

**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 | 张瑞豪 |
| 学号 | 1180800811 |
| 班号 | 1803002 |
| 电子邮件 | m17779349381@163.com |
| 手机号码 | 17779349381 |

**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

实验通过求解四个问题，训练基本的Java编程技能，能够利用Java OO开发基本的功能模块，阅读并补全代码，理解并测试程序。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

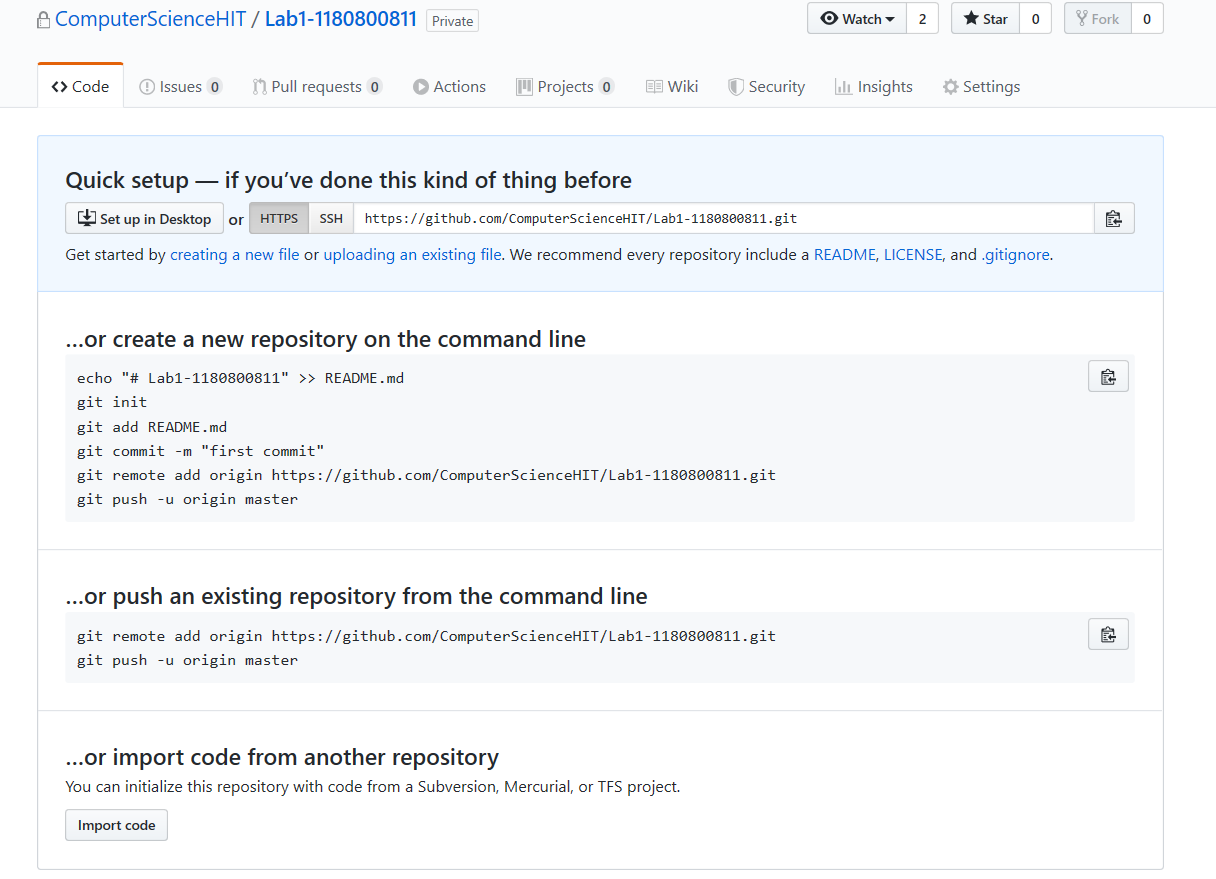
基于Git的代码配置管理

# 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. 由于在寒假前已经完成了jdk以及eclipse的安装、GitHub账号的申请以及git的下载和配置。所以此次实验只需加入GitHub classroom，建立第一次实验的仓库即可。



**2.遇到的困难：**由于我都是参考网上的教程来安装git、eclipse等环境，并未遇到特别显著的困难。

3.URL地址：https://github.com/ComputerScienceHIT/Lab1-1180800811

在这里给出你的GitHub Lab1仓库的URL地址（Lab1-学号）。

# 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

第一个实验应该是比较简单的，无非就是考察我们的java水平。一个简单的问题用java语言如何实现罢了。但是这个问题需要考虑很多的细节，如果没有很好的掌握java的一些类库的方法可能会有点迷。我觉得无非以下一些问题需要解决：

①java语言如何对文件进行读写操作并记录文件中的信息？

②怎么处理读取文件时出现的异常？

③怎么将文件中的矩阵转储到一个二维数组中？

④在判断矩阵是否为magic square的时候，有哪些特殊的情况是一定可以否定这个矩阵不是magic square？

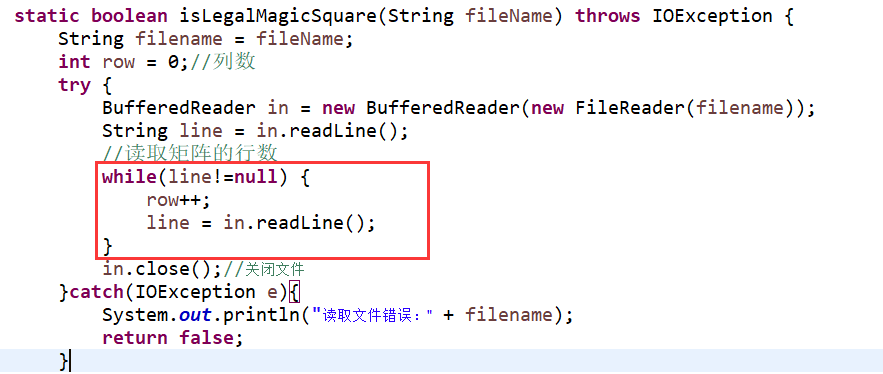
⑤按照常规操作判断矩阵是否为magic square。

### isLegalMagicSquare()

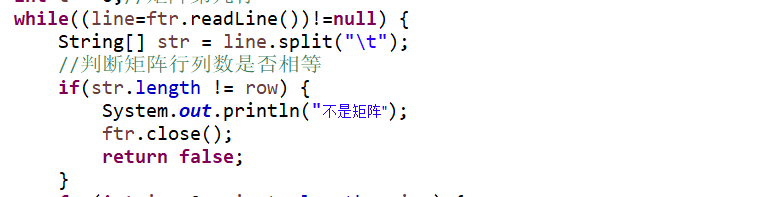
设计和实现思路：主要分成两个过程：首先把文件中的信息正确的读出来：矩阵的行数和列数、矩阵的每个元素，并且要进行异常处理：抛出异常或者捕获异常。其次就来判断矩阵是不是幻方矩阵，在判断的时候首先要判断矩阵是否满足幻方矩阵的要求：每个元素都是正整数(也就是说要排除矩阵元素为负数或者矩阵元素为小数的情况)。然后再判断矩阵的每行和每列以及对角线的元素之和是否相等。

