

**2019年春季学期  
计算机学院《软件构造》课程**

**Lab 1实验报告**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**目 录**

[1 实验目标概述 1](#_Toc1392798)

[2 实验环境配置 1](#_Toc1392799)

[3 实验过程 1](#_Toc1392800)

[3.1 Magic Squares 1](#_Toc1392801)

[3.1.1 isLegalMagicSquare() 1](#_Toc1392802)

[3.1.2 generateMagicSquare() 1](#_Toc1392803)

[3.2 Turtle Graphics 1](#_Toc1392804)

[3.2.1 Problem 1: Clone and import 2](#_Toc1392805)

[3.2.2 Problem 3: Turtle graphics and drawSquare 2](#_Toc1392806)

[3.2.3 Problem 5: Drawing polygons 2](#_Toc1392807)

[3.2.4 Problem 6: Calculating Bearings 2](#_Toc1392808)

[3.2.5 Problem 7: Convex Hulls 2](#_Toc1392809)

[3.2.6 Problem 8: Personal art 2](#_Toc1392810)

[3.2.7 Submitting 2](#_Toc1392811)

[3.3 Social Network 2](#_Toc1392812)

[3.3.1 设计/实现FriendshipGraph类 2](#_Toc1392813)

[3.3.2 设计/实现Person类 2](#_Toc1392814)

[3.3.3 设计/实现客户端代码main() 2](#_Toc1392815)

[3.3.4 设计/实现测试用例 3](#_Toc1392816)

[3.4 Tweet Tweet 3](#_Toc1392817)

[3.4.1 Problem 1: Extracting data from tweets 3](#_Toc1392818)

[3.4.2 Problem 2: Filtering lists of tweets 3](#_Toc1392819)

[3.4.3 Problem 3: Inferring a social network 3](#_Toc1392820)

[3.4.4 Problem 4: Get smarter 3](#_Toc1392821)

[4 实验进度记录 3](#_Toc1392822)

[5 实验过程中遇到的困难与解决途径 3](#_Toc1392823)

[6 实验过程中收获的经验、教训、感想 4](#_Toc1392824)

[6.1 实验过程中收获的经验和教训 4](#_Toc1392825)

[6.2 针对以下方面的感受 4](#_Toc1392826)

# 实验目标概述

本次实验通过求解四个问题(其中一个可选)，训练基本 Java 编程技能，能够利用JavaOO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

|  |  |
| --- | --- |
|  | 配置 |
| URL | https://github.com/ComputerScienceHIT/Lab1-1170300614 |
| 系统 | Win10 |
| JDK版本 | 1.8.0\_16 |
| JAVA\_PATH | JAVA\_HOME=/usr/libexec/java\_home -v 1.8 |

**2.2GitHub管理**

使用git与GitHub 关联

**2.3遇到的问题**

Git使用出现了问题。不会灵活使用git的指令。

解决方案：通过上网查询还有老师帮助解决问题。

# 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。

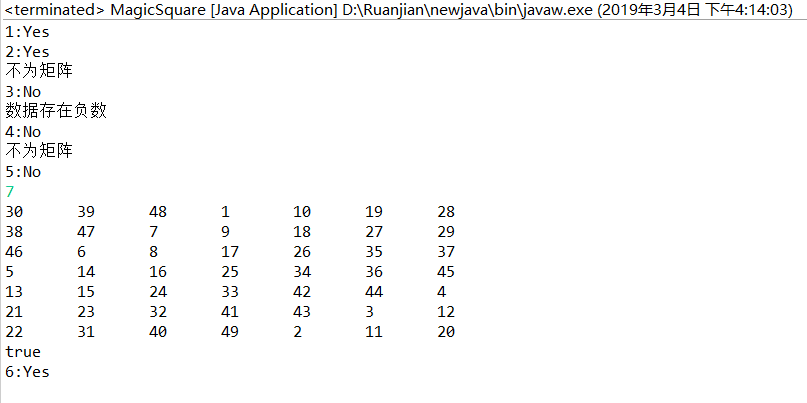
## Magic Squares

首先对MagticSquare的定义：n阶幻方方是一个正方形中n×n个数，通常是不同整数的排列，使得所有行，所有列和两个对角线中的n个数相加到相同的常数过通过两个函数解决问题isLegalMagicSquare()和generateMagicSquare()

