

**2019年春季学期  
计算机学院《软件构造》课程**

**Lab 1实验报告**

|  |  |
| --- | --- |
| 姓名 | 李奕鹏 |
| 学号 | 1170800716 |
| 班号 | 1737101 |
| 电子邮件 | 17895987@qq.com |
| 手机号码 | 18800418032 |

**目录**

[1 实验目标概述 1](#_Toc2711157)

[2 实验环境配置 1](#_Toc2711158)

[3 实验过程 2](#_Toc2711159)

[3.1 Magic Squares 2](#_Toc2711160)

[3.1.1 isLegalMagicSquare() 2](#_Toc2711161)

[3.1.2 generateMagicSquare() 2](#_Toc2711162)

[3.2 Turtle Graphics 4](#_Toc2711163)

[3.2.1 Problem 1: Clone and import 4](#_Toc2711164)

[3.2.2 Problem 3: Turtle graphics and drawSquare 4](#_Toc2711165)

[3.2.3 Problem 5: Drawing polygons 5](#_Toc2711166)

[3.2.4 Problem 6: Calculating Bearings 5](#_Toc2711167)

[3.2.5 Problem 7: Convex Hulls 6](#_Toc2711168)

[3.2.6 Problem 8: Personal art 6](#_Toc2711169)

[3.2.7 Submitting 6](#_Toc2711170)

[3.3 Social Network 6](#_Toc2711171)

[3.3.1 设计/实现FriendshipGraph类 7](#_Toc2711172)

[3.3.2 设计/实现Person类 7](#_Toc2711173)

[3.3.3 设计/实现客户端代码main() 7](#_Toc2711174)

[3.3.4 设计/实现测试用例 7](#_Toc2711175)

[3.4 Tweet Tweet 9](#_Toc2711176)

[3.4.1 Problem 1: Extracting data from tweets 9](#_Toc2711177)

[3.4.2 Problem 2: Filtering lists of tweets 9](#_Toc2711178)

[3.4.3 Problem 3: Inferring a social network 9](#_Toc2711179)

[3.4.4 Problem 4: Get smarter 9](#_Toc2711180)

[4 实验进度记录 10](#_Toc2711181)

[5 实验过程中遇到的困难与解决途径 10](#_Toc2711182)

[6 实验过程中收获的经验、教训、感想 11](#_Toc2711183)

[6.1 实验过程中收获的经验和教训 11](#_Toc2711184)

[6.2 针对以下方面的感受 11](#_Toc2711185)

# 实验目标概述

训练基本Java 编程技能，能够利用Java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。

# 实验环境配置

1. 下载安装Java jdk

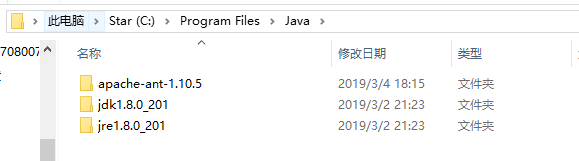


图1 JDK安装目录

1. 配置环境变量

①右击计算机——属性——高级系统设置

②点击环境变量，进入后在系统变量一栏点击新建，然后输入如下：

名称：JAVA\_HOME

变量值：C:\Program Files\Java\jdk-1.8.0\_201

③用户变量增加CLASS\_PATH

名称：CLASS\_PATH

变量值：.;%JAVA\_HOME%\lib;%JAVA\_HOME%\lib\tools.jar

④在系统环境变量中有一个path的环境变量，选中后选择编辑，将JAVA\_HOME添加进去 %JAVA\_HOME%\bin;

1. 安装Eclipse

我在官网下载了安装包，到本地安装即可

1. 安装Git以及环境配置

①生成RSA key 过程

（1）在lab文件夹下，右键打开git bash 执行 命名：ssh-agent bash

（2）生成RSA密钥，执行命令：ssh-keygen -t rsa -C [1170800716@qq.com](mailto:1170800716@qq.com)

