黑盒&白盒测试

小组名：致富经

组长：蔡佳泉 1711216

组员：阙定 1757127

杨阳 1859105

蔡心慧 1859209

王润炜 1859205

目录

1.实验目的

- 掌握单元测试软件 JUnit 的使用
- 掌握黑盒测试及白盒测试基本技术
- 实验内容及要求
- 编写测试用例

2.实验内容

- 对 Calculator 类，使用黑盒测试【等价划分法】
- 对 Triangle 类，先画出程序流程图，再使用白盒测试【条件覆盖】方法进行测试

3.实验步骤

1.1.对 Calculator 类，使用黑盒测试

1.1.1.原代码

Calculator.java

```
1. package com.ybq.calculator;
2.
3. public class Calculator {
4.     private static int result = 0;
5.
```

```
6.      public int add(int n) {
7.          result = result + n;
8.          return result;
9.      }
10.
11.     public int substract(int n) {
12.         result = result - n;
13.         return result;
14.     }
15.
16.     public int multiply(int n) {
17.         result = result * n;
18.         return result;
19.     }
20.
21.     public int divide(int n) {
22.         result = result / n;
23.         return result;
24.     }
25.
26.     public int square(int n) {
27.         result = n * n;
28.         return result;
29.     }
30.
31.     public int squareRoot(int n) {
32.         for (; ; ) ;
33.     }
34.}
```

1.1.2.等价类划分

输入等价类	有效等价类	无效等价类
一个整数	在-2^31 到 2^31 之间的整数 小数点后为 0 的浮点数	小于-2^31 的整数 大于 2^31 的整数 （即小于-2147483648 或大于 2147483647） 小数点后不为 0 的浮点数 含有非数字字符

1.1.3.测试用例

输入条件	测试用例说明	测试数据	期望结果	选取理由
一个整数	小于-2^31 的整数	-2147483649	显示出错	比有效整数小 1
	大于 2^31 的整数	2147483649	显示出错	比有效整数大 1
	小数点后为零的浮点数	1.0	输入有效	小数点后为 0
	小数点后不为零的浮点数	1.1	显示出错	小数点后不为 0
	有一个是非数字字符	1a	显示出错	只有一个非法字符
	全部是非数字字符	+vx	显示出错	三个非法字符
	在-2^31 到 2^31 之间的整数	5	输入有效	类型和范围均有效

测试模块	测试内容	输入数据	预期输出结果	运行输出结果	测试结果
Add 方法	“n”字段数据类型不匹配	-2147483649	提示输入字段超过可接受范围	提示输入字段超过可接受范围	通过
Add 方法	“n”字段数据类型不匹配	2147483649	提示输入字段超过可接受范围	提示输入字段超过可接受范围	通过
Add 方法	“n”字段数据类型不匹配	1.1	提示输入字段应为纯整数，类型不正确	提示输入字段应为纯整数，类型不正确	通过
Add 方法	“n”字段数据类型不匹配	1a	提示输入字段应为纯整数，类型不正确	提示输入字段应为纯整数，类型不正确	通过
Add 方法	“n”字段数据类型不匹配	+vx	提示输入字段应为纯整数，类型不正确	提示输入字段应为纯整数，类型不正确	通过
Add 方法	“n”字段数据类型匹配	1.0	输入成功	输入成功	通过
Add 方法	“n”字段数据类型匹配	5	输入成功	输入成功	通过
Substract 方法	“n”字段数据类型不匹配	-2147483649	提示输入字段超过可接受范围	提示输入字段超过可接受范围	通过
Substract 方法	“n”字段数据类型不匹配	2147483649	提示输入字段超过可接受范围	提示输入字段超过可接受范围	通过
Substract 方法	“n”字段数据类型不匹配	1.1	提示输入字段应为纯整数，类型不正确	提示输入字段应为纯整数，类型不正确	通过

