

**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 | 江建东 |
| 学号 | 1163450201 |
| 班号 | 1737102 |
| 电子邮件 | [1820985520@qq.com](mailto:1820985520@qq.com) |
| 手机号码 | 18846196989 |

**目录**

[1 实验目标概述 1](#_Toc3127409)

[2 实验环境配置 1](#_Toc3127410)

[3 实验过程 1](#_Toc3127411)

[3.1 Magic Squares 1](#_Toc3127412)

[3.1.1 isLegalMagicSquare() 2](#_Toc3127413)

[3.1.2 generateMagicSquare() 5](#_Toc3127414)

[3.2 Turtle Graphics 7](#_Toc3127415)

[3.2.1 Problem 1: Clone and import 7](#_Toc3127416)

[3.2.2 Problem 3: Turtle graphics and drawSquare 8](#_Toc3127417)

[3.2.3 Problem 5: Drawing polygons 8](#_Toc3127418)

[3.2.4 Problem 6: Calculating Bearings 9](#_Toc3127419)

[3.2.5 Problem 7: Convex Hulls 10](#_Toc3127420)

[3.2.6 Problem 8: Personal art 12](#_Toc3127421)

[3.2.7 Submitting 13](#_Toc3127422)

[3.3 Social Network 14](#_Toc3127423)

[3.3.1 设计/实现FriendshipGraph类 14](#_Toc3127424)

[3.3.2 设计/实现Person类 16](#_Toc3127425)

[3.3.3 设计/实现客户端代码main() 17](#_Toc3127426)

[3.3.4 设计/实现测试用例。 18](#_Toc3127427)

[3.4 Tweet Tweet 19](#_Toc3127428)

[3.4.1 Problem 1: Extracting data from tweets 19](#_Toc3127429)

[3.4.2 Problem 2: Filtering lists of tweets 20](#_Toc3127430)

[3.4.3 Problem 3: Inferring a social network 23](#_Toc3127431)

[3.4.4 Problem 4: Get smarter 25](#_Toc3127432)

[4 实验进度记录 26](#_Toc3127433)

[5 实验过程中遇到的困难与解决途径 27](#_Toc3127434)

[6 实验过程中收获的经验、教训、感想 27](#_Toc3127435)

[6.1 实验过程中收获的经验和教训 27](#_Toc3127436)

[6.2 针对以下方面的感受 27](#_Toc3127437)

# 实验目标概述

通过求解四个问题，训练基本Java编程技能，能够利用Java OO开发基本的功能模块，能够阅读理解已有代码框架并根据功能需求补全代码，能够为所开发的代码编写基本的测试程序并完成测试，初步保证所开发代码的正确性，另一方面，利用Git作为代码配置管理工具学会Git的基本使用方法。

# 实验环境配置

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. Eclipse的安装

从官网下载eclipse photon，<http://www.eclipse.org/photon/>，通过安装包安装Eclipse。

2. Git的安装

从官网下载Git，<https://git-scm.com/download/win>，通过安装包安装，输入GitHub的账号完成绑定。

GitHub Lab1的地址：https://github.com/ComputerScienceHIT/Lab1-1163450201

# 实验过程

## Magic Squares

这个任务的主题就是幻方，幻方就是横、纵、对角线的数字和相等的正方形n\*n矩阵。所以任务分解后首先就是设计一种能够判断是否是幻方的算法，其中还要考虑处理spec中的异常情况，增强健壮性。其次就是解释要求2中的奇数幻方的生成算法。最后就是对该算法进行扩展，增加健壮性和输出。

### isLegalMagicSquare()

对于这个函数，首先要拆分为三部分：

1.文件输入 2.数据正确性检查 3.幻方正确性检查

**1. 文件输入**

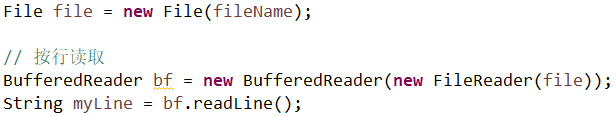
**设计：**

对于文件输入，根据spec的要求，按行输入而不是一次性输入，并采用字符串数组保存每行数据，对于每个数字之间的\t分隔符，使用split函数进行分割。最后对于每个数字（string形式），使用valueOf转换为数字。

**过程：**

对于文件输入，采用Java文件流输入，首先先读入第一行，进行分割后确定该矩阵的列数和应该有的行数，在做完正确性检查后，依次读取文件的每一行。这里使用的文件路径为相对路径，即从项目的根目录出发，这样读取文件相对更加灵活。

其中的关键代码为



按行读取



当文本不为空时，按行读取

**2. 数据正确性检查**

**设计：**

根据spec的要求，数据正确性的检查应该体现在一下这些方面

* 矩阵行列数不相等
* 矩阵并非矩阵，即每行的数字个数不同
* 数字并非正整数
* 数字之间并非使用\t分割

下面分别解释设计思路

1. 行列数不相等，由第一次文本输入解析出的列数与接下来读取的行数比较，如果行数不等于列数，即行列数不相等。所以设置一个行数计数器变量rowlen，每当输入一行符合标准的文件行时，rowlen，最后进行比较

2. 每行的数字不同，矩阵并非矩阵。在每一次用split函数解析完数据行时，检查string[]的大小是否和列数相等，若不想等，则说明缺少数字或多出数字。

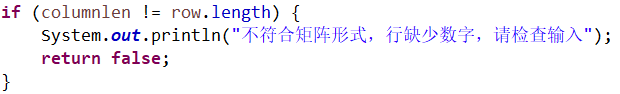
3. 数字并非正整数。单独添加一个辅助函数isNumeric使用正则表达式来匹配是否为正整数，并返回布尔值。

4. 数字之间非\t分割。添加辅助函数isDevidedByTab对传入的string[]遍历检查是否存在空串或者空白符，返回布尔值。

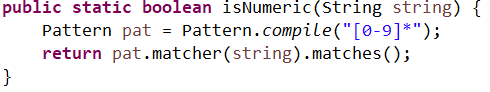
**过程：**

1. 

