

**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 | 罗瑞欣 |
| 学号 | 1170300821 |
| 班号 | 1703008 |
| 电子邮件 | [3102595709@qq.com](mailto:3102595709@qq.com) |
| 手机号码 | 15048260039 |

**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

IDEA+maven

Maven 安装在C盘上始终无法建好环境变量，改换E盘后解决

在这里给出你的GitHub Lab4仓库的URL地址（Lab4-学号）。

git@github.com:ComputerScienceHIT/Lab4-1170300821.git

# 实验过程

请仔细对照实验手册，针对每一项任务，在下面各节中记录你的实验过程、阐述你的设计思路和问题求解思路，可辅之以示意图或关键源代码加以说明（但千万不要把你的源代码全部粘贴过来！）。

## Error and Exception Handling

异常类的设计想法主要为以下几点

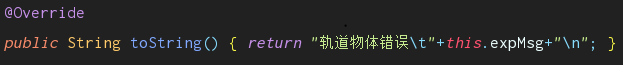
1. 在package：Exception中定义三个异常类：ObjectExp、ReadFileExp和RelationExp，分别对应轨道元素、读文件和依赖关系三种类型的错误。三个类的内部基本相似，如下：
   1. 异常信息，默认为MyException



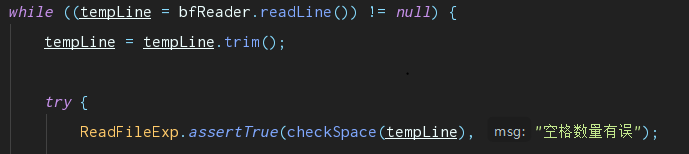
* 1. getter、setter、constructer
  2. 

其中cond在使用该异常类的类内部实现

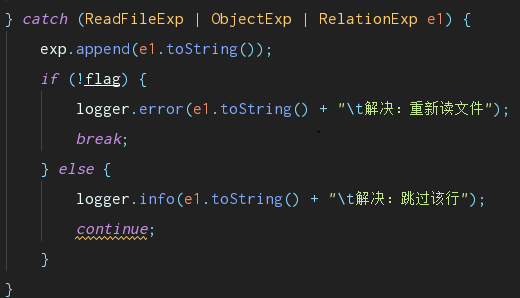
* 1. 输出信息：类对应的错误类型+具体错误信息



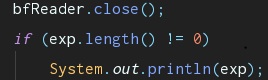
1. 在调用异常类的类内部：
   1. 在读文件方法中，将自定义异常类的try-catch放在按行读文件的循环中，以保证在某些不太严重的错误发生时，读文件的操作可以继续进行



