重 庆 交 通 大 学

学生实验报告

实验课程名称：《软件测试》

开 课 实验室：软件实验室（南岸）

学　　院：信息学院

专　　业：计算机科学与技术

班　　级：2014 级 一 班

学　　号：631406010109

学 生 姓 名：郭文浩

指 导 教 师：何　伟

开 课 时 间：2016 至 2017 学年第 2 学期

总 成 绩	
教师签名	

实验项目名称		黑盒测试			
姓名	郭文浩	学号	**631406010109**	实验日期	**2017 年 6 月 7 日**

教师评阅：

1:实验目的明确□A□B□C□D;　2:操作步骤正确□A□B□C□D;

3:实验报告规范□A□B□C□D;　4:实验结果符合要求□A□B□C□D

5:实验过程原始记录（数据、图表、计算等）符合要求□A□B□C□D;

6:实验分析总结全面□A□B□C□D;

签名：　　年　月　日

实验成绩：

一、实验目的

二、实验主要内容及原理

三、测试代码、测试方法及测试用例（可以以表格形式说明）

四、发现程序缺陷及修改方案

五、测试结果及分析(包括心得体会，本部分为重点，不能抄袭，不能复制)

一、实验目的

1、能熟练应用功能性测试技术进行测试用例设计。

2、对测试用例进行优化设计。

二、实验内容及原理

1、实验内容

题目一：三角形问题

根据下面给出的规格说明，利用等价类划分的方法，给出足够的测试用例。

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

题目二：日期问题

用决策表测试法测试以下程序:该程序有三个输入变量 month、day、year（month 、 day 和 year 均为整数值，并且满足：1≤month≤12 和 1≤day≤31),分别作为输入日期的月份、日、年份，通过程序可以输出该输入日期在日历上隔一天的日期。例如，输入为 2004 年 11 月 29 日,则该程序的输出为 2004 年 12 月 1 日。

(1) 分析各种输入情况，列出为输入变量 month 、 day 、 year 划分的有效等价类。

(2) 分析程序的规格说明，并结合以上等价类划分的情况，给出问题规定的可能采取的操作（即列出所有的动作桩）。

(3) 根据 (1) 和 (2) ，画出简化后的决策表。

2、实验原理

等价类测试

等价类测试方法是把所有可能的输入数据，即程序的输入域划分成若干部分，然后从每一部分中选取少数有代表性的数据作为测试用例。使用等价类划分方法设计测试用例要经历划分等价类（列出等价类表）和选取测试用例两步。

边界值测试

边界值测试包括：边界值分析、健壮性测试和最坏情况测试。边界值分析是考虑边界条件而选取测试用例的一种功能测试方法。边界值分析关注输入空间的边界，以标识测试用例，因为错误更可能出现在输入变量的极值附近。健壮性是

指在异常情况下，软件还能正常运行的能力。健壮性考虑的主要部分是预期输出，而不是输入。最坏情况测试将意味着更大工作量，n 变量函数的最坏情况测试会产生 5 的 n 次方个测试用例，而边界值分析只产生 4n+1 个测试用例。