### isLegalMagicSquare()

这个函数的功能是检验给出的矩阵是不是幻方，并且给出不是幻方时的错误。  
首先是读取矩阵，读入后存入一个字符串里，等待后边的检验。  
然后是按照一定顺序进行检查，看矩阵是否满足方阵的各个条件:  
分隔符是否符合要求 根据题目的意思，幻方的各个元素之间的分割元素必须是\t，那么我选择的方法是将整个方阵字符串中的\t``\n换成空格，然后进行分割存入字符串数组。检查这个数组，如果产生了空元素""那么说明这个数组之前存在非法的分隔符。然后检查分割后的元素，如果有非数字，则存在非法分隔符(这里包含了负数，浮点数等一系列错误)。  
检查是否为方阵   
根据题目要求，当矩阵的行数和列数不唯一时判定为非方阵。从字符串中读取元素，记录行数，然后记录最大的列数。如果不相等则为非方阵。  
检查是否缺少数字   
根据题目要求，在确认是方阵后进行检查。将数组按照分割符划分为数组，去除内部的空元素，记录元素数量。如果是完全平方数，表明是方阵，否则就是缺少数字。

实验截图



### generateMagicSquare()

首先判断输入的生成矩阵大小，若为偶数或负数则返回失败。  
生成方法：首先判断参数满足条件后，创建一个n×n大小二维数组，首先取n的一半，在矩阵第一行中间位置插入1，接下来在底部中间往右插入2，向右上方对角线一次插入3，4…直到碰到右边界。此时col归0，从左侧中间开始以此向右上方对角线继续依此插入，直到i被square整除，于是row - 1，往下一行开始又继续向右上方对角线依此插入知道row又为0，再从右下方开始以此按上述规律继续插入，不断折返直到右下角和中间到左上方被填满，再从中间下方和左上角仍按原规律继续填充直到整个矩阵被填满，至此Magic Square构造完毕。

## Turtle Graphics

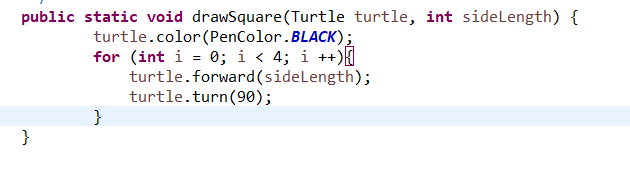
要求我们对海龟的各种操作来实现作图，包括重点是角度的调整和长度的把握。

### Problem 1: Clone and import

我采用的是从班级群中获得实验代码并且自己创建到自己的实验项目中。

### Problem 3: Turtle graphics and drawSquare

对于此要求我们直接采用一个 for 循环，运用 forward（）方法和 turn（）方法，就可以实现画正方形的需求。此方法比较直接简单，如下为代码截图：



### Problem 5: Drawing polygons

我们采用多个函数来解决此问题，采用了下面的四个函数来解决问题。

(1)calculateRegularPolygonAngle() 首先，这个方法要计算正多边形的内角，所以现根据正多边形内角和公式（n - 2） \*180 来计算正多边形的内角和，最后除以边数就可以实现这个方法。

(2)calculatePolygonSidesFromAngle() 这个方法要求我们根据给出的角度（正多边形内角）来计算正多边形的边数，通过运用公式 360 / n 即可进行计算。这里还是要注意浮点运算的时候取整问题。

