

# **软件质量保证与测试上机实验**

## 目录

一 测试中经典问题的实现 .....	1
1 三角形问题 .....	1
1.1 三角形问题描述 .....	1
1.2 三角形问题分析与设计流程图 .....	1
1.3 三角形问题实现（代码） .....	2
2 次日问题 .....	4
2.1 次日问题描述 .....	4
2.2 次日问题分析与设计（画流程图） .....	5
2.3 次日问题实现（代码） .....	5
二 黑盒测试 .....	9
1 三角形问题的黑盒测试用例设计 .....	9
1.1 边界值分析 .....	9
1.2 等价类划分 .....	10
1.3 决策表 .....	13
2 次日问题的黑盒测试用例设计 .....	14
2.1 边界值分析 .....	14
2.2 等价类划分 .....	15
2.3 决策表分析 .....	17
三 白盒测试 .....	20
1 三角形问题的白盒测试用例设计 .....	20
1.1 语句覆盖 .....	20
1.2 判定覆盖 .....	21
1.3 条件覆盖 .....	21
2 次日问题的白盒测试用例设计 .....	22
2.1 语句覆盖 .....	22
2.2 判定覆盖 .....	23
2.3 路径覆盖 .....	23
四 Eclipse 环境下用 Junit 进行单元测试 .....	25
1 Eclipse 环境下用 Junit 环境配置 .....	25
2 JUnit 测试脚本的设计 .....	25

2.1 三角形的测试脚本 .....	25
2.2 次日问题的测试脚本 .....	29
3 测试结果 .....	33
3.1 三角形问题 .....	33
3.2 次日问题 .....	42
五 测试结果分析 .....	42
1 缺陷分析 .....	42
1.1 缺陷记录 .....	42
1.2 发现缺陷的用例及设计方法 .....	42
2 测试总结 .....	43

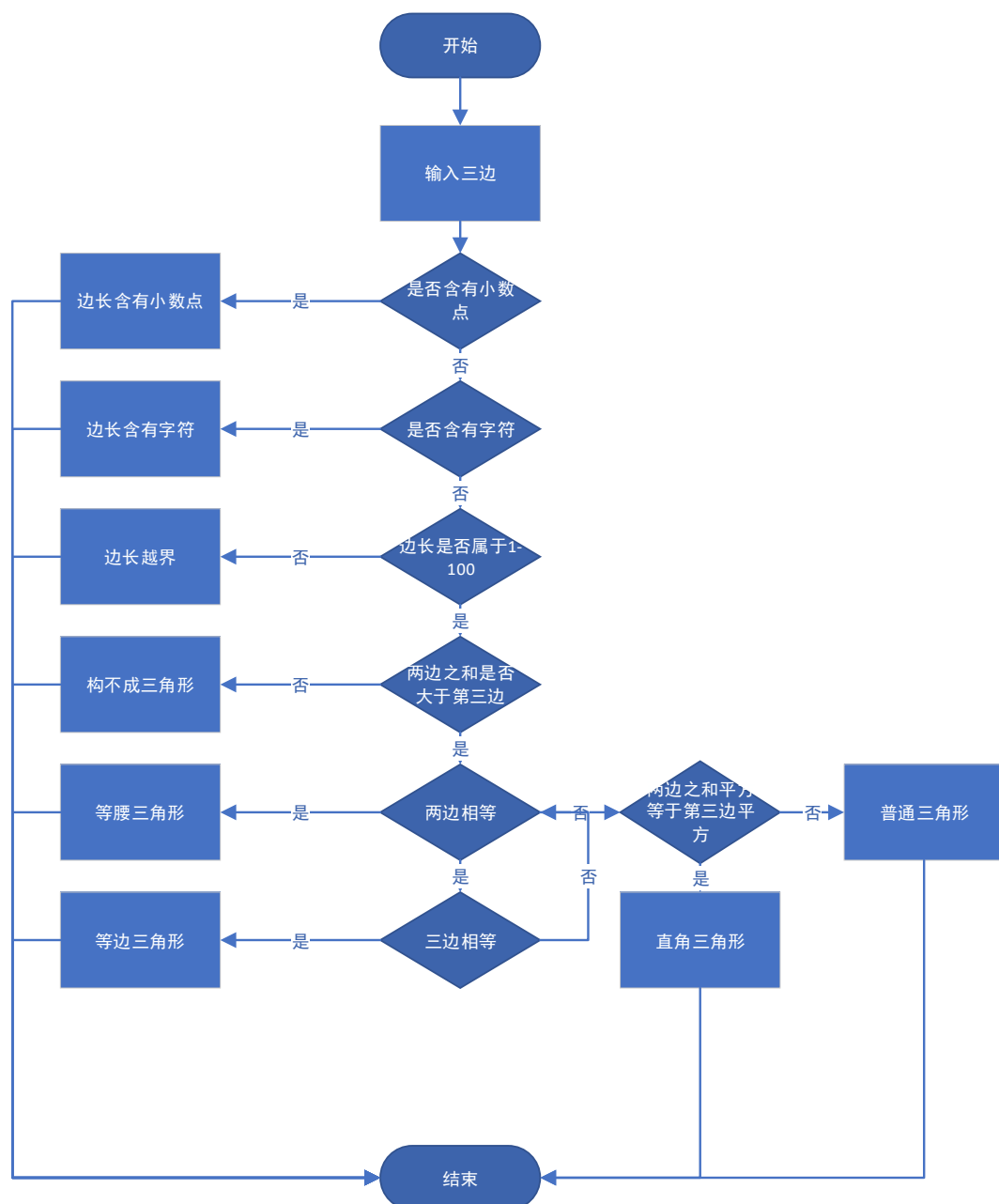
# 一 测试中经典问题的实现

## 1 三角形问题

### 1.1 三角形问题描述

从一个输入对话框中读取三个整数，这三个整数值代表了三角形三边的长度。程序显示提示信息，指出该三角形究竟是不是不规则三角形、等腰三角形、还是等边三角形。

### 1.2 三角形问题分析与设计流程图



### 1.3 三角形问题实现（代码）

```
package edu.dlu.triangle;
```

```
public class Demo1 {  
    public static void main(String[] args) {  
        new Util().isTriangle();  
    }  
}
```

```
package edu.dlu.triangle;
```

```
public class Util {
```

```
    // 判断字符串是否为数字
```

```
    public static boolean isNum(String str) {  
        return str.matches("^[-+]?([0-9]+)([.]( [0-9]+)?)?([.]( [0-9]+)?)?$");  
    }
```

```
    // 判断字符串是否含有"."分隔符
```

```
    public static boolean containsDot(String str) {  
        if (str.contains("."))  
            return true;  
        return false;  
    }
```

```
    // 判断字符串是否为空
```

```
    public static boolean isEmpty(String str) {  
        if ("".equals(str) || str == null)  
            return true;  
        else  
            return false;  
    }
```

```
    public void isTriangle(String strA, String strB, String strC) {
```

```

    if (!(isEmpty(strA) || isEmpty(strB) || isEmpty(strC))) {
        if (!(containsDot(strA) || containsDot(strB) ||
containsDot(strC))) {
            if (isNum(strA) && isNum(strB) && isNum(strC)) {
                int a = Integer.parseInt(strA);
                int b = Integer.parseInt(strB);
                int c = Integer.parseInt(strC);
                if ((a > 0 && a <= 100) && (b > 0 && b <= 100) &&
(c > 0 && c <= 100)) {
                    if (a + b <= c || a + c <= b || b + c <= a) {
                        System.out.println("构不成三角形");
                    } else {
                        if (a == b && b == c) {
                            System.out.println("等边三角形");
                        } else if (a == b || a == c || b == c) {
                            System.out.println("等腰三角形");
                        } else if ((a * a + b * b == c * c) || (a * a + c * c
== b * b)
|| (b * b + c * c == a * a)) {
                            System.out.println("直角三角形");
                        } else {
                            System.out.println("一般三角形");
                        }
                    }
                } else {
                    System.out.println("构不成三角形, 边长越界! ");
                }
            } else {
                System.out.println("构不成三角形, 边长含字符! ");
            }
        } else {
            System.out.println("边长不是整数");
        }
    } else {

```

```

        System.out.println("数据不完整");
    }
}
}

```

## 2 次日问题

### 2.1 次日问题描述

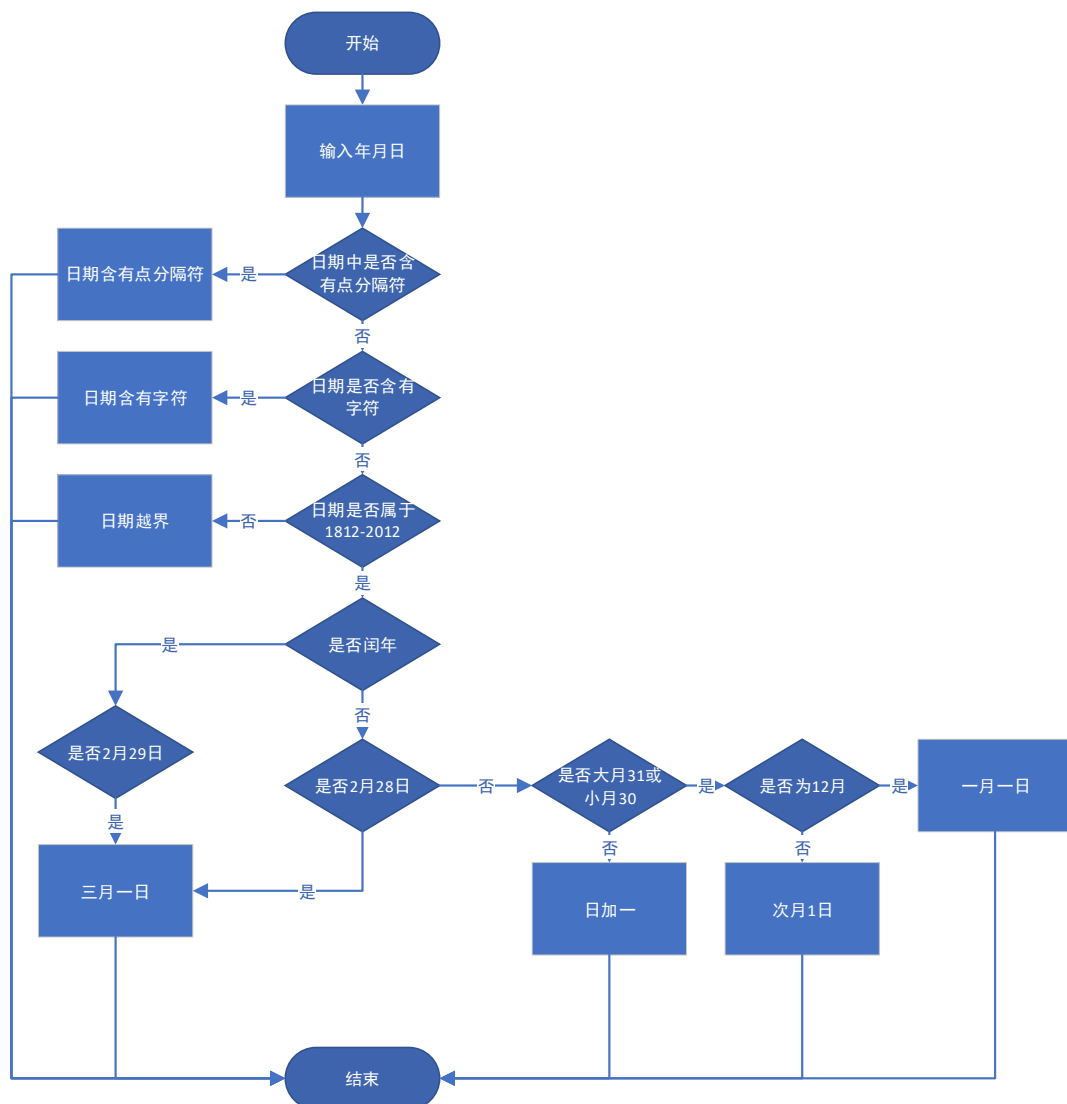
输入年月日，其中年份的有效取值范围为【1900，2100】，请输出输入日期的下一天，例如输入 2013 年 9 月 29 日，输出为 2013 年 9 月 30 日。若输入日期非法，例如输入 2013 年 2 月 30 日，则输出“输入日期不存在”，若输入日期超出取值范围，例如输入 2013 年 9 月 32 日，则输出“输入日期超出范围”。

