**软件测试实验课程**

**实 验 报 告**

学 院： 计算机科学与工程学院

班 级： 软件工程2018-1班

完 成 人：姓 名： KAFLE SAMRAT 学 号： 201801060933

指导教师：赵卫东

**山 东 科 技 大 学**

2020年11月

实验1 JUnit的安装与基本操作

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 组 别 |  | 实验人员 | KAFLE SAMRAT | | | | |
| 实验项目名称 | JUnit的安装与基本操作 | | | | | | |
| 实 验 类 别 | 验证型□ 设计型□ 综合型□ | | | | | | |
| 实 验 工 具 |  | | | 实验地点 | J3-132 | 实验日期 | 2020.12.7 |
| 学生实验小结 | Junit不需对源程序进行任何修改，减少风险，一次可以执行多个测试用例便于进行白盒测试，总之只需要写入参数就可以评测程序是否有错误，写覆盖测试时有些晕，不过在分析完程序流程图后问题得到了解决 | | | | | | |
| 教师评语 | 指导教师签名：  年 月 日 | | | | | | |
| 实验成绩 |  | | | | | | |

1实验内容和目标

1.1实验内容

本实验主要学习在开发工具中使用Junit4对程序进行单元测试的基本步骤和操作方法。多种开发环境都可以使用Junit进行测试，本次实验是在eclipse环境中编写一段简易计算器程序，使用Junit框架对该程序的各个方法进行测试，以熟悉Junit4的各种注解和断言

