

**2018年春季学期  
计算机学院大二软件构造课程**

**Lab 4实验报告**

|  |  |
| --- | --- |
| 姓名 | 穆添愉 |
| 学号 | 1160301008 |
| 班号 | 1603010 |
| 电子邮件 | [1417553133@qq.com](mailto:1417553133@qq.com) |
| 手机号码 | 15636094072 |

**目录**

[1 实验目标概述 1](#_Toc1900787419)

[2 实验环境配置 1](#_Toc1180222618)

[3 实验过程 2](#_Toc1904163637)

[3.1 Error and Exception Handling 2](#_Toc223646104)

[3.1.1 针对输入文本文件的异常/错误处理 3](#_Toc1767821510)

[3.1.2 针对输入图操作指令的异常/错误处理（可选） 6](#_Toc1057883075)

[3.2 Assertion and Defensive Programming 10](#_Toc65877188)

[3.2.1 checkRep()检查invariants 10](#_Toc864581994)

[3.2.2 Assertion保障pre-/post-condition 11](#_Toc811938435)

[3.3 Logging 12](#_Toc1944791666)

[3.3.1 写日志 12](#_Toc761568248)

[3.3.2 日志查询 13](#_Toc553459158)

[3.4 Testing for Robustness and Correctness 16](#_Toc1900938799)

[3.4.1 Testing strategy 16](#_Toc1729810613)

[3.4.2 测试用例设计 16](#_Toc1570358083)

[3.4.3 测试运行结果与覆盖度报告 20](#_Toc1020367898)

[3.5 FindBugs tool（可选） 22](#_Toc747807859)

[3.6 Debugging 24](#_Toc977229344)

[3.6.1 待调试程序 24](#_Toc624027122)

[3.6.2 理解待调试程序的过程 24](#_Toc559194996)

[3.6.3 发现并定位错误的过程 24](#_Toc1536498463)

[3.6.4 如何修正错误 27](#_Toc712176281)

[3.6.5 结果 27](#_Toc1058439024)

[4 实验进度记录 29](#_Toc589889823)

[5 实验过程中遇到的困难与解决途径 29](#_Toc1788520797)

[6 实验过程中收获的经验、教训、感想 30](#_Toc1254173498)

# 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(Graph<L,E>)代码和基于该 ADT 的四个应用(GraphPoet、社交网络、计算机网络拓扑、电影网络)的代码,使用以下技术进行改造,提高其健壮性和正确性

 (1)错误处理

 (2)异常处理

 (3)Assertion 和防御式编程

 (4)日志

 (5)调试技术

 (6)黑盒测试及代码覆盖度

# 实验环境配置

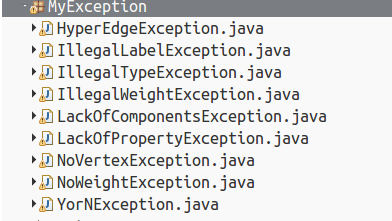
https://github.com/ComputerScienceHIT/Lab4-1160301008