Substract 方法	“n”字段数据类型不匹配	1a	提示输入字段应为纯整数,类型不正确	提示输入字段应为纯整数，类型不正确	通过
Substract 方法	“n”字段数据类型不匹配	+vx	提示输入字段应为纯整数,类型不正确	提示输入字段应为纯整数，类型不正确	通过
Substract 方法	“n”字段数据类型匹配	1.0	输入成功	输入成功	通过
Substract 方法	“n”字段数据类型匹配	5	输入成功	输入成功	通过
Multiply 方法	“n”字段数据类型不匹配	-2147483649	提示输入字段超过可接受范围	提示输入字段超过可接受范围	通过
Multiply 方法	“n”字段数据类型不匹配	2147483649	提示输入字段超过可接受范围	提示输入字段超过可接受范围	通过
Multiply 方法	“n”字段数据类型不匹配	1.1	提示输入字段应为纯整数,类型不正确	提示输入字段应为纯整数，类型不正确	通过
Multiply 方法	“n”字段数据类型不匹配	1a	提示输入字段应为纯整数,类型不正确	提示输入字段应为纯整数，类型不正确	通过
Multiply 方法	“n”字段数据类型不匹配	+vx	提示输入字段应为纯整数,类型不正确	提示输入字段应为纯整数，类型不正确	通过
Multiply 方法	“n”字段数据类型匹配	1.0	输入成功	输入成功	通过
Multiply 方法	“n”字段数据类型匹配	5	输入成功	输入成功	通过
Divide 方法	“n”字段数据类型不匹配	-2147483649	提示输入字段超过可接受范围	提示输入字段超过可接受范围	通过
Divide 方法	“n”字段数据类型不匹配	2147483649	提示输入字段超过可接受范围	提示输入字段超过可接受范围	通过
Divide 方法	“n”字段数据类型不匹配	1.1	提示输入字段应为纯整数,类型不正确	提示输入字段应为纯整数，类型不正确	通过
Divide 方法	“n”字段数据类型不匹配	1a	提示输入字段应为纯整数,类型不正确	提示输入字段应为纯整数，类型不正确	通过
Divide 方法	“n”字段数据类型不匹配	+vx	提示输入字段应为纯整数,类型不正确	提示输入字段应为纯整数，类型不正确	通过

Divide 方法	“n”字段数据类型匹配	1.0	输入成功	输入成功	通过
Divide 方法	“n”字段数据类型匹配	5	输入成功	输入成功	通过
Square 方法	“n”字段数据类型不匹配	-2147483649	提示输入字段超过可接受范围	提示输入字段超过可接受范围	通过
Square 方法	“n”字段数据类型不匹配	2147483649	提示输入字段超过可接受范围	提示输入字段超过可接受范围	通过
Square 方法	“n”字段数据类型不匹配	1.1	提示输入字段应为纯整数，类型不正确	提示输入字段应为纯整数，类型不正确	通过
Square 方法	“n”字段数据类型不匹配	1a	提示输入字段应为纯整数，类型不正确	提示输入字段应为纯整数，类型不正确	通过
Square 方法	“n”字段数据类型不匹配	+vx	提示输入字段应为纯整数，类型不正确	提示输入字段应为纯整数，类型不正确	通过
Square 方法	“n”字段数据类型匹配	1.0	输入成功	输入成功	通过
Square 方法	“n”字段数据类型匹配	5	输入成功	输入成功	通过

