

**2021年春季学期  
计算学部《软件构造》课程**

**Lab 1实验报告**

|  |  |
| --- | --- |
| 姓名 | 朱永燊 |
| 学号 | 1190200927 |
| 班号 | 1936606 |
| 电子邮件 | 874639353@qq.com |
| 手机号码 | 13769898106 |

**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

因为上学期选修过java，所以eclipse已经配置好了，只是从help里check for upgrade更新了一下

[ComputerScienceHIT/HIT-Lab1-1190200927: HIT-Lab1-mKethree created by GitHub Classroom](https://github.com/ComputerScienceHIT/HIT-Lab1-1190200927)

# 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

幻方（Magic Square）是一种将数字安排在正方形格子中，使每行、列和对角线上的数字和都相等的方法。

此部分意在考察java基础应用以及对输入输出的基本操作的理解。

### isLegalMagicSquare()

**public** **static** **boolean** isLegalMagicSquare(String fileName) **throws** IOException {

BufferedReader bReader = **new** BufferedReader(**new** FileReader(**new** File(fileName)));

String line = "";

**int** n = 0, m = 0, k = 0;

Arrays.*fill*(*flag*, **false**);

**while** ((line = bReader.readLine()) != **null**) {

String[] l = line.split("\t");

m = l.length;

**if** (k != 0 && k != m) {

System.***out***.println("The below txt's input is wrong");

bReader.close();

**return** **false**;

}

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

*square*[n][i] = Integer.*valueOf*(l[i].trim());

**if** (*square*[n][i] <= 0 || *flag*[*square*[n][i]]) {

bReader.close();

**return** **false**;

} **else**

*flag*[*square*[n][i]] = **true**;

}

k = m;

n++;

}

bReader.close();

**if** (n != m)

**return** **false**;

**int** s1 = 0, s2 = 0, s = 0;

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

s1 += *square*[i][i];

s2 += *square*[n - i - 1][i];

}

**if** (s1 == s2)

s = s1;

**else**

**return** **false**;

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

s1 = s2 = 0;

**for** (**int** j = 0; j < n; j++) {

s1 += *square*[i][j];

s2 += *square*[j][i];

}

**if** (s1 != s || s2 != s)

**return** **false**;

}

**return** **true**;

}

**1.首先读入txt文件**

a) 创建读入操作相关对象

b) 初始化string line

**2.再逐行将字符串转换为整型矩阵存储**

a) 将非空readline的字符串以\t分割

b) 去除头尾空格后转换为数字存储到二维数组中

c) 存储时判断该数字是否出现过

i. 出现过则报错

ii. 否则用Boolean表标记

d) 每读入一行数据纪录其中数据个数即列数，若下一次读入列数不同，则数据输入格式错误，报错

最后判断行列长度是否相等，若不等则输入格式错误，报错

**3.计算两条斜线的和并比较**

a) 分别得出主对角线和次对角线的和

b) 比较

i. 若相等则记录，作为基准值

ii. 若不相等则报错

**4.计算每条纵线和横线和和并比较**

a) 分别计算第i条横线和纵线的和

b) 与基准值比较

i. 不相等则报错

确定是否为幻方

另外需注意，打开的文件要记得关闭。

### generateMagicSquare()

**public** **static** **boolean** generateMagicSquare(**int** n) **throws** IOException {

**int** magic[][] = **new** **int**[n][n];

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

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

magic[row][col] = i;

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

row++;

**else** {

**if** (row == 0)

row = n - 1;

**else**

row--;

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

col = 0;

**else**

col++;

}

}

File file = **new** File("src/P1/txt/6.txt");

PrintWriter output = **new** PrintWriter(file);

**for** (i = 0; i < n; i++) {

**for** (j = 0; j < n; j++)

output.print(magic[i][j] + "\t");

output.println();

}

output.close();

**return** **true**;

}

}

**1.首先初始化**

i. 生成空矩阵

ii. Row=0，Col=n/2

**2.再循环n\*n次填充矩阵**

i. 将矩阵的[row, col]位置填充为i

ii. 在保证坐标在矩阵范围内的情况下使Row–，Col++

**3.打开文件，打印结果**

## Turtle Graphics

相当于完善TurtleSoup.java文件，这个实验是为了训练使用java自带的画图库，另外训练我们英文的阅读能力以及读代码的能力。

**注意！！！：海龟的旋转角度为顺时针，0度为正上方**

**与直角坐标系不论是初始位置亦或是角度方向都有不同**

### Problem 1: Clone and import

打开在实验说明内的链接，然后点击右上方有“…”标记，选择clone，克隆到本地后移进到本地仓库即可。

### Problem 3: Turtle graphics and drawSquare

这一个问题要求我们完成drawSquare函数，这个问题很简单，将给出的边长利用在turtle.forward方法中，表示前进的长度，然后每次转90度，循环4次即可画出一个正方形。

