

**2019年春季学期  
计算机学院《软件构造》课程**

**Lab 4实验报告**

|  |  |
| --- | --- |
| 姓名 | 李震宇 |
| 学号 | 1170300110 |
| 班号 | 1703001 |
| 电子邮件 | 1137859144@qq.com |
| 手机号码 | 18800419432 |

**目录**

[1 实验目标概述 1](#_Toc3922818)

[2 实验环境配置 1](#_Toc3922819)

[3 实验过程 1](#_Toc3922820)

[3.1 Error and Exception Handling 2](#_Toc3922821)

[3.2 Assertion and Defensive Programming 2](#_Toc3922822)

[3.2.1 checkRep()检查invariants 2](#_Toc3922823)

[3.2.2 Assertion保障pre-/post-condition 2](#_Toc3922824)

[3.3 Logging 2](#_Toc3922825)

[3.3.1 写日志 2](#_Toc3922826)

[3.3.2 日志查询 2](#_Toc3922827)

[3.4 Testing for Robustness and Correctness 2](#_Toc3922828)

[3.4.1 Testing strategy 2](#_Toc3922829)

[3.4.2 测试用例设计 2](#_Toc3922830)

[3.4.3 测试运行结果与EclEmma覆盖度报告 2](#_Toc3922831)

[3.5 SpotBugs tool 2](#_Toc3922832)

[3.6 Debugging 3](#_Toc3922833)

[3.6.1 理解待调试程序的代码思想 3](#_Toc3922834)

[3.6.2 发现并定位错误的过程 3](#_Toc3922835)

[3.6.3 如何修正错误 3](#_Toc3922836)

[3.6.4 结果 3](#_Toc3922837)

[4 实验进度记录 3](#_Toc3922838)

[5 实验过程中遇到的困难与解决途径 3](#_Toc3922839)

[6 实验过程中收获的经验、教训、感想 4](#_Toc3922840)

[6.1 实验过程中收获的经验和教训 4](#_Toc3922841)

[6.2 针对以下方面的感受 4](#_Toc3922842)

# 实验目标概述

本次实验重点训练学生面向健壮性和正确性的编程技能，利用错误和异常处 理、断言与防御式编程技术、日志/断点等调试技术、黑盒测试编程技术，使程序 可在不同的健壮性/正确性需求下能恰当的处理各种例外与错误情况，在出错后 可优雅的退出或继续执行，发现错误之后可有效的定位错误并做出修改。 实验针对 Lab 3 中写好的 ADT 代码和基于该 ADT 的三个应用的代码，使用 以下技术进行改造，提高其健壮性和正确性：