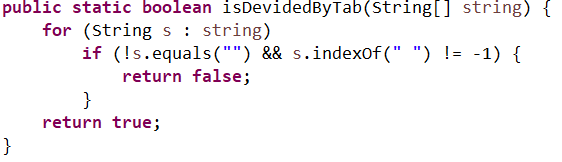
2.判断每行数量



3.判断是否是数字



4.判断是否为空白符



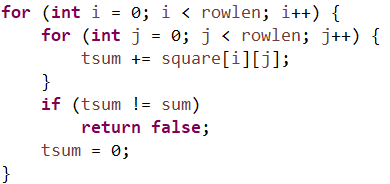
**3. 幻方正确性检查**

**设计：**

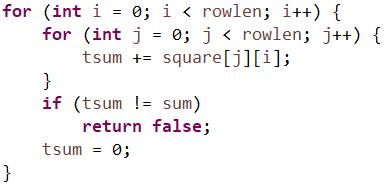
经过前两个步骤得到的矩阵数据，利用一个整型二维数组存储，分别对其每一行、每一列、对角线进行检查，判断是否相等，返回布尔值。

**过程：**

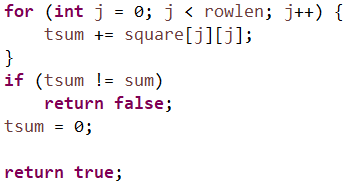
1. 行检查，首先取得第一行的和，然后对每一行进行比较



2. 列检查，同理

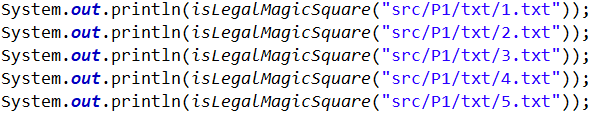


3. 对角线，同理

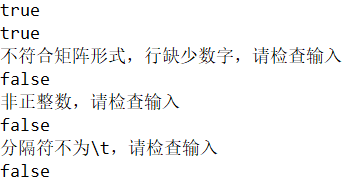


**结果：**

对P1的5个文件进行检查，

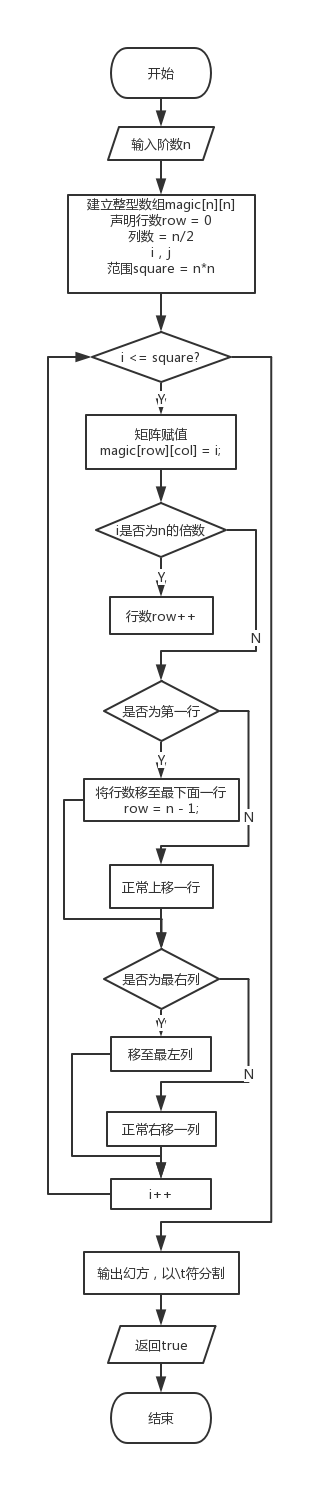


输出结果为



### generateMagicSquare()

**1. 该函数的流程图为**



生成奇数幻方的原理，网上只查到生成规则，具体原理还不清楚。

1. 数字1放在第一行中间

2. 依次放在上一个数的右上角

2.1如果右边出去了就回到左边

2.2 如果上面出去了就放下面

2.3 如果右上角有了就放在这个数的下面

**2. 有关手册中问题的解答：**

1. 如果n为偶数，出现异常java.lang.ArrayIndexOutOfBoundsException

该异常的含义为数组的下标越界，出现这种异常的原因在于代码的这一行

**if** (i % n == 0)

row++;

此时，如果n为偶数，在n为i的倍数时，row代表的行数就会越界，引起越界异常。

2. 如果n为负数，出现异常java.lang.NegativeArraySizeException

该异常的含义为数组的大小不能为负数，出现异常原因为

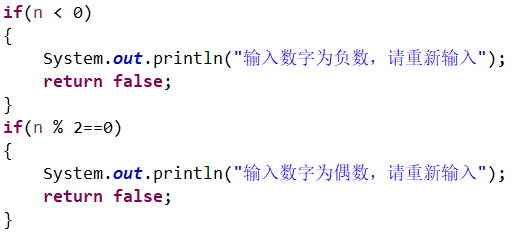
**int** magic[][] = **new** **int**[n][n];

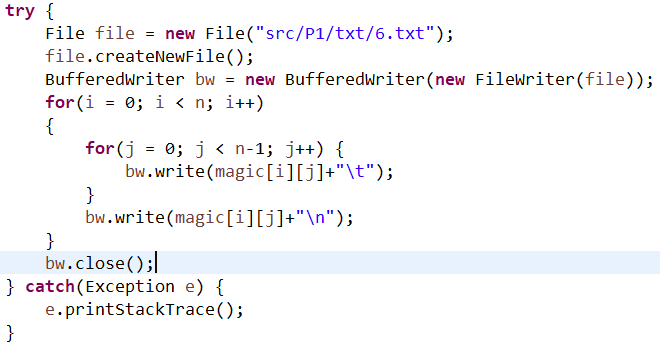
申请数组时大小为负数

**3. 扩展函数以及写入文件**

通过对数字的检查和Java文件流的使用实现功能，此时文件路径依然是相对路径。

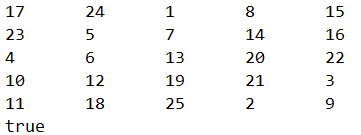
相关代码





**4. 结果**

对该函数进行检测，结果如下（参数为5）



## Turtle Graphics

这是一个Java实现的乌龟画图程序，MIT中设置了多个问题，主要步骤为补充完整代码的实现，项目中自带Junit单元测试，所以也不需要手动实现。问题主要有1.git的使用 2.画图 3.凸包算法

### Problem 1: Clone and import

P2文件夹的地址为：

<https://github.com/rainywang/Spring2019_HITCS_SC_Lab1/tree/master/P2>

因为仓库的单个文件不能单独下载，所以直接克隆整个仓库到本地文件夹，使用**git clone**命令，仓库的地址

https://github.com/rainywang/Spring2019\_HITCS\_SC\_Lab1.git



下载完成后，lab1文件夹下出现了仓库同名文件夹



在文件夹内，打开**git bash**，键入**git init**指令进行初始化



现在就建立了git仓库，之后每改动一个新的文件，使用**git status**命令查看改动的文件是否加入了缓冲区，如果存在，则使用**git add <filename>**将文件加入到缓冲区，最后使用**git commit -m <comment>** 进行本地推送。



