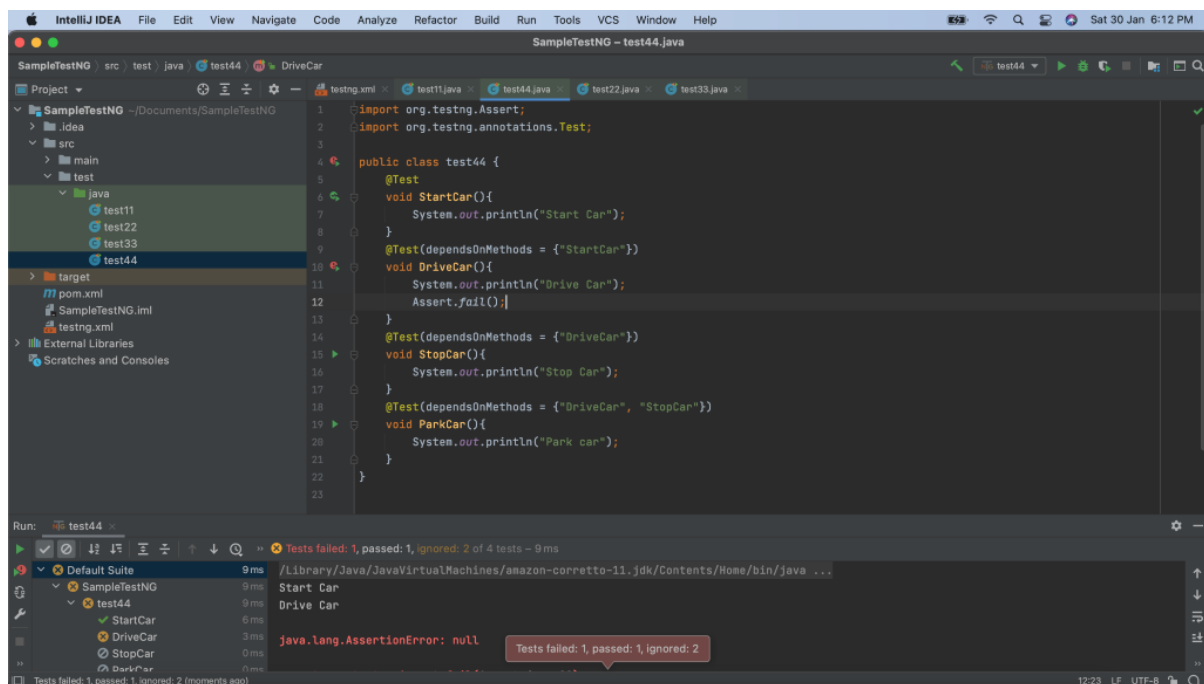
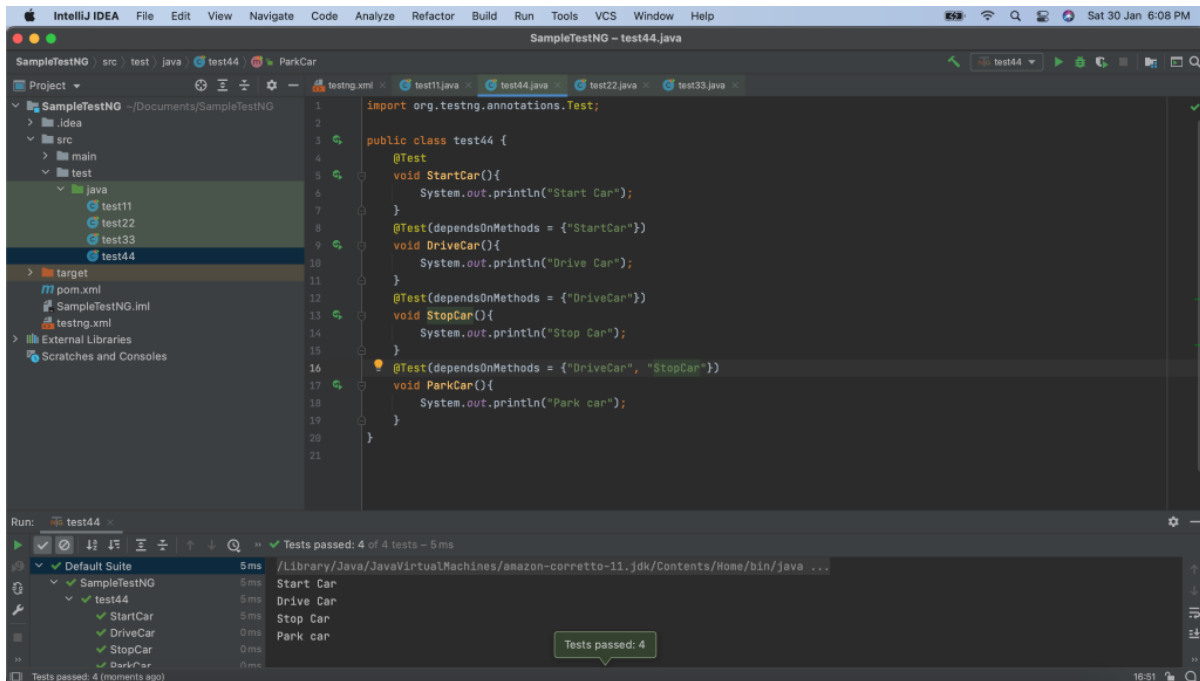
Set priority in TestNG: We can set priority for test cases in order of their execution, by giving priority to each test method. A test method having lower priority runs first then the test methods with higher priority are executed.

