

**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 | 任庆吉 |
| 学号 | 1160300422 |
| 班号 | 1603004 |
| 电子邮件 | 1091298335@qq.com |
| 手机号码 | 18846088521 |

**目录**

[1 实验目标概述 1](#_Toc511890111)

[2 实验环境配置 1](#_Toc511890112)

[3 实验过程 1](#_Toc511890113)

[3.1 Error and Exception Handling 1](#_Toc511890114)

[3.1.1 针对输入文本文件的异常/错误处理 1](#_Toc511890115)

[3.1.2 针对输入图操作指令的异常/错误处理（可选） 1](#_Toc511890116)

[3.2 Assertion and Defensive Programming 2](#_Toc511890117)

[3.2.1 checkRep()检查invariants 2](#_Toc511890118)

[3.2.2 Assertion保障pre-/post-condition 2](#_Toc511890119)

[3.3 Logging 2](#_Toc511890120)

[3.3.1 写日志 2](#_Toc511890121)

[3.3.2 日志查询 2](#_Toc511890122)

[3.4 Testing for Robustness and Correctness 2](#_Toc511890123)

[3.4.1 Testing strategy 2](#_Toc511890124)

[3.4.2 测试用例设计 2](#_Toc511890125)

[3.4.3 测试运行结果与覆盖度报告 2](#_Toc511890126)

[3.5 FindBugs tool（可选） 2](#_Toc511890127)

[3.6 Debugging 3](#_Toc511890128)

[3.6.1 待调试程序 3](#_Toc511890129)

[3.6.2 理解待调试程序的过程 3](#_Toc511890130)

[3.6.3 发现并定位错误的过程 3](#_Toc511890131)

[3.6.4 如何修正错误 3](#_Toc511890132)

[3.6.5 结果 3](#_Toc511890133)

[4 实验进度记录 3](#_Toc511890134)

[5 实验过程中遇到的困难与解决途径 3](#_Toc511890135)

[6 实验过程中收获的经验、教训、感想 3](#_Toc511890136)

# 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

实验环境设置请参见 Lab-0 实验指南。

除此之外，本次实验需要你在 Eclipse IDE 中安装配置 FindBugs（用于 Java代码静态分析的工具）。请访问 http://findbugs.sourceforge.net，了它并学习其安装、配置和使用。

