

**2019年春季学期  
计算机学院《软件构造》课程**

**Lab 5实验报告**

|  |  |
| --- | --- |
| 姓名 | 任永鹏 |
| 学号 | 1173710107 |
| 班号 | 1737101 |
| 电子邮件 | [805402160@qq.com](mailto:805402160@qq.com) |
| 手机号码 | 18800420869 |

**目录**

[1 实验目标概述 1](#_Toc3923017)

[2 实验环境配置 1](#_Toc3923018)

[3 实验过程 1](#_Toc3923019)

[3.1 Static Program Analysis 1](#_Toc3923020)

[3.1.1 人工代码走查（walkthrough） 1](#_Toc3923021)

[3.1.2 使用CheckStyle和SpotBugs进行静态代码分析 1](#_Toc3923022)

[3.2 Java I/O Optimization 1](#_Toc3923023)

[3.2.1 多种I/O实现方式 1](#_Toc3923024)

[3.2.2 多种I/O实现方式的效率对比分析 2](#_Toc3923025)

[3.3 Java Memory Management and Garbage Collection (GC) 3](#_Toc3923026)

[3.3.1 使用-verbose:gc参数 3](#_Toc3923027)

[3.3.2 用jstat命令行工具的-gc和-gcutil参数 3](#_Toc3923028)

[3.3.3 使用jmap -heap命令行工具 3](#_Toc3923029)

[3.3.4 使用jmap -clstats命令行工具 3](#_Toc3923030)

[3.3.5 使用jmap -permstat命令行工具 3](#_Toc3923031)

[3.3.6 使用JMC/JFR、jconsole或VisualVM工具 3](#_Toc3923032)

[3.3.7 分析垃圾回收过程 3](#_Toc3923033)

[3.3.8 配置JVM参数并发现优化的参数配置 3](#_Toc3923034)

[3.4 Dynamic Program Profiling 3](#_Toc3923035)

[3.4.1 使用JMC或VisualVM进行CPU Profiling 3](#_Toc3923036)

[3.4.2 使用VisualVM进行Memory profiling 3](#_Toc3923037)

[3.5 Memory Dump Analysis and Performance Optimization 3](#_Toc3923038)

[3.5.1 内存导出 3](#_Toc3923039)

[3.5.2 使用MAT分析内存导出文件 3](#_Toc3923040)

[3.5.3 发现热点/瓶颈并改进、改进前后的性能对比分析 3](#_Toc3923041)

[3.5.4 在MAT内使用OQL查询内存导出 4](#_Toc3923042)

[3.5.5 观察jstack/jcmd导出程序运行时的调用栈 4](#_Toc3923043)

[3.5.6 使用设计模式进行代码性能优化 4](#_Toc3923044)

[4 实验进度记录 4](#_Toc3923045)

[5 实验过程中遇到的困难与解决途径 4](#_Toc3923046)

[6 实验过程中收获的经验、教训、感想 5](#_Toc3923047)

[6.1 实验过程中收获的经验和教训 5](#_Toc3923048)

[6.2 针对以下方面的感受 5](#_Toc3923049)

# 实验目标概述

根据实验手册简要撰写。

# 实验环境配置

简要陈述你配置本次实验所需环境的过程，必要时可以给出屏幕截图。

特别是要记录配置过程中遇到的问题和困难，以及如何解决的。

在这里给出你的GitHub Lab5仓库的URL地址（Lab5-学号）。

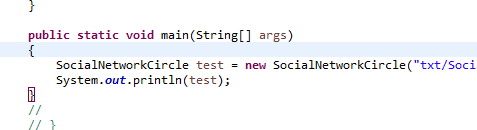
# 实验过程

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！）。

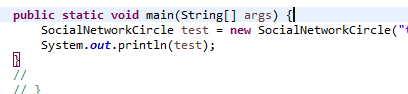
## Static Program Analysis

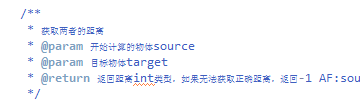
### 人工代码走查（walkthrough）

这里我在阅读了google的代码规范之后试着改了几处问题，剩下的大量类似的代码风格不规范的问题，因为一个一个改浪费时间，我都是直接在eclipse里面直接导入java的google代码风格文件，然后让eclipse修改的，值得一提的是这里的修改应该有局限性，比如注释等等很多问题还是没办法解决，剩下的问题需要用checkstyle解决。

1.

