

**2021年春季学期**

**计算学部《软件构造》课程**

**Lab 1实验报告**

|  |  |
| --- | --- |
| 姓名 | 李旻翀 |
| 学号 | 1190200208 |
| 班号 | 1936602 |
| 电子邮件 | 1190200208@stu.hit.edu.cn |
| 手机号码 | 18608109998 |

**目录**

2 实验环境配置

3 实验过程

3.1 Magic Squares

3.1.1 isLegalMagicSquare()

3.1.2 generateMagicSquare()

3.2 Turtle Graphics

3.2.1 Problem 1: Clone and import

3.2.2 Problem 3: Turtle graphics and drawSquare

3.2.3 Problem 5: Drawing polygons

3.2.4 Problem 6: Calculating Bearings

3.2.5 Problem 7: Convex Hulls

3.2.6 Problem 8: Personal art

3.2.7 Submitting

3.3 Social Network

3.3.1 设计/实现FriendshipGraph类

3.3.2 设计/实现Person类

3.3.3 设计/实现客户端代码main()

3.3.4 设计/实现测试用例

4 实验进度记录

5 实验过程中遇到的困难与解决途径

6 实验过程中收获的经验、教训、感想

6.1 实验过程中收获的经验和教训

6.2 针对以下方面的感受

# 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

本次实验采用IDE为 IDEA 2020.3,Java环境的配置、IDE的安装以及git的使用在学期初预习时便已完成，因此此处不再赘述。需要注意的是JDK安装时环境变量的配置以及IDEA的基本快捷键了解，相关资料均可在网上找到。

GitHub Lab1仓库的URL地址如下：

https://github.com/ComputerScienceHIT/Lab1-1190200208

# 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

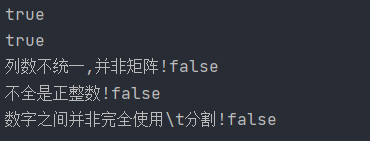
程序的目标是读入txt文档中的数表，并判断其是否为幻方矩阵，并返回true / false。在完成上述功能后，进一步实现生成幻方矩阵并保存至本地、调用函数判断其是否为幻方的功能。本题中的幻方矩阵需满足以下要求：①输入文件符合规范（不能出现行列数不等，并非矩阵，有非正整数，数字间不以\t分割等情况）②矩阵每一行元素和，每一列元素和，以及每一条对角线元素和都相等。

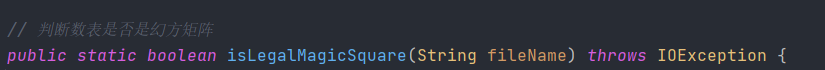
### isLegalMagicSquare()

函数的简要实现思路如下：

1. 创建输入流对象，读入txt文档，并按行保存在ArrayList中，方便后续处理。读取文件部分代码展示如下：
2. 对ArrayList中的每一行使用.split( )方法，按\t切分，得到行列所对应的元素。构造二维数组square[ ][ ]，将得到的元素转换成整形，保存在square中。另外，在这一步的过程中，也实现了对输入文件各种特殊情况（行列数不等，并非矩阵，有非正整数，数字间不以\t分割等）的处理。
3. 计算square中第0行元素之和，以此为基准值，利用for循环计算第i行，第j列，各对角线元素和，若均与基准值相等，则返回true，否则返回false。

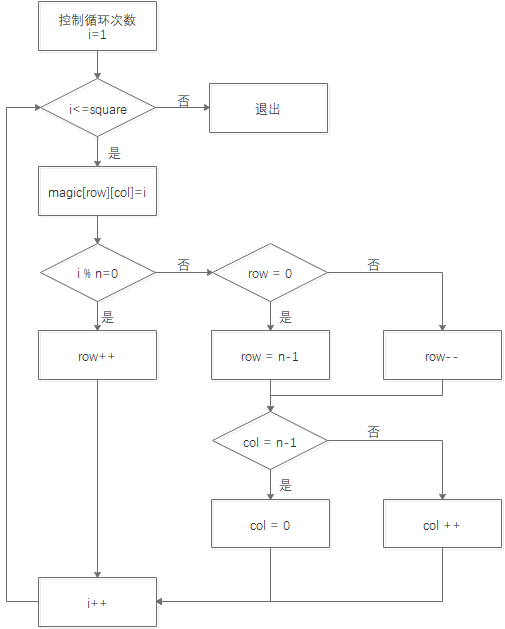
运行结果如下：



同时，函数能够重抛IOException异常。

### generateMagicSquare()

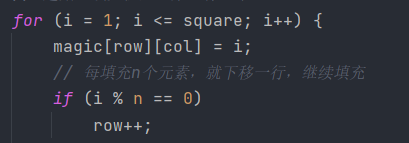
函数的流程图如下：



函数传入一个奇数n，首先，在函数中创建了二维数组magic[n][n]来表示矩阵，根据循环次数控制变量i，依次向矩阵中填充的数字。在第一次循环时，起点为第一行中间，之后每次循环，依次向上一次循环填充位置的右上角移动一格，若遇到上边界就返回到下边界，遇到第一行，则返回最后一行，遇到最后一列则返回第一列。同时，每输入n个数填充位置就下移一行，直到整个矩阵填充完毕。

