

**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 | 卢兑玧 |
| 学号 | L170300901 |
| 班号 | 1703009 |
| 电子邮件 | nty0725@naver.com |
| 手机号码 | 15663729008 |

**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

本次实验通过求解四个问题（其中一个可选），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作为代码配置管理的工具，学会Git的基本使用方法。

# 实验环境配置

由于在之前已经完成过Github的注册与Java开发环境的配置，所以只需完成github classroom的关联即可。

GitHub Lab1仓库的URL地址： <https://github.com/ComputerScienceHIT/Lab1-L170300901>

可使用git clone将空仓库克隆后在本地用命令行对仓库进行操作，命令行操作如下：

git init

git add .

git commit -m "first commit"

git remote add origin https://github.com/ComputerScienceHIT/Lab1-L170300901.git

git push -u origin master

# 实验过程

## Magic Squares

实验任务是关于幻方的，细化主要任务就是有读写txt文件，按格式存入数组，然后检验是否是幻方（每行每列对角线之和为同一个数字），还有生成幻方，还有包括对异常（输入）的处理，增加程序健壮性。

### isLegalMagicSquare()

这个方法的功能是判断文件的数据是否为幻方。是否是幻方又包括：是否是矩阵、是否由\t分割、是否全为正整数、然后才是是否行列对角线全为一个相同的数。

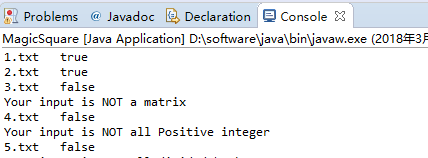
伪代码：

1. 定义一个200\*200的数组；
2. 打开文件讲数据读入数组（如果文件不存在，报错）
3. **while**(文件没有到末尾)
4. {
5. 读取一行；
6. 根据\t分割
7. 如果包含"，"、空格，说明至少不全是由\t分割的，返回**false**并报错
8. 将这一行的每一个字符串转到整形存到数组的对应行中
9. 如果不能出现不能转整形的异常，或者转换出来的整数小于等于0，说明数组文件并不全是正整数，返回**false**并报错
10. }
11. 记录行数和列数，如果读入总数不等于行数\*列数，则说明不是矩阵，返回**false**并报错
12. 求每一行每一列每对角线的和，再比较是否全部都相等
13. 如果相等 返回ture
14. 否则返回**false**

其中读入文件，判断是否分割正确、为矩阵的部分代码：

1. \\打开文件
2. **try** {
3. in = **new** BufferedReader(**new** FileReader(myFile));
4. String str;
5. **while** ((str = in.readLine()) != null) {
6. **int** rownum = 0;
7. line = str.split("\t");
8. **for** (String s : line) {
9. **try** {
10. **if** (s.contains(" ") || s.contains("\\,")) {
11. errkind = 2;  //  \\不全由”\t”分割
12. **return** **false**;
13. }
14. **if**(Integer.parseInt(s)<=0)
15. {
16. errkind = 3;   \\不是正整数
17. **return** **false**;
18. }
19. rownum++;
20. num++;
21. array[linenum][rownum] = Integer.parseInt(s);
22. } **catch** (Exception e) {
23. errkind = 3;  \\转换为整数出现异常，即输入不是整数
24. **return** **false**;
25. }
26. }

