哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

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

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

2 实验环境配置

2.1.1 下载 Eclipse

直接通过官网下载 64bit 的 eclipse（2019-09R）

2.1.2 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。

2.1.3 下载 Git

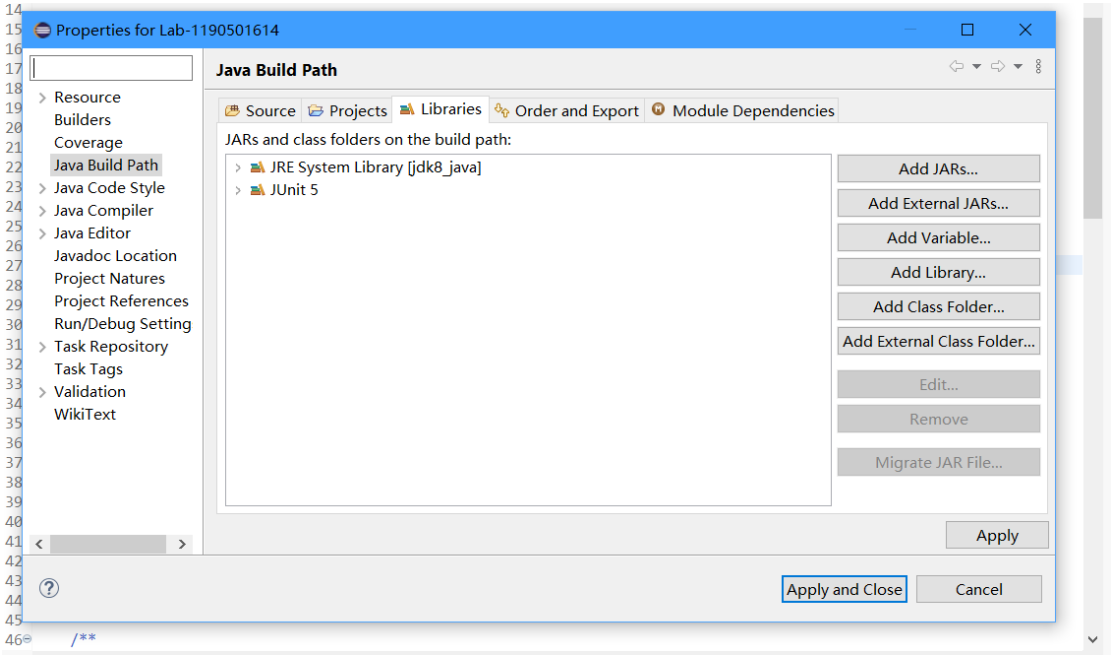
通过官网下载 Git，并安装 Git 到本地磁盘

2.1.4 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

2.1.5 创建项目工程文件并导入 Junit

通过 eclipse 创建一个 Java Project，选择 JDK 1．8 作为编译运行环境，并通过 building path 添加 Junit5。



3 实验过程

3.1 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。

3.1.1 `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();
```

2. 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。

3. 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;
}
```

4. 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;
    }
}
```

5. 判断是否为幻方

   如果前面的条件都满足，则为幻方，返回 true。

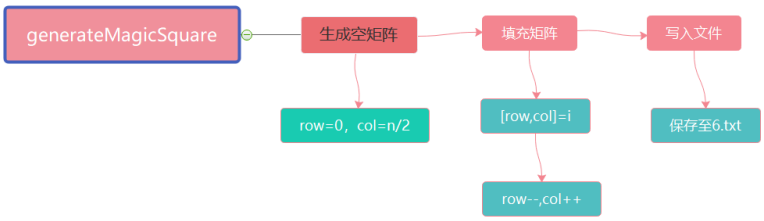
6. 程序运行结果输出案例：

```
MagicSquare [Java Application] D:\jdk8_java\bin\javaw.exe (2021年5月22日 下午3:35:18)
true
true
Not Square
false
data is not illegal
false
Not Square
false
```

3.1.2 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



5. 程序运行结果：

