**《软件测试》**

**实验报告五 ——基路径测试方法**

**姓 名： 吴志高 学 号： 2020105201**

**院 系：计算机与信息学院 专 业：计算机科学与技术**

**实 验 室： J-2301 实验日期： 2023/4/15**

**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

### 二、实验环境：

Idea2022，Gradle，Junit5

### 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，并把代码规范检查报告，jacoco测试覆盖率报告、spotbugs报告一同打包，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”

## **需求：电话账单计费**

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。

### **采用如下计费规则计算通话费：**

* 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。
* 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；
* 不到1分钟按1分钟计算

### **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：**

|  |  |  |  |
| --- | --- | --- | --- |
| **测试编号** | **测试用例** | **预测结果** | **实际结果** |
| **1** | *2022*,*12*,*25*,*10*,*12*,*12*,*2022*,*12*,*25*,*9*,*5*,*3* | *3.8元* | *3.8元* |
| **2** | *2019*,*12*,*25*,*16*,*4*,*5*,*2019*,*12*,*25*,*9*,*5*,*3* | *21.05元* | *21.05元* |
| **3** | *2018*,*12*,*25*,*16*,*4*,*5*,*2018*,*12*,*25*,*6*,*4*,*4* | *30.05元* | *30.05元* |

1. **代码实现：**

|  |
| --- |
| **实现代码：**  java/test3/fee.java package test3;  import java.text.DateFormat; import java.text.ParseException; import java.text.SimpleDateFormat; import java.util.Calendar; import java.util.Date;  public class Fee {   public static String classify(int aYear, int aMonth, int aDay,int aHour,int aMin,int aS,  int bYear, int bMonth, int bDay,int bHour,int bMin,int bS) throws ParseException {  DateFormat simpleFormat = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");  Date currentTime = simpleFormat.parse(aYear+"-"+aMonth+"-"+aDay+" "+aHour+":"+aMin+":"+aS); //当前系统时间  Date firstTime = simpleFormat.parse(bYear+"-"+bMonth+"-"+bDay+" "+bHour+":"+bMin+":"+bS);  String fee = String.*valueOf*(*getTime*(currentTime, firstTime))+"元"; return fee;  }  public static double getTime(Date currentTime, Date firstTime){  long diff = currentTime.getTime() - firstTime.getTime();//这样得到的差值是微秒级别  Calendar currentTimes =*dataToCalendar*(currentTime);//当前系统时间转Calendar类型  Calendar firstTimes =*dataToCalendar*(firstTime);//查询的数据时间转Calendar类型  int year = currentTimes.get(Calendar.*YEAR*) - firstTimes.get(Calendar.*YEAR*);//获取年  int month = currentTimes.get(Calendar.*MONTH*) - firstTimes.get(Calendar.*MONTH*);  int day = currentTimes.get(Calendar.*DAY\_OF\_MONTH*) - firstTimes.get(Calendar.*DAY\_OF\_MONTH*);  if (day < 0) {  month -= 1;  currentTimes.add(Calendar.*MONTH*, -1);  day = day + currentTimes.getActualMaximum(Calendar.*DAY\_OF\_MONTH*);//获取日  }  if (month < 0) {  month = (month + 12) % 12;//获取月  year--;  }  long days = diff / (1000 \* 60 \* 60 \* 24);  long hours = (diff-days\*(1000 \* 60 \* 60 \* 24))/(1000\* 60 \* 60); //获取时  long minutes = (diff-days\*(1000 \* 60 \* 60 \* 24)-hours\*(1000\* 60 \* 60))/(1000\* 60); //获取分钟  long s=(diff/1000-days\*24\*60\*60-hours\*60\*60-minutes\*60);//获取秒  String CountTime=""+"year"+"年"+month+"月"+day+"天"+hours+"小时"+minutes+"分"+s+"秒";    double fee;  double min=0;  int x=20\*60;  min=hours\*60+minutes+Math.*ceil*(s);  if(min<=x){  fee=0.05\*min;  }else{  fee=1+(min-x)\*0.1;  }  fee= Double.*parseDouble*(String.*format*("%.2f", fee));  return fee;  }   //Date类型转Calendar类型  public static Calendar dataToCalendar(Date date) {  Calendar calendar = Calendar.*getInstance*();  calendar.setTime(date);  return calendar;  } }  测试代码：  java/test3/feeTest.java  package test3;  import static org.junit.jupiter.api.Assertions.*assertEquals*;  import org.junit.jupiter.params.ParameterizedTest; import org.junit.jupiter.params.provider.CsvFileSource;  import java.text.ParseException;  class FeeTest {    @ParameterizedTest  @CsvFileSource(resources = "/t1.csv")  void t1(int aYear, int aMonth, int aDay,int aHour,int aMin,int aS,  int bYear, int bMonth, int bDay,int bHour,int bMin,int bS,  String expected) throws ParseException {  *assertEquals*(expected, Fee.*classify*(  aYear,aMonth,aDay,aHour,aMin,aS,  bYear,bMonth,bDay,bHour,bMin,bS));  }  }  测试结果：    覆盖率： |

### **五、结论分析与体会**

### 通过对决策表分析方法的实验，我深刻认识到这种方法在处理复杂的规则和条件方面的高效性和准确性。通过分析决策表，我能够快速识别出规则适用的条件，以及每个规则的结果。这种方法的优点是非常显然的，因为它不仅可以节省时间和精力，而且能够避免错误和失误。

### **六、仓库地址**

https://gitee.com/wu-zhi-gao/2020105201.git