# Junit

## 一、Junit基本概念

Junit是一个java语言的单元测试框架，是java的测试标准。

验证java中一些工具类的方法的正确性：

①法一：创建一个main方法，创建工具类的实例并调用要测试的方法，查看控制台的输出结果验证方法的正确性。这种方法在实际开发中，方法比较多的时候，这样的操作费时费力。

②法二：采用**Junit可以直接测试方法的执行结果**，能够高效地测试类中的方法。

## 二、Junit的使用

1.若在Eclipse中使用，可以使用Eclipse自带的Junit。右击项目-->Build Path-->Add Library-->JUnit，然后选择对应的版本即可。也可以右击项目-->Properties-->页面左边Java Build Path-->Libraries-->Add Library-->JUnit，然后选择版本。

2.若在其它地方使用，需要网上下载JUnit的jar包然后导入。

## 三、Junit3(次要)

### **1.编写测试类**：

在Junit3中，自己编写一个类的测试类**，该测试类必须继承TestCase类**，TestCase类是在Junit.framework包中的。同时**测试类中的测试方法必须以“test”为前缀，即以testXxx命名，且被void修饰，没有返回值，并且没有任何参数**。Junit3是通过分析方法名称来识别测试方法的，不遵循这个命名的测试方法将被Junit框架忽略，且不抛出任何异常，而是指示发生了错误。

**测试类最好用自动生成的，而不是自己写。自动生成的操作如下：**在一个工具类上，右键点击该类-->new-->JUnit Test Case(若找不到的话在Other中找)-->在弹出的JUnit Test Case窗口中勾选setUp和tearDown方法-->点击next(不是直接finish)-->勾选要测试的方法-->这时再点击finish。

**setUp()**方法：该方法会在每个测试方法执行前被调用以完成初始化的工作。**一般创建工具类的对象放在这里。**

**tearDown()**方法：该方法会在每个测试方法执行完之后被调用，用来释放测试中使用的资源。**一般工具类对象的销毁或指向null放在这里。**

**Assert.assertEquals(expectedValue，actualValue)**：参数1为方法结果的期望值，参数2为方法结果的实际值。

**void testXxx()**：**测试方法的写法**。

### 2.运行测试类：

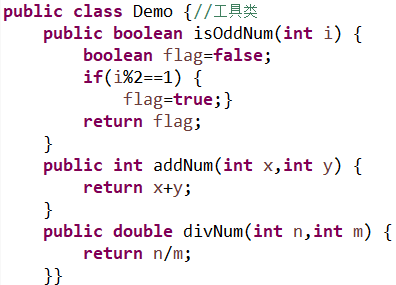
测试类没有明确的命名规定，但是最好命名为XxxTest。执行测试类不需要main方法，也不需要创建测试类的对象。

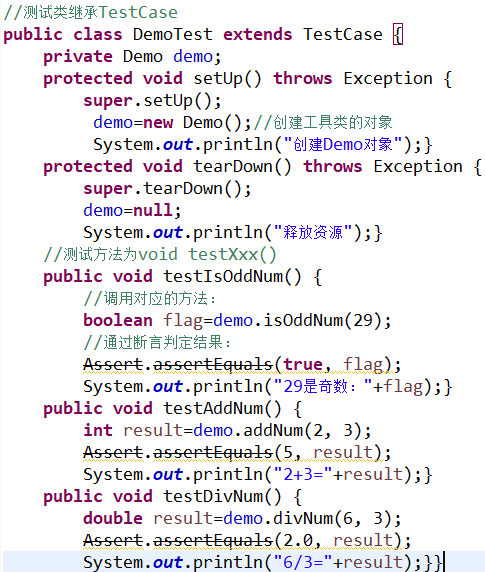


**直接在测试类中右键-->Run As-->JUnit Test即可**。

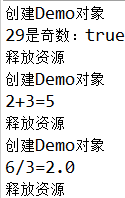
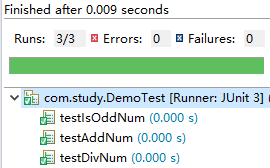
注意setUp()和tearDown()方法不是必须的，若不使用该方法的话，则在每个测试方法中都要创建一个工具类对象用来调用对应的方法。