* 1. 在catch部分可以按照两种情况处理：错误不严重时跳过该行；错误严重，涉及中心物体、或者对整体结构有影响时，break并返回false，提示重新读文件



* 1. 在读文件结束后，将记录在StringBuilder里面的异常信息在控制台输出



* 1. 上述Cond判断，在使用MyExp的类中定义不同的方法checkXXX，例如



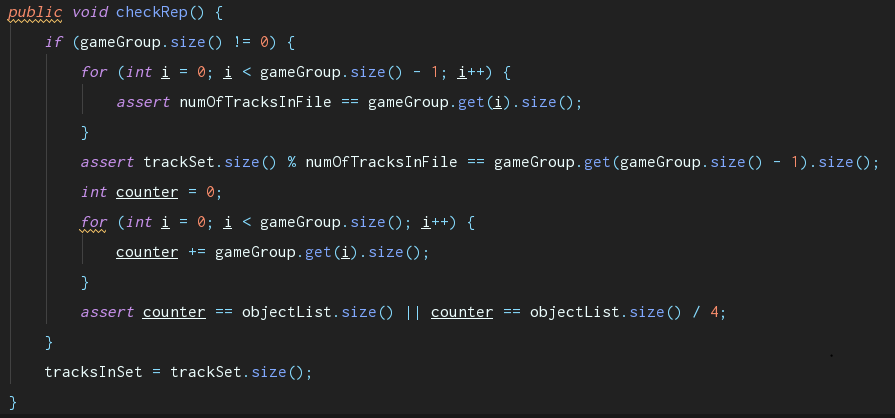




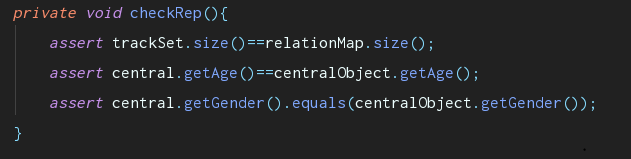
## Assertion and Defensive Programming

### checkRep()检查invariants

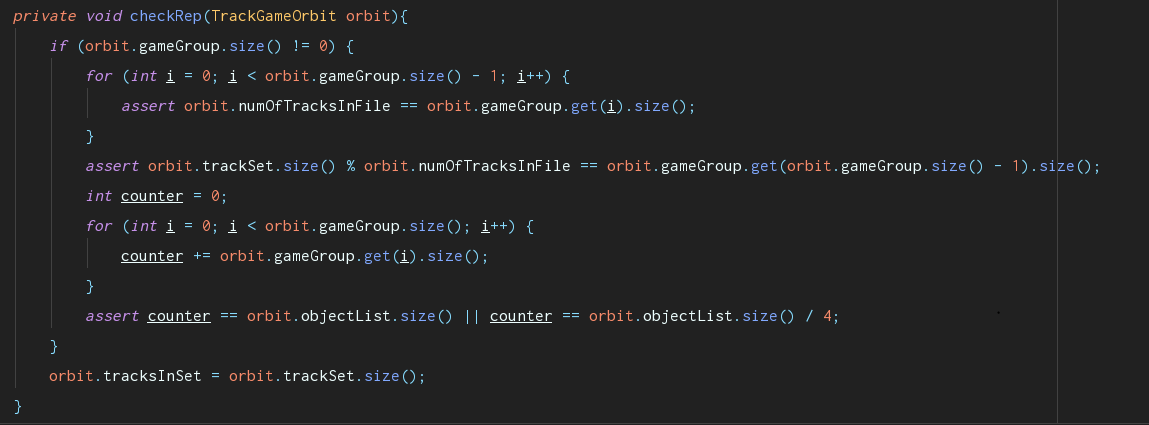
TrackGameOrbit



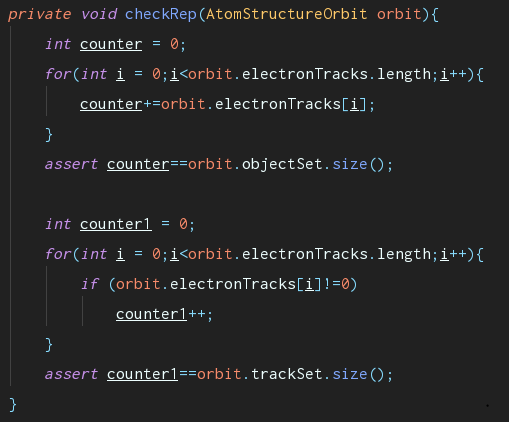
SocialNetworkCircleOrbit



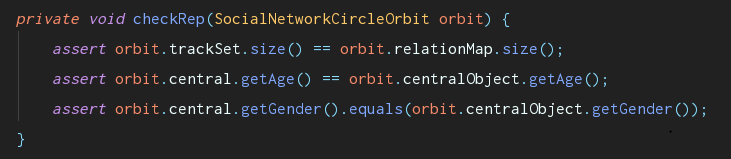
TrackGameAPI



AtomStructureAPI



SocialNetworkCircleAPI



### Assertion保障pre-/post-condition

Assert主要在各个方法的输入和返回时检查输入值和返回值，例如