Nextday 函数：

分析不同情况下的输入，可以得到不同输出

- (1) 当输入数值只要有一个不符合范围，提示“不合理范围”
- (2) 当输入非月头月尾数值时，输出日期将天数加 1
- (3) 当输入非 2 月月尾时，输出日期将月份加 1
- (4) 当输入的为年尾时，输出日期将年份加 1
- (5) 当输入闰年 28 日时，输出日期将天数加 1
- (6) 当输入闰年 29 日时，输出日期将月份加 1
- (7) 当输入非闰年 27 日时，输出日期将天数加 1
- (8) 当输入非闰年 28 日时，输出日期将月份加 1
- (9) 当输入非闰年 29 日时，提示“输入错误”

## 2.2 次日问题分析与设计（画流程图）



## 2.3 次日问题实现（代码）

```
package edu.dlu.nextDay;
public class Demo {
    public static void main(String[] args) {
        new Util().nextDay();
    }
}
```

```
package edu.dlu.nextDay;
```



```

public class Util {
    // 判断字符串是否为数字
    public static boolean isNum(String str) {
        return str.matches("^[-+]?([0-9]+)([.]( [0-9]+ )?)?([.]( [0-9]+ )?)?$");
    }

    // 判断字符串是否含有"."分隔符
    public static boolean containsDot(String str) {
        if (str.contains("."))
            return true;
        return false;
    }

    // 判断字符串是否为空
    public static boolean isEmpty(String str) {
        if ("".equals(str) || str == null)
            return true;
        else
            return false;
    }

    // 判断是否是闰年
    public static boolean isLeapYear(int year) {
        if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
            return true;
        } else {
            return false;
        }
    }

    public void nextDay(String yearOfDate, String monthOfDate, String
dayOfDate) {
        if (!(isEmpty(yearOfDate) || isEmpty(monthOfDate) ||
isEmpty(dayOfDate))) {

```

```

        if (!(containsDot(yearOfDate) || containsDot(monthOfDate) ||
containsDot(dayOfDate))) {
            if (isNum(yearOfDate) && isNum(monthOfDate) &&
isNum(dayOfDate)) {
                int year = Integer.parseInt(yearOfDate);
                int month = Integer.parseInt(monthOfDate);
                int day = Integer.parseInt(dayOfDate);
                if (year <= 2012 && year >= 1812) {
                    if (month <= 12 && month >= 1) {
                        if (day <= 31 && day >= 1) {
                            if (isLeapYear(year) && (month == 2) &&
(day == 29)) {
                                System.out.println("下一日是: " + year
+ "年3月1日");
                            } else if ((month == 2) && (day == 28)) {
                                System.out.println("下一日是: " + year
+ "年3月1日");
                            } else if ((month == 12) && (day == 31)) {
                                year = year + 1;
                                System.out.println("下一日是: " + year
+ "年1月1日");
                            } else if (((month == 1) || (month == 3) ||
(month == 5) || (month == 7) || (month == 8)
|| (month == 10)) && (day == 31))
{
                                    month = month + 1;
                                    System.out.println("下一日是: " + year
+ "年" + month + "月1日");
                                } else if (((month == 4) || (month == 6) ||
(month == 9) || (month == 11))
&& (day == 30)) {
                                    month = month + 1;
                                    System.out.println("下一日是: " + year
+ "年" + month + "月1日");
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

        } else if (!isLeapYear(year) && (month ==
2) && (day > 28)) {
            System.out.println("本年2月没有那么多
天");
        } else if (((month == 4) || (month == 6) ||
(month == 9) || (month == 11))
            && (day == 31)) {
            System.out.println(month + "月可没有
31天");
        } else {
            day = day + 1;
            System.out.println("下一日是:" + year +
"年" + month + "月" + day + "日");
        }
        } else {
            System.out.println("日越界");
        }
        } else {
            System.out.println("月越界");
        }
        } else {
            System.out.println("年越界");
        }
        } else {
            System.out.println("日期不合法，含异常字符。");
        }
        } else {
            System.out.println("日期不是整数");
        }
        } else {
            System.out.println("日期数据不全");
        }
    }
}

```

## 二 黑盒测试

### 1 三角形问题的黑盒测试用例设计

#### 1.1 边界值分析

人们从长期的测试工作经验得知，大量的错误是发生在输入或输出范围的边界上的，而不是在输入范围的内部。因此针对各种边界情况设计测试用例，可以查出更多的错误。例如，在做三角形计算时，要输入三角形的3个边长A、B和C。这3个数值应当满足 $A>0$ 、 $B>0$ 、 $C>0$ 、 $A+B>C$ 、 $A+C>B$ 、 $B+C>A$ ，才能构成三角形。但如果把6个不等式中的任何一个大于号“ $>$ ”错写成大于等于号“ $\geq$ ”，那就不能构成三角形。问题恰恰出现在容易被疏忽的边界附近。这里所说的边界是指相当于输入等价类和输出等价类而言，稍高于其边界值及稍低于其边界值的一些特定情况。

##### 1.1.1 一般边界值测试

在三角形问题描述中，除了要求边长是整数外，没有给出其它的限制条件。在此，我们将三角形每边边长的取范围值设值为 $[1, 100]$ 。

表 1 三角形问题的一般边界值测试					
测试用例	a	b	c	预期输出	实际输出
1	60	60	1	等腰三角形	等腰三角形
2	60	60	2	等腰三角形	等腰三角形
3	60	60	60	等边三角形	等边三角形
4	50	50	99	等腰三角形	等腰三角形
5	50	50	100	构不成三角形	构不成三角形
6	60	1	60	等腰三角形	等腰三角形
7	60	2	60	等腰三角形	等腰三角形
8	60	60	60	等边三角形	等边三角形
9	50	99	50	等腰三角形	等腰三角形
10	50	100	50	构不成三角形	构不成三角形
11	1	60	60	等腰三角形	等腰三角形
12	2	60	60	等腰三角形	等腰三角形
13	60	60	60	等边三角形	等边三角形
14	99	50	50	等腰三角形	等腰三角形
15	100	50	50	构不成三角形	构不成三角形

### 1.1.2 健壮边界值测试

表 2 三角形问题的健壮性边界值测试				
测试用例编号	数据列表			预期输出
	a	b	c	
1	0	50	50	抛出异常
2	1	50	50	等腰三角形
3	2	50	50	等腰三角形
4	50	50	50	等边三角形
5	99	50	50	等腰三角形
6	100	50	50	构不成三角形
7	101	50	50	抛出异常
8	50	0	50	抛出异常
9	50	1	50	等腰三角形
10	50	2	50	等腰三角形
11	50	99	50	等腰三角形
12	50	100	50	构不成三角形
13	50	101	50	抛出异常
14	50	50	0	抛出异常
15	50	50	1	等腰三角形
16	50	50	2	等腰三角形
17	50	50	99	等腰三角形
18	50	50	100	构不成三角形
19	50	50	101	抛出异常

## 1.2 等价类划分

等价类划分是一种典型的黑盒测试方法，用这一方法设计测试用例完全不考虑程序的内部结构，只根据对程序的要求和说明，即需求规格说明书。我们必须仔细分析和推敲说明书的各项需求，特别是功能需求。把说明中对输入的要求和输出的要求区别开来并加以分解。

等价类划分的办法是把程序的输入域划分成若干部分，然后从每个部分中选取少数代表性数据作为测试用例。每一类的代表性数据在测试中的作用等价于这一类中的其他值，也就是说，如果某一类中的一个例子发现了错误，这一等价类中的其他例子也能发现同样的错误；反之，如果某一类中的一个例子没有发现错误，则这一类中的其他例子也不会查出错误（除非等价类中的某些例子属于另一

等价类，因为几个等价类是可能相交的)。使用这一方法设计测试用例，首先必须在分析需求规格说明的基础上划分等价类，列出等价类表。

### 1.2.1 三角形等价类划分描述

三角形等价类划分描述			
输入条件	输入三个整数	有效等价类	无效等价类
		1. 整数 2. 三个整数 3. 非 0 整数 4. 正数	13. 一边为非整数 14. 两边为非整数 15. 三边为非整数 16. 边长个数小于 3 17. 边长个数大于 3 18. 一边为 0 19. 两边为 0 20. 三边为 0 21. 一边小于 0 22. 两边小于 0 23. 三边小于 0 24. 一边大于 100 25. 两边大于 100 26. 三边大于 100
输出条件	构成普通三角形	5. $a+b>c$ 6. $a+c>b$ 7. $b+c>a$	27. $a+b\leq c$ 28. $b+c\leq a$ 29. $c+a\leq b$
	构成等腰三角形	8. $a=b$ 且满足 5 9. $a=c$ 且满足 6 10. $b=c$ 且满足 7	
	构成等边三角形	11. $a=b=c$	
	构不成三角形	12. 至少不满足 5, 6, 7 中的一个	

