

**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 | 郭茁宁 |
| 学号 | 1183710109 |
| 班号 | 1837101 |
| 电子邮件 | 847738428@qq.com |
| 手机号码 | 13905082373 |

# 目录

[0 目录 2](#_Toc34065036)

[1 实验目标概述 1](#_Toc34065037)

[2 实验环境配置 2](#_Toc34065038)

[2.1 Java & Eclipse & Maven 使用配置方法 2](#_Toc34065039)

[2.1.1 下载Eclipse 2](#_Toc34065040)

[2.1.2 下载JDK13 2](#_Toc34065041)

[2.1.3 配置JDK 2](#_Toc34065042)

[2.1.4 编辑器字体修改 2](#_Toc34065043)

[2.1.5 修改默认编码 2](#_Toc34065044)

[2.1.6 设置代码自动提示 3](#_Toc34065045)

[2.1.7 配置 Maven 3](#_Toc34065046)

[2.2 JUnit 4](#_Toc34065047)

[2.2.1 新建Java项目 4](#_Toc34065048)

[2.2.2 编写代码 & 打包 5](#_Toc34065049)

[2.2.3 将JUnit加入Library 5](#_Toc34065050)

[2.2.4 创建Junit Test Case 6](#_Toc34065051)

[2.2.5 查看测试结果 7](#_Toc34065052)

[3 实验过程 8](#_Toc34065053)

[3.1 Magic Squares 8](#_Toc34065054)

[3.1.1 isLegalMagicSquare() 8](#_Toc34065055)

[3.1.2 generateMagicSquare() 9](#_Toc34065056)

[3.2 Turtle Graphics 9](#_Toc34065057)

[3.2.1 Problem 1: Clone and import 10](#_Toc34065058)

[3.2.2 Problem 3: Turtle graphics and drawSquare 10](#_Toc34065059)

[3.2.3 Problem 5: Drawing polygons 11](#_Toc34065060)

[3.2.4 Problem 6: Calculating Bearings 12](#_Toc34065061)

[3.2.5 Problem 7: Convex Hulls 13](#_Toc34065062)

[3.2.5.1 凸包问题 13](#_Toc34065063)

[3.2.5.2 算法描述 13](#_Toc34065064)

[3.2.5.3 JUnit测试结果 14](#_Toc34065065)

[3.2.6 Problem 8: Personal art 14](#_Toc34065066)

[3.2.6.1 多彩螺旋线 14](#_Toc34065067)

[3.2.6.2 代码 14](#_Toc34065068)

[3.2.6.3 效果 15](#_Toc34065069)

[3.2.7 Submitting 15](#_Toc34065070)

[3.2.7.1 初始化git 15](#_Toc34065071)

[3.2.7.2 添加远程仓库URL 16](#_Toc34065072)

[3.2.7.3 添加上传文件 16](#_Toc34065073)

[3.2.7.4 添加修改日志 17](#_Toc34065074)

[3.2.7.5 上传push 17](#_Toc34065075)

[3.2.7.6 查看上传情况 18](#_Toc34065076)

[3.2.7.7 下载/同步 18](#_Toc34065077)

[3.3 Social Network 18](#_Toc34065078)

[3.3.1 设计/实现FriendshipGraph类 18](#_Toc34065079)

[3.3.1.1 邻接表存储结构 19](#_Toc34065080)

[3.3.1.2 Class Node 20](#_Toc34065081)

[3.3.1.3 Method addVertex 21](#_Toc34065082)

[3.3.1.4 Method addEdge 21](#_Toc34065083)

[3.3.1.5 Method getDistance 21](#_Toc34065084)

[3.3.2 设计/实现Person类 21](#_Toc34065085)

[3.3.3 设计/实现客户端代码main() 22](#_Toc34065086)

[3.3.4 设计/实现测试用例 22](#_Toc34065087)

[4 实验进度记录 23](#_Toc34065088)

[5 实验过程中遇到的困难与解决途径 24](#_Toc34065089)

[6 实验过程中收获的经验、教训、感想 26](#_Toc34065090)

[6.1 实验过程中收获的经验和教训 26](#_Toc34065091)

[6.2 针对以下方面的感受 26](#_Toc34065092)