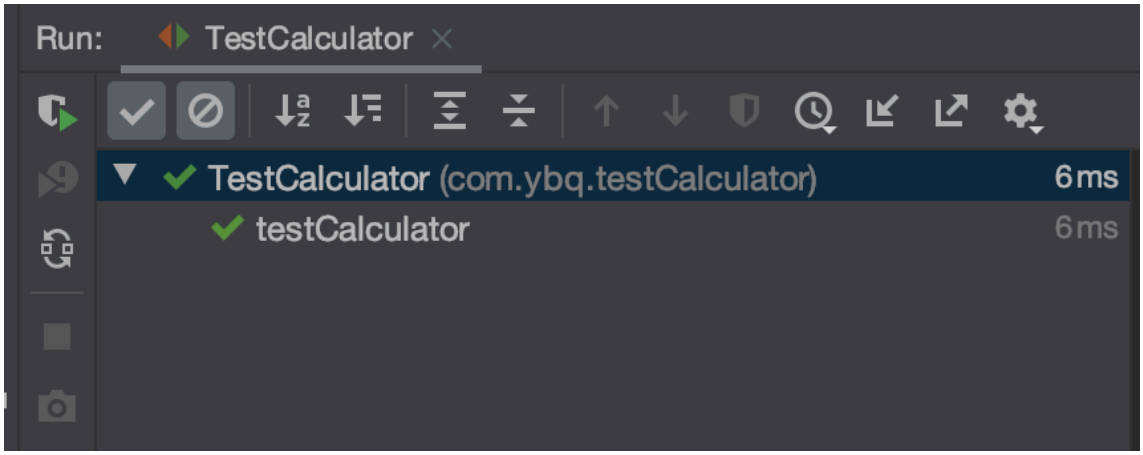
1.1.4. Junit 测试

TestCalculator.java

```
1. package com.ybq.testCalculator;
2.
3. import com.ybq.calculator.Calculator;
4. import org.junit.Test;
5.
6. import static org.junit.Assert.assertEquals;
7.
8. public class TestCalculator {
9.
10.     @Test
11.     public void testCalculator() {
12.         Calculator calculator = new Calculator();
13.         int sum = calculator.add(100);
```

```
14.         assertEquals(sum, 100);
15.
16.         sum = calculator.substract(50);
17.         assertEquals(sum, 50);
18.
19.         sum = calculator.multiply(2);
20.         assertEquals(sum,100);
21.
22.         sum = calculator.divide(2);
23.         assertEquals(sum,50);
24.
25.         sum = calculator.square(2);
26.         assertEquals(sum,4);
27.     }
28. }
```

