
哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

# 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 | 魏孝文 |
| 学号 | 1170500704 |
| 班号 | 1737101 |
| 电子邮件 | 706461857@qq.com |
| 手机号码 | 18145640839 |

# 目录

# 1 实验目标概述

本次实验通过求解四个问题，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

根据实验手册简要撰写。

# 2 实验环境配置

简要陈述你配置本次实验所需开发、测试、运行环境的过程，必要时可以给出屏幕截图。

特别是要记录配置过程中遇到的问题和困难，以及如何解决的。

https：//github.com/ComputerScienceHIT/Lab1-1170500704

在这里给出你的 GitHub Lab1 仓库的 URL 地址（Lab1-学号）。

# 3 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。

## 3.1 Magic Squares

MagicSquare 是指每一行的和，每一列的和，对角线的和都等于一个常数的矩阵，编写一个程序，判断一个矩阵是否是 Magicsquare，再改写一个生成 Magic Square 的程序，增加它的健壮性。

### 3.1.1 isLegalMagicSquare()

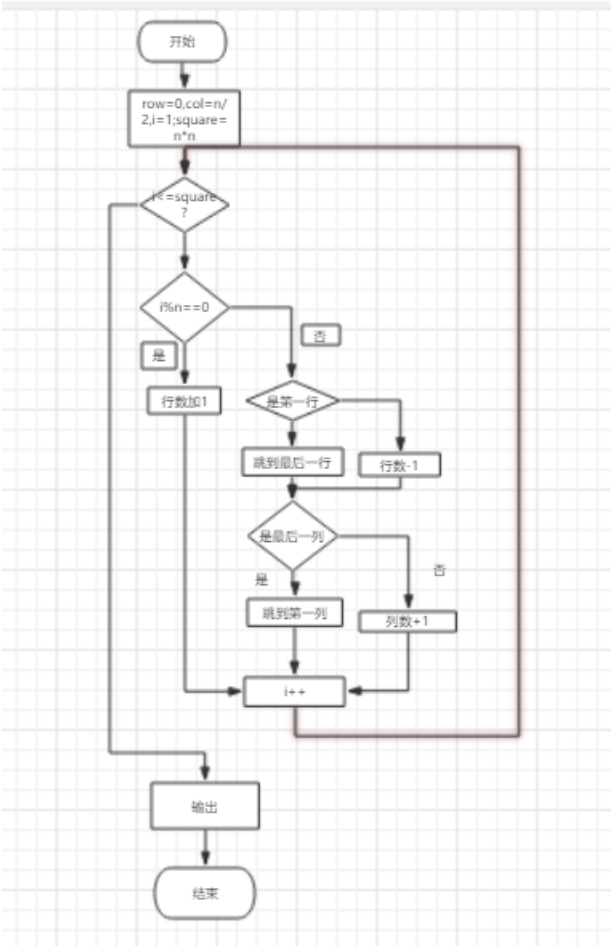
读取文件：我使用的是相对寻址的方式，在 src 文件夹下找到输入的 txt 文件，读取 txt 文件，根据读取的数字建立矩阵，如果发生异常则捕捉异常种类，输出并终止程序。
对于建立的矩阵，每行每列与对角线地扫描矩阵，如果发现和不相等，返回 false，如果全都相等，返回 true。
对五个测试用例检测：

```
1.txt: true

2.txt: true

不符合magic square的定义
3.txt: false

矩阵元素不是正整数
4.txt: false

不符合magic square的定义
5.txt: false
```

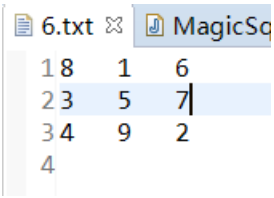
### 3.1.2 generateMagicSquare()

这个方法生成 Magic Square 的方法是：把"1"放在中间一列最上边的方格中，从它开始，按对角线方向（比如说按从左下到右上的方向）顺次把由小到大的各数放入各方格中，如果碰到顶，则折向底，如果到达右侧，则转向左侧，如果进行中轮到的方格中已有数或到达右上角，则退至前一格的下方。

输出到文件是新学习的，我采用的方法是通过 System.outSet 改变输出流到 6.txt。
程序流程图如下：




结果:



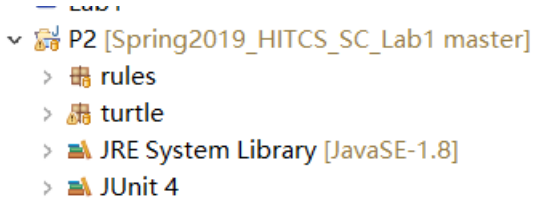

## 3.2 Turtle Graphics

Problem1:从 GitHub 上 clone 项目到本地，添加到 Eclipse 的文件中，对其进行修改。熟悉 turtle 作图的基本操作，计算一些角度和边，编写凸包算法。

### 3.2.1 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。

Fork 老师的仓库，clone 到本地，在 Eclipse，选择 *File* → *Import…* → *General* → *Existing Projects into Workspace*.完成将 P2 转化为 Eclipse 中的 project。




### 3.2.2 Problem 3: Turtle graphics and drawSquare

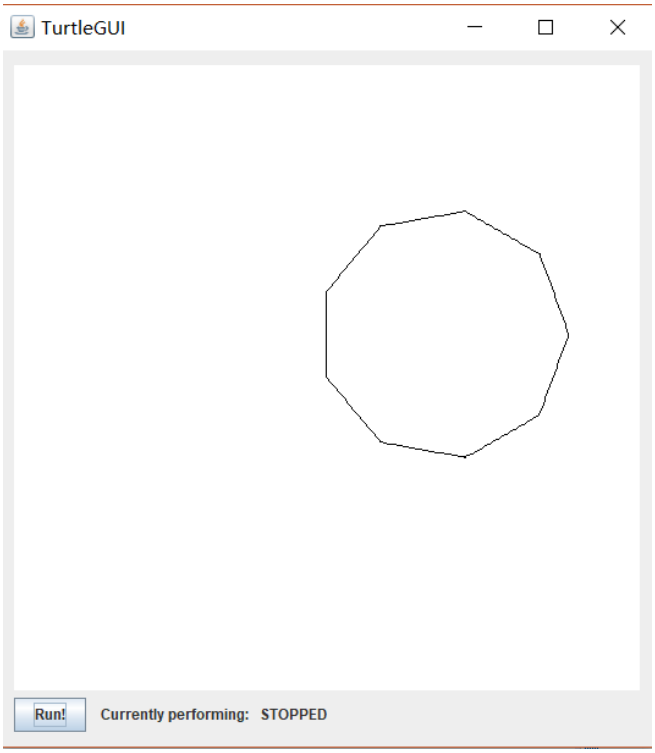
```
public static void drawSquare(Turtle turtle, int sideLength) {
        turtle.draw();
        turtle.forward(sideLength);
        turtle.turn(90.00);
        turtle.forward(sideLength);
        turtle.turn(90.00);
        turtle.forward(sideLength);
        turtle.turn(90.00);
        turtle.forward(sideLength);
        turtle.turn(90.00);
    }
```

### 3.2.3 Problem 5: Drawing polygons

1.public static double calculateRegularPolygonAngle(int sides)
此函数是给定边数来计算多边形的内角值，我们知道多边形的角度总和是 180*（sides-2），而正多边形的每个内角值相等，所以返回 `180*(sides-2)/sides`。

2.public static int calculatePolygonSidesFromAngle(double angle)
这个函数是跟定内角度数求边数，根据我们第一问得到的公式 angle=180*(sides-2)/sides，又因为外角和是 360 度，我们可以推算出 sides=360/(180-angle)，而等式右边是一个浮点数，我们用 Math.round()进行四舍五入，得到的是一个长整形数，强转成 int 类型，所以返回`(int)Math.round(360/(180-angle))`。

3.public static void drawRegularPolygon(Turtle turtle, int sides, int sideLength)
这个函数是做图的函数，我根据 sides 调用 calculateRegularPolygonAngle(int sides) 函数，算出内角值，再用 180-内角值得到每次转弯的角度值。用一个 for 循环，每走 sideLength 后转弯一次，绘制出图像。例如我画的 sideLength=70 的九边形。




### 3.2.4 Problem 6: Calculating Bearings

```
1.public static double calculateBearingToPoint(double currentBearing, int currentX, int currentY,int targetX, int targetY)
```

由几何关系知，要计算转角的度数就要先知道两个点所成的角度，所以用反正切函数算出：

```
double x=targetX-currentX,y=targetY-currentY,angle,temp=y/x;
angle = 90-currentBearing-Math.toDegrees(Math.atan(temp));
```

因为转弯是顺时针，所以如果 angle<180,直接返回 angle，否则返回 angle+360.

