

**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 | Arcwoc |
| 学号 | xxxxx |
| 班号 | xxxxxx |
| 电子邮件 | xxxxx |
| 手机号码 | xxxxx |

**目录**

目录

[1 实验目标概述 1](#_Toc26820)

[2 实验环境配置 1](#_Toc20522)

[3 实验过程 1](#_Toc24323)

[3.1 Error and Exception Handling 1](#_Toc3592)

[3.1.1 针对输入文本文件的异常/错误处理 1](#_Toc11835)

[3.1.2 针对输入图操作指令的异常/错误处理（可选） 7](#_Toc17930)

[3.2 Assertion and Defensive Programming 8](#_Toc21715)

[3.2.1 checkRep()检查invariants 8](#_Toc10124)

[3.2.2 Assertion保障pre-/post-condition 9](#_Toc27930)

[3.3 Logging 9](#_Toc29297)

[3.3.1 写日志 9](#_Toc29400)

[3.3.2 日志查询 10](#_Toc18707)

[3.4 Testing for Robustness and Correctness 11](#_Toc26112)

[3.4.1 Testing strategy 11](#_Toc32399)

[3.4.2 测试用例设计 11](#_Toc15439)

[3.4.3 测试运行结果与覆盖度报告 16](#_Toc11386)

[3.5 FindBugs tool（可选） 17](#_Toc24742)

[3.6 Debugging 18](#_Toc29514)

[3.6.1.1待调试程序 18](#_Toc13910)

[3.6.1.2理解待调试程序的过程 18](#_Toc8949)

[3.6.1.3发现并定位错误的过程 19](#_Toc1985)

[3.6.1.4如何修正错误 20](#_Toc13714)

[3.6.1.5结果 20](#_Toc17140)

[3.6.2.3 发现并定位错误的过程 23](#_Toc5122)

[3.6.3.3 发现并定位错误的过程 25](#_Toc15402)

[3.6.4.3 发现并定位错误的过程 27](#_Toc7406)

[4 实验进度记录 29](#_Toc11610)

[5 实验过程中遇到的困难与解决途径 29](#_Toc26779)

[6 实验过程中收获的经验、教训、感想 30](#_Toc9020)

# 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 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

**△**文件中存在不符合语法规则的语句：在使用正则表达式进行文本解析的过程中，程序一旦发现这些非法语句，应捕获异常，进行异常处理（提示错误），结束此文件的读取，允许用户选择其他文本文件。

处理方式：

先判断指令是否分为属性和label以及type两部分，然后再检查label和type两部分是否缺少。

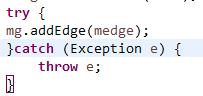


**△**边中使用的节点在节点部分未定义，进入自定义异常处理，提示错误，结束此文件的读取，允许用户选择其他文本文件。

处理方式：在加边的时候判断边内结点是否已经定义，如果没有定义则抛出异常。



判断是否存在，不存在则返回false，然后抛出异常，在工厂类中捕获异常，再次抛出，在工厂类中被捕获并做相应处理。



**△**为结点定义的属性的数目与特定应用的图的约束不符合：进入自定义异常处理，提示错误，结束此文件的读取，允许用户选择其他文本文件。

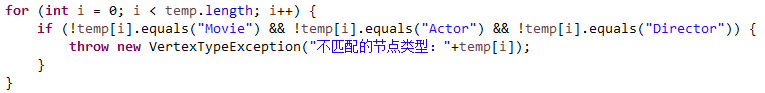
处理方式：

判断文档中属性值与预期是否符合，不符合则抛出异常。



**△**在某种类型的图应用中引入了不应出现的节点类型，例如在GraphPoet图中出现了类型为Person的节点：进入自定义异常处理，提示错误，结束此文件的读取，允许用户选择其他文本文件。

处理方式：在匹配到VertexType的时候检查一遍，比如在MovieGraph中：



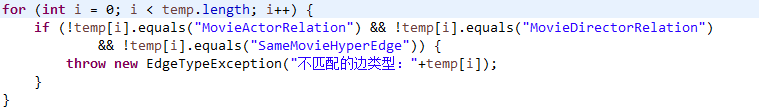
在建立顶点的时候再次处理：



都匹配不上的时候，抛出异常。

**△**在某种类型的图应用中引入了不应出现的边类型，例如在GraphPoet中出现了类型为FriendTie的边：进入自定义异常处理，提示错误，结束此文件的读取，允许用户选择其他文本文件。

处理方式：在匹配到EdgeType的时候检查一遍，比如在MovieGraph中：



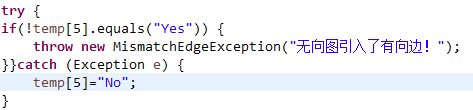
并且在建立边的时候再次处理（超边建立之前也要做类似判断）：



当提取的边类型与所有允许出现的边类型不一致时，抛出异常。

**△**在无向图中引入了有向边：自动去除这些边的方向，继续向下执行。

处理方式：在无向图中匹配到边的时候提取出是否有权的分量是”Yes”，则抛出异常并且捕获异常，更正错误，继续执行。



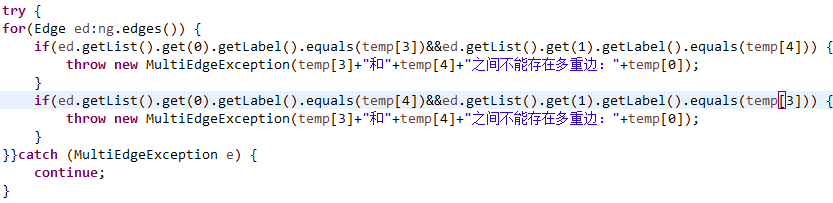
**△**在有向图中引入了无向边，例如SocialNetwork中出现了有向边：进入自定义异常处理，提示错误，结束此文件的读取，云南需用户选择其他文本文件。

处理方式：在匹配到边的时候发现是否有权的分量是用的”No”，则抛出异常，并给出异常信息：



**△**在单重图中存在了多重边：保留两个节点之间的第一条边，并且去除第2,3,…条边，继续向下执行。

处理方式：在匹配到一条新的边时，检查这条边的两个端点之间是否存在已有的边，如果存在则抛出异常，并且忽略此边，继续往下读取。



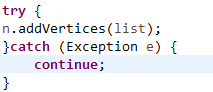
**△**在不应该存在边的两个节点之间存在了边：直接忽略这种非法边，继续向下执行。