git@github.com:ComputerScienceHIT/Lab4-1160300422.git

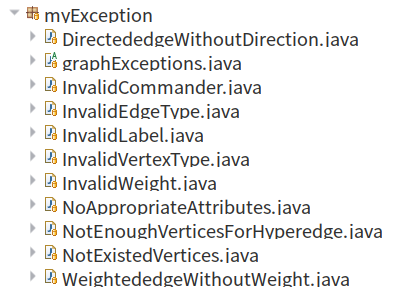
# 实验过程

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！）。

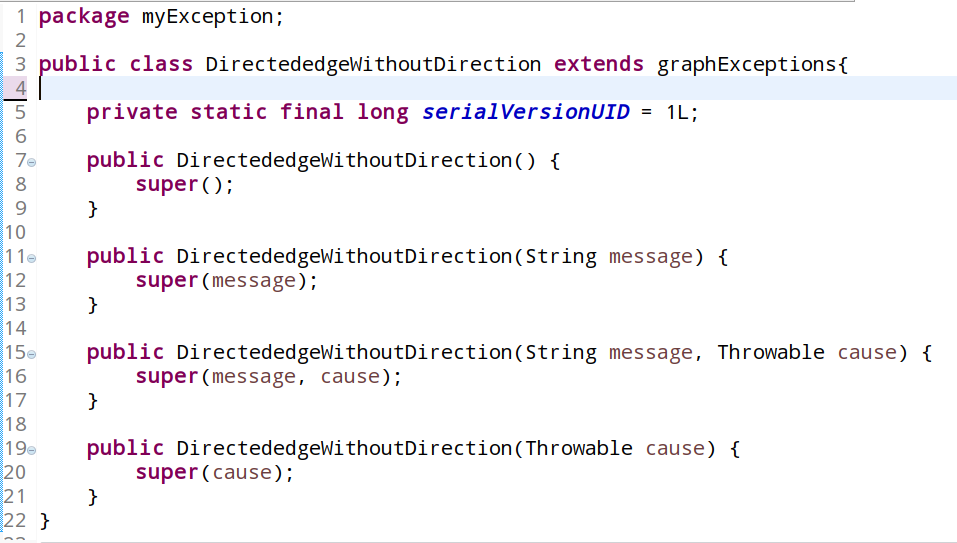
## Error and Exception Handling

### 针对输入文本文件的异常/错误处理

首先创建一个继承自父类Exception的类graphException，然后根据实验手册中提到的若干异常，从graphException中派生出来。



snapshot如图所示，异常的写法很简单，重写那些constructor就好，举例如下：



简述各异常触发的条件：

DirectededgeWithoutDirection：在有向图中引入了无向边

InvalidCommander：文件中存在不符合语法规则的语句

InvalidEdgeType：在某种类型的图应用中引入了不应出现的边类型

InvalidLabel：图、节点、边的 label 不满足正则表达式（ \w+） 的要求

InvalidVertexType：在某种类型的图应用中引入了不应出现的节点类型

InvalidWeight：带权边的权值不符合应用要求

NoAppropriateAttributes：为节点定义的属性的数目与特定应用的图的约束不符

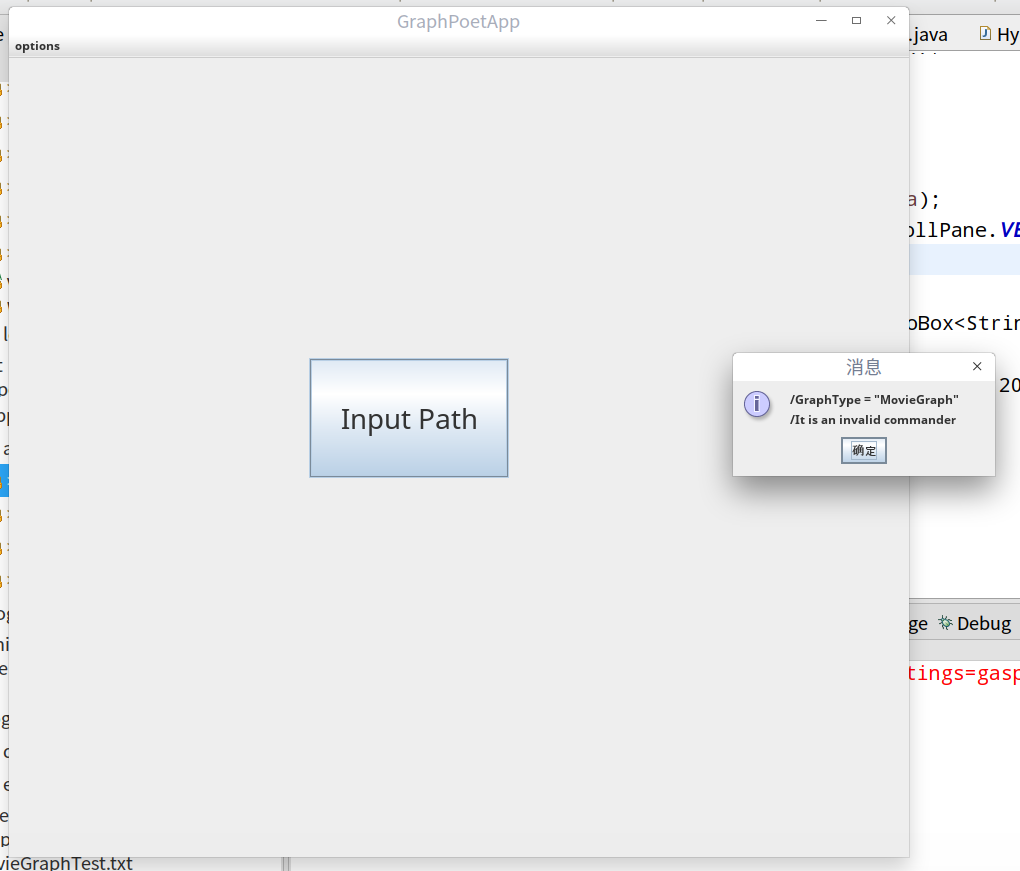
NotEnoughVerticesForHyperedge：某超边中包含的节点数小于 2