### 1.2.2 三角形等价类划分测试用例

表 3 三角形问题等价类划分测试						
测试用例编号	等价类条件	测试用例				预期结果
		a	b	c	d	
1	1, 2, 3, 4, 5, 6, 7	2	3	4		普通三角形
2	1, 2, 3, 4, 5, 6, 7, 8	2	2	3		等腰三角形
3	1, 2, 3, 4, 5, 6, 7, 9	2	3	2		等腰三角形
4	1, 2, 3, 4, 5, 6, 7, 10	3	2	2		等腰三角形
5	1, 2, 3, 4, 5, 6, 7, 11	2	2	2		等边三角形
6	1, 2, 3, 4, 12	1	2	3		构不成三角形
7	1, 2, 3, 4, 5, 6, 7, 13	2.1	3	4		抛出异常
8	1, 2, 3, 4, 5, 6, 8, 14	2.1	2.1	3		抛出异常
9	1, 2, 3, 4, 5, 6, 8, 15	2.1	2.1	2.1		抛出异常
10	1, 3, 4, 16	2	3	空		未测试
11	1, 2, 3, 17	2	3	4	3	未测试
12	1, 2, 18	0	2	3		构不成三角形, 边长越界
13	1, 2, 19	0	0	2		构不成三角形, 边长越界
14	1, 2, 20	0	0	0		构不成三角形, 边长越界
15	1, 2, 3, 21	-1	2	2		构不成三角形, 边长越界
16	1, 2, 3, 22	-1	-2	2		构不成三角形, 边长越界
17	1, 2, 3, 23	-2	-2	-2		构不成三角形, 边长越界
18	1, 2, 3, 4, 24	111	99	99		构不成三角形, 边长越界
19	1, 2, 3, 4, 25	111	111	99		构不成三角形, 边长越界
20	1, 2, 3, 4, 26	111	111	111		构不成三角形, 边长越界
21	1, 2, 3, 4, 27	5	3	2		构不成三角形
22	1, 2, 3, 4, 28	5	2	3		构不成三角形
23	1, 2, 3, 4, 29	2	5	3		构不成三角形

## 1.3 决策表

在一些数据处理问题中，某些操作是否实施依赖于多个逻辑条件的取值。在这些逻辑条件取值的组合所构成的多种情况下，分别执行不同的操作。处理这类问题的一个非常有力的分析和表达工具是判定表，或称决策表（Decision Table）。

在所有功能性测试方法中，基于判定表的测试方法是最严格的。

### 1.3.1 三角形问题的决策表

C1:a<b+c?	F	T	T	T	T	T	T	T	T	T	T
C2:b<a+c?	-	F	T	T	T	T	T	T	T	T	T
C3:c<a+b?	-	-	F	T	T	T	T	T	T	T	T
C4:a=b?	-	-	-	T	T	T	T	F	F	F	F
C5:a=c?	-	-	-	T	T	F	F	T	T	F	F
C6:b=c?	-	-	-	T	F	T	F	T	F	T	F
A1:	X	X	X								
A2:											X
A3:							X		X	X	
A4:				X							
A5:					X	X		X			

### 1.3.2 基于决策表的三角形测试用例

表 4 三角形问题的决策表测试					
用例编号	a	b	c	预期输出	实际输出
1	4	1	2	构不成三角形	构不成三角形
2	1	4	2	构不成三角形	构不成三角形
3	1	2	4	构不成三角形	构不成三角形
4	5	5	5	等边三角形	等边三角形
5	?	?	?	不可能	未测试
6	?	?	?	不可能	未测试
7	2	2	3	等腰三角形	等腰三角形



8	?	?	?	不可能	未测试
9	2	3	2	等腰三角形	等腰三角形
10	3	2	2	等腰三角形	等腰三角形
11	3	4	5	普通三角形	普通三角形

## 2 次日问题的黑盒测试用例设计

### 2.1 边界值分析

Month 的取值范围为：[1, 12]

Day 的取值范围为：[1, 31]

Year 的取值范围为：[1812, 2012]

表 5 次日问题的边界值测试					
用例编号	Year	Month	Day	期望输出	实际输出
1	1811	6	15	年越界	年越界
2	1812	6	15	1812 年 6 月 16 日	1812 年 6 月 16 日
3	1813	6	15	1813 年 6 月 16 日	1813 年 6 月 16 日
4	1912	6	15	1912 年 6 月 16 日	1912 年 6 月 16 日
5	2011	6	15	2011 年 6 月 16 日	2011 年 6 月 16 日
6	2012	6	15	2012 年 6 月 16 日	2012 年 6 月 16 日
7	2013	6	15	年越界	年越界
8	1912	6	-1	日越界	日越界
9	1912	6	1	1912 年 6 月 2 日	1912 年 6 月 2 日
10	1912	6	2	1912 年 6 月 3 日	1912 年 6 月 3 日
11	1912	6	29	1912 年 6 月 30 日	1912 年 6 月 30 日

12	1912	6	30	1912 年 7 月 1 日	1912 年 7 月 1 日
13	1912	6	31	6 月可没有 31 天	6 月可没有 31 天
14	1912	6	32	日越界	日越界
15	1912	-1	15	月越界	月越界
16	1912	1	15	1912 年 1 月 16 日	1912 年 1 月 16 日
17	1912	2	15	1912 年 2 月 16 日	1912 年 2 月 16 日
18	1912	11	15	1912 年 11 月 16 日	1912 年 11 月 16 日
19	1912	12	15	1912 年 12 月 16 日	1912 年 12 月 16 日
20	1912	13	15	月越界	月越界

## 2.2 等价类划分

NextDate 函数包含三个变量：month、day 和 year，函数的输出为输入日期后一天的日期。例如，输入为 2006 年 3 月 7 日，则函数的输出为 2006 年 3 月 8 日。要求输入变量 month、day 和 year 均为整数值，并且满足下列条件：

- ①  $1 \leq \text{month} \leq 12$
- ②  $1 \leq \text{day} \leq 31$
- ③  $1920 \leq \text{year} \leq 2050$

1) 有效等价类为：

$$\begin{aligned} M1 &= \{\text{月份: } 1 \leq \text{月份} \leq 12\} \\ D1 &= \{\text{日期: } 1 \leq \text{日期} \leq 31\} \\ Y1 &= \{\text{年: } 1812 \leq \text{年} \leq 2012\} \end{aligned}$$

2) 若条件 ① ~ ③中任何一个条件失效，则 NextDate 函数都会产生一个输出，指明相应的变量超出取值范围，比如“month 的值不在 1-12 范围当中”。显然还存在着大量的 year、month、day 的无效组合，NextDate 函数将这些组合作统一的输出：“无效输入日期”。其无效等价类为：

$$M2 = \{\text{月份: 月份} < 1\}$$

M3={月份: 月份>12}

D2={日期: 日期<1}

D3={日期: 日期>31}

Y2={年: 年<1812}

Y3={年: 年>2012}

健壮性等价类

表 6 次日问题的等价类测试					
用例编号	Year	Month	Day	预期输出	实际输出
1	1812	6	30	下一日是: 1812年7月1日	下一日是: 1812年7月1日
2	2012	6	30	下一日是: 2012年7月1日	下一日是: 2012年7月1日
3	2000	2	29	下一日是: 2000年3月1日	下一日是:2000 年3月1日
4	1912	7	31	下一日是: 1912年8月1日	下一日是: 1912年8月1日
5	1909	2	28	下一日是: 1909年3月1日	下一日是:1909 年3月1日
6	2005	10	31	下一日是: 2005年11月1日	下一日是: 2005年11月1日
7	1997	11	31	11月可没有31天	11月可没有31天
8	1999	12	31	下一日是: 2000年1月1日	下一日是:2000 年1月1日
9	1900	2	29	本年2月没有那么多天	本年2月没有那么多天
10	Abcd	6	6	日期不合法,含异常字符。	日期不合法,含异常字符。
11	6.6	6	6	日期不是整数	日期不是整数
12	1787	6	5	年越界	年越界
13	2018	7	2	年越界	年越界
14	2000	0	6	月越界	月越界
15	2000	13	6	月越界	月越界

16	2000	6	0	日越界	日越界
17	2000	6	32	日越界	日越界
18	1887	1	15	下一日是:1887年1月16日	下一日是:1887年1月16日
19	Null	Null	Null	日期数据不全	日期数据不全

## 2.3 决策表分析

NextDate 函数求解给定某个日期的下一个日期的可能操作（动作桩）如下：

变量 day 加 1 操作；

变量 day 复位操作；

变量 month 加 1 操作；

变量 month 复位操作；

变量 year 加 1 操作。

根据上述动作桩发现 NextDate 函数的求解关键是日和月的问题，通常可以在下面等价类（条件桩）的基础上建立决策表：

M1 = {month: month 有 30 天}

M2 = {month: month 有 31 天, 12 月除外}

M3 = {month: month 是 12 月}

M4 = {month: month 是 2 月}

D1 = {day:  $1 \leq \text{day} \leq 27$ }

D2 = {day: day = 28}

D3 = {day: day = 29}

D4 = {day: day = 30}

D5 = {day: day = 31}

Y1 = {year: year 是闰年}

Y2 = {year: year 不是闰年}

规则选项	1	2	3	4	5	6	7	8	9	10	11
条件:											
C1: month 在	M1	M1	M1	M1	M1	M2	M2	M2	M2	M2	M3

C2: day 在	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5	D1
C3: year 在	—	—	—	—	—	—	—	—	—	—	—
动作:											
A1: 不可能					√						
A2: day 加 1	√	√	√			√	√	√	√		√
A3: day 复位				√						√	
A4: month 加 1				√						√	
A5: month 复位											
A6: year 加 1											

规则选项	12	13	14	15	16	17	18	19	20	21	22
条件:											
C1: month 在	M3	M3	M3	M3	M4	M4	M4	M4	M4	M4	M4
C2: day 在	D2	D3	D4	D5	D1	D2	D2	D3	D3	D4	D5
C3: year 在	—	—	—	—	—	Y1	Y2	Y1	Y2	—	—
动作:											
A1: 不可能									√	√	√
A2: day 加 1	√	√	√		√	√					
A3: day 复位				√			√	√			

