

**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 | 张高玮 |
| 学号 | 1180300407 |
| 班号 | 1803004 |
| 电子邮件 | 1462034631@qq.com |
| 手机号码 | 15630559577 |

**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

①安装JDK8，去官网下载，选择下载目录

②配置环境变量：计算机→属性→高级系统设置→高级→环境变量→系统变量→新建JAVA\_HOME变量，值为JDK安装目录

Path变量→编辑：输入%JAVA\_HOME%\bin

③官网下载、安装eclipse、Git

④根据老师实验课指导了解Junit创建、使用testcase的方法

⑤注册Github账号，根据给出的URL地址绑定学号，创建仓库

在这里给出你的GitHub Lab1仓库的URL地址（Lab1-学号）。

https://github.com/ComputerScienceHIT/Lab1-1180300407.git

# 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

要求1中已经有了提示，训练我们读写文件、书写Java代码的基础能力。而关于要求2则是训练我们阅读、分析Java代码的能力，分析产生幻方矩阵的函数的原理。两个函数中都需要关注异常处理部分。

### isLegalMagicSquare()

在P1的实验指导中已经有了一些提示：从文件中逐行读取字符串，然后将字符串根据split方法以“\t”进行划分，将每一个元素转化为整数进行加和，验证其是否为幻方矩阵。

我在完成实验过程中总结出以下几个“碰壁”点：

①读取文件②字符串分割、转化时的格式问题：不符合格式该怎样处理

③保存元素和时用静态还是动态数组保存，需要几个数组

针对点①，从百度上找到一个读取文件的模版，使用Bufferreader



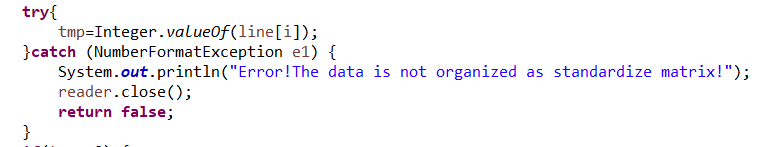






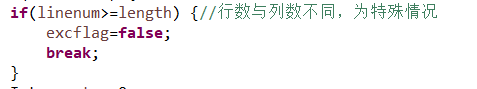


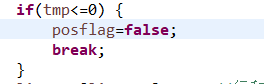
点②：首先，字符串转化为整数时采用try-catch捕获异常

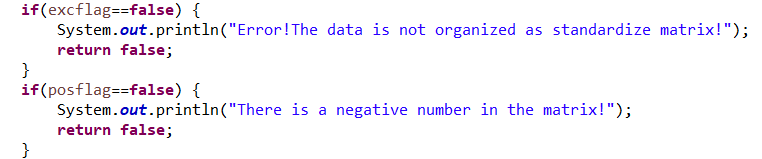


这样解决了不足“\t”情况以及浮点数情况，一旦出现则会捕获异常。

其次，设置excflag变量和posflag，如果出现非正整数以及行列不等情况，则设置为假，程序返回false并打印提示

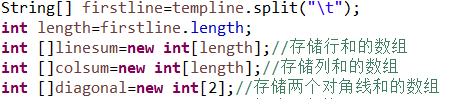
行列不同设置excflag

加和时同样可以检查非正整数，设置posflag

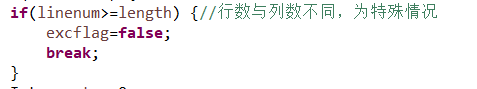


这里有一点是判断行列数是否相等，与第③点相关，放在下一点中

③保存加和问题：由于初学Java，对数据类型还不是很了解，写代码时并不知道List集合类，因此当时选择了普通数组来存储，但如果用普通静态数组，大小无法界定，因此我这里先计算矩阵第0行的元素个数length



