

**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 | 康瑞 |
| 学号 | 1160300514 |
| 班号 | 1636101 |
| 电子邮件 | 1160300514@stu.hit.edu.cn |
| 手机号码 | 18745142051 |

**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

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。

IntelliJ IDEA : Compiler, Builder

Junit4

Java 9

Findbugs

https://github.com/ComputerScienceHIT/Lab4-160300514

在这里给出你的GitHub Lab4仓库的URL地址（Lab4-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！）。

## Error and Exception Handling

### 针对输入文本文件的异常/错误处理

覆盖实验手册3.1节(1)中列出的各项任务。

I designed 4 Exceptions in the package: exception, and three of them is controlled by the Apps, Factory, Helper, and some of Vertex and Edges for the use of spec check, while these exceptions are in Apps, Helper, Factory, they are judging whether the input from CMD or File is true or not, and throw it in each class, and these exceptions is handled in the apps with catch, and if check Rep method (only in some of the vertex and edges) failed it would throw an exception named RepFieldException, when there being some wrong with Input: there are just two cases no matter from which the line are read: NoSuchTypeException and MessageFieldIncorrectException. NoSuchTypeException is thrown when there is error because of the type of input vertex or edge, such error can be: a(n) vertex/edge adds into a graph however this type of vertex or edge are not allowed. MessageFieldIncorrectException is used when the dataset has caused a Data parse exception or just because of the error of the misusing the field which is not allowed. When parsing Integer/Double elements and checking the label of vertex etc, such exception is thrown by The Class: parseInputHelper or parseCMDhelper. InnerClassException is often caused by inner error and used for labeling when the software is crashed, for almost all situation such exception cannot be thrown, so while testing the coverage and exception of the project, this exception is of 0%.

After all, some exception are thrown by checkRep method of someof the vertex or edges classes and can also be ,or say for the most situation, thrown by Apps and parseInputHelper and parseCMDhHelper, handled by the Apps.

For each error and the exception is:

1. Document input error: the leakiness of some variables or miss spelling of some variables: MessageFieldIncorrectException, thrown by Factories and dealt in Apps.
2. Definition missing of some labels: MessageFieldIncorrectException , Thrown by Factories: GraphFactory and Edge/VertexFactory handled by Apps/
3. Not confirm with the correct input: the same as 1
4. No such type of vertex in this Graph: NoSuchTypeException, Thrown by GraphFactory handled by Apps
5. Edge :4
6. Directed edge in undirected graph: pass and caused warning
7. Exception: NoSuchtypeException Thrown by GraphFactory handled by Apps
8. Same as 6
9. Continue: Warning

Others.

### 针对输入图操作指令的异常/错误处理（可选）

覆盖实验手册3.1节中(2)列出的各项任务。

Just using the exception mentioned above, the CMD input is handled by ParserCMDHelper and redirected into Parser Input Helper, so the CMD handling is dealt in parse CMD helper and parserInputHelper and the other classes, and the exception is handled in Apps.

Handle details: same as file input

## Assertion and Defensive Programming

### checkRep()检查invariants

almost all the input values are checked by factories, Apps, helpers, so the checkRep() is only happens on SocialNetwork and some vertex and edges to check the rep is vaild or not. SocialNetwork==1.0?

### Assertion保障pre-/post-condition

Assertion is used by exception throwing, only some of the class is uses assertions, for inner helps.

## Logging

### 写日志

Direction: package: src-logge, class: logGenerateStream.java in this class, the xml based log is formed and I provide method to parse the xml document, and return the message asked, the generator: with the method: Append: add warning, error, information and the operation to the log holder, the Append method will append new lines to file.

### 日志查询

The resolve method in logGenerateStream.class, accept the regex and parse the regex, read the generated file, and find the messages fit for the regex.

## Testing for Robustness and Correctness

### Testing strategy

Location: package: test, GeneralTest.java is test for Junit and test-test: Main.java is tests for Console input and used for a real app, I have test the whole Apps and nearly meets all the errors and exceptions using the equivalence of exceptions and apps.

### 测试用例设计

I tested the GraphPoetApp for all kinds of instructions to reaching a higher coverage.

data/NetworkTopology.dat --GraphTypeError

file --in data/WordGraph2wwewq.dat --file not found

file --in data/WordGraph.dat --right

save -s SQ1 --right, save graph

vertex --add R Word --add vertex

GraphPoet –help --helper

edge --add TMP SX R -- add edge

save -recall SQ1 --recall the save

degreeCentrality --calculate

degreeCentrality SX

closenessCentrality SX

betweennessCentrality SX ------------

vertex --add S Word

vertex --add SS dsdww

edge --add TMP SX R

edge --add TMP SX S

inDegreeCentrality SX

outDegreeCentrality SX

modify vertex SX SXX

distance SX JSX

radius

diameter

log --get information --get logs by regex

log --get error

log --get time

2018-05-19 11:02 --input the time and searching the logs happens after it, use the UTC time.

