

**2020年春季学期  
计算机学院《软件构造》课程**

**Lab 1实验报告**

|  |  |
| --- | --- |
| 姓名 | 张乙 |
| 学号 | 1180301022 |
| 班号 | 1803010 |
| 电子邮件 | 1430268665@qq.com |
| 手机号码 | 15839452898 |

**目录**

[1 实验目标概述 1](#_Toc34569045)

[2 实验环境配置 1](#_Toc34569046)

[3 实验过程 2](#_Toc34569047)

[3.1 Magic Squares 2](#_Toc34569048)

[3.1.1 isLegalMagicSquare() 2](#_Toc34569049)

[3.1.2 generateMagicSquare() 4](#_Toc34569050)

[3.2 Turtle Graphics 6](#_Toc34569051)

[3.2.1 Problem 1: Clone and import 6](#_Toc34569052)

[3.2.2 Problem 3: Turtle graphics and drawSquare 7](#_Toc34569053)

[3.2.3 Problem 5: Drawing polygons 7](#_Toc34569054)

[3.2.4 Problem 6: Calculating Bearings 8](#_Toc34569055)

[3.2.5 Problem 7: Convex Hulls 9](#_Toc34569056)

[3.2.6 Problem 8: Personal art 11](#_Toc34569057)

[3.2.7 Submitting 12](#_Toc34569058)

[3.3 Social Network 13](#_Toc34569059)

[3.3.1 设计/实现FriendshipGraph类 13](#_Toc34569060)

[3.3.2 设计/实现Person类 15](#_Toc34569061)

[3.3.3 设计/实现客户端代码main() 15](#_Toc34569062)

[3.3.4 设计/实现测试用例 16](#_Toc34569063)

[4 实验进度记录 19](#_Toc34569064)

[5 实验过程中遇到的困难与解决途径 20](#_Toc34569065)

[6 实验过程中收获的经验、教训、感想 20](#_Toc34569066)

[6.1 实验过程中收获的经验和教训 20](#_Toc34569067)

[6.2 针对以下方面的感受 20](#_Toc34569068)

# 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

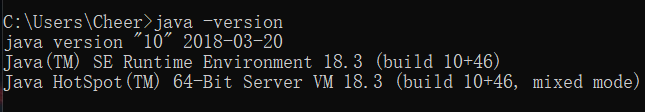
 基于 Eclipse IDE 进行 Java 编程

 基于 JUnit 的测试

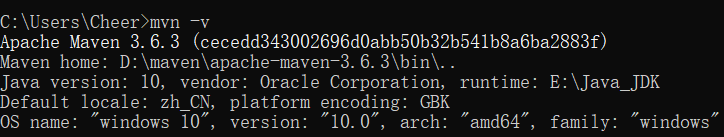
 基于 Git 的代码配置管理

# 实验环境配置

1.之前安装的JDK版本过低，重新下载了Java10并配置了环境变量。



2.我使用IntelliJ IDEA Community Edition 2019.3.3 x64进行本次实验。由于是第一次使用，我遇到了很多问题。其中主要问题是maven的配置。下载maven之后，需要修改settings.xml中的本地maven仓库位置和镜像网址（默认网址下载插件非常慢）。



3.Junit的配置问题。IntelliJ IDEA具有插件Junit Generator V2.0。在使用Junit4时，要通过修改pom.xml文件来导入外部库，之前忽视了这一点导致浪费了很多时间。

4. IntelliJ IDEA默认Java版本是Java1.5，尝试多种办法始终无法修改默认设置。最终也是通过修改pom.xml中的配置解决。下图是最终确定的pom.xml模板。



GitHub Lab1仓库的URL地址：

https://github.com/ComputerScienceHIT/Lab1-1180301022.git

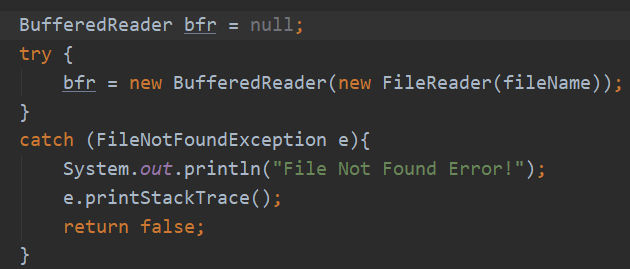
# 实验过程

## Magic Squares

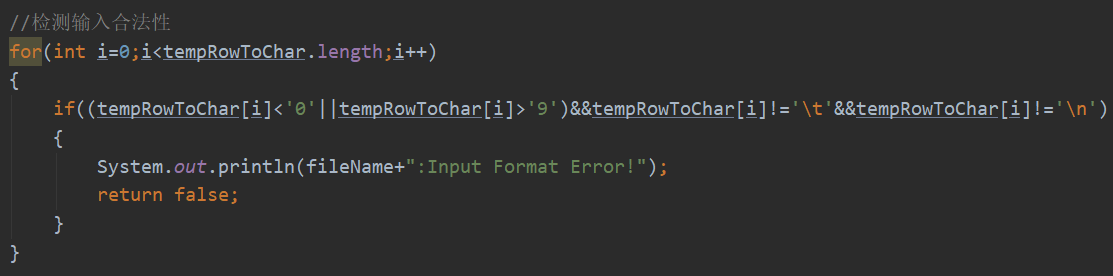
Magic Square是一些正整数组成的矩阵，其特点是每行、每列以及两条对角线上的元素之和都相等。给定的.txt文件中，同一行的数字使用制表符(\t)隔开，不同行用换行符(\n)隔开。实验的主要任务就是根据格式规则（按行）读入数据，进行格式判断，然后进行行、列、对角线的求和。若通过验证，则输入是Magic Square。产生Magic Square的代码已经给出。运用Java文件操作，将数组中的结果（如果输入可以产生合法Magic Square的话）写入目标文件6.txt中。

### isLegalMagicSquare()

1. 使用类java.io.BufferedReader的实例进行文件读取，其中的成员方法readLine()可以按行读取文件。



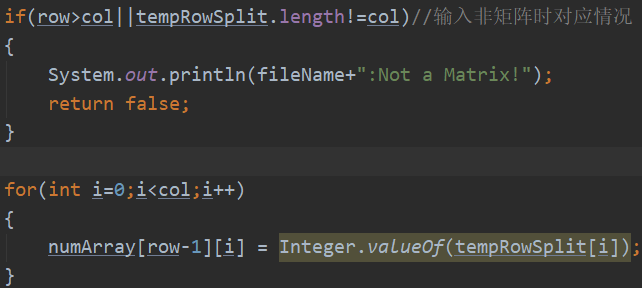
1. 创建读缓冲bfr后，开始逐行读取输入。对每行读入的String对象进行处理，即对每行进行一次循环。操作如下：将读入的String转化为字符型数组，遍历数组中的每个元素，检查是否有非法输入。



