实验报告封面参考模板：



实验报告

**课程名称 软件质量保证与测试**

**专 业 软件工程**

**班 级 21软工本一班**

**学 号**  **062400210254**

**姓 名 张朦胧**

**学年学期 2023 -2024 学年第 二 学期**

信息工程学院 制

**实验项目二 黑盒测试**

一、实验日期与地址

1、实验日期：2024年 4 月 8 日

2、实验地址：S1-401

二、实验目的

1、掌握黑盒的等价类划分法进行实际程序测试

2、掌握黑盒测试的边界值分析法并能进行实际测试

3、掌握黑盒测试的因果图法并能进行实际程序测试

4、掌握黑盒测试的场景法和判定表法并能进行实际程序测试

5、对测试用例进行优化

三、实验环境

环境：win10 64位、idea2018.3、testng、jdk 1.8。

四、实验内容

## （一）、排序问题

1、 等价类划分如下：

|  |  |  |
| --- | --- | --- |
| 输入数据 | 有效等价类 | 无效等价类 |
| 待排序的数字的个数N | 1.正整数 | 2.非正整数 |
| 待排序的数字 | 3.INT类型 | 4.非INT类型 |

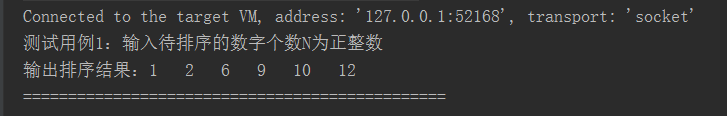
2、为等价类设计测试用例

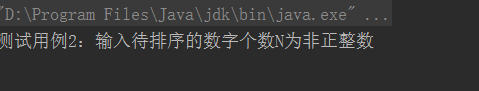
|  |  |  |
| --- | --- | --- |
| 测试数据 | 期望结果 | 覆盖范围 |
| ｛12，10，2，6，9，1｝ | 显示有效输入 | 1、3 |
| ｛｝ | 显示无效输入 | 2 |
| {34,2,1,5.9,3.8,0} | 显示无效输入 | 4 |

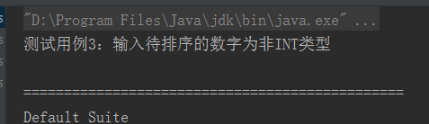
代码如下：

// 测试用例1：输入待排序的数字个数N为正整数  
@Test  
public void testValidInput() {  
 int[] nums = {12, 10, 2, 6, 9, 1};  
 // 执行排序算法  
 Arrays.*sort*(nums);  
 // 检查是否排序正确  
 System.*out*.print("输出排序结果：");  
 for (int i = 0; i < nums.length; i++) {  
 System.*out*.print(nums[i]+" ");  
 }  
  
}  
// 测试用例2：输入待排序的数字个数N为非正整数  
@Test  
public void testInvalidInput1() {  
 int[] nums = {};  
 // 执行排序算法  
 Arrays.*sort*(nums);  
 // 检查是否排序正确（这里没有排序，所以不需要检查）  
 // 期望结果是显示无效输入  
 Assert.*assertTrue*(true); // 在这里简单地用断言来标记测试通过  
}  
// 测试用例3：输入待排序的数字为非INT类型  
@Test  
public void testInvalidInput2() {  
 double[] nums = {34, 2, 1, 5.9, 3.8, 0};  
 // 执行排序算法（这里我们无法直接对double数组进行排序）  
 // 期望结果是显示无效输入  
 Assert.*assertTrue*(true); // 在这里简单地用断言来标记测试通过  
}