三、测试代码、测试方法及测试用例

1、测试代码（两个代码均为自己的代码）

题目一：三角形问题

```
package com.gwh.hhcs;
import java.util.Scanner;
/**
 * 三角形类 用于练习黑盒测试
 * @author guowenhao
 * @version 1.0
 */
public class Triangle {
    public static void main(String[] args) {
        // 接收器初始化
        Scanner sc = new Scanner(System.in);
        // 三角形三边初始化
        double edge1 = 0;
        double edge2 = 0;
        double edge3 = 0;
        // 接收三角形三边
        System.out.println("即将录入三角形的边长......");
        edge1 = receiveEdge(sc, "一");
        edge2 = receiveEdge(sc, "二");
        edge3 = receiveEdge(sc, "三");
        // 判断是否能够成三角形
        if (IsTriangle(edge1, edge2, edge3)) {
            // 判断能否构成等边三角形
            if (IsEquilateralTriangle(edge1, edge2, edge3)) {
                System.out.println("等边三角形");
            }
            // 判断能否构成等腰三角形
            else if (IsIsoscelesTriangle(edge1, edge2, edge3)) {
                System.out.println("等腰三角形");
            } else {
                System.out.println("三边不等的三角形");
            }
        } else {
            System.out.println("无法构成三角形");
        }
```

```
        sc.close();
    }
    /**
     * 判断是否为等边三角形
     * @param edge1
     *           第一条边的边长
     * @param edge2
     *           第二条边的边长
     * @param edge3
     *           第三条边的边长
     * @return 是等边三角形返回true 不是等边三角形返回false
     */
    private static boolean IsEquilateralTriangle(double edge1, double edge2, double
edge3) {
        if (edge1 == edge2 && edge1 == edge3)
            return true;
        else
            return false;
    }
    /**
     * 判断是否为等腰三角形
     * @param edge1
     *           第一条边的边长
     * @param edge2
     *           第二条边的边长
     * @param edge3
     *           第三条边的边长
     * @return 是等腰三角形返回true 不是等腰三角形返回false
     */
    private static boolean IsIsoscelesTriangle(double edge1, double edge2, double
edge3) {
        if (edge1 == edge2 || edge1 == edge3 || edge2 == edge3)
            return true;
        else
            return false;
    }
    /**
     * 判断输入的边长是否合法，合法返回输入，不合法则重新输入
     * @param sc
     *           接收器
     * @param index
     *           第几条边
     * @return 合法的边长
     */
```

```
    private static double receiveEdge(Scanner sc, String index) {
        double edge = 0;
        while (true) {
            try {
                System.out.println("请输入第" + index + "条边：");
                String input = sc.nextLine();
                edge = Double.parseDouble(input);
                if (edge > 0)
                    break;
                else {
                    System.out.println("第" + index + "条边输入有误，边长应为正数，请
重新输入！");
                    continue;
                }
            } catch (NumberFormatException e) {
                System.out.println("第" + index + "条边输入有误，边长应为数字，请重新
输入！");
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return edge;
    }
    /**
     * 通过输入的三边确定是否能构成三角形
     * @param edge1
     *            第一条边的边长
     * @param edge2
     *            第二条边的边长
     * @param edge3
     *            第三条边的边长
     * @return 能构成三角形返回true 不能构成三角形返回false
     */
    private static boolean IsTriangle(double edge1, double edge2, double edge3) {
        if (edge1 < edge2 + edge3 && edge2 < edge1 + edge3 && edge3 < edge1 + edge2
&& edge1 > Math.abs(edge2 - edge3)
                && edge2 > Math.abs(edge1 - edge3) && edge3 > Math.abs(edge1 -
edge2))
            return true;
        else
            return false;
    }
}
```

