

**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 | 任永鹏 |
| 学号 | 1173710107 |
| 班号 | 1737101 |
| 电子邮件 | 805402160@QQ.COM |
| 手机号码 | 18800420869 |

**目录**

[1 实验目标概述 1](#_Toc9181530)

[2 实验环境配置 1](#_Toc9181531)

[3 实验过程 2](#_Toc9181532)

[3.1 Error and Exception Handling 2](#_Toc9181533)

[3.2 Assertion and Defensive Programming 7](#_Toc9181534)

[3.2.1 checkRep()检查invariants 7](#_Toc9181535)

[3.2.2 Assertion保障pre-/post-condition 7](#_Toc9181536)

[3.3 Logging 8](#_Toc9181537)

[3.3.1 写日志 10](#_Toc9181538)

[3.3.2 日志查询 11](#_Toc9181539)

[3.4 Testing for Robustness and Correctness 13](#_Toc9181540)

[3.4.1 Testing strategy 13](#_Toc9181541)

[3.4.2 测试用例设计 13](#_Toc9181542)

[3.4.3 测试运行结果与EclEmma覆盖度报告 14](#_Toc9181543)

[3.5 SpotBugs tool 15](#_Toc9181544)

[3.6 Debugging 19](#_Toc9181545)

[3.6.1 理解待调试程序的代码思想 19](#_Toc9181546)

[3.6.2 发现并定位错误的过程 19](#_Toc9181547)

[3.6.3 如何修正错误 19](#_Toc9181548)

[3.6.4 结果 20](#_Toc9181549)

[4 实验进度记录 21](#_Toc9181550)

[5 实验过程中遇到的困难与解决途径 21](#_Toc9181551)

[6 实验过程中收获的经验、教训、感想 22](#_Toc9181552)

[6.1 实验过程中收获的经验和教训 22](#_Toc9181553)

[6.2 针对以下方面的感受 23](#_Toc9181554)