这样就可以实现本地管理开发。

### Problem 3: Turtle graphics and drawSquare

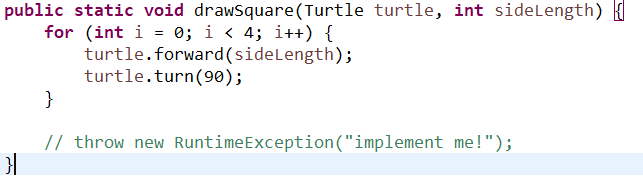
P3要求补全drawSquare函数，使得小海龟可以画出正方形。

**思路：**

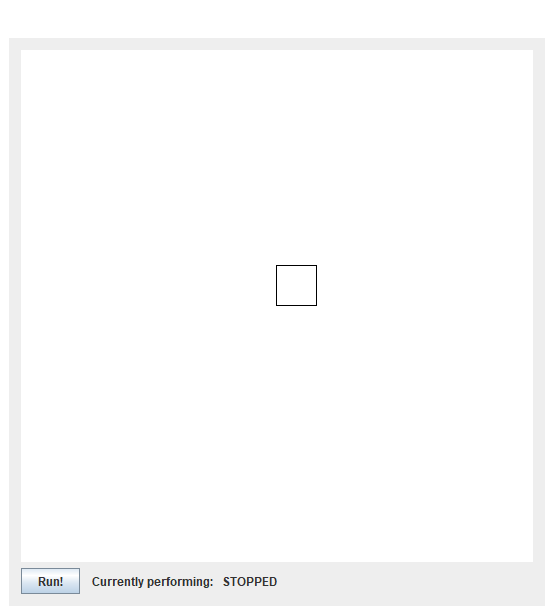
项目提供了两个函数forword(unit)和turn(degrees)，这两个函数分别控制海龟前进的长度和旋转的角度（顺时针），那么如果要画出正方形只需要将前进和旋转循环4遍就可以。

**过程：**

相关代码



**结果：**



### Problem 5: Drawing polygons

P5要求小海龟画出正多边形，输入边数，返回正多边形的内角

**思路：**

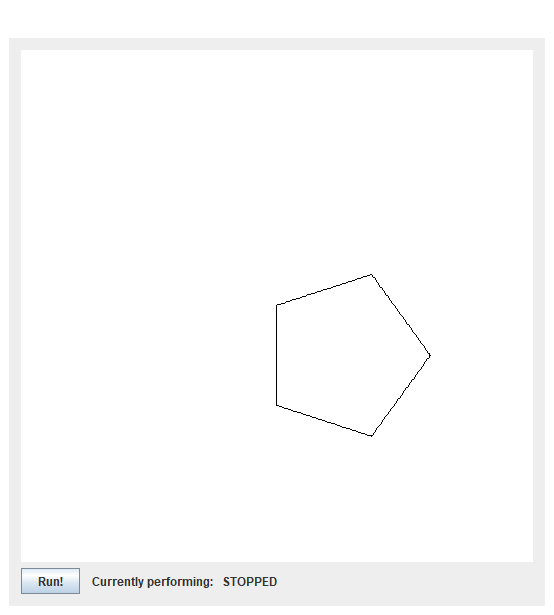
由正多变形的外角和为360°，所以输入边数大于3时，求出每个外角大小，在求出它的补角，即为所求

**过程：**

相关代码



**结果：**

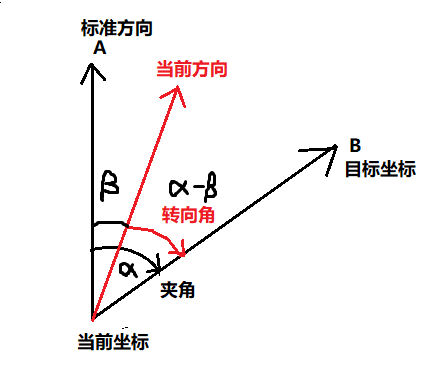


### Problem 6: Calculating Bearings

P4是计算小海龟所面向方向和即将前进方向的夹角，输入是List型位置坐标，返回List型转动方向。这个问题需要补全两个函数calculateBearingToPoint和calculateBearings，其中calculateBearingToPoint是根据输入的当前方向，当前坐标，目的坐标计算出所需要顺时针转动的方向。calculateBearings是通过上述函数计算一系列坐标所对应的旋转方向。

**思路：**

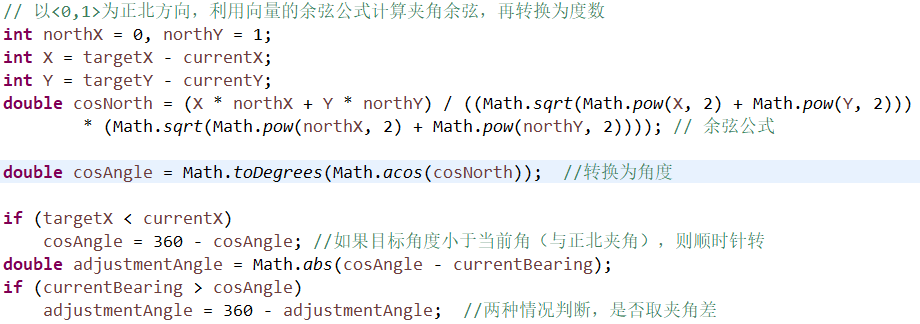
calculateBearingToPoint是通过点坐标来计算方向，所以可以使用向量的余弦公式计算夹角，以正北方向为标准方向，计算出当前点和目标点向量与标准方向的夹角，在与小海龟面对方向比较后，计算出应当转向的角度。



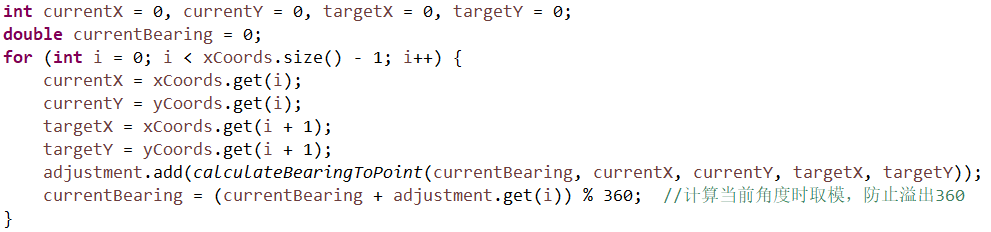
calculateBearings是借助上面函数实现计算一系列的转向角，开始的默认方向是正北，在计算过程中借助一个变量currentBearing保存转向后的海龟面对方向，注意对360求模防止角度溢出错误。

