哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

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	朱明彦
学号	1160300314
班号	1603003
电子邮件	1160300314@stu.hit.edu.cn
手机号码	18846082306

目录

1 实验目标概述

实验的主要目的就是通过解决 4（3+1）个问题，来熟悉 Java 编程的基本技能，并且学会简单地阅读已有的代码框架补全所需的功能。能够为已开发的代码编写基本的测试程序并完成测试以保证代码的正确性。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 代码配置管理

2 实验环境配置

Eclipse 以前使用的是 Mars 版本，这次临时改成了 Eclipse Neno2，好在 Java 环境配置不需要改变。Git 环境以前配置好了，测试后可以使用。在这里给出你的 GitHub Lab1 仓库的 URL 地址（Lab1-学号）。

https://github.com/ComputerScienceHIT/Lab1-1160300314

3 实验过程

3.1 Magic Squares

Magic Squares，中文名字“幻方”。主要有两个函数，一个是 `isLegalMagicSquare` 判断已有的数字矩阵是否是一个幻方；另一个是利用已有的 `generateMagicSquare` 函数，来生成一个奇数行数的幻方。

3.1.1 isLegalMagicSquare()

//按步骤给出你的设计和实现思路/过程/结果。

1、从文件中读取数字矩阵，采用按行读取的方式，并另外设计了一个函数 boolean isConSpeCharaters(String string) 来判断，读取的每一行是否含有除了 0-9, '\t', '\n', '-', '.'之外的字符。其判断方式如下

```
if (string.replaceAll("\\d*\\t*\\n*-*.*", "").length() == 0) {
            // 如果不包含除了\t \d \n - .之外的字符
            return true;
        }
        return false;
```

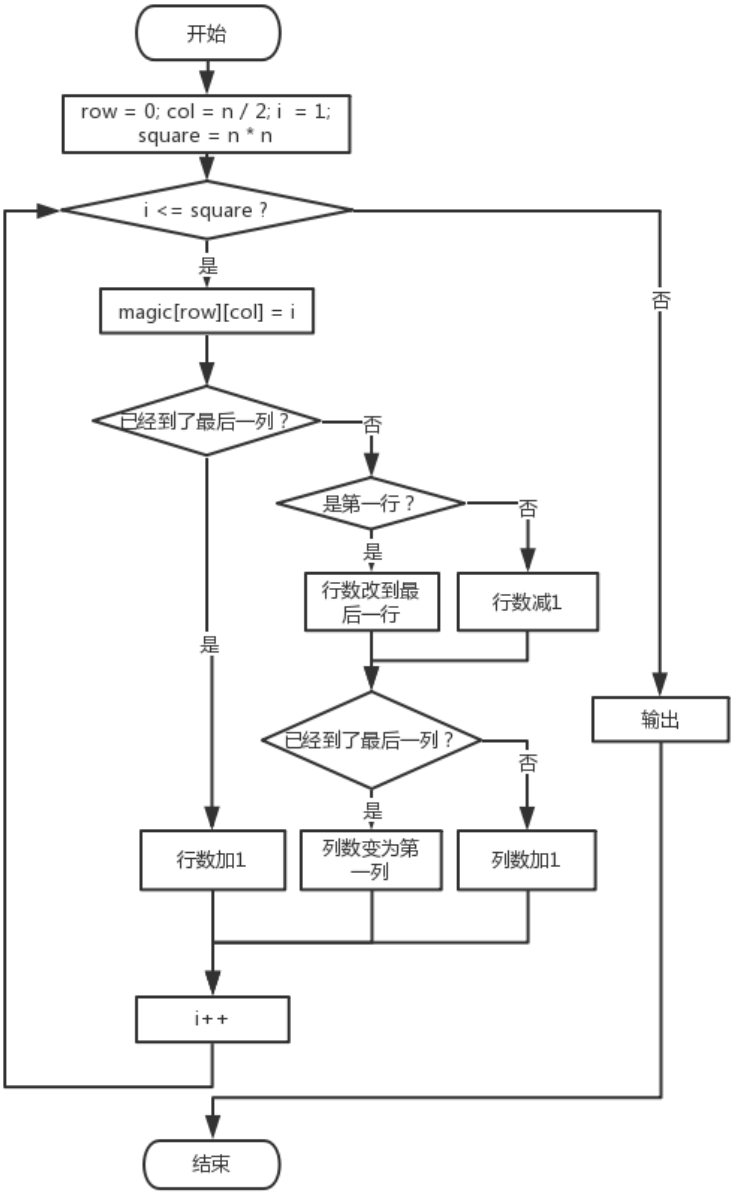
其中'\t'是用来分隔数字矩阵所采用的分隔符，而没有一次筛掉'-'和'.'是为了在后面更好地判断是否有负数和浮点数存在。