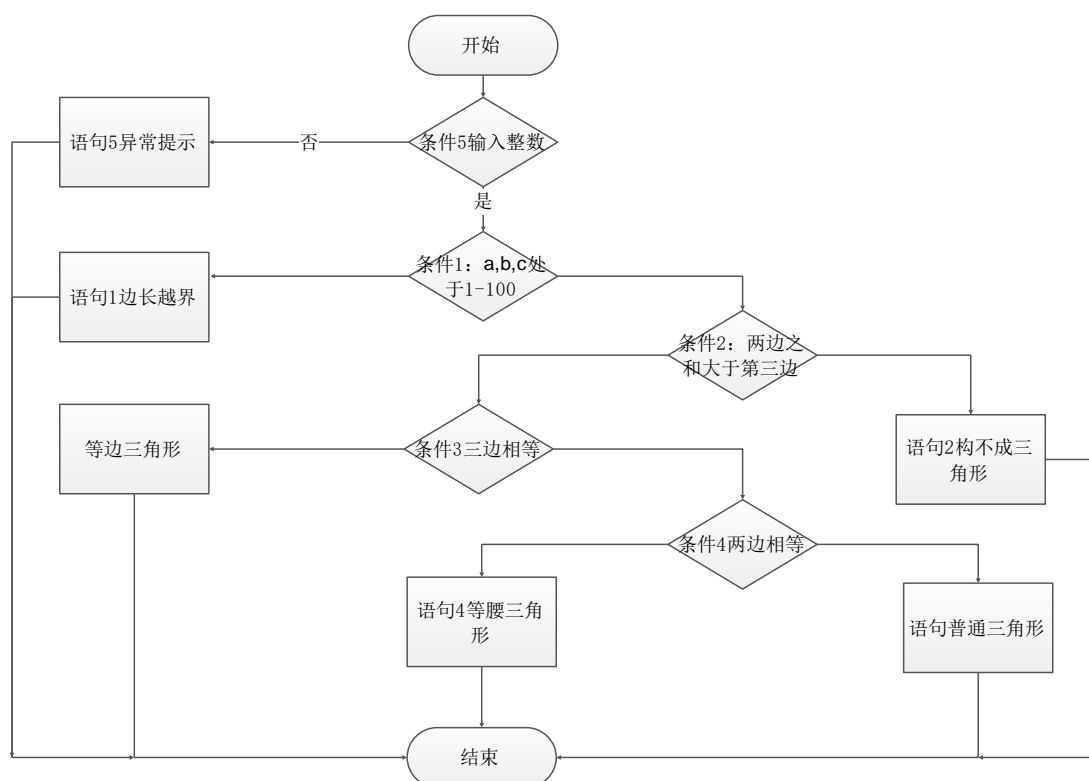
A4: month 加 1							√	√			
A5: month 复位				√							
A6: year 加 1				√							

表 7 次日问题的决策表测试					
测试用例	month	day	year	预期输出	实际输出
1-3	6	15	2007	下一日是:2007 年 6 月 16 日	下一日是:2007 年 6 月 16 日
4	6	30	2007	下一日是: 2007 年 7 月 1 日	下 一 日 是 : 2007 年 7 月 1 日
5	6	31	2007	6 月可没有 31 天	6 月可没有 31 天
6-9	1	15	2007	下一日是:2007 年 1 月 16 日	下一日是:2007 年 1 月 16 日
10	1	31	2007	下一日是: 2007 年 2 月 1 日	下 一 日 是 : 2007 年 2 月 1 日
11-14	12	15	2007	下一日是:2007 年 12 月 16 日	下一日是:2007 年 12 月 16 日
15	12	31	2007	下一日是: 2008 年 1 月 1 日	下一日是: 2008 年 1 月 1 日
16	2	15	2007	下一日是:2007 年 2 月 16 日	下一日是:2007 年 2 月 16 日
17	2	28	2000	下一日是: 2000 年 3 月 1 日	下一日是: 2000 年 3 月 1 日
18	2	28	2007	下一日是: 2007 年 3 月 1 日	下一日是: 2007 年 3 月 1 日
19	2	29	2000	本年 2 月没有那么多天	本年 2 月没有 那么多天

20	2	29	2007	本年 2 月没有那么多天	本年 2 月没有那么多天
21-22	2	30	2007	本年 2 月没有那么多天	本年 2 月没有那么多天

### 三 白盒测试

#### 1 三角形问题的白盒测试用例设计



#### 1.1 语句覆盖

表 8 三角形问题语句覆盖				
用例编号	用例输入	期望输出	实际输出	测试路径描述
1	-2, 3, 4	边长越界	边长越界	条件 5, 条件 1, 语句 1
2	A, 3, 4	构不成三角形, 边长含字符	构不成三角形, 边长含字符	条件 5, 路径 5
3	2.1, 3, 4	边长不是整	边长不是整	条件 5, 语句

		数	数	5
4	2, 3, 4	一般三角形	一般三角形	条件 5, 条件 1, 条件 2, 条件 3, 条件 4
5	2, 2, 3	等腰三角形	等腰三角形	条件 5, 条件 1, 条件 2, 条件 3, 条件 4
6	3, 3, 3	等边三角形	等边三角形	条件 5, 条件 1, 条件 2, 条件 3
7	Null, null, null	数据不完整	数据不完整	条件 1

## 1.2 判定覆盖

表 9 三角形问题的判定覆盖				
用例编号	用例输入	期望输出	实际输出	测试路径描述
1	50, 50, 50	等边三角形	等边三角形	条件 1, 2, 3, 5 真
2	1, 2, 3	构不成三角形	构不成三角形	条件 1, 5 真, 2 假
3	3.14, 5, 6	边长不是整数	边长不是整数	条件 5 假
4	A, 5, 6	构不成三角形, 边长含字符	构不成三角形, 边长含字符	条件 5 假
5	2, 2, 3	等腰三角形	等腰三角形	条件 1, 2, 4, 5 真, 3 假
6	4, 5, 6	一般三角形	一般三角形	条件 1, 2, 5 真, 3, 4 假

## 1.3 条件覆盖

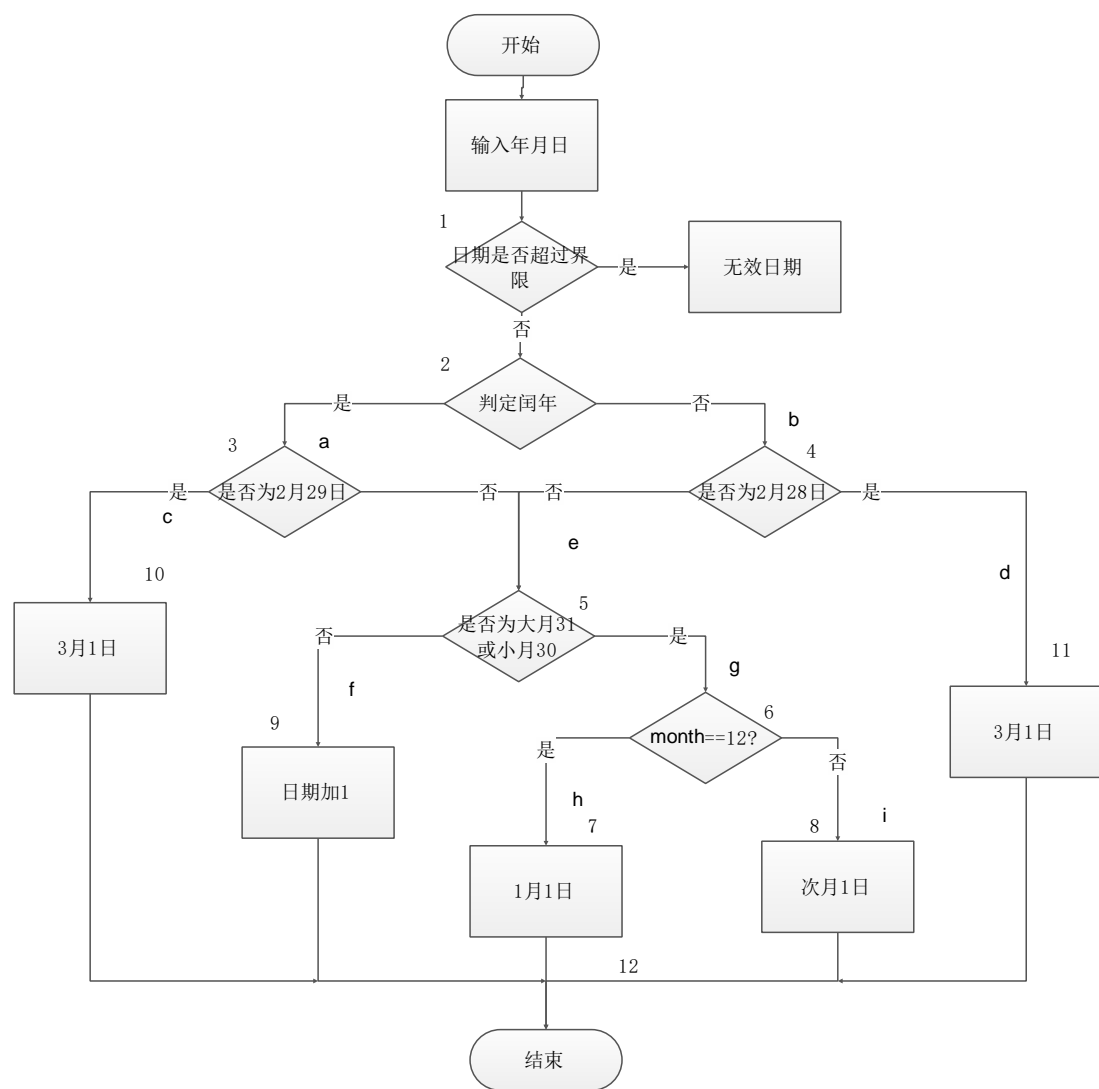
以下测试路径描述中, TX 表示 X 为 true, FX 表示 X 为 false

表 10 三角形问题的条件覆盖				
用例编号	用例输入	预期输出	实际输出	测试路径描述
1	-2, 2, 4	边长越界	边长越界	F1, T2, T3
2	3, -4, 120	边长越界	边长越界	T1, F2, F3
3	3, 5, 9	构不成三角形	构不成三角形	T4, T5, F6
4	8, 1, 3	构不成三角形	构不成三角形	F4, T5, T6