**例子：**





上面代码在控制台上输出的结果和Eclipse左侧显示的结果：

## 四、Junit4(主要)

在写好工具类后，最好也是采用自动生成的方式生成Junit4的测试类，也可以自己写测试类。Junit4导入的包是org.junit中的包。

**注意在Junit4中是通过注解的方式@Test标记测试方法的，测试类也不需要继承TestCase，测试方法也不需要以“test”作为前缀，测试方法名称可以随便写，但是测试方法也必须无返回值，被void修饰，同时无参数**。Junit4中直接用**assertEquals()**方法判断结果，然后导入**import** **static** org.junit.Assert.*assertEquals*。不要再使用**Assert.assertEquals()**方法判断结果，该方法已经被弃用了，且该方法是Junit3中的方法，导入的是**import** junit.framework.Assert。

### 1.Junit4中的注解：

**@BeforeClass**：类的数据的初始化，即**用来测试被public static修饰的方法、数据**。**该注解注释的方法在整个过程中只会执行一次，且是第一个运行的**。

**@AfterClass**：类的数据的销毁。**该注解注释的方法在整个过程中也只会执行一次，且是最后一个运行的**。**方法必须是被public static修饰**。

在自己写测试类时，**需要注意每个测试类中只能有一个方法被标注为@BeforeClass或@AfterClass，并且这个方法必须是被public static修饰的**。

**@Before**：会在每个测试方法执行之前调用，用来初始化资源。和Junit3中的setUp()效果一样。**每一个测试方法都应该使用自己的测试对象**，**@Before和setUp()的作用就是初始化该测试对象的**。会执行多次。**方法必须是public 修饰，但不能被static修饰**。

**@After**：会在每个测试方法执行完之后被调用，用来释放测试中使用的资源。@After和Junit3中的tearDown()效果一样，都是用来将测试对象释放的。会执行多次。**方法必须是public 修饰，但不能被static修饰。**

前面四个注解中的代码是Fixture，就是在某些阶段必然被调用的代码。

**@Test**：表示该方法是测试方法。**注意@Test不能加到静态方法上，且测试方法为void不能有返回值，同时不能有参数**。测试单元中的每个方法必须可以独立测试，测试方法之间不能有任何的依赖。

**@Ignore**：表示忽略该测试方法。

在Junit3中，如果想忽略某个测试方法，可以通过注释掉该方法或将测试方法的名字改成不以“test”开头，这样Runner就找不到该方法。而Junit4中则是用@Ignore标注来忽略测试方法。

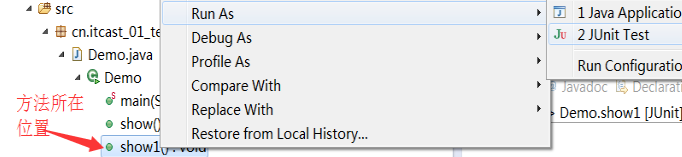
**限时测试@Test(timeout=1000)**：在测试的时候，若遇到方法是死循环时，这时我们可以给该测试方法设定执行时间，超过了设定时间，该方法就会被系统强行终止。只需给@Test加一个参数，如@Test(timeout=1000)，**timeout参数表明了设定的时间，单位为毫秒**。

[**测试异常@Test(expected=XxxException.class)**](mailto:测试异常@Test(expected=XxxException.class))：工具类中若一个方法应该抛出某种异常，但是它并没有抛出，在测试类中我们可以测试该方法是否抛出指定的异常。只需在@Test中加一个expected参数，并将要检测的异常传递给它。这样Junit框架就能帮我们自动检测是否抛出了我们指定的异常。