```
6
n is illegal
IO exception, this file is empty
false
```

```
7
true
```

```
-3
n is illegal
IO exception, this file is empty
false
```

3.2 Turtle Graphics

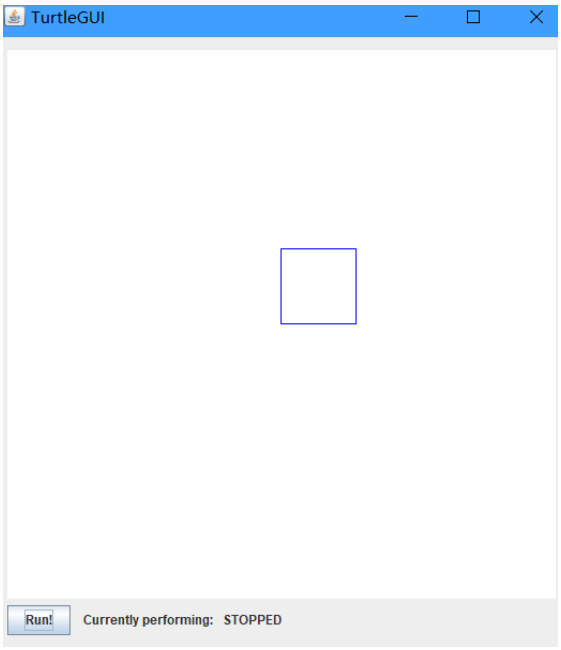
该任务要求我们 clone 和 import 已有的程序后，利用 turtle 的类和接口进行按照要求画图，主要是实现 TurtleSoup 类里面函数的具体功能。

3.2.1 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

3.2.2 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。

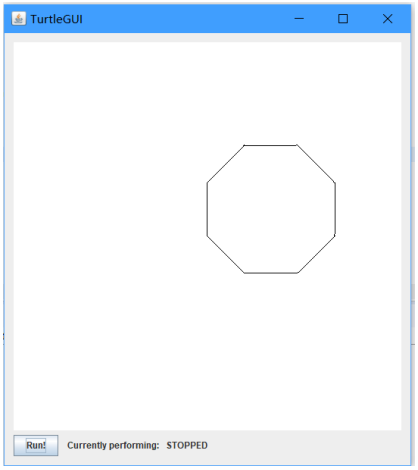


3.2.3 Problem 5: Drawing polygons

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：

$$(double)\ 180.0\ -\ (double)\ 360.0\ /\ sides$$

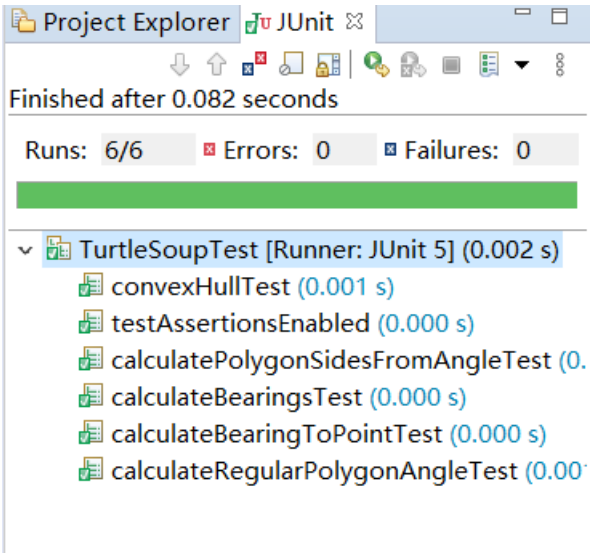
使用该公式，实现$calculateRegularPolygonAngle$



3.2.4 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$Math$ $.atan2$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
2. 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。

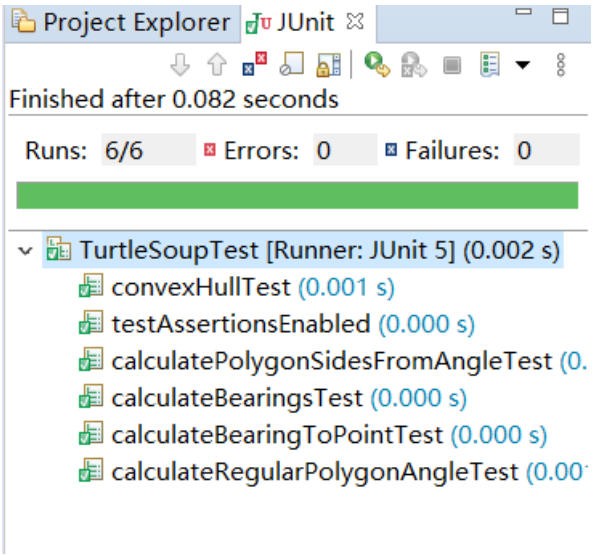


3.2.5 Problem 7: Convex Hulls

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……

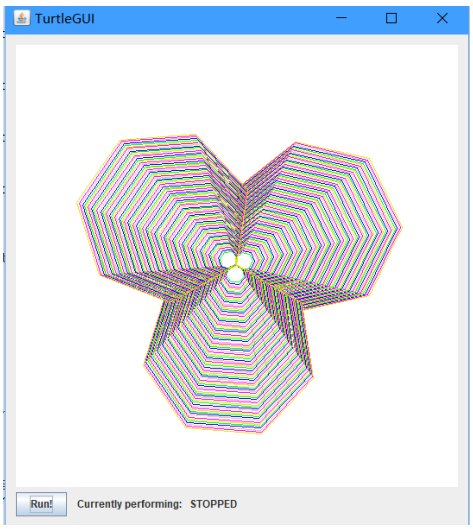


3.2.6 Problem 8: Personal art

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

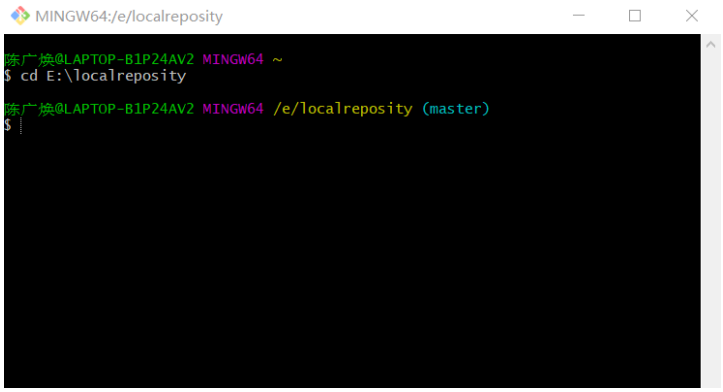
```
int size0 = 80, colornum = 5;
for(int j=0;j<3;j++)
{
 for (int i = 1; i <= size0; i++) {
     switch (i % colornum) {
        case 0:
            turtle.color(PenColor.YELLOW);
            break;
        case 1:
            turtle.color(PenColor.GREEN);
            break;
        case 2:
            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：