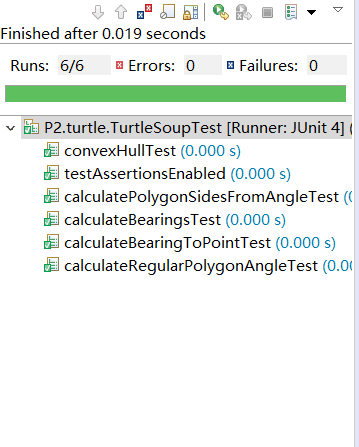
(3)drawRegularPolygon() 这个方法要画出一个正多边形，可以采用上一个函数，只需要计算出每次需要转的角度即可画出一个正多边形。

(4)calculateBearingToPoint() 这个方法需要一点思考，首先我采用转换坐标系的方法来进行思考这个问题，首先以起点为坐标原点，重新进行坐标系的构建，然后根据目标点的相对坐标来 进行情况的分类，可以根据对于 deltaY / deltaX 再进行反三角函数，来求出一个 一个角度和一些简单的几何知识，就可以解出这道题，但是在写这个方法的时候，遇到了一些问题，就是有时候我洗的方法和预期会相差180°或者360°，遇到的困难是我发现自己犯的错误，就是反三角函数 arctan 求出的角度不是一个解，而是多个，所以可以根据 if-else 来进行判断，或者采用 题目中提示的 arc2 函数，这样可以解决这个问题

### Problem 6: Calculating Bearings

(5)这个方法只需要循环调用 calculateBearingToPoint()即可, 获取 list 的长度，然 后首先将 list 中 第 一 个 点 作 为 起 始 点 ， 下 一 个 点 作 为 目 标 点 ， 传 入 calculateBearingToPoint()中，求出角度，加入返回的 list 中即可，每次更新起始 点和终止点还有初始朝向角度，即可实现。

结果截图



### Problem 7: Convex Hulls

本凸包算法中的我的大概思路为计算出最左边的点。依次选择和当前结点的方向夹角最小的点作为下一个点，并且将其加入到集合当中。当最左边的结点再次被选择的时候，算法结束。

经测试满足题意。

### Problem 8: Personal art

使用循环勾画自己的图型

### Submitting

没有进一步处理将实验完成在统一处理

## Social Network

这个实验首先要有 Person 类来存储每一个人的信息，然后在 Friendship 类中 来实例化 Person 对象，并根据题目要求来新建一个图，采用动态二维 List来存储并在此基础上来进行最短路的计算，最短路可以采取广度优先搜索来进行求取。

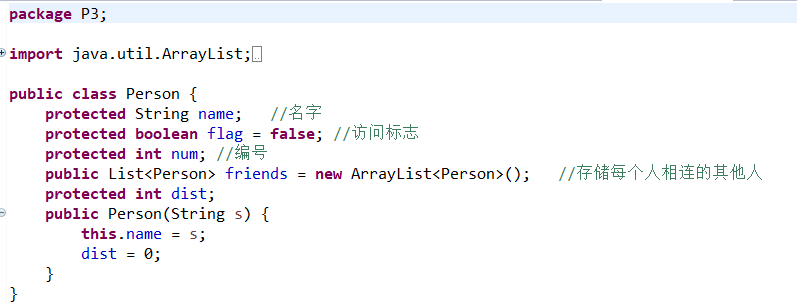
### 设计/实现FriendshipGraph类

用一个 List 来存储所有顶点，运用队列来进行广度优先搜索，即可找到每个 顶点到起始点的最短路径，直到搜到终止点，返回更新的终止点的 dist 属性，如 果与之相连的所有顶点都搜索过还没有找到目标点，则返回 -1。

