

**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 | 李大鑫 |
| 学号 | 1170300825 |
| 班号 | 1703008 |
| 电子邮件 | [hahalidaxin@163.com](mailto:hahalidaxin@163.com) |
| 手机号码 | 18145192593 |

**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

⚫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

项目配置以及目录信息：



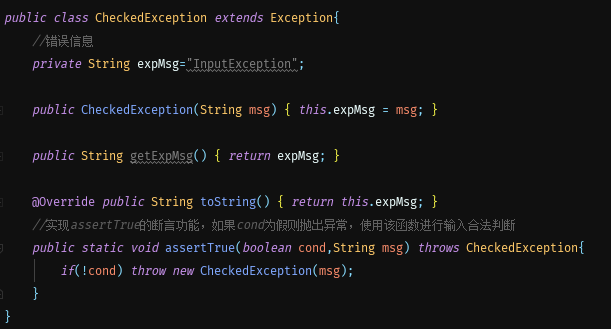
Lab4地址：<https://github.com/ComputerScienceHIT/Lab4-1170300825>

# 实验过程

## Error and Exception Handling

本实验中采用自定义Exception类型，自定义异常类分为两种，一种是Checked异常，一种是Unchecked异常。其中设置CheckedException与UncheckedExceptino，主要使用CheckedException对外部接口进行报错，在本次实验中，对于读入文件中发生的各种错误，使用CheckedException报错；对于程序中内部调用接口或者自身调用函数时可能发生的错误，使用UncheckedException进行报错。无论是CheckedException还是UncheckedException都持有一个String记录错误信息。

CheckedException代码如下：



**实验中构造的错误输入文件都是 正常名称\_E\_错误类型 的格式，如TrackGame\_E\_TrackNum等。**

如何处理输入文件中发生的错误：在Lab3中，使用正则表达式对输入文件进行匹配，在这里我们需要针对错误输入的情况进行判断，对于一行输入，如果不能匹配所有标准正则，那么分析错误原因，基本思想是将输入使用split函数分成不同部分，对于不同部分，再次使用正则表达式尝试匹配，从而定位错误原因。本实验中，一共设置三类输入错误分别继承自CheckedException，分别是输入文件中存在不符合语法规则的语句GrammarException、存在标签完全一样的元素SameLabelException、文件中各元素之间的依赖关系不确定DependencyException。

如何处理错误信息？在本实验中，用户之间的交互完全通过GUI完成，GUI上包括了实现交互的一系列控件。

首先如何捕获异常？

在Swing中，GUI处于与Main Thread不同的AWT Thread，因此对于在GUI交互中发生的异常不能在Main Thread进行捕获，需要在事件（点击、Itemselected等）触发的回调函数中进行try catch捕获并处理。如何处理？这里使用JOption.showMessage显示Exception持有的String，提示用户出错需要重新操作。