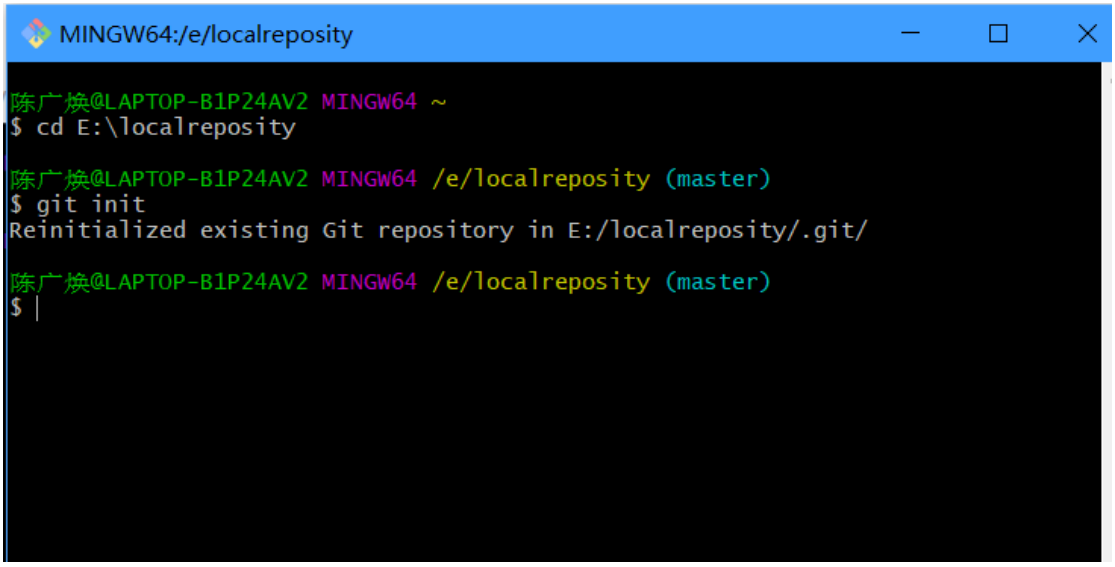


3.2.7 Submitting

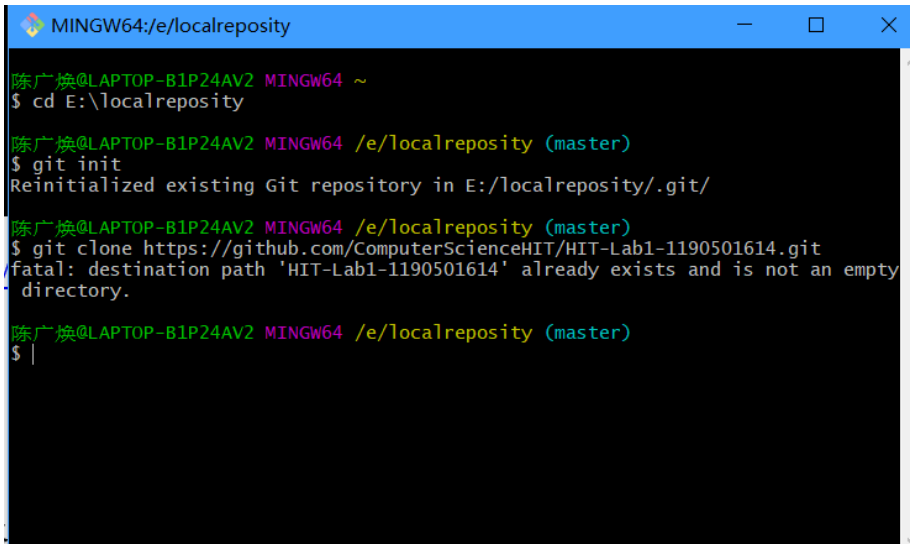
1. 建立文件夹（localreposity）
2. 打开 Git bash
3. cd E:\\localreposity