		形	形	
5	2, 8, 4	构不成三角 形	构不成三角 形	T4, F5, T6
6	3, 3, 3	等边三角形	等边三角形	T7, T8
7	2, 3, 4	普通三角形	普通三角形	F7, F8, F9, F10, F11
8	2, 2, 3	等腰三角形	等腰三角形	T9
9	3, 4, 3	等腰三角形	等腰三角形	T11
10	4, 5, 5	等腰三角形	等腰三角形	T10

## 2 次日问题的白盒测试用例设计



### 2.1 语句覆盖

表 11 次日问题的语句覆盖

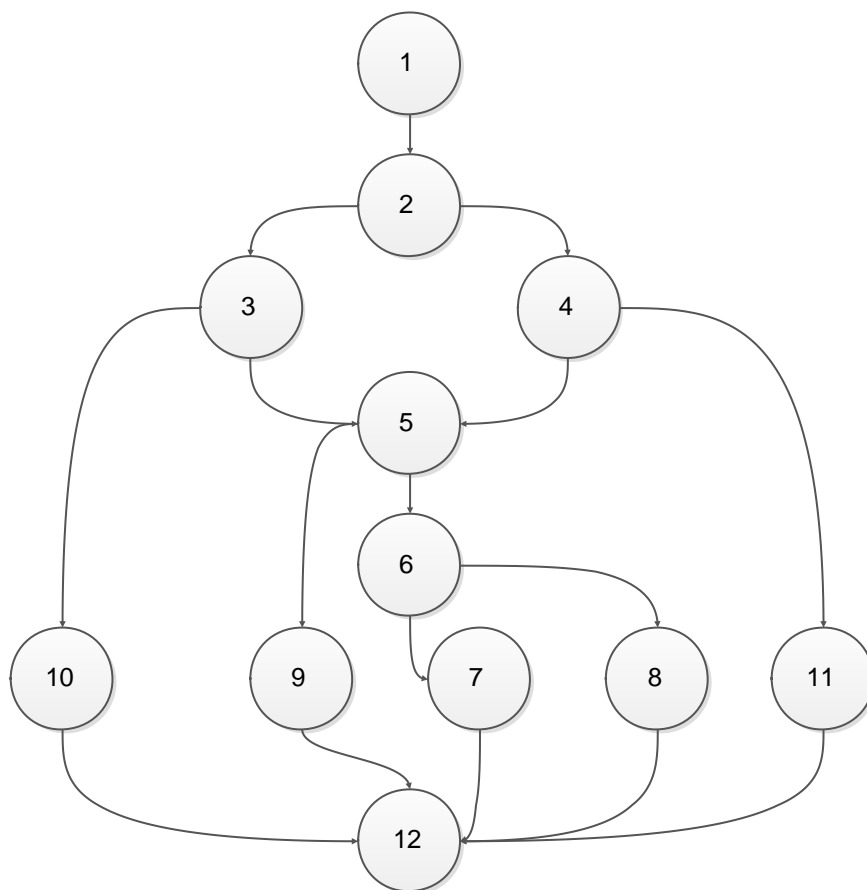
用例编号	用例输入	预期输出	实际输出	测试路径描述
1	2000 2 29	2000 年 3 月 1 日	2000 年 3 月 1 日	T1, T2
2	2001 2 28	2001 年 3 月 1 日	2001 年 3 月 1 日	F1, T3
3	2000 3 28	2000 年 3 月 29 日	2000 年 3 月 29 日	T1, F2, F4
4	2001 12 31	2002 年 1 月 1 日	2002 年 1 月 1 日	F1, F3, T4, T5
5	2001 11 30	2001 年 12 月 1 日	2001 年 12 月 1 日	F1, F3, T4, F5

## 2.2 判定覆盖

表 12 次日问题的判定覆盖测试				
用例编号	用例输入	预期输出	实际输出	测试路径描述
1	2000 2 29	2000 年 3 月 1 日	2000 年 3 月 1 日	ac
2	2001 2 28	2001 年 3 月 1 日	2001 年 3 月 1 日	bd
3	2000 3 28	2000 年 3 月 29 日	2000 年 3 月 29 日	aef
4	2001 12 31	2002 年 1 月 1 日	2002 年 1 月 1 日	begh
5	2001 11 30	2001 年 12 月 1 日	2001 年 12 月 1 日	beji

## 2.3 路径覆盖

### (1) 控制流图



## (2) 环形复杂度

方法一： $V(G)=5$ , (封闭区域)+1=6

方法二： $V(G)=16$ (边数)-12(节点)+2=6

方法三： $V(G)=5$ (判定节点)+1 = 6

## (3) 导出基本路径

根据上面的计算结果，可导出基本路径集，列出程序的独立路径，可得出程序段的基本路径集中有 6 条独立路径，每条独立路径为一个独立的测试用例，路径如下：

路径 1：1-2-3-10-12

路径 2：1-2-3-5-9-12

路径 3：1-2-3-5-6-7-12

路径 4：1-2-3-5-6-8-12

路径 5：1-2-4-5-6-8-12

路径 6：1-2-4-11-12

## (4) 设计测试用例

表 13 次日问题的路径覆盖测试

用例编号	用例输入	预期输出	实际输出	测试路径描述
1	2008 2 29	2008 年 3 月 1 日	2008 年 3 月 1 日	路径 1
2	2008 7 7	2008 年 7 月 7 日	2008 年 7 月 7 日	路径 2
3	2008 12 31	2009 年 1 月 1 日	2009 年 1 月 1 日	路径 3
4	2008 11 30	2008 年 12 月 1 日	2008 年 12 月 1 日	路径 4
5	2007 10 31	2008 年 11 月 1 日	2008 年 11 月 1 日	路径 5
6	2007 2 28	2007 年 3 月 1 日	2007 年 3 月 1 日	路径 6

## 四 Eclipse 环境下用 Junit 进行单元测试

### 1 Eclipse 环境下用 Junit 环境配置

右键单击项目名称, 选择 Build Path→Configure Build Path →Java Build Path → Libraries → Add Library → Junit 选择 Junit 版本, 然后新建一个测试类, 在测试类的方法前面加上@Test 注解, 便可单机方法名进行 JUnit 单元测试。

### 2 JUnit 测试脚本的设计

#### 2.1 三角形的测试脚本

```
package edu.dlu.triangle;
```

```
import org.junit.Test;
```

```
public class TestTriangle {
    Util util = new Util();

    @Test
    public void testTriangle() {
        util.isTriangle("a", "1", "2");
        util.isTriangle("-1", "50", "50");
        util.isTriangle("101", "50", "50");
        util.isTriangle("2.1", "3", "4");
    }
}
```

```

        util.isTriangle("100", "50", "50");
        util.isTriangle("1", "50", "50");
        util.isTriangle("50", "50", "50");
        util.isTriangle("4", "5", "6");
        util.isTriangle("3", "4", "5");
    }

```

@Test

```

public void testGeneralBoundaryOfTriangle() {
    System.out.println("这是三角形问题一般边界值测试的结果:");
    util.isTriangle("60", "60", "1");
    util.isTriangle("60", "60", "2");
    util.isTriangle("60", "60", "2");
    util.isTriangle("60", "60", "60");
    util.isTriangle("50", "50", "99");
    util.isTriangle("50", "50", "100");
    util.isTriangle("60", "1", "60");
    util.isTriangle("60", "2", "60");
    util.isTriangle("50", "99", "50");
    util.isTriangle("50", "100", "50");
    util.isTriangle("1", "60", "60");
    util.isTriangle("2", "60", "60");
    util.isTriangle("99", "50", "50");
    util.isTriangle("100", "50", "50");
}

```

@Test

```

public void testRobustBoundaryOfTriangle() {
    System.out.println("这是三角形问题健壮性边界值测试的结果:");
    util.isTriangle("0", "50", "50");
    util.isTriangle("1", "50", "50");
    util.isTriangle("2", "50", "50");
    util.isTriangle("50", "50", "50");
    util.isTriangle("99", "50", "50");
}

```

```

    util.isTriangle("100", "50", "50");
    util.isTriangle("101", "50", "50");
    util.isTriangle("50", "0", "50");
    util.isTriangle("50", "1", "50");
    util.isTriangle("50", "2", "50");
    util.isTriangle("50", "99", "50");
    util.isTriangle("50", "100", "50");
    util.isTriangle("50", "101", "50");
    util.isTriangle("50", "50", "0");
    util.isTriangle("50", "50", "1");
    util.isTriangle("50", "50", "2");
    util.isTriangle("50", "50", "99");
    util.isTriangle("50", "50", "100");
    util.isTriangle("50", "50", "101");
}

```

@Test

```

public void testEClassOfTriangle() {
    System.out.println("这是三角形问题等价类划分的结果:");
    util.isTriangle("2", "3", "4");
    util.isTriangle("2", "2", "3");
    util.isTriangle("2", "3", "2");
    util.isTriangle("3", "2", "2");
    util.isTriangle("2", "2", "2");
    util.isTriangle("1", "2", "3");
    util.isTriangle("2.1", "3", "4");
    util.isTriangle("2.1", "2.1", "3");
    util.isTriangle("2.1", "2.1", "2.1");
    util.isTriangle("0", "2", "3");
    util.isTriangle("0", "0", "2");
    util.isTriangle("0", "0", "0");
    util.isTriangle("-1", "2", "2");
    util.isTriangle("-1", "-2", "2");
    util.isTriangle("-2", "-2", "-2");
}

```

```

        util.isTriangle("111", "99", "99");
        util.isTriangle("111", "111", "99");
        util.isTriangle("111", "111", "111");
        util.isTriangle("5", "3", "2");
        util.isTriangle("5", "2", "3");
        util.isTriangle("2", "5", "3");
        util.isTriangle("", "", "");
        util.isTriangle("2", "3", "");
        util.isTriangle("3", "4", "5");
    }

```

@Test

```

public void testDesicionTableOfTriangle() {
    System.out.println("这是三角形问题决策表测试的结果:");
    util.isTriangle("4", "1", "2");
    util.isTriangle("1", "4", "2");
    util.isTriangle("1", "2", "4");
    util.isTriangle("5", "5", "5");
    util.isTriangle("?", "?", "?");
    util.isTriangle("2", "2", "3");
    util.isTriangle("2", "3", "2");
    util.isTriangle("3", "2", "2");
    util.isTriangle("3", "4", "5");
}

