

**2021年春季学期  
计算学部《软件构造》课程**

**Lab 1实验报告**

|  |  |
| --- | --- |
| 姓名 | 熊皓 |
| 学号 | 1190501809 |
| 班号 | 1903004 |
| 电子邮件 | 1439188741@qq.com |
| 手机号码 | 18323246430 |

**目录**

[1 实验目标概述 1](#_Toc29325521)

[2 实验环境配置 1](#_Toc29325522)

[3 实验过程 1](#_Toc29325523)

[3.1 Magic Squares 1](#_Toc29325524)

[3.1.1 isLegalMagicSquare() 1](#_Toc29325525)

[3.1.2 generateMagicSquare() 1](#_Toc29325526)

[3.2 Turtle Graphics 1](#_Toc29325527)

[3.2.1 Problem 1: Clone and import 2](#_Toc29325528)

[3.2.2 Problem 3: Turtle graphics and drawSquare 2](#_Toc29325529)

[3.2.3 Problem 5: Drawing polygons 2](#_Toc29325530)

[3.2.4 Problem 6: Calculating Bearings 2](#_Toc29325531)

[3.2.5 Problem 7: Convex Hulls 2](#_Toc29325532)

[3.2.6 Problem 8: Personal art 2](#_Toc29325533)

[3.2.7 Submitting 2](#_Toc29325534)

[3.3 Social Network 2](#_Toc29325535)

[3.3.1 设计/实现FriendshipGraph类 2](#_Toc29325536)

[3.3.2 设计/实现Person类 2](#_Toc29325537)

[3.3.3 设计/实现客户端代码main() 2](#_Toc29325538)

[3.3.4 设计/实现测试用例 3](#_Toc29325539)

[4 实验进度记录 3](#_Toc29325540)

[5 实验过程中遇到的困难与解决途径 3](#_Toc29325541)

[6 实验过程中收获的经验、教训、感想 3](#_Toc29325542)

[6.1 实验过程中收获的经验和教训 3](#_Toc29325543)

[6.2 针对以下方面的感受 3](#_Toc29325544)

# 实验目标概述

本次实验通过求解三个问题，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。

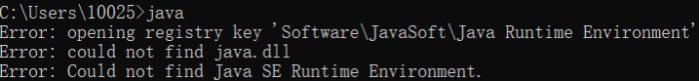
# 实验环境配置

**2.1配置环境过程**

1. 通过阅读实验手册在本地机器上安装了相应的开发环境（JDK、Eclipse、 Git）.
2. 通过查阅CSDN, 在IDEA官网安装了并配置了IDEA。
3. 阅读https://github.com/junit-team/junit4/wiki/Getting-started, 学习了如 何使用JUnit4为Java程序编写测试代码并执行测试

**2.2安装和配置环境时遇到的问题**

配置JDK是出现下列问题:



解决方案：卸载并重新安装

**2.3在这里给出你的GitHub Lab1仓库的URL地址**

**https://github.com/ComputerScienceHIT/HIT-Lab1-1190501809**

# 实验过程

## Magic Squares

Magic Squares主要包含两大任务：

1. 设计isLegalMagicSquare()函数进行判断给定矩阵是否为MagicSquare，同时能够处理输入文件的各种特殊情况（终止程序执行并输出错误信息），维持程序健壮性。
2. 为generateMagicSquare()函数绘制流程图，并解释其生成magicsquare的原理（添加中文注释），同时解释偶数与负数报错提示信息的含义，最后将生成的magicsquare写入6.txt并用isLegalMagicSquare()函数进行测试。

### isLegalMagicSquare()

思路：

要求函数能够读入文件中的数字并存储在矩阵中，所以首先需要判断读入的数据是否能够构成一个矩阵，然后分别就矩阵的每一行、每一列以及对角线的和时是否相等，同时需要判断几个错误：

1. 行列数不相等；
2. 并非矩阵；
3. 矩阵中的数字不是正整数；
4. 矩阵中数字未以\t作为分隔符

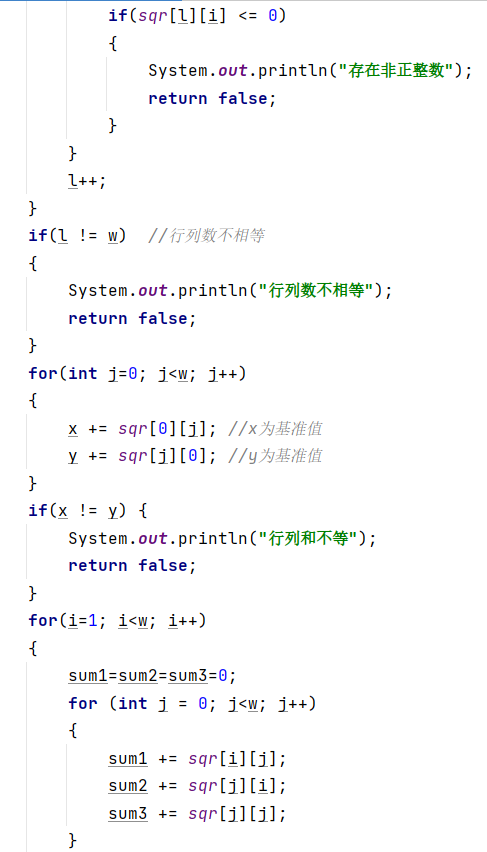
最后如所有条件都满足，则输出是MagicSquare。

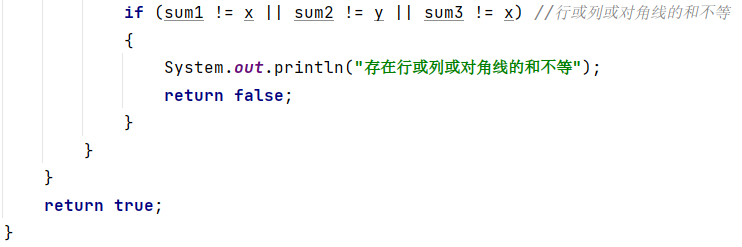
实现过程：

对于几个错误，判断方法如下：

1. 有数字未用/t分隔或某一行/列缺少数字：由于使用了split（“\t”）方法,因此判断方法就在于每次对\t分隔后的字符串的长度。若长度不等于基准值，则代表要么有数字未用\t分割或某一行/列缺少数字。
2. 对于矩阵中的数字不是正整数，我采用的方法为**catch**(NumberFormatException e1)的方法，若有浮点数或负数，则报错。
3. 对于行列数不相等，我的行数的基准是每次根据读的一行的数字来决定的，而列数则是一共读的行的次数，因此只需判断这两个数是否相等，就可判断行列数是否相等。
4. 对于行、列、对角线的和是否相等，我采用的是先计算三者的各自的一个基准值，然后比较这几个基准值，若相等，则继续比较剩余行、列的和；若不相等，则报错。