（3）根据指示路径文件中的密钥到ssh，执行的命令：ssh-add

（4）将密匙登入github并关联进去

5. 修改远程pull（push）方式

执行git remote set-url origin [git@github.com:ComputerScienceHIT/Lab1-1170800716.git](mailto:git@github.com:ComputerScienceHIT/Lab1-1170800716.git)

GitHub Lab1仓库的URL地址：https://github.com/ComputerScienceHIT/Lab1-1170800716

# 实验过程

## Magic Squares

编写一个MagicSquare.java文件利用main方法对下载的文件里面的矩阵进行判断是否符合每行每列以及对角线的和是否为同一个常量，这个函数需要有足够的健壮性，即需要能够判断题目提出的要求，优雅地抛出异常并且退出程序。然后是理解一段给出的代码添加注释并且画出流程图，并更具要求判断输入值的情况并且抛出异常。最后使用git将文件push到github仓库里面去。

### isLegalMagicSquare()

1. 考虑到不是在窗口输入，所以一开始要先创建能够读取文件，首先完成并测试读取文件的功能。
2. 在文件功能测试完毕以后思考文件时按照行读入的，于是根据提示使用spilt方法来分割每一行为一个字符数组，并将这些数组加到list里面去。然后通过第一行的数组长度和后面每一行的数组长度以及行数来比较，如果不同，即矩阵不为规则n\*n矩阵，报错，抛出异常。
3. 完成上面的工作以后就是将字符串转化为int了，在这个过程中可以先判断是否是整数，如果不是整数的话抛出异常。
4. 然后现在是已经转化为二维数组了，然后检查每行每列以及对角线上面的数的和是否相等，相等返回true，不等返回false

### generateMagicSquare()

1. 代码片段：（添加了注释）

int row = 0, col = n / 2, i, j, square = n \* n;

**for** (i = 1; i <= square; i++) {

magic[row][col] = i; */\*把1放在第一行正中间\*/*

**if** (i % n == 0)

row++;

*/\*如果这个数所要放的格已经有数填入，就把它放在前一个数的下一行同一列的格内\*/*

**else** {

*/\*如果这个数所要放的格已经超出了顶行那么就把它放在底行，仍然要放在右一列\*/*

**if** (row == 0)

row = n - 1;

**else**

row--;

*/\*如果这个数所要放的格已经超出了最右列那么就把它放在最左列，仍然要放在上一行\*/*

**if** (col == (n - 1))

col = 0;

**else**

col++;