题目二：日期问题

```
package com.gwh.hhcs;
import java.util.Scanner;
/**
 * 输入日期隔天日期
 *
 * @author guowenhao
 * @version 2.0
 */
public class NextAndNextDay {
    // 初始化每个月有几天，不可修改
    final private static int[] monthDay = new int[] { 29, 31, 28, 31, 30, 31, 30,
31, 31, 30, 31, 30, 31 };
    public static void main(String[] args) {
        // 得到输入的合法日期
        Scanner sc = new Scanner(System.in);
        int[] date = receiveDate(sc);
        sc.close();
        // 隔天
        date[2] = date[2] + 2;
        if (isCorrectDate(date)) {
            System.out.println(date[0] + "年" + date[1] + "月" + date[2] + "日");
        } else {
            // 当前天数=当前天数-这个月的最大天数
            date[2] = date[2] - monthDay[date[1]];
            date[1]++;
            if (date[1] < 13) {
                System.out.println(date[0] + "年" + date[1] + "月" + date[2] + "日
");
            } else {
                // 月份变为1月
                date[1] = 1;
                // 年份+1
                date[0]++;
                System.out.println(date[0] + "年" + date[1] + "月" + date[2] + "日
");
            }
        }
    }

    /**
     * 判断这一年是否为闰年
     *
     * @param year
```

```
     *           要判断的年份
     * @return 是闰年返回true 不是闰年返回falase
     */
    private static boolean isRunnian(int year) {
        if (year % 400 == 0)
            return true;
        else if (year % 4 == 0 && year % 100 != 0)
            return true;
        else
            return false;
    }
    /**
     * 判断输入的日期是否合法，合法返回输入，不合法则重新输入
     *
     * @param sc
     *           接收器
     * @return 合法的日期数组
     */

    private static int[] receiveDate(Scanner sc) {
        int[] date = new int[3];
        while (true) {
            try {
                System.out.println("请输入日期（例如1980-3-15）：");
                String input = sc.nextLine();
                String[] strDate = new String[3];
                // 分割输入的日期
                strDate = input.split("-");
                date[0] = Integer.parseInt(strDate[0]);
                date[1] = Integer.parseInt(strDate[1]);
                date[2] = Integer.parseInt(strDate[2]);
                // 日期应大于0
                if (date[0] < 1 || date[1] < 1 || date[2] < 1) {
                    System.out.println("日期应为正数，请重新输入！");
                    continue;
                }
                // 月份不能超过12
                else if (date[1] > 12) {
                    System.out.println("月份不能大于12，请重新输入！");
                    continue;
                } else if (isCorrectDate(date))
                    break;
                else {
                    if (date[1] == 2 && isRunnian(date[0]))
```

```
                        System.out.println("日期输入有误，" + date[0] + "年的" +
date[1] + "月最多只有" + monthDay[0] + "号，没有"
                                + date[2] + "号，请重新输入！");
                    else
                        System.out.println("日期输入有误，" + date[0] + "年的" +
date[1] + "月最多只有" + monthDay[date[1]] + "号，没有"
                                + date[2] + "号，请重新输入！");
                    continue;
                }
            } catch (NumberFormatException e) {
                System.out.println("输入日期应为不能超过" + Integer.MAX_VALUE + "的正
整数，请重新输入！");
            } catch (ArrayIndexOutOfBoundsException e) {
                System.out.println("输入日期应用 - 分隔，请重新输入！");
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return date;
    }

    /**
     * 判断输入的日期是否合法
     *
     * @param date
     *          输入的日期
     * @return 合法返回true 不合法返回false
     */
    private static boolean isCorrectDate(int[] date) {
        if (date[1] == 2 && isRunnian(date[0])) {
            if (date[2] > monthDay[0]) {
                return false;
            } else {
                return true;
            }
        } else {
            if (date[2] > monthDay[date[1]]) {
                return false;
            } else {
                return true;
            }
        }
    }
}
```

2、测试方法及测试用例

利用等价类测试方法解决三角形问题

满足条件	有效等价类		无效等价类	
是否构成三角形	(t1)	edge1>0	(t7)	edge1<=0
	(t2)	edge2>0	(t8)	edge2<=0
	(t3)	edge3>0	(t9)	edge3<=0
	(t4)	edge1+ edge2> edge3	(t10)	edge1+ edge2<= edge3
	(t5)	edge2+ edge3> edge1	(t11)	edge2+ edge3<=edge1
	(t6)	edge1+ edge3> edge2	(t12)	edge1+ edge3<= edge2
是否构成等腰三角形	(t13)	edge1= edge2	(t16)	edge1!= edge2 && edge2!= edge3 && edge1!= edge3
	(t14)	edge1= edge3		
	(t15)	edge2= edge3		
是否构成等边三角形	(t17)	edge1= edge2 && edge2= edge3 && edge1= edge3	(t18)	edge1!= edge2
			(t19)	edge1!= edge3
			(t20)	edge2!= edge3

有效等价类测试用例 (edge1,edge2,edge3)	预期输出	覆盖范围
2，3，4	三边不等的三角形	(t1)、(t2)、(t3)、(t4)、(t5)、(t6)
2，2，2	等边三角形	(t1)、(t2)、(t3)、(t4)、(t5)、(t6)、(t17)
2，2，3	等腰三角形	(t1)、(t2)、(t3)、(t4)、(t5)、(t6)、(t13)
2，3，2	等腰三角形	(t1)、(t2)、(t3)、(t4)、(t5)、(t6)、(t14)
3，2，2	等腰三角形	(t1)、(t2)、(t3)、(t4)、(t5)、(t6)、(t15)

无效等价类测试用例 (edge1,edge2,edge3)	预期输出	覆盖范围
0，3，4	第 1 条边输入有误	(t7)
2，0，4	第 2 条边输入有误	(t8)
2，3，0	第 3 条边输入有误	(t9)