2、判断行列是否相等，用 String.split('\t')来分隔字符串。采用的方式是记录第一行的列数和总的行数，在每读进来一行并分隔后，将列数与第一行列数比较，如果出现不一致则必可以判定该数字矩阵存在列数和行数不等的情况。在全部读入后，判断行数和列数是否一致，来判断是否为方阵。

3、在保证数字矩阵为方阵后，将分隔的字符串转为整数。如果有异常抛出，则存在浮点数；如没有，判断是否有负数。如均无，则将第一行数字的和作为基准，分别将每一行、每一列、正负对角线上的数字求和与基准比较，出现不一致则证明不是幻方；否则，可以判定该数字矩阵为幻方。

3.1.2 generateMagicSquare()

正奇数幻方的生成有一个口诀，也就是 generateMagicSquare()使用的策略"一居上行正中央，依次斜填切莫忘；上出框时向下放，右出框时向左放；排重便在下格填，右上排重一个样"，采用流程图的方式可以这样表示：



1、java.lang.ArrayIndexOutOfBoundsException 是常见的数组越界造成的异常，即在代码中使用了不合法的索引访问数组导致数组越界。在使用偶数 n 的情况下，每一次循环运行到第(n/2 * n)次的时候，都会执行 row++，然而此时已经在整个数组的第(n-1)列，所以下一次访问就出现了越界。

2、java.lang.NegativeArraySizeException 是由于程序尝试建立大小为负的数组时抛出的异常。原因是 n 取负数时，建立二维数组的时候会将行数和

列数取成负数，导致该异常抛出。

3.2 Turtle Graphics

这个 Turtle 原本是来自 python 中的一个库，这个实验的目的也就是通过 Java 来实现一个简单的 Turtle 并实现简单的图形绘制的功能。

3.2.1 Problem 1: Clone and import

从https://github.com/rainywang/Spring2018_HITCS_SC_Lab1/tree/master/P2获取此次实验的实验包，在本地使用git bash建立git仓库，并且导入eclipse中已建立的项目。

3.2.2 Problem 3: Turtle graphics and `drawSquare`

这次实验其实本质是在让我们读一读代码，了解我们需要做什么。其实我认为这个实验的目的是去写 GUI，但是阅读代码后发现 GUI 实现部分已经写好了。任务就变成了如何使用 GUI。

`drawSquare`，由于开始时 turtle 是朝向 Y 轴正向，且每一次只能右转。所以只需要画四条等长边，转三次 90°即可。

3.2.3 Problem 5: Drawing polygons

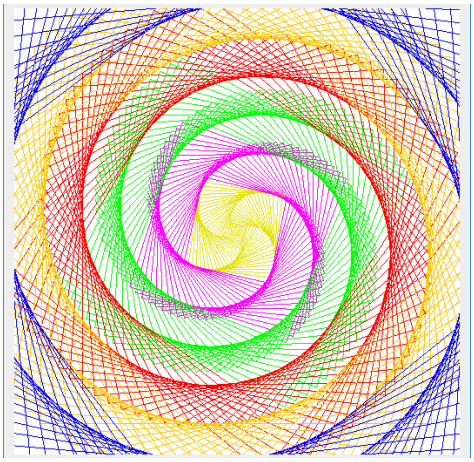
画正多边形的方法其实与画正方形一样，只是循环次数和每一次右转的角度更改一下即可。在实现 `drawRegularPolygon` 之前，已经实现了根据边数计算正 n 边形 `calculateRegularPolygonAngle` 函数，右转的角度就是正 n 边形的外角度数。

3.2.4 Problem 6: Calculating headings

实现 `calculatingHeadings` 其实主要在实现上一个 `calculateHeadingToPoint` 函数，根据当前的位置和朝向，判断去往目的位置需要右转的角度。直接利用 `Math.atan2` 函数计算出来由当前位置指向目的位置向量的与 X 轴正向的夹角，然后计算当前朝向的夹角，然后作差即可。主要注意不能出现负角度，所以合理增加 360 度。