### 设计/实现Person类

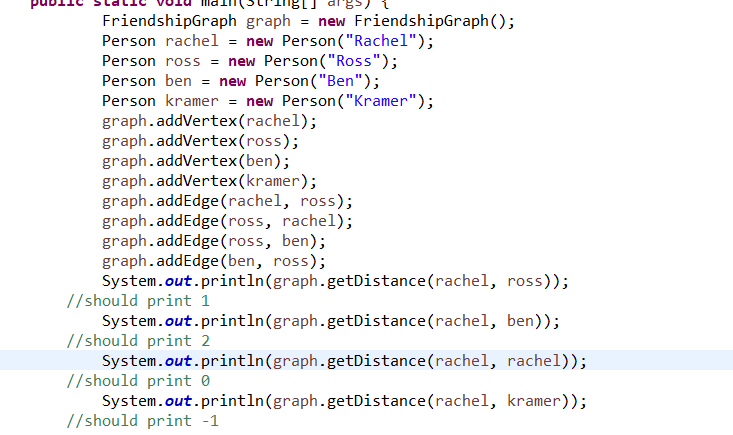
用一个 String 来存储每个人的名字，flag 标记来存储是否被访问过，在进行 广度优先搜索的时候可以用到，避免成环，friends 列表来存储与其相连的人，这 样可以更容易的进行遍历，也不用担心二维数组空间不够或者浪费空间，dist 来 存储距离起始点的最短距离，如果为 -1，则说明不相连。

代码如下：



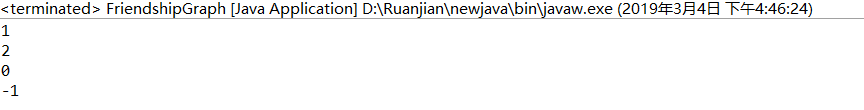
### 设计/实现客户端代码main()

Main()函数的测试内容代码如下：



。

结果输出如下：



### 设计/实现测试用例

给出你的设计和实现思路/过程/结果:

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");

/\*\*

\* Test addVertex

\*/

@Test

**public** **void** addVertexTest() { //加点测试

graph.addVertex(a);

graph.addVertex(b);

graph.addVertex(c);

graph.addVertex(d);

graph.addVertex(e);

graph.addVertex(f);

graph.addVertex(g);

*assertEquals*(a, graph.Vertex.get(0));

*assertEquals*(b, graph.Vertex.get(1));

*assertEquals*(c, graph.Vertex.get(2));

*assertEquals*(d, graph.Vertex.get(3));

*assertEquals*(e, graph.Vertex.get(4));

*assertEquals*(f, graph.Vertex.get(5));

*assertEquals*(g, graph.Vertex.get(6));

}

/\*\*

\* Test addEdge

\*/

@Test

**public** **void** addEdgeTest() { //加边测试

graph.addEdge(a, b);

graph.addEdge(b, a);

graph.addEdge(a, c);

graph.addEdge(c, a);

graph.addEdge(a, d);

graph.addEdge(d, a);

graph.addEdge(b, f);

graph.addEdge(f, b);

graph.addEdge(c, e);

graph.addEdge(e, c);

graph.addEdge(f, e);

graph.addEdge(e, f);

graph.addEdge(c, d);

graph.addEdge(d, c);

*assertEquals*(0, a.friends.indexOf(b));

*assertEquals*(1, a.friends.indexOf(c));

}

/\*\*

\* Test getDistance

\*/

@Test

**public** **void** getDistanceTest() {

graph.addVertex(a);

graph.addVertex(b);

graph.addVertex(c);

graph.addVertex(d);

graph.addVertex(e);

graph.addVertex(f);

graph.addVertex(g);

graph.addEdge(a, b);

graph.addEdge(b, a);

graph.addEdge(a, c);

graph.addEdge(c, a);

graph.addEdge(a, d);

graph.addEdge(d, a);

graph.addEdge(b, f);

graph.addEdge(f, b);

graph.addEdge(c, e);

graph.addEdge(e, c);

graph.addEdge(f, e);

graph.addEdge(e, f);

graph.addEdge(c, d);

graph.addEdge(d, c);

*assertEquals*(1, graph.getDistance(a, b));

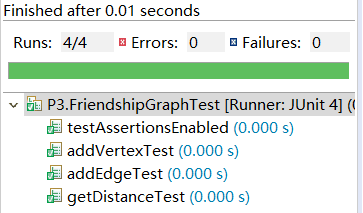
*assertEquals*(2, graph.getDistance(a, f));

