

**2019年春季学期  
计算机学院大二软件构造课程**

**Lab 1实验报告**

|  |  |
| --- | --- |
| 姓名 | 袁康 |
| 学号 | H170301801 |
| 班号 | 1703001 |
| 电子邮件 | isaacyuan@foxmail.com |
| 手机号码 | 17314957761 |

**目录**

[1 实验目标概述 1](#_Toc3390467)

[2 实验环境配置 1](#_Toc3390468)

[3 实验过程 1](#_Toc3390469)

[3.1 Magic Squares 1](#_Toc3390470)

[3.1.1 isLegalMagicSquare() 1](#_Toc3390471)

[3.1.2 generateMagicSquare() 2](#_Toc3390472)

[3.2 Turtle Graphics 3](#_Toc3390473)

[3.2.1 Problem 1: Clone and import 3](#_Toc3390474)

[3.2.2 Problem 3: Turtle graphics and drawSquare 4](#_Toc3390475)

[3.2.3 Problem 5: Drawing polygons 4](#_Toc3390476)

[3.2.4 Problem 6: Calculating headings 4](#_Toc3390477)

[3.2.5 Problem 7: Personal art 4](#_Toc3390478)

[3.2.6 Submitting 5](#_Toc3390479)

[3.3 Social Network 5](#_Toc3390480)

[3.3.1 设计/实现FriendshipGraph类 5](#_Toc3390481)

[3.3.2 设计/实现Person类 6](#_Toc3390482)

[3.3.3 设计/实现客户端代码main() 6](#_Toc3390483)

[3.3.4 设计/实现测试用例 7](#_Toc3390484)

[3.4 Tweet Tweet（选作，额外记分） 7](#_Toc3390485)

[4 实验进度记录 8](#_Toc3390486)

[5 实验过程中遇到的困难与解决途径 8](#_Toc3390487)

[6 实验过程中收获的经验、教训、感想 8](#_Toc3390488)

# 实验目标概述

通过完成四个较为基础的java基础程序，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。

# 实验环境配置

由于在这门课程开设之前已经配置好了环境，写过简单的Java程序，所以这次实验环境配置没有遇到问题。

我的GitHub Lab1仓库的URL地址如下：<https://github.com/ComputerScienceHIT/Lab1-H170301801XXX>

# 实验过程

## Magic Squares

本题要求完成一个程序可以读一个指定的纯数字文件，判断文件中的数据是否输入格式完全正确并且可以构成一个幻方（行列和对角线元素合都相同）。对于题目中给定的幻方生成函数，测试是否可以输出一个正确的幻方并完成写文件的操作。本题让我学习到了编写 java 程序的一些基本操作，如简单数据结构的操作、条件判断、循环分支、异常处理、函数返回、文件操作等。

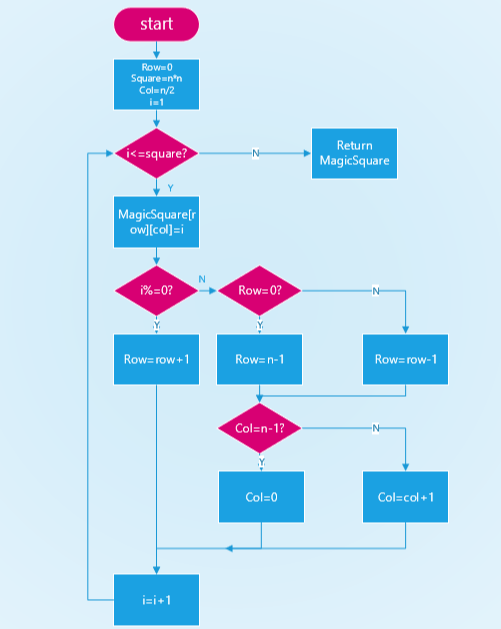
### isLegalMagicSquare()

首先设置一个二重循环，在这个二重循环内，实现了从文件内读入数据，同时计算出每一行，每一列，斜对角线的和；同时在遍历square的同时，判断读入的数据是否是非负的整数，是否每一行的数量都相等，如果有异常，就输出对应的错误。