NotExistedVertices：边中使用的节点在节点部分未定义

WeightededgeWithoutWeight：带权边却未能给出权值

在读文件时可能产生相应异常的地方，将特定的异常抛出。我采用的设计模式是，读文件时只向外抛出异常，在app中调用时，再去处理，处理办法是，首先将错误信息写入日志，然后以弹窗形式告知user，让他重新选择文件。

这部分内容占据本次实验的大半，难度在于如何合理的划分各种异常，以及如何判断异常。前者根据实验手册，可以了解大概。后者需要使用正则表达式，去匹配语句，再逐一排查到底哪里出了问题。



这里我用了一个错误的文件，成功抛出异常并处理。

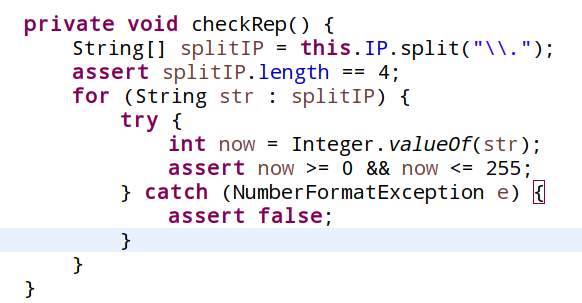
### 针对输入图操作指令的异常/错误处理（可选）

lab3我没做命令行，所以这部分也没做

## Assertion and Defensive Programming

### checkRep()检查invariants

在每个类开始，我都写清楚了RI，checkRep就是把RI部分翻译成assert语句，用于检查是否合法。举例：

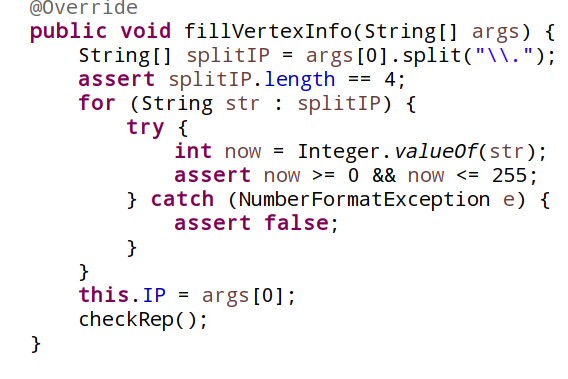


这是Vertex中的Computer类的checkRep，实验要求，Computer的属性IP是用“.”分割的四个整数，每个数的范围是[0,255]。只要是有一定约束的类，我都写了checkRep。

有一些类，比如Edge，他们几乎没有什么要求，有些要求比如，不能成环，但我觉得这是图的RI，所以我写在图里。

### Assertion保障pre-/post-condition

这部分内容是要求对每个函数传进来的参数先进行是否合法的判断，然后再交给函数执行。以及对方法执行后一些参数是否合法进行检查。其目的就是保证程序的健壮性。写起来很麻烦，但不是很难。举例如下：



Computer的fillVertexInfo方法，首先检查pre-condition，即传进来的参数是否是一个合法的IP地址，在执行函数后，再调用checkRep检查post-condition。