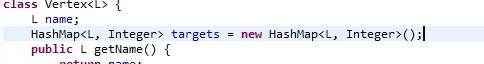
修改大括号之后变成



2. 

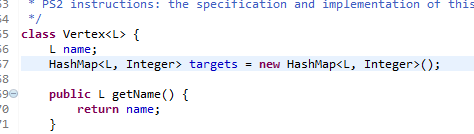
修改spec格式之后变成

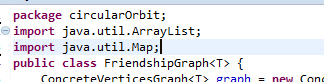


3. 

变量和方法之间要有间距

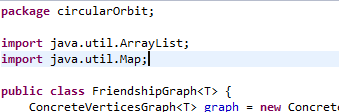
修改之后



4. 

Package import class之间要有间隔

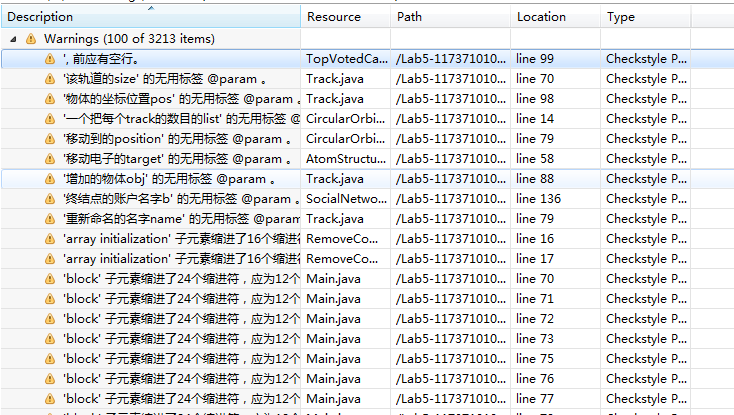
修改之后



### 使用CheckStyle和SpotBugs进行静态代码分析

列出你所发现的问题和所做的修改。每种类型的问题只需列出一个示例即可。

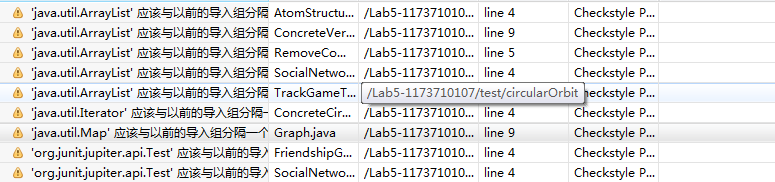
对比分析两种工具发现问题的能力和发现问题的类型上有何差异。



可以看到，主要问题就是三个，有空行，无用标签，缩进符的问题

有空行的问题只要分开声明就可以解决

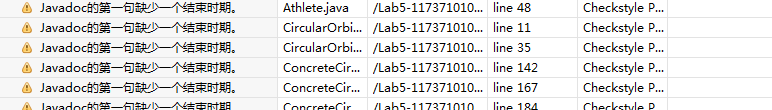
无用标签，我觉得可能是在不该写param的地方写了这个，或者是有多个param，删除就好了

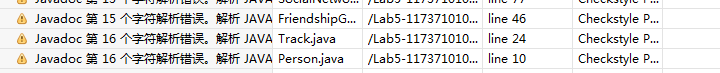


需要取消我之前设置的代码自动格式化，然后加上空行

然后还有数组大括号位置错误的提示，num[]变成[]num,更改之后就好了

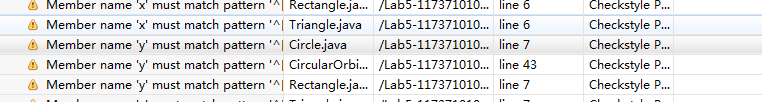
问题更多的在spec部分





这里检测到我的spec的格式不对，在@param中，参数名字要写在@param之后，中间空一格，然后参数名和定义中间也要空一格，在标注该方法的功能的时候，需要结尾加.