**过程：**

1. calculateBearingToPoint的实现相关代码



2. calculateBearings的实现相关代码



**结果：**

Junit测试通过



### Problem 7: Convex Hulls

P7是一个凸包问题，凸包问题可以采用问题中提到的giftwrapping算法解决。

**思路：**

凸包是所有点中最外则的点的集合，类似与用橡皮筋用把一堆钉子包起来所形成的形状。在这里二维情况使用giftwrapping算法解决，这个算法的思想是：首先，找一个最外则的点（最下和最左）作为起始点。然后对点集中的其他点做一次遍历，分别取两个点与起始点做向量，并进行叉积，比如向量A叉积向量B，如果结果为正，则A逆时针转向B，若为负，A顺时针转向B。这个问题中选择以顺时针方向寻找凸包点，即找到顺时针旋转角度最大的点。这里当两个点位于同一条线时，取距离起始点最远的点作为凸包的点集。

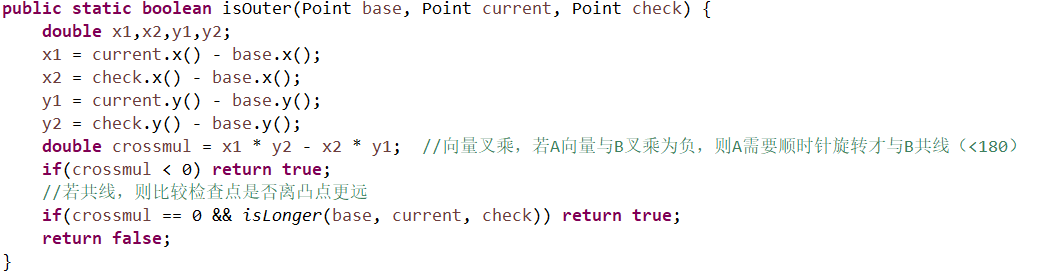
循环下去，直到下一个点是起始点时，结束循环，此时凸包的所有点都已找到。

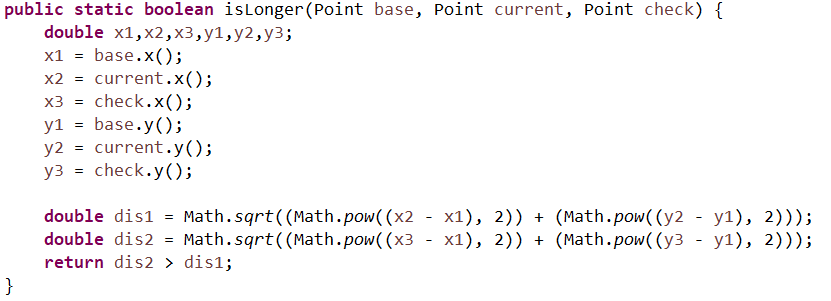
**过程：**

凸包算法实现中要实现的一个功能是，确定下一个点是否是凸包上的点，分解开来，子功能为，下一点与当前凸包点形成的向量是否比当前点的向量更加偏向顺时针方向，如果两点共线，是否下一个点比当前点更远？

所以这两个子功能拆分为isOuter和isLonger。

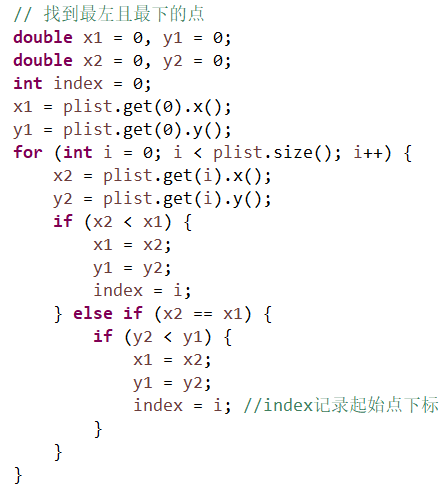
相关代码为



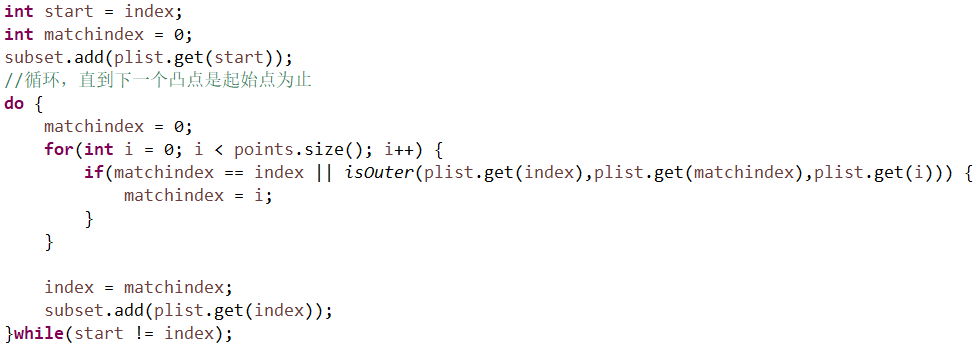


而实现凸包算法的框架为

首先找到最外侧的点，记录点的下标（set转为list）



从最外侧点开始对点集内进行遍历，找到合适的点添加入点集。



找到的点集即为凸包点集

**结果：**

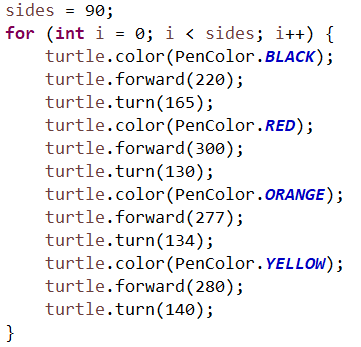
Junit测试通过



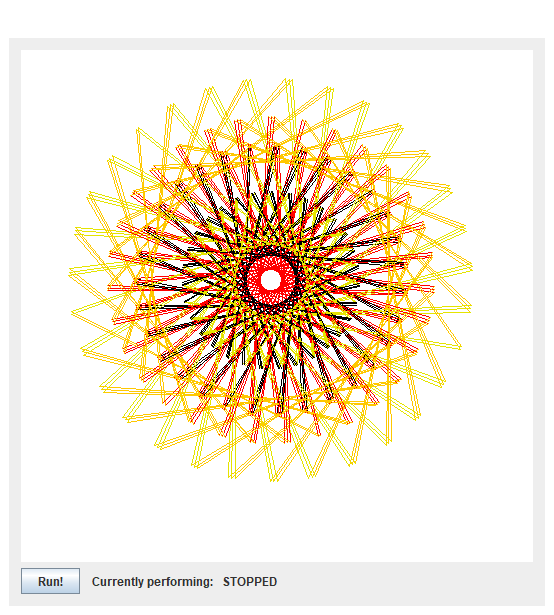
### Problem 8: Personal art

P8是一个自由发挥的问题，选择不同的颜色，重复不同的旋转角度和前进步数画出图案。

**过程：**



**结果：**



### Submitting

完成所有的工作后，使用git add . 命令添加所有文件到缓冲区，git commit -m “finish all the work”进行本地的新版本的提交

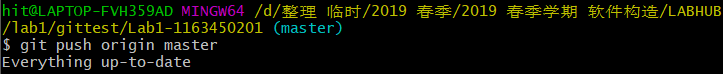




接下来添加远程仓库的链接，使用git remote add origin <address>添加远程仓库

推送，进行新版本的提交





## Social Network

这个任务是要构造一个朋友关系类和Person类，并由此构造一个朋友关系网，计算朋友之间最短的距离。这个任务可以用无向图的BFS解决，在这个问题中，使用队列来实现邻接表的BFS。

