**1.1 实验类型**

实验类型为综合设计型，2个学时。

**1.2 实验目的**

（1）能熟练应用黑盒测试技术进行测试用例设计；

（2）能够熟练应用JUnit框架，并能够实现数据与代码的分离测试

（3）能够熟练应用数据库或文件设计测试用例结构，并进行优化设计

**1.3 实验内容**

**题目1：日期问题**

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、1≤day≤31和1900≤year≤2050)，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2023年11月30日，则该程序的输出为2023年12月1日。

（1） 划分等价类；

（2） 按照等价类划分法设计测试用例；

（3） 按照边界值分析法设计测试用例；

（4） 按照判定表驱动法设计测试用例。

（5） 掌握Junit4的用法，使用Junit4测试nextDate 函数。

**要求：**

（1） 自行编写nextDate函数；

（2） 采用测试用例与测试代码分离的方法设计测试框架；

用例的结构需要设计

（3） 测试用例的存储可以采用数据库也可以采用文本文件的形式，但测试用例的结构需要设计；

（4） 采用黑盒测试方法中的等价类方法，边界值方法，决策表方法设计测

试用例，进行测试判断；和决策表等）、测试代码、测试用例库和运行结果

（5） 编写Junit测试用例；

（6） 根据实验结果分析测试用例设计有效性和被测程序的正确；

（7） 撰写实验报告，要求给出测试用例测试过程（等价类划分表、因果图 ）。

**解答：**

1. 划分等价类

|  |  |  |
| --- | --- | --- |
|  | 有效等价类 | 无效等价类 |
| 年 | ①1900≤year≤2500的闰年 | ⑩>2050 |
| ②1900≤year≤2500的平年 | ⑪<1900 |
|  | ⑫其他非年份值输入 |
| 月 | ③2 | ⑬<1 |
| ④1/3/5/7/8/10/12 | ⑭>12 |
| ⑤4/6/9/11 | ⑮其他非月份值输入 |
| 日 | ⑥1~28 | ⑯day<1 |
| ⑦29 | ⑰day>31 |
| ⑧30 | ⑱其他非日值输入 |
| ⑨31 | ⑲ 当①&&③时，day>29 |
|  | ⑳ 当②&&③时，day>28 |
|  | ㉑ 当④时，day>31 |
|  | ㉒ 当⑤时,day>30 |

1. 按照等价类划分法设计测试用例

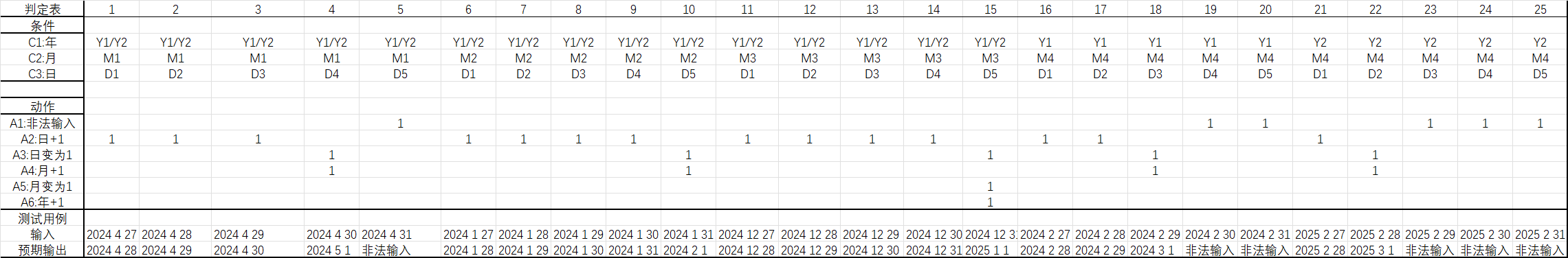
|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例编号 | 测试用例 | 覆盖等价类 | 预期输出 |
| 1 | 2025 4 23 | ②⑤⑥ | 2025年4月23日 |
| 2 | 2024 2 29 | ①③⑦ | 2024年3月1日 |
| 3 | 2024 5 31 | ①④⑨ | 2024年6月1日 |
| 4 | 2024 4 30 | ①⑤⑧ | 2024年5月1日 |
| 5 | 2051 1 1 | ⑩ | 非法输入 |
| 6 | 1888 1 1 | ⑪ | 非法输入 |
| 7 | ？ 1 1 | ⑫ | 非法输入 |
| 8 | 2000 0 1 | ⑬ | 非法输入 |
| 9 | 2000 13 1 | ⑭ | 非法输入 |
| 10 | 2000 ？ 1 | ⑮ | 非法输入 |
| 11 | 2000 1 0 | ⑯ | 非法输入 |
| 12 | 2000 1 32 | ⑰ | 非法输入 |
| 13 | 2000 1 ？ | ⑱ | 非法输入 |
| 14 | 2024 2 30 | ⑲ | 非法输入 |
| 15 | 2025 2 29 | ⑳ | 非法输入 |
| 16 | 2024 5 32 | ㉑ | 非法输入 |
| 17 | 2024 4 31 | ㉒ | 非法输入 |