}

}

代码1 部分generateMagicSquare()以及注释

2.流程图如下：

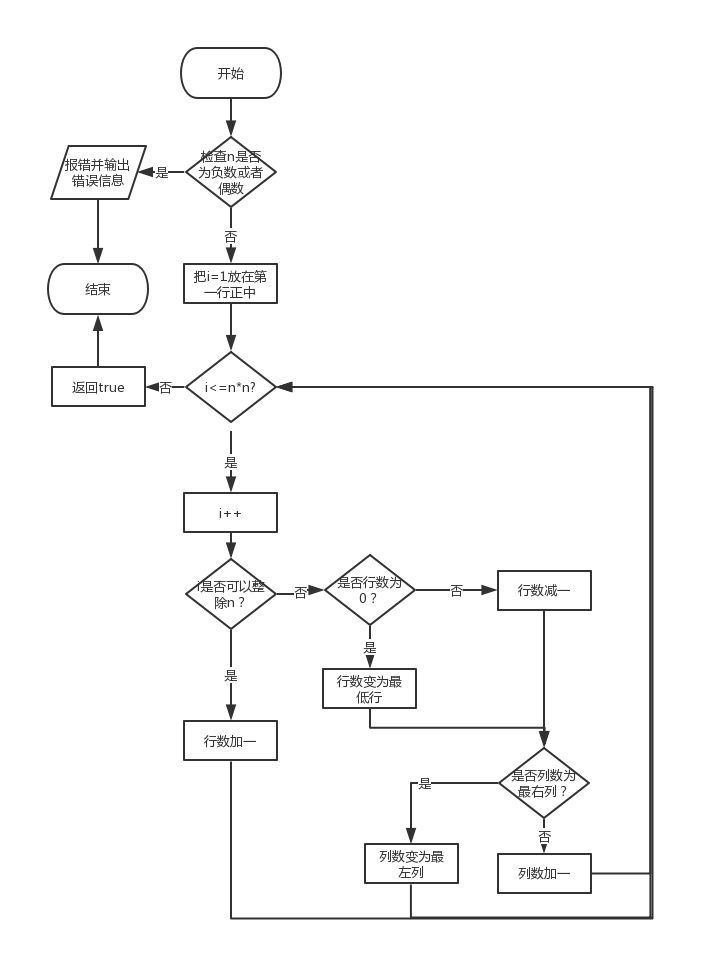


图2 函数流程图

## Turtle Graphics

这个呢是已经给好了大部分的代码，需要填充一些TurtleSoup.java里面的函数，来使得已经编写好的JUnit测试能够通过，并且学习如何通过git命令行来上传文件。

### Problem 1: Clone and import

1. 获取代码

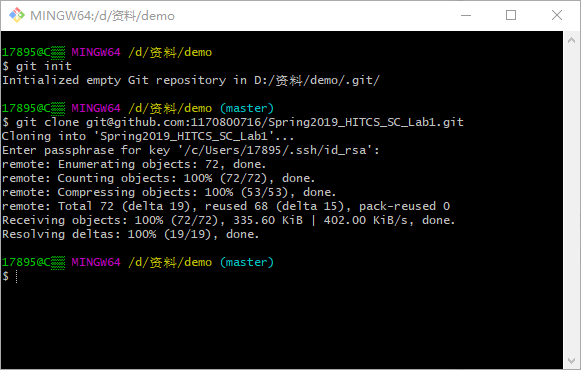


图3 使用git获取代码

2.然后我将代码完整复制到我的实验仓库文件夹下面统一管理以及上传。

### Problem 3: Turtle graphics and drawSquare

1. 我将代码打开以后查看了Turtle.java看了一下函数，意图就是画一个正方形，然后就很容易想到画一定长度，然后转个九十度然后再画这样四次就是一个正方形啦，然后编写了代码，运行截图如下：