比如在网络拓扑图中出现了环或者端点类型同为Conputer/Server的边：

处理方式：

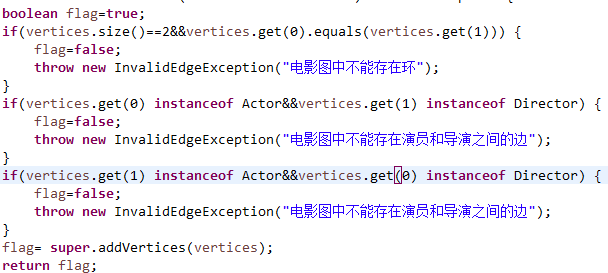


如果捕获到异常，则不再进行循环下面的操作，直接进行下一行的读取。

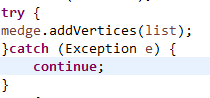


在电影图中存出现了演员和导演之间的边：

处理方式：

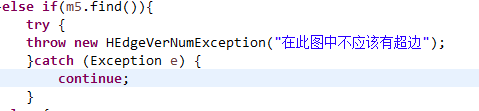


如果捕获到异常，则不再进行循环下面的操作，直接进行下一行的读取。



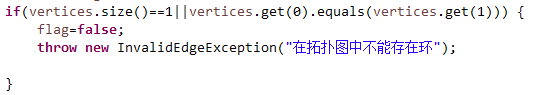
**△**在不应该出现超边的图中加入了超边：直接忽略非法边，继续向下执行。

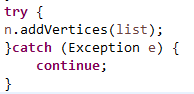
在不应该有超边的图中也进行超边的匹配，如果匹配到，则抛出异常。并且自动捕获异常，忽略该边，继续执行。



**△**在不应该出现loop的图中出现了loop：直接忽略这种非法边，继续向下执行。

处理方式：在社交图，拓扑图，电影图中都不能出现环，所以在向边中加入顶点的时候先判断端点的Label是否相同，如果相同则说明有环，抛出异常，并且在工厂类中catch异常，不再进行下面的操作，直接进行下一次循环迭代。





**△**某超边中包含的结点数目小于2：进入自定义异常处理，提示错误，结束此文件读取，并且允许用户选择其他文本文件。

处理方式：

在匹配到超边并且提取出相应信息后，做出如下判断。



如果超边数目少于2个，抛出异常，并给出异常信息。

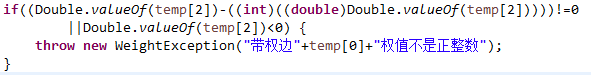
**△**带权边未能给出权值：进入自定义异常处理，提示错误，结束此文件的读取，允许用户选择其他文本文件。

处理方式：在提取出边的权值后，判断是否等于-1，如果该边是有权边，并且权值为-1，则抛出异常。



**△**带权边的权值不符合应用要求：进入自定义异常处理，提示错误，结束此文件的读取，允许用户选择其他文本文件。

在GraphPoet中出现权值非正整数的边，处理方式：：



在SocialNetwork中出现权值不在(0,1]之间的边，处理方式：



在NetworkToPology中出现权值为负数的边，处理方式：



**△**图，节点，边的Label不满足正则表达式(\w+)的要求：进入自定义异常处理，提示错误，结束此文件的读取，允许用户选择其他文本文件。

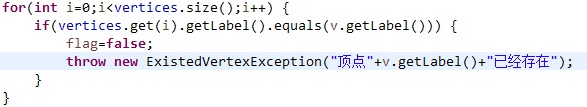
处理方式：为这种情况新建了异常类InvalidLabelException，并且在四个图的工厂类中加入如下代码：

IMG_256

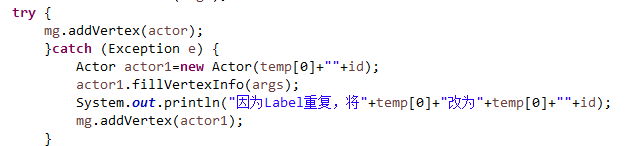
如果是顶点或者边，则分别在建边或者建点之前判断边和结点的Label是否合法。

**△**多个结点的label重复，多个边的label重复：通过自动修改后加的label来消除重复，重复的label依次变为label+id，并且提示用户做出的修改，继续执行。

处理策略：



在遇到已经存在相同label的结点时，抛出异常。



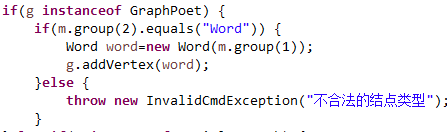
在catch到异常后，修改label，重新添加，并且向用户报出异常信息。

有重复边的处理方法相同。

### 针对输入图操作指令的异常/错误处理（可选）

**△**向图中加入新的结点：vertex --add label type

要检查指令中的type是否符合要求，如果不符合，则抛出异常。



**△**向图中加入新的边：edge --add label type [weighted=Y|N] [weight] [directed=Y|N] v1 v2

检查该指令中type代表的结点类型在图中是否应该存在，如果不存在则抛出异常。检查权重相关参数是否符合要求，若不符合则抛出异常，检查边的方向的参数是否有误，如果有误则抛出异常。检查该指令中的两个结点是否存在于图中，如果不存在抛出异常。

**△**修改结点属性：vertex --change label Attributes(e.g. age/sex/year)　to attrs

检查要改变属性的结点在图中是否存在，如果不存在则抛出异常。检查要改变的属性是否在该节点中存在，如果不存在则抛出异常。检查修改后的属性值是否符合要求，如果不符合要求则抛出异常。

**△**修改边的权重：edge --reweight label to x

检查要修改的边在图中是否存在，如果不存在则抛出异常。

**△**修改边的方向：edge --redirect label

检查要修改的边在图中是否存在，如果不存在则抛出异常。

**△**加入新的超边：hyperedge --addEdge label vertex1 vertex2...

检查要加入的超边包含的顶点个数是否大于等于2个，如果不满足，则抛出异常。

检查要加入的超边包含的顶点是否已经存在于图中，若不存在，则抛出异常。

**△**删除超边中的某个结点：hyperedge --remove edgelabel vertexlabel

检查超边和要删除的超边的顶点是否都存在于图中，若不存在，则抛出异常，并且在删除后，判断超边是否仍然满足顶点大于1个，如果不满足，则删除超边，抛出异常。

**△**向超边中加入顶点：hyperedge --add edgelabel vertexlabel

检查要加入的顶点是否存在于图中，若不存在，则抛出异常。检查超边是否存在，若不存在，则抛出异常。

**△**相关的计算：compute v1 v2/null

检查第一个参数是否为空或者合法，如果不合法，则抛出异常。

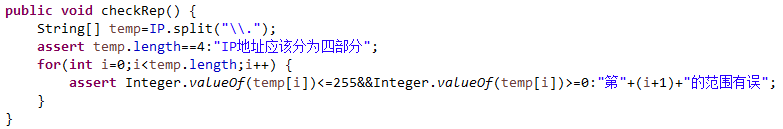
## Assertion and Defensive Programming

### checkRep()检查invariants

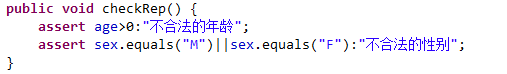
Vertex(检查标签)