# 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

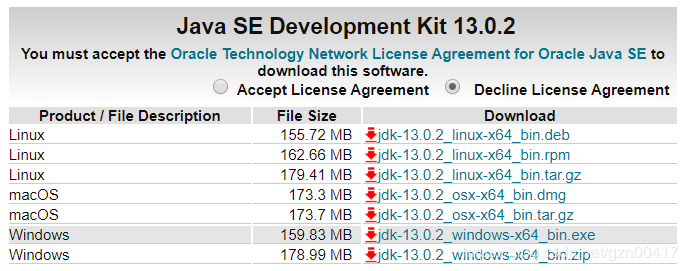
# 实验环境配置

## Java & Eclipse & Maven 使用配置方法

### 下载Eclipse

* Download 64 bit
* Select Another Mirror
* China - University of Science and Technology of China
* 作者选择中科大USTC镜像

### 下载JDK13

* 点击 Accept License Agreement 后，点击灰条选项  
  

### 配置JDK

* 点击 : Window -> Prefernces -> Java -> Installed JREs -> Add -> Standard VM -> 选择 JDK 目录 D:\Program Files\jdk-13.0.2 -> Finish -> 在列表勾选刚刚添加的 JDK -> Apply

### 编辑器字体修改

* 点击：Window -> Prefernces -> General -> Appearance -> Colors and Fonts -> Java -> Java Editor Text Font
* 推荐：Consolas+小四号字体

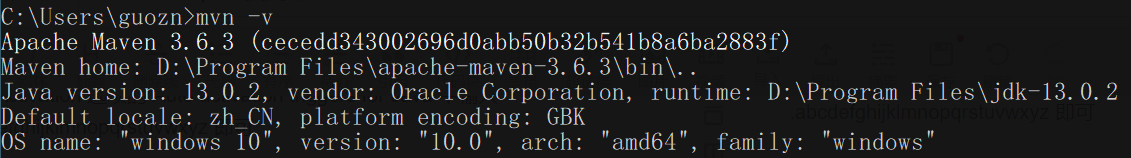
### 修改默认编码

* 点击 : Windows -> Preferences -> General -> Workspace , 在右侧的 Text file encoding 中选择 UTF-8
* 点击 : Windows -> Preferences -> General -> Content Types , 在右侧点开 Text , 选择 Java Source File , Java Properties File , JSP , 在下面的 Default encoding 输入 UTF-8 , 点击 Update
* 点击 : Windows -> Preferences -> Web -> JSP Files , 面板选择 ISO 10646/Unicode(UTF-8) 右键选择项目 -> Properties -> Resource -> 设置编码为 UTF-8