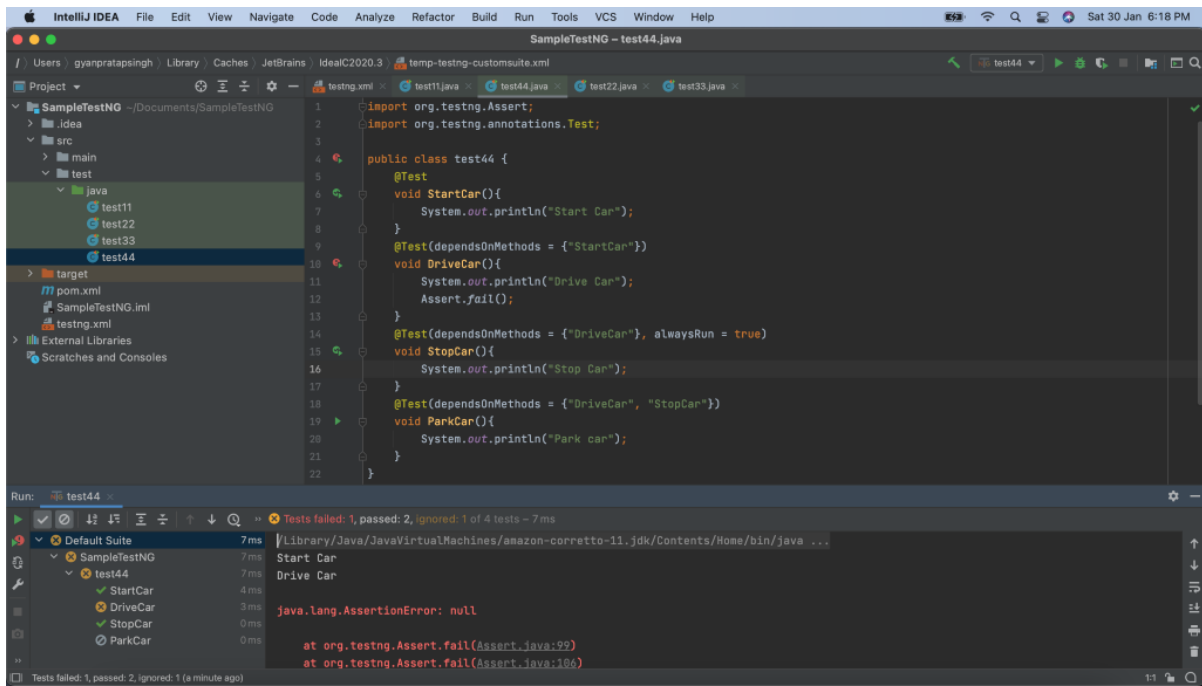


Disable a test in TestNG: Sometimes, it happens that our code is not ready and the test case written to test that method/code fails. In such cases, annotation `@Test(enabled = false)` helps to disable this test case. If a test method is annotated with `@Test(enabled = false)`, then the test case that is not ready to test is bypassed.

