****

**软件测试**

**软件测试课程作业说明文档**

|  |  |  |
| --- | --- | --- |
| 组 长 | 严祯颋 | 1750063 |
| 组 员 | 浦家瑞 | 1651200 |
| 王笑天 | 1751862 |
| 周宇东 | 1751659 |
| 指导老师 | 杜 庆 峰 | |
| 所在院系 | 软件学院 | |
| 专 业 | 软件工程 | |

二〇二〇年六月

目 录

[1. 练习1—三角形问题及万年历问题 4](#_Toc44333656)

[1.1. 三角形问题分析 4](#_Toc44333657)

[1.1.1. 边界值法 4](#_Toc44333658)

[1.1.2. 等价类法 5](#_Toc44333659)

[1.2. 三角形问题测试代码设计 6](#_Toc44333660)

[1.3. 三角形问题测试用例执行情况 7](#_Toc44333661)

[1.3.1. 边界值法 8](#_Toc44333662)

[1.3.2. 等价类法 9](#_Toc44333663)

[1.4. 万年历问题分析 10](#_Toc44333664)

[1.4.1. 边界值法 11](#_Toc44333665)

[1.4.2. 等价类法 11](#_Toc44333666)

[1.5. 测试代码设计 17](#_Toc44333667)

[1.6. 测试用例执行情况 18](#_Toc44333668)

[1.6.1. 边界值法 18](#_Toc44333669)

[1.6.2. 等价类法 20](#_Toc44333670)

[2. 练习2—佣金问题问题 24](#_Toc44333671)

[2.1. 问题分析 24](#_Toc44333672)

[2.2. 用例设计 27](#_Toc44333673)

[2.3. 测试代码设计 28](#_Toc44333674)

[2.4. 测试用例执行情况 30](#_Toc44333675)

[2.4.1. 销售额输出测试 30](#_Toc44333676)

[2.4.2. 佣金输出测试 31](#_Toc44333677)

[3. 练习3—使用Junit实践并了解原理 33](#_Toc44333678)

[3.1. Junit介绍 33](#_Toc44333679)

[3.2. Junit使用 34](#_Toc44333680)

[3.3. Junit原理 34](#_Toc44333681)

[3.4. 项目Junit实践 36](#_Toc44333682)

[4. 练习4—以类作为单位如何定义driver和stub 37](#_Toc44333683)

[4.1. 以类为单位定义Driver 37](#_Toc44333684)

[4.1.1. Driver介绍 37](#_Toc44333685)

[4.1.2. Driver实例 38](#_Toc44333686)

[4.1.3. Driver总结 40](#_Toc44333687)

[4.2. 以类为单位定义Stub 40](#_Toc44333688)

[4.2.1. Stub介绍 40](#_Toc44333689)

[4.2.2. Stub实例 40](#_Toc44333690)

[4.2.3. Stub总结 43](#_Toc44333691)

[5. 练习5—找相关的开源测试工具，缺陷跟踪工具等 43](#_Toc44333692)

[5.1. 开源测试工具 43](#_Toc44333693)

[5.1.1. 静态扫描工具 43](#_Toc44333694)

[5.1.1.1. SonarQube 43](#_Toc44333695)

[5.1.1.2. FindBugs 46](#_Toc44333696)

[5.1.1.3. Alibaba Java Coding Guidelines 47](#_Toc44333697)

[5.1.2. 性能测试工具 48](#_Toc44333698)

[5.1.2.1. Apache JMeter 48](#_Toc44333699)

[5.1.2.2. OpenSTA 49](#_Toc44333700)

[5.1.3. 自动化功能测试工具 50](#_Toc44333701)

[5.1.3.1. Selenium 50](#_Toc44333702)

[5.1.3.2. Appium 50](#_Toc44333703)

[5.2. 缺陷跟踪工具 51](#_Toc44333704)

[5.2.1. Teambition 51](#_Toc44333705)

[5.2.2. Github Issues 53](#_Toc44333706)

[6. 练习6—设计transition tree 算法 55](#_Toc44333707)

[6.1. 从状态图构建状态树算法 55](#_Toc44333708)

[6.2. ATM系统状态图 55](#_Toc44333709)

[6.3. State Transition Tree构建 56](#_Toc44333710)

[6.4. 测试用例构建 57](#_Toc44333711)

[6.4.1. 测试数据准备 57](#_Toc44333712)

[6.4.2. 基于状态转换的测试分析与用例设计 57](#_Toc44333713)

[7. 练习7—电信收费问题 60](#_Toc44333714)

[7.1. 练习分析 61](#_Toc44333715)

[7.1.1. 决策表法 62](#_Toc44333716)

[7.1.2. 等价类法 63](#_Toc44333717)

[7.1.3. 边界值法 67](#_Toc44333718)

[7.2. 测试代码设计 68](#_Toc44333719)

[7.3. 测试用例执行情况 69](#_Toc44333720)

[7.3.1. 决策表法 69](#_Toc44333721)

[7.3.2. 等价类法 71](#_Toc44333722)

[7.3.3. 边界值法 75](#_Toc44333723)

[7.4. 对比分析 76](#_Toc44333724)

[7.4.1. 综合测试用例设计 77](#_Toc44333725)

[7.4.2. 测试结果 79](#_Toc44333726)

[8. 练习8—销售系统问题问题 80](#_Toc44333727)

[8.1. 问题分析 81](#_Toc44333728)

[8.2. 用例设计 81](#_Toc44333729)

[8.2.1. 语句覆盖 81](#_Toc44333730)

[8.2.2. 判断覆盖 82](#_Toc44333731)

[8.2.3. 条件覆盖 82](#_Toc44333732)

[8.2.4. 判断—条件覆盖 83](#_Toc44333733)

[8.2.5. 条件组合覆盖 84](#_Toc44333734)

[8.3. 测试代码设计 88](#_Toc44333735)

[8.4. 测试用例执行情况 89](#_Toc44333736)

[8.4.1. 语句覆盖 89](#_Toc44333737)

[8.4.2. 判断覆盖 90](#_Toc44333738)

[8.4.1. 条件覆盖 90](#_Toc44333739)

[8.4.1. 判断—条件覆盖 91](#_Toc44333740)

[8.4.2. 条件组合覆盖 93](#_Toc44333741)

[9. 练习9—根据代码构建CFG图 94](#_Toc44333742)

[9.1. 流程图 95](#_Toc44333743)

[9.2. 控制流图（CFG） 97](#_Toc44333744)

# 练习1—三角形问题及万年历问题

判定三角形类型、万年历分别用边界值和等价类设计测试用例，需求描述如下：

三角形问题：

输入3个正数a、b和c，作为三角形的3条边。通过程序判断出由这3条边所构成的三角形的类型是等边三角形、等腰三角形还是一般三角形，并打印出相应的信息。

万年历问题：

输入3个整数，分别代表年月日，通过程序输出下一个天的日期

## 三角形问题分析

在本题中，我们需要根据输入的三个正数变量a、b、c，判断以这三个整数为边的三角形类型。

在这里我们对程序的输出格式加以规定，以方便之后的测试，对于不同类型的三角形，程序分别输出“等边三角形”、“等腰三角形”和“一般三角形”，同时针对可能存在的a、b、c三条边无法构成三角形的异常情况，输出“无法构成三角形”。

同时，由于在这道题中，三条边的长度只需要满足正数即可，不存在上界，因此在这里人工规定三角形各边取值的定义域为(0,100]。对于超出范围的输入，则增加输出“变量超出取值范围”。

### 边界值法

对于三角形问题来说，我们人工规定各边取值范围为(0,100]，因此，在这里，我们可以规定各边正常值为50，同时假定我们固定住a，b的值，由于构成三角形的条件存在，对于第三条边c，我们可以进一步得出c的一个限制条件，也就是|a-b|<c<a+b，因此此时c的取值范围为max(0,|a-b|)<c<min(100,a+b)。同样，我们可以构造出a，b的取值范围。同时，由于a，b，c的取值依赖于其他两项的取值，因此在本问题中，我们采用健壮性边界测试设计设计用例。

根据上述分析，我们得到以下测试用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | a | b | c | 预计输出 |
|  | 0 | 50 | 50 | 变量超出取值范围 |
|  | 2.2 | 50 | 50 | 等腰三角形 |
|  | -0.1 | 50 | 50 | 变量超出取值范围 |
|  | 90.1 | 50 | 50 | 等腰三角形 |
|  | 100 | 50 | 50 | 无法构成三角形 |
|  | 102 | 50 | 50 | 变量超出取值范围 |
|  | 50 | 0 | 50 | 变量超出取值范围 |
|  | 50 | 2.2 | 50 | 等腰三角形 |
|  | 50 | -0.1 | 50 | 变量超出取值范围 |
|  | 50 | 90.1 | 50 | 等腰三角形 |
|  | 50 | 100 | 50 | 无法构成三角形 |
|  | 50 | 102 | 50 | 变量超出取值范围 |
|  | 50 | 50 | 0 | 变量超出取值范围 |
|  | 50 | 50 | 2.2 | 等腰三角形 |
|  | 50 | 50 | -0.1 | 变量超出取值范围 |
|  | 50 | 50 | 90.1 | 等腰三角形 |
|  | 50 | 50 | 100 | 无法构成三角形 |
|  | 50 | 50 | 102 | 变量超出取值范围 |
|  | 50 | 50 | 50 | 等边三角形 |

### 等价类法

通过对题目的分析，我们可以看到，问题的输出有“等边三角形”、“等腰三角形”、“一般三角形”、“无法构成三角形”以及针对无效输入的输出“变量超出取值范围”。

因此我们根据问题输出构造等价类，可以构造出如下等价类：

D1={{a,b,c}|a,b,c构成等边三角形}

D2={{a,b,c}|a,b,c构成非等边等腰三角形}

D3={{a,b,c}|a,b,c构成非等边非等腰的一般三角形}

D4={{a,b,c}|a,b,c不构成三角形}

同时，我们考虑各边的取值范围(0,100]，对于每个参数的无效取值分别有低于最小值和高于最大值两种情况（其中各边取0已经包含于低于最小值这一情况）。

综上所述，在基于“单缺陷”假设的情况下，我们采用弱健壮性等价类测试，弱健壮性等价类的测试用例由弱一般等价类测试用例和额外弱健壮性等价类测试用例构成。弱一般等价类测试用例，应当由输出变量的取值划分等价类，而额外弱健壮等价类测试用例的设计应当从输入变量取值的有效范围来考虑。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | a | b | c | 预计输出 |
|  | 10 | 10 | 10 | 等边三角形 |
|  | 2.5 | 2.5 | 2 | 等腰三角形 |
|  | 60 | 50 | 32.5 | 一般三角形 |
|  | 10 | 20 | 10 | 无法构成三角形 |
|  | 0 | 2 | 2 | 变量超出取值范围 |
|  | 2 | 0 | 2 | 变量超出取值范围 |
|  | 2 | 2 | 0 | 变量超出取值范围 |
|  | 105 | 70 | 70 | 变量超出取值范围 |
|  | 70 | 70 | 105 | 变量超出取值范围 |
|  | 70 | 105 | 70 | 变量超出取值范围 |
|  | -1 | 2 | 2 | 变量超出取值范围 |
|  | 2 | -1 | 2 | 变量超出取值范围 |
|  | 2 | 2 | -1 | 变量超出取值范围 |

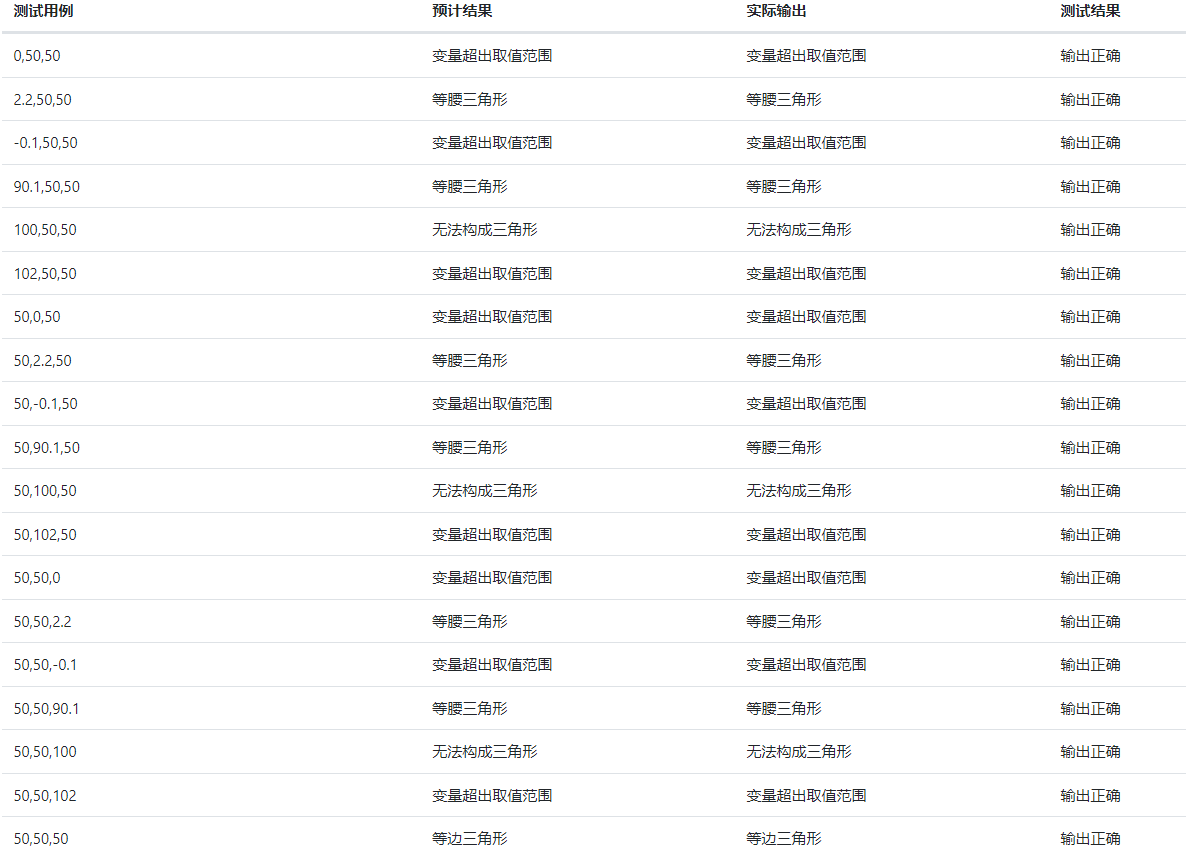
## 三角形问题测试代码设计

1. **public** **class** TriangleJudger{
2. **private** **static** **final** **double** MinV=0;
3. **private** **static** **final** **double** MaxV=100;
4. **private** **static** **boolean** comp(**double** a,**double** b){
5. **return** Math.abs(a-b)<1e-6;
6. }
7. **public** **static** String getTriangleType(**double** a,**double** b,**double** c){
8. **double** tmp;
9. **if**(a>b){
10. tmp=a;
11. a=b;
12. b=tmp;
13. }
14. **if**(b>c){
15. tmp=b;
16. b=c;
17. c=tmp;
18. }
19. **if**(a>b){
20. tmp=a;
21. a=b;
22. b=tmp;
23. }
24. **if**(a<=MinV||c>MaxV){
25. **return** "变量超出取值范围";
26. }
27. **if**(a+b<c||comp(a+b,c)){
28. **return** "无法构成三角形";
29. }
30. **if**(comp(a,b)&&comp(b,c)){
31. **return** "等边三角形";
32. }
33. **if**(comp(a,b)||comp(a,c)||comp(b,c)){
34. **return** "等腰三角形";
35. }
36. **return** "一般三角形";
37. }
38. }

## 三角形问题测试用例执行情况

使用自制代码测试工具对编写代码加以测试，得到如下结果：

### 边界值法

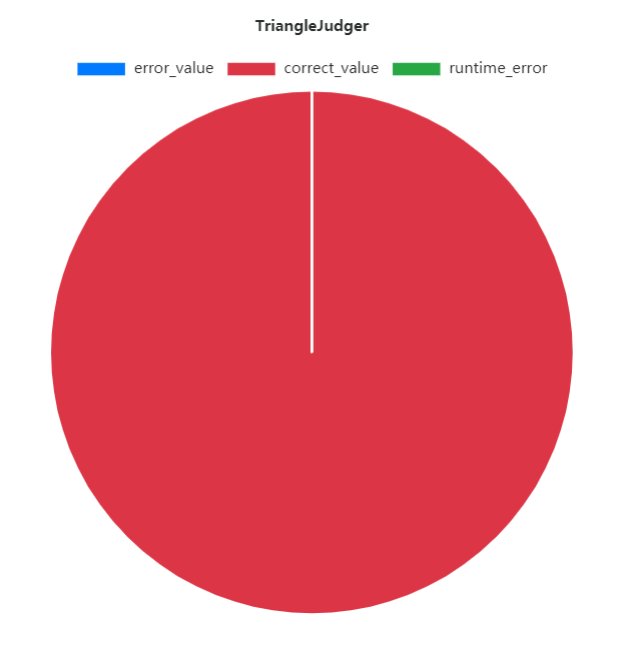




代码测试全部通过

### 等价类法

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 预计结果 | 实际输出 | 测试结果 |
| 10,10,10 | 等边三角形 | 等边三角形 | 输出正确 |
| 2.5,2.5,2 | 等腰三角形 | 等腰三角形 | 输出正确 |
| 60,50,32.5 | 一般三角形 | 一般三角形 | 输出正确 |
| 10,20,10 | 无法构成三角形 | 无法构成三角形 | 输出正确 |
| 0,2,2 | 变量超出取值范围 | 变量超出取值范围 | 输出正确 |
| 2,0,2 | 变量超出取值范围 | 变量超出取值范围 | 输出正确 |
| 2,2,0 | 变量超出取值范围 | 变量超出取值范围 | 输出正确 |
| 105,70,70 | 变量超出取值范围 | 变量超出取值范围 | 输出正确 |
| 70,70,105 | 变量超出取值范围 | 变量超出取值范围 | 输出正确 |
| 70,105,70 | 变量超出取值范围 | 变量超出取值范围 | 输出正确 |
| -1,2,2 | 变量超出取值范围 | 变量超出取值范围 | 输出正确 |
| 2,-1,2 | 变量超出取值范围 | 变量超出取值范围 | 输出正确 |
| 2,2,-1 | 变量超出取值范围 | 变量超出取值范围 | 输出正确 |



代码测试全部通过

## 万年历问题分析

在本题中，我们需要根据输入的年月日三个正整数变量，得到下一天的日期输出，在这里，我们规定程序输出格式为形如“yyyy-mm-dd”形式的的字符串，同时，对于非法的输入，程序应输出“非法输入”。

在这里，我们对各变量的代表符号定义如下：

Y：输入的年份

M：输入的月份

D：输入的日期

我们很容易得出，1<=M<=12，1<=D<=31，对于年份的范围，我们人为规定1900<=Y<=2500。

### 边界值法

通过对上述问题的分析，我们使用健壮边界分析，对于Y、M、D的普通值，分别取2020，6，15，得到以下测试用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | Y | M | D | 预计输出 |
|  | 2020 | 6 | 1 | 2020-06-02 |
|  | 2020 | 6 | 0 | 非法输入 |
|  | 2020 | 6 | 2 | 2020-06-03 |
|  | 2020 | 6 | 31 | 非法输入 |
|  | 2020 | 6 | 32 | 非法输入 |
|  | 2020 | 6 | 30 | 2020-07-01 |
|  | 2020 | 1 | 15 | 2020-01-16 |
|  | 2020 | 0 | 15 | 非法输入 |
|  | 2020 | 2 | 15 | 2020-02-16 |
|  | 2020 | 12 | 15 | 2020-12-16 |
|  | 2020 | 13 | 15 | 非法输入 |
|  | 2020 | 11 | 15 | 2020-11-16 |
|  | 1900 | 6 | 15 | 1900-06-16 |
|  | 1899 | 6 | 15 | 非法输入 |
|  | 1901 | 6 | 15 | 1901-06-16 |
|  | 2500 | 6 | 15 | 2500-06-16 |
|  | 2501 | 6 | 15 | 非法输入 |
|  | 2499 | 6 | 15 | 2499-06-16 |

### 等价类法

由于本问题中，年月日三个变量并非相互独立的，而在使用边界值的情况下并未考虑到这一点，导致效果不好，下面，我们使用等价类法对测试用例进行分析。

通过常识，我们可以发现，对于等价类的划分，我们主要应当考虑闰年问题以及加一天后就将跳转到下一月或下一年的那部分日期，因此我们可以划分出以下等价类

|  |  |  |
| --- | --- | --- |
| 输入数据 | 有效等价类 | 无效等价类 |
| Y | 1:[1900~2500]中的实际闰年 2:[1900~2500]中的普通闰年 3:[1900~2500]中的平年 | 1:(-∞~1900) 2:(2500~+∞) |
| M | 1:2月 2:4,6,9,11月 3:1,3,5,7,8,10月 4:12月 | 1:(-∞~1) 2:(12~+∞) |
| D | 1:[1~27] 2:28 3:29 3:30 4:31 | 1:(-∞~1) 2:(31~+∞) 3:当月最大天数不超过30时输入31  4:当月最大天数为29时输入30  5:当月最大天数为28时输入29 |

从划分的等价类来说，我们可以看到，本问题的无效等价类是“多缺陷”依赖的，因此采用强健壮性等价类分析，构造出以下用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | Y | M | D | 预计输出 |
|  | 2000 | 2 | 27 | 2000-02-28 |
|  | 2000 | 2 | 28 | 2000-02-29 |
|  | 2000 | 2 | 29 | 2000-03-01 |
|  | 2000 | 2 | 30 | 非法输入 |
|  | 2000 | 2 | 31 | 非法输入 |
|  | 2000 | 2 | 0 | 非法输入 |
|  | 2000 | 2 | 32 | 非法输入 |
|  | 2000 | 6 | 27 | 2000-06-28 |
|  | 2000 | 6 | 28 | 2000-06-29 |
|  | 2000 | 6 | 29 | 2000-06-30 |
|  | 2000 | 6 | 30 | 2000-07-01 |
|  | 2000 | 6 | 31 | 非法输入 |
|  | 2000 | 6 | 0 | 非法输入 |
|  | 2000 | 6 | 32 | 非法输入 |
|  | 2000 | 3 | 27 | 2000-03-28 |
|  | 2000 | 3 | 28 | 2000-03-29 |
|  | 2000 | 3 | 29 | 2000-03-30 |
|  | 2000 | 3 | 30 | 2000-03-31 |
|  | 2000 | 3 | 31 | 2000-04-01 |
|  | 2000 | 3 | 0 | 非法输入 |
|  | 2000 | 3 | 32 | 非法输入 |
|  | 2000 | 12 | 27 | 2000-12-28 |
|  | 2000 | 12 | 28 | 2000-12-29 |
|  | 2000 | 12 | 29 | 2000-12-30 |
|  | 2000 | 12 | 30 | 2000-12-31 |
|  | 2000 | 12 | 31 | 2001-01-01 |
|  | 2000 | 12 | 0 | 非法输入 |
|  | 2000 | 12 | 32 | 非法输入 |
|  | 2000 | 0 | 27 | 非法输入 |
|  | 2000 | 0 | 28 | 非法输入 |
|  | 2000 | 0 | 29 | 非法输入 |
|  | 2000 | 0 | 30 | 非法输入 |
|  | 2000 | 0 | 31 | 非法输入 |
|  | 2000 | 0 | 0 | 非法输入 |
|  | 2000 | 0 | 32 | 非法输入 |
|  | 2000 | 13 | 27 | 非法输入 |
|  | 2000 | 13 | 28 | 非法输入 |
|  | 2000 | 13 | 29 | 非法输入 |
|  | 2000 | 13 | 30 | 非法输入 |
|  | 2000 | 13 | 31 | 非法输入 |
|  | 2000 | 13 | 0 | 非法输入 |
|  | 2000 | 13 | 32 | 非法输入 |
|  | 2008 | 2 | 27 | 2008-02-28 |
|  | 2008 | 2 | 28 | 2008-02-29 |
|  | 2008 | 2 | 29 | 2008-03-01 |
|  | 2008 | 2 | 30 | 非法输入 |
|  | 2008 | 2 | 31 | 非法输入 |
|  | 2008 | 2 | 0 | 非法输入 |
|  | 2008 | 2 | 32 | 非法输入 |
|  | 2008 | 6 | 27 | 2008-06-28 |
|  | 2008 | 6 | 28 | 2008-06-29 |
|  | 2008 | 6 | 29 | 2008-06-30 |
|  | 2008 | 6 | 30 | 2008-07-01 |
|  | 2008 | 6 | 31 | 非法输入 |
|  | 2008 | 6 | 0 | 非法输入 |
|  | 2008 | 6 | 32 | 非法输入 |
|  | 2008 | 3 | 27 | 2008-03-28 |
|  | 2008 | 3 | 28 | 2008-03-29 |
|  | 2008 | 3 | 29 | 2008-03-30 |
|  | 2008 | 3 | 30 | 2008-03-31 |
|  | 2008 | 3 | 31 | 2008-04-01 |
|  | 2008 | 3 | 0 | 非法输入 |
|  | 2008 | 3 | 32 | 非法输入 |
|  | 2008 | 12 | 27 | 2008-12-28 |
|  | 2008 | 12 | 28 | 2008-12-29 |
|  | 2008 | 12 | 29 | 2008-12-30 |
|  | 2008 | 12 | 30 | 2008-12-31 |
|  | 2008 | 12 | 31 | 2009-01-01 |
|  | 2008 | 12 | 0 | 非法输入 |
|  | 2008 | 12 | 32 | 非法输入 |
|  | 2008 | 0 | 27 | 非法输入 |
|  | 2008 | 0 | 28 | 非法输入 |
|  | 2008 | 0 | 29 | 非法输入 |
|  | 2008 | 0 | 30 | 非法输入 |
|  | 2008 | 0 | 31 | 非法输入 |
|  | 2008 | 0 | 0 | 非法输入 |
|  | 2008 | 0 | 32 | 非法输入 |
|  | 2008 | 13 | 27 | 非法输入 |
|  | 2008 | 13 | 28 | 非法输入 |
|  | 2008 | 13 | 29 | 非法输入 |
|  | 2008 | 13 | 30 | 非法输入 |
|  | 2008 | 13 | 31 | 非法输入 |
|  | 2008 | 13 | 0 | 非法输入 |
|  | 2008 | 13 | 32 | 非法输入 |
|  | 1900 | 2 | 27 | 1900-02-28 |
|  | 1900 | 2 | 28 | 1900-03-01 |
|  | 1900 | 2 | 29 | 非法输入 |
|  | 1900 | 2 | 30 | 非法输入 |
|  | 1900 | 2 | 31 | 非法输入 |
|  | 1900 | 2 | 0 | 非法输入 |
|  | 1900 | 2 | 32 | 非法输入 |
|  | 1900 | 6 | 27 | 1900-06-28 |
|  | 1900 | 6 | 28 | 1900-06-29 |
|  | 1900 | 6 | 29 | 1900-06-30 |
|  | 1900 | 6 | 30 | 1900-07-01 |
|  | 1900 | 6 | 31 | 非法输入 |
|  | 1900 | 6 | 0 | 非法输入 |
|  | 1900 | 6 | 32 | 非法输入 |
|  | 1900 | 3 | 27 | 1900-03-28 |
|  | 1900 | 3 | 28 | 1900-03-29 |
|  | 1900 | 3 | 29 | 1900-03-30 |
|  | 1900 | 3 | 30 | 1900-03-31 |
|  | 1900 | 3 | 31 | 1900-04-01 |
|  | 1900 | 3 | 0 | 非法输入 |
|  | 1900 | 3 | 32 | 非法输入 |
|  | 1900 | 12 | 27 | 1900-12-28 |
|  | 1900 | 12 | 28 | 1900-12-29 |
|  | 1900 | 12 | 29 | 1900-12-30 |
|  | 1900 | 12 | 30 | 1900-12-31 |
|  | 1900 | 12 | 31 | 1901-01-01 |
|  | 1900 | 12 | 0 | 非法输入 |
|  | 1900 | 12 | 32 | 非法输入 |
|  | 1900 | 0 | 27 | 非法输入 |
|  | 1900 | 0 | 28 | 非法输入 |
|  | 1900 | 0 | 29 | 非法输入 |
|  | 1900 | 0 | 30 | 非法输入 |
|  | 1900 | 0 | 31 | 非法输入 |
|  | 1900 | 0 | 0 | 非法输入 |
|  | 1900 | 0 | 32 | 非法输入 |
|  | 1900 | 13 | 27 | 非法输入 |
|  | 1900 | 13 | 28 | 非法输入 |
|  | 1900 | 13 | 29 | 非法输入 |
|  | 1900 | 13 | 30 | 非法输入 |
|  | 1900 | 13 | 31 | 非法输入 |
|  | 1900 | 13 | 0 | 非法输入 |
|  | 1900 | 13 | 32 | 非法输入 |
|  | 1899 | 2 | 27 | 非法输入 |
|  | 1899 | 2 | 28 | 非法输入 |
|  | 1899 | 2 | 29 | 非法输入 |
|  | 1899 | 2 | 30 | 非法输入 |
|  | 1899 | 2 | 31 | 非法输入 |
|  | 1899 | 2 | 0 | 非法输入 |
|  | 1899 | 2 | 32 | 非法输入 |
|  | 1899 | 6 | 27 | 非法输入 |
|  | 1899 | 6 | 28 | 非法输入 |
|  | 1899 | 6 | 29 | 非法输入 |
|  | 1899 | 6 | 30 | 非法输入 |
|  | 1899 | 6 | 31 | 非法输入 |
|  | 1899 | 6 | 0 | 非法输入 |
|  | 1899 | 6 | 32 | 非法输入 |
|  | 1899 | 3 | 27 | 非法输入 |
|  | 1899 | 3 | 28 | 非法输入 |
|  | 1899 | 3 | 29 | 非法输入 |
|  | 1899 | 3 | 30 | 非法输入 |
|  | 1899 | 3 | 31 | 非法输入 |
|  | 1899 | 3 | 0 | 非法输入 |
|  | 1899 | 3 | 32 | 非法输入 |
|  | 1899 | 12 | 27 | 非法输入 |
|  | 1899 | 12 | 28 | 非法输入 |
|  | 1899 | 12 | 29 | 非法输入 |
|  | 1899 | 12 | 30 | 非法输入 |
|  | 1899 | 12 | 31 | 非法输入 |
|  | 1899 | 12 | 0 | 非法输入 |
|  | 1899 | 12 | 32 | 非法输入 |
|  | 1899 | 0 | 27 | 非法输入 |
|  | 1899 | 0 | 28 | 非法输入 |
|  | 1899 | 0 | 29 | 非法输入 |
|  | 1899 | 0 | 30 | 非法输入 |
|  | 1899 | 0 | 31 | 非法输入 |
|  | 1899 | 0 | 0 | 非法输入 |
|  | 1899 | 0 | 32 | 非法输入 |
|  | 1899 | 13 | 27 | 非法输入 |
|  | 1899 | 13 | 28 | 非法输入 |
|  | 1899 | 13 | 29 | 非法输入 |
|  | 1899 | 13 | 30 | 非法输入 |
|  | 1899 | 13 | 31 | 非法输入 |
|  | 1899 | 13 | 0 | 非法输入 |
|  | 1899 | 13 | 32 | 非法输入 |
|  | 2501 | 2 | 27 | 非法输入 |
|  | 2501 | 2 | 28 | 非法输入 |
|  | 2501 | 2 | 29 | 非法输入 |
|  | 2501 | 2 | 30 | 非法输入 |
|  | 2501 | 2 | 31 | 非法输入 |
|  | 2501 | 2 | 0 | 非法输入 |
|  | 2501 | 2 | 32 | 非法输入 |
|  | 2501 | 6 | 27 | 非法输入 |
|  | 2501 | 6 | 28 | 非法输入 |
|  | 2501 | 6 | 29 | 非法输入 |
|  | 2501 | 6 | 30 | 非法输入 |
|  | 2501 | 6 | 31 | 非法输入 |
|  | 2501 | 6 | 0 | 非法输入 |
|  | 2501 | 6 | 32 | 非法输入 |
|  | 2501 | 3 | 27 | 非法输入 |
|  | 2501 | 3 | 28 | 非法输入 |
|  | 2501 | 3 | 29 | 非法输入 |
|  | 2501 | 3 | 30 | 非法输入 |
|  | 2501 | 3 | 31 | 非法输入 |
|  | 2501 | 3 | 0 | 非法输入 |
|  | 2501 | 3 | 32 | 非法输入 |
|  | 2501 | 12 | 27 | 非法输入 |
|  | 2501 | 12 | 28 | 非法输入 |
|  | 2501 | 12 | 29 | 非法输入 |
|  | 2501 | 12 | 30 | 非法输入 |
|  | 2501 | 12 | 31 | 非法输入 |
|  | 2501 | 12 | 0 | 非法输入 |
|  | 2501 | 12 | 32 | 非法输入 |
|  | 2501 | 0 | 27 | 非法输入 |
|  | 2501 | 0 | 28 | 非法输入 |
|  | 2501 | 0 | 29 | 非法输入 |
|  | 2501 | 0 | 30 | 非法输入 |
|  | 2501 | 0 | 31 | 非法输入 |
|  | 2501 | 0 | 0 | 非法输入 |
|  | 2501 | 0 | 32 | 非法输入 |
|  | 2501 | 13 | 27 | 非法输入 |
|  | 2501 | 13 | 28 | 非法输入 |
|  | 2501 | 13 | 29 | 非法输入 |
|  | 2501 | 13 | 30 | 非法输入 |
|  | 2501 | 13 | 31 | 非法输入 |
|  | 2501 | 13 | 0 | 非法输入 |
|  | 2501 | 13 | 32 | 非法输入 |