1. 按照边界值分析法设计测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 年 | 月 | 日 | 预期输出 |
| 1 | 1899 | 4 | 25 | 非法输入 |
| 2 | 1900 | 4 | 25 | 1900年4月26日 |
| 3 | 1901 | 4 | 25 | 1901年4月26日 |
| 4 | 2024 | 4 | 25 | 2024年4月26日 |
| 5 | 2499 | 4 | 25 | 2499年4月26日 |
| 6 | 2500 | 4 | 25 | 2500年4月26日 |
| 7 | 2501 | 4 | 25 | 非法输入 |
| 8 | 2024 | 0 | 15 | 非法输入 |
| 9 | 2024 | 1 | 15 | 2024年1月16日 |
| 10 | 2024 | 2 | 15 | 2024年2月16日 |
| 11 | 2024 | 6 | 15 | 2024年6月16日 |
| 12 | 2024 | 11 | 15 | 2024年11月16日 |
| 13 | 2024 | 12 | 15 | 2024年12月16日 |
| 14 | 2024 | 13 | 15 | 非法输入 |
| 15 | 2024 | 2 | 30 | 非法输入 |
| 16 | 2024 | 2 | 29 | 2024年3月1日 |
| 17 | 2024 | 2 | 1 | 2024年2月2日 |
| 18 | 2024 | 2 | 15 | 2024年2月16日 |
| 19 | 2024 | 2 | 2 | 2024年2月3日 |
| 20 | 2024 | 2 | 28 | 2024年2月29日 |
| 21 | 2024 | 2 | 0 | 非法输入 |
| 22 | 2024 | 1 | 0 | 非法输入 |
| 23 | 2024 | 1 | 1 | 2024年1月2日 |
| 24 | 2024 | 1 | 2 | 2024年1月3日 |
| 25 | 2024 | 1 | 15 | 2024年1月16日 |
| 26 | 2024 | 1 | 30 | 2024年1月31日 |
| 27 | 2024 | 1 | 31 | 2024年2月1日 |
| 28 | 2024 | 1 | 32 | 非法输入 |
| 29 | 2024 | 4 | 0 | 非法输入 |
| 30 | 2024 | 4 | 1 | 2024年4月2日 |
| 31 | 2024 | 4 | 2 | 2024年4月3日 |
| 32 | 2024 | 4 | 15 | 2024年4月16日 |
| 33 | 2024 | 4 | 29 | 2024年4月30日 |
| 34 | 2024 | 4 | 30 | 2024年5月1日 |
| 35 | 2024 | 4 | 31 | 非法输入 |
| 36 | 2025 | 2 | 0 | 非法输入 |
| 37 | 2025 | 2 | 1 | 2025年2月2日 |
| 38 | 2025 | 2 | 2 | 2025年2月3日 |
| 39 | 2025 | 2 | 15 | 2025年2月16日 |
| 40 | 2025 | 2 | 27 | 2025年2月28日 |
| 41 | 2025 | 2 | 28 | 2025年3月1日 |
| 42 | 2024 | 2 | 29 | 非法输入 |