edge --delete LCI --vertex oper

vertex --delete $R[\\w]+

exit

And I test SocialNetwork for all errors and exceptions, below are the console input show:

data/NetworkTopology.dat

file --in data/MovieGraph.dat

file --in data/SocialNetwork\_01.dat

file --in data/SocialNetwork\_02.dat

file --in data/SocialNetwork\_03.dat

file --in data/SocialNetwork\_04.dat

file --in data/SocialNetwork.dat

file --in data/WordGraph.dat

vertex --add SX

vertex --delete SX

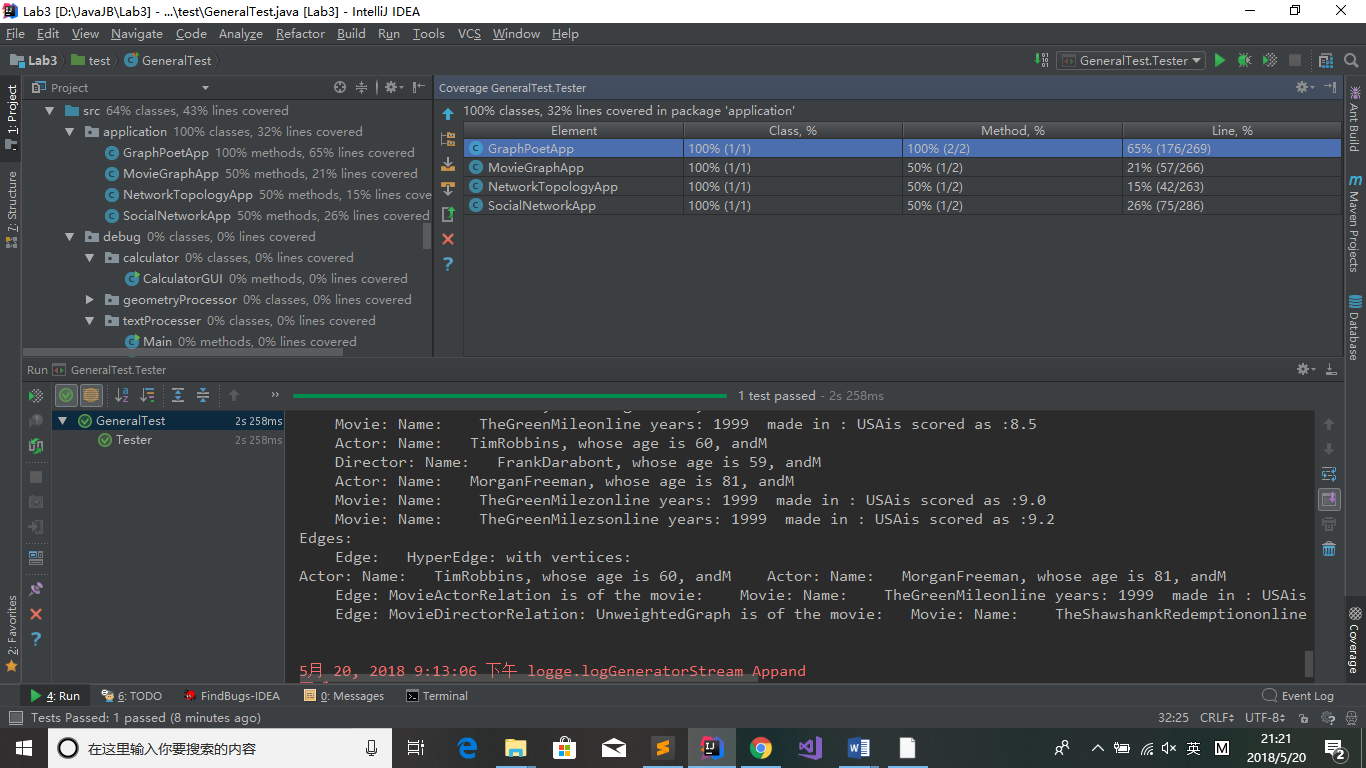
vertex --add JS Person

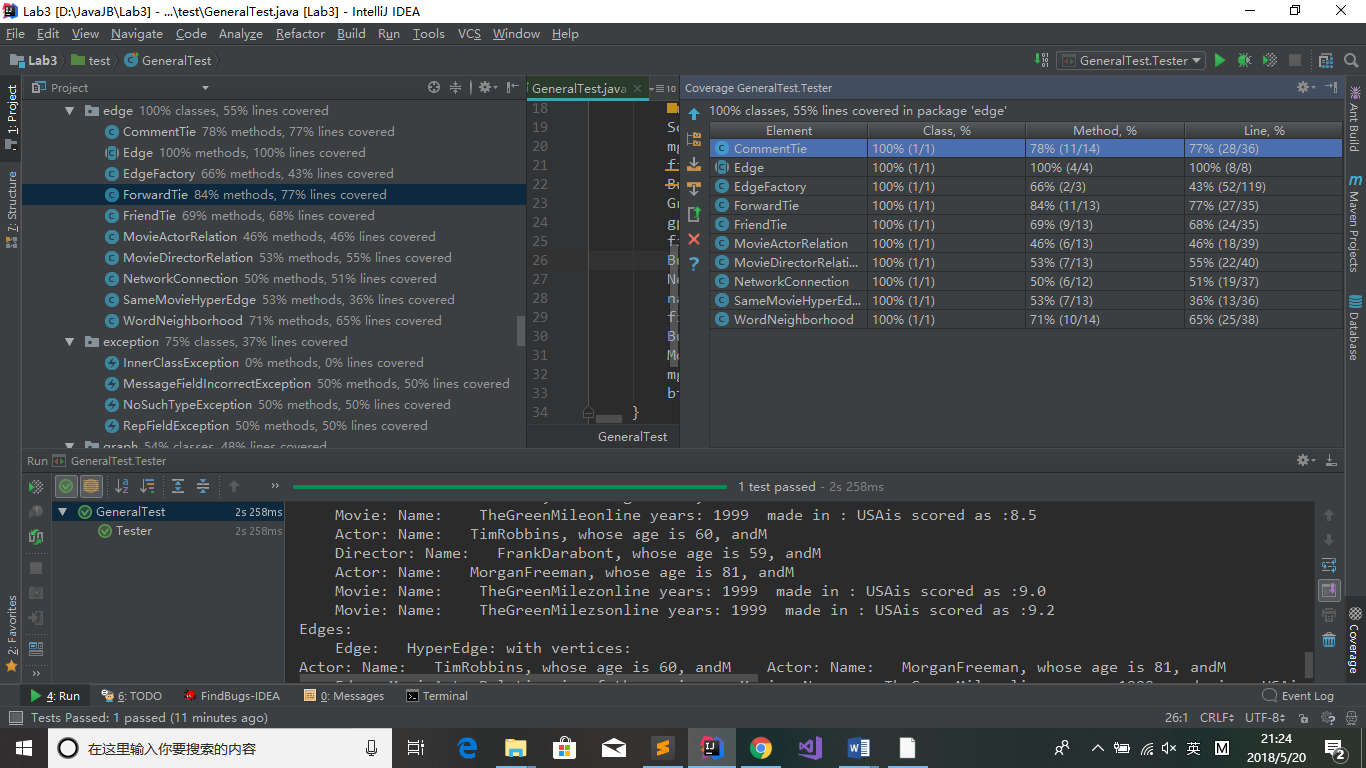
F/20

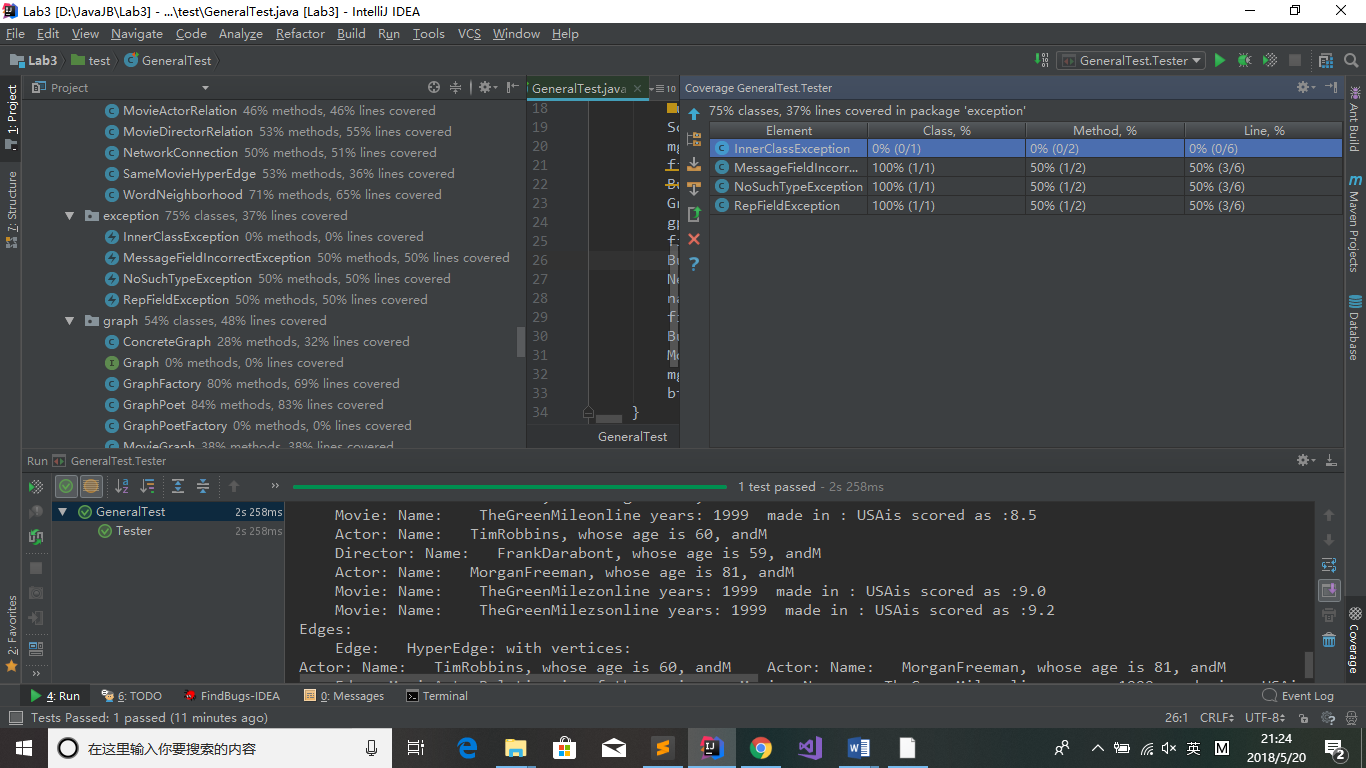
Exit

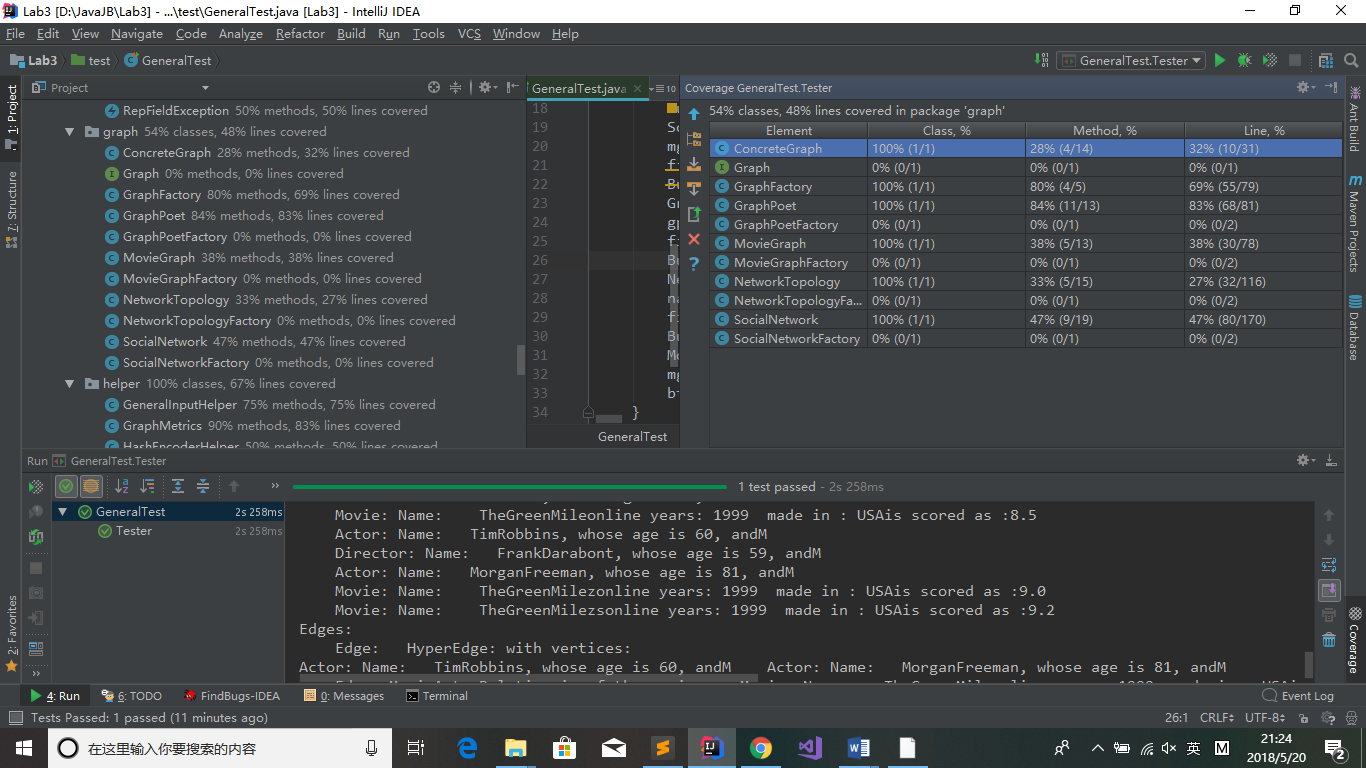
The test errors can be seen in package: data: lab3/data/…

