**《软件测试》**

**实验报告四 ——决策表分析方法**

**姓 名： 徐静 学 号： 2020112825**

**院 系： 计算机与信息学院 专 业：数字媒体技术\_\_\_\_\_**

**实 验 室： 实验日期： 2021/5/03**

**总评成绩： 审阅教师：**

### 一、实验目的：

1. 巩固 git 协作式管理工具的使用；
2. 巩固 gradle 项目构建 Java 应用的基本操作
3. 掌握决策表分析方法设计测试用例的方法
4. 能够运用功能测试、基于控制流和数据流的测试等软件测试的核心技术和原理，结合相关文献，对测试问题及其影响因素进行分析或计算，得出有效结论。

### 二、实验环境：

IDEA2020,JUnit

### 三、实验要求：

1. 实现电话账单收费功能
2. 采用决策表分析方法设计测试用例，并采用Junit5 编写电话账单收费程序测试用例，并提交到代码仓库
3. 使用checkstyle生成代码规范检查报告
4. 使用jacoco生成测试覆盖率报告
5. 使用spotbugs插件检查代码，生成代码安全检查报告，并修复检查出来的bug
6. 按照[实验报告模板](https://star.jmhui.com.cn/u/cms/www/202203/06160105hi3m.docx)编写实验报告，以“学号-姓名-软件测试实验四”命名，提交到雨课堂“软件测试实验四”

### **四、实验步骤与内容**

1. **题目：**

对于电话账单来说，当春季和秋季标准时间与夏令时时间进行转换时会产生一个很有意思的问题: 春季，这种转换发生在(3月某个)星期日凌晨2:00点，这时要将时钟设置为凌晨3:00点；秋季， 转换通常在11月的第一个星期日，时钟要从2:59:59调回2:00:00。

请为长途电话服务函数开发计费类，使用等价类分析方法构建测试用例并使用Junit5测试。

1. **采用如下计费规则计算通话费：**
2. 通话时间小于等于20分钟时，每分钟收费0.05美元，通话时间不足1分钟按1分钟计算。
3. 通话时间大于20分钟时，收费1.00美元，外加超过20分钟的部分每分钟0.10美元；
4. 不到1分钟按1分钟计算
5. **假设：**

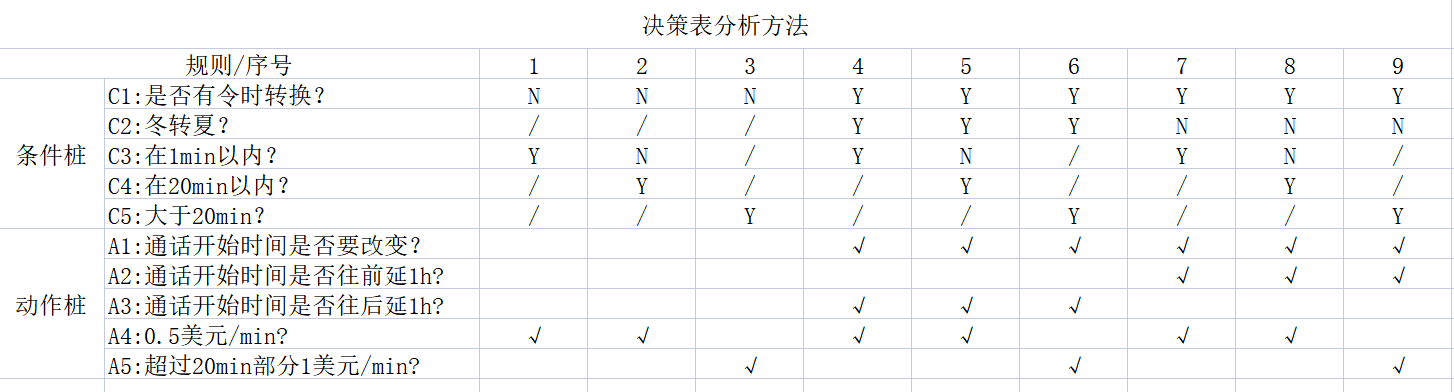
* 通话计费时间从被叫方应答开始计算，到呼叫方挂机时结束；
* 通话时间的秒数向上进位到分钟；
* 没有超过30个小时的通话。

