

**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 | 陈祥春 |
| 学号 | 1170300826 |
| 班号 | 173008 |
| 电子邮件 |  |
| 手机号码 |  |

**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 2](#_Toc1392800)

[3.1 Magic Squares 3](#_Toc1392801)

[3.1.1 isLegalMagicSquare() 3](#_Toc1392802)

[3.1.2 generateMagicSquare() 4](#_Toc1392803)

[3.2 Turtle Graphics 6](#_Toc1392804)

[3.2.1 Problem 1: Clone and import 6](#_Toc1392805)

[3.2.2 Problem 3: Turtle graphics and drawSquare 8](#_Toc1392806)

[3.2.3 Problem 5: Drawing polygons 8](#_Toc1392807)

[3.2.4 Problem 6: Calculating Bearings 10](#_Toc1392808)

[3.2.5 Problem 7: Convex Hulls 12](#_Toc1392809)

[3.2.6 Problem 8: Personal art 12](#_Toc1392810)

[3.2.7 Submitting 14](#_Toc1392811)

[3.3 Social Network 14](#_Toc1392812)

[3.3.1 设计/实现FriendshipGraph类 14](#_Toc1392813)

[3.3.2 设计/实现Person类 16](#_Toc1392814)

[3.3.3 设计/实现客户端代码main() 16](#_Toc1392815)

[3.3.4 设计/实现测试用例 16](#_Toc1392816)

[3.4 Tweet Tweet 17](#_Toc1392817)

[3.4.1 Problem 1: Extracting data from tweets 17](#_Toc1392818)

[3.4.2 Problem 2: Filtering lists of tweets 18](#_Toc1392819)

[3.4.3 Problem 3: Inferring a social network 19](#_Toc1392820)

[3.4.4 Problem 4: Get smarter 20](#_Toc1392821)

[4 实验进度记录 23](#_Toc1392822)

[5 实验过程中遇到的困难与解决途径 23](#_Toc1392823)

[6 实验过程中收获的经验、教训、感想 24](#_Toc1392824)

[6.1 实验过程中收获的经验和教训 24](#_Toc1392825)

[6.2 针对以下方面的感受 24](#_Toc1392826)

# 实验目标概述

根据实验手册简要撰写。

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

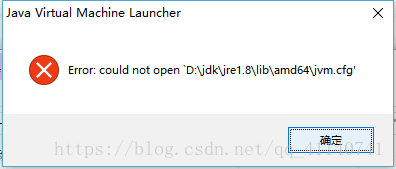
⚫ 基于 Git 的代码配置管理

# 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。

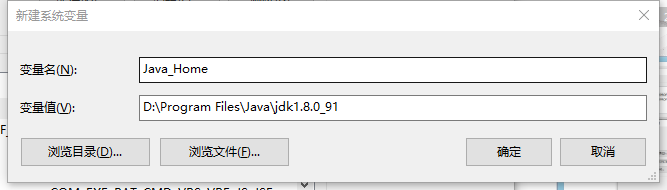
1. 首先在网上搜索安装eslipse和配置java8环境的教程，按教程进行操作。从官网上下载eslipse压缩包到电脑之后，解压，发现此时的eclipse是打不开的



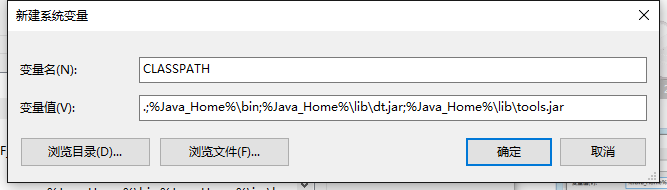
1. 出现以下情况都是因为还没有安装JDK的原因，然后开始安装JDK。从官网上下载JDK1.8版本，按照教程和默认操作进行安装
2. 最后配置JDK的环境变量。详细步骤如下：

win+E快捷键打开文件资源管理器，右键【此电脑】-》【属性】-》【高级】-》【环境变量】

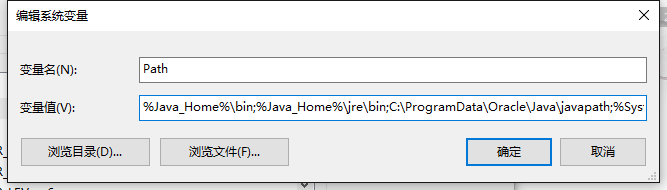
在”变量名“处填上”Java\_Home“，”变量值“为JDK安装路径：D:\Program Files\Java\jdk1.8.0\_91



在系统变量栏，新建CLASSPATH，配置变量值：%Java\_Home%\bin;%Java\_Home%\lib\dt.jar;%Java\_Home%\lib\tools.jar



最后是Path的配置，编辑Path变量，添加变量值为：%Java\_Home%\bin;%Java\_Home%\jre\bin



在这里给出你的GitHub Lab1仓库的URL地址（Lab1-学号）。

https://github.com/ComputerScienceHIT/Lab1-1170300826

# 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

在这里简要概述你对该任务的理解。

在Magic Squares.java，需要设计isLegalMagicSquare()函数判断文件中的矩阵是否是幻方，并且需要完备generateMagicSquare()函数，使其遇到负数和偶数能正常终止程序，然后调用isLegalMagicSquare()函数判断generateMagicSquare()生成的矩阵是否是幻方。

### isLegalMagicSquare()

按步骤给出你的设计和实现思路/过程/结果。

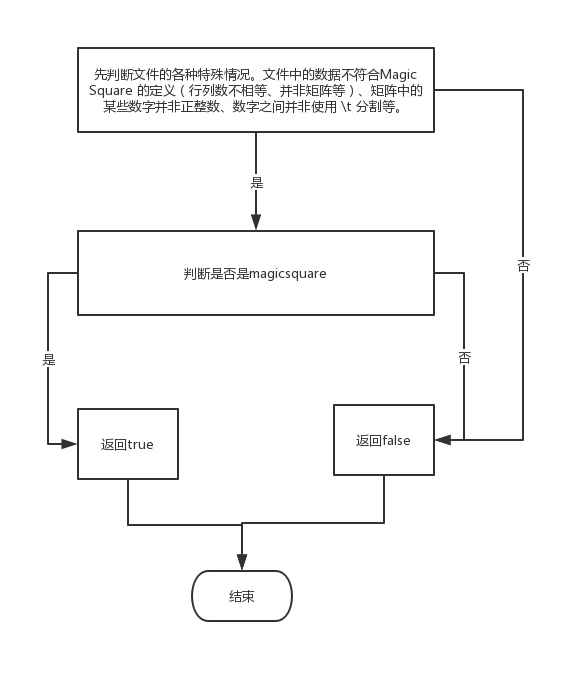
1. 功能：判断5个文本(1.txt到5.txt)中的矩阵是否是magicsquare，

