

**2020年春季学期  
计算机学院《软件构造》课程**

**Lab 1实验报告**

|  |  |
| --- | --- |
| 姓名 | 陈广焕 |
| 学号 | 1190501614 |
| 班号 | 1903006 |
| 电子邮件 | 2718458514@qq.com |
| 手机号码 | 15778541719 |

**目录**

[1 实验目标概述 1](#_Toc72651726)

[2 实验环境配置 1](#_Toc72651727)

[2.1.1 下载Eclipse 1](#_Toc72651728)

[**2.1.2 下载JDK13** 1](#_Toc72651729)

[**2.1.3 下载Git** 1](#_Toc72651730)

[**2.1.4 注册GitHub账号并建立仓库** 1](#_Toc72651731)

[**2.1.5 创建项目工程文件并导入Junit** 1](#_Toc72651732)

[3 实验过程 2](#_Toc72651733)

[3.1 Magic Squares 2](#_Toc72651734)

[3.1.1 isLegalMagicSquare() 2](#_Toc72651735)

[3.1.2 generateMagicSquare() 4](#_Toc72651736)

[3.2 Turtle Graphics 4](#_Toc72651737)

[3.2.1 Problem 1: Clone and import 4](#_Toc72651738)

[3.2.2 Problem 3: Turtle graphics and drawSquare 4](#_Toc72651739)

[3.2.3 Problem 5: Drawing polygons 5](#_Toc72651740)

[3.2.4 Problem 6: Calculating Bearings 5](#_Toc72651741)

[3.2.5 Problem 7: Convex Hulls 6](#_Toc72651742)

[3.2.6 Problem 8: Personal art 6](#_Toc72651743)

[3.2.7 Submitting 8](#_Toc72651744)

[3.3 Social Network 10](#_Toc72651745)

[3.3.1 设计/实现FriendshipGraph类 10](#_Toc72651746)

[3.3.2 设计/实现Person类 11](#_Toc72651747)

[3.3.3 设计/实现客户端代码main() 11](#_Toc72651748)

[3.3.4 设计/实现测试用例 12](#_Toc72651749)

[4 实验进度记录 14](#_Toc72651750)

[5 实验过程中遇到的困难与解决途径 14](#_Toc72651751)

[6 实验过程中收获的经验、教训、感想 14](#_Toc72651752)

[6.1 实验过程中收获的经验和教训 14](#_Toc72651753)

[6.2 针对以下方面的感受 15](#_Toc72651754)