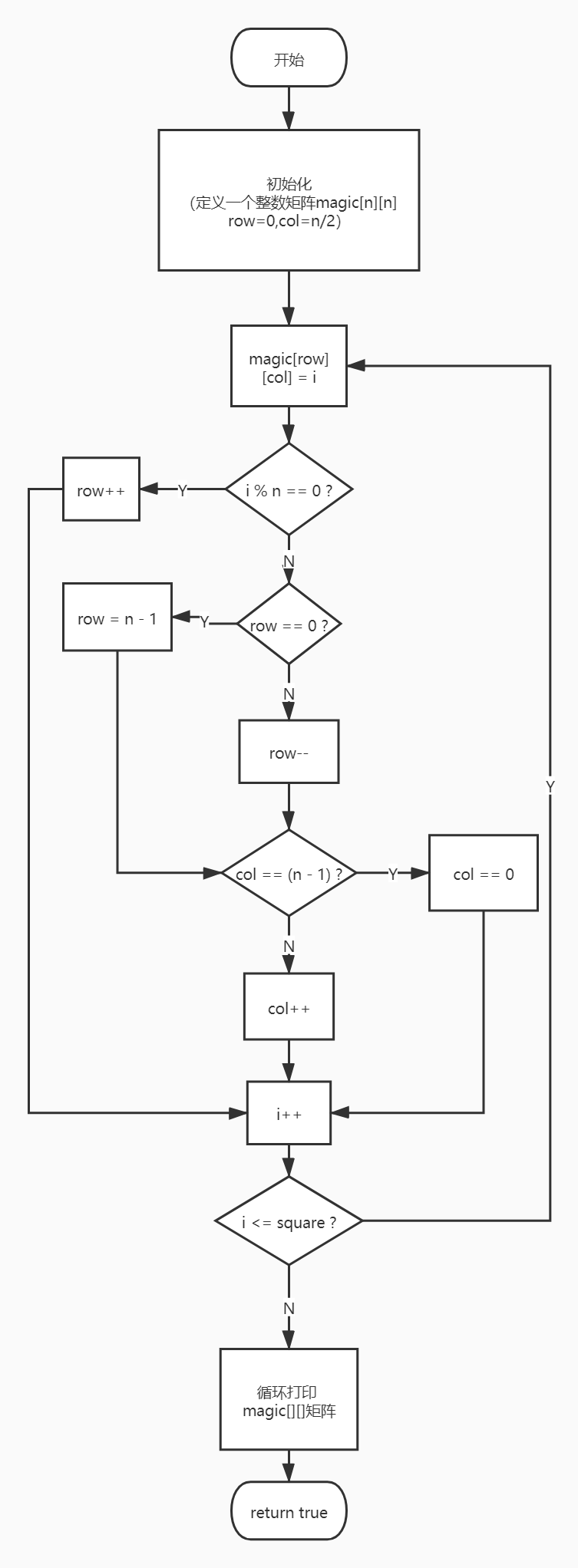






### generateMagicSquare()

程序流程图：



查阅JDK，解释异常：

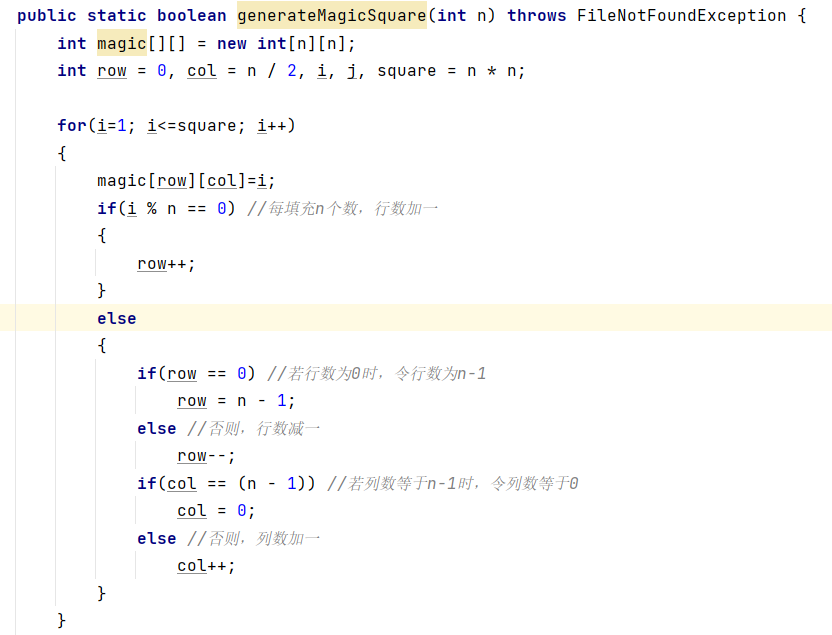
# java.lang.ArrayIndexOutOfBoundsException：数组越界

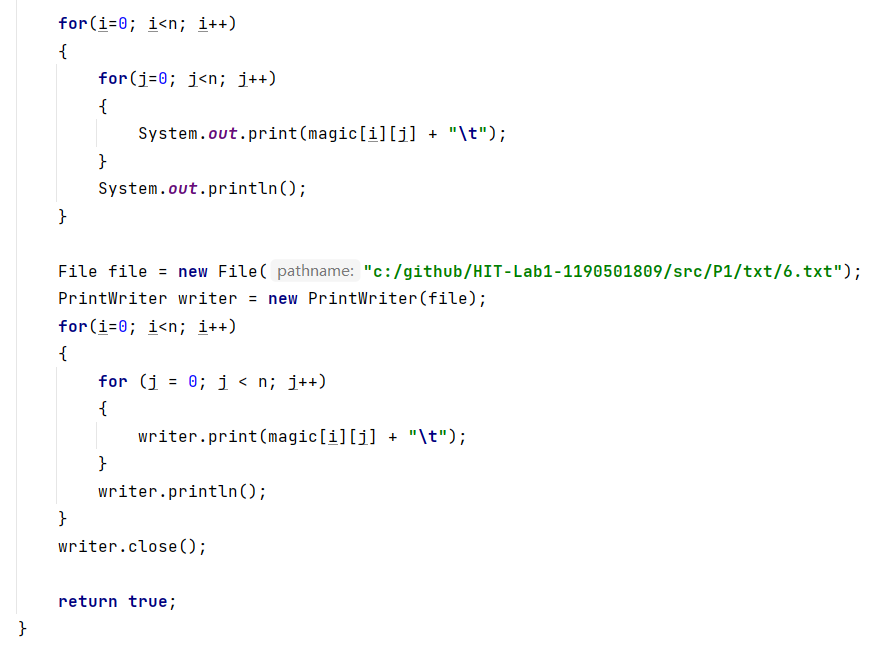
1. Java. lang. NegativeArraySizeException: 如果应用程序试图创建大小为负的数组，则抛出该异常