 错误处理

 异常处理

 Assertion 和防御式编程

 日志

 调试技术

 黑盒测试及代码覆盖度

# 实验环境配置

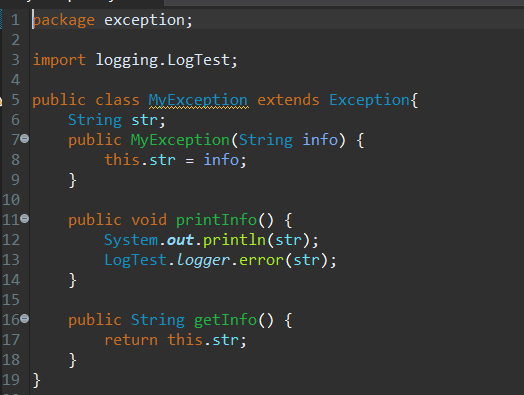
java + eclipse + log4j + spotBugs

<https://classroom.github.com/a/mnFCZ30Y>

# 实验过程

## Error and Exception Handling

### 编写异常类



其中主要是打印异常信息，然后写入日志文件和提取异常信息三个方法。

在创建异常对象时，通过构造器设定异常信息。

### 不符合语法规则异常

TrackGame：输入运动员信息顺序错误(两种)、输入小数位数错误、输入运动员信息却是。

AtomicStructure：输入元素名错误(两种)、输入轨道时使用小数错误、输入电子信息时使用小数错误。

SocialNetworkCircle：输入年龄使用小数、输入性别错误、输入亲密度小数位数错误。

### **存在标签相同的输入**

TrackGame：输入运动员名称相同。

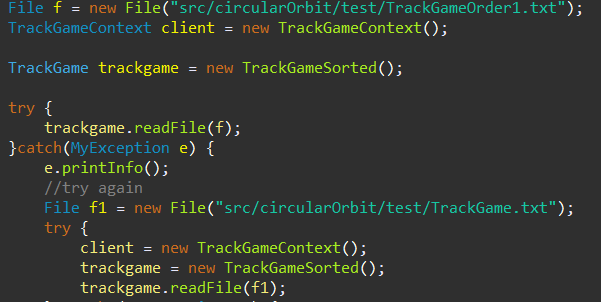
SocialNetworkCircle：输入了出发、指向相同的关系

### 依赖关系不正确

AtomicStructure：电子信息输入和轨道信息不匹配。

SocialNetworkCircle：没有创建Friend信息就使用了SocialTie

### 效果：





## Assertion and Defensive Programming

### checkRep()检查invariants

TrackGame：

使用Map构建Track-Objects的轨道映射，所以不管如何修改比赛方案，每组比赛的人数都不会超过轨道总数，因为Track数已经是确定的了，无法向不存在的轨道上添加物体。

AtomicStructure:

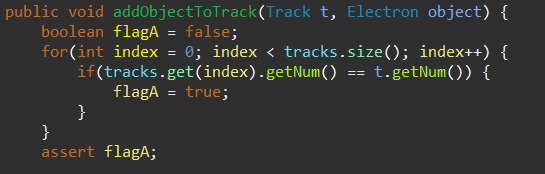
没什么RI

SocialNetwork:

每次添加或删除Tie的时候都会重新构建这个轨道系统，故可以保证不管社交关系如何增加或删除，第 i 层 轨道上的人与中心点的人之间的最短路径等于 i。

### Assertion保障pre-/post-condition

例如：



在每次调用方法时都对pre-condition进行检查，确保参数符合轨道系统的要求。对每

个新增、删除的函数都进行了检查，因为涉及到几乎每一个函数，因此就不一一截图了。

而且对于GUI应用，我已经做了足够的防御性工作，在每次调用函数之前都会判断这个输入的参数是否符合要求，如果不符合就会弹出Error对话框，因为每次调用都会检查，所以保证了在应用的使用过程中不会因为pre-condition导致程序结束，举一个例子：