# 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

### 下载Eclipse

直接通过官网下载64bit的eclipse（2019-09R）

* + 1. **下载JDK13**

通过官网下载64bit 的jdk8，然后运行程序依次按照步骤安装Java和配置JDK。并在系统环境变量添加classpath变量和Java\_Home变量。

* + 1. **下载Git**

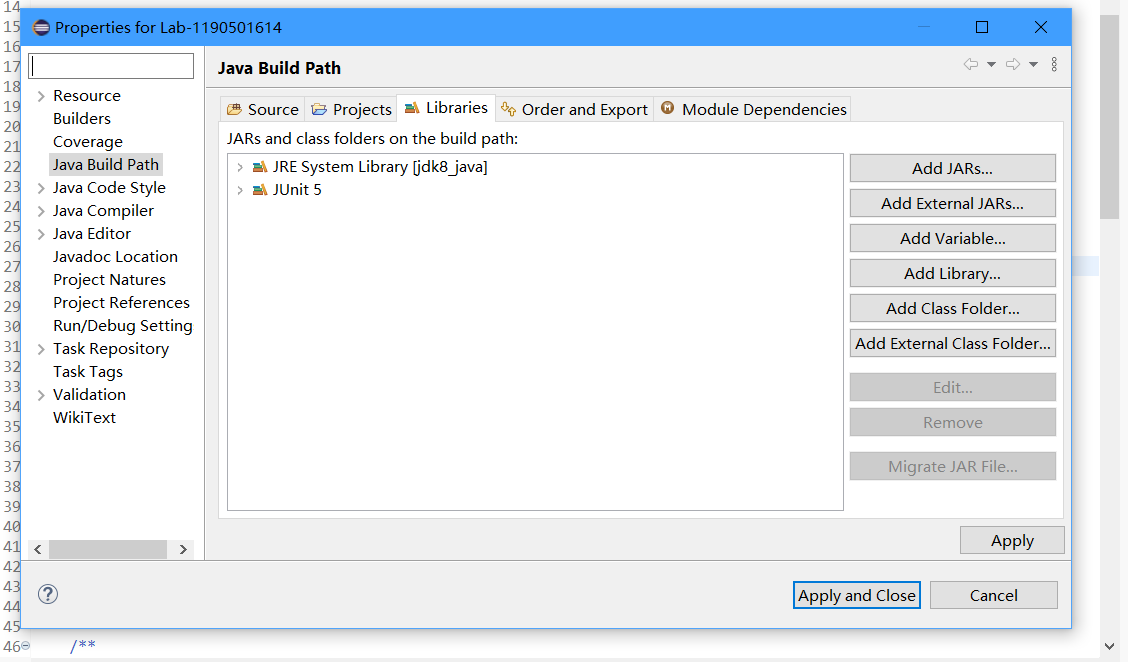
通过官网下载Git，并安装Git到本地磁盘

* + 1. **注册GitHub账号并建立仓库**

在GitHub官网，用自己的QQ邮箱注册一个GitHub账号，并按照老师给出的链接在classroom建立自己的Lab1仓库，命名为HIT-Lab1-1190501614。URL地址为<https://github.com/ComputerScienceHIT/HIT-Lab1-1190501614>

* + 1. **创建项目工程文件并导入Junit**

通过eclipse创建一个Java Project，选择JDK１.８作为编译运行环境，并通过building path 添加Junit5。



# 实验过程

## Magic Squares

任务1主要是判断一个矩阵是否为幻方和通过阶数n产生一个幻方，可分别构造函数Boolean isMagicSquare(String fileName)来实现，然后修改ppt中的generateMagicSquare函数对非法输入“优雅”退出。

### isLegalMagicSquare()

该函数功能为判断一个函数是否为幻方，如果是，则返回true，否则输出错误提示信息并返回false。

1. 首先读入文件

通过创建FileReader、BufferReader、StringBuilder对象读入txt中的文件，并抛出异常

File fP= **new** File(fileName);

InputStreamReader reader = **new** InputStreamReader(**new** FileInputStream(fP));

BufferedReader br = **new** BufferedReader(reader);

String line = "";

line = br.readLine();

1. 将字符数字转化为数值数字写入数组

通过split()函数将字符串按”\t”分割，由Integer.*valueOf()函数将其转化为int型数字，同*时同布尔数组vis判断数字是否重复出现过，如果重复出现则输出错误信息“Contain Same Number"并返回false。如果出现异常，则抛出异常，输出错误信息返回false，否则将数字写入一个二维数组中，同时检查每一行元素个数是否相等，最后检查行列数是否相等。

1. 通过数组判断对角线之和是否相等并作为判断基准

**for**(**int** i=0;i<num0;i++)

{

sum+=*MSquare*[i][i];

sum1+=*MSquare*[num0-1-i][i];

}

**if**(sum1!=sum)

{

System.***out***.println("sum diff");

**return** **false**;

}

1. 计算矩阵的行列和，判断是否相等

**for**(**int** i=0;i<num0;i++)

{

**int** tsum=0;

**int** tsum1=0;

**for** (**int** j=0;j<num0;j++)

{

tsum +=*MSquare*[i][j];

tsum1 +=*MSquare*[j][i];

}

**if**(tsum!=sum||tsum1!=sum)

{

System.***out***.println("sum diff");