**public** **static** **void** drawSquare(Turtle turtle, **int** sideLength) {

turtle.color(PenColor.***BLACK***);

**for** (**int** i = 0; i < 4; i++) {

turtle.forward(sideLength);

turtle.turn(90);

}

}

### Problem 5: Drawing polygons

该问题首先希望已知正多边形边数的情况下计算正多边形的内角度。根据几何知识可以推导得公式：

**public** **static** **double** calculateRegularPolygonAngle(**int** sides) {

**return** (**double**) 180.0 - (**double**) 360.0 / sides;

}

其次希望根据给定的多边形边数与边长画出一个正多边形，由180减内角即得旋转角度

同理：

**public** **static** **int** calculatePolygonSidesFromAngle(**double** angle) {

**double** exteriorAngle = 180.0 - angle;

**int** sides = (**int**)Math.*ceil*(360.0/exteriorAngle);

**return** sides;

}

### Problem 6: Calculating Bearings

这个方法要求用给定的参数求出从当前方向转到target点与current点连线顺时针需要旋转的角度。

不妨先使用Math.atan2函数计算两点之间的边在坐标系的角度，减去当前朝向的角度；

然后取相反数（海龟旋转的方向是顺时针，坐标轴角度的旋转角度的逆时针）；

再减去90°（海龟的0°线是向上，坐标轴的0°线是向右，向右到向上要逆时针旋转90°）；

最后调整为0-360°之间（可能大于360°或小于0°）。

**public** **static** **double** calculateBearingToPoint(**double** currentBearing, **int** currentX, **int** currentY,

**int** targetX, **int** targetY) {

**double** angle = Math.*atan2*(targetY - currentY, targetX - currentX) \* 180.0 / Math.***PI***;

**if** (angle < 0)

angle += 360.0;

**double** bearing = (360 - angle + 90 >= 360 ? 90 - angle : 360 - angle + 90) - currentBearing;

**return** bearing < 0 ? 360.0 + bearing : bearing;

}

### Problem 7: Convex Hulls

要求求解凸包问题

Jarvis步进法

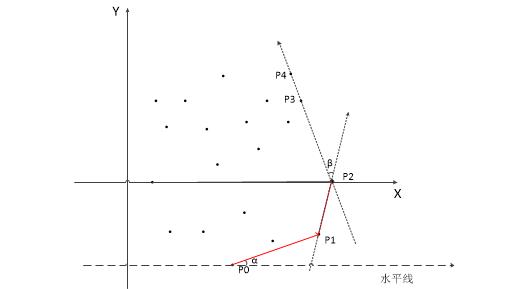
时间复杂度：O(nH)。（其中 n 是点的总个数，H 是凸包上的点的个数）

思路：

横坐标或纵坐标最小的那个点一定是凸包上的点，例如图上的 P0。

从 P0 开始，按逆时针的方向，逐个找凸包上的点，每前进一步找到一个点，所以叫作步进法。

怎么找下一个点呢？利用夹角。假设现在已经找到 {P0，P1，P2} 了，要找下一个点：剩下的点分别和 P2 组成向量，设这个向量与向量P1P2的夹角为 β 。当 β 最小时就是所要求的下一个点了，此处为 P3 。



**public** **static** Set<Point> convexHull(Set<Point> points) {

**if** (points.size() <= 3) **return** points;

Set<Point> convexHullPoints = **new** HashSet<Point>();

Point a = **new** Point(Double.***MAX\_VALUE***, Double.***MAX\_VALUE***);

**for** (Point i : points) {

**if** (i.x() < a.x() || (i.x() == a.x() && i.y() < a.y()))

a = i;

}

Point curPoint = a, minPoint = **null**, lastPoint = a;

**double** x1 = 0.0, y1 = -1.0;

**do** {

convexHullPoints.add(curPoint);