在这里给出你的GitHub Lab4仓库的URL地址（Lab4-学号）。

# 实验过程

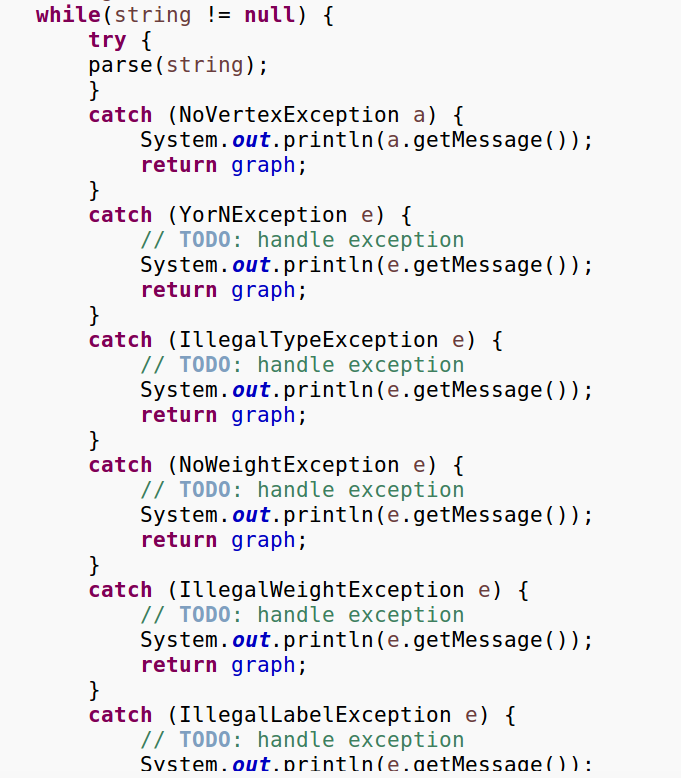
## Error and Exception Handling

针对于所有的异常，有如下的目录



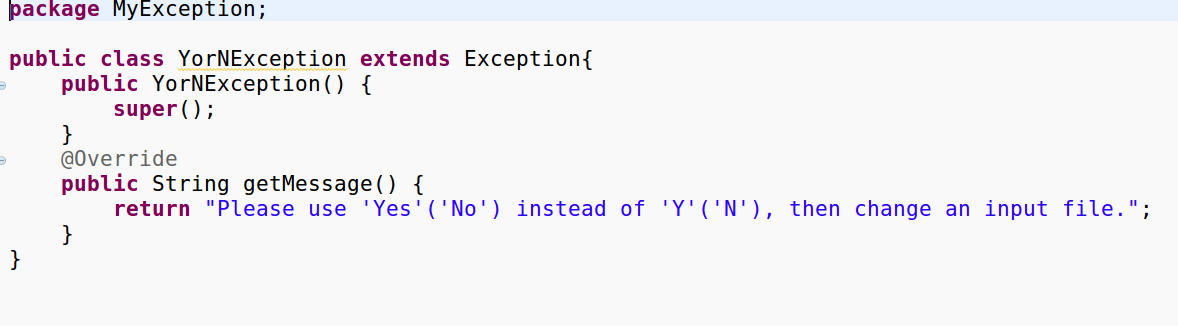
这是我写的一些自定义异常，都是一些需要抛出的异常

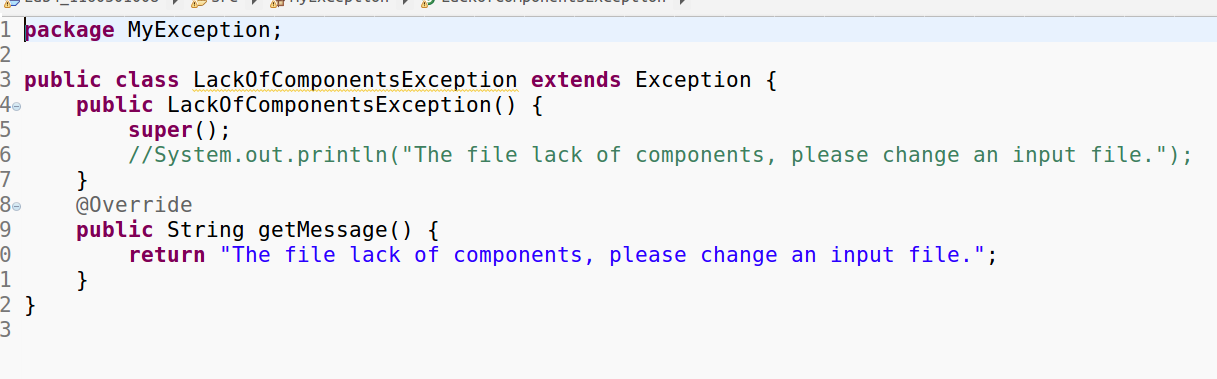
捕捉异常处理：



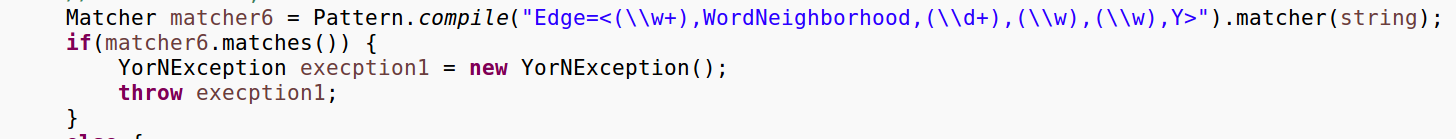
### 针对输入文本文件的异常/错误处理

1. 存在不合法语句，比如说缺少分量，用“N”来代替“No”，有如下的两个异常：



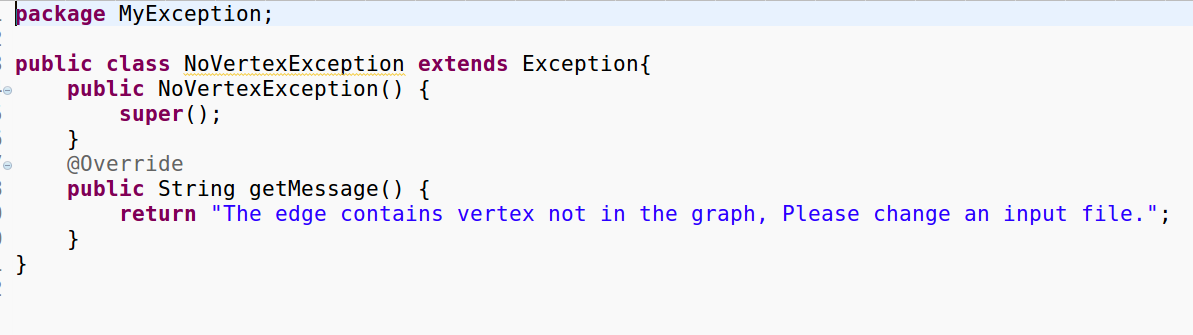


下面是代码中抛出异常的部分：

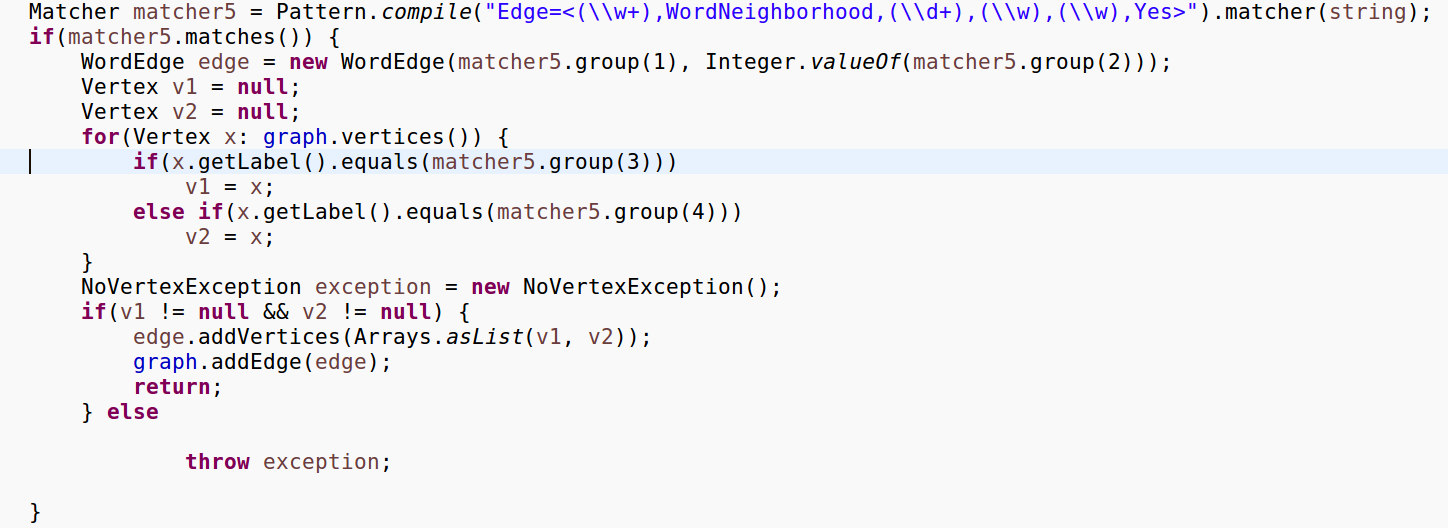


可见，当我们输入Y而不是Yes时，会抛出异常，捕捉到之后，会打印异常信息，从而实现自定义异常处理。

1. 边中使用的节点未定义

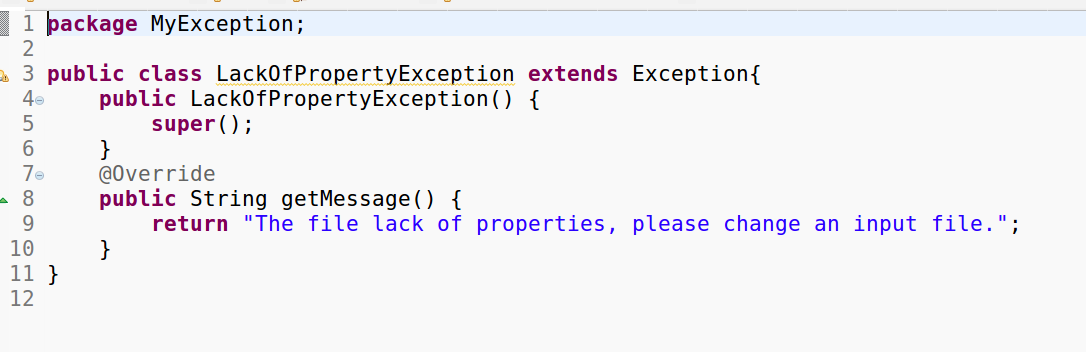


同样的方法来抛出异常：

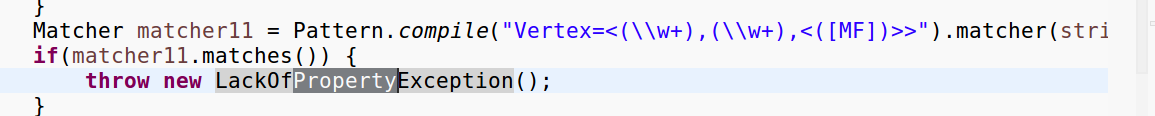


在加边的时候，去遍历顶点集来寻找边中存在的点，如果两个都能找到，那么进行加边操作，否则就抛出异常。

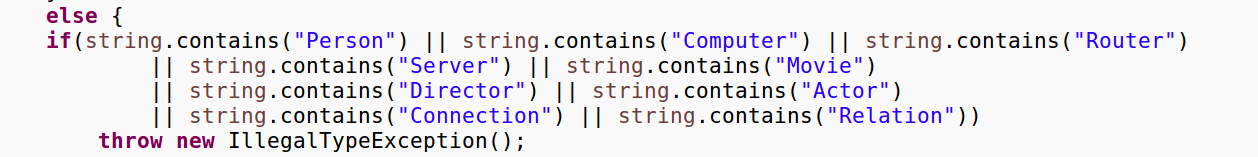
1. 为节点定义的属性的数目与特定应用的图的约束不符



同样的方法去进行异常的抛出，当输入的字符串缺少属性的时候，就进行异常的抛出。



1. 出现了不应出现的节点类型和边的类型，与上述一样，这个我没有设置正则表达式去进行匹配，因为可能出现太多种不该出现的边和节点类型，所以最后我是如下图来进行异常的抛出：



直接看字符串中是不是含有其他种类节点类型（Person，Actor等）和边的类型（Connection等），以此来进行判断，如果含有这些，那么就进行抛出，并打印异常信息。

（5）无向图中出现了有向边，那么就将方向去掉，正常执行加边操作，**这里我觉得这个异常有一点没有必要，因为首先已经不能存在不符合规定的其他类型的边了，也就是说在GraphPoet中只能出现WordNeighbor这一种边，而WordNeighbor已经确定是有向边，那么就不可能同时出现有向边和无向边，所以我觉得这个异常有一点牵强。**

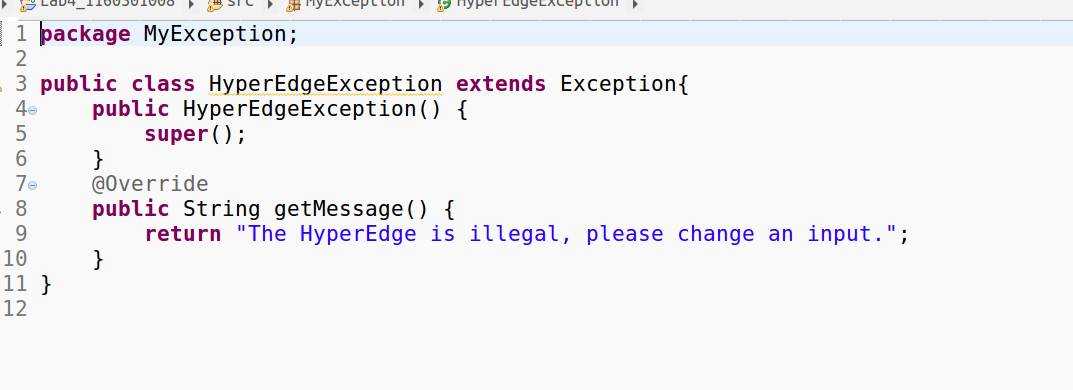
（6）单重图中出现了多重边，只需要在这两个点之间先去进行判断是不是已经存在边了，如果存在的话，忽略掉这条加边的指令，continue。

（7）不该出现边的节点之间连了边，这个由于情况比较少，可以直接根据具体节点来进行判断，例如Director和Actor之间出现了边。

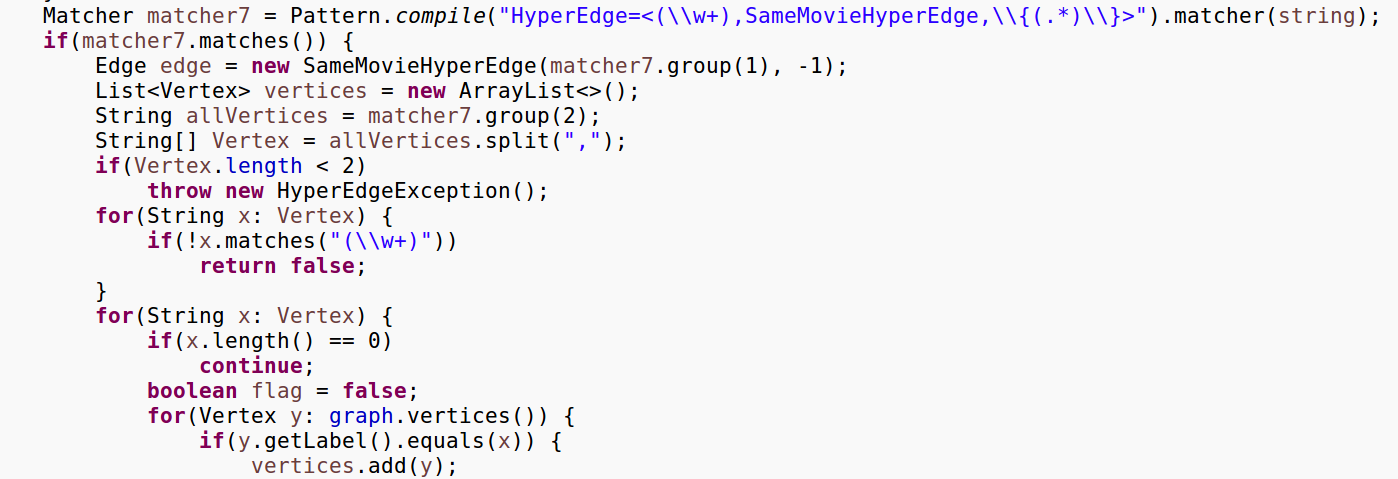
（8）不该出现loop的图中出现了loop，直接进行判断，如果出现了loop，跳过就好。

（9）不该出现超边的图中出现了超边，**这个异常我也觉得有一点多余，因为之前已经对于不该在图中出现的边抛出了异常，所以超边也可以进行异常的抛出。**

（10）超边中节点数小于2，有如下的自定义异常



一旦超边中的节点数小于2，就会抛出异常。



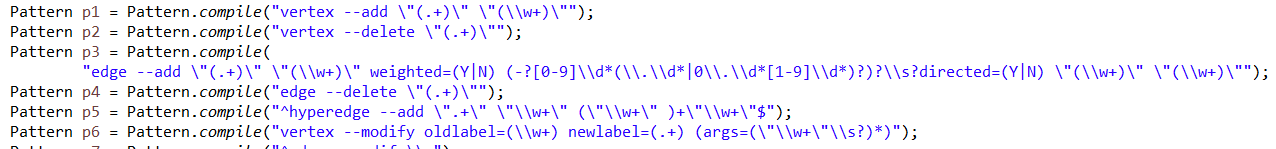
1. 带权边没有给出权重

像上面一样进行正则表达式的匹配，并且进行异常的抛出就好。

1. 权重不符合要求，这里主要选取权重小于0的情况，来进行正则匹配，抛出异常。
2. label不满足（\w+）的情况，先对label进行匹配，如果能匹配（.\*）之后，再去匹配（\w+），如果不能匹配，进行异常的抛出，也就意味着含有非法字符。
3. label重复，加边加点之前先遍历，不存在重复label的话，可以进行加边加点操作，否则忽略掉。

### 针对输入图操作指令的异常/错误处理（可选）

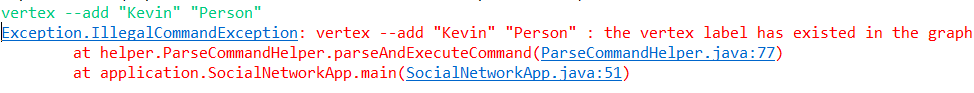
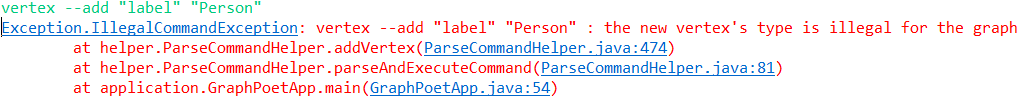
首先判断指令语法是否正确，如果不正确，直接抛出异常，允许用户输入新的指令。指令语法如下：



1. Vertex –add label type。

如果该语法匹配成功，则提取出label、type信息。首先判断label是否符合\w+或者输入的label已经在图中出现过，否则抛出异常，允许用户输入新的指令。根据不同类型的图，判断type对于该类图是否合法。



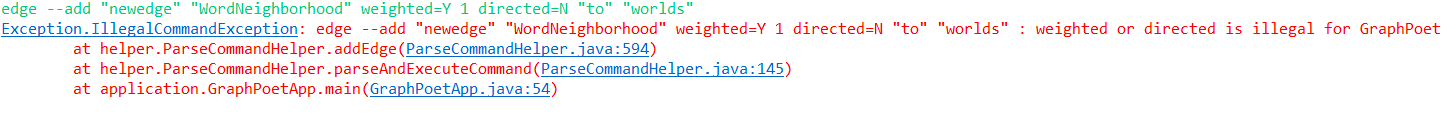


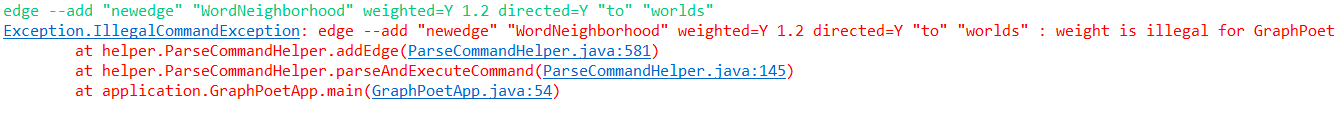
1. Vertex –delete regex

如果匹配成功该指令，则不需要抛出异常，只需要找到label符合该regex的点删除即可。

1. Edge –add label type weighted=Y|N weight directed=Y vertex1 vertex2

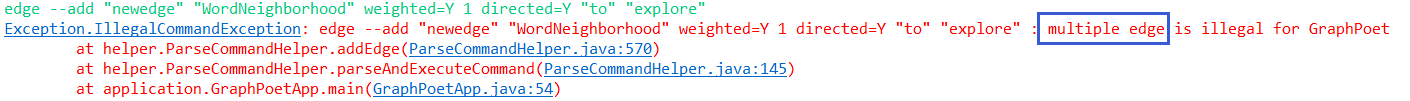
如果该语法匹配成功，提取出label，type，这两个属性的判断方法与vertex的一致。首先判断两个顶点是否都在图中。根据不同类型图或者不同类型边来判断weighted与directed是否合法。对于weighted=Y的边，需要判断所输入的weight对于该类型的边来说是否是合法的。如果是无权边，可以允许不输入权值，默认为-1，但是带权边必须手动输入权值。还需要判断是否在单重图中出现了多重边。对于不允许出现自环的图，如要判断新加的边是否为自环。对于无向边，判断是否给出了非零的权值。对于某些边，要判断顶点类型是否合法。如NetworkConnection要求两个顶点不能同时为Computer或Server.如果上述情况出现不合法的情况，均抛出异常，允许用户输入新的指令。

