

**2019年春季学期  
计算机学院《软件构造》课程**

**Lab 1实验报告**

|  |  |
| --- | --- |
| 姓名 | 强文杰 |
| 学号 | 1171000410 |
| 班号 | 1703005 |
| 电子邮件 | [672334335@qq.com](mailto:672334335@qq.com) |
| 手机号码 | 18800421389 |

**目 录**

[1 实验目标概述 1](#_Toc1392798)

[2 实验环境配置 1](#_Toc1392799)

[3 实验过程 1](#_Toc1392800)

[3.1 Magic Squares 1](#_Toc1392801)

[3.1.1 isLegalMagicSquare() 1](#_Toc1392802)

[3.1.2 generateMagicSquare() 1](#_Toc1392803)

[3.2 Turtle Graphics 1](#_Toc1392804)

[3.2.1 Problem 1: Clone and import 2](#_Toc1392805)

[3.2.2 Problem 3: Turtle graphics and drawSquare 2](#_Toc1392806)

[3.2.3 Problem 5: Drawing polygons 2](#_Toc1392807)

[3.2.4 Problem 6: Calculating Bearings 2](#_Toc1392808)

[3.2.5 Problem 7: Convex Hulls 2](#_Toc1392809)

[3.2.6 Problem 8: Personal art 2](#_Toc1392810)

[3.2.7 Submitting 2](#_Toc1392811)

[3.3 Social Network 2](#_Toc1392812)

[3.3.1 设计/实现FriendshipGraph类 2](#_Toc1392813)

[3.3.2 设计/实现Person类 2](#_Toc1392814)

[3.3.3 设计/实现客户端代码main() 2](#_Toc1392815)

[3.3.4 设计/实现测试用例 3](#_Toc1392816)

[3.4 Tweet Tweet 3](#_Toc1392817)

[3.4.1 Problem 1: Extracting data from tweets 3](#_Toc1392818)

[3.4.2 Problem 2: Filtering lists of tweets 3](#_Toc1392819)

[3.4.3 Problem 3: Inferring a social network 3](#_Toc1392820)

[3.4.4 Problem 4: Get smarter 3](#_Toc1392821)

[4 实验进度记录 3](#_Toc1392822)

[5 实验过程中遇到的困难与解决途径 3](#_Toc1392823)

[6 实验过程中收获的经验、教训、感想 4](#_Toc1392824)

[6.1 实验过程中收获的经验和教训 4](#_Toc1392825)

[6.2 针对以下方面的感受 4](#_Toc1392826)

# 实验目标概述

实验通过求解四个问题，训练基本的Java编程技能，能够利用Java OO开发基本的功能模块，阅读并补全代码，理解并测试程序。另一方面，利用Git作为代码配置管理的工具，学会Git的基本使用方法。

基本的Java OO编程

基于Eclipse IDE进行Java编程

基于Junit测试

基于Git的代码配置管理

# 实验环境配置

简要陈述你配置本次实验所需开发、测试、运行环境的过程，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。

1. 根据助教在群里发的链接，完成Eclipse和JDK11的安装，然后去网上查询Java运行环境的配置，配置计算机的环境变量。

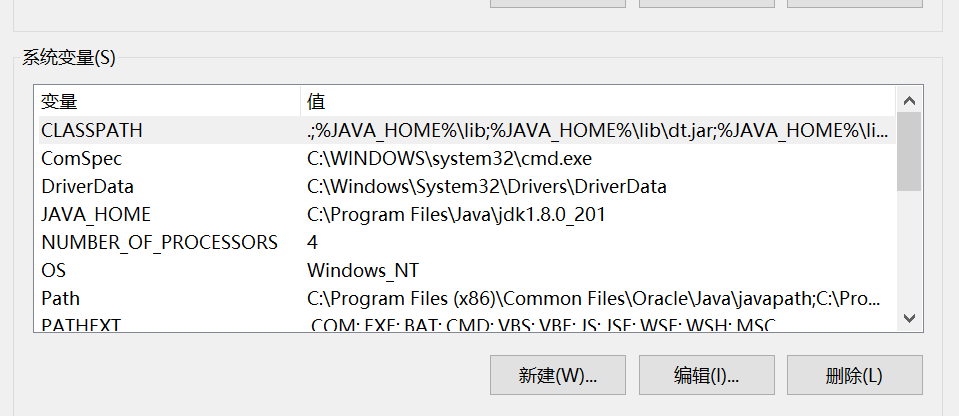


图2-1 环境变量配置

2.注册GitHub账号，并且授权GitHub classroom，获取lab1的GitHub classroom的URL地址。

3. Git代码配置管理。首先安装最新版Git，获取个人GitHub的URL地址，通过在Git bash输入指令建立管理本地仓库，将文件push到远程仓库。

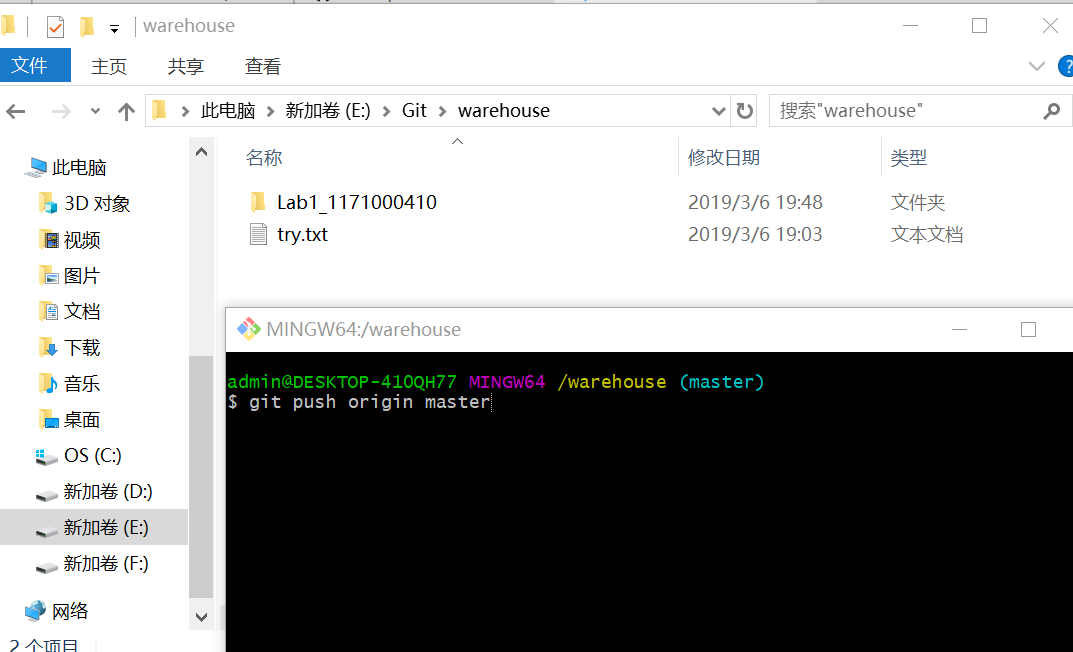


图2-2 Git将文件push到远程仓库

**遇到的困难：**

1.在配置环境时，最早的参考教程是针对Mac OS，并且本地路径的设置出错，导致走了很多弯路。

2. 在安装Git之后，未第一时间在Git的安装目录注册user.name和user.email，导致后面输入命令时bash显示commend not found。

URL地址：https://github.com/ComputerScienceHIT/Lab1-1171000410

# 实验过程

请仔细对照实验手册，针对四个问题中的每一项任务，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（但无需把你的源代码全部粘贴过来！）。