对整个函数进行注释后，研究其输入n为偶数或负数时产生的异常情况。

n为偶数时，错误提示为数组下标越界。其原因如下：



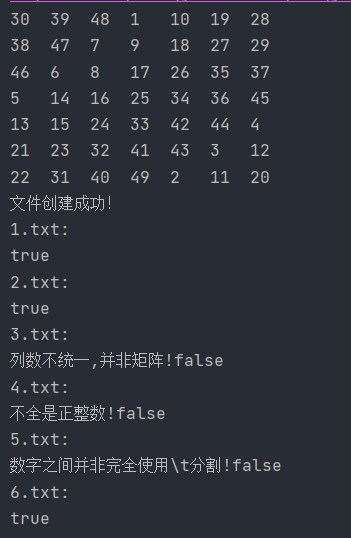
当i=n时，row刚好为n-1，此时经过如图示的if判断语句，会导致row++后row=n，数组下标越界。

当n为负数时，错误提示为数组大小为负数，这是因为在函数中需要根据n的值创建数组，自然n为负值时会弹出如此的报错提示。

之后对该函数做扩展，将产生的幻方矩阵写入文件6.txt，并在输入的 n 不合法时，返回false退出。

最后，利用前面已经写好的 isLegalMagicSquare()函数，在 main()函数判断新生成的6.txt是否符合幻方定义。

最终的运行结果如下（优化了输出显示）：



## Turtle Graphics

Turtle Graphics是一个广泛存在于不同编程语言中的图形库，在这一任务中，主要在java环境下完成一个简单Turtle类的搭建与使用，以此初步锻炼填补空缺代码与编写简单可视化程序的能力。

### Problem 1: Clone and import

<https://github.com/rainywang/Spring2021_HITCS_SC_Lab1/tree/master/P2>获取代码，git clone至本地，移至P2文件夹下，修改各个文件的package与import路径，使之能够正确运行。

修改前：





修改后:



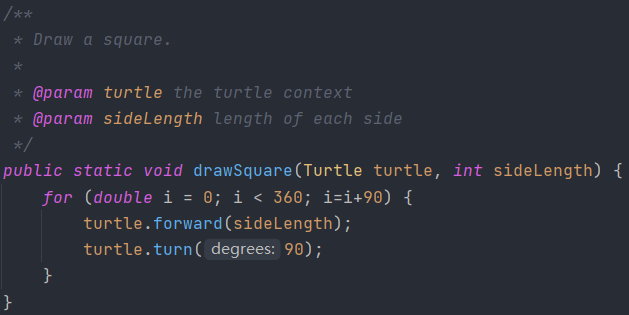
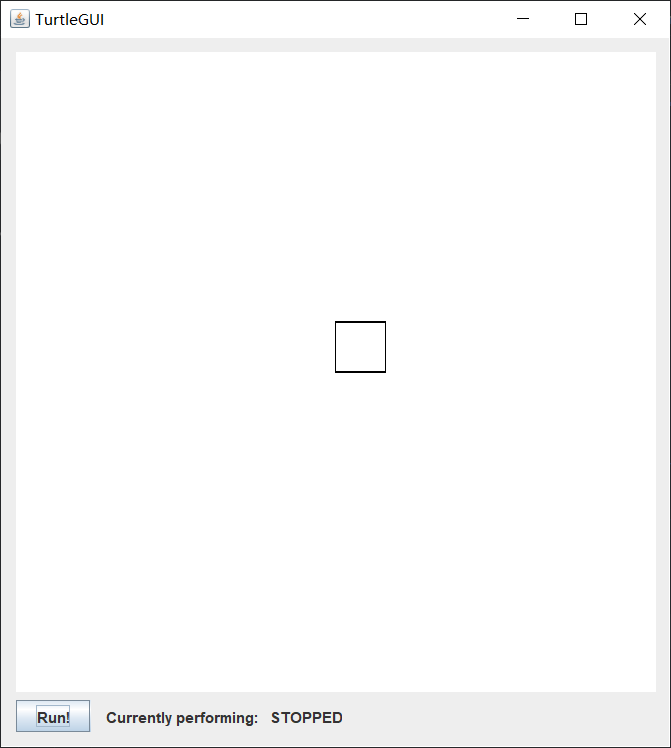


### Problem 3: Turtle graphics and drawSquare

根据MIT网页上的任务提示，此步骤中只用到forward(units)与turn(degrees)两个方法，并且浏览源代码可知，Turtle接口已经在DrawableTurtle.java文件中实现，这降低了我们完成这一步任务的难度。