实现过程：

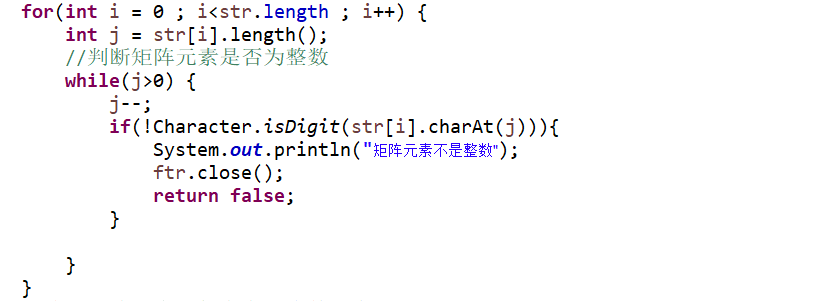
①首先读取文件中的信息，先每行每行的读取，由此可以确定矩阵的列数。同时应该注意对读取文件时的异常处理，我选择了捕获异常并且输出提示信息。



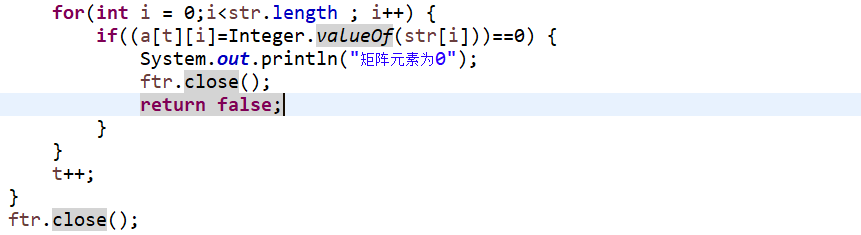
②然后把每行读取的字符串利用split函数分割为每一列。在这一步骤中可以判断矩阵的行数是否等于列数。



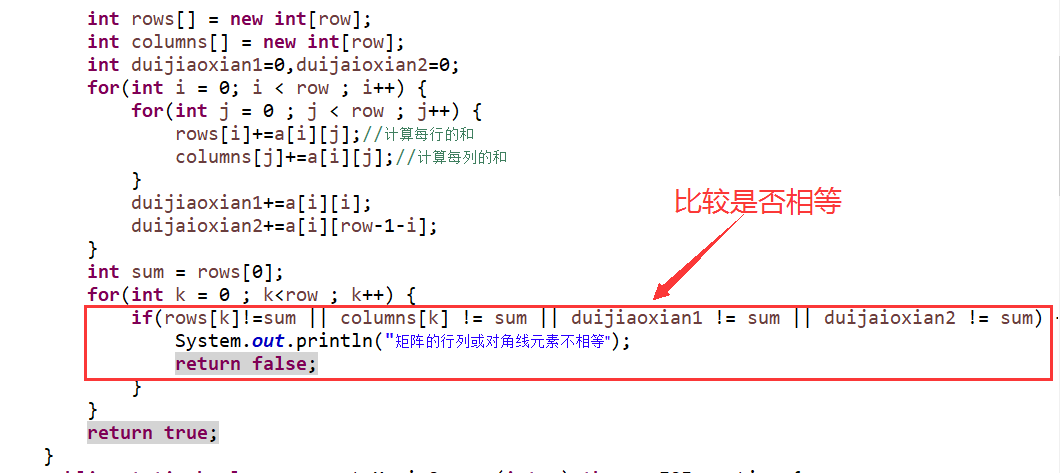
③当把每一行的字符串分割为每一列之后，就可以来读取矩阵的元素了。可以利用Integer容器类的valueOf方法，直接把读取的字符串转变为整数。在这个过程之前可以先判断矩阵的每一个元素是否为小数或者负数。方法是一次扫描字符串的每个字符，判断是否每个字符都是数字(若为小数则有”.”,若为负数则有”-”。)



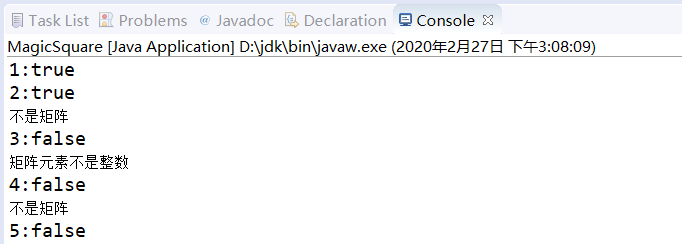
④当矩阵的每一个元素都是满足要求之后，就可以读出矩阵元素了。



⑤剩下的事情就是将每一行、每一列的、对角线的元素之和求出来再判断是否相等。



结果测试：



### generateMagicSquare()

这个函数就是来构造一个奇数阶的幻方矩阵。

该函数的思路:

①从第一行的正中间开始一次写入1-square这square个数，其中square=n\*n

②每次写下一个数的时候都是写在当前数的右上角

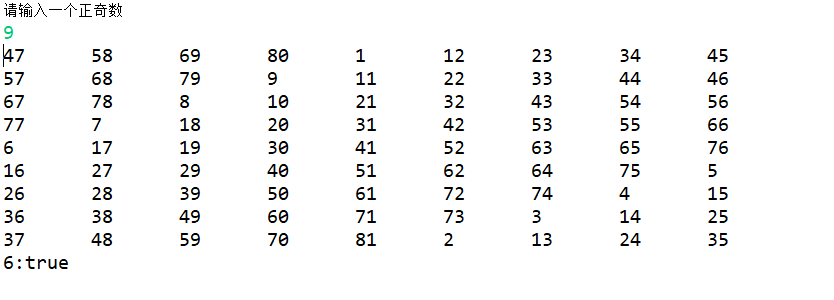
③第一行的下一个数的位置是：行数是最后一行，列数是当前列+1

④最后一列的下一个数的位置是：行数是当前行数-1，列数是第一列

⑤当某一个数的下一个数的位置已经被填入时，这个数的下一个数直接填入当前数的下一个位置，即行数+1，列数相等。

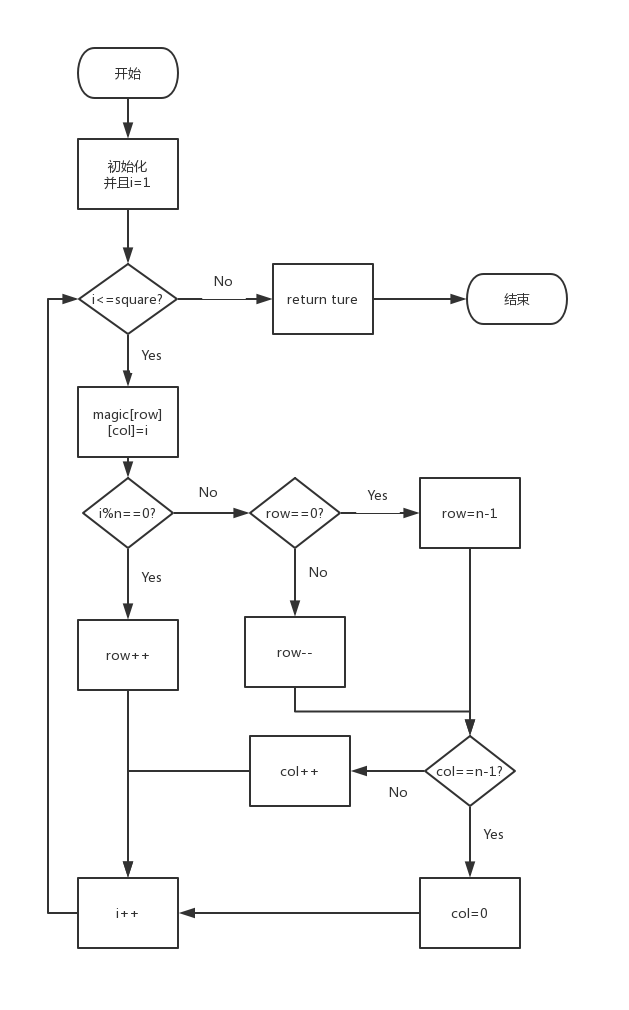
测试结果：

当输入一个数为9时，文件中的矩阵为



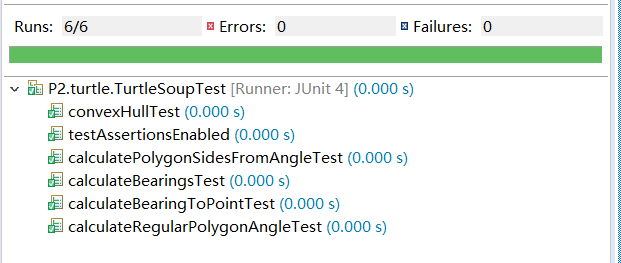
通过判断可知，该矩阵是幻方矩阵。

流程图



## Turtle Graphics

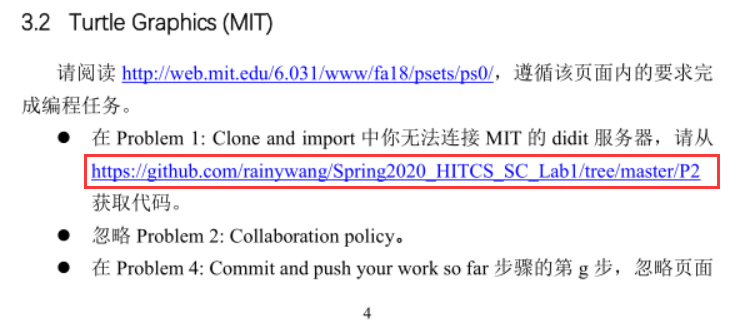
该任务需要我们帮助设计移动路线，比如problem2：设计正方形路线以及problem8：我们自己设置的各种图形。这就需要使用乌龟转弯turn()函数和行前走forword()函数，同时为了实现规则的正方形需要求一些几何问题，比如走多少路转弯一次？转弯角度是多少？转弯几次？已知几何形的边长求角度?已知几何形的角度求边长等等。(problem 5 和 probelm 6)problem 7是一个求凸包的问题。

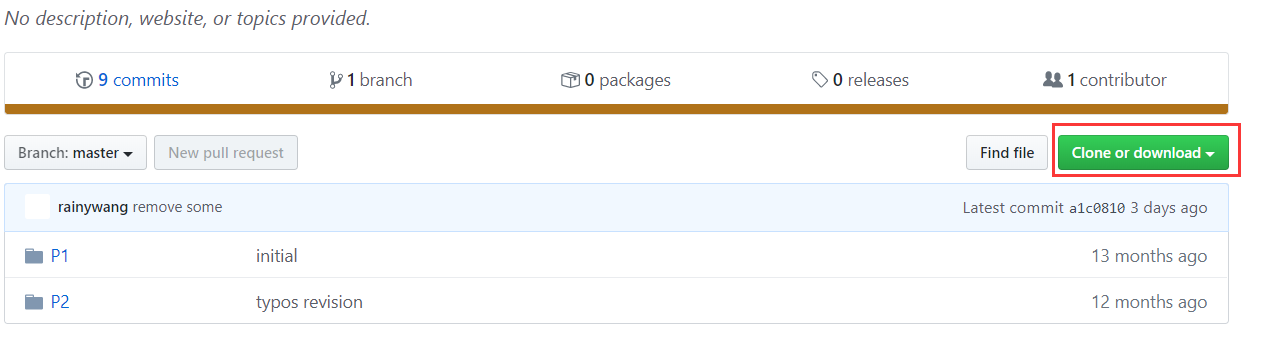


### Problem 1: Clone and import

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。

**获取任务的代码：通过lab1的指导书所给定的链接，去GitHub下载即可。**



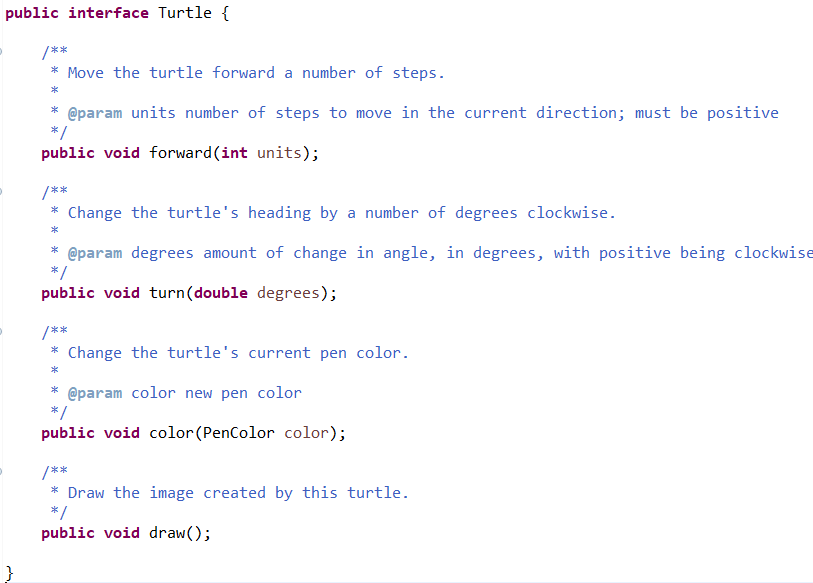


**在本地创建git仓库：**①首先在git中设置用户名和邮件②git init初始化本地仓库：在一个自己设置的文件夹中输入git init 即可把这个目录编程Git可以管理的本地仓库。

**使用git管理本地开发：**①git add remote origin 添加远程库源 ②在远程仓库创建master分支 ③git pull origin master将远程仓库同步到本地 ④git add->git commit->git push 将本地文件加入本地仓库，将本地仓库同步到远程仓库。

