****

**Java&.net软件开发**

**课内实习报告**

学号： /

班级： /

姓名： /

**中国地质大学（武汉）**

**地理与信息工程学院软件工程系**

**2019年 5 月**

目录

[1 实习1 1](#_Toc11617030)

[1.1 实习目的 1](#_Toc11617031)

[1.2 测试 1](#_Toc11617032)

[1.3 问题集 1](#_Toc11617033)

[1.4 思考和建议 4](#_Toc11617034)

[2 实习2 5](#_Toc11617035)

[2.1 实习目的 5](#_Toc11617036)

[2.2 问题集 5](#_Toc11617037)

[2.3 思考和建议 7](#_Toc11617038)

[3 实习3 7](#_Toc11617039)

[3.1 实习目的 7](#_Toc11617040)

[3.2 问题集 7](#_Toc11617041)

[3.3 思考和建议 10](#_Toc11617042)

[5 代码库位置 10](#_Toc11617043)

说明：

1）每次实习完成后提交源代码和此文档。代码部分去掉bin和库所在目录，压缩成一个文件。全部实习结束后，统一通过git工具把代码放到www.Github.com或www.oschina.net上。

2）文档增量更新，后一次实习提交时包含前一次实习内容，同时记得更新目录。全部实习结束时形成一个完整的实习报告。

3）实习文档命名：Java课内实习12017100xxxx-张三-实习x.doc，代码包命名：Java课内实习2017100xxxx-张三-实习x.zip。

4) 统一在deadline前发到邮箱： lsw4000@qq.com

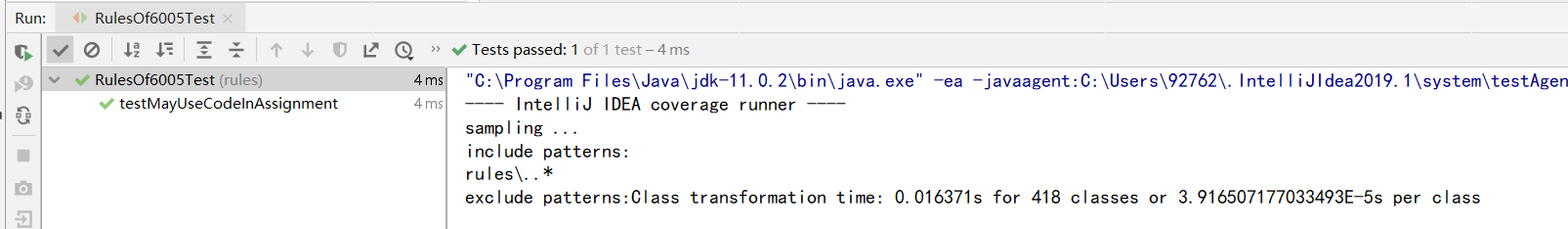
# 实习1

## 实习目的

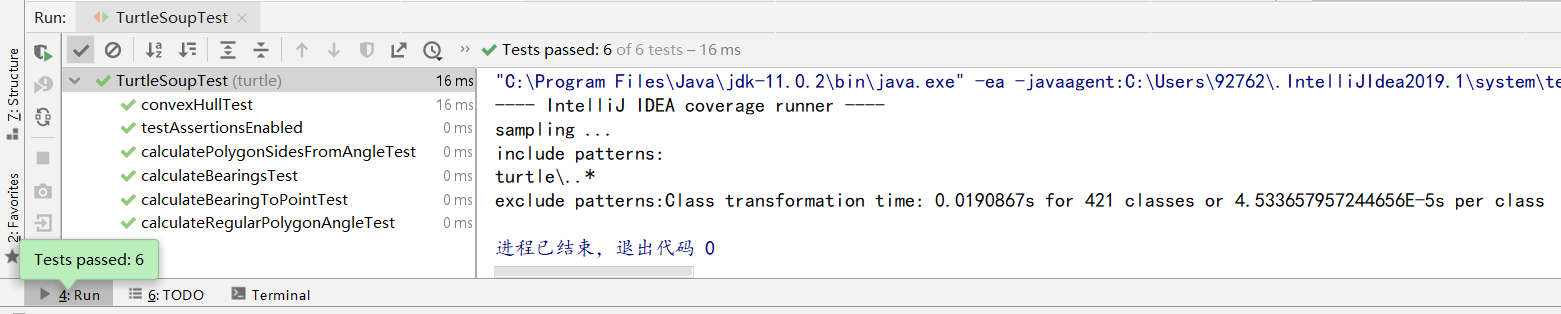
1. 熟悉*Java*开发*IDE（Eclipse/IDEA）*；
2. 练习*Java*基本语法；
3. 使用*turtle*库绘制；
4. 练习使用*Junit*进行单元测试；
5. 一些小算法练习。