**return** **false**;

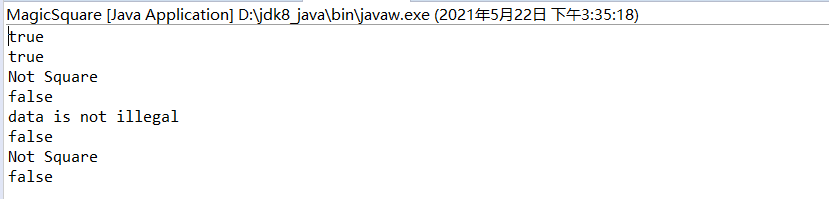
}

}

1. 判断是否为幻方

如果前面的条件都满足，则为幻方，返回true。

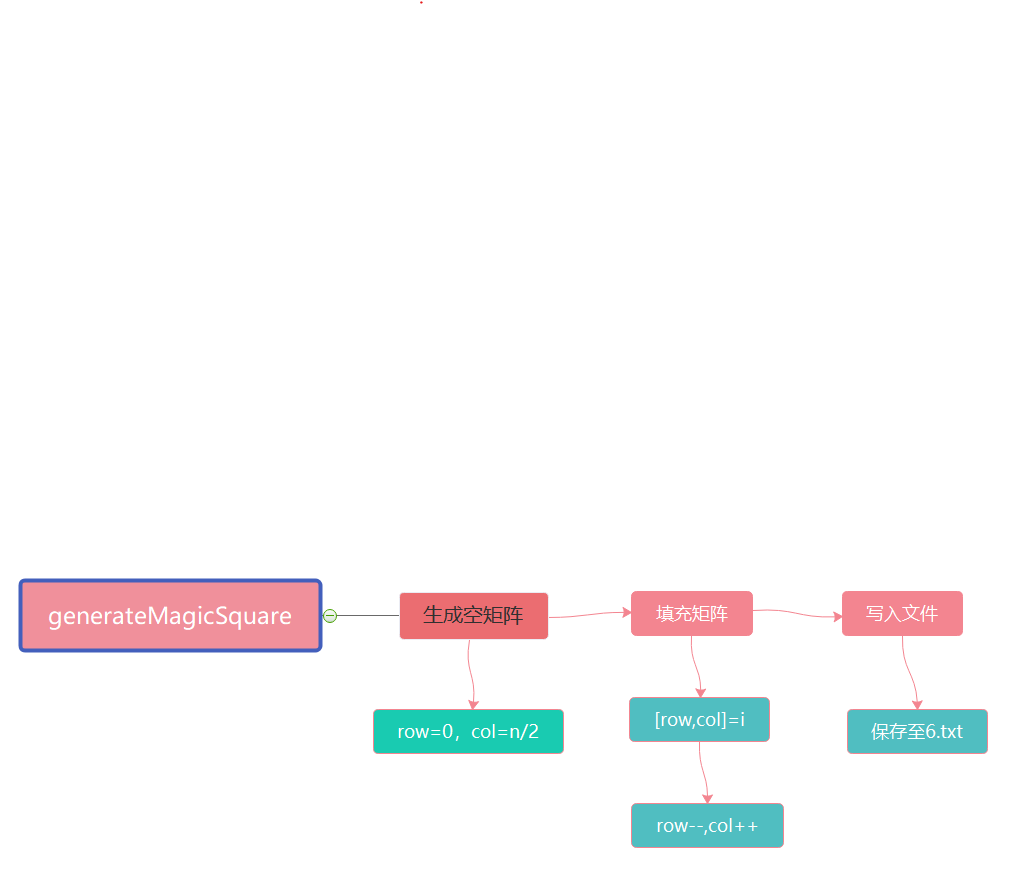
1. 程序运行结果输出案例：



### generateMagicSquare()

该函数主要功能是产生一个阶数为奇数的幻方，并允许对异常输入。

1. 生成一个n\*n的空矩阵
2. 循环n\*n次填充矩阵
3. 将结果保存至文件“src\\P1\\txt\\6.txt”
4. 程序流程图



1. 程序运行结果：







## Turtle Graphics

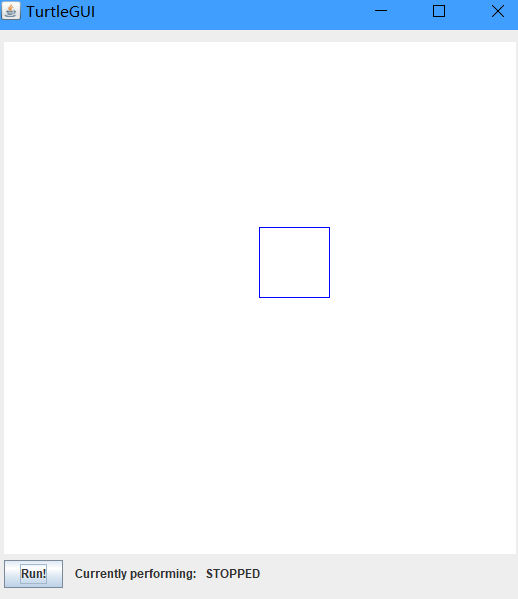
该任务要求我们clone和import已有的程序后，利用turtle的类和接口进行按照要求画图，主要是实现 TurtleSoup类里面函数的具体功能。