2，2，200	无法构成三角形	(t10)
200，2，3	无法构成三角形	(t11)
2，300，2	无法构成三角形	(t12)
3，5，6	三边不等的三角形	(t16)
3，5，6	三边不等的三角形	(t18)
3，5，6	三边不等的三角形	(t19)
3，5，6	三边不等的三角形	(t20)

利用决策表测试方法解决日期问题

(1) 划分有效等价类：

year 的有效等价类：

Y1:{平年};　Y2:{闰年};

month 的有效等价类：

M1:{1,3,5,7,8,10};

M2:{4,6,9,11};

M3:{12};

M4:{2};

day 的有效等价类：

D1:{1,2,3,…,26};

D2:{27};

D3:{28};

D4:{29};

D5:{30};

D6:{31};

(2) 动作桩：

A1：day=day+2;

A2：day=2;

A3：day=1;

A4：month=month +1;

A5：month=1;

A6：year=year+1;

A7：不可能;

(3) 决策表：

选项 / 规则		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
条件	Month	M1	M1	M1	M1	M1	M1	M2	M2	M2	M2	M2	M2	M3	M3	M3
	day	D1	D2	D3	D4	D5	D6	D1	D2	D3	D4	D5	D6	D1	D2	D3
	Year	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/
动作桩	A1	V	V	V	V			V	V	V				V	V	V
	A2						V					V				
	A3					V					V					
	A4					V	V				V	V				
	A5															
	A6															
	A7												V			

选项 / 规则		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
条件	Month	M3	M3	M3	M4	M4	M4	M4	M4	M4	M4	M4	M4	M4	M4	M4
	day	D4	D5	D6	D1	D1	D2	D2	D3	D3	D4	D4	D5	D5	D6	D6
	Year	/	/	/	Y1	Y2	Y1	Y2	Y1	Y2	Y1	Y2	Y1	Y2	Y1	Y2
动作桩	A1	V			V	V		V								
	A2			V					V			V				
	A3		V				V			V						
	A4						V		V	V		V				
	A5		V	V												
	A6		V	V												
	A7										V		V	V	V	V

(4) 化简后的决策表：

选项 / 规则		1~4	5	6	7~9	10	11	12	13`16	17	18	19~20	21	22	23	24	25	26	27~30
条件	Month	M1	M1	M1	M2	M2	M2	M2	M3	M3	M3	M4	M4	M4	M4	M4	M4	M4	M4
	day	D1~D4	D5	D6	D1~D3	D4	D5	D6	D1~D4	D5	D6	D1~D2	D2	D2	D3	D3	D4	D4	D5~D6
	Year	/	/	/	/	/	/	/	/	/	/	/	Y1	Y2	Y1	Y2	Y1	Y2	/
动作桩	A1	V			V				V			V		V					
	A2			V			V				V				V			V	
	A3		V			V				V			V			V			
	A4		V	V		V	V						V		V	V		V	
	A5									V	V								
	A6									V	V								
	A7							V									V		V

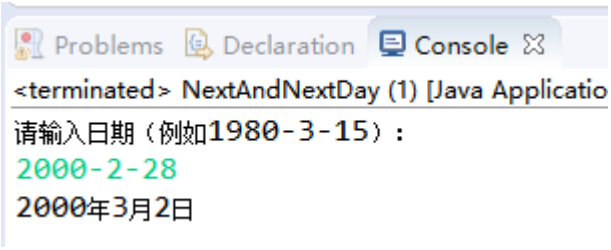
(5) 根据化简后的决策表设计测试用例：

测试用例	Month	day	Year	期望输出
1~4	3	25	2000	2000 年 3 月 27 日
5	3	30	2000	2000 年 4 月 1 日
6	3	31	2000	2000 年 4 月 2 日
7~9	11	25	2000	2000 年 11 月 27 日
10	11	29	2000	2000 年 12 月 1 日
11	11	30	2000	2000 年 12 月 2 日
12	11	31	2000	输入有误
13~16	12	25	2000	2000 年 12 月 27 日
17	12	30	2000	2001 年 1 月 1 日
18	12	31	2000	2001 年 1 月 2 日
19~20	2	25	2000	2000 年 2 月 27 日
21	2	27	2001	2001 年 3 月 1 日
22	2	27	2000	2000 年 2 月 29 日
23	2	28	2001	2001 年 3 月 2 日
24	2	28	2000	2000 年 3 月 1 日
25	2	29	2001	输入有误
26	2	29	2000	2000 年 3 月 2 日
27~30	2	30	2000	输入有误