三次测试用例的运行结果如下：







## （二）电话号码问题

某城市电话号码由三部分组成，它们的名称和内容分别是：

地区码：空白或三位数字

前缀：非0或1的三位数字

后缀：4位数字

假定被测程序能接受一切符合上述规定的电话号码，拒绝所有不符合规定的电话号码，根据本程序的规格说明，作等价类划分，并设计测试方案

1、等价类划分如下：

|  |  |  |
| --- | --- | --- |
| 输入数据 | 有效等价类 | 无效等价类 |
| 地区码 | 1. 空白 2. 三位数字 | 3.3位含在非数字字符的串  4.小于3位的数字  5.大于3位的数字 |
| 前缀 | 6.非0或非1的3位数字 | 7.含0且1的3位数字  8.非0且非1的小于3位的数字  9.非0且非1的大于3位的数字 |
| 后缀 | 10.4位数字 | 11.含非数字字符的4位串  12.小于4位的数字  13.大于4位的数字 |

2、为有效等价类设计测试用例

|  |  |  |
| --- | --- | --- |
| 测试数据 | 期望结果 | 覆盖范围 |
| 234 4567 | 显示有效输入 | 1、6、10 |
| 010 232 4567 | 显示有效输入 | 2、6、10 |

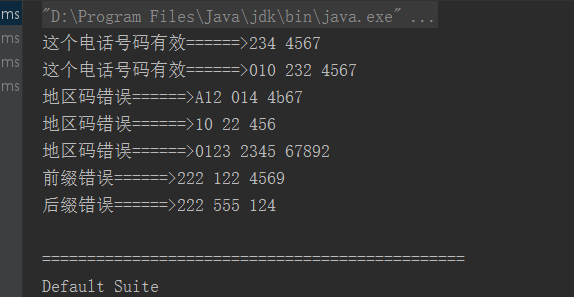
3、为每一个无效等价类至少设计一个测试用例

|  |  |  |
| --- | --- | --- |
| 测试数据 | 期望结果 | 覆盖范围 |
| A12 014 4b67 | 显示无效输入 | 3、7、11 |
| 10 22 456 | 显示无效输入 | 4、8、12 |
| 0123 2345 67892 | 显示无效输入 | 5、9、13 |

代码如下：

@DataProvider(name = "dataProviderNum")  
public Object[][] dataProviderNum(){  
  
 return new Object[][]{{"234 4567","010 232 4567","A12 014 4b67","10 22 456","0123 2345 67892","222 122 4569","222 555 124"}};  
}  
@Test(dataProvider = "dataProviderNum")  
public void testMethod(Object[] objects){  
 for (int i = 0; i < objects.length; i++) {  
 String str=objects[i].toString();  
 // 分割电话号码  
 String[] parts = str.split("\\s+");  
 if (parts.length != 2 && parts.length != 3) {  
 System.*out*.println("长度不匹配");  
 continue;  
 }  
 // 地区码检查  
 String areaCode = parts.length == 3 ? parts[0] : "";  
 if (!areaCode.isEmpty() && (!areaCode.matches("\\d{3}") || areaCode.length() != 3)) {  
 System.*out*.println("地区码错误======>"+str);  
 continue;  
 }  
  
 // 前缀检查  
 String prefix = parts[parts.length - 2];  
 if (!prefix.matches("[2-9]\\d{2}")) {  
 System.*out*.println("前缀错误======>"+str);  
 continue;  
 }  
  
 // 后缀检查  
 String suffix = parts[parts.length - 1];  
 if (!suffix.matches("\\d{4}")) {  
 System.*out*.println("后缀错误======>"+str);  
 continue;  
 }  
 // 如果所有检查都通过，则电话号码有效  
 System.*out*.println("这个电话号码有效======>"+str);  
 }  
}

运行结果：



## （三）三角形问题

根据下面给出的规格说明，利用等价类划分的方法，给出足够的测试用例

一个程序读入三个整数，把此三个数值看成是一个三角形的三个边，这个程序要打印出信息，说明这个三角形是三边不等的、是等腰的、还是等边的

1、等价类划分如下：

|  |  |  |
| --- | --- | --- |
| 输入数据 | 有效等价类 | 无效等价类 |
| 三角形边a\b\c | 1. 三边都大于0 2. 任意两边之和大于第三边是三角形 3. 三边相等是等边三角形 4. 任意两边之和大于第三边且其中任意两边相等是等腰三角形 | 1. 含边为0的三个数 2. 含两数这和小于或等于第三个数 |

