**《软件测试》**

**实验报告二 ——边界值分析方法**

**姓 名： 汪成肸 学 号： 2020105002**   **院 系： 计算机与信息学院 专 业：计算机科学与技术**

**实 验 室： 实验日期： 2023/03/15**

**总评成绩： 审阅教师：**

### 一、实验目的：

1. 练习基本的编程能力；
2. 掌握边界值分析法。

### 二、实验环境：

IntelliJ IDEA 2022.2.2

### 三、实验要求：

（1）针对实验一实现判断三角形类型程序Triangle，采用边界值分析方法设计测试用例，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。

### 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：**

**2.1 一般边界值方法**

有n个输入变量，设计测试用例使得一个变量在数据有效区域内取最大值、略小于最大值、正常值、略大于最小值和最小值。**对于有n个输入变量的程序，一般性边界值分析的测试用例个数为4n+1。**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试编号** | **边长a** | **边长b** | **边长c** | **预测结果** |
| **1** | **60** | **60** | 1 | 等腰三角形 |
| **2** | **60** | **60** | 2 | 等腰三角形 |
| **3** | **60** | **60** | 60 | 等边三角形 |
| **4** | **60** | **60** | 119 | 等腰三角形 |
| **5** | **60** | **60** | 120 | 非三角形 |
| **6** | **60** | **1** | 60 | 等腰三角形 |
| **7** | **60** | **2** | 60 | 等腰三角形 |
| **8** | **60** | **119** | 60 | 等腰三角形 |
| **9** | **60** | **120** | 60 | 非三角形 |
| **10** | **1** | 60 | 60 | 等腰三角形 |
| **11** | **2** | 60 | 60 | 等腰三角形 |
| **12** | **119** | 60 | 60 | 等腰三角形 |
| **13** | **120** | 60 | 60 | 非三角形 |

**2.2健壮边界值测试**

健壮性是指在异常情况下，软件还能正常运行的能力。健壮性考虑的主要部分是预期输出，而不是输入。健壮性测试是边界值分析的一种简单扩展。除了变量的5 个边界分析取值还要考虑略超过最大值（max）和略小于最小值（min）时的情况。健壮性测试的最大价值在于观察处理异常情况，它是检测软件系统容错性的重要手段。软件容错性的度量：从非法输入中恢复；健壮性有两层含义：容错能力和恢复能力。**对于有n个输入变量的程序，健壮性测试的测试用例个数为6n+1。**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试编号** | **边长a** | **边长b** | **边长c** | **预测结果** |
| **1** | **60** | **60** | 0 | 抛出异常 |
| **2** | **60** | **60** | 1 | 等腰三角形 |
| **3** | **60** | **60** | 2 | 等腰三角形 |
| **4** | **60** | **60** | 60 | 等边三角形 |
| **5** | **60** | **60** | 99 | 等腰三角形 |
| **6** | **50** | **50** | 100 | 非三角形 |
| **7** | **60** | **60** | 101 | 抛出异常 |
| **8** | **60** | 0 | **60** | 抛出异常 |
| **9** | **60** | 1 | **60** | 等腰三角形 |
| **10** | **60** | 2 | **60** | 等腰三角形 |
| **11** | **60** | 99 | **60** | 等腰三角形 |
| **12** | **50** | 100 | **50** | 非三角形 |
| **13** | **60** | 101 | **60** | 抛出异常 |
| **14** | 0 | **60** | **60** | 抛出异常 |
| **15** | 1 | **60** | **60** | 等腰三角形 |
| **16** | 2 | **60** | **60** | 等腰三角形 |
| **17** | 99 | **60** | **60** | 等腰三角形 |
| **18** | 100 | **50** | **50** | 非三角形 |
| **19** | 101 | **60** | **60** | 抛出异常 |

**2.3 最坏边界值测试**

最坏情况测试中，对每一个输入变量首先进行包含最小值、略高于最小值、正常值、略低于最大值、最大值等5个元素集合的测试，然后对这些集合进行笛卡尔积计算，以生成测试用例。**对于有n个输入变量的程序，健壮性测试的测试用例个数为5^n。**

**2.4 最坏健壮性测试**

健壮性最坏情况假设对每一个变量首先进行最小值、略小于最小值的值、略高于最小值的值、正常值、最大值、略高于最大值的值、略低于最大值的值等7个元素的集合。然后对这些集合进行笛卡尔积运算，以生成测试用例。**对于有n个输入变量的程序，健壮最坏情况测试的测试用例个数为7^n。**

1. **代码实现：**

|  |
| --- |
| **实现代码：**  public class Triangle {  final static class TriangleException extends Exception {  public TriangleException(String msg) {  super(msg);  }  }  public String classify(int a, int b, int c) throws TriangleException {  if (a <1 || a>100 || b<1 || b>100 || c<1 || c> 100) {  throw new TriangleException("参数异常");  }   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 org.wcx.code;  import org.junit.jupiter.params.ParameterizedTest; import org.junit.jupiter.params.provider.CsvFileSource;  import static org.junit.jupiter.api.Assertions.\*;   public class TriangleTest {   //内部静态三角形测试工具类，不希望被继承  final static class TriangleTestUtil {   //生成笛卡尔积  private static void generateCartesianProduct(int[] nums) {  int index = 1;  for (int i : nums) {  for (int i1 : nums) {  for (int i2 : nums) {  System.*out*.println(index + "," + i + "," + i1 + "," + i2);  index ++;  }  }  }  }  }   public static void main(String[] args) {  int[] common = {1, 2, 60, 99, 100};  int[] robust = {0, 1, 2, 60, 99, 100, 101};  System.*out*.println("以下是判断最坏边界值测试");  TriangleTestUtil.*generateCartesianProduct*(common);  System.*out*.println("以下是判断最坏健壮边界值测试");  TriangleTestUtil.*generateCartesianProduct*(robust);  }   private final Triangle triangle = new Triangle();    @ParameterizedTest  @CsvFileSource(resources = "/一般边界值测试用例.csv", numLinesToSkip = 1)  public void generalBoundaryValueTestCases(int index, int a, int b, int c, String expectedRes) throws Triangle.TriangleException {  *assertEquals*(expectedRes, triangle.classify(a, b, c));  }   @ParameterizedTest  @CsvFileSource(resources = "/健壮边界值测试用例.csv", numLinesToSkip = 1)  public void robustBoundaryValueTestCases(int index, int a, int b, int c, String expectedRes) {  try {  *assertEquals*(expectedRes, triangle.classify(a, b, c));  } catch (Triangle.TriangleException e) {  *assertEquals*(expectedRes, e.getMessage());  }  } }  测试结果：  一般边界值测试结果:    健壮边界值测试结果:    最坏边界值测试和健壮边界测试测试用例过于过多，考虑到不是实际工程，测试类给出了生成的测试用例，测试用户的预期结果需要人工判断，也可以使用程序判断，但我们需要对这个程序进行测试。 |

### 五、结论分析与体会

通过这次实验，我掌握了如何设计测试用例对程序进行一般边界值测试、健壮边界值测试、最坏边界值测试、最坏健壮边界值测试。边界值分析法就是对输入或输出的边界值进行测试的一种黑盒测试方法，通常作为对等价类划分法的补充，其测试用例来自等价类的边界。所谓边界值，是指相对于输入等价类和输出等价类而言，稍高于边界或稍低于边界的一些特定情况。

### 六、仓库地址

https://github.com/shuaishuaixi03/software-testing/tree/main/project-0315/code