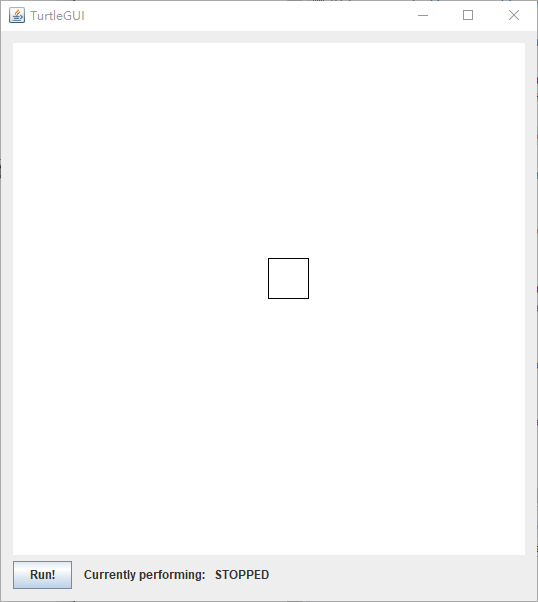


图4 drawSquare运行截图

### Problem 5: Drawing polygons

1.这里我也是根据前面写完的计算规则多边形内角的函数，根据算出来的度数，通过控制循环次数以及角度的转动来编写，截图如下：

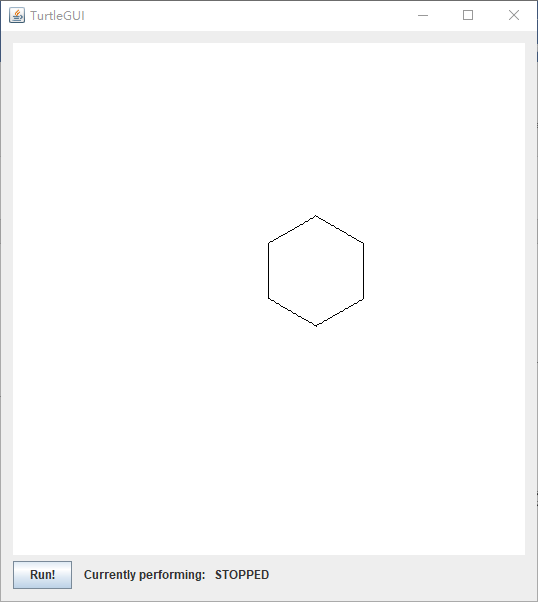


图5 Drawing polygons运行截图

### Problem 6: Calculating Bearings

1.这个函数还是相对来说比较棘手的，我通过计算了两个点的直线与x轴的夹角，然后通过条件判断（x1,x2,y1，y2的相对大小关系），不同的情况进行不同的处理，来实现输出角度的。

### Problem 7: Convex Hulls

1.这个函数我利用了上一个函数，上网搜索了所谓的卷包裹算法，然后利用上面的计算角度的函数，先选一个最左下角，然后通过逆时针旋转找到角度最小的那一个点，来判断下一个点，然后特判几种比较特殊的点，最后把所有的点卷成一个凸包。

### Problem 8: Personal art

1.这个函数我就天马行空了一点，按照自己的审美做了一个自我感觉还行的图，运行截图如下：

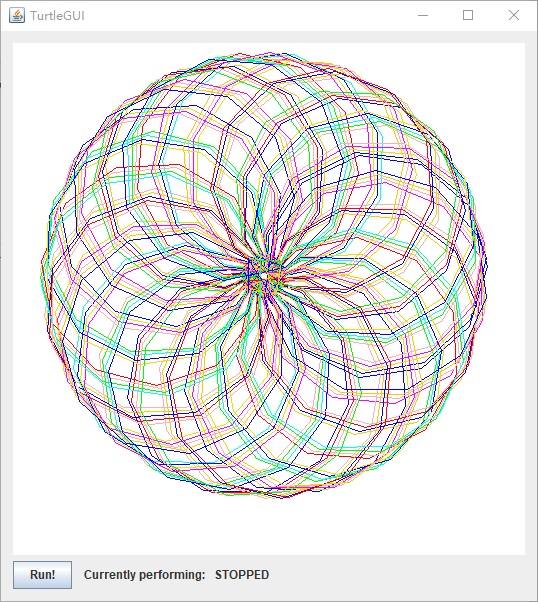


图6 drawPersonalArt运行截图

### Submitting

这里我先git add了我修改了的文件，然后git diff –staged保存了我的相关修改，然后git commit了。最后git push origin master完成所有的提交过程。

## Social Network

在这个任务里面就是需要创建一个无向图（为了以后的扩展目前创建为有向图），然后通过addVertex()以及addEedg()等函数将图的元素填充进去，然后实现getDistance()的方法来实现计算两点之间的距离，然后需要注意实验里面不允许使用static以及每个方法要使用合适的域，然后需要在编写的同时自己编写测试用例来验证自己的想法是不是正确。

### 设计/实现FriendshipGraph类

1.我的思路就是创建一个图，然后往图里面添加顶点以及边什么的，然后计算两个点之间的最短路径。

2.我的实现方法是：因为是无权图，所以这里我选择了邻接表，方便一点，然后我设计了一个结点类和链表类来储存我的每一个结点的出度。然后在计算距离上面按照要求使用广度优先搜索算法来实现搜索最短路径，最后输出结果。在主方法以及JUnit里面测试。

3.结果：能完整实现功能。

### 设计/实现Person类

这个类就是一个简单的构造方法，以及构建一个getName()方法来获取以及返回Person名字的字符串。

### 设计/实现客户端代码main()

这里的main方法就是手册上面给出的main方法，用来调试以及测试。

### 设计/实现测试用例

我将测试用例放入到JUnit里面进行测试了，我首先添加了一个如下图的图：



图7 测试用图

然后将这个图添加到我的JUnit测试里面去，具体代码如下：

**@Test**

**public** void getDistanceTest() **throws** Exception {

FriendshipGraph graph = **new** FriendshipGraph();