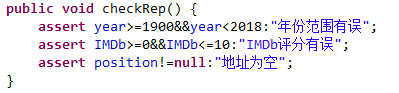
Computer,Router,Server（检查IP地址）



Person，Actor，Director（检查年龄和性别）



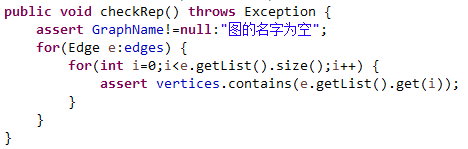
Movie（检查地点，年份和评分）



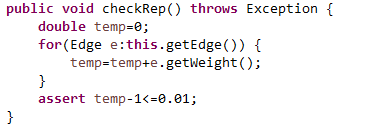
Edge（检查标签和年份，检查超边的顶点是否大于等于2）



ConcreteGraph（检查标签，以及边的端点是否在图中）



SocialNetwork（检查边的权重之和是否为1）



### Assertion保障pre-/post-condition

①在Vertex的构造函数中加入：**assert** label.matches("\\w+");

判断label是否满足正则表达式的要求。

②在graph的setGraphName()中加入**assert** graphName.matches("\\w+"):"图名字不符合正则表达式要求";判断图的名字是否符合要求。

③在edge的构造函数中加入**assert** label.matches("\\w+"):"边的label不符合正则表达式要求";判断label是否满足正则表达式要求。

③在edge的子类中加入assert语句保证参数数目正确，比如在Actor中加入：

**assert** args.length==2:"参数数目错误";

## Logging

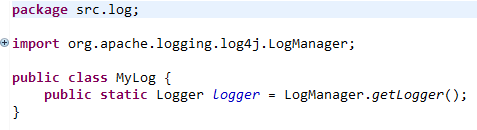
### 写日志

本实验用的是第三方库log4j来实现日志记录功能。

配置信息如下：



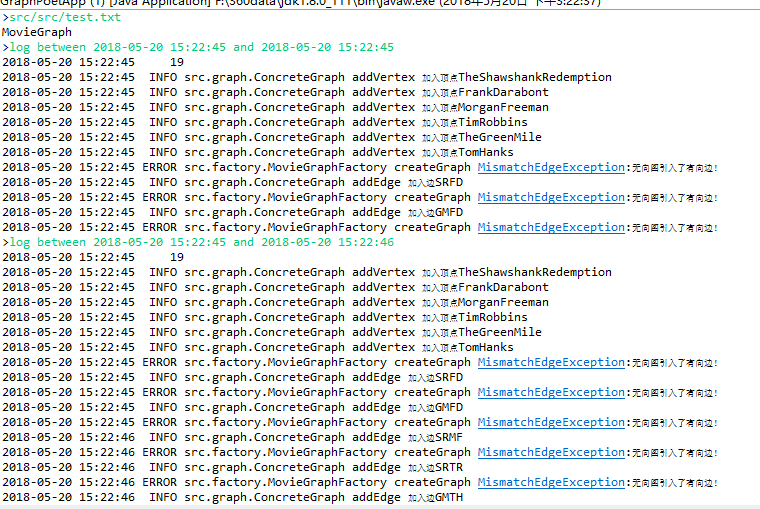
实现了一个MyLog类，里面有属性logger，满足在工程内任意位置可以记录操作和错误。



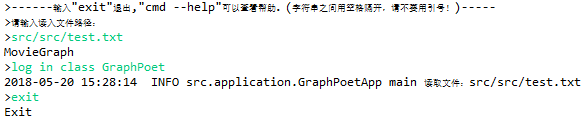
然后在其他类的每个操作和异常处向日志加入异常和操作，直接将日志信息写入src/src/log/graph.log中去。

### 日志查询

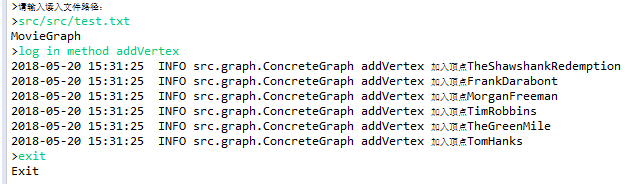
**按时间查询：**



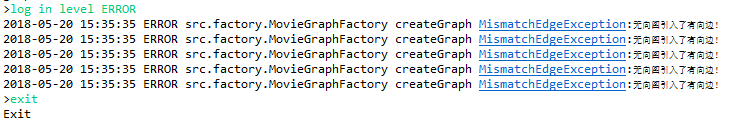
**按类查询：**



**按方法查询：**



按LEVEL查询：



## Testing for Robustness and Correctness

### Testing strategy

用不同的文件去建立GraphPoet图，考虑所有可能的异常情况，并且在Junit测试中捕捉异常，检验是否符合预期。

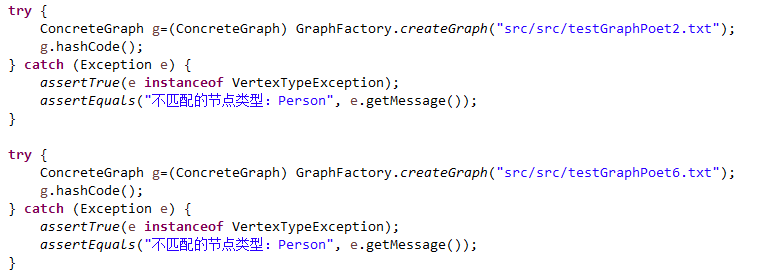
### 测试用例设计

测试ConcreteGraph时用的是GraphPoet的文档，一共分19个文档，分别覆盖：①test3.txt表示正常情况

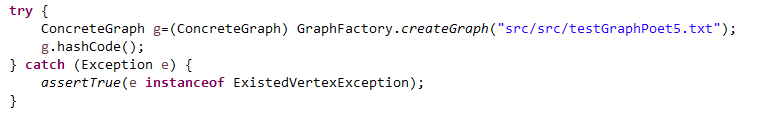
②testGraphPoet1.txt，testGraphPoet3.txt，testGraphPoet4.txt分别测试图的名字中，顶点的label中，边的label中有不合法字符时的情况抛出InvalidLabelException异常以及异常信息。



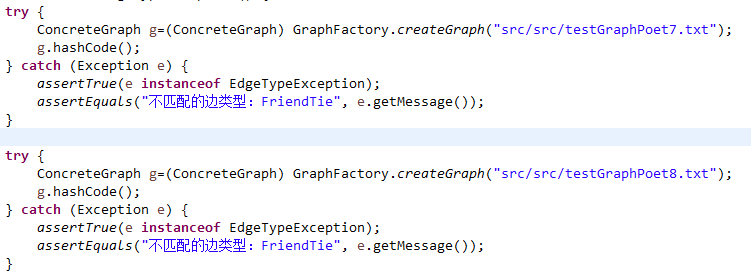
③testGraphPoet2.txt，testGraphPoet6.txt前者测试图中VertexType与图的类型是否匹配，如果不匹配，则抛出VertexTypeException异常。后者测试图中结点的类型判断，结点的类型与图中应有类型不匹配，则抛出VertexTypeException异常。



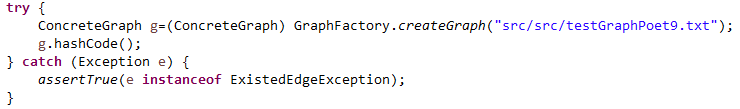
④testGraphPoet5.txt重复加入ask顶点，则会抛出ExistedVertexException异常，表示该顶点已经存在。