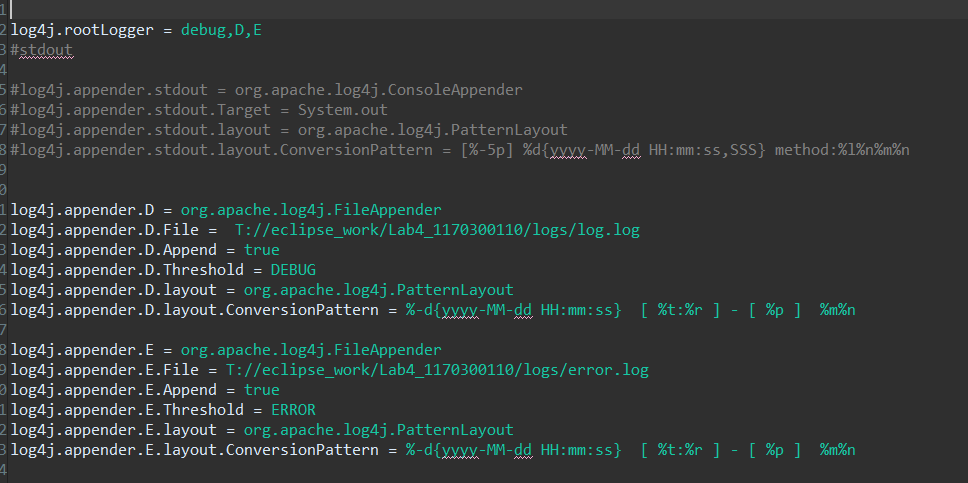


## Logging

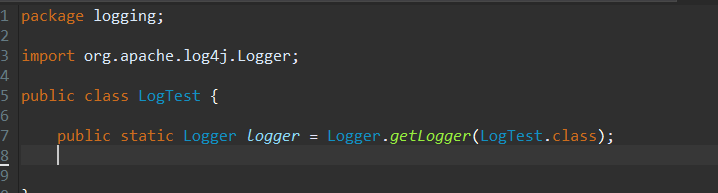
### 写日志

使用log4j写日志，选择将error类型的结果单独存储到一个文件中去。

log4j配置文件：



函数实现：



当需要写入日志时就可以调用

LogTest.logger.info(“xxxxx”);

或 LogTest.logger.error(“xxxx”);