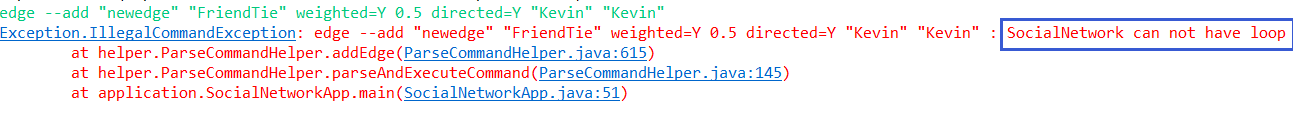


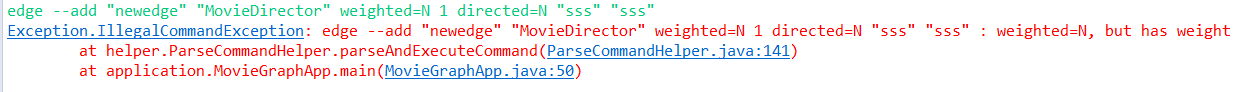


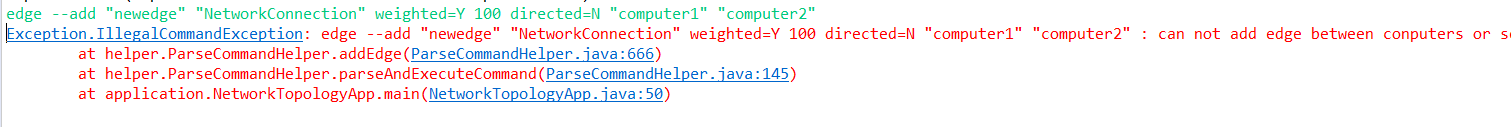


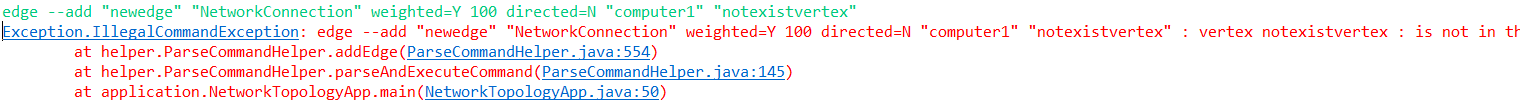


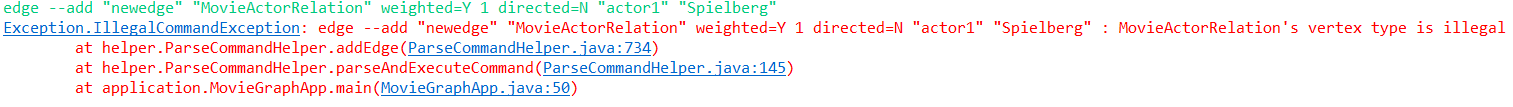










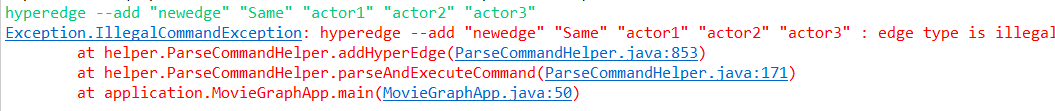


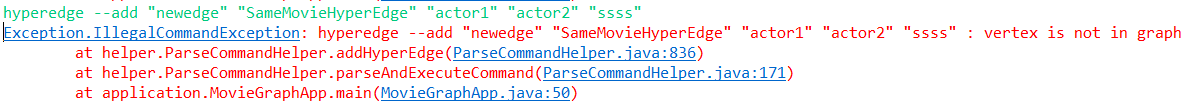
1. Edge –delete regex

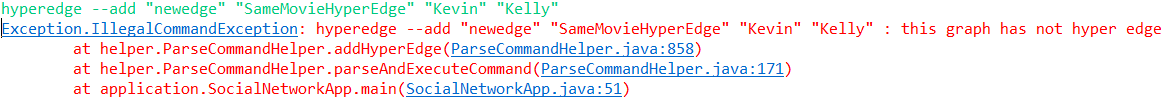
与 vertex –delete 相同，不需要抛出异常。

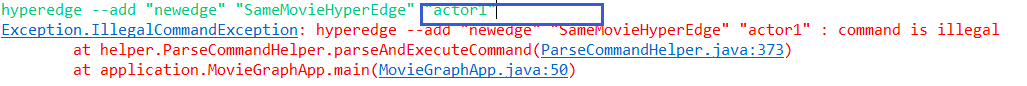
1. HyperEdge – add label type vertex1 vertex2……

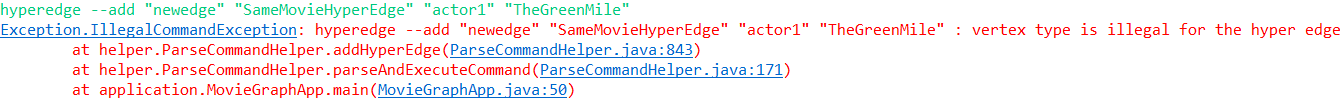
如果匹配该语句，首先判断label是否合法。然后判断该图中是否含有超边，对于GraphPoet、SocialNetwork、NetworkTopology不允许超边存在。如果MovieGraph，判断超边的类型是否正确。然后判断加入顶点的个数，是否大于等于2。如果上述合法，判断所加入的顶点是否都在图中，否则抛出异常。判断所加入的点的类型是否合法。最终判断所加入的一条超边上的顶点，是否出演同一个电影。如果图中已经有该label的超边，只需要将顶点加入到原有的超边上。





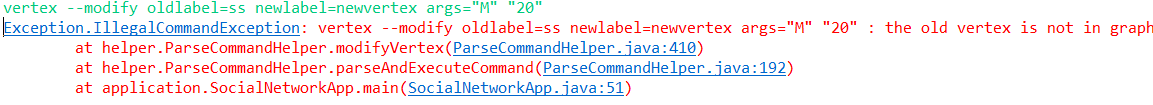






1. Vertex –modify oldlabel=xxxx newlabel=xxxx args=

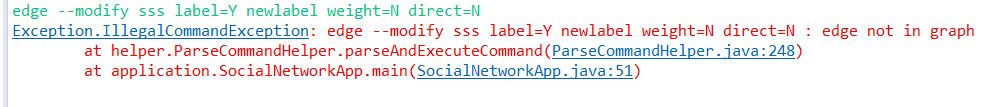
首先判断原label是否在图中，然后判断所给的newlabel是否符合\w+。两个label可以相同。根据不同类型的图来判断所给的属性是否合法。





1. Edge –modify oldlabel label=Y|N newlabel weight=Y|N newweight direct=Y|N

首先判断oldlabel是否在图中，否则抛出异常如果在label=Y或者weight=Y后面没有label或weight，抛出异常





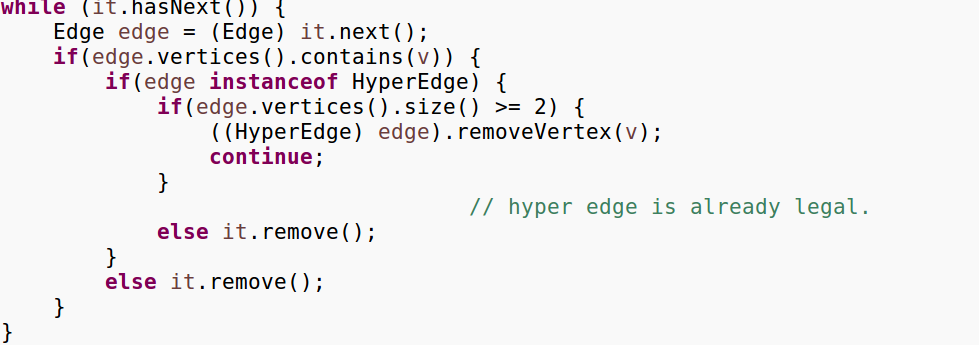
## Assertion and Defensive Programming

### checkRep()检查invariants

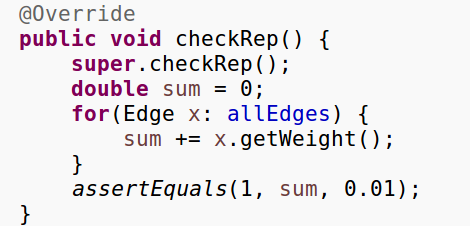
（1）在concreteGraph.java中有如下的checkRep（）：



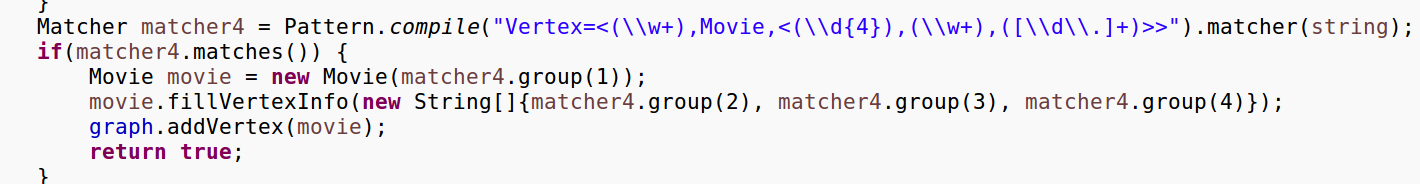
针对于超边，没有额外的去写checkRep（），因为我在进行加超边的时候已经进行了判断，一定要节点数大于2，



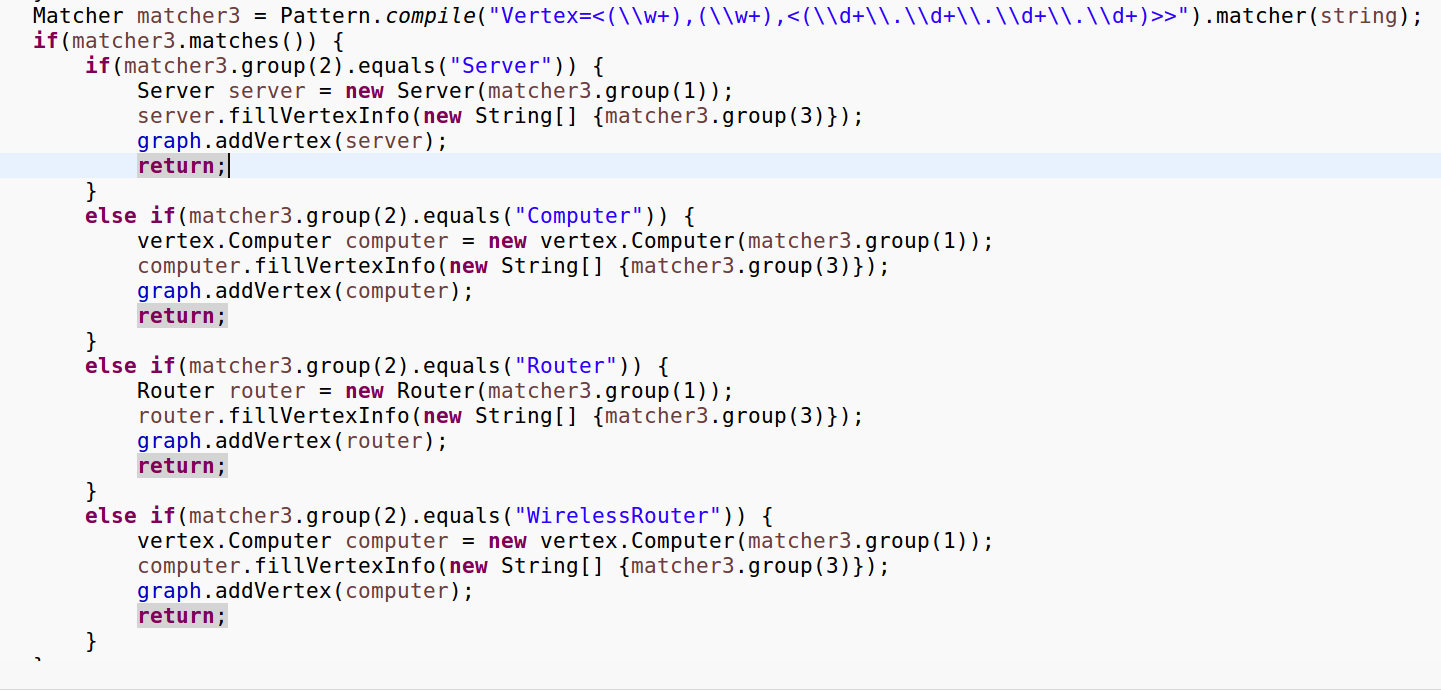
另外，针对于SocialNetwork，需要满足所有边权重之和为1，所以要重写checkRep（）如下：