**double** minTheta = Double.***MAX\_VALUE***, x2 = 0.0, y2 = 0.0;

**for** (Point i : points) {

**if** ((!convexHullPoints.contains(i) || i == a) && (i != lastPoint)) {

**double** x3 = i.x() - curPoint.x(), y3 = i.y() - curPoint.y();

**double** Theta = Math.*acos*((x1 \* x3 + y1 \* y3) / Math.*sqrt*(x1 \* x1 + y1 \* y1) / Math.*sqrt*(x3 \* x3 + y3 \* y3));

**if** (Theta < minTheta || (Theta == minTheta && x3 \* x3 + y3 \* y3 > Math.*pow*(i.x() - minPoint.x(), 2)

+ Math.*pow*(i.y() - minPoint.y(), 2))) {

minPoint = i;

minTheta = Theta;

x2 = x3;

y2 = y3;

}

}

}

x1 = x2;

y1 = y2;

lastPoint = curPoint;

curPoint = minPoint;

} **while** (curPoint != a);

**return** convexHullPoints;

}

不妨取最左边的点（横坐标相同，取纵坐标最小者）

则其必为凸包点

**注意：因为余弦0至180不同，正弦有重合，所以用余弦**

寻找思路为：以前两个凸包点的连线为初始方向，因为知道各点坐标，以余弦公式算出余弦再算出角度找出与连线夹角最小者为下一个凸包点

第一条线还需自找一点，不妨找（0 ，-1），此点也奠定了逆时针寻找

（-1,0）为顺时针

用一下分别纪录上一个凸包点，当前的凸包点（不一定是，在寻找中），最开始凸包点

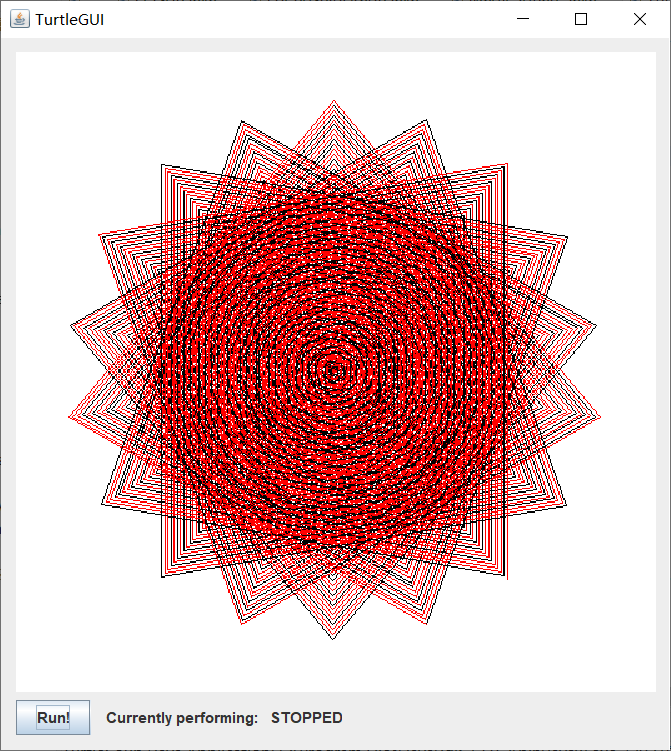
curPoint

minPoint

lastPoint

当凸包点又与开始的凸包点重和时停止

### Problem 8: Personal art



一朵花

### Submitting

[Eclipse 配置 GitHub - 天天向上327 - 博客园 (cnblogs.com)](https://www.cnblogs.com/yiwenrong/p/12665605.html)

在eclipse中直接就可以提交，首先右键，点击team，选择share，然后根据步骤与github仓库链接，然后将全部内容加到缓冲区，然后将缓冲区中的内容commit and push即可。