思路：

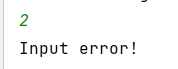
将产生的MagicSquare写入6.txt中，并调用isLegalMagicSquare进行判定。同时需要进行n的非法输入的处理。

实现过程：

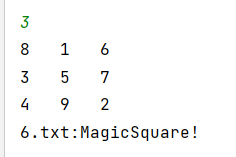




运行结果：







## Turtle Graphics

本次主要包含任务为：

1. 画一个正方形
2. 计算正多边形的内角
3. 由正多边形内角得到边数
4. 计算从一个点到另一个点转动的角度（顺时针）
5. 给定一系列点，计算每次转过的角度并返回在集合中
6. 计算一系列的凸包，调用函数绘制图形，使用junit进行单元测试

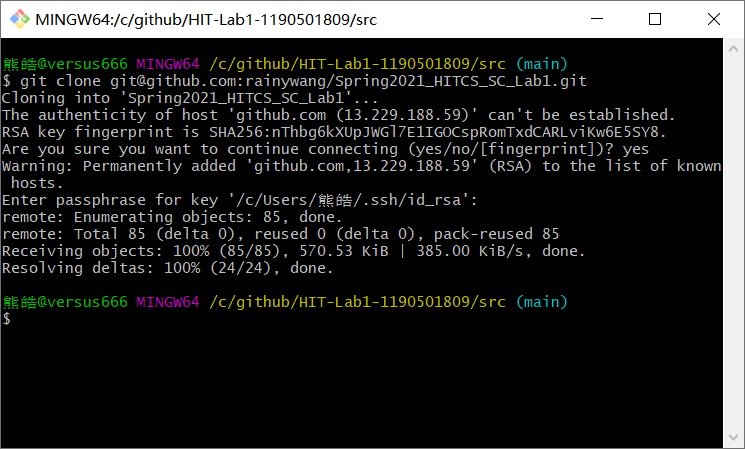
### Problem 1: Clone and import

打开目标存储文件夹，并点击Git Bash Here

在Git Bash中输入：

Git clone [git@github.com:rainywang/Spring2021\_HITCS\_SC\_Lab1.git](mailto:git@github.com:rainywang/Spring2021_HITCS_SC_Lab1.git)

将给定代码克隆至指定文件夹



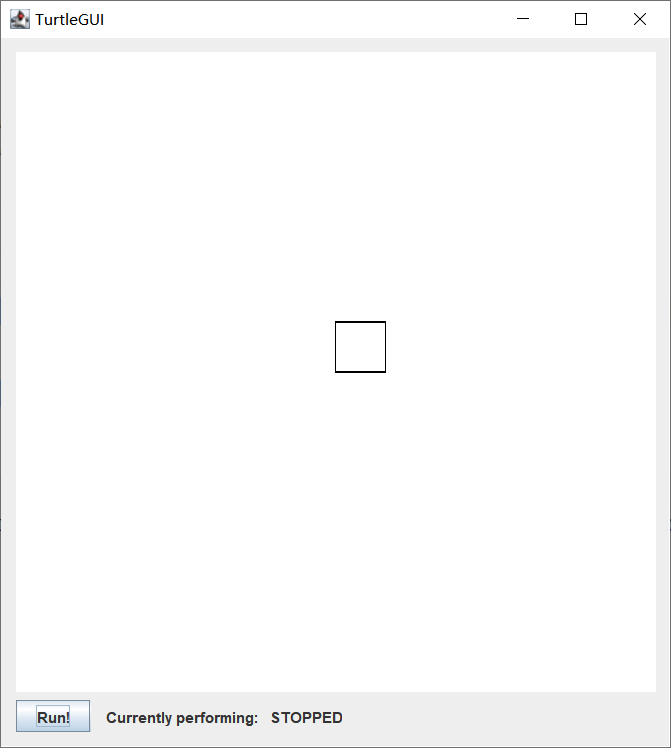
### Problem 3: Turtle graphics and drawSquare

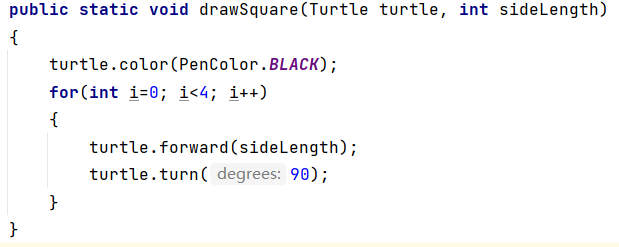
要求：

给定边长，使用forward和turn两种方法，面向turtle对象画出一个正方形。

实现过程：

设定画笔颜色为black，然后用一个循环依次画出正方形的四条边





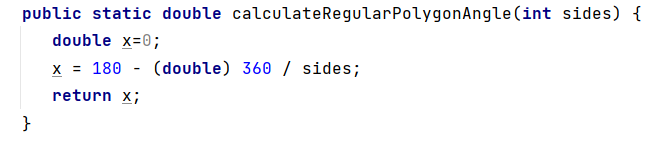
### Problem 5: Drawing polygons

要求：

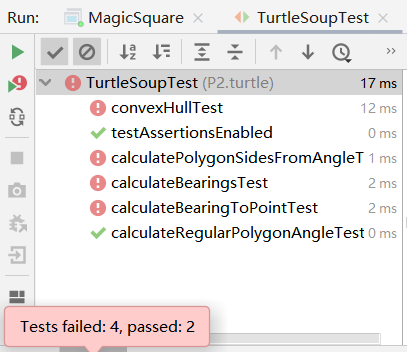
首先根据已知的正多边形的边数sides，然后推导内角度数：已知多边形的内角和为180\*(sides - 2)，然后除以边数sides，得公式为180 - 360 / sides，然后实现calculateRegularPolygonAngle函数，并运行test函数。