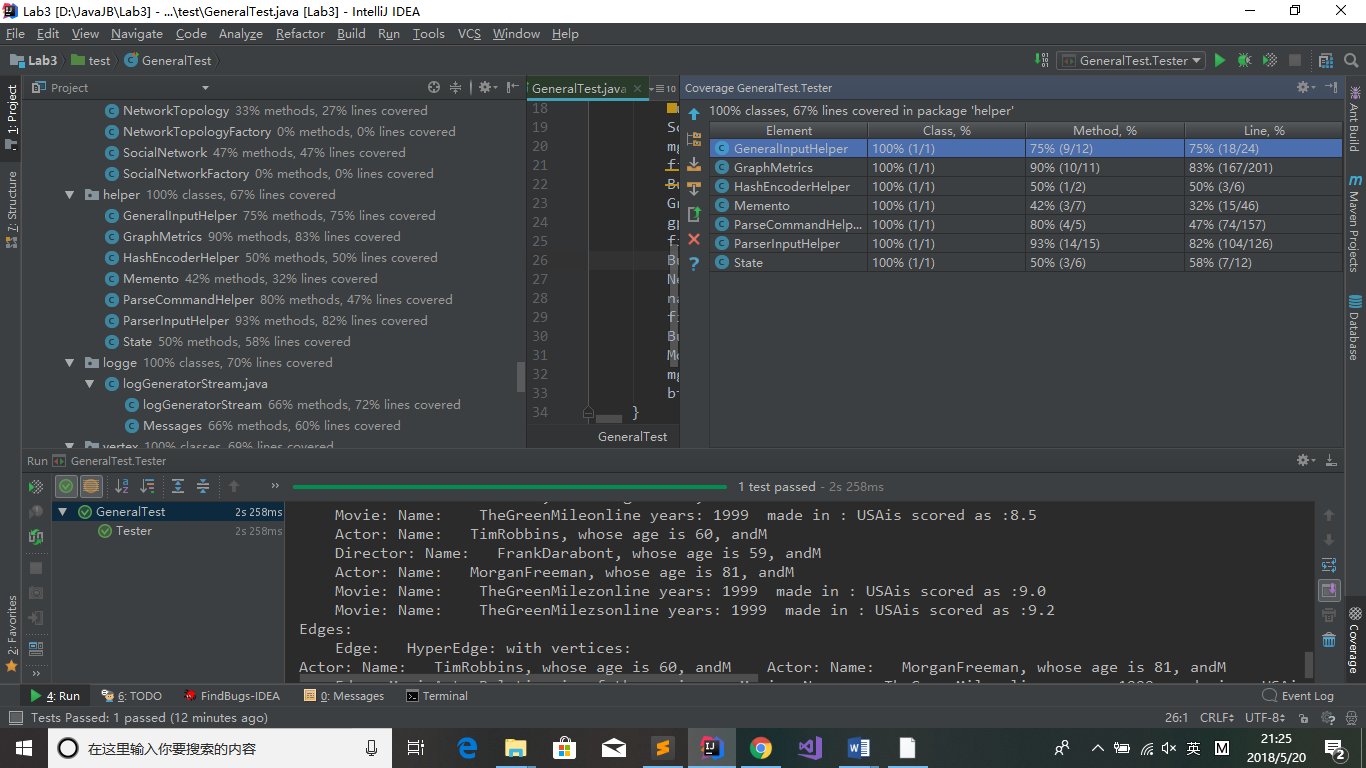
### 测试运行结果与覆盖度报告

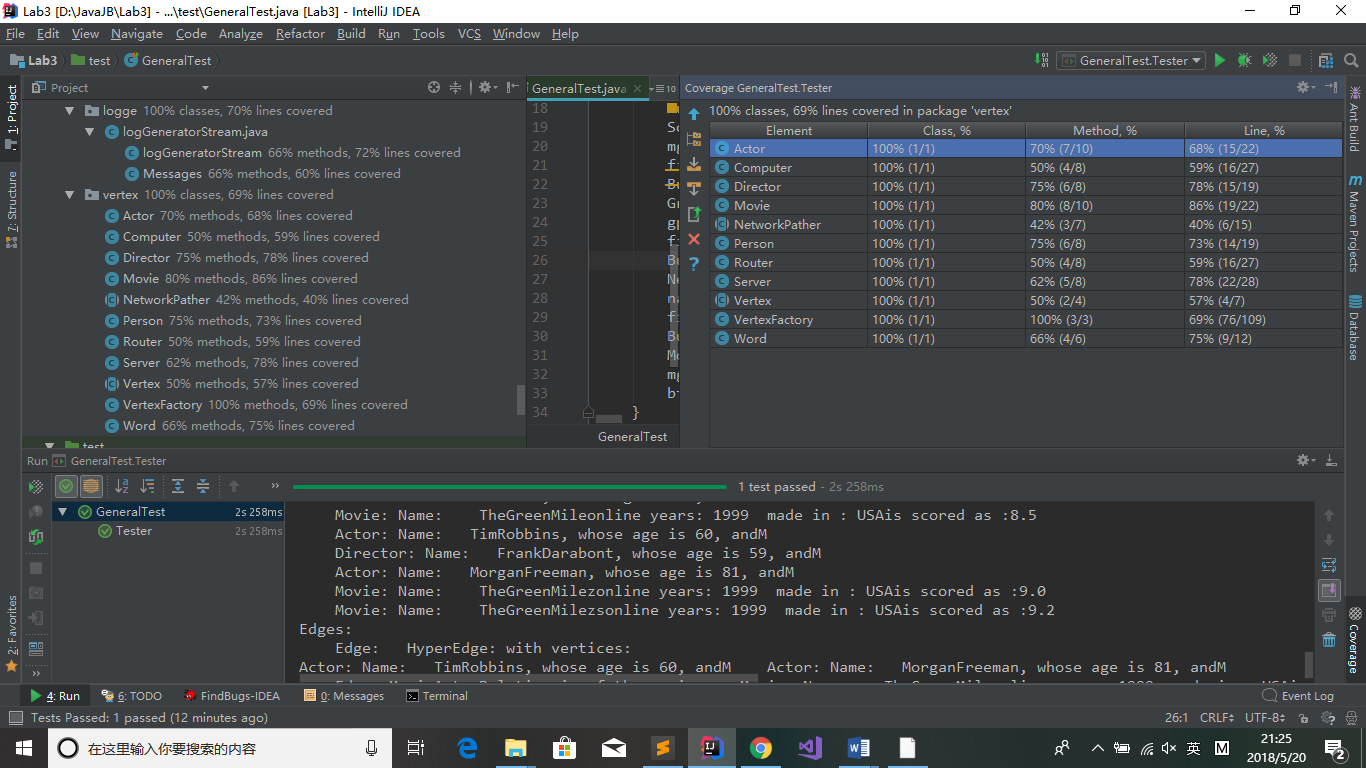












## FindBugs tool（可选）

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。

The warning found by FindBugs have fixed.

## Debugging

Overall, I debugged 4 packages.

### 待调试程序

CalculatorGUI.java GUI arrangement, logical.

Circle.java: Area

Main.java: Loops, first place

Triangle.java: Area

Main.java: new file generated: data.dat

Trie.java: Initialize, hashcode, node value

Main.java: choice==0?, setDownloader, print position and choice, exclude useless part

WebDownloader.java: endIndex, reuse of extension

### 理解待调试程序的过程

Feel it with heart, try it with love.

### 发现并定位错误的过程

Test and trace back, understanding.