## 测试代码设计

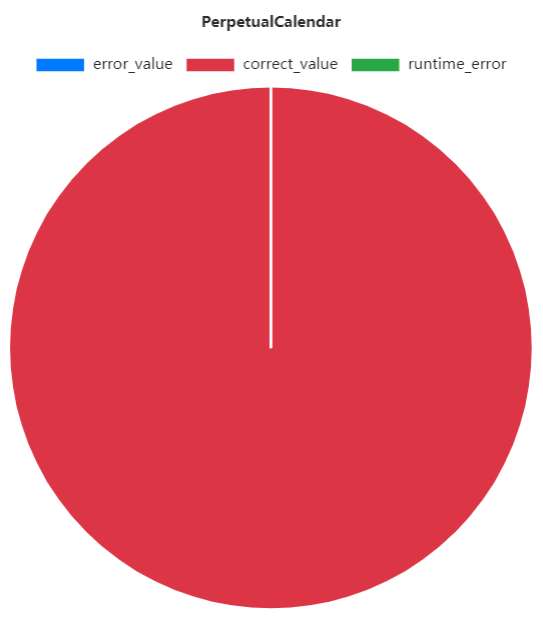
1. **import** java.text.SimpleDateFormat;
2. **import** java.util.Calendar;
4. **public** **class** PerpetualCalendar{
5. **public** **static** String getNextDay(**int** y,**int** m,**int** d){
6. **if**(y<1900||y>2500||m<1||m>12){
7. **return** "非法输入";
8. }
9. SimpleDateFormat dateFormat=**new** SimpleDateFormat("yyyy-MM-dd");
10. Calendar date=Calendar.getInstance();
11. date.set(Calendar.YEAR,y);
12. date.set(Calendar.MONTH,m-1);
13. var i=date.getActualMaximum(Calendar.DATE);
14. **if**(d<1||d>i){
15. **return** "非法输入";
16. }
17. date.set(Calendar.DATE,d);
18. date.add(Calendar.DATE,1);
19. **return** dateFormat.format(date.getTime());
20. }
21. }

## 测试用例执行情况

使用自制代码测试工具对编写代码加以测试，得到如下结果：

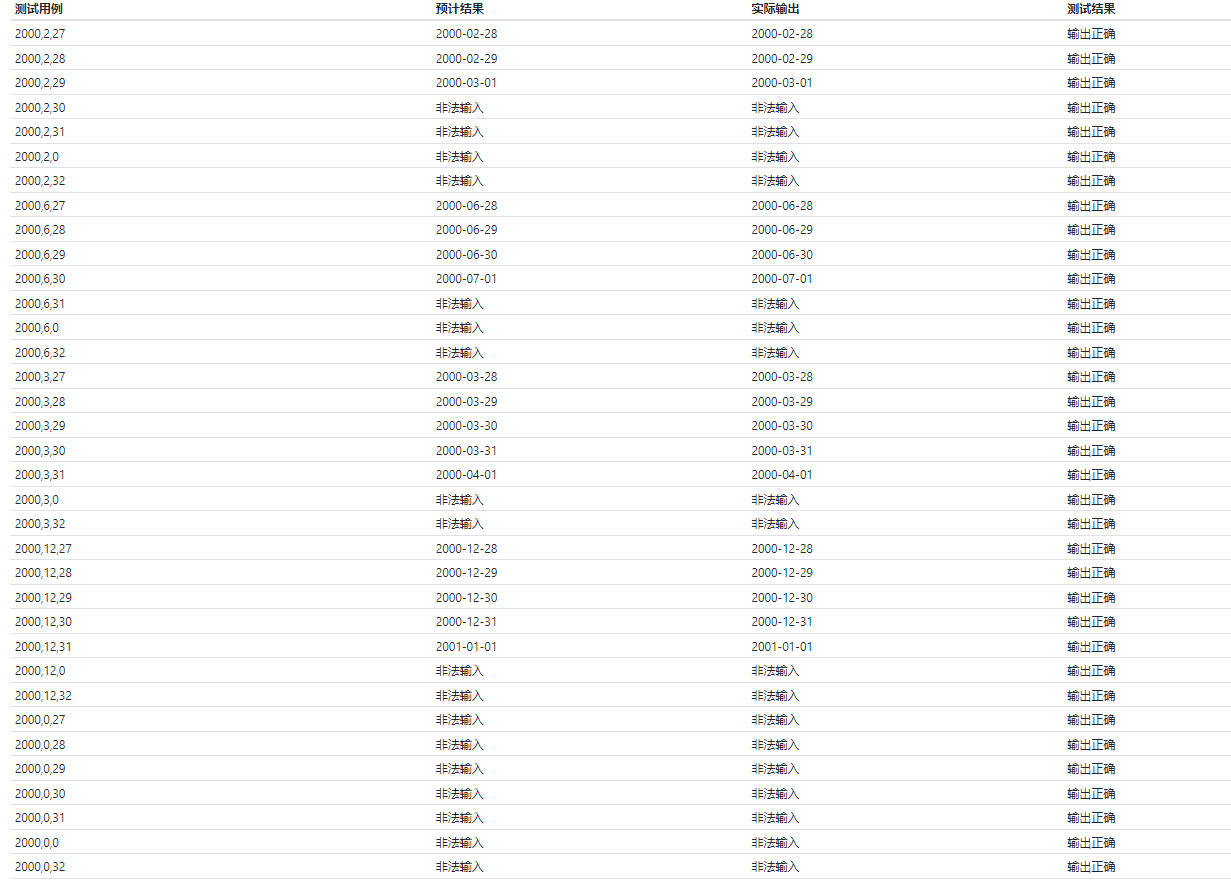
### 边界值法

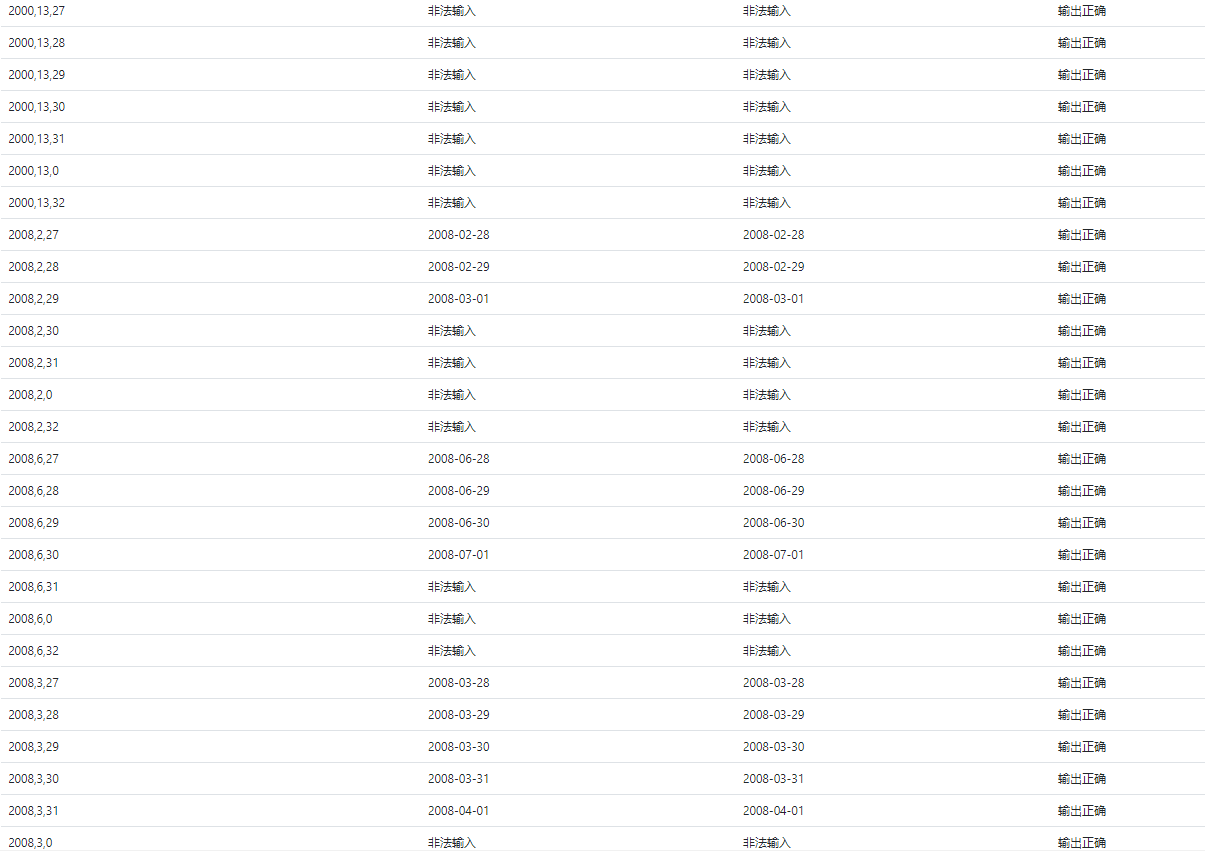


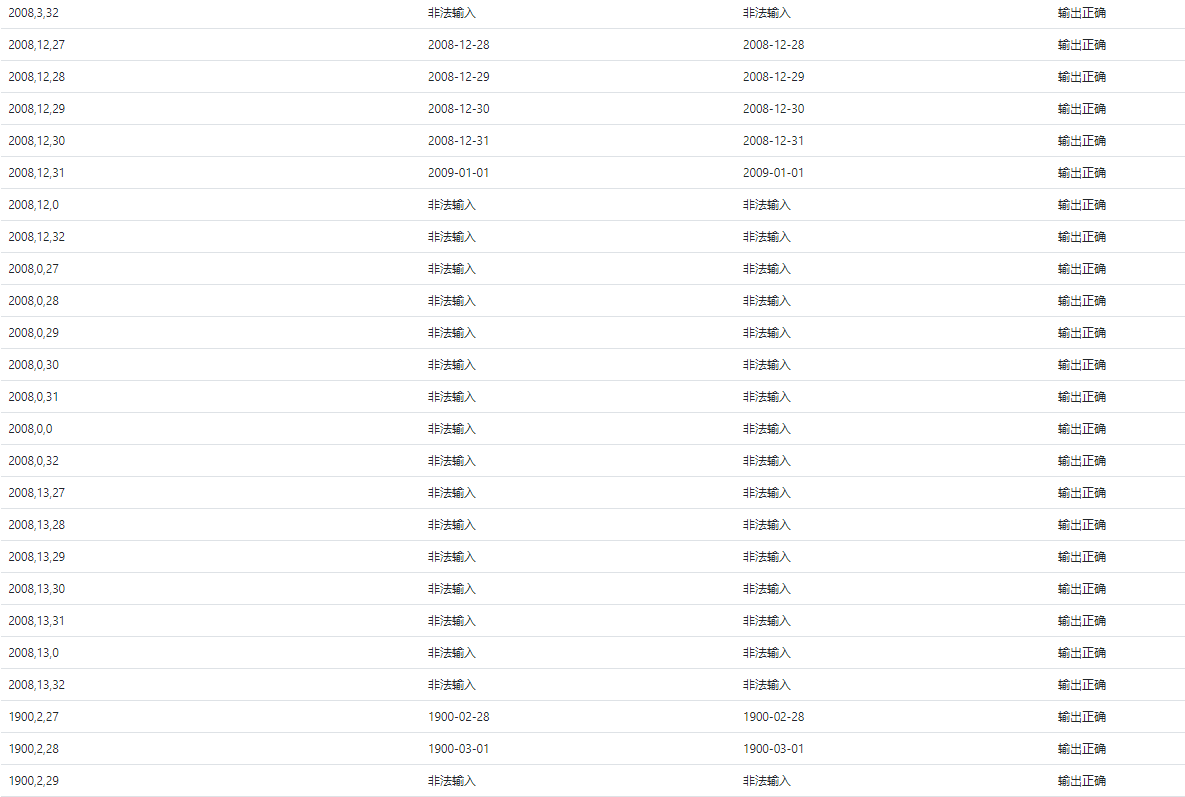


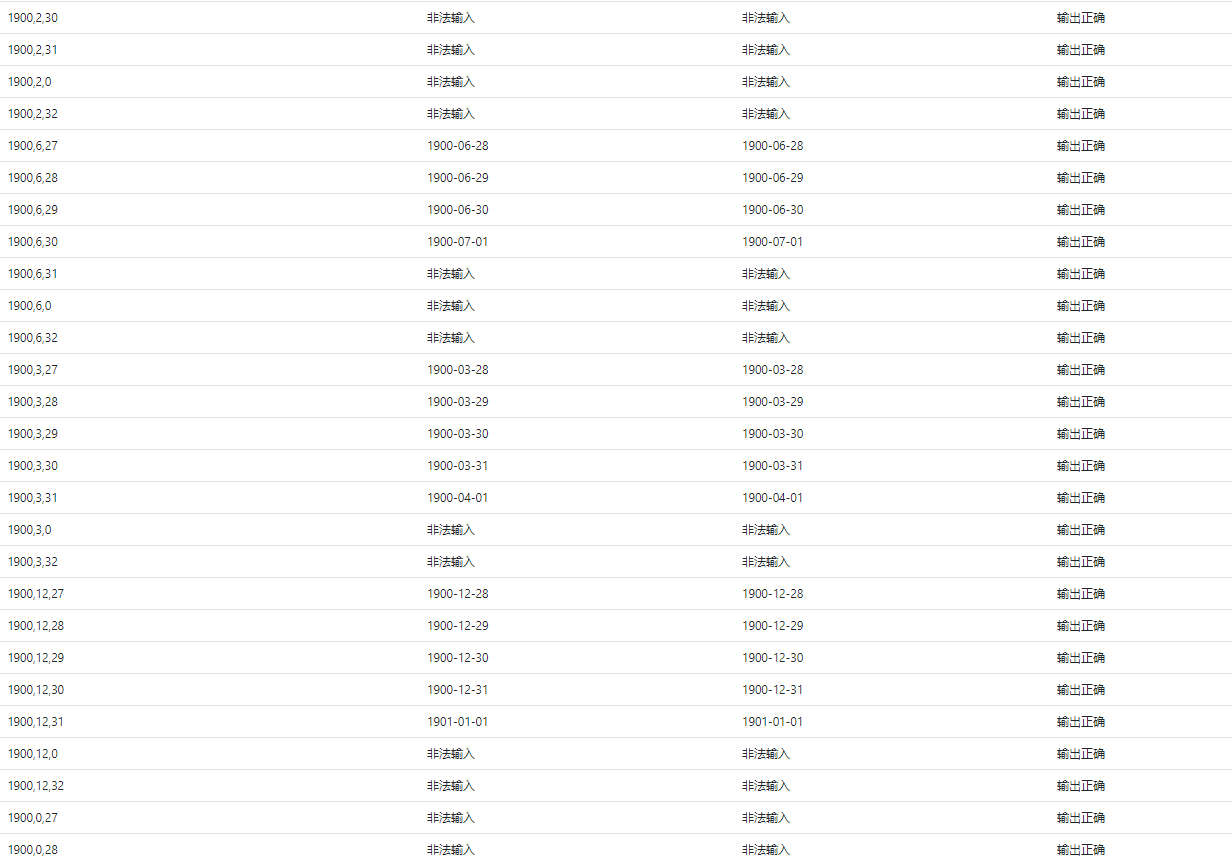
代码测试全部通过

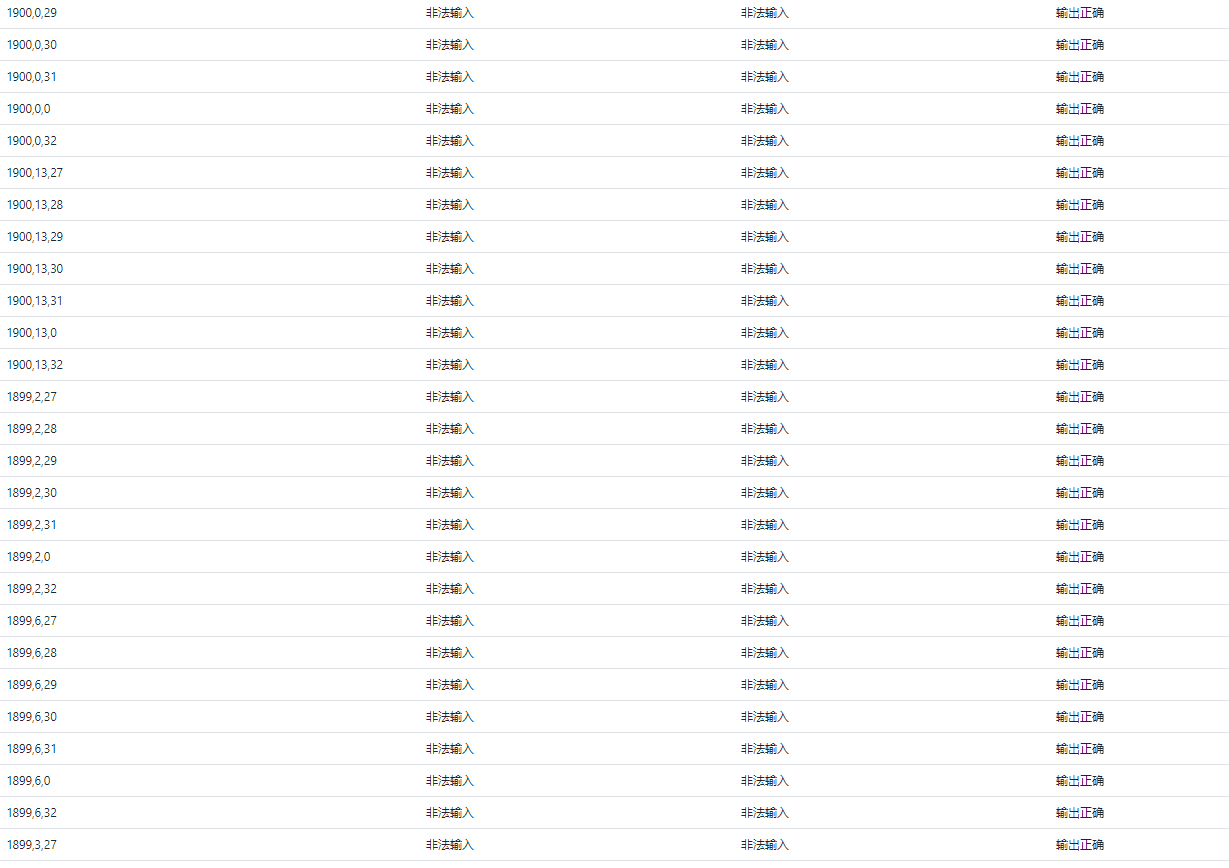
### 等价类法

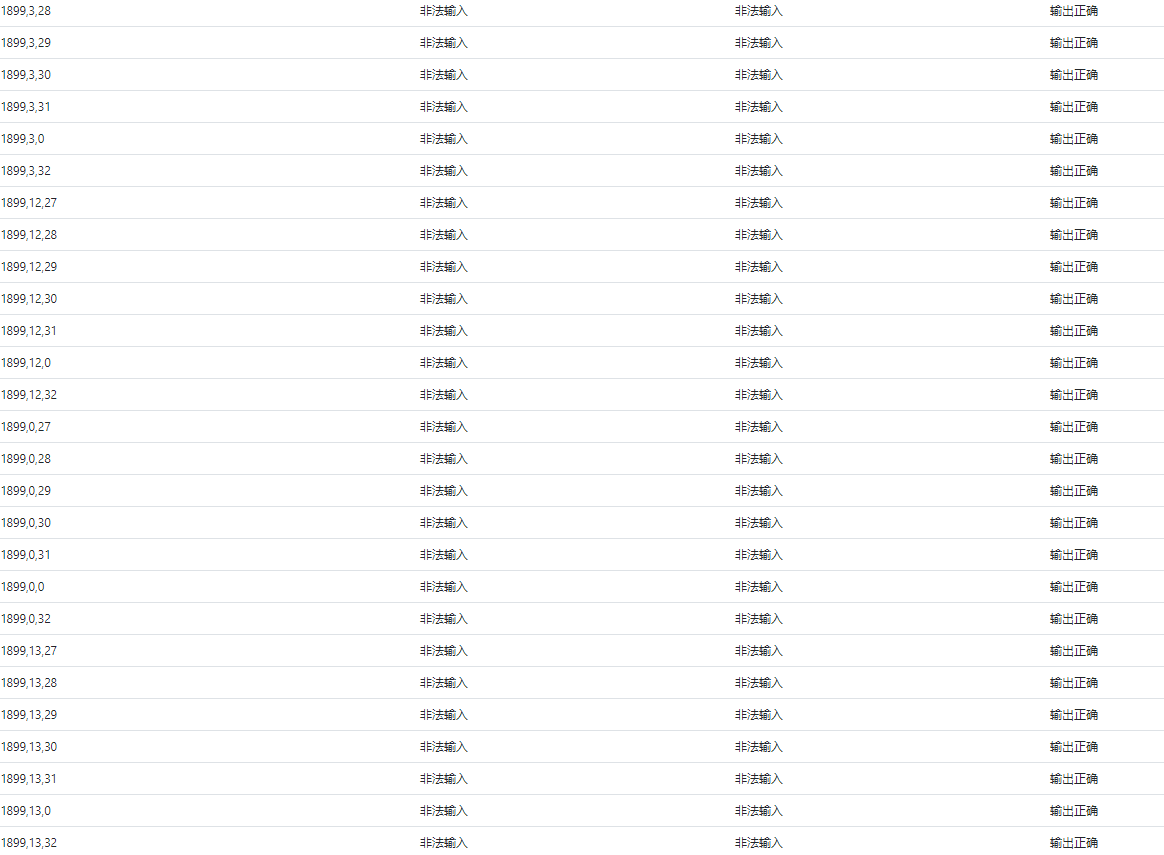


















代码测试全部通过

# 练习2—佣金问题问题

电脑销售系统，主机（25￥单位价格，每月最多销售的数量为70），显示器（30￥单位价格，每月最多销售数量为80），外设（45￥单位价格，每月最多销售的数量为90）；每个销售员每月至少销售一台完整的机器，当系统的主机这个变量接受到-1值的时候，系统自动统计该销售员本月的销售总额。当销售额小于等于1000（包括1000）按照10%提佣金，当销售额在1000-1800之间（包括1800）的时候按照15%提佣金，当销售额大于1800时按照20%提佣金。用边界值法设计测试用例。