为了条理清晰，可根据需要在各节增加三级标题。

## Magic Squares

这个实验就是设计一个Java小程序，熟悉Java的编程。所谓magic square就是各行、各列和各对角线的和都相等的矩阵，我们的任务就是设计程序判断txt中的矩阵是否为magic square，并且能够处理输入文件的各种特殊情况，例如：文件中的数据不符合Magic Square的定义，矩阵中某些数字并非正整数等，遇到异常情况，终止程序执行并在控制台输入错误提示信息。

### isLegalMagicSquare()

按步骤给出你的设计和实现思路/过程/结果。

**1.设计和实现思路：**该函数的功能是检验出矩阵是不是Magic Square，并且还得满足以下几个要求：分隔符是否符合要求；是不是矩阵；数字是否符合要求；是否满足Magic Square的数学关系。

**2.过程：**

**首先**，通过I/O流将 .txt文件按行读入，并且将该行存入字符串中，方便后面对其进行处理。

**其次，**使用Java API的String.split函数，对第一步读入的字符串进行处理。Magic Square中各个元素的分割是”\t”时 ，可以首先确定矩阵的各个参数，如列数，并且由此生成二维数组来存储矩阵。

**然后，**是对Magic Square输入文件各种特殊情况的判断（代码如下）：



图3.1.1-1 对输入特殊情况的判断

主要从以下方面判断输入的特殊情况：是否满足矩阵要求，通过使用String.split判断是否存在负数和浮点数等，行列数是否相等。

**最后，**判断Magic Square是否满足行列及对角线相等的数学关系。

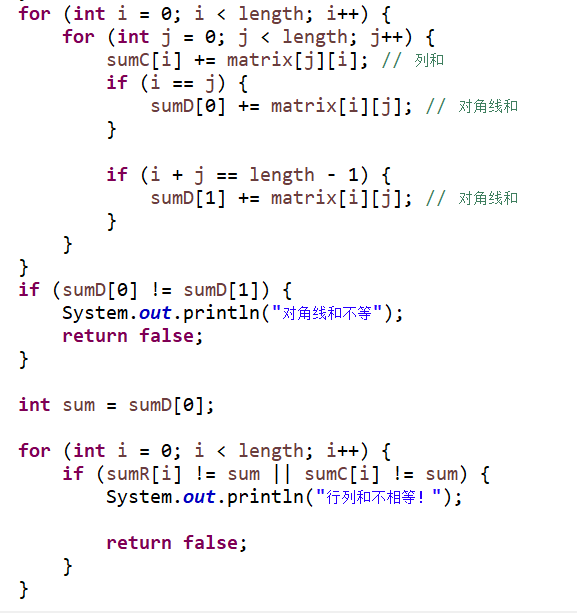


图3.1.1-2 判断矩阵是否满足数学关系

使用两个数组将各行各列的和保存，计算并比较两个对角线的和，再将对角线和sum和两个数组中的各个元素进行比较，如果其中出现不相等，直接返回false。

若以上所有情况都判断通过，函数将返回true。

**3.结果**

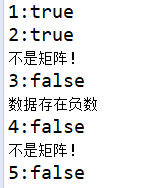


图3.1.1-2 运行结果

### generateMagicSquare()

按步骤给出你的设计和实现思路/过程/结果。

**1.设计和实现思路**

其实这个函数是奇数阶幻方最经典的--罗伯法。填写的方法是：

把最小的数放在第一行正中；按以下规律排列剩下的(n×n－1)个数：  
（1）每一个数放在前一个数的右上一格；

（2）如果这个数所要放的格已经超出了顶行那么就把它放在底行，仍然要放在右一列；

（3）如果这个数所要放的格已经超出了最右列那么就把它放在最左列，仍然要放在上一行；

（4）如果这个数所要放的格已经超出了顶行且超出了最右列，那么就把它放在底行且最左列；

（5）如果这个数所要放的格已经有数填入，那么就把它放在前一个数的下一行同一列的格内。

**2.过程（流程图展示）**

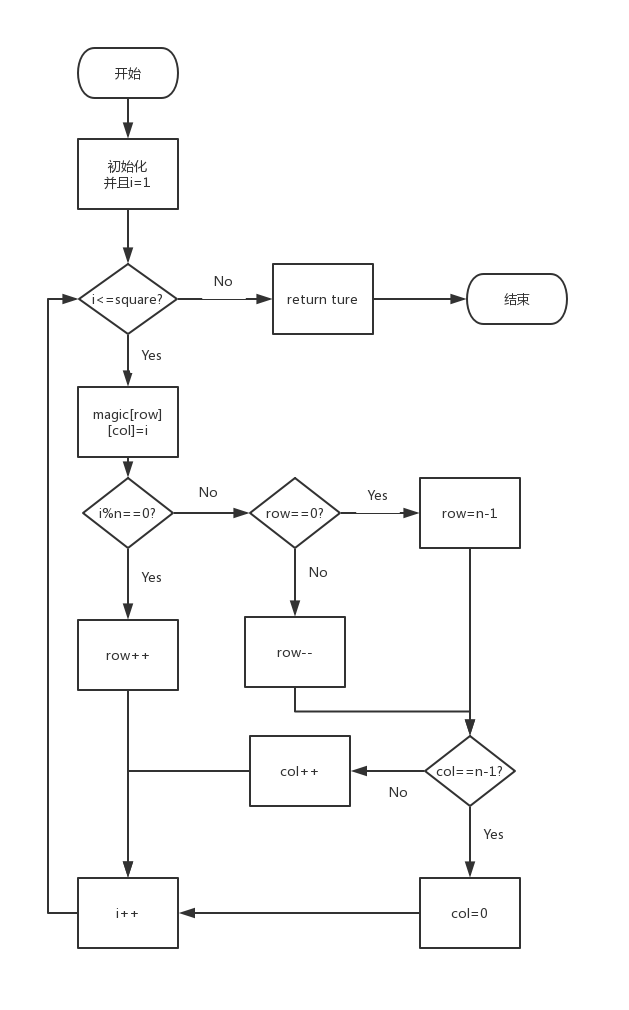
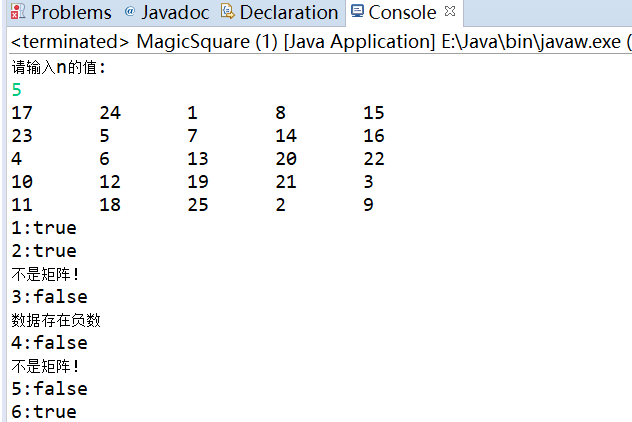


图3.1.2-1 generateMagicSquare流程图

**3.结果：**

（首先解释n不能是偶数的原因：若n是偶数，观察函数可知，row每轮循环都会比上个循环的值大，最后会发生数组越界）



## Turtle Graphics

这个实验的Turtle图形允许程序员向屏幕上的“乌龟”发出一系列的指令，这些“乌龟”会移动，随着它画出一条直线，我们可以构造出很多不同的图形。实验主要的部分是：1.使用forword()和turn()函数构造图形 2.解决一些数学方面的问题，如对角度的计算等。

