****

**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 |  |
| 王笑天 |  |
| 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

# 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，可以构造出如下10种测试用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

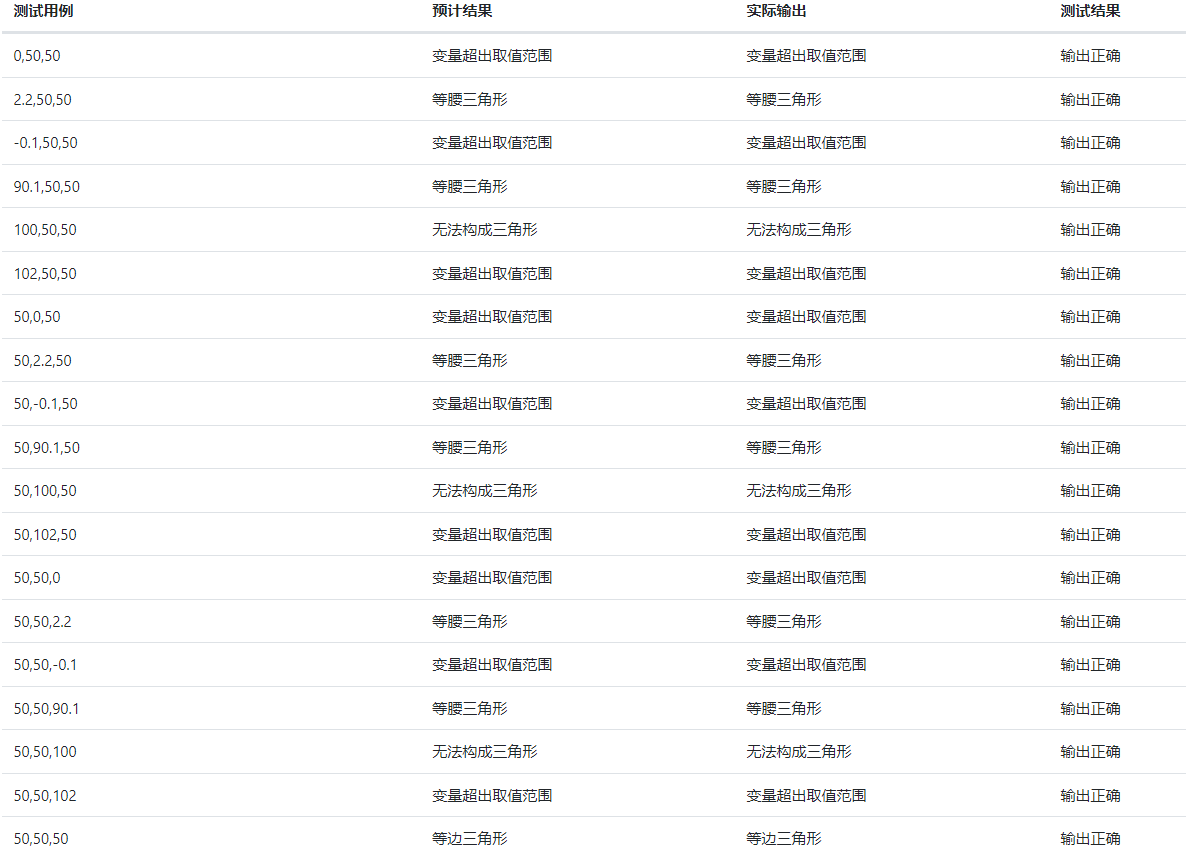
## 三角形问题测试代码设计

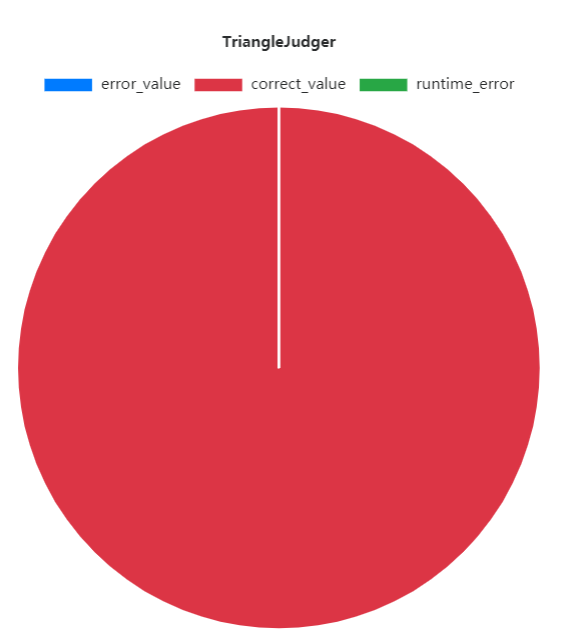
1. **public** **class** TriangleJudger{
2. **private** **static** **final** **double** MinV=0;
3. **private** **static** **final** **double** MaxV=100;
4. **private** **static** **boolean** comp(**double** a,**double** b){
5. **return** Math.abs(a-b)<1e-6;
6. }
7. **public** **static** String getTriangleType(**double** 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

