

**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 | Arcwoc |
| 学号 | \*\*\*\*\*\*\*\*\* |
| 班号 | \*\*\*\*\*\*\* |
| 电子邮件 | \*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 手机号码 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

**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。

①基本的Java OO编程

②基于 Eclipse IDE 进行 Java 编程

③基于 JUnit 的测试

④基于 Git 的代码配置管理

# 实验环境配置

在Ubuntu下配置Java运行环境，安装Eclipse，git，并且注册github账号并且绑定，安装并配置Junit，ant。

https://github.com/ComputerScienceHIT/Lab1-1160301012

# 实验过程

## Magic Squares

第一部分从格式和性质两个角度判断一个矩阵是否是Magic Squares，第二部分给出生成Magic Squares，需要完善代码并且判断生成的矩阵是否是Magic Squares.

### isLegalMagicSquare()

设计和实现思路：

①先判断格式是否正确，矩阵每行每列的元素个数是否相等，每个元素是否是有小数点，是否有符号，是否是正整数，是否都由\t隔开。不满足条件打印错误信息，并且返回false。

②按行、列、对角线分别判断元素之和是否相等。如果找到不相等的，打印错误信息，并且返回false

③都满足条件，返回true

结果：



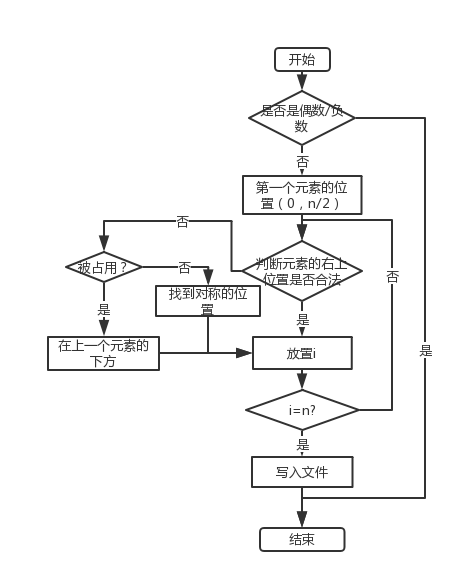
### generateMagicSquare()

