Let's take a scenario where sequencing will be required in order to pass all test cases:

**Scenario:** Generate a code where you are required to perform a Google search with a specific keyword say "Facebook". Now, verify that Browser title is changed to "Facebook - Google Search".

Note: Each step which you code should be in separate methods

**Method 1**: Open Browser say Firefox (openBrowser())

**Method 2**: Launch Google.com (launchGoogle())

**Method 3**: Perform a search using "Facebook" (performSearchAndClick1stLink())

**Method 4**: Verify Google search page title (FaceBookPageTitleVerification())

**Code for our scenario**:

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.Assert;

import org.testng.annotations.Test;

public class Priority\_In\_testNG {

WebDriver driver;

// Method 1: Open Brower say Firefox

@Test

public void openBrowser() {

driver = new FirefoxDriver();

}

// Method 2: Launch Google.com

@Test

public void launchGoogle() {

driver.get("http://www.google.co.in");

}

// Method 3: Perform a search using "Facebook"

@Test

public void peformSeachAndClick1stLink() {

driver.findElement(By.xpath(".//\*[@title='Search']")).sendKeys("Facebook");

}

// Method 4: Verify Google search page title.

@Test

public void FaceBookPageTitleVerification() throws Exception {

driver.findElement(By.xpath(".//\*[@value='Search']")).click();

Thread.sleep(3000);

Assert.assertEquals(driver.getTitle().contains("Facebook - Google Search"), true);

}

}

**Explanation of Code**

As mentioned above we have created 4 test cases for performing each action in an independent methods.

* The first method **(openBrowser)** states to initialize Firefox browser.
* The second method **(launchGoogle)** states that launch Google.com is in the initialized browser.
* The third method **(peformSeachAndClick1stLink)**states that perform a search in the search box (with xpath **(".//\*[@title='Search']")** with a search term as **Facebook** and
* The fourth and last method **(FaceBookPageTitleVerification)** states that click on search icon of Google and verify that browser title has been changed to **Facebook - Google Search.**

Now run this code using testNG as shown in the video you will find all the [Test Case](https://www.guru99.com/test-case.html) are failing. The reason for failure: as there is a dependency of previous test case to pass, only than current running test case will be passed.

In this case,

* First method which is executed is **openBrowser().** It got passed because it does not have any dependency.
* Second method executed is **FaceBookPageTitleVerification();** it is failing because we are trying to click search button and verifying browser title.
* You can see that if search activity is not process then how any other step can get passed. Hence, this is the reason my test cases are failing.

PASSED: openBrowser

FAILED: FaceBookPageTitleVerification

FAILED: launchGoogle

FAILED: peformSeachAndClick1stLink

## Demo of TestNG code without Priority in Alphabetical Order

If we don’t mention any priority, testng will execute the @Test methods based on alphabetical order of their method names irrespective of their place of implementation in the code.

package com.guru.testngannotations;

import org.testng.annotations.Test;

public class TestNG\_Priority\_Annotations {

@Test

public void c\_method(){

System.out.println("I'm in method C");

}

@Test

public void b\_method(){

System.out.println("I'm in method B");

}

@Test

public void a\_method(){

System.out.println("I'm in method A");

}

@Test

public void e\_method(){

System.out.println("I'm in method E");

}

@Test

public void d\_method(){

System.out.println("I'm in method D");

}

}

Output

I'm in method A

I'm in method B

I'm in method C

I'm in method D

I'm in method E

Though we defined the methods in a random manner (c, b, a, e, d), testng executed the methods based on their method names by considering alphabetical order and the same was reflected in the output as well.

**How to set Priority in TestNG**

As you have seen in the previous example that sequencing required in order to pass this scenario, so we will be modifying the previous piece of code with **Priority Parameter** so that each test should run against to the priority assigned to them.

Now as you can see we have assigned the Priority to each test case means test case will the lower priority value will be executed first.

**Priority in testNG in action**

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.Assert;

import org.testng.annotations.Test;

public class Priority\_In\_testNG {

WebDriver driver;

// Method 1: Open Browser say Firefox

@Test (priority=1)

public void openBrowser() {

driver = new FirefoxDriver();

}

// Method 2: Launch Google.com

@Test (priority=2)

public void launchGoogle() {

driver.get("http://www.google.co.in");

}

// Method 3: Perform a search using "Facebook"

@Test (priority=3)

public void peformSeachAndClick1stLink() {

driver.findElement(By.xpath(".//\*[@title='Search']")).sendKeys("Facebook");

}

// Method 4: Verify Google search page title.