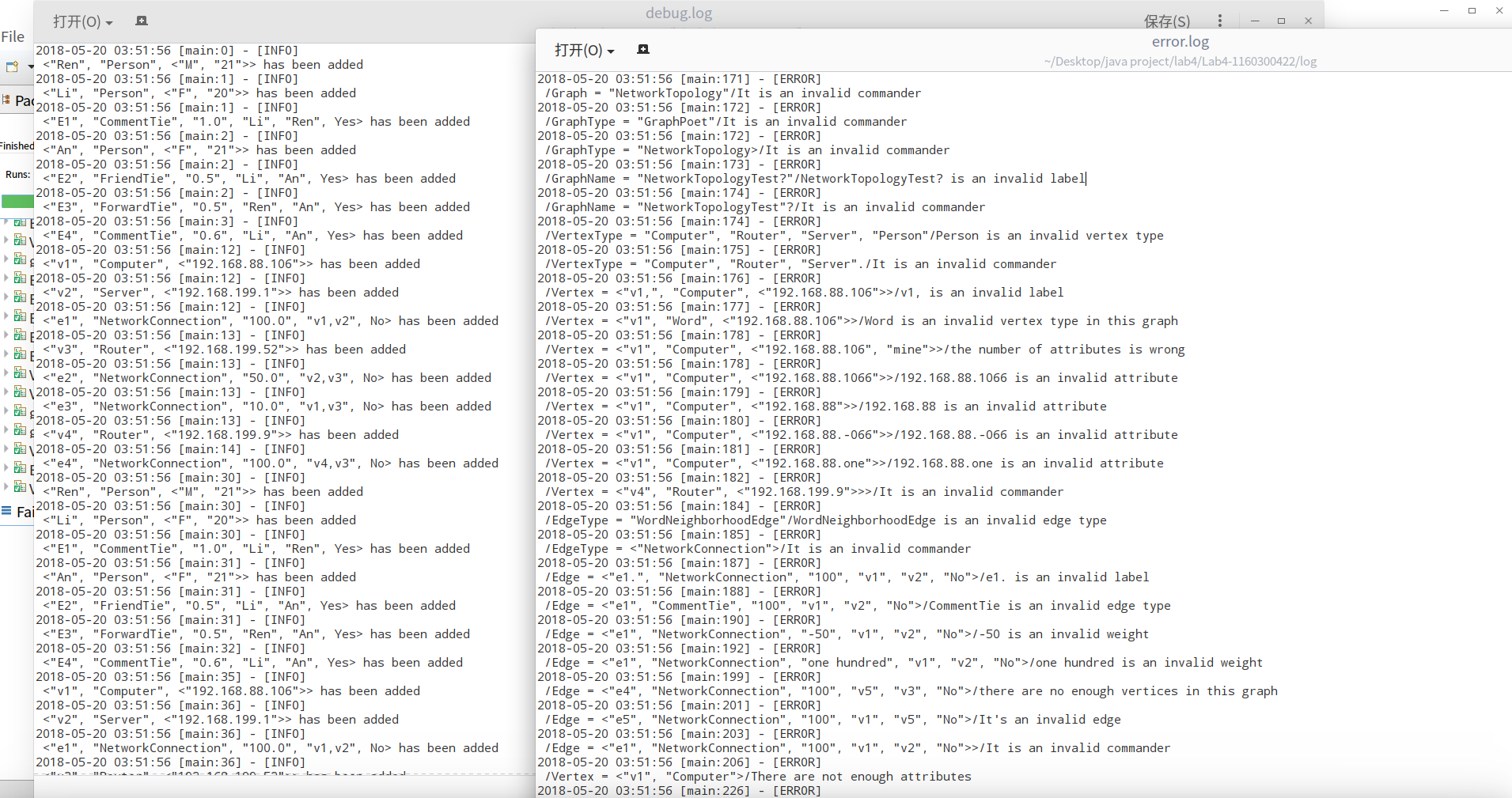
## Logging

### 写日志

通过我的学习，我选择了log4j来作为我的日志工具。嗯，具体怎么使用，我都是在网上找的教程，主要内容就是怎么写配置文件，我在这里没必要赘述。老师上课时说，这部分内容是用来锻炼我们复用别人代码的能力，我觉得没什么难度，只要有文档，认真读就可以了，所以也从这一方面提醒了我，好的程序要具有优良的可读性，要写好注释。这方面我一直在练习，我在每个类前都写清楚AF、RI等信息。对于每个变量、每个方法都给他一个合适的名字。这是我看的到的进步。

