



哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2019 年春季学期

计算机学院大二软件构造课程

Lab 1 实验报告

姓名	张景润
学号	1172510217
班号	1703002
电子邮件	2584363094@qq.com
手机号码	18530272728

目录

1 实验目标概述	1
2 实验环境配置	1
3 实验过程	2
3.1 Magic Squares	2
3.1.1 isLegalMagicSquare()	3
3.1.2 generateMagicSquare()	4
3.2 Turtle Graphics	6
3.2.1 Problem 1: Clone and import	7
3.2.2 Problem 3: Turtle graphics and drawSquare	7
3.2.3 Problem 5: Drawing polygons	8
3.2.4 Problem 6: Calculating headings	8
3.2.5 Problem 7: Personal art	8
3.2.6 Problem 8:convexHull	9
3.2.7 Submitting	10
3.3 Social Network	10
3.3.1 设计/实现 FriendshipGraph 类	11
3.3.2 设计/实现 Person 类	12
3.3.3 设计/实现客户端代码 main()	12
3.3.4 设计/实现测试用例	13
3.3.5 实验指导文档中的问题解决	14
3.4 Tweet Tweet (选作, 额外记分)	15
3.4.1 设计/实现 Extract 类	17
3.4.2 设计/实现 Filter 类	18
3.4.3 设计/实现 SocialNetWork 类	19
3.4.4 设计/实现 MySocialNetWork 类	20
3.4.5 设计/实现测试用例	20
3.4.6 设计实现 MySocialNetWork 测试用例	22
4 实验进度记录	23
5 实验过程中遇到的困难与解决途径	24

6 实验过程中收获的经验、教训、感想	24
--------------------------	----

1 实验目标概述

本次实验通过求解四个问题,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 的测试
- 基于 Git 的代码配置管理

2 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。

- 配置 java 环境

名称	修改日期	类型	大小
bin	2019/2/25 21:26	文件夹	
conf	2019/2/25 21:26	文件夹	
include	2019/2/25 21:26	文件夹	
jmods	2019/2/25 21:26	文件夹	
legal	2019/2/25 21:26	文件夹	
lib	2019/2/25 21:27	文件夹	
COPYRIGHT	2019/1/17 21:41	文件	4 KB
README	2019/2/25 21:27	Chrome HTML D...	1 KB
release	2019/2/25 21:27	文件	2 KB

- 配置 git 环境

遇到困难: 下载 git 总是失败

解决方法: 从同学那里拷贝

遇到困难: git 小白, 不会使用

解决困难: 网上百度学习, 以及实验室网站学习, 网上有廖雪峰的讲解。

名称	修改日期	类型	大小
bin	2019/3/1 15:22	文件夹	
cmd	2019/3/1 15:22	文件夹	
dev	2019/3/1 15:22	文件夹	
etc	2019/3/1 15:22	文件夹	
mingw64	2019/3/1 15:22	文件夹	
tmp	2019/3/1 15:22	文件夹	
usr	2019/3/1 15:22	文件夹	
git-bash	2019/2/26 19:48	应用程序	147 KB
git-cmd	2019/2/26 19:48	应用程序	146 KB
LICENSE	2018/3/12 17:58	文本文档	19 KB
ReleaseNotes	2019/2/26 20:10	Chrome HTML D...	142 KB
unins000	2019/3/1 15:22	DAT 文件	947 KB
unins000	2019/3/1 15:21	应用程序	1,267 KB
unins000	2019/3/1 15:22	Outlook 项目	23 KB

- 安装 eclipse

eclipse	2019/3/17 17:52	文件夹
Git	2019/3/1 15:22	文件夹

遇到困难: eclipse 首次使用, 上手生疏

解决办法: 熟能生巧, 经过实验 1 的联系, 对该软件已经初步熟悉

GitHub Lab1 仓库的 URL 地址 (Lab1-1172510217)

<https://github.com/ComputerScienceHIT/Lab1-1172510217>

3 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 (但千万不要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。

3.1 Magic Squares

- 1, 文本文件 IO 操作。比如按行读取文本内容, 将特定的内容写入文本。
- 2, 字符串操作。比如字符串按特定字符 (\t) 分割, 将字符转换为数值。
- 3, 矩阵数组数据提取与处理。比如矩阵每行、列元素相加, 对角线元素相加。
- 4, 产生幻方。分析产生方法, 处理非法输入情况。
- 5, 验证结果是 1 和 2 文件符合幻方要求, 3、4 和 5 不是; 同时我产生的 13 阶幻方是符合要求的。如截图 3.1-a 和 3.1-b

```
<terminated> MagicSquare (1) [Java Application] D:\java\bin\javaw.exe (2019年3月14日 下午3:32:18)
true
true
Illegal MagicSquare      false
Illegal MagicSquare      false
Illegal MagicSquare      false
Please input an odd number to generate a MagicSquare:13
true
```

图 3.1-a

TurtleSoup.java	MagicSquare.java	6.txt
1 93	108 123 138 153 168 1	16 31 46 61 76 91
2 107	122 137 152 167 13	15 30 45 60 75 90 92
3 121	136 151 166 12 14	29 44 59 74 89 104 106
4 135	150 165 11 26 28	43 58 73 88 103 105 120
5 149	164 10 25 27 42	57 72 87 102 117 119 134
6 163	9 24 39 41 56	71 86 101 116 118 133 148
7 8	23 38 40 55 70	85 100 115 130 132 147 162
8 22	37 52 54 69 84	99 114 129 131 146 161 7
9 36	51 53 68 83 98	113 128 143 145 160 6 21
10 50	65 67 82 97 112	127 142 144 159 5 20 35
11 64	66 81 96 111 126	141 156 158 4 19 34 49
12 78	80 95 110 125 140	155 157 3 18 33 48 63
13 79	94 109 124 139 154	169 2 17 32 47 62 77

图 3.1-b

3.1.1 isLegalMagicSquare()

我们要实现的逻辑无非是：从文本得到数据，判断是否是正整数，判断行之和与列之和与两条对角线之和为同一个常数。

1, 读取文件内容。

- 利用 IO 流处理 6 个文本文件，将文件名字符串保存到文件名字符串数组中，如图 a。
- 利用 FileReader 和 BufferedReader 函数，配合 readline 函数将文本字符串读取到字符串 line 中，如图 b。
- 利用 split 函数根据\t 字符串分割每一行文本，将得到的文本临时储存。

```
String []filename = new String [6];
filename[0]= "src/P1/txt/1.txt";
filename[1]= "src/P1/txt/2.txt";
filename[2]= "src/P1/txt/3.txt";
filename[3]= "src/P1/txt/4.txt";
filename[4]= "src/P1/txt/5.txt";
filename[5]= "src/P1/txt/6.txt";
```

图 3.1.1-a

```

FileReader fr1 = new FileReader(filename);
BufferedReader bf1 = new BufferedReader(fr1);
String line = null;
int [][] a = new int [row][row];
int k = 0;
while((line = bf1.readLine()) != null) {

```

图 3.1.1-b

2, 处理得到的文本。

- a) 利用 isDigit 函数循环判断得到的分割文本是否是纯数字。若不是纯数字(比如有小数点或者负号), 则返回 false。
- b) 利用 Integer.valueOf 函数将得到的文本储存到 int 数组中。同时判断是否有 0 的情况, 若是 0, 则返回 false; 否则继续进行。

3, 计算各行之和, 各列之和, 对角线之和, 并进行判断。

- a) 两层 for 循环计算各行之和, 各列之和, 对角线之和。为了便于比较, 将这些待比较的数据储存到两个 int 数组和两个 int 变量里。如图 c。
- b) 判断这些数据是否都等于一个数据(我们可以取这个数据是第一行之和), 一旦判断的过程中遇到了不相等的情况, 则直接 return false。

```

for(int i = 0; i < row; i++) {
    for(int j = 0; j < row; j++) {
        rows[i] += a[i][j];
        columns[j] += a[i][j];
    }
    diagonal1 += a[i][i];
    diagonal2 += a[i][row-1];
}

```

图 3.1.1-c

3.1.2 generateMagicSquare()

我们要实现的逻辑是: 将产生的数据输出到文件里, 判断输入的 n 是否为正奇数, 若为偶数, 输出提示信息并结束; 若是负数, 输出提示信息并结束。并了解相关异常的含义及产生原因

1, 输出得到的幻方数据到文本中。

- a) 改写原代码内容, 利用 bw.write 函数, 将得到的储存在数组中的正整数输出到 6.txt。同时注意输出的时候要将数据之间用 \t 分隔开, 因为我们要判断 6.txt 是否满足幻方格式需要注意分割 \t。如图 3.1.2-a。
- b) 利用 if 语句对非法输入进行筛选。若为偶数或者负数, 直接输出非法, 并结束。

2, 判断生成的幻方是否符合要求。

- a) 在 main 函数里调用 isLegalMagicSquare() 函数, 解析生成的 6.txt 文件是否符合要求, 根据题目开始时我的截图可知, 是符合要求的。

```
String filename = "src/P1/txt/6.txt";
FileWriter fw = new FileWriter(filename);
BufferedWriter bw = new BufferedWriter(fw);
for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
        bw.write(magic[i][j] + "\t");
    }
    if(i < n-1) {
        bw.write("\n");
    }
}
bw.close();
```