1. 对于movie来说，要保证IMDB得分在0~10分之间，这个已经在正则表达式的地方进行限制，也就是限制一位小数，还有拍摄年份限制为四位整数

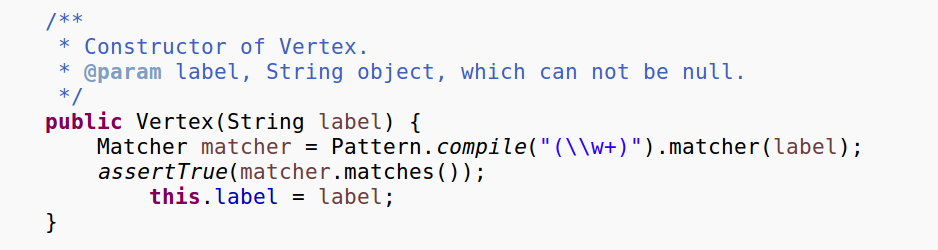


1. 对于网络拓扑图来说，要严格限制IP地址的格式，这个也已经在正则表达式中进行了限制：

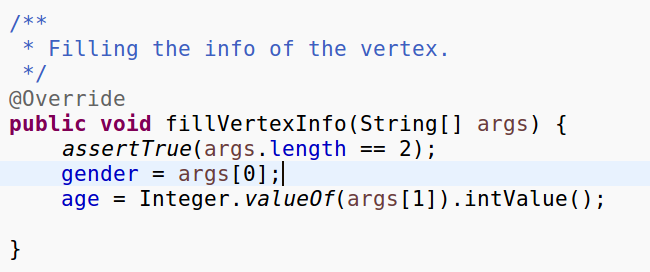


### Assertion保障pre-/post-condition

1. 针对于label，要保证所有的label都是（\w+），所以有如下的assert：



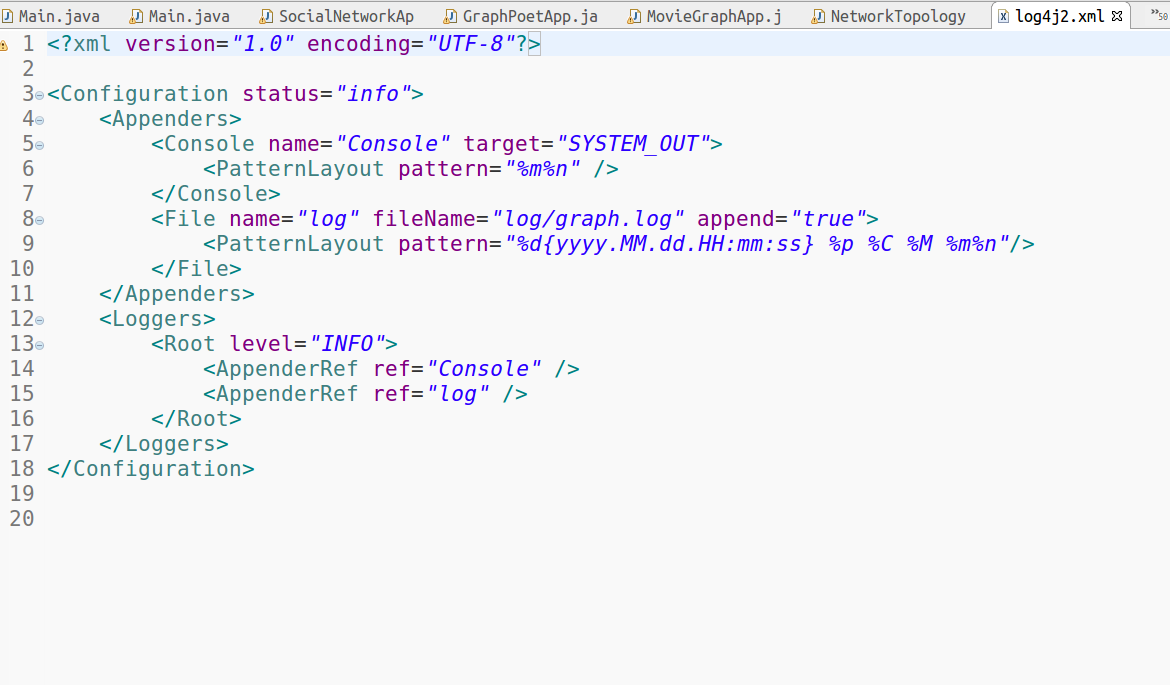
1. 要保障给节点传入信息的时候，数组的长度和要求属性的数量一致，这个就要每个类分别考虑了