具体程序实现非常简单：利用for循环，四次执行画线，改变角度的代码即可。

程序代码及运行结果如下：

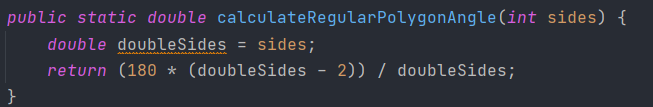
 

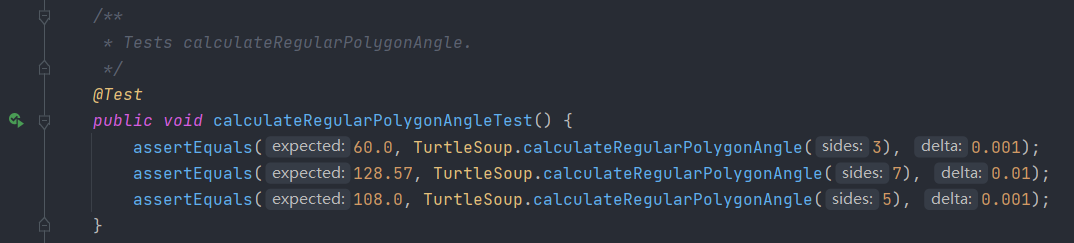
### Problem 5: Drawing polygons

查询得正多边形(regular polygon)内角度数公式：

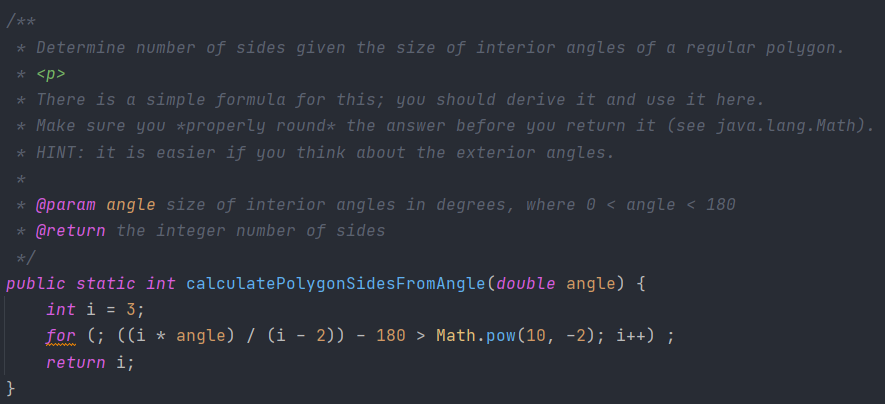


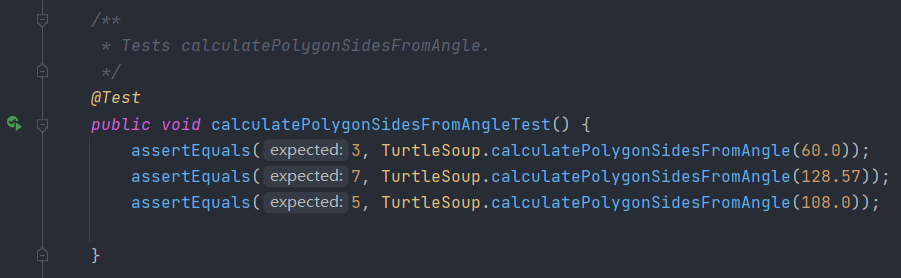
编写程序实现功能，在程序中将sides值赋值给double类型的变量doubleSides，使运算转为浮点运算，程序及测试结果如下：



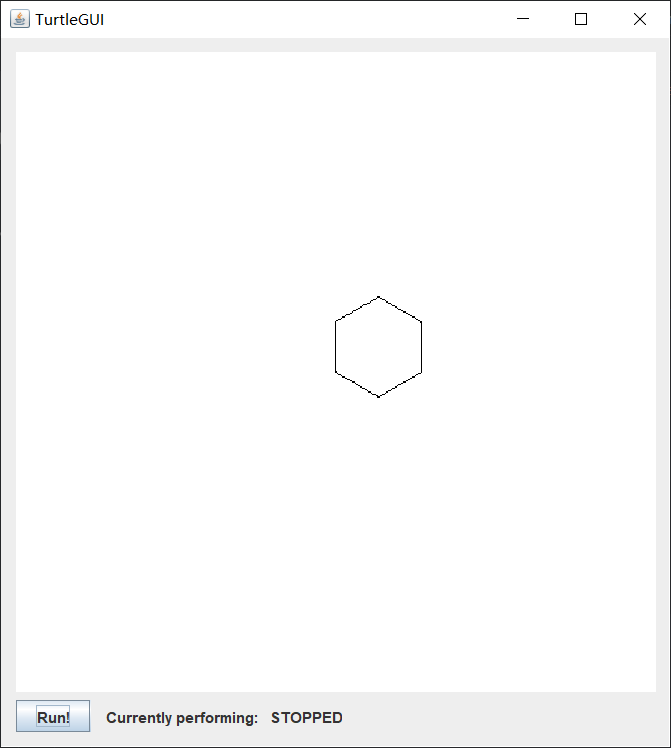


另外，根据上述公式逆推，也实现了由正多边形内角度数反推其边数的函数。





最后实现drawRegularPolygon ( )函数。保存正多边形内角为ang，需要注意每次旋转的角度为180-ang。最后运行结果如下：

### Problem 6: Calculating Bearings

首先设计calculateBearingToPoint函数，描述其功能如下:

已知当前点的坐标，下一点坐标，以及当前点目前所指方向，求需要顺时针转动的角度为多少，才能使当前点指向下一点。

关于此函数的设计，思路如下：

首先，可以根据两点间的与计算出两点连线与轴夹角的值，进而求出两点连线与轴的夹角角度，求出这一数据后，分如下六种情况，分别利用在六种情况下转动角度计算公式计算即可：



若求出的角度值为负，则用360加上它以调整为正；若求出角度值为，则更改其为0。最后编写的程序通过了测试用例。



根据设计好的calculateBearingToPoint函数，进一步设计calculateBearings函数，其功能是读入分别保存x坐标和y坐标的列表，并返回一个列表，其中保存相邻两点转动角度的值。

函数的具体设计思路如下：

每次从xCoords和yCoords中读取当前点与下一点坐标，输入calculateBearingToPoint函数计算两点转动角度，保存在angles列表中，直到循环n-1次后退出，函数返回angles列表。编写的程序最终通过了测试用例。



### Problem 7: Convex Hulls

在这一环节中，需要设计实现求凸包最小点集的函数。查阅Wiki，借鉴礼物包装算法的思路，设计convexHull函数如下：

首先对points中保存的点的个数进行判断，若点数小于等于3，则points中的所有点都构成了最小凸包，直接返回points；若点数大于3，建立保存最小凸包的点集minpoints，继续之后的步骤：

以y坐标最小的点作为起始点，起始转角bearing设为0，利用calculateBearingToPoint函数，计算当前点到其余点的转角，以转角最小对应的点作为下一点；若同时有多个点对应最小转角，则取距离当前点距离最大的点作为下一点；更新转角，对下一点重复以上步骤，直到计算出的下一点为起始点，结束循环。

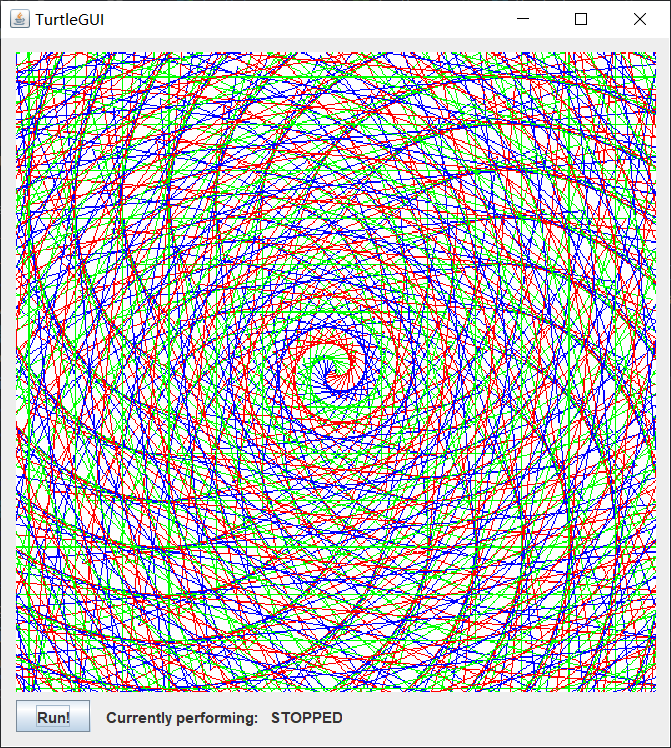
关键代码段展示如下：



另外，在完成此函数后，在debug模式下单步执行能够返回正确答案，将用例写入主函数也可正常运行，但直接运行测试用例会一直卡死，无响应。在Piazza上提问，经同学回答提示可能还是自己的代码出现了问题。经过修改，发现将初始点从x坐标最小的点改为y坐标最小的点，就可以让测试用例正常运行。问题得到解决，但为什么产生这样的错误，具体原因仍然不太清楚。