1.在检查输入的对象是否为空，数值是否在合理的范围之内





2.检查返回的对象是否为空，数值是否合乎要求





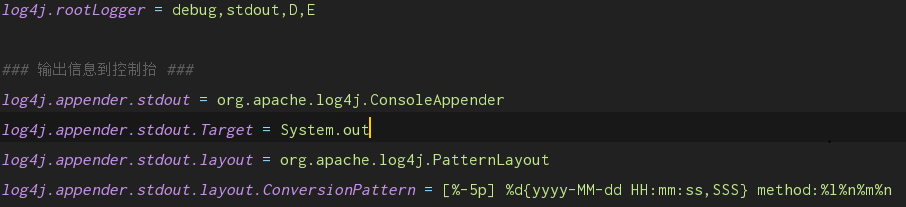
## Logging

在同学的建议下，使用了log4j

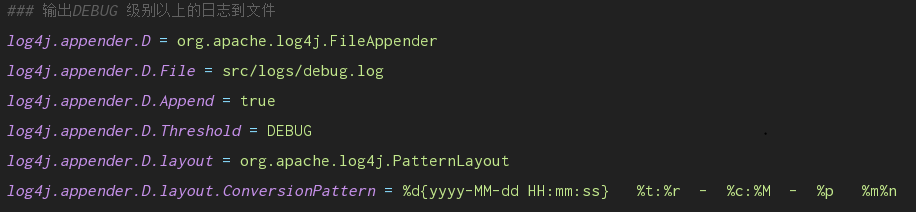
### 写日志

使用log4j写日志，主要包括配置文件和类内部具体调用两方面

1. 配置文件
   1. 控制台输出：在控制台打印等级+时间+发生位置、行数+具体异常信息

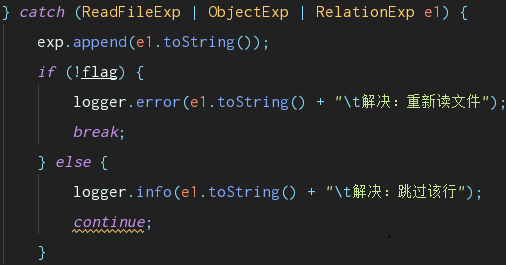


* 1. 写debug文件：
     1. 每天使用同一个文件
     2. 文件位置为src/logs/debug.log
     3. 继续上一行书写
     4. 书写内容：时间+线程+耗时+类+方法+优先级+具体信息

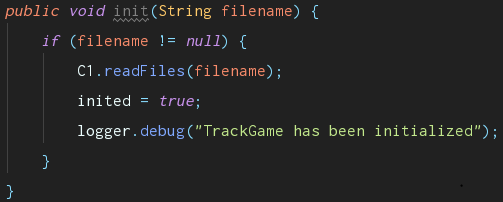


* 1. 写error文件
     1. 每天使用同一个文件
     2. 文件位置为src/logs/debug.log
     3. 继续上一行书写
     4. 书写内容：时间+线程+耗时+类+方法+优先级+具体信息