设计和实现思路：

①判断输入的数字是否是负数或者偶数，如果是，则打印错误信息并且结束程序，否则继续执行。

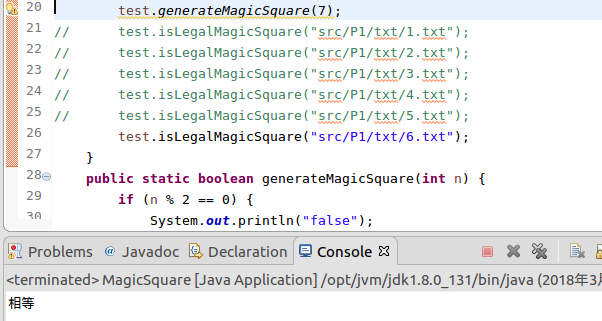
②完成矩阵

③写入文件

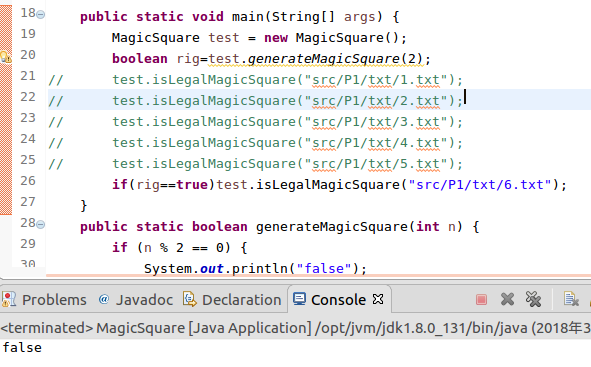


结果：

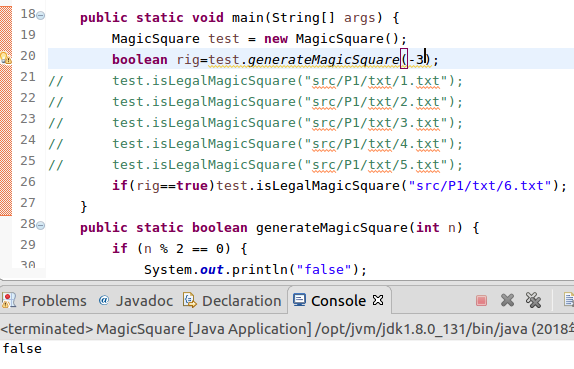
①奇数



②偶数



③负数



## Turtle Graphics

解决作图时会遇到的角度调整问题，并且利用得到的函数画出正方形，多边形，个人画作。

### Problem 1: Clone and import

打开终端输入：

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

然后import工程：

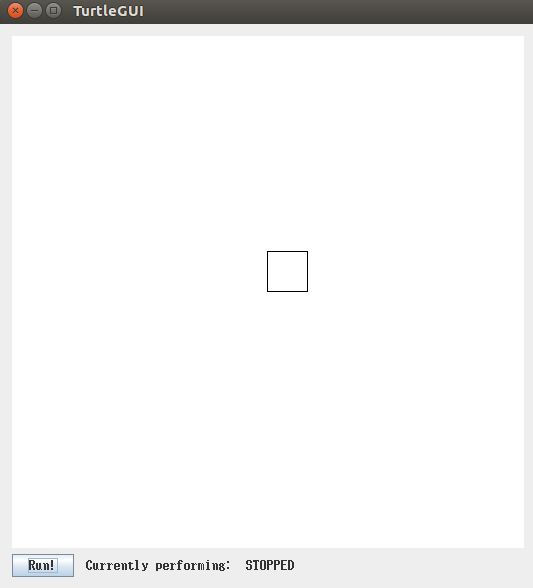
import->General->File System->选择文件->finish

### Problem 3: Turtle graphics and drawSquare

设计和实现思路：

初始方向竖直向上，向前sidelenghth长度并且旋转90°，循环4次.

结果：



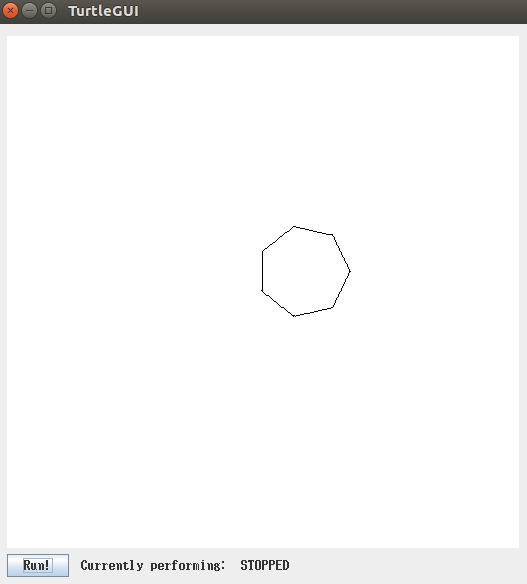
### Problem 5: Drawing polygons

设计和实现思路：

①初始方向竖直向上，先向前sidelength长度，再根据多边形的边数和calculateRegularPolygonAngle()计算要旋转的角度，旋转应转的角度。