错误处理如下：

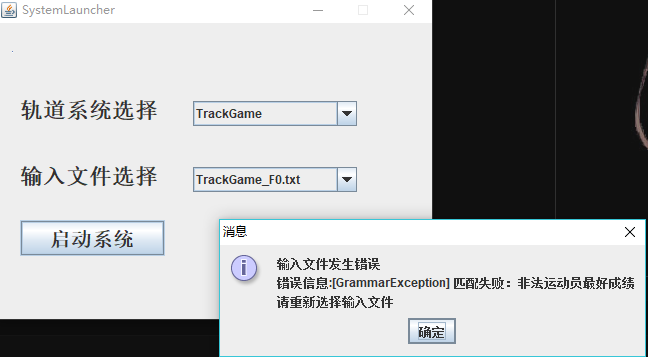


图3.1 TrackGame读入文件时的错误处理

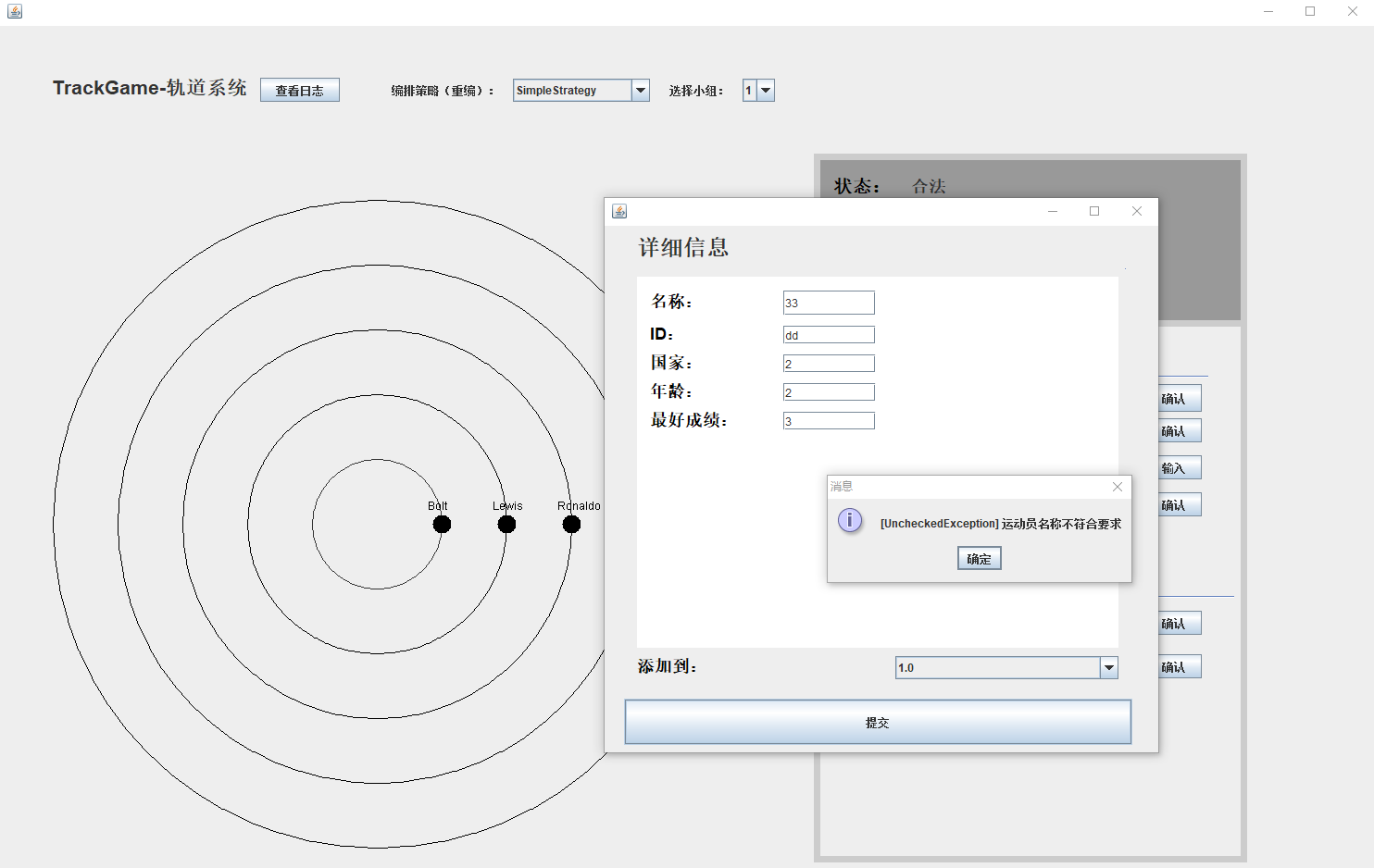


图3.2 TrackGame添加新的运动员时对于输入格式的错误处理

**读文件异常检查位于TrackGame，AtomStructure，SocialNetworkCircle等构造类。也可以通过OrbitLauncher入口GUI选择错误类型文件直接测试。**

## Assertion and Defensive Programming

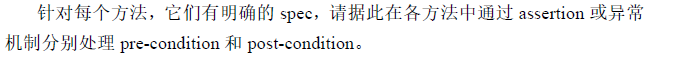
### checkRep()检查invariants

针对题目中涉及的每个ADT根据题目要求设计checkRep维护系统不变性，包括Track，CommonObject，ConcreteCircularOrbit以及继承他的子类等。

对于这些ADT，RI、AF以及Rep Explosure已经声明在类中，checkRep根据RI设计。

如果不满足ADT的循环不变性通过UncheckedException报错。

### Assertion保障pre-/post-condition



在本次实验中，主要工作针对ConcreteCircularOrbit以及其子类，这里采用的异常机制为检查pre-condition和post-condition，如果不满足抛出UncheckedException异常。

## Logging

### 写日志

本实验中使用log4j进行输出日志，首先对log4j进行配置，配置文件如下：

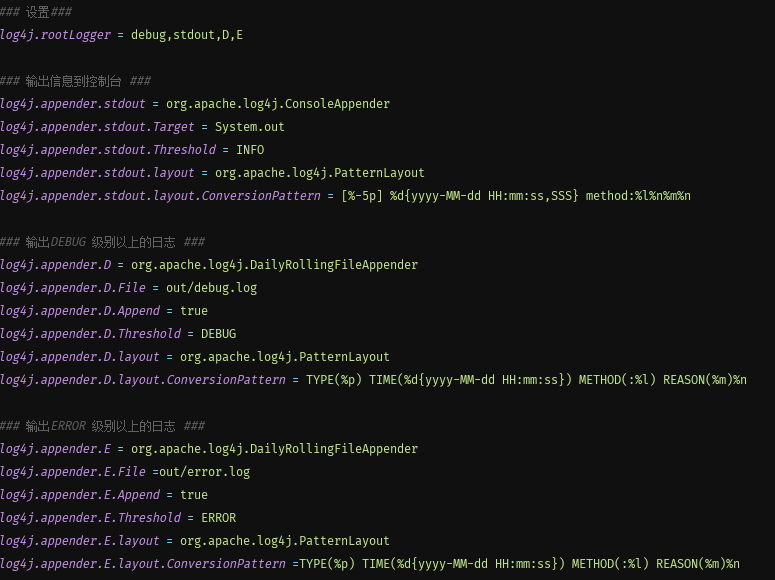


图3,.3 log4j.properties配置文件

可以看到，在本配置文件中，日志同时被打印到控制台和日志文件，对于DEBUG级别及以上的异常输出到debug.log，对于ERROR级别及以上的日志输出到error.log。其中ConversionPattern决定日志打印格式，日志文件采用了比较容易识别的格式，这里也方便了下一步的日志查询。