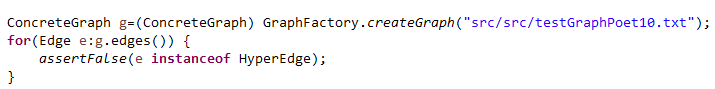
⑤testGraphPoet7.txt，testGraphPoet8.txt分别测试EdgeType中的边类型和要加入的边的类型是否符合该图接受的边的类型，如果不匹配，则抛出EdgeTypeException异常。



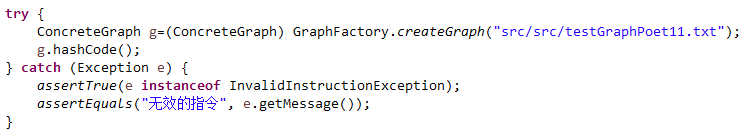
⑥testGraphPoet9.txt测试重复加入边是否抛出ExistedEdgeException异常



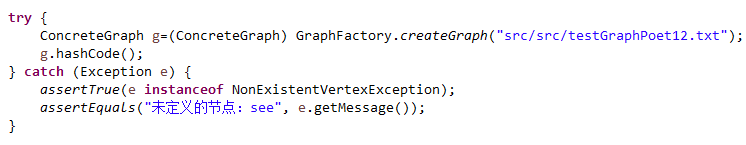
⑦testGraphPoet10.txt测试向图中加入超边是否能够成功。预期结果是忽略超边。



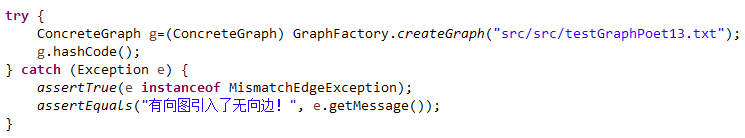
⑧testGraphPoet11.txt测试在文件中出现完全不合法的乱码指令时，是否抛出InvalidInstructionException异常。



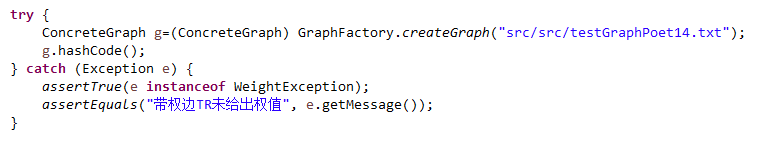
⑨testGraphPoet12.txt测试向图中加入一条新的边，边的端点包括未定义的顶点的情况，是否抛出NoExistentVertexException异常。



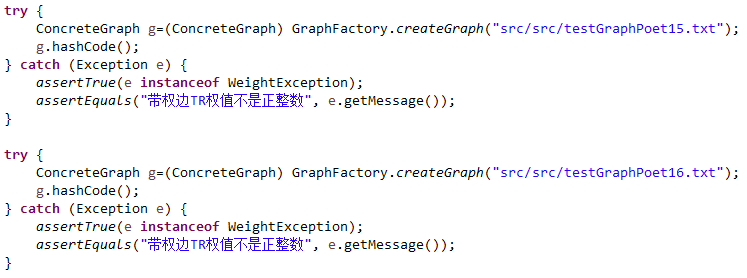
⑩testGraphPoet13.txt测试向有向图中加入无向边是否抛出异常MismatchEdgeException。



1. testGraphPoet14.txt测试给带权边但是不赋边的权重是否抛出异常WeightException。



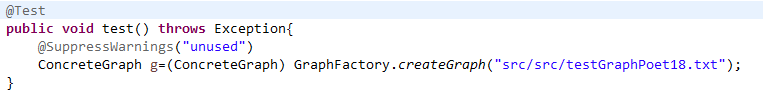
1. testGraphPoet15.txt测试给出的带权边的权值不是正数的情况下是否抛出WeightException。testGraphPoet16.txt测试给出的带权边的权值不是整数的情况下是否抛出异常WeightException。



1. testGraphPoet17.txt测试向图中加入多重边是否抛出MultiEdgeException异常.

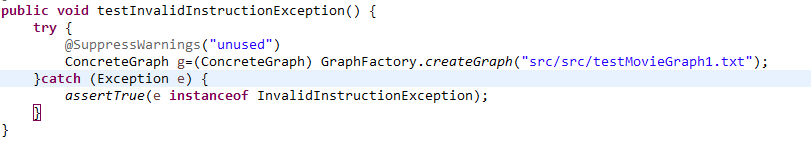


1. testGraphPoet18.txt测试图中只有类型没有名字没有结点没有边的时候是否正常运行。



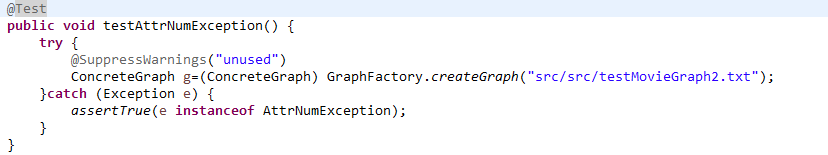
补充测试①：

测试当结点的label或者type缺失的时候是否抛出InvalidInstruction异常。



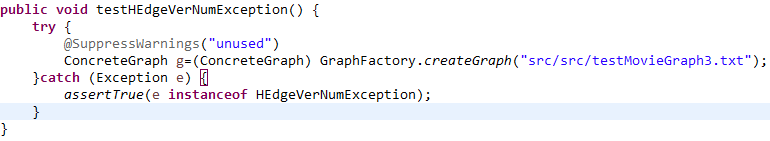
补充测试②：

测试结点的属性的数目缺少的时候是否抛出AttrNumException异常.