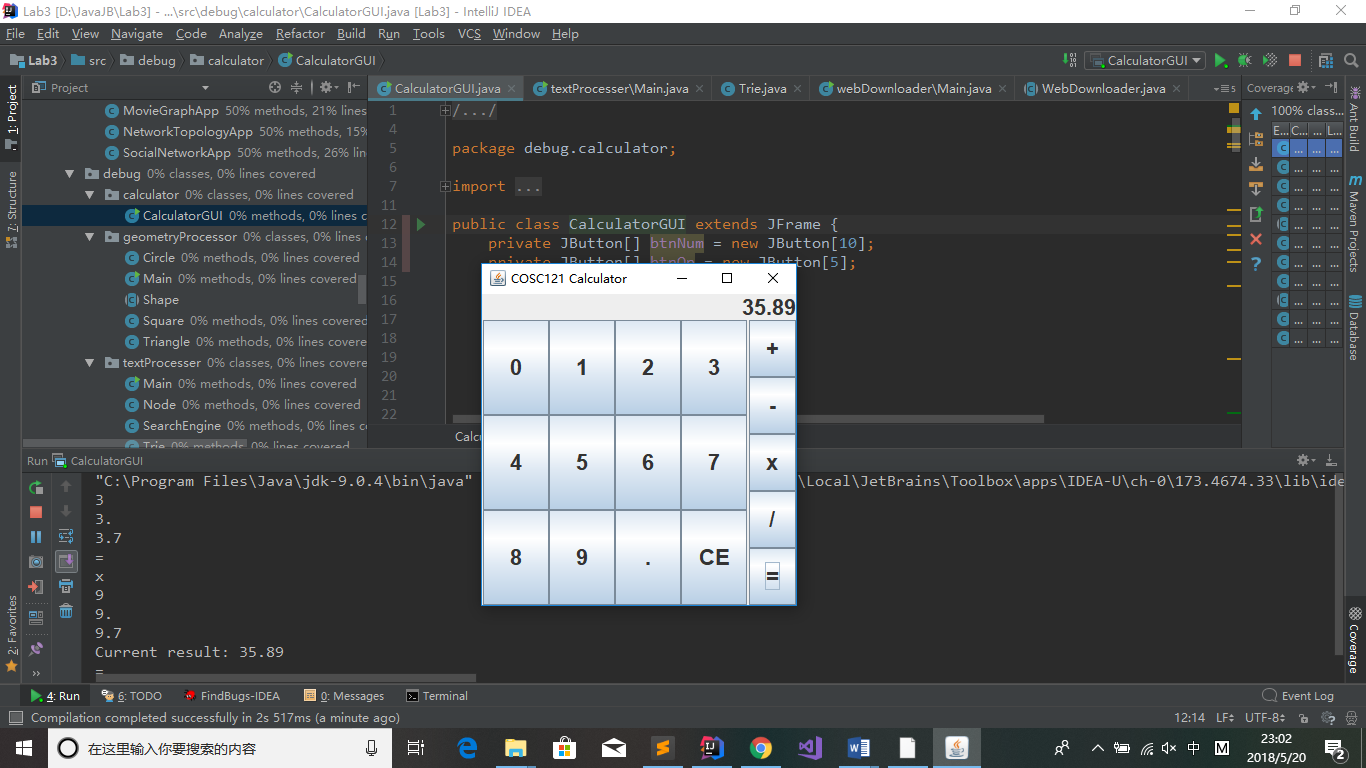
### 如何修正错误

Modify these lines.

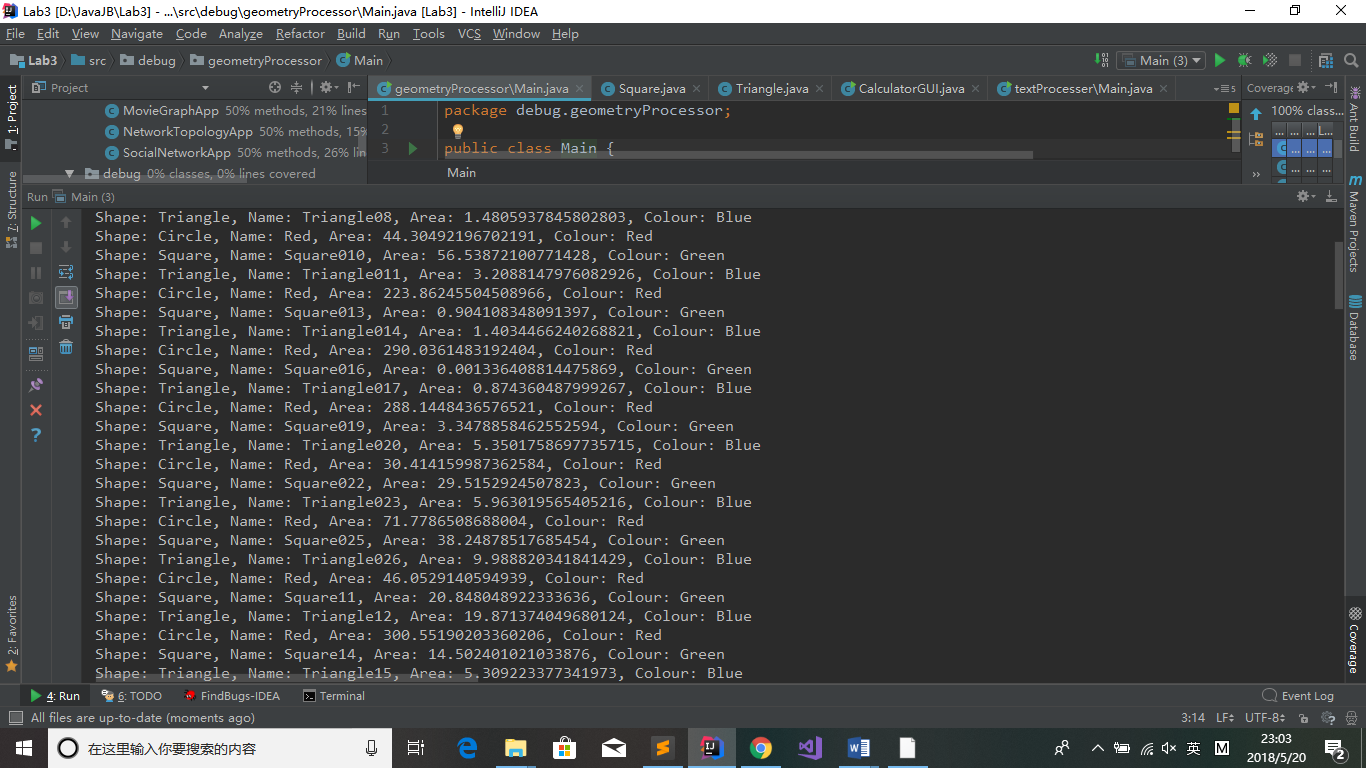
### 结果

All run with correctness.