# 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

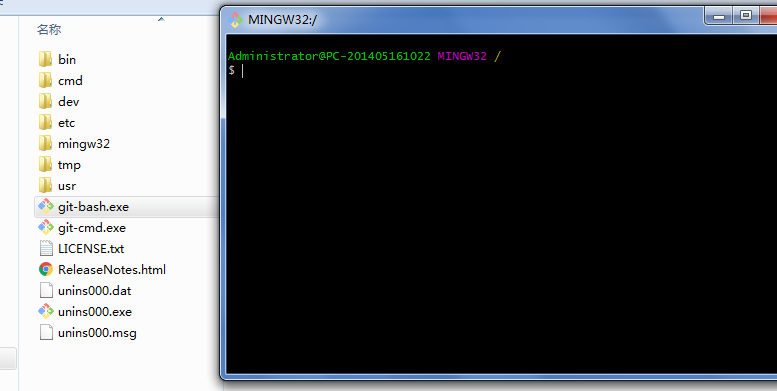
简要陈述你配置本次实验所需环境的过程，必要时可以给出屏幕截图。

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。

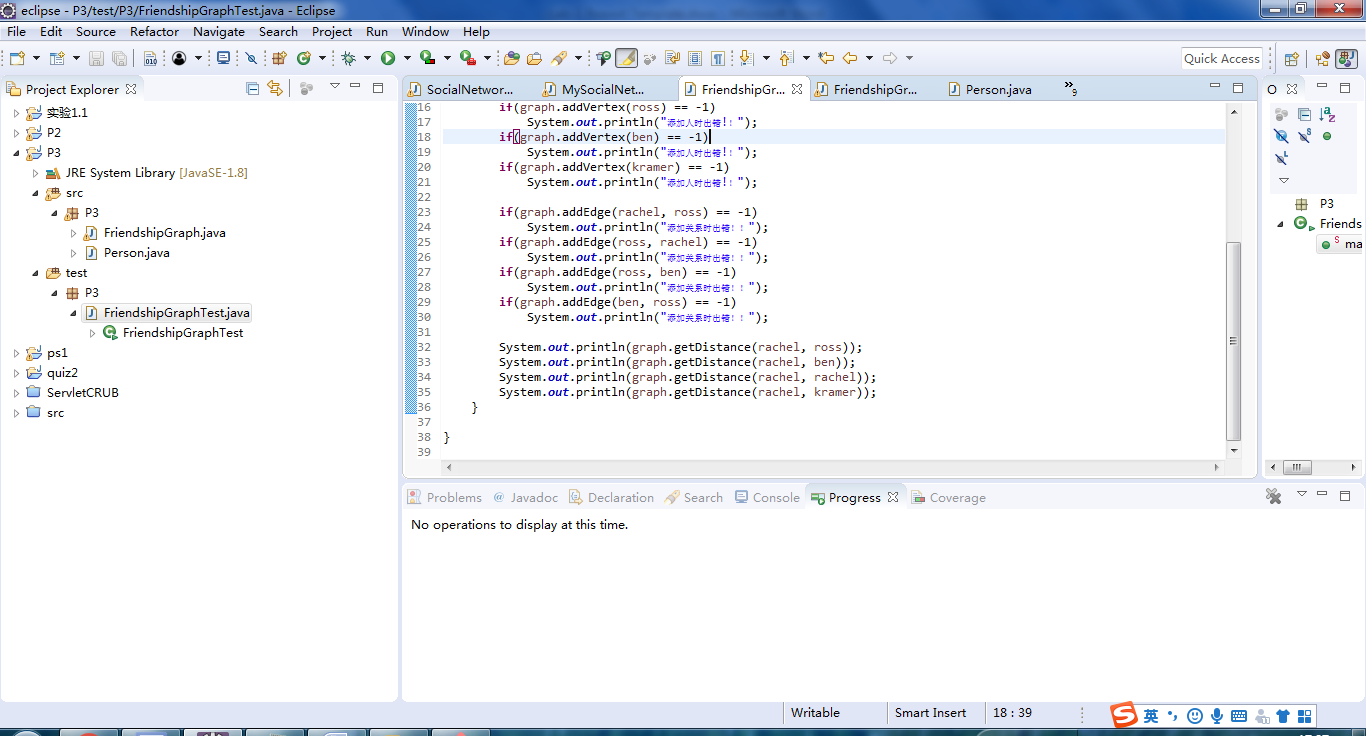
1安装Java

从Java Development Kit 10下载，下载Java SE Development Kit 10.0.2并运行安装程序。此处需要配置环境变量。

2安装Git



3安装eclipse



在这里给出你的GitHub Lab4仓库的URL地址（Lab4-学号）。

https://github.com/ComputerScienceHIT/Lab4-1173710107.git

# 实验过程

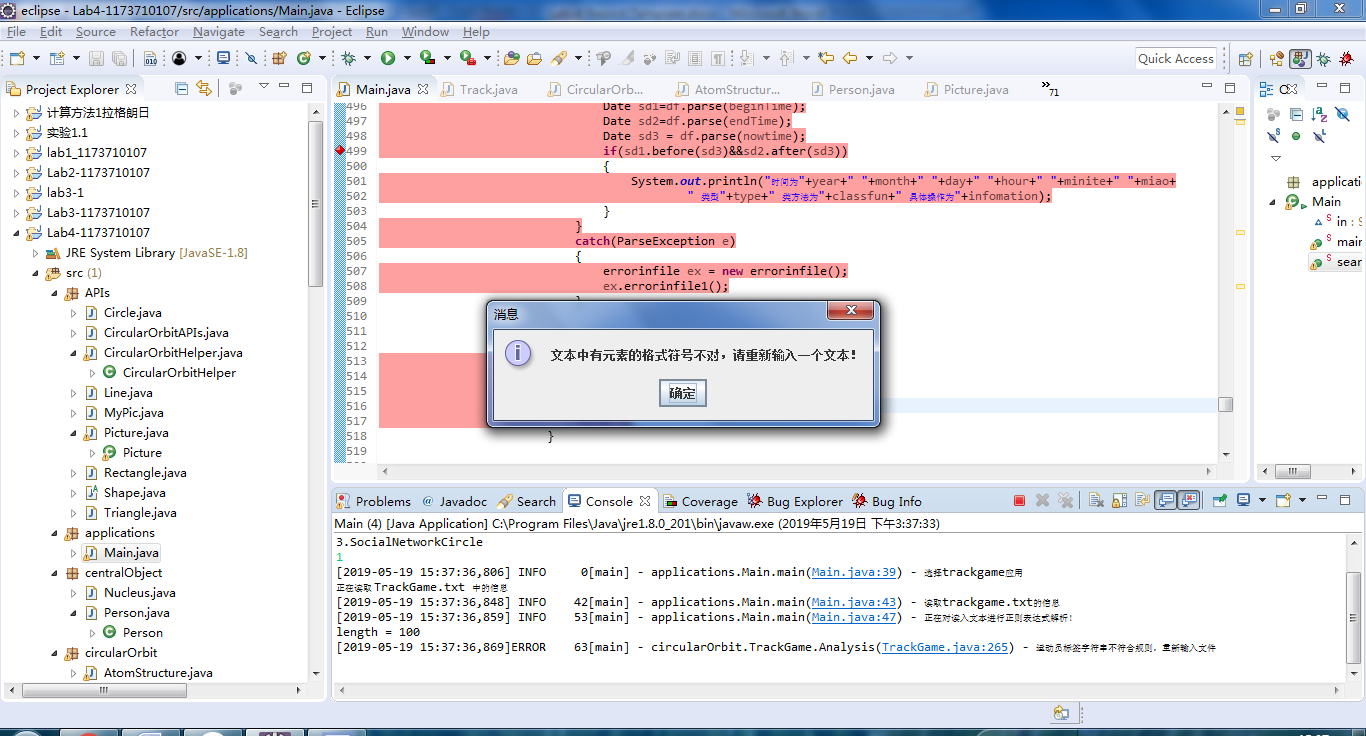
请仔细对照实验手册，针对每一项任务，在下面各节中记录你的实验过程、阐述你的设计思路和问题求解思路，可辅之以示意图或关键源代码加以说明（但千万不要把你的源代码全部粘贴过来！）。

