# [Differences Between JUnit3 and JUnit4](https://wiki.eisgroup.com/display/CRC/Differences+Between+JUnit3+and+JUnit4)

1. JUnit4 test cases no longer need to extend junit.framework.TestCase class. Any class with zero argument constructor can act as a test class
2. Test case names no longer need to follow testXXX pattern. Any method that we want to be considered as test method needs to be annotated with @Test in JUnit4
3. Instead of setUp() and tearDown() methods which are executed before and after each test case in JUnit3, we can annotate any number of methods that we want to execute before every test case by @Before annotation. Similarly methods that we want to be executed after each test case can be annotated by @After.
   1. However @Before and @After annotated methods must be public and return void.
4. If we want some common methods that must be executed not before and after each test case but a whole set of test, we can annotate them with @BeforeClass and @AfterClass.
   1. For example if we want some resource setup code which sets up same resources for each test case in class ex. database connectivity code can be annotated with @BeforeClass
5. We can ignore some test cases using @Ignore annotation.
6. Way of testing a test case for exception has also changed in JUnit4.

In JUnit3 we fail a test case deliberately if exception is not thrown, as shown below.

|  |
| --- |
| **public** **void** testSomeException() {     SomeClass c = **new** SomeClass();  **try** {         c.meth();         fail("Exception should have been thrown, but was not");     } **catch** (SomeException expected) {         assertTrue(**true**);     }  } |

In JUnit4 we use a expected in @Test annotation to denote that method is supposed to throw an exception of th type provided

|  |
| --- |
| @Test(expected=SomeException.**class**)  **public** **void** testSomeException() {     SomeClass c = **new** SomeClass ();     c.meth();  } |

7. In JUnit4 we have an additional parameter timeout which denote an upper time limit on execution of test. If this time limit exceeds, the test is marked as failed and n exception is raised. The code snapshot is shown below:

|  |
| --- |
| @Test(timeout=50)  **public** **void** testSomeMethodSupposedToBeFinishedIn50MilliSeconds() {     [...]  } |