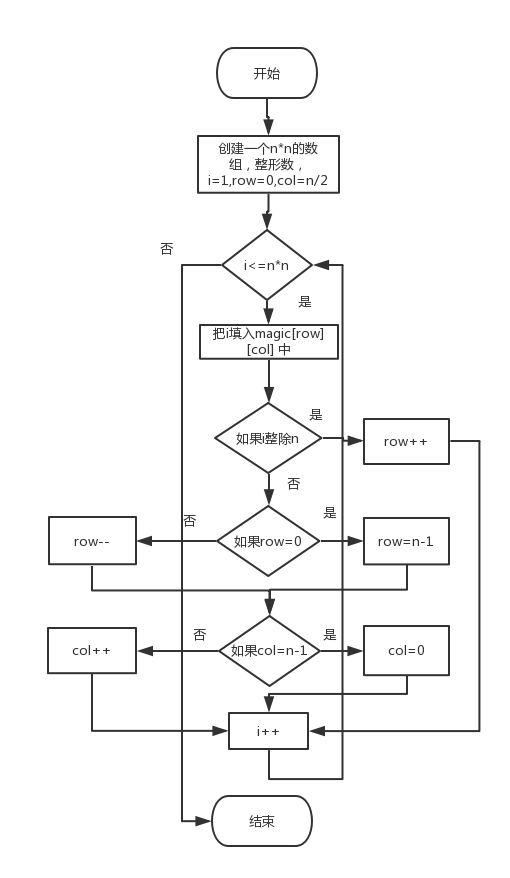
  输出结果：



### generateMagicSquare()

这个方法的功能是生成一个n阶幻方，n为正奇数，由用户输入，输入不正确会报错。由于生成幻方的主要部分题目已经给出，所有我们的重点是判断输入合法性和将生成的幻方写入一个新文件：6.txt。主要是学习了读写文件的基本操作。

流程图：



具体操作：

把1(或最小的数)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；

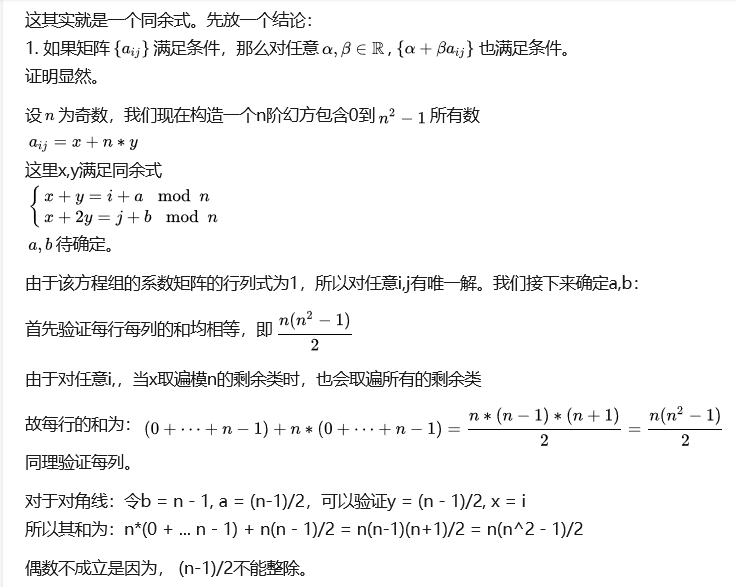
　　(2)、如果这个数所要放的格已经超出了顶行那么就把它放在底行，仍然要放在右一列；

　　(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，那么就把它放在前一个数的下一行同一列的格内；

　　(5)、如果这个数所要放的格已经有数填入，处理方法同(4)。

原理：



出处：<https://www.zhihu.com/question/30498489/answer/161657996>

<https://www.zhihu.com/question/30498489/answer/48393702>

报错原因：

**偶数：java.lang.ArrayIndexOutOfBoundsException：**

使用不合法的索引访问数组时会报数组越界，可能的原因是使用的角标大于等于数组的长度或为负数

**负数：java.lang.NegativeArraySizeException**

如果应用程序试图创建大小为负的数组，则抛出该异常。

测试结果：

C:\Users\杨依婕\AppData\Roaming\Tencent\Users\854823622\TIM\WinTemp\RichOle\AWBX7~0{I5AM]VHGEQC@N3I.png

## Turtle Graphics

Turtle Graphics任务目标为实现TurtleSoup.java中的多个函数并进行测试，通过实验熟悉Java语法和开发环境、Eclipse、Junit测试、Git版本控制。

### Problem 1: Clone and import

P2不是一个独立的仓库 git clone貌似不能直接clone下整个仓库 可以将整个整个都clone下来

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

如果有svn的话可以单独clone下P2文件夹

$ svn export <https://github.com/rainywang/Spring2019_HITCS_SC_Lab1/trunk/P2>

然后进入到clone下的文件夹，将其初始化为git仓库 commit初始版本

$ cd P1

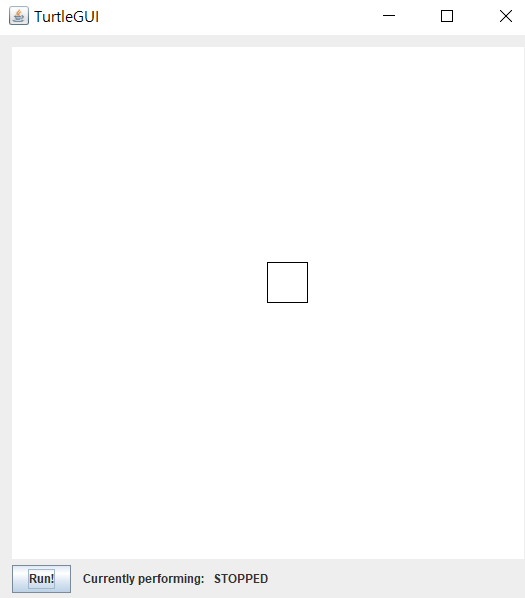
$ git init

$ git add .

