**《软件测试》**

**实验报告三 ——等价类分析方法**

**姓 名： 曹倜源 学 号： 2020112805**

**院 系： 计算机与信息学院 专 业：计算机科学与技术**

**实 验 室： J2301 实验日期： 2023/3/22**

**总评成绩： 审阅教师： 杨青**

### 一、实验目的：

巩固 git 协作式管理工具的使用；

巩固 gradle 项目构建 Java 应用的基本操作

掌握等价类分析方法设计测试用例的方法

### 二、实验环境：

IntelliJ IDEA2020,JUnit

### 三、实验要求：

1. 实现电话账单收费功能
2. 采用边等价类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. 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-软件测试实验三”命名，提交到雨课堂“软件测试实验三”

### 四、实验步骤与内容

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。

**代码实现：**

|  |
| --- |
| **实现代码：**  package cn.edu.ctgu;  import java.time.LocalDate;  import java.time.LocalDateTime;  public class LongDistanceCallService {  public double calculateCallCost(int callDurationSeconds, LocalDateTime callStartTime) {  double cost = 0.0;  // Calculate call duration in minutes, rounded up  int callDurationMinutes = (int) Math.ceil(callDurationSeconds / 60.0);  // Adjust call start time for daylight saving time  if (isDuringDaylightSavingTime(callStartTime)) {  callStartTime = callStartTime.plusHours(1);  }  // Apply call rates based on call duration  if (callDurationMinutes <= 0) {  cost = 0.0;  } else if (callDurationMinutes <= 20) {  cost = callDurationMinutes \* 0.05;  } else {  cost = 1.0 + (callDurationMinutes - 20) \* 0.1;  }  return cost;  }  private boolean isDuringDaylightSavingTime(LocalDateTime dateTime) {  // Check if date is between March and November (inclusive)  if (dateTime.getMonthValue() < 3 || dateTime.getMonthValue() > 11) {  return false;  }  // Check if date is in March or November  if (dateTime.getMonthValue() == 3 || dateTime.getMonthValue() == 11) {  // Check if date is on or after the second Sunday of the month  LocalDate date = dateTime.toLocalDate();  int dayOfWeek = date.getDayOfWeek().getValue();  int dayOfMonth = date.getDayOfMonth();  if (dateTime.getMonthValue() == 3) {  return (dayOfMonth >= 8 && dayOfWeek == 7) || dayOfMonth >= 15;  } else {  return (dayOfMonth >= 1 && dayOfWeek == 7) || (dayOfMonth >= 8 && dayOfMonth <= 14);  }  }  // Date is between April and October (inclusive)  return true;  }  }  测试代码：  package cn.edu.ctgu;  import static org.junit.jupiter.api.Assertions.\*;  import static org.junit.jupiter.api.Assertions.assertEquals;  import java.time.LocalDateTime;  import org.junit.jupiter.api.Test;  class LongDistanceCallServiceTest {  private LongDistanceCallService service = new LongDistanceCallService();  @Test  public void testZeroDurationCall() {  double cost = service.calculateCallCost(0, LocalDateTime.of(2023, 3, 26, 10, 0));  assertEquals(0.0, cost, 0.001);  }  @Test  public void testLessThanOneMinuteCall() {  double cost = service.calculateCallCost(30, LocalDateTime.of(2023, 3, 26, 10, 0));  assertEquals(0.05, cost, 0.001);  }  @Test  public void testShortDurationCall() {  double cost = service.calculateCallCost(10 \* 60, LocalDateTime.of(2023, 3, 26, 10, 0));  assertEquals(0.5, cost, 0.001);  }  @Test  public void testLongDurationCall() {  double cost = service.calculateCallCost(30 \* 60, LocalDateTime.of(2023, 3, 26, 10, 0));  assertEquals(2.0, cost, 0.001);  }  @Test  public void testDaylightSavingTimeStart() {  double cost = service.calculateCallCost(10 \* 60, LocalDateTime.of(2023, 3, 12, 1, 0));  assertEquals(0.5, cost, 0.001);  }  @Test  public void testStandardTime() {  double cost = service.calculateCallCost(10 \* 60, LocalDateTime.of(2023, 11, 1, 10, 0));  assertEquals(0.5, cost, 0.001);  }  }测试结果：    覆盖率： |
|  |

### 五、结论分析与体会

### 六、仓库地址

https://gitee.com/zhuyinl/software-test