```

@Test

```

public void testStmtCover() {
    System.out.println("这是三角形问题语句覆盖的结果:");
    util.isTriangle("-2", "3", "4");
    util.isTriangle("A", "3", "4");
    util.isTriangle("3.1", "3", "4");
    util.isTriangle("2", "3", "4");
    util.isTriangle("2", "2", "3");
    util.isTriangle("2", "2", "2");
}

```

```

        util.isTriangle("", "", "");
    }

```

@Test

```

public void testDesicionCover() {
    System.out.println("这是三角形问题判定覆盖的结果:");
    util.isTriangle("50", "50", "50");
    util.isTriangle("1", "2", "3");
    util.isTriangle("3.14", "5", "6");
    util.isTriangle("A", "5", "6");
    util.isTriangle("2", "2", "3");
    util.isTriangle("4", "5", "6");
}

```

@Test

```

public void testConditionCover() {
    System.out.println("这是三角形问题条件覆盖的结果:");
    util.isTriangle("-2", "2", "4");
    util.isTriangle("3", "-4", "120");
    util.isTriangle("3", "5", "9");
    util.isTriangle("8", "1", "3");
    util.isTriangle("2", "8", "4");
    util.isTriangle("3", "3", "3");
    util.isTriangle("2", "3", "4");
    util.isTriangle("2", "2", "3");
    util.isTriangle("3", "4", "3");
    util.isTriangle("4", "5", "5");
}
}

```

## 2.2 次日问题的测试脚本

```

package edu.dlu.nextDay;

```



```
import org.junit.Test;
```

```
public class TestNextDay {
```

```
    Util util = new Util();
```

```
    @Test
```

```
    public void testBoundaryOfNextDay() {
```

```
        System.out.println("以下是次日问题的边界值用例测试结果:");
```

```
        util.nextDay("1811", "6", "15");
```

```
        util.nextDay("1812", "6", "15");
```

```
        util.nextDay("1813", "6", "15");
```

```
        util.nextDay("1912", "6", "15");
```

```
        util.nextDay("2011", "6", "15");
```

```
        util.nextDay("2012", "6", "15");
```

```
        util.nextDay("2013", "6", "15");
```

```
        util.nextDay("1912", "6", "-1");
```

```
        util.nextDay("1912", "6", "1");
```

```
        util.nextDay("1912", "6", "2");
```

```
        util.nextDay("1912", "6", "29");
```

```
        util.nextDay("1912", "6", "30");
```

```
        util.nextDay("1912", "6", "31");
```

```
        util.nextDay("1912", "6", "32");
```

```
        util.nextDay("1912", "-1", "15");
```

```
        util.nextDay("1912", "1", "15");
```

```
        util.nextDay("1912", "2", "15");
```

```
        util.nextDay("1912", "11", "15");
```

```
        util.nextDay("1912", "12", "15");
```

```
        util.nextDay("1912", "13", "15");
```

```
    }
```

```
    @Test
```

```
    public void testEClassOfNextDay() {
```

```
        System.out.println("这是次日问题等价类测试结果:");
```

```
        util.nextDay("1812", "6", "30");
```

```

    util.nextDay("2012", "6", "30");
    util.nextDay("2000", "2", "29");
    util.nextDay("1912", "7", "31");
    util.nextDay("1909", "2", "28");
    util.nextDay("2005", "10", "31");
    util.nextDay("1997", "11", "31");
    util.nextDay("1999", "12", "31");
    util.nextDay("1900", "2", "29");
    util.nextDay("abcd", "6", "6");
    util.nextDay("6.6", "6", "6");
    util.nextDay("1787", "6", "5");
    util.nextDay("2018", "7", "2");
    util.nextDay("2000", "0", "6");
    util.nextDay("2000", "13", "6");
    util.nextDay("2000", "6", "0");
    util.nextDay("2000", "6", "32");
    util.nextDay("1887", "1", "15");
    util.nextDay("", "", "");
}

```

@Test

```

public void testDesicionTableOfNextDay() {
    System.out.println("这是次日问题的决策表测试结果:");
    util.nextDay("2007", "6", "15");
    util.nextDay("2007", "6", "30");
    util.nextDay("2007", "6", "31");
    util.nextDay("2007", "1", "15");
    util.nextDay("2007", "1", "31");
    util.nextDay("2007", "12", "15");
    util.nextDay("2007", "12", "31");
    util.nextDay("2007", "2", "15");
    util.nextDay("2007", "2", "28");
    util.nextDay("2007", "2", "29");
    util.nextDay("2007", "2", "30");
}

```

```

        util.nextDay("2000", "2", "28");
        util.nextDay("2000", "2", "29");
        util.nextDay("2000", "2", "30");
    }

```

@Test

```

    public void testStmtCover() {
        System.out.println("这是次日问题语句覆盖的结果:");
        util.nextDay("2000", "2", "29");
        util.nextDay("2001", "2", "28");
        util.nextDay("2000", "3", "28");
        util.nextDay("2001", "12", "31");
        util.nextDay("2001", "11", "30");
    }

```

@Test

```

    public void testDesicionCover() {
        System.out.println("这是次日问题判定覆盖的结果:");
        util.nextDay("2000", "2", "29");
        util.nextDay("2001", "2", "28");
        util.nextDay("2000", "3", "28");
        util.nextDay("2001", "12", "31");
        util.nextDay("2001", "11", "30");
    }

```

@Test

```

    public void testPathCover() {
        System.out.println("这是次日问题路径覆盖的结果:");
        util.nextDay("2008", "2", "29");
        util.nextDay("2008", "7", "7");
        util.nextDay("2008", "12", "31");
        util.nextDay("2008", "11", "30");
        util.nextDay("2007", "10", "31");
        util.nextDay("2007", "2", "28");
    }

```

```

    }
}

```

### 3 测试结果

#### 3.1 三角形问题

表 1 三角形问题的一般边界值测试结果

<terminated> TestTriangle.testGeneralBoundaryOfTriangle [JUnit] C:\Program Files\Java\jdk1.8.0_112\bin\javaw.exe (8 Jul 2018, 12:13:31)	
这是三角形问题一般边界值测试的结果:	
等腰三角形	
等腰三角形	
等腰三角形	
等边三角形	
等腰三角形	
构不成三角形	
等腰三角形	
等腰三角形	
等腰三角形	
构不成三角形	
等腰三角形	
等腰三角形	
等腰三角形	
构不成三角形	

表 1 三角形问题的一般边界值测试覆盖率







TestTriangle.testGeneralBoundaryOfTriangle (08-Jul-2018 12:13:33)				
Element	Coverage	Covered Instructio	Missed Instructions	Total Instructions
▼ Triangle	 26.4 %	211	589	800
▼ src	 26.4 %	211	589	800
▼ edu.dlu.triangle	 26.4 %	211	589	800
> TestTriangle.java	 15.4 %	96	529	625
▼ Util.java	 65.7 %	115	60	175
> Util	 65.7 %	115	60	175

表 2 三角形问题的健壮性边界值测试结果

<terminated> TestTriangle.testRobustBoundaryOfTriangle [JUnit] C:\Program Files\Java\jdk1.8.0\_112\bin\javaw.exe (8 Jul 2018, 12:28:58)

这是三角形问题健壮性边界值测试的结果:

构不成三角形, 边长越界!  
 等腰三角形  
 等腰三角形  
 等边三角形  
 等腰三角形  
 构不成三角形  
 构不成三角形, 边长越界!  
 构不成三角形, 边长越界!  
 等腰三角形  
 等腰三角形  
 等腰三角形  
 构不成三角形  
 构不成三角形, 边长越界!  
 构不成三角形, 边长越界!  
 等腰三角形  
 等腰三角形  
 等腰三角形  
 构不成三角形  
 构不成三角形, 边长越界!

表 2 三角形问题的健壮性边界值测试覆盖率

TestTriangle.testRobustBoundaryOfTriangle (08-Jul-2018 12:28:59)				
Element	Coverage	Covered Instructio	Missed Instructions	Total Instructions
Triangle	30.6 %	245	555	800
src	30.6 %	245	555	800
edu.dlu.triangle	30.6 %	245	555	800
TestTriangle.java	20.2 %	126	499	625
TestTriangle	20.2 %	126	499	625
Util.java	68.0 %	119	56	175
Util	68.0 %	119	56	175

表 3 三角形问题等价类划分测试结果

```
<terminated> TestTriangle.testClassOfTriangle [JUnit] C:\Program Files\Java\jdk1.8.0_112\bin\javaw.exe (8 Jul 2018, 12:33:40)
这是三角形问题等价类划分的结果:
一般三角形
等腰三角形
等腰三角形
等腰三角形
等边三角形
构不成三角形
边长不是整数
边长不是整数
边长不是整数
构不成三角形, 边长越界!
构不成三角形, 边长越界!
构不成三角形, 边长越界!
构不成三角形, 边长越界!
构不成三角形, 边长越界!
构不成三角形, 边长越界!
构不成三角形, 边长越界!
构不成三角形, 边长越界!
构不成三角形, 边长越界!
构不成三角形
构不成三角形
构不成三角形
数据不完整
数据不完整
直角三角形
```

表 3 三角形问题等价类划分测试覆盖率

Element	Coverage	Covered Instructio	Missed Instructions	Total Instructions
Triangle	40.9 %	327	473	800
src	40.9 %	327	473	800
edu.dlu.triangle	40.9 %	327	473	800
TestTriangle.java	25.0 %	156	469	625
TestTriangle	25.0 %	156	469	625
Util.java	97.7 %	171	4	175
Util	97.7 %	171	4	175

表 4 三角形问题的决策表测试结果

```
<terminated> TestTriangle.testDesicionTableOfTriangle [JUnit] C:\Program Files\Java\jdk1.8.0_112\bin\javaw.exe (8 Jul 2018, 12:36:32)
这是三角形问题决策表测试的结果:
构不成三角形
构不成三角形
构不成三角形
等边三角形
构不成三角形, 边长含字符!
等腰三角形
等腰三角形
等腰三角形
直角三角形
```

表 4 三角形问题的决策表测试覆盖率

TestTriangle.testDesicionTableOfTriangle (08-Jul-2018 12:36:33)				
Element	Coverage	Covered Instructio	Missed Instructions	Total Instructions
Triangle	25.0 %	200	600	800
src	25.0 %	200	600	800
edu.dlu.triangle	25.0 %	200	600	800
TestTriangle.java	10.6 %	66	559	625
TestTriangle	10.6 %	66	559	625
Util.java	76.6 %	134	41	175
Util	76.6 %	134	41	175