## 问题分析

本题考虑的是一个电脑销售系统，在触发特定输入值时，统计销售员的销售总额及销售佣金，为了确保销售额的计算以及对应销售佣金按照相应等级的激素啊安是正确的，需要对该系统进行测试。该系统的输入有三种：主机、显示器、和外设的销售数量，根据各产品的单价乘以各产品销售数量，并进行累加即可计算出销售总额和相应的销售佣金。

对于各产品，它们的销售数量定义域为：

1. 主机
2. 显示器
3. 外设

其中，需要特别强调的是，按照题目的要求，每个销售员每月**至少**销售一台完整的机器，所以各产品的销售数量的下限为1，同时各产品销售数量的上限也是有相关要求的，所以当系统的销售数量超出规定时，也需要进行相应的提示报错，在这里我们假定参数错误时输出-1，这就要求我们在使用边界值法进行测试时，要进行健壮性的边界值法测试，以测试当输入值适当低于下限、适当高于上限时系统的容错能力。

根据健壮性边界分析法，它是基于“单缺陷”假设，即由于缺陷导致程序失效极少是由两个或者多个缺陷的同时作用而引起，而是单个变量在其边界值附近取值引起的。在每一个测试用例中，所有变量取正常值，而令其中一个变量取***min、min+、nom、max-、max*。**

因此，各变量的取值集合为：

1. 主机
2. 显示器
3. 外设

根据上述分析，使用健壮性边界分析法，会产生3\*6+1=19个测试用例，列表如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例编号 | 输入 | | | 预计输出(销售额) | 预计输出(佣金) |
| 主机 | 显示器 | 外设 |
|  | 0 | 40 | 45 | -1 | -1 |
|  | 35 | 0 | 45 | -1 | -1 |
|  | 35 | 40 | 0 | -1 | -1 |
|  | 1 | 40 | 45 | 3250 | 650.0 |
|  | 35 | 1 | 45 | 2930 | 586.0 |
|  | 35 | 40 | 1 | 2120 | 424.0 |
|  | 2 | 40 | 45 | 3275 | 655.0 |
|  | 35 | 2 | 45 | 2960 | 592.0 |
|  | 35 | 40 | 2 | 2165 | 433.0 |
|  | 35 | 40 | 45 | 4100 | 820.0 |
|  | 69 | 40 | 45 | 4950 | 990.0 |
|  | 35 | 78 | 45 | 5240 | 1048.0 |
|  | 35 | 40 | 88 | 6035 | 1207.0 |
|  | 70 | 40 | 45 | 4975 | 995.0 |
|  | 35 | 80 | 45 | 5300 | 1060.0 |
|  | 35 | 40 | 90 | 6125 | 1225.0 |
|  | 71 | 40 | 45 | -1 | -1 |
|  | 35 | 81 | 45 | -1 | -1 |
|  | 35 | 40 | 91 | -1 | -1 |

从上述19个用例的输入以及预期输出，我们可以看出，预期输出可分为两种情况，一种是由于输入参数不合法导致直接输出-1（代表参数不合法），另一种情况则是参数合法，但全部都归属于计算佣金点时的第三种情况，即销售额大于1800元的情况，而另外两个佣金点对应的销售额范围（当销售额小于等于1000（包括1000）按照10%提佣金，当销售额在1000-1800之间（包括1800）的时候按照15%提佣金），并没有得到测试。

因此，对于这道题而言，单纯对于输入变量进行边界值分析是不够的，还应当对于输出值（预期）应当使用等价类划分，结合边界值测试的方法。

依据上述分析，首先，我们将三种佣金提点对应的销售额范围划分等价类，可得到一下三个等价类：

* 1. 佣金点10% 0<=销售额<=1000
  2. 佣金点15% 1000<销售额<=1800
  3. 佣金点20% 1800<销售额

在每一个类中，根据销售额的范围，依照健壮性边界分析的方法来设计测试用例。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例编号 | 输入 | | | 预计输出(销售额) | 预计输出(佣金) |
| 主机 | 显示器 | 外设 |
|  | 0 | 0 | 0 | -1 | -1 |
|  | 6 | 4 | 6 | 540 | 54 |
|  | 1 | 1 | 1 | 100 | 10 |
|  | 10 | 10 | 10 | 1000 | 100 |
|  | 11 | 10 | 10 | 1025 | 153.75 |
|  | 15 | 15 | 15 | 1500 | 225 |
|  | 18 | 18 | 18 | 1800 | 270 |
|  | 19 | 18 | 18 | 1825 | 365 |

## 用例设计

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例编号 | 输入 | | | 预计输出(销售额) | 预计输出(佣金) |
| 主机 | 显示器 | 外设 |
|  | 0 | 40 | 45 | -1 | -1 |
|  | 35 | 0 | 45 | -1 | -1 |
|  | 35 | 40 | 0 | -1 | -1 |
|  | 1 | 40 | 45 | 3250 | 650.0 |
|  | 35 | 1 | 45 | 2930 | 586.0 |
|  | 35 | 40 | 1 | 2120 | 424.0 |
|  | 2 | 40 | 45 | 3275 | 655.0 |
|  | 35 | 2 | 45 | 2960 | 592.0 |
|  | 35 | 40 | 2 | 2165 | 433.0 |
|  | 35 | 40 | 45 | 4100 | 820.0 |
|  | 69 | 40 | 45 | 4950 | 990.0 |
|  | 35 | 78 | 45 | 5240 | 1048.0 |
|  | 35 | 40 | 88 | 6035 | 1207.0 |
|  | 70 | 40 | 45 | 4975 | 995.0 |
|  | 35 | 80 | 45 | 5300 | 1060.0 |
|  | 35 | 40 | 90 | 6125 | 1225.0 |
|  | 71 | 40 | 45 | -1 | -1 |
|  | 35 | 81 | 45 | -1 | -1 |
|  | 35 | 40 | 91 | -1 | -1 |
|  | 0 | 0 | 0 | -1 | -1 |
|  | 6 | 4 | 6 | 540 | 54 |
|  | 1 | 1 | 1 | 100 | 10 |
|  | 10 | 10 | 10 | 1000 | 100 |
|  | 11 | 10 | 10 | 1025 | 153.75 |
|  | 15 | 15 | 15 | 1500 | 225 |
|  | 18 | 18 | 18 | 1800 | 270 |
|  | 19 | 18 | 18 | 1825 | 365 |

## 测试代码设计

1. **public** **class** DeviceSales {
2. **public** **int** getHostNum() {
3. **return** hostNum;
4. }
6. **public** **int** getScreenNum() {
7. **return** screenNum;
8. }
10. **public** **int** getPeripheralNum() {
11. **return** peripheralNum;
12. }
14. **private** **int** hostNum;
15. **private** **int** screenNum;
16. **private** **int** peripheralNum;
18. **private** **int** hostPrice = 25;
19. **private** **int** screenPrice = 30;
20. **private** **int** peripheralPrice = 45;
22. **private** **int** maxHostNum = 70;
23. **private** **int** maxScreenNum = 80;
24. **private** **int** maxPeripheralNum = 90;
26. **private** **double** firstLevel = 0.1;
27. **private** **double** secondLevel = 0.15;
28. **private** **double** thirdLevel = 0.2;

31. **private** **void** setDeviceNum(**int** hostNum, **int** screenNum, **int** peripheralNum){
32. **this**.hostNum = hostNum;
33. **this**.screenNum = screenNum;
34. **this**.peripheralNum = peripheralNum;
35. }
37. **public** Integer getSaleCount(**int** hostNum, **int** screenNum, **int** peripheralNum){
38. setDeviceNum(hostNum, screenNum, peripheralNum);
39. **if**(!checkParams()){
40. **return** -1;
41. }
42. **int** saleCount = hostNum \* hostPrice + screenNum \* screenPrice + peripheralNum \* peripheralPrice;
44. **return** saleCount;
45. }
47. **private** **boolean** checkParams(){
48. **if** (hostNum<1||hostNum>maxHostNum){
49. **return** **false**;
50. }
51. **if** (screenNum<1||screenNum>maxScreenNum){
52. **return** **false**;
53. }
54. **if** (peripheralNum<1||peripheralNum>maxPeripheralNum){
55. **return** **false**;
56. }
57. **return** **true**;
58. }
60. **public** Double getBrokerage(**int** hostNum, **int** screenNum, **int** peripheralNum){
61. setDeviceNum(hostNum, screenNum, peripheralNum);
62. **if**(!checkParams()){
63. **return** -1.0;
64. }
65. var saleCount=getSaleCount(hostNum, screenNum, peripheralNum);
66. **if** (saleCount <= 1000 ){
67. **return** firstLevel \* saleCount;
68. }
69. **else** **if** (saleCount <= 1800){
70. **return** secondLevel \* saleCount;
71. }
72. **else** **return** thirdLevel \* saleCount;
73. }
74. }

## 测试用例执行情况

使用自制代码测试工具对编写代码加以测试，得到如下结果：

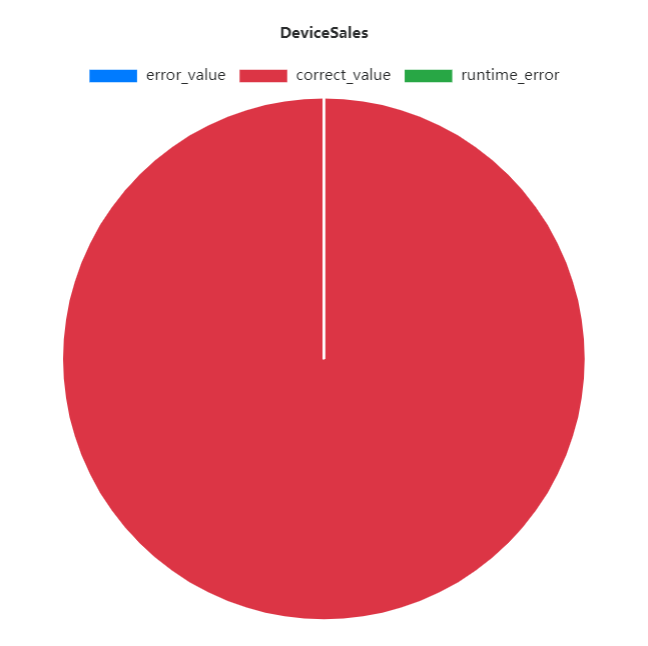
### 销售额输出测试

通过测试工具调用getSaleCount方法



结果如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 预计结果 | 实际输出 | 测试结果 |
| 0,40,45 | -1 | -1 | 输出正确 |
| 35,0,45 | -1 | -1 | 输出正确 |
| 35,40,0 | -1 | -1 | 输出正确 |
| 1,40,45 | 3250 | 3250 | 输出正确 |
| 35,1,45 | 2930 | 2930 | 输出正确 |
| 35,40,1 | 2120 | 2120 | 输出正确 |
| 2,40,45 | 3275 | 3275 | 输出正确 |
| 35,2,45 | 2960 | 2960 | 输出正确 |
| 35,40,2 | 2165 | 2165 | 输出正确 |
| 35,40,45 | 4100 | 4100 | 输出正确 |
| 69,40,45 | 4950 | 4950 | 输出正确 |
| 35,78,45 | 5240 | 5240 | 输出正确 |
| 35,40,88 | 6035 | 6035 | 输出正确 |
| 70,40,45 | 4975 | 4975 | 输出正确 |
| 35,80,45 | 5300 | 5300 | 输出正确 |
| 35,40,90 | 6125 | 6125 | 输出正确 |
| 71,40,45 | -1 | -1 | 输出正确 |
| 35,81,45 | -1 | -1 | 输出正确 |
| 35,40,91 | -1 | -1 | 输出正确 |
| 0,0,0 | -1 | -1 | 输出正确 |
| 6,4,6 | 540 | 540 | 输出正确 |
| 1,1,1 | 100 | 100 | 输出正确 |
| 10,10,10 | 1000 | 1000 | 输出正确 |
| 11,10,10 | 1025 | 1025 | 输出正确 |
| 15,15,15 | 1500 | 1500 | 输出正确 |
| 18,18,18 | 1800 | 1800 | 输出正确 |
| 19,18,18 | 1825 | 1825 | 输出正确 |



代码测试全部通过

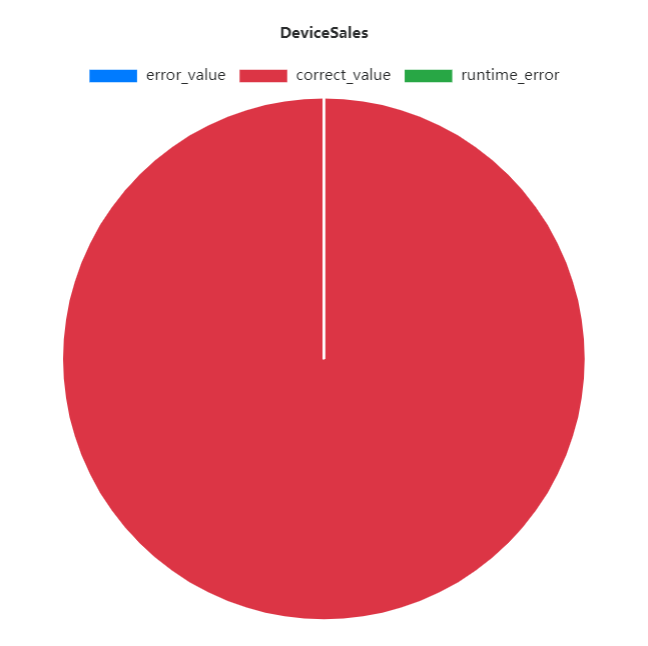
### 佣金输出测试

通过测试工具调用getBrokerage方法



结果如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 预计结果 | 实际输出 | 测试结果 |
| 0,40,45 | -1.0 | -1.0 | 输出正确 |
| 35,0,45 | -1.0 | -1.0 | 输出正确 |
| 35,40,0 | -1.0 | -1.0 | 输出正确 |
| 1,40,45 | 650.0 | 650.0 | 输出正确 |
| 35,1,45 | 586.0 | 586.0 | 输出正确 |
| 35,40,1 | 424.0 | 424.0 | 输出正确 |
| 2,40,45 | 655.0 | 655.0 | 输出正确 |
| 35,2,45 | 592.0 | 592.0 | 输出正确 |
| 35,40,2 | 433.0 | 433.0 | 输出正确 |
| 35,40,45 | 820.0 | 820.0 | 输出正确 |
| 69,40,45 | 990.0 | 990.0 | 输出正确 |
| 35,78,45 | 1048.0 | 1048.0 | 输出正确 |
| 35,40,88 | 1207.0 | 1207.0 | 输出正确 |
| 70,40,45 | 995.0 | 995.0 | 输出正确 |
| 35,80,45 | 1060.0 | 1060.0 | 输出正确 |
| 35,40,90 | 1225.0 | 1225.0 | 输出正确 |
| 71,40,45 | -1.0 | -1.0 | 输出正确 |
| 35,81,45 | -1.0 | -1.0 | 输出正确 |
| 35,40,91 | -1.0 | -1.0 | 输出正确 |
| 0,0,0 | -1.0 | -1.0 | 输出正确 |
| 6,4,6 | 54.0 | 54.0 | 输出正确 |
| 1,1,1 | 10.0 | 10.0 | 输出正确 |
| 10,10,10 | 100.0 | 100.0 | 输出正确 |
| 11,10,10 | 153.75 | 153.75 | 输出正确 |
| 15,15,15 | 225.0 | 225.0 | 输出正确 |
| 18,18,18 | 270.0 | 270.0 | 输出正确 |
| 19,18,18 | 365.0 | 365.0 | 输出正确 |



代码测试全部通过。

# 练习3—使用Junit实践并了解原理

## Junit介绍

Junit是一个Java语言的单元测试框架，它由 Kent Beck 和 Erich Gamma 建立，逐渐成为 xUnit 家族中最为成功的一个。 Junit有它自己的Junit扩展生态圈，多数Java的开发环境都已经集成了Junit作为单元测试的工具。在这里，一个单元可以是一个方法、类、包或者子系统。因此，单元测试是指对代码中的最小可测试单元进行检查和验证，以便确保它们正常工作。例如，我们可以给予一定的输入测试输出是否是所希望得到的结果。

Junit具有以下特点：

1. Junit提供了注释以及确定的测试方法；
2. Junit提供了断言用于测试预期的结果；
3. Junit测试优雅简洁不需要花费太多的时间；
4. Junit测试让大家可以更快地编写代码并且提高质量；
5. Junit测试可以组织成测试套件包含测试案例，甚至其他测试套件；
6. Junit显示测试进度，如果测试是没有问题条形是绿色的，测试失败则会变成红色；
7. Junit测试可以自动运行，检查自己的结果，并提供即时反馈，没有必要通过测试结果报告来手动梳理。

## Junit使用

Junit使用注解来完成多种测试功能，用户编写好测试代码后，使用注解对测试代码进行标识即可令Junit根据相应的注解完成相应的功能。

下面是Junit5中的一些常用注解：

@Test :表示方法是测试方法。但是与Junit4的@Test不同，他的职责非常单一不能声明任何属性，拓展的测试将会由Jupiter提供额外测试

@ParameterizedTest :表示方法是参数化测试，下方会有详细介绍

@RepeatedTest :表示方法可重复执行，下方会有详细介绍

@DisplayName :为测试类或者测试方法设置展示名称

@BeforeEach :表示在每个单元测试之前执行

@AfterEach :表示在每个单元测试之后执行

@BeforeAll :表示在所有单元测试之前执行

@AfterAll :表示在所有单元测试之后执行

@Tag :表示单元测试类别，类似于Junit4中的@Categories

@Disabled :表示测试类或测试方法不执行，类似于Junit4中的@Ignore

@Timeout :表示测试方法运行如果超过了指定时间将会返回错误

@ExtendWith :为测试类或测试方法提供扩展类引用

## Junit原理

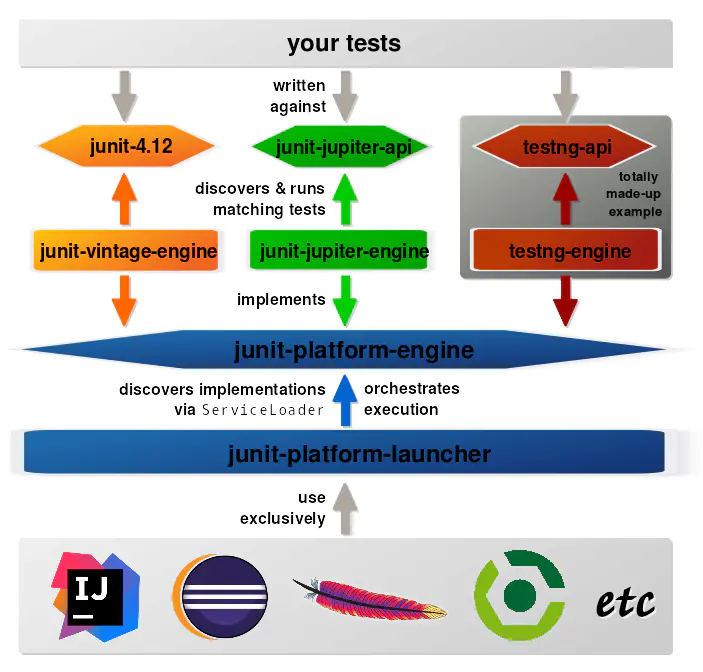
Junit5由Junit Platform、Junit Jupiter、Junit Vintage三个部分组成。

Junit Platform可以看作Junit提供的提供平台功能的模块，用户除了使用Junit测试引擎外，还可以接入其他测试引擎。

Junit Jupiter则是Junit5的核心，代表Junit5在Junit4上功能的演进。

Junit Vintage则是兼容性模块，用于兼容Junit4和Junit3。

下图为Junit5的系统架构：



第一层，用户基于引擎接口编写自动化测试

第二层，Junit 本身以及其他自动化测试框架实现的自动化引擎框架，Jupiter 和 Vintage 其实都是引擎实现，其他自动化测试框架如 TestNG 也可以通过实现引擎接入 Junit Platform

第三层，平台引擎，抽象上一层各种引擎实现的通用功能。

最底层，启动器。是面向各类 IDE 的接口，不同的 IDE 可以据此来实现对 Junit 的调度、运行。

## 项目Junit实践

在本项目组的软件测试项目中，单元测试和集成测试都需要基于Junit框架上来进行。

一个测试方法主要包括三部分：

1）Setup

2）执行操作

3）验证结果

在开始测试前，需要引入测试必须的测试框架：

SpringBoot 中的 pom.xml 文件需要添加的依赖：

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

为说明本项目的Junit实践，我们以集成测试中AdminOrderController的拒绝订单接口测试为例，其设计是基于等价类划分的方法，根据拒绝的订单ID是否存在于数据库中进行设计测试用例。

@Test

public void IT\_TD\_011\_004\_001\_002() throws Exception {

assertThrows(NestedServletException.class,()->mockMvc.perform(post("/rejectOrder.do").param("orderID","250")),"订单不存在！");

}

从测试代码中，我们可以看到，在测试方法的前面，需要使用注解@Test来说明这是一个方法，另外测试方法也不允许有任何的参数。在该测试用例中，是针对Controller的测试，所以需要使用mockMVC对象来模拟前端对该接口的请求，并且传递orderID的参数。由于传递的orderID在数据库中不存在，所以下层接口会抛出异常，这个时候，就需要使用Junit框架提供的Assert断言系列方法来对抛出的异常进行断言，如果确实抛出了异常，则说明该接口的设计是符合规约的。

# 练习4—以类作为单位如何定义driver和stub

在单元测试中，很多时候待测的模块不能独立运行，往往需要调用其他模块或被其他模块调用才能完成功能，因此我们在测试过程中往往需要额外构建辅助模块来模拟与待测模块相关联的模块，这些辅助模块可分为driver和stub两种模块。

## 以类为单位定义Driver

### Driver介绍

由于在单元测试中，待测模块往往是程序的最小单元，在没有主程序调用的情况下，无法独立运行。同时，即使主程序构建完成，若通过主程序调用待测模块，需要用户手动输入相应操作信息，并人工比对结果，十分繁琐并容易出错。

因此我们需要构建一个辅助模块来调用待测模块，并提供自动化测试的功能，这个辅助模块就称为Driver（驱动）。

Driver模块在程序中具有以下几点作用：

1. 接收测试输入
2. 对输入进行判断
3. 将输入传给待测单元，驱动被测单元执行
4. 接受被测单元执行结果，并对结果进行判断
5. 将判断结果作为用例执行结果并输出测试报告

### Driver实例

在本课程作业中，我们构造的代码往往是一个基本的类模块，需要其他程序调用，即需要Driver模块，在本课程作业中我们使用的测试工具就充当了Driver模块的作用，用于测试Java类中的相应方法，下面以该测试工具中调用待测模块的部分为例介绍Driver模块。

1. **public** **static** List<TestResult> testClass(String class\_name, String test\_case\_path, String method\_name) {
2. List<TestResult> results = **new** ArrayList<>();
3. Scanner scanner = **null**;
4. **try** {
5. // 从输入测试用例文件中获取待测模块输入
6. scanner = **new** Scanner(**new** File(test\_case\_path));
8. // 获取待测类
9. Class clazz = Class.forName(class\_name);
11. // 获取待测方法
12. var methods = clazz.getMethods();
13. Method method = **null**;
14. **for** (var item : methods) {
15. **if** (item.getName().equals(method\_name)) {
16. method = item;
17. }
18. }
20. **while** (scanner.hasNext()) {
21. var para\_num = method.getParameterCount();
22. List<Object> parameters = **new** ArrayList<>();
24. // 构造待测方法使用的参数列表
25. **for** (**int** i = 0; i < para\_num; ++i) {
26. parameters.add(convert\_value(scanner.next(), method.getParameterTypes()[i]));
27. }
28. Object return\_v;
29. var true\_result=method.getReturnType().getConstructor(String.**class**).newInstance(scanner.next());
30. **try**{
31. // 判断待测方法是否是static方法，从而以不同形式调用该方法
32. **if** (Modifier.isStatic(method.getModifiers())) {
33. return\_v = method.invoke(**null**, parameters.toArray());
34. } **else** {
35. return\_v = method.invoke(clazz.getDeclaredConstructor().newInstance(),parameters.toArray());
36. }
37. }**catch** (Exception e){
38. return\_v=**null**;
39. }

42. // 比对模块输出结果与预计结果的差异
43. // 并设置测试结果信息
44. TestResult testResult = **new** TestResult();
45. testResult.setParameters(parameters);
46. testResult.setResult(compareResult(return\_v,true\_result,method.getReturnType()));
47. testResult.setReal\_result(return\_v.toString());
48. testResult.setRight\_result(true\_result.toString());
49. testResult.setClass\_name(clazz.getSimpleName());
50. testResult.setMethod\_name(method\_name);
51. results.add(testResult);
52. }
54. } **catch** (Exception e) {
55. System.out.println(e.getMessage());
56. }**finally** {
57. scanner.close();
58. Runtime.getRuntime().gc();
59. }
60. **return** results;
61. }

该方法以待测模块类名，待测方法以及测试用例文件路径为参数。

下面是该方法的执行逻辑

1. 使用测试用例文件路径构造出输入器scanner
2. 使用反射机制通过类名和类方法名构造出待测方法并获取该方法相应信息
3. 根据待测方法的参数数目以及参数类型，从scanner中读出相应的参数，并转换为待测方法使用的参数
4. 使用获得的参数列表调用待测方法，将返回结果与从scanner中读出的预期结果进行对比
5. 构造测试结果报告

### Driver总结

总而言之，Driver模块的使命就是根据测试用例的设计去调用被测试模块，并且判断被测试模块的返回值是否与测试用例的预期结果相符。

## 以类为单位定义Stub

### Stub介绍

Stub模块用于代替待测模块调用的子模块，Stub模块可以进行少量的数据操作，不需要实现子模块的所有功能，但需要根据需要来实现或代替子模块的一部分内容，Stub模块是一次性模块，主要配合它的父模块工作。

### Stub实例

在本项目的单元测试中，Controller层的各模块需要调用下层Service层的接口提供的服务，由于单元测试是对于程序最小单元进行测试，需要对单元的上层以及下层调用进行隔离，所以对一下层Service层的调用需要进行插桩。为了体现Stub的作用，我们以IndexController层的String index(Model model)为例子进行说明。

单元测试中对该方法的测试代码如下：

1. @Test
2. **public** **void** return\_index\_html() **throws** Exception {
3. **int** venueID=1;
4. String venue\_name="venue";
5. String description="this is description";
6. **int** price=100;
7. String picture="";
8. String address="address";
9. String open\_time="08:00";
10. String close\_time="18:00";
11. Venue venue=**new** Venue(venueID,venue\_name,description,price,picture,address,open\_time,close\_time);
12. List<Venue> venues=**new** ArrayList<>();
13. venues.add(venue);
15. **int** id=1;
16. String title="title";
17. String content="this is content";
18. LocalDateTime ldt=LocalDateTime.now();
19. News news=**new** News(id,title,content,ldt);
20. List<News> news1=**new** ArrayList<>();
21. news1.add(news);
23. LocalDateTime ldt1=LocalDateTime.now().minusDays(1);
24. Message message=**new** Message(id,"user","this is a leave message", ldt1,1);
25. List<Message> messages=**new** ArrayList<>();
26. messages.add(message);
28. Pageable venue\_pageable= PageRequest.of(0,3, Sort.by("venueID").ascending());
29. Pageable news\_pageable= PageRequest.of(0,5, Sort.by("time").descending());
30. Pageable message\_pageable= PageRequest.of(0,5, Sort.by("time").descending());