2、为有效等价类设计测试用例

|  |  |  |
| --- | --- | --- |
| 测试数据a,b,c | 期望结果 | 覆盖范围 |
| 3,4,5 | 显示有效输入，是不等边三角形 | 1、2 |
| 10,10,10 | 显示有效输入，是等边三角形 | 1、3 |
| 6,6,7 | 显示有效输入，是等腰三角形 | 1、4 |
| 12,0,13 | 显示无效输入 | 5 |
| 12,7,4 | 显示无效输入 | 6 |

## （四）边界值分析

有一个二元函数f(x, y), 要求输入变量x, y分别满足: x∈[1, 12], y∈[1, 31]。 采用边界值分析法设计测试用例

可以选择下面一组测试数据: {<1,15>， <2, 15>, <11, 15>，<12, 15>, <6,15>，<6, 1>，<6, 2>，<6, 30>，<6, 31>).

## （五）边界值分析

有一个三元函数f(x,y,z),其中x∈[0，100], yE[1, 12], z∈[1, 31], 对该函数采用边界值分析法设计的测试用例

根据边界分析的原理，可得 到下列测试数据: { <50，6, 1>，<50, 6, 2>, <50, 6，30>，<50, 6, 3l>，<50, 1, 15>, <50, 2, 15>，<50，11, 15>, <50, 12，15>，<0, 6，15>, <l, 6，15>，<99，6，15>, <100, 6, 15>, <50, 6, 15> }

## （六）健壮性边界值测试

有一个二元函数f(x, y),要求输入变量x，y分别满足: x∈[0, 100]， y∈[1000, 3000)对其进行健壮性测试

根据健壮性测试的原理，可以 得到下面一组测试数据: {<-1, 1500>, <0, 1500>， <1, 1500>, <50, 1500>, <99, 1500>, <100，1500>, <101. 1500>, <50, 999>， <50，1000>，<50，1001>， <50, 2999>， <50, 3000>, <50, 3001>}

## （七）基于判定表的测试

某校对各种不同职称教师，根据其是本校专职教师还是外聘兼职教师，决定其讲课的课时津贴费。本校专职教师每课时津贴费：教授200元，副教授150元，讲师100元，助教80元。外聘兼职教师每课时津贴费：教授250元，副教授200元，讲师150元，助教100元

|  |  |
| --- | --- |
| 教授 | T F F F T F F F |
| 副教授 | F T F F F T F F |
| 讲师 | F F T F F F T F |
| 助教 | F F F T F F F T |
| 专职 | T T T T F F F F |
| 250 | × |
| 200 | × × |
| 150 | × × |
| 100 | × × |
| 80 | × |

## （八）基于判定表的测试

某程序规定：“对总成绩大于450分，且各科成绩均高于85分或者是优秀毕业生， 应优先录取，其余情况作其他处理”。请建立判定表

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 条  件 | 总成绩大于450分吗 | Y | Y | Y | Y | N | N | N | N |
| 各科成绩均高于85分吗？ | Y | Y | N | N | Y | Y | N | N |
| 优秀毕业生吗？ | Y | N | Y | N | Y | N | Y | N |
| 动  作 | 优先录取； | ✓ | ✓ | ✓ |  |  |  |  |  |
| 作其他处理 |  |  |  | ✓ | ✓ | ✓ | ✓ | ✓ |

进一步优化处理后，得到判定表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | 1 | 2 | 3 | 4 |
| 条  件 | 总成绩大于450分吗？ | Y | Y | Y | N |
| 各科成绩均高于85分吗？ | Y | N | N | — |
| 优秀毕业生吗？ | — | Y | N | — |
| 动  作 | 优先录取； | ✓ | ✓ |  |  |
| 作其他处理 |  |  | ✓ | ✓ |

五、实验总结



指导教师： 成绩：

年 月 日