### Problem 3: Turtle graphics and drawSquare

这个问题特别简单，只需要画一个正方形即可。我们找到Turtle接口，看看Turtle接口的方法。



3.2.2.1 Turtle 接口

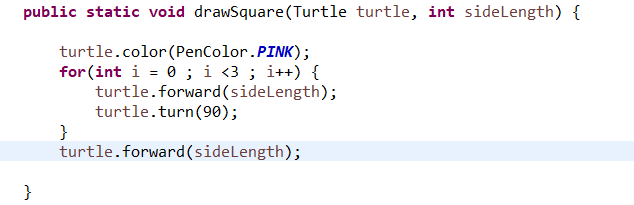
通过描述可知：forward()函数能够让turtle移动一段距离

turn()函数能够让乌龟旋转一定的角度

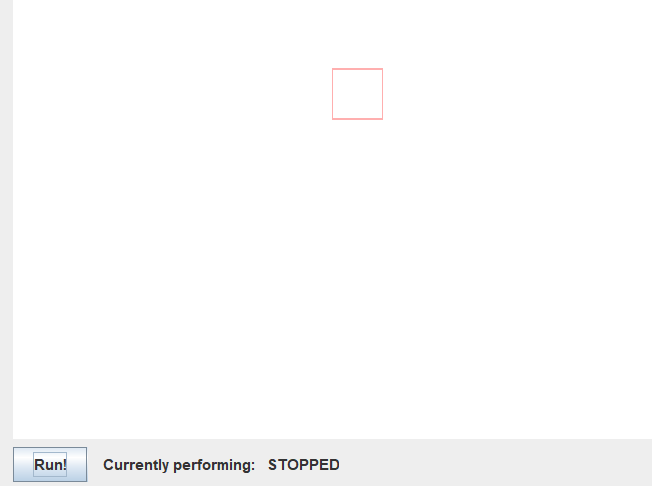
color()函数能够设定turtle的颜色

draw()函数就是turtle行走路线轨迹(就是一个图形)

该问题很简单：由于初始画笔的方向向上，画一个正方形只需要先forward再turn，循环三次。然后再forward一次。因为只需要转三次角度就行。



3.2.2.2 画正方形



3.2.2.3 正方形

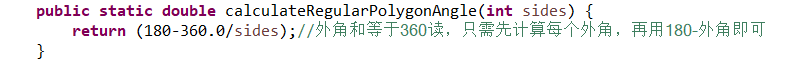
### Problem 5: Drawing polygons

这个问题需要我们实现三个函数

**3.2.3.1 public** **static** **double** calculateRegularPolygonAngle(**int** sides)函数

这个函数要求我们根据多边形的边数求出正多边形的每个内角

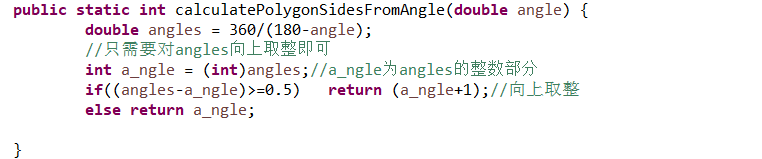
由于多边形的外角和为360度，根据边数就可以知道每个外角为360/sides，然后用180减去即可。



3.2.3.1 求角度

**3.2.3.2public** **static** **int** calculatePolygonSidesFromAngle(**double** angle)

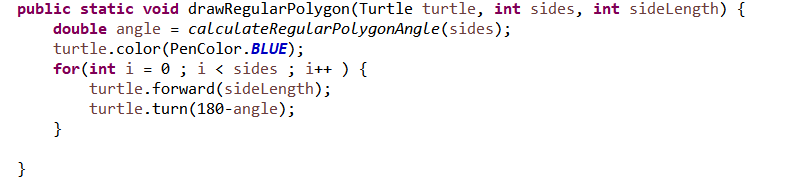
和第一个问题类似，这个问题在已知多边形的内角度数求多边形的边数。这个要注意的问题是注意浮点数转变为整数时的处理



3.2.3.2 求边数

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

这个问题就是要我们画任意的多边形，与画正方形的方法类似，无疑就是确定旋转角度以及旋转次数而已。但是要注意的是旋转角度是外角的度数。



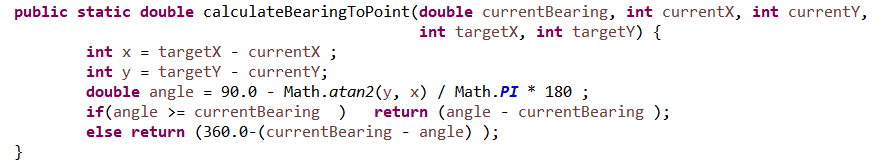
3.2.3.4画多边形

### Problem 6: Calculating Bearings

这个problem是计算轴承（向量的方向角），需要我们实现两个函数。

**3.2.4.1 public** **static** **double** calculateBearingToPoint(**double** currentBearing, **int** currentX, **int** currentY,**int** targetX, **int** targetY)

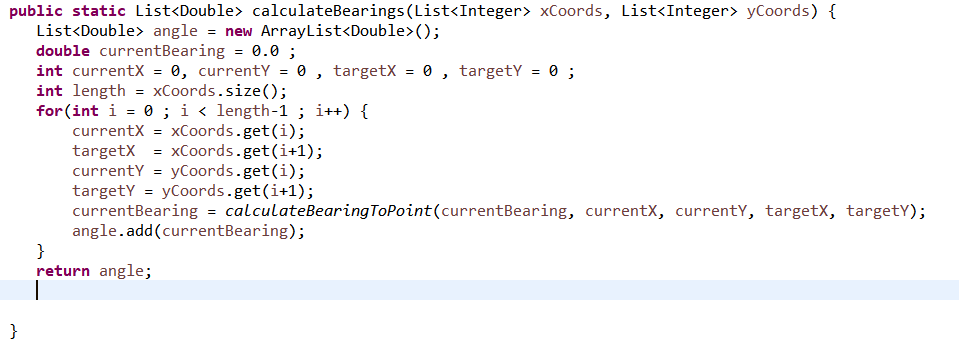
计算（当前点，当前朝向）构成的向量与（当前点，目标点）构成的向量之间的夹角，首先计算（当前点，目标点）构成的向量与y轴之间的tan，使用Math.atan2计算与y轴夹角，与当前朝向夹角（与y轴之间）作差计算两向量之间的夹角，需要注意负角变正角。