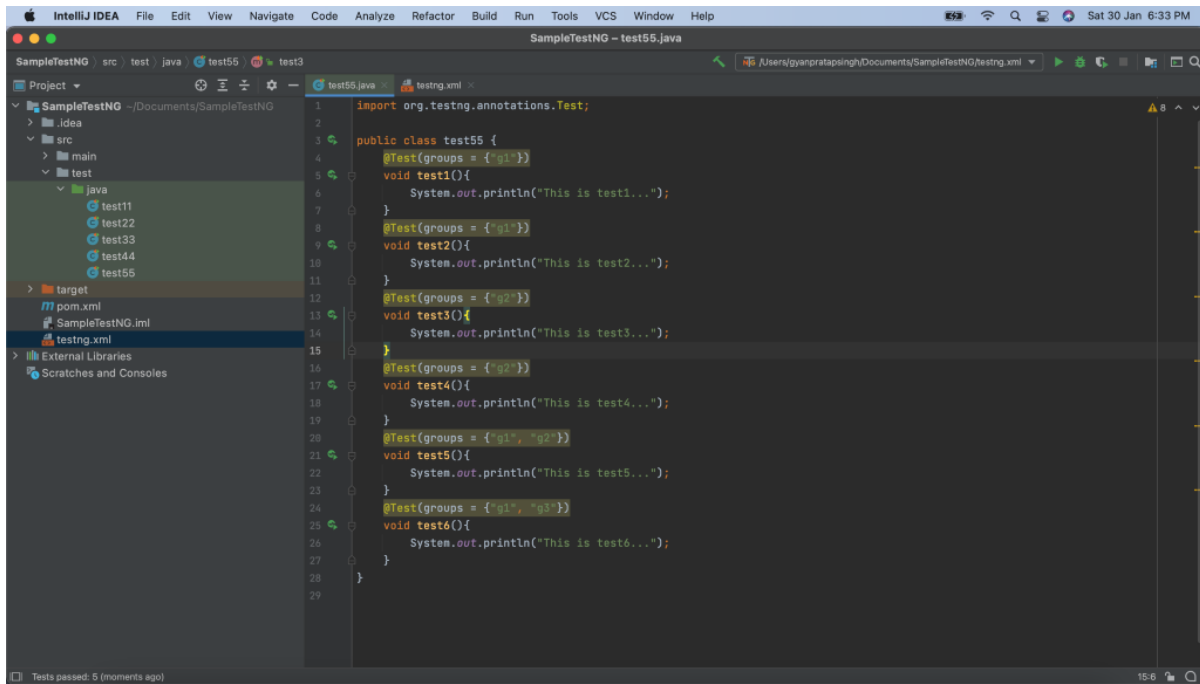


Dependency test in TestNG: Sometimes, you may need to invoke methods in a test case in a particular order, or you may want to share some data and state between methods. This kind of dependency is supported by TestNG, as it supports the declaration of explicit dependencies between test methods





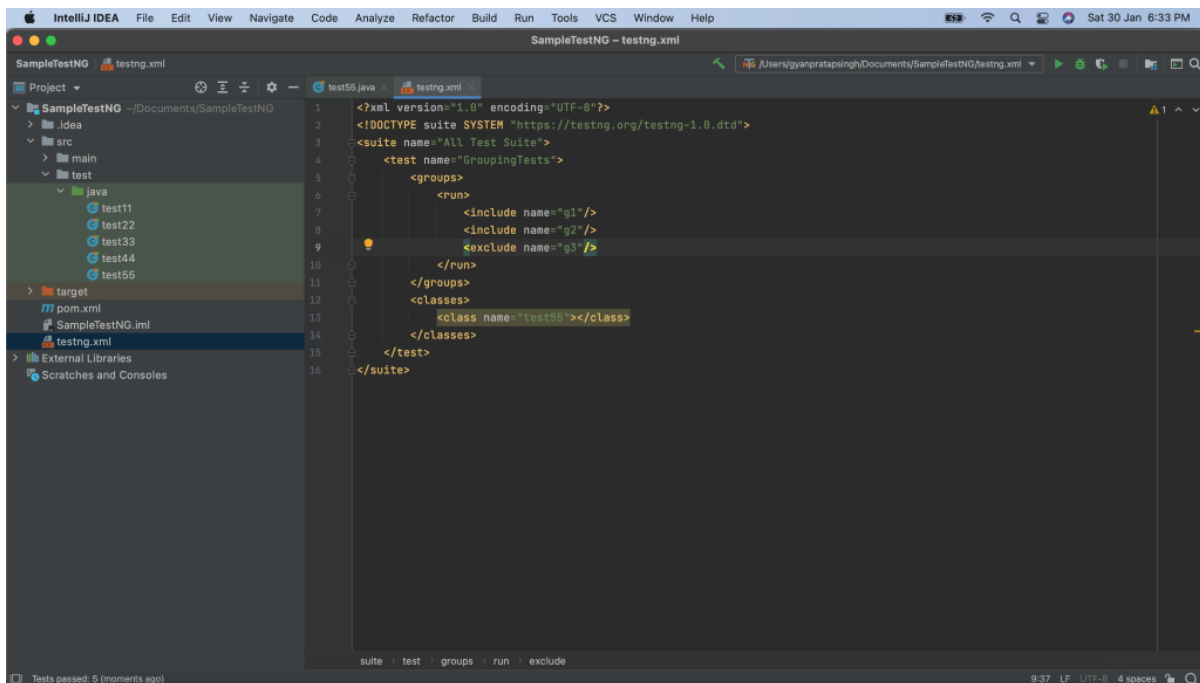
Group test in TestNG:



