**Lab3 实验报告**

测试用例定义过程

1. CalendarInterval类测试用例编写过程中，我深刻认识到测试用例本身的意义。它的测试不止要涵盖寻常的取值，还要涵盖边缘化的取值，以试探功能的边界。不仅仅要测试运行正确的情况，还要测试方法是否能正确抛出异常。因此我定义了如下测试用例：

public void testIntervalException(){

GregorianCalendar rightStartDate=new GregorianCalendar(2018, 4, 14, 15, 47);

GregorianCalendar wrongStartDate=new GregorianCalendar(2019, 5, 4, 12, 3);

GregorianCalendar rightEndDate=new GregorianCalendar(2018, 5, 4, 12, 3);

GregorianCalendar wrongEndDate=new GregorianCalendar(2017, 5, 4, 12, 3);

try {

CalendarInterval interval = new CalendarInterval(rightStartDate, wrongEndDate);

fail("Constructor is not able to handle IntervalException");

}catch (Exception ex){

assertTrue(ex instanceof IntervalException);

}

try {

CalendarInterval interval = new CalendarInterval(rightStartDate, rightEndDate);

interval.setEndTime(wrongEndDate);

fail("EndTime setter is not able to handle IntervalException");

}catch (Exception ex){

assertTrue(ex instanceof IntervalException);

}

try {

CalendarInterval interval = new CalendarInterval(rightStartDate, rightEndDate);

interval.setStartTime(wrongStartDate);

fail("StartTime setter is not able to handle IntervalException");

}catch (Exception ex){

assertTrue(ex instanceof IntervalException);

}

}

用于测试类中的方法是否能正确的抛出异常，正确地处理异常行为。而这一类测试非常重要，是功能安全性的重要保障。

2.在TodoList的测试用例编写时，对于任意输入时正确输出的检测显出其重要性。对于一些不合理的输入，抛出异常过于暴力。比如需要删除的项目不在列表中。此时就应该用空输出来代替。因此对于空输出的测试也是非常重要的。在本程序中，search的空输出我用空ArrayList表示（而非null），这样更具有机动性。

CalendarInterval searchInterval=new CalendarInterval(new GregorianCalendar(2018, 4, 13), new GregorianCalendar(2018, 4, 15));

TodoList list=new TodoList();

Todo todo=new Todo("Get up", new CalendarDate("2018-4-14"));

assertTrue(list.search(searchInterval).isEmpty());

我用如上代码对此进行测试。

而对于删除方法，我增添了返回值，以返回是否存在这一删除项的信息。在todo未添加入list的情况下，我增添了如下断言，以对删除行为进行判断。

assertFalse(list.delete(todo));

实现与测试过程

1. 对isOverlapped(date)方法的实验时，尝试直接用GregorianCalendar的before与after方法，test失败。

后来意识到date本身也是一个时间区间，应为它定义相应的时间区间，将代码构建为如下形态。

public boolean isOverlapped(CalendarDate date){

GregorianCalendar dateStartTime=new GregorianCalendar(date.getYear(), date.getMonth(), date.getDay(), 0, 0);

GregorianCalendar dateEndTime=new GregorianCalendar(date.getYear(), date.getMonth(), date.getDay(), 24, 0);

try {

CalendarInterval interval=new CalendarInterval(dateStartTime, dateEndTime);

return isOverlapped(interval);

} catch (IntervalException e) {

System.out.println(e.getMessage());;

}

return false;

}

成功通过测试。

2. 测试中关于构造函数处理IntervalException的单元测试未过。发现原因是DateUtil中相关的isValidInterval函数未实现，实现后问题解决。

public static boolean isValidInterval(GregorianCalendar start, GregorianCalendar end){

if(start.after(end))

return false;

return true;

}

3. 对TodoList的测试过程中，对设计和测试用例进行了微调，使代码更加清晰。添加了deleteInSingle，减少了重复代码。

4. 初次尝试TodoList的根据日期搜索功能失败，发现在用毫秒数作为HashMap键值时，0点所对应的毫秒数并不被86400000（一天的毫秒数）整除。因此在转换时需要有一个偏移量（计算知为57600000）。问题迎刃而解。公式如下

(ms-57600000L)/86400000L

5. 初次尝试时未能一次将整个时间段待办事项删除。发现是因为使用了HashMap数据结构之后，单个待办事项的引用可分布在多个键（我以日期为键）中，因此要将所有的引用都remove需要一个循环。增加了必要的循环后，功能完成。实现如下

boolean result=false;

long startKey=(todo.getInterval().getStartTime().getTimeInMillis()-57600000L)/86400000L;

long endKey=(todo.getInterval().getEndTime().getTimeInMillis()-57600000L)/86400000L;

for(long i=startKey; i<=endKey; i++){

result=deleteInArray(todoList.get(i), todo);

}

return result;