## Logging

### 写日志

这里采用老师推荐的第三方库log4j，下载安装之后，有如下的配置文件：



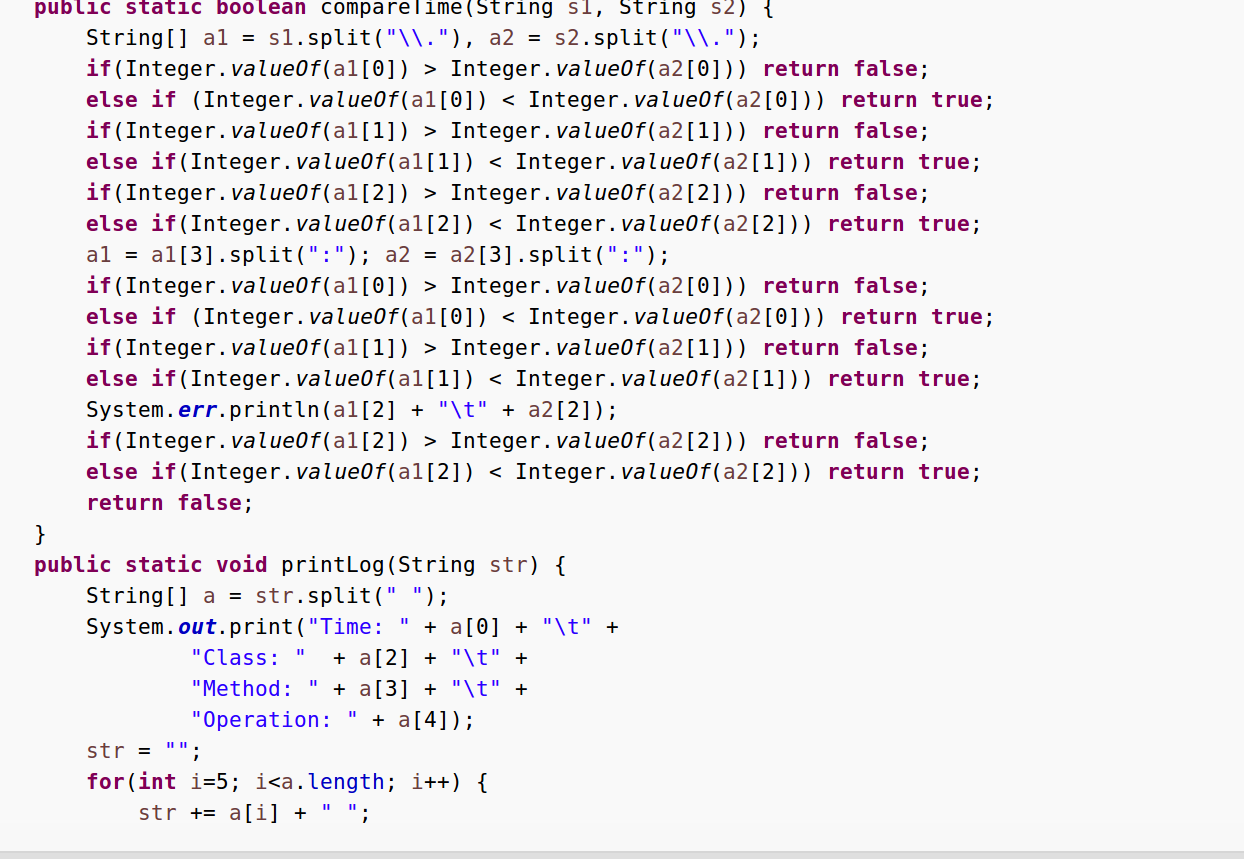
这里规定了日志的输出格式，以及输出的位置：src/graph/log.txt

在每个异常的位置和加边加点的位置我加入了记录日志，最终，日志效果如下：



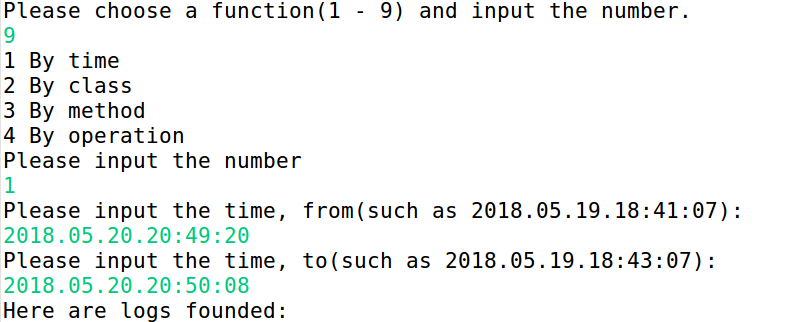
### 日志查询

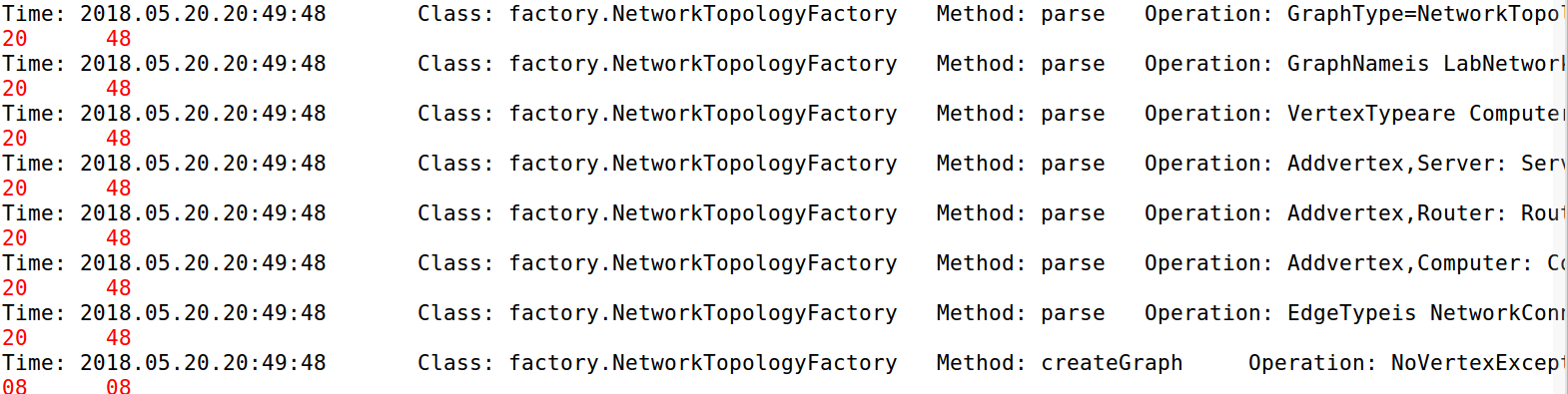
比较时间的部分比较麻烦，要考虑很多种情况，具体如下：



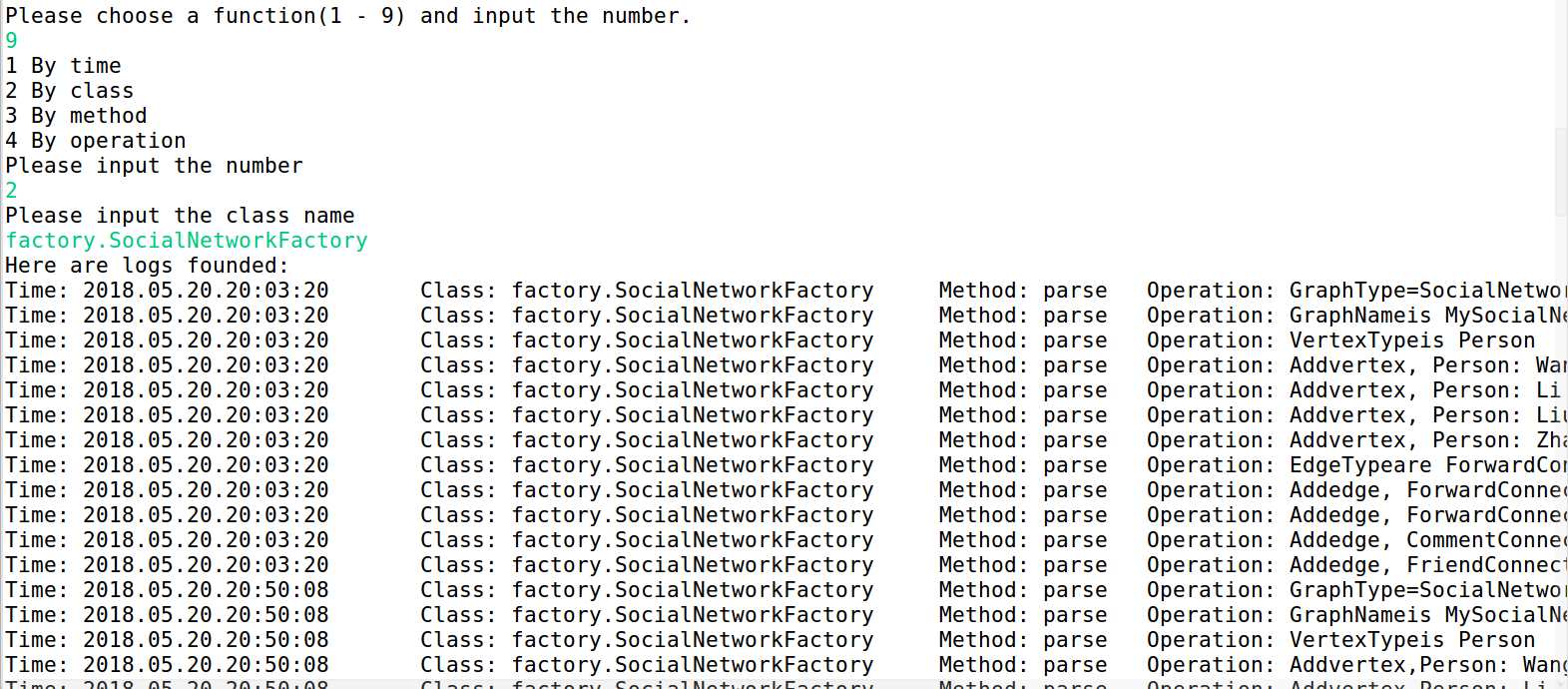
最终输出效果如下：

（1）按照时间段进行查询

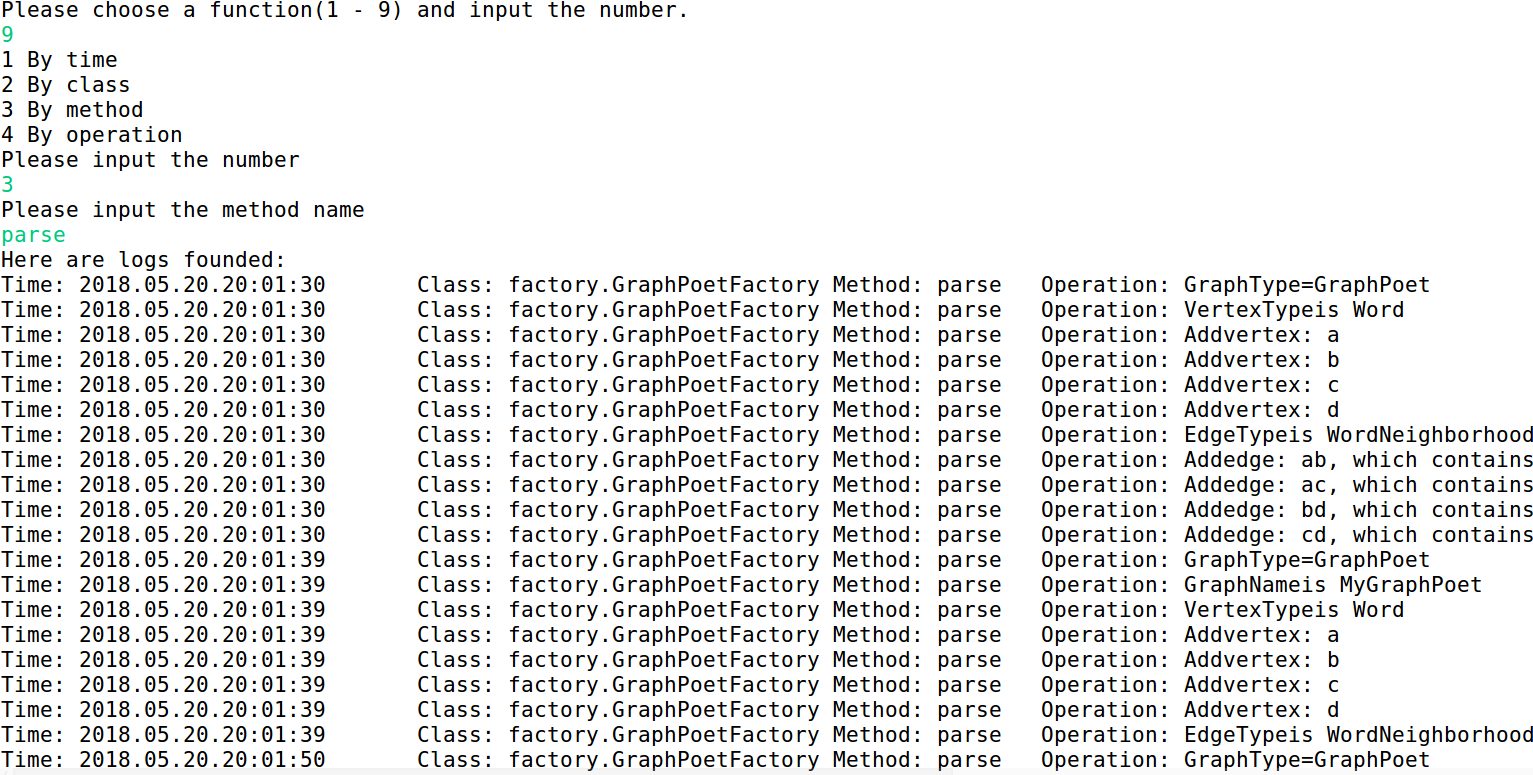




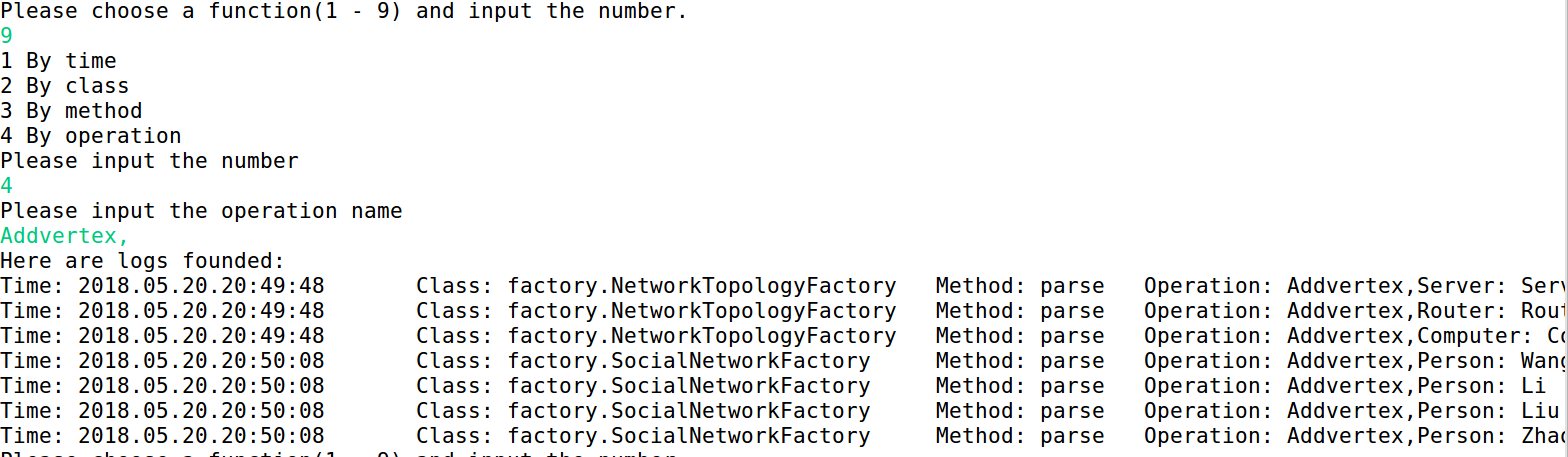
1. 按照类名进行查询



1. 按照方法名进行查询



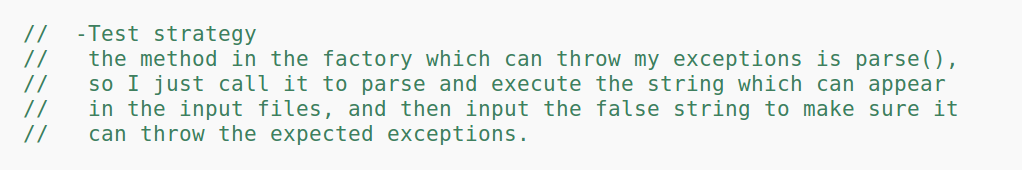
1. 按照操作进行查询



至此，四种查询方式已经全部实现。

## Testing for Robustness and Correctness

### Testing strategy



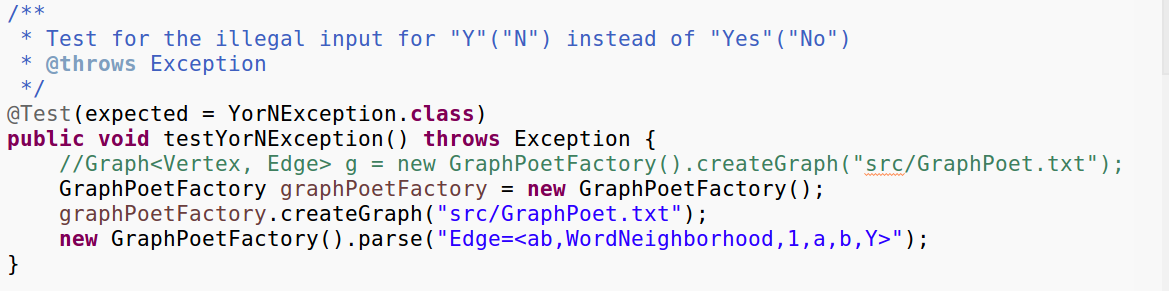
总之就是模仿各种异常输入，然后去判断是不是进行了期望的异常的抛出。

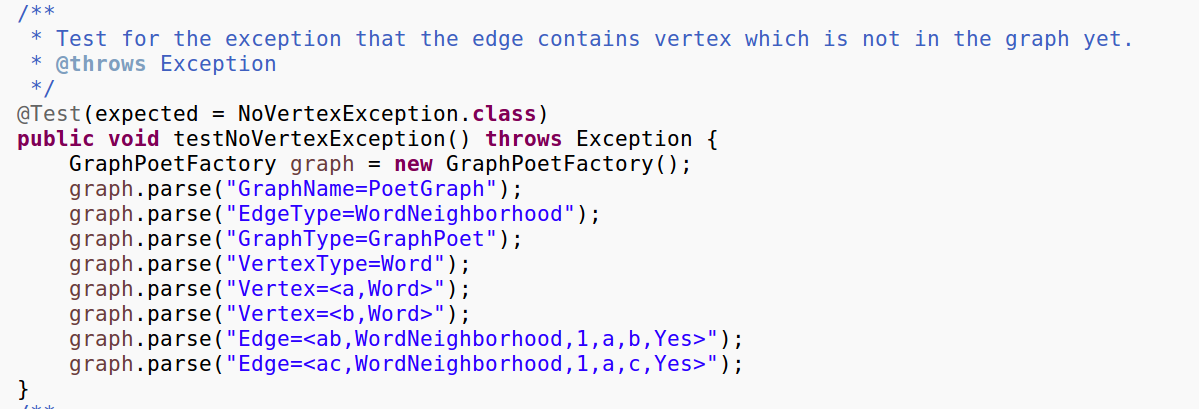
### 测试用例设计

（1）Exception测试

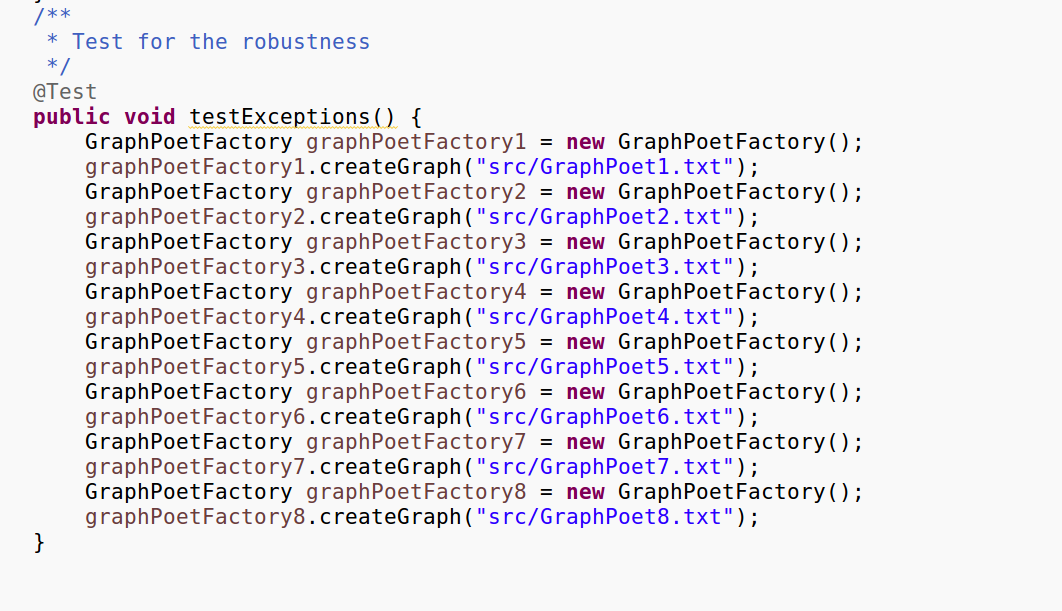
首先先要明确一点是factory中的一个方法parse（）来进行字符串的解析，从而进行加边加点的操作，所以，只需要让parse（）方法去解析不同的字符串去模仿加边加点，然后判断抛出的异常类型即可。