完成日志的写入。

效果：



### 日志查询

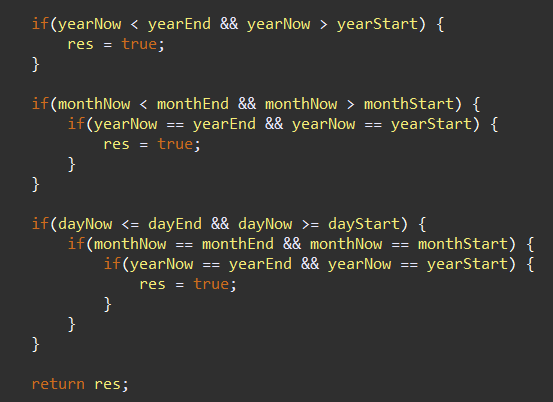
实现了三个不同的查询方式：按时间、按类型、按操作。

按时间：

输入指定时间范围，按行读取日志文件，提取处日志操作时间，判断是否在时间段内，

如果在，那么将这行操作日志打印出来。

判定是否在时间区间内的函数关键部分：



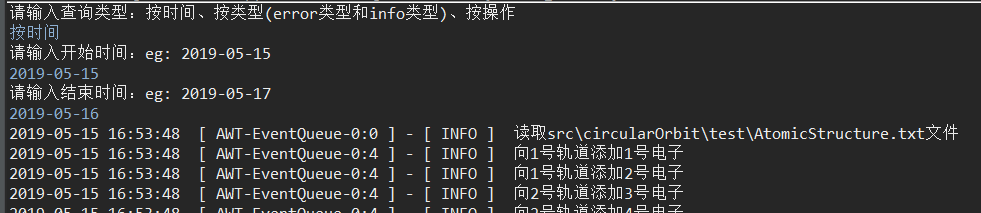
按类型：

按行读入日志文件，如果该行含有INFO则是info类型，如果改行含有ERROR则是error类型。按照用户需求打印即可。

按操作类型：

按行读入日志文件，如果改行有添加、新增等字眼，则是添加操作，如果有删除等字眼就是删除操作，按用户需求打印对应的日志。

效果：



