# 测试工具JUnit学习报告



姓名：何铭春

学号：E21514033

## 简介

JUnit是用于编写和运行可重复的自动化测试的开源测试框架， 这样可以保证我们的代码按预期工作。JUnit可广泛用于工业和作为支架(从命令行)或IDE(如Eclipse)内单独的Java程序。

## 使用方法

2.1 最简单使用

通过@Test注解来使得一个方法作为测试方法，能够运行。

例如：

@Test

public void testSum() {

System.out.println("@Test sum(): " + sum + " = " + testSum);

assertEquals(sum, testSum);

}

2.2 多种注解能够在合适时机运行函数

@Test

@Before

该方法必须在类中的每个测试之前执行，以便执行测试某些必要的先决条件。

@BeforeClass

类的所有测试之前，执行。

@After

在执行每项测试后执行

@AfterClass

类中所有测试结束后执行

@Ignore

被注解为@Ignore的方法将不被执行。

2.3 多种断言用于判断正误

oid assertEquals([String message], expected value, actual value) 断言两个值相等。

void assertTrue([String message], boolean condition) 断言一个条件为真

void assertFalse([String message],boolean condition) 断言一个条件为假

void assertNotNull([String message], java.lang.Object object) 断言一个对象不为空(null)

void assertNull([String message], java.lang.Object object) 断言一个对象为空(null)

2.4 参数化测试

通过构造器和参数函数来多次运行测试

### 测试实例

3.1 被测试类：用于实现两整数乘

public class Calculate {

public int sum(int var1,int var2){

System.out.println("相乘的值是"+var1+" \* "+var2);

return var1 \* var2;

}

}

3.2 测试类

@RunWith(Parameterized.class)

public class CalculateTest {

private int expected;

private int first;

private int second;

public CalculateTest(int expectedResult, int firstNumber, int secondNumber) {

this.expected = expectedResult;

this.first = firstNumber;

this.second = secondNumber;

}

@Parameters

public static Collection addedNumbers() {

return Arrays.asList(new Integer[][] { { 2, 1, 2 }, { 6, 2, 3 },

{ 12, 3, 4 }, { 20, 4, 5 }, });

}

@Test

public void sum() {

Calculate add = new Calculate();

System.out.println("Addition with parameters : " + first + " mul "

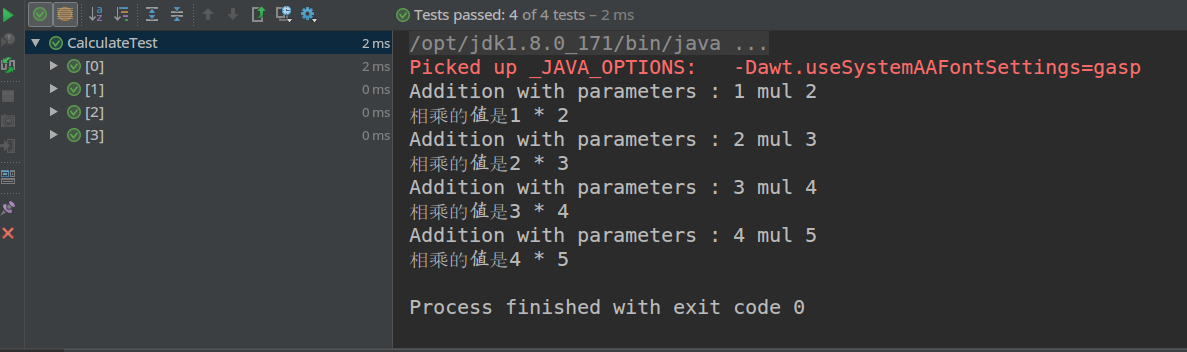
+ second);

assertEquals(expected, add.sum(first, second));

}

}

3.3 运行结果



### 小结和讨论

4.1 使用JUnit能够实现测试驱动编程，先设计好测试用例，然后不断测试和编程相结合，能够生产出高质量的代码。

4.2 通过注释、断言、参数化、规则，我们能够定制测试用例。并且一次编写多次运行。