1. **测试用例：**

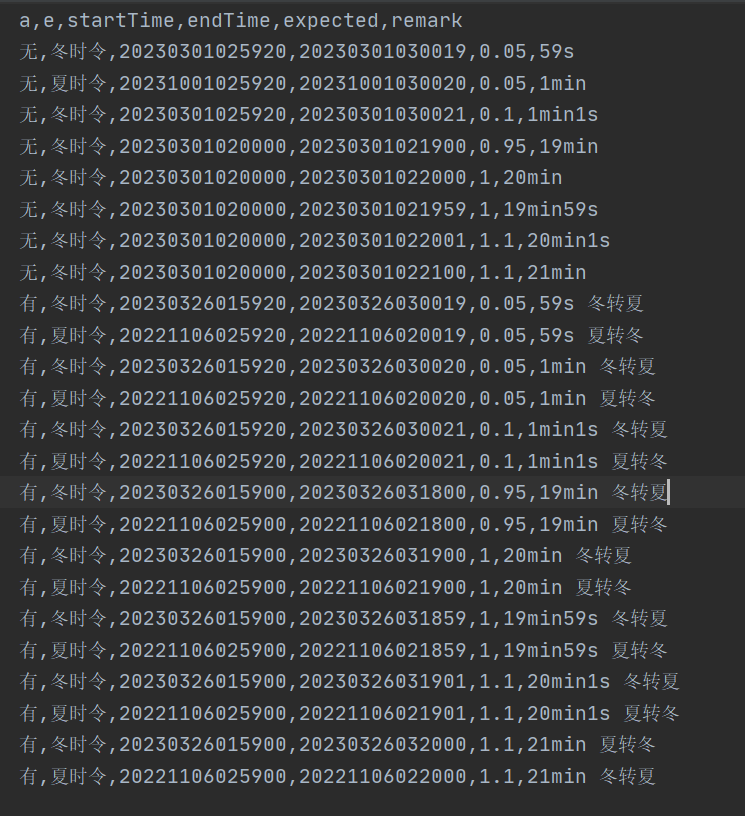
决策表分析：

1. **代码实现：**

首先需要说明一下的是本实验是基于实验三所做的另一种测试方法的分析，所以这里在实验三的代码上不做修改，将测试方法由上次实验的等价类分析方法改为决策表分析方法：



测试用例不发生改变：



|  |
| --- |
| **实现代码：**  package inittaskdemo3.app;  import java.util.Date;  public class Bill {  //name1有无时令转换 name2通话开始的时令  //StartTime起始时间 EndTime结束时间  public double get\_time(String name1, String name2, Date StartTime, Date EndTime)  {  //nm表示一分钟的毫秒数  double nm = 1000 \* 60;  long diff;  //通话时间  double min;  //返回自1970年1月1日以来，以此日期为准的00:00:00 GMT的毫秒数  long e = EndTime.getTime();  long s = StartTime.getTime();  double sum = 0;  if(name1.equals("有")) {  //通话时时令由冬时令转化为夏时令  if(name2.equals("冬时令"))  e -= nm \* 60;  //通话时时令由夏时令转化为冬时令  else if(name2.equals("夏时令"))  e += nm \* 60;  }  //计算通话时间  if(e<s)  min=-1;  else {  diff = e - s;  min = diff / nm;  }  //对一个数进行上舍入  min=Math.ceil(min);  if(min==-1)  return min;  else if(min <= 20)  {  sum = min \* 0.05;  }  else {  sum =1 + ( min - 20 ) \* 0.1;  }  String str = String.format("%.2f",sum);  sum = Double.parseDouble(str);  return sum;  }  }  测试代码：  package inittaskdemo3.app;  import org.junit.jupiter.api.DisplayName;  import org.junit.jupiter.params.ParameterizedTest;  import org.junit.jupiter.params.provider.CsvFileSource;  import static org.junit.jupiter.api.Assertions.assertEquals;  import java.text.ParseException;  import java.text.SimpleDateFormat;  import java.util.Date;  public class BillTest1 {  public Date StrToDate(String str) {  //SimpleDateFormat 允许你选择任何用户自定义日期时间格式来运行  SimpleDateFormat format = new SimpleDateFormat("yyyyMMddHHmmss");  Date date = null;  //当try语句中出现异常时，会执行catch 中的语句  try {  //parse()将字符串转换为日期类型，以进行后续操作  date = format.parse(str);  } catch (ParseException e) {  //printStackTrace()：在命令行打印异常信息在程序中出错的位置及原因  e.printStackTrace();  }  return date;  }  @DisplayName(value="等价类划分")  @ParameterizedTest  @CsvFileSource(resources = "/Bill1.csv",numLinesToSkip =1,encoding = "GB2312")  void parameters\_error\_test(String a,String e,String b, String c,double d, String remark) {  Bill bill = new Bill();  Date st=StrToDate(b);  Date et=StrToDate(c);  double fee=bill.get\_time(a,e,st,et);  assertEquals(d, fee);  }  }  测试结果：    覆盖率：    CheckStyle使用：    SpotBugs使用： |

### **结论分析与体会**

这次实验又是一个新的知识点——决策表，相较于其他测试方法，这个测试方法最为严密，逻辑最为严谨，在表格中将条件桩和动作桩一一列举之后使得其更加一目了然。正如上次实验所说，这次实验又涉及到了另一个插件spotbugs，它也是用于规范代码，使用比较简单，这里就不赘述了。

### **六、仓库地址**

https://github.com/cdwvcf/rjce1.git