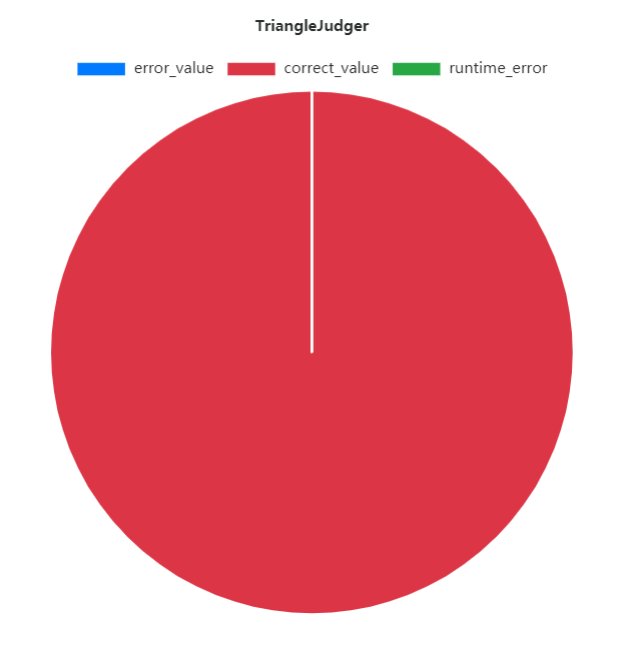




代码测试全部通过

### 等价类法





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 |

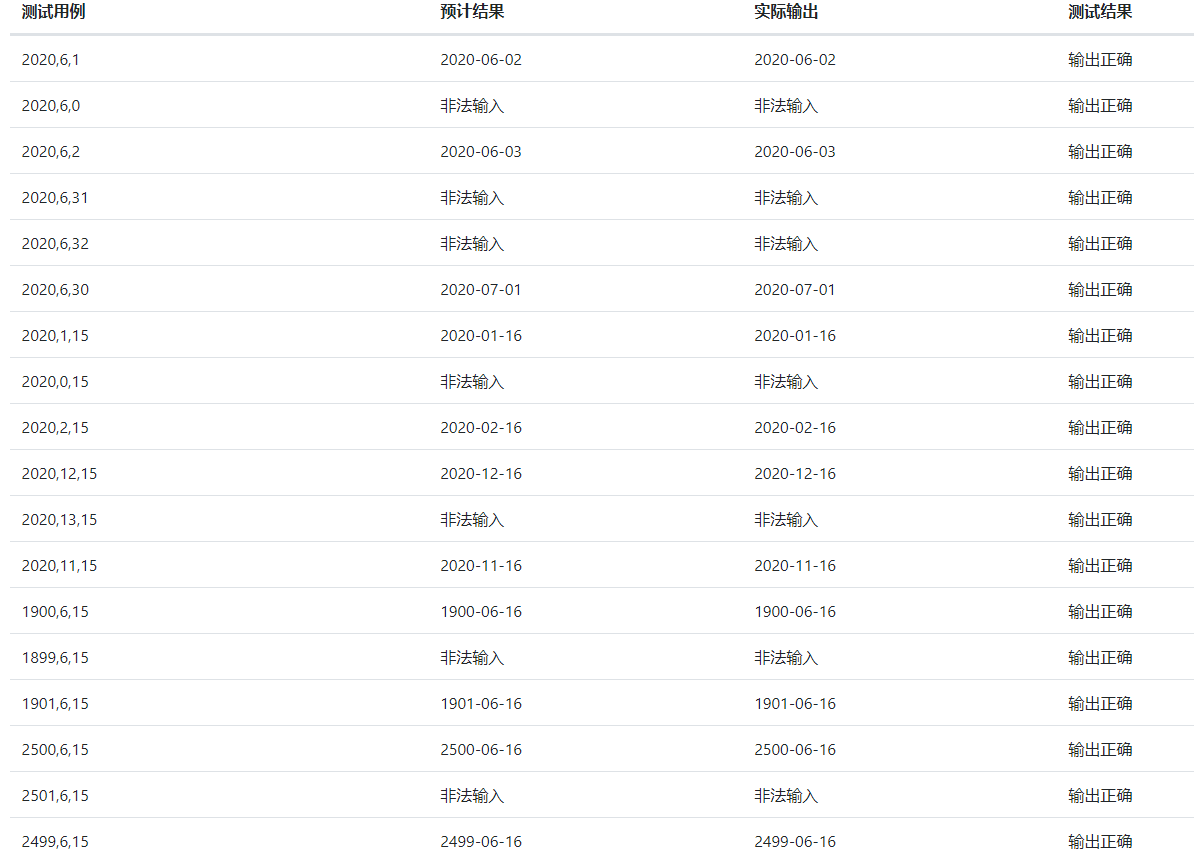
## 测试代码设计

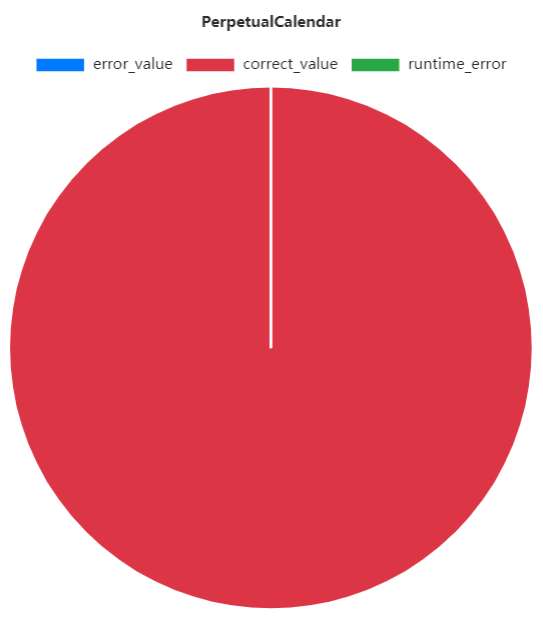
1. **import** java.text.SimpleDateFormat;
2. **import** java.util.Calendar;
4. **public** **class** PerpetualCalendar{
5. **public** **static** String getNextDay(**int** y,**int** m,**int** d){
6. **if**(y<1900||y>2500||m<1||m>12){
7. **return** "非法输入";
8. }
9. SimpleDateFormat dateFormat=**new** SimpleDateFormat("yyyy-MM-dd");
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：