实现过程：

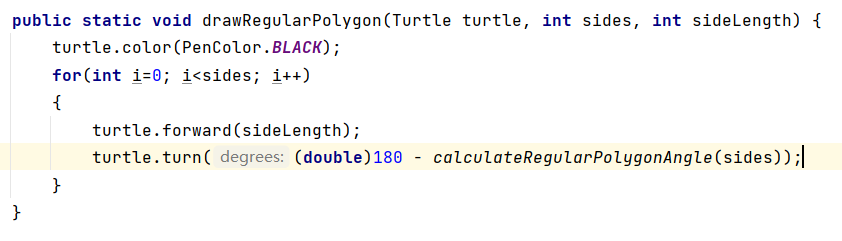
实现calculateRegularPolygonAngle函数为：

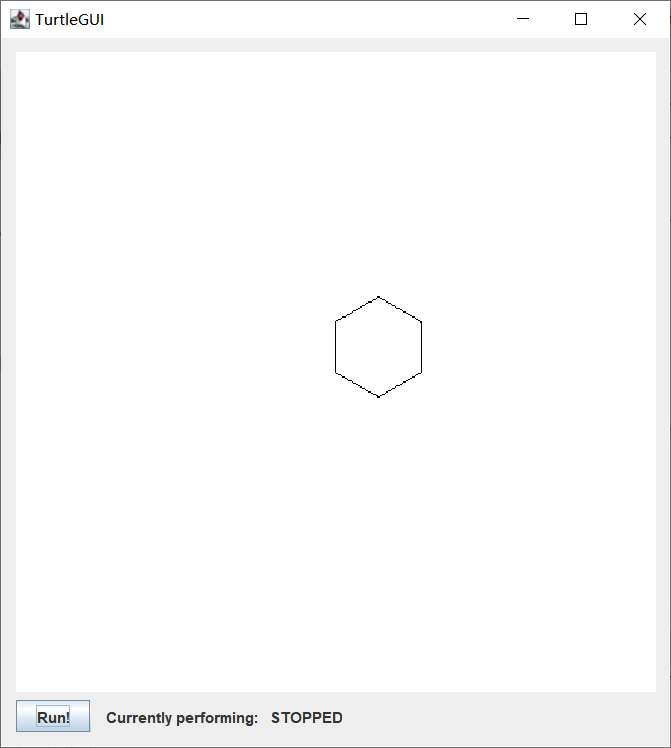


此时运行TurtleSoupTest中的Junit测试：



然后再完成drawRegularPolygon函数（本次写的是正六边形）：





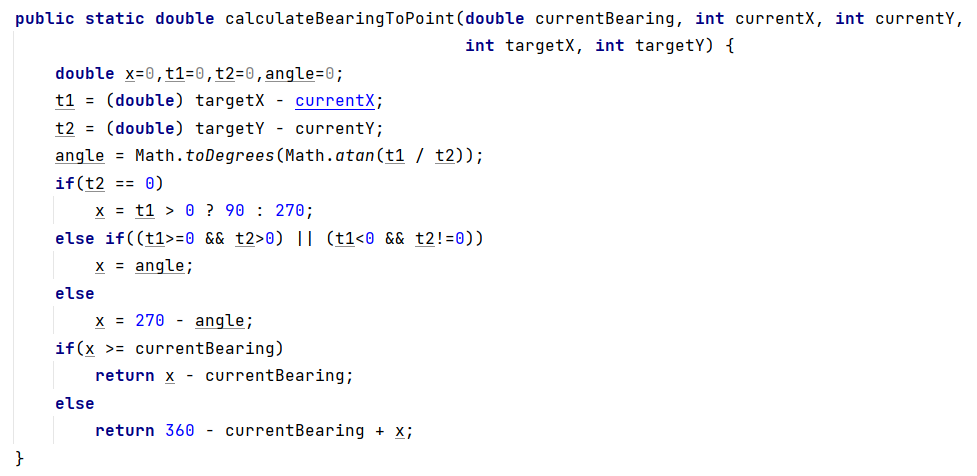
### Problem 6: Calculating Bearings

要求：

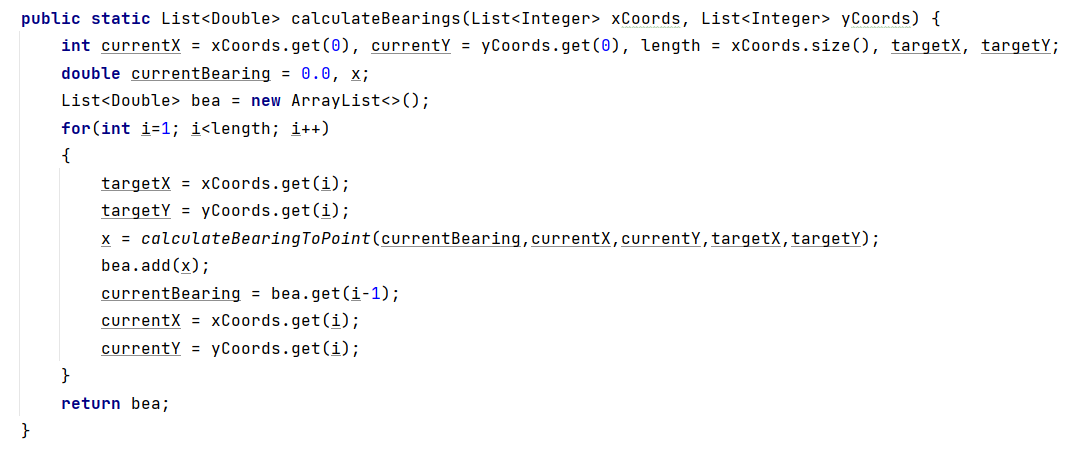
1. 通过给定起始点坐标、目标点坐标以及起始点的角度设计函数计算从起始点指向目标点需要转过的角度（顺时针）。
2. 给定两个数组，分别为x坐标、y坐标的数组，利用数组构成的点集，分别计算从初始点到目标点的转动角度（初始点与目标点按点集顺序后移）。