需要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（行列数不相等、并非矩阵等）、矩阵中的某些数字并非正整数、数字之间并非使用 \t 分割等。若遇到这些情况，终止程序执行（isLegalMagicSquare 函数返回 false），并在控制台输出错误提示信息。

1. 参数：文件名。
2. 输出：如果是输出true，否则输出false。

设计和实现思路：

流程图如下：



先判断文件的各种特殊情况。文件中的数据不符合Magic Square 的定义（行列数不相等、并非矩阵等）、矩阵中的某些数字并非正整数、数字之间并非使用 \t 分割等。

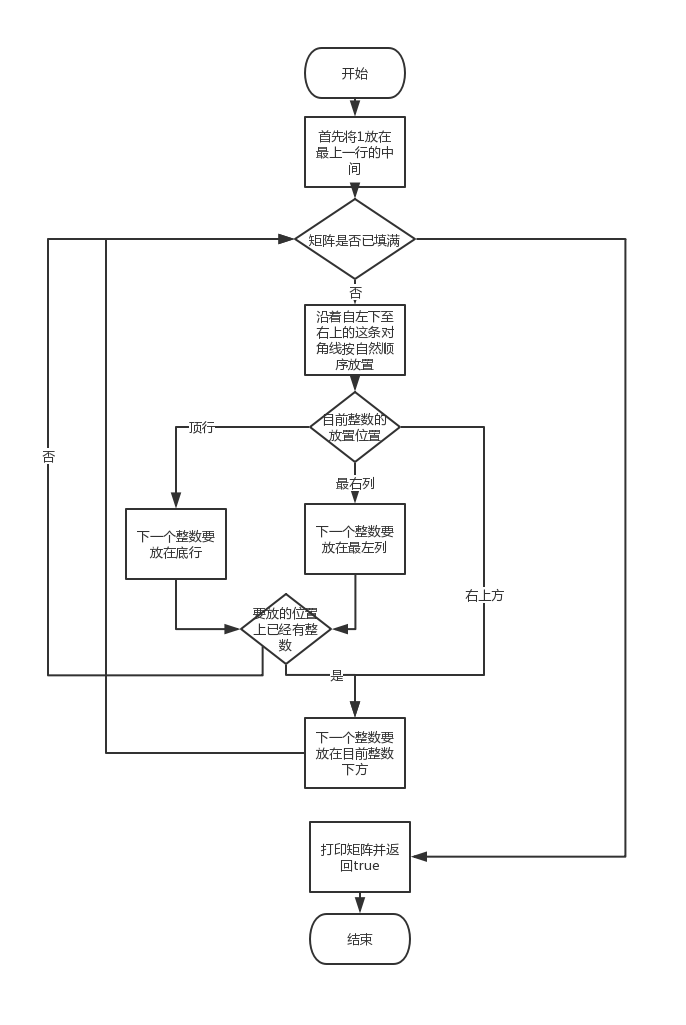
如果不存在使得程序终止的特殊情况，判断矩阵是否符合MagicSquare的定义。

结果：如果文件中的矩阵是幻方，则输出true，否则输出false

### generateMagicSquare()

按步骤给出你的设计和实现思路/过程/结果。

1. 该函数程序流程图



（2）根据生成奇数阶Magic Square的算法为：

首先将1放在最上一行的中间。其后沿着自左下至右上的这条对角线按自然顺序放置，但同时须作如下修正：

（a）在到达顶行时，下一个整数要放在底行，所放位置就是把底行当做顶行上边一行时该数应该放置的位置。

（b）当到达最右边的一列时，下一个整数要放在最左边的一列上，

所放位置就是把最左边的一列当作最右边那列的右边的列时该数应该放置的位置。

（c）当要放的位置上已经填好了整数，或上一个整数已经放在了幻方的右上角时，则当前要摆放的整数将放在紧挨上述位置的下方。

（3）扩展后的函数如下并添加中文注释。

public static boolean generateMagicSquare(int n) {

if(n <=0 || n % 2 == 0)

{

System.out.println("输入错误，请输入正奇数");

return false;

}

int magic[][] = new int[n][n];//创建一个n\*n的矩阵

int row = 0, col = n / 2, i, j, square = n \* n;

//square为矩阵元素总数量

for (i = 1; i <= square; i++)

{//首先将1放在最上一行的中间。其后沿着自左下至右上的这条对角线按自然顺序放置

magic[row][col] = i;

if (i % n == 0)//当要放的位置上已经填好了整数，或上一个整数放在右上角时，则当前要摆放的整数将放在紧挨上述位置的下方

row++;

else {

if (row == 0)//在到达顶行时，下一个整数要放在底行，所放位置就是把底行当做顶行上边一行时该数应该放置的位置

row = n - 1;

else

row--;

if (col == (n - 1))//当到达最右边的一列时，下一个整数要放在最左边的一列上，

col = 0;

else

col++;

}

}

//将产生的 magic square 写入文件\src\P1\txt\6.txt

try {

File file = new File("src\\P1\\txt\\6.txt");

FileWriter writer = new FileWriter(file);

BufferedWriter out = new BufferedWriter(writer);

try

{

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

out.write(magic[i][j]+"\t");

}

out.write("\r\n");

}

out.flush();

}catch (IOException e) {

e.printStackTrace();

}

} catch (IOException e) {

e.printStackTrace();

}

return true;

}

## Turtle Graphics

在这里简要概述你对该任务的理解。

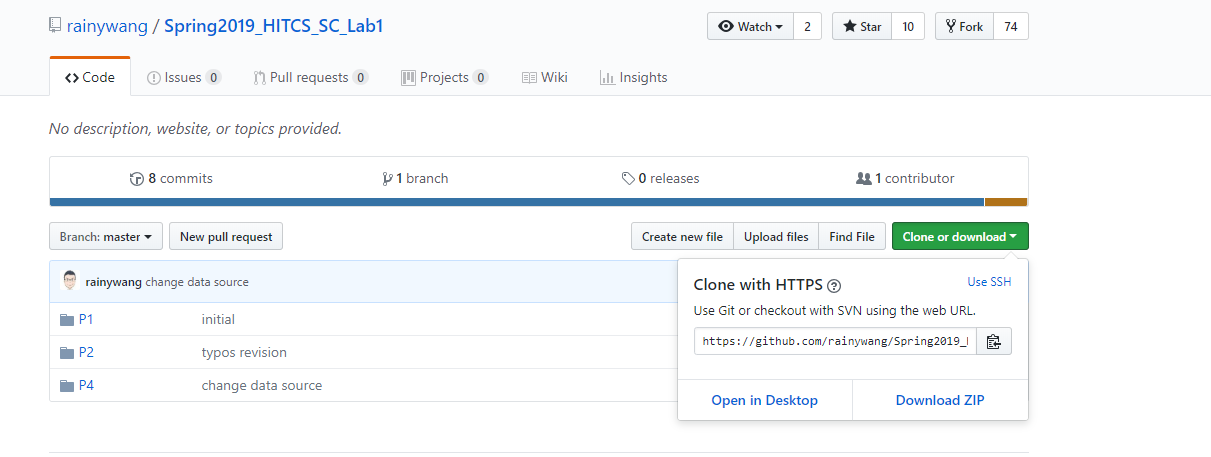
补全TurtleSoup.java文件，通过使用logo语言，在java画布上完成绘图任务以及通过TurtleSoupTest.java的测试。