### Problem 1: Clone and import

通过老师给出另一条链接直接到GitHUb下载压缩包，然后解压到本地磁盘，导入到项目工程中

### Problem 3: Turtle graphics and drawSquare

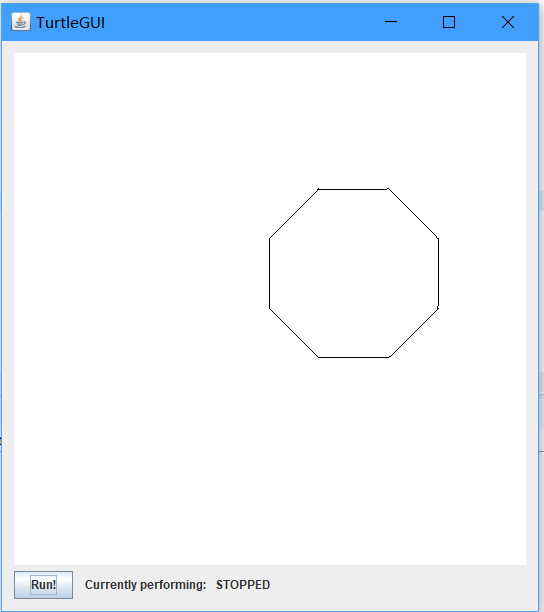
画一个正方形，每次向前移动相同距离，然后旋转90°，循环四次即可得到正方形。



### Problem 5: Drawing polygons

首先在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：

使用该公式，实现

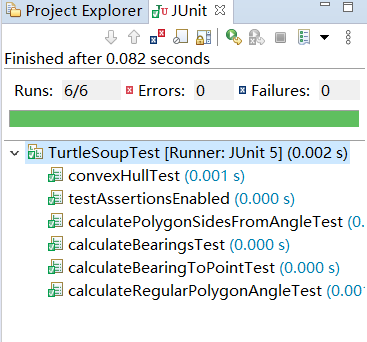


### Problem 6: Calculating Bearings

* 1. 已知起点和当前朝向角度，想知道到终点需要转动的角度。首先使用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，再减去90°，最后模360°调整角度

1. 基于上一个问题，此时有若干个点，想知道从第一个点开始到第二个点，再从第二个点到第三个点……以此类推每次转向的角度。可通过循环实现，起点为第一个点，终点为最后一个点。每次计算结果保存至List中。

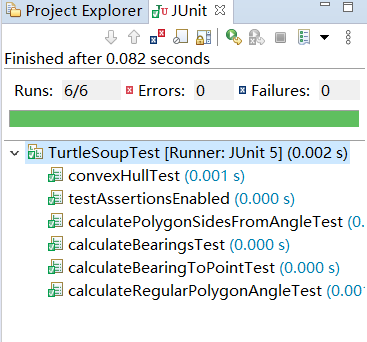


### Problem 7: Convex Hulls

**凸包算法**

这里使用Gift-Wrapping算法。我们发现任意凸包上的点，你会发现以该点建立一个极角坐标系，该点连结其它所有点的极角中，该点逆时针方向的第一凸包点到该点极角最小，例如P0，到所有点的极角中P0P1极角最小。

算法中首先找到最左边的点，这个点必然在凸包上，然后计算该点连接点极角最小的，这里计算有技巧，算法中进行试验,直到找到到最右端的点，找到P1后，就可以从P1开始，接着顺次找到P2，又以P2为起点……



### Problem 8: Personal art

**public** **static** **void** drawPersonalArt(Turtle turtle) {

**int** size0 = 80, colornum = 5;

**for**(**int** j=0;j<3;j++)

{

**for** (**int** i = 1; i <= size0; i++) {

**switch** (i % colornum) {

**case** 0:

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

**break**;

**case** 1:

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

**break**;

**case** 2:

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

**break**;

**case** 3:

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

**break**;

**case** 4:

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

**break**;

**case** 5:

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

**break**;

}

*drawRegularPolygon*(turtle,7,8+i);