### Problem 1: Clone and import

**获取任务的代码：**通过实验指导上的链接访问老师的仓库，将相关的代码下载到本地即可。

**在本地创建git仓库：**先完成git的下载和安装，然后在git的安装目录设置user.name和user.email 。在一个自己设置的文件夹中输入git init 即可把这个目录编程Git可以管理的本地仓库。

**使用git管理本地开发：**获取个人GitHub的URL地址，通过在Git bash输入git add把文件添加到仓库，再输入git commit把文件提交到仓库，最后将文件git push origin master推送到远程仓库。

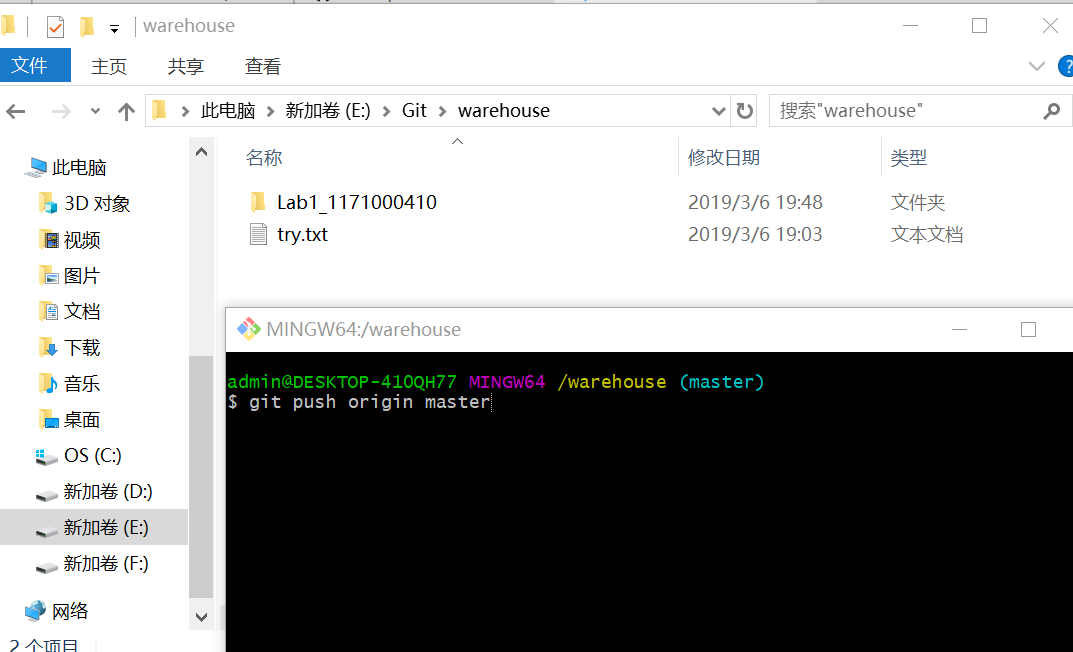


图3.2.1-1 Git将文件push到远程仓库

### Problem 3: Turtle graphics and drawSquare

这个任务是通过“龟”画一个正方形。

其中Turtle.java中定义了画图所需要的接口。

**public** **void** forward(**int** units); 向当前方向画一条长度为units的直线

**public** **void** turn(**double** degrees); 面朝方向顺时针旋转90度

**public** **void** color(PenColor color); 更改“乌龟”画笔的颜色

初始画笔的方向向上，画一个正方形只需要先forward再turn，循环四次。

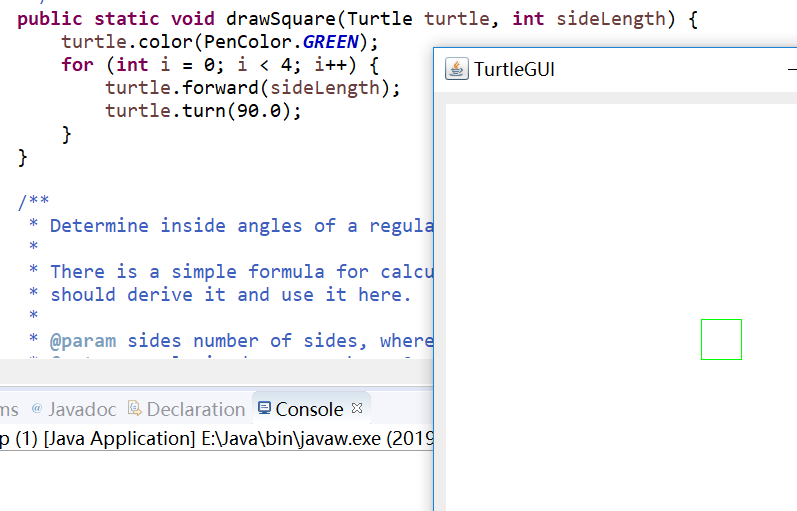


图3.2.2-1 画正方形

### Problem 5: Drawing polygon

绘制多边形这个problem包含了几个数学计算方面的任务。

1. **public** **static** **double** calculateRegularPolygonAngle(**int** sides)

由多边形的边数计算它的内角。

有两种方式： 根据 (sides - 2) \*180 / sides的数学公式来计算；或者根据多边形外角和恒为360度来计算。

这两种方法经过测试都是可行的。

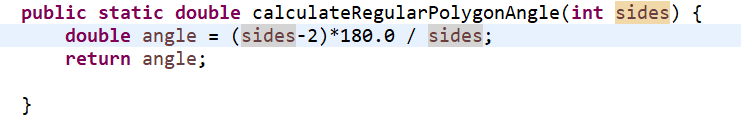


图3.2.3-1 计算多边形内角

2. **public** **static** **int** calculatePolygonSidesFromAngle(**double** angle)

根据多边形的内角，计算多边形的边数。

其实这个函数相当于上个函数的逆运算。由多边形的内角即可求出其外角，再由外角和是360度即可求出边数。

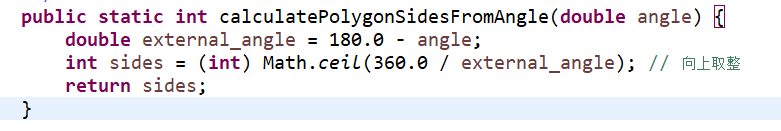


图3.2.3-2 计算多边形的边数

3. **public** **static** **void** drawRegularPolygon(Turtle turtle, **int** sides, **int** sideLength)

画一个多边形和画一个正方形很类似，只是每次旋转的角度不同，并且旋转的次数也不同。通过calculateRegularPolygonAngle函数计算出内角，从而算出外角，旋转角度即外角大小，旋转次数即边数。

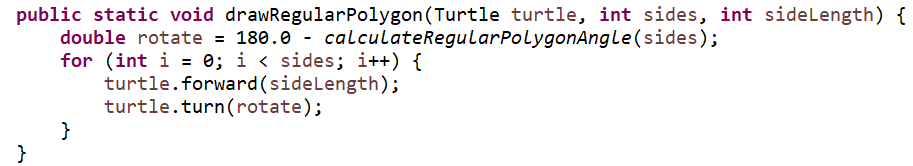


图3.2.3-3 绘制多边形

### Problem 6: Calculating Bearings

这个problem是计算轴承（向量的方向角），包括了两个函数。第一个函数是计算两个向量的方向角，第二个函数则是建立在第一个函数的基础上，计算一系列向量的方向角。

1. **public** **static** **double** calculateBearingToPoint(**double** currentBearing, **int** currentX, **int** currentY, **int** targetX,**int** targetY)