剩下的问题就是



这个我不知道该怎么改了，就是简单的定义变量xy，结果会有不匹配正则表达式的问题



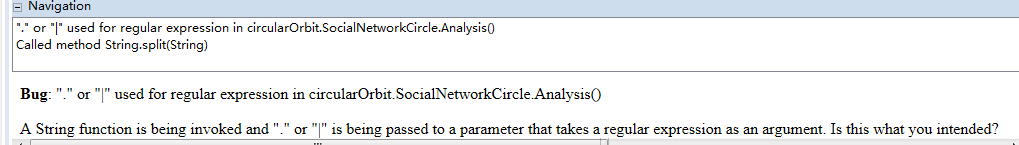
这个因为我要逐步的读入控制台信息，所以没办法调换代码的位置，这个没法改

Spotbugs检查

对代码修改，消除这些错误。

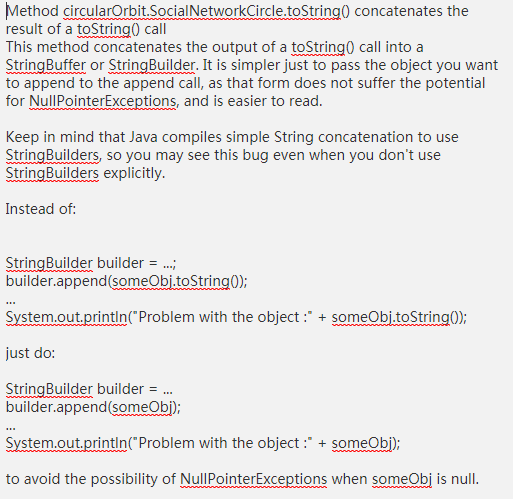
错误一：





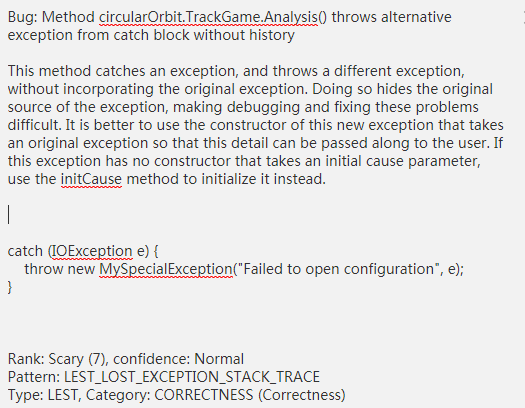
符号没\\，加上[\\就可以了](file:///\\就可以了)