下面展示我的日志文件

我按照优先级error和info写了两个.log文件，名为debug.log和error.log



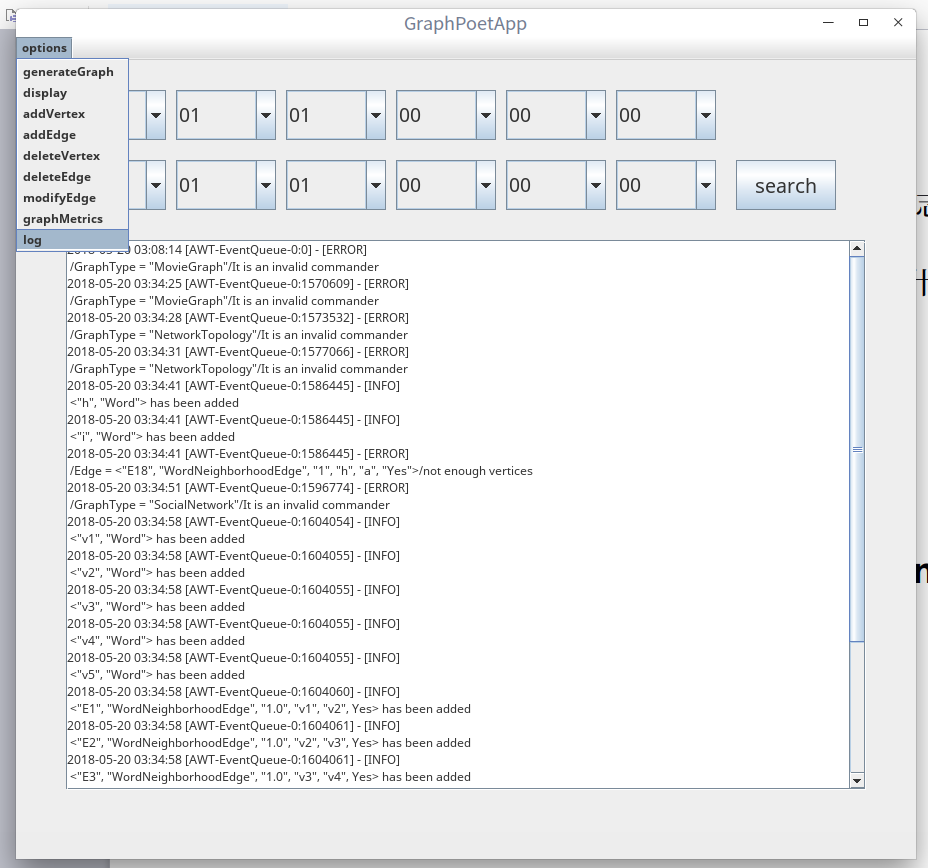
日志在项目下的log目录下（不是Log，Log是我对日志进行处理的类）

### 日志查询

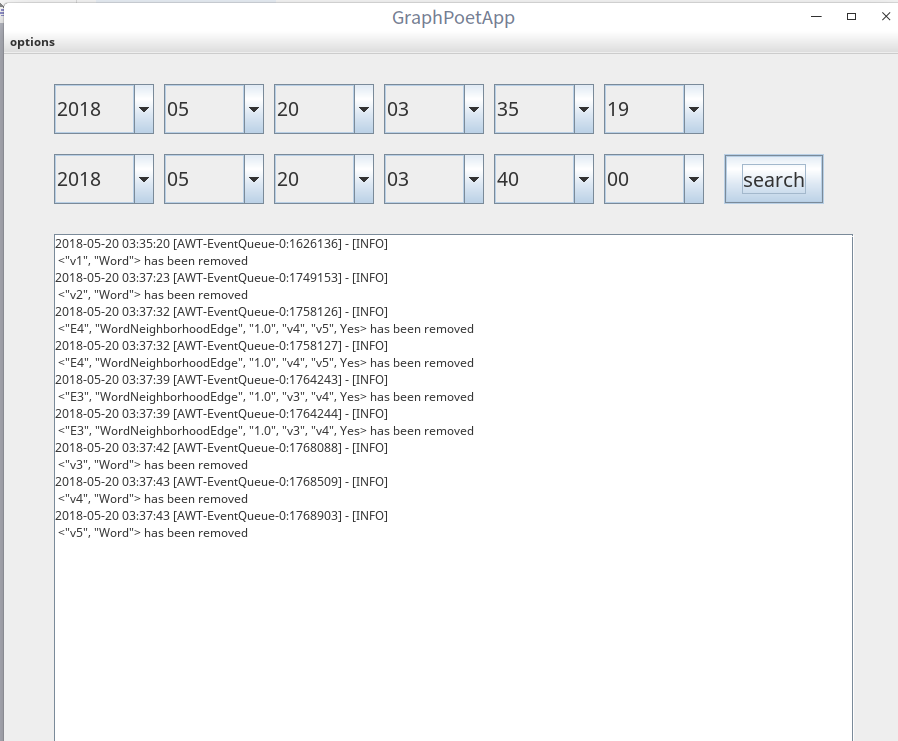
由于时间原因，我只实现了根据时间查找日志。其他的，例如按类型、

按类、按方法、按操作类型，在我看来没有什么区别，都是从日志文件中读取日志，再把符合要求的那部分提供给用户。

下面展示我实现的根据时间查找：



点击options中的log功能，可以看到所有的日志，因为user还没有选择过滤，所以都展示给user。当user选择一个时间段后，点击search可以看到想要看的那部分日志



