

**2018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 | 任庆吉 |
| 学号 | 1160300422 |
| 班号 | 1603004 |
| 电子邮件 | RenQJHIT@163.com |
| 手机号码 | 18846088521 |

**目录**

[1 实验目标概述 1](#_Toc506282558)

[2 实验环境配置 1](#_Toc506282559)

[3 实验过程 1](#_Toc506282560)

[3.1 Magic Squares 1](#_Toc506282561)

[3.1.1 isLegalMagicSquare() 1](#_Toc506282562)

[3.1.2 generateMagicSquare() 1](#_Toc506282563)

[3.2 Turtle Graphics 1](#_Toc506282564)

[3.2.1 Problem 1: Clone and import 2](#_Toc506282565)

[3.2.2 Problem 3: Turtle graphics and drawSquare 2](#_Toc506282566)

[3.2.3 Problem 5: Drawing polygons 2](#_Toc506282567)

[3.2.4 Problem 6: Calculating headings 2](#_Toc506282568)

[3.2.5 Problem 7: Personal art 2](#_Toc506282569)

[3.2.6 Submitting 2](#_Toc506282570)

[3.3 Social Network 2](#_Toc506282571)

[3.3.1 设计/实现FriendshipGraph类 2](#_Toc506282572)

[3.3.2 设计/实现Person类 2](#_Toc506282573)

[3.3.3 设计/实现客户端代码main() 2](#_Toc506282574)

[3.3.4 设计/实现测试用例 2](#_Toc506282575)

[3.4 Tweet Tweet（选作，额外记分） 3](#_Toc506282576)

[4 实验进度记录 3](#_Toc506282577)

[5 实验过程中遇到的困难与解决途径 3](#_Toc506282578)

[6 实验过程中收获的经验、教训、感想 3](#_Toc506282579)

# 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

Eclipse和jdk是早就有的，没花时间去配置，git是在软件源

apt install git

开始对git很陌生，不知道是干什么的，摸索了两天懂了大概。其他的就没什么了，基本就是写代码了。

我的仓库：

git@github.com:ComputerScienceHIT/Lab1-1160300422.git

# 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（但千万不要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

这个实验就是写一个小程序，我猜想老师设计这个实验的目的是让我们熟悉一下Java编程，整体来看没什么难度，但是涉及到的一些方法实现，如读写文件，数组的遍历等等，还是花了一些时间去学习。

程序要求是从txt中读取一个矩阵，判断是不是magicsquare，所谓magicsquare就是每行、每列、每个对角线的和都相等的矩阵。因为只要有一点不符合要求就判定为false，所以处处设卡，一旦出现问题则返回false，全部都通过即为true。这就是整体的思路。

### isLegalMagicSquare()

先读矩阵第一行，确定基数N，并确定每个String是不是整数，然后生成N×N的数组，然后在循环中依次读每一行，仍然要判断每个String是不是整数，读完一行后还要判断当前行是不是N个元素。读完矩阵后还要判断列数是不是N。然后要计算该矩阵的每一行每一列以及对角线的和，判断是否符合要求。

### generateMagicSquare()

这个函数是把1到N×N依次填入N×N矩阵，从中间开始，填的顺序是斜向上填，到边界就顺延到另一侧。可以保证每行每列对角线的和都相同。

### 运行截图



## Turtle Graphics

Logo是麻省理工学院发明的一种编程语言，最初用于在太空中移动机器人。海龟图形，添加到Logo语言中，允许程序员向屏幕上的“海龟”发出一系列指令，并在其移动时画出一条线。海龟图形也被添加到许多不同的编程语言中，包括Python，它是标准库的一部分。

  这次实验就是学着用turtle画一些图案，以及对图案进行一些角度计算。很简单，主要使用两个函数，forward(units)和turn(degrees) ，forward(units)是指前进多少长度，turn(degrees)是指顺时针转多少度。

### Problem 1: Clone and import

git clone <https://github.com/rainywang/Spring2018_HITCS_SC_Lab1.git>

下载代码

git init

创建本地仓库

git push [git@github.com:ComputerScienceHIT/Lab1-1160300422.git](mailto:git@github.com:ComputerScienceHIT/Lab1-1160300422.git)

把本地仓库上传到GitHub

### Problem 3: Turtle graphics and drawSquare

画一个正方形很简单，

while (i < 4) {

            i++;

            turtle.forward(100);

            turtle.turn(90);

      }

每次前进相同的长度，每次都转90°。

### Problem 5: Drawing polygons

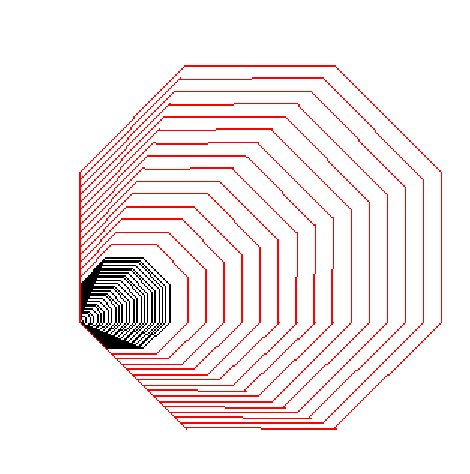
画一个多边形。之前实现了计算正多边形内角的函数，利用这个函数得到每次改变的方向，循环n次得到正n边形。