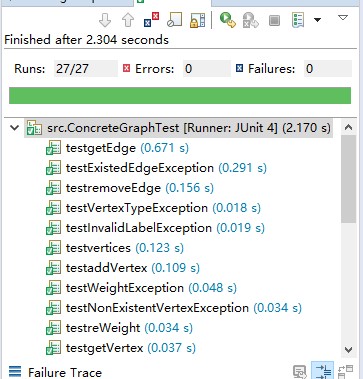
补充测试③：

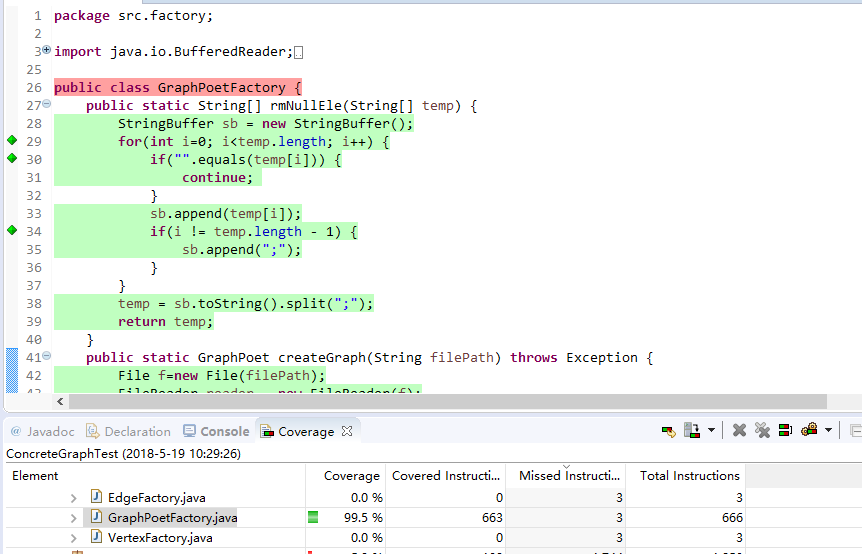
测试超边中只有一个顶点是否会抛出HEdgeNumException异常。



### 测试运行结果与覆盖度报告

ConcreteGraph传入GraphPoet的测试文件测试结果如下：





## FindBugs tool（可选）

①toString函数在一些情况下可能返回null，导致其他程序中断。

修改：把null改为””空字符串。

②用等号来比较两个浮点数大小，可能导致错误。

修改，把等号改为Math.abs(x - y) < .0000001 的形式

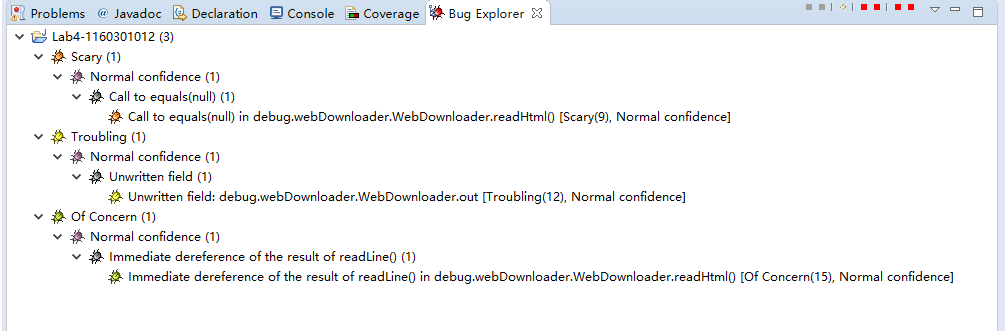
③把一些String类型的对象初始化为null，可能导致错误。

修改，把null改为一些有意义的字符串。

不良习惯：经常把一些引用初始化为null

修改结果：

只剩下3.6也就是debug部分还有问题。四个图以及其相关的代码隐藏的bug均被处理。



## Debugging

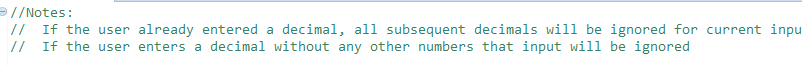
**3.6.1** **CalculatorGUI**

### 3.6.1.1待调试程序

calculator包下的CalculatorGUI.java

### 3.6.1.2理解待调试程序的过程

首先查看程序，注意到：



然后查看程序具体代码：

先看main()函数，很简单，只有一行，新建了一个计算器实例，并且将setVisible设置为true。然后下面开始看每一个部分：

result存储计算结果，buffer存储数字输入，operation存储运算符号，init记录计算器的变量状态，看变量是否被有效存储。

然后下面的代码对GUI初步布局，设置了窗口大小，按钮和文本框。并且将它们添加到两个JPanel对象上，pnlNum上放置数字按键，pnlOp上放置运算操作符。并且将它们放在合适的位置上。这些都在CalculatorGUI的构造方法中执行，每当一个CalculatorGUI类被新建，这些操作都得到实现。

接下来就是对用户输入的处理以及计算和显示。

**①public** **void** processDigitInput(JButton value)

该方法处理输入的数字和小数点。如果之前的输入中没有小数点，并且目前从按键读入了小数点，那么我们将为输入的串中加入小数点。也就是说，如果之前已经有了小数点，那么就忽略后序的小数点。

**②public** **void** processOperationInput(String in)

该方法处理输入的操作符号，利用用户的输入相应的设置operation参数。

**③public** **void** performOperation()

该方法根据状态变量执行适当的算术运算。

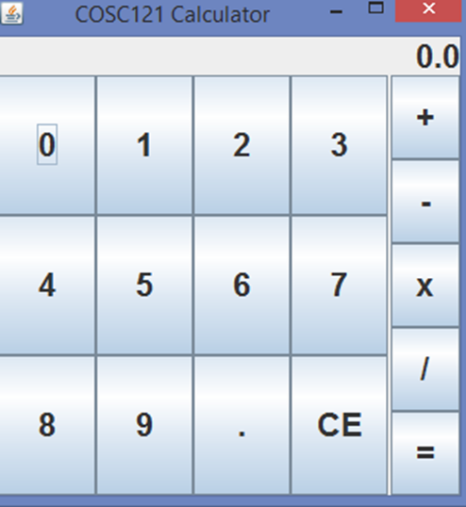
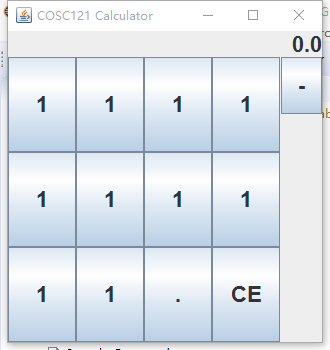
**④public** **void** clear()

该方法重置了计算器的状态，恢复到默认状态。

最后是三个事件监听类，分别是OpActionListener,DigitActionListener和CEActionListener.分别用来监听运算操作符按钮，数字按钮，以及CE按钮。

### 3.6.1.3发现并定位错误的过程

错误①：首先运行了一下程序，发现界面如下：

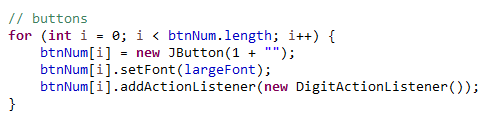


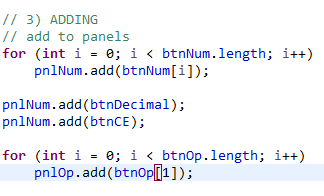
与样例比较，初步发现GUI如下问题：

数字按键全部为1

缺少减法符号以外的运算符号

找到问题在于按钮循环初始化的时候，参数应该是i而不是1.