33. when(venueService.findAll(any())).thenReturn(**new** PageImpl<>(venues,venue\_pageable,1));
34. when(newsService.findAll(any())).thenReturn(**new** PageImpl<>(news1,news\_pageable,1));
35. when(messageService.findPassState(any())).thenReturn(**new** PageImpl<>(messages,message\_pageable,1));
36. when(messageVoService.returnVo(any())).thenReturn(**null**);
38. ResultActions perform=mockMvc.perform(get("/index"));
39. perform.andExpect(status().isOk());
40. verify(venueService).findAll(any());
41. verify(newsService).findAll(any());
42. verify(messageService).findPassState(any());
43. verify(messageVoService).returnVo(any());
45. MvcResult mvcResult=mockMvc.perform(get("/index")).andReturn();
46. ModelAndView mv=mvcResult.getModelAndView();
47. assertAll("",()-> assertModelAttributeAvailable(mv,"user"),
48. ()-> assertModelAttributeAvailable(mv,"news\_list"),
49. ()->assertModelAttributeAvailable(mv,"venue\_list"),
50. ()->assertModelAttributeAvailable(mv,"message\_list"));
52. verify(venueService,times(2)).findAll(any());
53. verify(newsService,times(2)).findAll(any());
54. verify(messageService,times(2)).findPassState(any());
55. verify(messageVoService,times(2)).returnVo(any());
56. }

在IndexController中的该方法，是映射到了“/index”也就是主界面的URL上，当用户访问主界面时，就会调用该接口，该接口再去调用下层各Service提供的服务获取最新的故事列表、公告列表、共享空间列表、用户个数、订单个数等信息，保存在Model中。由于单元测试需要对下层调用的模块进行插桩，所以在本项目的单元测试中，使用了Mockito框架。

Mockito是mocking框架，它让你用简洁的API做测试。而且Mockito简单易学，它可读性强和验证语法简洁。Mockito 是一个针对 Java 的单元测试模拟框架，它与 EasyMock 和 jMock 很相似，都是为了简化单元测试过程中测试上下文 ( 或者称之为测试驱动函数以及桩函数 ) 的搭建而开发的工具。

测试代码的1—30行都是在模拟下层接口返回的数据，33—36行就是在使用Mockito框架进行插装的一个过程，这里使用了Mockito框架的Mockito.when(methodCall).thenReturn(value1).thenReturn(value2)方法，当调用methodCall时，返回预定义的内容，从而实现插桩，之后，可以通过verify来验证返回的Model中的内容，是否和插桩时预定义的内容一致。这样就实现了对于该接口的单元测试。

### Stub总结

桩模块的除了使得程序能够编译通过之外，还需要模拟返回被代替的模块的各种可能返回值。

# 练习5—找相关的开源测试工具，缺陷跟踪工具等

## 开源测试工具

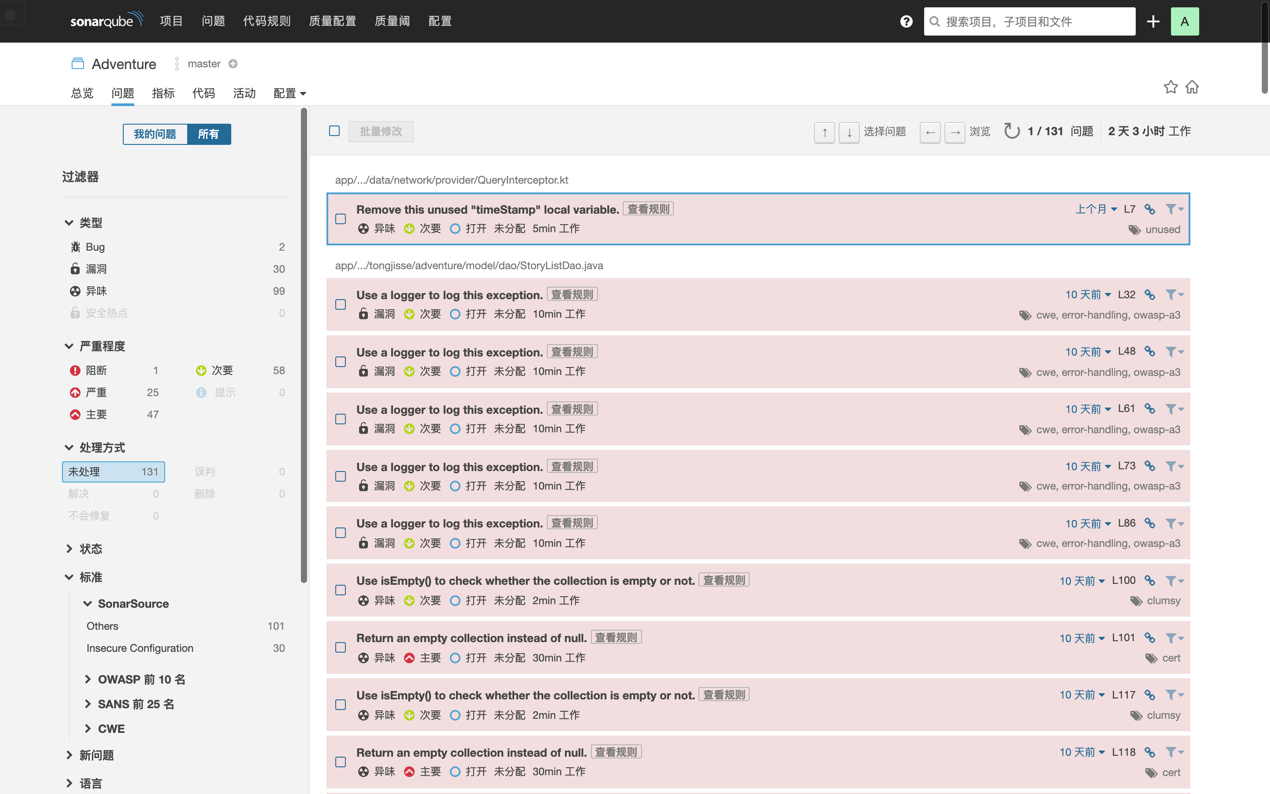
### 静态扫描工具

#### SonarQube

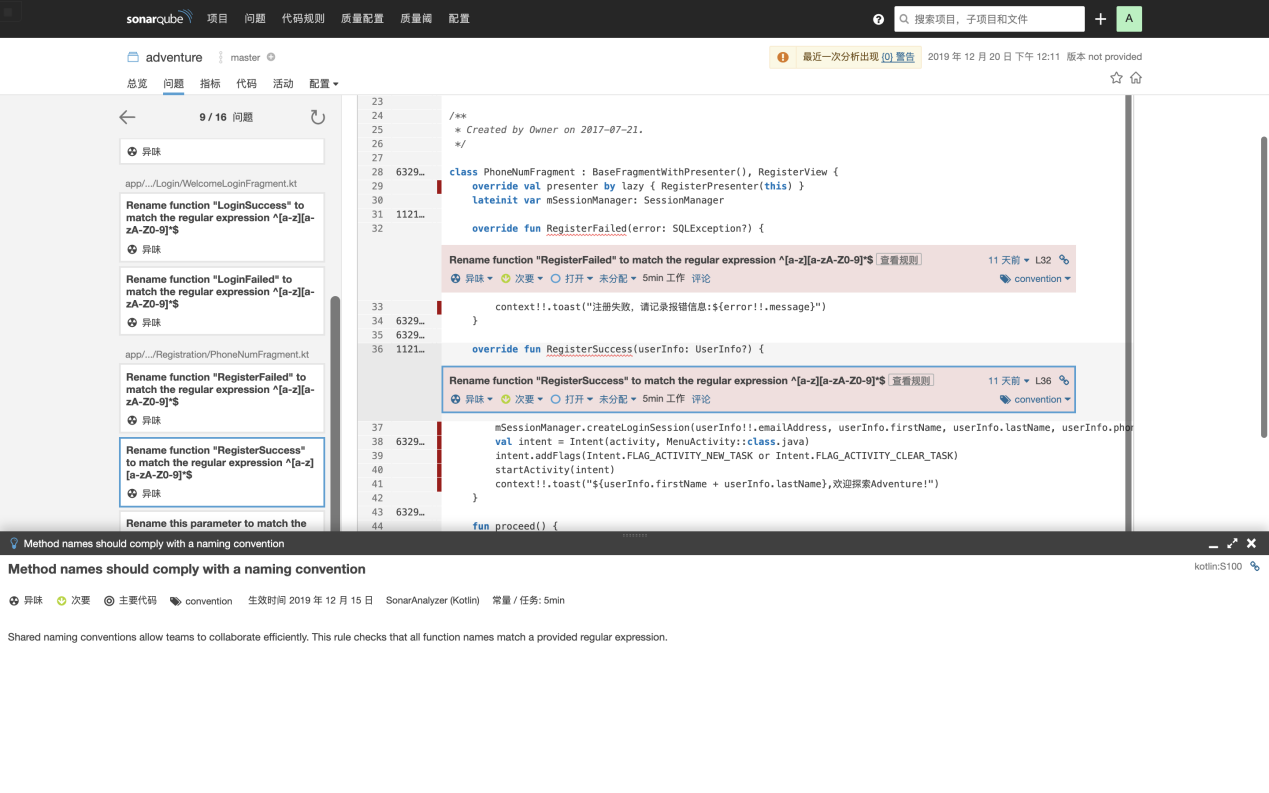
SonarQube 是一款用于代码质量管理的开源工具，它主要用于管理源代码的质量。 通过插件形式，可以支持众多计算机语言，比如 java, C#, go，C/C++, PL/SQL, Cobol, JavaScrip, Groovy 等。sonar可以通过PMD,CheckStyle,Findbugs等等代码规则检测工具来检测你的代码，帮助你发现代码的漏洞，Bug，异味等信息。

Sonar 不仅提供了对 IDE 的支持，可以在 Eclipse和 IntelliJ IDEA 这些工具里联机查看结果；同时 Sonar 还对大量的持续集成工具提供了接口支持，可以很方便地在持续集成中使用 Sonar。

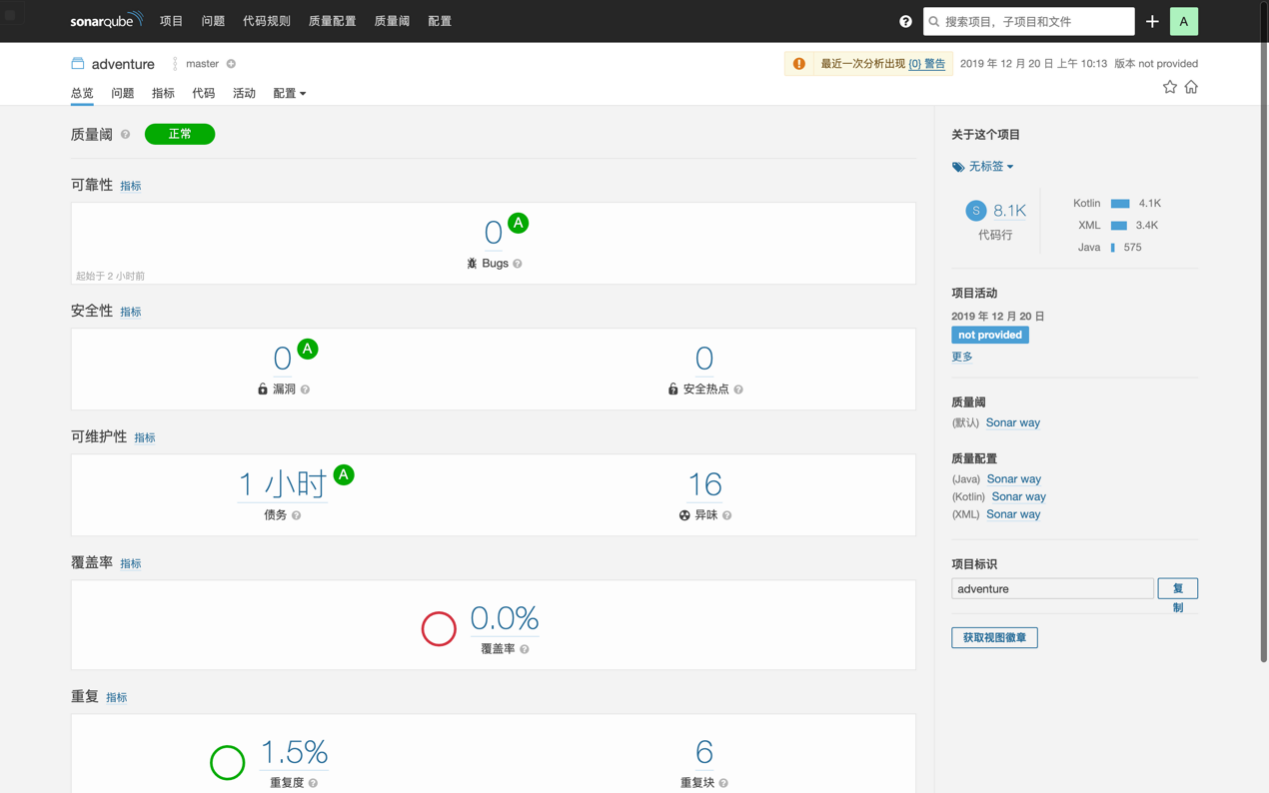
在本项目团队以前的项目中，曾经使用过SonarQube作为静态审查工具，结果如下图所示：

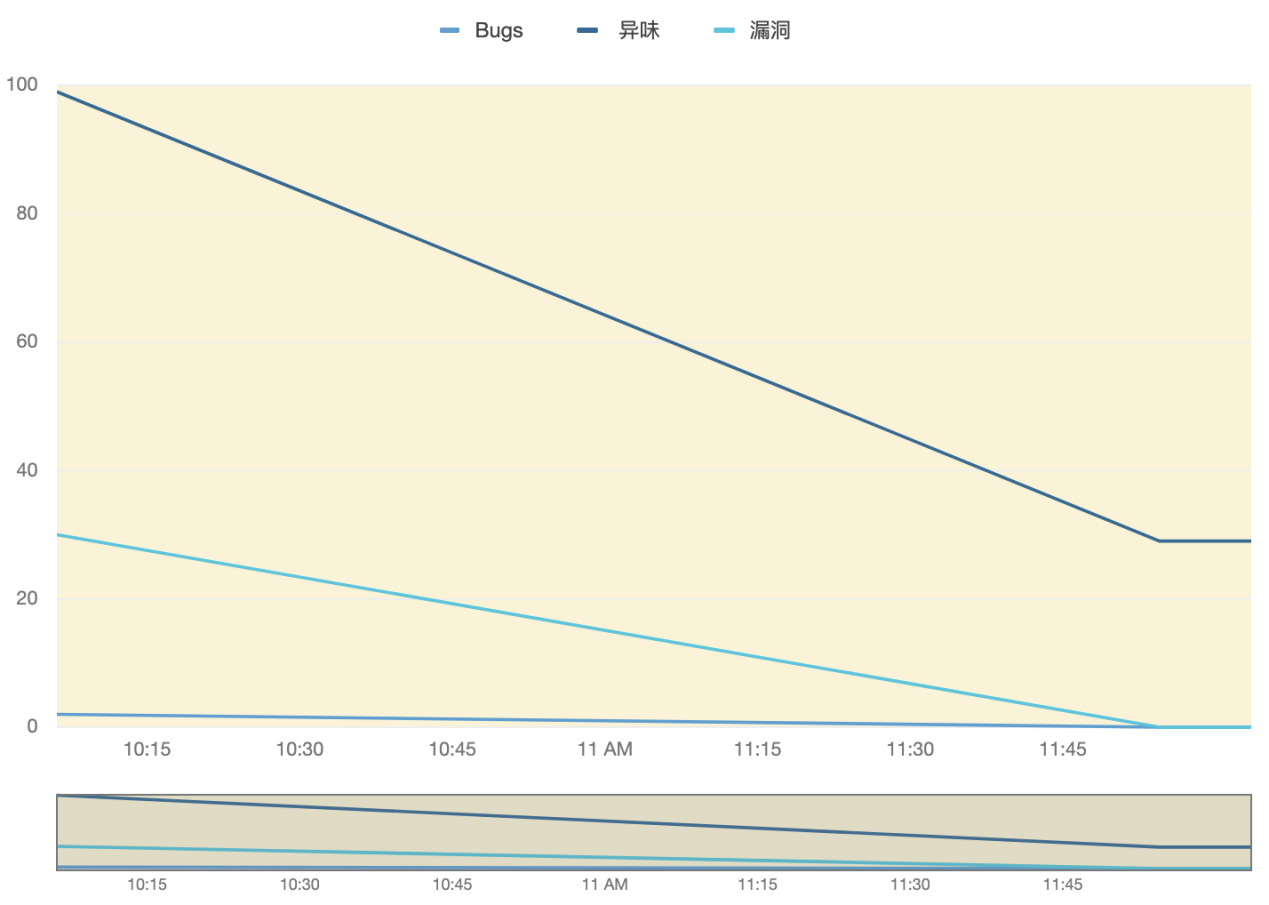


存在2个潜在bug、30个漏洞和99个异味。此时程序安全性和稳定性较差，代码可读性较低。因此，我们决定对代码进行重构，提高代码质量。



根据SonarQube给出的提示与解释规范代码，修复bug和漏洞。





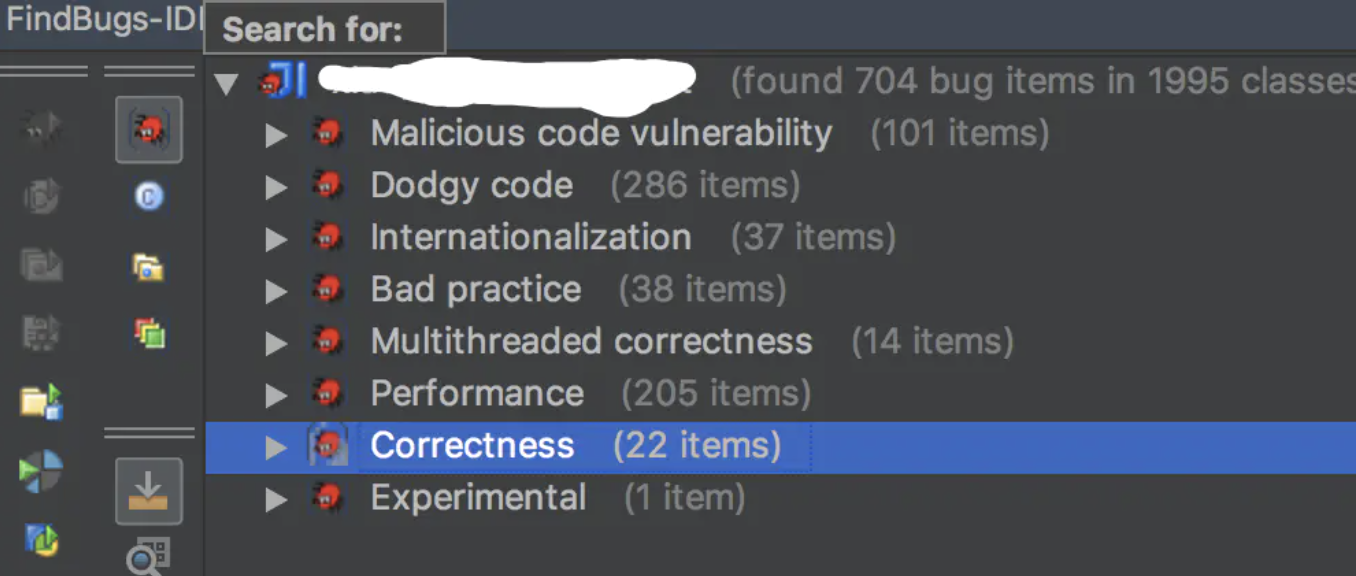
由此可见，在代码重构后，代码质量得到显著提升。

#### FindBugs

在本项目团队的软件测试项目中，使用的是FindBugs进行静态扫描。

FindBug 是一款开源的 Java 代码检查工具,遵循 GNU 公共许可协议。它可以检查 Java 类或者 JAR 文件,运行的是 Java 字节码而不是源码,检查原理是:将字节码与一组缺陷模式进行对比来发现可能存在的问题,这些问题包括空指针引用、无限递归循环、死锁等。检查的 bug 类型包括:

* Malicious code vulnerability：恶意代码
* Dodgy code：不符合规范的代码
* Internationalization：国际化相关问题，如错误的字符串转换;
* Bad practice：坏的实践:常见代码错误,序列化错误,用于静态代码检查时进行缺陷模式匹配;
* Multithreaded correctness：多线程的正确性:如多线程编程时常见的同步,线程调度问题;
* Performance：运行时性能问题，如由变量定义,方法调用导致的代码低效问题。
* Correctness：可能导致错误的代码,如空指针引用等;
* Experimental：可能受到的恶意攻击,如访问权限修饰符的定义等;
* Security：安全性



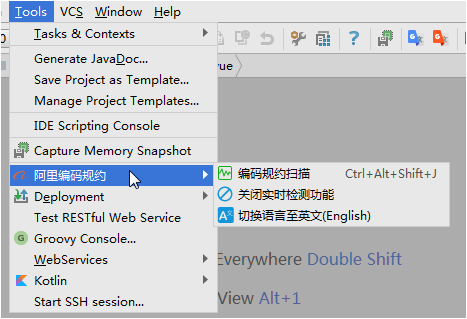
#### Alibaba Java Coding Guidelines

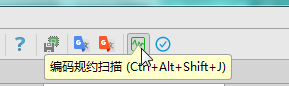
在本项目的静态测试中，还是用了Alibaba Java Coding Guidelines作为静态测试工具，它会根据你的代码情况给出致命错误，错误，警告等等级的提示让程序员快速定位问题代码进行修改.这个插件也是阿里开发近半年后在2017年10月14日上午9：00 阿里巴巴于在杭州云栖大会《研发效能峰会》上，正式发布《阿里巴巴Java开发手册》扫描插件。

如图操作,直接去IDEA插件里面搜索即可;



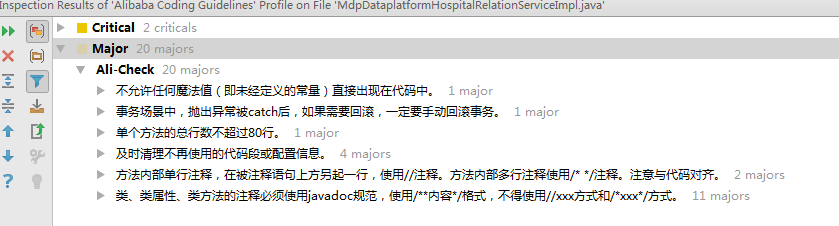
此时会看到以下效果





就可以检测代码了,但是如果电脑配置比较尴尬的话,最好是把实时检测关掉,要用的时候再去检测;

检测结果如下：



### 性能测试工具

#### Apache JMeter

Apache JMeter是Apache组织开发的基于Java的压力测试工具。用于对软件做压力测试，它最初被设计用于Web应用测试，但后来扩展到其他测试领域。 它可以用于测试静态和动态资源，例如静态文件、Java 小服务程序、CGI 脚本、Java 对象、数据库、FTP 服务器， 等等。JMeter 可以用于对服务器、网络或对象模拟巨大的负载，来自不同压力类别下测试它们的强度和分析整体性能。另外，JMeter能够对应用程序做功能/回归测试，通过创建带有断言的脚本来验证你的程序返回了你期望的结果。为了最大限度的灵活性，JMeter允许使用正则表达式创建断言。

Apache JMeter可以用于对静态的和动态的资源（文件，Servlet，Perl脚本，java 对象，数据库和查询，FTP服务器等等）的性能进行测试。它可以用于对服务器、网络或对象模拟繁重的负载来测试它们的强度或分析不同压力类型下的整体性能。你可以使用它做性能的图形分析或在大并发负载测试你的服务器/脚本/对象。

本项目中使用该工具进行性能测试。

#### OpenSTA

随着 Web 应用技术的发展和 Web 应用程序的迅速普及，Web 系统的性能和服务质量显得越来越重要。性能测试在软件的质量保证中起着越来越重要的作用。性能测试是通过自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试。性能测试是一种信息的收集和分析的过程，测试过程中通过收集的数据用来预测怎样的负载水平将耗尽系统资源。性能测试保证程序具有良好的性能，它考察在不同的用户负载下，Web 对用户请求做出的相应情况，以确保将来系统运行的安全性、可靠性和执行效率。性能测试的主要目的是为维护系统的性能找到有效地改善策略。

OpenSTA 是一个免费的、开源的的 web 性能测试工具。OpenSTA 是专用于 B/S 结构的、免费的性能测试工具。它的优点除了免费、源代码开放等优点外，还能对录制的测试脚本进行分析，并且按指定的语法进行编辑。软件测试工程师在录制完测试脚本后，只需要了解该脚本语言的特定语法知识，就可以对测试脚本进行编辑，以便于再次执行性能测试时获得所需要的参数，而后进行特定的性能指标分析。OpenSTA 以最简单的方式让大家对性能测试的原理有较深的了解，其较为丰富的图形化测试结果大大提高了测试报告的可阅读性。

### 自动化功能测试工具

#### Selenium

Selenium在软件测试中的主要应用就是可以完全模拟人对浏览器的操作，对数据进行获取动态数据由代码生成，在页面初始化的过程中是没有也无法获取，但是可以通过Selenium来进行获取。有些数据是需要登录以后才能获取的，比如说好友列表，评论，消费记录等等。登录后获取cookie才能进行以上的操作，但是使用Selenium以后，可以避免人工登录，只需要得到账号密码即可实现Selenium代替登录。

Selenium的特点：

1、由程序控制浏览器进行操作，而不是手动操作浏览器。

2、程序控制浏览器进行操作的时候，速度非常慢，所以要谨慎使用Selenium。

3、使用Selenium控制浏览器的时候，需要下载浏览器对应的驱动程序。

4、Selenium为开源、免费 ，但是更新速度没有浏览器快  。

在本团队课程项目的系统测试中，采用自动化与非自动化相结合的方式，在自动化测试中采用Selenium作为自动化功能测试工具。

#### Appium

appium 是一个自动化测试开源工具，支持 iOS 平台和 Android 平台上的原生应用，web应用和混合应用。

* “移动原生应用”是指那些用iOS或者 Android SDK 写的应用（Application简称app）。
* “移动web应用”是指使用移动浏览器访问的应用（appium支持iOS上的Safari和Android上的 Chrome）。
* “混合应用”是指原生代码封装网页视图——原生代码和 web 内容交互。比如，像 Phonegap，可以帮助开发者使用网页技术开发应用，然后用原生代码封装，这些就是混合应用。

重要的是，appium是一个跨平台的工具：它允许测试人员在不同的平台（iOS，Android）使用同一套API来写自动化测试脚本，这样大大增加了iOS和Android测试套件间代码的复用性。

appium类库封装了标准Selenium客户端类库，为用户提供所有常见的JSON格式selenium命令以及额外的移动设备控制相关的命令，如多点触控手势和屏幕朝向。

appium客户端类库实现了Mobile JSON Wire Protocol（一个标准协议的官方扩展草稿）和W3C WebDriver spec（一个传输不可预知的自动化协议，该协议定义了MultiAction 接口）的元素。