1. 具体调用：
   1. 记录异常：可跳过的异常设为INFO，将错误信息+“跳过该行”写入文件；严重异常设为ERROR，错误信息+“重读文件”写入文件



* 1. 记录操作：在application中的类中，每正常进行一项操作，将操作信息设为DEBUG写入文件



### 日志查询

定义查询log的类SelectLog（在otherAPIs中），使用正则表达式读文件筛选

正则表达式：

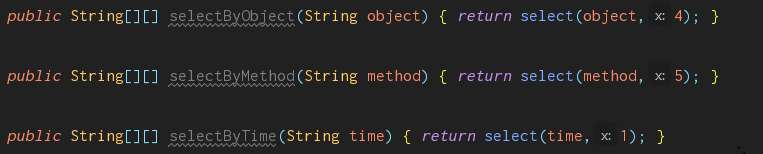


分别读取：日期+时间+线程+耗时+类+方法+等级+具体信息

方法select根据匹配信息（item）和匹配条目（int x对应match的group），被具体的筛选方法调用，对时间、方法、类、等级等条目筛选log信息



具体按条目筛选：



## Testing for Robustness and Correctness

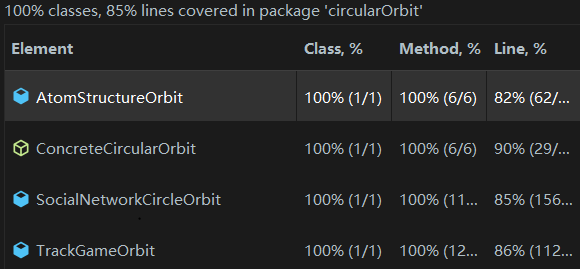
### Testing strategy

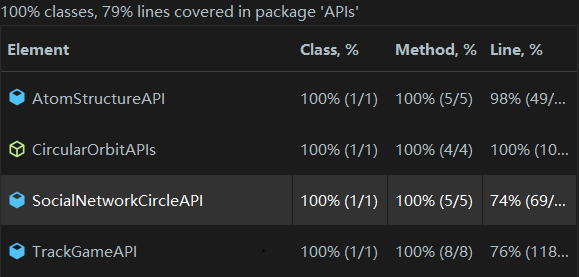
由于在lab3的设计中，实现或支持绝大部分功能的方法集中在API和orbit的类当中，主要测试CircularOrbit和APIs两个package

### 测试用例设计

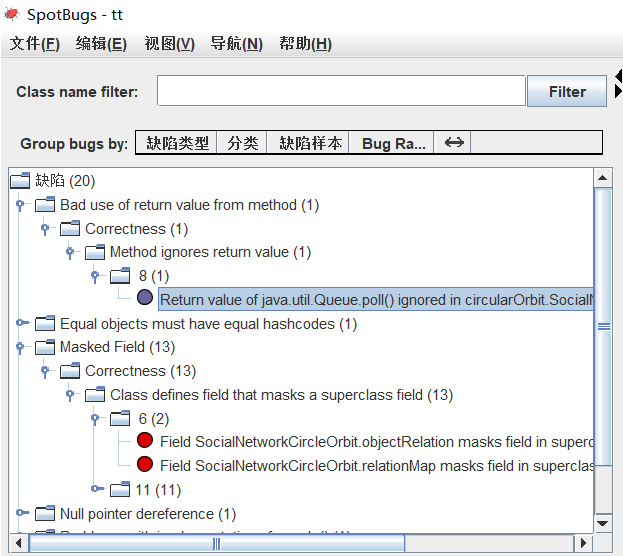
### 测试运行结果与EclEmma覆盖度报告

覆盖度基本在80%以上





## SpotBugs tool



共发现20个缺陷，主要分为四个方面

1. 未获取返回值：在利用BFS构造SocialNetworkCircle的轨道结构时，分别使用队列的peek和poll获取元素和出队，未获取第二个poll的元素
2. 重写方法equals，与使用hashCode的方法冲突
3. 子类中用相同名的对象覆盖父类的对象
4. Position类（未使用）中的constructer书写错误，无用赋值



## Debugging

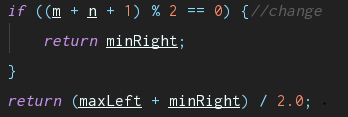
### 理解待调试程序的代码思想

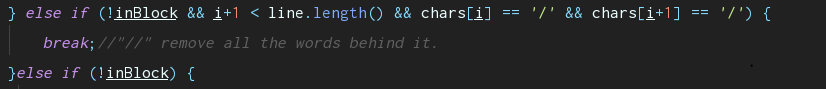
FindMedianSortedArrays：查找两个有序数组的中位数。示例中都是升序，

RemoveComments: 消掉给定字符串序列中的注释部分。

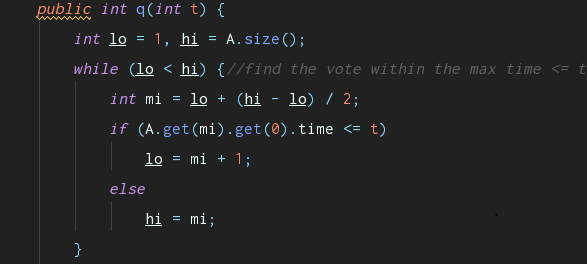
TopVotedCandidate：计算t时刻选举成功的候选人

### 发现并定位错误的过程

1. 发现此处之前判断两个数组的奇偶性的时候出现偏差，但是对于总个数为偶数个数的测试用例总是通过，调试发现只是奇数的时候返回的值有误。
2. 直接运行输出，发现有一行“//”的输出。然后就加了一个elseif，处理了一下直接将那一行删除掉。