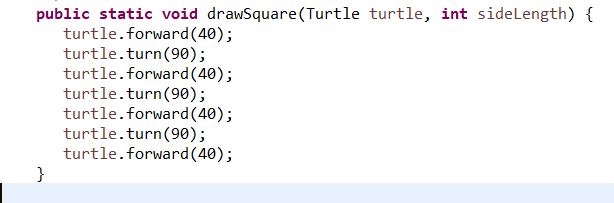
$ git commit –m “origin”

### Problem 3: Turtle graphics and drawSquare

forward(units)，直行，turn(angle)顺时针转弯，通过调用这两个方法画出正方形。很简单：



代码是：



### Problem 5: Drawing polygons

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

**try**{

turtle.turn(270);

**double** angle = 360.0 - *calculateRegularPolygonAngle*(sides);

**for**(**int** i = 0; i < sides; i++){

turtle.forward(sideLength);

turtle.turn(angle);

}

}**catch**(Exception e) {

**throw** **new** RuntimeException("implement me!");

}

### Problem 6: Calculating Bearings

该阶段需实现两个函数：calculatingBearingToPoint 和calculatingBearings。使用向量点乘的数学原理及公式可算出结果。可以把从起点到终点抽象为向量，旋转的角度即向量与y轴之间的夹角。点乘与向量长度的商即角度的余弦值。对角度的取值范围应进行处理。可调用Java中的Math类中的常量与函数进行数学运算。

double length = Math.sqrt(x\*x + y\*y);

double angle = Math.acos((double)y / length)/Math.PI\*180;

return angle >= currentBearing ? angle - currentBearing : angle - currentBearing + 360;

运行calculatingBearingTest可对自己的函数进行测试与修改。通过JUnit测试后的图片如下所示：



### Problem 7: Convex Hulls

在这里使用暴力的算法——枚举来选择Convex hulls。Convex hull即最小的包括全部点的定点集，在这里使用了简单的数学方法去测试点是否为Convex hull的点：（debug很久）

具体思想：对于一个n个点集合中的两个点p1和p2，这两个点可以确定一条直线。当且仅当该集合中的其它点都位于这条直线的同一边时，这条直线就是Convex hulls的边界，也就是说p1和p2就是Convex hulls中的顶点。

在坐标平面上穿过两个点(x1, y1)，(x2, y2)的直线公式如下：

ax + by = c

其中 a = y2 – y1, b = x1 – x2, c = x1\*y2 – y1\*x2。

根据上面的公式，我们只需要把每个点代入公式ax+by-c，判断公式计算结果的符号是否全部大于0或者小于0，如果是则是Convex hulls上的点，否则不是。该算法的时间效率为O（n^3）。

//统计点在直线上侧和下侧的情况

for (Point point : points) {

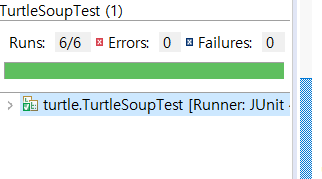
double result = a\*point.x() + b\*point.y() - c;

if (result > 0) flag++;

if (result < 0) flag--;

}

测试结果如图所示：（get a green bar :D）



### Problem 8: Personal art

不好意思，，，我没做

### Submitting

首先使用add将全部内容加载到工作区，commit提交后push到github远程仓库。

$ git add .