1. 在所有可能抛出错误的类中声明一个Logger，对于每一个可能抛出的错误，在抛错的同时使用logger.error打印ERROR级别日志。异常错误的发生时间、类名、方法名、具体信息都能够在日志文件获得。**因为在本实验中针对异常错误的处理方式相同（弹窗显示错误信息提示重新处理）所以异常处理结果不计入日志。**
2. 在ConcreteCircularOrbit以及其子类中为每一种轨道操作使用logger.info打印INFO级别日志。

### 日志查询

日志查询的基本思想如下：首先设计一个日志处理配LogParser，这个类提供读取日志文件以及筛选日志的功能。

读入文件时，因为log4j针对不同日的日志分别产生后缀名不同的日志文件，再读入的时候需要将这些日志全部读入，对于每一条日志，使用正则表达式获得日志基本信息（与日志文件输出格式匹配），设置类LogItem对日志基本信息进行存储，存放到LogParser的一个List中。

筛选日志时，根据提供的可用条件在List中筛选满足条件的所有日志，并整合成易于阅读的格式打印。

具体日志查询功能如下：

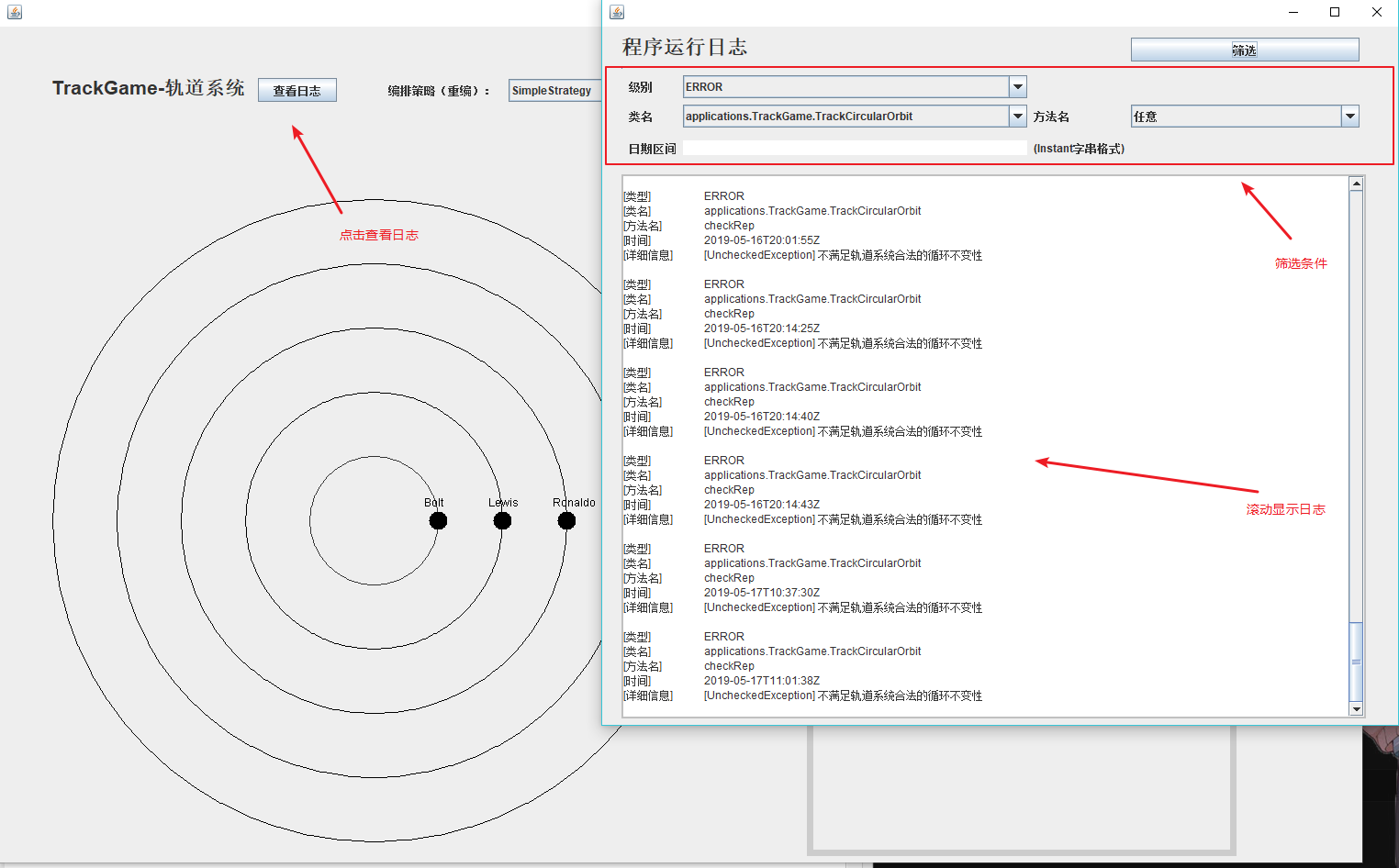


图3.4 查看日志功能演示

对于时间，起始时间以及终止时间都需要满足能够配Instant.parse正常解析的格式，中间使用 , 分隔。如2019-05-16T09:02:04Z,2019-05-16T09:38:39Z，如果输入错误，则提示错误信息。

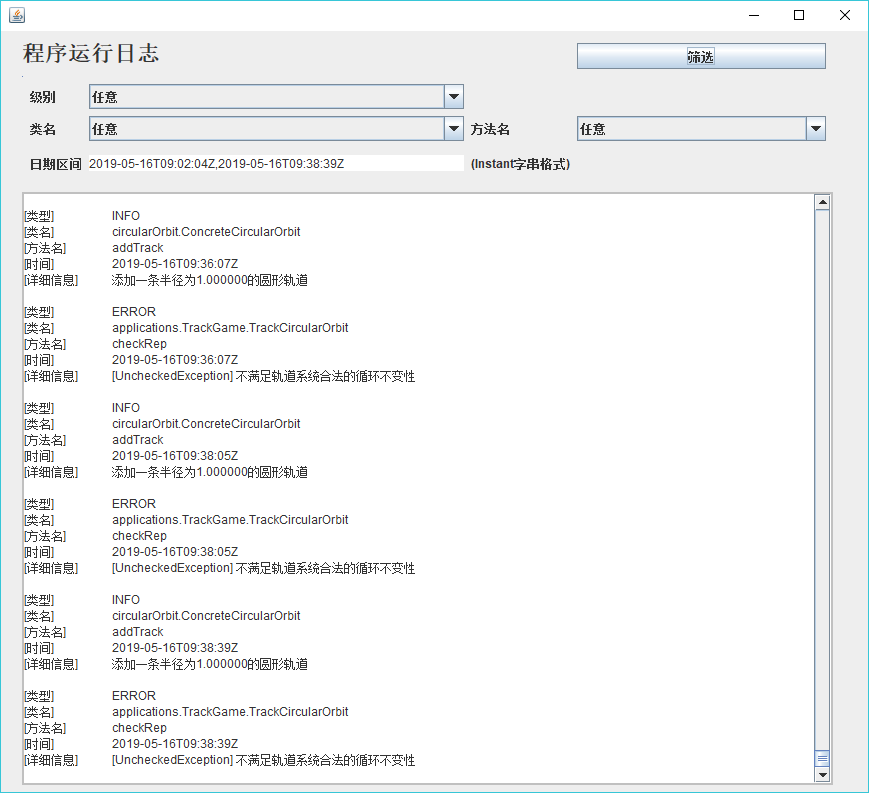


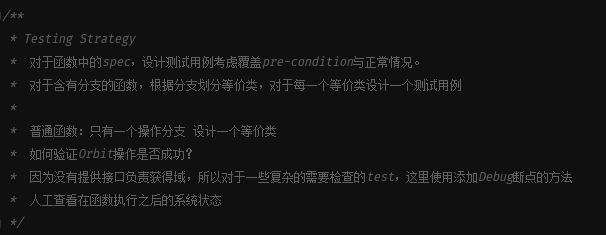
图3.5 根据时间筛选日志

## Testing for Robustness and Correctness

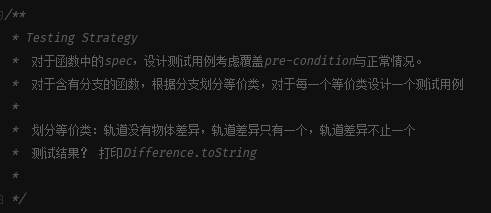
### Testing strategy

对于测试，主要思想相同。而且排除了pre-condition之后的函数大多只有一个分支，所以设计测试数据的思路相似。一些主要类的testing strategy设计如下：