### Problem 6: Calculating headings

这个函数要计算一系列点转动的角度和，只是重复利用calculateHeadingToPoint函数求和。

### Problem 7: Personal art

这个随便写的，一般来说只要边长和角度按照某一规律变化都能得到一个看起来很规整的图案。



这是一支铅笔

### Submitting

git push [git@github.com:ComputerScienceHIT/Lab1-1160300422.git](mailto:git@github.com:ComputerScienceHIT/Lab1-1160300422.git)

## Social Network

主要目标是构造一个图的结构描述人与人之间的关系。需要用到上学期学到的数据结构的知识，但是上学期学数据结构时用的语言是C语言，用Java来实现还是有些陌生。

### 设计/实现FriendshipGraph类

FriendshipGraph描述出这个关系图，经过分析，人际关系图应该是一个稀疏图，所以选择邻接表来实现。为每个成员创建一个List来存放他的朋友，用一个Map来存取这些映射关系。

Public Map<Person,List<Person>> map=new HashMap<Person,List<Person>>();

图有两个要素，点和边，所以要实现addVertex和addEdge两个函数。

**public** **void** addVertex(Person Vertex) **throws** Exception {

**if** (**this**.map.containsKey(Vertex)) {//为避免重复添加这个判断

**throw** **new** Exception("Wrong!Each person has a unique name");

} **else** {

List<Person> friend = **new** ArrayList<Person>();//为他创建一个List存朋友们

map.put(Vertex, friend);

}

}

**public** **void** addEdge(Person m1, Person m2) {

**int** n = map.get(m1).size(), i = 0;

**boolean** choice = **true**;

**while** (i < n) {

**if** (map.get(m1).get(i) == m2) {//防止重复

choice = **false**;

**break**;

}

i++;

}

**if** (choice) {

map.get(m1).add(m2);//如果m1认识m2，则在m1的List添加m2

}

}

求两个人A、B之间有多少层关系只需要用深度优先遍历寻找B在A的第几层。这个算法上学期学过，用java实现一下就好。

### 设计/实现Person类

Person类用于描述每个成员的性质，可以有姓名、学校、学号等等信息，但是本实验只需要姓名这一信息。代码很简单。粘上来

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

**private** String name;

**public** Person(String Name) {

**this**.name = Name;

}

**public** String getName() {

**return** **this**.name;

}

}

写了一个用名字来实例化的构造函数，因为希望name是一个private所以又加了一个getName方法（但没用上）。

### 设计/实现客户端代码main()

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

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));

}

使用了实验手册中的关系图，如果想自己新建一个也是同样的过程。

### 设计/实现测试用例

addVertexTest : 这个函数没有返回值，根据它造成的影响测试。

*assertTrue*(graph.map.size() == 4);

*assertTrue*(graph.map.containsKey(rachel));

*assertTrue*(graph.map.containsKey(ross));

*assertTrue*(graph.map.containsKey(ben));

*assertTrue*(graph.map.containsKey(kramer));

addEdgeTest ： 同样void函数，要根据影响测试，

*assertEquals*(Rachel, graph.map.get(rachel));

*assertEquals*(Ross, graph.map.get(ross));

*assertEquals*(Ben, graph.map.get(ben));

*assertEquals*(Kramer, graph.map.get(kramer));

getDistanceTest ： 这个函数有返回值，判断返回值是否正确就行了

*assertEquals*(1, graph.getDistance(rachel, ross));

*assertEquals*(2, graph.getDistance(rachel, ben));

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

*assertEquals*(-1, graph.getDistance(rachel, kramer));

## Tweet Tweet（选作，额外记分）

这个实验写起来很麻烦，首先理解题意就有些困难，因为没用过twitter，对于有些概念，如timestamp，getmention不知道什么意思，花了很长时间来理解。

### 实现Extract类

这个类是实现从twitter列表中获取信息的方法。

首先是getTimespan得到这个tweet列表的开始时间和结束时间

**public** **static** Timespan getTimespan(List<Tweet> tweets) {

**int** n = tweets.size();

Instant min = tweets.get(0).getTimestamp();

Instant max = tweets.get(0).getTimestamp();

**for** (**int** i = 1; i < n; i++) {

Instant now = tweets.get(i).getTimestamp();

**if** (now.isBefore(min))

min = now;

**if** (now.isAfter(max))

max = now;

}

Timespan timespan = **new** Timespan(min, max);

**return** timespan;

}

这个很简单遍历一遍所有timestamp，找到最前和最后。

然后要写getMentionedUsers方法，该函数是从所有推文中找到所有被@的用户名，根据要求构造出一个正则表达式来对字符串进行匹配，usernameInText是Set不用考虑重复的问题，但是要考虑大小写，所以在添加之前我都转化成小写了。

**public** **static** Set<String> getMentionedUsers(List<Tweet> tweets) {

Set<String> usernameInText = **new** HashSet<String>();