## 测试

Problem1：



Problem2-8：



## 问题集

### Problem 1

[问题描述] 热身任务：实现*mayUseCodeInAssignment*函数。

[求解思路] 刚开始看不懂这个题目，只知道是根据多个*Boolean*得出一个Boolean，但不知道规则，通过阅读函数说明以及测试函数，我的推测是：只有当*writtenByYourself*为真、*availableToOthers*为假、*writtenAsCourseWork*为真、*citingYourSource*为真、*implementationRequired*为真时返回真，其余情况返回假。

[遇到的困难等]

。。。

### problem 2

[问题描述] 用*forward*(前进)和*turn*(转弯)实现*drawSquare* (绘制一个正方形)。

[求解思路] 循环四次：每次前进边长，转90°。

[遇到的困难等]

。。。。

### problem 3

[问题描述] 实现*calculateRegularPolygonAngle*函数（计算正多边形内角度数）。

[求解思路] 运用公式：即可。

[遇到的困难等]

要注意一点：类型的转换，如果不给分母强制转换成*double*，是先进行除法之后再转换的，所以会丢弃小数点后的数。

### problem 4

[问题描述] 实现*calculatePolygonSidesFromAngle*函数（通过正多边形内角度数计算边数）

[求解思路] 根据上一题的公式进行简单的计算即可得出公式：。

[遇到的困难等]

公式得出的结果不一定为整数，所以需要四舍五入，此处我使用的是*Integer.parseInt(new java.text.DecimalFormat("0").format(double)))。*原理就是先将double按格式转换成字符串，再通过字符串转换成整数。

### problem 5

[问题描述] 实现*drawRegularPolygon*函数（绘制正多边形）。

[求解思路] 利用到*problem3*实现的函数，*sides*个循环：每个循环前进*sideLength*，转*180-(calculateRegularPolygonAngle(sides))*度。

[遇到的困难等]

。。。。

### problem 6

[问题描述]实现*calculateHeadingToPoint*函数：此函数计算turn从当前点到目标点所需的参数，并将当前方向作为附加参数。例如，如果乌龟在（0,1）面向30度，并且必须到达（0,0），它必须再转150度，所以*calculateHeadingToPoint(30, 0, 1, 0, 0)*会返回150.0。

[求解思路] 数学问题：先用atan2计算两坐标与x，y平行线形成的一个角的度数-tmp；通过画图得出:

。

[遇到的困难等]

这个计算还挺难的，耗尽了我的中学所学啊。

### problem 7

[问题描述] 实现*calculateHeadings*函数：给出n个点，依次计算*0->1、1->2、…、n-1->n*，所需的*turn*参数

[求解思路] 直接依次遍历调用*calculateBearingToPoint*即可。

[遇到的困难等]

。。。。

### problem 8

[问题描述] 实现*convexHull*函数：给定二维平面上的点集，凸包就是将最外层的点连接起来构成的凸多边形，它能包含点集中所有的点。

[求解思路] 我的思路是：

1. 先找出y值最小的那个点，若有y值相同的点，选择x较小的那个，这个点必然包含在凸包中；
2. 初始*currentBearing*=270，用找到的点与剩下的点作*calculateBearingToPoint*运算，找出值最大的那个即为第二个点；
3. 接下来，*currentBearing=currentBearing+上一步最大值-180*，依次找出剩下的点即可。