计算单个轴承

**3.2.4.2 public** **static** List<Double> calculateBearings(List<Integer> xCoords, List<Integer> yCoords)

这个问题就是要求我们对列表中的每两个相邻节点调用calculateBearingToPoint函数计算夹角，并用列表保存。



计算多个轴承

### Problem 7: Convex Hulls

这个问题就是凸包问题，即求一组最小的点集，使得这些点集依次连线能够把所有的点给包围起来。

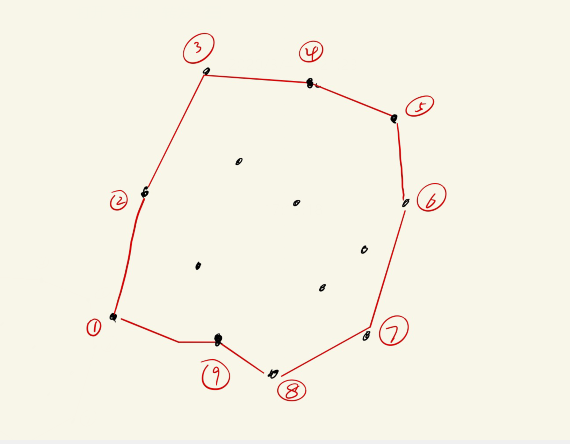
通过查阅资料，我打算采用边界漫游法。

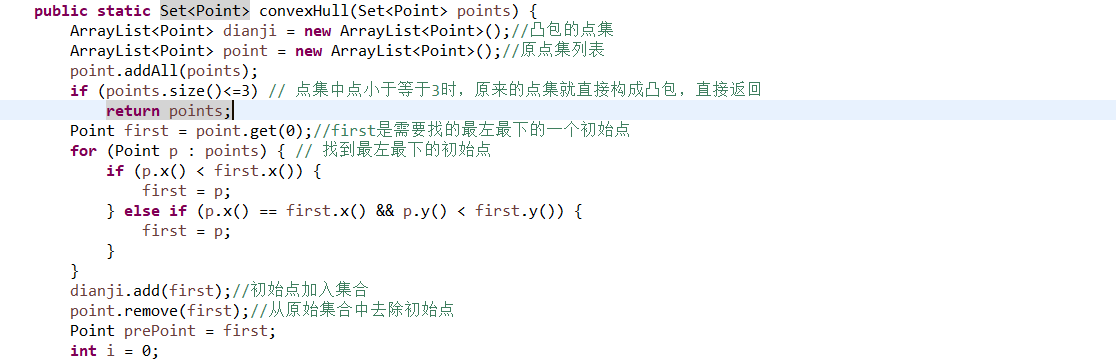
①当点集的点个数小于等于3时，直接返回即可。否则，转②

②首先遍历所有的点，找到最左下角的点。把这个点设置为基点

③接下来，以当前点为基点，y轴正向为当前偏移角，找到顺时针转角最小的角，如果有两个或多个最小转角相同的点，则取最远的点。将这个点加入凸包点集。并将这个加入的点集作为当前的点。并且在原来点集删除这个点。

④循环②③直到再次遇到第一个加入到凸包的点集。

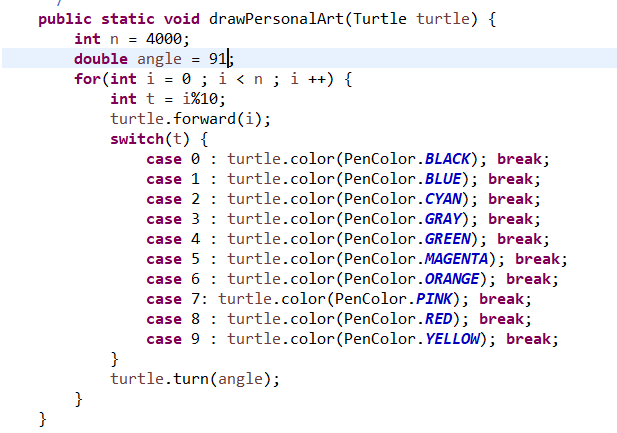


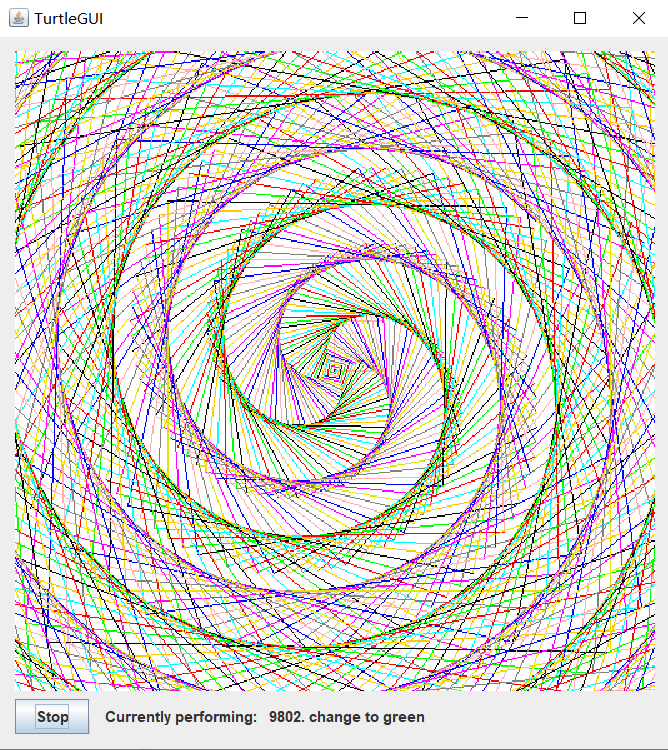




### Problem 8: Personal art

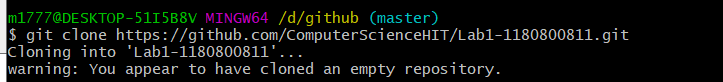
这个就是自己设计图形，我是打算是一个迷眩一点的。所以让他每次转91度，每一次走的距离又0-4000，而且每次的颜色都会变。就像迭代的图形一样