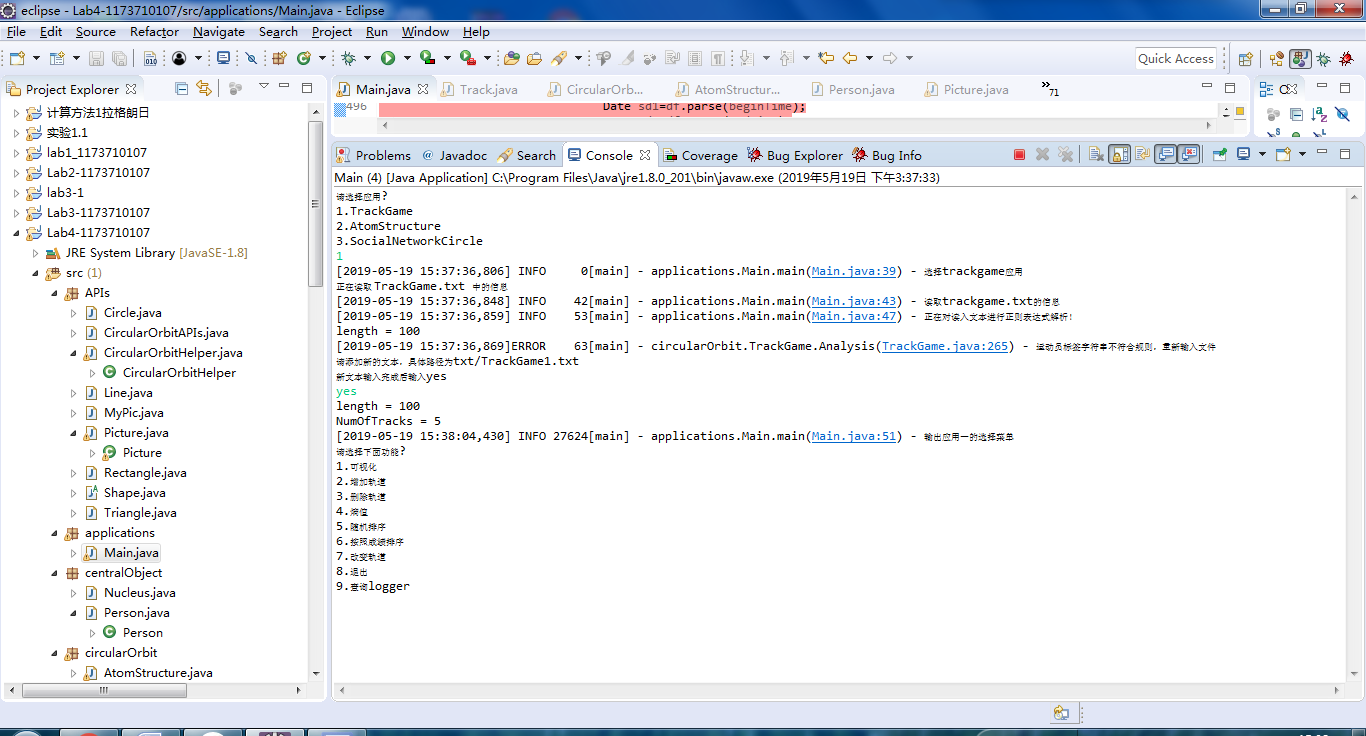
## Error and Exception Handling

首先我先介绍一下自己写的异常文本

用trackgame为例，txt文件夹中包括tackgame,trackgame1,trackgame2,trackgame3,trackgame4,trackgame5一共有六个文本文档，其中只有trackgame1是文本正确的文档，其他都是我自己写的各种异常文件，主函数只读入trackgame的文件，当发现trackgame的异常之后，如果是我自己内部分析可以处理的错误，只会给出提示信息，否则会给用户端提示，让用户重新输入信息到trackgame1文件中，因为trackgame1的文本文档的信息时正确的，所以直接就可以进行各个功能的体验了。

下面我截一个过程图





在exception的package里面新建一个newfile的接口，里面有一个方法来进行重新输入文本的操作，然后新建一个newfile1的接口实现类,同时extends异常类，然后下面分别有各个需要通过重新读文本操作的各种异常的子类，最后在子类里面写构造器提示各种错误的问题根源所在，然后在父类里面进行newfile2的方法，根据里面的参数分别对三个应用重新读入问题，然后重新进行分析。

这里我分别通过应用来介绍：

首先介绍第一个应用场景：

在这里我对原来的代码做了修改，原来是通过将读入文本的信息用readfile方法进行返回一个文本的字符串，然后再使用analysis函数进行正则表达式的匹配工作，但是这样的问题在于：因为整个文本都是一个string，所以在其他情况无法匹配正则表达式 的语句出现的时候，就会无法分析是因为什么原因而导致的问题，也无法获取不匹配正则表达式的语句，我曾经尝试过使用正则表达式的部分匹配的功能，但是具体实现起来的额难度还是太大了，所以我变成使用arraylist把文本的每一行当做一个string，然后遍历list进行对每一句的string进行分析。

分析的过程如下：

首先先用三个正则表达式分别对给定的字符串进行匹配，并用visited来标识是否匹配成功，如果匹配成功的话，直接函数结束就行了，匹配失败的话，即visited=false，这样使用，的分隔符对字符串进行分割，得到的数组的length如果是5，那么进入运动员的分析，否则进一步进行分析。

在运动员的额分析过程中，使用<分隔符对数组0位进行分解，如果前面的字符串和运动员应该有的字符串是相同的话，那么确定是运动员的，否则抛出异常，标签非法。

然后对使用，分隔符获得的数组进行逐个数据的分析，对名字进行检查看有没有一样名字的运动员，对号码和年龄进行检查是不是纯数字，对国家进行检查是不是三个大写的字母，其他情况直接报异常，对于成绩的判断是通过先使用分隔符.将数据分开，看是不是两位小数，如果不是的话，判断小数的位数，如果小于两位，自动补零，如果是大于等于三位的话，将后面的几位去除掉。

在进一步分析的过程中，将：：作为分隔符判断前面的标签是属于哪个类的，如果找不到的话，则抛出非法标签的异常。在判断game的过程中，首先判断是不是在length部分有非法字符的存在，如果判断没有非法字符，都是纯数字的话，可以判断是所给的length不是给定的几个数值，这时候对具体的数值进行判断和比较，小于100的划分为100米跑，100-200米被划分为200米跑，200-400米被划分为400米跑，其他情况被划分为800米跑，然后重新构造字符串，进行正则表达式的匹配。在判断tracknum的过程中，判断判断是不是纯数字，检测是否有非法字符的存在。

下面分析第二个应用场景：

同样和上面的应用场景一样，对文本的每一行进行分析。对于这个应用场景，首先先遍历一遍整个文本的文件，对tracknum的正则表达式进行匹配，确定整个文本中有强调tracknum的信息，其中在find的循环里面进行tracknum++的操作，进行确定文本中的tracknum语句的个数，如果tracknum==0或者大于1的话，就会分别抛出缺少tag和sametag的异常。

在确定trackNum之后，再进行一个文本文件的遍历，重新匹配另外的两个正则表达式的语句，如果还是匹配到了上面的tracknum那一行，选择直接跳过当前行，直接进行下一行文本文件的分析。在匹配numberofElectron那一行的时候，进行判断是否存在该标签，如果没有该标签的话，抛出异常存在缺少异常的情况，接下来再匹配这一行数据的正则表达式的格式，对每一个语句匹配得到的信息进行分析，如果该电子层数所在的轨道数超过了总轨道数，则该轨道不存在，这样也会抛出异常显示依赖关系出现了错误。

如果每行的文本在上面的有匹配成功则visited=true，在analysis的最后进行判断，如果visited=false的话，对该行读入的文本进行分析，使用 ：：<作为分隔符分割字符串，首先判断该字符串是否含有该分隔符，不存在的话抛出异常格式错误，存在的话，判断前面的标签是否和前面的三个标签相同，不相同的话抛出异常显示非法标签的存在。