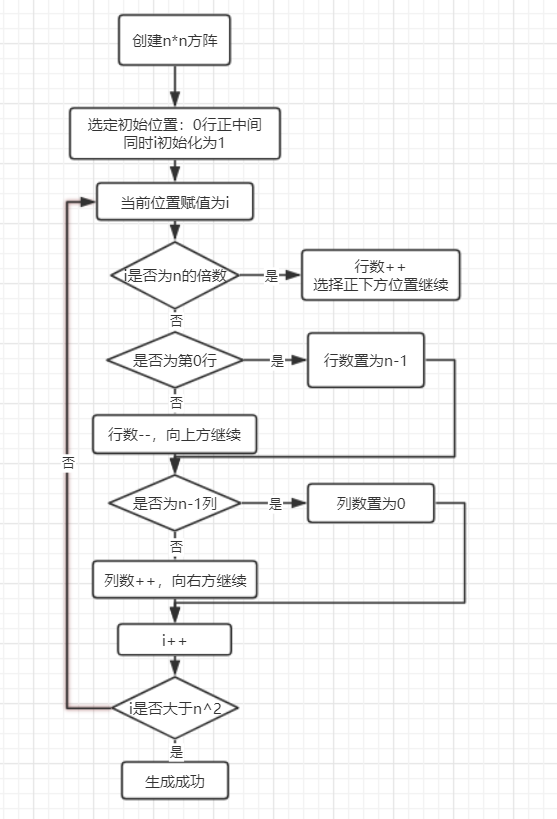
行列数都认定为length，这样如果是标准方阵并不会出现问题，而若不是标准方阵则可以在循环内部增添判断条件，直接让excflag置true，跳出循环而不会发生越界



我在实验过程中主要在这三个地方遇到了问题，解决了这些之后这个函数基本就完成了

### generateMagicSquare()

根据源代码给出的程序流程图如下：



对其分析如下：

①初始位置选择第0行的正中间，将1置于此，下一个整数从最后一行开始

②沿着左下至右上的对角线方向放置连续整数，行数递减，列数递增

③行数递减至第0行时，再置为最后一行继续循环

④列数递增至n-1列时，置为第0列继续循环

⑤当下一个应放置位置已有整数或者到达矩阵的右上顶点时，选择其下一行 的正下方位置作为代替（用i%n==0来判断）

产生异常的原因如下：

①n为偶数时矩阵左下角的元素会出现越界现象，因此出现异常

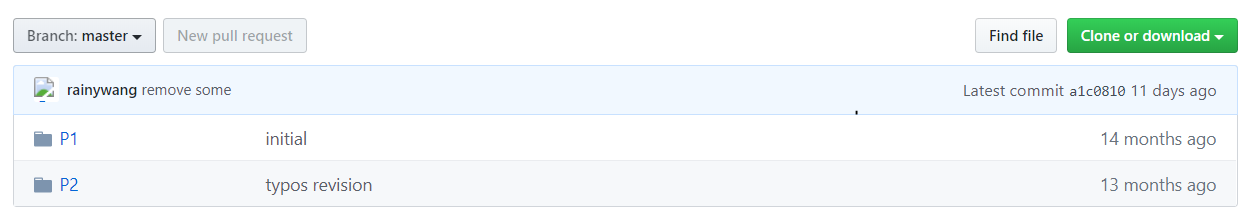
②n为负数时创建二维数组时便会出现异常，矩阵的大小不能是负数

## Turtle Graphics

这个任务与Turtle库联系紧密，通过调用其中的一些方法来实现一些图形操作，同时一些函数也要依赖一些数学知识，主要是想训练我们Java的基础编程能力以及使用架包、开发工具的能力，并让我们熟悉Git的使用以及Junit的基本使用方法。

### Problem 1: Clone and import

这里直接通过实验指导中的网站进入Github，通过绿色的Clone and download将源码下载。



根据老师实验课的指导，选择使用eclipse完成提交工作Team→Share Project

根据MIT网页提示，每完成一个函数便进行提交，进行版本控制

### Problem 3: Turtle graphics and drawSquare

这个函数非常基础，利用一个for循环，完成4次简单的前进（forward）-转向90度（turn）的操作即可

### Problem 5: Drawing polygons

这里需要设计两个函数：

首先需要根据多边形的边数计算内角，这里利用内角和定理即可算出，不过要注意边数小于等于2的特殊情况

然后画出这个多边形：利用上边的函数计算出内角，每一次转向的角度为外角，因此转向角也就得到了，利用一个for循环，完成边数次的“前进边长-转向外角”的操作即可。

### Problem 6: Calculating Bearings

这个问题要求我们完成两个函数

3.2.4.1 calculateBearingToPoint

根据现有转向角，计算当前点与目标点之间所需的转向角，这里主要利用Math库中的atan函数，根据正切值计算角度值，我这里根据当前点与目标点之间的位置关系划分九个维度：重合、左上、左、左下、下、右下、右、右上、上