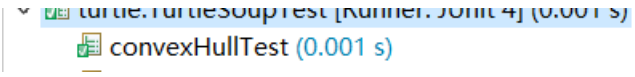
```
2. public static List<Double> calculateBearings(List<Integer> xCoords, List<Integer> yCoords)
```

记 len 是 xCoords 的长度，对于每个从 0 到 len-1 的 i，用反正切函数和 yCoords.get(i)/xCoords.get(i)算出当前的角度值，再调用 calculateBearingToPoint 计算出转角值，将结果 add 进入结果的 list 即可。

### 3.2.5 Problem 7: Convex Hulls

这个问题是在给定的点集里面，找出形成 Convex Hull 之后能包含所有的点的最少的点集。这一题判断共线用了叉积的知识：设(x1,y1)是一个向量，那么它和另一个向量(x2,y2)共线的充要条件是 x1*y2-x2*y1=0。

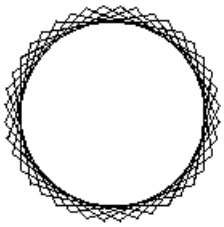
用暴力法解决问题：点集中最左下角的点一定是凸包中的点，从这一点出发，逆时针查找下一个点，如果有一个点满足起始点到这个点的向量与起始点到其他任何点的向量的叉积都是负数，那么将这个点加入凸包。最后我对于凸包中所有共线的元素（及叉积为 0）进行查找，将共线而且不是端点的元素删除。




### 3.2.6 Problem 8: Personal art

```
public static void drawPersonalArt(Turtle turtle) {
    int i;
    turtle.draw();
    for(i=0;i<50;i++) {
    turtle.forward(70);
    turtle.turn(63);
    }
//throw new RuntimeException("implement me!");
}
```

效果图:



### 3.2.7 Submitting

git add -A

git commit

git push

## 3.3 Social Network

创建一个社交网络的模型，一个社交网络包括一些人，这些人通过朋友关系联系在一起。

1. FriendshipGraph graph = new FriendshipGraph();通过调用 FriendshipGraph()方法来创建一个新的人际网络。
2. Person rachel = new Person("Rachel")创建一个新的人节点，她的名字叫 Rachel。
3. graph.addVertex(rachel);将 Rachel 这个节点加入图中。
4. graph.addEdge(rachel, ross);在图中加入一条边，从 Rachel 指向 Ross，代表 Rachel 和 Ross 新建了朋友关系。
5. graph.getDistance(rachel, ross)得到 Rachel 和 Ross 在社交网中的距离。

在这里简要概述你对该任务的理解。

### 3.3.1 设计/实现 FriendshipGraph 类

我用List类实现FriendshipGraph，其中List中的每个节点都是一个person类的对象，
ArrayList people=new ArrayList<Person>();这样就可以把所有的person加入了社交网中。

1. void addVertex(Person p)
   如果 people 里面已经包含了节点 p，那么就输出异常，否则将节点 p add 进 people 中。

2. void addEdge(Person p1,Person p2)
   如果 p1 的朋友中已经有 p2，那么打印异常，否则在 p1 的 friend 中加入 p2。

3. int getDistance(Person p1,Person p2)
   这个函数是计算节点 p1 和节点 p2 在社交网络中的距离，我的想法来源于广度优先遍历，设一个初值为 1 的计数器 count，一个名为 l1 的 list，初值为 p1 的所有朋友。
   While（l1 不为空与 l1 中没有 p2）{
   新建 list l2 存储 l1 中所有节点的朋友；
   Count++；
   L1=l2；
   }
   如果 l1 最终为空，那么代表网络中不存在 p1 到 p2 的边，返回-1，否则返回两个点之间的距离 count。

结果：

```
<terminated> Frienc
1
2
0
-1
```

给出你的设计和实现思路/过程/结果。

### 3.3.2 设计/实现 Person 类

```
import java.util.ArrayList;

public class Person {
    String name;//name用来存储person的名字
boolean visited=false;//标记位，在计算两个节点之间距离时用来标记一个节点是否被访问过
    Person (String s){//初始化person
        name=s;
    }
```

```
    ArrayList friend=new ArrayList<Person>();//新建一个名为friend的list,元素类型是Person，用来存储这个人的朋友

    public void addfriend(Person p) {//增加朋友函数，如果这个人认识了一个新朋友，那么将新朋友加入friend中
    this.friend.add(p);
    }
}
```