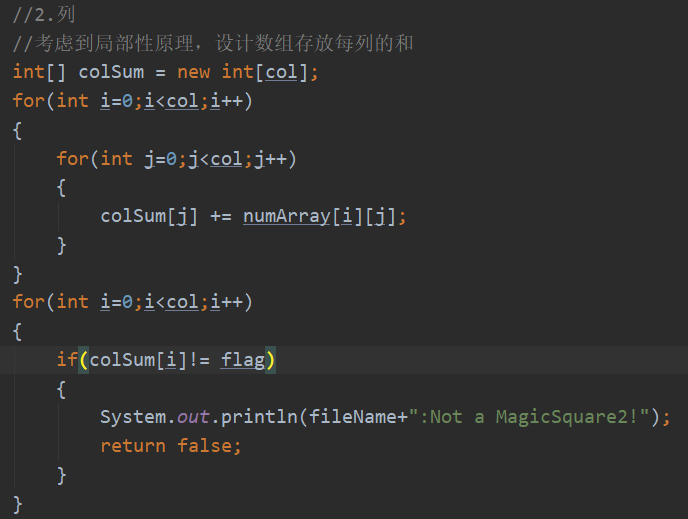
1. 输入合法后，利用String类的成员方法split()将该行对应的字符串划分为若干字符串，得到字符串数组tempRowSplit。如果当前行是第一行，则将tempRowSplit.length作为输入的维度（列数col），创建存放最终矩阵的二维数组numArray。



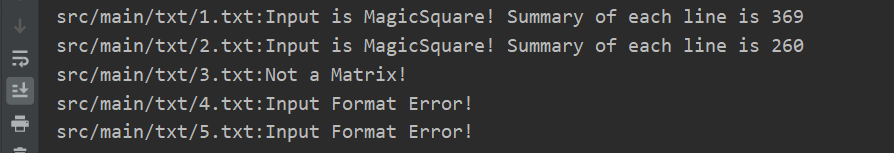
1. 每读取一行，记录行数的变量row++。每次循环时，检查输入是否满足矩阵行列元素相等的特性。满足条件时，将字符串数组的每个元素转换成整数值，存放到对应位置。



1. 最后一步，验证各行列以及对角线的元素和是否相等，如果相等，则输入满足Magic Square的条件。在求各列的元素和时，考虑到局部性原理，创建一个数组存放各列的和，一次仍然读入数组的一行。



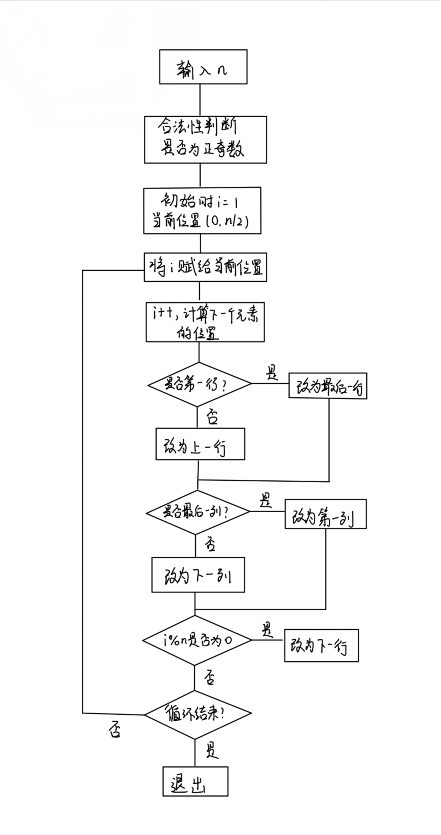
1. 实验结果：



注：地址是完成代码时测试用的，最后修改的地址满足所有题意。

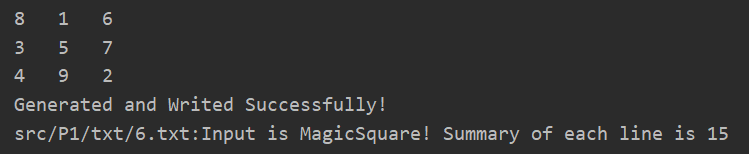
### generateMagicSquare()

generateMagicSquare()的总体流程如下：

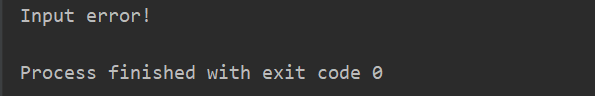


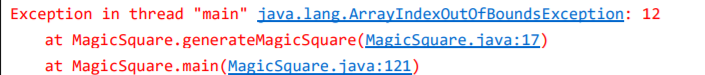
运行结果：

1.正确运行

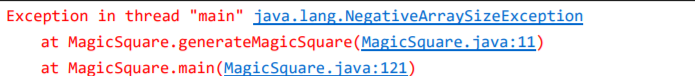


2.输入n=-1时（输入偶数同理，显示输入错误）





在MagicSquare.java的第17行和第121行出现了数组越界访问异常。当输入是偶数时出现该异常，其原因是没有考虑到输入的n为偶数的情况，i随着循环递增时，会出现超出数组界限的情况，导致异常。



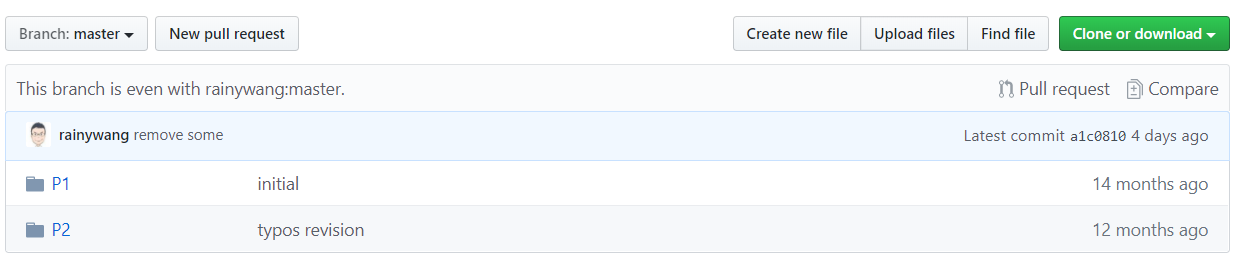
在MagicSquare.java的第11行和第121行出现了数组大小是负数的异常。输入n为负数时，在建立数组magic[n][n]时导致数组大小为负，显然是错误的。