实现过程：

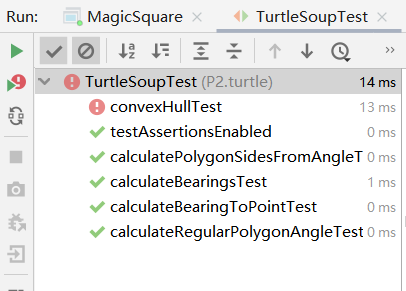
首先依据给定的起始点、目标点的坐标分坐标轴的四个象限来讨论计算出它们连线与north方向的夹角x（此时假设currentbearing为0），最后分两种情况：1.夹角角度大于起始角度：转动角度就等于夹角与起始角度的差值；2.夹角角度小于起始角度：转动角度就等于360.0与起始角度的差值再加上x。依据此原理，设计的calculateBearingToPoint函数：



然后来设计第二个函数，依据给定的两个集合，先取currentX、currentY的值为xCoords(0)、yCoords(0)和currentBearing为0，然后利用一个循环，依次取targetX、targetY,然后调用calculateBearingToPoint函数并将返回值存入一个double数组。每经过一次循环，都将currentX、currentY、targetX、targetY依次后移，并取上一次的结果作为下一次的currentBearing。依据此原理，设计的calculateBearings函数为：



同时利用Junit测试集进行测试：



### Problem 7: Convex Hulls

#### 凸包问题

凸包的顶点定义：

给定空间上的有限点集IMG_257，由点集中有限个极点（Extreme Point）构成的凸对象（Convex Object），就称为凸多胞体。极点就可以看成是多胞体的顶点，以二维多边形为例，极点就相当于凸多边形的顶点。使用严格的数学定义来说，极点是指那些不能由凸多胞体内的其它点的凸组合（Convex Combination）表示出来的点。给定点IMG_258共IMG_259个点，它的凸组合可以用下面的等式表示：

IMG_260

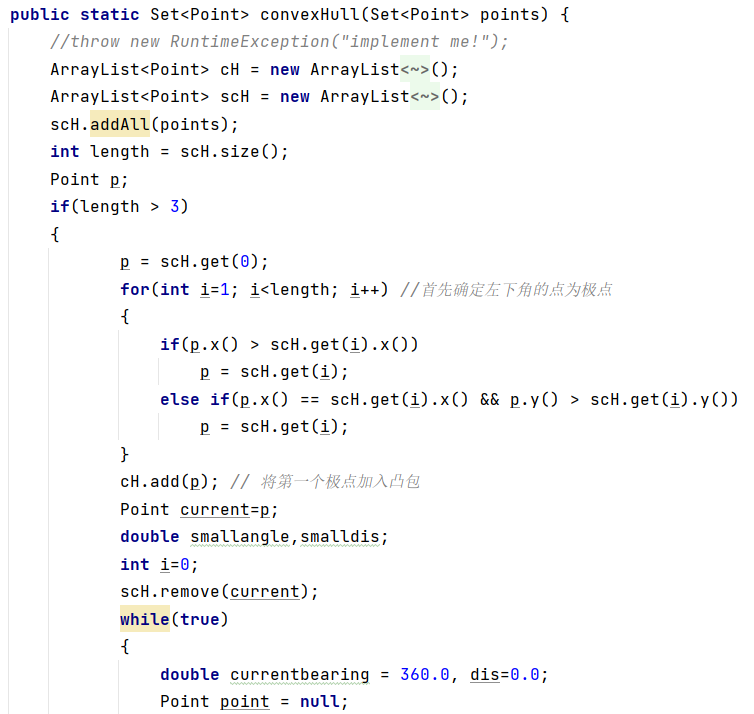
一条线段就是它的端点的所有的凸组合，三角形就是它的三个顶点的凸组合，一个四面体是它的四个顶点的凸组合。