$ git commit –m “finish P2”

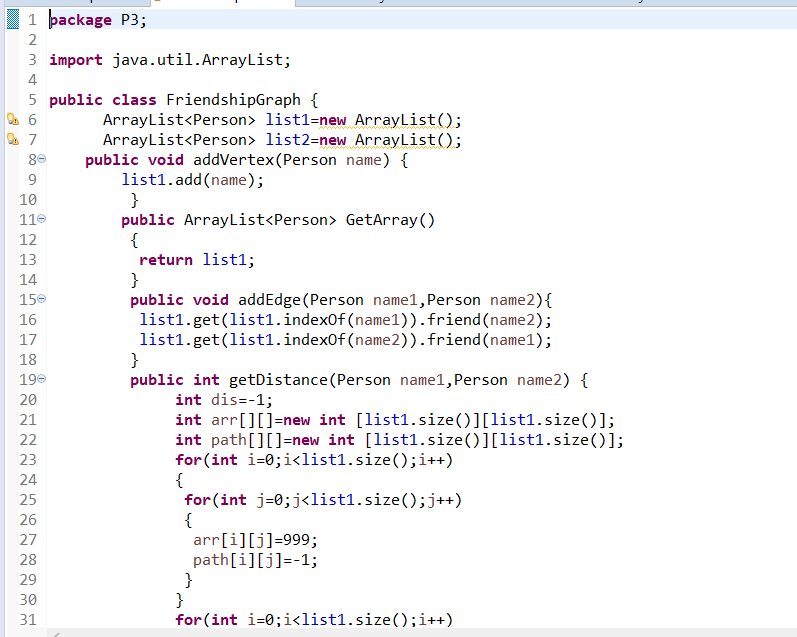
$ git push

。

## Social Network

这个任务中的需求为建立一个无向图，并能够计算任意两个点之间的最短距离，并使用JUnit对代码中的函数进行测试。

### 设计/实现FriendshipGraph类







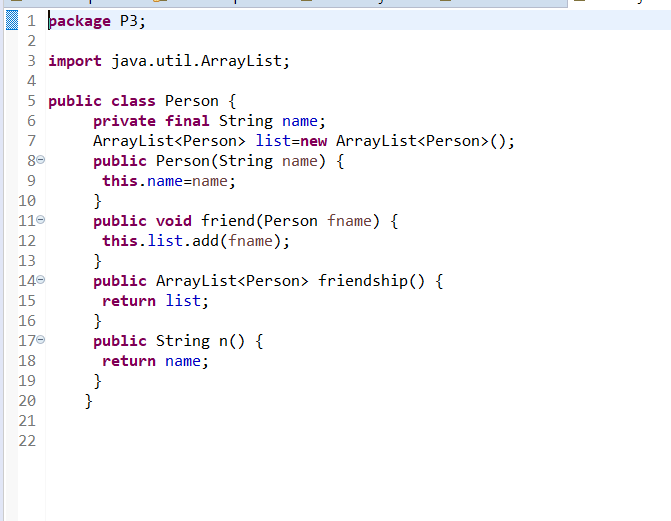
### 设计/实现Person类

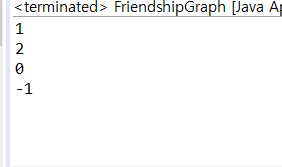
思考Person的特征，抽象出以下两点特征：

- 姓名 （String）

* - 朋友／认识的人 （使用List<Person>储存）
* 为了保证变量的私有性，可设置两个get接口供外界访问成员变量。
* 若新增此Person对象的认识的人，可直接在list中添加Person对象。

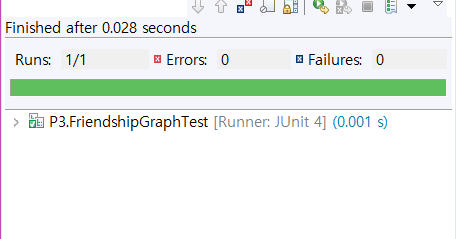
### 设计/实现客户端代码main()





### 设计/实现测试用例

在Test中应对FriendshipGraph的三个函数都进行测试。实现难点为实验要求中对权限进行了要求：“Use proper access modifiers (public, private, etc.) for your fields and methods. If a field/method can be private, it should be private.”而三个函数均可为private，JUnit只能直接测试public函数。在询问了TA之后选择在查找解决方案。最终选择使用Java反射的语言特性进行测试，核心代码如下所示：



## Tweet Tweet

### Problem 1: Extracting data from tweets

第一个方法：public static Timespan getTimespan(List<Tweet> tweets)