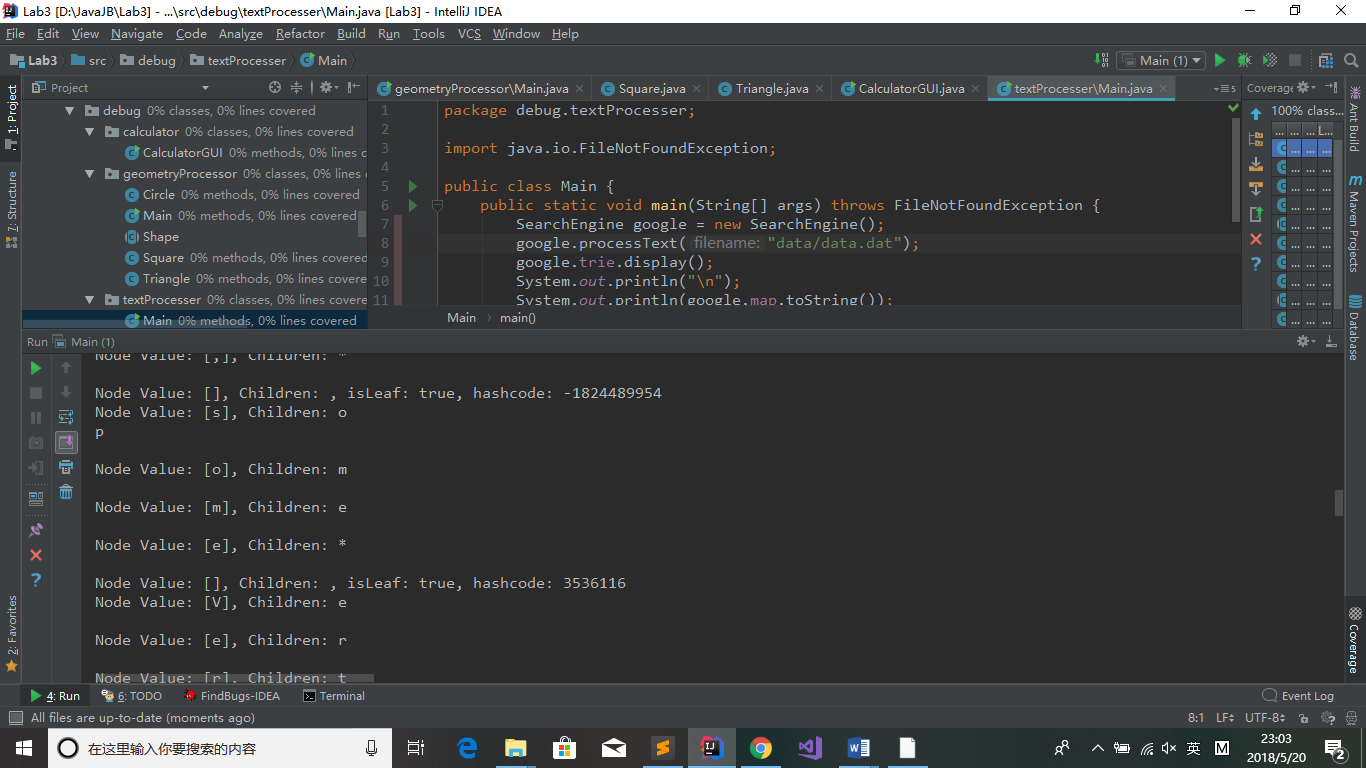
1.