日志内容很清晰，非常容易理解，具有可读性。

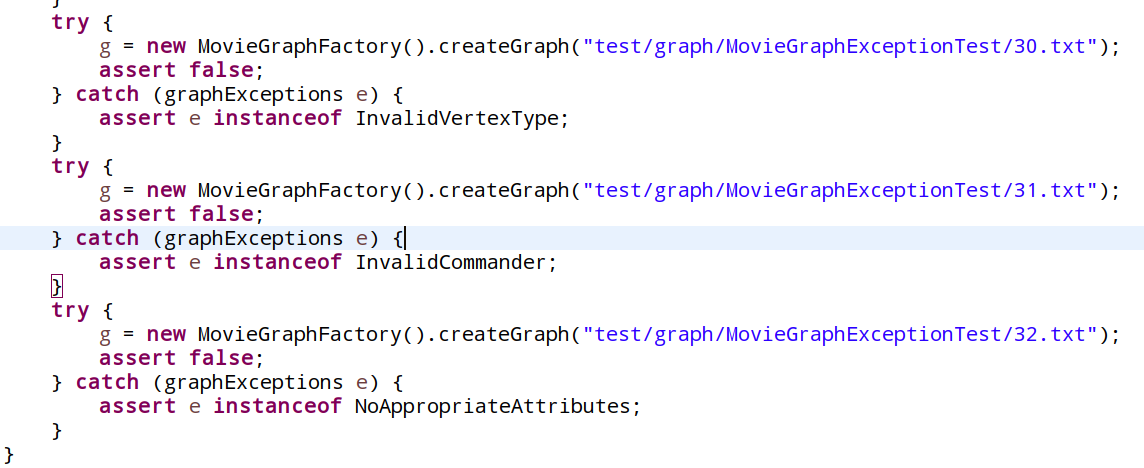
## Testing for Robustness and Correctness

### Testing strategy

对于不同的方法有不同的测试方法，主要思想是使用等价类和边界值的测试思想。这部分工作，我做的不是那么认真，没有在test中写清test strategy，但是，测试文件中的测试样例都是我认真思考过的，都是“很强”的测试。举例如下：



…………………………



对于每个图的exception测试，我都力求完美重现每个异常，所以我为每个图都编写了数十个带有错误的文件来让factory方法来处理，检测是否能够正确处理这些异常，只有先经过自己设置的这一关，才能交付用户使用，保证程序具有一定的健壮性。

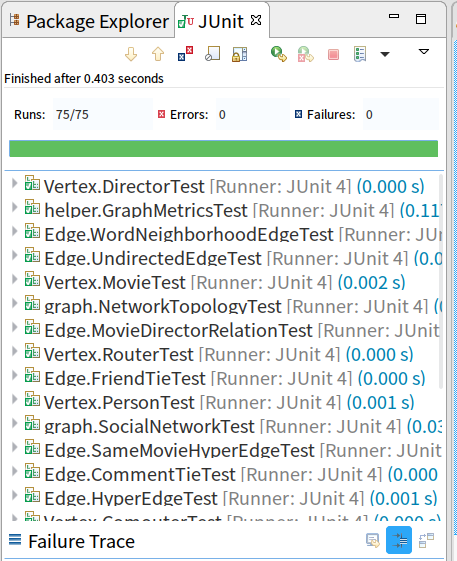
测试文件在test目录下，对应每个图有一个文件夹

test文件实在太多了，无法一一列举，以此为例来说明我在测试自己代码上用的心血。

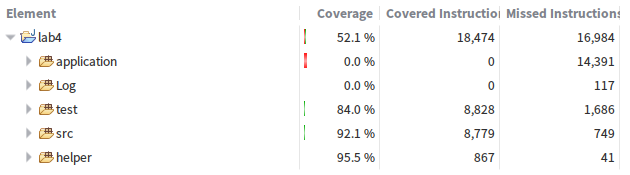
### 测试用例设计

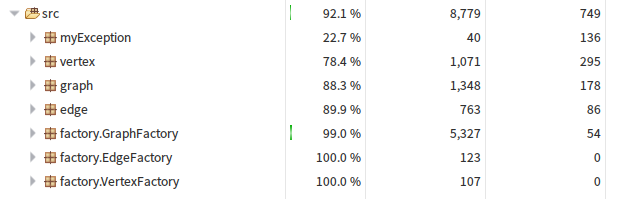
与上部分内容有些重叠，主要思想是力求跑遍程序的所有角落，为每一类错误都编写测试样例，来保证程序的正确性，健壮性。