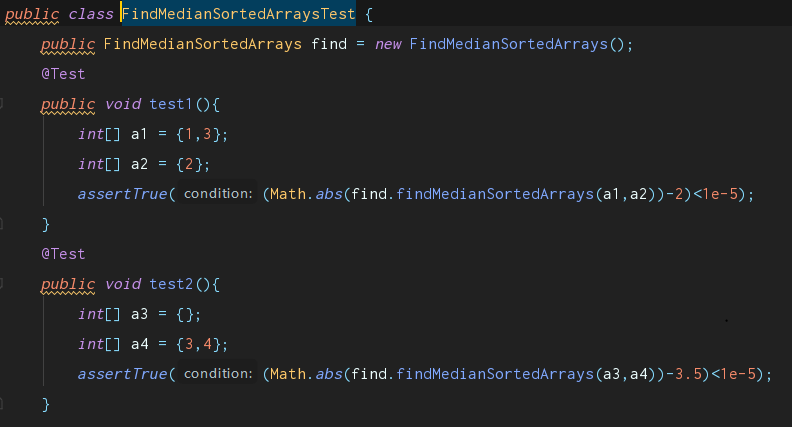
1. 运行，进入死循环。进入debug，发现死在了计算第一个二分法处，查了一下二分法的具体实施。

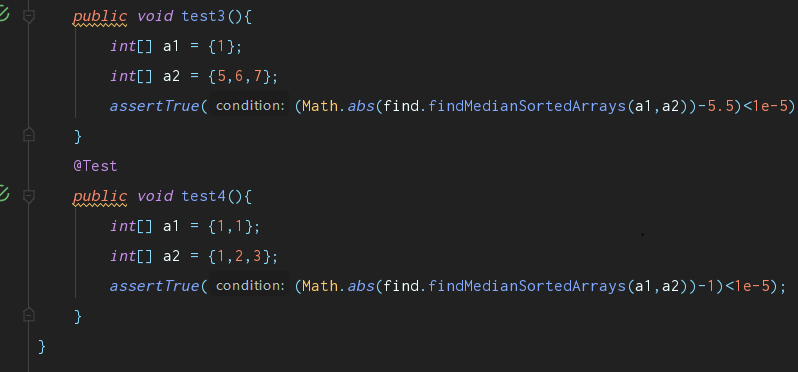


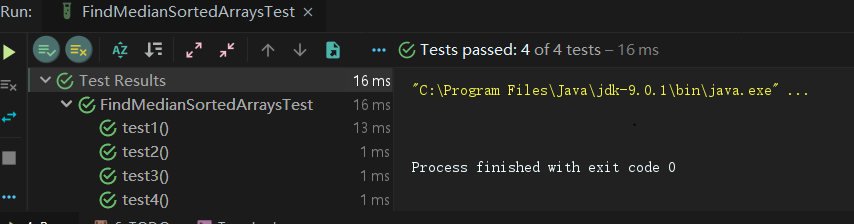
第一层遍历票数，找到离给定的时间最近的投票；按时间遍历，最先录入的就是获得相同票的人中最早的人；查找时间上满足的，同等票中离时间最近的人。