结果：在 FriendshipGraph 中测试正确。

给出你的设计和实现思路/过程/结果。

### 3.3.3 设计/实现客户端代码 main()

给出你的设计和实现思路/过程/结果。

### 3.3.4 设计/实现测试用例

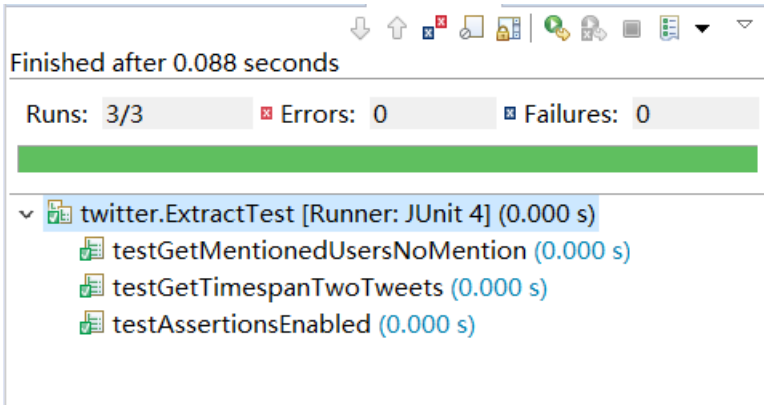
给出你的设计和实现思路/过程/结果。

## 3.4 Tweet Tweet

这个问题是对给出的一系列 Tweet 博文进行查找时间段，作者，计算每个人关注的人，查找最受欢迎的人的一系列操作。

### 3.4.1 Problem 1: Extracting data from tweets

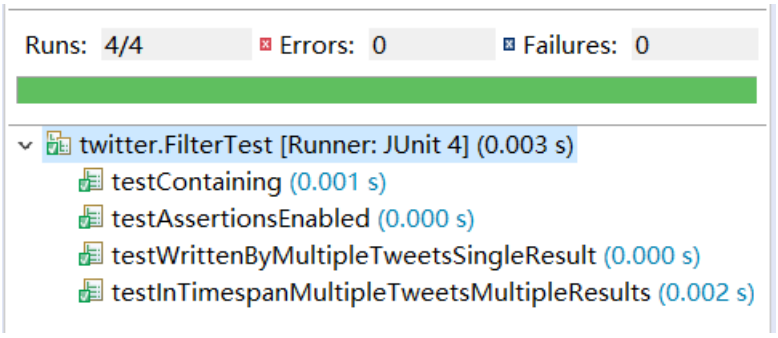
1. 查找 tweets 的最早的时间和最晚的时间
   先设最早的时间 start 和最晚的时间 end 都是第一个 tweet 的时间，遍历所有 tweet，如果有 tweet 的时间比 start 还早，那么就将 start 设为这个 tweet 的时间，同理如果有 tweet 的时间比 end 还晚，那么就将 end 等于这个时间。
2. 找到被@的用户:
   先把所有用户的名字记录在 list 中，对于每一条 tweet 的内容进行遍历，如果出现了@用户名的内容，对其进行检测，如果用户名是 tweet 的最后一个字符串，加入答案，如果的下一个字符是空格，加入答案。




### 3.4.2 Problem 2: Filtering lists of tweets

1. 在 tweet 中查找特定作者写的 tweet
   遍历 tweet，如果某个 tweet 的作者是要查找的这个人，那么将这个 tweet 加入结果的 list 中。
2. 在 tweet 中查找在特定时间段写的 tweet
   遍历 tweet，如果某个 tweet 发表的时间是特定的时间段，那么将这个 tweet 加入结果的 list 中。
3. 在 tweet 中查找具有特定单词的 tweet
   遍历 tweet，对于每一个 tweet，遍历我们要查找的词，如果找到了，那么将这个 tweet 加入结果的 list 中。