图 3.1.2-a

3, 异常含义及产生原因。

- a) 若输入的 n 为偶数。产生的异常是数组下标越界问题：
java.lang.ArrayIndexOutOfBoundsException。如图 3.1.2-b。
因为考虑一种情况，若 $row == 0$ 时， row 会变为 $n - 1$ ；此时有可能在进行循环时， i 正好整除 n ，则 $row++$ 会使 $magic$ 数组下标越界。现在分析这种情况只有偶数可以发生异常。首先 row 的变化只有三种情况： i 整除 n 、 i 不整除 n 且 row 等于 0 和 i 不整除 n 且 row 不等于 0。在偶数 n 的情况下， row 有可能在越界的。
- b) 若输入的 n 为负数。产生的异常是数组大小为负数的异常：
java.lang.NegativeArraySizeException。如图 3.1.2-c。
因为在函数内部根据 n 申请了一个 int 二维数组，若 n 为负数，数组申请报错。

```
Please input an odd number to generate a MagicSquare:10
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 10
at P1.MagicSquare.generateMagicSquare(MagicSquare.java:88)
at P1.MagicSquare.main(MagicSquare.java:131)
```

图 3.1.2-b

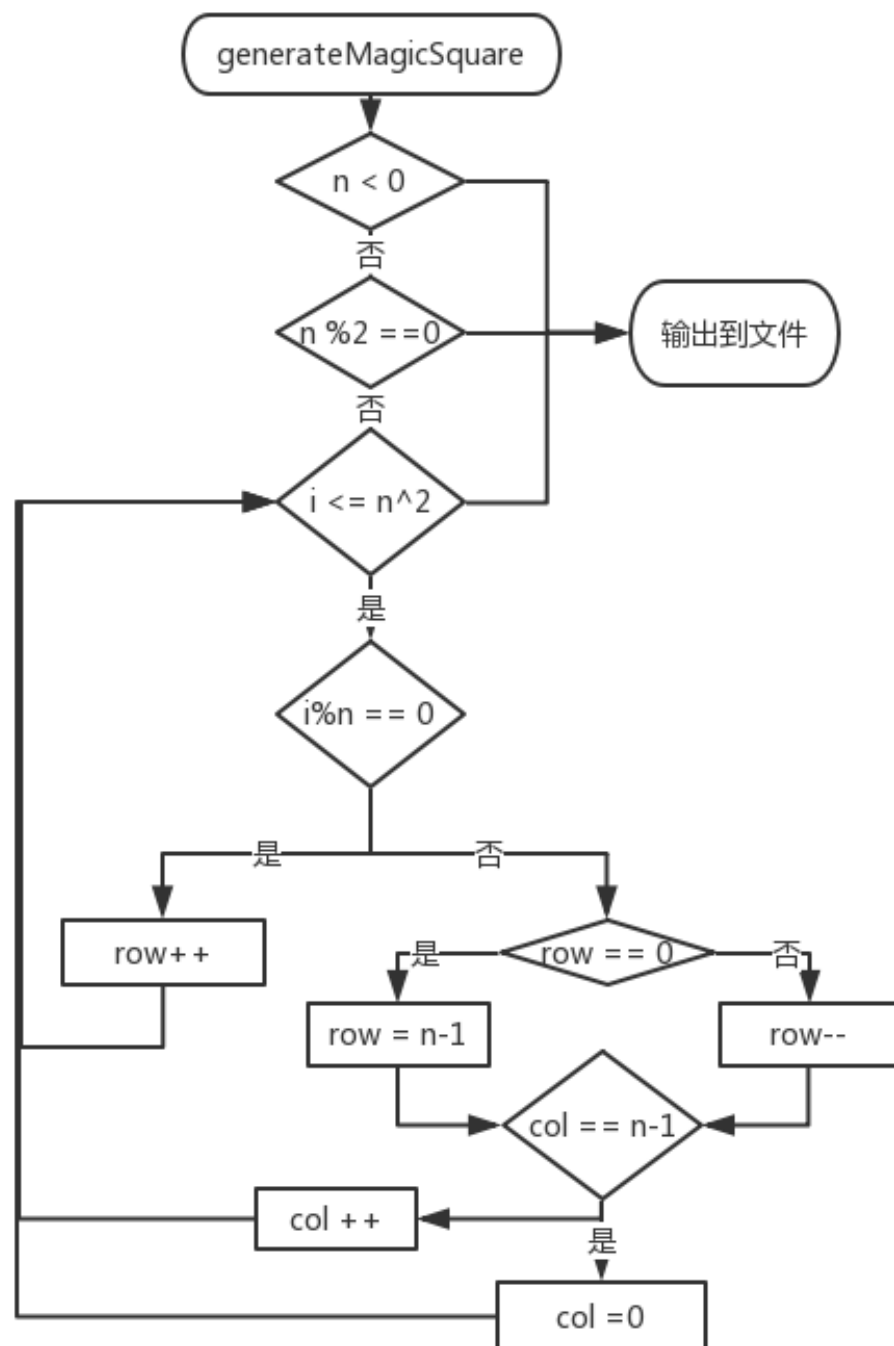
```
Please input an odd number to generate a MagicSquare:-3
Exception in thread "main" java.lang.NegativeArraySizeException: -3
at P1.MagicSquare.generateMagicSquare(MagicSquare.java:85)
at P1.MagicSquare.main(MagicSquare.java:131)
```

图 3.1.2-c

4, 说明产生幻方的算法步骤（罗伯法）以下是助记口诀

- a) 1 居上行正中央——数字 1 放在首行最中间的格子中
 - b) 依次斜填切莫忘——向右上角斜行依次填入数字
 - c) 上出框界往下写——如果右上方向出了上边界，就以出框后的虚拟方格位置为基准，将数字竖直降落至底行对应的格子中
 - d) 右出框时左边放——同上，向右出了边界，就以出框后的虚拟方格位置为基准，将数字平移至最左列对应的格子中
 - e) 重复便在下格填——如果数字 $\{N\}$ 右上的格子已被其它数字占领，就将 $\{N+1\}$ 填写在 $\{N\}$ 下面的格子中
- 右上重复一个样——如果朝右上角出界，和“重复”的情况做同样处理。

5, 附上该函数的程序框图



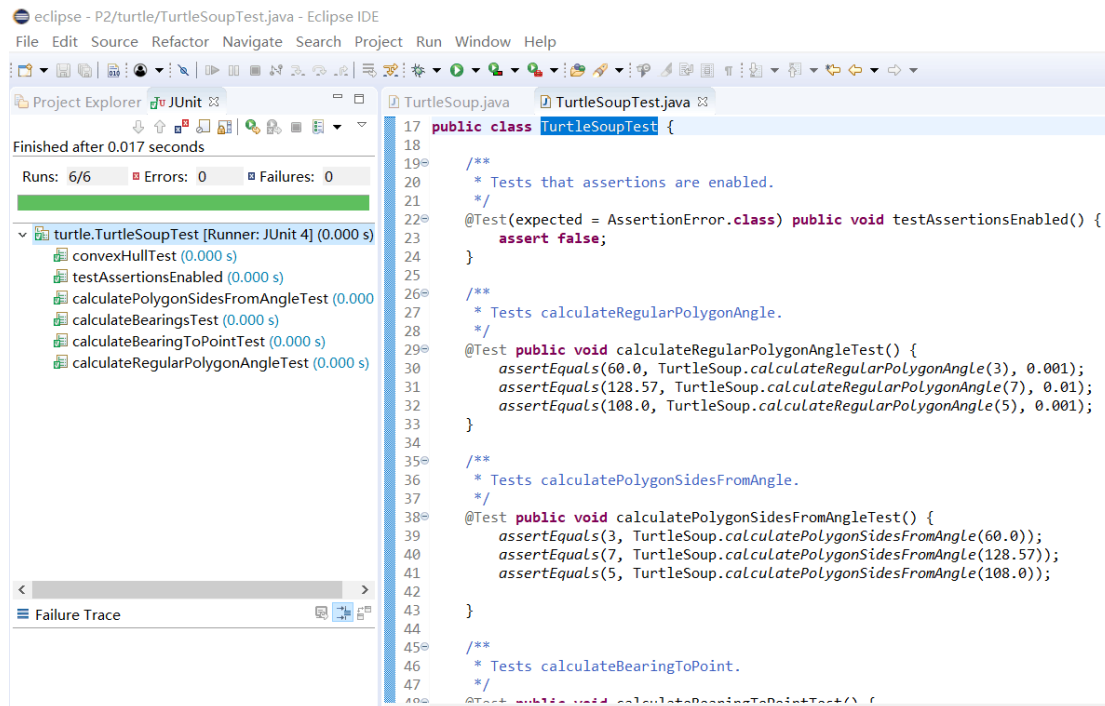
3.2 Turtle Graphics

在这里简要概述你对该任务的理解。

本问题的主要考察方向是正多边形角度与边数的关系，以及循环寻找最小偏

转角。需要注意的是: `int` 与 `double` 数据类型的变换容易出现问題, 要多加注意, 合适的转换类型, 以及多边形内角和外角的关系。

附上实验成功截图。



3.2.1 Problem 1: Clone and import

如何从 GitHub 获取该任务的代码、在本地创建 git 仓库、使用 git 管理本地开发。

1, 从 GitHub 网页上下载源代码, 网址是 https://github.com/rainywang/Spring2019_HITCS_SC_Lab1/tree/master/P2