然后分别对各个表达式的数据进行分析：

对ELementName判断该表达式的元素是否是第一个字母大写，第二个字母小写，如果不是抛出异常显示是格式问题

用分隔符判断numberofElectron这一行的数据是否是两个整数，如果不是抛出异常显示输入的数据非法格式

接下来对第三个应用进行分析：

和上面的两个应用一样，我们首先对文本的每一行的信息进行判断，先对他们进行三个正则表达式的匹配，如果他们有正则表达式可以匹配上的话，设置visited=true，在中心点语句匹配上之后进行centnum++，随后在每一行语句进行中心点的匹配之后，进行centnum的判断，如果centnum==2，抛出异常sametag，值得一提的是在SocialTie的正则表达式的匹配过程中需要对数据进行分析，如果socialtie中的两个account中有一个并不在allfriends这个集合中，抛出异常显示应用三有关系没有对应该人名，或者有friends标签放在了socialtie之后。

如果最后visited==false，和之前一样使用::<的分隔符对字符串进行分割，然后对前面的tag进行比较，如果不是规定的三个tag的话，抛出异常显示非法标签的存在。下面对三个tag分别进行分析：

对于centrualuser该标签的参数分别进行判断，判断第一个参数是否是由字符串组成的，第二个参数是否是由纯数字组成的，判断第三个参数是否是由M或者F。

对于friend标签判断第二个参数是否是纯数字组成的，判断第三个参数是否是由M或者F

对于socialtie标签判断前面两个用户是否已经存在于friends的集合之中，判断第三个参数两个用户之间的权重是否是由三位小数构成的。这里如果不是三位小数，但是是小数的话，进行判断，如果是小于三位小数，进行自动补零的操作，然后重新进行正则表达式的匹配，如果是大于三位小数的情况的话，将大于三位之后的数字进行舍去，然后重新进行正则表达式的匹配。

## Assertion and Defensive Programming

### checkRep()检查invariants

在第一个应用在每一次操作的时候首先检查运动员的集合的size是否小于等于整个轨道的个数，如果不是的话，抛出异常

在第二个应用检查tracknum具体是否大于0

在第三个应用检查每一个用户在哪个轨道和该用户和中心点距离的是否相同

### Assertion保障pre-/post-condition

第一个应用

public void setinfoall(ArrayList<String> info)

确定传入的参数是arraylist类型的参数

第二个应用确定setinfoall 前置条件确定传入参数是string类型

public void transitElectron(int source,int target)

确定传入参数是两个int类型

public AtomStructure(String path)

确定传入参数是string类型

在第三个应用

public SocialNetworkCircle(String path)

检查传入的参数是否是string类型的

public final void setinfoall(ArrayList<String> info)

检查传入参数是不是string类型的

public int getLogicalDistance(String a,String b)

public void removeEdge(String a, String b)

## Logging

这里使用log4j的外部库进行实现，

首先将log4j.jar包加入工程中，这个时候，如果我们进行run as application，会得到如下提示:

log4j:WARN No appenders could be found for logger (com.foo).

log4j:WARN Please initialize the log4j system properly.

log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.

因为程序找不到log4j.properties这一配置文件。所以，需要把这个配置文件添加到当前的工程中了。

在当前工程上右键。new->folder,在弹出的对话框中的”folder name”输入一个文件夹的名字，这里用log4j。然后点击”advanced”,选中”link to folder in the file system”,“browse”一个目录，选择src的文件夹。这样才建立的一个关联的目录。接着在package explorer中找到刚刚建立的文件log4j文件夹，import你定义好的log4j.properties(import的过程省了)，下面这个可做参考:

log4j.rootLogger=DEBUG, R, O

log4j.appender.O=org.apache.log4j.ConsoleAppender

log4j.appender.R=org.apache.log4j.RollingFileAppender

log4j.appender.R.File=log4j.log

//Control the maximum log file size

log4j.appender.R.MaxFileSize=100KB

//Archive log files (one backup file here)

log4j.appender.R.MaxBackupIndex=1

log4j.appender.R.layout=org.apache.log4j.PatternLayout

log4j.appender.O.layout=org.apache.log4j.PatternLayout

log4j.appender.R.layout.ConversionPattern=[%d{ISO8601}]%5p%6.6r[%t]%x - %C.%M(%F:%L) - %m%n

log4j.appender.O.layout.ConversionPattern=[%d{ISO8601}]%5p%6.6r[%t]%x - %C.%M(%F:%L) - %m%n

在之前package()单击右键，run as->run configuration，然后在弹出的页面的tab页中选择Classpath,选中当前的工程，然后发现右侧的”Advanced”进入可选状态，点击以后，会接着弹出一个对话框，选择”Add folders”，找到当前的工程，选择log4j文件夹，然后确定。这样，我们就把log4j.properties添加到当前工程的class path中了。选择run。

到此，就可以在eclipse里面边写代码边调试了。

在文件中有这么一行代码

log4j.appender.R.File=log4j.log

这的意思是输出到文件中

因此，这个程序在当前工程下还存在一个 log4j.log的文件。

### 写日志

import org.apache.log4j.Logger;

import org.apache.log4j.Level;

在每一个要添加logger的异常类上面import，然后在分析文件进行正则表达式的方法上面写

Logger logger = Logger.getLogger(X .class);

这里的X是该代码所在的类名字

logger.setLevel(Level.INFO);

然后使用logger.error() 在throw new 错误的地方，添加错误提示

Logger.info（）在关键操作的地方，添加信息

下面是以部分的log文件内容

[2019-05-15 21:52:44,885] INFO 0[main] - applications.Main.main(Main.java:32) - 选择trackgame应用

[2019-05-15 21:52:44,963] INFO 78[main] - applications.Main.main(Main.java:36) - 读取trackgame.txt的信息

[2019-05-15 21:52:44,963] INFO 78[main] - applications.Main.main(Main.java:40) - 正在对读入文本进行正则表达式解析！

[2019-05-15 21:52:44,980]ERROR 95[main] - circularOrbit.TrackGame.Analysis(TrackGame.java:377) - 文本中存在非法标签，重新输入文件