The screenshot shows the IntelliJ IDEA IDE with the 'SampleTestNG' project open. The 'test55.java' file is selected in the editor, displaying the following code:

```
1 import org.testng.annotations.Test;
2
3 public class test55 {
4     @Test(groups = {"g1"})
5     void test1(){
6         System.out.println("This is test1...");
7     }
8     @Test(groups = {"g1"})
9     void test2(){
10        System.out.println("This is test2...");
11    }
12    @Test(groups = {"g2"})
13    void test3(){
14        System.out.println("This is test3...");
15    }
16    @Test(groups = {"g1"})
17    void test4(){
18        System.out.println("This is test4...");
19    }
20    @Test(groups = {"g1", "g2"})
21    void test5(){
22        System.out.println("This is test5...");
23    }
24    @Test(groups = {"g1", "g1"})
25    void test6(){
26        System.out.println("This is test6...");
27    }
28 }
29
```

The left sidebar shows the project structure with the 'test' directory containing files 'test11', 'test22', 'test33', 'test44', and 'test55'. The bottom status bar indicates 'Tests passed: 5 (moments ago)'.

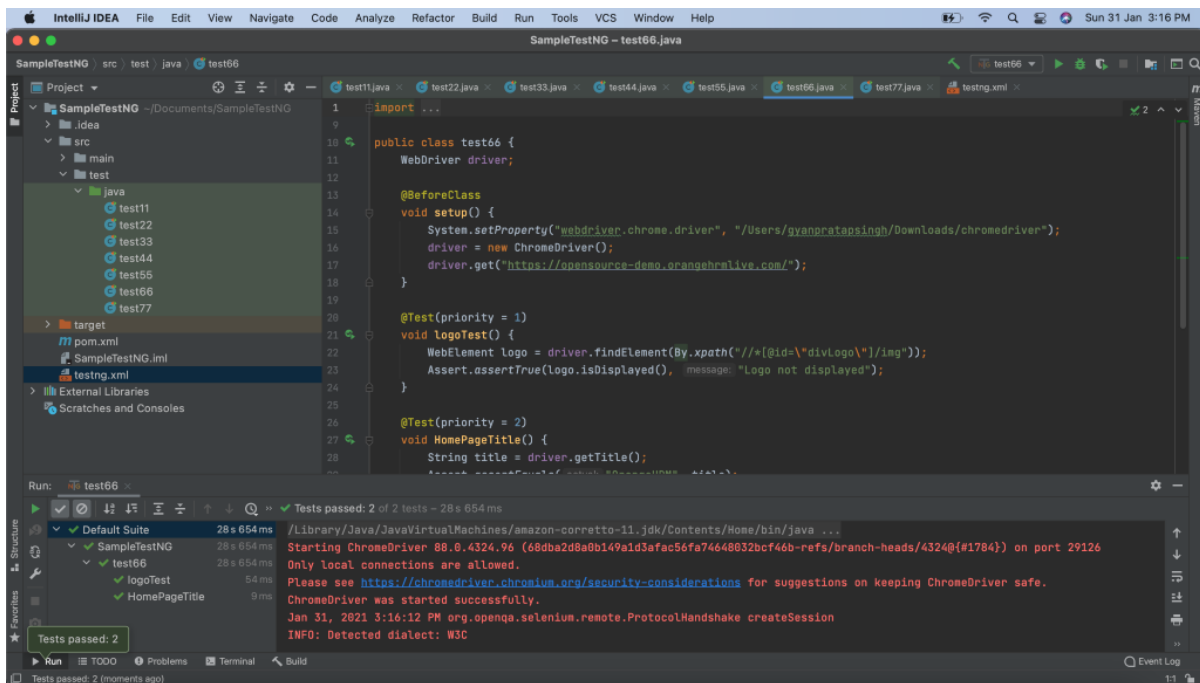
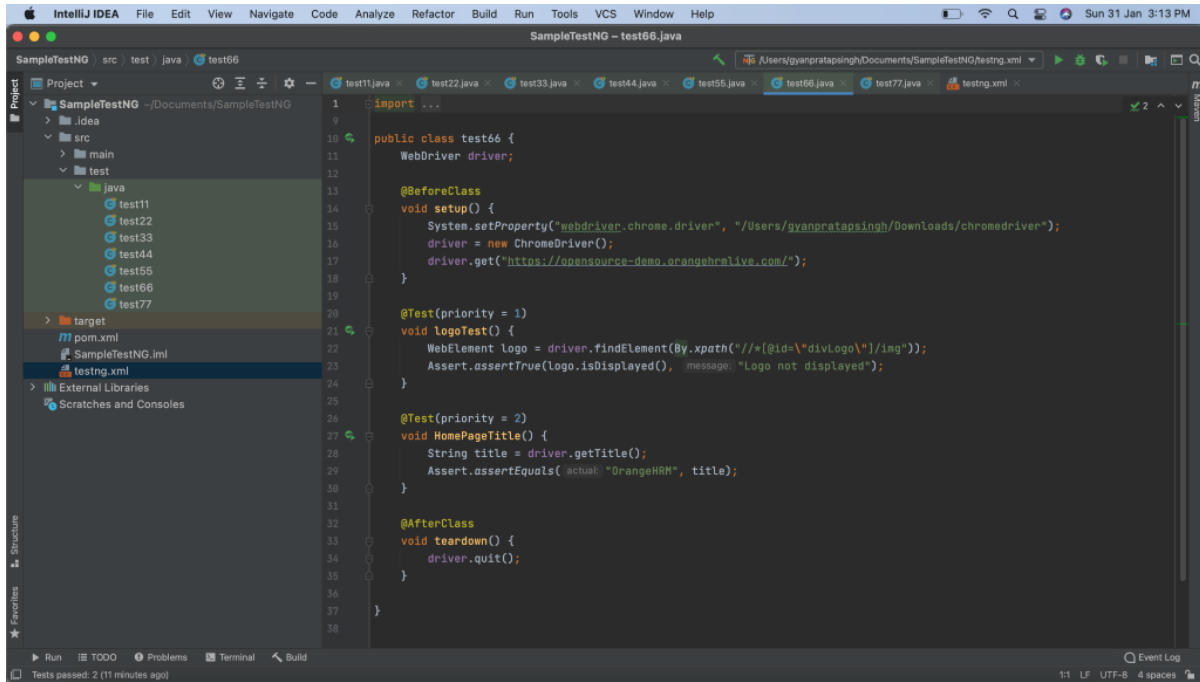