1.2实验目标

（1）掌握安装JUnit的方法

（2）掌握使用JUnit进行单元测试的基本步骤和操作方法

2实验方案设计

2.1实验环境

Eclipse 、Junit、Windows

2.2实验方法设计

2.2.1Triangle类的核心代码

**package** helloWorld;

**public** **class** Triangle {

**public** **int** istriangle(**int** a,**int** b,**int** c) {

**if**((a+b>c)&&(a+c>b)&&(b+c>a))

**return** 1;

**else**

**return** 0;

}

**public** **int** getType(**int** a,**int** b,**int** c) {

**if**(a == b && b == c)//equilateral

**return** 1;

**else** **if**(a == b || b == c || a == c)//isosceles

{

**if**((a\*a + b\*b == c\*c)||(a\*a+c\*c == b\*b)||(b\*b+c\*c == a\*a))

**return** 3;//Isosceles right-angled

**else**

**return** 2;//Ordinary isosceles

}

**else** **if**((a\*a + b\*b == c\*c)||(a\*a+c\*c == b\*b)||(b\*b+c\*c == a\*a))//Right Angle

**return** 4;//Ordinary right angle

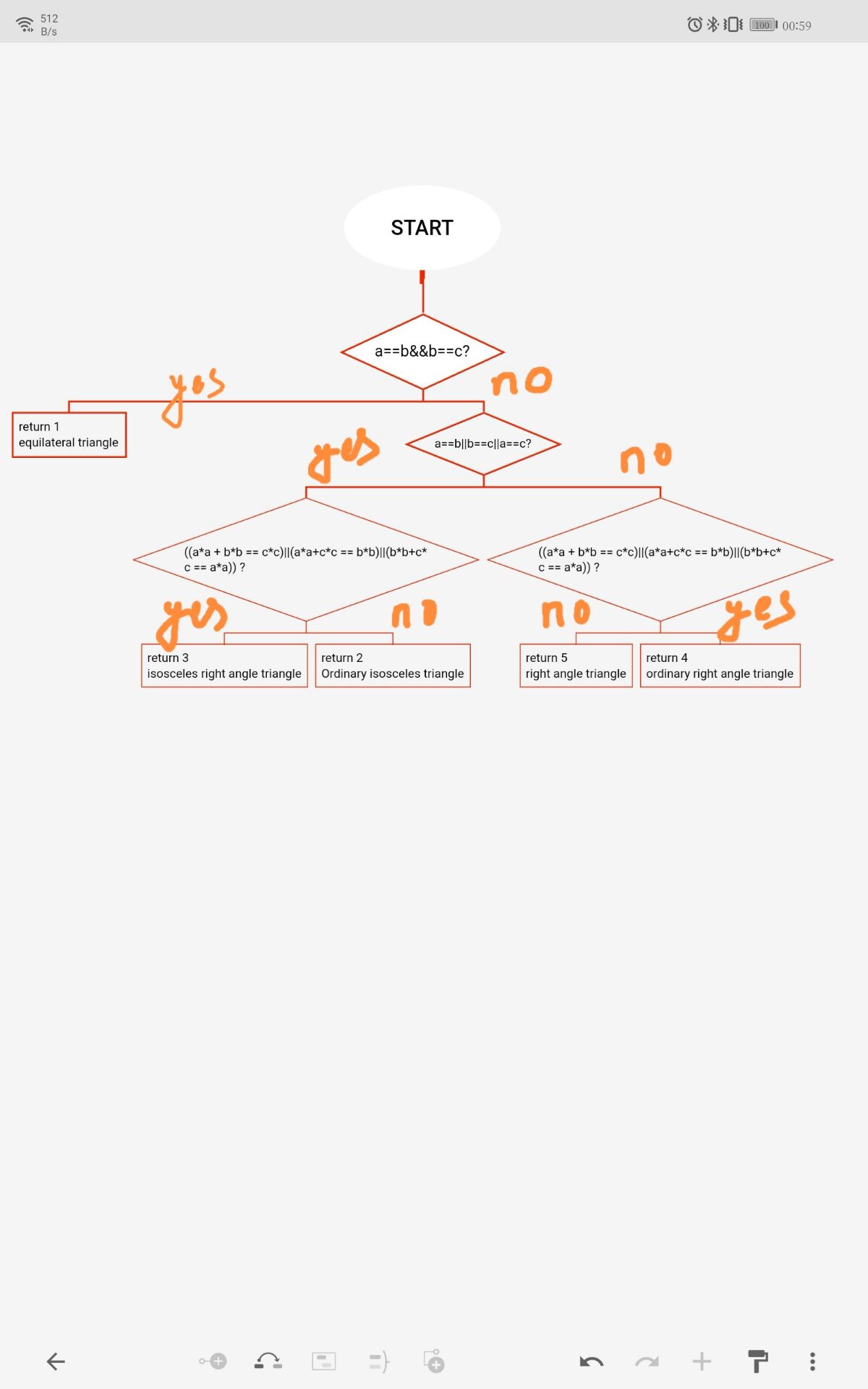
**else**

**return** 5;

}

}

2.2.2Triangle类的程序流程图



2.2.3Triangle Test测试类的核心代码

**package** helloWorld;

**import** **static** org.junit.Assert.~~assertThat~~;

**import** **static** org.junit.Assert.*assertTrue*;

**import** **static** org.junit.jupiter.api.Assertions.\*;

**import** org.junit.jupiter.api.AfterAll;

**import** org.junit.jupiter.api.AfterEach;

**import** org.junit.jupiter.api.BeforeAll;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Test;

**class** TriangleTest {

**private** Triangle t;

@BeforeAll

**static** **void** setUpBeforeClass() **throws** Exception {

}

@AfterAll

**static** **void** tearDownAfterClass() **throws** Exception {

}

@BeforeEach

**void** setUp() **throws** Exception {

t = **new** Triangle();

}

@AfterEach

**void** tearDown() **throws** Exception {

t = **null**;

}

// @Test

// void test() {

// fail("Not yet implemented");

// }

**public** **void** testistriangle()

{

*assertEquals*(1,t.istriangle(2, 3, 4));

*assertEquals*(1,t.istriangle(6, 6, 6));

}

@Test

**public** **void** testGetType() {

*assertEquals*(1,t.getType(3, 3, 3));

*assertSame*(2,t.getType(4, 4, 6));

*assertTrue*("true",4 == t.getType(3, 4, 5));

}

}

2.2.4测试用例

(1) Statement overwrite: Make every statement in the program can be executed at least once

|  |  |  |
| --- | --- | --- |
| S.No | Test case | Results |
| 1 | (2,3,4) | An Ordinary Triangle |
| 2 | (6,6,6) | An Equilateral Triangle |
| 3 | (1,3,9) | Not Triangle |

(2) Determine coverage: each branch of the program passes at least once

|  |  |  |
| --- | --- | --- |
| S.No | Test case | Results |
| 1 | (6,6,6) | An Equilateral Triangle |
| 2 | (2,3,4) | An Ordinary Triangle |
| 3 | (3,4,5) | Right angle triangle |
| 4 | (3,3,4) | An Ordinary isosceles triangle |
| 5 | (2,2,2√2) | An isosceles right angle triangle |