2, 在本地创建 git 仓库, 使用 git 管理本地开发。

初始化本地仓库: `git init`; 添加远程库源: `git add remote origin`; 在远程仓库创建 master 分支; 将远程仓库同步到本地: `git pull origin master`; 将本地文件加入本地仓库, 将本地仓库同步到远程仓库: `git add->git commit->git push`。

3.2.2 Problem 3: Turtle graphics and drawSquare

问题思路: 画正四边形 (知道边长即可), 可循环四次: 前进边长个单位长度, 并旋转 90 度。

- 1, 先调用函数 `forward(int a)`, 前进 a 距离
- 2, 调用函数 `turn(double angle)`, 旋转 90 度
- 3, 重复上述过程 4 次, 至产生闭合曲线即可 (旋转 90 度也可只做 3 次)

3.2.3 Problem 5: Drawing polygons

问题思路: 根据正多边形外角和内角的关系: 外角+内角=180 度, 且外角和=360 度; 边数代表循环次数, 外角代表偏转角度, 边长代表前进距离, 即可做出普通正多边形。

- 1, 根据边数和公式外角和等于 360° 得到外角大小: $360.0/\text{边数}$ (注意转换为 double 类型)
- 2, 根据外角+内角=180°, 得到内角大小: $180-\text{外角}$
- 3, 边数代表循环次数, 边长代表前进距离, 外角代表偏转角度。

3.2.4 Problem 6: Calculating headings

问题思路: 利用起点和终点的坐标计算出相对于 y 轴的角度, 然后计算相对于原来角度的偏转即可。但需要考虑特殊情况: 起点和终点重合, 起点和终点 x 相等。

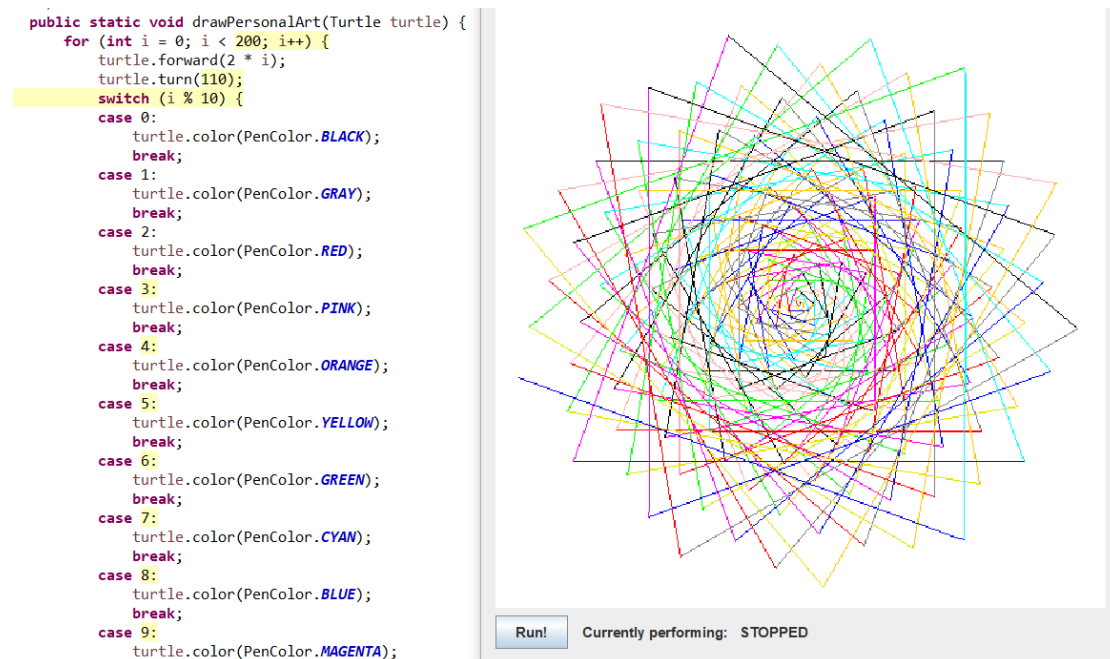
- 1, 若起点和终点 x 重合, 若终点 y 大, 则计算相对 y 轴角度为 0 度; 若终点 y 小或者等于, 则为 180 度, 在进行第三步。
- 2, 若 x 点不相等。运用函数 `Math.atan` 将相对位置转化为弧度, 然后转化为角度。如下图

```
    } else {  
        angle = Math.atan((double) (targetY - currentY) / (double) (targetX - currentX));  
        angle = Math.toDegrees(angle);  
        if (targetX < currentX) {  
            angle -= 180;  
        }  
        if (angle <= 90) {  
            angle = 90 - angle;  
        } else {  
            angle = 450 - angle;  
        }  
    }
```

- 3, 减去起始角度, 经过分析处理可得偏转角度。
- 4, 对于计算多个坐标的偏转角, 只需要将上一次的偏转角作为本次的起始角度, 然后调用以上过程即可。尤其要注意第一次取起始角为 0 度; 同时注意类的运用。

3.2.5 Problem 7: Personal art

问题思路: 要画出好看的图形并不难, 只需要利用 for 循环配合函数 `forward` 和函数 `turn` 即可。我画的截图如下。同时要想有规律的改变颜色可以使用 `switch` 语句, 通过循环增量 `i` 的变化可以利用 `i%10` 得到颜色变化。



3.2.6 Problem 8:convexHull

问题思路：主要是依次选取相对于一个点偏转角最小的点，直到形成一个闭合的曲线。这里为了方便选取最左下角的点为第一个点（也可左上角，右上角，右下角），同时利用函数 `calculateBearingToPoint` 函数计算偏转角。

- 1, 遍历寻找最左下角的点作为起始点。
- 2, 循环依次寻找下一个偏转角最小的点，利用函数 `calculateBearingToPoint` 计算偏转角。值得注意的是，我们的起始角度对于我们的所有待选点结果偏转角度的影响是一致的，因此我们可以统一取起始角为 0 度，降低了计算量。如截图

```

for (int i = 0; i < count; i++) {
    if ((int)pointsSet.get(i).x() < xleft) { // 判断x值的大小，找到较小的x及其所在的位置
        xleft = pointsSet.get(i).x();
        index = i;
        continue;
    }
    if ((int)pointsSet.get(i).x() == xleft) { // 若该点与我们选的x值相等，则判断y值大小
        if ((int)pointsSet.get(i).y() < pointsSet.get(index).y()) {
            index = i;
        }
    }
}
}

```

```
for (int i = 0; i < count; i++) { // 寻找一个起始的计算点
    if (!flags[i] || i == index) {
        angle1 = calculateBearingToPoint(0, (int) pointsSet.get(indexfinal).x(),
            (int) pointsSet.get(indexfinal).y(), (int) pointsSet.get(i).x(), (int) pointsSet.get(i).y());
        k = i;
        break;
    }
}
for (int i = 0; i < count; i++) {
    if (!flags[i] || i == index) {
        angle2 = calculateBearingToPoint(0, (int) pointsSet.get(indexfinal).x(),
            (int) pointsSet.get(indexfinal).y(), (int) pointsSet.get(i).x(), (int) pointsSet.get(i).y());
        if (angle2 < angle1) {
            angle1 = angle2;
            k = i;
            continue;
        }
        if (angle2 == angle1) { // 若有两个点偏转角度一样, 则选择距离计算目前点较远的点
            if (Math.abs(pointsSet.get(indexfinal).x() - pointsSet.get(i).x()) > Math
                .abs(pointsSet.get(k).x() - pointsSet.get(indexfinal).x())) {
                k = i;
            }
        }
    }
}
```

- 3, 循环的出口是起始点和终点是同一个点。
- 4, 在此函数内还设置了一个 Boolean 数组, 用于标记一个点是否已经被选取过。但是与此同时我们要注意起始点虽然被标记过, 但是还要参与比较, 因为否则 while 循环为死循环。(这里自然用到了前面提到的两点重合角度计算函数设计的呼应)