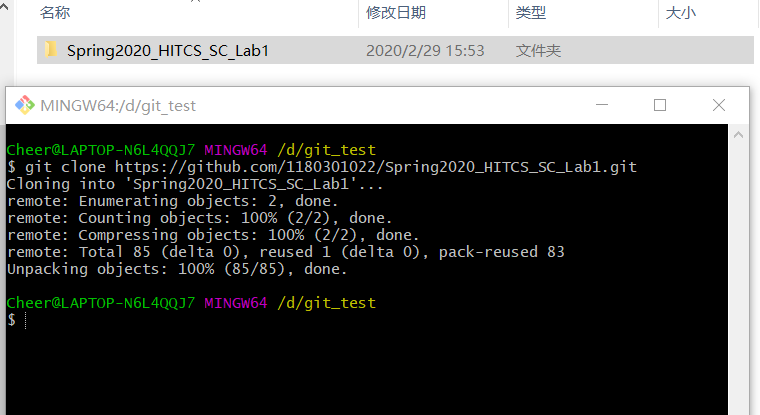
## Turtle Graphics

利用现有的类，由浅入深，完成一系列画图以及计算操作。在补全成员方法后使用Junit进行测试，保证方法及类编写的正确性，有效地提高了效率。同时涉及到一些常用类的使用（如Set和List），我通过查阅书籍和阅读博客学习了对这些类的使用。

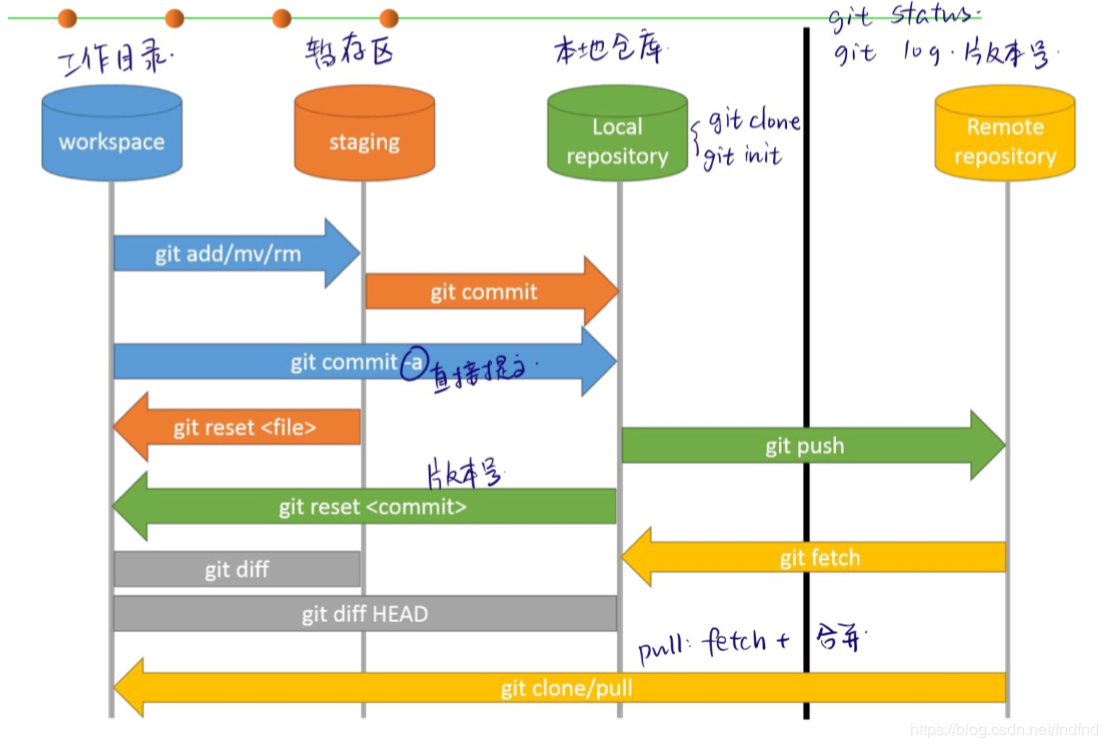
### Problem 1: Clone and import

获取代码：打开实验指导所给的GitHub链接，点击fork，再点击Clone or Download获取链接，在git bash中使用git clone命令下载远程库的文件。





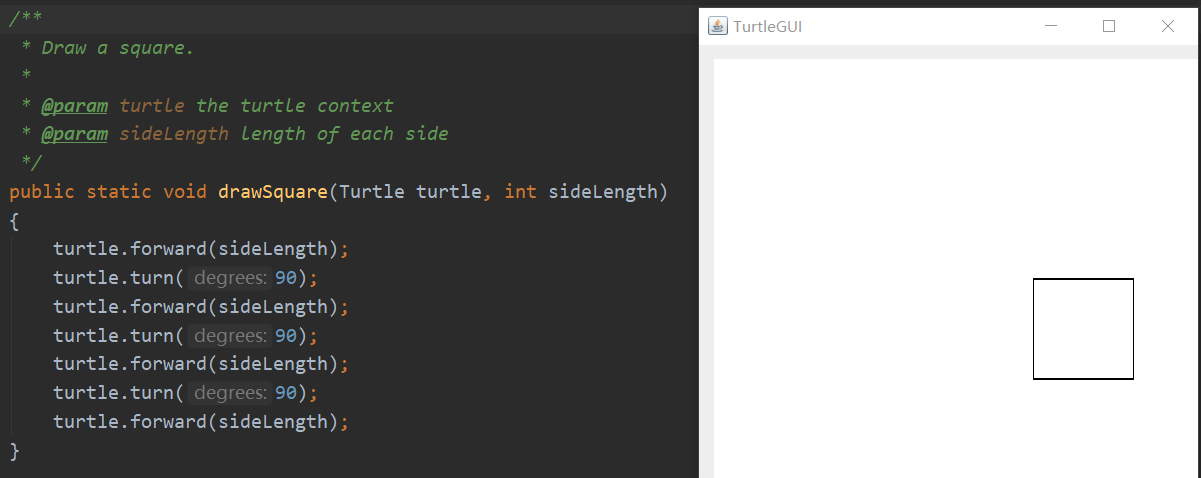
创建本地git仓库、使用git管理本地开发：在git bash中使用git init命令创建本地git仓库。通过SSH key将本地仓库和远程仓库链接。下图为我在博客中找到的git的常用命令。通过git pull或者git clone获取远程仓库的文件，通过git add、git commit、git push将本地仓库的文件上传到远程仓库。