首先currentBearing是当前方向和正上方的夹角，清楚这个后面的计算才不会出错。根据参数提供的current点和target点的坐标，可以计算出二者横坐标和纵坐标之差，并在current点处建立坐标系。再通过调用Math.*atan2*(y, x)反正切函数，可以计算出两点连线向量的方位角。

然后将返回单位为弧度的方位角转化为角度，并且被90减去，算出两点连线的向量与正上方向的夹角degree。

最后讨论“乌龟”朝向与正上方向夹角和degree的大小，即可返回需要旋转的角度。

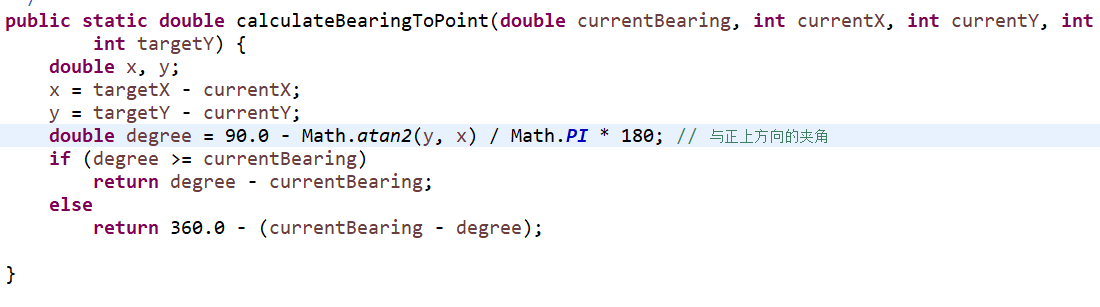


图3.2.4-1 计算轴承

2. **public** **static** List<Double> calculateBearings(List<Integer> xCoords, List<Integer> yCoords)

这个函数是建立在第一个计算轴承函数的基础上的。函数的参数是一系列点的横坐标和纵坐标，需要返回的是旋转到最后一个点的所有角度的集合。

先new一个所有角度的容器，并将所有点的个数设为length。循环，对（length-1）次旋转的度数进行求解，求解方式是借用第一个函数，代入两点的坐标和current点的方向，即可返回需要旋转的角度。最后将所有的角度都存入容器中并返回即可。

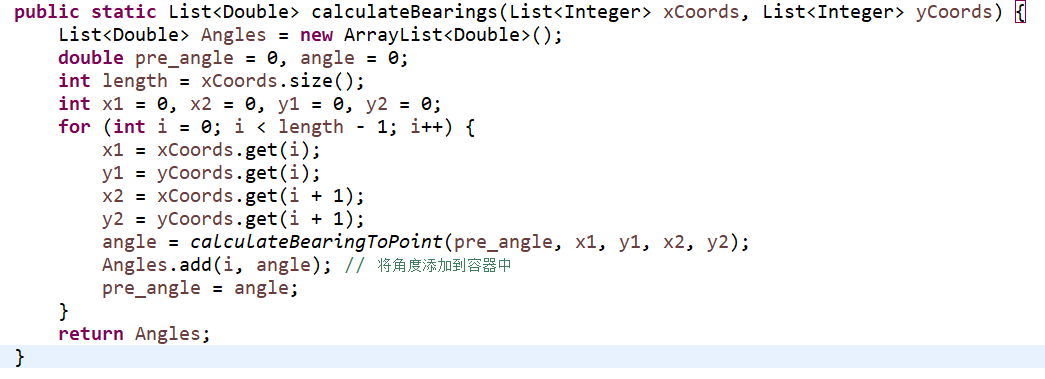


图3.2.4-2 计算多个轴承

### Problem 7: Convex Hulls

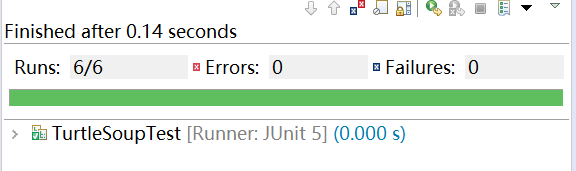
点集的凸包是一个最小的凸多边形P，满足点集中的每个点都在P的边界上或者在P的内部。因为需要对每个点进行遍历并知道它们的下标，我选择将Set中的点全部用Arraylist来存储，中间使用一个迭代器Iterator来过渡，集中遍历集合中的元素。

得到Arraylist后，先找一个固定点p0作为x坐标最小的点，如果有多个这样的点，则选取最下面的点。然后找一个基准点p1，并且遍历点集中的其他点，取一个为p2，若p2p0与p1p0的叉乘积大于零，则将p2设为p1，遍历完点集后把p1添加到凸包中。一直循环，直到p0再次称为待添加的点，循环终止。