3.2.7 Submitting

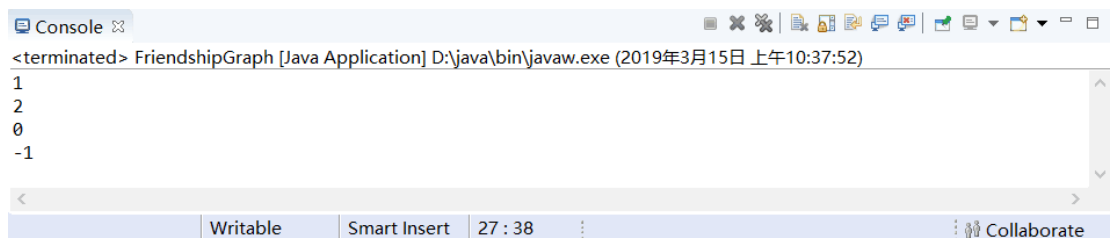
如何通过 Git 提交当前版本到 GitHub 上你的 Lab1 仓库。

运用指令 `git push` 即可提交

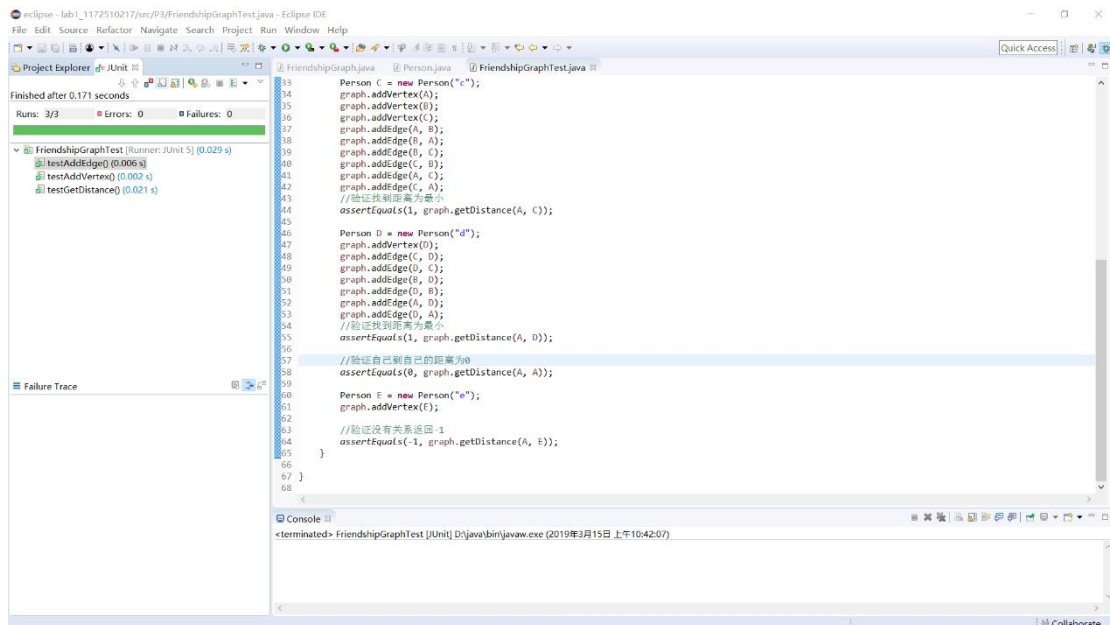
3.3 Social Network

主要是实现各种类的属性, 比如 `person` 类中有姓名、与之有关系的人等等。主要实现的算法是 `getDistance`, 用于实现两人有关系时, 返回关系距离, 无关系时返回 -1, 自己时返回 0。主要运用广度优先算法即可。

本题 main 函数以及 test 截图如下。



```
Console
<terminated> FriendshipGraph [Java Application] D:\java\bin\javaw.exe (2019年3月15日 上午10:37:52)
1
2
0
-1
```



3.3.1 设计/实现 FriendshipGraph 类

给出你的设计和实现思路/过程/结果。

- 1, 实现 `addVertex(Person person)`。首先在类中引入 `Allperson` 用于储存所有的 `Person`，在这里判断是否已经重复加入，若重复加入，则不再加入该 `person`。
构造映射 `Map<String, Person> nameCheck`，用于判断该人的名字是否已经在 `map` 中存在，若存在，则报错返回；若不存在，则加入 `map` 中。
- 2, 实现 `addEdge(Person person1, Person perosn2)`。调用 `Person` 类中的 `addRelation` 函数实现该方法。
- 3, 实现 `getDistance(Person person1, Person person2)`。
 - a) 特殊情况判断：两者是同一个人，则返回 0。
 - b) 构造两个映射 `map: Map<Person, Boolean> visited` 和 `Map<Person, Integer> dis` 分别用于记录 `person` 是否被访问过，以及 `person1` 到 `person` 的距离。
 - c) 构造队列：`Queue<Person> queue`，用于广度优先搜索遍历。
 - d) 两层循环：第一层用于控制队列非空时终止，终止意味着无关系；第二层用于控制遍历与某 `person` 相连的所有人。
 - e) 循环增量或循环内部条件控制：若未找到，则 `map` 的 `dis` 自加加；若找到，则返回 `dis` 中的距离。如图。

```

while (!queue.isEmpty()) { // 循环直到队列为空
    Person head = queue.poll(); // 得到队首元素，并将其出队
    Person temp = head.getRelation().peek(); // 得到与其有关的第一个人
    int i = 0;
    while (temp != null) { // 循环直到无人与head有直接关系
        if (!visited.get(temp)) { // 若temp未被访问
            if (temp.equals(person2)) { // 若找到person2
                return dis.get(head) + 1;
            } else { // 若未找到
                visited.put(temp, true);
                dis.put(temp, dis.get(head) + 1);
                queue.add(temp); // 将当前person入队
            }
        }
        if (++i < head.getRelation().size()) { // 继续寻找与其有关系的人
            temp = head.getRelation().get(i);
        } else {
            break;
        }
    }
}
}

```

3.3.2 设计/实现 Person 类

给出你的设计和实现思路/过程/结果。

该类要实现 person 的基本属性：姓名+关系

- 1, 类中设置一个链表结构 `LinkedList<Person> connect` 用于储存与该 person 有关系的人。
- 2, 实现代码的安全性: 使用 `getName` 函数获取人名; 使用 `getRelation` 函数获取关系。
- 3, 实现方法 `addRelation`: 在链表 `connect` 中加入 person 关系。如下图。

```

public void addRelation(Person person) {
    this.connect.add(person); // 得到person关系
}

```

3.3.3 设计/实现客户端代码 main()

给出你的设计和实现思路/过程/结果。

报告上已给代码，完全一样，截图如下。

```
public static void main(String[] args) {  
  
    FriendshipGraph graph = new FriendshipGraph();  
    Person rachel = new Person("Rachel");  
    Person ross = new Person("Rachel");  
    Person ben = new Person("Ben");  
    Person kramer = new Person("Kramer");  
    graph.addVertex(rachel);  
    graph.addVertex(ross);  
    graph.addVertex(ben);  
    graph.addVertex(kramer);  
    graph.addEdge(rachel, ross);  
    graph.addEdge(ross, rachel);  
    graph.addEdge(ross, ben);  
    graph.addEdge(ben, ross);  
    System.out.println(graph.getDistance(rachel, ross));  
    // should print 1  
    System.out.println(graph.getDistance(rachel, ben));  
    // should print 2  
    System.out.println(graph.getDistance(rachel, rachel));  
    // should print 0  
    System.out.println(graph.getDistance(rachel, kramer));  
    // should print -1  
}
```