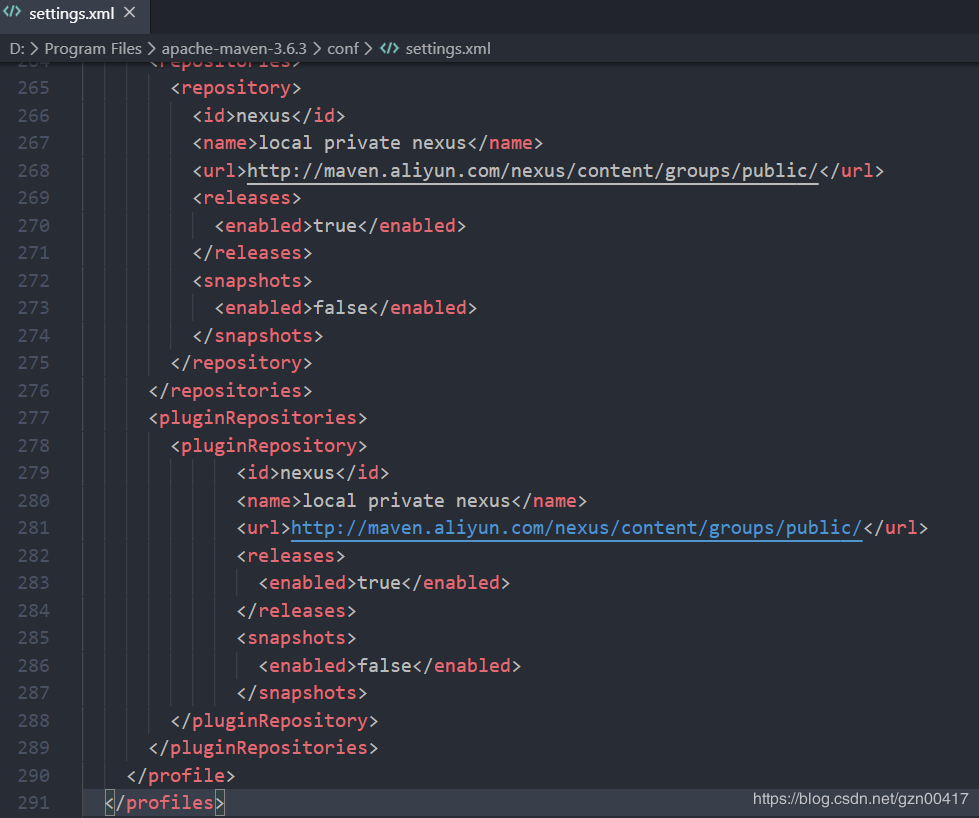
### 设置代码自动提示

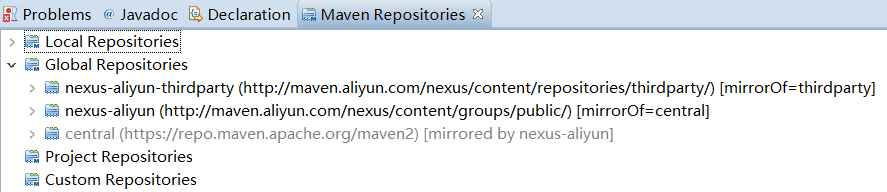
* 点击 Window -> Preferences -> Java -> Editor -> Content Assist , 在右侧的 Auto Activation 选项的 Auto activation triggers for Java 中输入 .abcdefghijklmnopqrstuvwxyz 即可

### 配置 Maven

* 首先配置系统环境变量 :
* 右键计算机 -> 属性 -> 高级系统设置 -> 高级 -> 环境变量 , 在下方的 系统变量 添加 MAVEN*HOME , 值是 Maven 文件夹路径 , 如 : C:\MyProgram\Maven , 修改 Path 变量 , 添加 %MAVEN*HOME%\bin; , 注意分号 ; 配置完成之后查看是否成功 : 运行 CMD 输入 mvn -v  
  
* 然后配置 Eclipse :
* 打开 Eclipse -> Window -> Preferences -> Maven -> User Settings -> Global Settings: 和 User Settings: -> Browse... , 选择 Maven 目录下的 \conf\settings.xml 文件 , 然后点击 Update Settings 按钮即可
* 添加镜像（国内阿里云镜像）
* conf/setting.xml中插入代码
* 在mirrors中间插入

<mirror>  
 <id>nexus-aliyun</id>  
 <mirrorOf>central</mirrorOf>  
 <name>Nexus aliyun</name>  
 <url>http://maven.aliyun.com/nexus/content/groups/public/</url>  
 </mirror>  
 <mirror>  
 <id>nexus-aliyun-thirdparty</id>  
 <mirrorOf>thirdparty</mirrorOf>  
 <name>Nexus aliyun thirdparty</name>  
 <url>http://maven.aliyun.com/nexus/content/repositories/thirdparty/</url>  
 </mirror>  
 - 在profiles中间插入

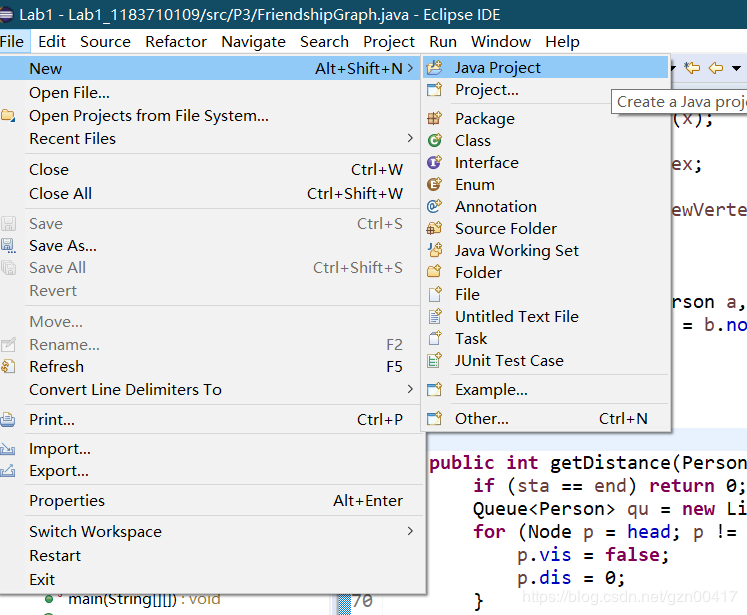


* 查看是否成功
  + Window -> Show View -> Other -> Maven -> Maven Repositories -> Open
  + 若打开后更改，可以点击Maven Repositories查看视图右边的Reload setting.xml  
    

