**《软件测试》**

**实验报告四 ——决策表分析方法**

**姓 名： 黄勇斌 学 号：2019112321**

**院 系： 计算机与信息学院 专 业： 计算机科学与技术**

**实 验 室： 实验日期：2022/4/2**

**总评成绩： 审阅教师：**

目录

[**《软件测试》** 1](#_Toc99813008)

[**实验报告四 ——决策表分析方法** 1](#_Toc99813009)

[一、实验目的： 3](#_Toc99813010)

[二、实验环境： 3](#_Toc99813011)

[三、实验要求： 3](#_Toc99813012)

[四、实验步骤与内容 3](#_Toc99813013)

[五、结论分析与体会 9](#_Toc99813014)

[六、仓库地址 9](#_Toc99813015)

### 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：

Eclipse2020,JUnit

### 三、实验要求：

1. 打开[educoder](https://www.educoder.net/classrooms/19905?code=Q35M2),完成黑盒测试过关题。
2. 实现电话账单收费功能
3. 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
4. 使用checkstyle生成代码规范检查报告
5. 使用jacoco生成测试覆盖率报告
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”

### 四、实验步骤与内容

1. **题目：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

**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. **决策表：**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **规则** | 1 | 2 | 3 | 4 | 5 | 6 |
| **开始时间在夏令时** | Y | N | Y | N | - | - |
| **结束时间在夏令时** | N | Y | Y | N | - | - |
| **总共时长超过30分钟** | N | N | N | N | Y | - |
| **开始时间比结束时间大** | N | N | N | N | N | Y |
| **动作** |  |  |  |  |  |  |
| **时间加上60分钟，并输出费用** |  | R |  |  |  |  |
| **时间减去60分钟，并输出费用** | R |  |  |  |  |  |
| **时间不变，并输出费用** |  |  | R | R |  |  |
| **输出‘输入错误’** |  |  |  |  |  | R |
| **输出‘超过30小时’** |  |  |  |  | R |  |

1. **测试用例：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 开始时间 | 是否在夏令时 | 结束时间 | 是否在夏令时 | 预测输出 | 实际输出 |
| 20220321161205 | 0 | 20220321161105 | 0 | 输入错误 | 输入错误 |
| 20220321161205 | 0 | 20230321161205 | 0 | 超过30小时 | 超过30小时 |
| 20220321160905 | 1 | 20220321161206 | 0 | 17.40 | 17.40 |
| 20220321160905 | 0 | 20220321161206 | 1 | 5.40 | 5.40 |
| 20220321160905 | 1 | 20220321161206 | 1 | 11.40 | 11.40 |
| 20220321160905 | 0 | 20220321161206 | 0 | 11.40 | 11.40 |