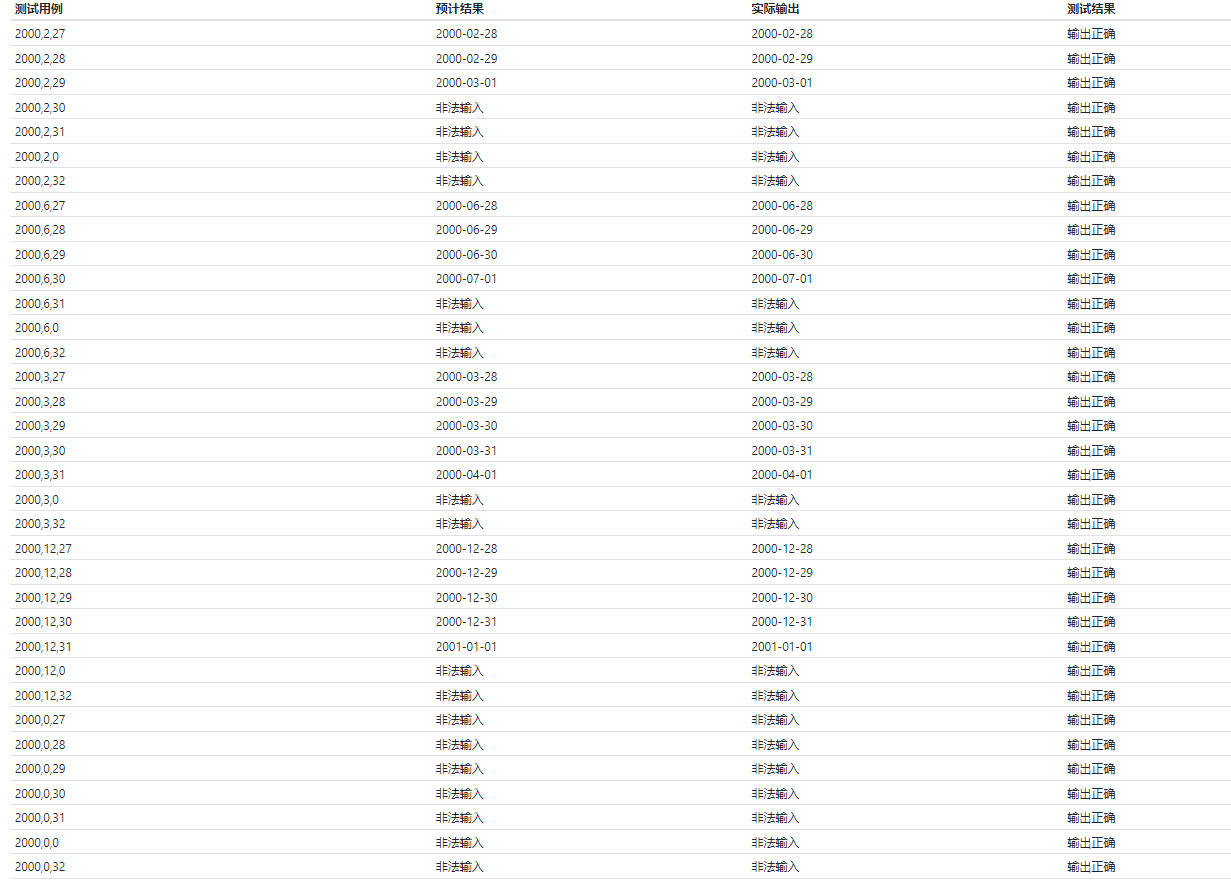
### 边界值法

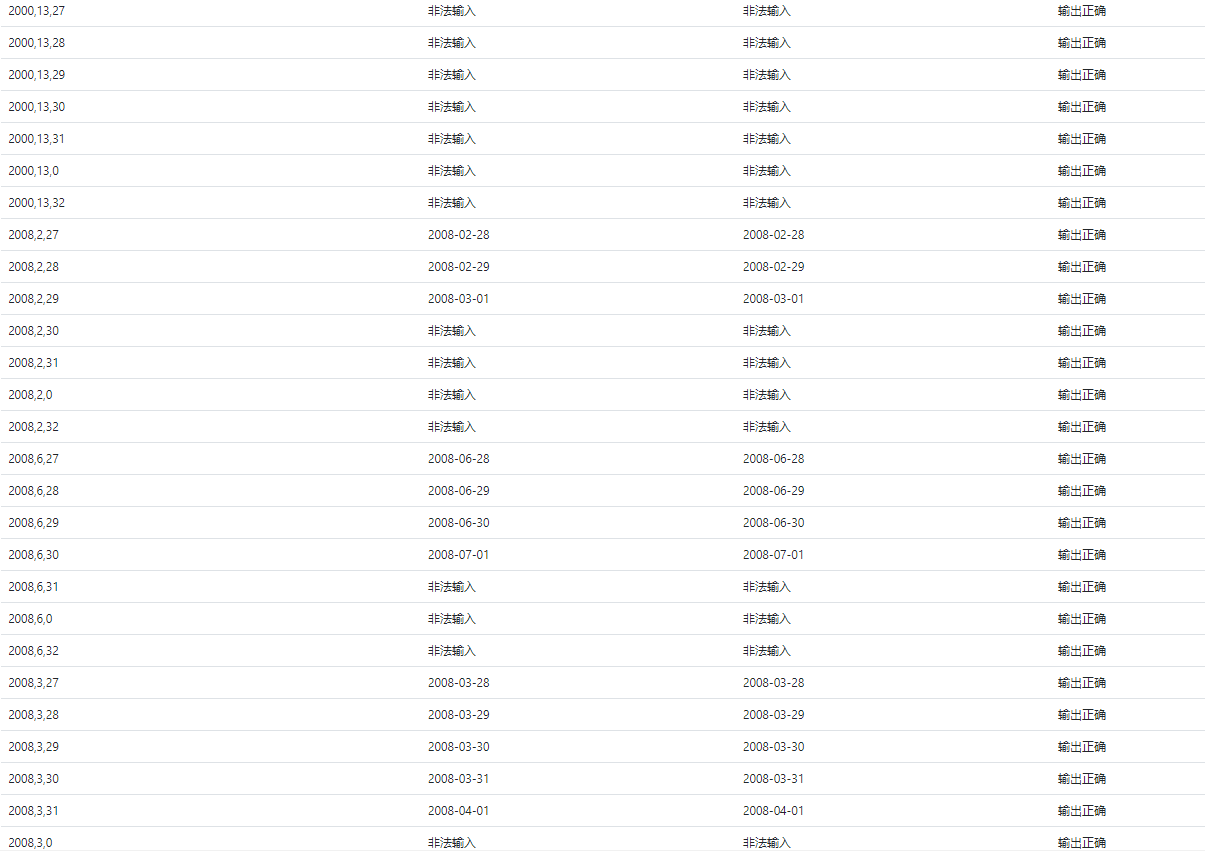


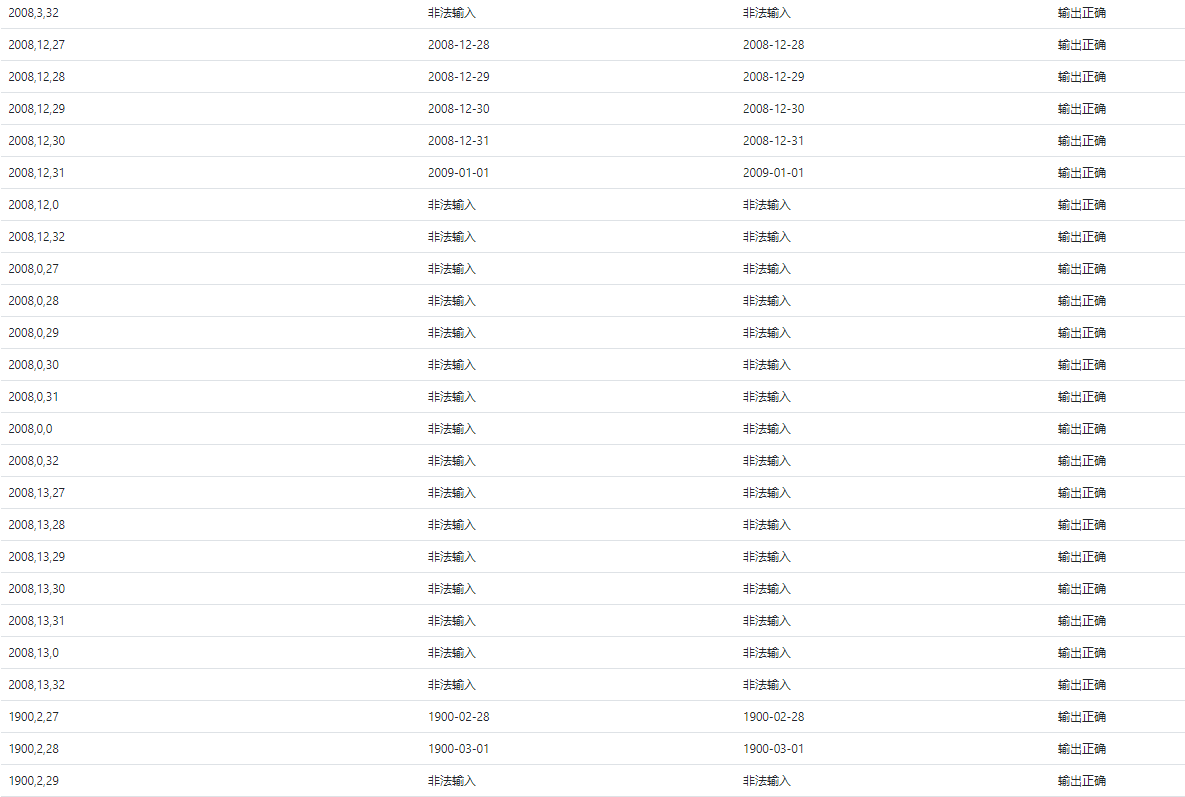


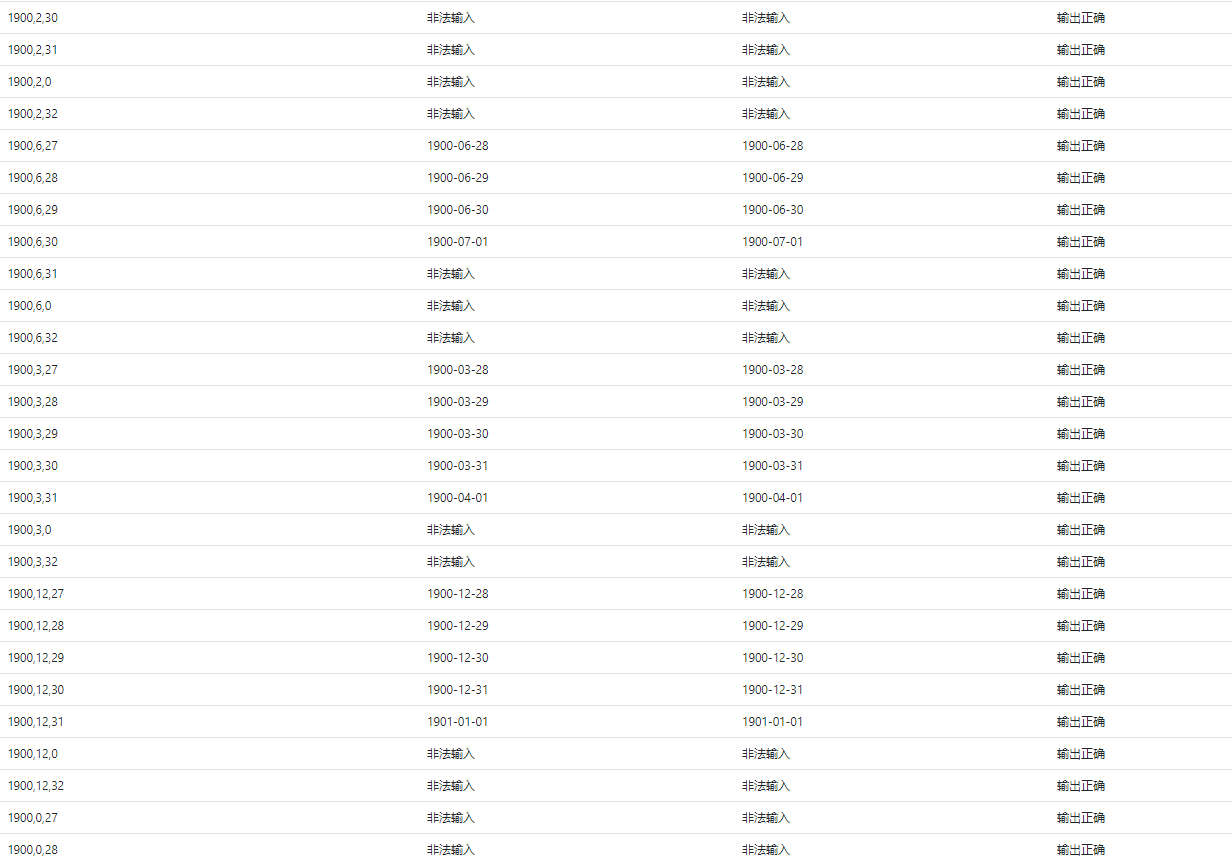
代码测试全部通过

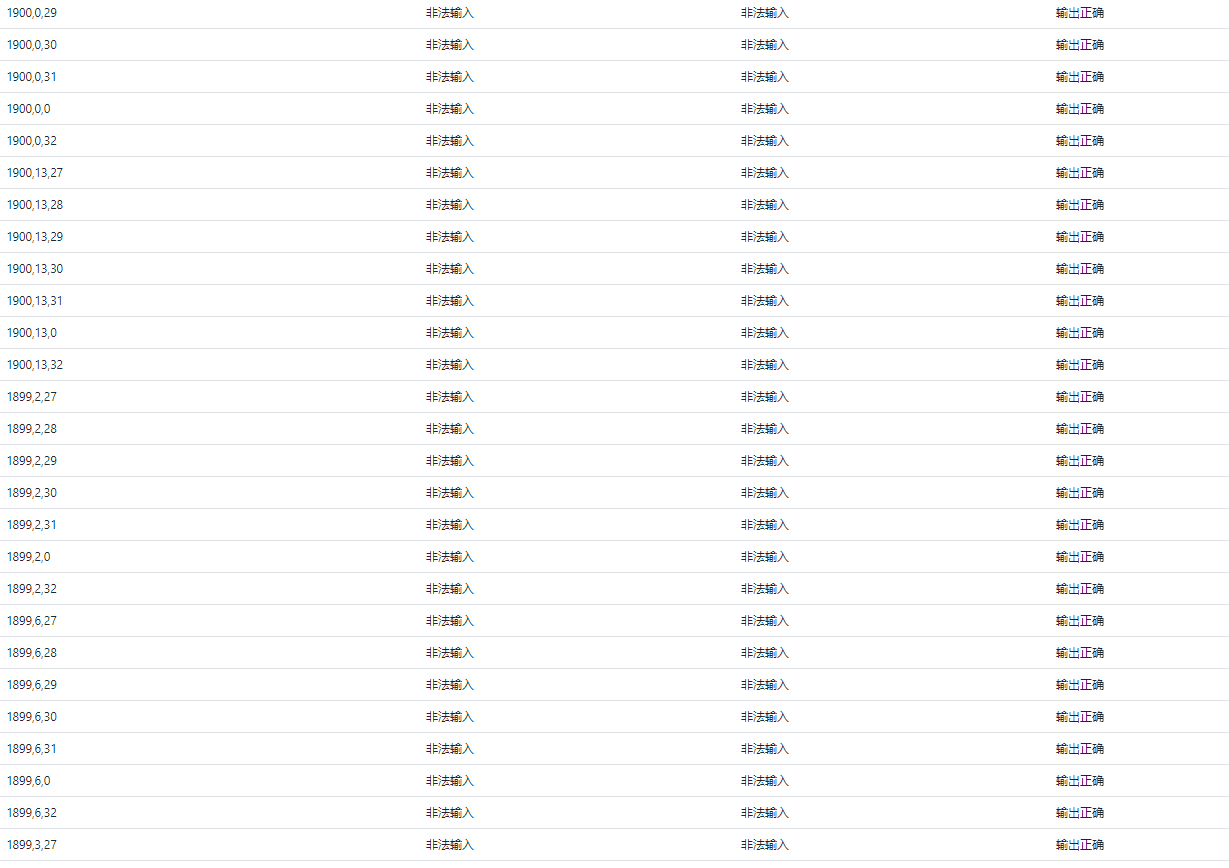