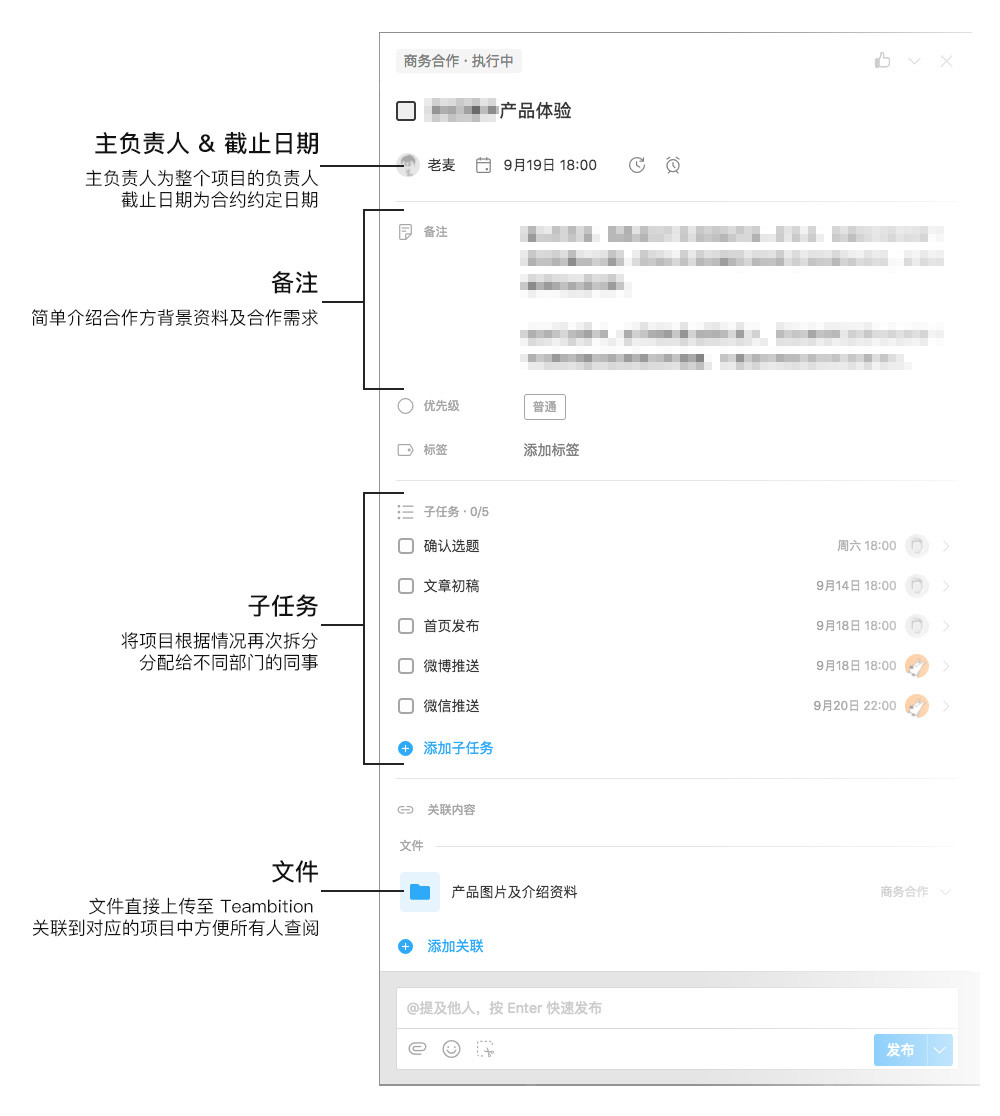
appium服务端定义了官方协议的扩展，为appium 用户提供了方便的接口来执行各种设备动作，例如在测试过程中安装/卸载App。这就是为什么我们需要appium特定的客户端，而不是通用的Selenium 客户端。当然，appium 客户端类库只是增加了一些功能，而实际上这些功能就是简单的扩展了Selenium 客户端，所以他们仍然可以用来运行通用的Selenium会话。

## 缺陷跟踪工具

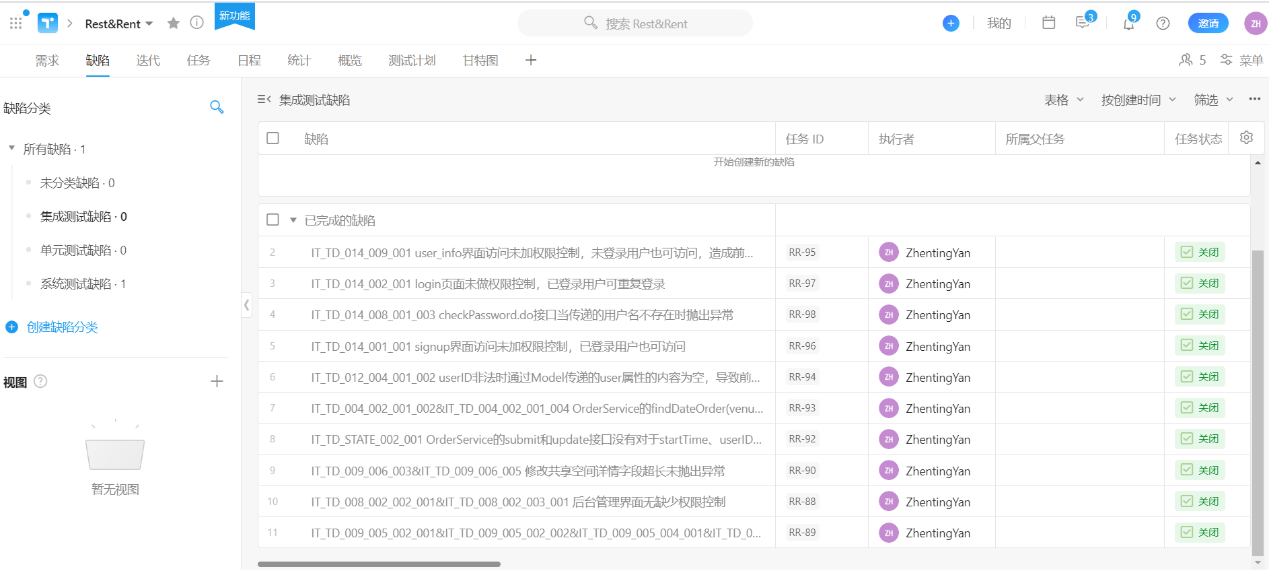
### Teambition

Teambition是一套基于「看板系统」的项目管理工具，它保持了「看板系统」在项目管理中直观易用的特性，同时也针对企业在项目管理中经常遇到的问题开发了很多特色功能，让整套系统变得更加符合使用习惯。

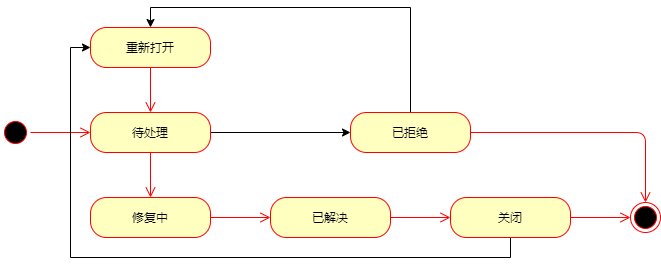
在本项目的开发阶段，使用Teambition进行项目管理，在Teambition中，一个项目的完成需要许多个子任务的完成，而一个子任务又可以细分为若干个子任务。



缺陷跟踪管理也是软件项目管理的一部分，因此在Teambition中也有缺陷模块，本项目的缺陷模块如下所示：



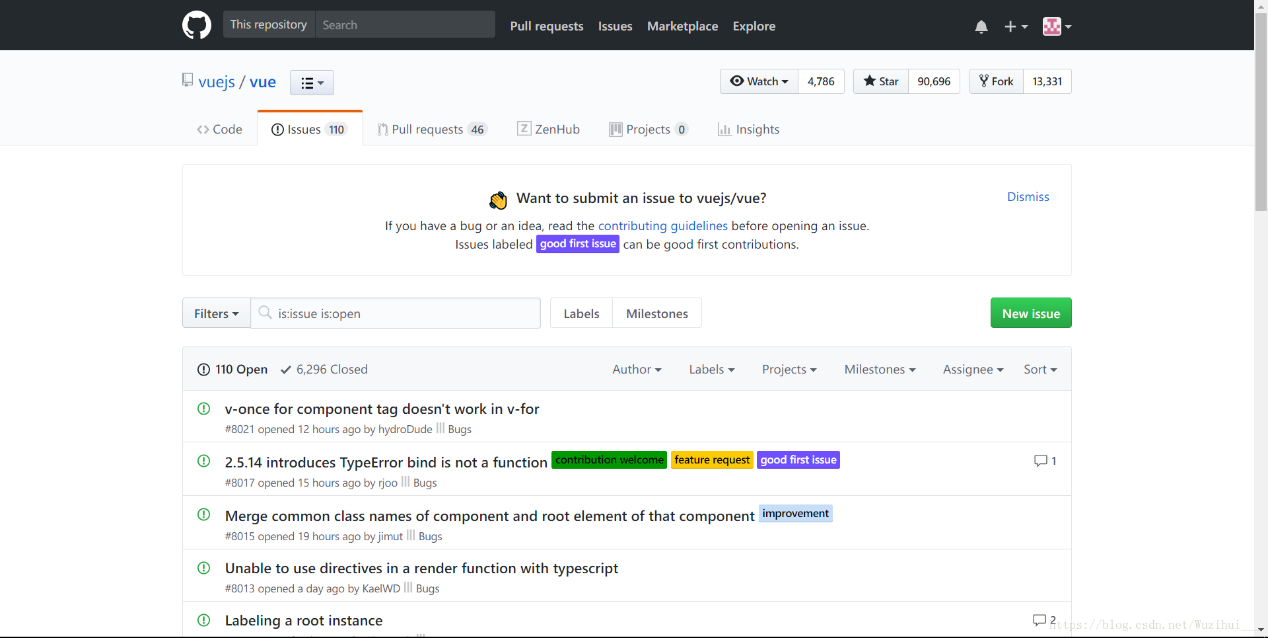
团队的成员可以根据测试情况创建缺陷任务，形成缺陷任务看板，在Teambition的缺陷模块中，一个缺陷的状态如下图所示：



首先，团队测试人员创建新的缺陷，新的缺陷为待处理状态，经项目经理确认评估以后，分配给对应的开发人员处理缺陷，开发人员可以根据实际情况经批准后拒绝缺陷，如果没有拒绝缺陷，则进入缺陷修复状态。在缺陷修复完成以后，开发人员将缺陷状态转为已解决状态，经过项目经理确认以后，缺陷进入关闭状态。当缺陷处于已拒绝或者关闭状态时，该缺陷均属于完成状态，如果后续发现缺陷未完全解决或者已拒绝修复的缺陷对实际业务产生重大影响时，经过项目经理批准可重新打开该缺陷任务。

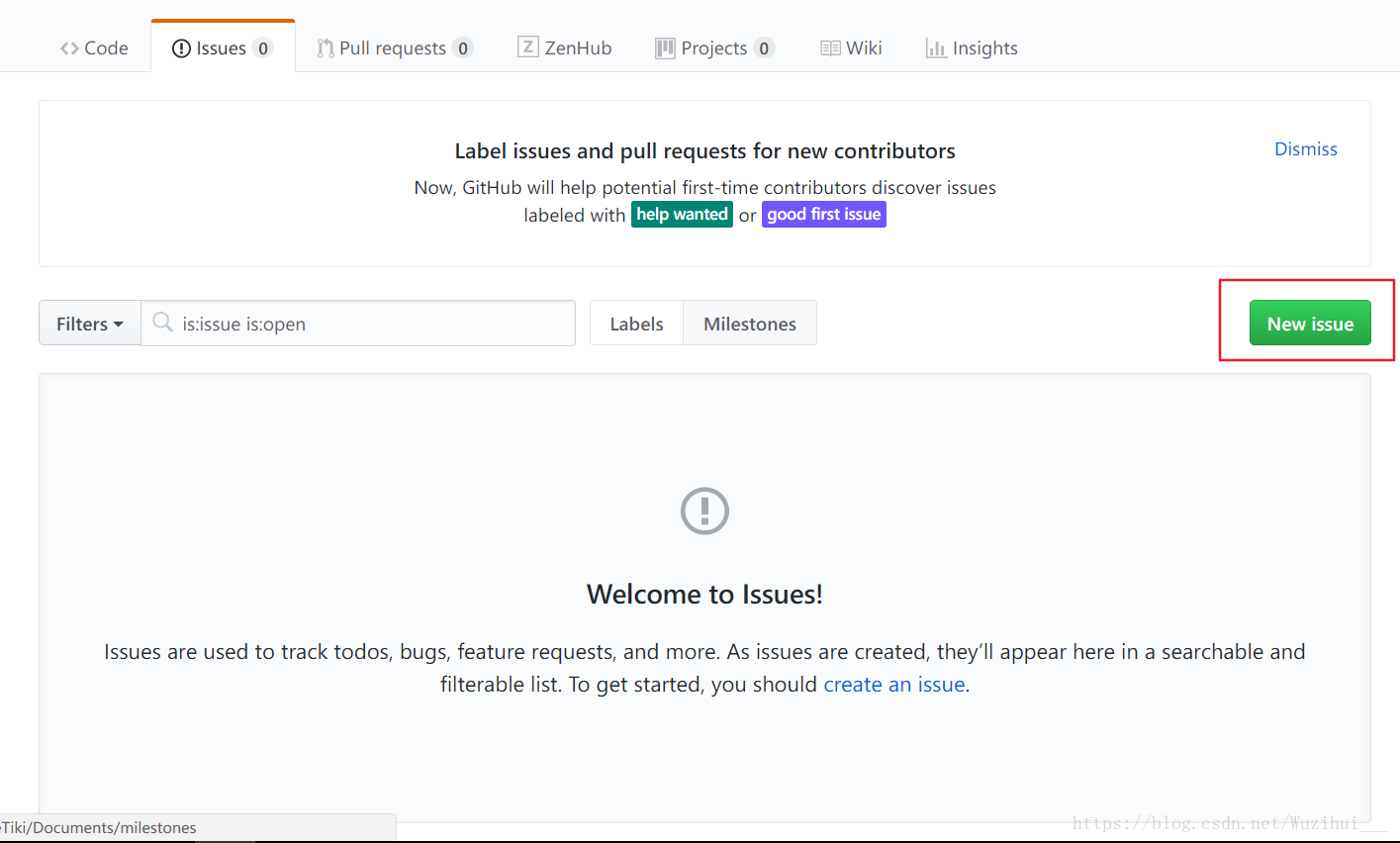
### Github Issues

Github为开发者提供了许多便于开发的功能，其中，issues功能被用来追踪各种想法，增强功能，任务，bug等。许多的项目通过centeral bug tracker收集用户的反馈，Github通过issues来实现同样的功能。



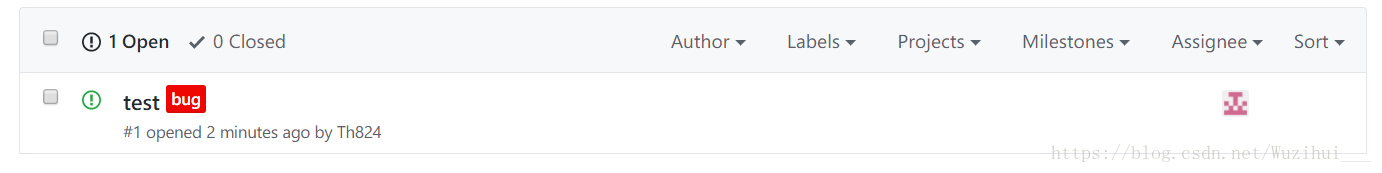
Issues经常被误解为一个报告软件bug的地方，其实不然。报告软件bug只是Issues众多功能中的一个。项目维护者可以通过Issues来组织需要完成的任务，例如增加新特性或者审计一个已经上线的功能。同时，还可以将Issues关联某些pull request，一旦合并了某个pull request，这个issue会被自动关闭。同时，你可以将issues添加到看板，也可以将看板的任务转化为issue。

可以通过选项卡中的issues选项，点击New issue即可建立一个新的issue。



如果你是项目的维护者，github还支持你如下的操作：

* 将一个issue分配给参与者
* 为一个issue打上标签
* 将一个issue添加到项目的看板
* 为一个issue关联到某个里程碑



# 练习6—设计transition tree 算法

构建一个打印机系统或复印件系统或ATM系统或一个复杂业务类的状态图或基于典型的前端界面的内容对象、交互、导航的状态图使用State Transition Testing 方法进行测试的分析和设计，并尝试用例的设计。

## 从状态图构建状态树算法

首先生成0-Switch树，其步骤为：

（1） 将初始状态或者开始状态作为状态转换树的根，根在整个状态转换树中的层次是1。

（2）假设当前生成状态转换树的层次为K，那么从左到右检查所有层次为K上的节点；将该节点对应的所有下一个可能的状态作为它的子节点，状态之间的转换作为两个状态的边。

（3）重复步骤2，直到一个位于层次K上的节点出现在层次J上，且J小于等于K，那么这个节点就成为最终的叶节点，而无需继续生成其子节点；或者节点的状态是结束状态，也不需要针对该节点继续进行状态转换。

简而言之，就是当一个节点没有子节点或这个节点在其之上的层中出现过，这个节点就是终止节点。

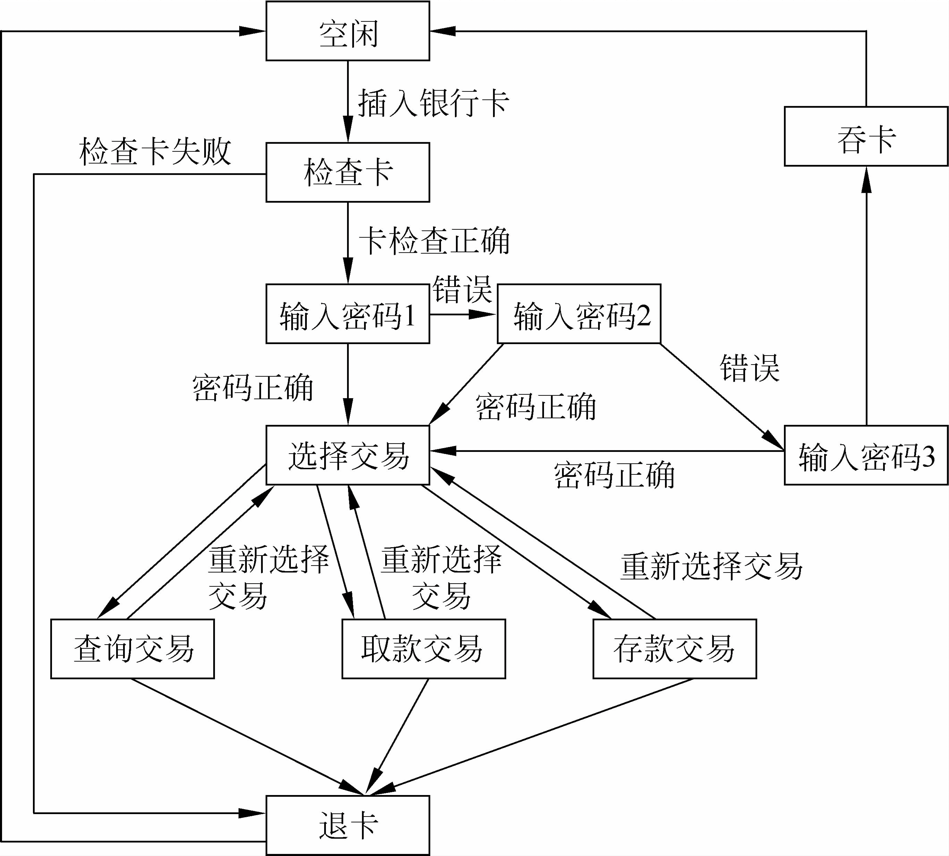
生成0-Switch树后，在其上扩充1个层次（即把0-Switch树中所有带有子节点的终止节点扩展，把他们的子节点作为终止节点。），这样就产生了1-Switch树。

同理，N-Switch树就是在（N-1）-Switch树的基础上扩充1个层次。

这样，我们可以知道，N-Switch树中生成的case中，能够覆盖全部长度为N+1的迁移条件的序列。这就使得我们能够量化的评估状态图测试的覆盖度和深度。

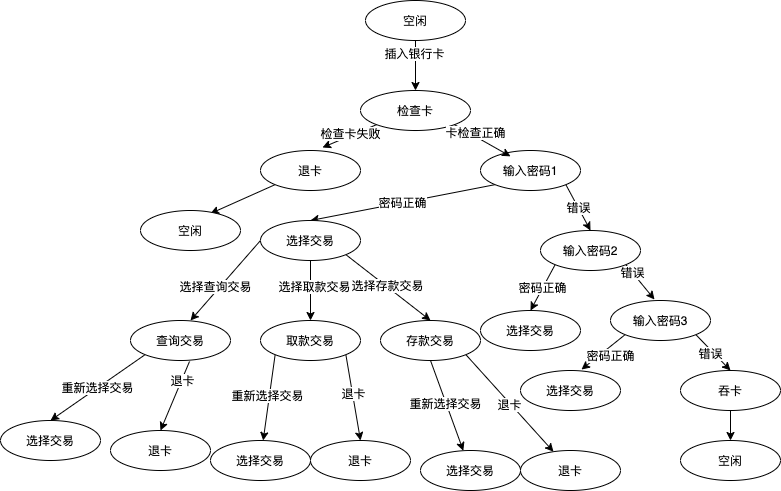
## ATM系统状态图

根据ATM的功能，我们初步得出以下的状态图：



## State Transition Tree构建

根据我们得到的状态图，我们构建出如下的State Transition Tree：



如图所示，构建的State Transition Tree涵盖了所有状态以及所有的状态转换过程。

## 测试用例构建

根据上述构建的State Transition Tree，我们可以进行测试用例的构建。

首先是定义测试用例的总体信息，如下表所示：

|  |  |
| --- | --- |
| 初始化状态： | ATM系统闲置 |
| 测试输入： | 用户对ATM机的操作 |
| 预期行为： | 用户完成一系列业务办理 |
| 结束状态： | 用户结束操作 |

根据状态图转换测试用例设计法的规则，状态图中的每个状态、每条边都应该至少覆盖一次，状态转换树中从根节点到任意终结点的路径都应该视作为一个测试用例。

### 测试数据准备

|  |  |  |
| --- | --- | --- |
| 卡号 | 预期密码 | 账户余额(元) |
| 201288345608(本系统的A银行账号) | 123456 | 8000 |
| 101288345609(非本系统的B银行账号) | 321456 | 2000 |

### 基于状态转换的测试分析与用例设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | | ATM\_TEST\_CASE\_FUN\_001 | | |
| 覆盖的状态转换路径 | | 空闲=>检查卡=>退卡=>空闲 | | |
| 输入 | | | | |
| 初始输入 | 预期交互输入、输出1 | | 预期交互输入、输出2 | 预期交互输入、输出3 |
| 插卡：101288345609 | 输入：无  输出：正在检查卡 | | 输入：无  输出：非本系统银行卡，交易失败，请取卡 |  |
| 最后预期输出 | | | | |
| 系统提示非本银行的银行卡，并退卡，显示交易失败，系统转到空闲状态 | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | | ATM\_TEST\_CASE\_FUN\_002 | | |
| 覆盖的状态转换路径 | | 空闲=>检查卡=>输入密码1=>选择交易=>查询交易=>退卡 | | |
| 输入 | | | | |
| 初始输入 | 预期交互输入、输出1 | | 预期交互输入、输出2 | 预期交互输入、输出3 |
| 插卡：  201288345608 | 输入：123456  输出：密码正确，正在进入交易界面 | | 输入：点击查询交易按钮  输出：显示查询交易界面，显示余额为8000元 | 输入：点击退卡按钮  输出：显示退卡界面 |
| 最后预期输出 | | | | |
| 系统退卡，转到空闲状态 | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | | ATM\_TEST\_CASE\_FUN\_003 | | |
| 覆盖的状态转换路径 | | 空闲=>检查卡=>输入密码1=>选择交易=>查询交易=>选择交易 | | |
| 输入 | | | | |
| 初始输入 | 预期交互输入、输出1 | | 预期交互输入、输出2 | 预期交互输入、输出3 |
| 插卡：  201288345608 | 输入：123456  输出：密码正确，正在进入交易界面 | | 输入：点击查询交易按钮  输出：显示查询交易界面 | 输入：点击继续交易按钮  输出：显示选择交易类型按钮 |
| 最后预期输出 | | | | |
| 系统跳转到选择交易界面 | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | | ATM\_TEST\_CASE\_FUN\_004 | | |
| 覆盖的状态转换路径 | | 空闲=>检查卡=>输入密码1=>选择交易=>取款交易=>退卡 | | |
| 输入 | | | | |
| 初始输入 | 预期交互输入、输出1 | | 预期交互输入、输出2 | 预期交互输入、输出3 |
| 插卡：  201288345608 | 输入：123456  输出：密码正确，正在进入交易界面 | | 输入：点击取款交易按钮  输出：显示查询交易界面 | 输入：点击退卡按钮  输出：显示退卡界面 |
| 最后预期输出 | | | | |
| 系统退卡，转到空闲状态 | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | | ATM\_TEST\_CASE\_FUN\_005 | | |
| 覆盖的状态转换路径 | | 空闲=>检查卡=>输入密码1=>选择交易=>取款交易=>选择交易 | | |
| 输入 | | | | |
| 初始输入 | 预期交互输入、输出1 | | 预期交互输入、输出2 | 预期交互输入、输出3 |
| 插卡：  201288345608 | 输入：123456  输出：密码正确，正在进入交易界面 | | 输入：点击取款交易按钮  输出：显示查询交易界面 | 输入：点击继续交易按钮  输出：显示选择交易类型按钮 |
| 最后预期输出 | | | | |
| 系统跳转到选择交易界面 | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | | ATM\_TEST\_CASE\_FUN\_006 | | |
| 覆盖的状态转换路径 | | 空闲=>检查卡=>输入密码1=>选择交易=>存款交易=>选择交易 | | |
| 输入 | | | | |
| 初始输入 | 预期交互输入、输出1 | | 预期交互输入、输出2 | 预期交互输入、输出3 |
| 插卡：  201288345608 | 输入：123456  输出：密码正确，正在进入交易界面 | | 输入：点击存款交易按钮  输出：显示查询交易界面 | 输入：点击退卡按钮  输出：显示退卡界面 |
| 最后预期输出 | | | | |
| 系统退卡，转到空闲状态 | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | | ATM\_TEST\_CASE\_FUN\_007 | | |
| 覆盖的状态转换路径 | | 空闲=>检查卡=>输入密码1=>选择交易=>存款交易=>选择交易 | | |
| 输入 | | | | |
| 初始输入 | 预期交互输入、输出1 | | 预期交互输入、输出2 | 预期交互输入、输出3 |
| 插卡：  201288345608 | 输入：123456  输出：密码正确，正在进入交易界面 | | 输入：点击存款交易按钮  输出：显示查询交易界面 | 输入：点击继续交易按钮  输出：显示选择交易类型按钮 |
| 最后预期输出 | | | | |
| 系统跳转到选择交易界面 | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | | ATM\_TEST\_CASE\_FUN\_008 | | |
| 覆盖的状态转换路径 | | 空闲=>检查卡=>输入密码1=>输入密码2=>选择交易 | | |
| 输入 | | | | |
| 初始输入 | 预期交互输入、输出1 | | 预期交互输入、输出2 | 预期交互输入、输出3 |
| 插卡：  201288345608 | 输入：654321  输出：密码不正确，再输错两次即吞卡 | | 输入：123456  输出：密码正确，正在进入交易界面 |  |
| 最后预期输出 | | | | |
| 系统进入选择交易界面 | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | | ATM\_TEST\_CASE\_FUN\_009 | | |
| 覆盖的状态转换路径 | | 空闲=>检查卡=>输入密码1=>输入密码2=>输入密码3=>选择交易 | | |
| 输入 | | | | |
| 初始输入 | 预期交互输入、输出1 | | 预期交互输入、输出2 | 预期交互输入、输出3 |
| 插卡：  201288345608 | 输入：654321  输出：密码不正确，再输错两次即吞卡 | | 输入：653421  输出：密码不正确，再输错一次即吞卡 | 输入：123456  输出：密码正确，正在进入交易界面 |
|  |  | |  |  |
| 最后预期输出 | | | | |
| 系统进入选择交易界面 | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | | ATM\_TEST\_CASE\_FUN\_010 | | |
| 覆盖的状态转换路径 | | 空闲=>检查卡=>输入密码1=>输入密码2=>输入密码3=>吞卡=>空闲 | | |
| 输入 | | | | |
| 初始输入 | 预期交互输入、输出1 | | 预期交互输入、输出2 | 预期交互输入、输出3 |
| 插卡：  201288345608 | 输入：654321  输出：密码不正确，再输错两次即吞卡 | | 输入：653421  输出：密码不正确，再输错一次即吞卡 | 输入：666666  输出：密码验证失败，错误次数已达3次，系统将吞卡，请联系发卡行。 |
| 最后预期输出 | | | | |
| 系统吞卡，进入吞卡提示界面，然后进入空闲状态 | | | | |