**int** n = tweets.size();

Pattern pattern = Pattern.*compile*("(^|[^A-Za-z0-9\_-]+)@([A-Za-z0-9\_-]+)");//满足条件的字符字串的正则表达式

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

String text = tweets.get(i).getText();

Matcher username\_mention = pattern.matcher(text);

**while** (username\_mention.find()) {

String now = **new** String(username\_mention.group(2).toString().toLowerCase());//全用小写

usernameInText.add(now);//将匹配到的username都添加到usernameInText

}

}

**return** usernameInText;

}

### 实现Filter类

这个类实现对推特列表进行分类，根据时间，作者，推文内容等分类。这几个函数都很好写，writtenBy就遍历每篇推文的作者，inTimespan就遍历时间，containing要遍历推文内容，要忽略大小写。

### 实现SocialNetwork类

这个类是要根据推文内容推测用户之间的关系，我采用了最简单的方法，即如果A的 推文中@了B，那么就认为A认识B。最终想要得到Map<String, Set<String>>的结构，即每个人映射一个他@过的所有人。用之前实现的函数writtenBy和getMentionedUsers可以很容易得到。

**public** **static** Map<String, Set<String>> guessFollowsGraph(List<Tweet> tweets) {

Map<String, Set<String>> followMap = **new** HashMap<String, Set<String>>();

Set<String> author = **new** HashSet<String>();

**int** n = tweets.size();

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

Tweet now = tweets.get(i);

author.add(now.getAuthor().toLowerCase());

}

**for** (String str : author) {

List<Tweet> fromStr = Filter.*writtenBy*(tweets, str);

Set<String> usernameInText = Extract.*getMentionedUsers*(fromStr);

followMap.put(str, usernameInText);

}

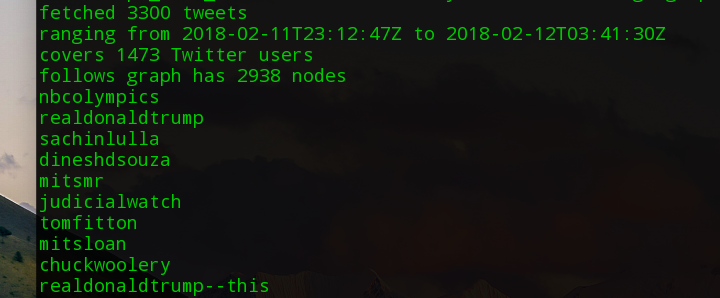
**return** followMap;

}

首先把所有推特遍历一遍，找到所有作者，构成一个集合，然后遍历这个集合。对于每个user，先用writtenBy找到他写的文章，然后用getMentionedUsers得到他@的人。

然后是一个测试，要你根据给出的推特（很多），找出谁被最多的人@过（who have the greatest influence）。用Map<String, Integer>这样的映射记录每个人和他的influence值。得到这个映射之后就进行排序，我选择了插入排序，边生成边排序。

### 运行结果截图



# 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 |
| 2018-02-28 | 14:00-15:30 | 编写问题1的isLegalMagicSquare函数并进行测试 | 按计划完成 |
| 2018-03-03 | 全天 | 编写问题2的函数并进行测试 | 按计划完成 |
| 2018-03-04 | 全天 | 编写问题3的函数并进行测试 | 按计划完成 |
| 2018-03-05 | 14:00-15:30 | 编写问题4的Extract类并进行测试 | 按计划完成 |
| 2018-03-06 | 下午 | 编写问题4的Filter类并进行测试 | 按计划完成 |
| 2018-03-08 | 下午 | 完成问题4 | 按计划完成 |
| 2018-03-15 | 全天 | 学着写build.gradle | 按计划完成 |

# 实验过程中遇到的困难与解决途径

实验过程中遇到的问题主要来自两个方面，一是对Java语法的不熟悉，二是对英语语法的不熟悉，分别通过搜索引擎和有道词典解决。

实验开始我没有着急写代码，而是学习git的原理与用法，这花费了我两个晚上，但是我觉得很有必要。

因为一个学期没用过Java所以有些陌生，在实现一些功能时都在网上查了一些文档或者看一看以前写过的代码，都很有帮助。

另外一个在实验中掌握的技术是正则表达式，以前只是听说过，因为没用过也就没掌握，但是在写实验三的时候我需要用到，所以去学习了一下。

还自学了如何写gradle脚本来自动化build。

# 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？//非常合
2. 关于Eclipse IDE //没用过别的，不做比较，但是我觉得还挺好用的
3. 关于Git//第一次使用这种版本控制工具，觉得很神奇，还在不断适应
4. 关于CMU和MIT的作业//题目都挺有意思，做完后能感到学了不少东西
5. 关于本实验的工作量、难度、deadline//难度主要体现在英语阅读上
6. 关于初接触“软件构造”课程//这门课是一门主要由实验构成的课，我非常喜欢自己动手去做一些东西，我相信实践出真知。

还有一点感受，就是在听课的时候，老师说的好多东西，我听起来都很抽象，不知道在说什么，等到自己真正用的时候才忽然想起这东西好像在哪听过。希望老师讲课的时候融入一些实例来讲解。