[2019-05-15 21:58:58,647] INFO 0[main] - applications.Main.main(Main.java:32) - 选择trackgame应用

[2019-05-15 21:58:58,679] INFO 32[main] - applications.Main.main(Main.java:36) - 读取trackgame.txt的信息

[2019-05-15 21:58:58,679] INFO 32[main] - applications.Main.main(Main.java:40) - 正在对读入文本进行正则表达式解析！

[2019-05-15 21:58:58,694]ERROR 47[main] - circularOrbit.TrackGame.Analysis(TrackGame.java:377) - 文本中存在非法标签，重新输入文件

### 日志查询

在main函数的每个菜单中添加搜索logger的功能，然后再单独输出一个查询logger的菜单。

System.***out***.println("1.按时间查询");

System.***out***.println("2.按类型查询");

System.***out***.println("3.按类查询");

System.***out***.println("4.按方法查询");

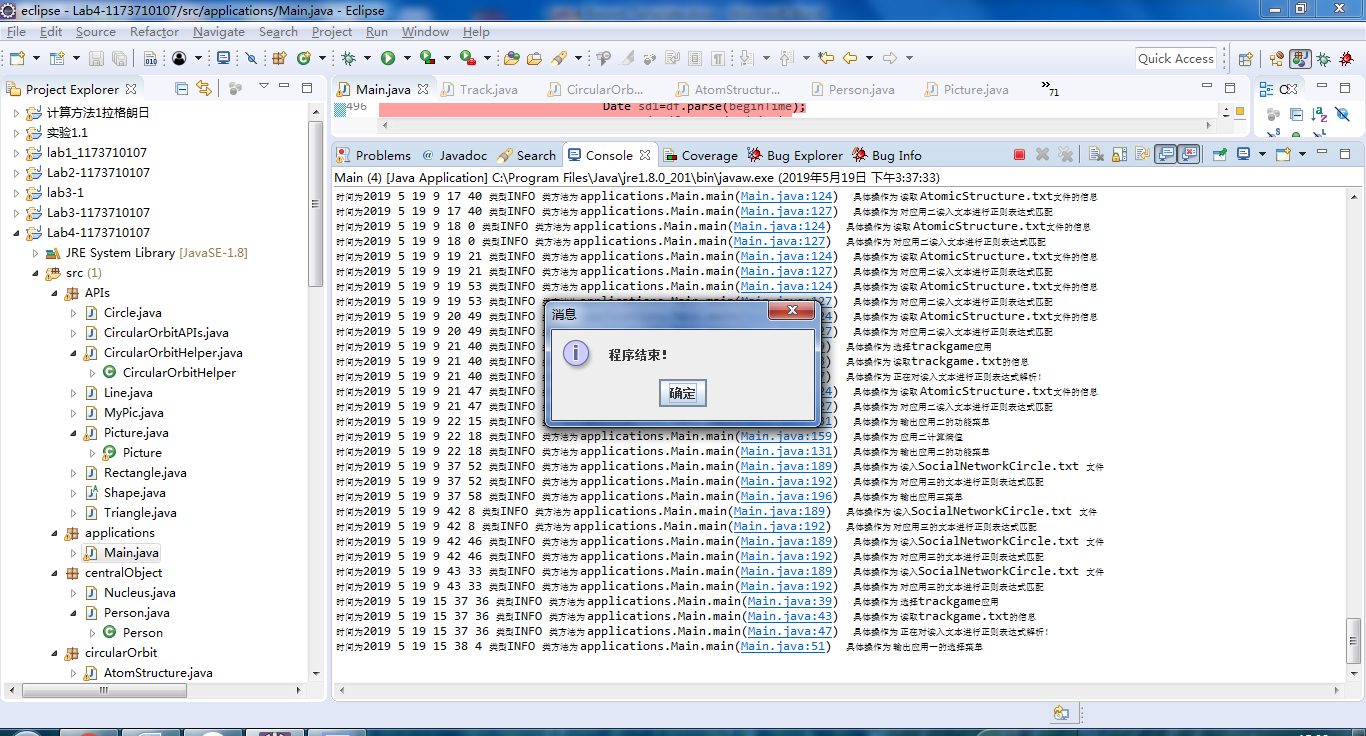
System.***out***.println("5.按操作类型查询");

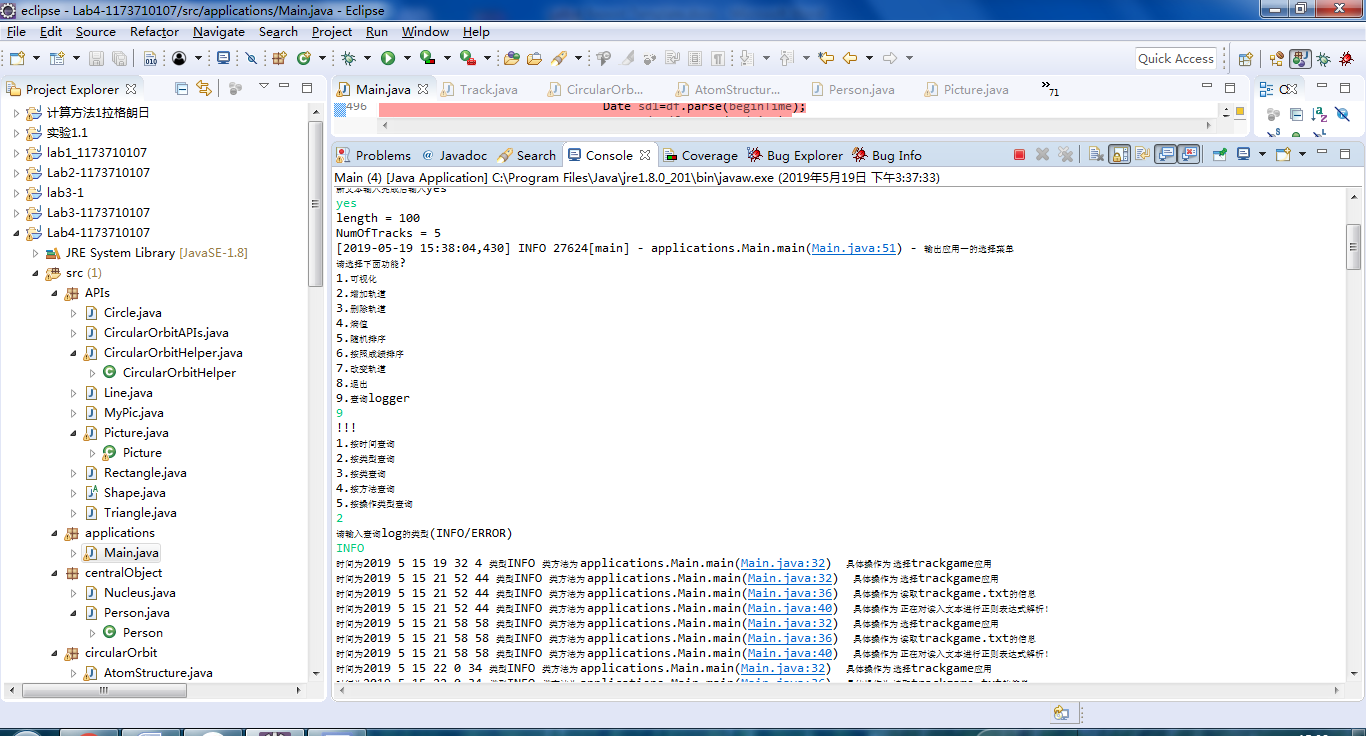
然后通过读入信息，判断用户需要去查询哪一个信息，通过switch语句去选择性的要求用户去输入需要查询的信息，然后通过遍历文本信息，进行逐行读取，然后把这一行的文本信息进行分割，获取具体各个信息，然后通过switch语句选择对应的的功能，然后将用户输入的信息和文本分析的信息进行比较，符合条件的话，输出之前获取的各个部分的信息，否则跳过，继续读入分析下一行文本。