The screenshot shows the IntelliJ IDEA IDE with the 'SampleTestNG' project open. The 'testng.xml' file is selected in the editor, displaying the following configuration:

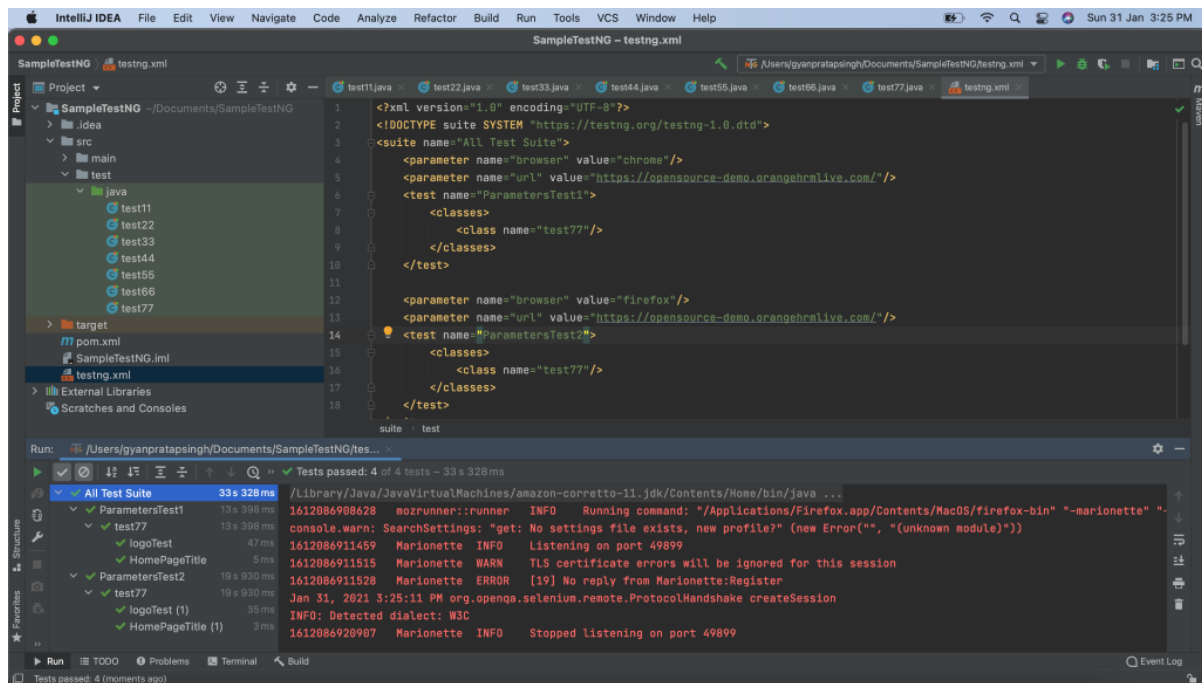
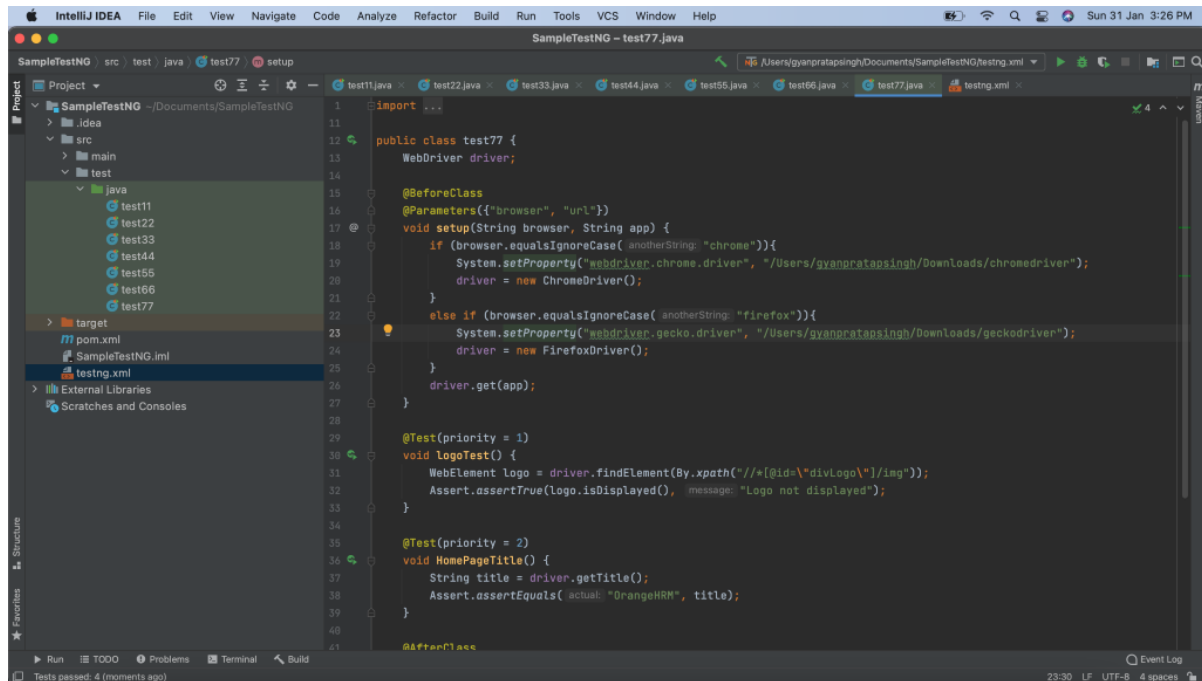
```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
3 <suite name="All Test Suite">
4     <test name="GroupingTests">
5         <groups>
6             <run>
7                 <include name="g1"/>
8                 <include name="g2"/>
9                 <exclude name="g3"/>
10            </run>
11        </groups>
12        <classes>
13            <class name="test55"/></class>
14        </classes>
15    </test>
16 </suite>
```