### 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。

从GitHub获取该任务的代码:

因为无法连接无法连接 MIT 的 didit 服务器，所以直接选择Download ZIP



在本地创建git仓库:

首先选择一个合适的文件目录，我选择\git\Lab\Lab1-1170300826

通过git init命令把这个目录变成Git可以管理的仓库：$ git init

使用git管理本地开发:

使用git init命令建立好本地git仓库后，关联远程仓库

以提交scr文件夹为例：

使用命令git add把文件添加到仓库，$ git add src。

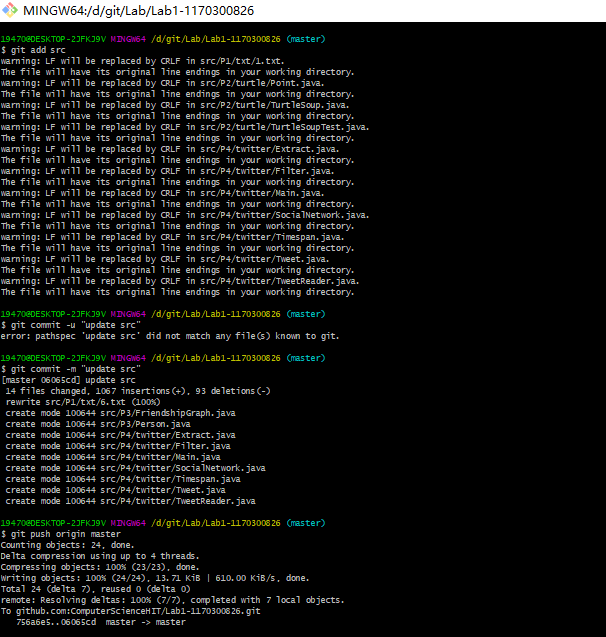
使用命令git commit把文件添加到仓库，$ git commit -u "update src"

在本地的Lab1-1170300826仓库下运行命令：

$git remote add origin [git@github.com:ComputerScienceHIT/Lab1-1170300826.git](mailto:git@github.com:ComputerScienceHIT/Lab1-1170300826.git)

最后提交至远程仓库：（首次提交使用$ git push -u origin master

）$ git push origin master



### Problem 3: Turtle graphics and drawSquare

按步骤给出你的设计和实现思路/过程/结果。

设计和实现思路：在Turtle graphics这个问题集中包含Logo语言的受限子集：

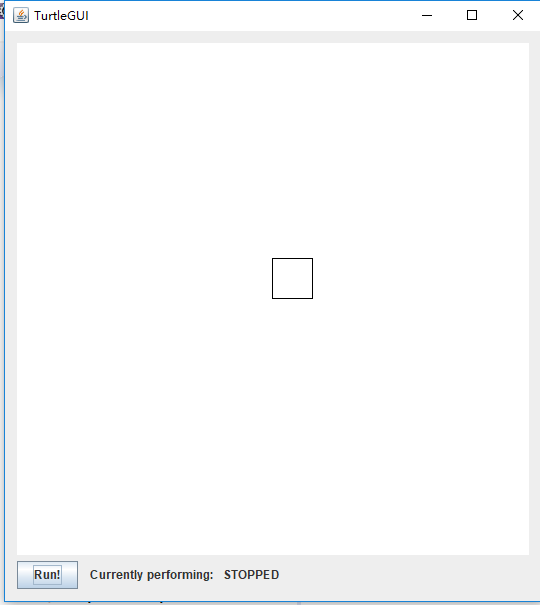
forward(units) :以单位像素为单位在当前方向上移动乌龟，其中单位为整数。

turn(degrees) :将乌龟向右旋转角度（顺时针），其中度数是双精度浮点数。

drawSquare函数的功能是画一个正方形，根据正方形内角为90且边长相等，用forward让turtle沿当前方向走sideLength距离，用turn使得turtle旋转90度，依次循环直到turtle回到起点。

过程：将turtle.forward(sideLength);turtle.turn(90);语句循环四次即可得到正方形。

结果：

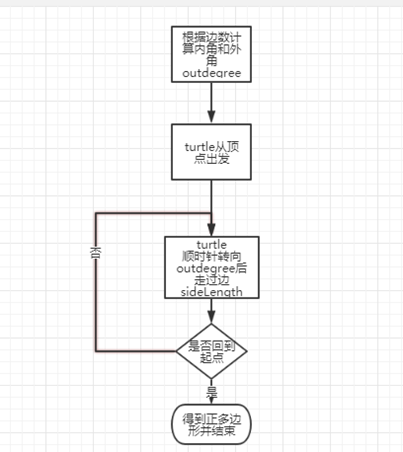


### Problem 5: Drawing polygons

设计和实现思路：

要画一个正多边形首先要求内角，根据正多边形所有外角和等于360度且每个内角相等可得公式，内角度数 = (边数 - 2)\*180/边数。根据正多边形的性质，turtle每走过边长的距离（即画出一条边）后需要顺时针调整的角度恰好等于外角度，因此循环多次，直至turtle回到起点，可画出一个正多边形。

过程：



关键代码如下：

int i;

double indegree = (sides - 2)\*180/sides;

double outdegree = 180 - indegree;

for(i=0;i<sides;i++)

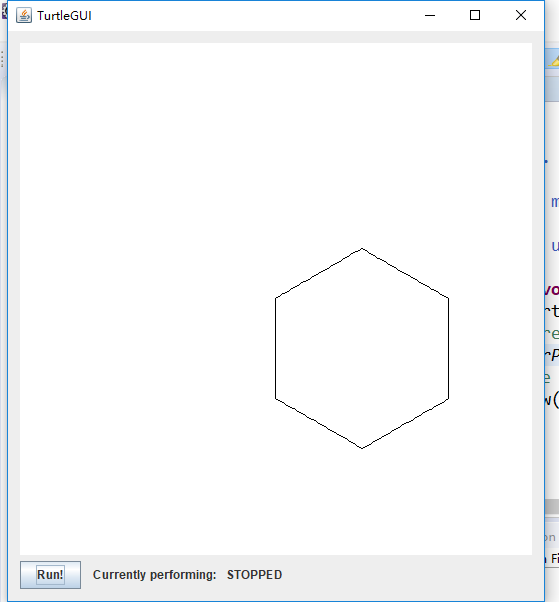
{

turtle.turn(outdegree);

turtle.forward(sideLength);