## Social Network

建立一些点，这些点看做是人，点之间连边，有边代表这两点有联系，这样就构成了一个关系图，然后完成获取两点之间最近距离。

### 设计/实现FriendshipGraph类

用矩阵来实现FriendshipGraph类的表示

第一种方法可以初始化一个比较大的矩阵来存储，但这有缺点，比较模板，且后续程序不好优化，当数据量超过程序时无法运行

此方法用n记录点数，peson类的number来对应行列，

第二种方法，不妨用arraylist套arraylist来构建一个动态矩阵

每加入一个点就在原基础上加入对应的行与列，并进行赋值

每一个点即person类都有唯一的number来对应矩阵相应行与列

最后求解距离用floyd算法即可

其中INF用-1表示，则算法中需修改与-1相关的代码

三层循环k I j

若kj 为-1且ki ij不为-1 则赋值

或若kj不为-1 且ki ij不为-1 则比较后赋值

其中关于重复人名，在外部弄一个hashset 收录已有人名

每一次录入点时判断是否有这个人名，若有则录入，若无则不

### 设计/实现Person类

Person类较为简单，直接贴代码

主要含string即人名 number即对应矩阵/动态矩阵行列数

**public** **class** Person {

**private** String name;

**public** **int** number;

**public** Person(String name) {

**this**.setName(name);

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

### 设计/实现客户端代码main()

相当于写测试用例，创FriendshipGraph后输入输出就可以

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

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));

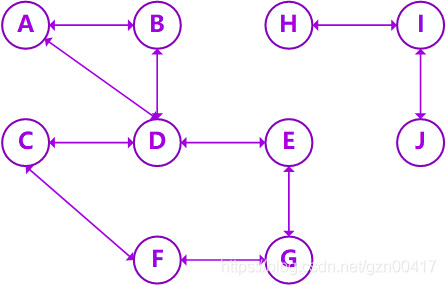
System.***out***.println(graph.getDistance(rachel, kramer));

}

### 设计/实现测试用例

简单图测试用例参照main函数即可

再可以加入复杂图测试用例



**public** **void** GrpahTest2() {

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");

Person h = **new** Person("H");

Person i = **new** Person("I");

Person j = **new** Person("J");

graph.addVertex(a);

graph.addVertex(b);

graph.addVertex(c);

graph.addVertex(d);

graph.addVertex(e);

graph.addVertex(f);

graph.addVertex(g);

graph.addVertex(h);

graph.addVertex(i);

graph.addVertex(j);

graph.addEdge(a, b);

graph.addEdge(a, d);

graph.addEdge(b, d);

graph.addEdge(c, d);

graph.addEdge(d, e);

graph.addEdge(c, f);

graph.addEdge(e, g);

graph.addEdge(f, g);

graph.addEdge(h, i);

graph.addEdge(i, j);

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

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

*assertEquals*(3, graph.getDistance(a, g));

*assertEquals*(3, graph.getDistance(b, f));

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

*assertEquals*(2, graph.getDistance(h, j));

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

*assertEquals*(-1, graph.getDistance(d, j));

*assertEquals*(-1, graph.getDistance(c, i));

*assertEquals*(-1, graph.getDistance(f, h));

}

# 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 |
| 2021-05-17 | 17:30-20:30 | 编写P1并进行测试 | 按计划完成 |
| 2021-05-18 | 15:50-17:30 | 编写P2 | 遇到困难，未完成 |
| 2021-05-19 | 18:30-20:30 | 完善P2 | 完成 |
| 2021-05-20 | 15:50-17:30 | 编写P3 | 完成 |
| 2021-05-21 | 8:00-9:45 | 写报告 | 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 |
| 不知道在eclipse里怎么把txt带到编程文件夹里 | 问同学，直接在外部复制粘贴再fresh就行 |
| Junit包的添加 | 上网查，找到教程 |
| Eclipse 与 Git的使用和互通 |  |
| 实验要求基本全英文，比较晦涩难懂 | 上网翻译 |

# 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

实验一定要提前先预习先做，不会的东西就到网上去查

写实验之前要多熟悉语言，对Java语言不熟悉的我在这次实验中遇到了很多困难，有很多Java自带的函数和数据结构都不了解，导致走了很多弯路。这个时候千万不能放弃，应该努力通过各种途径去学习相关知识，来弥补自己的不足。

## 针对以下方面的感受

1. Java编程语言是否对你的口味？

c用习惯了，用java还是很不习惯，不过java各种方法的封装用起来很方便，能够提高编码速度。

（但是有些东西找实现方法最后发现没有，浪费了好多时间）

1. 关于Eclipse IDE；

用起来还行

1. 关于Git和GitHub；

有点麻烦，而且老是出毛病，clone push各种报错

1. 关于CMU和MIT的作业；

难度适中（如果会java的话），个人觉得好难（不会java）

1. 关于本实验的工作量、难度、deadline；

适中

1. 关于初接触“软件构造”课程；

好难