1. 按照判定表驱动法设计测试用例

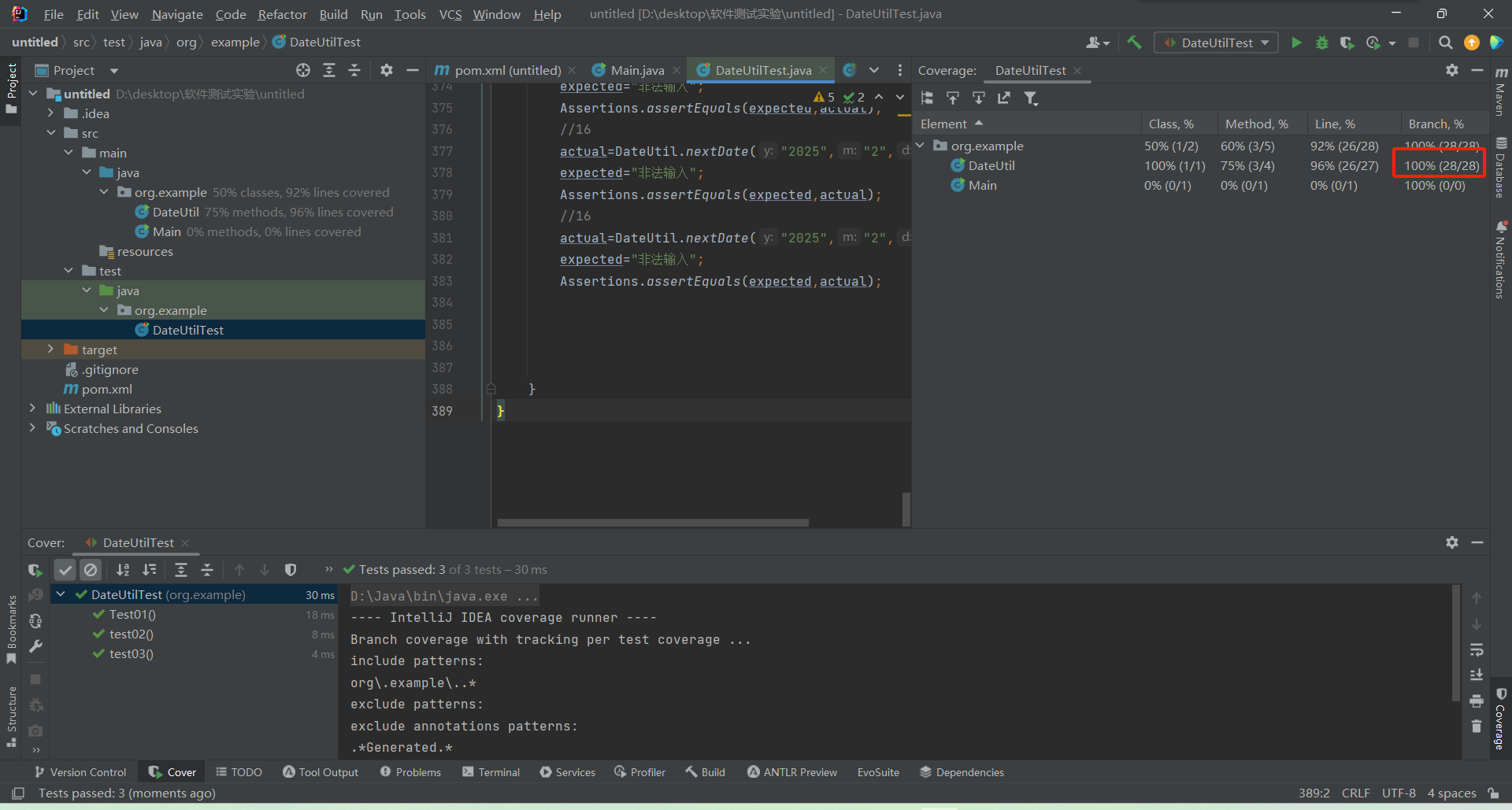
|  |  |
| --- | --- |
| 条件桩 |  |
| Y1 | 闰年 |
| Y2 | 平年 |
|  |  |
| M1 | 每月有30天的月份 |
| M2 | 每月有31天的月份（除12月） |
| M3 | 12月 |
| M4 | 2月 |
|  |  |
| D1 | 1≤day≤27 |
| D2 | day=28 |
| D3 | day=29 |
| D4 | day=30 |



1. 掌握Junit4的用法，使用Junit4测试nextDate 函数。

**nextDate 函数**：

package org.example;  
  
public class DateUtil {  
 */\*\* 判断是否为闰年 \*\*/* public static boolean isLeapYear(int year) {  
 return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);  
 }  
 */\*\*\* 判断输入的日期是否合法 \*/* public static boolean isValidDate(int year, int month, int day) {  
 if (year < 1900 || year > 2050) return false;  
 if (month < 1 || month > 12) return false;  
  
 int[] daysInMonth = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};  
 if (*isLeapYear*(year)) {  
 daysInMonth[1] = 29; // 闰年2月29天  
 }  
  
 return day >= 1 && day <= daysInMonth[month - 1];  
 }  
 */\*\* 计算下一天 返回格式 YYYY年MM月DD日 \*/* public static String nextDate(String y, String m, String d) {  
  
 int year ;  
 int month ;  
 int day ;  
 try { */\*\* 排除一些非数值的输入 例如年份为 "dsda" \*/* year = Integer.*parseInt*(y);  
 month = Integer.*parseInt*(m);  
 day = Integer.*parseInt*(d);  
  