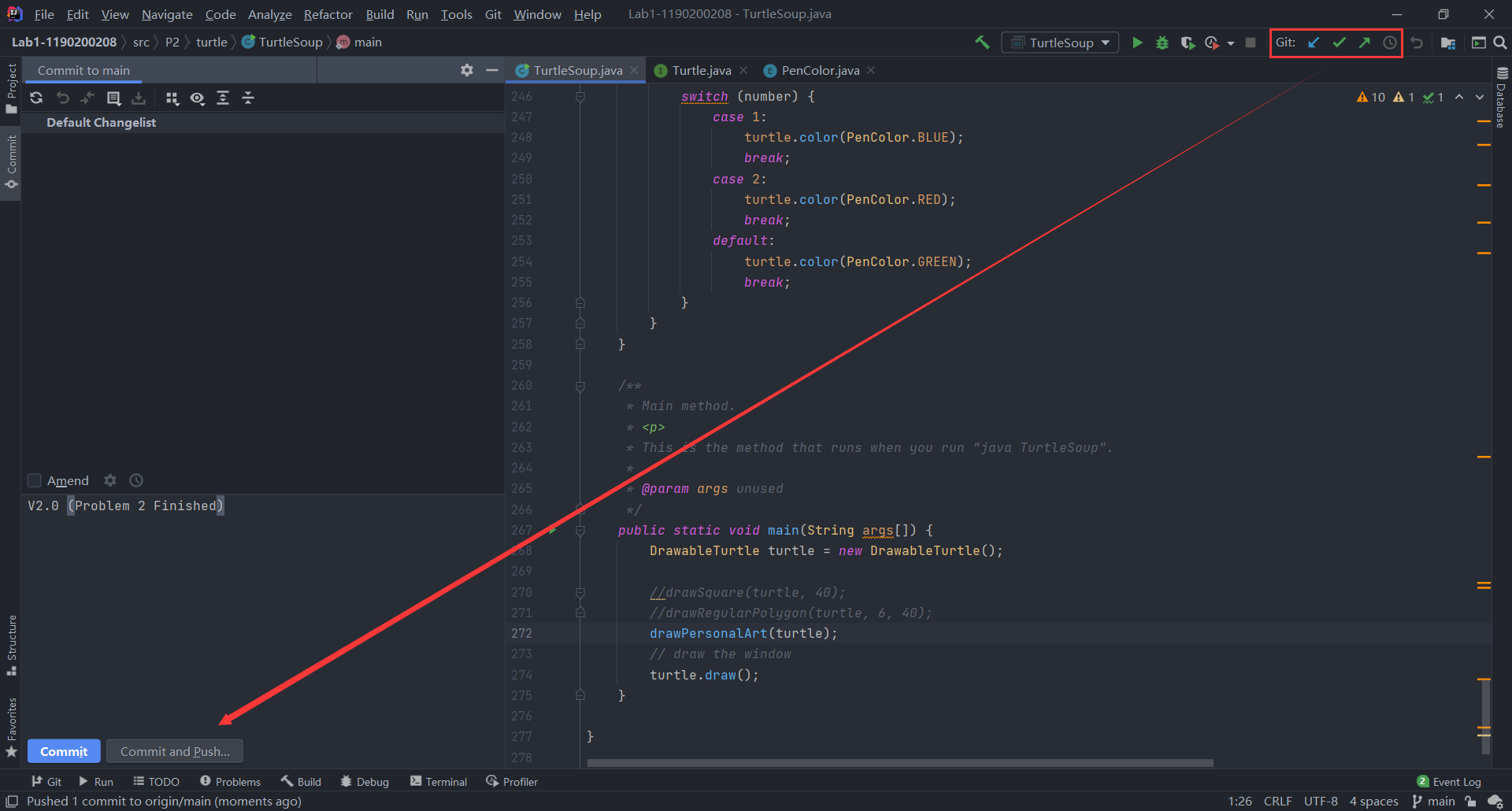
### Problem 8: Personal art

画出的图像如下：



### Submitting

由于IDEA中集成了git功能，所以直接通过IDEA进行commit and push操作。



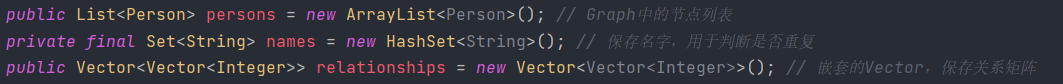
## Social Network

构建一个程序，实现人际关系图的表示。具体来说，需要设计出两个类：一个类表示人与人之间的关系网络，需要能够拓展为有向图；另一个类为网络中的节点，表示关系网络中的个体，另外，还需要为设计的人际关系图编写测试用例，以验证程序的正确性。