结果：



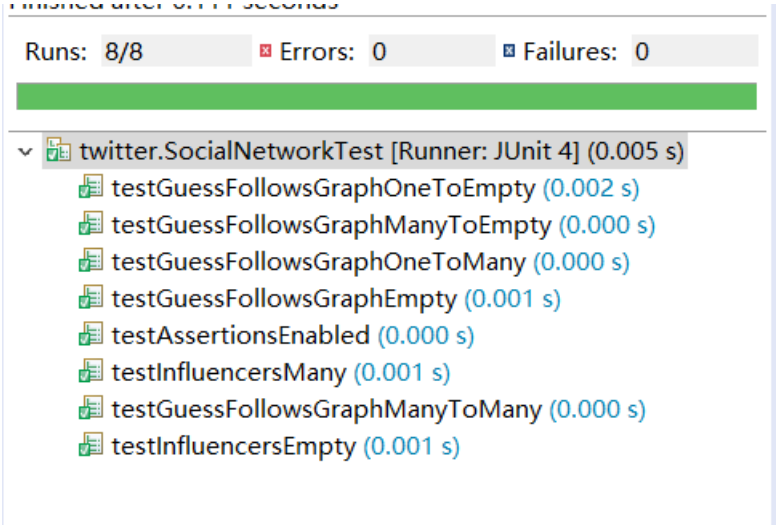

### 3.4.3 Problem 3: Inferring a social network

1. `public static Map<String, Set<String>> guessFollowsGraph(List<Tweet> tweets)`
   这个问题是给定一系列 tweet，返回表示关注关系的一个图。
   我的想法是先遍历一遍 tweets，找到所有的不重复的作者名，加入到名为 name 的 list 中，再对 name 中的每一个元素 s，遍历 tweets，如果某个 tweet 的作者是 s，查找这个 tweet 中有没有@某人，如果有，将这个人名字加入到 set 中，当遍历结束时，如果 set 不为空，将（s，set）加入到题目要返回的 map 中。
2. `public static List<String> influencers(Map<String, Set<String>> followsGraph)`

我的想法是先声明一个 class user：

```
class user{
    String username;
    int atnumber;
    public user(String s) {
        username=s;
        atnumber=1;
    }

    boolean equal(String s) {
        if(username.equals(s))
            return true;
        else
            return false;
    }
}
```

建立一个名为 name，元素类型为 user 的 list 用来存储所有的被关注者遍历 followGraph 中的每一个 Set<String>，将 Set<String>记为 follow，再遍历 follow，如果 follow 的某个元素第一次出现，那么新建一个名为这个元素的 user 节点，将这个节点加入 name 中，否则将名为这个元素的 user 的节点的 atname+1.当遍历结束时，找出 name 中 atnumber 最大的元素，将它们的 uesrname 加入到结果的 list 中。




### 3.4.4 Problem 4: Get smarter

对使用共同标签（#XXX）的人进行查找，我们约定标签和 text 内容是分开的，及#前面不能是除了空格键的其它字符。
对于给出的一系列 tweet，对于某一个特定的人，找出他所有 tweet 中使用的所有标签，对于其他人写的 tweet 进行遍历，如果发现了使用相同标签的人，将这两个人建立联系。

# 4 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 | 按计划完成 |
| 2019-3-4 | 14：00-22：00 | 编写问题二并进行测试 | 没有学习过叉积，延期很长时间完成 |
| 2019-3-5 | 18：00-23：00 | 编写 P3,P4 | 按计划完成 |
| 2019-3-7 | 15：30-21：30 | 学习 travis-ci 进行测试 | 遇到问题。未完成 |
| 2019-3-8 | 早上 8：00-12：00 | 学习在线测试 | 请教同学完成 |

# 5 实验过程中遇到的困难与解决途径

| 遇到的难点 | 解决途径 |
|---|---|
| 凸包问题中的判断共线 | 开始我使用斜率进行判断，但是斜率实在是难判断，还要考虑斜率无穷大的情况，后来我在网上查资料学习使用叉积，解决了问题 |
| 文件的读取和写入 | 这方面之前没怎么接触过，后来用 `BufferedReader` 和 `System.setout` 解决文件的读取和写入。 |
| 代码脱离 LDE 进行测试 | 之前没有将代码脱离 IDE 进行测试过，不知道该怎么生成 pom.xml 或者 build.xml，后来在网上学习其他同学的经验才 build 成功 |

# 6 实验过程中收获的经验、教训、感想

## 6.1 实验过程中收获的经验和教训

经验：Java 集合类的使用，熟悉了 Java 的基本语法

教训: 写程序之前要事先想清楚再写, 凸包问题没有想到好的解决策略就写, 卡了很长时间。

## 6.2 针对以下方面的感受

(1) Java 编程语言是否对你的口味？

对

(2) 关于 Eclipse IDE

较熟练地掌握了 Eclipse 的使用方法。

(3) 关于 Git 和 GitHub

第一次使用 Github，学会了建立仓库，上传文件查看文件变化等基本操作。

(4) 关于 CMU 和 MIT 的作业

第一次的实验内容不是很难

(5) 关于本实验的工作量、难度、deadline

比较合理

(6) 关于初接触“软件构造”课程

上课的内容比较抽象，但是实验比较具体。