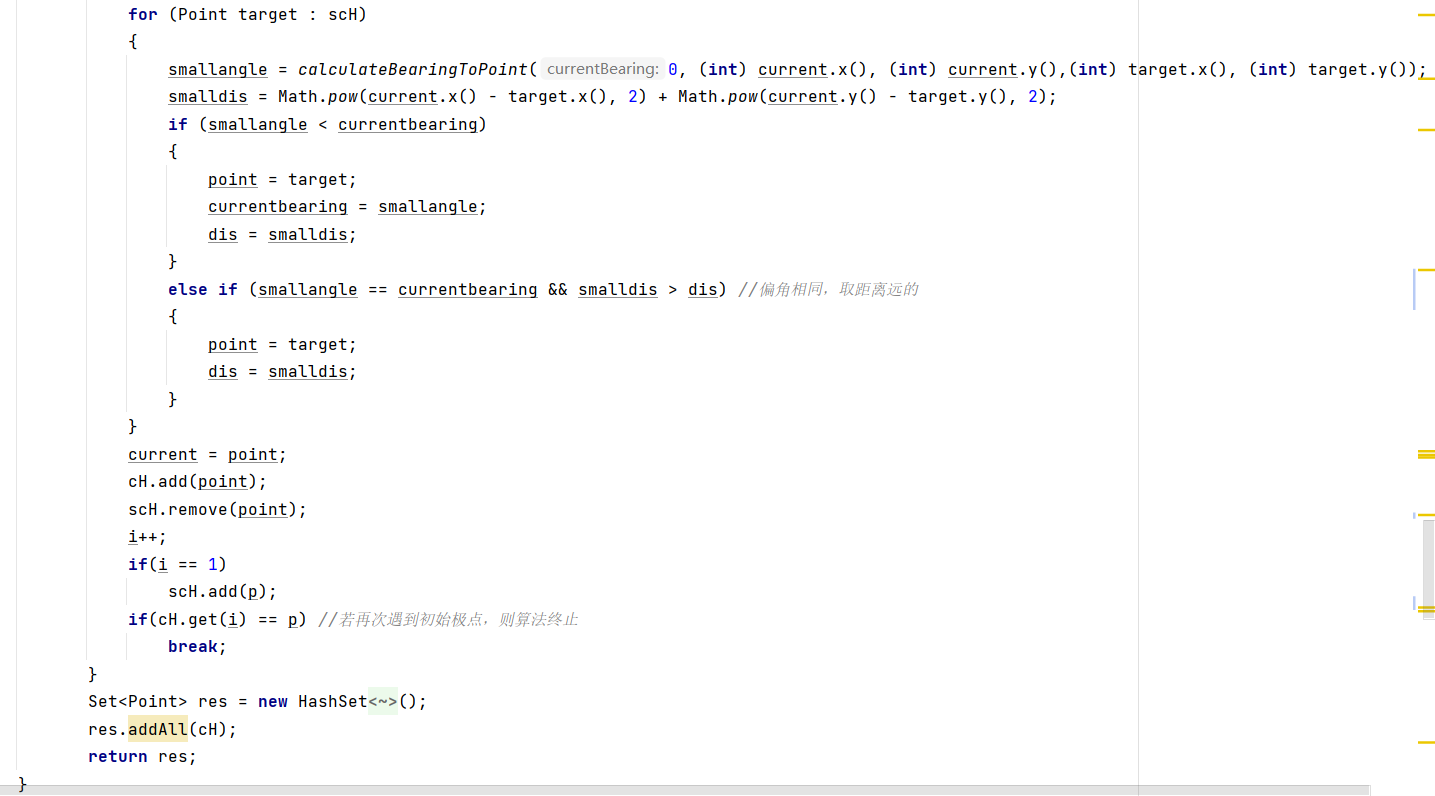
题中给定一组点，要求返回构成凸包周长顶点的输入点的最小子集。

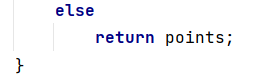
#### 算法描述

1. 找一个极点（左上角、左下角、右上角、右下角）
2. 遍历所有点，找出和极点偏转角度最小的（顺时针）
3. 将当前点作为下一个极点，继续找点
4. 直到找到第一个极点，算法结束

根据这个算法，完成的convexHull函数：



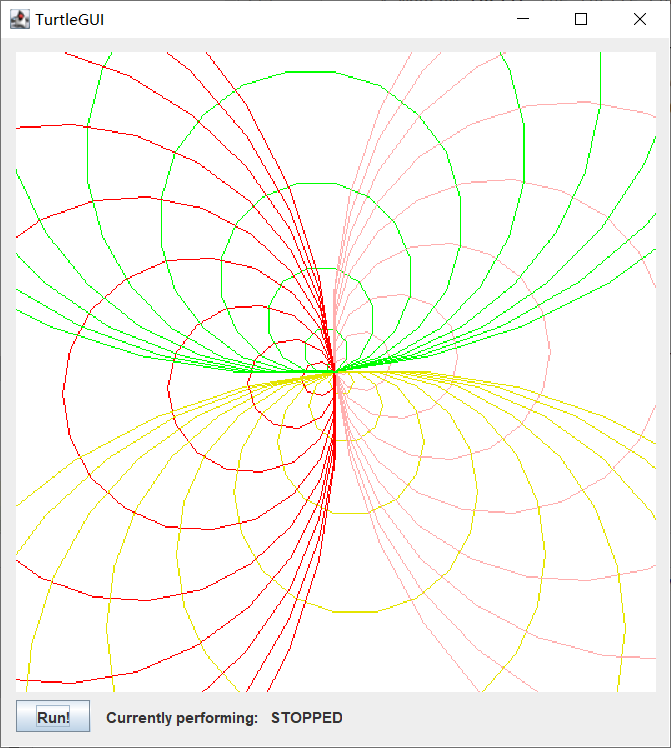




#### Junit测试

#### 

### Problem 8: Personal art



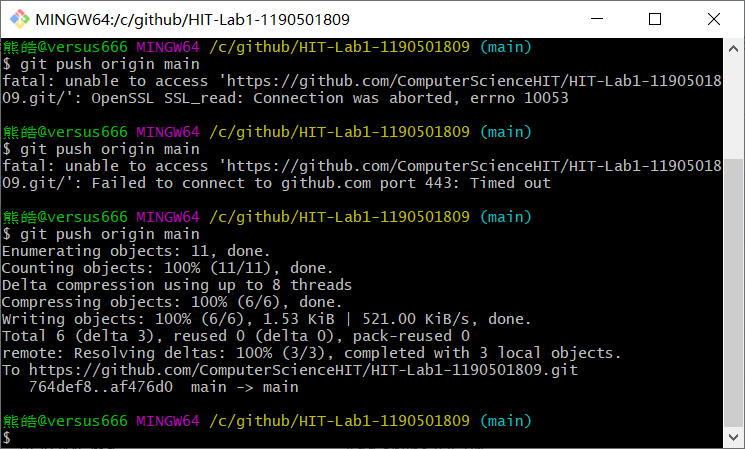
### Submitting

首先在本地库右键Git Bash Here，然后依次输入：

Git add src

Git commit -m””

Git push origin main



## Social Network

任务要求：

基于两个类FriendshipGraph和Person来设计一个社交网络图，每个人作为顶点，通过两个人的最短距离来反映两者之间的联系。

### 设计/实现FriendshipGraph类

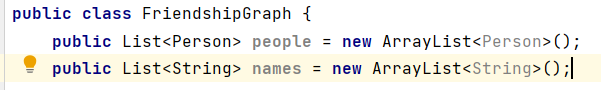
思路：

该类的意义是一张社交网络图，包括了增加每个Person的顶点addVertex()（若重复，则报错提示并结束程序）、增加每个顶点之间的有向边addEdge()、计算任意两个顶点之间最短距离的方法getDistance()（采用广度优先计算最短路径）。