### 测试运行结果与覆盖度报告

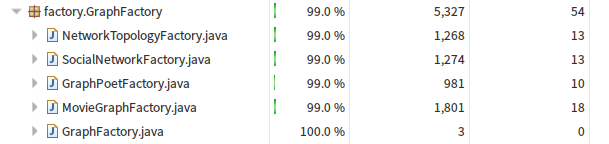


完全通过。





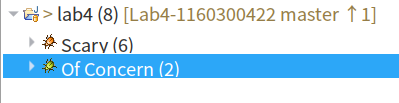
对于项目主要代码的测试覆盖度达到92％，除了异常类（我真的不知道测试他们干嘛，就重写了几个constructor，其他的都是继承自Exception的）其他类，比如Vertex、Edge、graph都达到了很高的覆盖度。其中我觉得最重要的是关于graph的工厂的测试，因为他几乎调用了所有的功能，这部分我写的测试样例非常详细，



全部达到99％的覆盖度，我很满意。

application我不知道怎么用junit测试，只是手动调试了一下，功能都实现的很好。

## FindBugs tool（可选）



findBugs结果

## Debugging

### 待调试程序

这部分实验内容，我完成了两个，geometryProcessor和calculator

### 理解待调试程序的过程

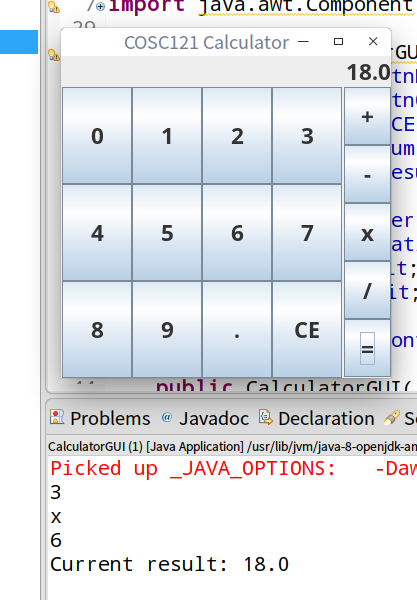
总的说来，老师提供给我们的这四个包含bug的代码，逻辑都很清晰，也不算很难改，大概不到一个小时就能改的差不多。代码里出现的主要问题，都是很明显的错误，比如在calculator中没有显示四则运算的按钮，没有实现相应计算功能，就去相应模块修改代码就行，但是因为时间太久，而且错误也很琐碎，所以我记不清调试的全部过程，我也没有时间再去重做一次，所以这部分实验报告不太完整，请TA、老师见谅。