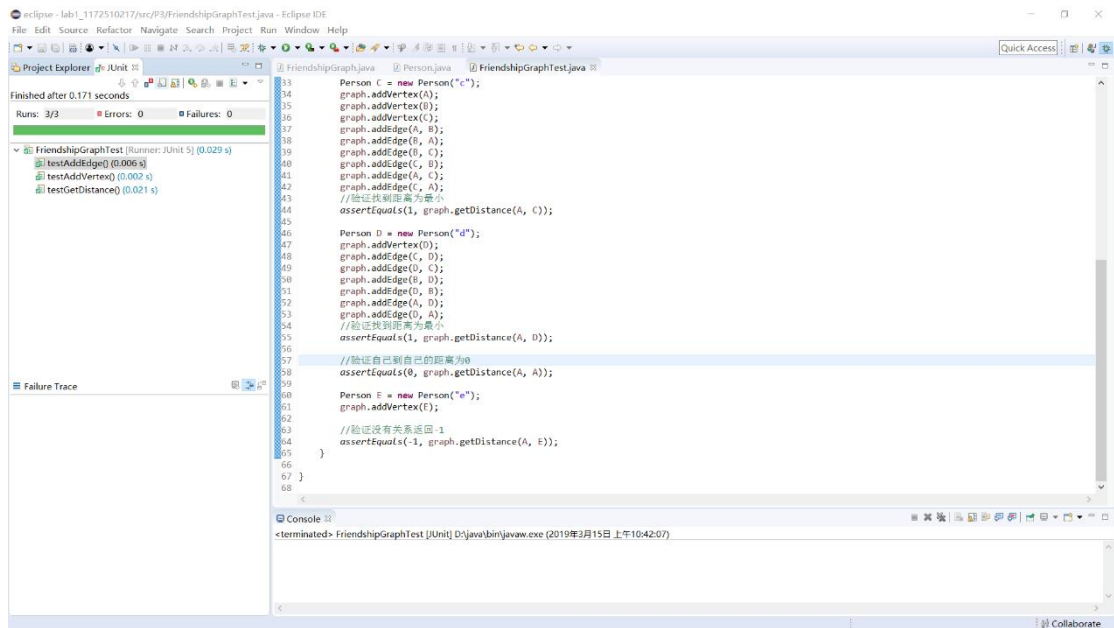
3.3.4 设计/实现测试用例

给出你的设计和实现思路/过程/结果。

设计思路：首先要判断特殊情况：自己与自己的距离为 0，没有关系的人距离为-1。再设计一般情况测试类。最后要注意测试选取距离是否为最小。

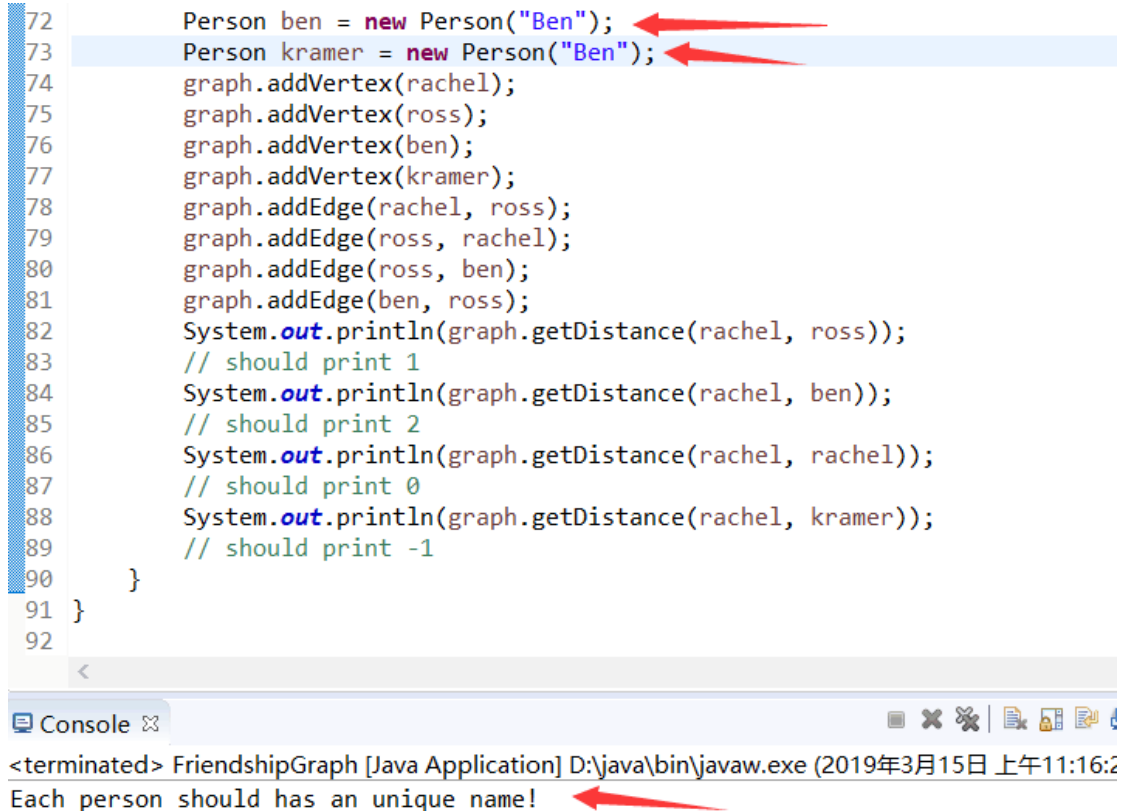
- 1, 测试自己与自己。传入参数为 A 和 A，若返回结果==0，则 asserttrue 成立；
- 2, 测试没有关系的人。传入参数为 A 和 E，其中 E 和任何人都无关系，若返回结果==-1，则 asserttrue 成立；
- 3, 测试一般情况。构造几个关系网，测试一般的情况即可。
- 4, 测试距离为最小。构造几个特殊的关系网，其中 A 与所有人都有关系，而其他人与部分人有关系，若 A 与其他人的距离为 1 则 asserttrue 成立。

附上实验 test 截图。



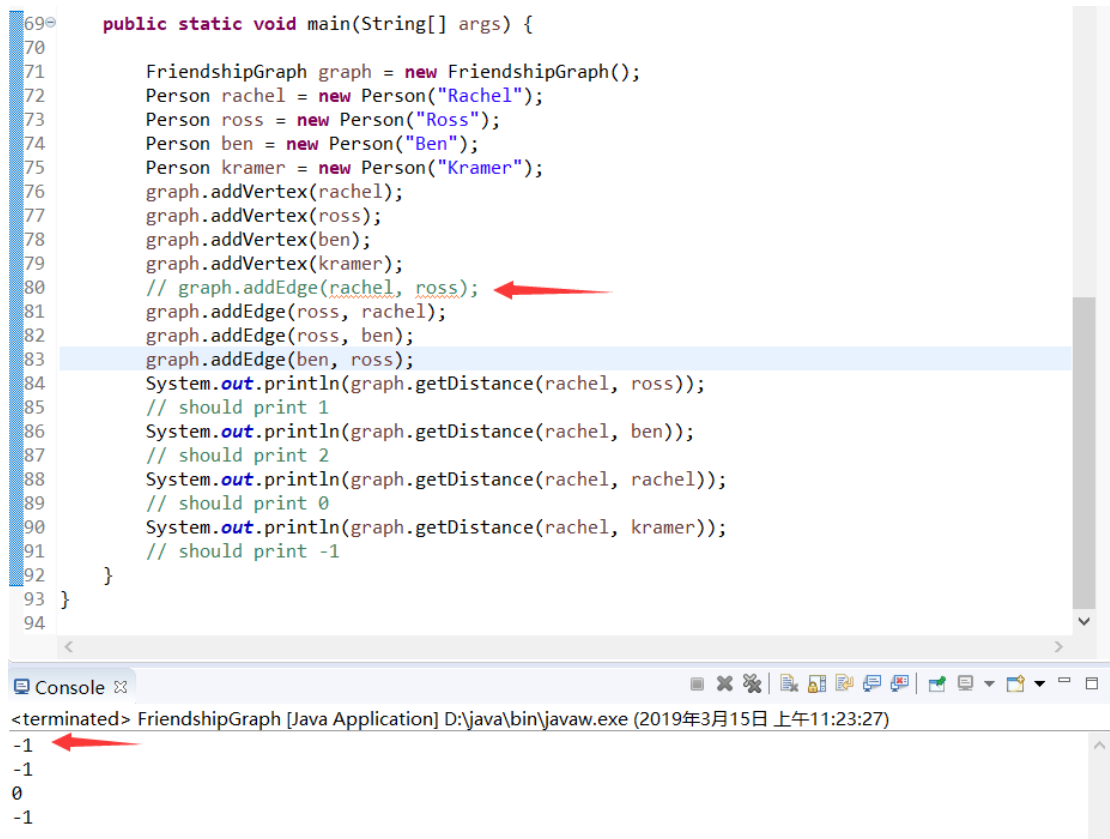
3.3.5 实验指导文档中的问题解决

- 1, 若加入的一个人已经存在, 则我的算法不再将该人加入。
- 2, 若破坏一人一个姓名的原则, 则程序退出并在终端输出错误。如图



- 3, 若将加关系边语句 addEdge (A, B) 和 addEdge (B, A) 删除一句, 则会当成有向图处理, 即相当于单向的 follow 关系。输出也会相应的变化, 终端变化是和预

想的结果一致的。例如若将我的代码 main 函数中将此句话注释掉: 10. graph.addEdge(rachel, ross), 我的终端关于 rachel 和 ross 的距离将会变化输出-1。如截图。



```
69 public static void main(String[] args) {
70
71     FriendshipGraph graph = new FriendshipGraph();
72     Person rachel = new Person("Rachel");
73     Person ross = new Person("Ross");
74     Person ben = new Person("Ben");
75     Person kramer = new Person("Kramer");
76     graph.addVertex(rachel);
77     graph.addVertex(ross);
78     graph.addVertex(ben);
79     graph.addVertex(kramer);
80     // graph.addEdge(rachel, ross);
81     graph.addEdge(ross, rachel);
82     graph.addEdge(ross, ben);
83     graph.addEdge(ben, ross);
84     System.out.println(graph.getDistance(rachel, ross));
85     // should print 1
86     System.out.println(graph.getDistance(rachel, ben));
87     // should print 2
88     System.out.println(graph.getDistance(rachel, rachel));
89     // should print 0
90     System.out.println(graph.getDistance(rachel, kramer));
91     // should print -1
92 }
93 }
94
```