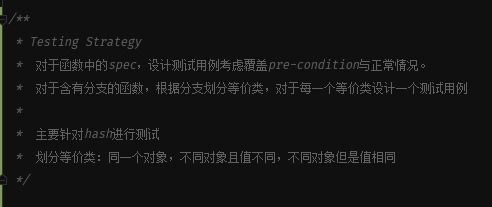
Orbit系统类



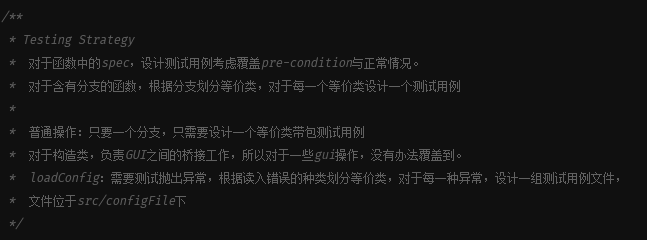
Difference差异类



类的hash测试



构造类AtomStructure测试：



### 测试用例设计

测试用例遵循上小节中的testing strategy设计。在代码中实现，此处不赘述。

### 测试运行结果与EclEmma覆盖度报告

测试运行成果如下：

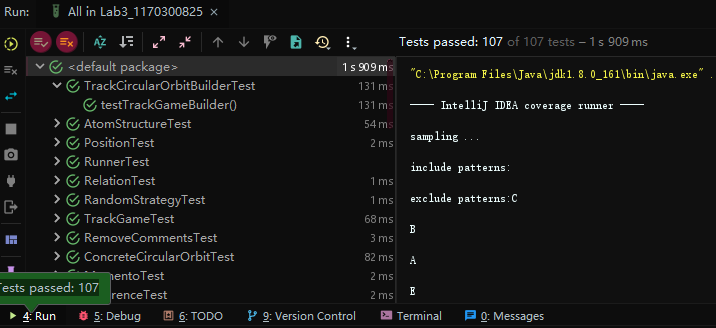


图3..6 测试运行结果

测试报告使用IDEA自带覆盖度测试功能生成，生成的HTML测试覆盖度报告已经存放在了converage目录下。图如下：

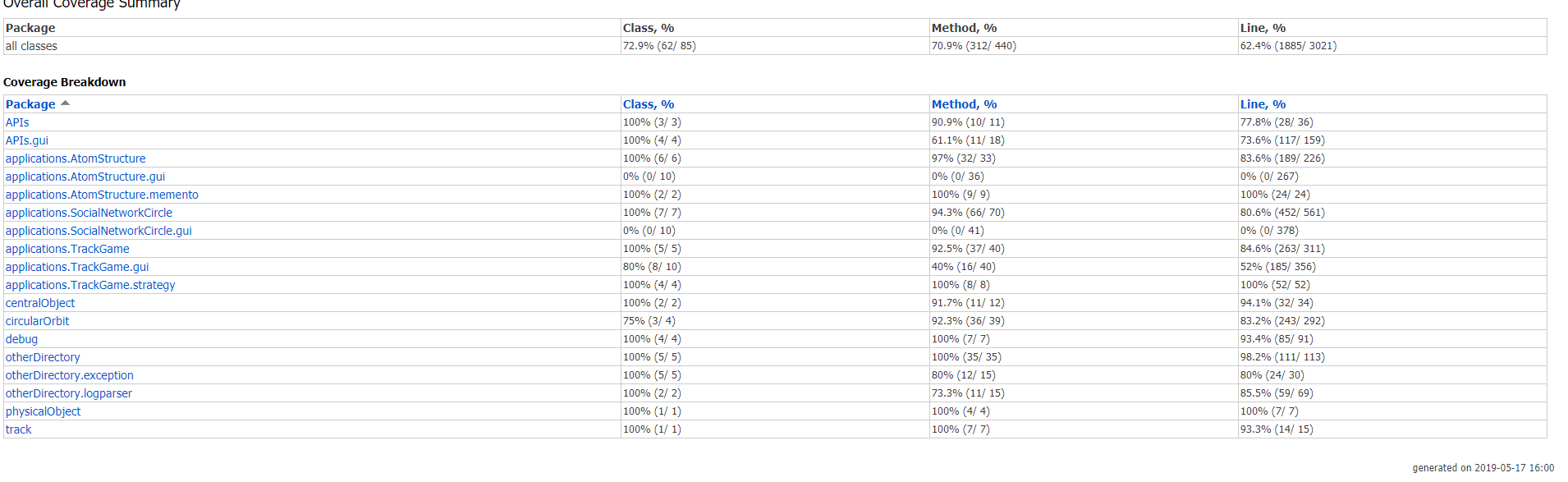
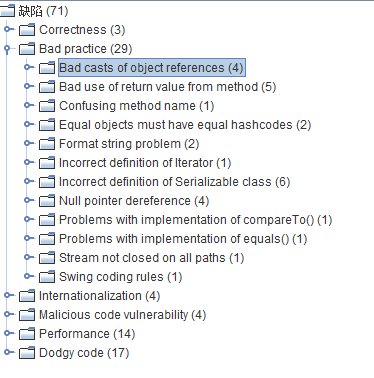
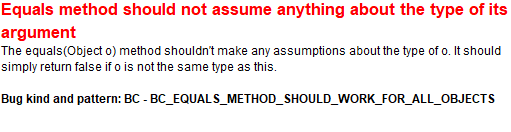


