# Junit初识

【参考：<https://www.cnblogs.com/fsjohnhuang/p/4061902.html#a2】>

## Junit概述：

### 什么是Junit：

1. Junit是用于编写可复用测试集的简单框架，是xUnit的一个子集。XUnit是一套基于测试驱动开发的测试框架，有PythonUnit、CppUnit、Junit等。
2. Junit测试程序员测试，即所谓的白盒测试，因为程序员知道被测试的软件如何完成功能（How）和完成什么功能（What）。
3. 多数的Java的开发工具都已经集成了Junit作为单元测试的工具，如Eclipse。

### 为什么要使用测试工具：

1. 测试框架可以帮助我们对编写的程序进行有目的的测试，帮助我们最大限度的避免代码中的bug，以保证系统的正确性和稳定性。
2. 很多人对自己写的代码，测试时就简单的写main方法，然后sysout输出到控制台观察结果。这样非常枯燥繁琐，不规范。缺点：测试方法不能一起运行，测试结果要程序员自己观察才可以判断程序逻辑是否正确。
3. Junit的断言机制，可以直接将我们的预期结果和程序运行的结果进行一个对比，确保对结果的可预知性。

## 入门案例：

1. 编写一个简单的计算类：Calculate.java

|  |
| --- |
| **public** **class** Calculate {  /\*\*  \* 两数相加求和  \*/  **public** **int** add(**int** x,**int** y) {  **return** x + y;  }    /\*\*  \* 两数相减  \* **@return**  \*/  **public** **int** subtract(**int** x,**int** y) {  **return** x - y;  }      /\*\*  \* 两数相乘  \* **@return**  \*/  **public** **int** multiply(**int** x,**int** y) {  **return** x \* y;  }    /\*\*  \* 两数相除  \* **@return**  \*/  **public** **int** divide(**int** x,**int** y) {  **return** x / y;  }  } |

1. new 一个测试类，并为这四个方法生成测试方法：

|  |
| --- |
| **public** **class** CalculateTest {  /\*\*  \* 测试两数相加  \*/  @Test  **public** **void** testAdd() {  Calculate calculate = **new** Calculate();  **int** result = calculate.add(2, 5);    *assertEquals*(7, result);  System.***out***.println(1);    *assertEquals*(10, result);  System.***out***.println(2);    /\*\*  \* 输出结果: 1  \*  \* 从输出结果可以知道,当断言为true的时候会继续向下执行,知道方法结束;当断言为false的时候,Junit会中断执行并抛出AssertError  \*/  }  /\*\*  \* 测试两数相减  \*/  @Test  **public** **void** testSubtract() {  Calculate calcuate = **new** Calculate();  **int** result = calcuate.subtract(12, 2);  *assertEquals*("减法有问题", 10000, result); //故意设置减法期望值为10000  }  /\*\*  \* 测试两数相乘  \*/  @Test  **public** **void** testMultiply() {  Calculate calcuate = **new** Calculate();  **int** result = calcuate.multiply(2, 0);  *assertEquals*("乘法有问题", 0, result);  }  /\*\*  \* 测试两数相除  \*/  @Test  **public** **void** testDivide() {  Calculate calcuate = **new** Calculate();  **int** result = calcuate.divide(6, 0);  System.***out***.println(1);  *assertEquals*("除法有问题", 6, result);    /\*\*  \* 执行 calculate.divide(6,0)的时候抛出了异常,程序中断执行,所以下面的代码并没有执行,因此Junit执行的结果才会出现 Errors:1 Failures:0  \*/  }    /\*\*  \* 从这几个测试案例中可以知道：  \* 1. 如果程序本身抛出了异常,那么就不会执行Junit的断言(参考testDevide()方法):  \* 1)程序会在该代码处中断执行,并抛出异常,那么之后的代码都不会执行(不考虑捕获的情况),那么JUnit的Assert根本就没有执行  \* 2)正是因为程序本身抛出了异常,所以Junit的测试结果是 Errors:1 Failures:0  \* 2. 如果程序本身没有抛出异常,那么会继续执行JUnit断言:  \* 1)如果Junit的断言通过,代码会向下继续执行,直到方法结束;  \* 2)如果JUnit的断言不通过,代码会在此处中断,并排除Junit自定义的异常,所以JUnit的执行结果是 Errors:0 Failures:1 (参考testAdd()方法)  \*/  } |