# 练习7—电信收费问题

研究一个与我们的生活息息相关的电信收费问题系统，需求描述如下：

每月的电话总费用=基本月租费+折扣后的实际的通话费，如果没有折扣则按实际通话费计算，基本月租费为25元，每分钟通话费为0.15元。

实际通话费是否有折扣与当月的通话时间（分钟）和本年度至本月的累计未按时缴费的次数有关。跨年度未交费与折扣无关，但跨年未交部分每月需要交付总额5%的滞纳金。

当月的通话分钟数和折扣比例及本年度未按时缴费次数之间有直接的对应关系，如果本年度的未按时缴费的次数超过本月通话时间所对应的容许值则免于折扣，并按实际的通话费计算。

电话费的收取，采用在线支付，支付方式为：支付宝或银行卡（开发简单的模拟子系统），支付后打印支付成功或不成功清单。

通话时间和折扣比例及未按时缴费次数的关系为：

|  |  |  |
| --- | --- | --- |
| 本月通话的分钟数 | 通话时间段的最大容许不按时缴费次数 | 通话时间段的折扣率 |
| 0＜ 通话时间≤60 | 1 | 1.0％ |
| 60＜ 通话时间≤120 | 2 | 1.5％ |
| 120＜ 通话时间≤180 | 3 | 2.0％ |
| 180＜ 通话时间≤300 | 3 | 2.5％ |
| 300＜ 通话时间 | 6 | 3.0％ |

分别用边界值、等价类和决策表设计测试用例，并综合分析得出合理的测试用例集。

## 练习分析

在本题中，通话时间段的折扣率应当是我们所需要验证的结果，本月通话的分钟数和通话时间段的最大容许不按时缴费次数应当是我们所需要的设计的测试用例中的输入变量。

在这里，我们对各变量的代表符号定义如下：

T：本月通话的分钟数

C：通话时间段的不按时缴费次数。

Dis：通话时间段的折扣率

LastYearCost：上一年未交费用

题目中虽然没有对于这些变量有明确的取值边界的限制，但是实际上这些变量都有隐含的取值范围：

T最大为31天\*24小时\*60分钟 = 44640分钟

本年度的通话时间段的不按时缴费次数C最大为11次（隐含条件）。

在此问题中，我们编写的程序作为底层的话费计算工具，不考虑当前月份的影响，对于跨年未缴费用，考虑到现实业务逻辑，若一直不缴费，则停止号码使用，因此最大值应为第12月的话费最大值，即25+44640\*0.15=6721元。

即每月应缴费用为25+T\*0.15\*(1-Dis)+LastYearCost\*0.05，若输入参数错误，则输出-1，同时我们规定，当月通话时长为0时，仍需缴纳相应月租费和滞纳金。

### 决策表法

根据上述对本题的分析，我们可以设计出如下所示的决策表：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| T | 0<T<=60 | | | | 60<T<=120 | | | | 120<T<=180 | | | | 180<T<=300 | | | | T>300 | | | |
| C | C<=1 | | C>1 | | C<=2 | | C>2 | | C<=3 | | C>3 | | C<=3 | | C>3 | | C<=6 | | C>6 | |
| 跨年未缴 | 有 | 无 | 有 | 无 | 有 | 无 | 有 | 无 | 有 | 无 | 有 | 无 | 有 | 无 | 有 | 无 | 有 | 无 | 有 | 无 |
| Dis = 0 |  | | X | |  | | X | |  | | X | |  | | X | |  | | X | |
| Dis=1.0 | X | |  | |  | |  | |  | |  | |  | |  | |  | |  | |
| Dis = 1.5 |  | |  | | X | |  | |  | |  | |  | |  | |  | |  | |
| Dis = 2.0 |  | |  | |  | |  | | X | |  | |  | |  | |  | |  | |
| Dis = 2.5 |  | |  | |  | |  | |  | |  | | X | |  | |  | |  | |
| Dis = 3.0 |  | |  | |  | |  | |  | |  | |  | |  | | X | |  | |
| 缴纳滞纳金 | X |  | X |  | X |  | X |  | X |  | X |  | X |  | X |  | X |  | X |  |

其中，每一列都可以作为一个测试用例，因此，针对于本题的决策表法，我们可以设计出如下表所示的测试用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 本月通话时间 | 未按时缴费次数 | 跨年未缴费用 | 预计输出 |
|  | 30 | 1 | 6721 | 365.505 |
|  | 30 | 2 | 6721 | 365.55 |
|  | 90 | 2 | 6721 | 374.3475 |
|  | 90 | 3 | 6721 | 374.55 |
|  | 150 | 3 | 6721 | 383.1 |
|  | 150 | 4 | 6721 | 383.55 |
|  | 210 | 3 | 6721 | 391.7625 |
|  | 210 | 4 | 6721 | 392.55 |
|  | 350 | 6 | 6721 | 411.975 |
|  | 350 | 11 | 6721 | 413.55 |
|  | 30 | 1 | 0 | 29.455 |
|  | 30 | 2 | 0 | 29.5 |
|  | 90 | 2 | 0 | 38.2975 |
|  | 90 | 3 | 0 | 38.5 |
|  | 150 | 3 | 0 | 47.05 |
|  | 150 | 4 | 0 | 47.5 |
|  | 210 | 3 | 0 | 55.7125 |
|  | 210 | 4 | 0 | 56.5 |
|  | 350 | 6 | 0 | 75.925 |
|  | 350 | 11 | 0 | 77.5 |

### 等价类法

根据输入信息，我们可以针对三个不同变量构造出相应的等价类。

对于变量T来说，划分出的等价类为：

T1 = { T | 0=<T<=60 }

T2 = { T | 60<T<=120 }

T3 = { T | 120<T<=180 }

T4 = { T | 180<T<=300 }

T5 = { T | 300<T<=44640 }

无效等价类为

T6 = { T | T<0 }

T7 = { T | 44640<T }

对于变量C来说，划分出的等价类为：

C1 = { C | 0<=C<=1 }

C2 = { C | 1<C<=2 }

C3 = { C | 2<C<3 }

C4 = { C | 3<C<=6 }

C5 = { C | 6<C<=11 }

无效等价类为

C6 = { C | C<0 }

C7 = { C | 11<C }

对于变量LastYearCost，有且只有一个等价类：

LastYearCost1 = { LastYearCost | 0=<LastYearCost<=6721 }

无效等价类为：

LastYearCost2 = { LastYearCost | LastYearCost<0 }

LastYearCost3 = { LastYearCost | 6721<LastYearCost }

由于本问题输出与多个变量的取值有关，因此应该基于“多缺陷”假设使用强健壮性等价类的划分，构成测试样例如下所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 本月通话时间 | 未按时缴费次数 | 跨年未缴费用 | 预计输出 |
|  | -1 | -1 | -1 | -1 |
|  | -1 | -1 | 1000 | -1 |
|  | -1 | -1 | 6722 | -1 |
|  | -1 | 0 | -1 | -1 |
|  | -1 | 0 | 1000 | -1 |
|  | -1 | 0 | 6722 | -1 |
|  | -1 | 2 | -1 | -1 |
|  | -1 | 2 | 1000 | -1 |
|  | -1 | 2 | 6722 | -1 |
|  | -1 | 3 | -1 | -1 |
|  | -1 | 3 | 1000 | -1 |
|  | -1 | 3 | 6722 | -1 |
|  | -1 | 6 | -1 | -1 |
|  | -1 | 6 | 1000 | -1 |
|  | -1 | 6 | 6722 | -1 |
|  | -1 | 11 | -1 | -1 |
|  | -1 | 11 | 1000 | -1 |
|  | -1 | 11 | 6722 | -1 |
|  | -1 | 12 | -1 | -1 |
|  | -1 | 12 | 1000 | -1 |
|  | -1 | 12 | 6722 | -1 |
|  | 60 | -1 | -1 | -1 |
|  | 60 | -1 | 1000 | -1 |
|  | 60 | -1 | 6722 | -1 |
|  | 60 | 0 | -1 | -1 |
|  | 60 | 0 | 1000 | 83.91 |
|  | 60 | 0 | 6722 | -1 |
|  | 60 | 2 | -1 | -1 |
|  | 60 | 2 | 1000 | 84 |
|  | 60 | 2 | 6722 | -1 |
|  | 60 | 3 | -1 | -1 |
|  | 60 | 3 | 1000 | 84 |
|  | 60 | 3 | 6722 | -1 |
|  | 60 | 6 | -1 | -1 |
|  | 60 | 6 | 1000 | 84 |
|  | 60 | 6 | 6722 | -1 |
|  | 60 | 11 | -1 | -1 |
|  | 60 | 11 | 1000 | 84 |
|  | 60 | 11 | 6722 | -1 |
|  | 60 | 12 | -1 | -1 |
|  | 60 | 12 | 1000 | -1 |
|  | 60 | 12 | 6722 | -1 |
|  | 120 | -1 | -1 | -1 |
|  | 120 | -1 | 1000 | -1 |
|  | 120 | -1 | 6722 | -1 |
|  | 120 | 0 | -1 | -1 |
|  | 120 | 0 | 1000 | 92.73 |
|  | 120 | 0 | 6722 | -1 |
|  | 120 | 2 | -1 | -1 |
|  | 120 | 2 | 1000 | 92.73 |
|  | 120 | 2 | 6722 | -1 |
|  | 120 | 3 | -1 | -1 |
|  | 120 | 3 | 1000 | 93 |
|  | 120 | 3 | 6722 | -1 |
|  | 120 | 6 | -1 | -1 |
|  | 120 | 6 | 1000 | 93 |
|  | 120 | 6 | 6722 | -1 |
|  | 120 | 11 | -1 | -1 |
|  | 120 | 11 | 1000 | 93 |
|  | 120 | 11 | 6722 | -1 |
|  | 120 | 12 | -1 | -1 |
|  | 120 | 12 | 1000 | -1 |
|  | 120 | 12 | 6722 | -1 |
|  | 180 | -1 | -1 | -1 |
|  | 180 | -1 | 1000 | -1 |
|  | 180 | -1 | 6722 | -1 |
|  | 180 | 0 | -1 | -1 |
|  | 180 | 0 | 1000 | 101.46 |
|  | 180 | 0 | 6722 | -1 |
|  | 180 | 2 | -1 | -1 |
|  | 180 | 2 | 1000 | 101.46 |
|  | 180 | 2 | 6722 | -1 |
|  | 180 | 3 | -1 | -1 |
|  | 180 | 3 | 1000 | 101.46 |
|  | 180 | 3 | 6722 | -1 |
|  | 180 | 6 | -1 | -1 |
|  | 180 | 6 | 1000 | 102 |
|  | 180 | 6 | 6722 | -1 |
|  | 180 | 11 | -1 | -1 |
|  | 180 | 11 | 1000 | 102 |
|  | 180 | 11 | 6722 | -1 |
|  | 180 | 12 | -1 | -1 |
|  | 180 | 12 | 1000 | -1 |
|  | 180 | 12 | 6722 | -1 |
|  | 300 | -1 | -1 | -1 |
|  | 300 | -1 | 1000 | -1 |
|  | 300 | -1 | 6722 | -1 |
|  | 300 | 0 | -1 | -1 |
|  | 300 | 0 | 1000 | 118.875 |
|  | 300 | 0 | 6722 | -1 |
|  | 300 | 2 | -1 | -1 |
|  | 300 | 2 | 1000 | 118.875 |
|  | 300 | 2 | 6722 | -1 |
|  | 300 | 3 | -1 | -1 |
|  | 300 | 3 | 1000 | 118.875 |
|  | 300 | 3 | 6722 | -1 |
|  | 300 | 6 | -1 | -1 |
|  | 300 | 6 | 1000 | 120 |
|  | 300 | 6 | 6722 | -1 |
|  | 300 | 11 | -1 | -1 |
|  | 300 | 11 | 1000 | 120 |
|  | 300 | 11 | 6722 | -1 |
|  | 300 | 12 | -1 | -1 |
|  | 300 | 12 | 1000 | -1 |
|  | 300 | 12 | 6722 | -1 |
|  | 350 | -1 | -1 | -1 |
|  | 350 | -1 | 1000 | -1 |
|  | 350 | -1 | 6722 | -1 |
|  | 350 | 0 | -1 | -1 |
|  | 350 | 0 | 1000 | 125.925 |
|  | 350 | 0 | 6722 | -1 |
|  | 350 | 2 | -1 | -1 |
|  | 350 | 2 | 1000 | 125.925 |
|  | 350 | 2 | 6722 | -1 |
|  | 350 | 3 | -1 | -1 |
|  | 350 | 3 | 1000 | 125.925 |
|  | 350 | 3 | 6722 | -1 |
|  | 350 | 6 | -1 | -1 |
|  | 350 | 6 | 1000 | 125.925 |
|  | 350 | 6 | 6722 | -1 |
|  | 350 | 11 | -1 | -1 |
|  | 350 | 11 | 1000 | 127.5 |
|  | 350 | 11 | 6722 | -1 |
|  | 350 | 12 | -1 | -1 |
|  | 350 | 12 | 1000 | -1 |
|  | 350 | 12 | 6722 | -1 |
|  | 44641 | -1 | -1 | -1 |
|  | 44641 | -1 | 1000 | -1 |
|  | 44641 | -1 | 6722 | -1 |
|  | 44641 | 0 | -1 | -1 |
|  | 44641 | 0 | 1000 | -1 |
|  | 44641 | 0 | 6722 | -1 |
|  | 44641 | 2 | -1 | -1 |
|  | 44641 | 2 | 1000 | -1 |
|  | 44641 | 2 | 6722 | -1 |
|  | 44641 | 3 | -1 | -1 |
|  | 44641 | 3 | 1000 | -1 |
|  | 44641 | 3 | 6722 | -1 |
|  | 44641 | 6 | -1 | -1 |
|  | 44641 | 6 | 1000 | -1 |
|  | 44641 | 6 | 6722 | -1 |
|  | 44641 | 11 | -1 | -1 |
|  | 44641 | 11 | 1000 | -1 |
|  | 44641 | 11 | 6722 | -1 |
|  | 44641 | 12 | -1 | -1 |
|  | 44641 | 12 | 1000 | -1 |
|  | 44641 | 12 | 6722 | -1 |

### 边界值法

通过对问题的分析，我们可以很快得到变量的边界值，分别为：

0<=T<=44640

0<=C<=11

0<=LastYearCost<=6721

我们设T，C，LastYearCost的标准值分别为20000，6，3000

则得到健壮边界测试用例为：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 本月通话时间 | 未按时缴费次数 | 跨年未缴费用 | 预计输出 |
|  | -1 | 6 | 3000 | -1 |
|  | 0 | 6 | 3000 | 175 |
|  | 10 | 6 | 3000 | 176.5 |
|  | 20000 | -1 | 3000 | -1 |
|  | 20000 | 0 | 3000 | 3085 |
|  | 20000 | 1 | 3000 | 3085 |
|  | 20000 | 6 | -1 | -1 |
|  | 20000 | 6 | 0 | 2935 |
|  | 20000 | 6 | 10 | 2935.5 |
|  | 20000 | 6 | 6000 | 3235 |
|  | 20000 | 6 | 6721 | 3271.05 |
|  | 20000 | 6 | 6722 | -1 |
|  | 20000 | 10 | 3000 | 3175 |
|  | 20000 | 11 | 3000 | 3175 |
|  | 20000 | 12 | 3000 | -1 |
|  | 40000 | 6 | 3000 | 5995 |
|  | 44640 | 6 | 3000 | 6670.12 |
|  | 44641 | 6 | 3000 | -1 |
|  | 20000 | 6 | 3000 | 3085 |

## 测试代码设计

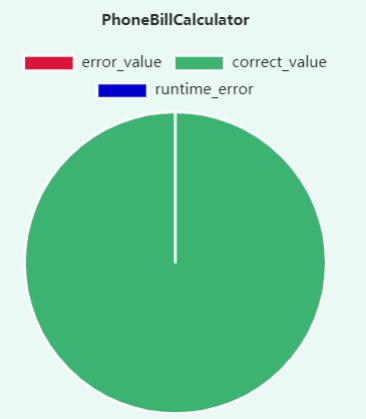
1. **public** **class** PhoneBillCalculator{
3. **private** **static** **double** getDiscount(**int** phone\_time,**int** uncost\_times){
4. **if**(phone\_time>0&&phone\_time<=60&&uncost\_times<=1){
5. **return** 0.01;
6. }
7. **if**(phone\_time>60&&phone\_time<=120&&uncost\_times<=2){
8. **return** 0.015;
9. }
10. **if**(phone\_time>120&&phone\_time<=180&&uncost\_times<=3){
11. **return** 0.02;
12. }
13. **if**(phone\_time>180&&phone\_time<=300&&uncost\_times<=3){
14. **return** 0.025;
15. }
16. **if**(phone\_time>300&&uncost\_times<=6){
17. **return** 0.03;
18. }
19. **return** 0.0;
20. }
21. **private** **static** **int** MaxPhoneTime=44640;
22. **private** **static** **double** MaxLastYearCost=6721.0;
23. **private** **static** **int** MaxUnCostTimes=11;
24. **public** **static** Double calPhoneBill(**int** phone\_time,**int** uncost\_times,**double** last\_year\_cost){
25. **if**(phone\_time<0||phone\_time>MaxPhoneTime||uncost\_times<0||uncost\_times>MaxUnCostTimes||last\_year\_cost<0||last\_year\_cost>MaxLastYearCost){
26. **return** -1.0;
27. }
28. **return** 25.0+phone\_time\*0.15\*(1-getDiscount(phone\_time, uncost\_times))+last\_year\_cost\*0.05;
29. }
30. }

## 测试用例执行情况

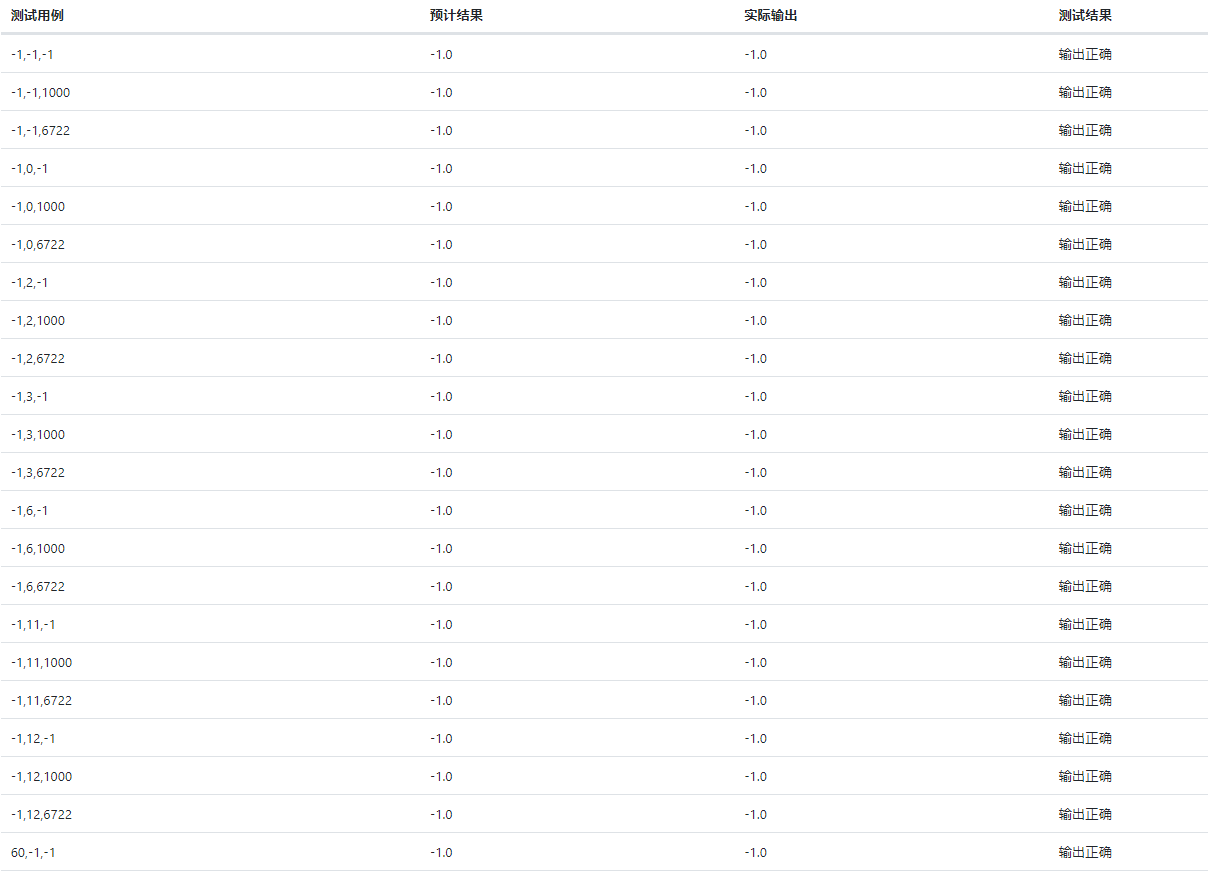
使用自制代码测试工具对编写代码加以测试，得到如下结果：

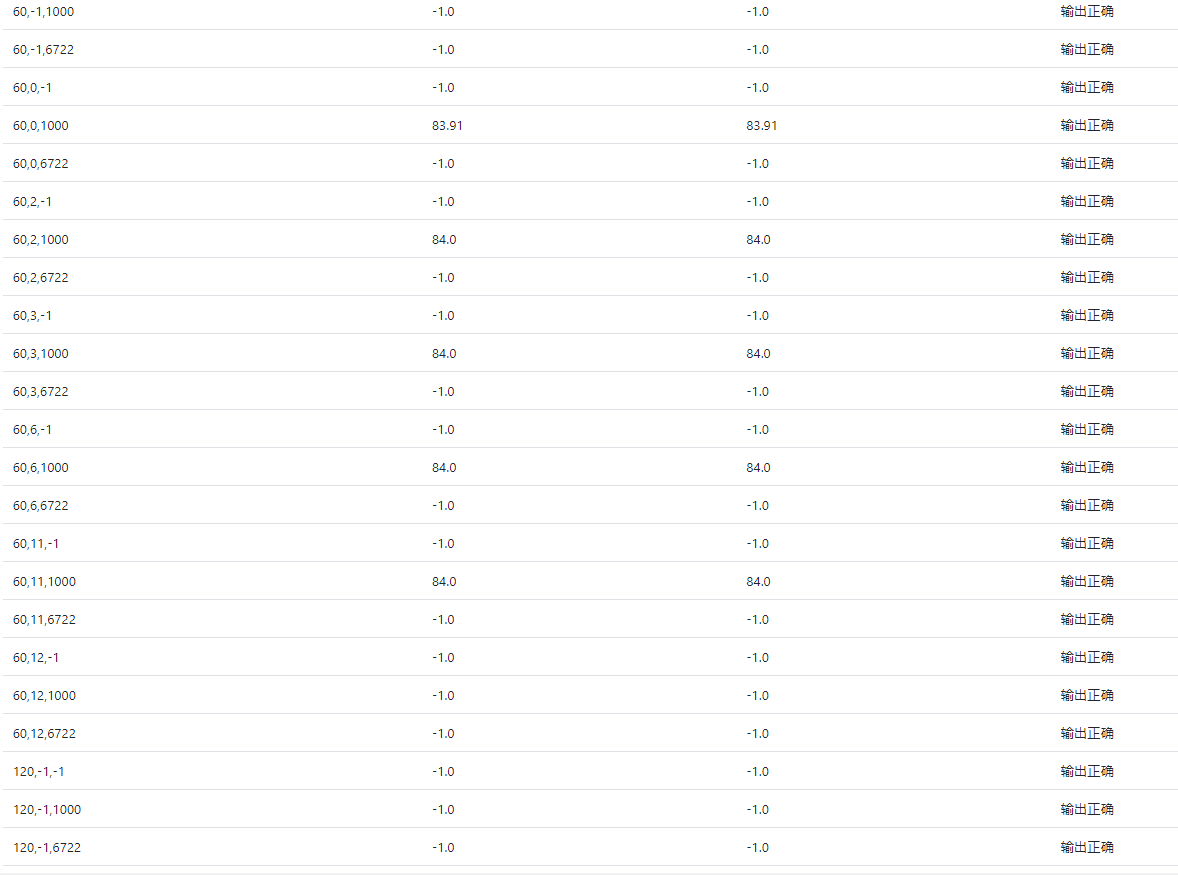
### 决策表法

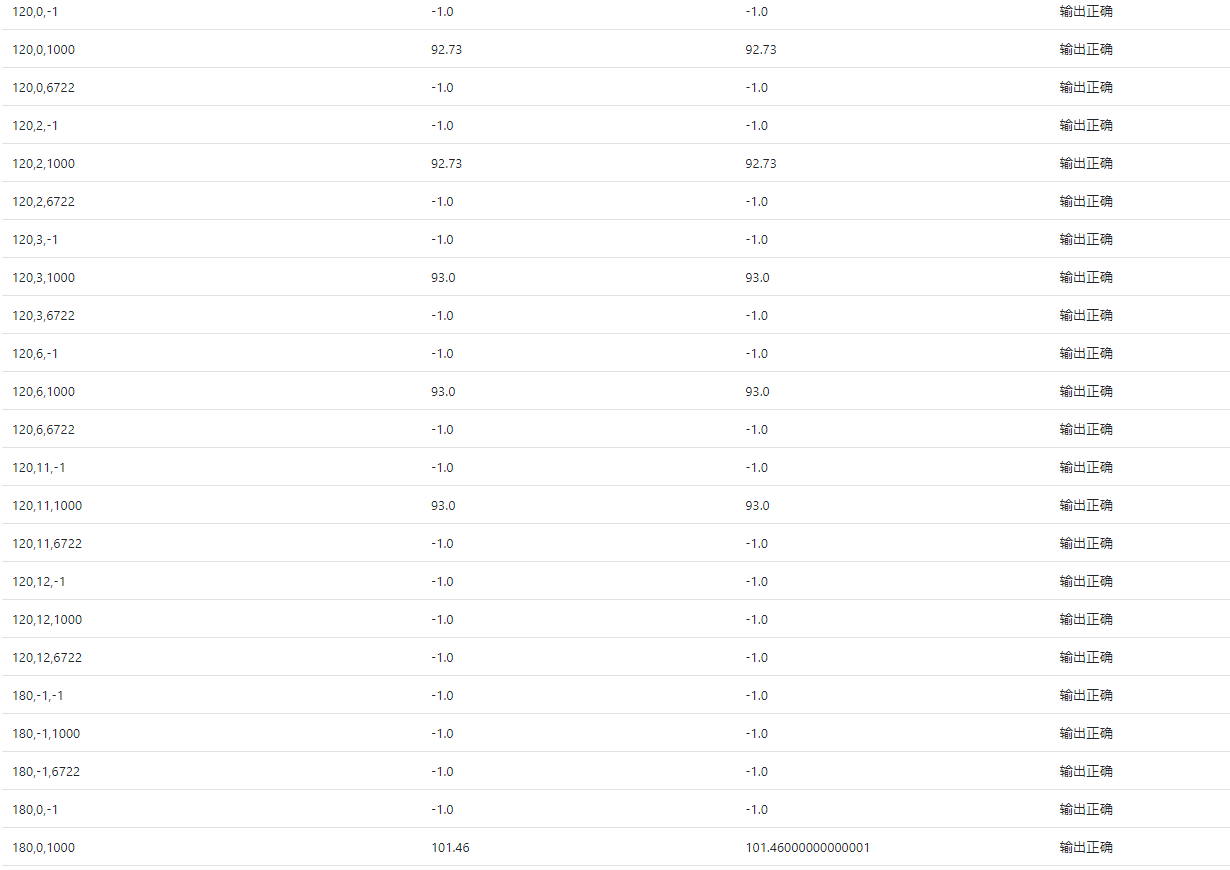
|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 预计结果 | 实际输出 | 测试结果 |
| 30,1,6721 | 365.505 | 365.505 | 输出正确 |
| 30,2,6721 | 365.55 | 365.55 | 输出正确 |
| 90,2,6721 | 374.3475 | 374.3475 | 输出正确 |
| 90,3,6721 | 374.55 | 374.55 | 输出正确 |
| 150,3,6721 | 383.1 | 383.1 | 输出正确 |
| 150,4,6721 | 383.55 | 383.55 | 输出正确 |
| 210,3,6721 | 391.7625 | 391.7625 | 输出正确 |
| 210,4,6721 | 392.55 | 392.55 | 输出正确 |
| 350,6,6721 | 411.975 | 411.975 | 输出正确 |
| 350,11,6721 | 413.55 | 413.55 | 输出正确 |
| 30,1,0 | 29.455 | 29.455 | 输出正确 |
| 30,2,0 | 29.5 | 29.5 | 输出正确 |
| 90,2,0 | 38.2975 | 38.2975 | 输出正确 |
| 90,3,0 | 38.5 | 38.5 | 输出正确 |
| 150,3,0 | 47.05 | 47.05 | 输出正确 |
| 150,4,0 | 47.5 | 47.5 | 输出正确 |
| 210,3,0 | 55.7125 | 55.7125 | 输出正确 |
| 210,4,0 | 56.5 | 56.5 | 输出正确 |
| 350,6,0 | 75.925 | 75.925 | 输出正确 |
| 350,11,0 | 77.5 | 77.5 | 输出正确 |