### 设计/实现FriendshipGraph类

**设计：**

由于采用BFS的非递归邻接表，所以要实现一个队列以及头结点列表。FriendshipGraph的其他要求在spec中提到的有addVertex，addEdge，getDistance三个函数，以及一系列的健壮性检查，包括重名检查。

**实现思路：**

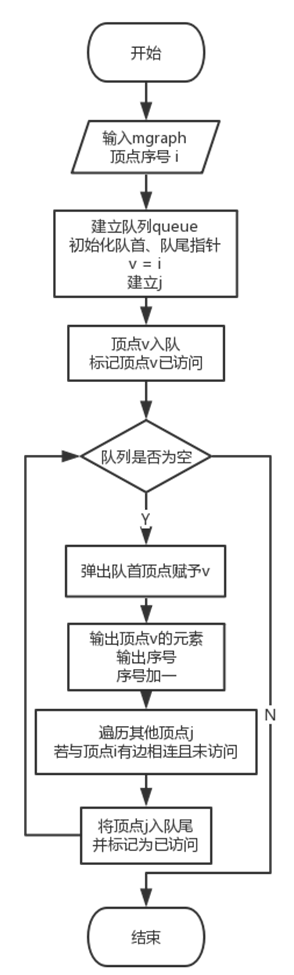
由于要实现头结点列表，所以增加一个List类保存Person对象。

addVertex实现添加结点，在里面增加健壮性检查，成功后会在list中添加顶点。

addEdge实现添加边，因为这里要求的是添加双向边，所以视作有向图处理，例如addEdge（a, b），调用a的addFriend方法，将b添加入a的朋友关系中。

getDistance获取最短路径，首先要依次进行：空对象检查、顶点集是否包含输入检查、重名检查，随后执行BFS非递归搜索。

BFS非递归搜索需要用到队列Queue，在这个类中定义一个队列类Queue，并定义一系列操作方法。BFS会生成BFS生成树，我们所求的最短距离就是这个数的结点层数，由于Person类不能添加无关元素，所以在函数中添加了depth数组，用来记录每个结点的层数。下面是邻接表的BFS代码框架，在每添加队列一个新的结点时，该结点的层数是父结点加一：

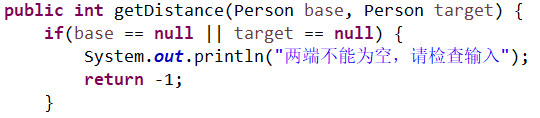


**实现过程：**

1. 建立顶点集vertex

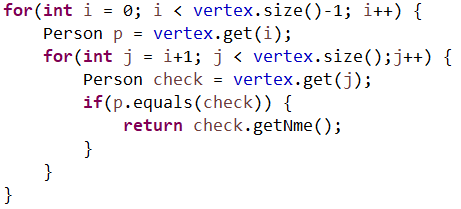


2. getDiatance正确性检查



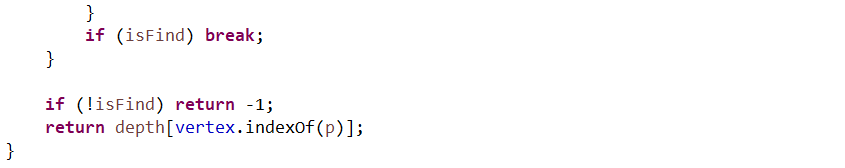


3. 检查重名的相关代码



4. BFS实现





### 设计/实现Person类

**设计：**

Person类根据spec要求应该至少拥有：name姓名、friendship朋友关系集合、addFriend添加朋友关系、getFriendship获得该朋友关系、getName获得姓名、setName设置姓名、Person构造器。

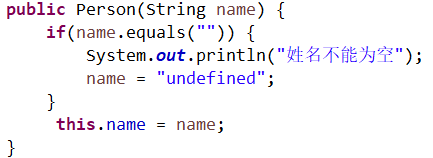
由于是按照邻接表设计，所以Person应该自带相邻的链表，这里采用Set来保存。

**过程：**

**1.** 朋友关系Set



2. Person构造器

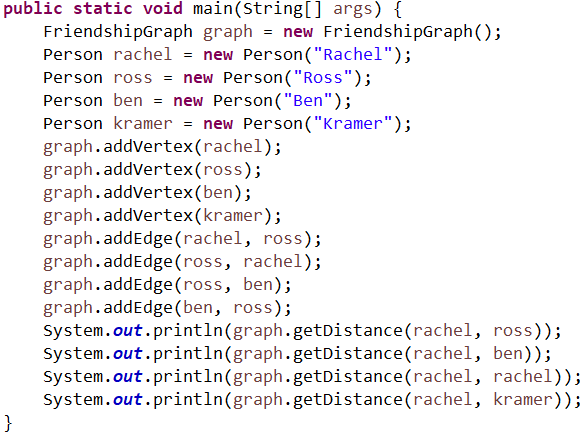


### 设计/实现客户端代码main()

**设计：**

按照spec要求来编写

**过程：**



**结果：**



**手册中的问题：**

如果将上述代码的第 10 行注释掉（意即 rachel 和 ross 之间只存在单向的 社交关系 ross->rachel），请人工判断第 14-17 行的代码应输出什么结果？让 程序执行，看其实际输出结果是否与你的期望一致

由于采用的是邻接表的BFS搜索，少添加一条边就相当于只存在单向关系。也就是从Rachel出发的所有getDistance都会返回-1的结果。执行后与预期一致。



一旦出现重名的顶点，函数会检测并退出。

例如



结果



### 设计/实现测试用例。

**设计：**

测试用例分别测试addVertex、addEdge、getDistance

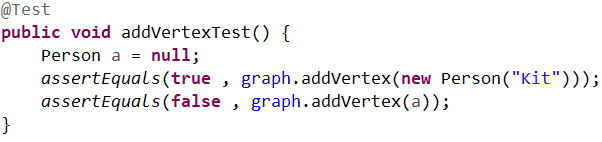
addVertex主要测试两种，空对象和正确输入

addEdge主要测试，两端输入为空，一端为空，正确输入

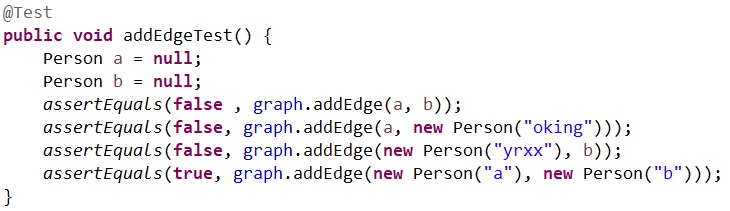
getDistance主要测试，存在顶点未加入点集，不存在边，存在空对象，自身比较，多点无向图输入。