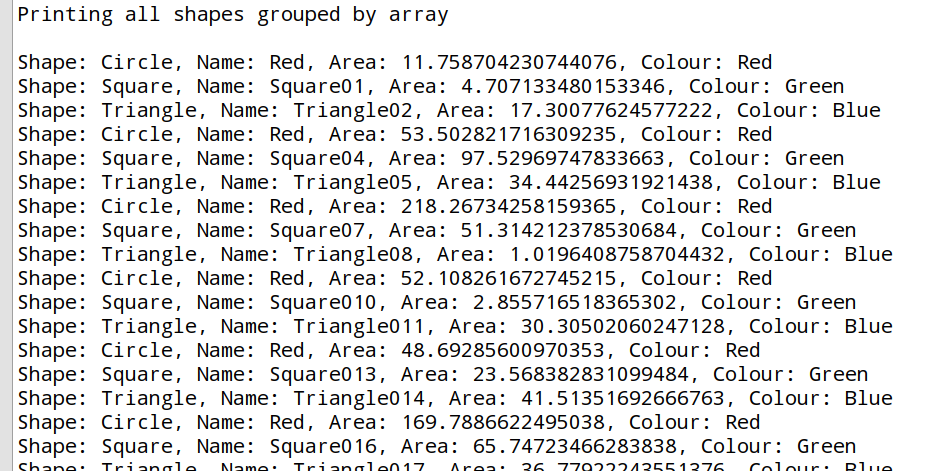
### 发现并定位错误的过程

### 如何修正错误

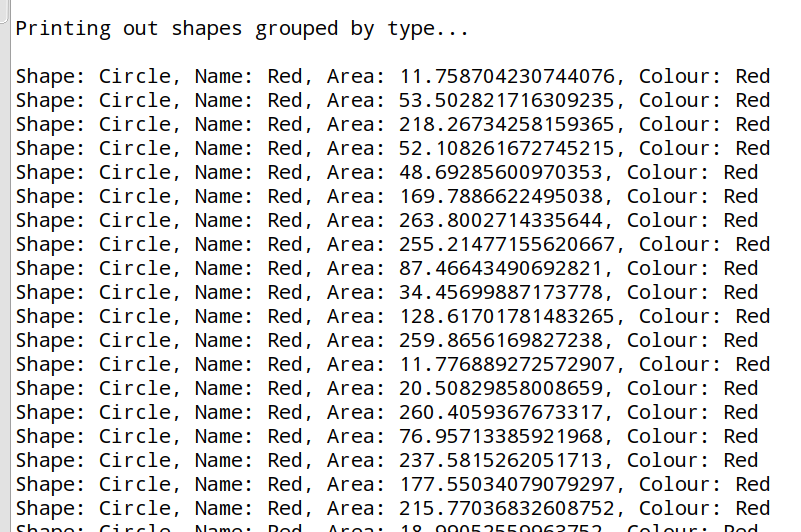
### 结果

下面展示一下这两个改好的程序运行效果





…………………………



# 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 |
| 5.7 | 下午 | 阅读实验手册，弄清楚要干什么 | 对整个任务有了大概理解 |
| 5.8 | 下午 | 阅读实验手册，弄清楚那些异常指的是什么 | 不太清楚，咨询了TA |
| 5.9 | 下午 | 感觉清楚了，于是开始写异常 | 进展顺利 |
| 5.10 | 晚上 | 写完GraphPoet的异常处理，及测试 | 完成 |
| 5.12 | 上午 | 写完SocialNetwork的异常处理，及测试 | 完成 |
| 5.12 | 晚上 | 写完NetworkTopology的异常处理，及测试 | 完成 |
| 5.13 | 全天 | 写完MovieGraph的异常处理，及测试 | 完成 |
| 5.14 | 下午 | 调试程序 | 完成一个  caculator |
| 5.15 | 下午 | 调试程序 | 完成  geometryProcessor |
| 5.16 | 晚上 | 学习如何使用log4j | 基本了解 |
| 5.18 | 下午 | 有些注释没写清楚，补一补 | 完成 |
| 5.20 | 凌晨 | 添加日志功能，写入文件，按时间搜索 | 完成 |

# 实验过程中遇到的困难与解决途径

实验过程遇到了很多困难，有些是对实验内容的不理解，这方面我询问了  
TA，得到了很耐心的解答。还有些是对陌生知识的使用，比如log4j，我通过读大牛写的博客，慢慢去理解。还有一个问题就是如何分配时间，我这两周的课特别多，所以没给实验太多时间，有些事一拖再拖，所以只能熬夜补，这样不好，我会合理安排时间，按时完成每项作业。

# 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？首先实现正确性，然后在站在user角度，尽可能地进行一些破坏性的输入，希望程序崩溃，从而使程序健壮性更好。
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？看具体情况，看这个错误和异常是否可以接受。
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？一个是让自己的程序健壮性更好，一个是使用程序时按照一定规定。给别人的程序最好有很强的健壮性，但是自己用的程序，可以适当降低要求。
4. 过分谨慎的“防御”（excessively defensive）真的有必要吗？有必要，你永远不知道谁会用你的代码。
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？写日志是一个很好的方法，断言也是，他们都能清楚地指出是哪里出了问题。
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？不一定，有些奇怪的组合可能让程序崩掉。
7. 关于本实验的工作量、难度、deadline。工作量很大，难度不大，deadline有点急，有些功能来不及实现。
8. 到目前为止你对《软件构造》课程的评价和建议。软件构造实验使我编程能力有了很大提升，学会了很多解决问题的办法，能看得到自己在慢慢进步。