}

结果：sides = 6,sidelength = 100,画出六边形。

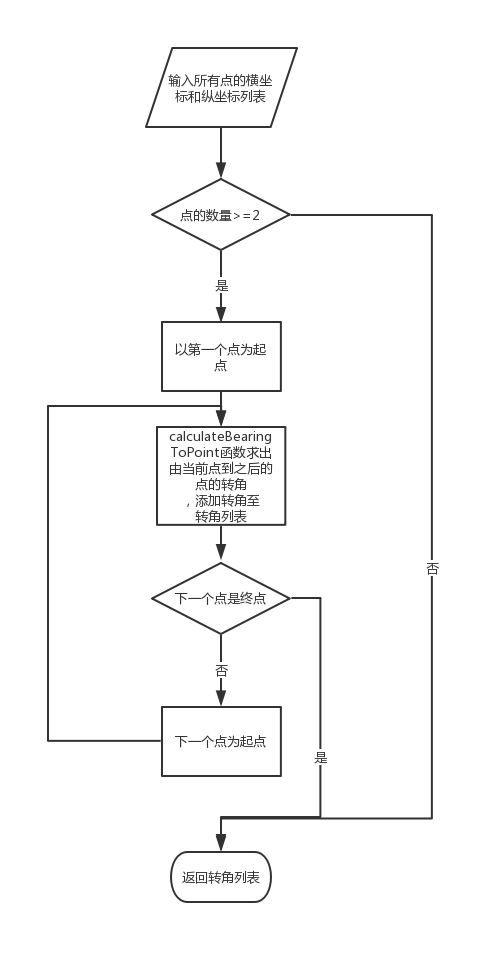


### Problem 6: Calculating Bearings

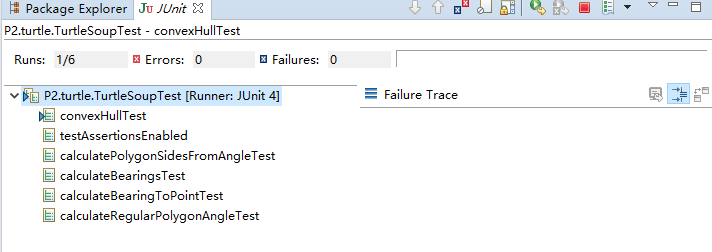
设计和实现思路：该函数的功能是根据点的x集合和y集合，得到turtle每次转向的角度，返回所有转角角度的集合，当点集为空时，返回集合为空。

数组的第一个点为起点，利用calculateBearingToPoint函数求出由当前点到下一个点的转角。然后以下一个点为起点，利用calculateBearingToPoint函数求出由当前点到之后的点的转角。以此类推，直到遍历至最后一个点结束。最后得到所有转角角度的集合。

过程：



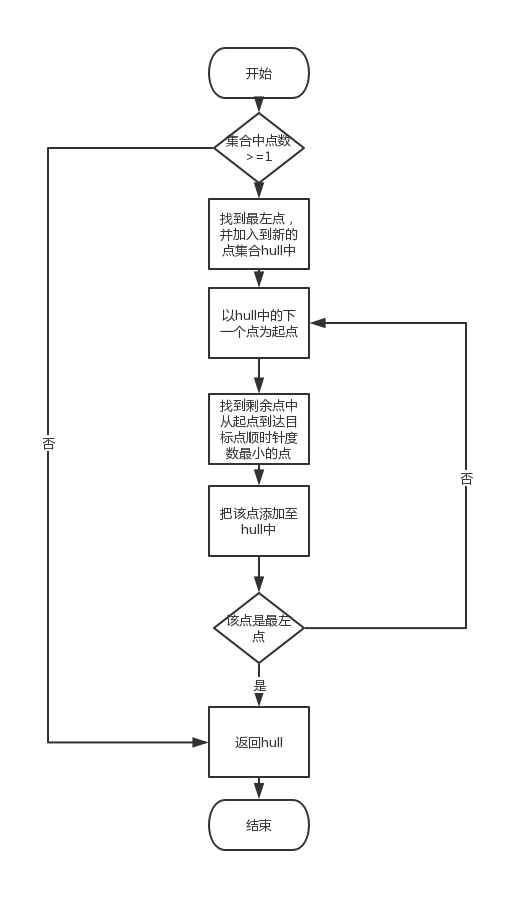
结果：返回所有转角角度的集合，通过测试。



### Problem 7: Convex Hulls

设计和实现思路：要解决凸包问题可以采用礼物包装算法。首先集合中点数是否小于1，若是直接返回。否则从最左侧点开始，顺时针方向继续移动，在未访问过的点中求更顺时针（即从起点到达目标点偏离度数最小）的点，并把该点作为下一次移动的起点，继续在未访问过的点中求更顺时针（即从起点到达目标点偏离度数最小）的点，以此类推，直到再次到达最左点

过程：流程图如图：

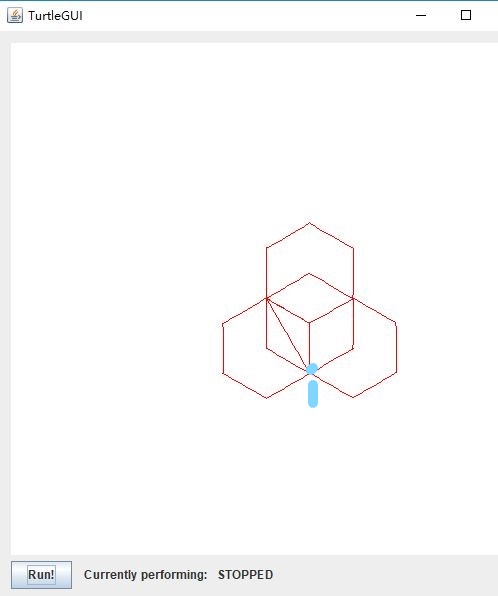


结果：得到一个包含所有点的最小凸多边形，返回该最小凸多边形边上的点集合并通过测试。

### Problem 8: Personal art

设计和实现思路：该图形由一个中央正多边形和三个红色正六边形组成。因此先画中央的正多边形，再画周围三个对称的正多边形。

过程：



1.先设置画笔为红色。turtle.color(PenColor.RED);

2.根据正六边形的性质，画图

for(i=0;i<sides;i++)