**过程：**

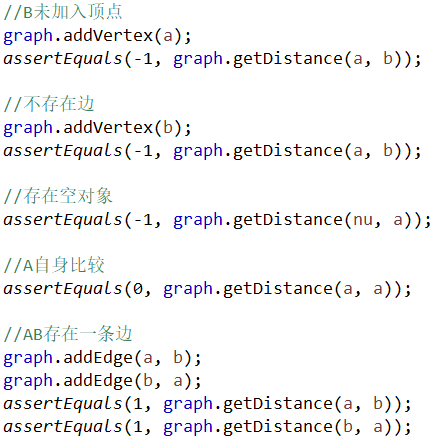
1. addVertex测试用例

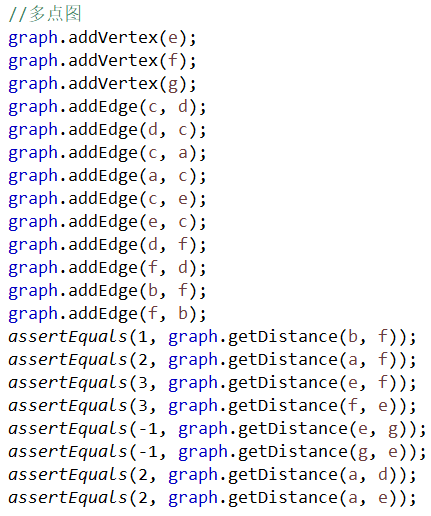


2. addEdge测试用例

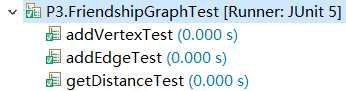


3. getDistance测试用例





**结果：**



## Tweet Tweet

这是一个有关Tweet的任务，任务中涉及到了正则表达式，信息过滤，推测关系网等。需要用到Java的正则表达式、映射、集合等。

### Problem 1: Extracting data from tweets

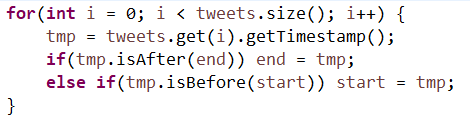
**设计：**

P1中要求实现对推特最小时间间隔的查询和@用户的查询，时间间隔是用Instant类来实现。这两个功能分别用getTimespan和getMentionedUsers实现，getTimespan中只需要遍历一遍推特的时间，找出最早最晚的时间即可。提取@用户名时，spec要求不区分大小写并且只需要出现一次，且类似于邮箱之类的不能作为用户名。这里使用正则表达式实现提取用户名，为了不区分大小写，添加一个checkuser的集合保存用户名的小写形式，每提取一个用户名，在小写集合中检查是否重名，不是则作为新的用户名添加。

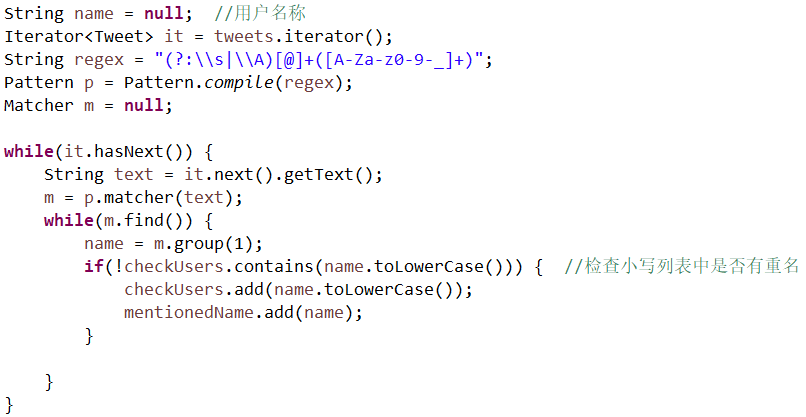
同时，P1还要求实现ExtractTest，对函数进行测试，测试用例选择以单个、多个来进行测试。

**过程：**

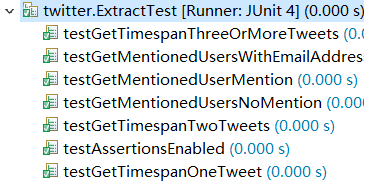
**1.** getTiemspan的相关代码



2. getMentionedUsers相关代码



**结果：**



### Problem 2: Filtering lists of tweets

P2任务是实现一个过滤器功能，即查找某位用户所发布的推特和查找某一时间段内所发布的所有推特还有查找关键字。以及它的测试单元。

**设计：**

这个过滤器功能由三个函数组成，writtenBy、inTimespan、containing。

**writtenBy**：

首次对用户名进行合法性检测，接着对推特list进行合法性检查。检查完成后，为了实现大小写不敏感，将用户名转换为小写，对tweets进行遍历，getAuthor获得用户名的小写形式比较。如果匹配，加入list。