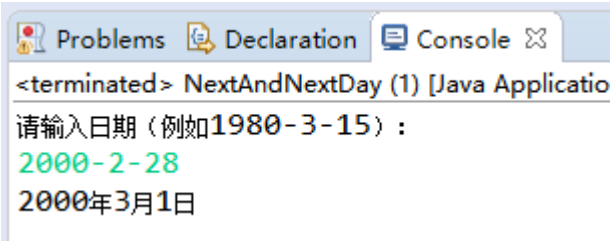
四、发现程序缺陷及修改方案

1、程序缺陷

日期问题中如果输入 2000-2-28 的话，应该输出 2000 年 3 月 1 日，因为 2000 年的 2 月有 29 天，从 28 号开始，隔一天就是 3 月 1 号。而我的程序中忘记了天数这个限定条件，就导致输出了 2000 年 3 月 2 日。具体如图：



错误结果（左图）



正确结果（右图）

2、修改方案

将我的闰年判断和月份放在一起判断，再将我之前设定好的用于存放每个月最大天数的一个全局数组：

【`final private static int[] monthDay = new int[] { 29, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };`】

两者结合起来判断，对日期进行重新计算就解决了这个问题，代码截图如下：

有缺陷的代码：

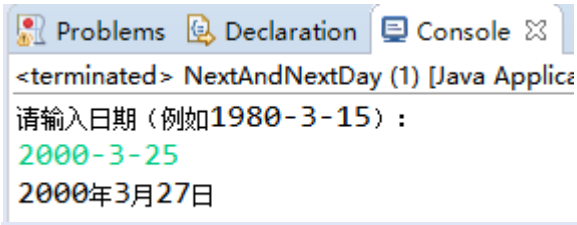
```
// 当前天数=当前天数-这个月的最大天数
date[2] = date[2] - monthDay[date[1]];
date[1]++;
```

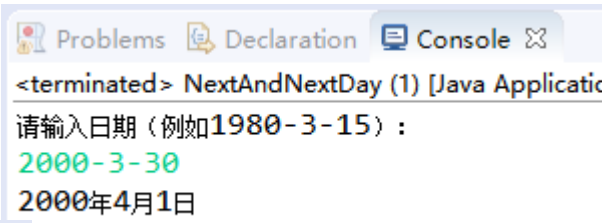
完善后的代码：

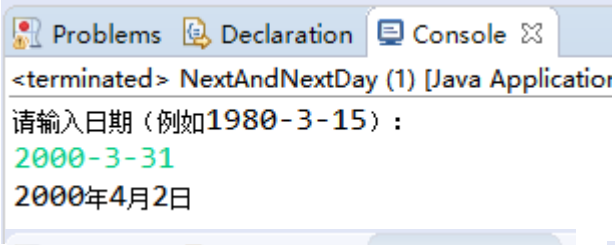
```
// 当前天数=当前天数-这个月的最大天数
if(isRunnian(date[0])&&date[1]==2){
    date[2] = date[2] - monthDay[0];
}else{
    date[2] = date[2] - monthDay[date[1]];
}
date[1]++;
```