实现过程：

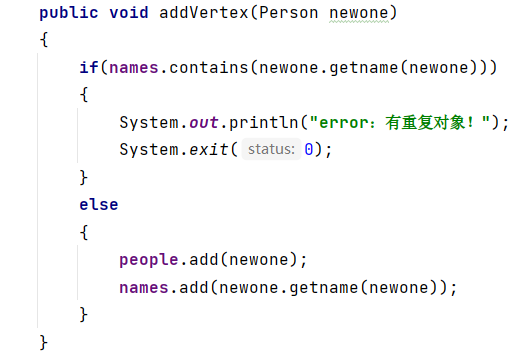
#### 3.3.1.1 存储结构

定义一个Person集合和一个String集合来分别存储所有的人对象和名字，并初始化。



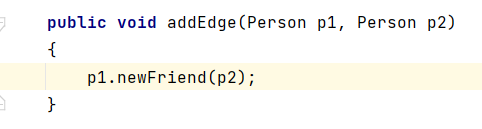
#### 3.3.1.2 method1: addVertex

该方法中添加顶点前，需要先判断names集合中是否有重复对象，若是，则提示错误并退出程序；若否，则将新的顶点加入people集合及将新的名字加入names集合。



#### 3.3.1.3 Method2: addEdge

对两个对象调用Person类中的newFriend方法。

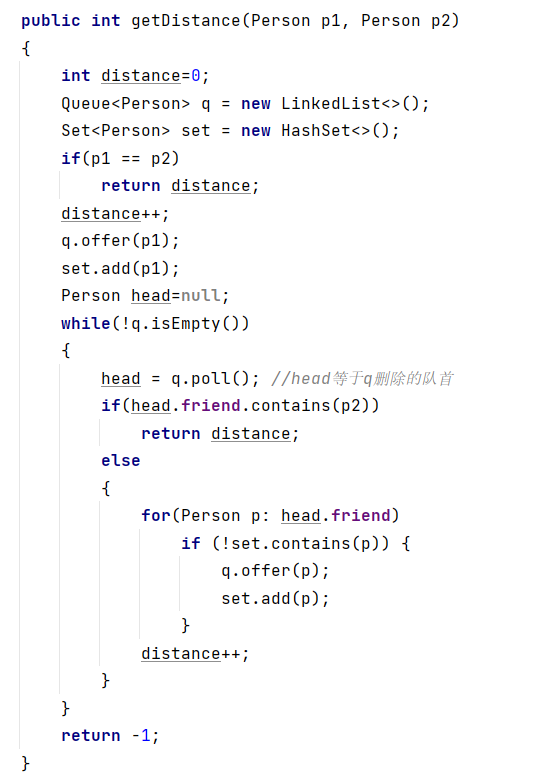


#### 3.3.1.4 Method3: getDistance

考虑到顶点集的数据可能较大，本方法中我们采用广度优先求最短路径：

首先考虑特殊情况，若两个对象重复，则返回0；若两个对象之间没有联系，则返回-1.

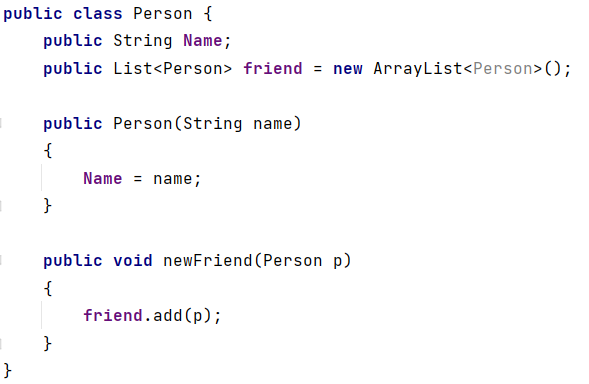
构造一个队列q和一个集合set，队列q用来存储广搜的遍历结果，集合set负责标记已经访问过的顶点。首先将第一个顶点入队，当队列非空时，弹出队首head，然后遍历head的friend列表，若有终点顶点，则返回distance；若没有，则将head的friend列表全部入队（前提是入队顶点未被访问过），依次循环（条件是队列非空）。



### 设计/实现Person类

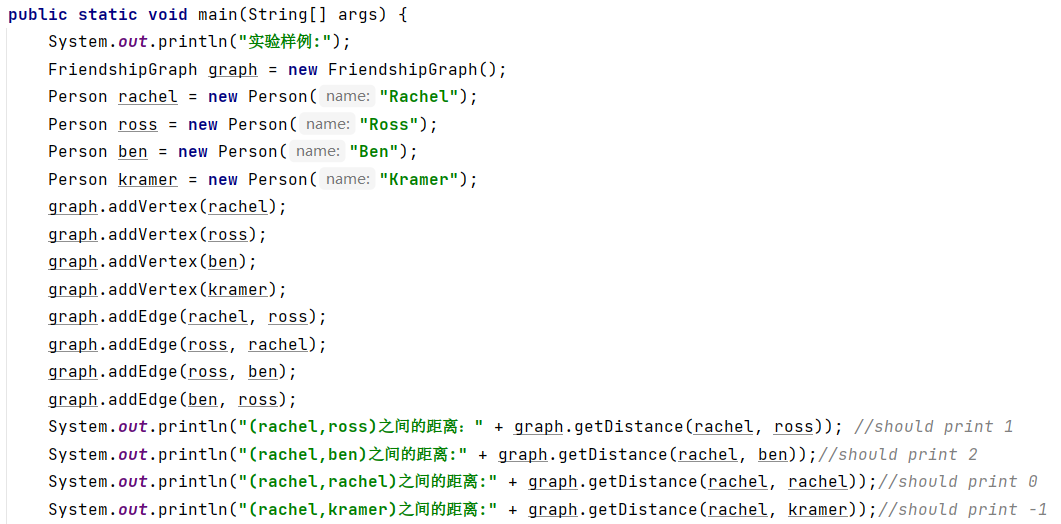
在Person类中设两个public量，Name（Person对象的名字）以及friend（Person对象的朋友列表）。

同时包含两个方法，Person(String name)：负责将输入的name存储到Person的Name中，以及newFriend(Person p)：负责将对象p存储在Person的friend列表中。



### 设计/实现客户端代码main()

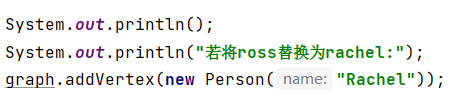
测试实验样例：



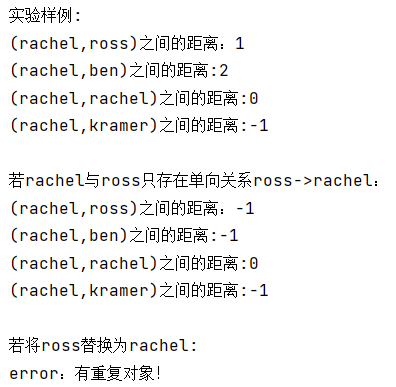
测试rachel与ross之间只存在单向关系：



测试有重复对象名时：



运行结果：



### 设计/实现测试用例

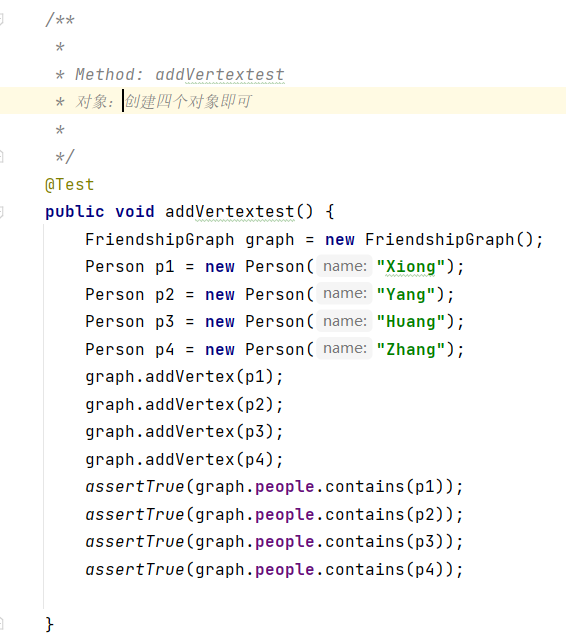
需要对addVertex、addEdge、getDistance三个方法进行测试。