示例如下：





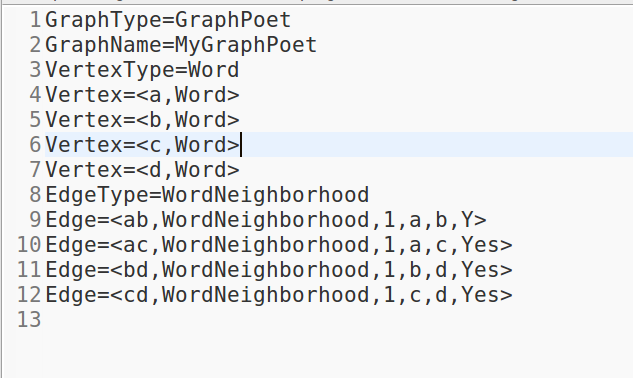
以下是针对于GraphPoet来进行所有异常的测试



其中这些测试文件就是模仿各种不同的错误输入。

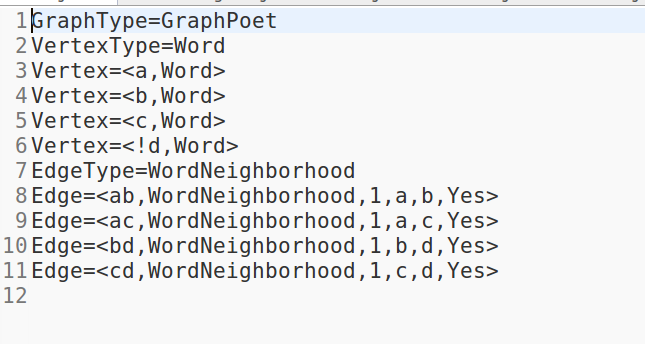
示例如下：

这是用“Y”来代替 “Yes”