Person A = **new** Person("A");

Person B = **new** Person("B");

Person C = **new** Person("C");

Person D = **new** Person("D");

Person E = **new** Person("E");

Person F = **new** Person("F");

Person G = **new** Person("G");

addVertex(graph, A);

addVertex(graph, B);

addVertex(graph, C);

addVertex(graph, D);

addVertex(graph, E);

addVertex(graph, F);

addVertex(graph, G);

addEdge(graph, A, D);

addEdge(graph, D, A);

addEdge(graph, A, C);

addEdge(graph, C, A);

addEdge(graph, B, C);

addEdge(graph, C, B);

addEdge(graph, E, C);

addEdge(graph, C, E);

addEdge(graph, C, G);

addEdge(graph, G, C);

assertEquals(2, getDistance(graph, A, G));

assertEquals(1, getDistance(graph, B, C));

assertEquals(3, getDistance(graph, D, G));

assertEquals(0, getDistance(graph, C, C));

assertEquals(-1, getDistance(graph, B, F));

assertEquals(2, getDistance(graph, E, B));

assertEquals(2, getDistance(graph, B, E));

}

代码2 getDistanceTest()代码片段

测试结果如下：

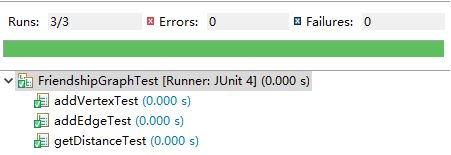


图8 测试结果

## Tweet Tweet

在这个任务中，需要做的是，首先把先计算时间戳，以便于能够获取所测试的tweet里面最早好最晚的时间间隔，然后是获取每条推特里面提到的人以便于后面的problem 3里面挑选最有影响力的人，然后problem 2里面的writtenby函数是看所有的推特里面相应的人发布了多少条tweet，以及inTimespan就是筛选出在给定时间内的tweet，最后的problem 3里面就是猜每个人follow了那些人，然后是那些人最有影响力，以及problem4里面就是要优化那个测试的算法。

### Problem 1: Extracting data from tweets

首先是编写测试根据要求我扩充了一下tweet然后进行相对应的测试。

在这个任务中需要实现获取最早的和最晚的tweet之间的时间间隔，所以我就把所有的tweet都遍历一遍，从中挑出最早的和最晚的，然后将其储存起来。

第二个函数我使用了正则表达式，首先判断每一条tweet里面是否含有“@”然后如果有的话再对这一条tweet进行更进一步的处理，然后特判一下函数说明里面提到的邮箱的问题，然后就完成了。

### Problem 2: Filtering lists of tweets

在这个任务里面徐要做的是提取出来指定哪个人写的那些tweet这个就很简单，就遍历一下，然后看谁的名字符合要求就提取出来。最后加到list里面。然后就是计算某一个时间段里面发布的哪些tweet，和上面一样，遍历一下，筛选出符合条件的加到list里面。

### Problem 3: Inferring a social network

在这个任务里面需要做的是猜谁follow了谁，根据spec里面的说法，就是谁@了谁，谁就Follow了谁，根据这个想法，我们可以使用前面编写的提取某一个tweet里面@了谁，这样的话就基本上可以直接导出谁follow谁了。

然后就是计算影响力再排序，这个里面就是从前面第一个函数里面提取出来，只需要被follow的那一个表，然后看里面出现的次数来增加权重并且以此为依据来输出影响力序列。

### Problem 4: Get smarter

在这个任务里面我采用的是方案一，就是如果两个人用了同一个tag的话，这两个人就互相关注了，实现这个和提取名字的那个很相似，就是提取出来一份tag，然后看后面的tweet里面有没有相同的，如果有相同的话，就让他们互相关注一下。