 } catch (NumberFormatException e) {  
 return "非法输入";  
 }  
  
  
 if(!*isValidDate*(year,month,day)){  
 return "非法输入";  
 }  
  
  
 int[] daysInMonth = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};  
  
 // 处理闰年2月  
 if (*isLeapYear*(year)) {  
 daysInMonth[1] = 29;  
 }  
  
 // 增加一天  
 day++;  
  
 // 检查是否超过了当前月份的天数  
 if (day > daysInMonth[month - 1]) {  
 day = 1;  
 month++;  
 // 检查是否超过了12月  
 if (month > 12) {  
 month = 1;  
 year++;  
 }  
 }  
  
 return String.*format*("%d年%d月%d日", year, month, day);  
 }  
  
 public static void main(String[] args) {  
  
 }  
}



电脑萤幕的截图

描述已自动生成

**测试代码：**

package org.example;  
  
import org.junit.jupiter.api.Assertions;  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class DateUtilTest {  
  
 */\*\*  
 \*  
 \* 等价类测试用例  
 \* \*/* @Test  
 public void Test01(){  
 String year="2025";  
 String month="4";  
 String day="23";  
 // 1  
 String actual=DateUtil.*nextDate*(year,month,day);  
 String expected="2025年4月24日";  
 Assertions.*assertEquals*(actual,expected);  
 // 2  
 actual=DateUtil.*nextDate*("2024","2","29");  
 expected="2024年3月1日";  
 Assertions.*assertEquals*(actual,expected);  
 // 3  
 actual=DateUtil.*nextDate*("2024","5","31");  
 expected="2024年6月1日";  
 Assertions.*assertEquals*(actual,expected);  
 //4  
 actual=DateUtil.*nextDate*("2024","4","30");  
 expected="2024年5月1日";  
 Assertions.*assertEquals*(actual,expected);  
 //5  
 actual=DateUtil.*nextDate*("2051","1","1");  
 expected="非法输入";  
 Assertions.*assertEquals*(actual,expected);  
 //6  
 actual=DateUtil.*nextDate*("1888","1","1");  
 expected="非法输入";  
 Assertions.*assertEquals*(actual,expected);  
 //7  
 actual=DateUtil.*nextDate*("sda","1","1");  
 expected="非法输入";  
 Assertions.*assertEquals*(actual,expected);  
 //8  
 actual=DateUtil.*nextDate*("2000","0","1");  
 expected="非法输入";  
 Assertions.*assertEquals*(actual,expected);  
 //9  
 actual=DateUtil.*nextDate*("2024","13","1");  
 expected="非法输入";  
 Assertions.*assertEquals*(actual,expected);  
 //10  
 actual=DateUtil.*nextDate*("2000","asda","1");  
 expected="非法输入";  
 Assertions.*assertEquals*(actual,expected);  
 //11  
 actual=DateUtil.*nextDate*("2000","1","0");  
 expected="非法输入";  
 Assertions.*assertEquals*(actual,expected);  
 //12  
 actual=DateUtil.*nextDate*("2000","1","32");  
 expected="非法输入";  
 Assertions.*assertEquals*(actual,expected);  
 //13  
 actual=DateUtil.*nextDate*("2000","1","sda");  
 expected="非法输入";  
 Assertions.*assertEquals*(actual,expected);  
 //14  
 actual=DateUtil.*nextDate*("2024","2","30");  
 expected="非法输入";  
 Assertions.*assertEquals*(actual,expected);  
 //15  
 actual=DateUtil.*nextDate*("2025","2","29");  
 expected="非法输入";  
 Assertions.*assertEquals*(actual,expected);  
 //16  
 actual=DateUtil.*nextDate*("2024","5","32");  
 expected="非法输入";  
 Assertions.*assertEquals*(actual,expected);  
 //17  
 actual=DateUtil.*nextDate*("2024","4","31");  
 expected="非法输入";  
 Assertions.*assertEquals*(actual,expected);  
  