图3.2.5-1 凸包循环主体

**以上所有函数的测试结果：**

****

### Problem 8: Personal art

我自己设计的图形是一个彩色的五角星。外层循环是改变方向，内层是调用for each循环，使得每次前进的时候画笔的颜色都会不断改变。

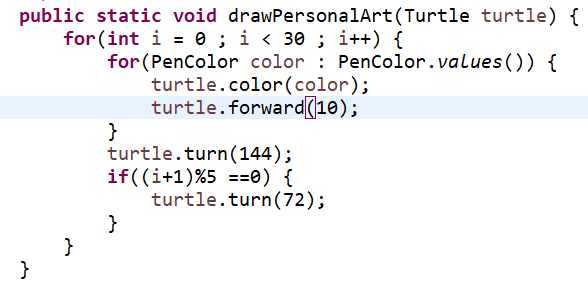


图3.2.6-1 Personal art函数主体

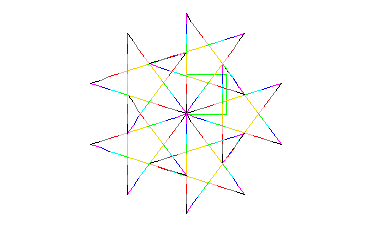


图3.2.6-2 Personal art

### Submitting

先用当前版本把老版本覆盖掉，其余操作和之前一样。git add Lab1\_1171000410

git commit -m “第二次提交” git push origin master 。

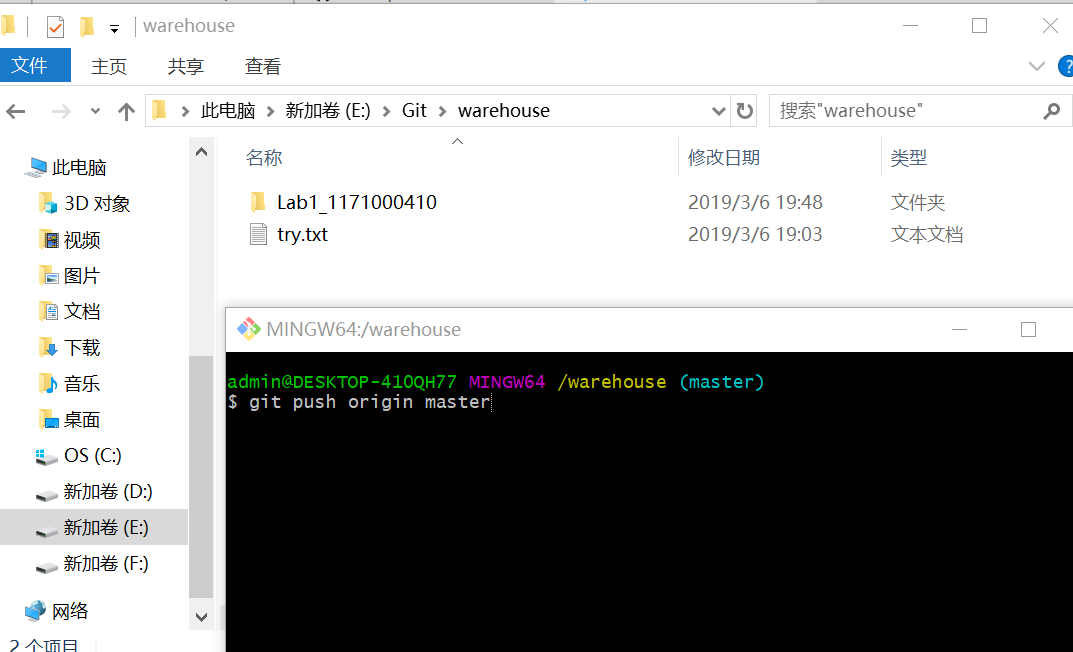


图3.2.7-1 Git将文件push到远程仓库

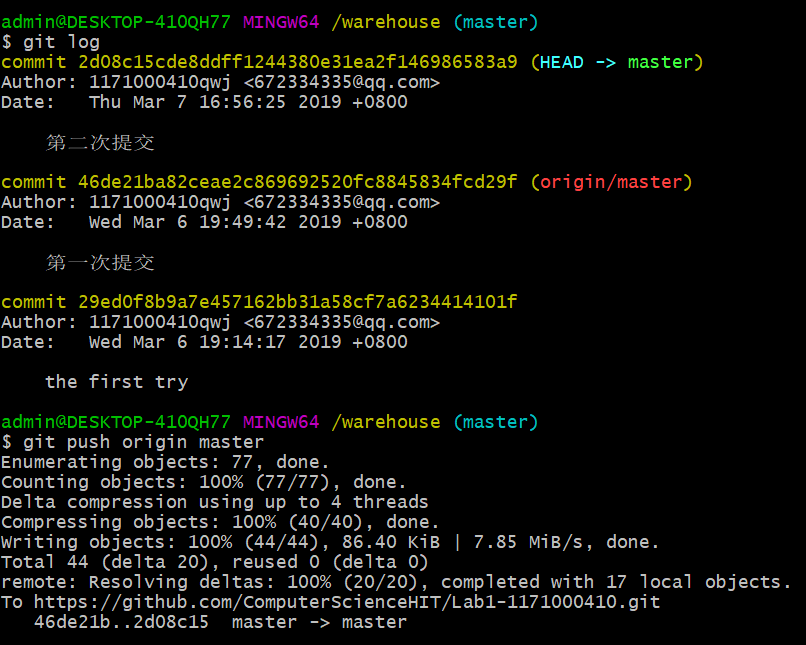


图3.2.7-2 Git提交当前版本到远程仓库

## Social Network

这个任务是实现一个社交网络，基本的实现方式是图。社交网络建模为无向图，但是能够支持扩展到有向图。主要的函数是getDistance，可以返回两人的最短距离。测试时图表应能合理拓展，测试数据可以是很多的点和边。另外，这个任务需要对一些错误情况进行处理，比如人重复输入等。

### 设计/实现FriendshipGraph类

给出你的设计和实现思路/过程/结果。

**设计和实现思路：**不仅得补充FriendshipGraph类的相关函数。而且还需要设计合理的数据结构。根据分析，FriendshipGraph描述的这个关系图应该是稀疏图，因此选择用邻接表，每个成员的朋友放在一个List中，并且整个图的映射关系通过Map来存取。另外，根据实验指导的提示，类中需要增加一些非法情况的判断，例如人已经存在，或者边已经存在等。

**过程：public** Map<Person,ArrayList<Person>> map = **new** HashMap<Person, ArrayList<Person>>();

//每个成员的朋友放在List中，Map存取图的映射关系

1. **public** **void** addVertex(Person p)

这个函数是为把参数添加到图中，作为图的一个顶点，并且为该点创建一个List来存他的朋友。这个函数需要注意的就是点已经存在的约束情况，增加判断条件。

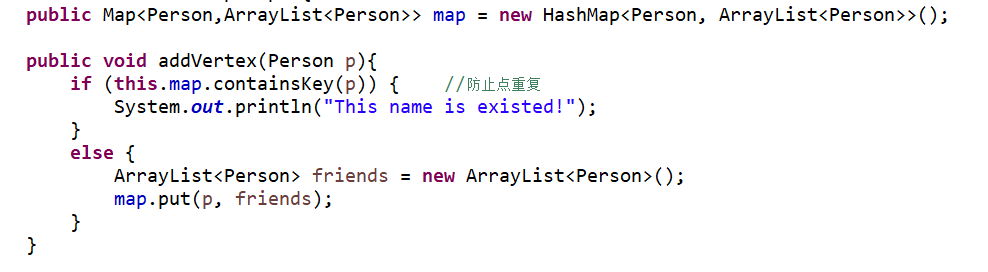


图3.3.1-1 addvertex函数

2. **public** **void** addEdge(Person p1, Person p2)

构建图的要素，添加了点后就该添加边了。通过Map里的一些方法，遍历对应点P1的朋友的集合，如果List容器中有P2，则设置标志sign为false，停止遍历，否则就把P2添加进朋友的集合。

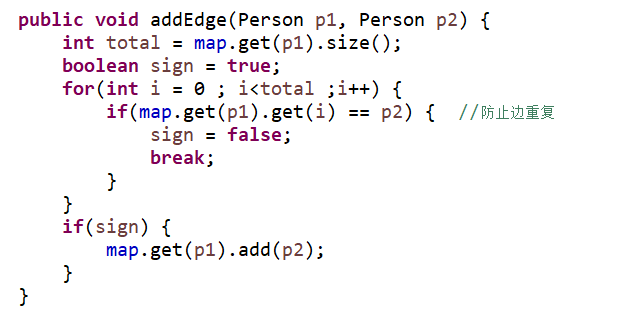


图3.3.1-2 addEdge函数

3. **public** **int** getDistance(Person p1, Person p2)

最主要的就是找两个人距离的函数，因为根据题设要求最短距离，因此采用广度遍历的方式，此处需要用到Queue的数据结构，并且设置了一个List来存放已经访问过的person。

广度遍历的主要思路是：将P1进队，设置访问过，然后出队，依次访问其相邻点，并且把它们进队，然后P1访问完再取队列第一个元素访问，直到某一相邻的点为P2则返回最短距离。如果所有点都访问完仍无最短距离，说明P1和P2没有关系，此时返回-1 。



图3.3.1-3 广度遍历主要过程

**测试结果：**

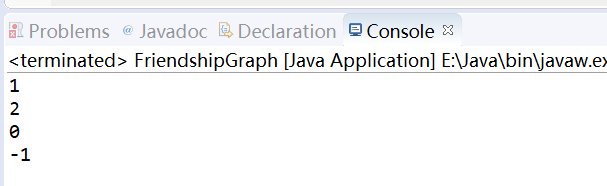


图3.3.1-4 正常输出的测试结果

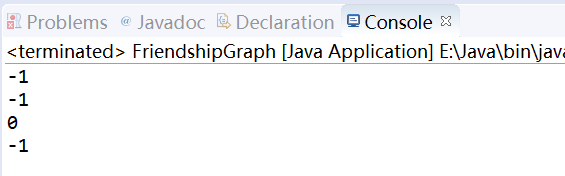


图3.3.1-5 注释掉rachel->ross后的输出结果

### 设计/实现Person类

Person类较为简单，主要是根据FriendshipGraph类的需求编写的。它用于描述每个成员的性质，主要是实例化姓名的构造方法，getName方法，判断姓名是否重复的nameSameWith方法。