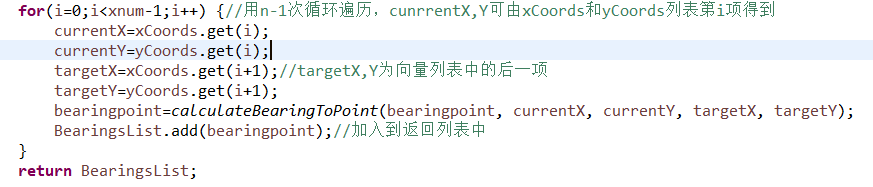
完成后通过Junit测试

3.2.4.2 calculateBearings

初始转向角为0，计算向量列表中每两个相邻点间所需转向角

利用3.2.4.1已完成的函数，用一个for循环，前一个元素作为当前点，后一个为目标点，上一次计算的转向角作为当前转向角，计算出的转向角存入列表并作为下一次计算时的当前转向角，遍历整个向量列表

核心部分代码如下：



### Problem 7: Convex Hulls

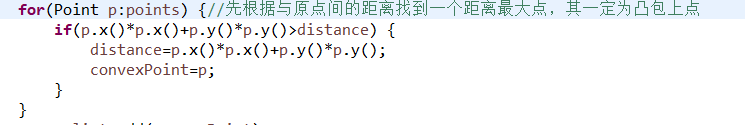
这个问题相比前面的问题复杂了许多，这里对凸包大致分析如下：

如果能找到一个凸包上的点，由于凸包要将所有点包围在内，因此如果利用已知点找到一个点：使得所有点都在该点与已知点的连线的一侧（无论左、右），则这一点必为凸包上点，如果其余点都位于连线一侧，说明从已知的凸包上点到该点所需的转向角最小，因此只需遍历点集，找到转向角最小的点即可。

这是总体的分析思路，但是实现细节时还有一些关键点需要注意

①点个数小于等于3的情况，所有点都是凸包上点，直接返回

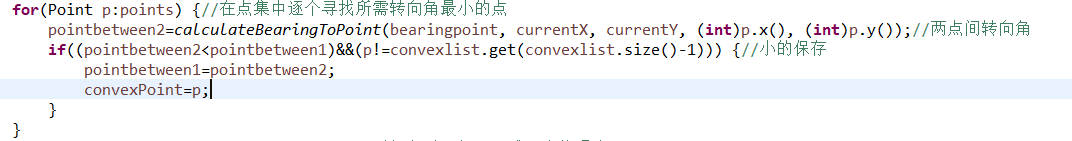
②找到一个一定在凸包上的点：这里选择整个坐标内距离原点最远的点，这个点一定是凸包图形的一个“角”，在凸包上，故遍历点集，找到距离最大点



③找到一个凸包上点后立刻将其作为新的起点进行搜寻

④关于转向角问题，这个问题导致的bug让我调试了很长时间：找到一个凸包上点后该怎样变化转向角

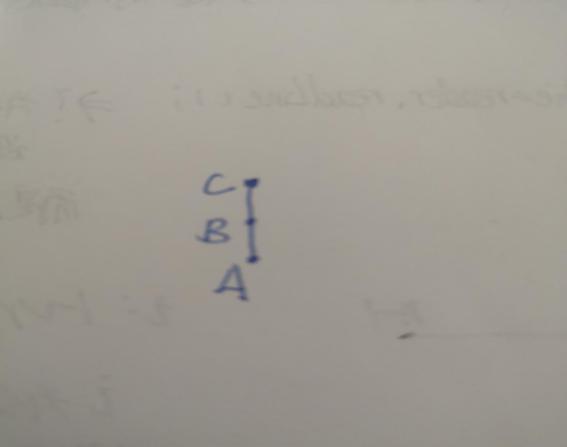
受到Problem6的影响，我一开始的解决方法是直接将新计算得到的calculateBearingToPoint作为新的转向角，这样写以后Junit的测试样例总是不能通过，后来经过画图分析发现如果转向角是这样选定的，那么转向角并不能代表两个目标点之间的旋转角度，而是两个旋转角度的差值，这个角度最小并不能保证其余点位于连线两侧。经过分析后终于明白，转向角应该采用累加方式，这样才能真正的表示两个点之间的旋转角度。