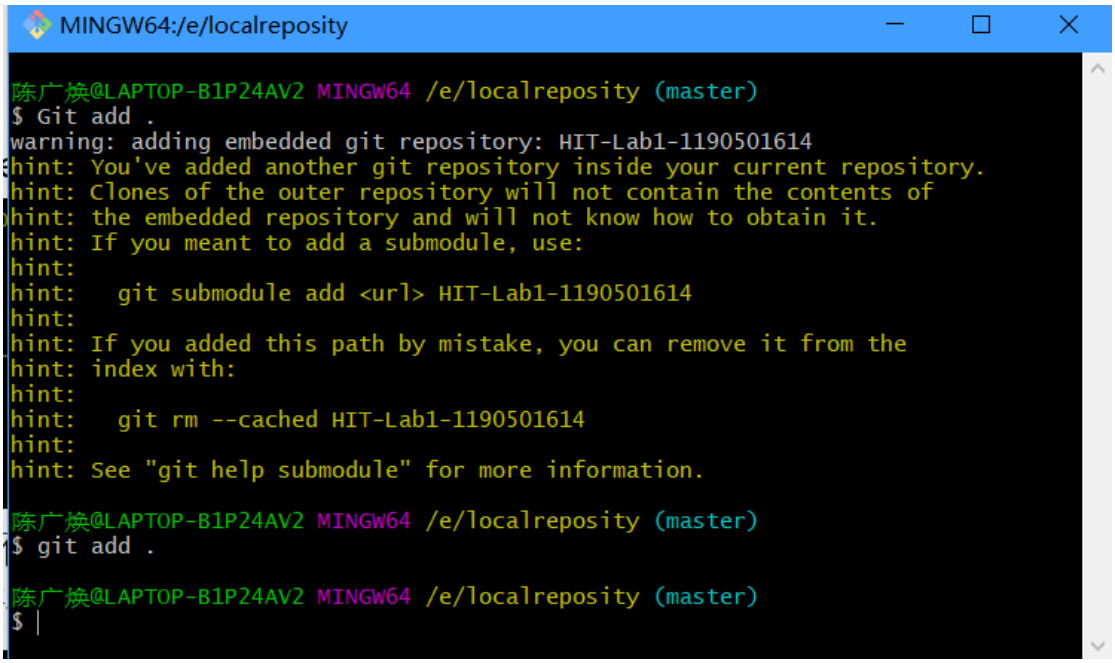
4. git init



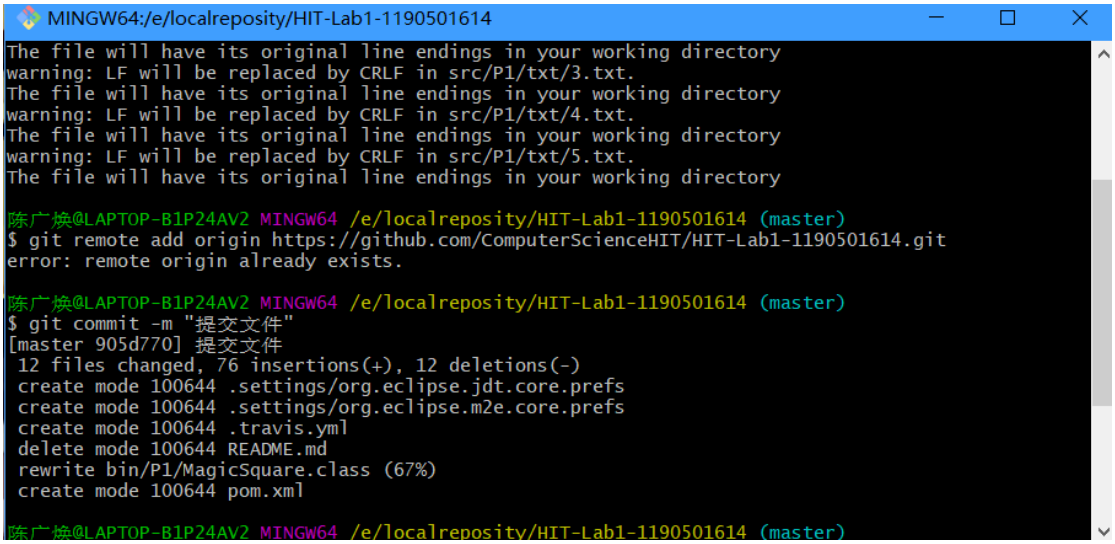
5. git clone https://github.com/ComputerScienceHIT/HIT-Lab1-1190501614.git



6. Git add .



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



9. `git push -u origin master`

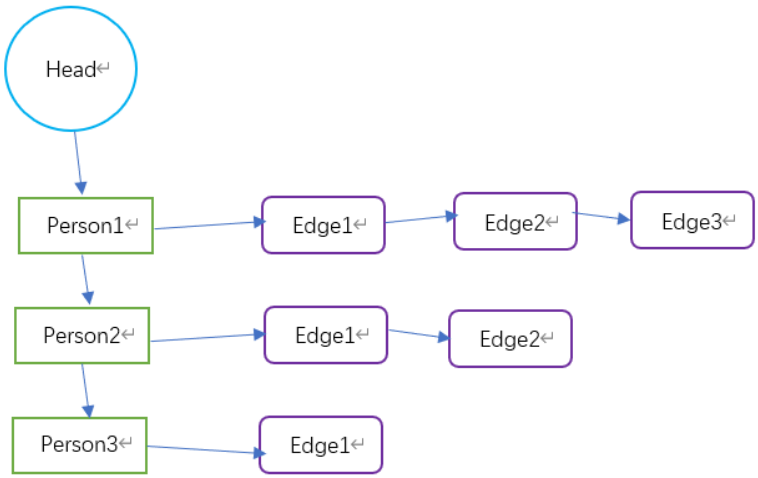
3.3 Social Network

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。

3.3.1 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：



2. 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

3. 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
4. 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
5. getDistance()
   获得两个点之间的距离，使用广度优先搜索即可

3.3.2 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;
}
}
```

3.3.3 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);
  }
```