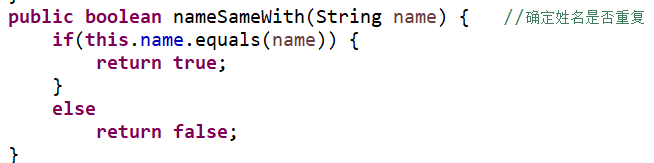


图3.3.2-1 nameSameWith方法

### 设计/实现客户端代码main()

main()函数主体部分就是实验指导上的要求：先new一个FriendshipGraph类的对象，然后添加顶点，添加边。

不同之处在于我增加了一个测试，如果Person的对象姓名字符串相同，违反了“Each person has a unique name”，将会提示出错并结束程序运行。

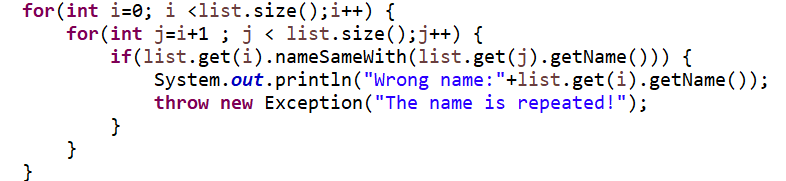


图3.3.3-1约束条件主体部分

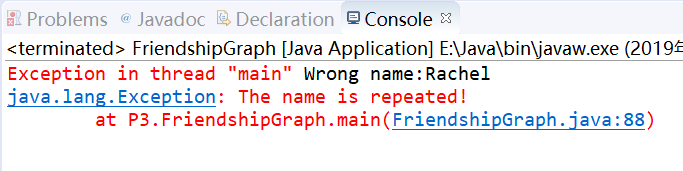


图3.3.3-2 违反“Each person has a unique name”

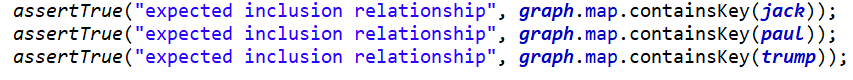
### 设计/实现测试用例

**设计和实现思路：**在测试包Test中创建了一个 FriendshipGrapgTest.java文件后，通过注解的方式@Test表示这是一个测试方法。

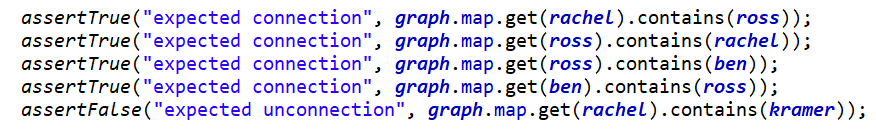
主要使用的方法：assertEquals、assertTrue和assertFalse方法。都是通过判断预期值和实际值是否相等来返回true或false。

**过程：**

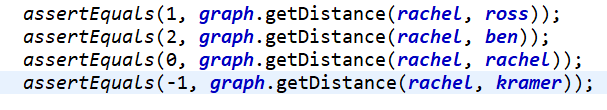
对于addVertex()方法，主要测试了添加的点是否存在。



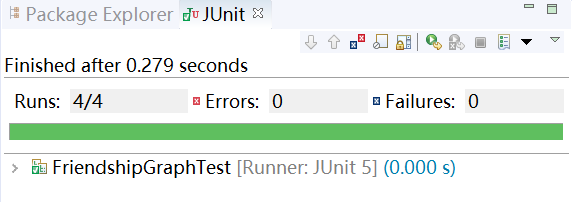
对于addEdge()方法，测试了边是否存在和是否有超出unconnected边。



对于getDistance方法，测试了距离实际值是否和预期值相等。



**结果：**



## Tweet Tweet

这个任务比较有趣，问题集的主题是构造一个方法工具箱，可以从twitter下载的一组推文中提取信息。不仅如此，我们还需要实现很多twitter上的功能，比如过滤推文列表，推测受欢迎程度等。在处理这个任务之前，有必要先熟悉一下整个package中类中的方法，这样有助于思考出编程策略。

### Problem 1: Extracting data from tweets

Extract类中包含从推文列表中提取信息的方法，包含了两个方法：一是获取推文间隔的最大时间段；二是获取推文中提到的用户名。

1. **public** **static** Timespan getTimespan(List<Tweet> tweets)

输入的参数是一系列的推文，我们需要获取推文所涵盖的时间段，即求time interval最大区间。

我们获取了推文的数量后，可以先将表示时间点的min和max分别设为第一篇推文的时间，获取时间是通过Tweet类中的getTimestamp()方法。然后遍历所有的推文，每当有推文的时间在min之前或是max之后，就更新min和max的值，保证其中涵盖了最大的时间间隔。最后，max和min的差值即是我们需要返回的timespan 。

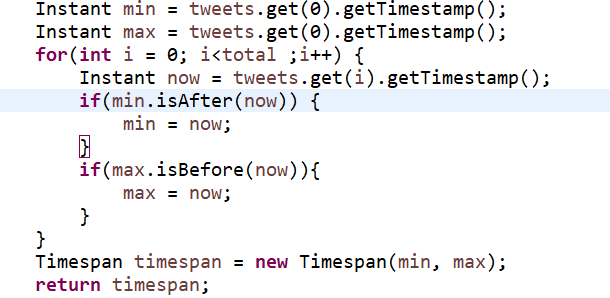


图3.4.1-1 getTimespan函数主要部分

2. **public** **static** Set<String> getMentionedUsers(List<Tweet> tweets)

该方法的参数是一系列不同ID推文的列表，需要返回推文中@的用户名集。

这个函数的主要思路是遍历所有的tweet，并且用字符串存储每篇tweet，扫描这个整个字符串，找到”@”后面的用户即可。

这个函数难处在于构造正则表达式，Pattern和Matcher的组合对整篇的文章进行匹配，匹配到即加入到Set/

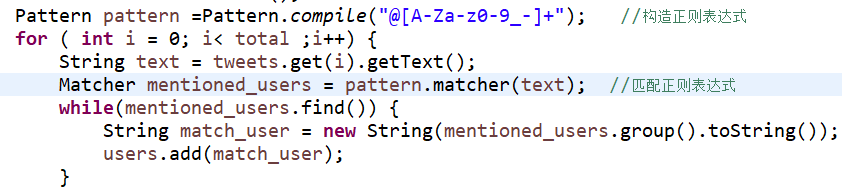
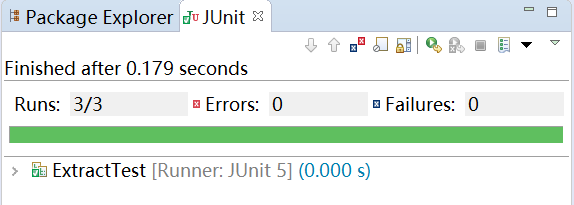


图3.4.1-2 匹配正则表达式的主要过程

**结果：**

****

### Problem 2: Filtering lists of tweets

这个任务是通过完成三个函数实现过滤推文列表，根据时间、作者、推文内容等。

主要采用的方法是遍历。

1. **public** **static** List<Tweet> writtenBy(List<Tweet> tweets, String username)

获取推文总数，遍历所有推文的作者，如果推文的作者与username相同，则加入容器中，最后返回List的对象即可。



图3.4.2-1 遍历tweet的作者

2. **public** **static** List<Tweet> inTimespan(List<Tweet> tweets, Timespan timespan)