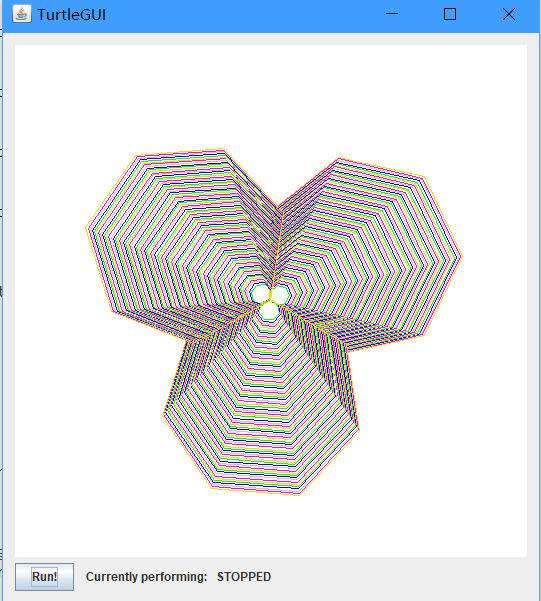
}

turtle.turn(120);

}

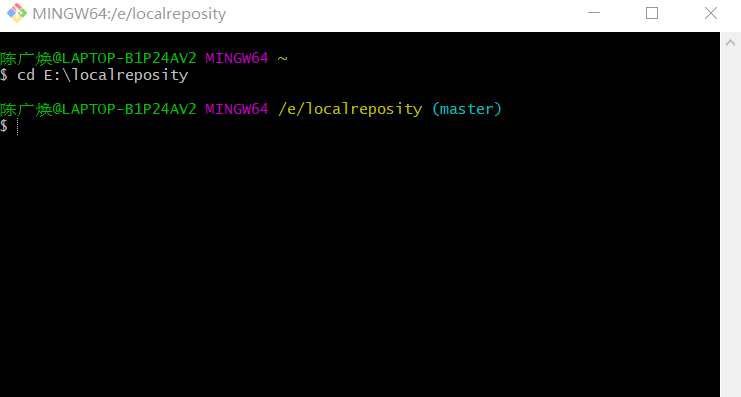
}

结果图：

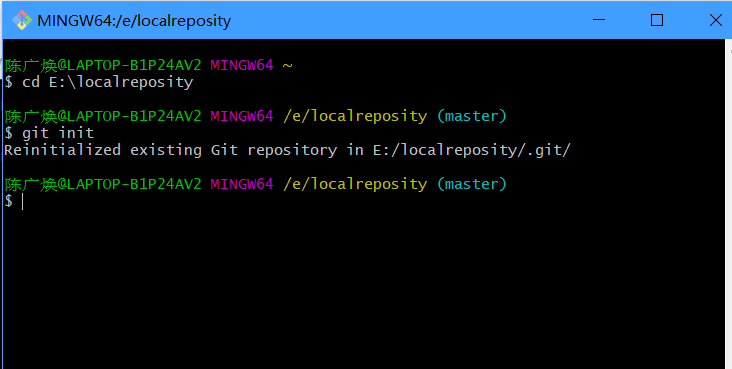


### Submitting

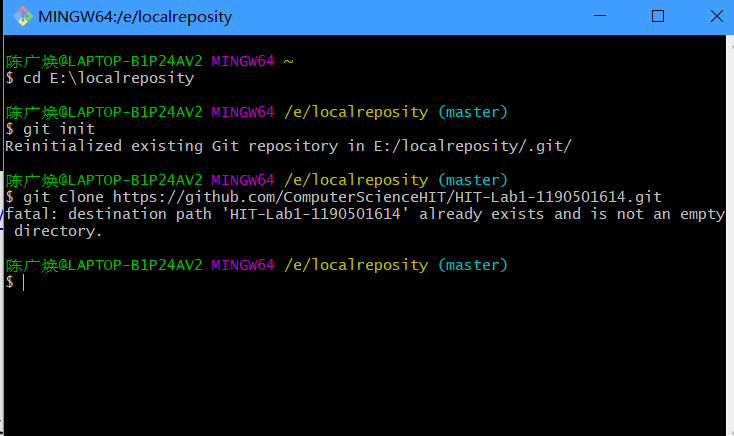
1. 建立文件夹（localreposity）
2. 打开Git bash
3. cd E:\\localreposity