## JUnit

* JUnit is a simple framework to write repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks.
* 使用 JUnit 为 Java 程序编写测试代码并执行测试

### 新建Java项目

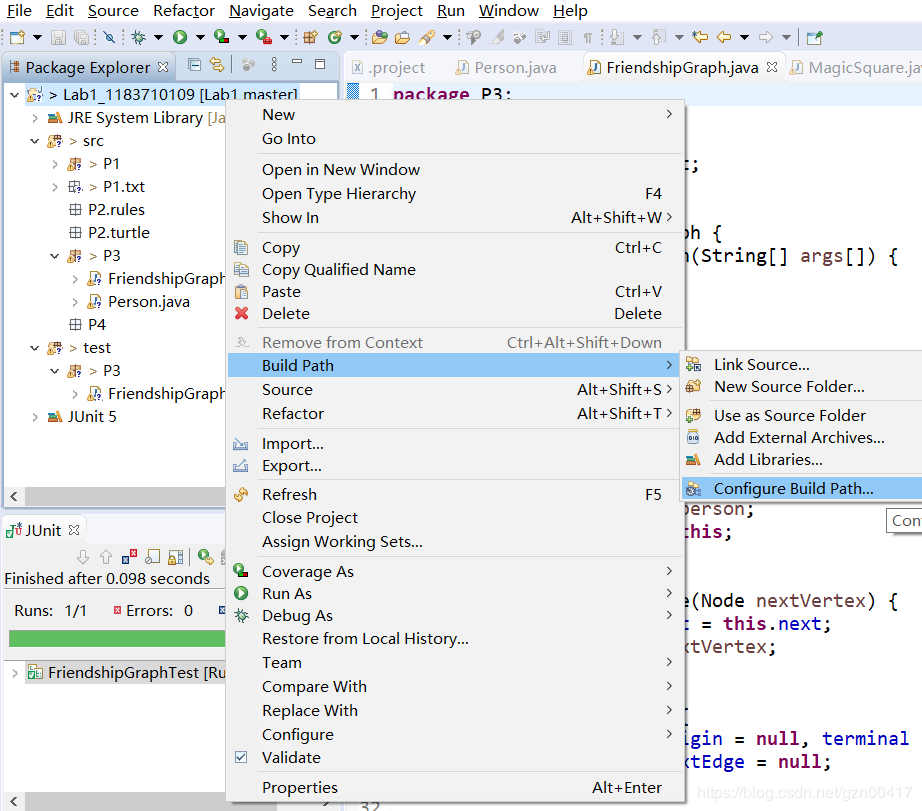
* File -> New -> Java Project
* 根据自己的需求配置  
  