### 等价类法



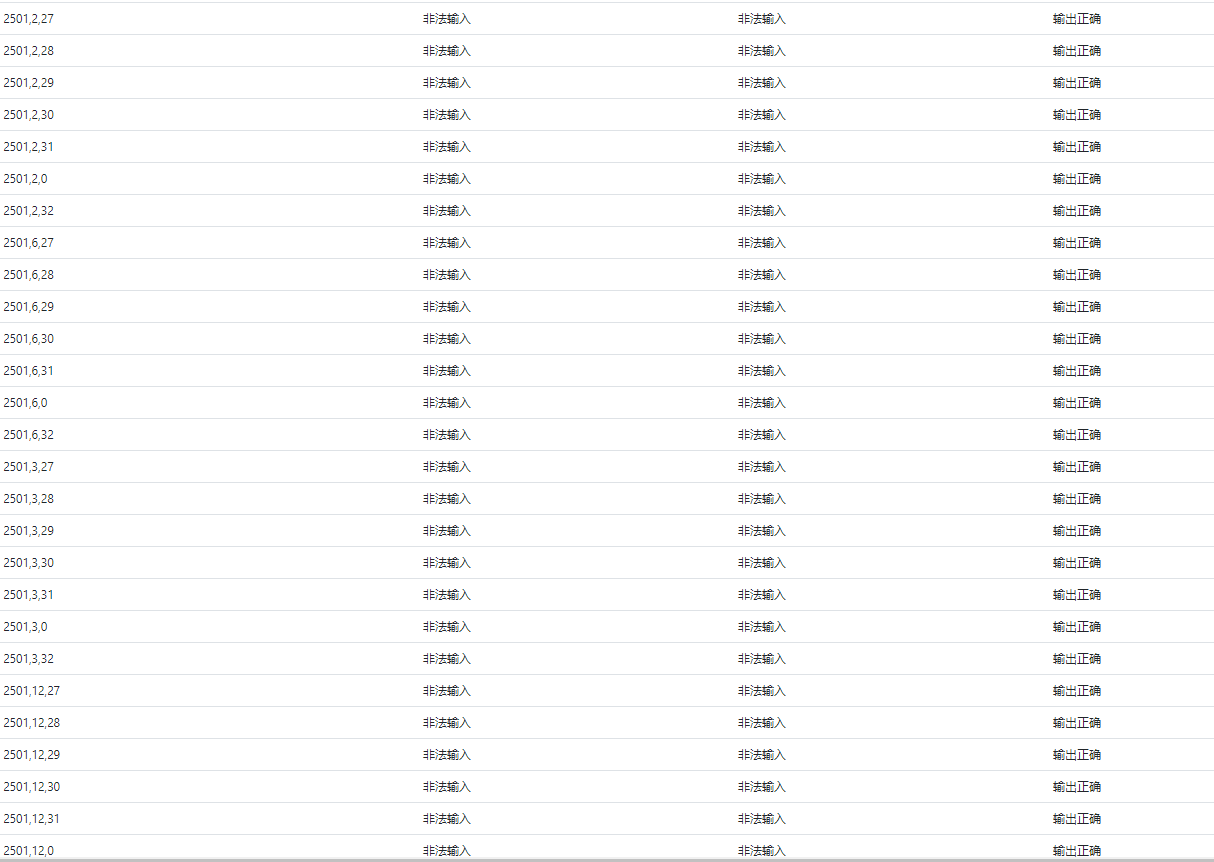


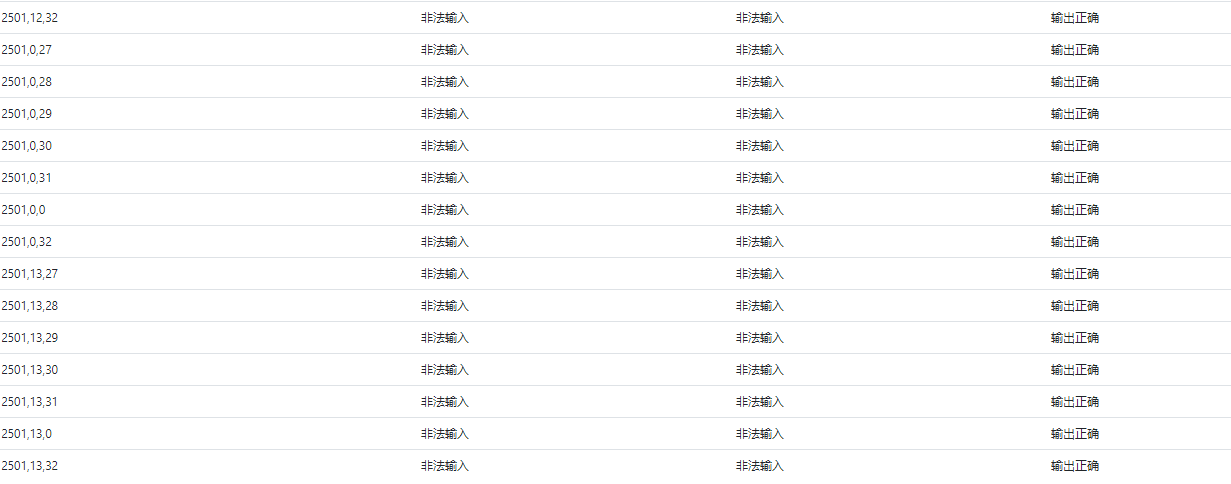














代码测试全部通过

# 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

## 用例设计

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 |

## 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;

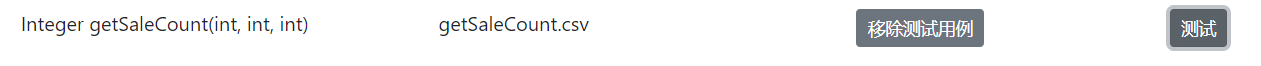
31. **private** **void** setDeviceNum(**int** hostNum, **int** screenNum, **int** peripheralNum){
32. **this**.hostNum = hostNum;
33. **this**.screenNum = screenNum;
34. **this**.peripheralNum = peripheralNum;
35. }
37. **public** Integer getSaleCount(**int** hostNum, **int** screenNum, **int** peripheralNum){
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