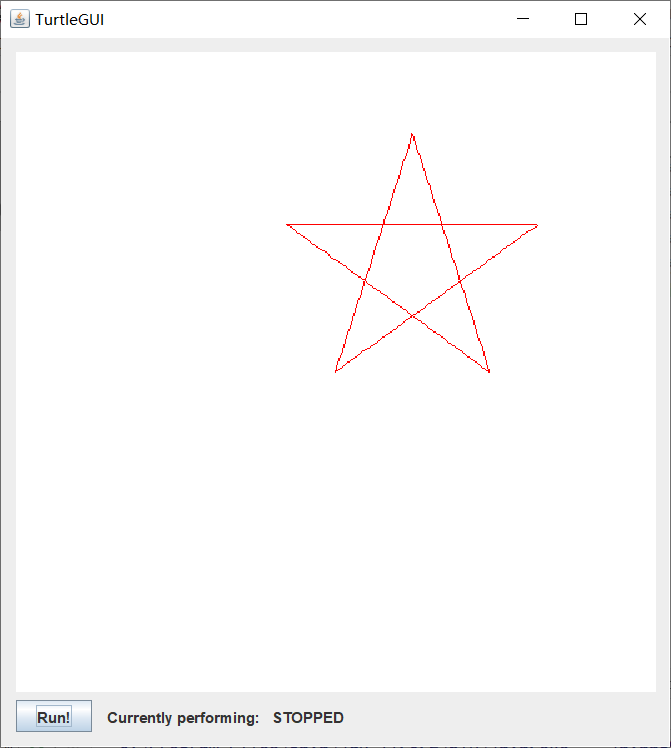
[遇到的困难等]

注意：当给出的点集的个数小于3时，给出的点直接形成一个凸包，所以直接返回这个点集即可。

### problem 9

[问题描述] 个人艺术创作。

[求解思路] 简单的画了一个五角星。每次转的角度是144°



[遇到的困难等]

。。。。

## 思考和建议

直接在上算法课的时候，每周的*leetcode*练习都尽量用*Java*写的，所以对*Java*不算陌生，但是也不太熟悉，其中的*Junit*单元测试是之前从来没有见过的，对这方面有些陌生。在之前学习*Python*时，接触过*python*中的*turtle*库，所以也不陌生。

通过本次的实习，对*Junit*有了基本的了解，并知道如何在项目中添加*Junit*测试，了解了凸包的概念和算法（可能我的不太严谨），对*Java*的编写也更熟练了。

建议：有些题目是真的看不懂（可能是语言的原因），像第一题，刚开始看到就是懵的。

# 实习2

## 实习目的

1. 熟悉Java的一些集合框架（List，Set，Map）
2. 学习一个计算文本相似度的方法：余弦相似度
3. 练习面向对象的编程

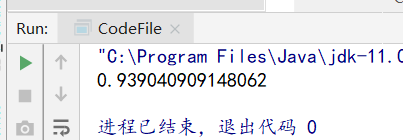
## 问题集

### Problem 1

[问题描述]利用余弦相似度来计算两个代码文件的相似度。

[求解思路]先后按照*Scanner.next()*读取两个文件，读取到的字符串分别加入（修改）各自文件的*Map*（用于计数）以及一个*Set*，再用两个*set.size()*大小的数组，纪录两个文件的词频向量，最后用公式计算即可。

[遇到的困难等]给出的两个文件的编码一个为*utf-8*，一个是*GBK*，读取*GBK*编码的文件遇到中文会出现问题，更改编码后成功：



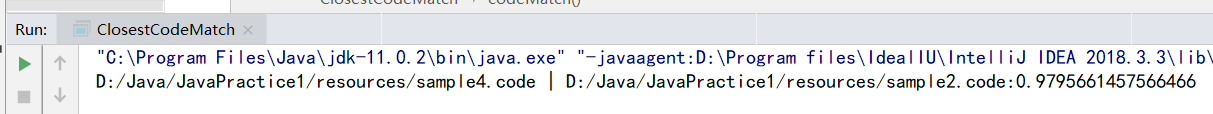
### problem 2

[问题描述]给出n个文件，找出相似度最高的两个文件，比较*n(n+1)/2*次。

[求解思路] 将n个文件名放入一个集合中，每次pop一个文件，与Set中剩下的文件分别比较，将结果纪录在一个*Map<(url1,url2),Similarity>*中,最后按照Similarity从大到小排序（将Map转为List），输出*list.get(0)*。