 }  
 //边界值  
  
 */\*\*  
 \*  
 \* 用边界值  
 \*/* @Test  
 public void test02(){  
 String actual="";  
 String expected="";  
 //年份 的健壮性边界值分析  
 //1  
 actual=DateUtil.*nextDate*("1899","4","25");  
 expected="非法输入";  
 Assertions.*assertEquals*(actual,expected);  
 //2  
 actual=DateUtil.*nextDate*("1900","4","25");  
 expected="1900年4月26日";  
 Assertions.*assertEquals*(actual,expected);  
 //3  
 actual=DateUtil.*nextDate*("1901","4","25");  
 expected="1901年4月26日";  
 Assertions.*assertEquals*(actual,expected);  
 //4  
 actual=DateUtil.*nextDate*("2024","4","25");  
 expected="2024年4月26日";  
 Assertions.*assertEquals*(actual,expected);  
 //5  
 actual=DateUtil.*nextDate*("2049","4","25");  
 expected="2049年4月26日";  
 Assertions.*assertEquals*(actual,expected);  
 //6  
 actual=DateUtil.*nextDate*("2050","4","25");  
 expected="2050年4月26日";  
 Assertions.*assertEquals*(actual,expected);  
 //7  
 actual=DateUtil.*nextDate*("2051","4","25");  
 expected="非法输入";  
 Assertions.*assertEquals*(actual,expected);  
 //8  
 actual=DateUtil.*nextDate*("2024","0","15");  
 expected="非法输入";  
 Assertions.*assertEquals*(actual,expected);  
 //9  
 actual=DateUtil.*nextDate*("2024","1","15");  
 expected="2024年1月16日";  
 Assertions.*assertEquals*(actual,expected);  
 //10  
 actual=DateUtil.*nextDate*("2024","2","15");  
 expected="2024年2月16日";  
 Assertions.*assertEquals*(actual,expected);  
 //11  
 actual=DateUtil.*nextDate*("2024","6","15");  
 expected="2024年6月16日";  
 Assertions.*assertEquals*(actual,expected);  
 //12  
 actual=DateUtil.*nextDate*("2024","11","15");  
 expected="2024年11月16日";  
 Assertions.*assertEquals*(actual,expected);  
 //13  
 actual=DateUtil.*nextDate*("2024","12","15");  
 expected="2024年12月16日";  
 Assertions.*assertEquals*(actual,expected);  
 //14  
 actual=DateUtil.*nextDate*("2024","13","15");  
 expected="非法输入";  
 Assertions.*assertEquals*(actual,expected);  
 //15  
 actual=DateUtil.*nextDate*("2024","2","30");  
 expected="非法输入";  
 Assertions.*assertEquals*(actual,expected);  
 //16  
 actual=DateUtil.*nextDate*("2024","2","29");  
 expected="2024年3月1日";  
 Assertions.*assertEquals*(expected,actual);  
 //17  
 actual=DateUtil.*nextDate*("2024","2","1");  
 expected="2024年2月2日";  
 Assertions.*assertEquals*(expected,actual);  
 //18  
 actual=DateUtil.*nextDate*("2024","2","15");  
 expected="2024年2月16日";  
 Assertions.*assertEquals*(expected,actual);  
 //19  
 actual=DateUtil.*nextDate*("2024","2","2");  
 expected="2024年2月3日";  
 Assertions.*assertEquals*(expected,actual);  
 //20  
 actual=DateUtil.*nextDate*("2024","2","28");  
 expected="2024年2月29日";  
 Assertions.*assertEquals*(expected,actual);  
 //21  
 actual=DateUtil.*nextDate*("2024","2","0");  
 expected="非法输入";  
 Assertions.*assertEquals*(expected,actual);  
 //22  
 actual=DateUtil.*nextDate*("2024","1","0");  
 expected="非法输入";  
 Assertions.*assertEquals*(expected,actual);  
 //23  
 actual=DateUtil.*nextDate*("2024","1","1");  
 expected="2024年1月2日";  
 Assertions.*assertEquals*(expected,actual);  
 //24  
 actual=DateUtil.*nextDate*("2024","1","2");  
 expected="2024年1月3日";  
 Assertions.*assertEquals*(expected,actual);  
 //25  
 actual=DateUtil.*nextDate*("2024","1","15");  
 expected="2024年1月16日";  
 Assertions.*assertEquals*(expected,actual);  
 //26  
 actual=DateUtil.*nextDate*("2024","1","30");  
 expected="2024年1月31日";  
 Assertions.*assertEquals*(expected,actual);  
 //27  
 actual=DateUtil.*nextDate*("2024","1","31");  
 expected="2024年2月1日";  
 Assertions.*assertEquals*(expected,actual);  
 //28  
 actual=DateUtil.*nextDate*("2024","1","32");  
 expected="非法输入";  
 Assertions.*assertEquals*(expected,actual);  
 //29  
 actual=DateUtil.*nextDate*("2024","4","0");  
 expected="非法输入";  
 Assertions.*assertEquals*(expected,actual);  
 //30  
 actual=DateUtil.*nextDate*("2024","4","1");  
 expected="2024年4月2日";  
 Assertions.*assertEquals*(expected,actual);  
 //31  
 actual=DateUtil.*nextDate*("2024","4","2");  
 expected="2024年4月3日";  
 Assertions.*assertEquals*(expected,actual);  
 //32  
 actual=DateUtil.*nextDate*("2024","4","15");  
 expected="2024年4月16日";  
 Assertions.*assertEquals*(expected,actual);  
 //33  
 actual=DateUtil.*nextDate*("2024","4","29");  
 expected="2024年4月30日";  
 Assertions.*assertEquals*(expected,actual);  
 //34  
 actual=DateUtil.*nextDate*("2024","4","30");  
 expected="2024年5月1日";  
 Assertions.*assertEquals*(expected,actual);  
 //35  
 actual=DateUtil.*nextDate*("2024","4","31");  
 expected="非法输入";  
 Assertions.*assertEquals*(expected,actual);  
 //36  
 actual=DateUtil.*nextDate*("2025","2","0");  
 expected="非法输入";  
 Assertions.*assertEquals*(expected,actual);  
 //37  
 actual=DateUtil.*nextDate*("2025","2","1");  
 expected="2025年2月2日";  
 Assertions.*assertEquals*(expected,actual);  
 //38  
 actual=DateUtil.*nextDate*("2025","2","2");  
 expected="2025年2月3日";  
 Assertions.*assertEquals*(expected,actual);  
 //39  
 actual=DateUtil.*nextDate*("2025","2","15");  
 expected="2025年2月16日";  
 Assertions.*assertEquals*(expected,actual);  
 //40  
 actual=DateUtil.*nextDate*("2025","2","27");  
 expected="2025年2月28日";  
 Assertions.*assertEquals*(expected,actual);  
 //41  
 actual=DateUtil.*nextDate*("2025","2","28");  
 expected="2025年3月1日";  
 Assertions.*assertEquals*(expected,actual);  
 //42  
 actual=DateUtil.*nextDate*("2025","2","29");  
 expected="非法输入";  
 Assertions.*assertEquals*(expected,actual);  
  