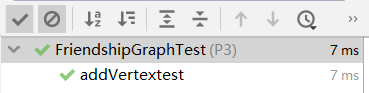
#### addVertextest

思路：

在addVertextest中建立四个Person对象全部执行addVertex，然后判断people中是否都包含这些对象。

实现过程：





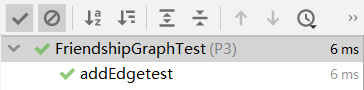
#### addEdgetest

思路：

建立四个Person对象，执行addVertex后，执行addEdge，然后判断每个对象的friend中是否包含彼此。

实现过程：





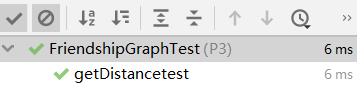
#### getDistancetest

思路：

建立四个Person对象，执行addVertex、addEdge后，保证三种距离类都存在（-1、0、正整数）。

实现过程：





# 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 |
| 2021-05-11 | 15:45-17:30 | 编写问题1的isLegalMagicSquare函数并进行测试 | 按时完成任务 |
| 2021-05-16 | 15:00-17:00 | 编写问题1的generateMagicSquare并绘制流程图、解释异常 | 延期2小时完成 |
| 2021-05-17 | 18:30-21:00 | 编写问题2的Problem1,3,5 | 按时完成任务 |
| 2021-05-18 | 15:45-17:30 | 编写问题2的Problem6,7,8 | 遇到困难，未全部完成 |
| 2021-05-20 | 18:00-22:00 | 编写问题2的Problem7,8 | 延期1小时完成 |
| 2021-5-21 | 9:00-12:00 | 编写问题3的FriendshipGraph，Person类以及main()函数 | 按时完成 |
| 2021-5-21 | 15:00-17:00 | 编写问题3的测试样例 | 按时完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的困难 | 解决途径 |
| 第一个遇到的问题是在问题1的generateMagicSquare中，题中要求画出程序流程图，但之前没画过正式的程序流程图，因此画这个图的时候非常拿不稳 | 到CSDN上学习标准的程序流程图的画法，同时查阅了较好的画此类的网站Procession. |
| 第二个问题是在问题2的Convex Hulls处。首先自己对凸包问题的定义模糊，对gift-wrapped algorithm的思想内核不熟。同时编程时因为涉及较多变量，一开始自己写的程序有很多bug，甚至包含几个死循环。 | 首先我查阅了有关凸包问题的定义，了解了这类问题的背景信息。然后我上CSDN查阅了有关gift-wrapped algorithm的思想。此时已经有了一个清晰的编程思路，然后我在草稿本上画出了有关convex hulls编程中各变量的关系与程序的大致流程图，使自己在编程的过程中避免了很多逻辑上的错误。最后再编程的时候就比较顺利了。 |
| 最后遇到问题是在问题3的getDistance方法中。因为这个方法中涉及使用先广求最短路径问题，然而我因为对java的不熟悉，导致自己不知道该如何选择存储结构和一些后续的编程实现。 | 通过与熟悉java的学长交流，我自己对这个编程有了一个较清晰的思路。然而在我第一次编程完成时，发现前面几类输出都满足（正整数、0），但一到-1时就出现了死循环。后来经过自己仔细审阅代码，发现自己忘记标记已访问的顶点，导致死循环。后来加上一个集合来记录已访问过的点，输出结果就OK了。 |

# 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

首先自己最大的收获是自己尝试并实践了java编程，虽然自己在java编程中还出现了很多c语言的影子。在拿到各类问题后，我发现自己能够独立清晰地分析题意，并且心中能有一个大致的框架与思想，并能在实践中实现自己的想法。尽管有时结果有bug，但经过一番debug，使自己更能适应使用java编程同时也有过一个较舒适的体验。但经过自己多次编程，我发现了java编程时必不可少的几点，首先要书写规约，同时尽量在一些代码进行中文注释，方便他人阅读以及自己后面的debug。另一个比较重要的点就是写测试用例，在P2、P3中，都有测试用例的涉及，同时我也深切感受到测试用例于一个程序的健壮性的重要性。

## 针对以下方面的感受

1. Java编程语言是否对你的口味？

java的很多类使用起来很方便，但我觉得稍有不方便的就是涉及各种包的引用。

1. 关于Eclipse IDE；

我觉得Eclipse IDE不是很适合我，一是操作不是很方便，同时界面稍显老套，我更倾向于使用IDEA。

1. 关于Git和GitHub；

Git与GitHub的操作最开始学的时候觉得很繁琐，但当自己上手后，就觉得GitHub这个平台是真的很方便。

1. 关于CMU和MIT的作业；

第一次阅读名校的作业来完成编程，一开始挺不适应全英文，但逐渐发现大部分设计的英文都较直观，阅读起来就感觉还好。而且当自己完成这些名校的作业后，心中会有一种舒适与满足感。

1. 关于本实验的工作量、难度、deadline；

本次实验的工作量我认为还是比较合适的，但主要发布的那一周我有考试与大创需要准备，因此本次实验我完成得就有点赶。难度的话，我认为大部分还是比较基础的，但也有比较有挑战性的（如Convex Hulls以及P3的getDistance）。Deadline还是比较合理的。

1. 关于初接触“软件构造”课程；

最开始对软构的印象就是JAVA（主要因为自己对java不熟），但真正接触后，发现其实软构也挺能接受的。自己可以接触更规范的编程方式以及编程思想，可以很有效地提高自己的代码设计与实践能力！