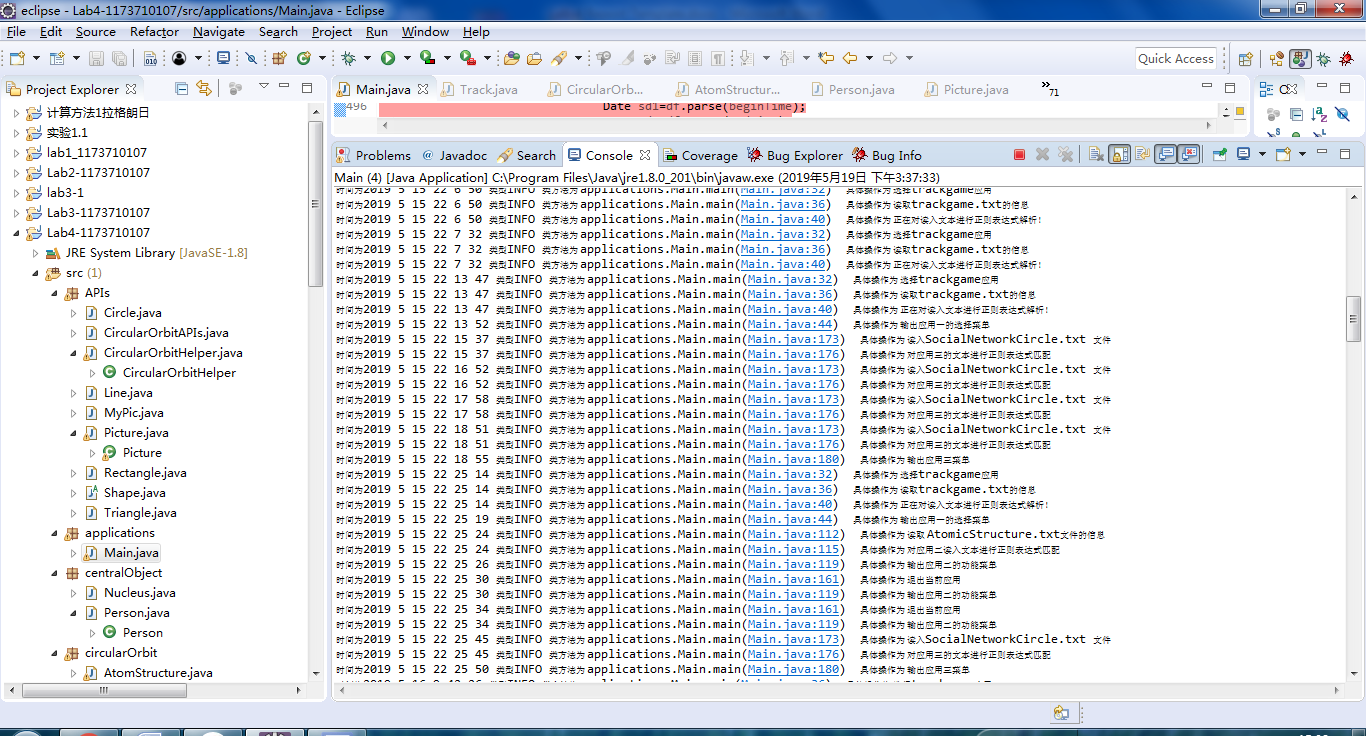
需要注意的是我之前想通过直接分别获取年月日时分秒，然后比较，但是发现代码比较难写，代码量比较大，所以直接用SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");，然后用Date自己带的before或者after来进行比较。