 }  
 */\*\*  
 \*  
 \* 判定表  
 \*/* @Test  
 public void test03(){  
 String actual="";  
 String expected="";  
 //1  
 actual=DateUtil.*nextDate*("2024","4","27");  
 expected="2024年4月28日";  
 Assertions.*assertEquals*(expected,actual);  
 //2  
 actual=DateUtil.*nextDate*("2024","4","28");  
 expected="2024年4月29日";  
 Assertions.*assertEquals*(expected,actual);  
 //3  
 actual=DateUtil.*nextDate*("2024","4","29");  
 expected="2024年4月30日";  
 Assertions.*assertEquals*(expected,actual);  
 //4  
 actual=DateUtil.*nextDate*("2024","4","30");  
 expected="2024年5月1日";  
 Assertions.*assertEquals*(expected,actual);  
 //5  
 actual=DateUtil.*nextDate*("2024","4","31");  
 expected="非法输入";  
 Assertions.*assertEquals*(expected,actual);  
  
 //6  
 actual=DateUtil.*nextDate*("2024","1","27");  
 expected="2024年1月28日";  
 Assertions.*assertEquals*(expected,actual);  
 //7  
 actual=DateUtil.*nextDate*("2024","1","28");  
 expected="2024年1月29日";  
 Assertions.*assertEquals*(expected,actual);  
 //8  
 actual=DateUtil.*nextDate*("2024","1","29");  
 expected="2024年1月30日";  
 Assertions.*assertEquals*(expected,actual);  
 //9  
 actual=DateUtil.*nextDate*("2024","1","30");  
 expected="2024年1月31日";  
 Assertions.*assertEquals*(expected,actual);  
 //10  
 actual=DateUtil.*nextDate*("2024","1","31");  
 expected="2024年2月1日";  
 Assertions.*assertEquals*(expected,actual);  
 //11  
 actual=DateUtil.*nextDate*("2024","12","27");  
 expected="2024年12月28日";  
 Assertions.*assertEquals*(expected,actual);  
 //12  
 actual=DateUtil.*nextDate*("2024","12","28");  
 expected="2024年12月29日";  
 Assertions.*assertEquals*(expected,actual);  
 //13  
 actual=DateUtil.*nextDate*("2024","12","29");  
 expected="2024年12月30日";  
 Assertions.*assertEquals*(expected,actual);  
 //14  
 actual=DateUtil.*nextDate*("2024","12","30");  
 expected="2024年12月31日";  
 Assertions.*assertEquals*(expected,actual);  
 //15  
 actual=DateUtil.*nextDate*("2024","12","31");  
 expected="2025年1月1日";  
 Assertions.*assertEquals*(expected,actual);  
  