{

turtle.turn(outdegree);

turtle.forward(sideLength);

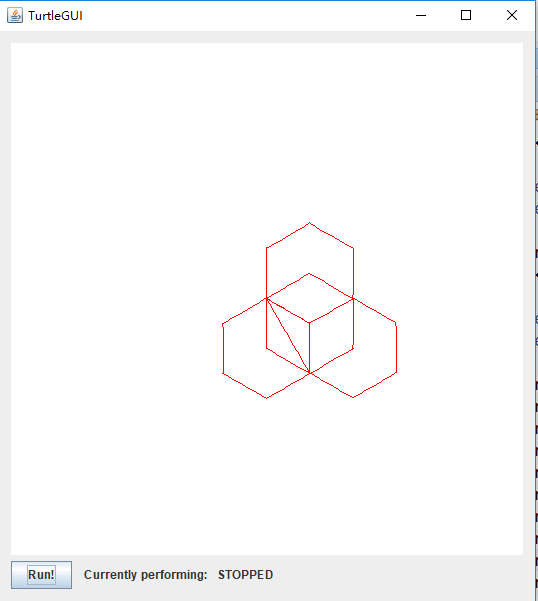
}

3.turtle回到起点后，调整方向，turtle.turn(outdegree)，然后同步骤2画出第2个正六边形

4. turtle回到起点后，调整方向，turtle.turn(180+outdegree)，然后同步骤2画出第3个正六边形

5. turtle回到起点后，需要从起点到达点1。先调整角度turtle.turn(indegree + 90)，经过数学计算turtle.forward((int)(sideLength\*1.732))，到达点1，然后根据正六边形的性质画第4个正六边形。

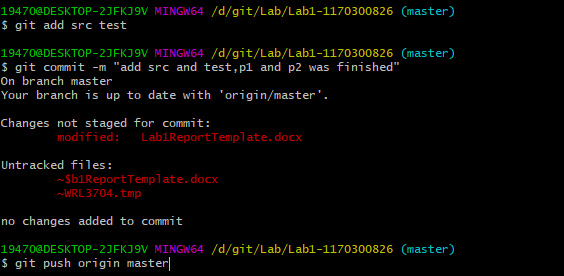
结果：得到一个由一个中央正多边形和三个对称的正六边形组成的红色图形



### Submitting

如何通过Git提交当前版本到GitHub上你的Lab1仓库。

此时需要提交的文件夹有src和test文件夹。执行以下操作：



## Social Network

在这里简要概述你对该任务的理解。

实现Person类，实现并测试代表社交网络中友谊的FriendshipGraph类，并可以计算图中两个人之间的距离。 将社交网络建模为无向图，其中每个人都连接到零个或多个人。

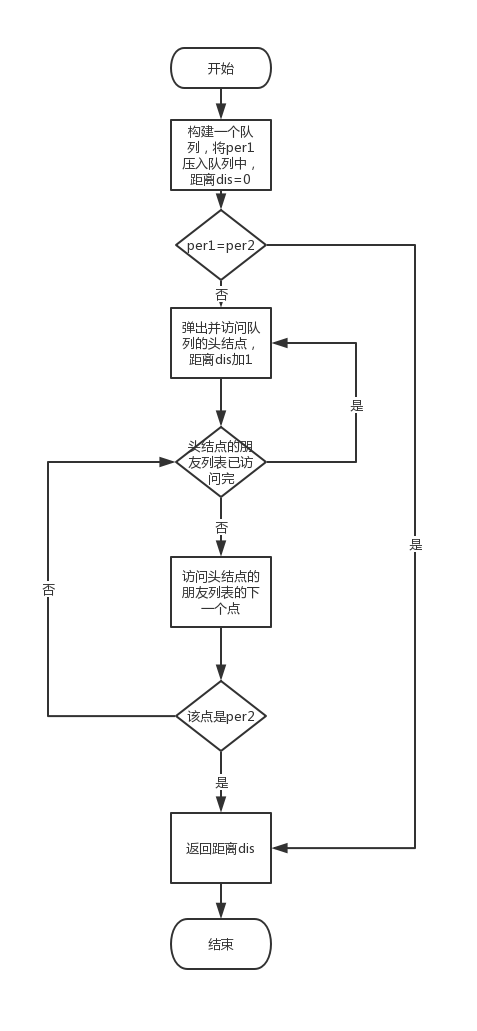
### 设计/实现FriendshipGraph类

给出你的设计和实现思路/过程/结果。

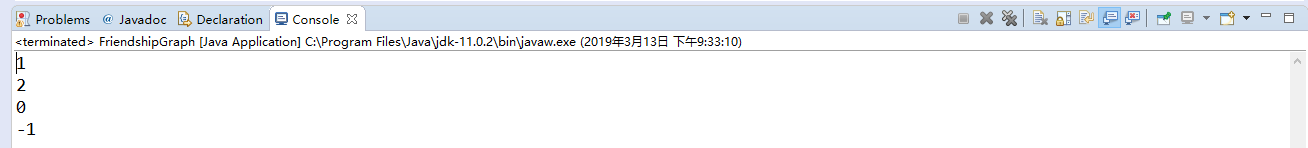
设计和实现思路：

1. addEdge函数：直接调用per1.addFriend(per2)将per2添加至per1的朋友列表中
2. addVertex函数：定义一个person类列表people, 直接调用people.add(per)，添加对象至列表
3. getDistance函数：使用广度优先搜索计算两个person类对象的距离。构建队列结构实现广度优先搜索。

过程：getDistance函数的流程图如下：



结果：返回两个person类对象之间的距离



### 设计/实现Person类

给出你的设计和实现思路/过程/结果。

设计和实现思路：构建了一个person类和三个方法：addFriend()，getName()和getFriendList()。

1. person类：对person对象，初始化的成员变量name(this.name = name)，并创建一个当前person的相邻节点列表。定义一个局部变量namelist用于检查重名。
2. addFriend()的功能是添加相邻节点
3. getName()的功能是获取当前Person的名字
4. getFriendList()的功能是获得当前Person相邻节点列表

过程与结果：

public Person(String name) {// Person类的构造方法

if(namelist.contains(name))

{

System.out.println("输入重名，结果无效");

System.exit(0);

}

this.name = name;

namelist.add(name);

friendList = new ArrayList<>();

}

public void addFriend(Person pb) {// 添加相邻节点

friendList.add(pb);}

public String getName() {// 获取当前Person的名字

return name;}

//获得当前person的相邻节点列表

public List<Person> getFriendList() {

return this.friendList;