### Problem 3: Turtle graphics and drawSquare

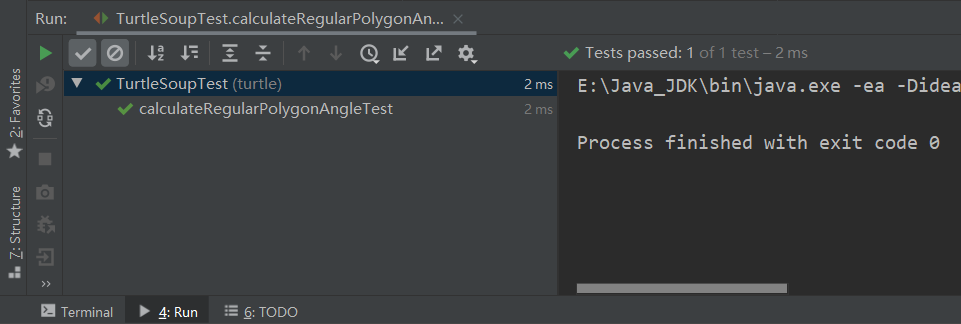
由任务说明可知默认方向向上，通过简单的画线（使用turtle.forward()）和转向(turtle.turn())方法即可完成。

代码及运行结果：

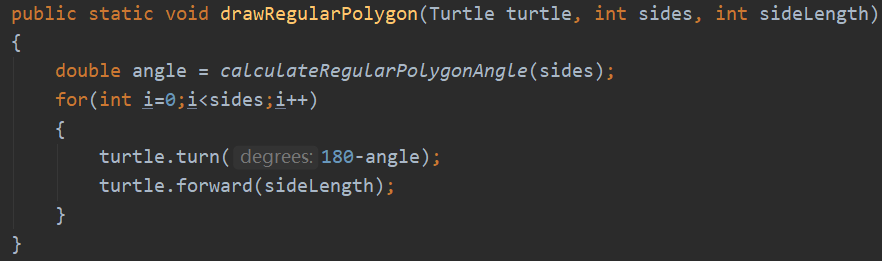


### Problem 5: Drawing polygons

首先通过规则多边形内角公式，补全方法calculateRegularPolygonAngle。该方法输入规则多边形边数，输出内角值。然后使用Junit对该方法进行测试。我使用IntelliJ IDEA Community Edition 2019.3.3 x64进行开发，运行测试方法的操作是在该方法名处点击右键->go to->test，运行测试方法即可。下图显示测试通过。

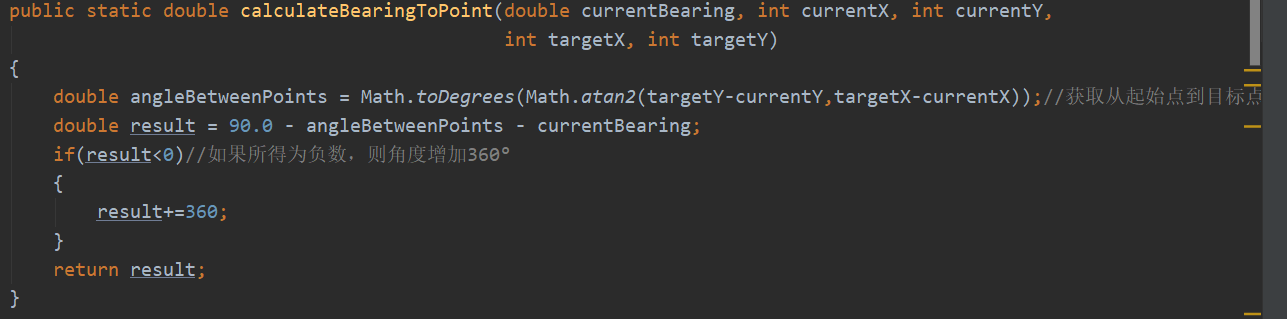


然后补全方法drawRegularPolygon。该方法同样非常简单，输入turtle、边数和边长，进行边长次数的循环，每次循环画出一条边，并将方向旋转内角的补角度数即可。

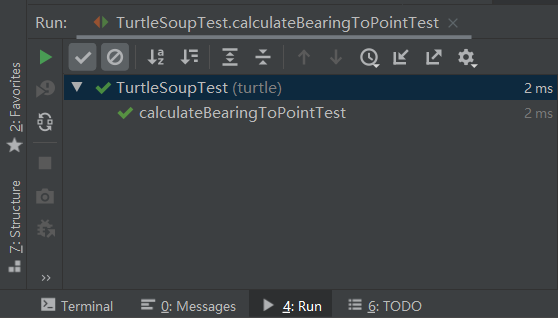


### Problem 6: Calculating Bearings

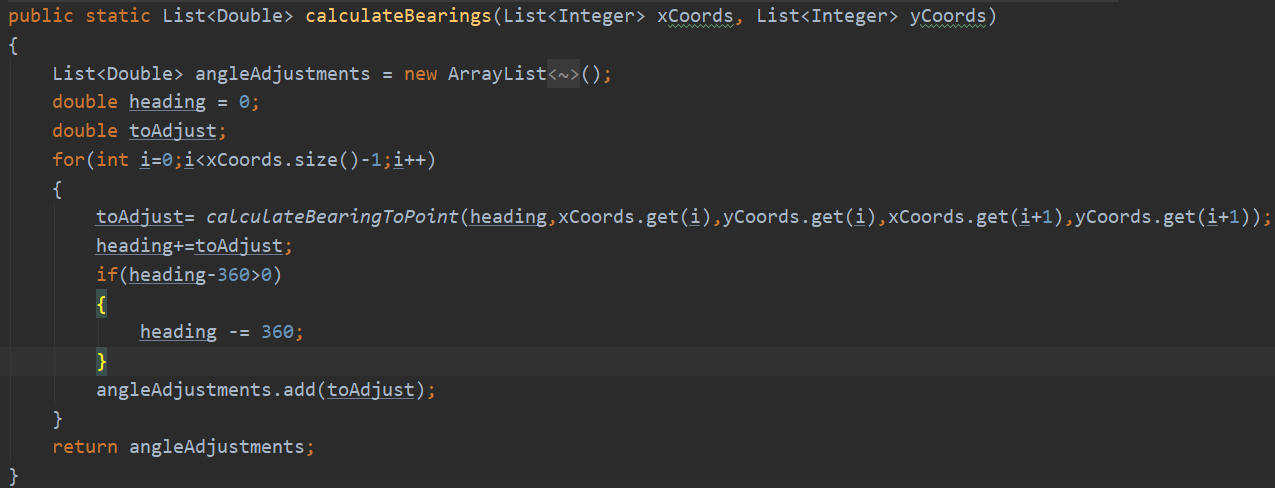
calculateBearingToPoint方法计算起始点沿顺时针方向相对原方向需要旋转多少度才能到达目标点。使用Math.atan2()求出夹角的值（注意该方法返回的是弧度值，要调用Math.toDegrees将其转化成角度值）。我在平面直角坐标系中利用几何知识推导出了公式，即下图中求result的语句。如果此时得到的result为负，则将其增加360°。