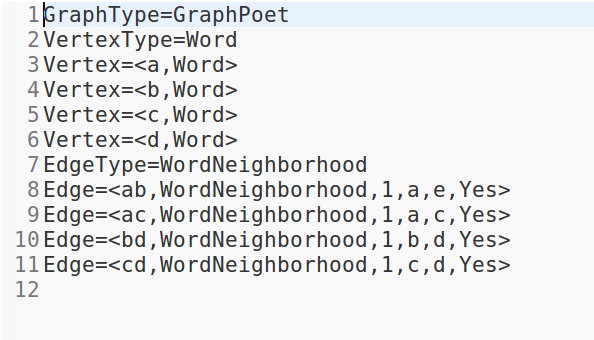


这是带权边没有输入权值：

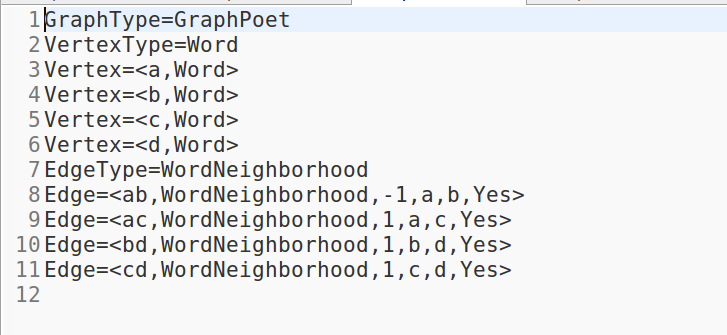
输入边的label不符合（\w+）



边中的点不在图中：



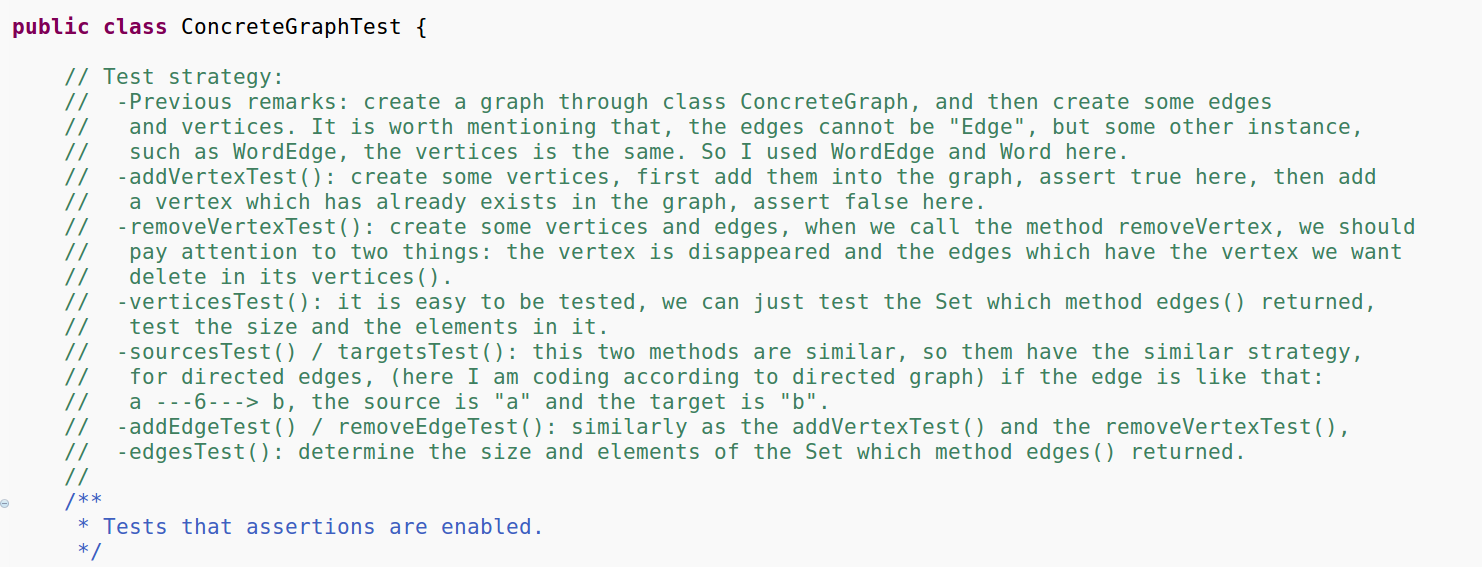
输入的权值不合法：

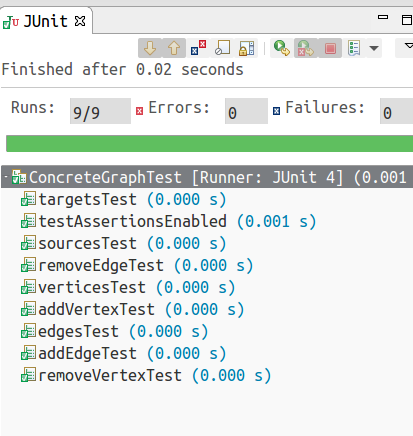


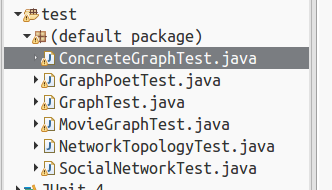
1. 关于点和边的测试

已经在lab3中的代码中有体现，所以这里只给出测试结果截图

如下的test strategy：

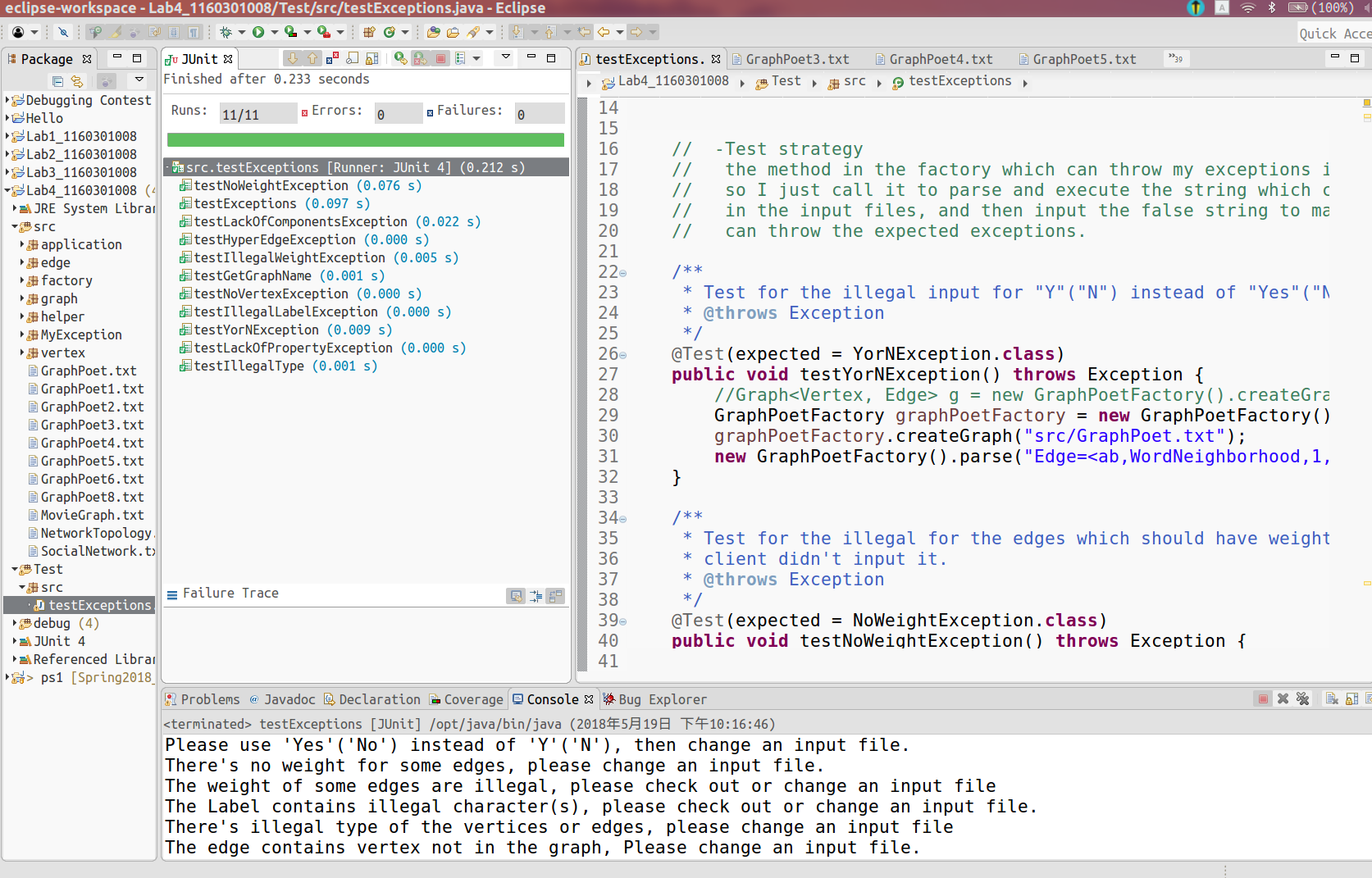




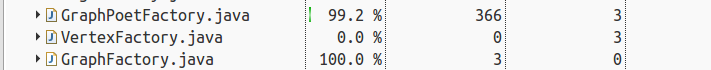


### 测试运行结果与覆盖度报告

测试结果如下：

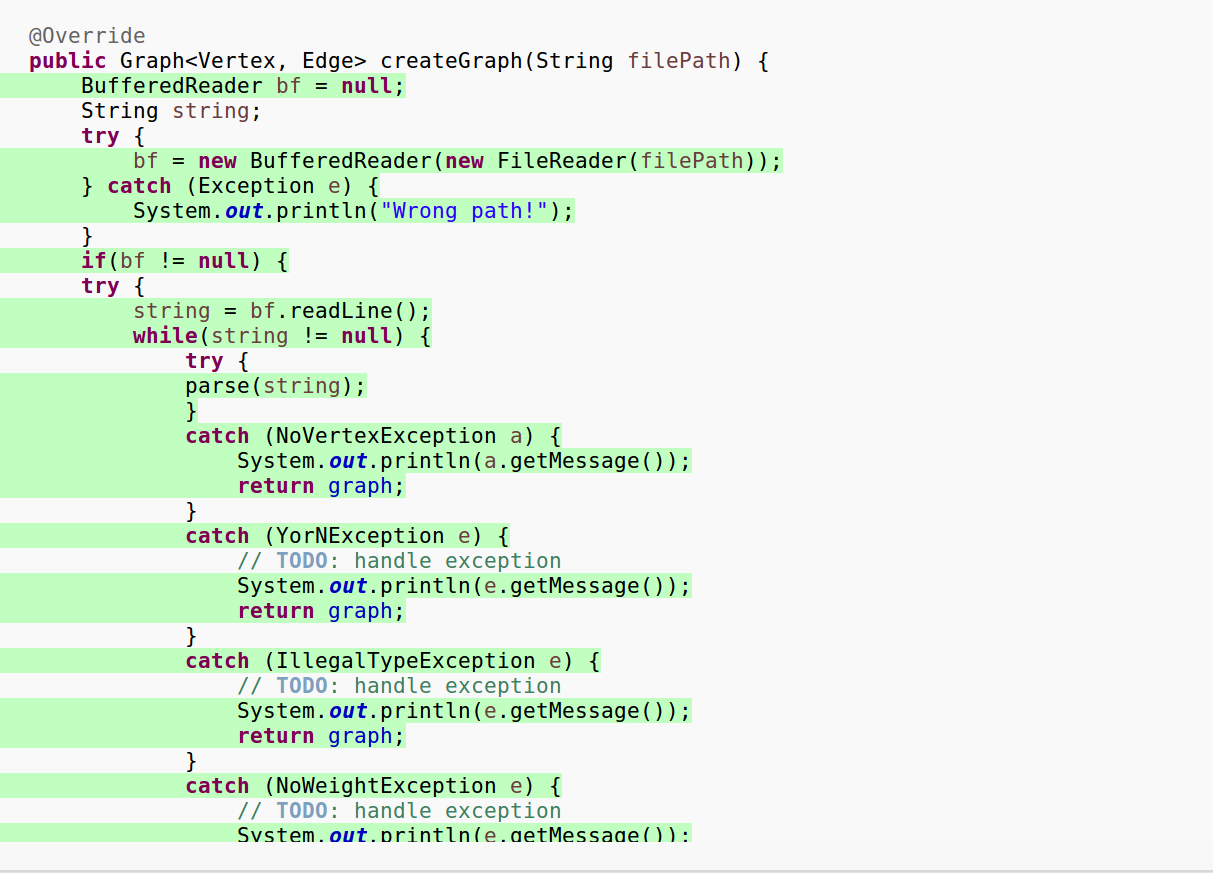


覆盖度如下（保证GraphPoet为100%）：



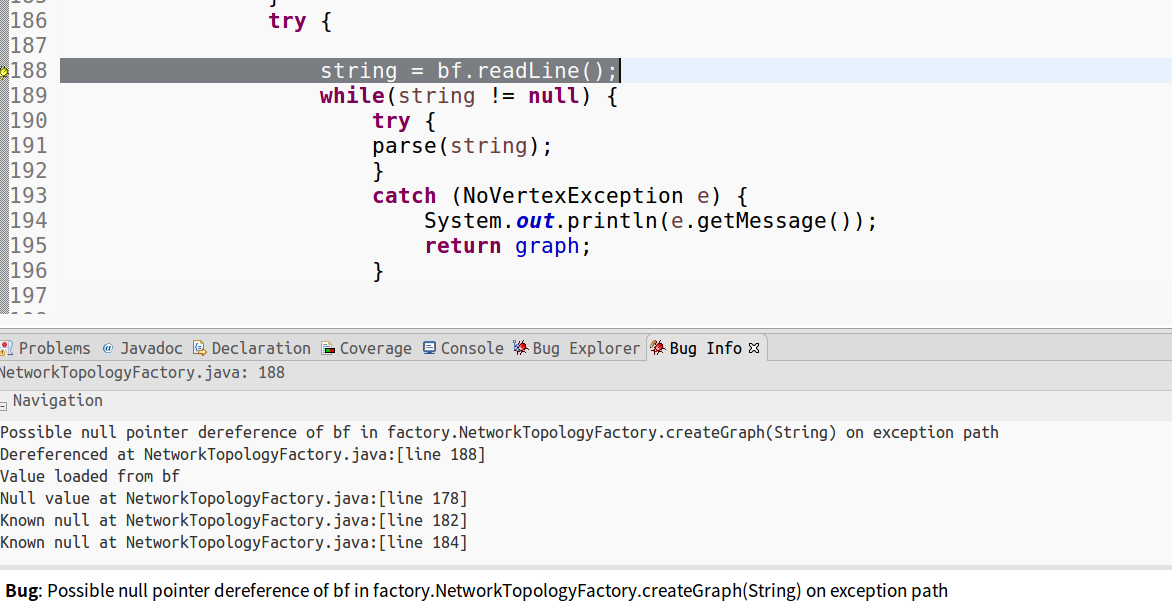
然后看一下GraphPoetFactory中的代码，确保都是绿色



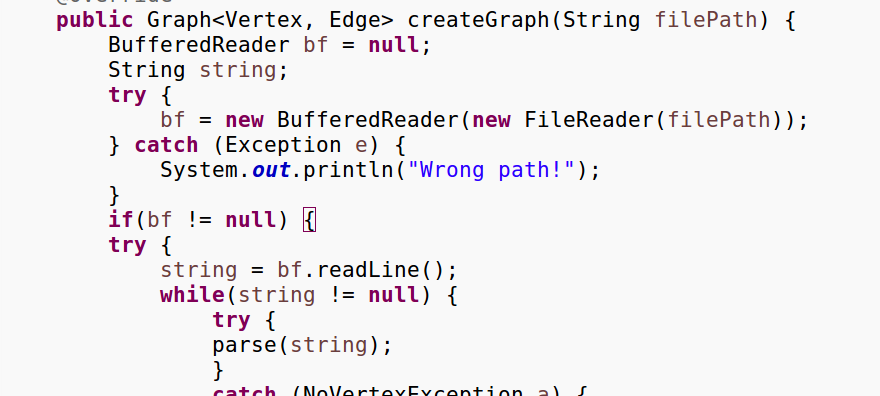


## FindBugs tool（可选）

首先，发现的第一个bug，就是在进行文件读取的时候，有可能会有空指针，bugInfo如下：

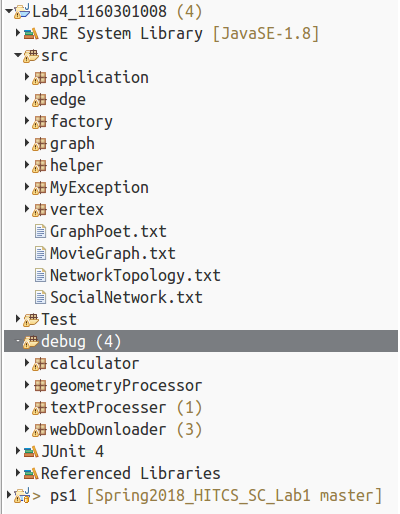


也就是说，我将bf初始化为null，但是其实是不允许为null的，所以，为了改正，我进行了如下的修改



这样就可以避免bf为null，，避免抛出异常。今后我会注意到这一点的。

至此，之前的代码已经将bug全部消除。（显示为4是debug部分的bug）



## Debugging

### 待调试程序

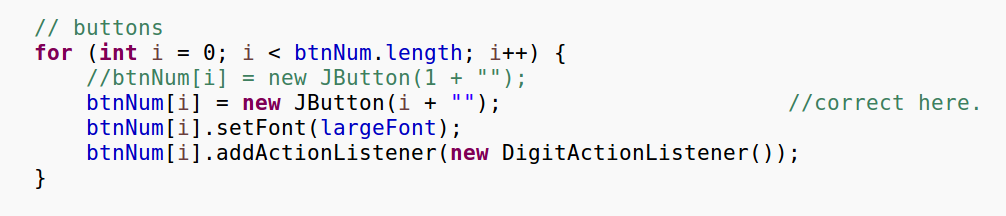
1. CalculatorGUI
2. GeometryProcessor
3. textProcessor
4. webDownloader

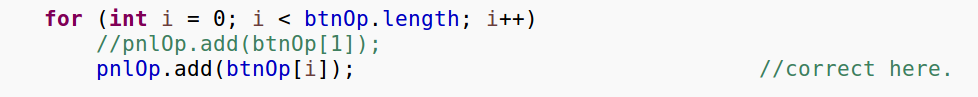
### 理解待调试程序的过程

1. 这个文件其实是一个计算器的GUI可视化，首先，发现了不能将数字显示在上面，同时运算符也只能显示“-”，由此意识到应该是将代码修改到可以正常运行为止。
2. 首先实现接口的类发现有代码没有实现，并且在初始化对象的时候没有调用实现类，而是调用了接口类，并根据SampleOutput文件，知道应该是要输出一些面积的计算之类的操作。
3. 字典树，在维基百科中查找了字典树的定义，理解了字典树，才知道该怎么去进行debug。
4. 文件下载，看了SampleOutput，大概知道了是类似于脚本的东西，自动的去进行文件的下载。

### 发现并定位错误的过程

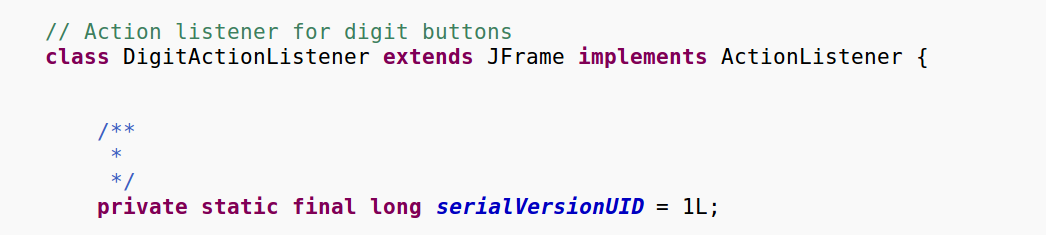
1. 首先，先去思考为什么数字和运算符不会显示在GUI界面，后来仔细读了一遍代码，大概了解了逻辑，发现了最终的错误原因





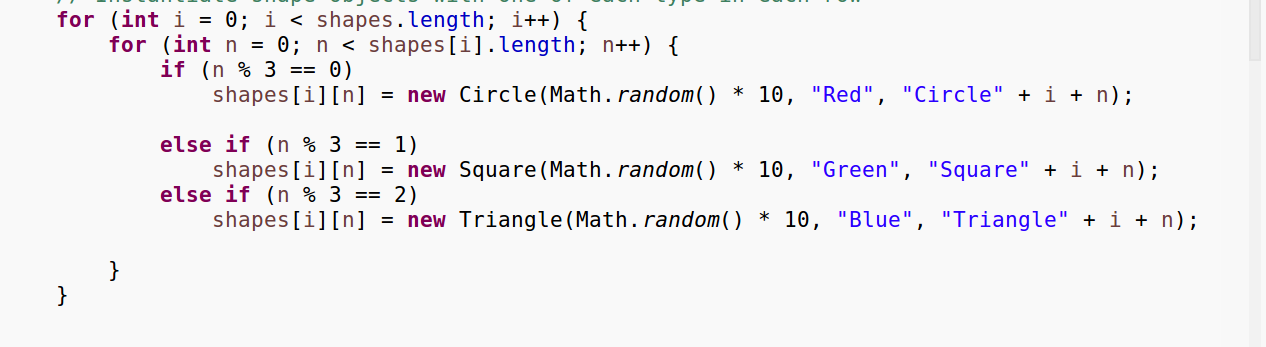
这两个错误都是将循环变量“i”写成了“1”，导致无法循环的去加入值。

其次，发现了有的方法显示没有被调用，例如下面的方法：



然后，我看了一下eclipse的提供的修改方式，采用自动修改方法，为每一个操作加上了serialVersionUID，还有很多很细小的bug，比如“+”“-”写反，“×”“/”逻辑写反，init没有设置为“true”等等。

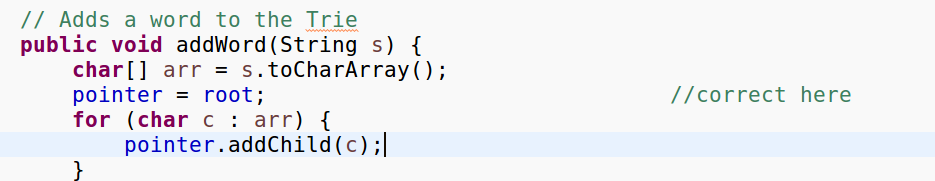
1. 这个比较简单，首先就是eclipse的报错（语法错误），实例化一个对象的时候，采用了接口类而不是实现了接口的实现类，所以，首先将这里改好