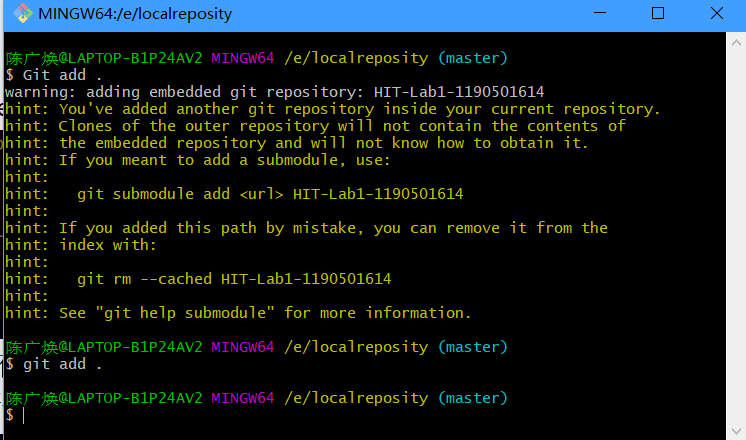
1. git init



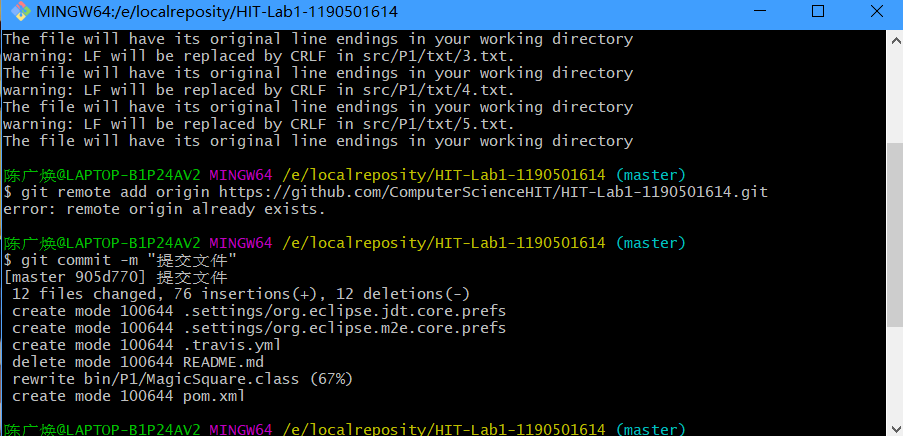
1. git clone <https://github.com/ComputerScienceHIT/HIT-Lab1-1190501614.git>



1. Git add .



1. git commit -m “提交文件”
2. git remote add origin https://github.com/ComputerScienceHIT/HIT-Lab1-1190501614.git



1. git push -u origin master

## Social Network

该任务要求设计一张社交网络图，连接人与人，并且能计算任意两人之间的联系情况。网络图基于FriendshipGraph类和Person类。

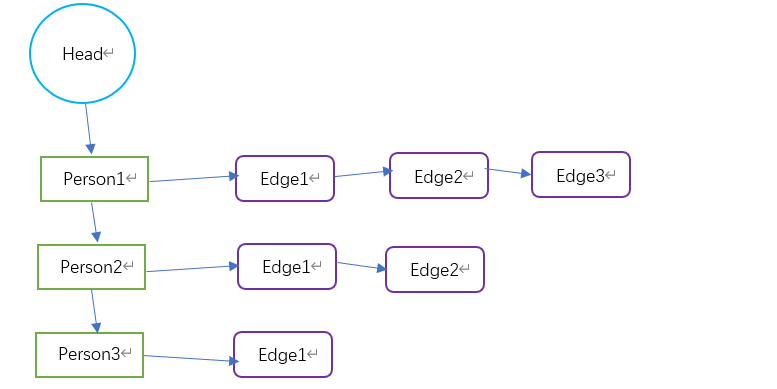
### 设计/实现FriendshipGraph类

该类实际是一个关系图，包括了代表每个Person的结点、代表人与人之间关系的边、以及建立点和联系和计算距离的方法函数。

* 1. 通过邻接表存储结点和边

每次new person就建立一个新结点，并判断是否与之前的结点同名，每次加一条边就在相应结点的链上加边。

具体关系如下图：



* 1. 结点类（class Node）

将每一个person转化为邻接表里面的结点，通过Node next指向下一个结点，通过int dis记录距离。通过addNode()和addNodeEdge()方法实现添加结点和边

* 1. addVertex()方法

增加一个新的节点，参数是要加入的Person类。首先，方法要对Person的名字进行判重：用哈希集合HashSet记录下已加入的所有Person的名字，每当新加入一个Person则进行判断是否在集合中；然后则新建一个Node类，使每一个Person与一个Node对应起来

* 1. addEdge()