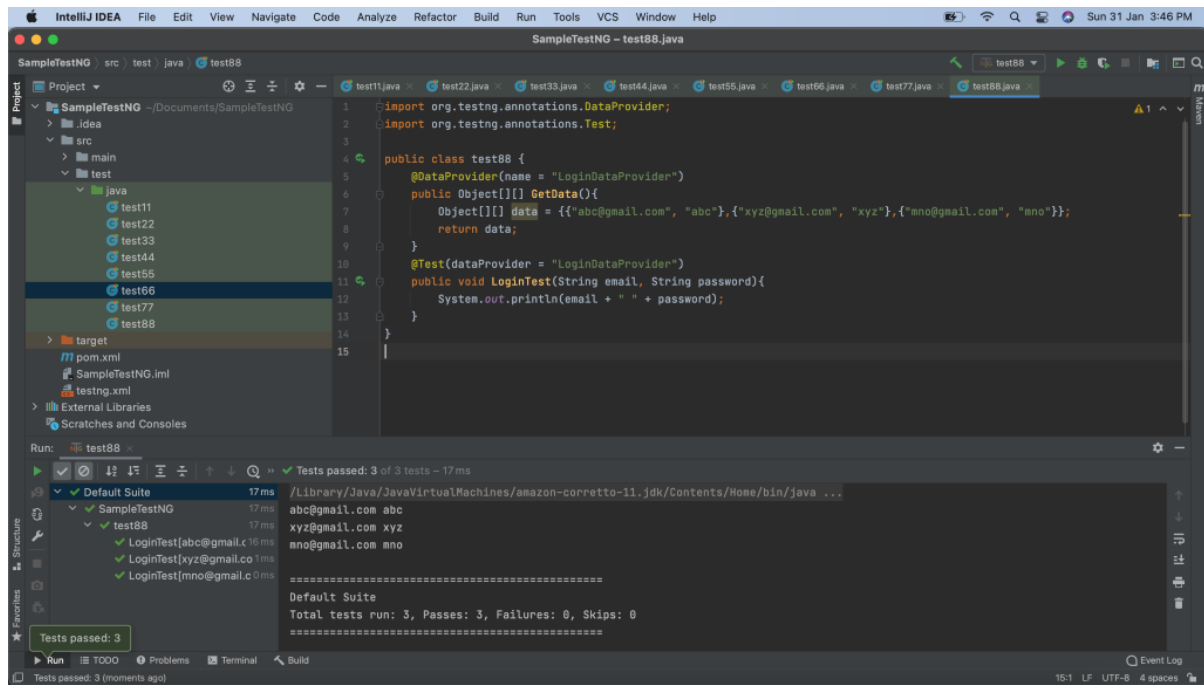
The left sidebar shows the project structure with the 'test' directory containing files 'test11', 'test22', 'test33', 'test44', and 'test55'. The bottom status bar indicates 'Tests passed: 5 (moments ago)'.