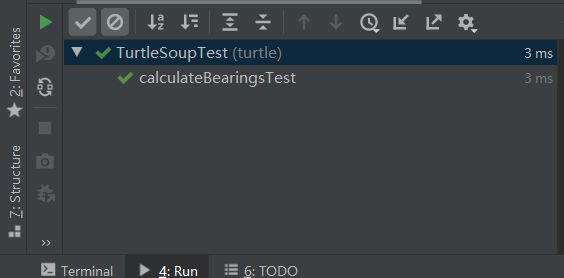
测试结果：



calculateBearings方法通过List输入一系列点，返回存放需要旋转角度的List。在该方法中，我使用变量heading存放当前边的方向，使用toAdjust作为临时变量存放需要旋转的角度值。每轮循环向输出List写入一个toAdjust值，遍历输入点的List。每轮循环中，如果heading的值大于360°，则减去360°作为当前的方向。

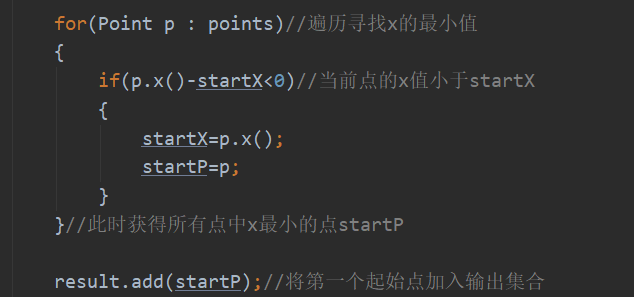


测试结果：

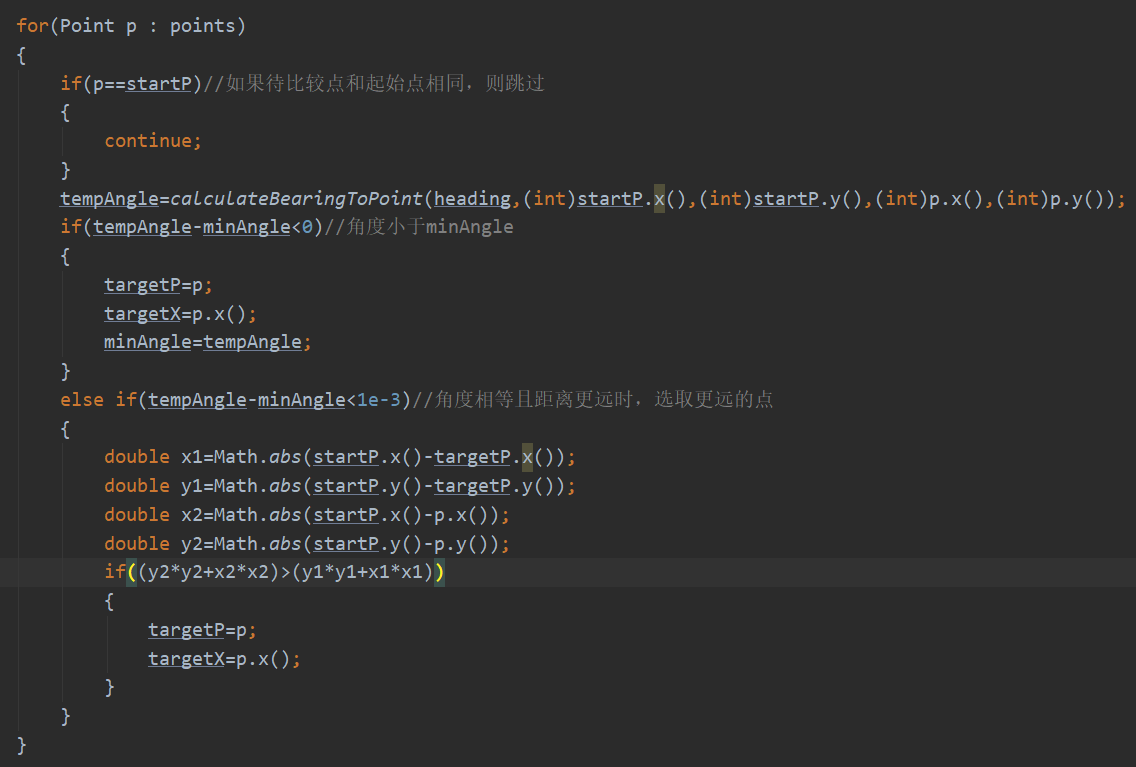


### Problem 7: Convex Hulls

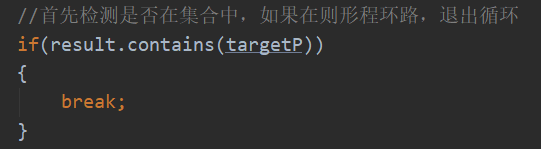
1. 选取横坐标最小的点为起始点



1. 从其余点中找到顺时针角度最小的点，每次循环遍历除自身外的所有点

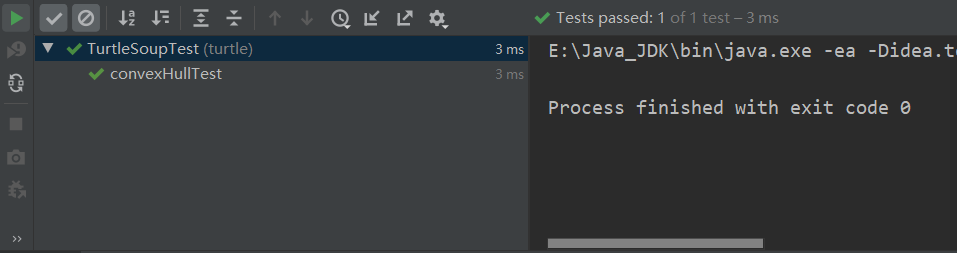