由于题目默认关系是双向的，每次添加边，调用两次addNodeEdge添加正反方向的边

* 1. getDistance()

获得两个点之间的距离，使用广度优先搜索即可

### 设计/实现Person类

该类的目标是将每一个人对应到一个Person对象，并存储名字的信息。通过Node变量对应一个结点。

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

**public** String Name;

**public** Node node = **null**;

**public** Person(String PersonName) {

Name = PersonName;

}

}

### 设计/实现客户端代码main()

1. 名字重复引发错误测试：

**public** **void** GTest2() {

FriendshipGraph graph = **new** FriendshipGraph();

Person per1 = **new** Person("per1");

graph.addVertex(per1);

Person per2 = **new** Person("per2");

graph.addVertex(per2);

//同名时出现异常

Person per3 = **new** Person("per1");

graph.addVertex(per3);

}

1. 基本测试：

**public** **void** GTest1() {

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));// 1

\* System.out.println(graph.getDistance(rachel, ben));// 2

\* System.out.println(graph.getDistance(rachel, rachel));// 0

\* System.out.println(graph.getDistance(rachel, kramer));// -1

\*/

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

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

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

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

}

### 设计/实现测试用例

1. 重复名字测试用例

三个Person对象，其中per1和per3具有相同名字为“per1”

Person per1 = **new** Person("per1");

graph.addVertex(per1);

Person per2 = **new** Person("per2");

graph.addVertex(per2);

Person per3 = **new** Person("per1");

graph.addVertex(per3);

（2）简单测试

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);

（3）复杂测试：

Person pa = **new** Person("A");

Person pb = **new** Person("B");

Person pc = **new** Person("C");

Person pd = **new** Person("D");

Person pe = **new** Person("E");

Person pf = **new** Person("F");

graph.addVertex(pa);

graph.addVertex(pb);

graph.addVertex(pc);

graph.addVertex(pd);

graph.addVertex(pe);

graph.addVertex(pf);

graph.addEdge(pa, pb);

graph.addEdge(pa, pd);

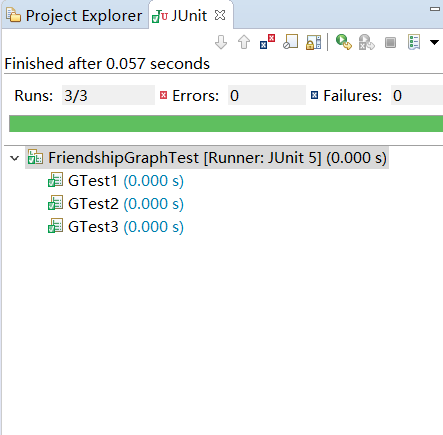
graph.addEdge(pb, pd);

graph.addEdge(pc, pd);

graph.addEdge(pd, pe);

graph.addEdge(pc, pf);

测试结果：



# 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 |
| 2021-5-17 | 下午 | 编写问题1的isLegalMagicSquare函数并进行测试 | 出现bug，超时 |
| 2021-5-17 | 晚上 | 完善isLegalMagicSquare | 按时完成 |
| 2021-5-18 | 晚上 | Turtle | 遇到困难，未完成 |
| 2021-5-18 | 晚上 | Turtle | 遇到BUG，未完成 |
| 2021-5-19 | 晚上 | Turtle | 超时完成 |
| 2021-5-20 | 下午 | 准备FriendshipGraph | 未完成 |
| 2021-5-21 | 下午，晚上 | FriendshipGraph | 完成 |
| 2021-5-22 | 下午，晚上 | 写报告 | 完成 |
| 2021-5-23 | 下午 | 提交文件 | 完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| Eclipse导入turtle和rules时，出现bug | 1. 没有导入Junit，通过build path导入了Junit5解决问题 2. 导入包错误，将turtle改为P2.turtle，rules改为P2.rules |
| 不会用Git | 通过网上查找资源知道了如何提交文件 |
| 不会Java编程 | 查找网上资源现学 |

# 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

仍需提高自己的代码能力和算法水平

## 针对以下方面的感受

1. Java编程语言是否对你的口味？

符合，和C++编程习惯类似

1. 关于Eclipse IDE

不习惯，之前用到的是IDEA

1. 关于Git和GitHub

还行

1. 关于CMU和MIT的作业

英文题，看着不舒服

1. 关于本实验的工作量、难度、deadline

大

1. 关于初接触“软件构造”课程

难