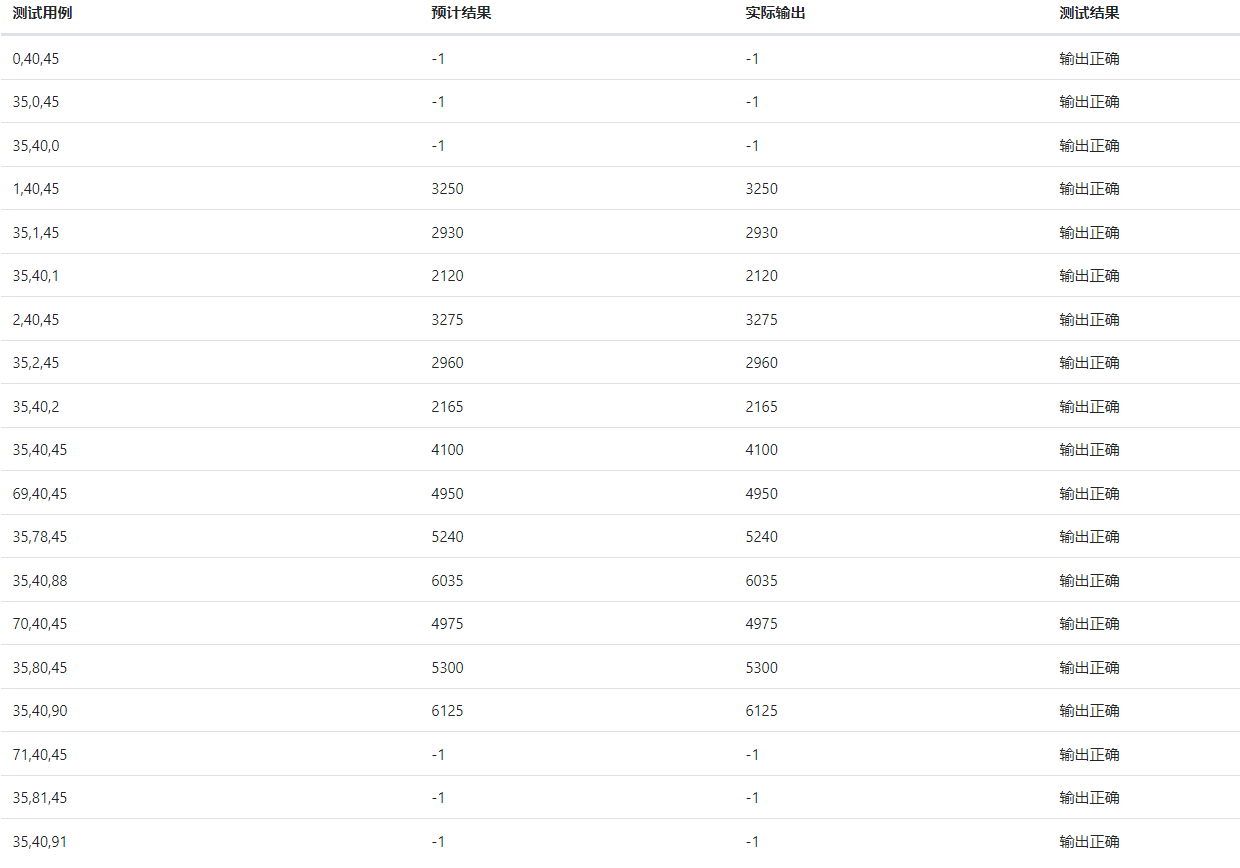
使用自制代码测试工具对编写代码加以测试，得到如下结果：

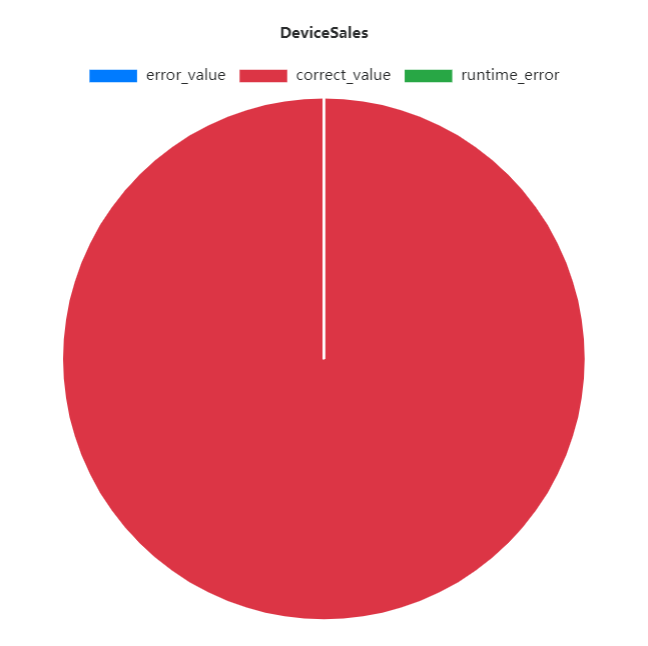
### 销售额输出测试

通过测试工具调用getSaleCount方法



结果如下：





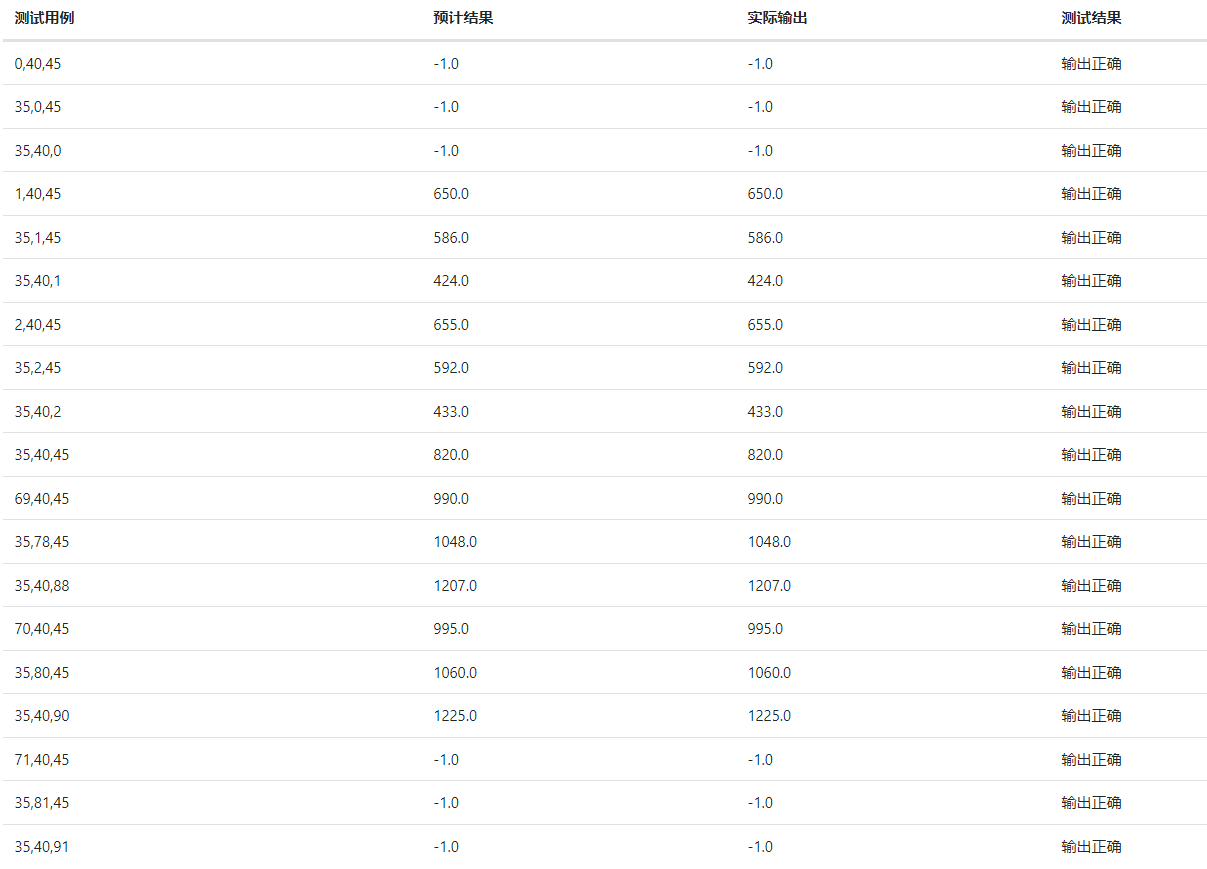
代码测试全部通过

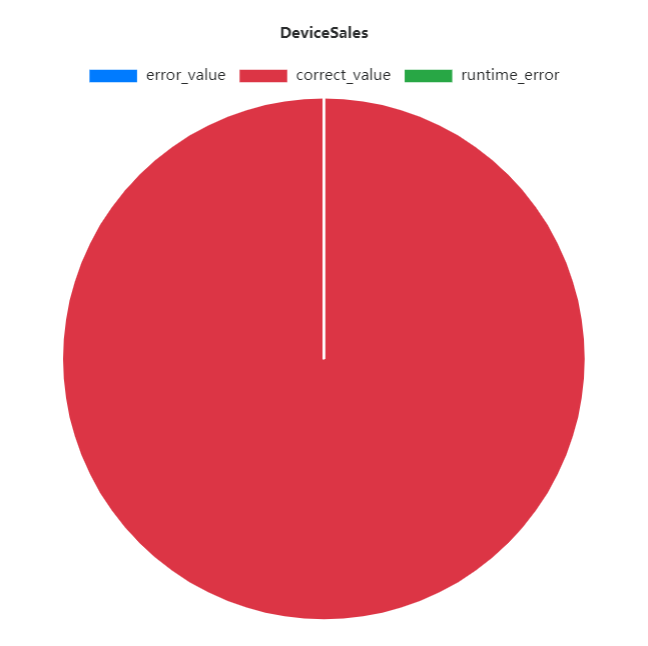
### 佣金输出测试

通过测试工具调用getBrokerage方法



结果如下：





代码测试全部通过

# 练习3—使用Junit实践并了解原理

# 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(true\_result.equals(return\_v));
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

接下来来举例说明Sub模块的作用。

在下面这段代码中，待测模块的PrintUsers方法需要调用类UserDao的getUsers方法从数据库中获取所有的用户名，但由于数据库还未构建完成，需要使用Stub模块来获取相应信息。于是我们定义了一个类UserDaoStub继承自UserDao，并重写了getUsers方法，从而让待测模块能够正常运行。

1. **public** **class** UserDao{
2. **public** String[] getUsers(){
3. // 还未实现
4. **return** **null**;
5. }
6. }
8. **public** **class** UserDaoStub **extends** UserDao{
9. @Override
10. **public** String[] getUsers() {
11. **return** **new** String[]{"XiaoMing","XiaoHong","ZhangSan"};
12. }
13. }
15. **public** **class** TestClass {
16. **private** UserDao userDao=**new** UserDaoStub();