 //16  
 actual=DateUtil.*nextDate*("2024","2","27");  
 expected="2024年2月28日";  
 Assertions.*assertEquals*(expected,actual);  
 //16  
 actual=DateUtil.*nextDate*("2024","2","28");  
 expected="2024年2月29日";  
 Assertions.*assertEquals*(expected,actual);  
 //16  
 actual=DateUtil.*nextDate*("2024","2","29");  
 expected="2024年3月1日";  
 Assertions.*assertEquals*(expected,actual);  
 //16  
 actual=DateUtil.*nextDate*("2024","2","30");  
 expected="非法输入";  
 Assertions.*assertEquals*(expected,actual);  
 //16  
 actual=DateUtil.*nextDate*("2024","2","31");  
 expected="非法输入";  
 Assertions.*assertEquals*(expected,actual);  
 //16  
 actual=DateUtil.*nextDate*("2025","2","27");  
 expected="2025年2月28日";  
 Assertions.*assertEquals*(expected,actual);  
 //16  
 actual=DateUtil.*nextDate*("2025","2","28");  
 expected="2025年3月1日";  
 Assertions.*assertEquals*(expected,actual);  
 //16  
 actual=DateUtil.*nextDate*("2025","2","29");  
 expected="非法输入";  
 Assertions.*assertEquals*(expected,actual);  
 //16  
 actual=DateUtil.*nextDate*("2025","2","30");  
 expected="非法输入";  
 Assertions.*assertEquals*(expected,actual);  
 //16  
 actual=DateUtil.*nextDate*("2025","2","31");  
 expected="非法输入";  
 Assertions.*assertEquals*(expected,actual);  
  
  
  