错误二：



方法：把tostring那个代码重新单独写一行，变成一个string添加就可以了

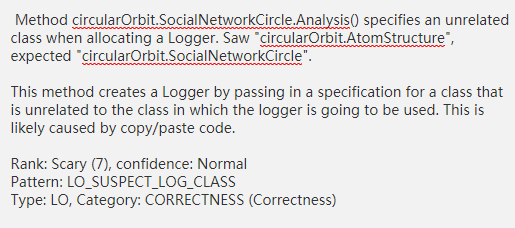
错误三：



问题出错原因：

在没有抛出相对应异常的情况下，catch异常并处理

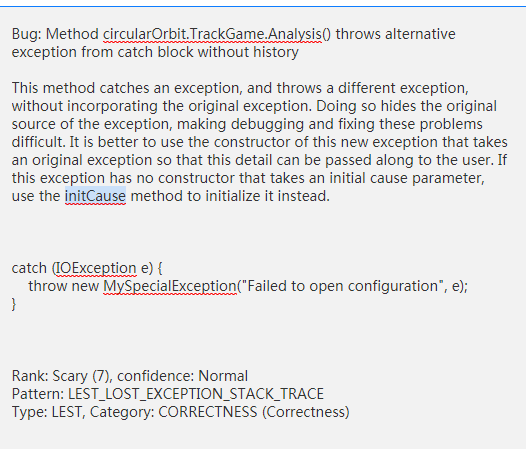
错误四：





原因：这里的logger括号里面的类名忘记改了

错误五：





这个我真的不会改。。。。

下面是我查到的资料

<https://www.imooc.com/qadetail/110953>

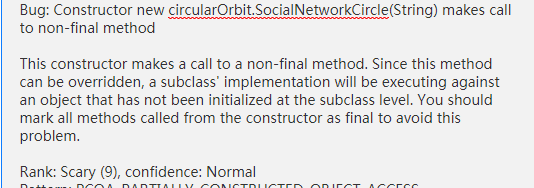
我看了这个资料，大概知道是要进行异常的包装，然后方便找到根源的问题，但是我这里感觉并没有出现所说的捕获异常A，然后抛出异常B的问题啊。。。

我尽力了！！！！

。。。。。

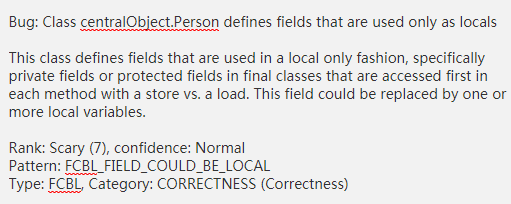
突然又好了，又不报错了，我啥也没改，为了防止到时候检查出来，我还是先写报告里面了