功能为获取这些tweet的时间跨度，即最早时间到最晚时间。

首先完成测试：

需要测试的部分有 tweet 列表为空，1个tweet,2个tweet两个时间，3个tweet3个时间

伪代码：

1. Instant starttime=Instant.MAX,endtime=Instant.MIN ;
2. **if** (tweets为空){
3. **return** **new** Timespan(Instant.now(), Instant.now());
4. }
5. **else** {
6. **for** (Tweet tweet : tweets) {
7. **if** (tweet.getTimestamp().在endtime之后) {
8. endtime = tweet.getTimestamp();
9. }
10. **if** (tweet.getTimestamp().在starttime之前) {
11. starttime = tweet.getTimestamp();
12. }
13. }
14. **return** **new** Timespan(starttime, endtime);

第二个方法：public static Set<String> getMentionedUsers(List<Tweet> tweets)

功能为返回每个tweet提到的人

首先完成测试，分类包括没有提到任何人，提到一个人，提到多个（三个）人。

1. **public** **static** Set<String> getMentionedUsers(List<Tweet> tweets) {
2. //   throw new RuntimeException("not implemented");
3. Set<String> mentionedUsers = **new** HashSet<>();
4. Pattern p = Pattern.compile("\\B@[A-Za-z0-9\_-]+\\b");//正则匹配“ @xxx  ”
5. **for** (Tweet tweet: tweets) {
6. String text = tweet.getText().toLowerCase();//全部转化为小写
7. Matcher m = p.matcher(text);
9. **while** (m.find()) {
10. String userMentioned = m.group().substring(1);//取@之后的内容
11. mentionedUsers.add(userMentioned);//增加@的人
12. }
13. }
14. **return** mentionedUsers;
15. }

### Problem 2: Filtering lists of tweets

第一个方法**public** **static** List<Tweet> writtenBy(List<Tweet> tweets, String username)

它的功能是返回这些tweets中是username写的，只需要循环然后依此使用getAuthor()即可。

首先完成测试，包括空推特、写了一篇tweet,写了多篇（三篇）tweets

1. **public** **static** List<Tweet> writtenBy(List<Tweet> tweets, String username) {
2. // throw new RuntimeException("not implemented");
3. List<Tweet> tweetswrittenBy = **new** ArrayList<>();
4. **for** (Tweet tweet : tweets) {
5. **if** (tweet.getAuthor().equalsIgnoreCase(username)) {
6. tweetswrittenBy.add(tweet);
7. }
8. }
9. **return** tweetswrittenBy;
10. }

第二个方法是**public** **static** List<Tweet> inTimespan(List<Tweet> tweets, Timespan timespan)它的功能是找到某个时间范围内的所有推特，也只需循环判断时间即可。

首先完成测试：返回包括有0篇、1篇、多篇（三篇）推特

1. **public** **static** List<Tweet> inTimespan(List<Tweet> tweets, Timespan timespan) {
2. **throw** **new** RuntimeException("not implemented");
3. List<Tweet> tweetsinTimespan = **new** ArrayList<>();
4. **for** (Tweet tweet : tweets) {
5. **if** (tweet.getTimestamp().isBefore(timespan.getEnd()) && tweet.getTimestamp().isAfter(timespan.getStart())) {
6. tweetsinTimespan.add(tweet);
7. }
8. }
9. **return** tweetsinTimespan;
10. }

第三个方法是**public** **static** List<Tweet> containing(List<Tweet> tweets, List<String> words)

功能是找到含有words的所有推特，但是要注意大小写的问题所以直接把文本和words全部转换为小写，把每个tweet的文本切割成一个list然后使用List.contains（）方法即可。

首先完成测试，包括没有包含单词，有一篇包含，三篇包含。

1. **public** **static** List<Tweet> containing(List<Tweet> tweets, List<String> words) {
2. **throw** **new** RuntimeException("not implemented");
3. List<String> List = **new** ArrayList<>();
4. **for** (String word : words) {
5. List.add(word.toLowerCase());
6. }
7. List<Tweet> tw = **new** ArrayList<>();
8. **for** (Tweet tweet : tweets) {
10. List<String> wordsnew = **new** ArrayList<String>(Arrays.asList(tweet.getText().split(" ")));
11. **for** (String s : wordsnew) {
12. **if** (List.contains(s.toLowerCase())) {
13. tw.add(tweet);
14. **break**;
15. }
16. }
17. }