@Test (priority=4)

public void FaceBookPageTitleVerification() throws Exception {

driver.findElement(By.xpath(".//\*[@value='Search']")).click();

Thread.sleep(3000);

Assert.assertEquals(driver.getTitle().contains("Facebook - Google Search"), true);

}

}

**Output** :

PASSED: openBrowser

PASSED: launchGoogle

PASSED: peformSearchAndClick1stLink

PASSED: FaceBookPageTitleVerification

**Practice examples:**package com.guru.testngannotations;

import org.testng.annotations.Test;

public class TestNG\_Priority\_Annotations {

@Test(priority=6)

public void c\_method(){

System.out.println("I'm in method C");

}

@Test(priority=9)

public void b\_method(){

System.out.println("I'm in method B");

}

@Test(priority=1)

public void a\_method(){

System.out.println("I'm in method A");

}

@Test(priority=0)

public void e\_method(){

System.out.println("I'm in method E");

}

@Test(priority=3)

public void d\_method(){

System.out.println("I'm in method D");

}

}

Output

I'm in method E

I'm in method A

I'm in method D

I'm in method C

I'm in method B

Here we have provided the priorities as 0,1,3,6,9. So, method having 0 as priority is executed first and then method having priority-1 and so on. Here alphabetical order method name won’t be considered as we provided the priorities

## Methods with Same Priority:

There may be a chance that methods may contain same priority. In those cases, testng considers the alphabetical order of the method names whose priority is same.

package com.guru.testngannotations;

import org.testng.annotations.Test;

public class TestNG\_Priority\_Annotations {

@Test(priority=6)

public void c\_method(){

System.out.println("I'm in method C");

}

@Test(priority=9)

public void b\_method(){

System.out.println("I'm in method B");

}

@Test(priority=6)

public void a\_method(){

System.out.println("I'm in method A");

}

@Test(priority=0)

public void e\_method(){

System.out.println("I'm in method E");

}

@Test(priority=3)

public void d\_method(){

System.out.println("I'm in method D");

}

}

Output

I'm in method E

I'm in method D

I'm in method A

I'm in method C

I'm in method B

Here ‘e’ and ‘d’ are executed based on their priority values. But the methods ‘a’ and ‘c’ contains the same priority value(6). So, here testng considers the alphabetical order of ‘a’ and ’c’ and executes them accordingly.

## Combining both prioritized(having same priority) and non-prioritized methods:

In this case, we’ll cover two cases in one testng class.

1. Methods having same priority value.
2. More than one non-prioritized methods.

package com.guru.testngannotations;

import org.testng.annotations.Test;

public class TestNG\_Priority\_Annotations {

@Test()

public void c\_method(){

System.out.println("I'm in method C");

}

@Test()

public void b\_method(){

System.out.println("I'm in method B");

}

@Test(priority=6)

public void a\_method(){

System.out.println("I'm in method A");

}

@Test(priority=0)

public void e\_method(){

System.out.println("I'm in method E");

}

@Test(priority=6)

public void d\_method(){

System.out.println("I'm in method D");

}

}

**Output:**

I'm in method B

I'm in method C

I'm in method E

I'm in method A

I'm in method D

PASSED: b\_method

PASSED: c\_method

PASSED: e\_method

PASSED: a\_method

PASSED: d\_method

**Explanation:**

**First preference:** Non-prioritized methods: ‘c’ and ‘b’: Based on alphabetical order ‘b’ was executed first and then ‘c’.

**Second preference:** Prioritized methods: ‘a’, ‘e’ and ‘d’: ‘e’ was executed first as it was having highest priority(0). As the priority of ‘a’ and ‘d’ methods were same, testng considered the alphabetical order of their methods names. So, between them, ‘a’ was executed first and then ‘d’.

**Case-sensitive in TestNG**

Just for your information there is a standard syntax for defining priority in testNG i.e. **@Test (priority=4)**,suppose you are defining it in some other syntax say **@Test (PRIORITY=1)** then your IDE will show it as a compilation error. Refer image below:

[](https://www.guru99.com/images/Sap-QM/Test_case_priority_TestNG.jpg)