②循环n次

结果：



### Problem 6: Calculating headings

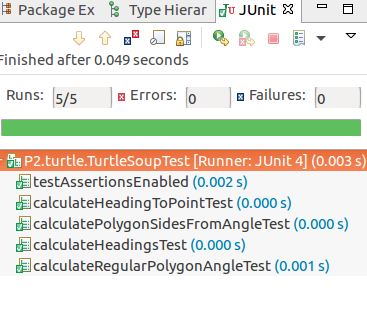
设计和实现思路：

①调用calculateHeadingToPoint()求第一个顶点到第二个顶点所需转的角度

②循环处理知道最后一个顶点，将结果加入List中

③返回List

结果：



### Problem 7: Personal art

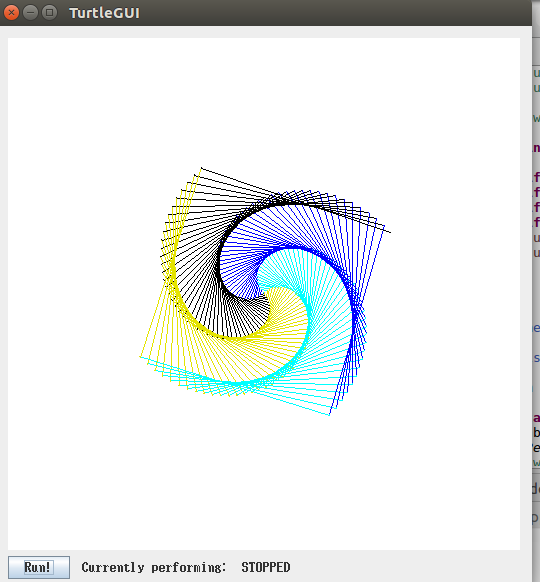
设计和实现思路：

①每次根据i的值改变画笔的颜色

②画笔移动距离i后旋转91°

③循环递增i

结果：



### Submitting

git add.

git commit -m ”your message”

git push origin master

## Social Network

用无向图来存储人与人之间的联系，可以实现加边，加点，用广度优先搜索求两点之间最短路径长度等功能。

### 设计/实现FriendshipGraph类

设计和实现：

①**public** **void** addVertex(Person name)

先判断要入的人是否已经存在，如果存在就结束程序。

如果不存在则加入顶点所在List

②**public** ArrayList<Person> addEdge(Person name1, Person name2)

将name1的List里加入name2，表示name2与name1邻接，返回一个表示与name1邻接的人的list

③**public** **int** getDistance(Person name1, Person name2)

利用广度优先搜索，从name1开始，利用邻接表访问与name1邻接的人，存入队列并且做相应的访问和距离标记，如果能够访问到name2，则返回最近距离。否则返回-1。

### 设计/实现Person类

设计和实现：

①**public** Person(String name)

构造函数，传入name并且初始化

②**public** **void** setDis(**int** dis)

设置从起始点到该点的距离

③**public** **int** getDis()

返回从起始点到该点的距离

④**public** **void** setFlag(**int** tf)

设置该点是否被访问过

⑤**public** **int** getFlag()

返回该点是否被访问过

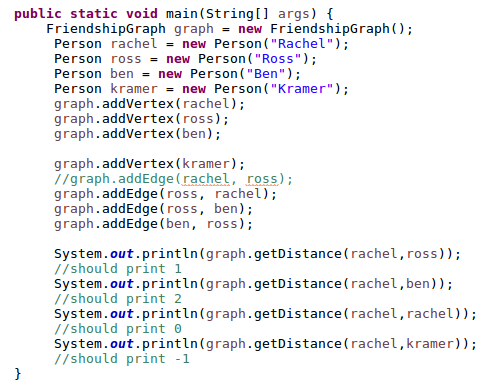
⑥**public** **void** setNum(**int** num)

设置该点的编号

⑦**public** **int** getNum()

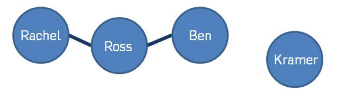
得到该点的编号

### 设计/实现客户端代码main()



### 设计/实现测试用例

测试用图：



**① public** **void** addVertexTest()

依次加入顶点Rachel，Ross，Ben，Kcramer，其编号分别是0,1,2,3

比较Kcramer的编号是否是3

*assertEquals*(kramer.getNum(),3);

②**public** **void** addEdgeTest()

*assertTrue*(graph.addEdge(ben,ross).contains(ross));

*assertTrue*(graph.addEdge(rachel,ross).contains(ross));

addEdge（)函数返回一个List，表示与name1相邻接的人，检查列表里是否存在name2，即可判断加边是否成功。

③**public** **void** getDistanceTest()

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

*assertEquals*(2,graph.getDistance(rachel,ben)); *assertEquals*(0,graph.getDistance(rachel,rachel)); *assertEquals*(-1,graph.getDistance(rachel,kramer));

如上图建立一个graph后，判断各个顶点之间的距离是否符合预期。

④如果将加入顶点的代码改成如下代码：

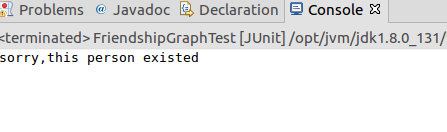
graph.addVertex(rachel);

graph.addVertex(rachel);

graph.addVertex(ben);