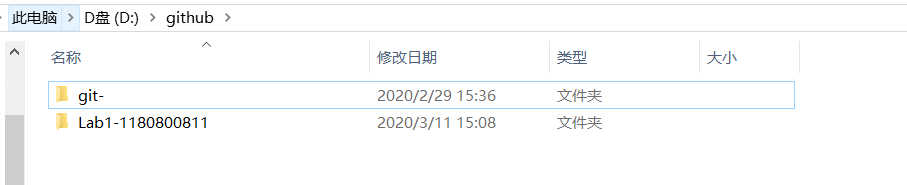




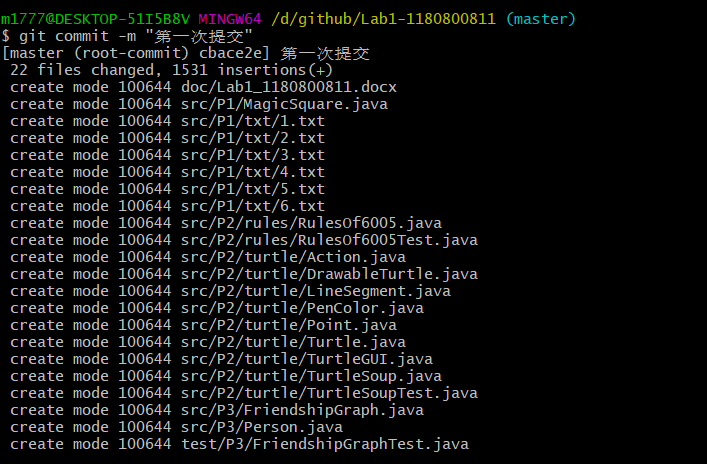
### Submitting

1. 首先git clone

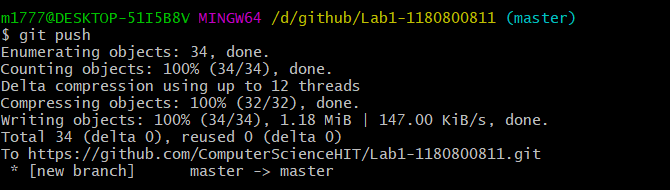


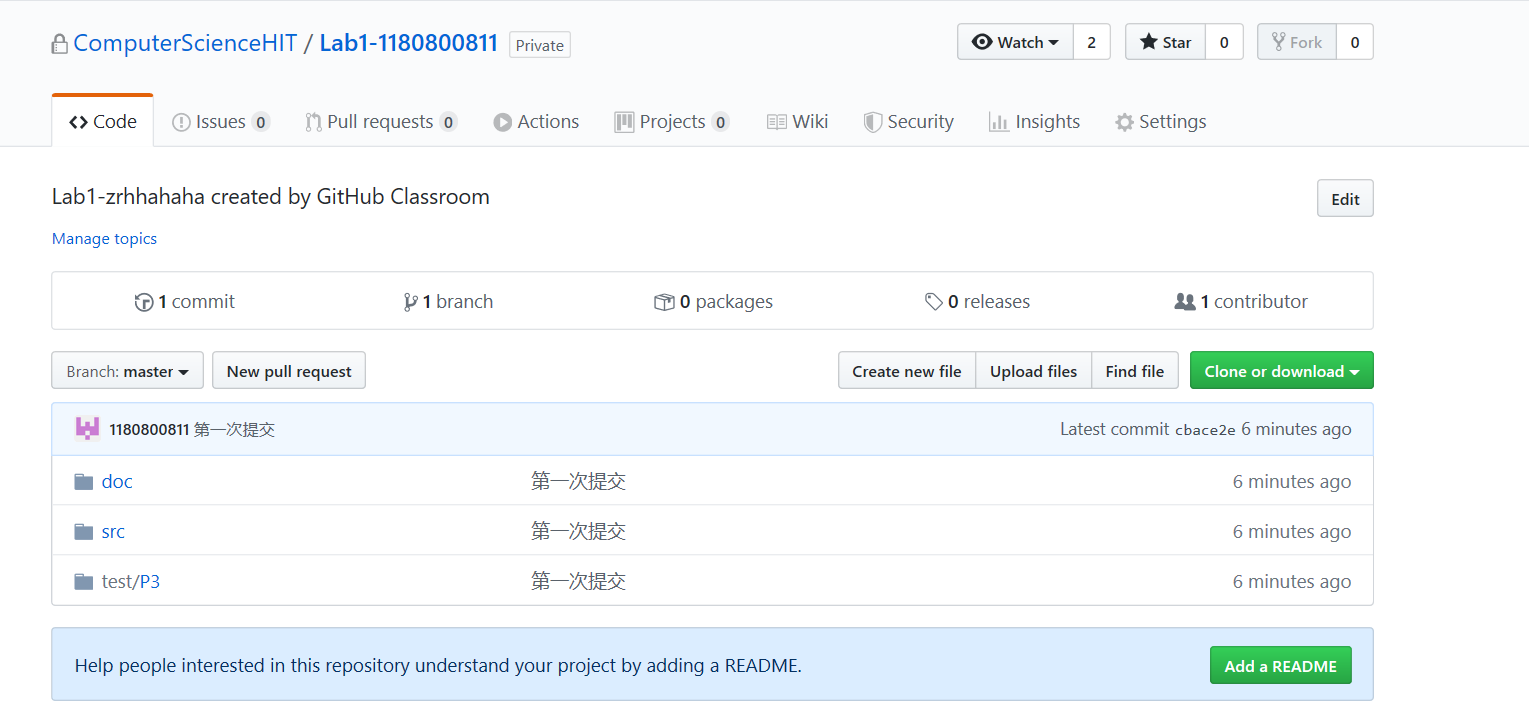


1. git add 和 git commit



1. git push





## Social Network

### 设计/实现FriendshipGraph类

设计思想：friendshipGraph类主要就是实现类似于求有向图人的最短距离。

因此：表示这个无向图：我采用Map映射关系来描绘这个关系图。Map<Person,ArrayList<Person>,每个Person对象作为一个键映射一个朋友集即Person列表作为值。



①**public** **boolean** addVertex(Person p)

这个函数就是把一个Person对象加入到图中，但是必须要判断会不会重名！如果添加成功返回true，否则返回false，并输出提示信息

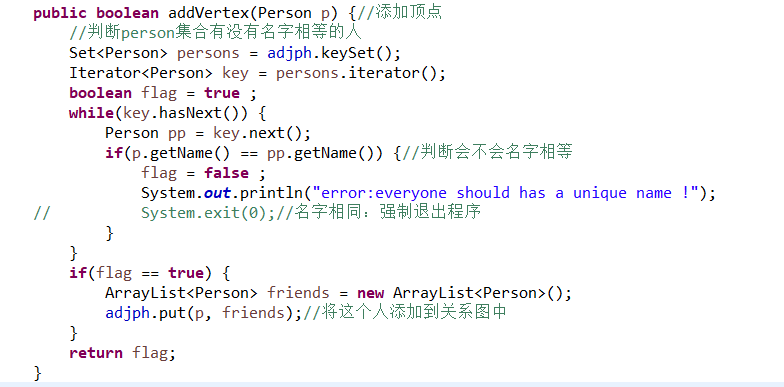


图3.3.1.1 addVertex的代码

②**public** **boolean** addEdge(Person p1 , Person p2)

该方法即为p1添加边p2，但是必须先判断p1原来的边集有没有包括p2.添加成功返回true，否则返回false。

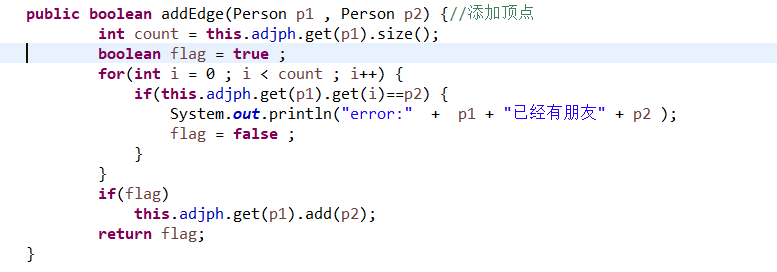


图3.3.1.2 addEdge的代码

③**public** **int** getDistance(Person p1, Person p2)

该方法实现原理类似于图的深度搜索，借助于队列来实现。并且构建一个点集是否访问的数组。初始时将p1压入队列中，将这个点标记为访问过，然后循环以下步骤直到队列为空：将队首元素s弹出，如果s==p2，直接返回，否则访问所有和p1有边的点集，如果该点没有被访问过，如果这个点就是p2，直接返回，否则将该点标记为访问过，并压入队列中。