图3.6 Emma代码覆盖度生成报告

**对于覆盖度，需要声明，这里没有针对gui目录下的文件进行测试，对最终覆盖度产生很大影响。因为这些类都是使用JFormDesigner直接生成的，而且测试工作都是通过GUI交互完成的。**

## SpotBugs tool

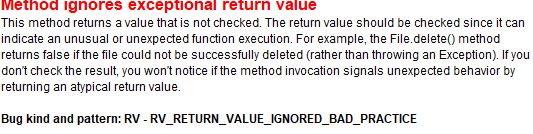




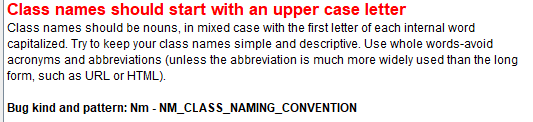
覆盖equals不能假设传入参数类型，需要使用instanceof进行判断



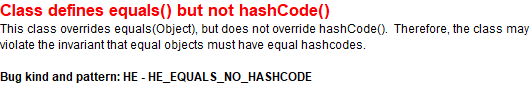
覆盖equals需要检查传入参数为null的情况



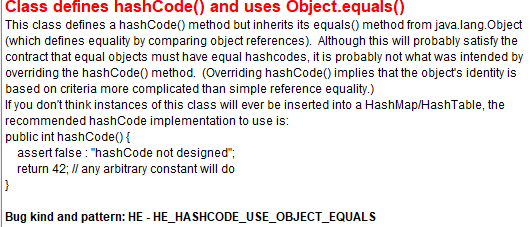
Queue.offer没有检查返回值获取是否入队成功的信息。



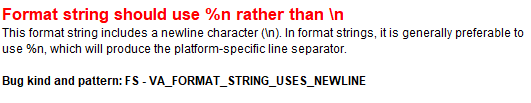
类名开头需要大写



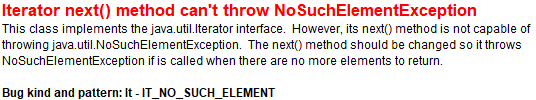
覆盖equals但是没有重写hashCode



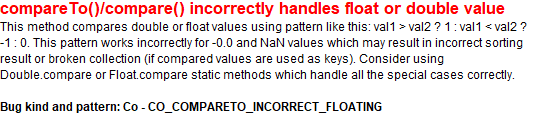
没有覆盖equals但是重写了hashCode



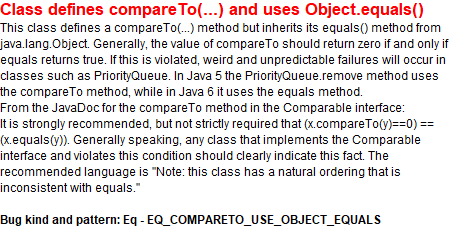
在String.format中使用%n会根据平台不同产生不同效果。



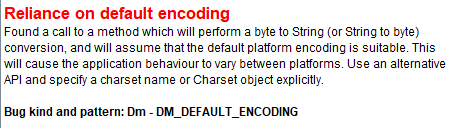
Iterator.next没有抛出NoSuchElementException



Double或者Float进行比较的时候不能正确处理-0.0或者NaN的情况，最好调用Double.compare静态方法。

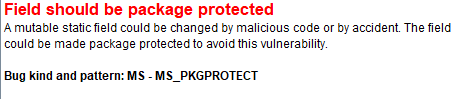


定义了CompareTo但是没有重写equals

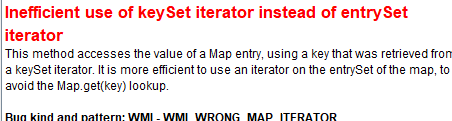


没有规定读入的编码格式，对于不同的平台可能产生不同的效果。需要明确指出读入时使用的编码格式。将读入代码改写成以下格式。

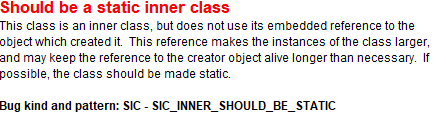




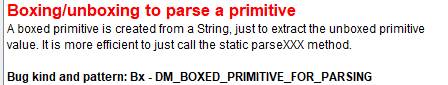
静态公共变量可能被其他类所修改，最好修改成为protected以上级别。



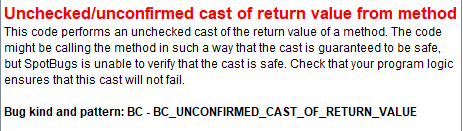
将使用keySet对Map进行遍历修改为使用Map.Entry进行遍历。



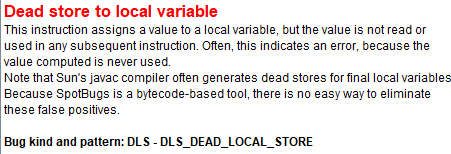
内部类，但是不拥有创建这个类的引用，可能会使得owner拿到这个类的时间过长。建议声明成为static。



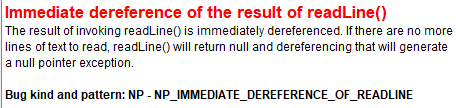
从一个String类型中解析获得一个基础数据类型，建议使用静态方法parseXXX方法更加高效。