1. 标记该点为起始点，重复2，直到下一个点为已加入输出集合的点，结束程序。

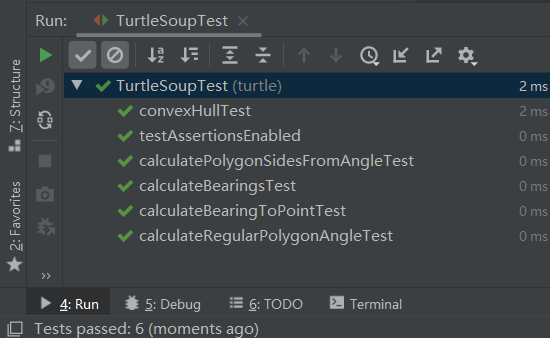


特殊情况：输入点数为0，1，2，3时，返回输入集合。多个点同时在一条边上时，创建并加入一个临时集合，在遍历完成后如果该集合的点满足角度最小，则选择边长最大的。

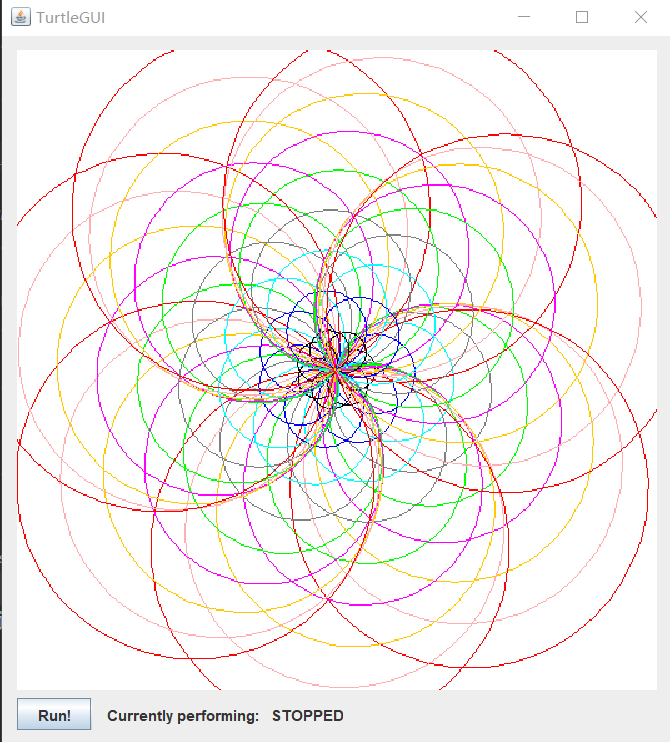
测试结果：



总体测试结果：



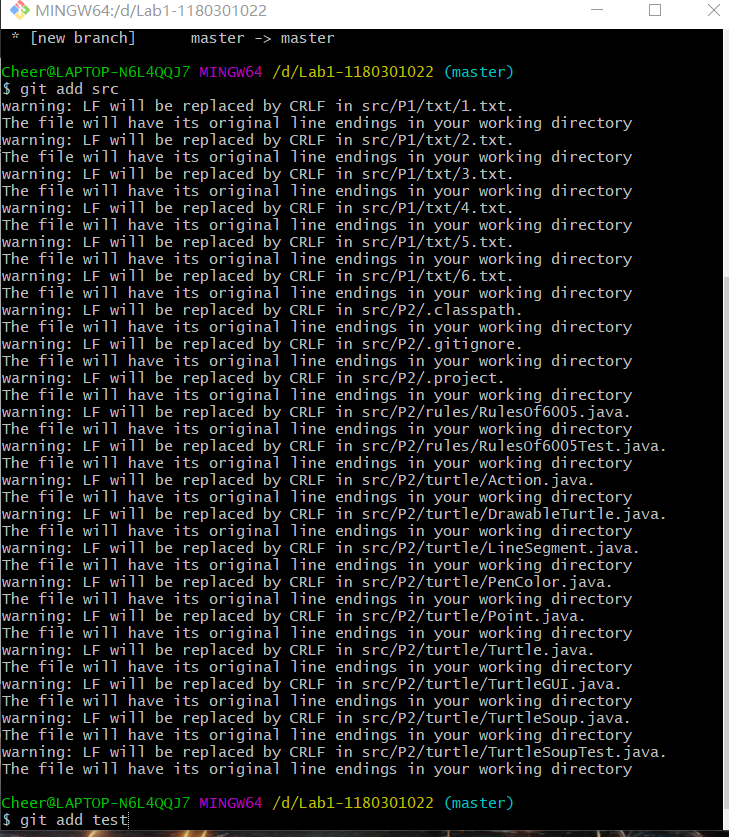
### Problem 8: Personal art

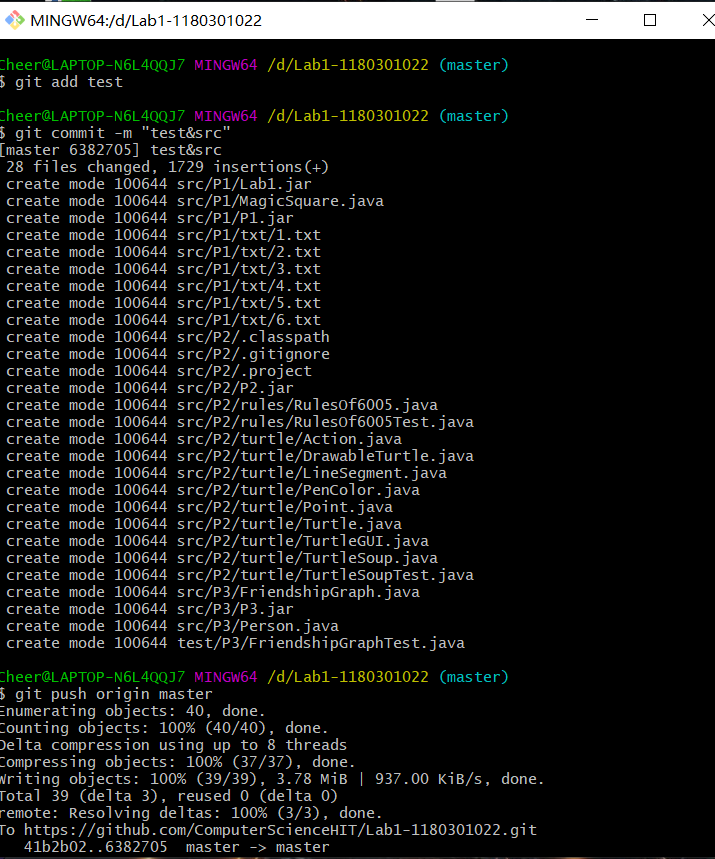


利用之前写的画多边形的方法，进行70次循环，以i为循环控制变量，每次循环都以i%10作为边的大小，旋转5个角度，则图像总共有7个起点。考虑到枚举中有十种颜色，利用switch语句，依次选取每种颜色作为每个多边形边的颜色。