## Testing for Robustness and Correctness

### Testing strategy

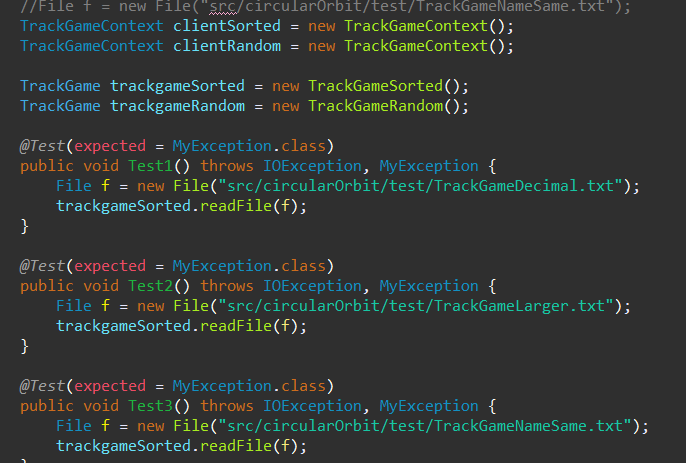
输入含有不同的输入异常的文件路径，确定抛出正确的异常提示。

进行输入正确的测试。

进行输入错误的测试。并确定程序抛出指定的AssertionError.class

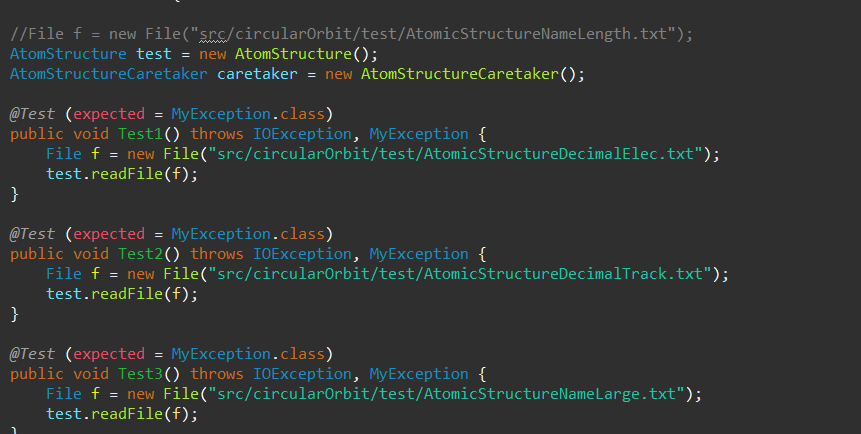
### 测试用例设计

TrackGame:



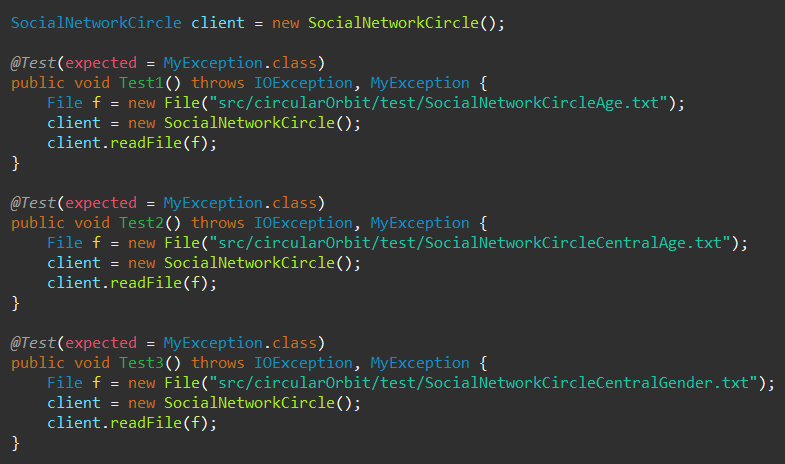


AtomicStructure:



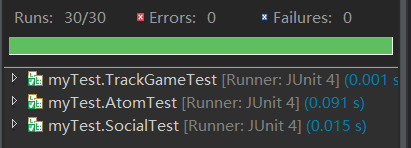


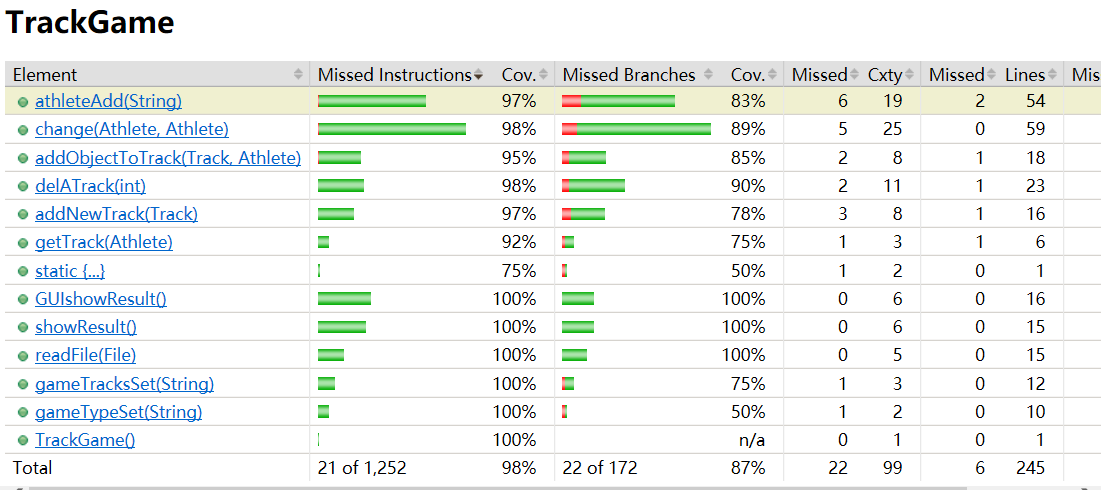
SocialNetwork:

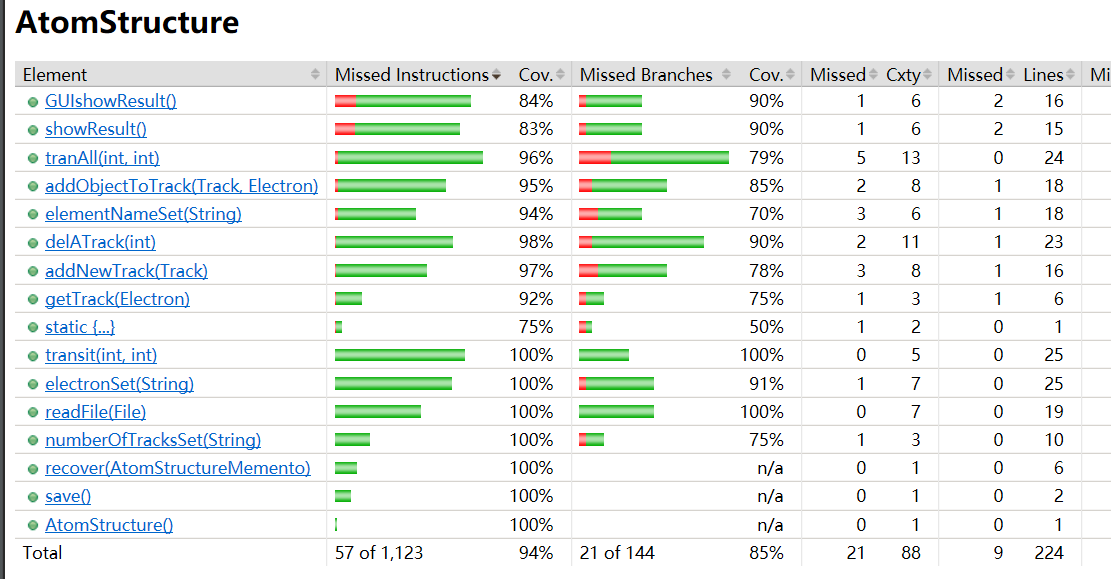


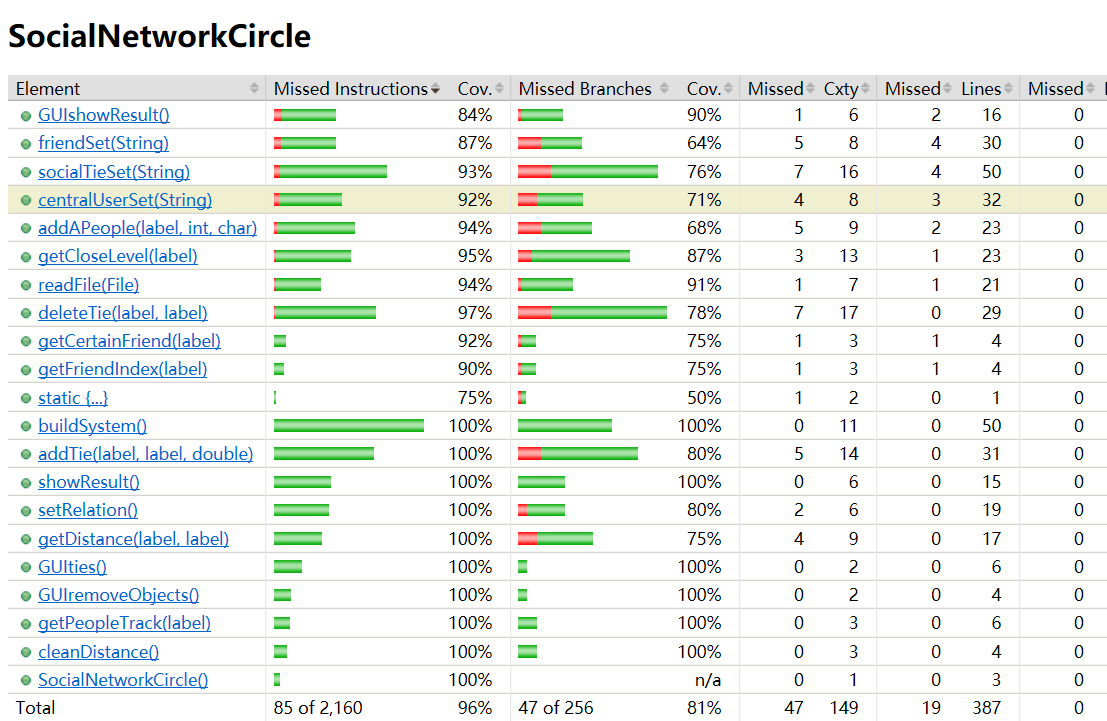


### 测试运行结果与EclEmma覆盖度报告









## SpotBugs tool

发现的错误都是：

重写的equals方法没有考虑到传入null的情况。

重写了equals方法却没有重写hashcode方法

提醒我equals方法和hashcode方法是紧密相连的，要同时写。

## Debugging

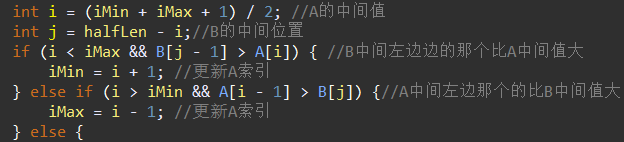
### 理解待调试程序的代码思想

FindMedianSortedArrays:

寻找两个数组AB合在一起的中位数，如果合在一起是偶数长度，则找到中间两位的值，计算出中位数等于(left+right)/2

如果合在一起是奇数长度，则找到中间位置值返回。

寻找方式：







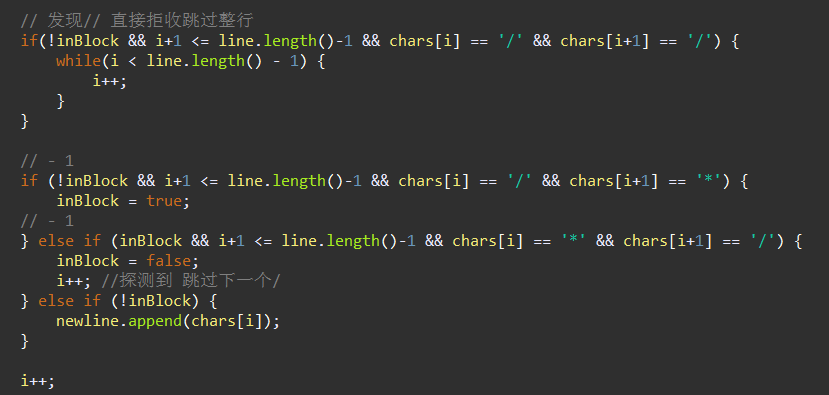
偶数：

奇数：

RemoveComments:

删除注释的算法。isblock是屏蔽模式，当为false时读入，当为true时屏蔽输入，当读入扫描到/\*时开始isblock = true开始屏蔽输入，直到扫描到\*/时关闭isblock，正常读入。

当扫描到//时，开始屏蔽输入，跳过整行，直接进入下一行。



TopVotedCandidate:

在线选举算法。

首先我们可以在一次扫描计算出每个时间点 time[i] 的当选者。因为每次票数只会增加 1，所以我们可以根据上一个时间点的当选者来推出当前时间的当选者。

然后每询问一个时间，二分查找小于等于查询时间的时间点，返回该时间点的当选者。

### 发现并定位错误的过程

使用java自带的debug功能，单步调试，寻找和预期不相同的位置，错误就出现在那

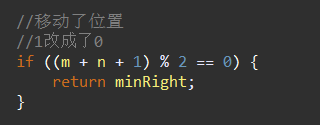
里，然后进行修改，然后测试，若通过，更换测试用例，如果发现有错误，就再重复debug过程寻找错误。

### 如何修正错误

这些错误一般都出现在奇偶条件判断，边界值问题，二分法上下界更换上。

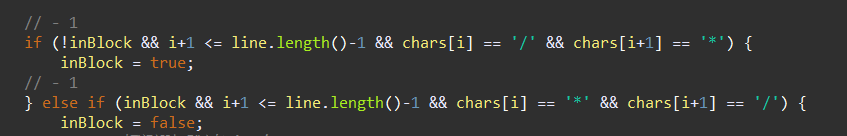
FindMedianSortedArrays:

奇偶判断错误

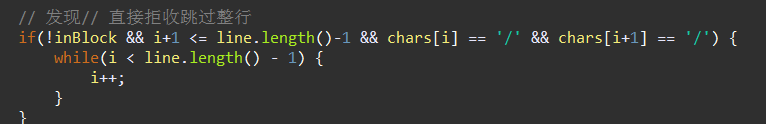


RemoveComments:

判断时出现了数组越界

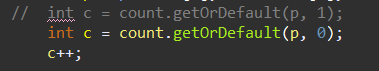


缺少一种情况

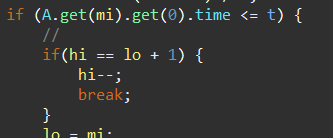


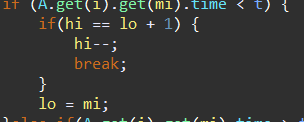
TopVotedCandidate:

如果没有这个人，票数从零记录，然后c++代表获得一票。



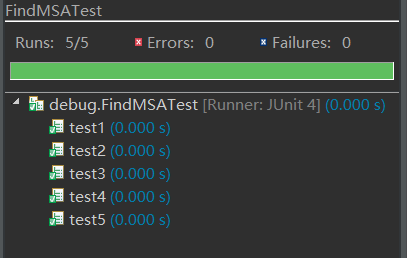
二分法边界条件出现错误：

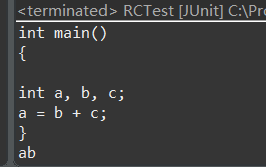


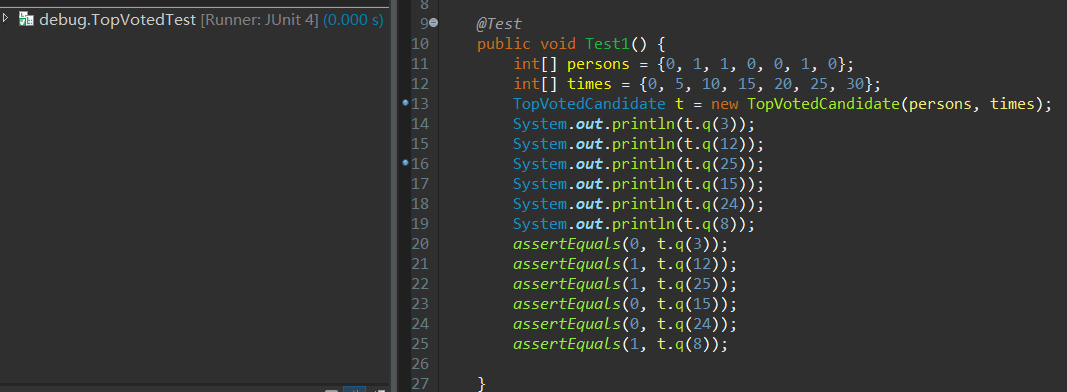


### 结果

依次通过了测试样例。







# 实验进度记录

请使用表格方式记录你的进度情况，以超过半小时的连续编程时间为一行。

每次结束编程时，请向该表格中增加一行。不要事后胡乱填写。

不要嫌烦，该表格可帮助你汇总你在每个任务上付出的时间和精力，发现自己不擅长的任务，后续有意识的弥补。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 2019.5.13 | 下午+晚上 | 完成3.1 | 成功 |
| 2019.5.14 | 晚上 | 完成3.2 | 失败 |
| 2019.5.15 | 晚上 | 完成3.2 | 成功 |
| 2019.5.16 | 下午+晚上 | 完成3.3 | 成功 |
| 2019.5.17 | 下午+晚上 | 完成3.4+3.5 | 成功 |
| 2019.5.18 | 下午+晚上 | 完成3.6 | 成功 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 3.6Debug一开始读不懂题，搞不明白算法是什么意思 | 百度搜索到了相对应的题，读懂了算法 |

# 实验过程中收获的经验、教训、感想

## 实验过程中收获的经验和教训

这个还好没踩太多坑，主要是学习了异常的定义方式处理方式，如何和用户进行更好的交互。充分理解了程序的健壮性的重要意义。

## 针对以下方面的感受

1. 健壮性和正确性，二者对编程中程序员的思路有什么不同的影响？
2. 为了应对1%可能出现的错误或异常，需要增加很多行的代码，这是否划算？（考虑这个反例：民航飞机上为何不安装降落伞？）
3. “让自己的程序能应对更多的异常情况”和“让客户端/程序的用户承担确保正确性的职责”，二者有什么差异？你在哪些编程场景下会考虑遵循前者、在哪些场景下考虑遵循后者？
4. 过分谨慎的“防御”（excessively defensive）真的有必要吗？如果你在完成Lab5的时候发现Lab5追求的是I/O大文件时的性能（时间/空间），你是否会回过头来修改你在Lab3和本实验里所做的各类defensive措施？如何在二者之间取得平衡？
5. 通过调试发现并定位错误，你自己的编程经历中有总结出一些有效的方法吗？请分享之。Assertion和log技术是否会帮助你更有效的定位错误？
6. 怎么才是“充分的测试”？代码覆盖度100%是否就意味着100%充分的测试？
7. Debug一个错误的程序，有乐趣吗？
8. 关于本实验的工作量、难度、deadline。
9. 到目前为止你对《软件构造》课程的评价和建议。
10. 期末考试临近，你对占成绩60%的闭卷考试有什么期望或建议？//请严肃的提出，杜绝开玩笑，教师会认真考虑你们的建议。

1.健壮性是用户输入有问题时能够及时提醒，并接受第二次输入，而正确性是程序必须正确，为程序员提供的方法，程序员自己调用，必须满足正确性。

2.emm，我感觉有些没必要。

3.让自己的程序能够应对更多的异常情况主要是在用户可能输入许许多多奇怪的输入的情况下吧，如果用户明确程序目的，懂得程序的使用方法，那么就可以让程序的用户承担正确性的职责。

4.我觉着没必要。不会修改。取得平衡emm，牺牲防御性提高效率吧。

5.DEBUG单步调试我觉着可以应付大多数情况了。我觉得不会，因为Assertion直接结束了，并不知道问题出现在哪里，log也是，有可能bug出现在好几行代码前，无法准确定位。

6.等价类思想吧。尽可能多的不同等价类的情况。我认为不意味着充分的测试

7.没有乐趣，非常抓狂。

8.工作量可以接受，难度可以接受，deadline也还好。主要debug那里如果不百度很难明白他要干什么。

9.都还好，希望汉化ppt。再一个讲的有点快，点多而细，很难掌握全面。

10.希望少考一些边边角角的细节问题，多一些硬核？emm就是那种很核心的问题，因为这个课的知识点太多太杂，真的很难全记住，而且英文的翻译也有误差，感觉很费劲。