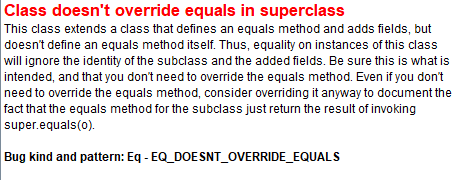
没有确定函数的返回类型直接进行强制类型转换。



本地没有被使用过的变量

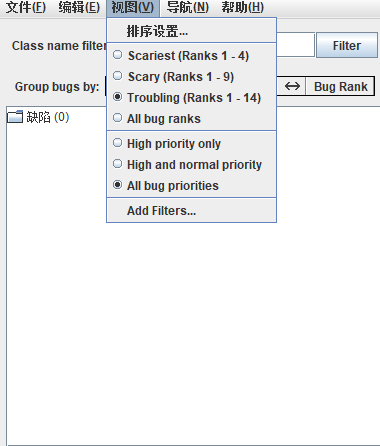


在readLine之后立即调用trim方法，没有判断结果非空



父类重写了equals方法，但是作为子类并没有自身实现这一方法。这样在于父类进行比较的时候可能会出现错误。

最终测试结果如下：



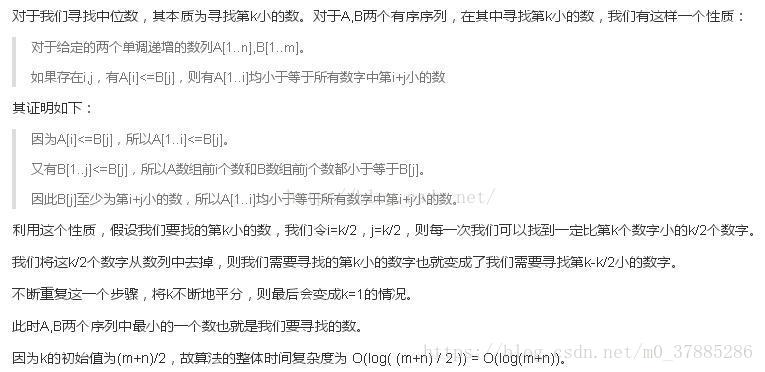
## Debugging

### 理解待调试程序的代码思想

1. FindMedianSoretedArrays。

题目理解：对于两个给定的有序序列，求出这两个序列的中位数。、

解题思路：分治思想的经典应用。



1. RemoveComens

题目理解：去除c++语言中的注释。包括//、/\*\*/等。另外请注意，如果//不是从行首开始的时候，其之前的字符都要进行保存。同理/\*\*/也是。总之不要删除注释范围以内的东西。

解题思路：遍历的方式是加入multi变量是不是一个多行的注释。根据这个变量进行针对性的操作。如果不是多行注释，遇到//直接跳过，遇到/\*要把multi改成多行标记并把i+1来跳过\*号，如果不是上述两种注释则为正常的代码，加入字符串变量。如果是多行注释时，遇到\*/修改multi结束多行字符串并把i+1来跳过/号。

需要注意的是line重新变成空字符串的位置应该在append之后，因为我们认为这部分字符串已经结束了。只要这样的操作才能满足题目中Example 2返回ab的结果。

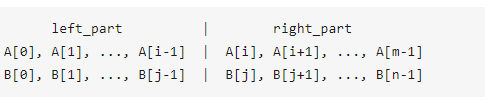
1. TopVotedCandidate

题目理解：在一个选举中第i次投票的情况为persons[i]和times[i]，实现一个函数能实现在时间t返回得票数多的lead。

解题思路：设置一个目前得票最多的变量count，先循环得票人和时间，用hashmap或数组数组统计每个人的得票数，如果次数大于count，则更新count，用数组记录到目前时间得票最多的人(按时间顺序排列的目前得票多的人)。查找某一时间点得票最多的人时，用二分法查找按时间排序的数组。

### 发现并定位错误的过程

1. FindMedianSoretedArrays。
2. 将两个序列划分成为以下情况：



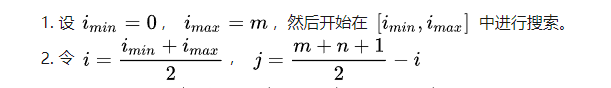
如果存在len(left\_part)==len(right\_part)，max(left\_part)<=min(right\_part)我们就可以确定中位数。为了确保这两个条件需要满足：

1’i+j=m-i+n-j 如果满足n>=m的话，只需要使i=0~m，j=(m+n+1)/2-i

2’B[j-1]<=A[i] and A[i-1]<=B[j].