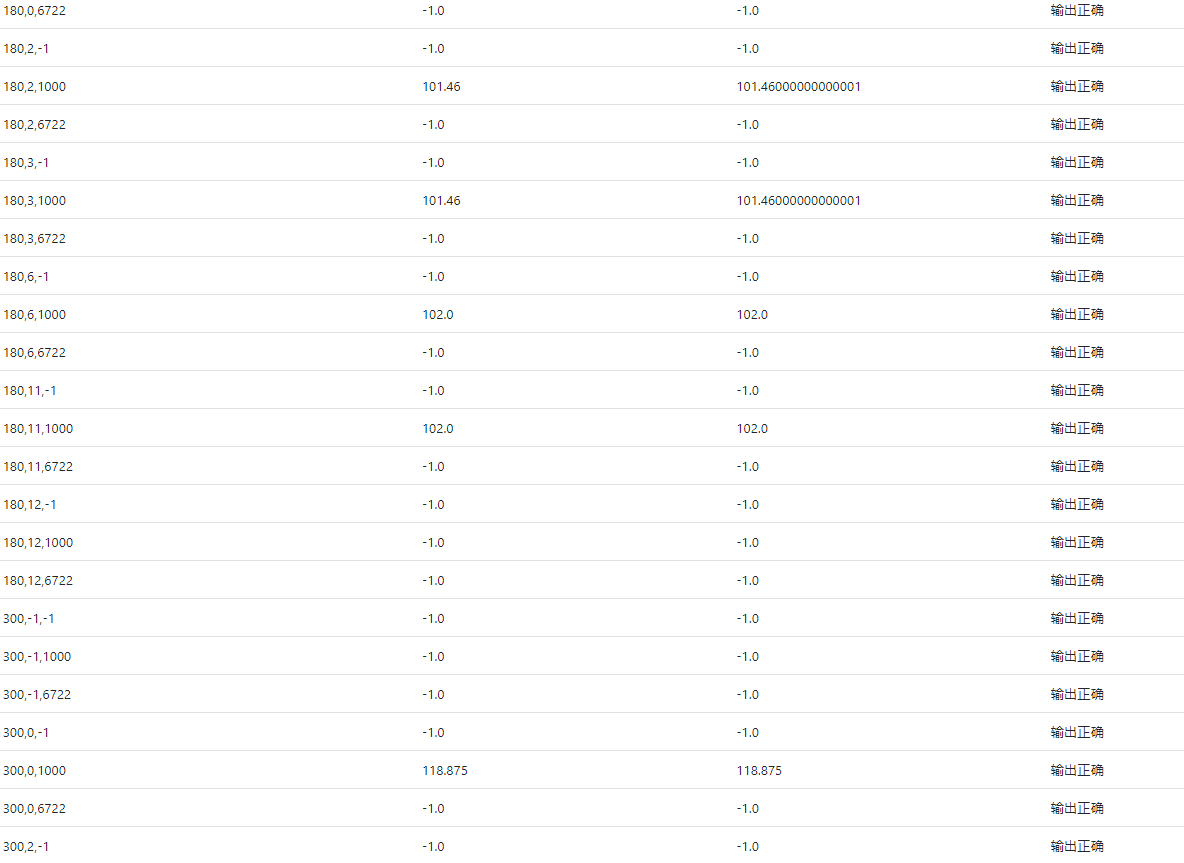


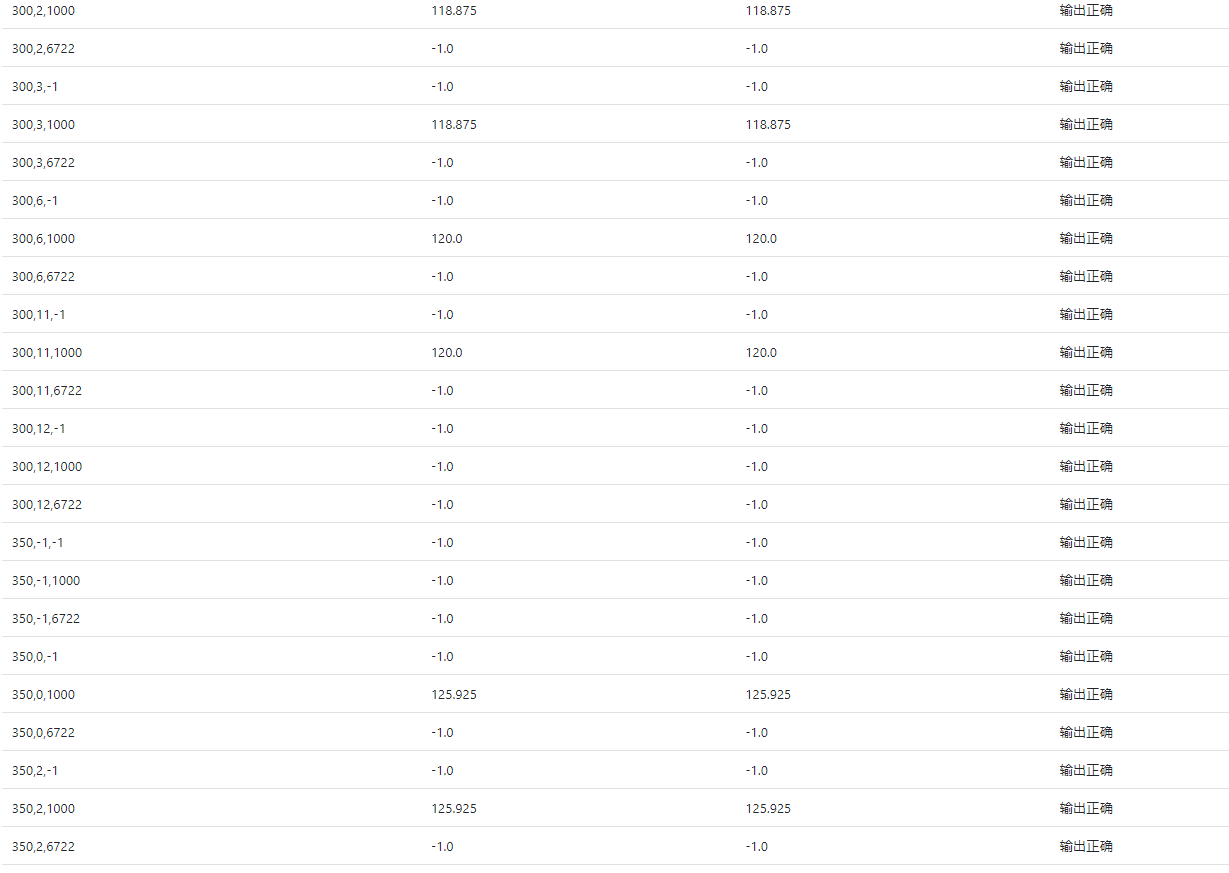
### 等价类法

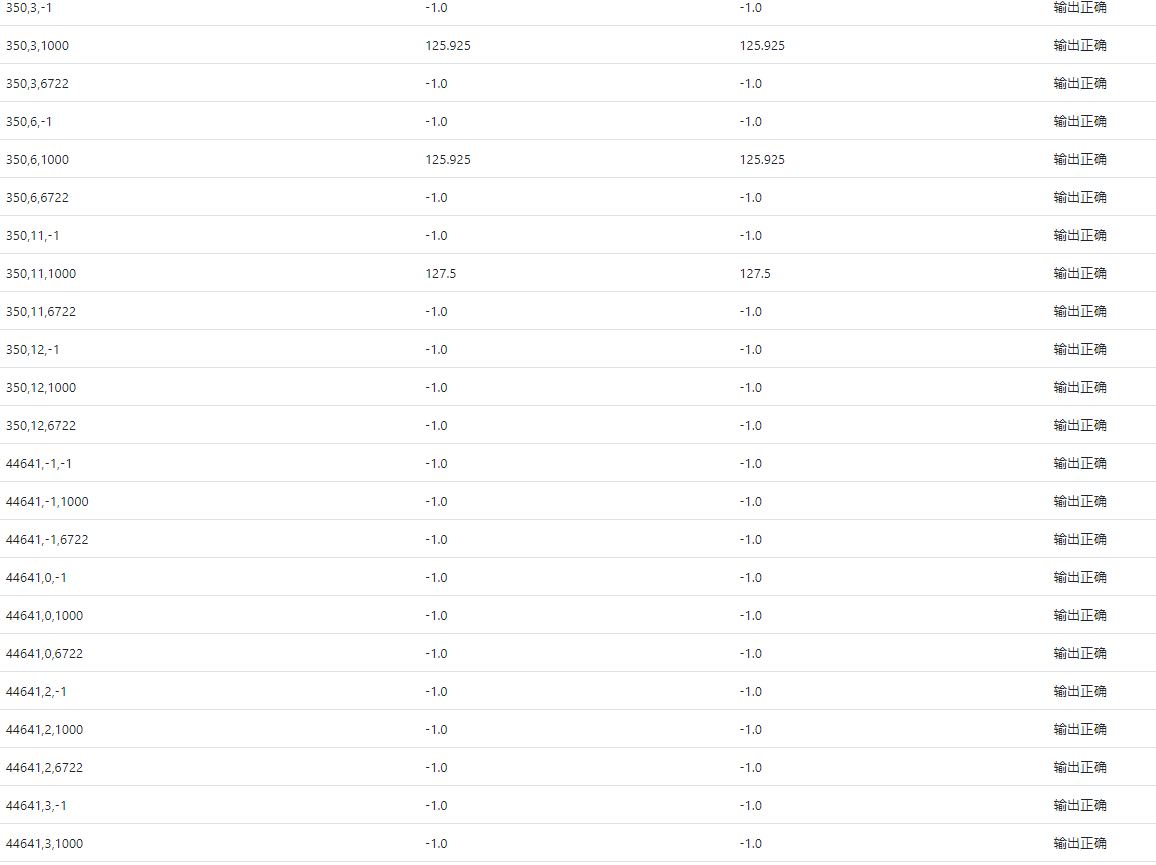


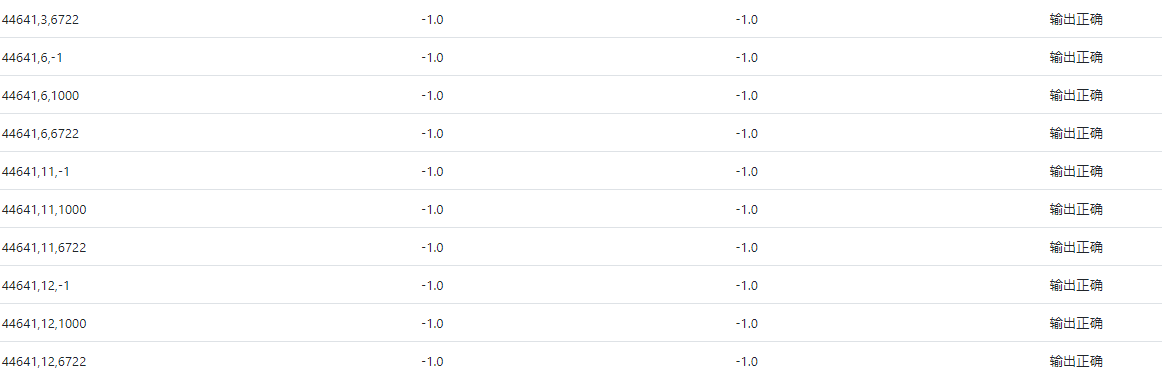


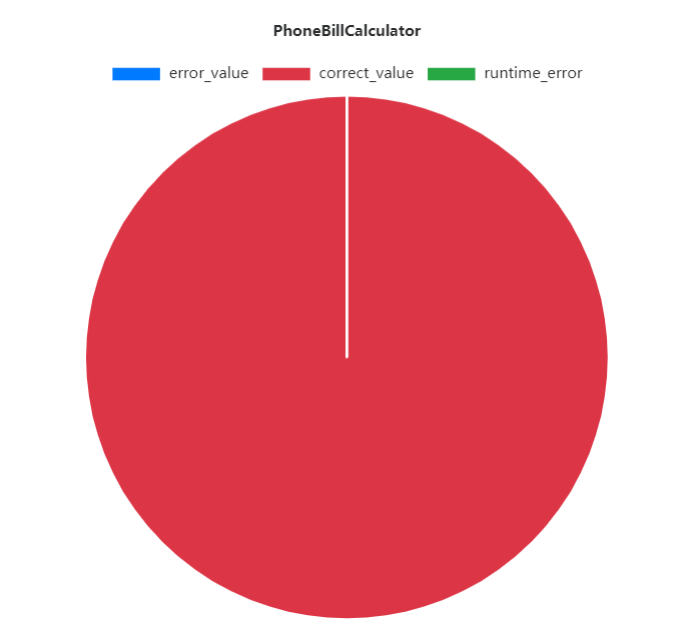








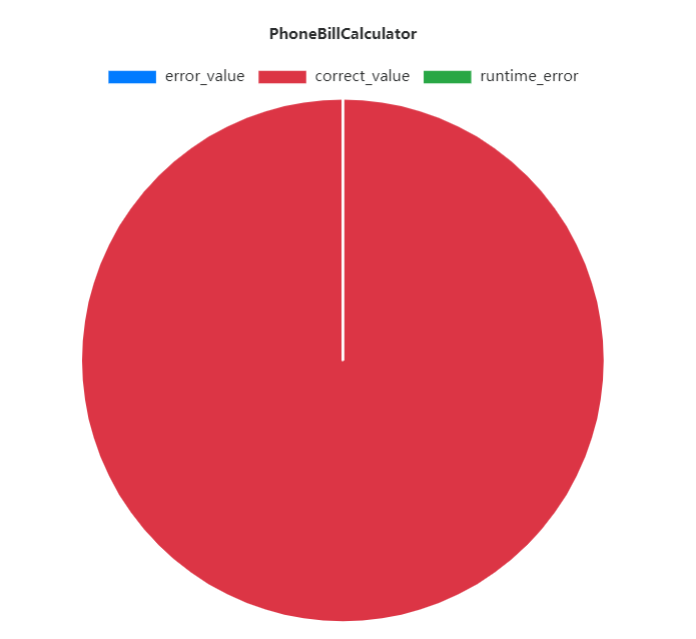




测试全部通过

### 边界值法





测试全部通过

## 对比分析

在本问题中，由于各变量相互间并不独立，同时在取值范围内根据不同取值程序执行不同动作，因此在本问题中采取边界值法测试程序不能很好覆盖所有情况，效果不好。

而使用等价类法进行分析，可以将各变量取值范围进行进一步细分，得到更精确的划分方式，同时由于变量间并不独立，因此应采取强健壮等价类的方法构建测试样例，但这种做法虽然可以覆盖到所有的情况，但仍存在很多冗余的测试样例。

而采用决策表的方式构建测试用例则是根据输入数据与程序执行的动作进行构建，既可以覆盖到所有情况，同时构建出的测试用例也十分简洁，没有冗余情况。在对本测试用例的设计中，采用的是扩展决策表，实际上在决策表的测试用例设计中已经考虑到了等价类的问题。

### 综合测试用例设计

但在本问题中，我们使用的决策表未能覆盖到无效输入的情况，因此要构建最优的测试用例，应当以本问题中构建的决策表为基础，在各变量的等价区间内根据**基本边界值**法补充设计测试用例。

如上所述，各变量的取值范围为：

0<=T<=44640

0<=C<=11

0<=LastYearCost<=6721

我们设T，C，LastYearCost的标准值分别为20000，6，3000。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 本月通话时间 | 未按时缴费次数 | 跨年未缴费用 | 预计输出 |
|  | 30 | 1 | 6721 | 365.505 |
|  | 0 | 1 | 6721 | 361.05 |
|  | 60 | 1 | 6721 | 369.96000000000004 |
|  | 61 | 1 | 6721 | 370.06275 |
|  | 1 | 1 | 6721 | 361.1985 |
|  | 30 | 0 | 6721 | 365.505 |
|  | 30 | 1 | 3000 | 179.45499999999998 |
|  | 30 | 1 | 6722 | -1.0 |
|  | 30 | 2 | 6721 | 365.55 |
|  | 90 | 2 | 6721 | 374.3475 |
|  | 60 | 2 | 6721 | 370.05 |
|  | 61 | 2 | 6721 | 370.06275 |
|  | 120 | 2 | 6721 | 378.78000000000003 |
|  | 121 | 2 | 6721 | 378.837 |
|  | 90 | 0 | 6721 | 374.3475 |
|  | 90 | 1 | 6721 | 374.3475 |
|  | 90 | 2 | 3000 | 188.2975 |
|  | 90 | 2 | 6722 | -1.0 |
|  | 90 | 3 | 6721 | 374.55 |
|  | 150 | 3 | 6721 | 383.1 |
|  | 120 | 3 | 6721 | 379.05 |
|  | 121 | 3 | 6721 | 378.837 |
|  | 180 | 3 | 6721 | 387.51 |
|  | 181 | 3 | 6721 | 387.52125 |
|  | 150 | 0 | 6721 | 383.1 |
|  | 150 | 1 | 6721 | 383.1 |
|  | 150 | 3 | 3000 | 197.05 |
|  | 150 | 3 | 6722 | -1.0 |
|  | 150 | 4 | 6721 | 383.55 |
|  | 210 | 3 | 6721 | 391.7625 |
|  | 300 | 3 | 6721 | 404.925 |
|  | 301 | 3 | 6721 | 404.8455 |
|  | 210 | 2 | 6721 | 391.7625 |
|  | 210 | 0 | 6721 | 391.7625 |
|  | 210 | 1 | 6721 | 391.7625 |
|  | 210 | 4 | 6721 | 392.55 |
|  | 210 | 3 | 3000 | 205.7125 |
|  | 210 | 3 | 6722 | -1.0 |
|  | 210 | 4 | 6721 | 392.55 |
|  | 350 | 6 | 6721 | 411.975 |
|  | 300 | 6 | 6721 | 406.05 |
|  | 301 | 6 | 6721 | 404.8455 |
|  | 44640 | 6 | 6721 | 6856.17 |
|  | 44641 | 6 | 6721 | -1.0 |
|  | 350 | 0 | 6721 | 411.975 |
|  | 350 | 1 | 6721 | 411.975 |
|  | 350 | 3 | 6721 | 411.975 |
|  | 350 | 7 | 6721 | 413.55 |
|  | 350 | 12 | 6721 | -1.0 |
|  | 350 | 6 | 3000 | 225.925 |
|  | 350 | 6 | 6722 | -1.0 |
|  | 350 | 11 | 6721 | 413.55 |
|  | 30 | 1 | 0 | 29.455 |
|  | 30 | 2 | 0 | 29.5 |
|  | 90 | 2 | 0 | 38.2975 |
|  | 90 | 3 | 0 | 38.5 |
|  | 150 | 3 | 0 | 47.05 |
|  | 150 | 4 | 0 | 47.5 |
|  | 210 | 3 | 0 | 55.7125 |
|  | 210 | 4 | 0 | 56.5 |
|  | 350 | 6 | 0 | 75.925 |
|  | 350 | 11 | 0 | 77.5 |
|  | 30 | 1 | 6722 | -1.0 |

### 测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 预计结果 | 实际输出 | 测试结果 |
| 30,1,6721 | 365.505 | 365.505 | 输出正确 |
| 0,1,6721 | 361.05 | 361.05 | 输出正确 |
| 60,1,6721 | 369.96000000000004 | 369.96000000000004 | 输出正确 |
| 61,1,6721 | 370.06275 | 370.06275 | 输出正确 |
| 1,1,6721 | 361.1985 | 361.1985 | 输出正确 |
| 30,0,6721 | 365.505 | 365.505 | 输出正确 |
| 30,1,3000 | 179.45499999999998 | 179.45499999999998 | 输出正确 |
| 30,1,6722 | -1.0 | -1.0 | 输出正确 |
| 30,2,6721 | 365.55 | 365.55 | 输出正确 |
| 90,2,6721 | 374.3475 | 374.3475 | 输出正确 |
| 60,2,6721 | 370.05 | 370.05 | 输出正确 |
| 61,2,6721 | 370.06275 | 370.06275 | 输出正确 |
| 120,2,6721 | 378.78000000000003 | 378.78000000000003 | 输出正确 |
| 121,2,6721 | 378.837 | 378.837 | 输出正确 |
| 90,0,6721 | 374.3475 | 374.3475 | 输出正确 |
| 90,1,6721 | 374.3475 | 374.3475 | 输出正确 |
| 90,2,3000 | 188.2975 | 188.2975 | 输出正确 |
| 90,2,6722 | -1.0 | -1.0 | 输出正确 |
| 90,3,6721 | 374.55 | 374.55 | 输出正确 |
| 150,3,6721 | 383.1 | 383.1 | 输出正确 |
| 120,3,6721 | 379.05 | 379.05 | 输出正确 |
| 121,3,6721 | 378.837 | 378.837 | 输出正确 |
| 180,3,6721 | 387.51 | 387.51 | 输出正确 |
| 181,3,6721 | 387.52125 | 387.52125 | 输出正确 |
| 150,0,6721 | 383.1 | 383.1 | 输出正确 |
| 150,1,6721 | 383.1 | 383.1 | 输出正确 |
| 150,3,3000 | 197.05 | 197.05 | 输出正确 |
| 150,3,6722 | -1.0 | -1.0 | 输出正确 |
| 150,4,6721 | 383.55 | 383.55 | 输出正确 |
| 210,3,6721 | 391.7625 | 391.7625 | 输出正确 |
| 300,3,6721 | 404.925 | 404.925 | 输出正确 |
| 301,3,6721 | 404.8455 | 404.8455 | 输出正确 |
| 210,2,6721 | 391.7625 | 391.7625 | 输出正确 |
| 210,0,6721 | 391.7625 | 391.7625 | 输出正确 |
| 210,1,6721 | 391.7625 | 391.7625 | 输出正确 |
| 210,4,6721 | 392.55 | 392.55 | 输出正确 |
| 210,3,3000 | 205.7125 | 205.7125 | 输出正确 |
| 210,3,6722 | -1.0 | -1.0 | 输出正确 |
| 210,4,6721 | 392.55 | 392.55 | 输出正确 |
| 350,6,6721 | 411.975 | 411.975 | 输出正确 |
| 300,6,6721 | 406.05 | 406.05 | 输出正确 |
| 301,6,6721 | 404.8455 | 404.8455 | 输出正确 |
| 44640,6,6721 | 6856.17 | 6856.17 | 输出正确 |
| 44641,6,6721 | -1.0 | -1.0 | 输出正确 |
| 350,0,6721 | 411.975 | 411.975 | 输出正确 |
| 350,1,6721 | 411.975 | 411.975 | 输出正确 |
| 350,3,6721 | 411.975 | 411.975 | 输出正确 |
| 350,7,6721 | 413.55 | 413.55 | 输出正确 |
| 350,12,6721 | -1.0 | -1.0 | 输出正确 |
| 350,6,3000 | 225.925 | 225.925 | 输出正确 |
| 350,6,6722 | -1.0 | -1.0 | 输出正确 |
| 350,11,6721 | 413.55 | 413.55 | 输出正确 |
| 30,1,0 | 29.455 | 29.455 | 输出正确 |
| 30,2,0 | 29.5 | 29.5 | 输出正确 |
| 90,2,0 | 38.2975 | 38.2975 | 输出正确 |
| 90,3,0 | 38.5 | 38.5 | 输出正确 |
| 150,3,0 | 47.05 | 47.05 | 输出正确 |
| 150,4,0 | 47.5 | 47.5 | 输出正确 |
| 210,3,0 | 55.7125 | 55.7125 | 输出正确 |
| 210,4,0 | 56.5 | 56.5 | 输出正确 |
| 350,6,0 | 75.925 | 75.925 | 输出正确 |
| 350,11,0 | 77.5 | 77.5 | 输出正确 |
| 30,1,6722 | -1.0 | -1.0 | 输出正确 |

# 练习8—销售系统问题问题

一销售系统，如果销售员的**年销售额**大于**200万**RMB**且请假天数不超过10天的情况下**，现金到帐大于等于60%，则佣金（提成）系数为7，即佣金值为销售额除以佣金系数；现金到帐小于60%，佣金不予计算。所有其他情况且现金到帐小于等于85%，则按佣金系数均为6计算佣金，现金到账大于85%，佣金系数按5处理。

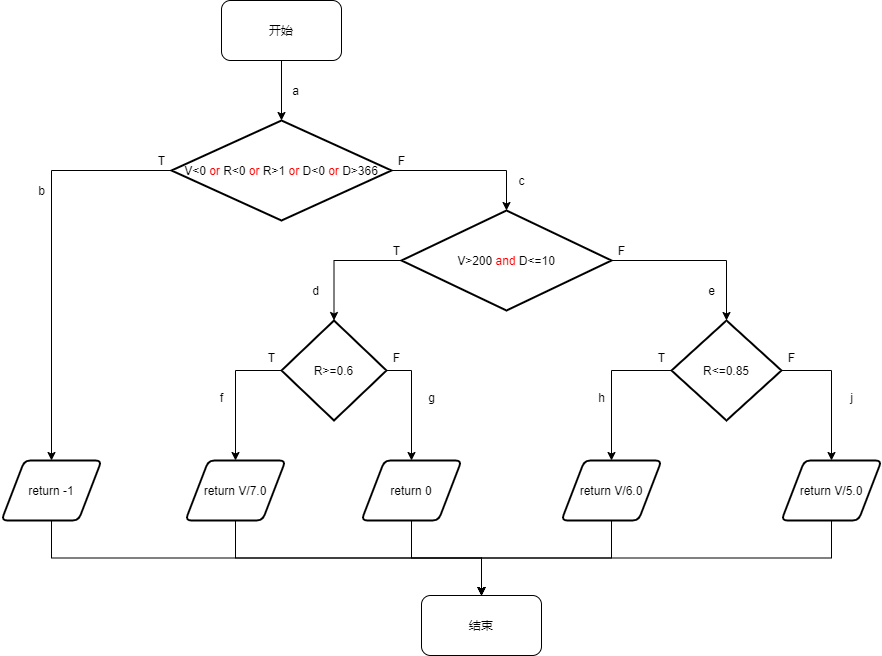
根据题意设计流程图并设计测试用例实现白盒测试（White Box Test）的1）语句覆盖，2）判断覆盖，3）条件覆盖，4）判断—条件覆盖，5）条件组合覆盖（测试用例及覆盖表示要清晰）