*assertEquals*(0, graph.getDistance(a, a));

*assertEquals*(-1, graph.getDistance(a, g));

*assertEquals*(1, graph.getDistance(e, f));

}



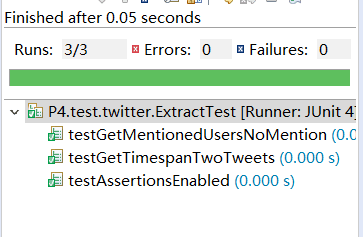
## Tweet Tweet

首先我们要处理的是主要是对文本处理的考察，比如涉及到的 getUsersMentioned()，寻找一条推文中@到的用户）可以采用正则表达式来进行实现或者直接暴力匹配就可以完成，Problem 4 提取#标签也涉及文本处理，但是在 get smater中我选择了另一个建议，Triadic closure, 这其中涉及对 Set 的修改及遍历，但是在实验过程中，发现 Set 并不能一边修改一边遍历，因为 for each 遍历是采用迭代器 的，在遍历过程中修改是会破坏迭代器的，所以这个难题我想了一下午采用了一个新的 Map 来存储需要额外添加关系的人，然后在遍历好 Map 之后，去遍历这个辅助 Map 来修改之前的 Map。

### Problem 1: Extracting data from tweets

这个问题的难点就在于时间戳之间不能直接进行比较，我们要转换成秒数来进行 比较，这个秒数是相对同一时间之间的时间差，这样就可以获取最早时间的推文 和最晚时间的推文，达到题目要求。

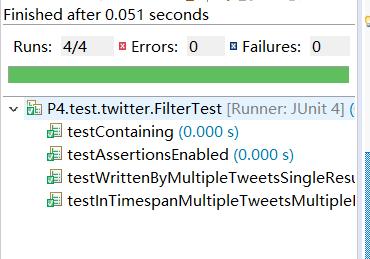
试验结果如下：



### Problem 2: Filtering lists of tweets

要按要求返回作者、返回在规定时间段之内的tweets、返回包含特殊单词的 tweets 就可以实现

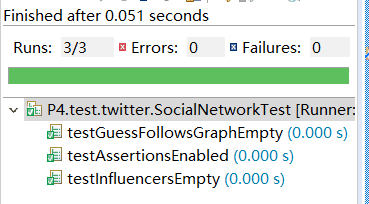
测试结果如下：



### Problem 3: Inferring a social network

最难的一部分就是要提取被提及的用户名，采取直接暴力匹配，提取 @ 后面的字符串 来作为用户名，并存入一个 list 返回，这样我就得到了被提及用户的列表，然后 根据列表去建立关系图，后面的步骤就不是很麻烦了，然后统计每个人的 followers，按照每个人的 followers 数量来进行排序就完成了。

测试结果如下：



### Problem 4: Get smarter

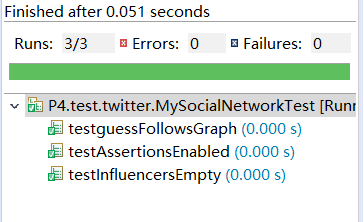
这里我选择了 Triadic closure 方法，也就是如果两个人都互相和第三个人互相@，

那么他们两个人就也应该互相关注。这里遇到最大的难题就是 Set 的遍历问题，

只能采取迭代器来进行遍历，就不能在一边遍历一边修改，所以采取一个辅助 list

来进行在 Problem 3 基础上返回的 Map 进行修改，最终实现要求.