Console

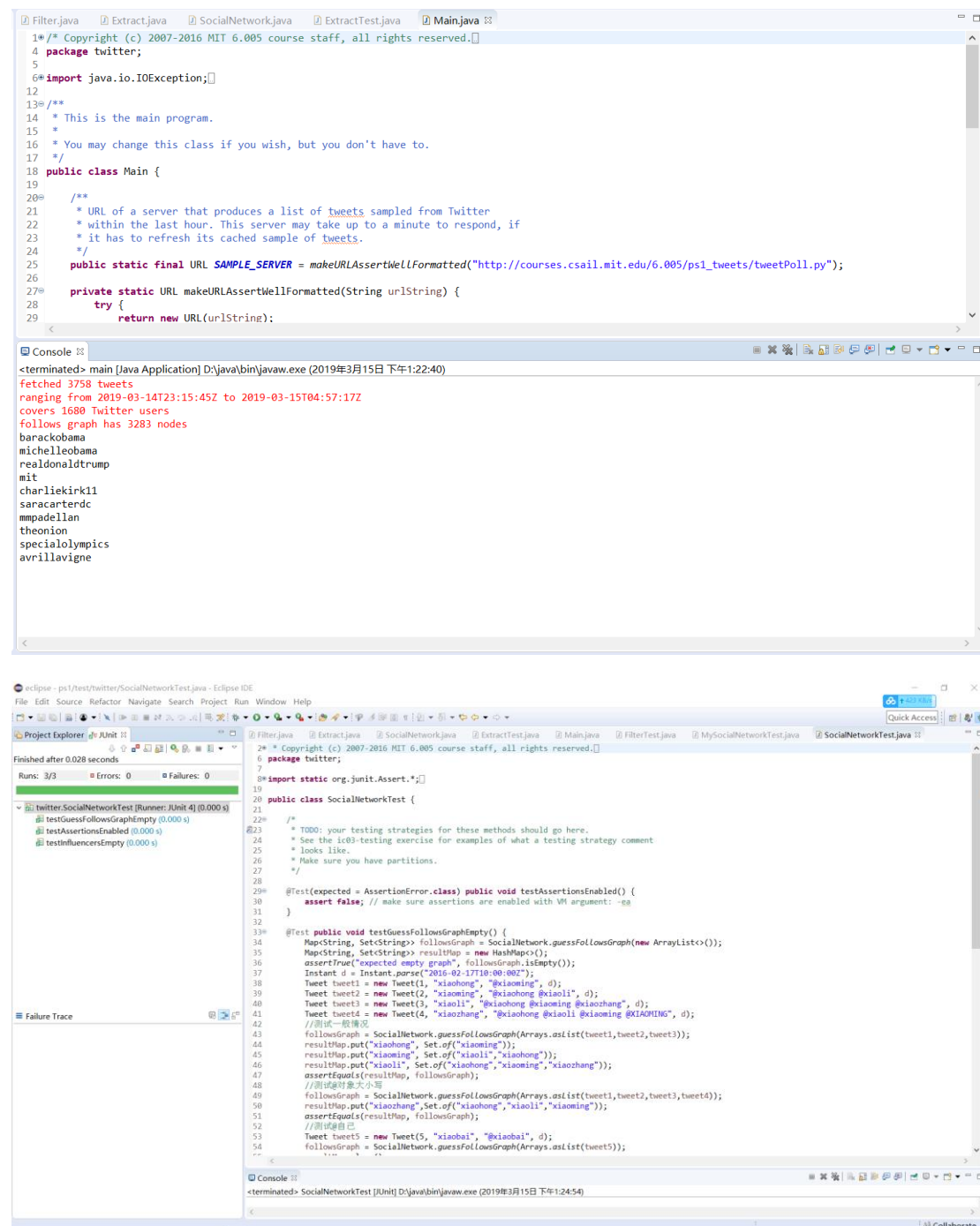
<terminated> FriendshipGraph [Java Application] D:\java\bin\javaw.exe (2019年3月15日 上午11:23:27)

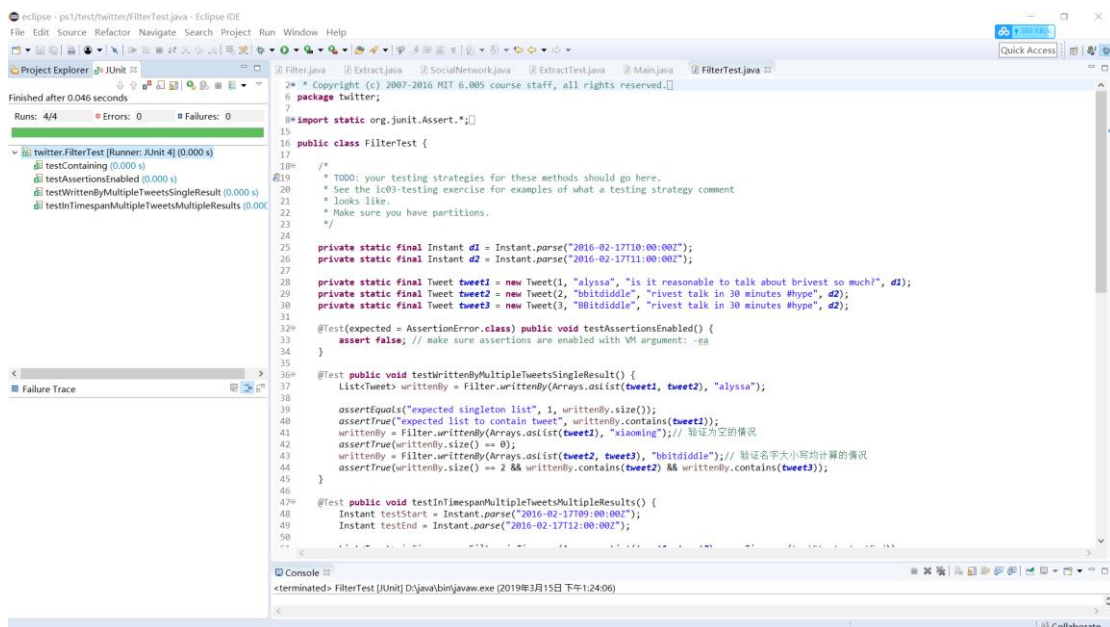
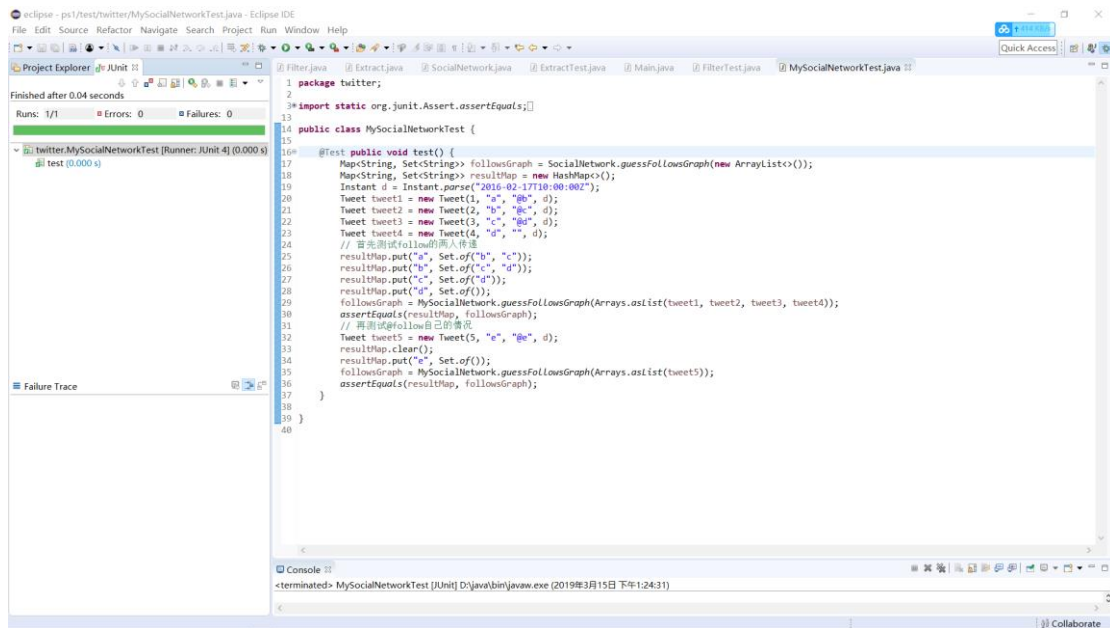
-1
-1
0
-1

3.4 Tweet Tweet (选作, 额外记分)

我的理解: Extract 类实现对象发表推文时间获取与比较, 以及推文内容按关键词提取; Filter 类实现同作者推文划分到一起, 在一个时间段的推文划分到一起, 包含关键词的推文划分在一起; socialNetWork 类实现作者与其追随的人形成映射, 同时将作者按追随者数目降序排列; mySocialNetWork 类实现三人追随关系的一般传递, 从而实现更合理的猜测相互影响关系。对于 test 类则要关注边界条件, 极限条件, 比如: 加入的推文是否按照原来的顺序排列, 是否忽略了大小写等

附上实验验证及 test 截图。





- twitter.ExtractTest [Runner: JUnit 4] (0.000 s)
 - testGetMentionedUsersNoMention (0.000 s)
 - testGetTimespanTwoTweets (0.000 s)
 - testAssertionsEnabled (0.000 s)