之后在一个一重循环内，比较每一行的和，每一列的和是否相等，同时判断行和列的数量是否相等，如果不相等，就输出对应的报错信息。

最后比较行的和与斜对角线数字的和是否相等，如果相等，则返回true，表明这是一个MagicSquare，如果不相等，就返回false，表明这不是agicSquare。

### generateMagicSquare()



Exception in thread "main"

java.lang.ArrayIndexOutOfBoundsException

: 12

at MagicSquare.generateMagicSquare(

MagicSquare.java:17

)at MagicSquare.main(

MagicSquare.java:121

)

这个报错的原因是数组的下标访问越界，因为输入的n是偶数，在计算过程中，按照n是奇数的边界条件判断就会出现这种异常情况，在下标超过数组的大小时，没有相应的判断检测。

Exception in thread "main"

java.lang.NegativeArraySizeException

at

MagicSquare.generateMagicSquare(

MagicSquare.java:11

)

at MagicSquare.main(

MagicSquare.java:121

)

出现这个错误的原因是因为数组的下标是负数，而数组的下标必须是非负数。出现负数下标的原因是传入函数的n是负数，在执行过程中，就会出现负数的下标去访问数组。

解决这两个异常的思路是 在generateMagicSquare()函数的入口处，判断是否是非负整数，是否是奇数，就可以避免这两种异常情况的发生。

## Turtle Graphics

这个实验是实现turtle框架中的一些功能函数，从而实现最终使用这个turtle工具画出所想的图像。这个实验要求自己能理解整个turtle工具的大体结构，理解要实现的功能函数的原理，就可以顺利的完成这个任务。

### 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。

在合适的目录下， 在命令行工具下依次输入以下指令就可以完成。

git clone https://github.com/rainywang/Spring2019\_HITCS\_SC\_Lab1.git

mkdir Lab1 #创建本地项目文件夹

cd Lab1

git init #创建本地git仓库

提交代码的三步命令：

1.【git add 文件名.后缀名】将文件加入到缓存区

2.【git commit -m"注释"】将缓存区的文件添加到分支（可以多次add一次commit）

3.【git push origin master】将分支推送到github

### Problem 3: Turtle graphics and drawSquare

这个函数需要用到一个一重循环结构，在循环内部，只需要直走传入参数sideLength长的距离，然后转90度，循环次数是4，这样这个正方形就画出来了。

### Problem 5: Drawing polygons

这个任务分为两个函数，第一个函数calculatePolygonSidesFromAngle 是用来根据传入的多边形边的数量计算内角的角度，这一计算的理论依据是 多边形的外角和是360度来确定的，可以先算出每个外角的度数，从而计算出内角的度数。

然后像画正方形一样，设置一个一重循环，循环次数是边的个数，每次循环都先走变得距离，然后转过计算出的角度，就可以画出规定的规则多边形。

同时发现，当边数大于20时，这个多边形的形状已经接近一个圆了，只要边的个数足够多，就可以无限趋近于圆了。

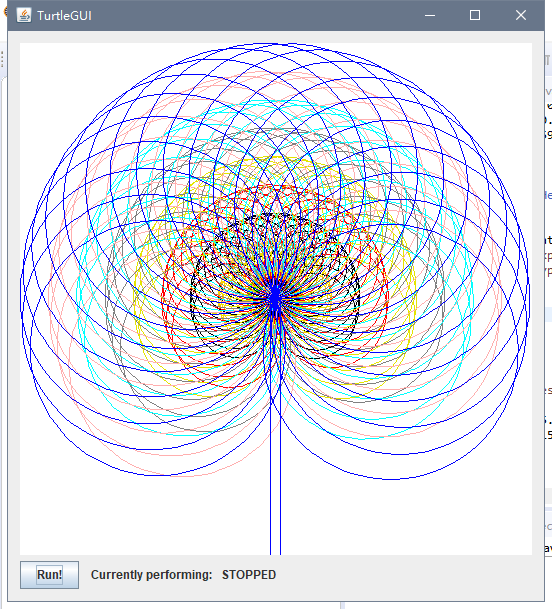
### Problem 6: Calculating headings

首先计算出目标点与当前点形成的角度，然后减去当前点的朝向角度就可以算出需要转过的角度。为了计算目标点与当前点形成的角度，需要先算出角对应的正切值，然后调用Math库中的atan()函数，计算出角度的弧度值，再根据弧度值，换算成角度值，同时还要根据得到的值，判断角度的象限，写出正确的分支结构语句，才能得到正确的角度值。这是这个实验最难的地方。

