**《软件测试》**

**实验报告一 ——JUnit测试方法**

**姓 名： 曹倜源 学 号： 2020112805**

**院 系： 计算机与信息学院 专 业：信息管理与信息系统**

**实 验 室： J-2301 实验日期： 2021/3/19**

**总评成绩： 审阅教师：**

### 一、实验目的：

1. 掌握 git 协作式管理工具；
2. 掌握 gradle 项目构建基本操作
3. 锻炼基本代码编程能力

### 二、实验环境：

IDEA,JUnit

### 三、实验要求：

1. 学习使用 JUnit5 编写测试用例，参考[Junit5 五分钟教程](https://www.jianshu.com/p/3183fb69fd74" \o "Junit5 五分钟教程)
2. 使用 git 管理代码（gitee或GitHub均可）
3. 使用 gradle 项目构建工具，参考 [gradle 入门教程](https://www.cnblogs.com/hellxz/p/helloworld-gradle.html" \o "gradle 入门教程)，深入学习可以参考B站学习视频[来自Gradle开发团队的Gradle入门教程](https://www.bilibili.com/video/BV1DE411Z7nt?p=1" \o "来自Gradle开发团队的Gradle入门教程);或者使用maven构建工具创建项目；
4. 按照以下实验要求，使用 Java 语言实现判断三角形类型程序Triangle，并采用Junit5 编写完成三角形类型程序测试用例，并提交到代码仓库
5. 按照[实验报告模板](https://star.jmhui.com.cn/u/cms/www/202203/06160105hi3m.docx) 编写实验报告，以“学号-姓名-软件测试实验一”命名，提交到雨课堂“软件测试实验一”

### **四、实验步骤与内容**

1. **题目：三角形题目**

**输入**3个整数a、b和c分别作为三角形的三条边，要求a、b和c必须满足以下条件：

1、整数

2、3个数

3、边长大于等于1，小于等于100

4、任意两边之和大于第三边

**输出**为5种情况之一：

如果不满足条件1、2、3，则程序输出为“输入错误”。

如果不满足条件4，则程序输出为“非三角形”。

如果三条边相等，则程序输出为“等边三角形”。

如果恰好有两条边相等，则程序输出为“等腰三角形”。

如果三条边都不相等，则程序输出为“一般三角形”。

1. **测试用例：**

|  |  |  |  |
| --- | --- | --- | --- |
| **测试编号** | **测试用例** | **预测结果** | **实际结果** |
| **1** | 3,4,3=6 | 不等边三角形 | 不等边三角形 |
| **2** | 3,3,3 | 等边三角形 | 等边三角形 |
| **3** | 3,3,6 | 非三角形 | 非三角形 |

1. **代码实现：**

|  |
| --- |
| **实现代码：**  package cn.edu.ctgu;  public class Triangle {  public String classify(int a, int b, int c) {  if (a <1 || a>100 || b<1 || b>100 || c<1 || c> 100) {  return "输入错误";  }  if (!((a + b > c) && (a + c > b) && (b + c > a))) {  return "非三角形";  } else if (a == b && a == c && b == c) {  return "等边三角形";  } else if (a != b && a != c && b != c) {  return "不等边三角形";  } else {  return "等腰三角形";  }  } }  测试代码：  package cn.edu.ctgu;  import org.junit.jupiter.api.DisplayName; import org.junit.jupiter.api.Test;  import static org.junit.jupiter.api.Assertions.\*;  class TriangleTest {   @Test  @DisplayName("输入错误")  void parameters\_error\_test() {  Triangle triangle = new Triangle();  String type = triangle.classify(0, 4, 5);  assertEquals("输入错误", type);  }  @Test  @DisplayName("不等边三角形")  void scalene\_test() {  Triangle triangle = new Triangle();  String type = triangle.classify(3, 4, 6);  assertEquals("不等边三角形", type);  } }  测试结果：  IMG_256  覆盖率：  IMG_256 |

### **结论分析与体会**

JUnit自动测试框架是Java语言中最为流行的测试框架之一，它可以帮助我们快速、准确地测试我们的代码，确保其在各种情况下的正确性和稳定性。在学习JUnit自动测试框架时，我主要掌握了非参数化和参数化测试方法，并对这两种方法有了一些深刻的体会。

首先，非参数化测试是指在测试中，我们手动指定测试的输入值和预期结果。这种方法的优点是测试结果可靠，可以精确地检查代码是否正确，而缺点是测试时间长、难以保证完全覆盖所有可能的输入值。在我的学习中，我通过编写一些简单的测试用例，掌握了JUnit中如何编写非参数化测试。例如，我测试了一个计算器类中的加法方法，输入了一些数值进行测试，验证了方法的正确性。通过这个过程，我意识到，只有对所有可能的输入值进行测试，才能保证代码的健壮性和正确性。因此，对于比较复杂的代码，非参数化测试的覆盖率可能比较低，这时候就需要用到参数化测试。

其次，参数化测试是指在测试中，我们使用数据驱动的方式，自动化地生成各种不同的输入值和预期结果进行测试。这种方法的优点是可以大大提高测试的覆盖率和效率，同时还能减少手动编写测试用例的工作量。在学习JUnit中的参数化测试时，我了解了如何使用@Parameterized注解和@ValueSource注解来生成不同的输入值进行测试。我也了解了如何使用@CsvSource注解，从CSV文件中读取数据进行测试。在实践中，我发现参数化测试能够有效地提高测试效率，尤其是在测试中需要涉及多种输入值的情况下，可以极大地节省测试时间。

总的来说，学习JUnit自动测试框架，掌握非参数化和参数化测试方法是一项非常有价值的工作。通过我的学习体验，我认识到测试在软件开发中的重要性和必要性。测试能够帮助我们确保代码的正确性和稳定性，避免代码在运行时出现各种问题。而掌握JUnit中的测试方法，可以使我们的测试更加高效和精确，从而提高我们的开发效率。此外，JUnit的学习也让我更好地了解了Java语言中的一些重要概念，例如注解、反射等，对我的Java编程能力也有了一定的提高。

### **六、仓库地址**

https://github.com/Ctyring/soft-test.git