**inTimespan**：

同样是遍历的思路，注意到timespan是包括endpoint的时间点，也就是说它是一个闭区间，在遍历tweets要注意这一点。

**containing：**

这个函数是查询包含一系列关键字的推特。首先对关键字list进行合法性检查，排除空白字符和空字符集，接下来就是遍历，注意大小写不敏感。

对于单元测试的用例，遵循以下几点

1.正确输入，包括查找成功和失败

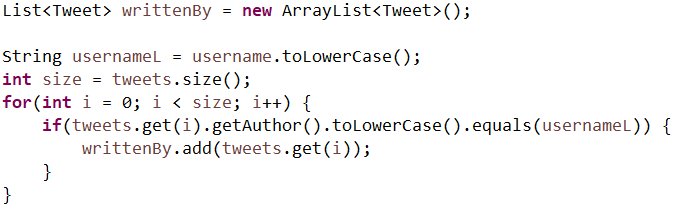
2. 异常输入

3. 边界条件

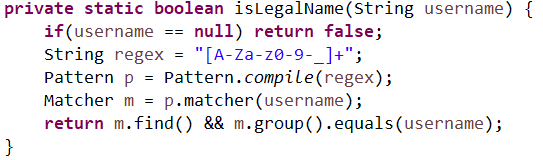
**过程：**

相关代码

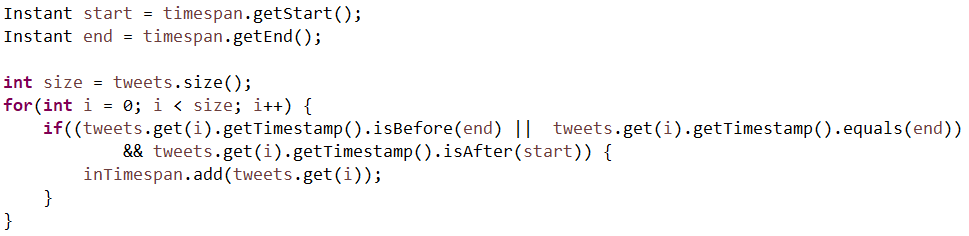
1. 获取用户的推特



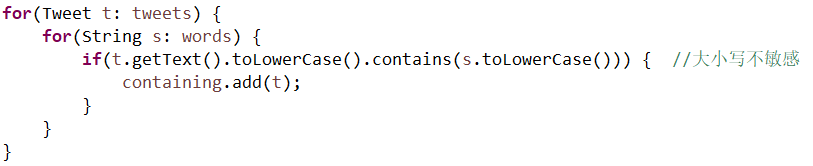
其中，检测用户名合法性



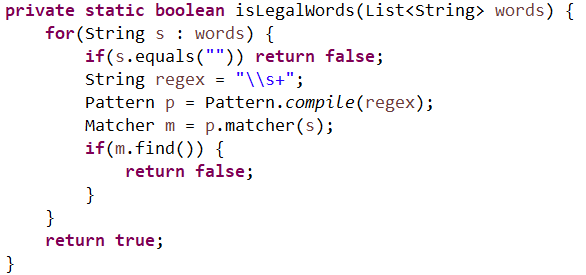
获取时间段推特



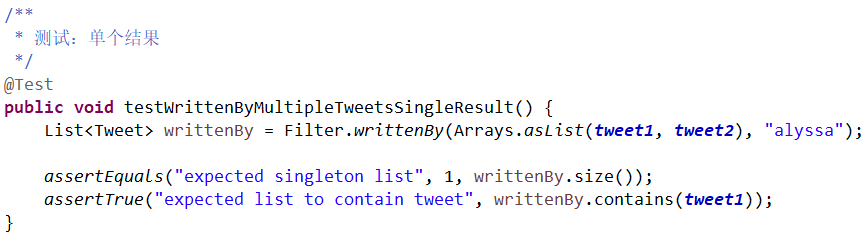
2. containing函数

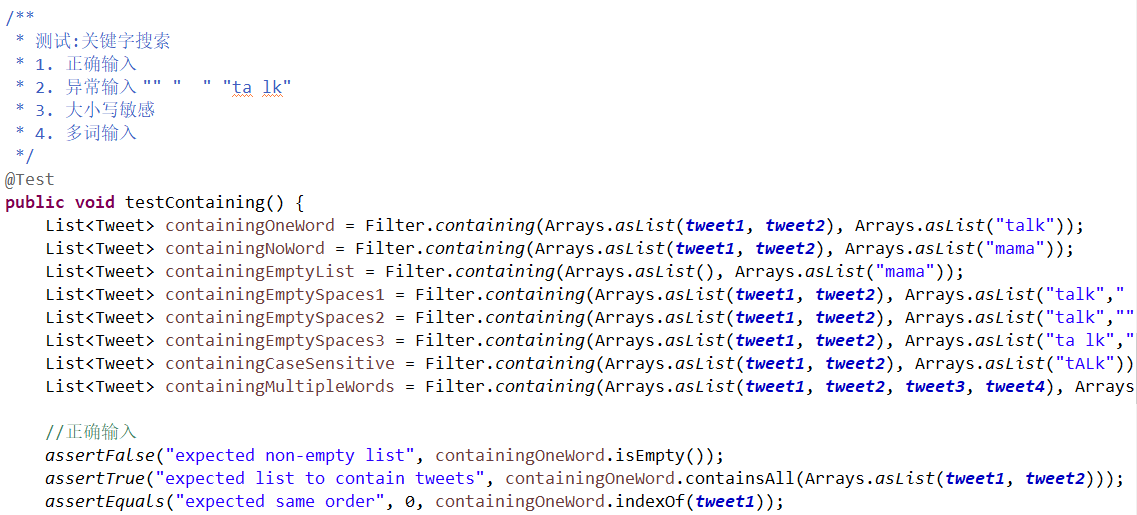


3. 单词合法性检查

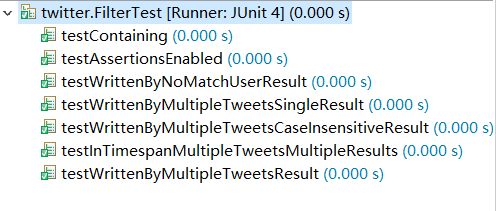


4. 测试用例部分代码





**结果：**



### Problem 3: Inferring a social network

P3任务是建立社交网络，以及获得影响力从大到小用户的列表

**设计：**

任务包含两个函数gussFollowsGraph和influencers。

**gussFollowsGraph：**

这个函数的实现类似与上一问，就是将list换为了map，基本流程是：通过正则表达式获取用户名，检查后添加入Set中，不断循环更新Set，最后添加入Map中，返回Map。

**influencers：**

统计最受欢迎的人，利用Map做一个简单的计数器，每个人对应着收到@的数量。接着做一个Comparator接口的comparerTo函数比较Map中的大小顺序，关键点在于，如果A和B的受欢迎次数相同，则他们按照姓名的字典排序排序。