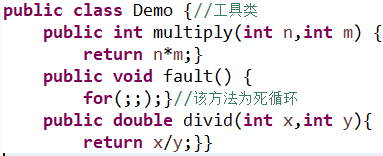
### 2.Junit4中高级用法：

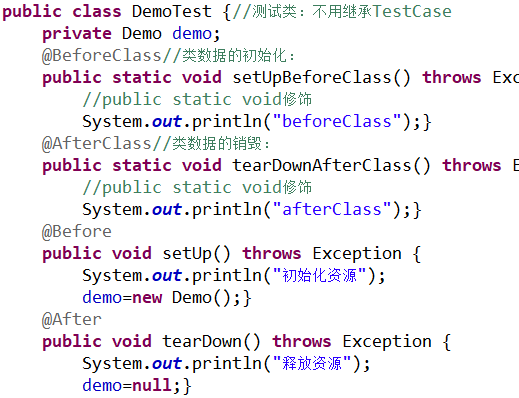
**@RunWith(**TestClass**Runner.class)**：把测试代码提交给JUnit框架之后，JUnit是使用**Runner(运行器)**来调用测试代码的。每一个Runner都有各自特殊的功能，需要根据不同的需求选择不同的Runner来运行测试代码。**JUnit中有一个默认的Runner，若没有指定，系统自动使用默认的Runner来运行测试代码，这个默认的Runner就是**(org.junit.internal.runners.**TestClassRunner**)。要想指定Runner，需要使用标注@RunWith()，并把指定的Runner作为参数传递给它。**注意@RunWith()是用来修饰类的，而不是修饰函数的。只要一个类指定了Runner，则这个类中的所有测试函数都被这个Runner调用**。当类被@RunWith注解修饰，或者类继承了一个被该注解修饰的类，JUnit将会使用这个注解所指明的Runner来运行测试，而不使用JUnit默认的运行器。

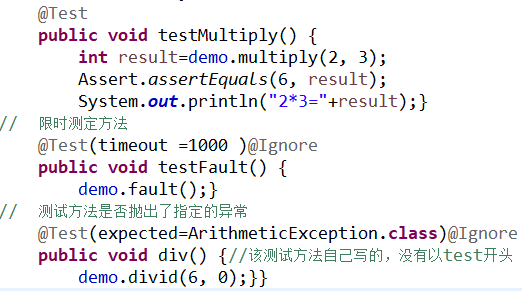
不选择方法而执行Junit，则所有的@Test注解的方法都会被执行一遍(被Ignore注解的方法不会)。可以指定要执行哪个测试方法。在Eclipse的左边，选中测试类中要执行的测试方法-->右键Run As-->Junit Test该方法就会被执行。



**例子：**







**参数测试**：参数化测试适用于测试函数根据不同的传参，有很多类型的结果或者参数有许多特殊值。

**采用参数化测试的步骤**：

**①**专门生成一个新的测试类，不能与其它测试共用同一个类。为该测试类指定特殊的运行器**@RunWith(Parameterized.class)**。

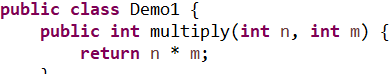
**②**在测试类中，定义变量用来保存测试数据或保存期望值。

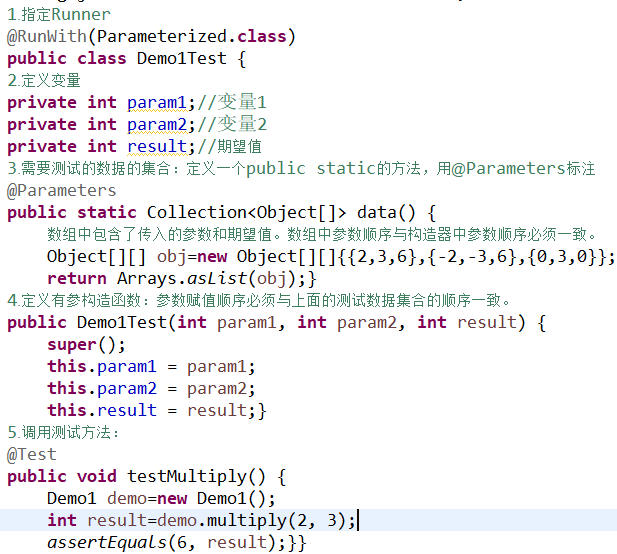
**③**定义一个public static 修饰的方法，方法名随意，返回值类型为Collection类型(不是必须，也可以是其它类型)，该方法要用**@Parameters**标注。该方法中填写实际传入的参数和期望值。**@Parameters是给方法标注的，@Parameter是给属性标注的，属性必须为public类型才可以。**