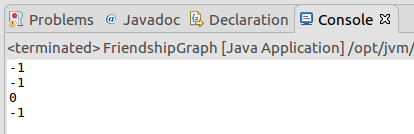
graph.addVertex(kramer);

运行结果为：

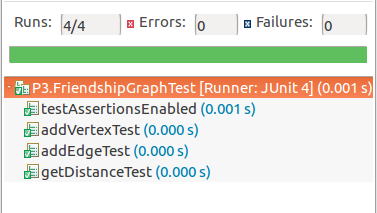


⑤如果去掉加入graph.addEdge(rachel, ross)这句代码

结果会是：



Test文件测试结果：



## Tweet Tweet（选作，额外记分）

**3.4.1 设计/实现Extract类**

设计和实现：

①**public** **static** Timespan getTimespan(List<Tweet> tweets)

将tweets的List传入函数，计算所有推文的起始时间和结束时间。

②**public** **static** Set<String> getMentionedUsers(List<Tweet> tweets)

循环遍历所有推文，正则匹配找到合法的名字，加入集合。

测试用例：

①

**private** **static** **final** Instant ***d1*** = Instant.*parse*("2016-02-17T10:00:00Z");

**private** **static** **final** Instant ***d2*** = Instant.*parse*("2016-02-17T11:00:00Z");

**private** **static** **final** Instant ***d3*** = Instant.*parse*("2016-02-17T11:30:00Z");

*assertEquals*("expected start", ***d1***, timespan.getStart());

*assertEquals*("expected end", ***d3***, timespan.getEnd());

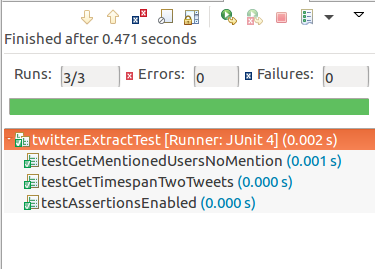
②

推文1："RTitution - interview I did - check out @FreedomofPress"

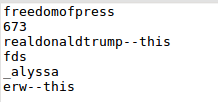
推文2："@673 @ d@432dsa. rivestundermine @RealDonaldTrump--this"

推文3："\n@fds\u2026 RT .@\_alyssa:(@erw--this): commentators, at -@er"

结果：



分离出来的姓名：



**3.4.2 设计/实现Filter类**

设计和实现：

**①public** **static** List<Tweet> writtenBy(List<Tweet> tweets, String username)

遍历推文，获得用户名，如果与想要的用户名相匹配，则把这个推文加入List

②**public** **static** List<Tweet> inTimespan(List<Tweet> tweets, Timespan timespan)

遍历推文，获得发文时间，如果在时间段内，则把这个推文加入List

③**public** **static** List<Tweet> containing(List<Tweet> tweets, List<String> words)

遍历所有推文，对每篇推文遍历所有的单词，如果找到相应的单词，则加入要返回的推文列表

用例

测试用例：

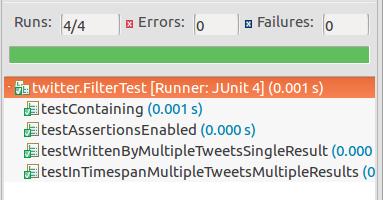
**private** **static** **final** Instant ***d1*** = Instant.*parse*("2016-02-17T10:00:00Z");

**private** **static** **final** Instant ***d2*** = Instant.*parse*("2016-02-17T11:00:00Z");

**private** **static** **final** Tweet ***tweet1*** = **new** Tweet(1, "alyssa", "is it reasonable to talk about rivest so much?", ***d1***);

**private** **static** **final** Tweet ***tweet2*** = **new** Tweet(2, "bbitdiddle", "rivest talk in 30 minutes #hype", ***d2***);

结果：



**3.4.3 设计/实现SocialNetwork类**

设计和实现：

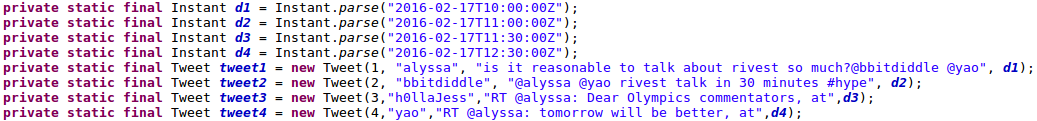
①**public** **static** Map<String, Set<String>> guessFollowsGraph(List<Tweet> tweets)