### 结果

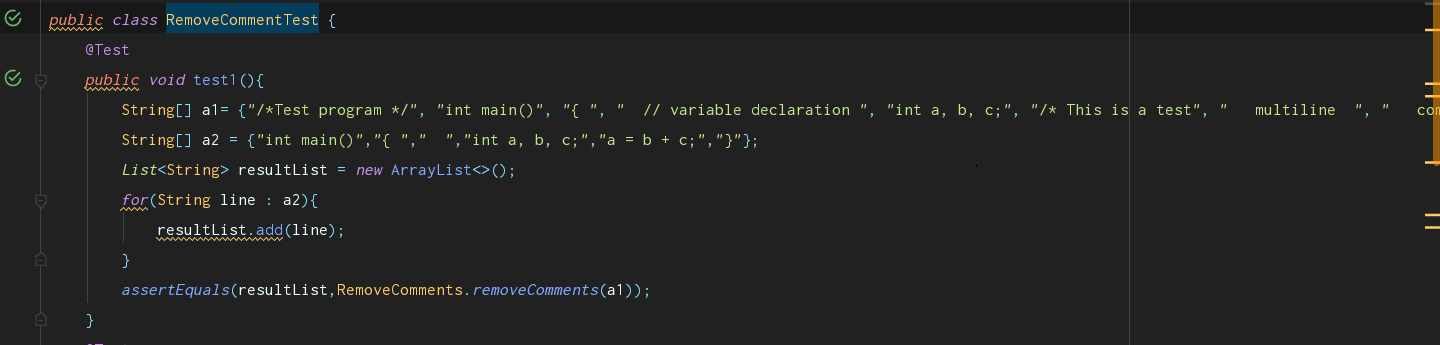
（1）



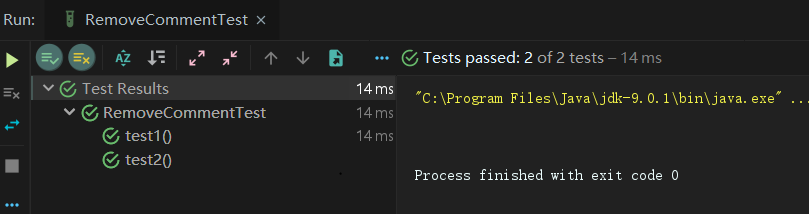




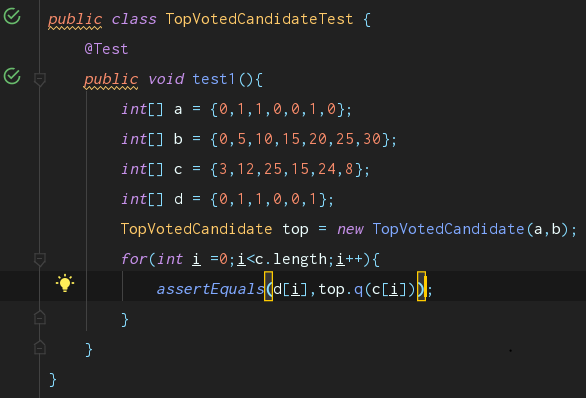
（2）

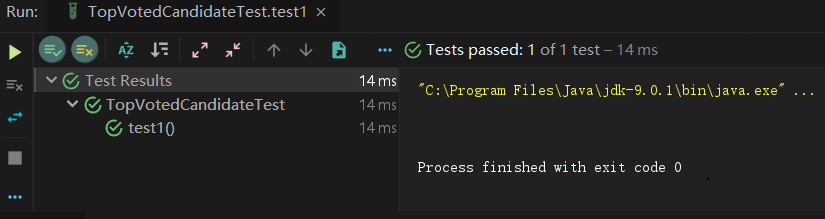






（3）





# 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 |
| 5.13 | 下午+晚上 | Exception | 完成类的创建 |
| 5.14 | 晚上 | Exception | 完成在文件内部调用 |
| 5.15 | 晚上 | Assert+checkrep | 完成 |
| 5.16 | 晚上 | 学习Log4j+配置 | 完成 |
| 5.17 | 晚上 | 调用和筛选log | 完成筛选查看 |
| 5.18 | 全天 | Test | 完成 |
| 5.19 | 全天 | Splotbug+debug | 完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| Maven环境变量 | 安装到E盘 |
| Log4j的文件配置 | Google+照葫芦画瓢 |
|  |  |

# 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

经验主要在于学习log、log4j和maven的使用，通过文件配置插件

## 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？

答：健壮性在于程序接收信息的广度，准确性在于程序的精心设计以及考虑多种情况

1. 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？）

答：应该是划算的吧

1. “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？

答：前者对一般广大用户群体，后者对各种terminal

1. 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？

答：不出错的情况下尽量削减

1. 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？

答：print+debug+log+Assert

1. 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？

答：对各种情况的测试，覆盖到边界特殊情况

1. Debug一个错误的程序，有乐趣吗？

答：海星，比写bug强

1. 关于本实验的工作量、难度、deadline。

答：如果不是正好赶上一个很难的期末考试，都还好

1. 到目前为止你对《软件构造》课程的评价和建议。
2. 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。

答：手写代码不要多，可能写不完