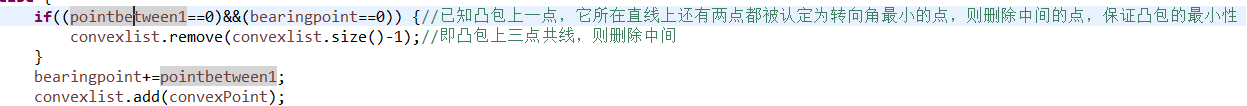


⑤循环终止问题：回到凸包上最开始找到的点时终止

⑥排除多余点问题：如图

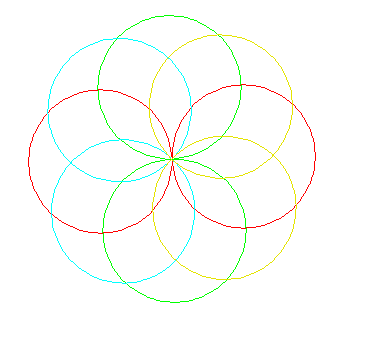


如果三点共线，那么两两之间的转向角都是0，一定会在循环中加入凸包中，但是凸包是最小点集，简单分析便可知道这种情况中间点一定是多余的，因此代码中要有对这一部分的判断、删除



### Problem 8: Personal art

这道题属于自由发挥的范畴，我这里是用了多个循环用多个圆形构成了一个图案。截图如下：



### Submitting

这里我是使用eclipse直接向Github提交，觉得比起直接使用Git可能更简单一些：右键点击→Team→Commit

## Social Network

这个问题的实质便是图问题：添加顶点、添加边等基础操作，会涉及很多基础数据类型：一维数组、二维数组、List类实现的动态数组等；求最小距离的算法，这一问题是需要解决的关键问题，通过这一任务可以加深对各种数据类型的了解，并锻炼我们将C语言已经实现的算法快速迁移至java实现的能力。

### 设计/实现FriendshipGraph类

首先设计该类的属性：顶点集以及表示边的邻接矩阵。顶点集我采用List类实现的动态数组本来邻接矩阵我也想用两个嵌套的List实现动态二维数组，但其他算法实现起来就会比较复杂便放弃了，使用了普通的二维数组。这里假定了顶点数不超过1000



然后设计方法：

1. 添加顶点：直接加入顶点集，但在加入之前要遍历顶点集检查是否重复添加，重复添加则需要打印提示
2. 添加边：①检查边的两个顶点是否重复②检查这两个顶点是否在顶点集中③检查正常后，通过List类的indexof方法得到其在顶点集中的序号，然后Edge数组中置1
3. 求两点距离函数：任务中的关键函数

根据实验指导中的提示，这里的总体思路是广度优先搜索求最短距离。相比其他算法，使用广度优先搜索的思路求最短距离比较简单，但实现起来有一处还是困扰了我很久，之后会介绍到。

①判断两个顶点是否在顶点集中，不在则打印提示并返回-1

②判断是否为同一顶点，若是，省去遍历操作，直接返回0

③关键部分：利用一般的广度优先搜索进行遍历，这里用List类创建一个数组作为队列来使用。

大体上与一般的广度优先搜索没什么不同，但困扰我的一点时如何保证这距离是最短的。一个顶点可能与多个顶点相连，这多个顶点只有全部遍历完成以后并且没有找到目标点时目前的距离才能进行加1操作，我把这多个顶点称作位于“同一级”，即同级元素遍历完全后距离才能加1.那么同级元素就需要额外的空间进行存储，并且随着遍历的进行，顶点在变，顶点的朋友也在变，因此同级元素也在变，因此当与一个顶点相关联的顶点遍历完成时这个同级元素需要及时更换。