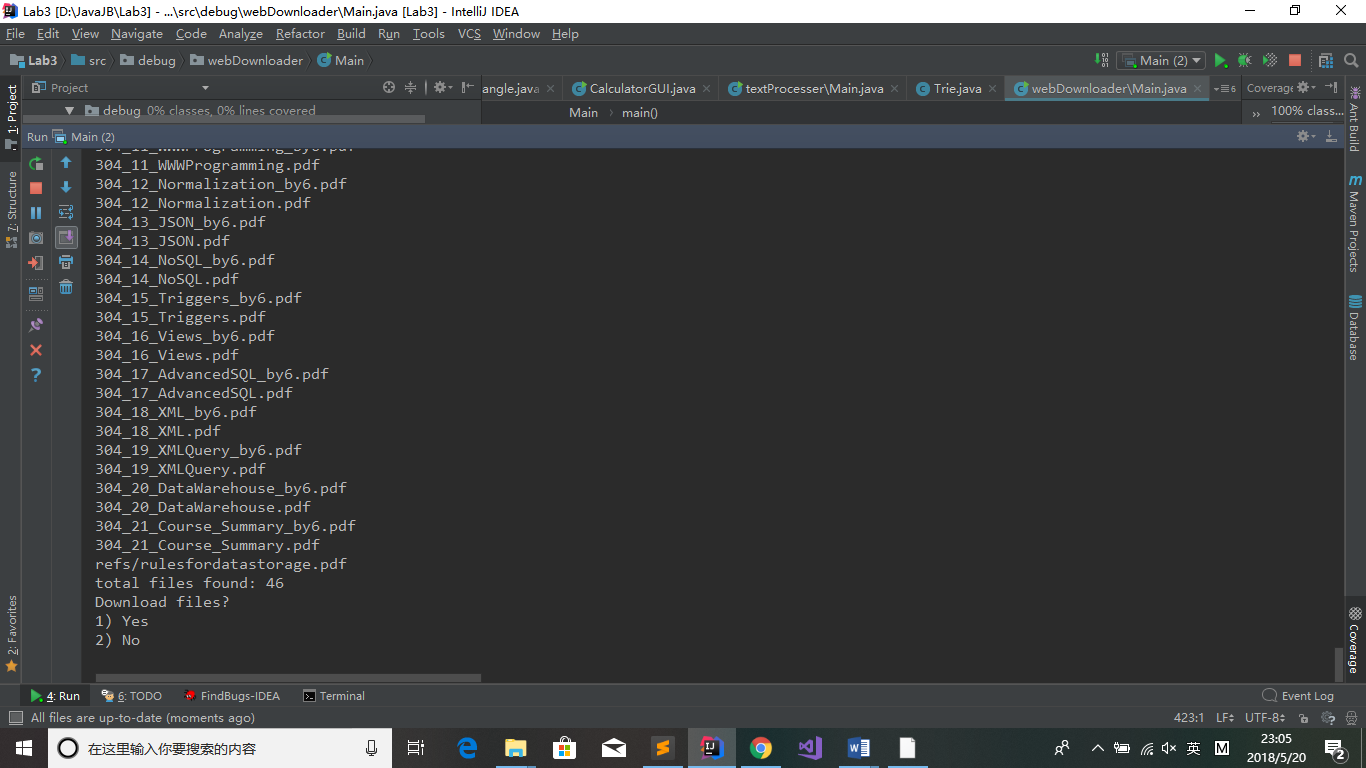
2.

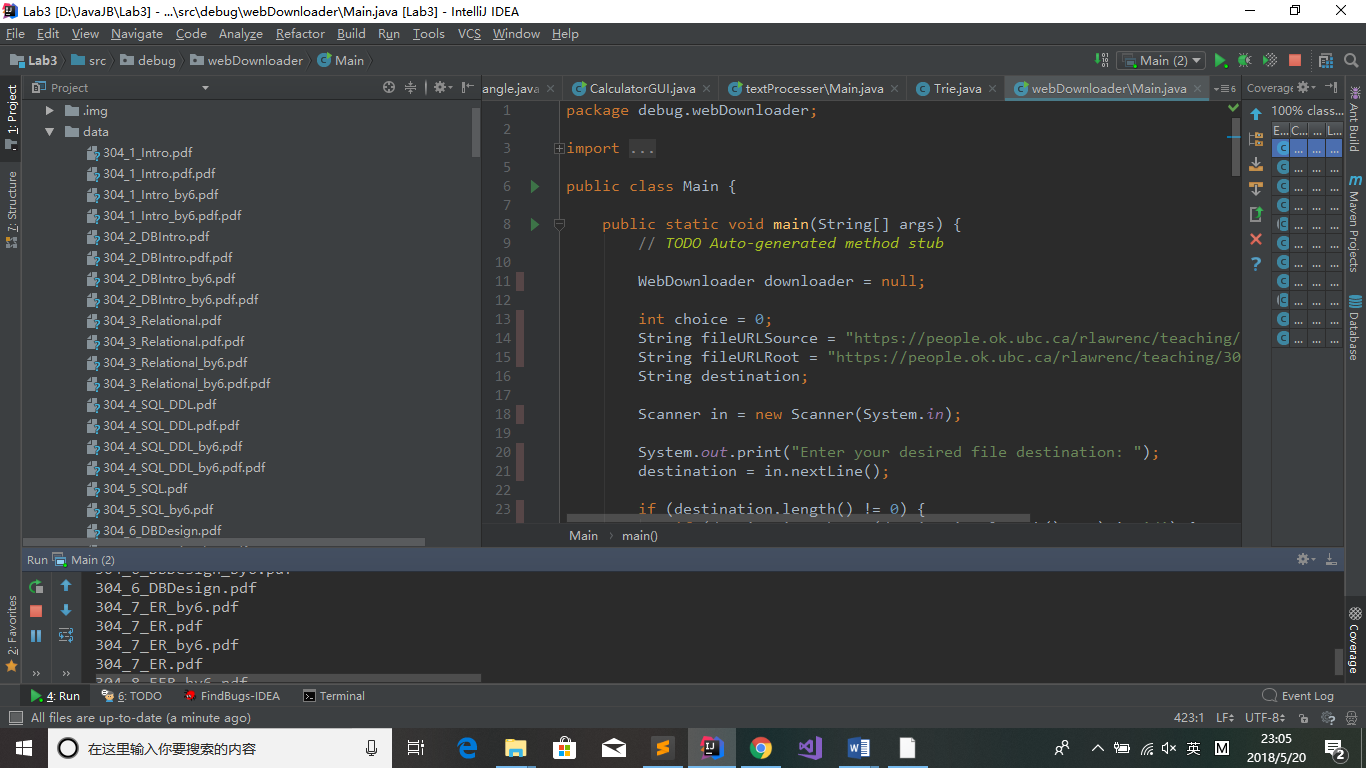


3.



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 |
| 5.20 | 0-24 | All | Done. |
|  |  |  |  |
|  |  |  |  |

# 实验过程中遇到的困难与解决途径

Google, StackOverflow

# 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。