2. 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));
}
```

3.3.4 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

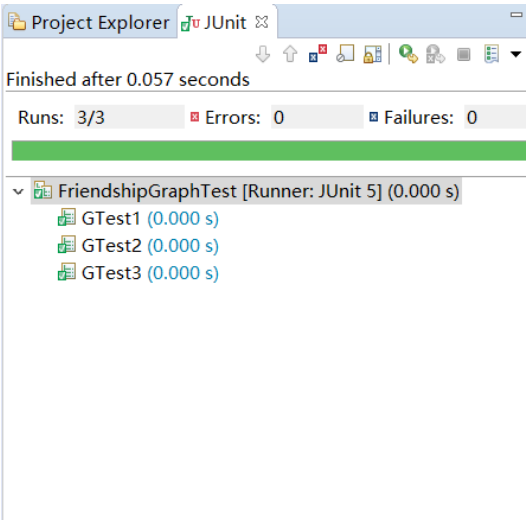
```
Person rachel = new Person("Rachel");
Person ross = new Person("Ross");
Person ben = new Person("Ben");
Person kramer = new Person("Kramer");

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：



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
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

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
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

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

仍需提高自己的代码能力和算法水平

6.2 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

(2) 关于 Eclipse IDE

不习惯，之前用到的是 IDEA

(3) 关于 Git 和 GitHub

还行

(4) 关于 CMU 和 MIT 的作业

英文题，看着不舒服

(5) 关于本实验的工作量、难度、deadline

大

(6) 关于初接触“软件构造”课程

难