错误②：阅读代码，很容易发现123~130行以及150~176行符号匹配有问题，运算符号与相对应的处理不相匹配，符号错乱，并且在165行的除号的情况下的处理缺失。

错误③：运行代码很容易发现，在performOperation()中的if-else语句中，总是选择了第一个分支，很容易发现，是init在每次操作后都没有被更新的结果。

错误④：上面的错误全部修改后，运行程序发现控制台的计算结果是正确的，但是在GUI中的更新不够同步。

### 3.6.1.4如何修正错误

错误①修改：

67行：btnNum[i] = **new** JButton(1 + "");

改为：btnNum[i] = **new** JButton(i + "");

104行：pnlOp.add(btnOp[1]);

改为：pnlOp.add(btnOp[i]);

错误②修改：

将123~130行以及150~176行的符号匹配修正，并且在165行下面加上：

result = result / Float.*parseFloat*(buffer);

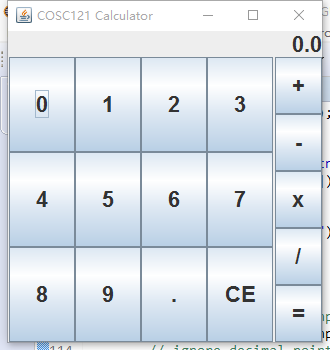
错误③修改：

在第一个分支**if** (!init && !buffer.equals(""))的最后加上init=**true**;

错误④修改：

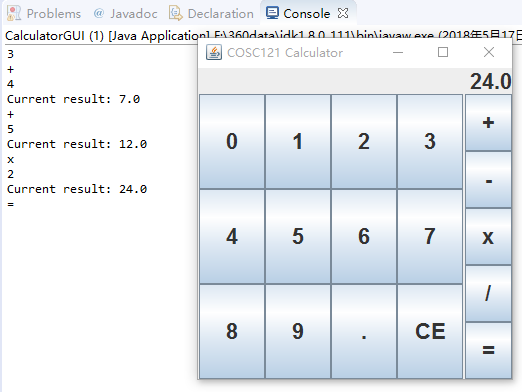
在每次用户输入新的数字后，将其显示到文本框内，在processDigitInput(JButton value)的if语句的末尾加上：txtResult.setText("" + buffer);

### 3.6.1.5结果

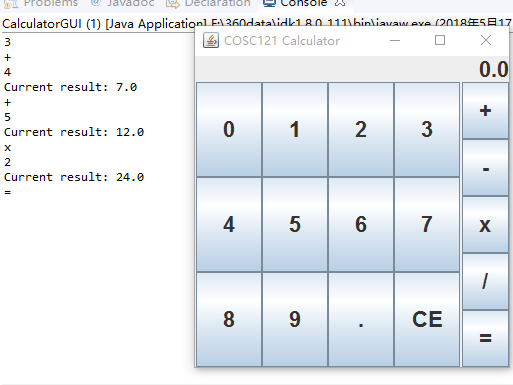


运行程序：

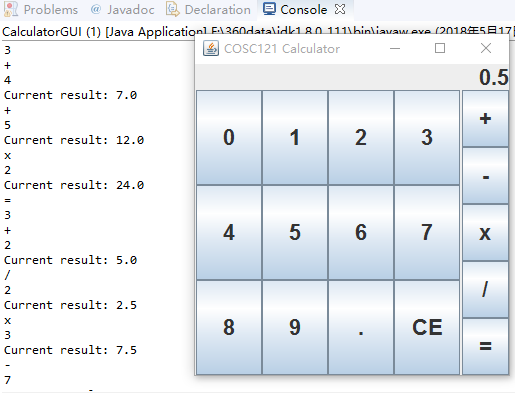
①（3+4+5）\*2



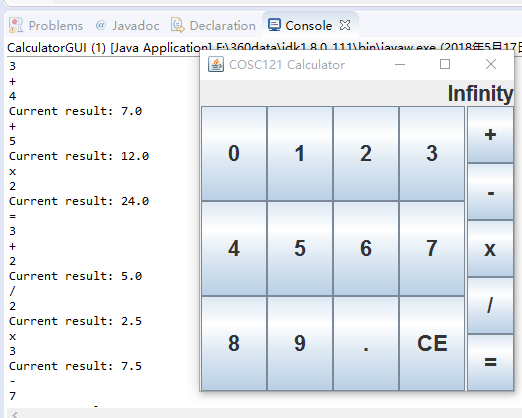
②点击CE



③计算（3+2）/2 \*3-7



④计算3/0



**3.6.2 geometryProcessor**

**3.6.2.1 待调试程序**

geometryProcessor包下的java文件

**3.6.2.2 理解待调试程序的过程**

Main.java：main方法是程序执行的入口，先声明了一个3行n列的数组，数组元素是Shape类型的。然后初始化数组，并且按照数组打印数组所有元素，然后按照数组元素的形状打印数组。

Shape.java:定义了Shape接口，里面有七个没有实现的方法。

Circle.java：定义了圆形类，实现了接口里的方法。

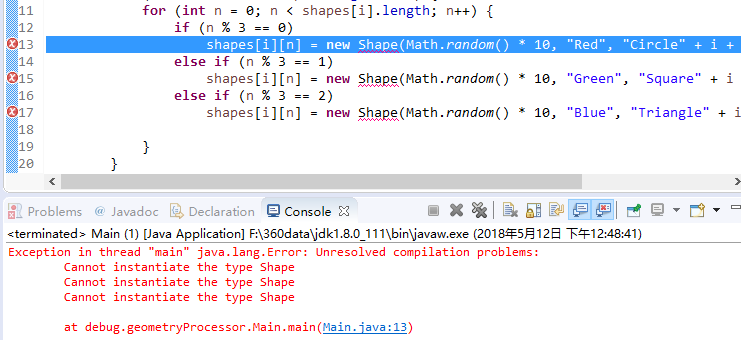
Square.java：定义了正方形类，实现了接口里的方法。

Triangle.java：定义了三角形类，实现了接口里的方法。

### 3.6.2.3 发现并定位错误的过程

错误①：读代码很容易发现，Main.java里的数组初始化有错误。

错误②：运行程序可以看到：



打开Shape.java的代码可以看到，Shape是一个接口，不能被实例化。

错误③：Circle.java中有未实现的接口方法**public** String getColour()

错误④：Square.java中有未实现的接口方法**public** **double** getArea()

错误⑤：Triangle.java中三角形面积计算错误

错误⑥：Circle.java中圆面积计算错误

**3.6.2.4 如何修正错误**

错误①修改：

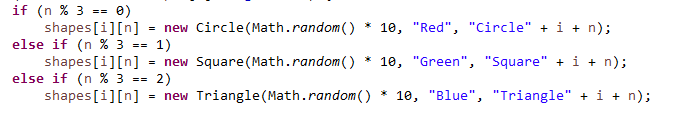
11行**for** (**int** n = 0; n < shapes[n].length; n++) {改为

**for** (**int** n = 0; n < shapes[i].length; n++) {

错误②修改：

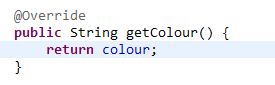
根据代码可以知道：

当n%3等于0,1,2时，Shapes[i][n]分别为圆形，正方形和三角形，所以修改12~17行代码为：



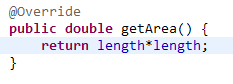
错误③修改：

在Circle.java末尾加上：



错误④修改：

在Square.java末尾加上：



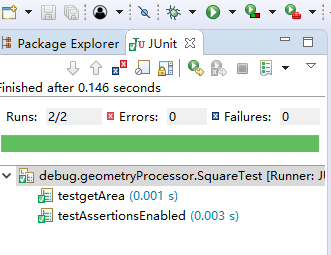
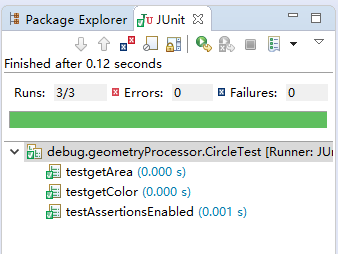
错误⑤修改：Triangle.java中第30行改为：