}

### 设计/实现客户端代码main()

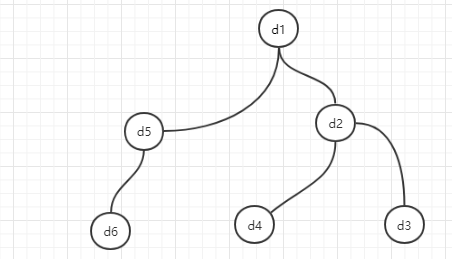
给出你的设计和实现思路/过程/结果。

该部分已给出

### 设计/实现测试用例

给出你的设计和实现思路/过程/结果。

设计和实现思路：根据FriendshipGraph类和person类构建如下社交网络：



过程：

测试d1和d5的距离，d1和d6的距离，d5和d5的距离，d4和d6的距离

assertEquals(1, graph.getDistance(d1, d5), 0.001);

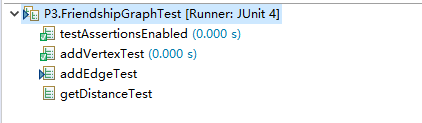
assertEquals(2, graph.getDistance(d1, d6), 0.001);

assertEquals(0, graph.getDistance(d5, d5), 0.001);

assertEquals(-1, graph.getDistance(d4, d6), 0.001);

对函数addVertexTest()和addEdgeTest()，直接运行函数测试

结果：



## Tweet Tweet

请自行组织子目录结构。

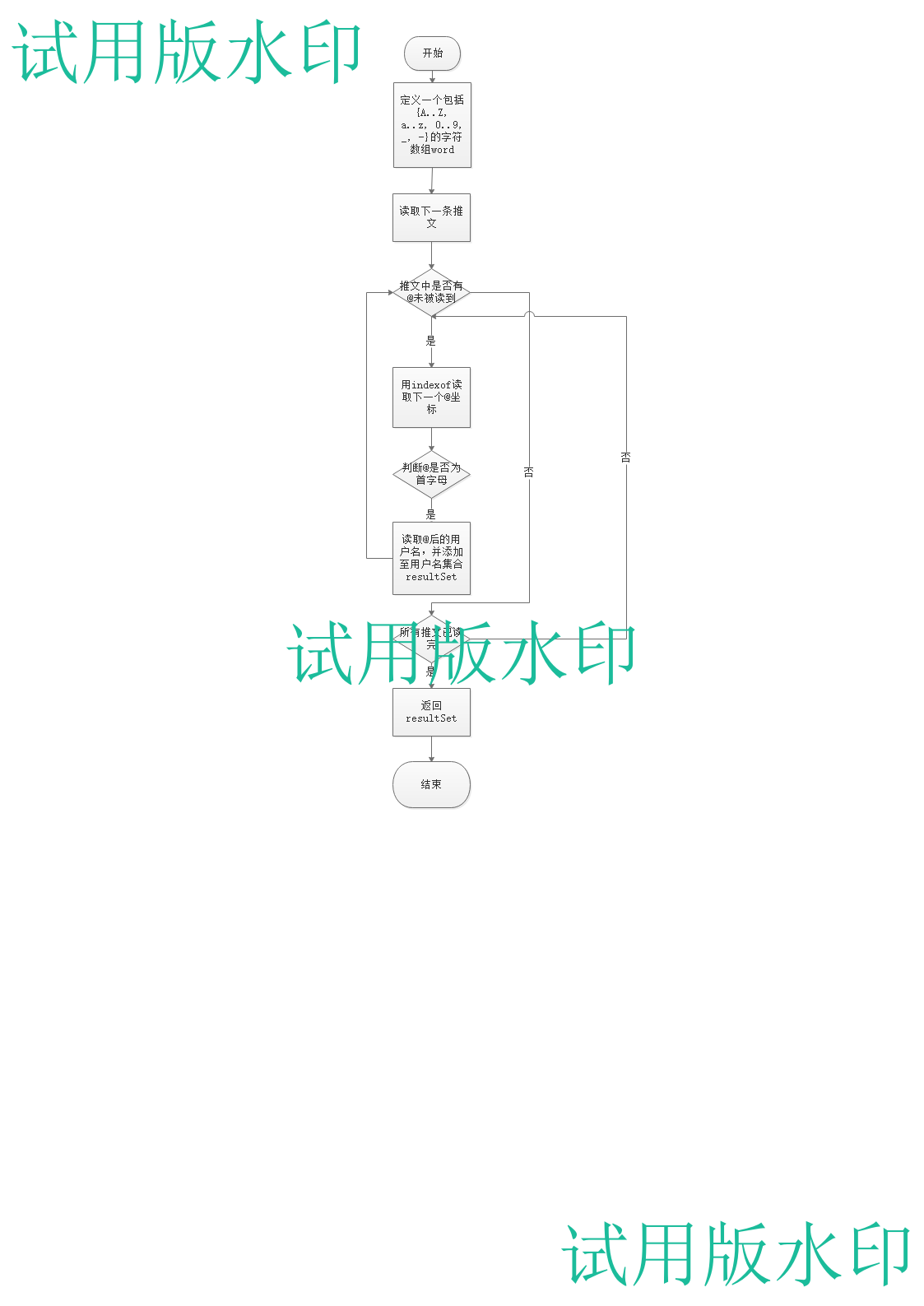
### Problem 1: Extracting data from tweets

1. 问题1的主要任务是从推文中提取数据。为getTimespan()和getMentionedUsers()设计，记录和实现测试用例。
2. 函数getTimespan功能是：计算所有推文所跨越的时间段。

设计思路与实现过程：遍历每一个推文的发布时间，通过比较得到所有推文中的最早发布时间和最晚发布时间，跨越的时间段 = 最晚发布时间 - 最早发布时间

1. 函数getMentionedUsers功能是：获得推文中提到的所有用户名

设计思路与实现过程：流程图如下：

：

设计思路为：定义一个包括{A..Z, a..z, 0..9, \_, -}的字符数组word，用于判断字母是否为用户名的合法字符。利用for循环读取推文，对于每一条推文，从左到右读取推文中的@，若不存在@，则不存在用户名，直接进入下一次循环。若存在@，找到@位置，判断@是否为首字母，否则判断@前面的字符是否为合法字符，若合法则@后用户名无效，若不合法，读取@后的用户名（遇到非法字符停止）。以此类推，找到下一个@并读取用户名。读取完所有@后进入下一次循环。推文全部读取后循环终止。

### Problem 2: Filtering lists of tweets

1. 问题2的主要任务是过滤与条件匹配的推文列表。设计writtenBy() ， inTimespan()和containing()并通过测试用例
2. 函数writtenBy()功能是：找到一个用户发的所有推文

设计思路与实现过程：定义一个该用户的推文列表name\_result，循环一遍所有推文，若推文的作者是该用户则添加至name\_result，循环结束后name\_result即为结果