## 问题分析

由题意可知，销售额V大于等于0（同时规定V为以万为单位的整数），现金到账率R的范围为[0,1]，由于一年最多366天，因此请假天数D的范围为[0,366]。

对于非法输入，在这里我们返回-1

并且，根据题意，我们可以构造出以下程序流程图和控制流图。



## 用例设计

### 语句覆盖

由上述流程图和控制流图可得知，若要覆盖全部语句，可选择的程序执行路径集合为{ab, acdf, acdg, aceh, acei}，由此可构建出如下测试用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | V | R | D | 预计输出 |
|  | -1 | 0 | 0 | -1 |
|  | 700 | 0.8 | 5 | 100 |
|  | 700 | 0.1 | 5 | 0 |
|  | 600 | 0.7 | 15 | 100 |
|  | 500 | 0.9 | 15 | 100 |

### 判断覆盖

由上述流程图和控制流图可得知，若要完成判断覆盖，可选择的程序执行路径集合为**{ab, acdf, acdg, aceh, acei}**，由此可构建出如下测试用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | V | R | D | 预计输出 |
| T1 | -1 | 0 | 0 | -1 |
| T2 | 700 | 0.8 | 5 | 100 |
| T3 | 700 | 0.1 | 5 | 0 |
| T4 | 600 | 0.7 | 15 | 100 |
| T5 | 500 | 0.9 | 15 | 100 |

### 条件覆盖

由上述流程图和控制流图可得知，本程序共有4个判断语句。

对于第一个判断语句，有以下取值：

V<0为真，记为T1

V<0为假，记为F1

R<0为真，记为T2

R<0为假，记为F2

R>1为真，记为T3

R>1为假，记为F3

D<0为真，记为T4

D<0为假，记为F4

D>366为真，记为T5

D>366为假，记为F5

对于第二个判断语句，有以下取值：

V>200为真，记为T6

V>200为假，记为F6

D<=10为真，记为T7

D<=10为假，记为F7

对于第三个判断语句，有以下取值：

R>0.6为真，记为T8

R>0.6为假，记为F8

对于第四个判断语句，有以下取值：

R<=0.85为真，记为T9

R<=0.85为假，记为F9

通过上述条件，我们得到以下用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | V R D | 所走路径 | 覆盖条件 | 预计输出 |
| T1 | -1 -1 -1 | ab | T1 T2 F3 T4 F5 F6 T7 F8 T9 | -1 |
| T2 | 300 2 367 | ab | F1 F2 T3 F4 T5 T6 F7 T8 F9 | -1 |

该用例覆盖了所有条件，但仅覆盖了一个分支，故效果不好。

### 判断—条件覆盖

由上述流程图和控制流图可得知，本程序共有4个判断语句。

对于第一个判断语句，有以下取值：

V<0为真，记为T1

V<0为假，记为F1

R<0为真，记为T2

R<0为假，记为F2

R>1为真，记为T3

R>1为假，记为F3

D<0为真，记为T4

D<0为假，记为F4

D>366为真，记为T5

D>366为假，记为F5

对于第二个判断语句，有以下取值：

V>200为真，记为T6

V>200为假，记为F6

D<=10为真，记为T7

D<=10为假，记为F7

对于第三个判断语句，有以下取值：

R>0.6为真，记为T8

R>0.6为假，记为F8

对于第四个判断语句，有以下取值：

R<=0.85为真，记为T9

R<=0.85为假，记为F9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | V R D | 所走路径 | 覆盖条件 | 预计输出 |
|  | -1 -1 -1 | ab | T1 T2 F3 T4 F5 F6 T7 F8 T9 | -1 |
|  | 300 2 367 | ab | F1 F2 T3 F4 T5 T6 F7 T8 F9 | -1 |
|  | 700 0.8 5 | acdf | F1 F2 F3 F4 F5 T6 T7 T8 T9 | 100 |
|  | 700 0.1 5 | acdg | F1 F2 F3 F4 F5 T6 T7 F8 T9 | 0 |
|  | 600 0.7 15 | aceh | F1 F2 F3 F4 F5 T6 F7 T8 T9 | 100 |
|  | 500 0.9 15 | acei | F1 F2 F3 F4 F5 T6 F7 T8 F9 | 100 |

### 条件组合覆盖

对于第一个判断语句，有以下取值：

V<0为真，记为T1

V<0为假，记为F1

R<0为真，记为T2

R<0为假，记为F2

R>1为真，记为T3

R>1为假，记为F3

D<0为真，记为T4

D<0为假，记为F4

D>366为真，记为T5

D>366为假，记为F5

对于第二个判断语句，有以下取值：

V>200为真，记为T6

V>200为假，记为F6

D<=10为真，记为T7

D<=10为假，记为F7

对于第三个判断语句，有以下取值：

R>0.6为真，记为T8

R>0.6为假，记为F8

对于第四个判断语句，有以下取值：

R<=0.85为真，记为T9

R<=0.85为假，记为F9

根据上面得到的条件，我们针对每个判断语句对条件进行分组:

对于第一个判断语句，要使条件覆盖，V的取值分别为V<0和0<=V，R的取值分别为R<0，0<=R<=1，1<R，D的取值分别为D<0，0<=D<=366，366<D，除此之外，还需要排除一些互斥条件，例如在第一个判断语句中，T2与T3不能共同时组合，T4与T5不能够同时组合等。

因此条件组合有：

1. V<0 R<0 D<0 记为T1 T2 F3 T4 F5
2. V<0 R<0 0<=D<=366 记为T1 T2 F3 F4 F5
3. V<0 R<0 366<D 记为T1 T2 F3 F4 T5
4. V<0 0<=R<=1 D<0 记为T1 F2 F3 T4 F5
5. V<0 0<=R<=1 0<=D<=366 记为T1 F2 F3 F4 F5
6. V<0 0<=R<=1 366<D 记为T1 F2 F3 F4 T5
7. V<0 1<R D<0 记为T1 F2 T3 T4 F5
8. V<0 1<R 0<=D<=366 记为T1 F2 T3 F4 F5
9. V<0 1<R 366<D 记为T1 F2 T3 F4 T5
10. 0<=V R<0 D<0 记为F1 T2 F3 T4 F5
11. 0<=V R<0 0<=D<=366 记为F1 T2 F3 F4 F5
12. 0<=V R<0 366<D 记为F1 T2 F3 F4 T5
13. 0<=V 0<=R<=1 D<0 记为F1 F2 F3 T4 F5
14. 0<=V 0<=R<=1 0<=D<=366 记为F1 F2 F3 F4 F5
15. 0<=V 0<=R<=1 366<D 记为F1 F2 F3 F4 T5
16. 0<=V 1<R D<0 记为F1 F2 T3 T4 F5
17. 0<=V 1<R 0<=D<=366 记为F1 F2 T3 F4 F5
18. 0<=V 1<R 366<D 记为F1 F2 T3 F4 T5

对于第二个判断语句，有以下分组取值

1. V>200 D<=10 记为T6 T7
2. V>200 D>10 记为T6 F7
3. V<=200 D<=10 记为F6 T7
4. V<=200 D>10 记为F6 F7

对于第三个判断语句，有以下分组取值

1. R>0.6 记为T8
2. R<=0.6 记为F8

对于第四个判断语句，有以下分组取值

1. R<=0.85 记为T9
2. R>0.85 记为F9

由此，可得到如下测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例编号 | V R D | 覆盖组合号 | 所走路径 | 覆盖条件 | 预计输出 |
|  | -1 -1 -1 | 1+21+24+25 | ab | T1 T2 F3 T4 F5 F6 T7 F8 T9 | -1 |
|  | -1 -1 15 | 2+22+24+25 | ab | T1 T2 F3 F4 F5 F6 F7 F8 T9 | -1 |
|  | -1 -1 367 | 3+22+24+25 | ab | T1 T2 F3 F4 T5 F6 F7 F8 T9 | -1 |
|  | -1 0.2 -1 | 4+21+24+25 | ab | T1 F2 F3 T4 F5 F6 T7 F8 T9 | -1 |
|  | -1 0.2 15 | 5+22+24+25 | ab | T1 F2 F3 F4 F5 F6 F7 F8 T9 | -1 |
|  | -1 0.2 367 | 6+22+24+25 | ab | T1 F2 F3 F4 T5 F6 F7 F8 T9 | -1 |
|  | -1 2 -1 | 7+21+23+26 | ab | T1 F2 T3 T4 F5 F6 T7 T8 F9 | -1 |
|  | -1 2 15 | 8+22+23+26 | ab | T1 F2 T3 F4 F5 F6 F7 T8 F9 | -1 |
|  | -1 2 367 | 9+22+23+26 | ab | T1 F2 T3 F4 T5 F6 F7 T8 F9 | -1 |
|  | 200 -1 -1 | 10+21+24+25 | ab | F1 T2 F3 T4 F5 F6 T7 F8 T9 | -1 |
|  | 200 -1 15 | 11+22+24+25 | ab | F1 T2 F3 F4 F5 F6 F7 F8 T9 | -1 |
|  | 200 -1 367 | 12+22+24+25 | ab | F1 T2 F3 F4 T5 F6 F7 F8 T9 | -1 |
|  | 200 0.2 -1 | 13+21+24+25 | ab | F1 F2 F3 T4 F5 F6 T7 F8 T9 | -1 |
|  | 700 0.8 5 | 14+19+23+25 | acdf | F1 F2 F3 F4 F5 T6 T7 T8 T9 | 100 |
|  | 200 0.2 367 | 15+22+24+25 | ab | F1 F2 F3 F4 T5 F6 F7 F8 T9 | -1 |
|  | 200 2 -1 | 16+21+23+26 | ab | F1 F2 T3 T4 F5 F6 T7 T8 F9 | -1 |
|  | 200 2 15 | 17+22+23+26 | ab | F1 F2 T3 F4 F5 F6 F7 T8 F9 | -1 |
|  | 200 2 367 | 18+22+23+26 | ab | F1 F2 T3 F4 T5 F6 F7 T8 F9 | -1 |
|  | 700 0.1 5 | 14+19+24+25 | acdg | F1 F2 F3 F4 F5 T6 T7 F8 T9 | 0 |
|  | 60 0.7 5 | 14+21+23+25 | acdh | F1 F2 F3 F4 F5 F6 T7 T8 T9 | 10 |
|  | 50 0.9 15 | 14+22+23+26 | acdi | F1 F2 F3 F4 F5 F6 F7 T8 F9 | 10 |

如上表所示，构成的用例覆盖每个判断的所有条件组合。

## 测试代码设计

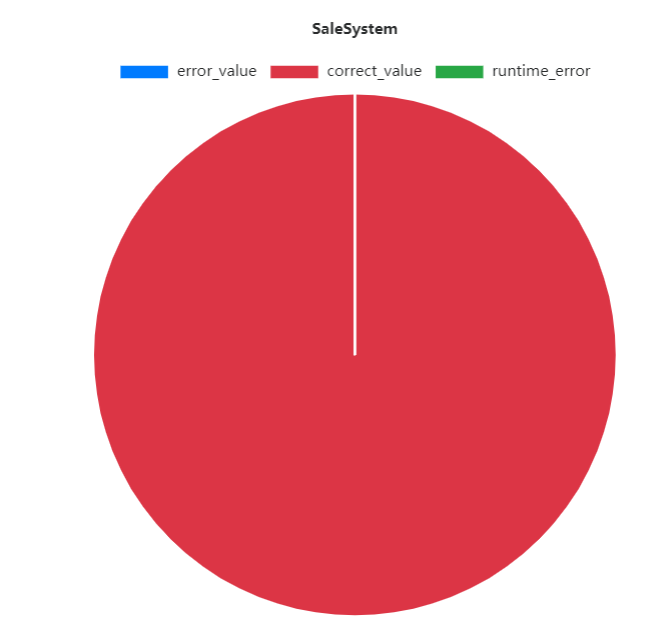
1. **public** **class** SaleSystem{
2. **static** **public** Double getBrokerage(Integer sales\_value, Double arrival\_rate, Integer leave\_days){
3. **if**(sales\_value<0||arrival\_rate<0||arrival\_rate>1||leave\_days<0||leave\_days>365){
4. **return** -1;
5. }
6. **if**(sales\_value>200&&leave\_days<=10){
7. **if**(arrival\_rate>0.6){
8. **return** sales\_value/7.0;
9. }**else**{
10. **return** 0;
11. }
12. }**else**{
13. **if**(arrival\_rate<0.85){
14. **return** sales\_value/6.0;
15. }**else**{
16. **return** sales\_value/5.0;
17. }
18. }
19. }
20. }

## 测试用例执行情况

使用自制代码测试工具对编写代码加以测试，得到如下结果：

### 语句覆盖

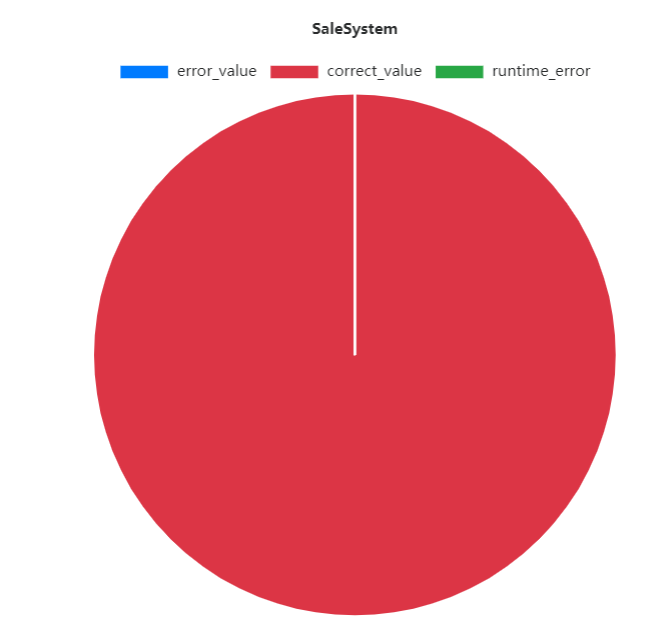




测试全部通过

### 判断覆盖

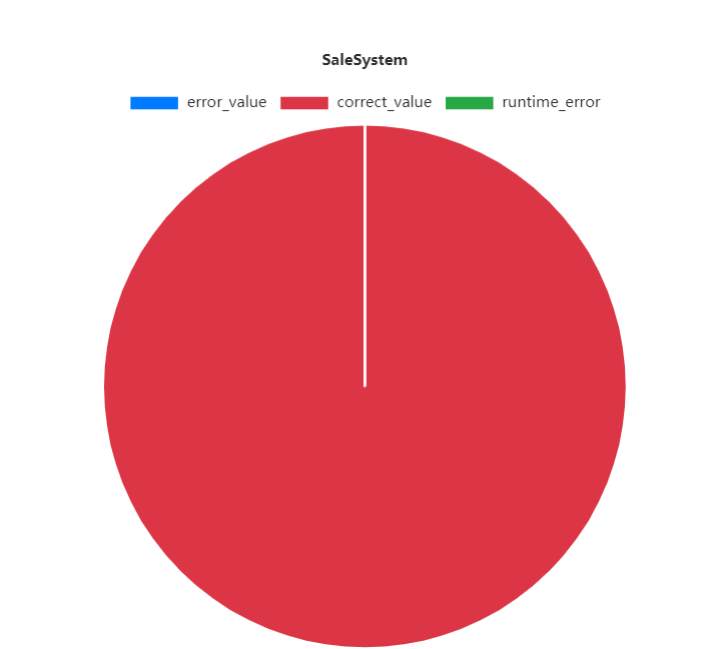




测试全部通过

### 条件覆盖





测试全部通过

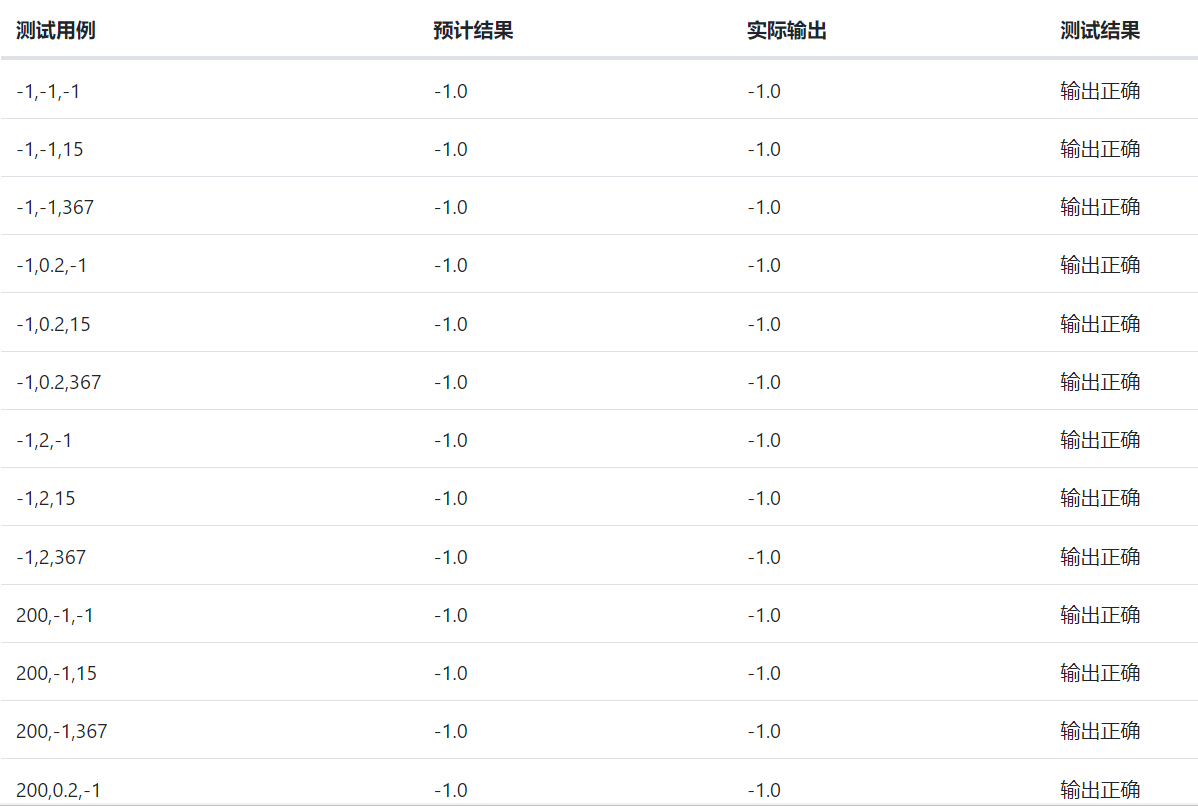
### 判断—条件覆盖



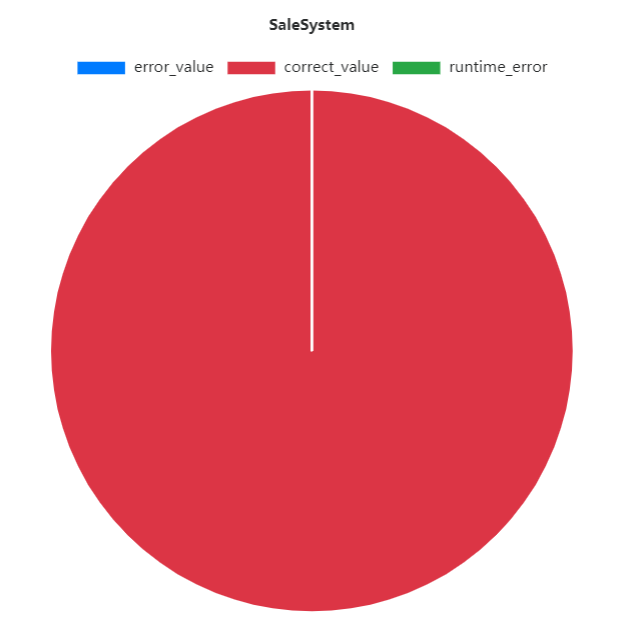


测试全部通过

### 条件组合覆盖







测试全部通过

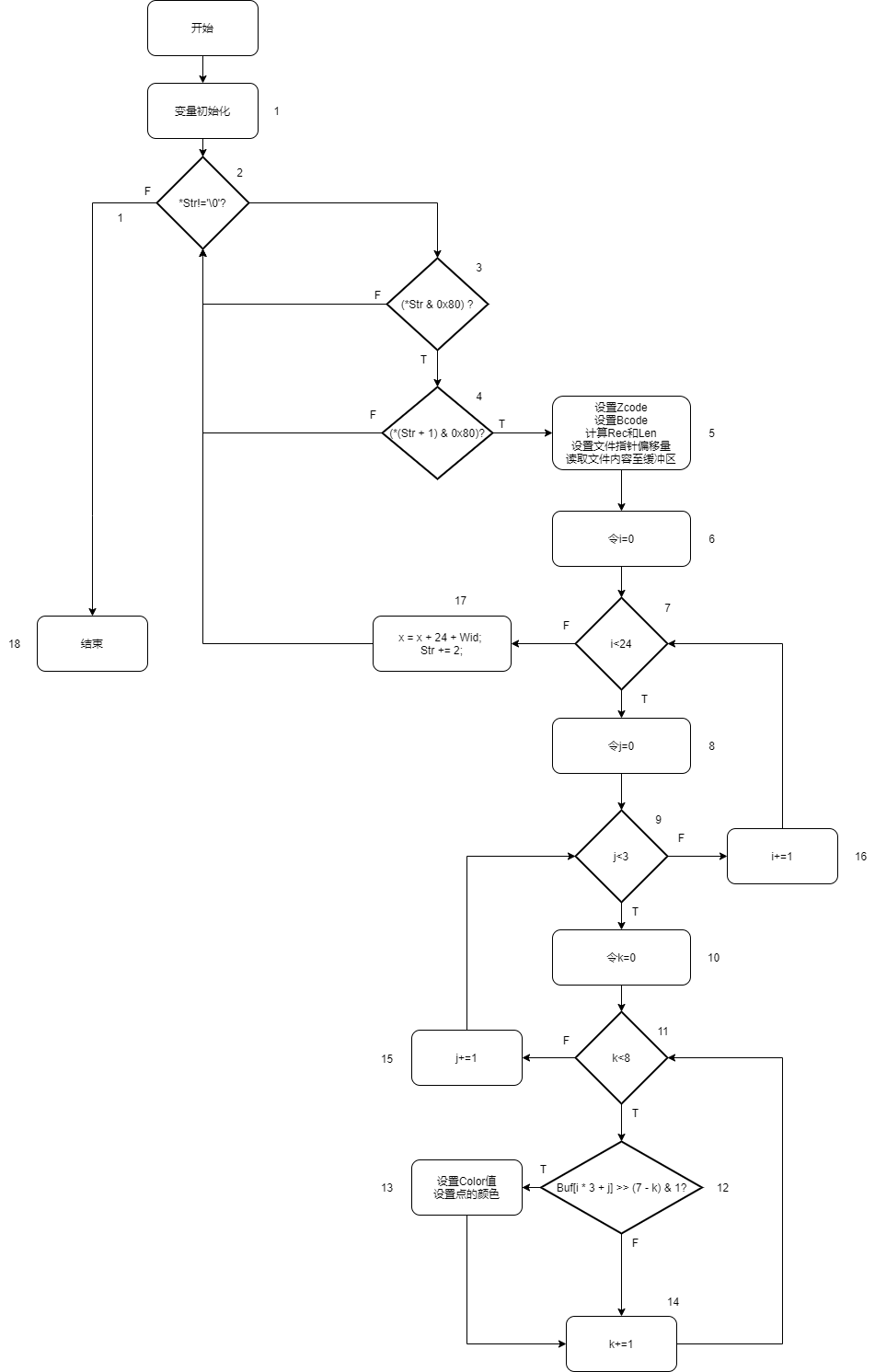
# 练习9—根据代码构建CFG图

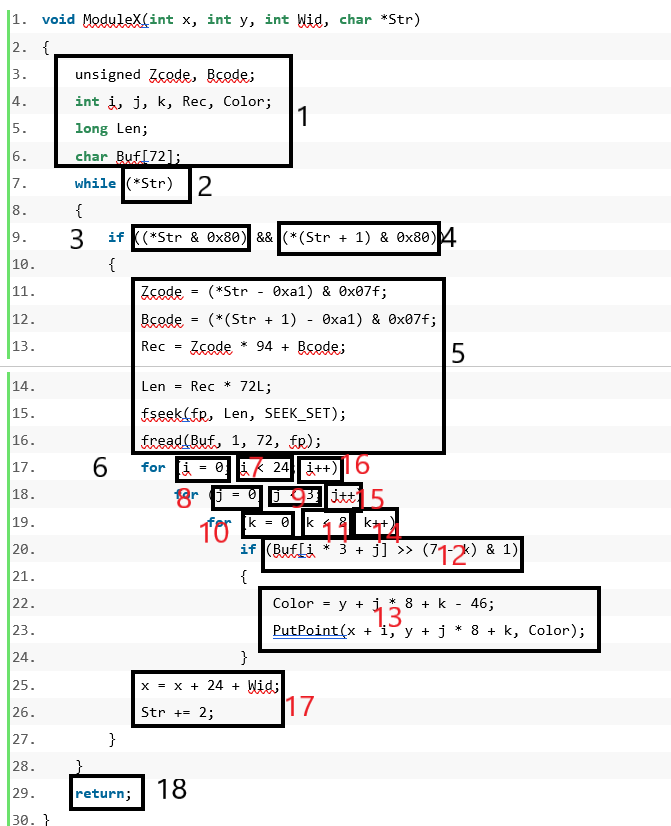
根据下列c语言代码构建控制流图

1. **void** ModuleX(**int** x, **int** y, **int** Wid, **char** \*Str)
2. {
3. unsigned Zcode, Bcode;
4. **int** i, j, k, Rec, Color;
5. **long** Len;
6. **char** Buf[72];
7. **while** (\*Str)
8. {
9. **if** ((\*Str & 0x80) && (\*(Str + 1) & 0x80))
10. {
11. Zcode = (\*Str - 0xa1) & 0x07f;
12. Bcode = (\*(Str + 1) - 0xa1) & 0x07f;
13. Rec = Zcode \* 94 + Bcode;
14. Len = Rec \* 72L;
15. fseek(fp, Len, SEEK\_SET);
16. fread(Buf, 1, 72, fp);
17. **for** (i = 0; i < 24; i++)
18. **for** (j = 0; j < 3; j++)
19. **for** (k = 0; k < 8; k++)
20. **if** (Buf[i \* 3 + j] >> (7 - k) & 1)
21. {
22. Color = y + j \* 8 + k - 46;
23. PutPoint(x + i, y + j \* 8 + k, Color);
24. }
25. x = x + 24 + Wid;
26. Str += 2;
27. }
28. }
29. **return**;
30. }

## 流程图

根据代码，我们画出如下流程图，并按照程序图分析划分各个节点并标号





## 控制流图（CFG）

根据上述流程图及各节点编号，我们可以画出如下CFG图