# 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 |
| 2019-02-25 | 14:00-15:30 | 编写问题1的isLegalMagicSquare函数并进行测试 | 遇到困难，未完成 |
| 2019-02-26 | 10:00-12:00 | 编写问题1的isLegalMagicSquare函数并进行测试 | 按计划完成 |
| 2019-02-26 | 14:00-15:30 | 绘制问题1的generateMagicSquare函数流程图并进行扩展 | 按计划完成 |
| 2019-02-27 | 15:30-23:00 | 任务2完成到Problem6 | 按计划完成 |
| 2019-02-28 | 18:00-23:00 | 任务2基本完成 | 按计划完成 |
| 2019-03-01 | 18:00-23:00 | 完成任务3 | 遇到困难未完成 |
| 2019-03-02 | 10:00-12:00 | 完成任务3 | 遇到困难未完成 |
| 2019-03-02 | 14:00-23:00 | 完成任务3 | 遇到困难未完成 |
| 2019-03-03 | 08:00-12:00 | 完成任务3 | 按计划完成 |
| 2019-03-03 | 14:00-23:00 | 完成任务4 | 遇到困难未完成 |
| 2019-03-04 | 14:00-15:00 | 完成任务4 | 遇到困难未完成 |
| 2019-03-04 | 20:00-23:00 | 完成任务4 | 遇到困难未完成 |
| 2019-03-04 | 18:00-21:00 | 完成任务4 | 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 |
| 任务1的isLegalMagicSquare函数中文件读入方式出了问题，没有看到提示里面的split方法，导致没有做出来 | 重构了代码，将原来的按个读入换成了按行读入，然后使用split方法来将文件读取进入矩阵里面再进行验证和计算。 |
| 任务2的凸壳算法实在做不出来了不知道问题处出在了哪里 | 通过自己编写main方法测试函数，检测相应的值，然后找到了问题出在上一个计算角度那里，然后重写了上一个函数再回来看这个函数就可以正确运行了。 |
| 任务3里面有一个要求是，每个人的名字不同，这可咋整啊，我又不能遍历所有的对象，又不让创建静态变量来储存已经加入的人名，不知道怎么办了 | 后来在舍友的提示下，我将判断的机制加入到了addVertex函数里面，然后就可以不使用static然后同样的保存了数据在主类FriendshipGraph里面。 |
| 在P3里面的JUnit除夕拿了问题，怎么样也运行不了，在Stack Overflow里面也找不到答案。 | 最后我发现导入的P2的能使用，P3不能用，这两个项目唯一不同的地方是JRE版本不一样，然后我把eclipse卸载了，然后把jdk卸载了，统一了jdk为java8，最后再打开就可以运行了。 |

# 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

1. 做实验的时候一定要沉住气，不急不躁，慢慢来，会做出来的，不可做到一半就想着放弃。
2. 写任何代码都需要提前构思，不可以有一点思路就写，有了较为完整地思路以后再开始敲键盘，不然很容易出现问题然后重头来。
3. 一定要多思考
4. 不会就去问，没什么不好意思的

## 针对以下方面的感受

1. Java编程语言是否对你的口味？

说实话，不太符合我的口味，一方面java看起来不如c++简单易懂，另一方面java又不如python易上手实用性强。

1. 关于Eclipse IDE

这个IDE给我的感觉，不能自动填充代码有点不好，不过对于我们初学者来说，这个可能是一个训练自己的好机会，其他的就是比idea稳定吧，然后比idea丑。

1. 关于Git和GitHub

这也是我第一次接触git，git可以说是一大利器，完全托管代码，可以随时恢复到想要的版本还会自动标记哪里修改了，然后不同的分支可以为不同功能的实现埋下方便，可以说是一大利器，就是有时候github需要科学上网非常不好。

1. 关于CMU和MIT的作业

内容非常好，很锻炼人的编程能力，以及思维能力，还有英语能力哈哈哈。

1. 关于本实验的工作量、难度、deadline

可以说，工作量巨大，难度巨大（对我来说）deadline还行，还算有时间

1. 关于初接触“软件构造”课程

对于这一门课程，我的感觉是对于一个程序员来说是必经之路，算是踏上起点了吧，这门课的重要性是毋庸置疑的。