1. 函数inTimespan()功能是：找到某个时间段内发布的所有推文

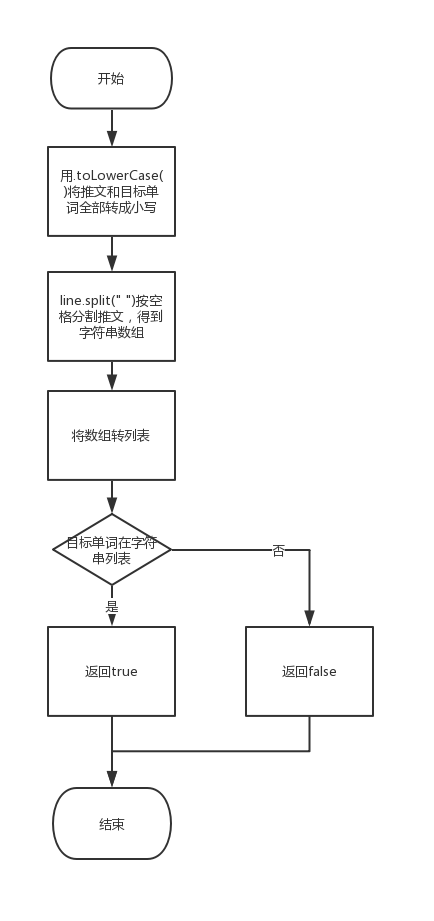
设计思路与实现过程：定义一个该时间段内发布的推文列表time\_result，循环一遍所有推文，若推文是在该时间段发布的则添加至time\_result，循环结束后time\_result即为结果

1. 函数containing()功能是：找到包括目标单词的所有推文

设计思路与实现过程：定义一个推文列表word\_result，循环一遍所有推文，调用Isexit ()是否需包括目标单词若包括则添加至word\_result，循环结束后word\_result即为结果

1. 新增public函数Isexit ()功能是：判断推文中是否包括目标单词

设计思路与实现过程：流程图如下



### Problem 3: Inferring a social network

1. 问题3的主要任务是设计guessFollowsGraph()和influencers()，并通过测试用例。

guessFollowsGraph()方法为推文列表中提到的人创建社交网络。influencers()方法返回按其影响排序的人员列表（关注者总数）。

1. 新增函数getMentionedUsers功能：返回某条推文提到的用户名集合

设计思路与实现过程：与Extract.java中的getMentionedUsers类似，Extract.java中的getMentionedUsers需要处理许多推文，该文件中的getMentionedUsers只需处理一条推文并返回该用户名关注的用户名集合

1. 函数guessFollowsGraph ()功能是： 返回一个用户与被关注者的关系网络

设计思路与实现过程：定义一个推文列表result\_map，循环一遍所有推文，获取发布每一条推文的用户，调用getMentionedUsers获取该用户关注的用户集合，并将该用户和关注的用户集合放入result\_map中。循环结束后result\_map即为结果

1. 函数influencers功能是： 在社交网络中找到最多粉丝影响最大的人

设计思路与实现过程：

1.定义一个字符串列表name，用来记录名字

对Map进行遍历，从记录社交网络中的每一个被追随者，并计数

第一次循环得到所有被追随者名字集合name，可以重复

for(String o:followsGraph.keySet())

{

for(String temp:followsGraph.get(o))

{

name.add(temp);

}

}

2.第二次循环，遍历name，定义一个表用户与粉丝数量的社交网络的地图mentioned（Map<String, Integer> mentioned）

对每个被追随者名字计数，并对每个被追随者名字计数，然后放入mentioned中

3.对mentioned排序，使得mentioned按value降序排列

4.定义一个字符串列表influencer\_set，将mentioned中的键依次存入influencer\_set中，得到按粉丝数量降序的用户列表

### Problem 4: Get smarter

1. 问题4的主要任务是在guessFollowsGraph（）的基础上除了A的推文中出现@B表示A关注B外实现一种额外的证据：如果A和B相互关注，B和C相互关注，则判定A和C相互关注。设计 和实现MyGuessFollowsGraph ()和MySocialNectworkTest()，并要求MyGuessFollowsGraph ()通过MySocialNectworkTest()的测试用例。
2. 新增函数getMentionedUsers功能：返回某条推文提到的用户名集合

设计思路与实现过程：与Extract.java中的getMentionedUsers类似，Extract.java中的getMentionedUsers需要处理许多推文，该文件中的getMentionedUsers只需处理一条推文并返回该用户名关注的用户名集合。

1. 函数MyGuessFollowsGraph ()功能是： 返回在增加额外的证据后用户与被关注者的关系网络

设计思路与实现过程：

1.定义一个推文列表result\_map，循环一遍所有推文，获取发布每一条推文的用户，调用getMentionedUsers获取该用户关注的用户集合，并将该用户和关注的用户集合放入result\_map中。

循环结束后得到在@的证据下的关系网络result\_map。（这部分设计过程同guessFollowsGraph）

2.遍历result\_map。

循环列举每一个作者的关注人数集合。如果被关注者的关注集合无作者,则删除该被关注者。

for(String au:result\_map.keySet())

{//第二重循环遍历作者的关注人数集合

for(String mention:result\_map.get(au))

{

temp\_set.add(mention);

if((result\_map.get(mention)) == null||!(result\_map.get(mention)).contains(au))//被关注者的关注集合无作者,则删除

{

delList.add(mention);

}

}

temp\_set.removeAll(delList);

mutual\_map.put(au,temp\_set);

delList = new ArrayList<String>();

temp\_set = new HashSet<String>();

}

在result\_map的基础上得到相互关注的关系网络mutual\_map。

3.遍历相互关注的用户集合mutual\_map。

循环列举每一个作者的相互关注人数集合。在相互关注者的相互关注集合中若有作者未关注的人，则添加关注。

for (String ps: mutual\_map.keySet())

{

for(String temp:result\_map1.get(ps))//避免改变result\_map1

{

tempset.add(temp);

}

if(!(mutual\_map.get(ps)).isEmpty())//如果不为空

{

for(String end:mutual\_map.get(ps))//第二次遍历集合中相互关注者

{

for(String per:mutual\_map.get(end))

//第三次遍历相互关注者的关注集合，若有作者未关注的人则添加关注

{

if(!(result\_map1.get(ps)).contains(per))//若有作者未关注的人则添加关注

{

delList.add(per);

}

}

}

}

tempset.addAll(delList);

smartFollow.put(ps,tempset);

tempset = new HashSet<String>();

delList = new ArrayList<String>();