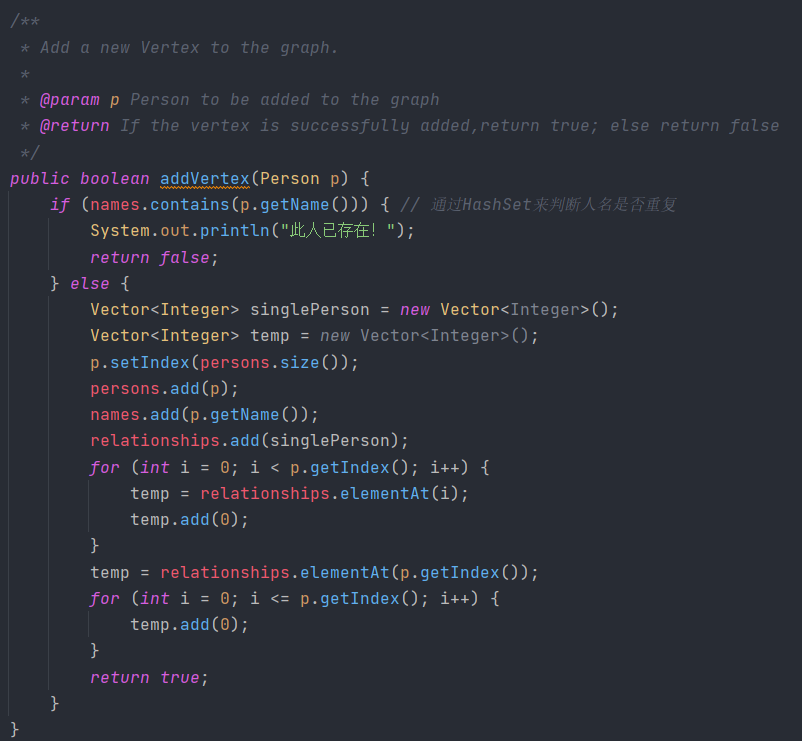
### 设计/实现FriendshipGraph类

FriendshipGraph类包含三个主要方法：addVertex，addEdge，getDistance

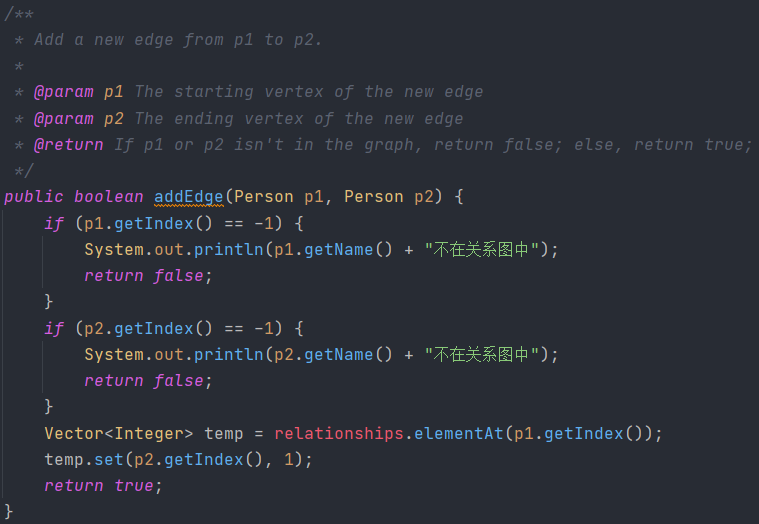
在 FriendshipGraph类中，声明了如下结构，其作用如下：



addVertex的功能是为关系图添加顶点。首先通过Hashset的不可重复性验证名字是否已经出现过，若是，则打印错误提示信息，并返回false。若不是，则将其添加到节点列表persons中，并更新关系矩阵，与顶点内部的相关数据。



addEdge的功能是为关系图添加顶点间的边。由于类要保留拓展到有向图的功能，所以每一次运行addEdge都只会添加一条单向边。首先，判断两个参数p1与p2是否都在节点列表persons中，若否，则打印错误提示信息，并返回false。若是，则修改关系矩阵relationships中的对应值。



getDistance的功能是计算并返回两人间的最短距离。利用了BFS算法：在类中设计了一个队列queue与一个标志是否访问过的数组visit[]（初始化为全0），每次弹出队首元素，并将其直接相连的顶点入队，标记为访问过。每次完成一轮广搜，distance++。直到队头为终点，返回此时的distance，若直到队空也没遇见目标顶点，则返回-1。

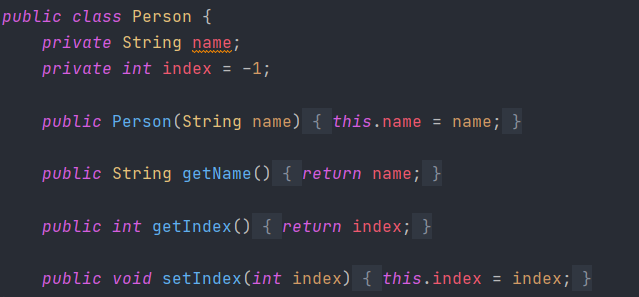
另外，针对输入的两个person相同/输出参数中有的不在图中，设计了相应的错误提示。

以下是代码的关键部分:



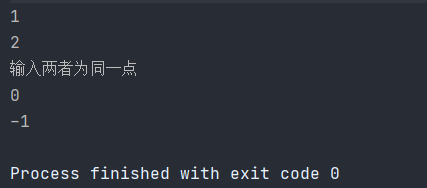
### 设计/实现Person类

Perosn类中包含两个私有成员，一个为其名字，一个为index，表示其在关系矩阵中表示第index行与第index列。另外定义了四个公共方法，便于对其两个成员进行修改与查找的工作。