因此我在这里的实现方法是新建两个列表：nowList和friendList

nowList中存储的是与目前遍历的元素同级的元素，friendList中存储的是nowList中元素的所有朋友元素。这样每遍历一个同级元素后，就将其从nowList中移除，遍历时的朋友元素都已经加入到friendList中，当nowList为空时说明同级遍历完成，将friendList中元素全部复制给nowList，再将friendList置空，便实现了同级元素的变换。具体代码如下：



### 设计/实现Person类

两个属性：name与visit。visit是用来标记在计算距离的遍历时其是否被访问过。

一个构造方法：根据名字构建对象

以及相关的get、set方法，对visit进行初始化的方法

### 设计/实现客户端代码main()

将实验指导中的测试用例作为main函数进行测试

### 设计/实现测试用例

测试策略如下：

首先根据图的连通性进行测试：没有边的图、有边但不连通的图、连通图

其次根据图的顶点数进行测试：一个顶点、两个顶点、n(n>2)个顶点

最后根据两个点间距离进行测试：两个点不连通，两个点重合、两个点距离为n(n>0)

详细测试用例请参见实现代码

# 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。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 任务 | 实际完成情况 |
| 2020-02-27 | 21:00-22:30 | 由于java是零基础，目标是熟悉java中类与对象的基本操作，熟悉文件的操作，学习调试 | 完成 |
| 2020-02-29 | 9:30-11:25 | 编写P1中的isLegalMagicSquare函数 | 整体构建完成，但对于4.txt，5.txt测试结果错误，存在bug |
| 2020-02-29 | 13:40-15:00 | 调试isLegalMagicSquare函数 | 找出bug并解决，完成 |
| 2020-02-29 | 19:00-20:00 | 分析generateMagicSquare函数，进行改进、测验，撰写实验报告 | 完成 |
| 2020-03-01 | 9:00-11:10 | 分析、解决P2 | 剩余convexHull、drawPersonalArt函数未完成 |
| 2020-03-02 | 9:00-11:00 | 完成P2中convexHull函数 | 未完成，Junit中有一个样例未通过 |
| 2020-03-02 | 15:00-16:40 | 调试、解决convexHull函数中bug | 完成 |
| 2020-03-04 | 14:00-15:10 | 完成drawPersonalart函数 | 完成 |
| 2020-03-04 | 18:00-19:50 | 完成P3 | 测试用例未编写，实验指导中的样例通过 |
| 2020-03-06 | 15:00-16:30 | 完成P3测试用例编写 | 测试用例测试过程中发现程序存在bug，算法有问题，后来解决了问题，完成目标 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 不熟悉java中的数据类型：尤其是集合类与动态数组的实现与使用 | 网上查询，通过java API进行查阅，参考其中的一些小例子进行编写 |
| P2凸包函数的设计以及其中一些特殊情况考虑不周 | 网上查阅凸包相关定义与算法，确定总体思路：利用最小转向角来设计凸包。其中转向角的计算通过调试改正了错误。还有三点共线的特殊情况是从测试用例中受到了启发才想到这种特殊情况 |
| P3最小距离的同级元素问题 | 画图分析，debug逐步分析代码，分析何时才能对距离进行加1 |

# 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

①掌握了java编程的基本技术，熟悉了java中的基本数据类型

②对软件中的鲁棒性质量目标有了一些体会，设计时总要“惦记”这异常情况

③学会了利用Junit进行单元测试，利用debug进行调试

## 针对以下方面的感受

1. Java编程语言是否对你的口味？

目前感觉还可以，对于集合类数据类型非常感兴趣，觉得它们很强大，对于类和对象的编程思想还不是很了解

1. 关于Eclipse IDE

觉得很全面，已经包括很多的java库，不需要下载什么插件，也可以在其上进行Git提交，已经包括了Junit，全面、便捷

1. 关于Git和GitHub

初始使用感觉很繁琐，但是能方便地记录每一次的版本变化情况，利于维护、修改代码

1. 关于CMU和MIT的作业

对于有语言基础但是是java初学者的我，能够帮助我快速地上手，熟悉java

1. 关于本实验的工作量、难度、deadline

工作量、难度适中，本实验deadline较为宽松

1. 关于初接触“软件构造”课程

对于我来说，拓宽了编程视野，以前是个只鼓捣算法企图提高效率的人，现在编程时会考虑其它的一些质量目标，比如鲁棒性、可维护性