结果



1.2.对 Triangle 类，先画出程序流程图，再使用白盒测试

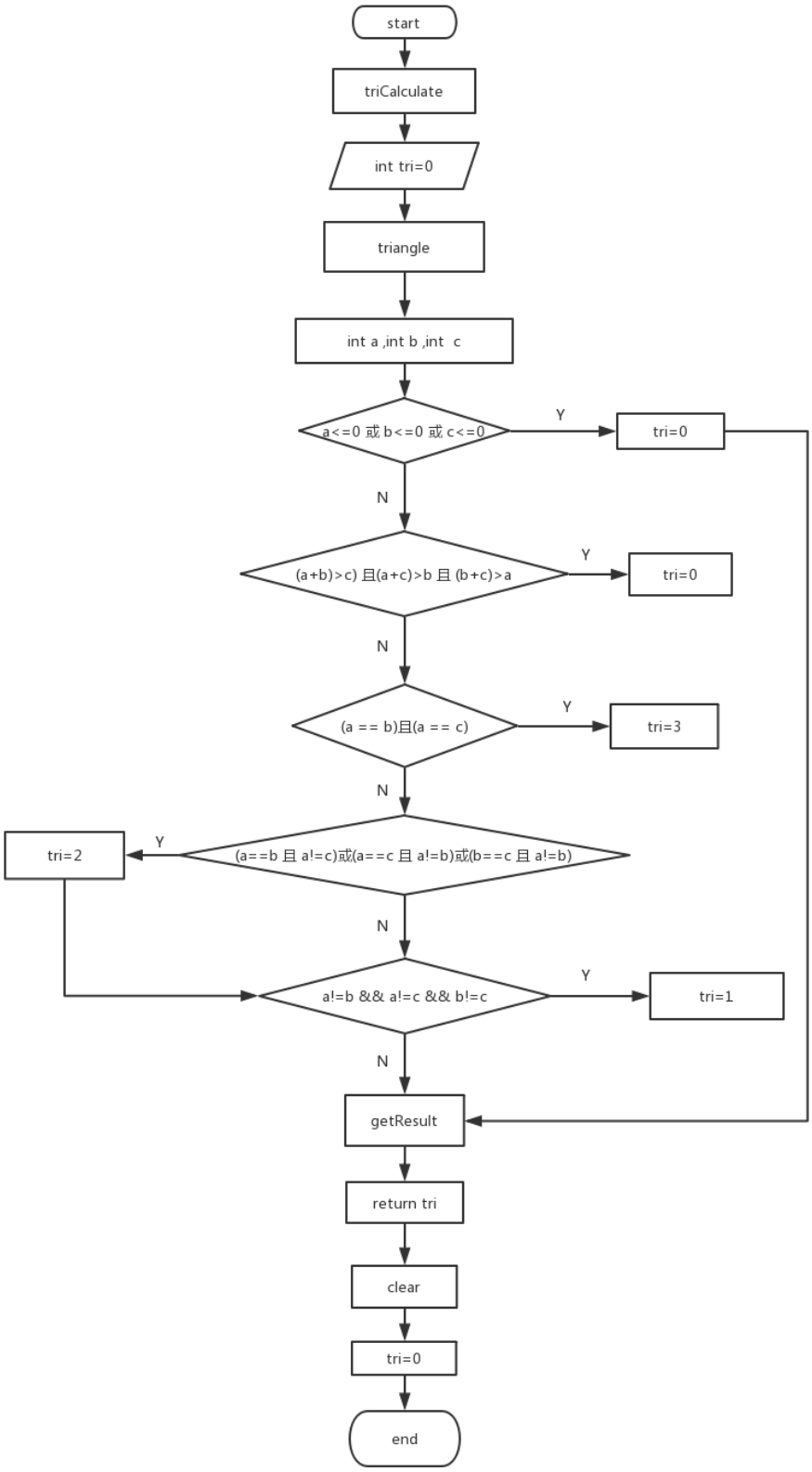
1.2.1.原代码

TriCalculate.java

```
1. package com.ybq.triangle;
2.
3. public class TriCalculate {
4.     private static int tri = 0;
```

```
5.
6.      public void triangle(int a, int b, int c) {
7.          //有一条边为非正数,不是三角形
8.          if (a <= 0 || b <= 0 || c <= 0) {
9.              tri = 0;
10.         } else {
11.             //三边任意两边大于第三边,是三角形
12.             if (((a + b) > c) && ((a + c) > b && (b + c) > a)) {

13.                 if ((a == b) && (a == c)) {
14.                     tri = 3;//等边三角形
15.                 }
16.                 if ((a == b && a != c) || (a == c && a != b) || (
    b == c && a != b)) {
17.                     tri = 2;//等腰三角形
18.                 }
19.                 if (a != b && a != c && b != c) {
20.                     tri = 1;//一般三角形
21.                 }
22.             } else {
23.                 tri = 0;//三边长度无法构成三角形
24.             }
25.         }
26.     }
27.
28.     public int getReuslt() {
29.         return tri;
30.     }
31.
32.     public void clear() {
33.         tri = 0;
34.     }
35. }
```

1.2.2.程序流程图


start
triCalculate
int tri=0
triangle
int a ,int b ,int c
a<=0 或 b<=0 或 c<=0
Y
tri=0
N
(a+b)>c) 且(a+c)>b 且 (b+c)>a
Y
tri=0
N
(a == b)且(a == c)
Y
tri=3
N
tri=2
Y
(a==b 且 a!=c)或(a==c 且 a!=b)或(b==c 且 a!=b)
N
a!=b && a!=c && b!=c
Y
tri=1
N
getResult
return tri
clear
tri=0
end