错误六：



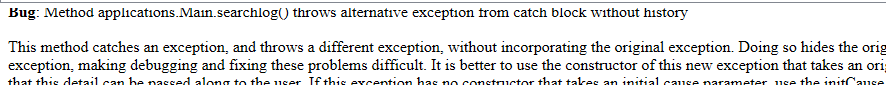
方法加final就可以了

错误七：



删掉private就可以了

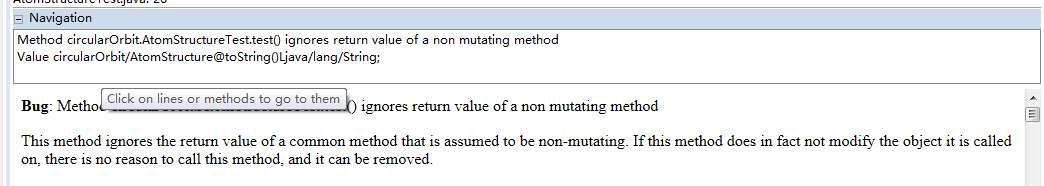
错误八：

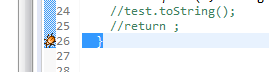




这个就是上面我说的那个，用initcause就可以了

剩下一个问题改不了：





这个不知道什么原因，会出现这样的问题，明明是一个void方法，结果会出现返回值的问题，这个不会改，，，

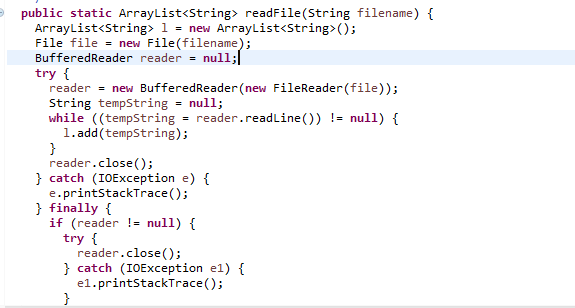
## Java I/O Optimization

### 多种I/O实现方式

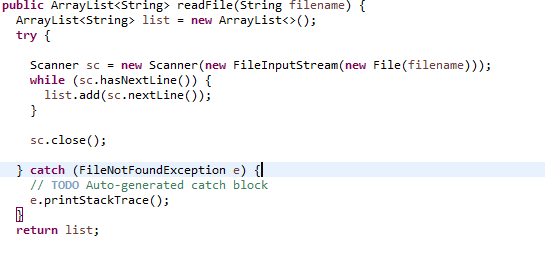
实现了哪些I/O方式来读写文件，具体如何实现的。

如何用strategy设计模式实现在多种I/O策略之间的切换。

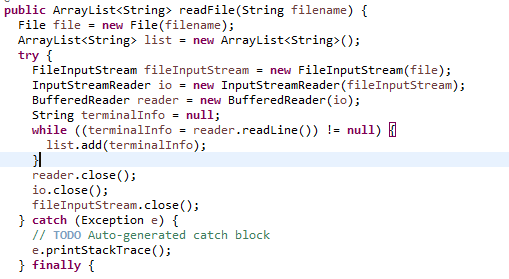
读写文件使用了bufferreader



Scanner

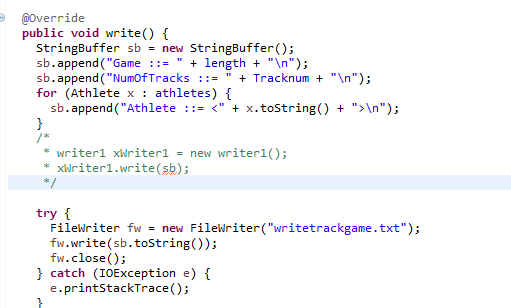


Stream

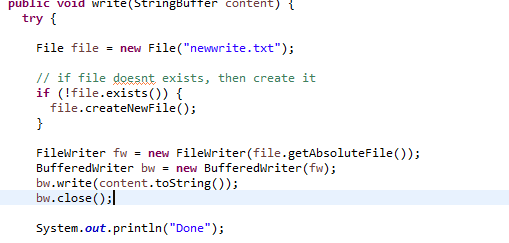


写文件

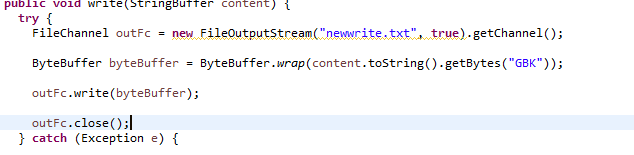
Filewriter



Bufferwriter



Channel



设计模式的应用

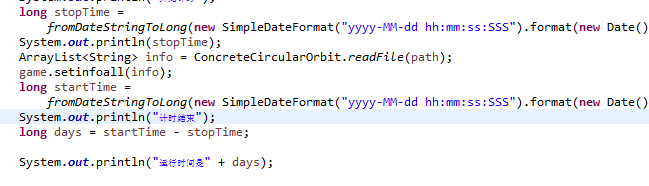
写一个读文件的接口，然后写一个写文件的接口，再多写两个实现类，然后通过调用实现类的方法来实现

### 多种I/O实现方式的效率对比分析

如何收集你的程序I/O语法文件的时间。

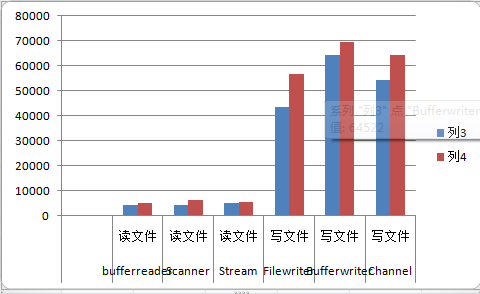
表格方式对比不同I/O的性能。

图形对比不同I/O的性能。



通过计算代码运行开始的时间和结束的时间，然后两者相减，获得程序运行的时间

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | TestTrackGame.txt | socialNetWorkCircle(2).txt |
| bufferreader | 读文件 | 4367 | 5234 |
| Scanner | 读文件 | 4352 | 6556 |
| Stream | 读文件 | 5332 | 5432 |
| Filewriter | 写文件 | 43553 | 56544 |
| Bufferwriter | 写文件 | 64522 | 69353 |
| Channel | 写文件 | 54332 | 64325 |
|  |  |  |  |



## Java Memory Management and Garbage Collection (GC)

### 使用-verbose:gc参数

Java HotSpot(TM) Client VM (25.201-b09) for windows-x86 JRE (1.8.0\_201-b09), built on Dec 16 2018 00:13:48 by "java\_re" with MS VC++ 10.0 (VS2010)