### Problem 7: Personal art

在这个问题中，我设计了一个近似于蘑菇的图案，由蘑菇帽和蘑菇柄两部分组成。其中蘑菇帽部分是由一个二重循环组成，外层循环是用来控制蘑菇的半径，每次循环都使蘑菇的半径有规律的变大，同时控制画笔颜色规律的变化，内层循环是用来控制画圆，围绕一个中心点，在一定的角度范围内画出圆。其中颜色的规律变化是通过将各种颜色值放入一个数组内，使用外层循环的i值来更换画笔的颜色，使得蘑菇的每一层颜色都不相同，达到美观的效果。

蘑菇柄的实现，是调用画直线的函数，让画笔画一个长条矩形。通过这个图画，我就同时验证了之前设计的全部功能函数的正确性。



### Submitting

提交代码的三步命令：

1.【git add 文件名.后缀名】将文件加入到缓存区

2.【git commit -m"注释"】将缓存区的文件添加到分支（可以多次add一次commit）

3.【git push origin master】将分支推送到github

## Social Network

这个任务是一个从0 开始自己构造类的实践，需要完全自己去设计类的成员变量，类的成员函数，以及每个变量，函数的权限等级，兼顾函数的restful以及安全性，是对面向对象编程的一次重要实践。

### 设计/实现FriendshipGraph类

在这个类中，成员变量有一个关系矩阵，一个用来统计图中人数的personNum变量其中关系矩阵我使用的是ArrayList类，这个类有个好处是可以根据图的人数来动态的规定数组的大小，避免了直接使用数组造成的空间浪费，同时当人数增多时，可以动态的增加空间来存储信息，不会发生溢出。当然，使用关系矩阵的缺点是当人际间的关系不太复杂时，空间消耗较大，当人数过多时，计算机的内存不够用，就无法申请到足够的空间。

在这个类中，有三个成员函数，一个是addVertex，这个函数是用来为关系图添加新的人，需要扩展关系矩阵的大小，判断这个人是否已经存在于关系矩阵中，判断关系矩阵中是否已经有和这个人名字相同的Person对象，如果不满足条件，就输出提示信息，停止执行程序。addEdge 这个函数是将两个人之间建立关系，对应的代码是将关系矩阵中对应的值由false变成true。

getDistance 这个函数是从第一个传入的person对象出发，开始广度优先搜索，寻找到第二个传入的Person对象的最短路径就是所求的distance，这个函数中需要用到队列，一个标记每个人是否被遍历过的布尔数组，需要用到广度优先搜索算法。

### 设计/实现Person类

这个类中有两个Private属性的变量，一个是Person对象的名字，一个是对应的Id,用以区分两个person对象，因为这两个成员变量都是Private属性，不能直接从外部直接访问和修改，所以提供了setId，getId,getName这三个用于操作成员变量的函数，同时还有一个构造函数，用于给person对象赋一个初始的名字。

### 设计/实现客户端代码main()

**public** **static** **void** main(String[] argv) {

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

Person rachel = **new** Person("Rachel");

Person ross = **new** Person("Ross");

Person ben = **new** Person("Ben");

Person kramer = **new** Person("Kramer");

graph.addVertex(rachel);

graph.addVertex(ross);

graph.addVertex(ben);

graph.addVertex(kramer);

graph.addEdge(rachel, ross);

graph.addEdge(ross, rachel);

graph.addEdge(ross, ben);

graph.addEdge(ben, ross);

System.***out***.println(graph.getDistance(rachel, ross));

System.***out***.println(graph.getDistance(rachel, ben));

System.***out***.println(graph.getDistance(rachel, rachel));