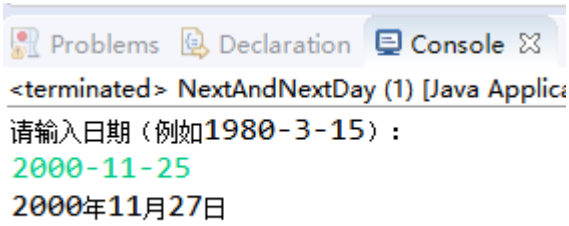
五、测试结果及分析

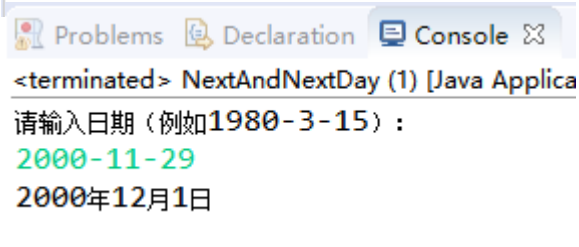
1、测试结果及分析

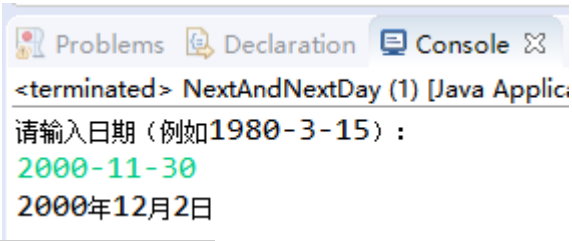


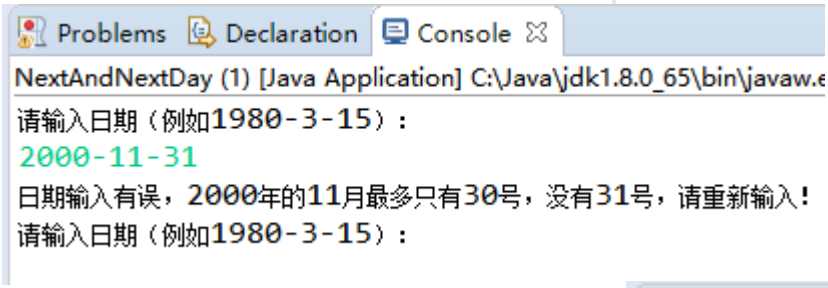


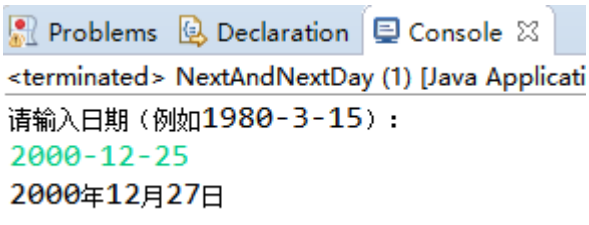


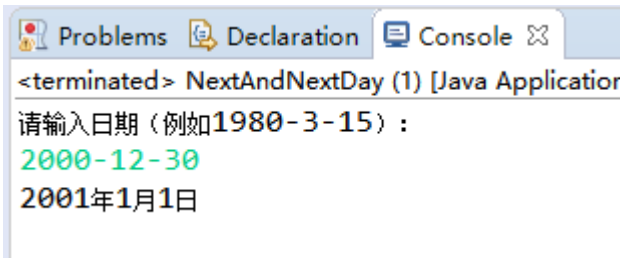


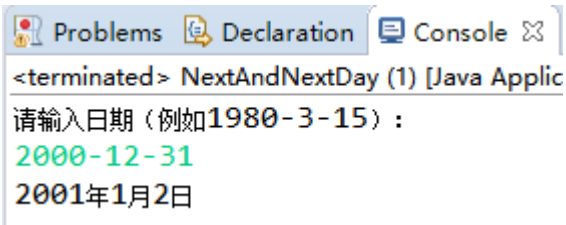


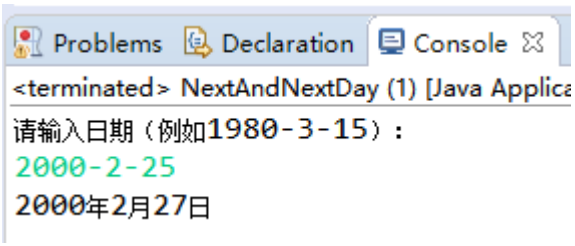


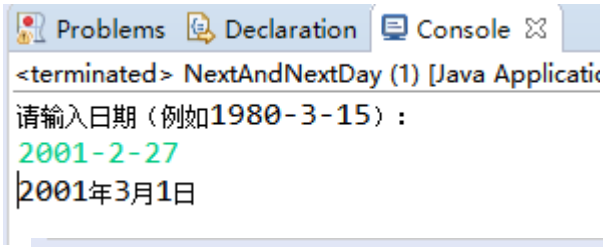


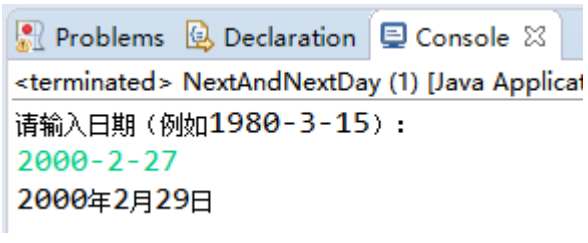


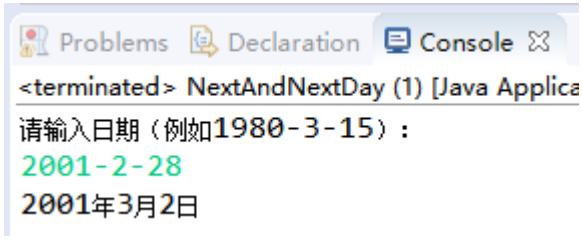