[遇到的困难等]刚开始写的有问题，在计算相似度函数中，直接用wordSet计算多个文件时会纪录多个文件的词，后改为用一个临时拷贝的Set。

测试文件为给出的文件夹中的几个*TurtleSoup.java*。



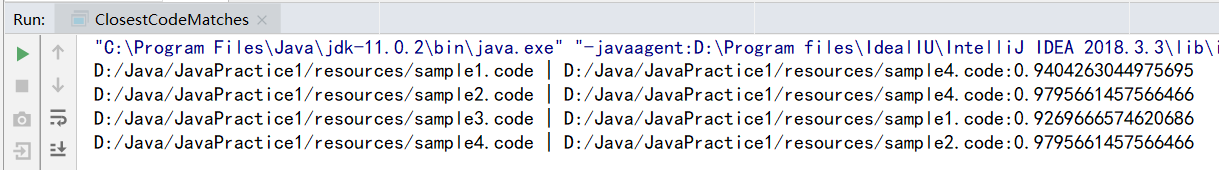
### problem 3

[问题描述]与问题2类似，给出一组文件，为每个文件求出相似度最高的那个文件。

[求解思路] 与问题2的求解也类似，比较得出Similarity，不同的是为每个文件在Map<url1,(url2,Similarity)>纪录相似度最高的文件及相似度。

[遇到的困难等]

无大问题，测试文件同Problem2：



### problem 4

[问题描述]给出一个文件夹，用于评价各相关目录下，指定文件（用正则表达式过滤）的相似性，输出相似度高于*topRate*的文件夹，可选择移除注释或不移除。

[求解思路] 先用一个*findFile()*函数找出给出目录的各个子目录下匹配的文件，每个子目录中的可匹配的文件存于一个List中，并纪录每组文件对应的子文件夹。在*CodeFile*类中，增加两个函数，用于接收一组文件，即由原来的两个文件比较改为两组文件的比较（合并），并增加可选是否移除（忽略）注释。然后类似*Problem3*，纪录每两个子文件夹的相似度，最后排序，输出Similarity高于topRate的子文件夹。

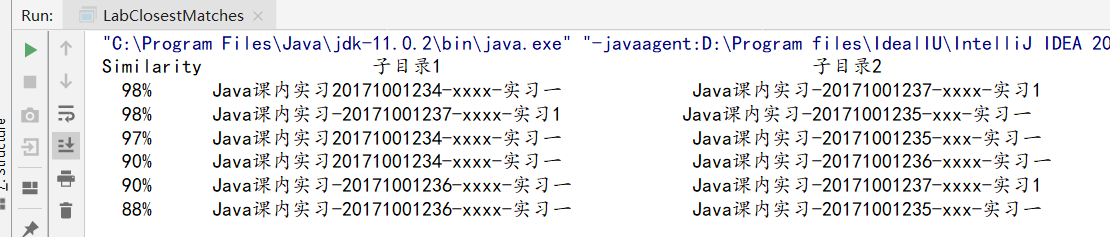
移除注释：一个boolean变量flag（标记是否是非注释文件），初始flag为true，

当flag为true，每次取得的字符串判断前两个字符是否为/\*，若是则接下来为/\*\*/段注释部分，记flag为false，若前两个字符为//，则为行注释，直接忽略，若都不是则为非注释部分，进行相关操作；当flag为false，说明当前还是/\*\*/段注释，若后两个字符为\*/，结束当前注释，flag=true，否则忽略。

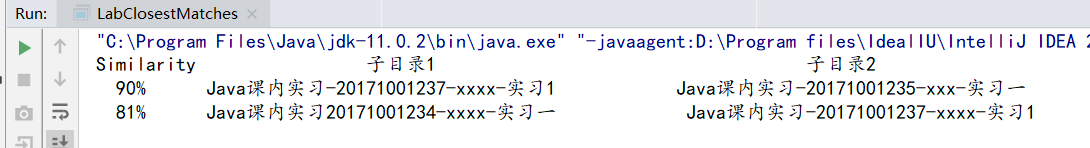
[遇到的困难等]

刚开始想的是用正则表达式分割整个文件，但是会爆栈。。（我用的是**"(/\\\*{1,2}(.|\\s)\*\\\*/)|(//( |\t|.)+\n)|(\\s+)"**可能是我的正则表达式有问题？）所以改用了flag标记。。。

测试：*topRate=80，fileNameMatches=".\*TurtleSoup.java"，removeComments=false:*



*removeComments=true:*



## 思考和建议

收获：余弦相似度的计算：分词->得到磁频向量->公式计算得出cosθ->得到结果；练习了Java的集合框架的使用。

# 实习3

## 实习目的

1. 理解并应用多态性、封装和约束等概念，还包括合理的使用Java接口。
2. 能够基于非正式的规范来解释、设计和实现软件，并展示对基本软件设计原则的理解。
3. 编写单元测试，并使用JUnit测试。
4. 理解代码覆盖率度量的好处和限制，并解释覆盖率度量的结果。
5. 良好的Java编码规范，测试实践及风格。