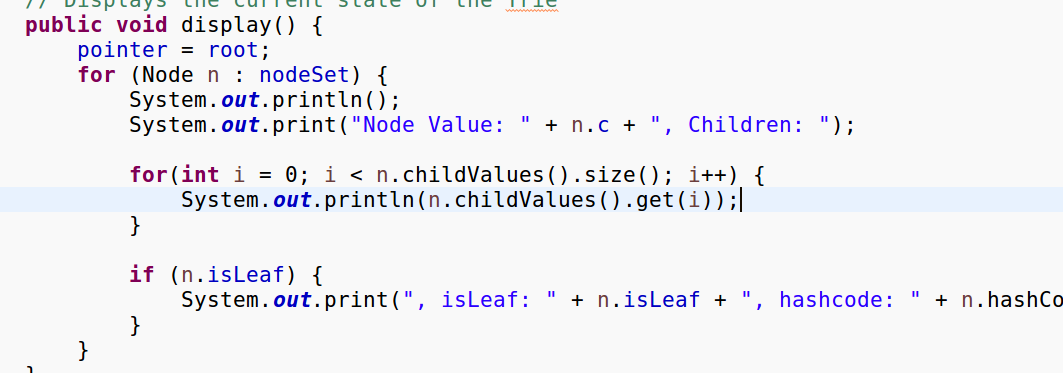
其次，是逻辑上的一些问题，我先读了一边代码，先弄清他的逻辑在实现接口的时候，我发现有的方法没有实现，先将其实现，然后，我运行了一下，发现

面积那里算的不太对，最终，我发现了是计算面积的公式写错了，然后进行改正即可，第二个还是比较好写的。

1. 首先通读代码，发现了在初始化树的时候没有为pointer指针初始化，所以改正如下：



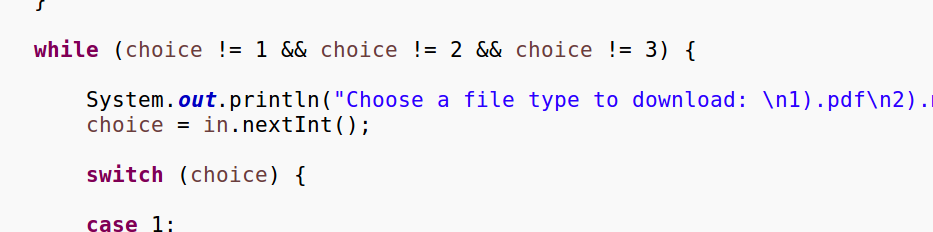
然后我运行了一下代码，发现在打印树的信息的时候，打印的是节点的地址值，然后仔细看代码，发现了问题，修改如下：



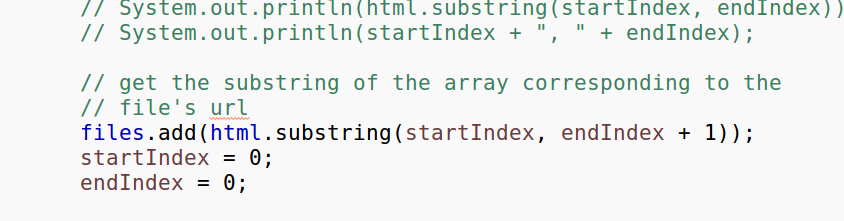
然后，发现了一个小的逻辑错误，在执行操作的时候，没有修改计数器的值



1. 通读代码，发现了和（2）类似的问题，也是在进行实例化对象的时候采用了接口类，修改之后，发现了一些逻辑问题，将“&&”写成了“||”如下：



然后发现了打印文件信息的时候不能正常输出文件后缀名，仔细的去看代码之后，发现是边界处理问题，修改如下：

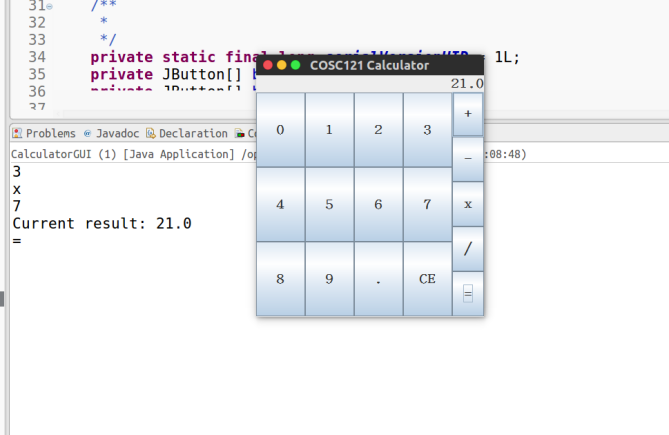


### 如何修正错误

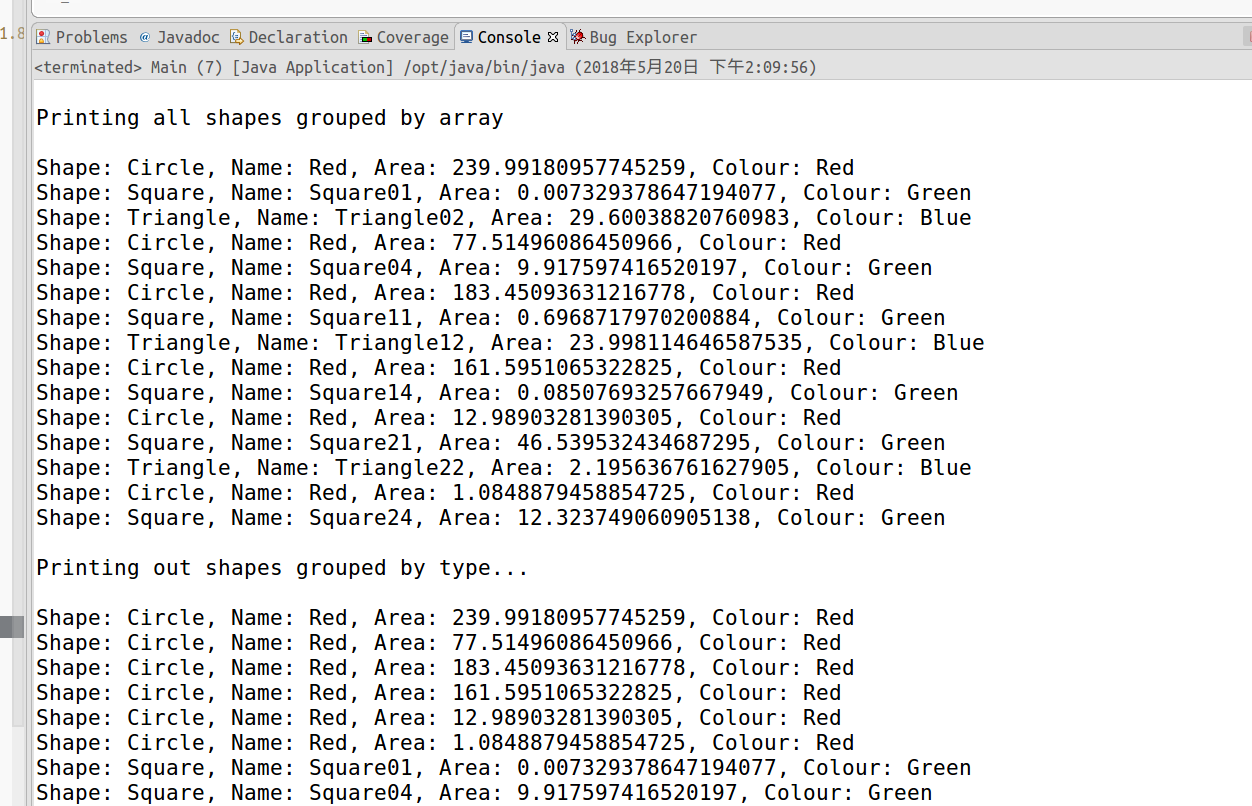
见3.6.3

### 结果

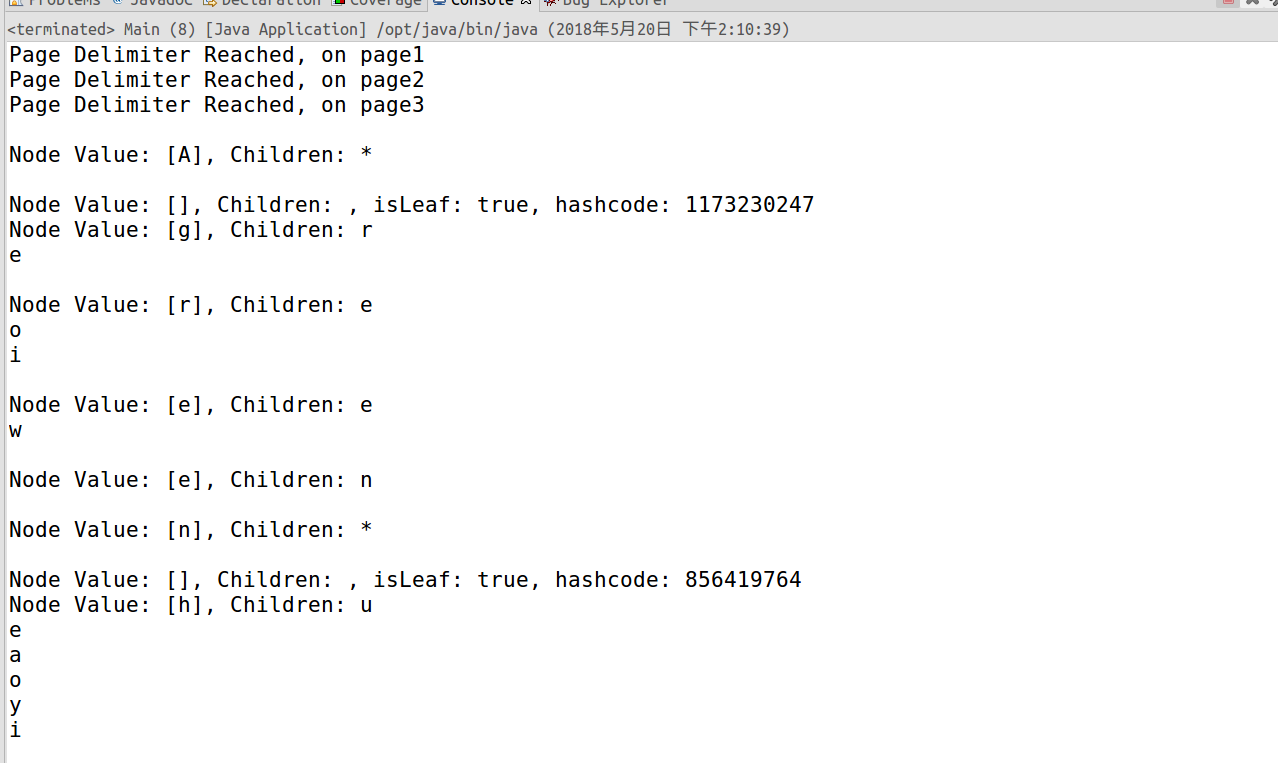
（1）



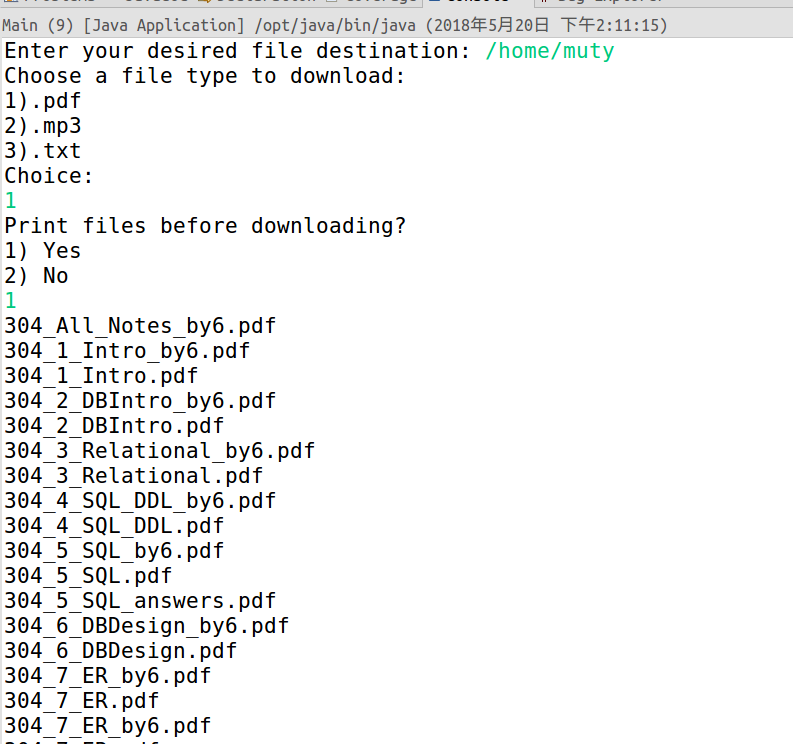
（2）



(3）



（4）



# 实验进度记录

请尽可能详细的记录你的进度情况。

|  |  |  |  |
| --- | --- | --- | --- |
| 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

# 实验过程中收获的经验、教训、感想

本节除了总结你在实验过程中收获的经验和教训，也可就以下方面谈谈你的感受（非必须）：

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？
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？
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. 关于本实验的工作量、难度、deadline。
8. 到目前为止你对《软件构造》课程的评价和建议。