图3.3.1.3 getDinstance的代码

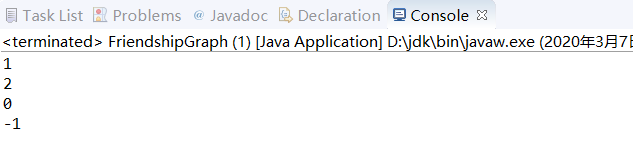


图3.3.1.4 正常输出的测试结果

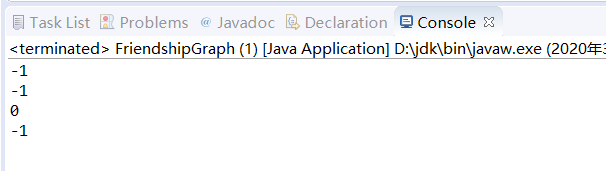


图3.3.1.5 注释掉rachel->ross后的输出结果

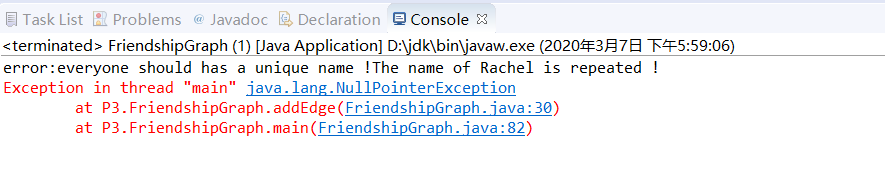
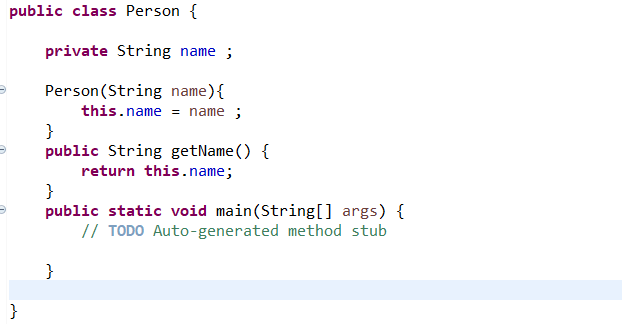


图3.3.1.6 同名输出结果

### 3.3.1 设计/实现Person类

Person类主要是实现一个构造方法，用于初始化，另外一个就是getName方法，用于后续判断同名的情况。



### 设计/实现客户端代码main()

实现思想：首先根据实验指导书的给出的例子，给出正确的结果。

然后注释掉

然后测试当出现同名时的结果。

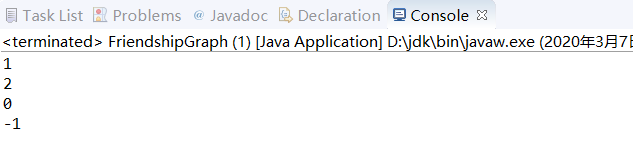


图3.3.2.1 正常输出的测试结果

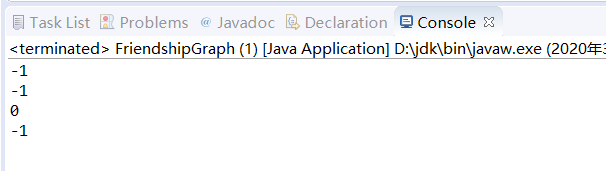


图3.3.2.2 注释掉rachel->ross后的输出结果

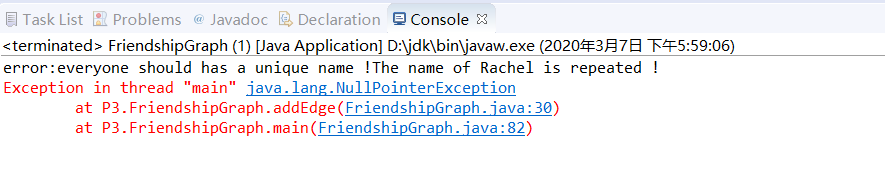


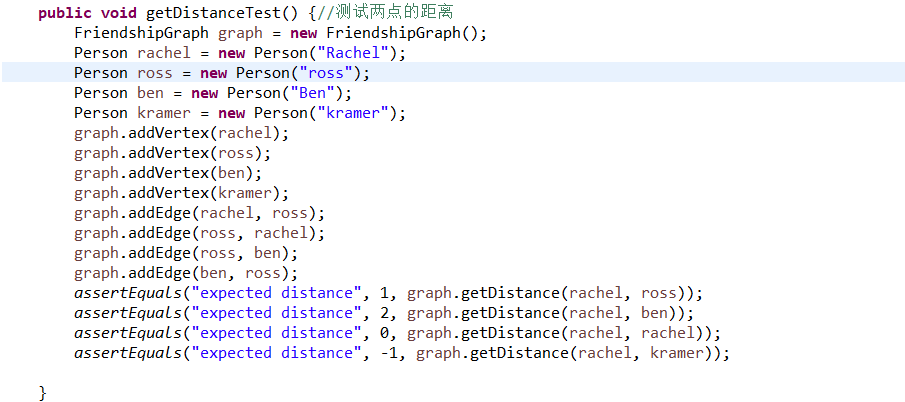
图3.3.2.3 同名输出结果

### 设计/实现测试用例

### 

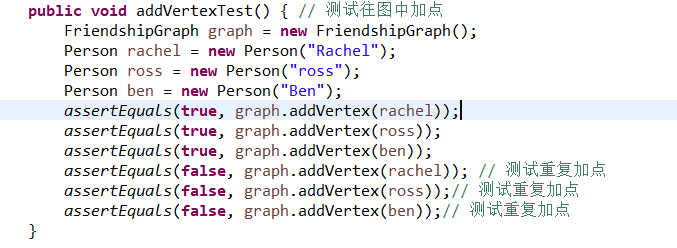
对于每一个方法都需要有测试用例。在第三个问题中，实现的方法有:addEdge(),addVertex(),getDistance(),getName(),因此对这些方法进行测试即可。因此在测试类中添加这些方法的测试。主要是使用assetEquals方法。

1. **public** **void** getDistanceTest()：

测试思想，用实验指导书给出的用例进行测试，来测试我们的预期结果和实际结果是否相等。

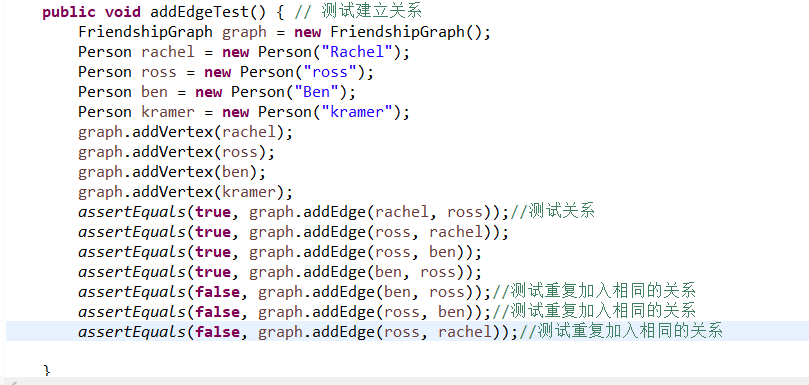
②**public** **void** addVertexTest()