Assertions in TestNG:

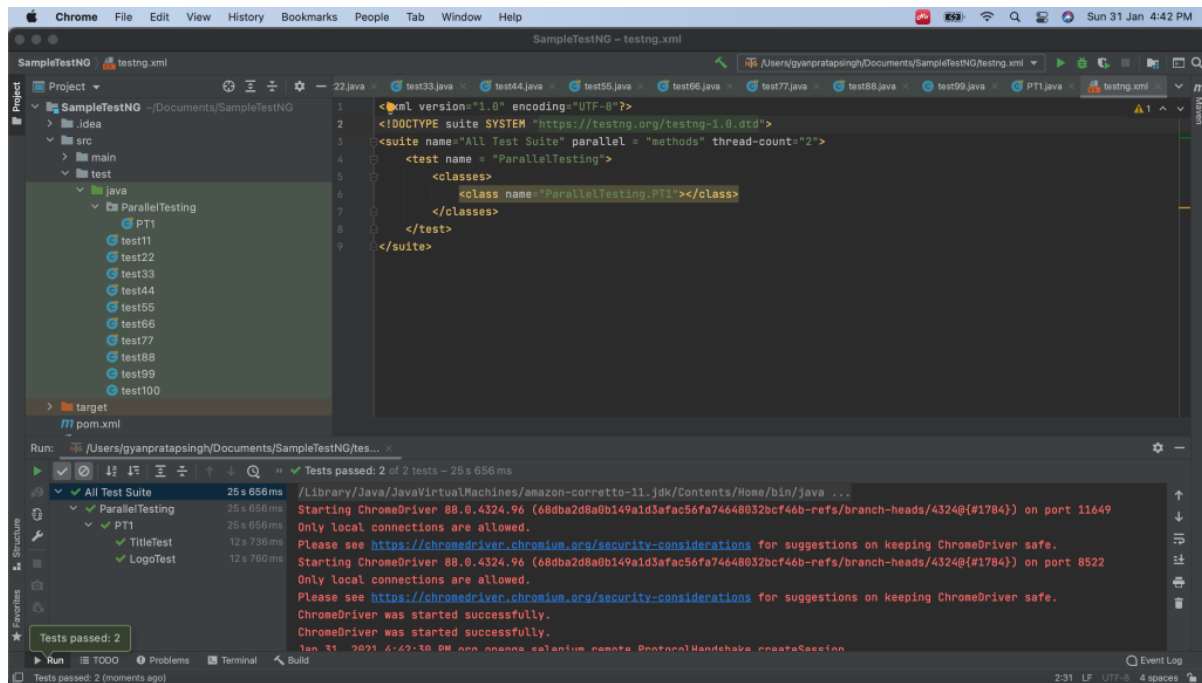
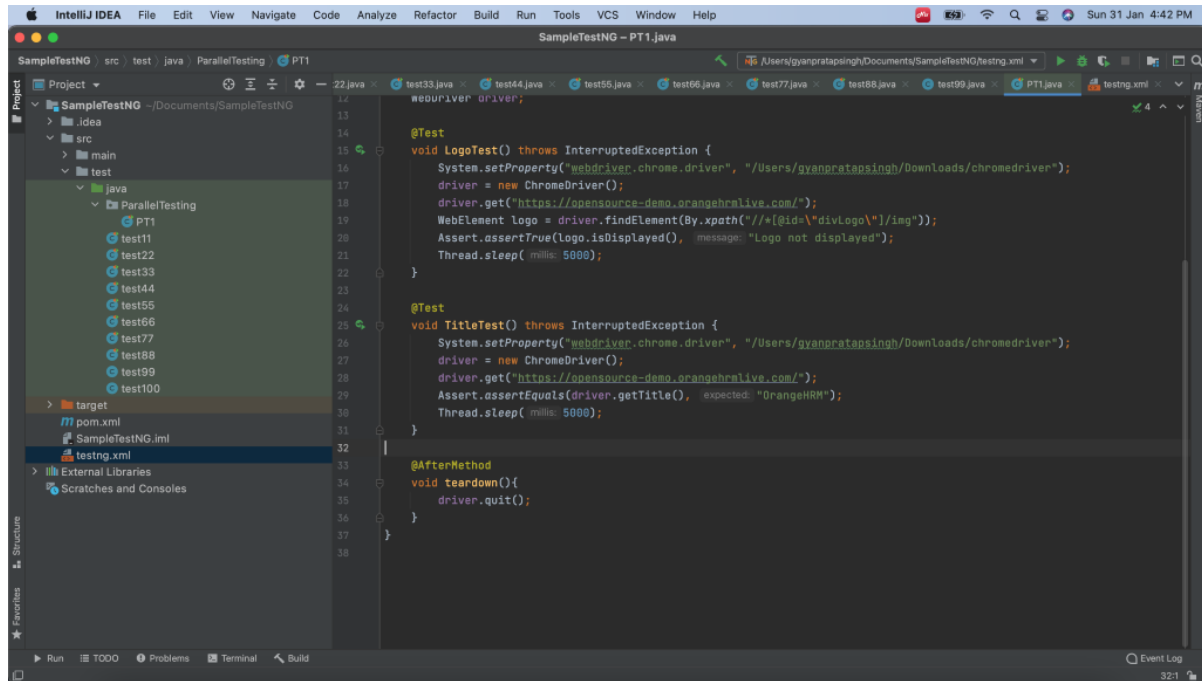