3.2.5 Problem 7: Personal art

个人艺术环节，注意到在 `drawSquare` 函数的书写，如果将右转的角度稍微增加，那么每一次如果多旋转一点，画出来的形状类似正方形向外抛出的形状。在此基础上同时再更改颜色与边长就会有下面这个图形：



3.2.6 Submitting

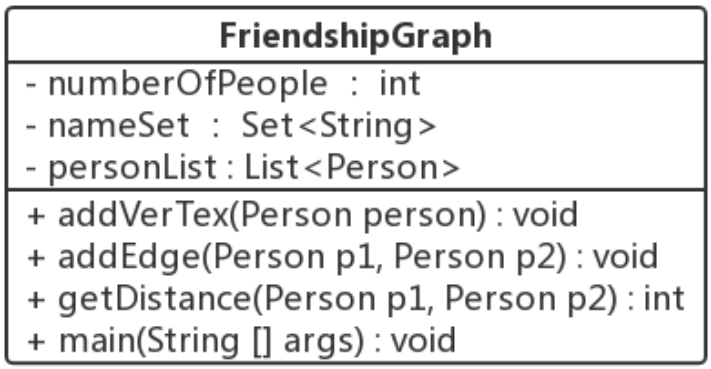
git add -A
git commit
git push
将所有的变化均 push 到 GitHub 中 Private 仓库中。

3.3 Social Network

主要任务就是实现两个类，`Person` 和 `FriendshipGraph` 类，并通过对图的操作，来判断两个人之间的最亲近距离。

3.3.1 设计/实现 `FriendshipGraph` 类

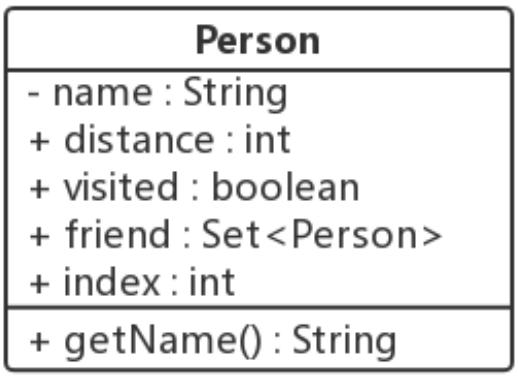
设计 UML 图如下



实现过程，`addVertex` 函数主要注意判断是否有重复的人名出现，利用 `Set` 来判断人名之间是否重复，同时记录初始化每个人的 index 变量并加入 `personList`，方便记录。`addEdge` 函数注意判断边的两个端点是否已经存在，如果存在则将对应的 person2 加入 person1 的“朋友”中。`getDistance` 利用 bfs 来寻找最短路径，只需要从 person1 开始进行广度搜索直到 person2 为止，如果找不到之间输出-1，否则输出 person2 的成员变量 distance 即可。（对于 Person 类的介绍 3.3.2 中有着详细介绍）

3.3.2 设计/实现 `Person` 类

设计 UML 图如下



`Person` 类有 5 个成员变量，`name` 为该对象的名字，与其对应的还有获得其名字的方法 `getName()`；其余的成员变量，distance 为在某一次询问中距离起点的距离，visited 为访问标志，friend 为该成员拥有的单向朋友的集合，index 为该成员加入某个 graph 时的顺序。

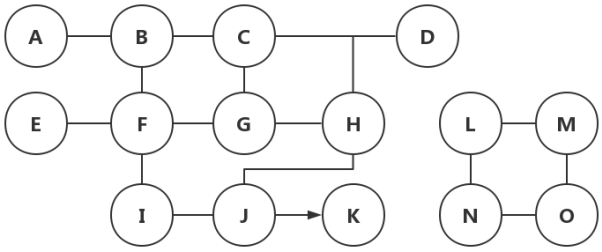
3.3.3 设计/实现客户端代码 `main()`

客户端代码的测试主要利用可以确定答案且与实验手册中已经给出的样例相同。建立新的社交网络，在不改变测试样例的情况下检验输出结果是否一致；然后将 rachel→ross 的边删掉，手工检验结果是否一致；将 ross 的名字替换为“Rachel”，检验程序能否正常退出。

在建立与原社交网络相同的结构后，输出与样例相同；删除掉 rachel→ross 的单向边，与手工检验的结果“-1, -1, 0, -1”相同；在将 ross 的名字替换后，程序提示错误“Rachel name has been used”，结束程序运行。