测试结果如下:



# 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 | 15:45-17:30 | 接收到了本次实验任务开始理解和计划任务的分配 | 按计划完成 |
| 2019-02-25 | 19:00-22:00 | 开始进行Java语言的系统学习 | 遇到问题,未完成 |
| 2019-02-26 | 13:00-15:00 | 进行Java的学习 | 遇到困难，未完成 |
| 2019-02-26 | 19:00-22:00 | 进行Java容器等知识学习 | 按计划完成 |
| 2019-02-27 | 19:00-22:00 | 进行Java的类与对象的学习 | 按计划完成 |
| 2019-02-28 | 10:00-13:00 | 编写问题 1 的 isLegalMagicSquare 函数并进行测试 | 按计划完成 |
| 2019-02-28 | 19:00-22:00 | 完成P1要求的尾部问题 | 按计划完成 |
| 2019-03-01 | 13:00-17:00 | 完成P2的所有问题 | 延期 3 小时，因为一些数学问题造成了 延期、耽误了时间 |
| 2019-03-01 | 22:00-23:00 | 进行P2的测试 | 按时完成 |
| 2019-03-02 | 8:00-10:00 | 进行P3的问题研究以及算法分析 | 遇到困难未完成 |
| 2019-03-02 | 13:00-16:00 | 确定算法后进行P3问题解决 | 按计划完成 |
| 2019-03-03 | 8:00-11:00 | 想通过正则表达式完成P4,进行学习 | 失败,遇到困难未完成 |
| 2019-03-03 | 12:00-14:00 | 退而求其次选择暴力匹配的方法实现P4的问题 | 按时完成任务 |
| 2019-03-03 | 16:00-19:00 | 完成P4的前两个小问题 | 按计划完成 |
| 2019-03-04 | 20:00-22:00 | 完成P4的第三个小问题 | 按计划完成 |
| 2019-03-05 | 13:00-15:00 | 进行get smarter 问题的理解和讨论 | 按计划完成 |
| 2019-03-05 | 15:00-18:00 | 进行P4 最后问题的解决 | 按计划完成 |
| 2019-03-14 | 20:00-21:00 | 进行代码工作的第一次提交 | 按计划完成 |
| 2019-03-15 | 20:00-23:00 | 进行报告书写 | 按计划完成 |
| 2019-03-17 | 9:00-12:00 | 重新检查报告 | 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 |
| 对于大量的英文文档阅读有着不小的困难 | 通过浏览器的翻译功能解决语言不畅通问题 |
| 对于Java基本语法存在不理解和错误使用的问题 | 花取大量时间(大约一周时间)进行Java语法的学习,同时向同学请教java的编程问题.在同学和网课的支持下完成了,基本的Java学习. |
| 在解决Magic Square问题中出现了判断问题被困住了不少时间 | 和同学讨论正确的方法进行函数的书写 |
| 在解决turtle Graphic问题中对于一些数学知识出现了遗漏和错用的情况 | 在网上复习高中数学知识,请教同学解决数学方面的问题 |
| 在解决social work的问题中出现了,Floyd和深度搜索还有光度搜索选择的错误,长期困扰了我 | 对上个学期的数据结构科目进行复习,对比比较不懂算法之间的区别与联系.做出选择 |
| 在解决tweet问题中出现的问题较多首先体现在对问题的理解一开始出现了理解偏差没有按照题目要求进行代码书写 | 和同学讨论之后得到正确的解决方案 |

# 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

经验和教训总结来说就是要加强自己对英文文献的理解和掌握.对于Java语言的使用还有待提升,最后需要强调的是自己要注重对实验题目的理解,减少错误认识.

## 针对以下方面的感受

1. Java编程语言是否对你的口味？

java对于我们来说真的非常重要,虽然计算机类的同学暂时还没有系统的学习Java语言但是我们应该自主得学习各种编程语言.

1. 关于Eclipse IDE

加强使用还有理解Eclipse IDE,注意自己的操作.

1. 关于Git和GitHub

孰能生巧,GitHub还有Git是非常有用的代码工程工具.我们应该注重git还有GitHub的使用

1. 关于CMU和MIT的作业

难度较大,但是有利于加强学生解决问题的能力.

1. 关于本实验的工作量、难度、deadline

难度较大,需要学生对Java有着较深的理解

1. 关于初接触“软件构造”课程

这门课程的设立不同于其他大量的计算机课程,比较立体的介绍了软件的构造过程.