下面截几个图

：







里面的logger都是之前留下来的

## Testing for Robustness and Correctness

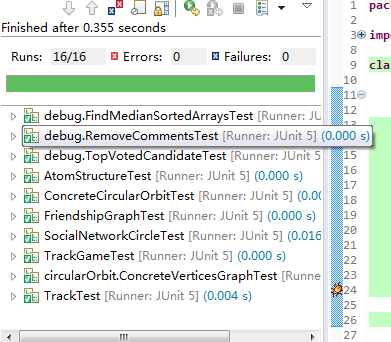
### Testing strategy

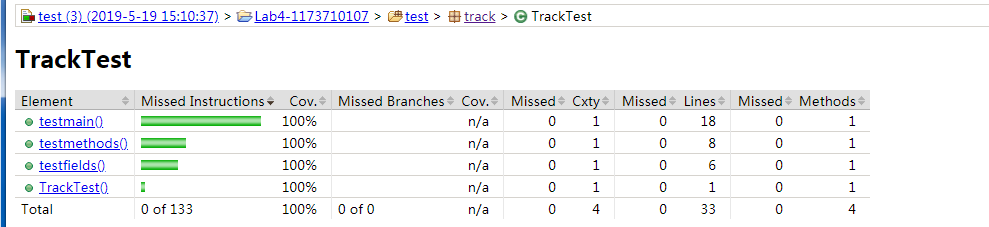
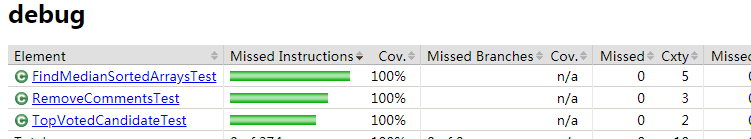
API这部分的可视化GUI不需要去做测试，main函数也不用去写，然后中心物体和轨道上的物体类也没什么写的，这里就写circularOrbit，debug和track’就可以了

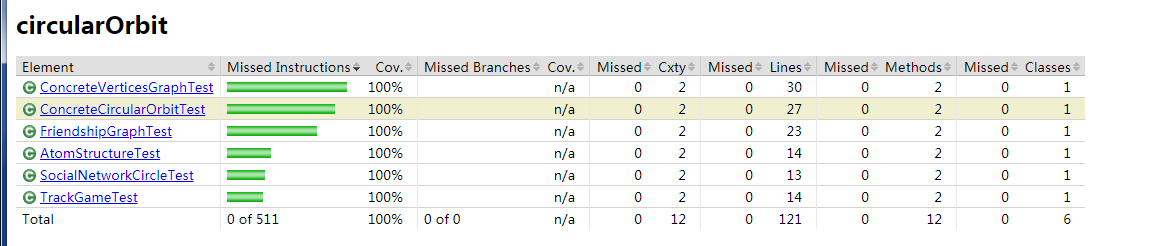
### 测试用例设计

对需要、test的类生成junit测试，然后尽量对每一个该方法都调用到，然后使用assertequals函数对预期的结果进行测试

### 测试运行结果与EclEmma覆盖度报告





上传到github了

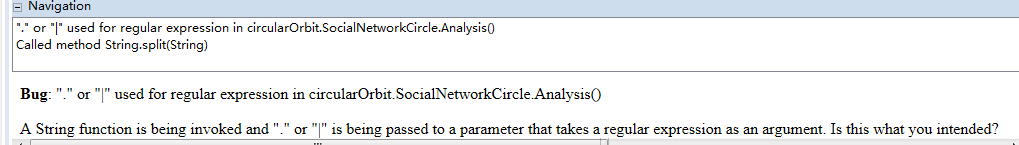
## SpotBugs tool

发现了哪些错误，每种错误代表什么不良的编程习惯

对代码修改，消除这些错误。

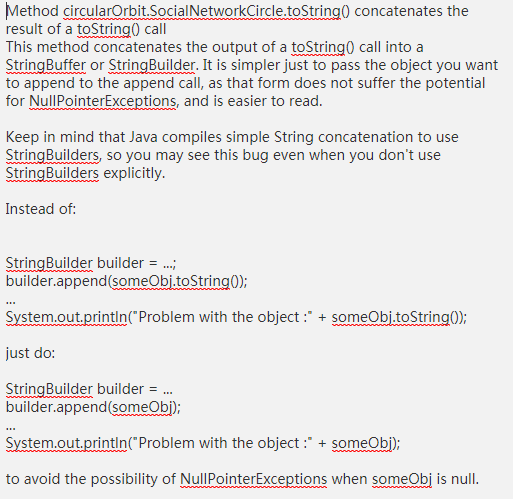
错误一：





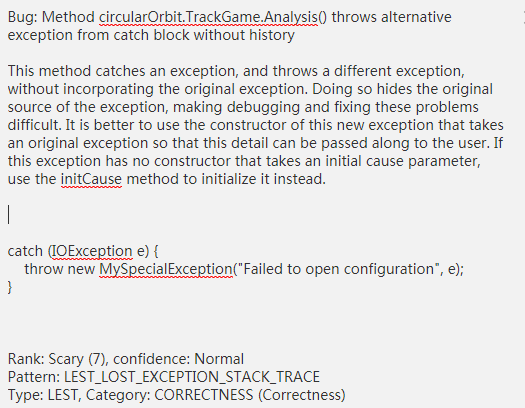
符号没\\，加上[\\就可以了](file:///\\就可以了)

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

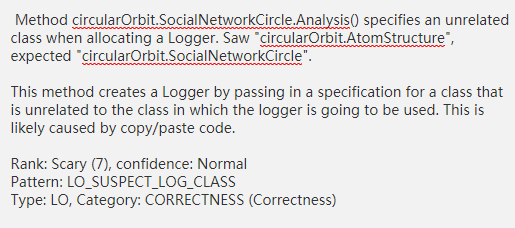
错误三：



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

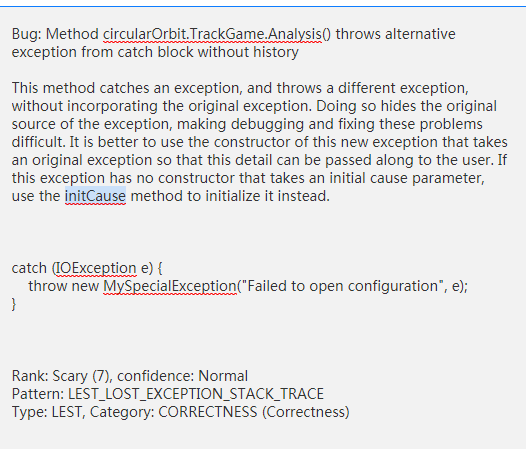
错误四：





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>

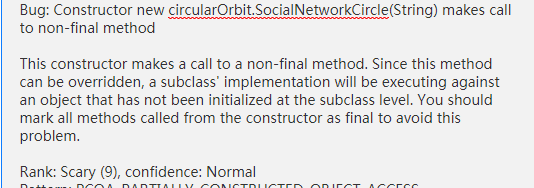
我看了这个资料，大概知道是要进行异常的包装，然后方便找到根源的问题，但是我这里感觉并没有出现所说的捕获异常A，然后抛出异常B的问题啊。。。