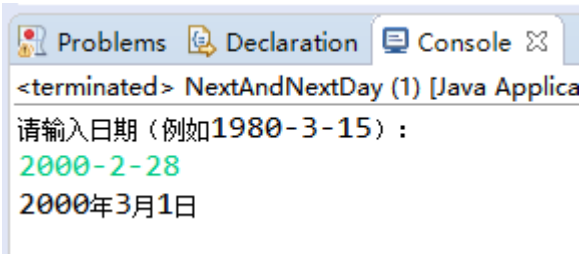


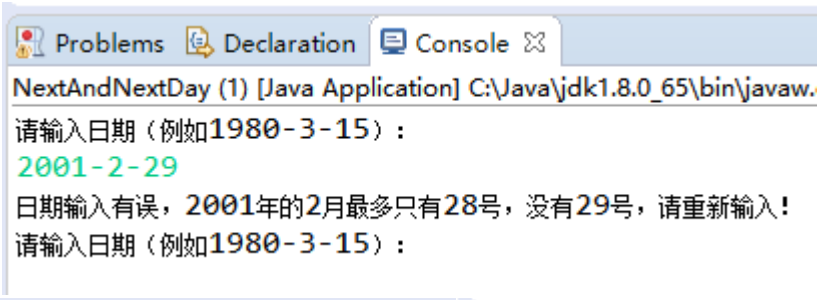


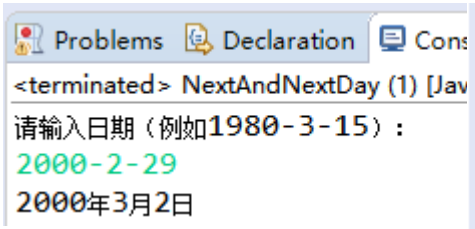


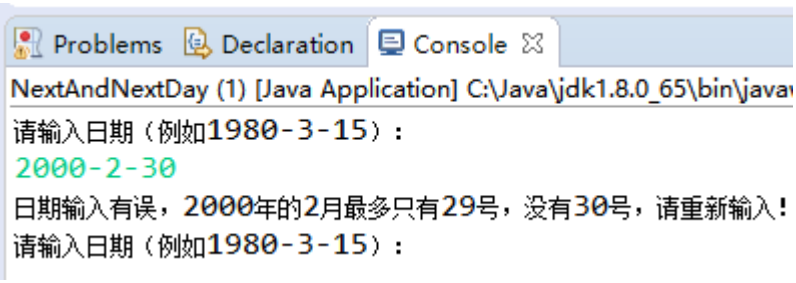












2、心得体会

之前一直以为黑盒测试不用看代码，直接测就好了，感觉一定很简单，比白盒测试简单多了。可是自己做了之后才发现，原来黑盒测试也很麻烦。因为黑盒测试虽然不用了解程序内部的构造，但是正是因为不懂程序的构造才要去设计更多、更完善的测试用例。然而当找测试用例还是像以前一样想到什么找什么，那一定会漏掉测试用例的，而往往漏掉的那些测试用例就是我们程序中出 bug 的地方。所以通过这次实验我充分体会到了找测试用例时选对方法的重要性。这次主要练习了等价类划分、边界值分析和决策表三种方法，个人觉得其实就是递进式的，一级比一级完善，只要照着方法做了，将有效等价类、无效等价类、边界值、动作桩这些东西都设计完善了，那么程序的 bug 也就没地方藏了。总之，这次黑盒测试很成功，收获了很多。