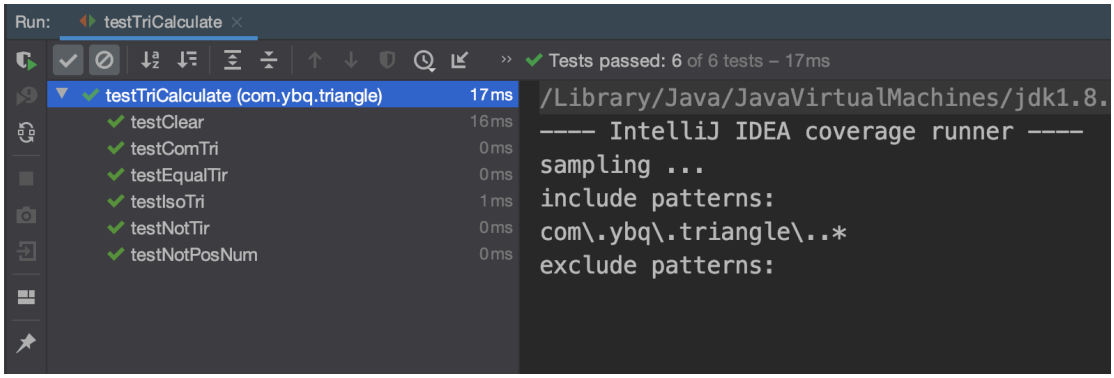
1.2.3.Junit 测试

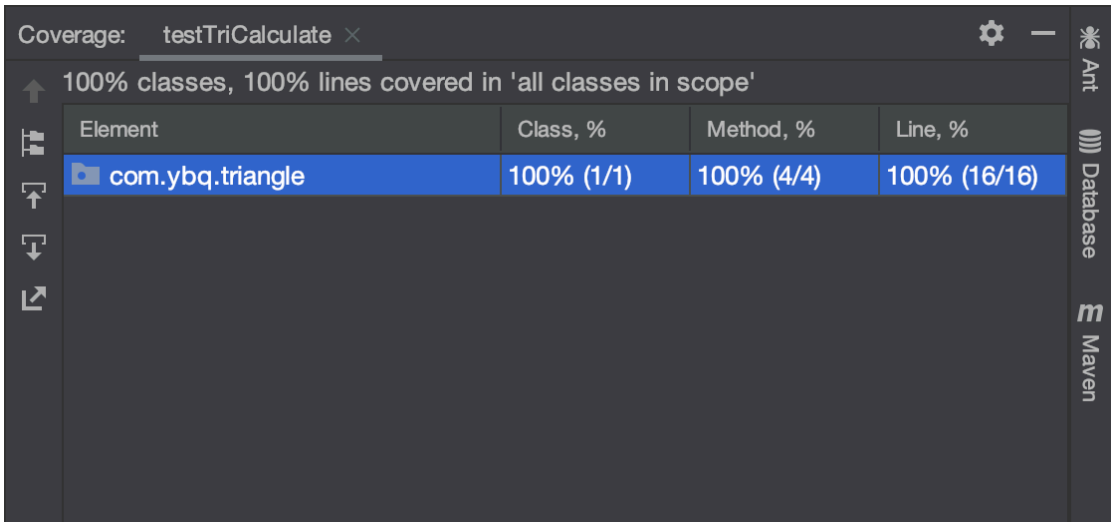
TestTriCalculate.java

```java
package com.ybq.testTriangle;

import com.ybq.triangle.TriCalculate;
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class TestTriCalculate {
    /**
     * 测试 clear()
     */
    @Test
    public void testClear() {
        TriCalculate triCalculate = new TriCalculate();
        triCalculate.triangle(1,1,1);
        assertEquals(triCalculate.getReuslt(), 3);
        triCalculate.clear();
        assertEquals(triCalculate.getReuslt(), 0);
    }

    /**
     * 测试等边三角形
     */
    @Test
    public void testEqualTir() {
        TriCalculate triCalculate = new TriCalculate();
        triCalculate.triangle(1,1,1);
        assertEquals(triCalculate.getReuslt(), 3);
    }

    /**
     * 测试等腰三角形
     */
    @Test
    public void testIsoTri() {
        TriCalculate triCalculate = new TriCalculate();
        triCalculate.triangle(1,2,2);
        assertEquals(triCalculate.getReuslt(), 2);
    }
```

```
40.
41.     /**
42.      * 测试等腰三角形
43.      */
44.     @Test
45.     public void testComTri() {
46.         TriCalculate triCalculate = new TriCalculate();
47.         triCalculate.triangle(3,4,5);
48.         assertEquals(triCalculate.getReuslt(), 1);
49.     }
50.
51.     /**
52.      * 测试边是非正数
53.      */
54.     @Test
55.     public void testNotPosNum() {
56.         TriCalculate triCalculate = new TriCalculate();
57.         triCalculate.triangle(-1,-2,-3);
58.         assertEquals(triCalculate.getReuslt(), 0);
59.     }
60.
61.     /**
62.      * 测试三边任意两边不大于第三边
63.      */
64.     @Test
65.     public void testNotTir() {
66.         TriCalculate triCalculate = new TriCalculate();
67.         triCalculate.triangle(1,2,3);
68.         assertEquals(triCalculate.getReuslt(), 0);
69.     }
70. }
```

结果





4.成员分工

学号	姓名	任务分工
1711216	蔡佳泉	Junit 测试
1859105	杨阳	等价类划分&测试用例
1859205	王润炜	Junit 测试
1859209	蔡心慧	程序流程图
1757127	阙定	文档排版