(3) Conditional coverage: each condition in the determination can obtain various possible results

|  |  |  |
| --- | --- | --- |
| S.No | Test case | Results |
| 1 | (3,4,5) | Right angle triangle |
| 2 | (6,6,6) | An Equilateral Triangle |

(4) Judgment condition coverage: each condition in the judgment gets a variety of possible values, and makes each judgment get a variety of possible results

|  |  |  |
| --- | --- | --- |
| S.No | Test case | Results |
| 1 | (2,2,2√2) | An isosceles right angle triangle |
| 2 | (3,4,5) | Right angle triangle |
| 3 | (2,3,4) | An Ordinary Triangle |
| 4 | (6,6,6) | An Equilateral Triangle |

(5) Conditional combination coverage: all possible combinations of the conditions in each decision should occur at least once

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Test case | Results |  |
| 1 | (2,3,4) | An Ordinary Triangle |  |
| 2 | (3,4,5) | Right angle triangle |  |
| 3 | (4,3,5) | Right angle triangle |  |
| 4 | (5,3,4) | Right angle triangle |  |
| 5 | (2,2,3) | An isosceles triangle |  |
| 6 | (2,3,2) | An isosceles triangle |  |
| 7 | (6,6,6) | An Equilateral Triangle |  |
| 8 | (2,2,2√2) | An isosceles right angle triangle |  |

(6) Path coverage: Every path in the program needs to be covered

|  |  |  |
| --- | --- | --- |
| S.No | Test case | Results |
| 1 | (2,2,2) | An Equilateral triangle |
| 2 | (2,2,3) | An isosceles triangle |
| 3 | (2,2,2√2) | An isosceles right angle triangle |
| 4 | (2,3,4) | An Ordinary Triangle |
| 5 | (3,4,5) | Right angle triangle |

2.2.5TriangleParamTest测试类的核心代码

**package** helloWorld;

**import** **static** org.junit.jupiter.api.Assertions.\*;

**import** java.util.Arrays;

**import** java.util.Collection;

**import** org.junit.After;

**import** org.junit.Before;

**import** org.junit.jupiter.api.AfterAll;

**import** org.junit.jupiter.api.AfterEach;

**import** org.junit.jupiter.api.BeforeAll;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Test;

**import** org.junit.runners.Parameterized.Parameters;

**class** TriangleParamTest {

**private** Triangle t;

**private** **int** param1;

**private** **int** param2;

**private** **int** param3;

**private** **int** except;

@Before

**public** **void** setUp() **throws** Exception {

t = **new** Triangle();

}

@After

**public** **void** tearDown() **throws** Exception {

t = **null**;

}

**public** TriangleParamTest(**int** param1,**int** param2,**int** param3,**int** except) {

**this**.param1 = param1;

**this**.param2 = param2;

**this**.param3 = param3;

**this**.except = except;

}

**public** **static** Collection<Object[]> paradata(){

**return** Arrays.*asList*(**new** Object[][] {

{3,3,3,1},{2,2,3,2},{3,4,5,4},{5,6,9,5}

});

}

@Test

**public** **void** testGetType() {

*assertEquals*(except, t.getType(param1, param2, param3));

}

}

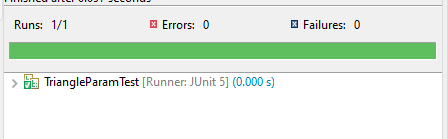
3实验步骤和结果

1. 编写被测程序Triangle类，包含以下两个方法

IsTriangle():判断是否为三角形

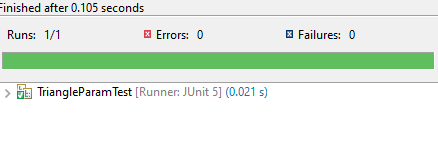
GetType():判断三角形类型，包括等边三角形、等腰三角形、直角三角形、等腰直角三角形、普通三角形。

以Calculator类为例，编写Triangle类的测试类TriangleTest，要求熟练使用Junit5各种注解和断言。



1. 使用白盒测试中的逻辑覆盖方法和基本路径测试设计测试用例。

因测试用例较多，使用参数化设置对Triangle类进行测试，编写参数化测试类TrianglrParamTest。



4实验结果分析

实验方案设计较为合理，istriangle():判断是否为三角形，getType()：判断三角形类型，通过Junit辅助函数对测试用例进行测试，皆具有合理性。