### Submitting

在gitbash中使用git add,git commit,git push origin master命令上传。





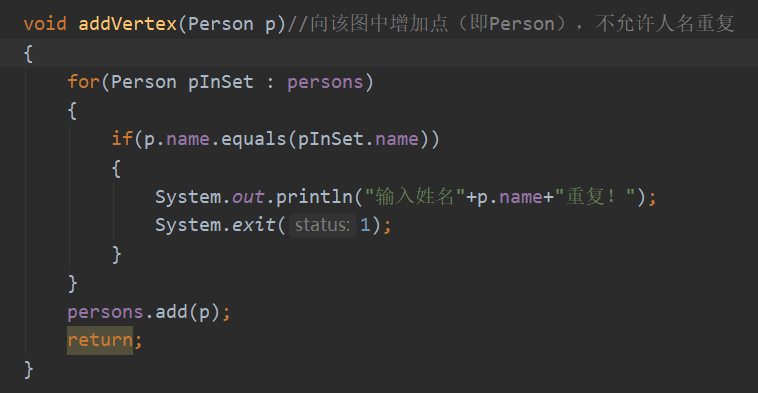
## Social Network

该任务的目标是创造一个无向图，将点（Person）按照输入要求连接起来（建立关系）。同时建立一个方法，该方法通过广度优先搜索计算并返回两点之间的最短距离。

### 设计/实现FriendshipGraph类

构造方法：在FriendshipGraph类中设置一个HashSet（标识符为persons），将加入该类的Person实例存放在集合中，以便重复姓名的检查。

addVertex方法：使用HashSet类中的contains方法，检测当前集合中是否含有该元素。在当前条件下，即检测是否重复输入人名。如果输入人名重复，则在控制台中输出错误提示并退出程序。



addEdge方法：根据题目要求，如果在两点之前添加一条边，需要调用两次addEdge方法。则调用一次addEdge方法只建立一条有向边。我在Person类中设置了向量类型的friends成员域，存放和该点直接连接的点。所以在addEdge方法中只需要将参数2加入到参数1的friends成员域中即可。

getDistance方法：

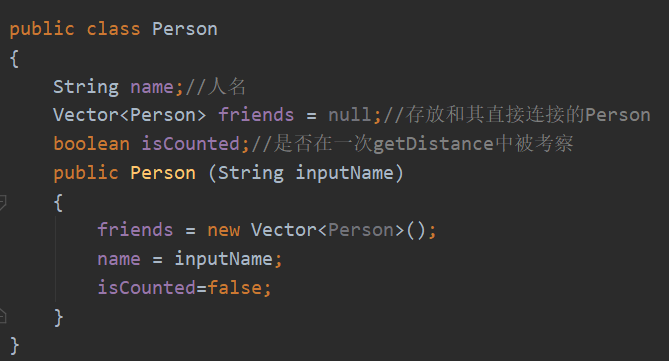
总体思路：如果输入的两个参数相同，直接返回0；输入两个参数不相同，则以Person类的成员域friends作为邻接表，利用队列进行广度优先搜索，如果可达返回最小距离，否则返回-1。

实现过程：将起始点入队，将该点标记为已考察，队列不为空时进行循环。在循环中，首先将队首元素出队，遍历其friends域的元素。如果发现目标点，则将persons集合中所有的点标记为未考察，输出结果。否则，如果元素考察过则直接进入下一次循环，如果没有考察过则将其加入队列，并标记为已考察。当队列为空时，起始点和目标点不可达，将persons集合中所有的点标记为未考察，输出-1。



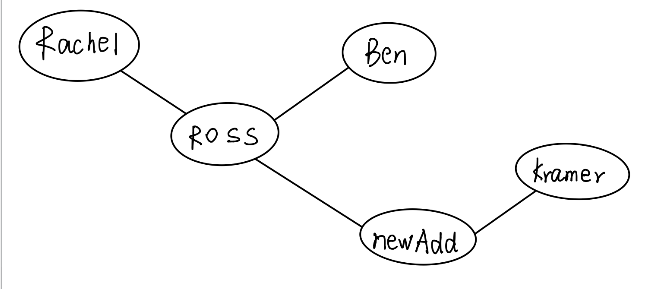
### 设计/实现Person类

为了方便FriendshipGraph中方法的实现，在Person类中设置了三个成员域：String name（存放姓名），Vector<Person> friends（存放和该点直接相连的点），boolean isCounted（标记在一次getDistance调用中，该点是否考察过）。