3.3.4 设计/实现测试用例

设计一个更大的图进行测试，其图示如下：



利用 JUnit 来测试 getDistance 函数在当前的测试样例中的表现

```
assertEquals(-1, graph.getDistance(iPerson, lPerson));
//验证有多个连通分量
assertEquals(2, graph.getDistance(iPerson, hPerson));
//验证最短路
assertEquals(2, graph.getDistance(lPerson, oPerson));
assertEquals(3, graph.getDistance(aPerson, hPerson));
assertEquals(4, graph.getDistance(aPerson, jPerson));
assertEquals(-1, graph.getDistance(dPerson, oPerson));
assertEquals(4, graph.getDistance(dPerson, kPerson));
assertEquals(-1, graph.getDistance(kPerson, dPerson));
```

与手工推导结果相同来验证正确性。

对于 void 类型的 addVertex 和 addEdge 函数则通过能否获取到由加入顶点和加入边所带来的副作用来判断其正确性：

```
assertTrue(graph.hasTheVertex(hPerson));
assertFalse(graph.hasTheVertex(new Person("None")));
assertTrue(graph.hasTheEdge(jPerson, kPerson));
assertFalse(graph.hasTheEdge(kPerson, jPerson));
```

另外为了测试更大的图，采用 Dijstra（Test\P3\FriendshipGraphForTest 中为用 Dijstra 进行的最短路实现）和 bfs 分别从两个不同的角度进行跑一个自动生成的图，然后利用 JUnit 进行测试判断，可以判断从 100-1000 点加入单向边随机生成的图均得到相同答案。

3.4 Tweet Tweet（选作，额外记分）

3.4.1 实现 Extract 类

1、实现 getTimepan 函数

getTimepan 目的即找到一个最小的区间能够覆盖所有的 Tweets 发布的时间，将所有的 tweet 一次扫过去，记录最早和最晚的时间点，然后将这两个时间作为起始时间返回即可。

2、实现 getMentionedUsers 函数

getMentionedUsers 函数的目的就是找到所有在 tweet 中被@过的用户名单，最重要的一点是将 tweet 里面的正文进行划分，找到所有可能的 username。此处通过正则匹配实现，基于所有的给出的 tweet 中每两个被@的 username 之间一定存在非用户名合法的字符（除了英文字母、数字、-、_之外）所以以下可用

```
String reg = "@[a-zA-Z0-9_-]{1, }";
Pattern pattern = Pattern.compile(reg);
Matcher matcher = pattern.matcher(tweet.getText());
```

然后简单加入 Set 中即可。

3.4.2 实现 Filter 类

1、实现 writtenBy 函数

writtenBy 函数的主要目的找到某一个特定用户所发的所有 tweet 的列表，简单将所有列表扫过，然后将某个人的 tweet 加入链表尾部即可。注意用户名是大小写不敏感的，只需要使用 equalsIgnoreCase 函数进行比较即可。

2、实现 inTimepan 函数

inTimepan 函数的目的是找到所有 tweet 中在某个时间段内所发的 tweet，只需要将所有 tweet 扫一遍，保证时间恰好处于合理区间内即可加入列表内。

3、实现 containing 函数

containing 函数的目的就是找到包含某些词的 tweet 的集合，只需要将所有的 tweet 内的正文对关键词进行扫描，如果包含就将其加入到最终的答案中即可。

3.4.3 实现 SocialNetwork 类

1、实现 guessFollowGraph 函数

guessFollowGraph 函数的目的就是找到某一个用户所@过的所有用户的集合，借助 Extract 类中的分离用户名的函数，然后将每一个用户的合在一起返回即可。

2、实现 influencer 函数

主要是将一系列的 tweet 中分析最有影响力一些用户，主要的评价标准就是利用有多少位 followers 来判断。利用 guessFollowGraph 函数将每一个用户@过的用户分别进行加入。然后根据 followers 的数量进行排序，就可以得到最有影响力的用户。

3、get Smarter

MIT 在这个问题上面的描述又开始变得模糊不清，因为这其中有很多的因素不是可以简单的量化的。比如就关注用户而言，如果 A 与 B 互相关注，B 和 C 互相关注，那么 A 和 C 之间肯定存在着某种关系，但是这种关系的却是不明确的，很难直接表达，A 和 C 之间有可能没有互相关注，甚至都不知道彼此；同样对于剩下的比如某个用户关注了一些话题“#something”，并且有很多的用户也关注这个问题，那么有两种可能，就是这个用户影响力很大，他关注了这个之后其余用户