1. 结论：

从这几个测试案例中可以知道：

1. 如果程序本身抛出了异常,那么就不会执行Junit的断言(参考testDevide()方法):

1)程序会在该代码处中断执行,并抛出异常,那么之后的代码都不会执行(不考虑捕获的情况),那么JUnit的Assert根本就没有执行

2)正是因为程序本身抛出了异常,所以Junit的测试结果是 Errors:1 Failures:0

2. 如果程序本身没有抛出异常,那么会继续执行JUnit断言:

1)如果Junit的断言通过,代码会向下继续执行,直到方法结束;

2)如果JUnit的断言不通过,代码会在此处中断,并排除Junit自定义的异常,所以JUnit的执行结果是 Errors:0 Failures:1 (参考testAdd()方法)

4、源码：

|  |
| --- |
|  |

【JUnitDemo01/src/chenming/basic/Calculate.java】

【JUnitDemo01/test/chenming/basic/CalculateTest.java】

## Junit规范：

1. 测试方法上必须使用@Test进行修饰
2. 测试方法必须使用public void进行修饰，不能带任何参数
3. 新建一个源代码目录来存放我们的测试代码，也就是测试代码和项目业务代码分开。
4. 测试类所在的包名应该和被测试类所在的包名保持一致
5. 测试单元中的每个测试方法都可以独立测试，测试方法之间不能有任何的依赖
6. 测试类使用Test作为类名的后缀（不是必须）
7. 测试方法使用test作为方法名的前缀（不是必须）

## 测试失败的两种情况：

**【注意】**测试用例是用来达到测试想要的预期结果，而不能测试出程序的逻辑错误。

### Error：

Error是由代码异常引起的，它可以产生于测试代码本身的错误，也可以是被测试代码中的一个隐藏的bug，这种测试情况一般与Junit无关。

下面是产生Error的常见情况：

|  |
| --- |
| /\*\*  \* 下面是测试产生Error的常见情况(产生Error的情况一般与JUnit无关)：  \* 1.被测试的代码有bug  \* 2.测试的代码有bug  \*/  @Test  **public** **void** testError\_1() {  //1.被测试的代码有bug  Calculate calcuate = **new** Calculate();  **int** result = calcuate.divide(6, 0);  *assertEquals*("除法有问题", 6, result);  /\*\*  \* 测试结果产生了Error,这是因为在执行Calculate对象的divide()方法的时候,产生了异常,所以Junit测试结果为Error  \*/  }  @Test  **public** **void** testError\_2() {  //2.测试的代码有bug  **int** temp = 5 / 0;    Calculate calcuate = **new** Calculate();  **int** result = calcuate.divide(6, 0);  *assertEquals*("除法有问题", 6, result);  /\*\*  \* 测试结果产生了Error,这是因为在执行 "int temp = 5 / 0;" 这一行代码的时候产生了异常,所以Junit测试结果为Error  \*/  } |

### Failure：

Failure一般是由单元测试使用的断言方法判断失败引起的，这说明预期结果与实际结果不相符。

下面是产生Failure的常见情况：

|  |
| --- |
| /\*\*  \* 下面是测试产生Failure的常见情况(产生Failure的情况一般就是因为测试结果与我们的期望不一致)  \*/  @Test  **public** **void** testFailure() {  Calculate calcuate = **new** Calculate();  **int** result = calcuate.add(6, 8);  *assertEquals*("加法有问题", 20, result);  /\*\*  \* 测试结果产生Failure,这是因为Junit断言的时候发现期望结果(20)与实际结果(14)不一致,所以Junit测试结果为Failure  \*/  } |

【源码】

|  |
| --- |
|  |

【JUnitDemo01/src/chenming/basic/Calculate.java】

【JUnitDemo01/test/chenming/basic/CalculateTest.java】