System.***out***.println(graph.getDistance(rachel, kramer));

}

### 设计/实现测试用例

我的测试分为两种测试思路，一种是靠大循环次数来验证稳定性和效率，同时也能测出正确性，在这个测试用例中，我设了1000次循环，即往关系表中添加1000个person对象，添加1000个边，最后比对getdistance的运行结果是否正确。

**for** (**int** i = 0; i < 1000; i++) {

persons[i] = **new** Person(String.*valueOf*(i));

graphTest.addVertex(persons[i]);

}

第二种测试用例就是手动添加一些人，添加一些复杂的边，形成一个复杂的关系图，然后手动计算某些人之间的距离，与getDistance函数计算出的值进行比较，如果有不同的值，就无法通过jtest的测试。

## Tweet Tweet（选作，额外记分）

Problem1:

筛选tweet的时间段，思路就是遍历整个tweet列表，然后找出最早的发表时间和最晚的发表时间，就是所求的时间段。

提取被提到的人：我使用了java的正则表达式匹配，表达式如下

Pattern regex = Pattern.*compile*("[^\\w\\-]@[\\w\\-]+");

然后将匹配到的字符串的@之前的内容剪掉，剩余的就是提到的用户的username.

测试：我在测试用例中编写了4条tweet信息，然后对我写的程序使用junit进行测试。

Problem2:

筛选某个用户发布的所有tweet信息，这个就是简单的遍历所有tweet信息，然后找到特定用户的信息即可。筛选某个时间段内的信息的思路同上。

筛选用户发的text中的某句话，这个依然是使用正则表达式去匹配要求的字符串。

Problem3：

得到guessFollowsGraph，就是首先筛选得到没有重复用户的一个用户列表，然后按照这个用户列表去匹配它们@过的用户，就得到了FollowsGraph。

然后在FollowsGraph中找到follow数量最多的用户，需要遍历FollowsGraph，然后将对应的用户和它们的follow数存在一个hashMap中，key是methioned用户，value是follow他们的数量，最后调用collection对象的sort方法，将存于hashMap中的无序键值对转换成从大到小的list对象，返回这个对象即可。

这些函数都已经通过了main函数中的3300条tweet信息的验证，输出结果经与其他同学们输出的结果比对相同无误。

# 实验进度记录

请尽可能详细的记录你的进度情况。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 2019-03-02 | 13:30-17:30 | 编写task1，完成测试 | 延期1小时完成 |
| 2019-03-03 | 8:30-11:00 | 编写task2，完成测试 | 延期1小时完成 |
| 2019-03-09 | 13:40-17:00 | 编写task3，完成测试 | 按计划完成 |
| 2019-03-10 | 13:30-17:50 | 编写task4，完成测试 | 延期1小时完成 |

# 实验过程中遇到的困难与解决途径

实验中遇到的困难主要来源于实验四，在实验四中遇到了两个疑难的问题，一个是字符串的比较不能直接用==来比较，而是要用字符串的equal()函数，由这个问题造成的bug非常难排查，结果总是错的。

另一个问题是在匹配@符号时，用正则表达式匹配到一个字符串，我接下来直接用字符串的剪切，将@和@符号之前的内容直接剪去，因为@符号之前只有一个字符。但是有一个特殊的情况，有一个@符号之前是一个表情，表情占了两个字节，再按之前的字符串剪裁，会多出一个@符号，所以程序运行结果不正确。

# 实验过程中收获的经验、教训、感想

本节除了总结你在实验过程中收获的经验和教训，也可就以下方面谈谈你的感受（非必须）：

1. Java编程语言是否对你的口味？

还可以，感觉不如C++用得习惯。

1. 关于Eclipse IDE

亮色的界面不如IDEA的舒服。

1. 关于Git

第一次使用，很新奇。

1. 关于CMU和MIT的作业

很有趣，也很难。

1. 关于本实验的工作量、难度、deadline

工作量和难度都好像有点大。

1. 关于初接触“软件构造”课程

学习的内容很系统，易于把握。但是课堂节奏有些快，细节有些多，课后复习看课件有些吃力。