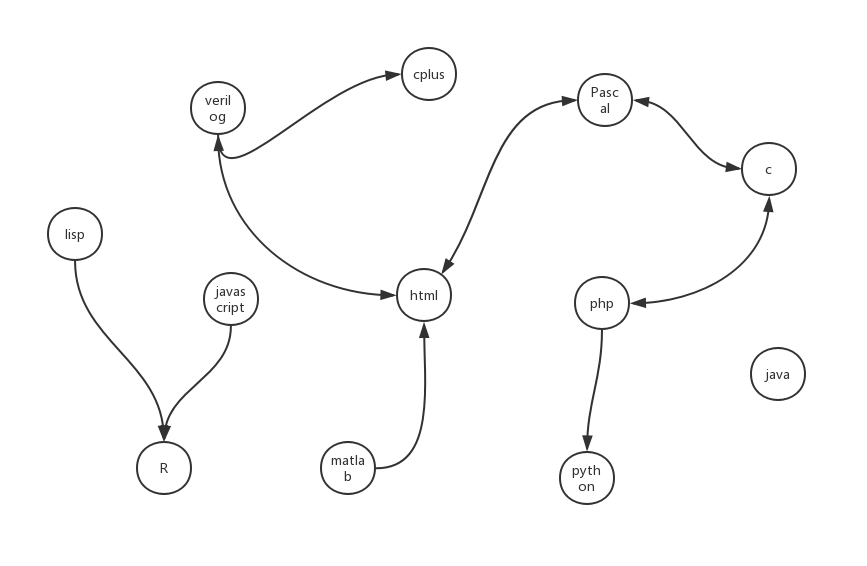
}

在mutual\_map的基础上得到新的关系网络smartFollow。

1. 函数MySocialNectworkTest()功能是： 测试函数MyGuessFollowsGraph ()

设计思路与实现过程：

测试样例中的关注和被关注者之间的关系网络。



1. 检测Pascal与php是否相互关注，在guessFollowsGraph两者不相互关注，在MyGuessFollowsGraph中两者相互关注

assertTrue("Expected Pascal contains php", (smartGraph.get("Pascal")).contains("php"));

assertTrue("Expected php contains Pascal", (smartGraph.get("php")).contains("Pascal"));

1. 检测c与html是否相互关注，在guessFollowsGraph两者不相互关注，在MyGuessFollowsGraph中两者相互关注

assertTrue("Expected c contains html", (smartGraph.get("c")).contains("html"));

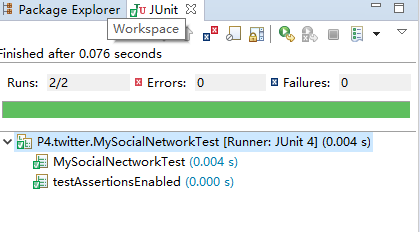
assertTrue("Expected html contains c", (smartGraph.get("html")).contains("c"));

1. 检测Pascal与verilog是否相互关注，在guessFollowsGraph两者不相互关注，在MyGuessFollowsGraph中两者相互关注

assertTrue("Expected verilog contains Pascal", (smartGraph.get("verilog")).contains("Pascal"))

assertTrue("Expected Pascal contains verilog", (smartGraph.get("Pascal")).contains("verilog"))

1. 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 | 18:00-22:30 | 编写问题1并进行测试 | 按计划完成 |
| 2019-02-29 | 18:00-22:30 | 编写问题2并进行测试 | 按计划完成 |
| 2019-03-01 | 18:00-22:30 | 编写问题3并运行 | 延期2小时完成 |
| 2019-03-02 | 18:00-22:30 | 编写问题3的测试文件并通过测试 | 按计划完成 |
| 2019-03-03 | 18:00-22:30 | 编写问题4的Extract.java和Filter.java文件并通过测试 | 按计划完成 |
| 2019-03-04 | 18:00-22:30 | 编写问题4的SocialNetwork.java并通过测试  并 | 按计划完成 |
| 2019-03-05 | 18:00-22:30 | 编写问题4Get smarter部分和相应的测试文件并通过测试 | 按计划完成 |
| 2019-03-11 | 19:00-22:30 | 写P1和P2部分实验报告 | 延期1小时完成 |
| 2019-03-12 | 19:00-22:30 | 写P4部分实验报告 | 遇到困难，未完成 |
| 2019-03-13 | 20:00-22:30 | 完成剩下的实验报告 | 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 |
| P1问题上对generateMagicSquare()函数，用户输入非整数不知道如何处理 | 通过CSDN博客得到可以用try {}catch(Exception e) 处理用户输入非整数的情况 |
| P2问题上对凸包问题的解决方案（礼物包装算法）不了解 | 通过礼物包装算法的维基百科了解了其基本原理，并通过查看类似问题的解决方法学会了解决凸包问题 |
| P3问题上由于之前未系统java,对创建对象和类相关操作不了解 | 在mooc上学习了java创建对象和类相关操作的课程视频 |
| P4问题上getMentionedUsers不知道如何从推文中提取@后的用户名 | 通过与同学讨论得到解决方案 |

# 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

经验和教训：学会了利用github对软件进行管理，通过问题P1学会了如何利用文件操作对文件中的数据进行判断和处理。通过问题P2了解了java图形库，以及凸包问题的解决方案——礼物包装算法。在问题P3中，学会了创建person类和对象，面向person类和对象编写函数。在问题P4中，我利用了map和list等相关的操作，并学会了测试文件的编写。在调试过程中，我遇到了以下问题：

1. temp\_set = result\_map，每次改变temp\_set ，result\_map也随着改变。原因是 java中赋值传对象（地址）

2.使用for循环时出现ConcurrentModificationException异常报错，原因时每次循环不能改变被读取循环变量的值

## 针对以下方面的感受

1. Java编程语言是否对你的口味？

Java语言功能齐全，虽然不如python简洁，但Java源代码的书写不拘泥于特定的环境，可以用记事本、文本编辑器等编辑软件来实现。Java导包比较麻烦，其他还好。

1. 关于Eclipse IDE

Eclipse是最容易学习上手的IDE,快捷键丰富。经过设置后有代码自动补全的功能，调试时可以在java和debug模式中切换，比较方便。（但还是IDEA更好，超级方便的GIT项目管理，更好的调试）

1. 关于Git和GitHub

Git是目前世界上最先进的分布式版本控制系统。git有以下功能：

从服务器上克隆完整的Git仓库（包括代码和版本信息）到单机上；

根据不同的开发目的，创建分支，修改代码；在单机上自己创建的分支上提交代码；在单机上合并分支。Git适合用于管理和存储代码。

5、把服务器上最新版的代码fetch下来，然后跟自己的主分支合并。

1. 关于CMU和MIT的作业

作业有趣但难度太大了

1. 关于本实验的工作量、难度、deadline

工作量适度，难度中等，deadline适合

1. 关于初接触“软件构造”课程

由于缺少课本，以及对PPT概念过于抽象，上课听不太懂