### 设计/实现客户端代码main()

将实验手册中的代码改写到自己程序中即可，最后的运行结果如下（包含错误提示）：

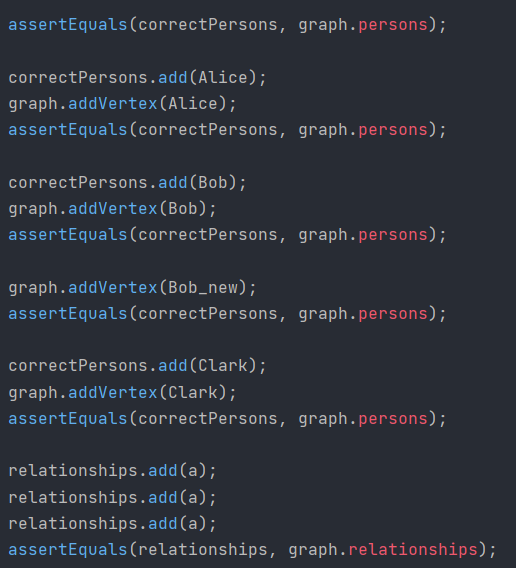


### 设计/实现测试用例

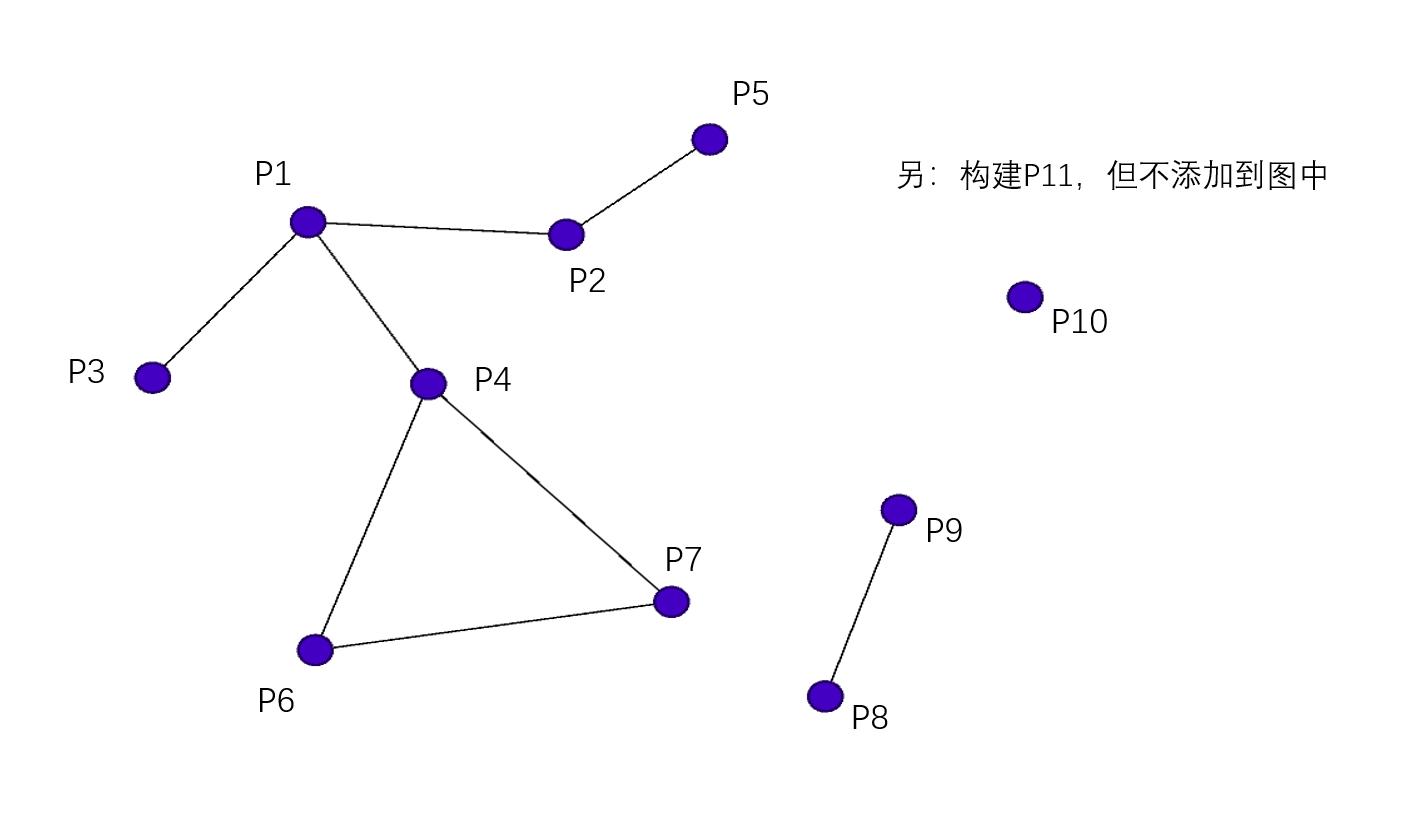
针对FriendshipGraph类中的主要函数，均设计了测试用例。

1. 对于addVertex函数，设计了addVertexTest。在其中调用addVertex添加节点，判断添加后graph中的节点列表graph.persons是否符合预期。针对异常，考虑了添加同名节点的情况；另外，还验证了关系矩阵graph.relationships是否符合预期。