表 5 次日问题的边界值测试结果

<terminated> TestNextDay.testBoundaryOfNextDay [JUnit] C:\Program Files\Java\jdk1.8.0\_112\bin\javaw.exe (8 Jul 2018, 12:37:54)

以下是次日问题的边界值用例测试结果:

年越界

下一日是:1812年6月16日

下一日是:1813年6月16日

下一日是:1912年6月16日

下一日是:2011年6月16日

下一日是:2012年6月16日

年越界

日越界

下一日是:1912年6月2日

下一日是:1912年6月3日

下一日是:1912年6月30日

下一日是: 1912年7月1日

6月可没有31天

日越界

月越界

下一日是:1912年1月16日

下一日是:1912年2月16日

下一日是:1912年11月16日

下一日是:1912年12月16日

月越界

表 5 次日问题的边界值测试覆盖率

TestNextDay.testBoundaryOfNextDay (08-Jul-2018 12:37:55)				
Element	Coverage	Covered Instructio	Missed Instructions	Total Instructions
NextDay	48.2 %	363	390	753
src	48.2 %	363	390	753
edu.dlu.nextDay	48.2 %	363	390	753
TestNextDay.java	29.6 %	132	314	446
Util.java	75.2 %	231	76	307
Util	75.2 %	231	76	307
nextDay(String, String, String)	72.9 %	194	72	266
containsDot(String)	75.0 %	6	2	8
isEmpty(String)	80.0 %	8	2	10
isLeapYear(int)	100.0 %	16	0	16
isNum(String)	100.0 %	4	0	4

表 6 次日问题的等价类测试结果

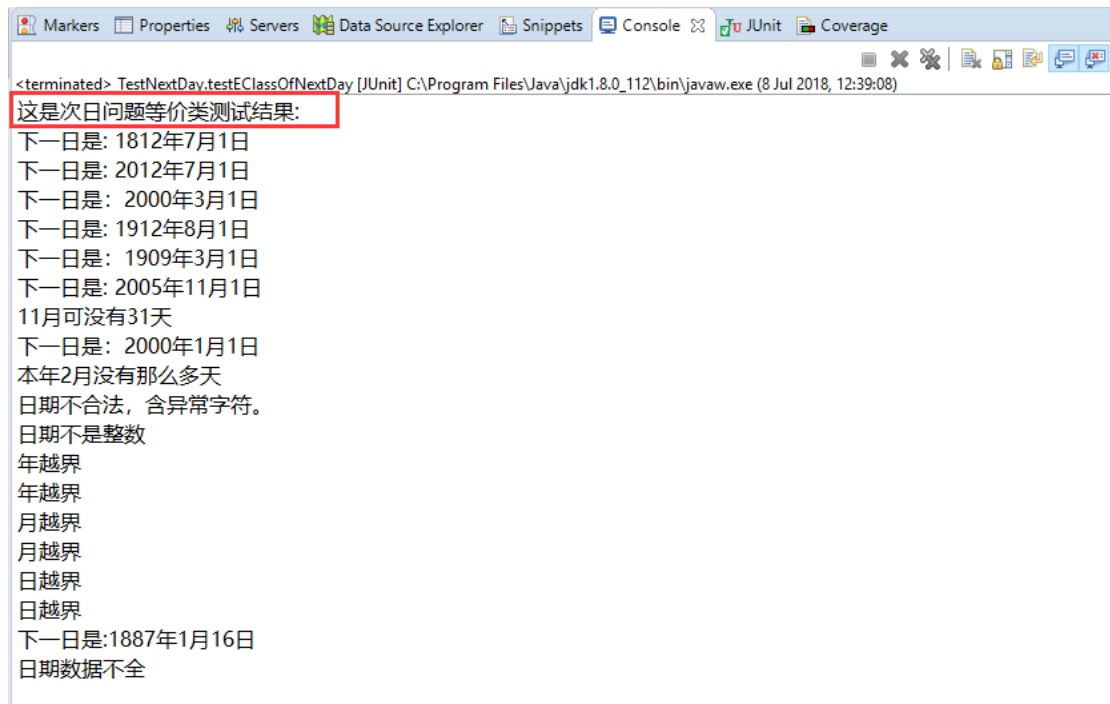


表 6 次日问题的等价类测试覆盖率

TestNextDay.testEClassOfNextDay (08-Jul-2018 12:39:09)				
Element	Coverage	Covered Instruction	Missed Instructions	Total Instructions
NextDay	57.5 %	433	320	753
src	57.5 %	433	320	753
edu.dlu.nextDay	57.5 %	433	320	753
TestNextDay.java	28.3 %	126	320	446
TestNextDay	28.3 %	126	320	446
testBoundaryOfNextDay()	0.0 %	0	124	124
testDesicionTableOfNextDay()	0.0 %	0	88	88
testPathCover()	0.0 %	0	40	40
testDesicionCover()	0.0 %	0	34	34
testStmntCover()	0.0 %	0	34	34
testEClassOfNextDay()	100.0 %	118	0	118
Util.java	100.0 %	307	0	307
Util	100.0 %	307	0	307
containsDot(String)	100.0 %	8	0	8
isEmpty(String)	100.0 %	10	0	10
isLeapYear(int)	100.0 %	16	0	16
isNum(String)	100.0 %	4	0	4
nextDay(String, String, String)	100.0 %	266	0	266

表 7 次日问题的决策表测试结果



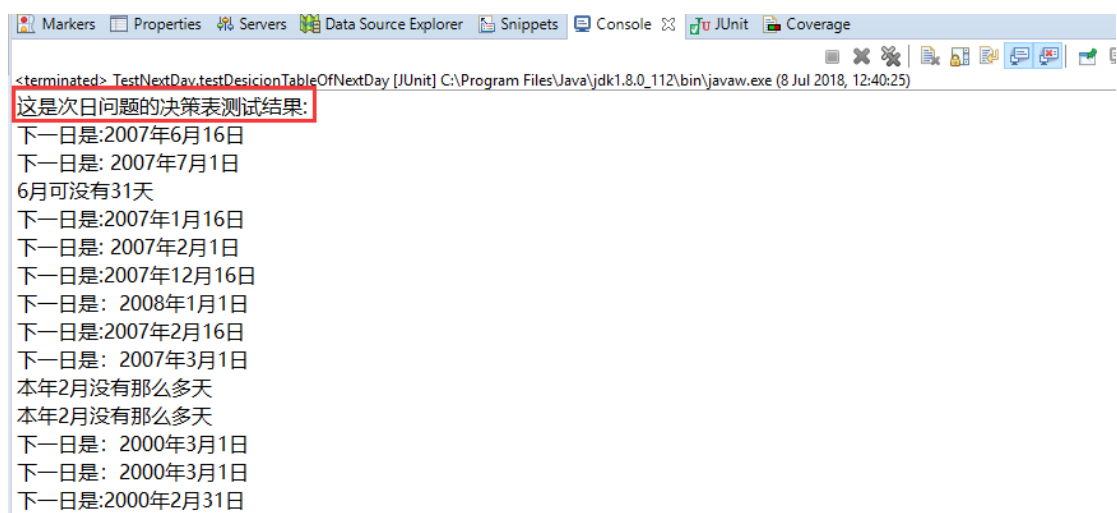


表 7 次日问题的决策表测试覆盖率

Element	Coverage	Covered Instructio	Missed Instructions	Total Instructions
NextDay	49.9 %	376	377	753
src	49.9 %	376	377	753
edu.dlu.nextDay	49.9 %	376	377	753
TestNextDay.java	21.5 %	96	350	446
TestNextDay	21.5 %	96	350	446
testBoundaryOfNextDay()	0.0 %	0	124	124
testEClassOfNextDay()	0.0 %	0	118	118
testPathCover()	0.0 %	0	40	40
testDesicionCover()	0.0 %	0	34	34
testStmntCover()	0.0 %	0	34	34
testDesicionTableOfNextDay()	100.0 %	88	0	88
Util.java	91.2 %	280	27	307
Util	91.2 %	280	27	307
nextDay(String, String, String)	91.4 %	243	23	266
containsDot(String)	75.0 %	6	2	8
isEmpty(String)	80.0 %	8	2	10
isLeapYear(int)	100.0 %	16	0	16
isNum(String)	100.0 %	4	0	4

表 8 三角形问题语句覆盖结果

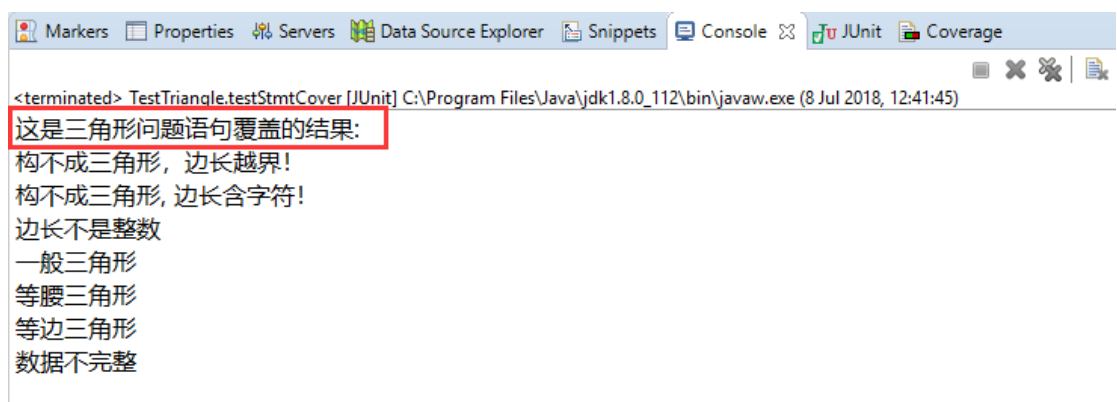


表 8 三角形问题语句覆盖覆盖率

TestTriangle.testStmtCover (08-Jul-2018 12:41:46)				
Element	Coverage	Covered Instructio	Missed Instructions	Total Instructions
Triangle	27.6 %	221	579	800
src	27.6 %	221	579	800
edu.dlu.triangle	27.6 %	221	579	800
TestTriangle.java	8.6 %	54	571	625
TestTriangle	8.6 %	54	571	625
testEClassOfTriangle()	0.0 %	0	148	148
testRobustBoundaryOfTriangle()	0.0 %	0	118	118
testGeneralBoundaryOfTriangle()	0.0 %	0	88	88
testConditionCover()	0.0 %	0	64	64
testDesicionTableOfTriangle()	0.0 %	0	58	58
testTriangle()	0.0 %	0	55	55
testDesicionCover()	0.0 %	0	40	40
testStmntCover()	100.0 %	46	0	46
Util.java	95.4 %	167	8	175
Util	95.4 %	167	8	175
isTriangle(String, String, String)	94.7 %	142	8	150
containsDot(String)	100.0 %	8	0	8
isEmpty(String)	100.0 %	10	0	10
isNum(String)	100.0 %	4	0	4

表 9 三角形问题的判定覆盖结果

<terminated> TestTriangle.testDesicionCover [JUnit] C:\Program Files\Java\jdk1.8.0\_112\bin\javaw.exe (8 Jul 2018, 12:43:14)

这是三角形问题判定覆盖的结果:

等边三角形

构不成三角形

边长不是整数

构不成三角形, 边长含字符!