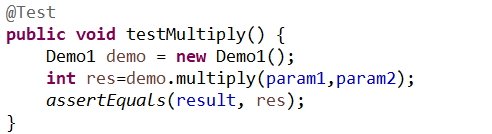
**④**定义类的构造方法，构造方法的参数为②中定义的变量。构造方法中参数赋值的顺序与③中初始化的数据的顺序一致。

**⑤**编写测试方法，用初始化的参数进行测试。

**参数测试的例子(多个参数)：**

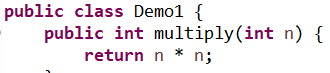


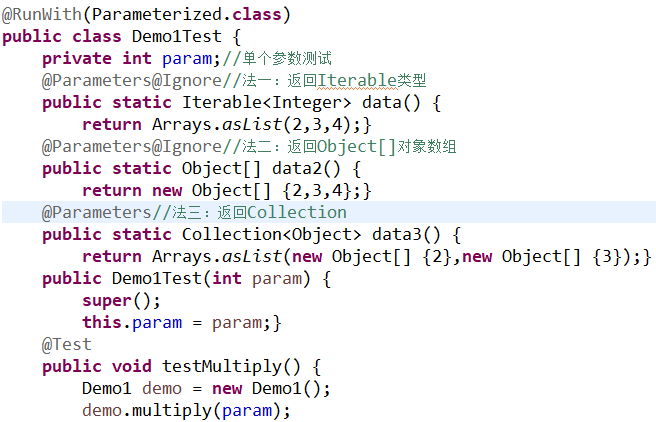




若单个参数进行测试，则不需要将数据包装成集合，可用迭代器或者对象数组替换Collection。

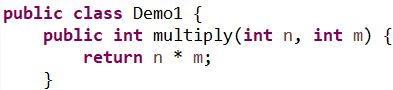
**参数测试的例子(单个参数)**：

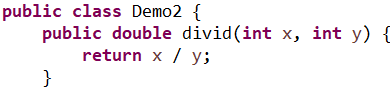


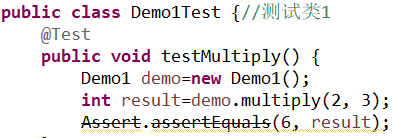


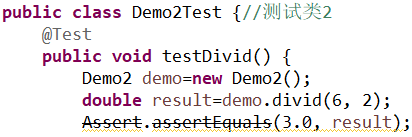
**打包测试**：一个项目中往往会有多个测试类，一个一个执行测试类是比较麻烦。可以将所有需要运行的测试类集中起来一次性的运行完毕。**需要编写一个单独的类，该类中不需要写任何东西。**[**然后将该类用@RunWith()标注，并将Suite.class**](mailto:然后将该类用@RunWith()标注，并将Suite.class)**作为参数传入。**[**然后使用@Suite.SuiteClasses()标注，将要运行的测试类以类名.class**](mailto:然后使用@Suite.SuiteClasses()标注，将要运行的测试类以类名.class)**作为参数传入标注中，用{}将测试类括起来，测试类之间用逗号隔开**。

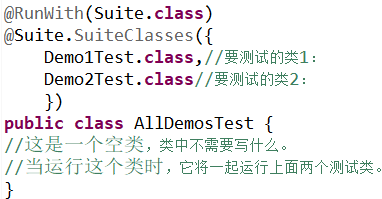
**打包测试的例子**：











，[只要加@Test就是测试方法。junit4不需要继承类。而junit3](mailto:只要加@Test就是测试方法。junit4不需要继承类。而junit3)的测试方法只能在写在测试用例的类里面，不能写在其他地方。而junit4可以在任何地方写测试方法，只要加@Test

多个测试方法是不能共用一个测试对象