测试往图中增加点的函数，等价类就是增加点成功，增加点不成功，特殊用例就是同名现象

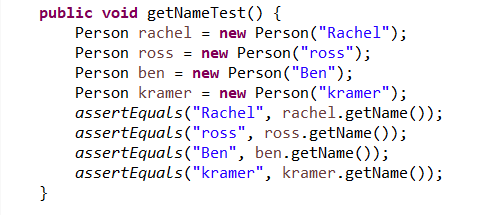


③**Public void** addEdgeTest()

测试建立关系的函数是否成，即等价类为成功和不成功。



1. 测试getName方法



# 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 |
| 2020-02-26 | 20:00-21:00 | 了解了实验的目标和要求 | 按计划完成 |
| 2020-02-27 | 09:12-09:46 | 学会java怎么读入文件 | 按计划完成 |
| 2020-02-27 | 9:00-12:20 | 完成实验第一个任务Magic Squares | 延期完成 |
| 2020-02-27 | 14:50-15:25 | 写了Magic Squares实验部分的实验报告 | 提前完成 |
| 2020-02-28 | 8:17-11:44 | 学习git如何对本地仓库进行管理 | 延期完成 |
| 2020-02-28 | 19:03-22:36 | 完成Turtle Graphics问题的problem3/4/5/6/8 | 按计划完成 |
| 2020-02-29 | 09:00-10:26 | 完成Turtle Graphics问题的problem7 | 未完成 |
| 2020-02-29 | 10:30-12:30 | 深入了解了git的使用方法并发表自己的博客 | 按计划完成 |
| 2020-02-29 | 14:40-15:10 | 写Turtle Graphics问题部分的实验报告 | 按计划完成 |
| 2020-03-01 | 20:20-21:03 | 完成Person类和FriendshipGraph类 | 提前完成 |
| 2020-03-06 | 14:40-15:47 | 完成Turtle Graphics问题的problem7 | 按时完成 |
| 2020-03-07 | 14:00-15:30 | 解决了第三个问题的测试用例编写 | 延期完成 |
| 2020-03-07 | 16:40-17:50 | 完成实验报告的第三个问题 | 按时完成 |
| 2020-03-08 | 10:20-11:30 | 完成实验报告的填写 | 按时完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 对String类的split方法不懂 | 查阅Java API文档并上csdn查看别人撰写的博客 |
| 在写Magic Squares这个实验的时候，出现的debug主要有：charAt函数的下标出界，判断是否是对角线元素的和时，限制是：i+j=row-1,而不是i+j=n | 通过搜索java报错可知是charAt函数访问越界。纠结了很久才发现i+j=row - 1 |
| 对Magic Squares这个实验的文件的输入输出不懂 | 看了清华大学java程序设计的mooc，然后通过看PPT试着写就看懂了 |
| 不会使用git | 上网查阅别人的博客，跟着一个步骤一个步骤来实践，最后总结写出自己关于git使用方法的博客 |
| 对于java的关于队列以及一些集合的接口的实现不熟悉 | 上网查阅博客 |
| 在为第三个问题编写测试用例的时候，总是通不过去 | 忘了在每个测试方法前面标注：@test |
| 求凸包问题不理解 | 网上查阅博客 |

# 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

本次实验确实任务量有点大，对于java不熟和英语差的我来说挑战更是有点艰巨。

从第一个问题着手，理解起来很简单，但是实现起来的细节很多，需要考虑矩阵的特殊情况，比如矩阵元素是不是整数，分隔符号是不是“\t”等等，并且第一个问题对于异常处理、文件读写和java类库的一些知识都有要求比如split方法的使用、Interger.valueOf方法的使用等，因此就只能去网上百度学习文件读写、异常处理的知识再来做这个题目。当学会了这些知识之后并应用了之后，就变成了给定一个二维数组，判定是不是幻方矩阵，这个问题就特别简单了。然后就是一个构造奇数阶的幻方矩阵的函数，一开始有点迷，只有当自己用手一步一步写出来的时候才真正懂了这个原理。

第二个问题曾经一度让我不知道咋整。首先是git的使用方法，看着指导书真的有点摸不着头脑，辛亏在网上找到了一位大神的博客，手把手的教，让我对git有了一定的了解。在学习git的使用就花费了很多的时间。然后就是看不懂的turtle英文。辛亏有Google翻译，带着闯一闯的心态，慢慢的看懂了第二个问题。但是当我把老师给的P2导入的时候遇到很多的问题，总是出现错误，上网查阅资料也没有解决，最后发现原来是Junit版本太低了....，把Junit3改变成Junit4问题一下子解决了，这个问题卡了我一下午。于是解决该问题还是特别简单，就是画画图，求一求数学知识层面的一些参数，比较简单。但是这个问题还是要自己去学习一个集合和接口的知识，Map、List、Set等等。最难的应该就是凸包问题，单纯看这个问题有点难懂，于是去网上搜了一下。道理比较简单，但是实现起来bug很多。一开始怎么都过不了测试用例。于是先把这个问题放了放，到了后面才返回来重新做这个题目。换了个方法解决就通过了。

第三个问题就是把我困在测试用例上面。这个问题就是类似写一个广度优先，求无向图的两顶点之间的距离。所以实现起来不是很麻烦，唯一的不懂的地方就是java怎么实现队列，于是又得百度。测试用例这一块真的高炸我心态，明明感觉自己写的测试用例是对的，确怎么也通不过去，直到我发现我没有在每个测试方法前面加@Test。。

**经验和教训**：java基本知识要掌握牢固，在写完每一个方法应该就写测试用例。

## 针对以下方面的感受

1. Java编程语言是否对你的口味？
2. 关于Eclipse IDE
3. 关于Git和GitHub
4. 关于CMU和MIT的作业
5. 关于本实验的工作量、难度、deadline
6. 关于初接触“软件构造”课程
7. 非常合我的口味，java类库实在太方便了，提供的功能也特别的多，只是自己学识太浅，还不能很好的发掘更多的java简便的方法和类库。
8. Eclipse IDE也很好使用，比如给一些代码，然后选择右键->Refactor就可以很容易的将代码打包，构建一个新的方法。另外Eclipse对于错误的提示信息也特别的好，很容易发现自己错在哪里。
9. Git是真的很好使用，GitHub用来管理仓库也特别的方便，最重要的是在GitHub可以看到很多开源的项目，真的特别棒
10. CMU和MIT的课程和实验还是很有难度的，参考的价值很大。
11. 工作量看肯定不用说的，比计算机系统任务大的多。难度适中，deadline给的很充足。但是后面的实验可能就很难了
12. 这门课真的改变了我对软件的认可，原来一个好的软件需要考虑这么多的问题，上大学以来觉得这么课的价值应该特别大，真的会提高我们的java水平和各种其他知识，比如git的使用等等。