单元测试遵循三个方面：

1. 正确输入

2. 异常输入

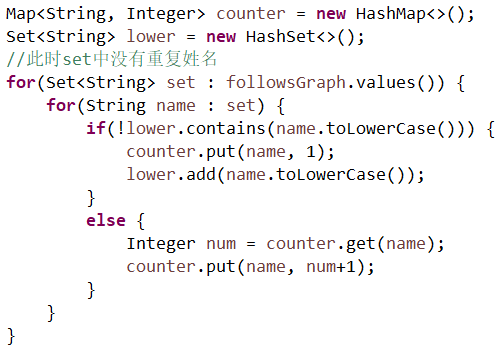
3. 边界输入

**过程：**

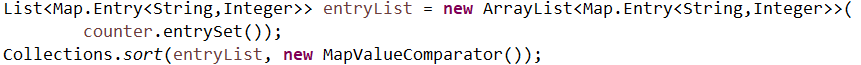
1. guessFollowsGraph相关代码

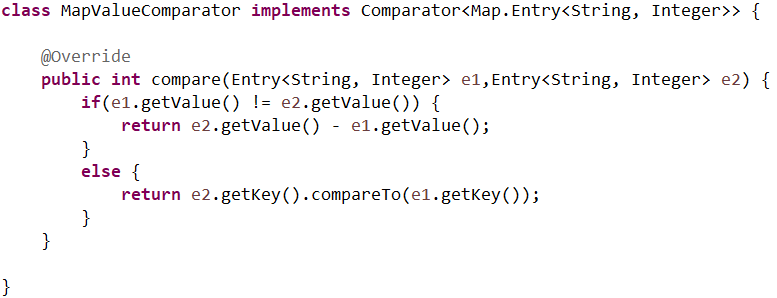


2. influencers里的Map计数器相关代码

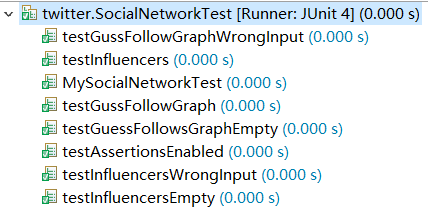


3. 比较器相关代码





**结果：**



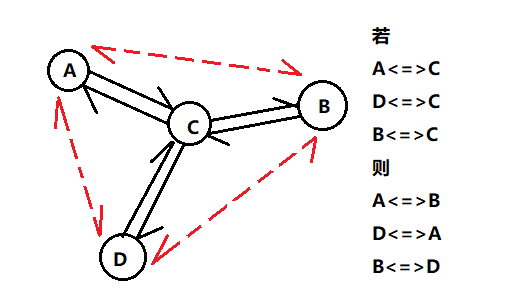
### Problem 4: Get smarter

P4任务为使得guessFollowsGraph函数更加智能，可以选择下面三个方面之一实现：共同标签、三元闭包、间接回复（？）

这里选择三元闭包实现。

**设计：**

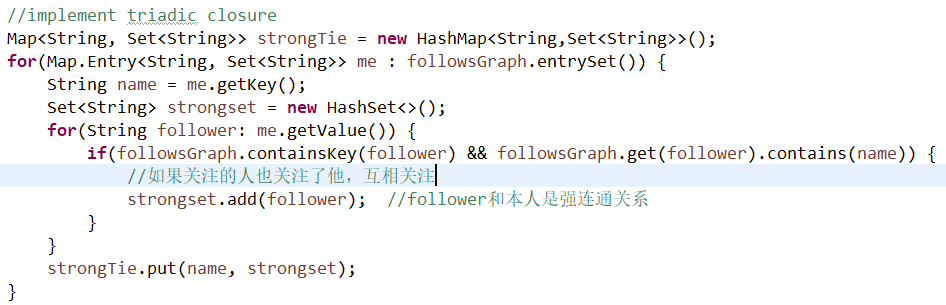
所谓三元闭包指，A与B互相关注，B与C互相关注，那么在某种程度上，A与C也具有互相关注的可能，因此要将A和C互相加入对方的followers中。为了找到这种相互关系，需要找到所有的强连通关系，如果将所有强连通关系提取出来，那么任意两个存在中间结点的结点都需要添加到对方的followers中。



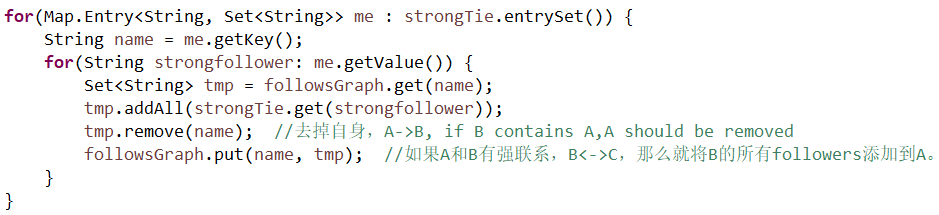
在此基础上，如果C有多个强关系，而A又与C有强关系，为了给A添加朋友关系，只需要将C的所有强关系（除了A自身）添加到A的followers即可。

**过程：**

在原函数基础上，添加一张Map，strongTie用来记录强关系网。



双重循环，添加关系



特别的，由于使用了用同一个函数，所以测试用例有所区分，**原函数测试用例不包含强关系网**。

**结果：**



# 实验进度记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 任务 | 实际完成情况 |
| 2019-02-27 | 20:00-22:00 | 编写问题1的isLegalMagicSquare函数并进行测试 | 按计划完成 |
| 2019-02-28 | 14:30-16:00 | 编写问题1的generatingSquare相关资料 | 延期1小时完成 |
| 2019-02-28 | 20:39-22:40 | 编写Turtle问题partⅠ | 按计划完成 |
| 2019-03-03 | 10:00-14:00 | TurtleⅡconvexHull | 延期1小时完成 |
| 2019-03-06 | 18:00-23:00 | FriendshipGraph && Person | 按计划完成 |
| 2019-03-07 | 15:00-23:00 | TweetFilter & TweetExtract | 延期1小时完成 |
| 2019-03-08 | 18:00-22:00 | TweetSocialNetwork | 延期1小时完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| git 切换账号 | 查阅资料，更改global config即可 |
| 不理解奇数幻方的生成方法 | 查阅资料，查到了奇数幻方的步骤 |
| 凸包问题giftwrapping算法的实现 | <https://www.cnblogs.com/hxsyl/p/3222688.html>讲解了凸包求解的原理和实现 |
| Map的比较器实现 | <https://blog.csdn.net/xiaokui_wingfly/article/details/42964695>讲解了如何做 |
| Set、Map不会用 | 查阅Java核心技术卷1 |
| Java文件输入输出流 | 查阅博客 |
| 将Set转为List | 查阅博客 |

# 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

1.先构思好，再进行编码。先将主体的框架构思好，采用什么算法，使用什么数据结构，再动手。编码是所有环节的最后一步。

2.测试驱动。根据所要实现的功能，先编写测试用例，接着再实现功能函数，这样就保证了函数的正确性，同时在后面的重构中也避免出错。

3.保持类和方法功能的单一性。如果一个函数要实现一个功能需要多个子环节，尽可能将子环节抽象出来作为单独的辅助函数，这样不仅能思路清晰，还可以在debug时减少抓狂几率:)。

## 针对以下方面的感受

1. Java编程语言是否对你的口味？

比较适合，主要是面向对象的编程思想相较于c更加抽象，合理。

1. 关于Eclipse IDE

挺好用的，带有的代码自动补全很好用。

1. 关于Git和GitHub

作为版本管理平台，两者搭配起来对于版本控制用处非常大，作为其他文件管理也很适用。

1. 关于CMU和MIT的作业

趣味性很强，同时综合性也很好。在tweet任务中同时涉及到多个Java概念，而且本身这个任务也可以作为大数据分析的一个小小实践。

1. 关于本实验的工作量、难度、deadline

难度适中，工作量相当于0.8个C语言期末大作业，8个Java平时作业，作为第一份作业工作量不小。

1. 关于初接触“软件构造”课程

这门课应该挺有意思，先上几周再看看。