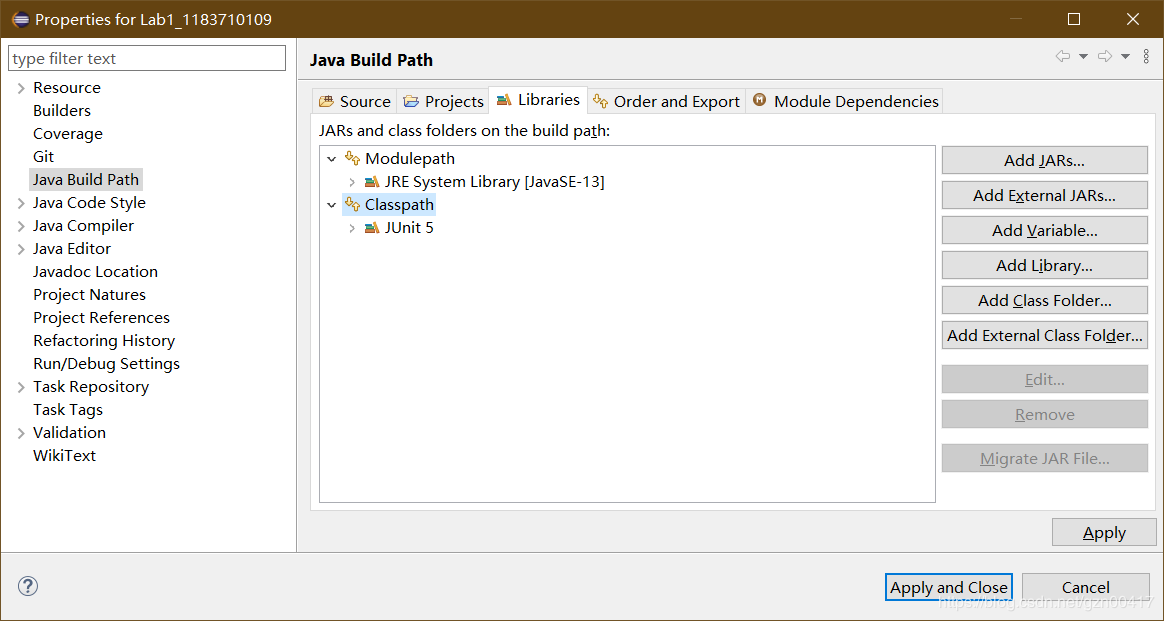
### 编写代码 & 打包

* 准备好src文件夹和test文件夹
* src文件夹中
* package名与test中相同（必须）

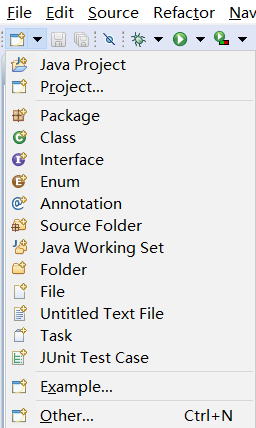
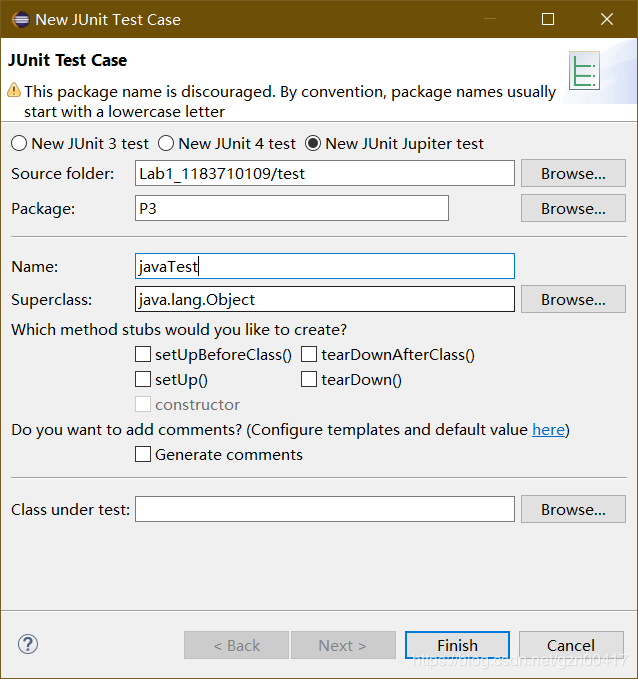
### 将JUnit加入Library