3.4.1 设计/实现 Extract 类

3.4.1.1 getTimespan

a) for 循环遍历寻找最早的发 tweet 的时间和最晚的发 tweet 的时间。

- b) 初始时, 令最早和最晚的时间均为第一篇 tweet 的时刻, 然后循环比较不断更新该时刻。最终将得到的最早和最晚时刻赋值给返回值即可。

3.4.1.2 getMentionedUsers

- a) 设计方法 `boolean judgeLegalChar(char ch)`。用于判断推文中的一个字符是否符合用户名要求。语句如下:

```
boolean condition1 = (ch >= 'a' && ch <= 'z');
boolean condition2 = (ch >= 'A' && ch <= 'Z');
boolean condition3 = (ch >= '0' && ch <= '9');
boolean condition4 = (ch == '_' || ch == '-');
return (condition1 || condition2 || condition3 || condition4);
```

- b) 基本逻辑是遍历每一篇推文文本, 依次寻找文本中字符@所在的位置, 然后判断: @是否是文本第一个字符或者@左边第一个字符传入 a) 中的方法是否返回真; @后的第一个元素传入 a) 中的方法是否返回真。判断语句如下。

```
if ((k == 0 || !judgeLegalChar(s.charAt(k - 1))) && judgeLegalChar(s.charAt(k + 1))
    while (judgeLegalChar(s.charAt(l))) {
        if (l == s.length() - 1) {
            l++;
            break;
        }
        l++;
    }
    userNames.add(s.substring(k + 1, l).toLowerCase());
```

- c) 经过上一步的判断处理, 只需要将合法字符串从原文本中截断即可。

3.4.2 设计/实现 Filter 类

3.4.2.1 containing

- a) 遍历所有的推文, 提取所有这些对象中的 text 字符串。
 b) 将这些字符串按照空格格式调用 `String.split` 函数分割得到所有的单词。
 c) 遍历所有分割得到的单词, 判断是否含有 words 中的单词, 若含有, 则将该推文加入 tweets 返回值中。如截图。

```
for (int i = 0; i < count; i++) {
    String[] s = tweets.get(i).getText().split(" ");
    Label: for (int j = 0; j < s.length; j++) {
        for (int k = 0; k < words.size(); k++) {
            if (s[j].equalsIgnoreCase(words.get(k))) {
                if (containing.indexOf(tweets.get(i)) == -1) {
                    containing.add(tweets.get(i));
                    break Label;
                }
            }
        }
    }
}
```

- d) 比较的同时仍要注意忽略英文字母的大小写。
 e) 比较巧妙的是在函数内部设置了一个 Label, 用于 break 跳出循环, 提高算

法效率。因为我们只要在一篇推文中找到了一个 words 中的单词，即可将该推文加入返回值中，无需进行下面的比较。

3.4.2.2 inTimespan

- 运用 for 循环遍历所有的推文，得到发推文的时刻 instant。
- 调用 Instant 类的 isAfter 和 isBefore 函数判断得到的 instant 是否在要求的区间内。若在，则加入该推文对象；若不在，则继续遍历。

3.4.2.3 writtenBy

- 运用 for 循环遍历所有的推文，得到他们的作者。
- 将每一篇推文的作者与传入的 username 进行比较，注意要忽略英文字母大小写的情况。调用 String 类中的 equalsIgnoreCase 函数即可比较。
- 注意顺序遍历，因为要求输出结果中的 tweets 要和原来输入的顺序一致。

3.4.3 设计/实现 SocialNetWork 类

3.4.3.1 guessFollowsGraph

- 建立多个数据结构类型。Set<String> allUserNames 用于保存所有的作者小写的名字；List<Tweet> writtenBy 用于保存作者写的所有推文；Set<String> followNames 用于保存作者@的所有人。
- for each 循环得到所有的作者；for each 循环得到一个作者写的所有推文，并从这些推文中提取作者@的所有对象。
- 要注意将作者自己从@群体里面删除，因为自己不可以“影响”自己。
- 如下图

```
for (Tweet temp : tweets) { // 忽略大小写得到所有的作者
    allUserNames.add(temp.getAuthor().toLowerCase());
}
for (String temp : allUserNames) {
    writtenBy = Filter.writtenBy(tweets, temp); // 得到作者写的所有推文
    followNames = Extract.getMentionedUsers(writtenBy); // 得到作者@的所有人
    followNames.remove(temp); // 将作者自己从@群体中删除
    followsGraph.put(temp, followNames);
}
```

3.4.3.2 influencers

- 设计 map 数据结构：Map<String, Integer> valueMap 用于储存所有的用户及其被关注人数。

- b) 遍历传入的 map followsGraph, 将所有的被关注者加入 valueMap 中, 若再次出现该被关注者的姓名, 则粉丝数目++, 否则将其加入 map, 同时粉丝数设为 1。
- c) 注意还有可能出现粉丝数为 0 的情况。这时要将他们加入 valueMap, 同时设置粉丝数为 0。
- d) 调用 Collections.sort 函数实现 map 的 value 降序排列并输出到 influencers 中。
- e) 如截图

```
for (String string : followsGraph.keySet()) {
    if (!valueMap.keySet().contains(string)) {
        influencers.add(string);
        valueMap.put(string, 0); // 若有人未被关注, 则直接加入
    }
}
Collections.sort(influencers, new Comparator<String>() { // 降序排列

    public int compare(String o1, String o2) {
        return valueMap.get(o2).compareTo(valueMap.get(o1));
    }
});
```

3.4.4 设计/实现 MySocialNetWork 类

- a) 此方法是上一个类方法的拓展, 描述为: 当一个人 follow 的对象同时 follow 了另外一个人, 则这个人很有可能受到另外一个人的影响 (例如: 若 A 追随 B, B 追随 C, 则 A 受 C 影响)。
- b) 在原来算法的基础上, 增加了一个 for 循环, 用于得到一个人的追随人的所有追随人。如图所示。

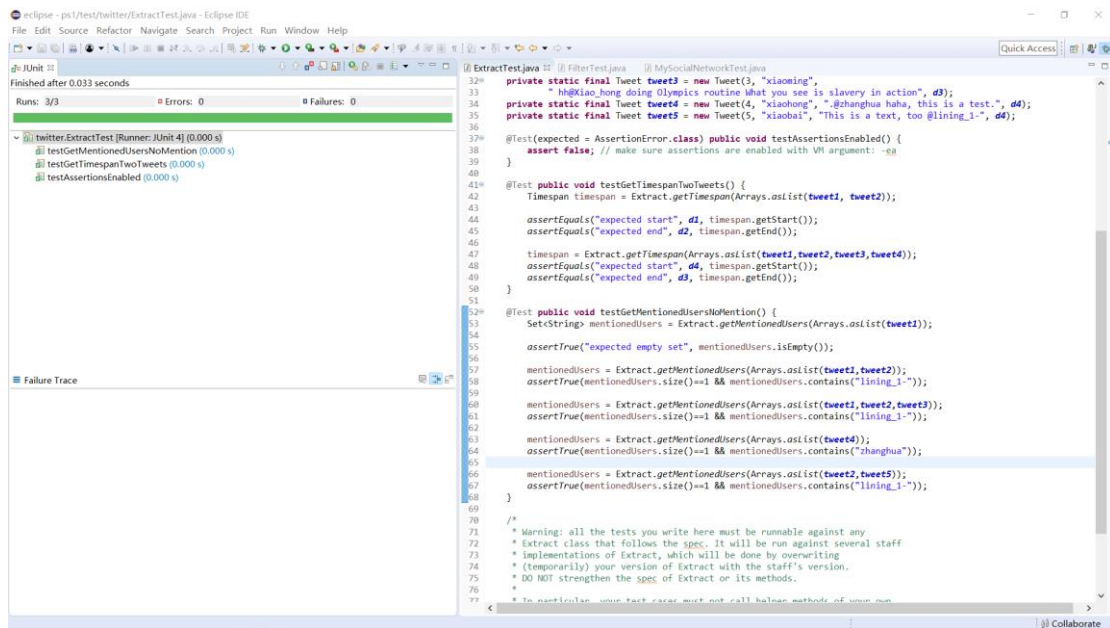
```
for (String temp : followsGraph.keySet()) {
    Set<String> mySocilSet = new HashSet<>();
    for (String follow : followsGraph.get(temp)) {
        mySocilSet = followsGraph.get(follow);
        mySocilSet.addAll(followsGraph.get(follow));
    }
    mySocilSet.remove(temp);
    followsGraph.put(temp, mySocilSet);
}
```