这个函数也是遍历通过解决。先通过Timespan类中的方法获得参数timespan的起始时间和结束时间，分别用start和end保存。遍历所有tweet，获取每篇tweet的时间，时间比start晚并且比end早的tweet即加入容器中。

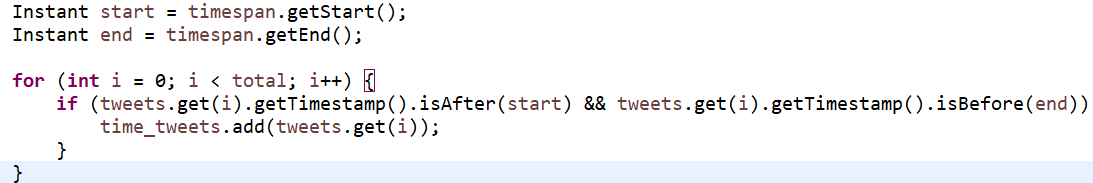


图3.4.2-2 找timespan中的tweet

3. **public** **static** List<Tweet> containing(List<Tweet> tweets, List<String> words)

这个函数是查找包含特定词的推文。主要采取方式仍然是遍历。

函数包含有个两重循环：第一个是遍历所有的tweet，对每篇tweet使用string保存，然后构造容器中word的正则表达式，第二个循环是遍历容器，使用Pattern和Matcher结合，从string中找到匹配的特定字。

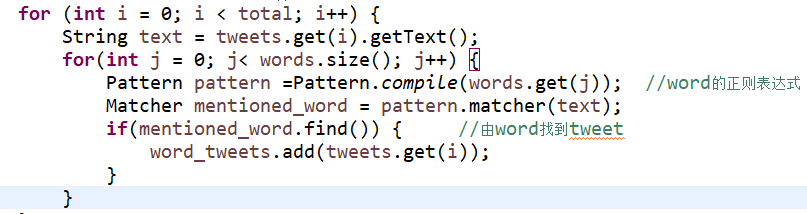


图3.4.2-3 根据特定词匹配tweet

**结果：**

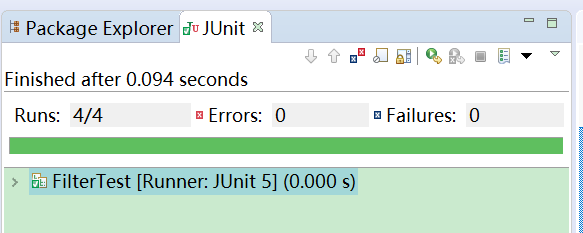
****

图3.4.2-4 Filter测试结果

### Problem 3: Inferring a social network

SocialNetwork类才是真正的社交网络的体现。我们需要根据社交网络Map中存在的“@”关系来分析和总结出信息。因为推文中的证据有限，因此选择“@”的关系来推测两人是否认识最为有效。

1. **public** **static** Map<String, Set<String>> guessFollowsGraph(List<Tweet> tweets)

这个函数的参数还是一系列的tweets，需要我们返回一张关注图。

我们的思路是这样的：先通过getAuthor()方法找到所有tweets的作者并存在HashSet中，然后通过Filter.writtenBy()分别找到这些作者写的所有文章并存在List中，通过遍历所有文章使用Extract.getMentionedUsers()即可找到每个作者文章中“@”过的人，最后存入map中返回。

有一点需要注意的就是：题干中说到Twitter用户名不区分大小写，因此我将authors的名字全部转化为了小写。（不过最后toLowerCase并没有影响测试结果）

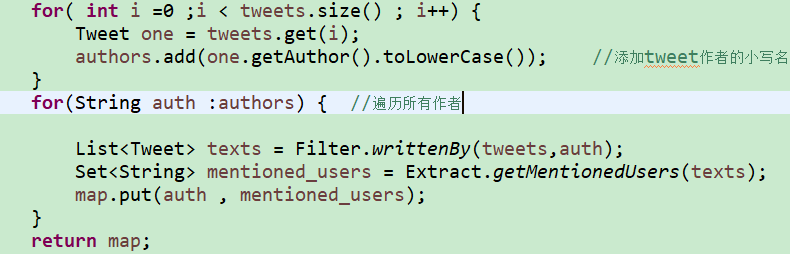


图3.4.3-1 由tweets找到被关注的人

2. **public** **static** List<String> influencers(Map<String, Set<String>> followsGraph)

这个函数是找到影响力最大的人，即被“@”次数最多的人。函数参数是一个followGraph，我们通过这个图可以找到所有的“@”关系。然后我们将influencers和他们被“@”的数量建立对应的关系，用一个Map来存储。最后将Map里的数量排序即可。

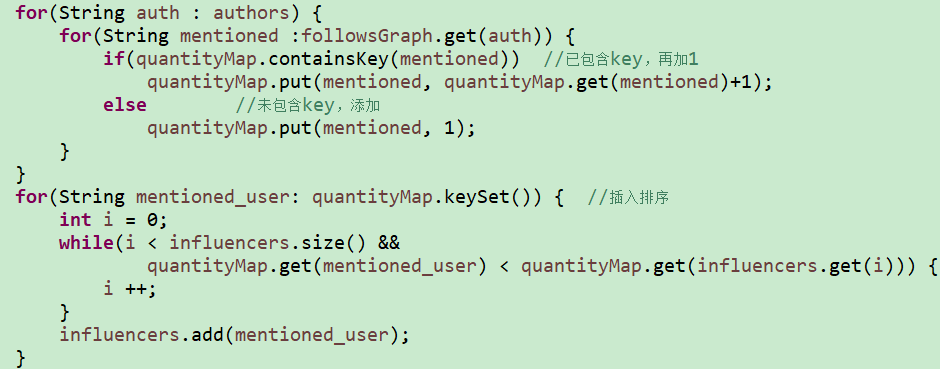


图3.4.3-2 influencers被关注次数并排序

**结果：**

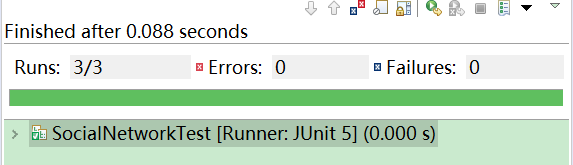


图3.4.3-3 SocialNetwork测试结果

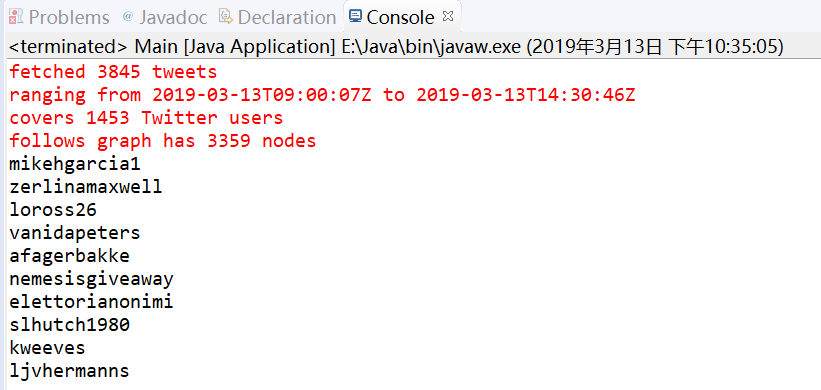


图3.4.3-4 main函数运行结果

### Problem 4: Get smarter

对于这个部分，我选择的是Common hashtags 。在作者的推文中，使用相同标签的人可能相互影响。

首先构造正则表达式”([#A-Za-z]+)”，遍历所有文章，通过这个表达式找到所有带”#”的标记并把它们存入容器tags中。建立一个authorMap，再次对所有文章匹配tags中的标记，把每个标记和文章的作者对应起来。其中每个标记是authorMap的key值，对应的作者是values。最后把每个key对应的作者全部取出，建立互相关注的关系即可。