因此可以看出，这里的haltLen不应该是(m+n)/2

1. 进行二叉树搜索，如下



发现i=(iMax+iMin+1)/2错误。

3）考虑什么情况下maxLeft就是最后答案？当m+n为奇数的时候最中间的也就是maxLeft应该为最后答案，所以发现(m+n+1)%2==1错误。

1. RemoveComments
2. 通过单行注释的测试用例，可以发现，该代码中不存在单行注释的处理情况。
3. 通过比较\*/的处理情况可以看出，最后i++的目的是要跳过这两个字符，所以对于/\*的处理缺少一个i++。
4. TopVotedCondidate

本题中发现错误基于理解代码思路与应用二分查找的定式。

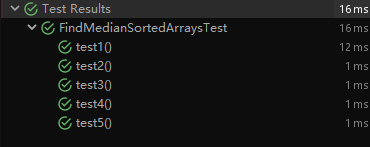
1. 根据变量名count可以猜测这是一个用来计数的，所以推断c = count.getOrDefualt少了一个加1.\
2. 对于q负责查询t时间谁是leader。A的第一个下标代表被投票的次数，A的第二个下标代表Vote索引，Vote是个从前到后的顺序插入的，所以都满足时间顺序。观察q的第一个二分查找，不难发现其目的是要找第一个下标尽可能大的满足list.get(0)<=t的list，这种二分查找是有定式的，可以发现lo=mi是不对的。同时最后得到的结果lo也是不对的。
3. 第二个二分查找是在time有序的Vote列表中查找一个时间<=t且满足下标最大的Vote，同上一个，这里可以发现<t是不对的，j=Math.max(lo,0)是不对。

### 如何修正错误

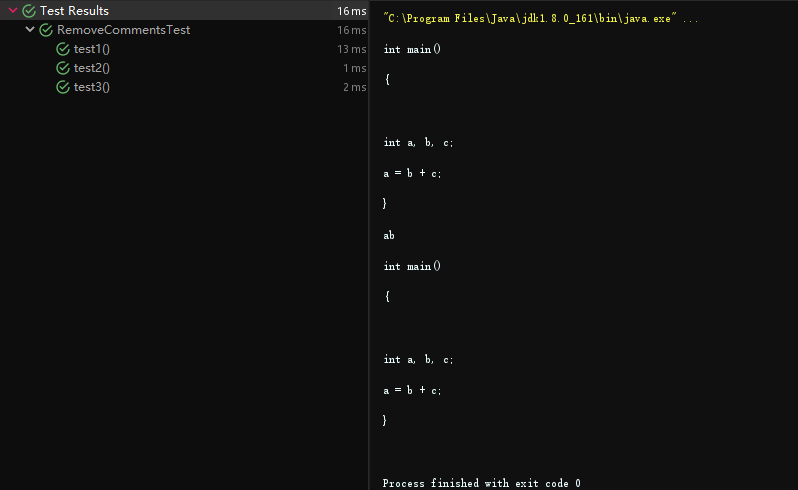
1. FindMedianSoretedArrays
2. 将halfLen=(m+n)/2改成halfLen=(m+n+1)/2
3. 将i=(iMin+iMax+1)/2 改成(iMin+iMax)/2
4. 将(m+n)%2==1改成(m+n)%2==1
5. RemoveComments
6. 当遇到单行注释的时候，只需要跳过改行即可。
7. 对于/\*情况添加 i++跳过
8. TopVotedCondidate
9. 将count.getOrDefault改成count.getOrDefault+1
10. 将lo=mi改成lo=mi+1
11. I=lo改成i=lo-1
12. 将<改成<=t
13. 将j = Math.max(lo,0)改成j=Math.max(lo-1,0)

### 结果

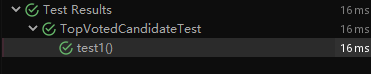
1. FindMedianSoretedArrays



1. RemoveComments



1. TopVotedCondidate



# 实验进度记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 5/14 | 18:00-21:55 | 3.1 | 完成 |
| 5/15 | 14:00-16:47 | 3.3 | 完成 |
| 5/15 | 16:47-22:36 | 3.3 3.5 | 完成 |
| 5/16 | 16:00-19:00 | 3.4 | 完成 |
| 5/17 | 16:00-22:00 | 写报告 | 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 |
| GUI，滚动区域不滚动 | 需要根据需要显示的子串设置ScrollPane下的TextArea的行数 |
| Test实在太多 | 慢慢写 |

# 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

## 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？

健壮性使程序员考虑更多如边界条件等，正确性可能更多关注程序是否正确。

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？）

如果你的系统需要经受大规模、频繁的使用的话的确值得。

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？

保守与激进的编程方法。

在外部接口处倾向于第一种，如果是内部接口的话倾向于第一种。

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？

如果是想要将自己的ADT公共使用的话很有必要。不知道，还没看Lab5。

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？

首先通过试错获得一个错误的输入用例，然后进入debug在关键位置设置断点确定错误区间，进一步确定错误语句。

会。

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？

否，代码100%代表运行到了所有的代码，但是不能够保证每一种等价类都能够有一个代表测试用例。

1. Debug一个错误的程序，有乐趣吗？

Debug成功之后挺有乐趣的。

1. 关于本实验的工作量、难度、deadline。

Test的工作量很大。

1. 到目前为止你对《软件构造》课程的评价和建议。

挺好的。

1. 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。

希望老师按照同学们实际的学习情况出题。