Parameterized test in TestNG: Another interesting feature available in TestNG is parametric testing. In most cases, you'll come across a scenario where the business logic requires a hugely varying number of tests. Parameterized tests allow developers to run the same test over and over again using different values.





Parallel testing in TestNG:



Listeners in TestNG:

This screenshot shows the IntelliJ IDEA interface with the 'SampleTestNG' project open. The 'Project' view on the left shows the file structure, including 'src/test/java/LisExam/LTE1'. The main editor displays the code for 'LTE1.java'. The code imports 'org.testng.Assert', 'org.testng.SkipException', 'org.testng.annotations.Listeners', and 'org.testng.annotations.Test'. It uses the '@Listeners(LisExam.LTE2.class)' annotation on the 'LTE1' class. The 'LTE1' class contains three test methods: 'test1()', 'test2()', and 'test3()'. 'test1()' and 'test2()' use 'Assert.assertEquals()' to verify test results. 'test3()' throws a 'SkipException' with the message 'This is skip exception'.

```
1 import org.testng.Assert;
2 import org.testng.SkipException;
3 import org.testng.annotations.Listeners;
4 import org.testng.annotations.Test;
5
6 @Listeners(LisExam.LTE2.class)
7 public class LTE1 {
8     @Test
9     void test1() {
10         System.out.println("This is test1");
11         Assert.assertEquals(actual: "A", expected: "A");
12     }
13
14     @Test
15     void test2() {
16         System.out.println("This is test2");
17         Assert.assertEquals(actual: "A", expected: "B");
18     }
19
20     @Test
21     void test3() {
22         System.out.println("This is test3");
23         throw new SkipException("This is skip exception");
24     }
25 }
```

This screenshot shows the IntelliJ IDEA interface with the 'SampleTestNG' project open. The 'Project' view on the left shows the file structure, including 'src/test/java/LisExam/LTE2'. The main editor displays the code for 'LTE2.java'. The code imports 'org.testng.ITestContext', 'org.testng.IEventListener', and 'org.testng.ITestResult'. It implements the 'IEventListener' interface in the 'LTE2' class. The 'LTE2' class contains several methods: 'onStart()', 'onFinish()', 'onTestStart()', 'onTestSkipped()', 'onTestSuccess()', 'onTestFailure()', and 'onTestFailedButWithinSuccessPercentage()'. Each method prints a message to the console indicating the state of the test execution.

```
1 package LisExam;
2
3 import org.testng.ITestContext;
4 import org.testng.IEventListener;
5 import org.testng.ITestResult;
6
7 public class LTE2 implements IEventListener {
8     @Override
9     public void onStart(ITestContext arg0) { System.out.println("Starts test execution..." + arg0.getName()); }
10
11     @Override
12     public void onFinish(ITestContext arg0) { System.out.println("Finish test execution..." + arg0.getName()); }
13
14     @Override
15     public void onTestStart(ITestResult arg0) { System.out.println("Starts test..." + arg0.getName()); }
16
17     @Override
18     public void onTestSkipped(ITestResult arg0) {
19         System.out.println("Skipped test..." + arg0.getName());
20     }
21
22     @Override
23     public void onTestSuccess(ITestResult arg0) {
24         System.out.println("Passed test..." + arg0.getName());
25     }
26
27     @Override
28     public void onTestFailure(ITestResult arg0) {
29         System.out.println("Failed test..." + arg0.getName());
30     }
31
32     @Override
33     public void onTestFailedButWithinSuccessPercentage(ITestResult arg0) {
34         // TODO
35     }
36 }
```