也开始关注这个话题；或者这个话题原本就是很热的话题，比如 tweetpoll.py 里面给出的一样“#Olympic”就是一个很热的话题，这样不能简单的判断两个关注同一话题的用户之间的关系。

所以在这里，我只选用了一个策略来是”guessFollowGraph”变得“smarter”，就是三元关系，如果三个人之间有三元开环，那么就认为 A 和 C 之间有关系，即 A 和 C 互相关注。

4、设计 mySocialNetworkTest 类

要测试三元关系的成立，设计了一个包含 A 和 B 互相关注，B 和 C 互相关注，但是 A 和 C 之间没有相互关系的 tweet 列表，然后测试之后可以看到 A 的关注列表也有 C 出现。

3.4.4 Main 类

Main 类是进行测试其余三个任务的一个主类，在一个比较真实的情况下测试已有的代码。直接得到的结果如下图。

可以看到一共有 3300 条推特，覆盖了 1473 个用户，在用户中影响力最大的前十名用户分别是 NBColympics、RealdonaldTrump、SachinLulla、DineshDSouza、mitsmr、JudicialWatch 等等。

```
fetched 3300 tweets
ranging from 2018-02-11T23:12:47Z to 2018-02-12T03:41:30Z
covers 1473 Twitter users
follows graph has 2938 nodes

nbcolympics
realdonaldtrump
sachinlulla
dineshdsouza
mitsmr
judicialwatch
tomfitton
mitsloan
chuckwoolery
realdonaldtrump--this
```

4 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
2018-02-26	14:00-15:30	编写问题 1 的 `isLegalMagicSquare` 函数并进行测试	延期 1 小时完成
2018-02-26	16:30-17:30	编写测试 `generateMagicSquare`	正常完成
2018-02-27	8:00-9:30	编写 `drawSquare`、`calculateRegularPolygonAngle`、`calculatePolygonSidesFromAngle` 函数	正常完成
2018-02-27	19:00-23:00	编写 P2 中 `TurtleSoup` 剩余函数和简单实现 P3 中样例	正常完成
2018-02-28	10:15-11:30	编写 P3 中部分个人样例测试	正常完成
2018-02-28	18:30-22:00	尝试重构 P3 最短路实现	部分完成
2018-02-28	22:30-23:30	编写 `Extract` 和 `Filter`	正常完成
2018-03-01	8:00-11:30	编写 `SocialNetwork` 类	正常完成
2018-03-01	16:00-18:00	尝试修复 `Main` 类，自行挂 ftp 服务器	后发现 MIT 可以直接用，失败
2018-03-01	19:00-23:00	验证 `Main` 输出结果正确性，修改 `Extract` 类	未完成，延期到 2018-03-02
2018-03-02	8:00-10:00	写 P1/P2 实验报告	完成
2018-03-02	14:00-15:30	写 P3 实验报告	完成

5 实验过程中遇到的困难与解决途径

遇到的困难主要有 Java 好久不写确实语法忘记比较多，需要经常翻书复习；此外还有就是脱离 IDE 进行 build，没有接触过，试过了 gradle、maven 然后才在 Travis-CI 成功，主要还是通过 stackOverflow 上查找和询问学长来找一些经验来解决。

6 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？

(2) 关于 Eclipse IDE

(3) 关于 Git

(4) 关于 CMU 和 MIT 的作业

(5) 关于本实验的工作量、难度、deadline

(6) 关于初接触“软件构造”课程

（1）Java 编程好久不使用，但是捡起了一些基本语法编写起来还是十分顺手的，感觉比 C++更加好用。

（2）Eclipse IDE，用起来很流畅也很顺手，但是很多插件下载过慢，其余地相比 IDEA 都相差无几。

（3）Git 是个好工具，能够记录所有变化，虽然使用仍然极其不熟练，但是“无 git 不编程”，还是促使我好好学 git。

（4）CMU 的实验感觉更抽象，而且给的很多要求更死板一些；MIT 的实验看起来更开放，而且 turtle 等一些实验更加有趣，给的教程也更加详细让人更有兴趣做下去。

（5）工作量总的来说还是适中的，第一次实验来说还看不出什么，但作为 Java 语法复习的实验感觉这个实验还是相当友好的。但有一点，实验手册中的文件夹格式比较特殊跟 maven 的文件夹格式相差还是挺大的

（6）软件构造课程刚开始学习还是有太多概念，这一点让开始有一种“劝退”的感觉，还是需要仔细阅读材料，感觉概念理解比代码要难写挺多。