### Problem 3: Inferring a social network

第一个方法是**public** **static** Map<String, Set<String>> guessFollowsGraph(List<Tweet> tweets)功能是给出一个Map每个人对应他关注的人（此时的关注就是根据如果有@某人就一定关注他）

首先写测试，关注0人，1人，3人

伪代码：

1. **public** **static** Map<String, Set<String>> guessFollowsGraph(List<Tweet> tweets) {
2. //     throw new RuntimeException("not implemented");
3. Map<String, Set<String>> SocialNetwork = **new** HashMap<String, Set<String>>();
4. List<Tweet> socialtweet = **new** ArrayList<Tweet>(tweets);
5. **for** (Tweet tweet: socialtweet){
6. 获得该推特提到的所有人
7. SocialNetwork.put(改tweet的作者,该推特提到的所有人);
8. **for** (String author: 该推特提到的所有人){
9. **if**(不是自己提到自己){
10. SocialNetwork.get(tweet作者.add(author);
11. }
12. }
13. }
14. **return** SocialNetwork;
15. }

第二个方法是 **public** **static** List<String> influencers(Map<String, Set<String>> followsGraph) 功能是返回一个影响力表，从高到低，这里就是被关注的人越多，影响力越大。

首先完成测试，包括影响力列表含有0人、1人、3人。

伪代码：

1. **public** **static** List<String> influencers(Map<String, Set<String>> followsGraph) {
2. //      throw new RuntimeException("not implemented");
3. List<String> List = **new** ArrayList<String>();
4. 建立一个map，映射每个人，和他的影响力指数（这里指被关注的次数）
5. **int** influenceCount;
6. **for** (每个人){
7. **for**(他关注的人中){
8. **if**(列表里有这个人)){
9. 影响力指数加一
10. }
11. **else**{
12. map加入这个人，影响力为1
13. }
14. }
15. }
17. //sort the influencers and put them in InfluenceList
18. **while**(影响力不为空){
19. 根据每个人的影响力指数，排序建立InfluenceList
20. }
21. **return** List
22. }

### Problem 4: Get smarter

这里我选择增加一个evidence: **Awareness**. If A follows B and B follows C, and B retweets a tweet made by C, then A sees the retweet and is influenced by C. 即如果A follow B ,B follow C ,B有转载过 C的推特，那么C也看到A的推特，A的影响力也会加一。那样AC也存在关系网。

测试是在上一个的基础之上增加有包含“RT @ “的推特。

伪代码：

1. **public** **static** Map<String, Set<String>> guessFollowsGraphplus(List<Tweet> tweets) {
2. //     throw new RuntimeException("not implemented");
3. Map<String, Set<String>> SocialNetwork = **new** HashMap<String, Set<String>>();
4. List<Tweet> socialtweet = **new** ArrayList<Tweet>(tweets);
5. 同guessFollowsGraphp部分；
6. Pattern p = Pattern.compile("rt\\s@[A-Za-z0-9\_-]+\\b");//匹配" @RTXXXX"
7. **for** (每个人) {
8. String text = tweetx.getText().toLowerCase();
9. Matcher m = p.matcher(text);
10. Set<String> RTUsers = Extract.getMentionedUsers(Arrays.asList(tweetx));
12. **while** (m.find()) {
13. 提取出这个人RT的人
14. }
15. 关注这个人的人，如果没有关注他RT的人，那么他和这个RT的人也增加关系。
17. }
18. **return** SocialNetwork;
19. }

# 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 | 编写问题1的isLegalMagicSquare函数并进行测试 | 按计划完成 |
|  |  |  | 延期1小时完成 |
|  |  |  | 遇到困难，未完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
|  |  |
|  |  |
|  |  |

# 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

## 针对以下方面的感受

1. Java编程语言是否对你的口味？

我本来没有学过Java，这次做实验的时候基本就是边学边做，需要什么就去查什么的用法，觉得就像是用胶水粘程序一样的感觉，不够流畅那种。但是马上就喜欢上了Java这种有内在哲学的语言。

1. 关于Eclipse IDE

学习它的使用真的用了很长时间,功能超多，但是很喜欢。

1. 关于Git

第一次使用版本控制系统，发现真的很方便。

1. 关于CMU和MIT的作业
2. 关于本实验的工作量、难度、deadline
3. 关于初接触“软件构造”课程

以上三个问题，一个回答：

Hard, difficult, tough, but interesting .