17. **public** **void** PrintUsers(){
18. var items=userDao.getUsers();
19. **for**(var item:items){
20. System.out.println(item);
21. }
22. }
23. }

### 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

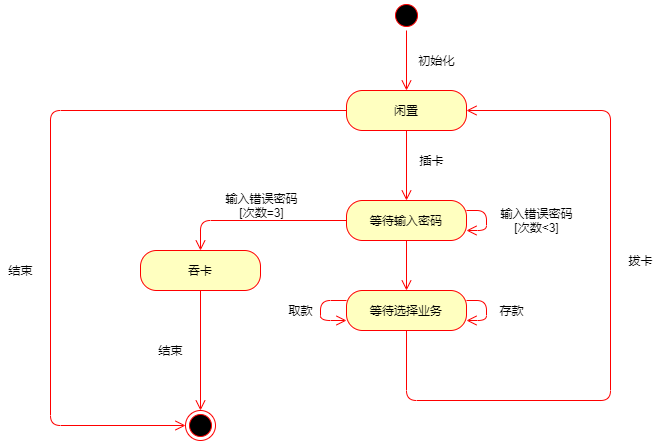
# 练习6—设计transition tree 算法

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

这里我们以ATM系统为例进行分析

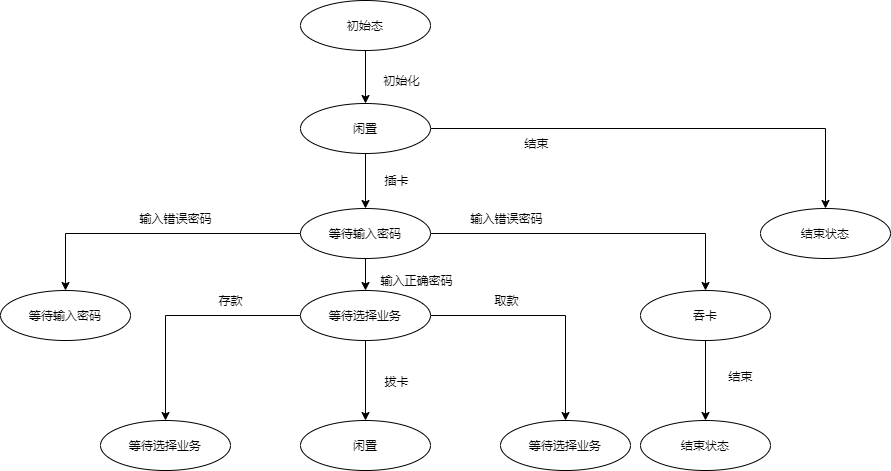
## 系统状态图

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 |
| 结束态： | 用户结束操作 |

接下来这是根据State Transition Tree中每一个转换过程进行构建，结果如下表所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 当前状态 | 触发该转换行为 | 该转换的预期效果 | 下一状态 |
| 闲置 | 用户插入银行卡 | ATM机进入登录界面 | 等待用户输入密码 |
| 等待用户输入密码 | 用户输入错误密码，且总次数小于3 | ATM机回到登录界面 | 等待用户输入密码 |
| 等待用户输入密码 | 用户输入错误密码，且总次数等于3 | ATM机吞掉用户银行卡 | 吞卡 |
| 吞卡 | 用户结束操作 | 无 | 结束态 |
| 等待用户输入密码 | 用户输入正确密码 | ATM机进入业务办理界面 | 等待选择业务 |
| 等待选择业务 | 用户选择存款 | ATM机完成存款业务 | 等待选择业务 |
| 等待选择业务 | 用户选择取款 | ATM机完成取款业务 | 等待选择业务 |
| 等待选择业务 | 用户拔出银行卡 | ATM机回到闲置页面 | 闲置 |
| 闲置 | 用户结束操作 | 无 | 结束态 |

如上表所示，在该测试用例下，可以达到ATM机的所有状态以及所有转换。