1. **代码实现：**

|  |
| --- |
| **实现代码：**  **CalculateTime类：**  import java.time.LocalDateTime;  public class CalculateTime {  LocalDateTime startTime;  LocalDateTime endTime;  int inDayLightSavingOfStartTime;  int inDayLightSavingOfEndTime;  long Millis;  long Minutes;  public CalculateTime(LocalDateTime startTime, int inDayLightSavingOfStartTime, LocalDateTime endTime, int inDayLightSavingOfEndTime){  this.startTime = startTime;  this.endTime = endTime;  this.inDayLightSavingOfEndTime = inDayLightSavingOfEndTime;  this.inDayLightSavingOfStartTime = inDayLightSavingOfStartTime;  }  public void calculateMinutes(){  if(Millis %(60\*1000) == 0){  this.Minutes = Millis/(60\*1000);  }  else {  this.Minutes = Millis /(60\*1000)+1;  }  }  long getTime(){  java.time.Duration duration = java.time.Duration.between(startTime, endTime );  if(inDayLightSavingOfStartTime == 0 && inDayLightSavingOfEndTime == 1){  this.Millis = duration.toMillis()-60\*60\*1000;  }  else if(inDayLightSavingOfStartTime == 1 && inDayLightSavingOfEndTime == 0)  this.Millis = duration.toMillis()+60\*60\*1000;  else  this.Millis = duration.toMillis();  calculateMinutes();  return this.Minutes;  }  public long getMillis() {  return Millis;  }  }  **CalculateBill类：**  public class CalculateBill {  long Minutes;  double bill;  public CalculateBill(long Minutes){  this.Minutes = Minutes;  calculateBill();  }  public void calculateBill(){  if(Minutes <= 20){  bill = 0.05\*Minutes;  }  else{  bill = 1 + (Minutes-20)\*0.1;  }  }  public double getBill(){  return bill;  }  }  **Solve类：**  import java.time.LocalDateTime;  public class Solve {  String startTime;  int inDayLightSavingOfStartTime;  String endTime;  int inDayLightSavingOfEndTime;  LocalDateTime localDateTime1;  LocalDateTime localDateTime2;  public Solve(String startTime, int inDayLightSavingOfStartTime, String endTime, int inDayLightSavingOfEndTime) {  this.startTime = startTime;  this.endTime = endTime;  this.inDayLightSavingOfEndTime = inDayLightSavingOfEndTime;  this.inDayLightSavingOfStartTime = inDayLightSavingOfStartTime;  }  public String getResults() {  if(startTime.compareTo(endTime) > 0){  return "输入错误";  }  else{  this.localDateTime1 = LocalDateTime.of(  Integer.parseInt(startTime.substring(0, 4)), //2022  Integer.parseInt(startTime.substring(4, 6)), //04  Integer.parseInt(startTime.substring(6, 8)),//02  Integer.parseInt(startTime.substring(8, 10)),//15  Integer.parseInt(startTime.substring(10, 12)),//57  Integer.parseInt(startTime.substring(12, 14))//32  );  }  this.localDateTime2 = LocalDateTime.of(  Integer.parseInt(endTime.substring(0, 4)), //2022  Integer.parseInt(endTime.substring(4, 6)), //04  Integer.parseInt(endTime.substring(6, 8)),//02  Integer.parseInt(endTime.substring(8, 10)),//15  Integer.parseInt(endTime.substring(10, 12)),//57  Integer.parseInt(endTime.substring(12, 14))//32  );  CalculateTime calculateTime = new CalculateTime(localDateTime1, inDayLightSavingOfStartTime, localDateTime2, inDayLightSavingOfEndTime);  if(calculateTime.getTime() > 60\*30)  return "超过30小时";  CalculateBill calculateBill = new CalculateBill(calculateTime.getTime());  //return String.valueOf(localDateTime1);  //return String.valueOf(calculateTime.getTime());  return String.format("%.2f", calculateBill.getBill());  }  }  **测试代码：**  import org.junit.jupiter.params.ParameterizedTest;  import org.junit.jupiter.params.provider.CsvSource;  import java.time.LocalDateTime;  import static org.junit.jupiter.api.Assertions.\*;  class mainTest {  Solve solve;  @ParameterizedTest  @CsvSource({  "20220321161205,0,20220321161105,0,输入错误",  "20220321161205,0,20230321161205,0,超过30小时",  "20220321140905,1,20220321161206,0,17.40",  "20220321140905,0,20220321161206,1,5.40",  "20220321140905,1,20220321161206,1,11.40",  "20220321140905,0,20220321161206,0,11.40"  })  //输入year-month-day-minute-seconds 组成的字符串  void test(String startTime, int inDayLightSavingOfStartTime, String endTime, int inDayLightSavingOfEndTime, String results){  solve = new Solve(startTime, inDayLightSavingOfStartTime, endTime, inDayLightSavingOfEndTime);  String result = solve.getResults();  assertEquals(results, solve.getResults());  }  }  测试结果：    jacoco报告  checkstyle报告 |

### 五、结论分析与体会

这次深刻的了解了关于决策表的知识，但是在关于之前的实验内容发现有所遗忘。

### 六、仓库地址

<https://github.com/hyb1041739742/Software-Testing>