图3.4.4-1 MyguessFollowGraph主要部分

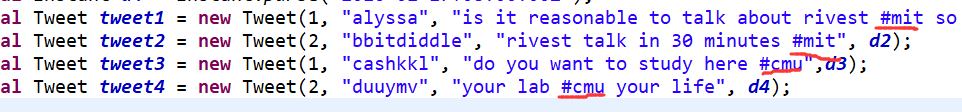


图3.4.4-2 测试用例

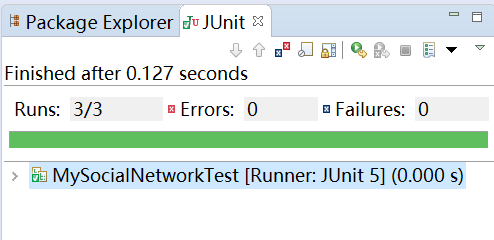


图3.4.4-3 测试结果

# 实验进度记录

请使用表格方式记录你的进度情况，以超过半小时的连续编程时间为一行。

每次结束编程时，请向该表格中增加一行。不要事后胡乱填写。

不要嫌烦，该表格可帮助你汇总你在每个任务上付出的时间和精力，发现自己不擅长的任务，后续有意识的弥补。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 任务 | 实际完成情况 |
| 2019-02-25 | 15:45-17:30 | 配置java的相关环境 | 按计划完成 |
| 2019-03-01 | 13:45-17:30 | 完成isMagicSquare函数并测试 | 未完成debug工作 |
| 2019-03-01 | 18:30-22:30 | 完成debug工作并解决P1 | 延时2小时，完成 |
| 2019-03-02 | 08:00-11:30 | 学会git的操作并自学java | 按计划完成 |
| 2019-03-02 | 14:00-17:00 | P2写到计算轴承 | 延时1小时完成 |
| 2019-03-03 | 13:00-18:00 | 写完P2 | 未完成凸壳 |
| 2019-03-03 | 20:00-23:00 | 换算法写完凸壳 | 未完成，共线问题没解决 |
| 2019-03-04 | 18:30-23:00 | 完成P2，并且P3尽可能向前处理 | 按计划完成 |
| 2019-03-05 | 18:00-22:30 | 设计好FriendshipGraph类 | 按计划完成 |
| 2019-03-06 | 18:30-23:00 | 写好P3的Junit并开始P4 | Junit有些问题，P4开始了前面的部分 |
| 2019-03-07 | 18:30-23:00 | P4写到SocialNetwork | 未完成，测试程序显示问题 |
| 2019-03-08 | 13:30-17:30 | 完成P4 | 按计划完成 |
| 2019-03-08 | 18:30-23:00 | 写实验报告 | 写到P3的Person类 |
| 2019-03-09 | 13:30-17:30 | 写实验报告 | 写到P4的Filter类 |
| 2019-03-09 | 18:30-23:00 | 完成实验报告 | 未完成，还有一些不足 |
| 2019-03-09 | 13:30-17:30 | 完成实验报告并提交 | 按计划完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 配置Java环境的时候总是出现问题 | 重新去搜索了环境变量的设置，发现自己之前设置的是Mac OS里使用的环境变量，并且正确配置了路径。 |
| 不熟悉isMagicSquare()函数中读文件和写文件，同时在数学关系确认行列对角线相等时程序也一直不输出想要的结果 | 在网上查询了很多关于Java的I/O流，读文件和写文件的操作，并且使用的是相对路径。  数学关系编写有误时，我设置断点进行debug，最后发现是有一处存入二维数组时，我行和列弄反了，导致后面的结果不正确。 |
| 程序多次把报错需要定义我调用的类，我使用方法有误等等。 | 我去仔细学习了package和import的机制，正确地引用了一些类文件。 |
| 很多长篇英文不易读懂和正确理解 | 谷歌翻译 |
| 写turtle的轴承函数，算方向角的时候总是讨论不对 | 深入学习了Math里的一些方法，选择反正切函数解决问题 |
| turtle的凸包问题我起初用的是自己想的算法，是一种扫描法，时间复杂性大约是O(n^3)，最后处理共线总是有问题，测试用例的point P12过不了 | 考虑了很久否认了我的算法，它很难处理共线的情况。我最后采用的是类似于礼品包装的一个遍历算法，时间复杂度是O(n^2)，共线问题不用容易处理，而且对所有测试用例很快就通过了。 |
| FriendshipGraph类很容易想到怎么去写，主要就是一个广度遍历，但是对于容器的操作，hashmap的操作和队列的实现不熟练 | 老老实实去网上找资料，翻编程思想的书，把一些基本的java工具弄明白了，并且重新复习了上学期的广度遍历。 |
| 拿到P4感觉很懵，似乎无从下手。 | 这次不是针对一个类，我需要把package中所有的类、接口全部弄清楚，才能清楚地理解整个问题 |
| Extract中getMentionedUsers函数中，我采用的方法是构造正则表达式，但是由于刚开始不太熟悉，构造出来的问题很大 | 重新学习了一番正则表达式的构造 |
| SocialNetwork应该是本实验最难的部分了吧，起初不知道怎么去做 | 还是从Java的图的学习为起点，反复阅读Mit给的指导，最后终于做出来了 |
| P4无法测试 | 把Lab1\_1171000410/src和Lab1\_1171000410/test文件夹放在一个source |

# 实验过程中收获的经验、教训、感想

## 实验过程中收获的经验和教训

**教训：**由于写实验之前Java的学习很浅，并且没有什么实践，于是我采取的是边写边学的方式。但是在实验的过程中，感觉对容器和Map相关的知识掌握不够，导致走了很多弯路。

**经验：**最大的收获还是加强了对Java的练习，学会了Git和GitHub的使用，毕竟第一个实验更多的是为后面的实验做铺垫。另外，我认为以后确定为某个problem使用的策略后，先得仔细学习和掌握有关的知识，这样方法运用起来会更加轻松高效。

## 针对以下方面的感受

1. Java编程语言是否对你的口味？

在学习Java语言之前主要用的是C语言，虽然是Java的初学者，但是感觉在很多情况下，面向对象的Java语言更加好用，并且强大的类库使我们编写代码更加方便。

1. 关于Eclipse IDE

虽然刚接触Eclipse的全英文有点不适应，但是用习惯了觉得挺好用的，IDE也是对程序员很友好，可以让我们减少很多语法错误。

1. 关于Git和GitHub

Git版本控制工具非常强大，很适合软件构造这门课。Github作为程序员的“交流社区”，当然是很棒的。

1. 关于CMU和MIT的作业

CMU和MIT作为世界顶级的计算机学校，他们设计的实验感觉都非常有意思，在编程学习的同时，感觉自己的实践能力也有所提高。

1. 关于本实验的工作量、难度、deadline

工作量挺大的，毕竟每个problem都有不少函数需要完成。难度适中，P4理解起来会有些困难。deadline给的很充裕，我还是做到了提前一周交。

1. 关于初接触“软件构造”课程

初接触这门课，感觉很高端，毕竟ppt都是长篇的英文。实验很有意思且具有挑战性，就是大部分时间都被都投入了这门课，感觉其他课有点慌哈哈。