* 项目名字（右键）-> Build Path -> Configure Build Path



* Library（中间上面） -> classpath（左边） -> Add Libraries（右边）
* 

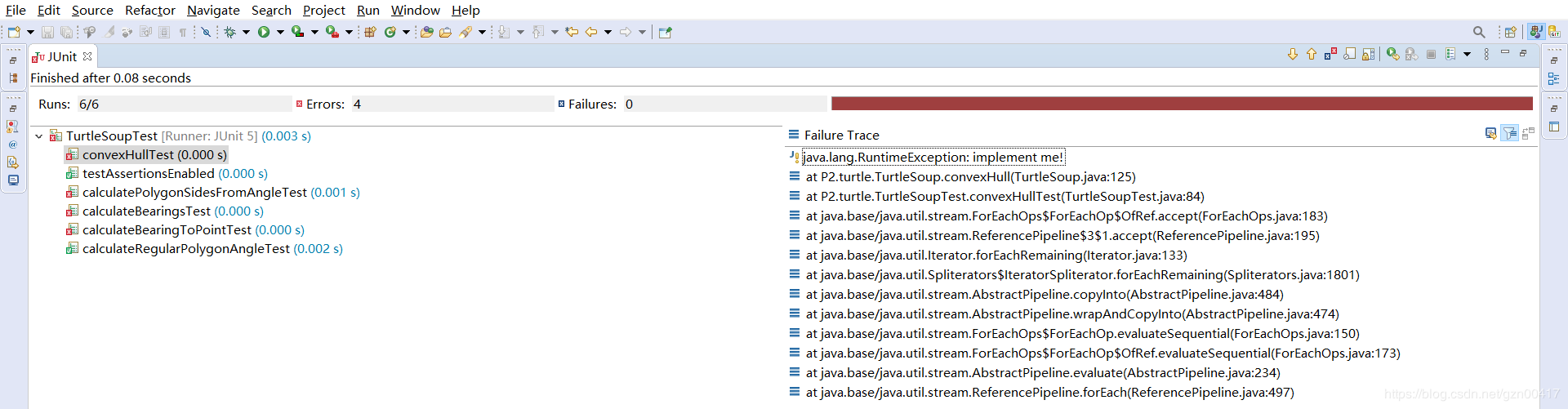
### 创建Junit Test Case

* 和创建project类似的方法创建JUnit Test Case  
  
* 弹框选项
  + New Junit Jupiter Test
  + Source folder：./.../test
  + package P3（与之前写好的类的package一样）  
    
* 创建好的示例：

package P3;  
import static org.junit.jupiter.api.Assertions.\*;  
import org.junit.jupiter.api.Test;  
class Test1 {  
 @Test  
 void test() {  
 assertEquals(2.0, Calc(1.0, 1.0), 0);  
 fail("Not yet implemented");  
 }  
}

