TestNG Parallel Execution Explanation

# Response 1: Parallel="methods" and parallel="classes"

If you change parallel="methods" to parallel="classes", the behavior will change significantly   
 because TestNG will execute entire test classes in parallel rather than individual test methods.  
   
 Let's break it down:  
   
 With parallel="classes", TestNG will run entire test classes in parallel. Each class will run in a separate thread,   
 but within each class, the test methods will be executed sequentially (not in parallel).  
   
 Example Configuration:  
 <suite name="Class Parallel Suite" parallel="classes" thread-count="3">  
 <test name="Test Group">  
 <classes>  
 <class name="com.example.tests.Class1"/>  
 <class name="com.example.tests.Class2"/>  
 <class name="com.example.tests.Class3"/>  
 <class name="com.example.tests.Class4"/>  
 </classes>  
 </test>  
 </suite>  
   
 Test Classes with 4 Methods Each:  
 public class Class1 {  
 @Test  
 public void test1() {  
 System.out.println("Test1 from Class1");  
 }  
   
 @Test  
 public void test2() {  
 System.out.println("Test2 from Class1");  
 }  
   
 @Test  
 public void test3() {  
 System.out.println("Test3 from Class1");  
 }  
   
 @Test  
 public void test4() {  
 System.out.println("Test4 from Class1");  
 }  
 }  
 public class Class2 {  
 @Test  
 public void test1() {  
 System.out.println("Test1 from Class2");  
 }  
   
 @Test  
 public void test2() {  
 System.out.println("Test2 from Class2");  
 }  
   
 @Test  
 public void test3() {  
 System.out.println("Test3 from Class2");  
 }  
   
 @Test  
 public void test4() {  
 System.out.println("Test4 from Class2");  
 }  
 }  
 public class Class3 {  
 @Test  
 public void test1() {  
 System.out.println("Test1 from Class3");  
 }  
   
 @Test  
 public void test2() {  
 System.out.println("Test2 from Class3");  
 }  
   
 @Test  
 public void test3() {  
 System.out.println("Test3 from Class3");  
 }  
   
 @Test  
 public void test4() {  
 System.out.println("Test4 from Class3");  
 }  
 }  
 public class Class4 {  
 @Test  
 public void test1() {  
 System.out.println("Test1 from Class4");  
 }  
   
 @Test  
 public void test2() {  
 System.out.println("Test2 from Class4");  
 }  
   
 @Test  
 public void test3() {  
 System.out.println("Test3 from Class4");  
 }  
   
 @Test  
 public void test4() {  
 System.out.println("Test4 from Class4");  
 }  
 }  
   
 Execution Flow with parallel="classes" and thread-count=3:  
 TestNG starts executing with 3 threads since thread-count=3:  
 - Class1, Class2, and Class3 will run in parallel on 3 separate threads.  
 - Each class runs sequentially within its thread (4 methods of Class1, then Class2, and so on).  
 - After one class finishes, TestNG will start executing Class4 in the remaining thread.  
   
 Visual Flow:  
 1. Initially: Class1, Class2, and Class3 will run in parallel, each using a separate thread.  
 2. After one class finishes: Class4 will begin execution in the next available thread.  
   
 Summary:  
 - parallel="classes" ensures that entire test classes run in parallel.  
 - The methods within each class still run sequentially in the same thread (not parallelized).  
 - thread-count=3 means TestNG will run 3 test classes in parallel.

# Response 2: Clarification on Execution with 4 Methods in Each Class

If each class contains 4 test methods and you set parallel="classes", TestNG will run entire test classes in parallel   
 (as explained previously). Each class will run in a separate thread, but within each class, the test methods will be executed sequentially,   
 not in parallel (because TestNG is parallelizing at the class level, not the method level).  
   
 Let's look at an example to understand this better:  
   
 Example Configuration:  
 <suite name="Class Parallel Suite" parallel="classes" thread-count="3">  
 <test name="Test Group">  
 <classes>  
 <class name="com.example.tests.Class1"/>  
 <class name="com.example.tests.Class2"/>  
 <class name="com.example.tests.Class3"/>  
 <class name="com.example.tests.Class4"/>  
 </classes>  
 </test>  
 </suite>  
   
 Test Classes with 4 Methods Each:  
 public class Class1 {  
 @Test  
 public void test1() {  
 System.out.println("Test1 from Class1");  
 }  
   
 @Test  
 public void test2() {  
 System.out.println("Test2 from Class1");  
 }  
   
 @Test  
 public void test3() {  
 System.out.println("Test3 from Class1");  
 }  
   
 @Test  
 public void test4() {  
 System.out.println("Test4 from Class1");  
 }  
 }  
 public class Class2 {  
 @Test  
 public void test1() {  
 System.out.println("Test1 from Class2");  
 }  
   
 @Test  
 public void test2() {  
 System.out.println("Test2 from Class2");  
 }  
   
 @Test  
 public void test3() {  
 System.out.println("Test3 from Class2");  
 }  
   
 @Test  
 public void test4() {  
 System.out.println("Test4 from Class2");  
 }  
 }  
 public class Class3 {  
 @Test  
 public void test1() {  
 System.out.println("Test1 from Class3");  
 }  
   
 @Test  
 public void test2() {  
 System.out.println("Test2 from Class3");  
 }  
   
 @Test  
 public void test3() {  
 System.out.println("Test3 from Class3");  
 }  
   
 @Test  
 public void test4() {  
 System.out.println("Test4 from Class3");  
 }  
 }  
 public class Class4 {  
 @Test  
 public void test1() {  
 System.out.println("Test1 from Class4");  
 }  
   
 @Test  
 public void test2() {  
 System.out.println("Test2 from Class4");  
 }  
   
 @Test  
 public void test3() {  
 System.out.println("Test3 from Class4");  
 }  
   
 @Test  
 public void test4() {  
 System.out.println("Test4 from Class4");  
 }  
 }  
   
 Execution Flow with parallel="classes" and thread-count=3:  
 1. TestNG starts executing with 3 threads.  
 2. Class1, Class2, and Class3 run in parallel using 3 separate threads.  
 3. After one class finishes, Class4 begins execution in the next available thread.  
   
 Summary:  
 - parallel="classes" runs entire test classes in parallel, not individual methods.  
 - Methods within each class are still executed sequentially.  
 - thread-count=3 means 3 test classes run in parallel, one after the other.