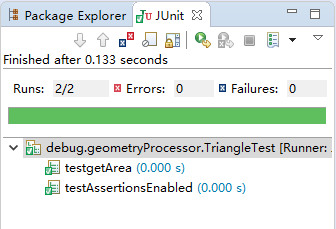
**return** (Math.*sqrt*(3) / 4) \* length\*length;

错误⑥修改：Circle.java中第26行改为：

**return** Math.***PI*** \* length\*length;

**3.6.2.5结果**





**3.6.3 textProcesser**

**3.6.3.1 待调试程序** textProcesser包下的java文件

**3.6.3.2 理解待调试程序的过程**

Main.java新建一个SearchEngine类并且读取文件，然后调用该实例的tire属性的display方法，并且打印出该实例的map属性里的内容。

Node.java是一个Node类，有Trie类型的属性t，表示当前的结点所属，c存储字符列表，children存储这个结点的所有子结点，isLeaf表示该结点是否是一个叶结点，leafValue表示如果该节点是叶结点，该结点的值。addChild()方法向该结点加入字符，addLeafChild()方法向该树加入叶结点，getTire()方法得到当前结点所属的Tire，childValues()方法可以得到一个包含所有孩子结点字符的列表。

SearchEngine.java里有一个map属性，存储一个单词在所有页面的出现次数，trie存储一个text文档的所有单词，pagecounter计数，pageDelimiter用来确定是否要递增pageCounter,processText读文件并且将信息存储在map中。

Trie.java的addWord()方法向trie中加入单词，display()方法表示当前trie的状态，hasLeaf()打印所有的结点在一个结点集合中，并且表示它是否有一个叶结点作为子结点。

### 3.6.3.3 发现并定位错误的过程

错误①：Tire.java中30行Node Value应该是n.c而不是n

错误②：Tire.java中37行hashcode应该用n.leafValue而不是n

错误③：SearchEngine.java中18行PageCounter初始化错误。

错误④：SearchEngine.java中在 processText(String filename)中遇到pageDelimiter没有递增pageCounter

错误⑤：Tire.java的32行循环终止边界错误。

错误⑥：Tire.java的addWord()方法中未对pointer设置。

**3.6.3.4 如何修正错误**

错误①改正：将Tire.java中30行改为：

System.***out***.print("Node Value: " + n.c + ", Children: ");

错误②改正：将Tire.java中37行改为：

System.***out***.print(", isLeaf: " + n.isLeaf + ", hashcode: " + n.leafValue);

错误③改正：将SearchEngine.java中18行PageCounter初始化为1

错误④改正：在SearchEngine.java中39行加上pageCounter++;

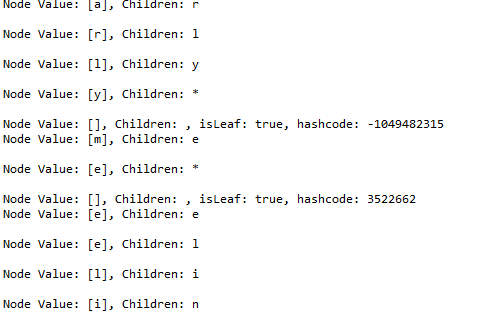
错误⑤改正：在Tire.java的32行改为：

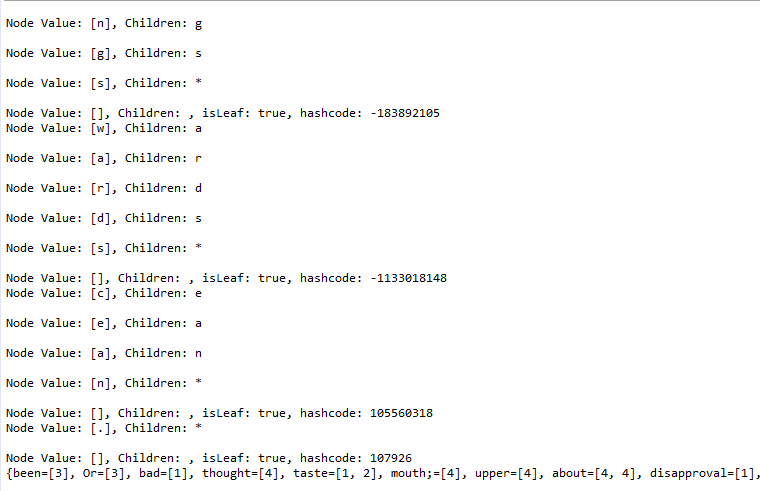
**for**(**int** i = 0; i < n.childValues().size(); i++) {

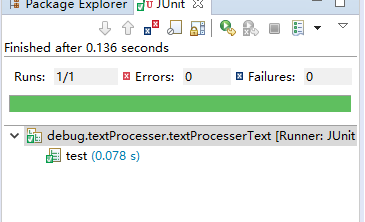
错误⑥改正：在Tire.java的addWord()方法里第一行加上pointer=root;

**3.6.3.5结果**

**部分运行结果：**







**3.6.4 webDownloader**

**3.6.4.1 待调试程序**

webDownloader包下的java文件

**3.6.4.2 理解待调试程序的过程**

Main.java让用户输入要把文件下载到的路径，要下载文件的格式，下载之前是否打印文件名字等等，然后调用相应的类的方法，满足用户的需求。

WebDownloader.java是一个抽象类。里面有属性website，存储网站URL，fileExtension，存储文件扩展名，destination存储文件下载路径，fileRoot存储你要下载的文档在服务器上的路径，url存储URL，files存储这个类当前URL的文档列表，html存储当前网站的完整的html代码。

**①public** **void** init()

初始化该类的url，新建一个ArrayList实例赋值给files，并且调用readHtml()和findFiles()。

②**public** **void** readHtml()

读出当前URL对应网站的HTML代码并且存储成一个字符串。一行一行读取HTML代码直到读取到"</html>"为止。

③**public** **void** findFiles()

通过遍历HTML并且查找相符合的文件扩展名的文件，加入files列表。

④**public** **void** setFileDestination(String d)

设置目标路径

⑤**public** **void** setFileRoot(String r)

设置fileRoot

⑥**public** **void** downloadFiles()

遍历files列表，从服务器下载文件。

Mp3Downloader.java,PdfDownloader.java,TextDownloader.java分别是三个继承自WebDownloader类的具体的下载类。

### 3.6.4.3 发现并定位错误的过程

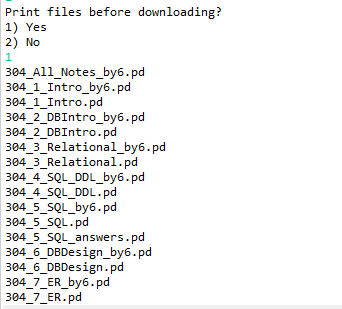
错误①：Main.java利用eclipse的静态检查，可以看到39行，45行，50行试图把抽象类WebDownloader实例化。