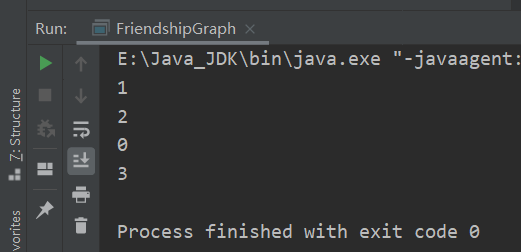


### 设计/实现客户端代码main()

在实验指导给的main代码的基础上，增加了点newAdd，分别连接点ross和kramer。



测试结果如下：



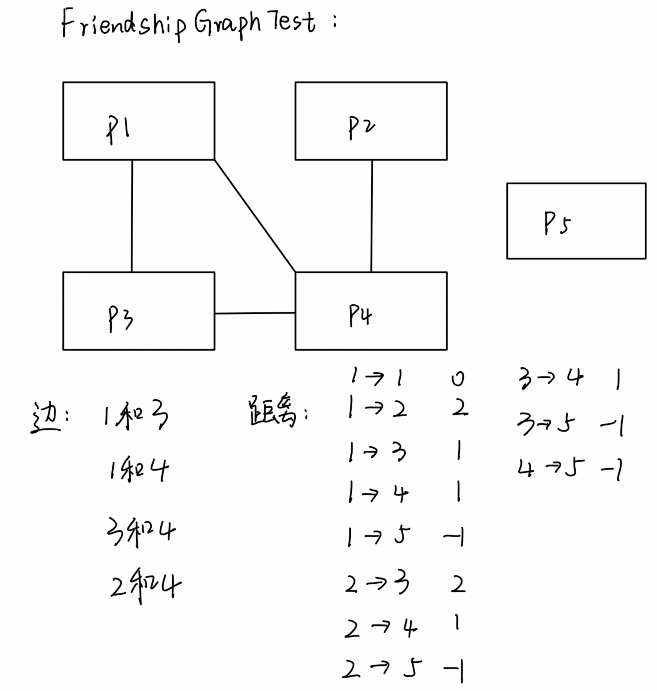
经验证，输出结果正确。

### 设计/实现测试用例

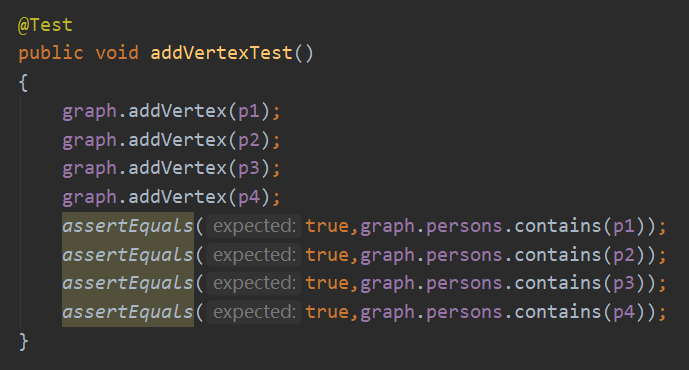
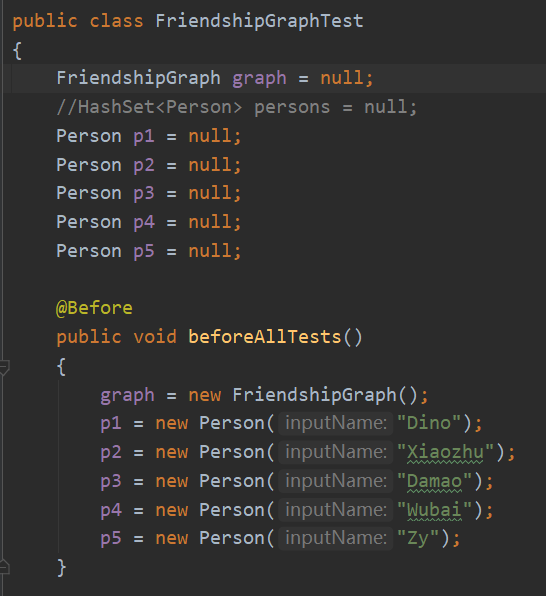
测试思路：

构建一个结构已知的图，用assert方法证明FriendshipGraph类中每个方法的正确性。

测试用例：



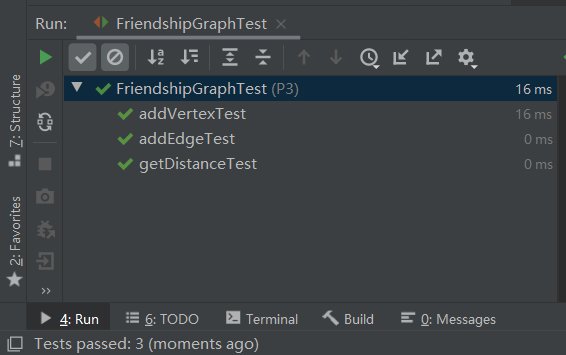
测试代码：



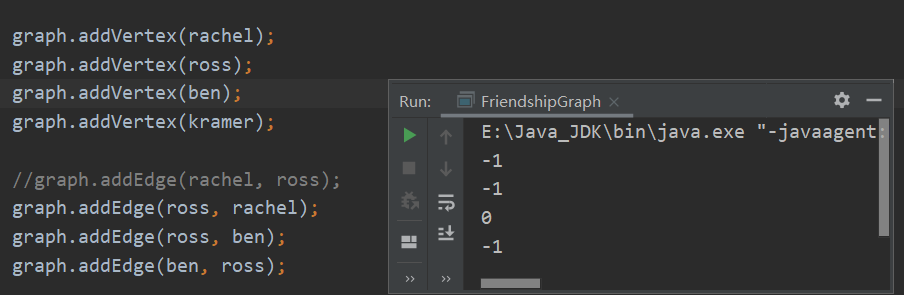




测试结果：

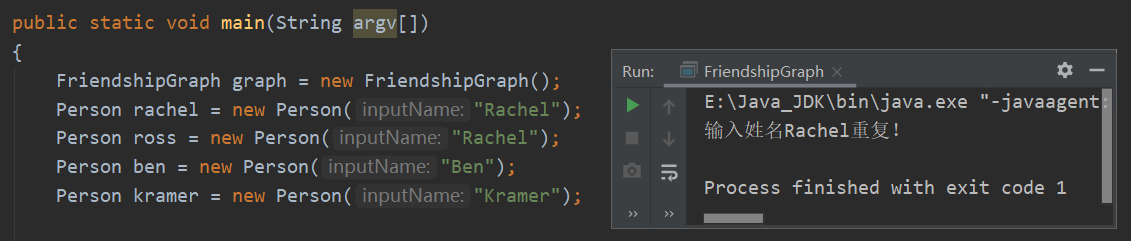


**思考题1**

****

和人工判断一致。

**思考题2**

****

如果输入重复的姓名，则出现错误信息并退出。

# 实验进度记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 任务 | 实际完成情况 |
| 2020-02-28 | 8:00-10:00 | 编写问题1的isLegalMagicSquare函数并进行测试 | 按计划完成 |
| 2020-02-28 | 10:00-10:30 | 补全问题1的generateMagicSquare方法 | 按计划完成 |
| 2020-02-29 | 8:30-11:30 | 完成问题2的环境配置，完成除convexHull的方法 | 延期大约半小时 |
| 2020-02-29 | 14:30-18:00 | 编写convexHull方法，但是测试没有通过 | 没有完成 |
| 2020-03-01 | 8:45-11:00 | 完成问题3 | 提前完成 |
| 2020-03-03 | 10:00-12:00 | Debug问题2的convexHull方法 | 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 |
| 不了解GitHub及其使用方法，不会建立本地仓库 | 通过搜索并阅读相关博客，按步骤完成了本地仓库和SSH密钥配置，建立本地和远程的连接。 |
| convexHull方法测试时遇到错误。单独运行测试方法时可以通过，但是运行测试类会报错。 | 在piazza上提出问题，在老师和同学的帮助下，发现是源码有些地方不够严谨，修改后问题解决。 |

# 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

通过这次实验，我的Java编程能力有了一定提高。同时意识到，光是寒假对着书本学习Java是不够的，只有亲自动手操作才能暴露真正的问题。我还对IDEA的各种操作更加熟悉，通过查阅相关资料解决了Junit、JDK、maven的配置问题，同时掌握了IDEA的debug方法。通过问题2和问题3，我学会了使用Junit进行测试以及写测试代码。

## 针对以下方面的感受

1. Java编程语言是否对你的口味？

是的，Java的库相比较C语言来说更丰富，用起来更方便。

1. 关于Eclipse IDE

去年暑假学Java的时候用的是Eclipse，用起来不差。这次实验我用的是IDEA，安装插件非常方便，代码补齐也很智能。

1. 关于Git和GitHub

第一次接触GitHub让我觉得有些吃力。本次实验让我了解到它是一个很好的版本控制工具和代码仓库。通过后续实验我想多探索一些它的功能。

1. 关于CMU和MIT的作业

英文的题目有些不适应。不过这些作业做起来难度适中，我觉得很好。

1. 关于本实验的工作量、难度、deadline

很好。

1. 关于初接触“软件构造”课程

软件构造是一门基础而核心的课程。这门课程偏向于应用，提倡用现有的工具而不是手工来完成任务。我希望通过这门课程的学习，提高自己开发和测试软件的水平，缩小学习和实际应用的差距。