- c) 在算法中, 即是这样, 若 A follow B, 则将 B 的所有 follow 的人加入 A 的 follow 对象 Map 中。

3.4.5 设计/实现测试用例

3.4.5.1 ExtractTest

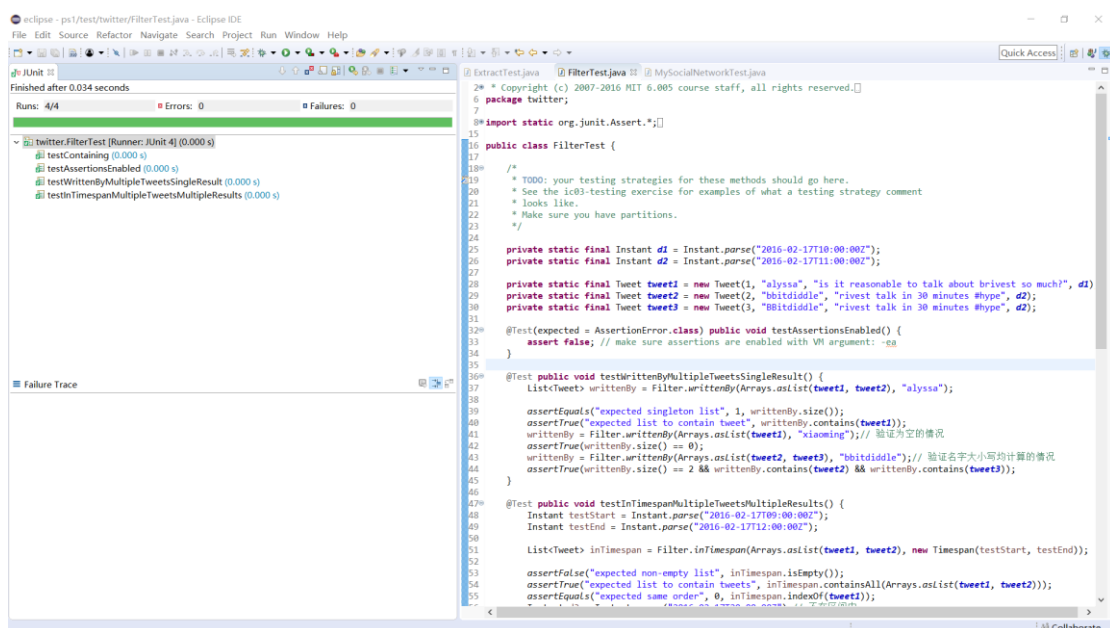
测试通过截图



- a) testGetTimespan。测试 tweets 全在范围内，和部分在范围内的情况。
- b) testGetMentionedUsers。测试@前有非法字符；测试@后无合法字符；测试大小写均有的情况；测试@在字段最前面；测试@后的用户名结尾同时是文本结尾。

3.4.5.2 FilterTest

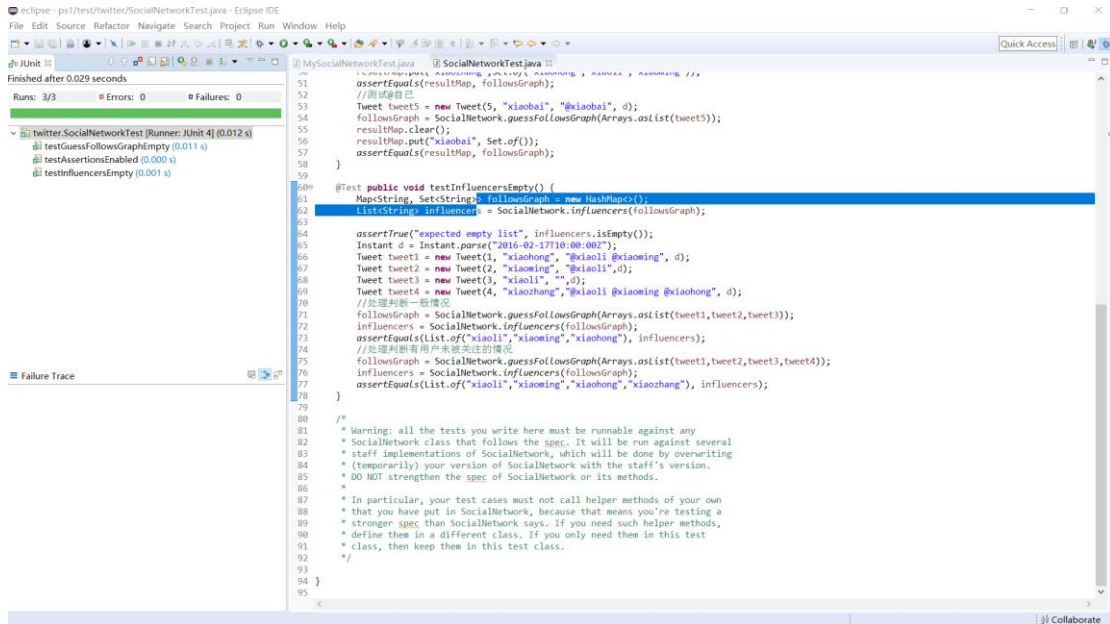
测试通过截图



- a) testWrittenBy。测试作者没写文章；测试名字大小写不区分。
- b) testInTimespan。测试结果顺序与输入顺序一致；测试部分时刻不在范围内。
- c) testContaining。测试作者不区分大小写；测试单词必须被空格隔开；测试是否重复加入某 tweet；测试多个 tweet 作为输入。

3.4.5.3 SocialNetworkTest

附 test 通过截图



- a) testGuessFollowsGraph。测试多个输入对象情况；测试@对象大小写不区分；测试@自己要去除的情况；
- b) testInfluencers。处理一般情况下多个对象作为输入；测试有用户未被关注的情况；还要测试输出顺序确实是按照降序排列的。

3.4.6 设计实现 MySocialNetWork 测试用例

附上测试通过截图

```
1 package twitter;
2
3 import static org.junit.Assert.assertEquals;
4
5 public class MySocialNetworkTest {
6
7     @Test public void test() {
8         Map<String, Set<String>> followsGraph = SocialNetwork.guessFollowsGraph(new ArrayList<>());
9         Map<String, Set<String>> resultMap = new HashMap<>();
10         Instant d = Instant.parse("2016-02-17T10:00:00Z");
11         Tweet tweet1 = new Tweet(1, "a", "@a", d);
12         Tweet tweet2 = new Tweet(2, "b", "@c", d);
13         Tweet tweet3 = new Tweet(3, "c", "@d", d);
14         Tweet tweet4 = new Tweet(4, "d", "@", d);
15         // 首先测试follow的两人传递
16         resultMap.put("a", Set.of("b", "c"));
17         resultMap.put("b", Set.of("c", "d"));
18         resultMap.put("c", Set.of("d"));
19         resultMap.put("d", Set.of());
20         followsGraph = MySocialNetwork.guessFollowsGraph(Arrays.asList(tweet1, tweet2, tweet3, tweet4));
21         assertEquals(resultMap, followsGraph);
22         // 再测试@自己的情况
23         Tweet tweet5 = new Tweet(5, "e", "@e", d);
24         resultMap.clear();
25         resultMap.put("e", Set.of());
26         followsGraph = MySocialNetwork.guessFollowsGraph(Arrays.asList(tweet5));
27         assertEquals(resultMap, followsGraph);
28     }
29 }
30
31 }
```

JUnit 4

Finished after 0.025 seconds

Runs: 1/1 Errors: 0 Failures: 0

twitter.MySocialNetworkTest [Runner: JUnit 4] (0.010 s)

test (0.010 s)

Failure Trace

由于一般情况的测试已经在上面提及验证。如下只测试特殊的情况。
测试@自己的情况；测试@的传递情况。

4 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
2019-02-28	19:00-21:30	编写问题 1 的 isLegalMagicSquare 函数并进行测试	按计划完成
2019-2-28	21:30-20:30	编写问题 1 的 generateMagicSquare 函数并进行测试	按时完成
2019-03-01	晚上	编写 P2	凸包遇到困难，未完成
2019-03-02	晚上	编写 P2	凸包遇到困难，未完成
2019-03-05	晚上	完成 P2	测试通过
2019-03-06	下午	完成 P3	遇到困难，未完成
2019-03-07	晚上	完成 P3	遇到困难，未完成
2019-03-08	晚上	完成 P3	完成，并通过测试
2019-03-09	晚上	完成 P4 前两部分	完成，并通过测试
2019-03-10	晚上	完成 P4	完成，并通过测试
2019-03-11	下午	完善所有 test	完成，并通过测试

5 实验过程中遇到的困难与解决途径

- 1, 凸包算法遇到困难, 开始与结束状态不好控制。解决途径, 设置循环增量, 设置真值数组。
- 2, 单源最短路径遇到困难。再次复习广度优先遍历算法
- 3, 问题四中的 getSmarter 遇到困难, 集合与映射数据结构还是不够熟悉。上网百度搜索相关数据结构知识。

6 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教训, 也可就以下方面谈谈你的感受 (非必须):

- (1) Java 编程语言是否对你的口味?

还好, 主要是之前完全没接触过 java, 上手有点难。

- (2) 关于 Eclipse IDE

很方便, 极大的提高了我的编程效率

- (3) 关于 Git

之前未曾使用, 第一次实验感觉不出来 git 的作用

- (4) 关于 CMU 和 MIT 的作业

量大, 感觉有部分地方要求不是特别清楚

- (5) 关于本实验的工作量、难度、deadline

deadline 合理, 难度中等偏上

- (6) 关于初接触“软件构造”课程

有点懵逼, 不过经过 lab1 的考验, 已经开始找到门道了。