  
 }  
}

**题目2：自动售货机**

有一个处理单价为5角钱的饮料的自动售货机，其规格说明如下：

（1）若投入5角钱或1元钱的硬币，押下〖橙汁〗或〖啤酒〗的按钮，则相应的

饮料就送出来；

（2）若售货机没有零钱找，则一个显示〖零钱找完〗的红灯亮，这时在投入1

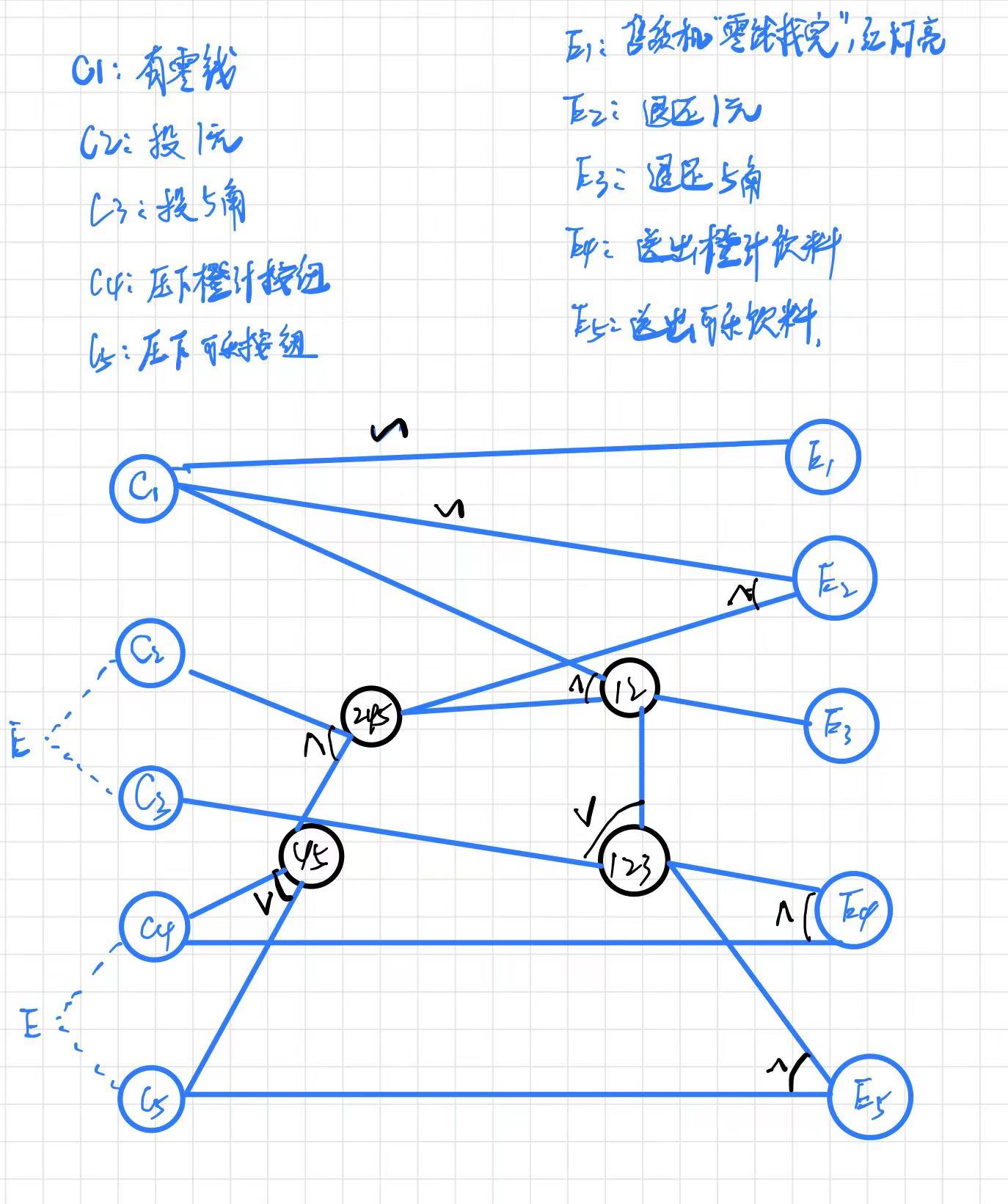
元硬币并押下按钮后，饮料不送出来而且1元硬币也退出来；

（3）若有零钱找，则显示〖零钱找完〗的红灯灭，在送出饮料的同时退还5角

硬币。

**要求：**

请绘制出因果图和判定表，并给出相应的测试用例。



图片包含 日历

描述已自动生成

输入1：售货机没有零钱找，投入1元，压下橙汁按钮

预期结果：售货机"零钱找完"灯亮，不送出饮料，退还1元

输入2：售货机没有零钱找，投入1元，压下啤酒按钮

预期结果：售货机"零钱找完"灯亮，不送出饮料，退还1元

输入3：售货机没有零钱找，投入1元

预期结果：售货机"零钱找完"灯亮

输入4：售货机没有零钱找，投入1元

预期结果：售货机"零钱找完"灯亮

输入5：售货机没有零钱找，投入5角，压下橙汁按钮

预期结果：售货机"零钱找完"灯亮，送出饮料

输入6：售货机没有零钱找，投入5角，压下啤酒按钮

预期结果：售货机"零钱找完"灯亮，送出饮料

输入7：售货机没有零钱找，投入5角

预期结果：售货机"零钱找完"灯亮

输入8：售货机没有零钱找，压下啤酒按钮

预期结果：售货机"零钱找完"灯亮

输入9：售货机没有零钱找，压下橙汁按钮

预期结果：售货机"零钱找完"灯亮

输入9：售货机有零钱找，投入1元，压下橙汁按钮

预期结果：送出橙汁饮料，找回5角硬币

输入10：售货机有零钱找，投入1元，压下啤酒按钮

预期结果：送出啤酒饮料，找回5角硬币

输入11：售货机有零钱找，投入1元

预期结果：无反应

输入12：售货机有零钱找，投入5角，压下橙汁按钮

预期结果：送出橙汁饮料

输入13：售货机有零钱找，投入5角，压下啤酒按钮

预期结果：送出啤酒饮料

输入14：售货机有零钱找，投入5角

预期结果：无反应

输入15：售货机有零钱找，压下橙汁按钮

预期结果：无反应

输入16：售货机有零钱找，压下啤酒按钮

预期结果：无反应

**1.4 实验步骤**

（1） 根据黑盒测试技术设计测试用例；

（2） 根据所学知识确定优化策略（原则：用最少的用例检测出更多的缺陷、

软件测试的充分性与冗余性考虑）；

（3） 使用Junit为题目1编写并运行测试用例。

**1.5 实验思考**

（1） 在实际的测试中，如何设计测试用例才能达到用最少的测试用例检测

出最多的缺陷；

要用最少的测试用例检测出最多的缺陷，首先需进行充分的需求分析和等价类确定，找出所有边界值并设计交集用例。根据测试用例的使用频率和重要程度排序，并优先选择包含边界值和边界值组合的测试用例，确保每个等价类的每个代表值至少在一个测试用例中出现。

（2） 在进行用例设计时，如何考虑软件测试用例的充分性和减少软件测试

用例的冗余性。

要考虑软件测试用例的充分性和减少冗余性，需要熟悉需求和架构设计，精心选择测试数据以达到需求覆盖率，并安排测试用例执行顺序以重用测试数据，注意各模块之间的关联，合理设计测试用例。