等腰三角形

一般三角形

表 9 三角形问题的判定覆盖覆盖率

TestTriangle.testDesicionCover (08-Jul-2018 12:43:15)				
Element	Coverage	Covered Instructio	Missed Instructions	Total Instructions
Triangle	26.2 %	210	590	800
src	26.2 %	210	590	800
edu.dlu.triangle	26.2 %	210	590	800
TestTriangle.java	7.7 %	48	577	625
TestTriangle	7.7 %	48	577	625
testEClassOfTriangle()	0.0 %	0	148	148
testRobustBoundaryOfTriangle()	0.0 %	0	118	118
testGeneralBoundaryOfTriangle()	0.0 %	0	88	88
testConditionCover()	0.0 %	0	64	64
testDesicionTableOfTriangle()	0.0 %	0	58	58
testTriangle()	0.0 %	0	55	55
testStmntCover()	0.0 %	0	46	46
testDesicionCover()	100.0 %	40	0	40
Util.java	92.6 %	162	13	175
Util	92.6 %	162	13	175
isTriangle(String, String, String)	92.7 %	139	11	150
isEmpty(String)	80.0 %	8	2	10
containsDot(String)	100.0 %	8	0	8
isNum(String)	100.0 %	4	0	4

表 10 三角形问题的条件覆盖结果

```
<terminated> TestTriangle.testConditionCover [JUnit] C:\Program Files\Java\jdk1.8.0_112\bin\javaw.exe (8 Jul 2018, 12:44:18)
这是三角形问题条件覆盖的结果:
构不成三角形, 边长越界!
构不成三角形, 边长越界!
构不成三角形
构不成三角形
构不成三角形
等边三角形
一般三角形
等腰三角形
等腰三角形
等腰三角形
```

表 10 三角形问题的条件覆盖覆盖率

TestTriangle.testConditionCover (08-Jul-2018 12:44:19)				
Element	Coverage	Covered Instruction	Missed Instructions	Total Instructions
Triangle	28.5 %	228	572	800
src	28.5 %	228	572	800
edu.dlu.triangle	28.5 %	228	572	800
TestTriangle.java	11.5 %	72	553	625
TestTriangle	11.5 %	72	553	625
testEClassOfTriangle()	0.0 %	0	148	148
testRobustBoundaryOfTriangle()	0.0 %	0	118	118
testGeneralBoundaryOfTriangle()	0.0 %	0	88	88
testDesicionTableOfTriangle()	0.0 %	0	58	58
testTriangle()	0.0 %	0	55	55
testStmtCover()	0.0 %	0	46	46
testDesicionCover()	0.0 %	0	40	40
testConditionCover()	100.0 %	64	0	64
Util.java	89.1 %	156	19	175
Util	89.1 %	156	19	175
isTriangle(String, String, String)	90.0 %	135	15	150
containsDot(String)	75.0 %	6	2	8
isEmpty(String)	80.0 %	8	2	10
isNum(String)	100.0 %	4	0	4

表 11 次日问题的语句覆盖结果

```
<terminated> TestNextDay.testStmtCover [JUnit] C:\Program Files\Java\jdk1.8.0_112\bin\javaw.exe (8 Jul 2018, 12:45:50)
这是次日问题语句覆盖的结果:
下一日是: 2000年3月1日
下一日是: 2001年3月1日
下一日是:2000年3月29日
下一日是: 2002年1月1日
下一日是: 2001年12月1日
```

表 11 次日问题的语句覆盖覆盖率

TestNextDay.testStmntCover (08-Jul-2018 12:45:51)				
Element	Coverage	Covered Instructio	Missed Instructions	Total Instructions
NextDay	37.3 %	281	472	753
src	37.3 %	281	472	753
edu.dlu.nextDay	37.3 %	281	472	753
TestNextDay.java	9.4 %	42	404	446
Util.java	77.9 %	239	68	307
Util	77.9 %	239	68	307
nextDay(String, String, String)	75.9 %	202	64	266
containsDot(String)	75.0 %	6	2	8
isEmpty(String)	80.0 %	8	2	10
isLeapYear(int)	100.0 %	16	0	16
isNum(String)	100.0 %	4	0	4

表 12 次日问题的判定覆盖测试结果

<terminated> TestNextDay.testDesicionCover [JUnit] C:\Program Files\Java\jdk1.8.0_112\bin\javaw.exe (8 Jul 2018, 12:47:17)	
这是次日问题判定覆盖的结果:	
下一日是: 2000年3月1日	
下一日是: 2001年3月1日	
下一日是:2000年3月29日	
下一日是: 2002年1月1日	
下一日是: 2001年12月1日	

表 12 次日问题的判定覆盖测试覆盖率

TestNextDay.testDesicionCover (08-Jul-2018 12:47:18)				
Element	Coverage	Covered Instructio	Missed Instructions	Total Instructions
NextDay	37.3 %	281	472	753
src	37.3 %	281	472	753
edu.dlu.nextDay	37.3 %	281	472	753
TestNextDay.java	9.4 %	42	404	446
TestNextDay	9.4 %	42	404	446
testBoundaryOfNextDay()	0.0 %	0	124	124
testEClassOfNextDay()	0.0 %	0	118	118
testDesicionTableOfNextDay()	0.0 %	0	88	88
testPathCover()	0.0 %	0	40	40
testStmntCover()	0.0 %	0	34	34
testDesicionCover()	100.0 %	34	0	34
Util.java	77.9 %	239	68	307
Util	77.9 %	239	68	307
nextDay(String, String, String)	75.9 %	202	64	266
containsDot(String)	75.0 %	6	2	8
isEmpty(String)	80.0 %	8	2	10
isLeapYear(int)	100.0 %	16	0	16
isNum(String)	100.0 %	4	0	4

表 13 次日问题的路径覆盖测试结果

<terminated> TestNextDay.testPathCover [JUnit] C:\Program Files\Java\jdk1.8.0_112\bin\javaw.exe (8 Jul 2018, 12:48:11)	
这是次日问题路径覆盖的结果:	
下一日是: 2008年3月1日	
下一日是:2008年7月8日	
下一日是: 2009年1月1日	
下一日是: 2008年12月1日	
下一日是: 2007年11月1日	
下一日是: 2007年3月1日	

表 13 次日问题的路径覆盖测试覆盖率

TestNextDay.testPathCover (08-Jul-2018 12:48:12)

Element	Coverage	Covered Instructio	Missed Instructions	Total Instructions
NextDay	40.4 %	304	449	753
src	40.4 %	304	449	753
edu.dlu.nextDay	40.4 %	304	449	753
TestNextDay.java	10.8 %	48	398	446
TestNextDay	10.8 %	48	398	446
testBoundaryOfNextDay()	0.0 %	0	124	124
testEClassOfNextDay()	0.0 %	0	118	118
testDesicionTableOfNextDay()	0.0 %	0	88	88
testDesicionCover()	0.0 %	0	34	34
testStmtCover()	0.0 %	0	34	34
testPathCover()	100.0 %	40	0	40
Util.java	83.4 %	256	51	307
Util	83.4 %	256	51	307
nextDay(String, String, String)	82.3 %	219	47	266
containsDot(String)	75.0 %	6	2	8
isEmpty(String)	80.0 %	8	2	10
isLeapYear(int)	100.0 %	16	0	16
isNum(String)	100.0 %	4	0	4

## 3.2 次日问题

# 五 测试结果分析

## 1 缺陷分析

### 1.1 缺陷记录

次日问题

缺陷编号	测试用例	预期输出	实际输出
1	1900 2 28	1900 年 3 月 1 日	1900 年 2 月 29 日
2	1997 11 30	1997 年 12 月 1 日	1997 年 11 月 31 日
3	1997 10 31	1997 年 11 月 1 日	1997 年 10 月 32 日
4	3.1415926 6 6	日期不是整数	抛出异常

### 1.2 发现缺陷的用例及设计方法

缺陷编号	测试用例	设计方法	修复方法
1	1900 2 28	边界值分析	增加闰年判断工具 isLeapYear(int year) 并调用
2	1997 11 30	边界值分析	增加判断条件
3	1997 10 31	边界值分析	增加判断条件
4	3.1415926 6 6	等价类划分	调用已写好的工具 containsDot(String string) 进行判断

## 2 测试总结

通过软件三角形问题与次日问题的分析与测试，深刻地理解到以前思考问题是多么的片面，是多么的不周全，实际的软件测试虽然麻烦，但是却非常地能够锻炼人的思维。在测试的过程中仔细地考虑了输入数据的可能情况以及如何提高程序的健壮性，就像是软件测试中的“聪明的猴子”到“愚蠢的猴子”一样。输入的数据完全是不可预知的，各式各样的。

在三角形问题的设计中，从一开始只知道输入正整数，然后考虑到输入数据可能为多边或少边，可能为负数，可能为小数，可能为字符，可能为字符串，可能包含有特殊字符，可能包含“点”操作符，可能是普通三角形，可能是等边三角形，可能是等腰三角形，还有可能是直角三角形，还可能是等腰直角三角形，在进行直角三角形的判定时做的还不是很完善，只能够判断边长为整数的直角三角形，后期仍需完善判别边长为根号的三角形，具体的实现办法可以使用绝对值误差判断，例如两边的平方之和与第三边的平方之差小于  $10e-6$  可以认为近似相等。

考虑到这些问题后然后再一一加以解决，同时提高了思维能力和编程能力和处理细节的能力。

次日问题的设计与测试中与三角形问题类似，但是需要考虑的情况则是更多。

同时也系统的学习并且实践了软件测试的具体方法，比如黑盒测试中的边界值分析，等价类划分，决策表以及白盒测试的语句覆盖，分支覆盖，判定覆盖，路径覆盖等等。还学习了在 Eclipse 环境中利用 Junit 进行单元测试。