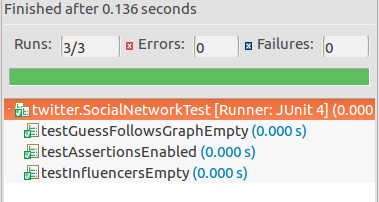
遍历所有推文，在推文中如果有提到合法的人名，表示该推文的作者关注了提到的人名，然后将提到的人名加入推文作者对应的关注的人员集合里。

②**public** **static** List<String> influencers(Map<String, Set<String>> followsGraph)

遍历所有人关注的用户集合，将某个用户的名字和被关注次数放入一个Map中，然后根据value的值排序，将用户的名字按照其被关注次数加入列表中。

测试用例：

结果：



**3.4.4 设计/实现MySocialNetwork类**

设计和实现：

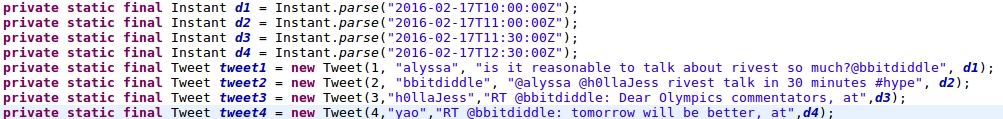
**①public** **static** Map<String, Set<String>> guessFollowsGraph(List<Tweet> tweets)

遍历所有推文，在推文中如果有提到合法的人名，表示该推文的作者关注了提到的人名，然后将提到的人名加入推文作者对应的关注的人员集合里。再根据A用户与B用户相互关注，B用户与C用户相互关注，得到A用户与C用户相互关注，将用户关系加入Map中。

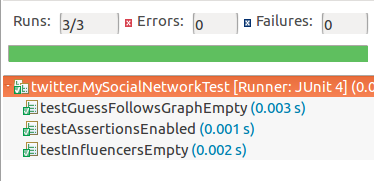
②**public** **static** List<String> influencers(Map<String, Set<String>> followsGraph)

遍历所有人关注的用户集合，将某个用户的名字和被关注次数放入一个Map中，然后根据value的值排序，将用户的名字按照其被关注次数加入列表中。

测试用例：



结果：



# 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 |
| 2018-03-01 | 15:00-17:30 | 编写问题2并进行测试 | 延期1小时完成 |
| 2018-03-01 | 19:00-23:00 | 编写问题3并进行测试 | 延期1小时完成 |
| 2018-03-02 | 15:00-22:00 | 编写问题4并进行测试 | 延期2小时完成 |

# 实验过程中遇到的困难与解决途径

①语言阅读有问题，解决途径：用谷歌翻译，同时记住常用单词，认真阅读要求，谨防出错

②git使用比较生疏，解决途径：找博客熟悉，自己建立仓库练习

③junit的安装和使用，解决途径：阅读博客和相关资料

④对工程进行build，解决途径：阅读相关文档和博客

⑤正则表达式的使用不熟练，解决途径：寻找相关教程，阅读尝试，最后写出合适的表达式匹配

# 实验过程中收获的经验、教训、感想

（1）我开始接触Java的时间比较晚，接触了之后我被Java丰富的库函数和面向对象编程的精妙吸引，虽然现在还不能很好的使用Java，还经常犯一些错误，但是我还是很享受这个过程和这个语言的。（2）Eclipse感觉还是比较好用的吧（3）Git很方便，是很强大的版本管理软件，常用操作简单，快速。（4）作业还没做过，老师在乐学网上留的问题挺好的（5）本实验的工作量可能是因为英语阅读问题，感觉有点大，可能后面会习惯吧，而且有时候细节理解上的错误会导致徒劳无功，比如task4的extract.java里要求包含所有推文的最短时间，结果理解错误，导致还得重新写。（6）初接触软件构造课程，觉得这个课程很重要，让我了解了软件开发的一些相对细节的东西。