## 问题集

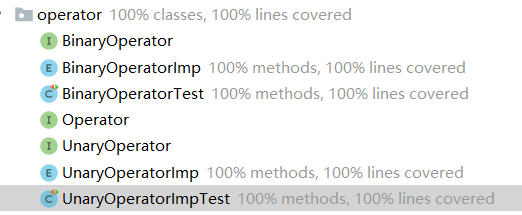
### Problem 1

[问题描述] 使用多态性实现计算器操作

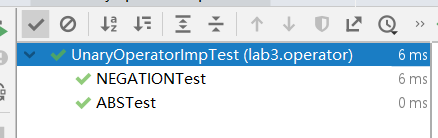
[求解思路] 即实现BinaryOperatorImp和UnaryOperatorImp中几个接口。很简单不赘述。然后在guicalc.Main中实现计算器。及将运算符当作参数传递给src.lab3.guicalc.GuiCalculator的构造函数并启动计算器。

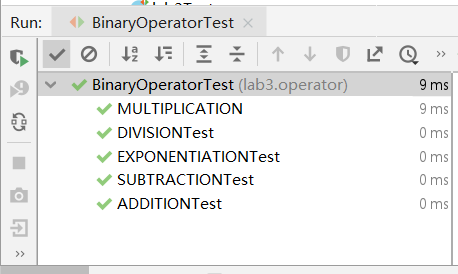
[遇到的困难等]测试：

覆盖：



结果：





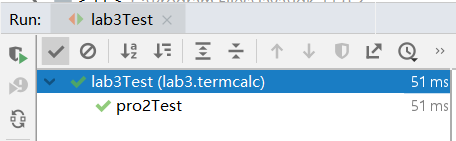
### problem 2

[问题描述] 实现计算器表达式

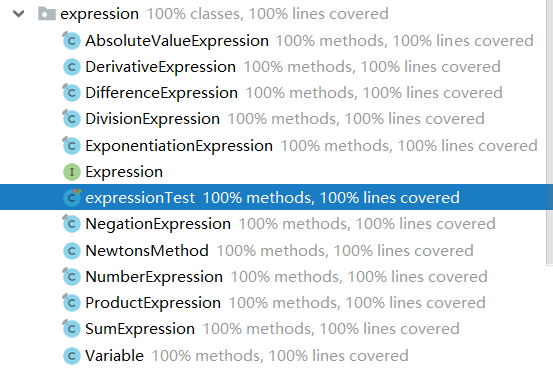
[求解思路] 先实现八个运算符表达式类，即实现其中的toString()与eval()方法，按照计算逻辑即可。然后实现termcalc.ExpressionMakerImp，调用之前实现的运算符表达式类即可，算数表达式的读取分割已经给出。

[遇到的困难等]测试：

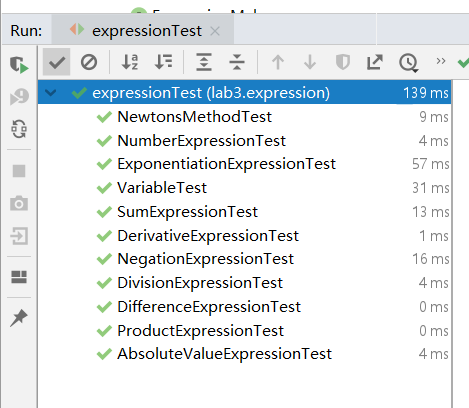
算数表达式运算测试：



覆盖：



结果：



### problem 3

[问题描述] 函数式编程:导数与牛顿法；即实现表达式中可以带未知数与一个函数求导和求零点的方法。

[求解思路] 实现带命名变量的表达式很简单与实现运算符表达式类似。

求导数表达式题中已经给出了一个算法，实现即可。

牛顿法求零点：同样按照描述实现，描述有点模糊，可能实现的不太对。。

[遇到的困难等]测试：

与problem2的测试写在了一起。

## 思考和建议

1. 了解了更多的Java编程的结构；
2. 练习了接口的使用；
3. 复习Junit测试的编写。

# 代码库位置

[*https://github.com/Xhofe/Java*](https://github.com/Xhofe/Java)