错误②：Main.java 31行，62行会造成死循环，88行同理，因为不适合的终止条件。

错误③：测试发现，在输入1的时候并没有在下载之前打印出文件信息，发现是73~78行的分支语句对应情况处理交叉错误。

错误④：测试发现在询问用户是否要下载文档之后的case语句也有不匹配的问题。

错误⑤：修改完以上代码后，运行发现出现以下结果：



经过定位发现是WebDownloader.java中第91行中对endIndex更新错误，导致后缀读入不完整

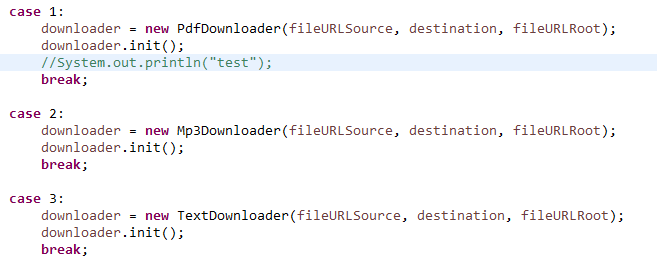
错误⑥：WebDownloader.java中70行计算files的大小的时候，减一是错误的。

错误⑦：WebDownloader.java中154行计算时间有误，毫秒与秒之间单位转化处理错误。

错误⑧：WebDownloader.java中下载文档部分路径有问题。

**3.6.4.4 如何修正错误**

错误①修改：



错误②修改：



88行做相同修改，31行做类似修改。

错误③修改：

在73~82行将case1和case2的处理语句互换。

错误④修改：

在99~108行将case1和case2的处理语句互换。

错误⑤修改：

将WebDownloader.java中第91行改为endIndex = i + 4;

错误⑥修改：

将WebDownloader.java中第70行改为System.***out***.println("total files found: " + files.size());

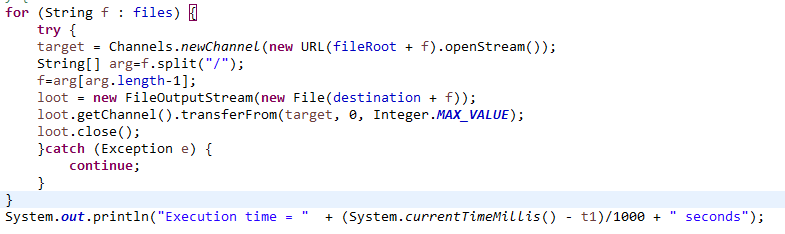
错误⑦修改：

将WebDownloader.java中第154行改为System.***out***.println("Execution time = " + (System.*currentTimeMillis*() - t1)/1000 + " seconds");

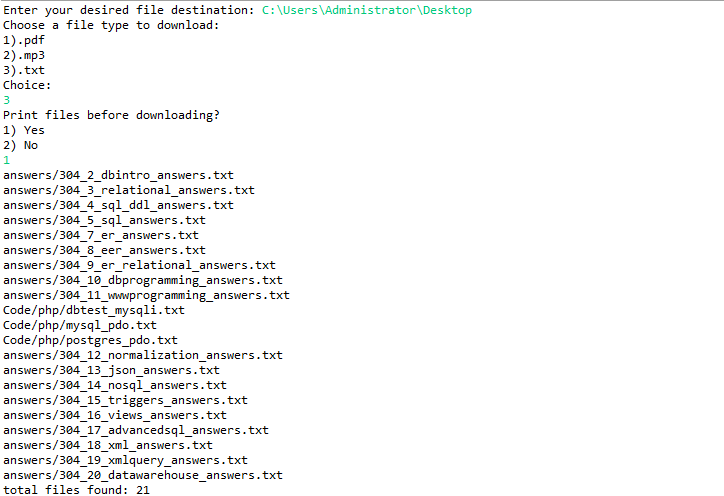
另外在Main.java的最后加上in.close();

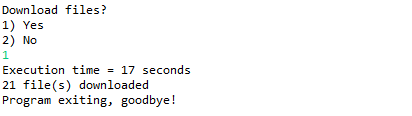
错误⑧修改：

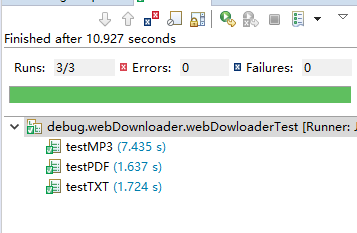
WebDownloader.java该部分改为：



**3.6.4.5结果**







# 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 |
| 5.7-5.10 | 空闲时间 | 完成异常的处理（指令和工厂方法） | 基本完成 |
| 5.11-5.14 | 空闲时间 | 完成异常的测试和日志的记录 | 基本完成 |
| 5.15-5.20 | 空闲时间 | 完成3.6的debug和前面的debug | 基本完成 |

# 实验过程中遇到的困难与解决途径

不了解日志的用法，通过查阅资料选择符合需求的第三方库，并且阅读其文档，了解该库的安装与配置，以及使用。

# 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？

健壮性是为了让程序的容错性更强，正确性是为了让程序出错的可能性更小。考虑健壮性的时候会考虑更多的要处理的输入的情况，考虑正确性则是要考虑对待确定的输入是否能够得出正确的输出。

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？

我认为这是划算的。为了应对小概率增加很多行的代码，但是却增强了程序的正确性和健壮性。但这往往不意味着程序效率的减弱，但是却为程序的维护提供了方便，因为好的异常处理能够让代码逻辑更加清晰，在出问题的时候更容易找到问题。

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？

让自己的程序能应对更多的异常情况是把各种情况由自己的程序来处理，在遇到非正常情况的时候，程序也能完成特定的行为。让客户端或者用户来承担正确性的职责则是把规则告诉用户，让用户自行遵守，以满足程序正确顺利执行。在规则比较繁杂的时候或者比较细节的地方，比如用户输入的格式，应该交给程序来完成异常处理。

1. 过分谨慎的“防御”（excessively defensive）真的有必要吗？

防御对于程序的健壮性是非常必要的。

1. 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？

断点和assert都是定位错误有效的方法，前者可以让我们看到随着程序运行，变量代表的值的变化，后者可以在容易犯错的地方及时防止我们犯错并且精准的定位错误的位置。Assertion和log技术可以帮助我更有效的定位错误。

1. 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？

充分的测试意味着对可能的输入的等价类比较全面的覆盖，对程序所有可能情况的分支都有覆盖。代码覆盖度100%不以为这100%充分的测试，因为代码与问题的两个不同的东西，代码覆盖度100%只意味着代码得到了充分的测试，不意味着问题的所有等价类得到了覆盖。

1. 关于本实验的工作量、难度、deadline。

本实验工作量较大，难度不大，deadline恰当。

1. 到目前为止你对《软件构造》课程的评价和建议。

《软件构造》是一门可以学到很多东西的课程。