1.Memory: 4k page, physical 3376960k(1173092k free), swap 6752244k(2985608k free)

2.0.292: [GC (Allocation Failure) 4416K->1606K(15872K), 0.2238031 secs]

6.089: [GC (Allocation Failure) 6022K->2908K(15872K), 0.0129736 secs]

3. CommandLine flags: -XX:InitialHeapSize=16777216 -XX:MaxHeapSize=268435456 -XX:+PrintGC -XX:+PrintGCTimeStamps -XX:-UseLargePagesIndividualAllocation

### 用jstat命令行工具的-gc和-gcutil参数

### 使用jmap -heap命令行工具

### 使用jmap -clstats命令行工具

### 使用jmap -permstat命令行工具

### 使用JMC/JFR、jconsole或VisualVM工具

### 分析垃圾回收过程

### 配置JVM参数并发现优化的参数配置

## Dynamic Program Profiling

### 使用JMC或VisualVM进行CPU Profiling

### 使用VisualVM进行Memory profiling

## Memory Dump Analysis and Performance Optimization

### 内存导出

### 使用MAT分析内存导出文件

### 发现热点/瓶颈并改进、改进前后的性能对比分析

### 在MAT内使用OQL查询内存导出

### 观察jstack/jcmd导出程序运行时的调用栈

### 使用设计模式进行代码性能优化

## Git仓库结构

请在完成全部实验要求之后，利用Git log指令或Git图形化客户端或GitHub上项目仓库的Insight页面，给出你的仓库到目前为止的Object Graph，尤其是区分清楚本实验中要求的多个分支和master分支所指向的位置。

# 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 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
|  | 1.下载eclipse插件 <http://sourceforge.net/projects/eclipse-cs/> 下载后，把里面的两个文件夹plugins和 features下面的文件分别拷贝到eclipse目录下面对应的plugins和features目录 2.如果你就这样直接启动eclipse 可能会出现ClassNotFoundExceptions 等错误，所以必须在启动eclipse的时候加上一个参数 -clean. 这样eclipse就可以更新它当前安装的许多插件信息 比如你得eclipse在d盘 那么就是 d:\eclipse\eclipse.exe -clean |
|  |  |
|  |  |

# 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. 代码“看起来很美”和“运行起来很美”，二者之间有何必然的联系或冲突？哪个比另一个更重要些吗？在有限的编程时间里，你更倾向于把精力放在哪个上？
2. 诸如SpotBugs和CheckStyle这样的代码静态分析工具，会提示你的代码里有无数不符合规范或有潜在bug的地方，结合你在本次实验中的体会，你认为它们是否会真的帮助你改善代码质量？
3. 为什么Java提供了这么多种I/O的实现方式？从Java自身的发展路线上看，这其实也体现了JDK自身代码的逐渐优化过程。你是否能够梳理清楚Java I/O的逐步优化和扩展的过程，并能够搞清楚每种I/O技术最适合的应用场景？
4. JVM的内存管理机制，与你在《计算机系统》课程里所学的内存管理基本原理相比，有何差异？有何新意？你认为它是否足够好？
5. JVM自动进行垃圾回收，从而避免了程序员手工进行垃圾回收的麻烦（例如在C++中）。你怎么看待这两种垃圾回收机制？你认为JVM目前所采用的这些垃圾回收机制还有改进的空间吗？
6. 基于你在实验中的体会，你认为“通过配置JVM内存分配和GC参数来提高程序运行性能”是否有足够的回报？
7. 通过Memory Dump进行程序性能的分析，JMC/JFR、VisualVM和MAT这几个工具提供了很强大的分析功能。你是否已经体验到了使用它们发现程序热点以进行程序性能优化的好处？
8. 使用各种代码调优技术进行性能优化，考验的是程序员的细心，依赖的是程序员日积月累的编程中养成的“对性能的敏感程度”。你是否有足够的耐心，从每一条语句、每一个类做起，“积跬步，以至千里”，一点一点累积出整体性能的较大提升？
9. 关于本实验的工作量、难度、deadline。
10. 到目前为止，你对《软件构造》课程的意见与建议。