我尽力了！！！！

。。。。。

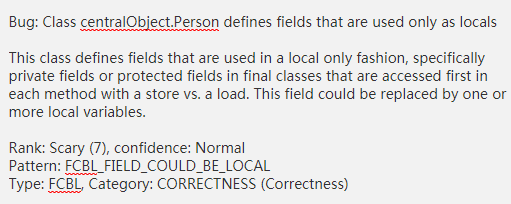
突然又好了，又不报错了，我啥也没改，为了防止到时候检查出来，我还是先写报告里面了

错误六：



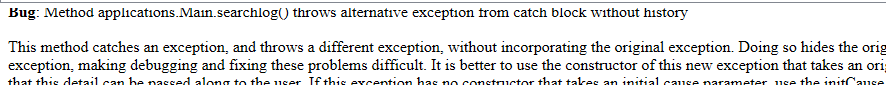
方法加final就可以了

错误七：



删掉private就可以了

错误八：

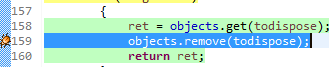




这个就是上面我说的那个，用initcause就可以了

错误九：





这个没办法改

因为这个方法删除的话就要返回删除的东西，那我不get怎么返回要删除的东西。。。。

## Debugging

代码一：find

代码二：remove

代码三：top

### 理解待调试程序的代码思想

代码一：这段代码是要输入两个数组，然后求出这些数中的中位数

代码二：这段代码是要通过输入的一个数组，然后通过函数的处理，把注释的语句去掉，然后返回剩下的代码

代码三：这段代码用于计算某个时间的投票，看看哪个候选人将会获胜，然后如果票数一样多的话，先得票的获得胜利

### 发现并定位错误的过程

代码一：自己那两个数组，然后自己一步一步通过理想与现实的误差/xyx，判断代码错误/xk

代码二：通过将一段小程序，自己手动运行每一行代码

代码三：首先要看明白要干什么，然后再看清楚每一个的方法具体要实现什么，分开看

### 如何修正错误

代码一：int i = (iMin + iMax + 1) / 2; 去掉+1

B[j - 1] > A[i] 去掉-1

(m + n + 1) % 2 == 1) 去掉+1或者后面改成0

代码二：

if (!inBlock && i+1 <= line.length() && chars[i] == '/' && chars[i+1] == '\*') { 把第一个条件删除

if (inBlock && newline.length() > 0) 把inblock变成！

else if (inBlock && i+1 < line.length() && chars[i] == '\*' && chars[i+1] == '/')

条件下添加i++;

代码三：

int c = count.getOrDefault(p, 1); 1改成0 然后c++

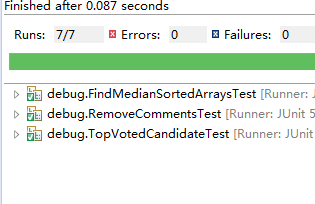
因为q方法是返回在T时间选举成功选举的候选人的编号，所以

hi = A.size() size需要-1

hi = A.get(i).size();需要-1

具体函数里面用二分计算

### 结果



# 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 |
| 5.12 | 下午加晚上 | 整明白怎么写，然后写完4.1 | 是整明白怎么写异常类了，但是只写完第一个应用的异常抛出，异常还没有处理 |
| 5.13 | 整天，包括上课 | 写完4.1 | 写完了 |
| 5.14 | 整天，除了英语课 | 写完4.2 4.3 | 4.3没写完，只写了logger的添加，没写查询 |
| 5.15 | 整天 | 写完4.3，然后写debug | Debug就写了一个。。。 |
| 5.16 | 整天 | 上课写debug，然后飞机上写测试用例（去图灵大会） | Debug写完了，飞机上没写 |
| 5.18 | 晚上 | 写测试用例 | 写了，贼垃圾。。 |
| 5.19 | 晚上 | 写报告 | End |
|  |  |  |  |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 在进行异常处理的过程中，运行时出现下面的异常    让我很疑惑，我明明还没有输入信息，结果就出现了显示scanner没有读入信息的问题  就像是有人执行到这一步自动帮我执行了回车一样 | 网上找到原因：  导致这个异常的原因是因为没有读取到数据，而为什么会没有读取到数据呢？通过close（）这个方法可以看到一旦scanner被close，再次唤醒scanner时没有用处的，所以建议在项目中，scanner类修饰为static的一个工具类，起到一个全局变量的作用，最后在项目关闭的时候回调一个方法，这个时候才调用scanner.close（）这个方法 |
|  | 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。。。  然后自己好了 |
|  |  |

# 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

尽管这次的实验比上次简单多了，但是因为我个人原因，时间安排出现问题，所以写的不咋地，以后会注意

## 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？

首先要正确，才能进而要求健壮性

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？）

民航飞机不装降落伞是因为降落也需要一些技术，不是说有降落伞就能活下来的，但是这个自己修改了错误的代码时确实可以解决问题的。

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？
2. 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？
3. 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？
4. 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？

我觉得充分的测试不是测试多少代码行数，而是测试了多少主要的线程

1. Debug一个错误的程序，有乐趣吗？

第一次用那个spotbugs，感觉还挺有意思的，自己也找了很多资料去了解错误根源

1. 关于本实验的工作量、难度、deadline。

我的时间有点不够，但是还是不算很难的

1. 到目前为止你对《软件构造》课程的评价和建议。

Good。，就是学不明白，还要慢慢自己静下来钻研

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。

和之前有的专业课一样，一纸开卷，我觉得真的挺好的，有的要考的知识点不需要一定记住，只知道就可以了，但是不一直开卷的话，就需要去自己从0开始完全知道了，这样当然学的更好，但是这样的话可能我们复习的重点会偏移。