关键部分代码：



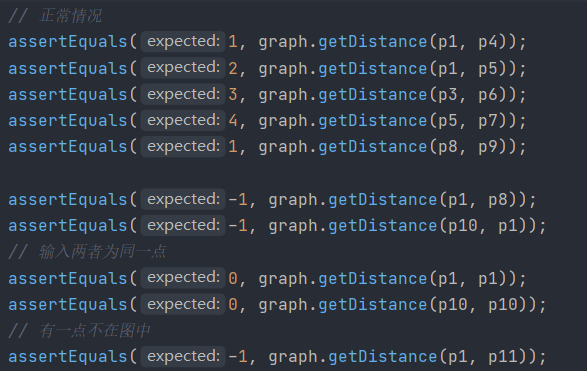
1. 对于addEdge函数，设计了addEdgeTest。验证添加边后图的邻接矩阵是否符合预期。并在向图中添加新的顶点后再次验证，以确保程序正确无误。
2. 对于getDistance函数，设计了getDistanceTest。构造了如下的关系图。



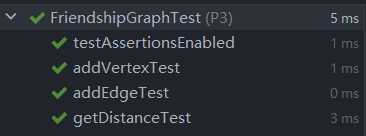
对此图输入如下几类参数，进行测试：

1. 相连通两顶点（如P3与P6）
2. 不连通两顶点（如P1与P8）
3. 参数为两相同顶点
4. 某一参数不在图中（如P1与P11）

测试用例如下：



最终的测试结果：



# 实验进度记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 任务 | 实际完成情况 |
| 2021-05-11 | 13:45-5:30 | 编写问题1的isLegalMagicSquare函数并进行测试 | 按计划完成 |
| 2021-05-13 | 9:10-9:35 | 完成问题一的部分报告撰写 | 按计划完成 |
| 2021-05-17 | 9:10-9:45 | 分析问题1的generateMagicSquare函数，并撰写部分报告 | 按计划完成 |
| 2021-05-17 | 21:05-22:05 | 完成问题1的generateMagicSquare函数拓展与对应部分的报告撰写 | 按计划完成 |
| 2021-05-18 | 13:45-15:15 | 完成问题2的Problem 3,5及对应部分的报告撰写 | 按计划完成 |
| 2021-05-18 | 18:50-21:00 | 完成问题2的Problem 6及对应部分的报告撰写 | 按计划完成 |
| 2021-05-19 | 19:40-22:30 | 编写问题2的Problem 6代码 | 遇到困难，未完成 |
| 2021-05-20 | 8:00-10:30 | 解决问题2，Problem 6之前存在的问题 | 按计划完成 |
| 2021-05-20 | 14:50-15:40 | 完成问题2剩余部分的编程与报告撰写 | 按计划完成 |
| 2021-05-20 | 19:30-21:00 | 完成问题3的部分代码编写 | 按计划完成 |
| 2021-05-21 | 8:00-11:00 | 完成问题3的部分代码编写 | 按计划完成 |
| 2021-05-22 | 8:20-9:15 | 完成问题3的代码编写 | 按计划完成 |
| 2021-05-22 | 9:15-11:00 | 完成问题3及后续报告撰写，并优化代码 | 按计划完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的困难 | 解决途径 |
| 对java方法不熟悉 | 参考网上博客，针对遇到的困难搜索一般的解决方案，并在自己的程序中应用 |
| Problem2，问题6的测试用例无法正常运行，在debug模式下单步执行能够返回正确答案，将用例写入主函数也可正常运行。 | 在Piazza上提问，经同学回答提示可能还是自己的代码出现了问题。经过修改，发现将初始点从x坐标最小的点改为y坐标最小的点，就可以让测试用例正常运行。问题得到解决，但为什么产生这样的错误，具体原因仍然不太清楚。 |
| 在完成问题三的getDistance函数时，由于对BFS算法的特性不了解，导致花费了较长时间编写程序 | 查阅网络博客，深入理解BFS算法的特性。 |

# 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：在遇到困难时，及时查阅技术文档，很多问题就能迎刃而解

教训：在动手编写程序前做好规划，避免因为使用了不当的数据结构造成后续程序难以编写。

## 针对以下方面的感受

1. Java编程语言是否对你的口味？

Java内置了许多函数，容器，功能非常强大，实验一算是一个不断适应Java编程的过程，通过这次的实验，不断查阅技术文档，感受到运用Java编程能够很方便地实现目标功能。

1. 关于Eclipse IDE；

本人使用的是IDEA，感觉非常方便好用。

1. 关于Git和GitHub；

通过这次实验熟悉了Git与GitHub的使用。感受到两者都是非常强大，非常有用的工具。体会到了先进生产力的魅力。

1. 关于CMU和MIT的作业；

难度适中，但建议将我们自己作业与CMU/MIT作业中要求不同的地方归纳在一起列出，不然容易做漏。

1. 关于本实验的工作量、难度、deadline；

对于第一次实验来说，个人感觉难度，工作量适中，piazza网站上提问反馈也很及时，但因为实验过程中穿插了自动机考试和计算机系统的实验，所以dealine显得有些紧张。

1. 关于初接触“软件构造”课程；

应该多安排一些学时，个人对软件构造理论以及具体的“编程艺术”比较感兴趣，这门课给了我焕然一新的感觉。我能体会到这门课对未来实际工作的帮助是非常大的。