* 更改void test()中的内容即可！

### 查看测试结果

* 双击JUnit视图放大查看
* 每一个@Test为一组测试
* 每组测试会显示测试结果，正确为绿色，错误为红色，还有其他错误提示  
  

# 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（但无需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

幻方是一个有n\*n个不同数字、且每行、每列和斜线上都有相同的和的方形结构。要求写出程序判断输入一个矩阵是否是幻方，并且构造幻方。

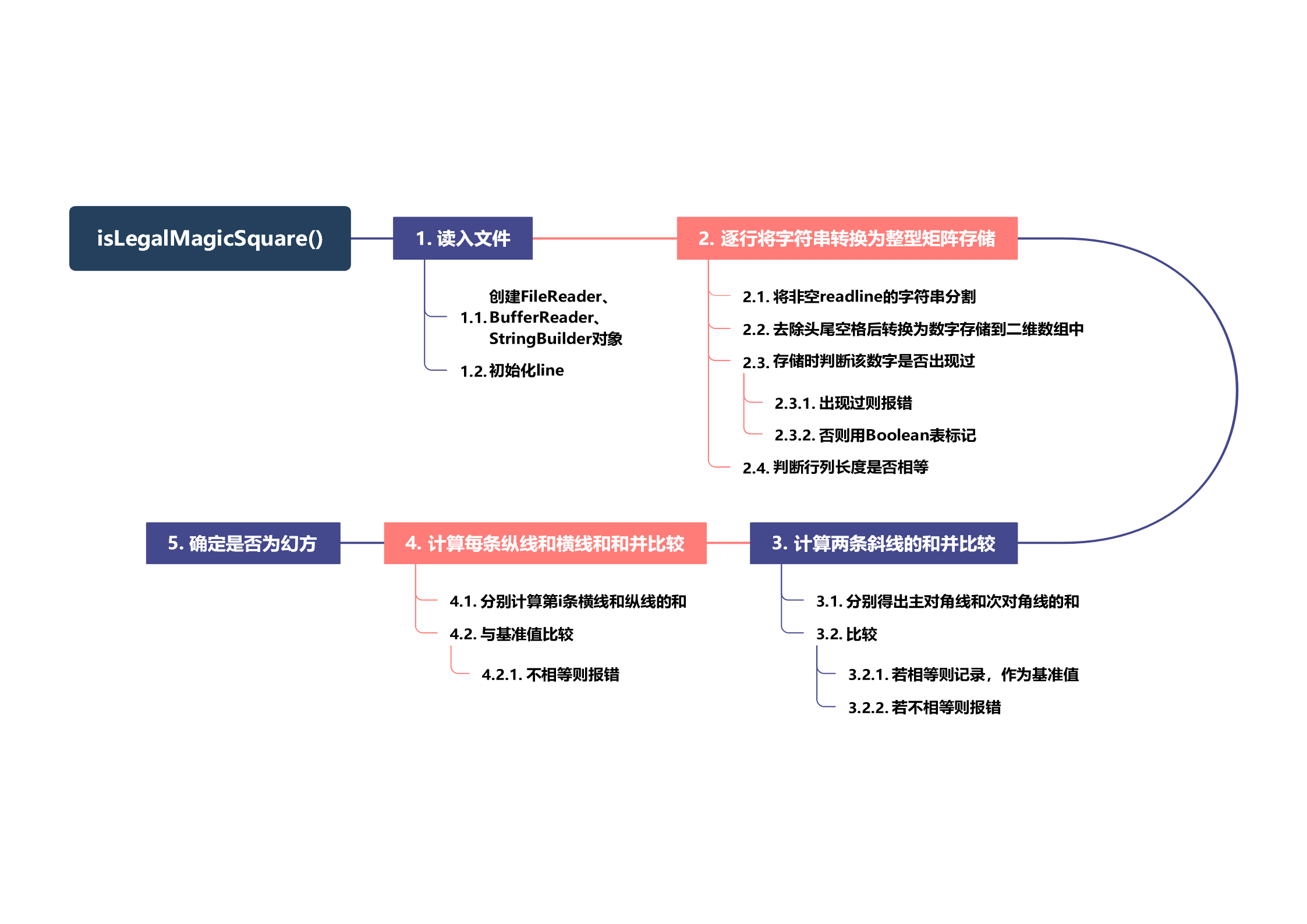
Main函数有两个部分，分别是：读取5个矩阵并判断、生成一个矩阵判断后输出到文件中。在读取时，用for循环分别将字符1到5拼入地址中，输出同理。其中在生成幻方之前，判断n的合法性。



### isLegalMagicSquare()

该函数要实现判断一个矩阵是否为幻方。

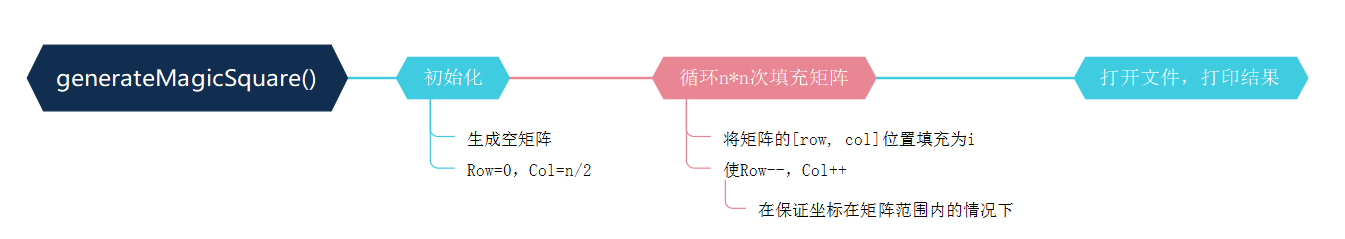
1. 读入文件
   1. 创建FileReader、BufferReader、StringBuilder对象
   2. 初始化line
2. 逐行将字符串转换为整型矩阵存储
   1. 将非空readline的字符串分割
   2. 去除头尾空格后转换为数字存储到二维数组中
   3. 存储时判断该数字是否出现过
      1. 出现过则报错
      2. 否则用Boolean表标记
   4. 判断行列长度是否相等
3. 计算两条斜线的和并比较
   1. 分别得出主对角线和次对角线的和
   2. 比较
      1. 若相等则记录，作为基准值
      2. 若不相等则报错
4. 计算每条纵线和横线和和并比较
   1. 分别计算第i条横线和纵线的和
   2. 与基准值比较
      1. 不相等则报错
5. 确定是否为幻方



### generateMagicSquare()

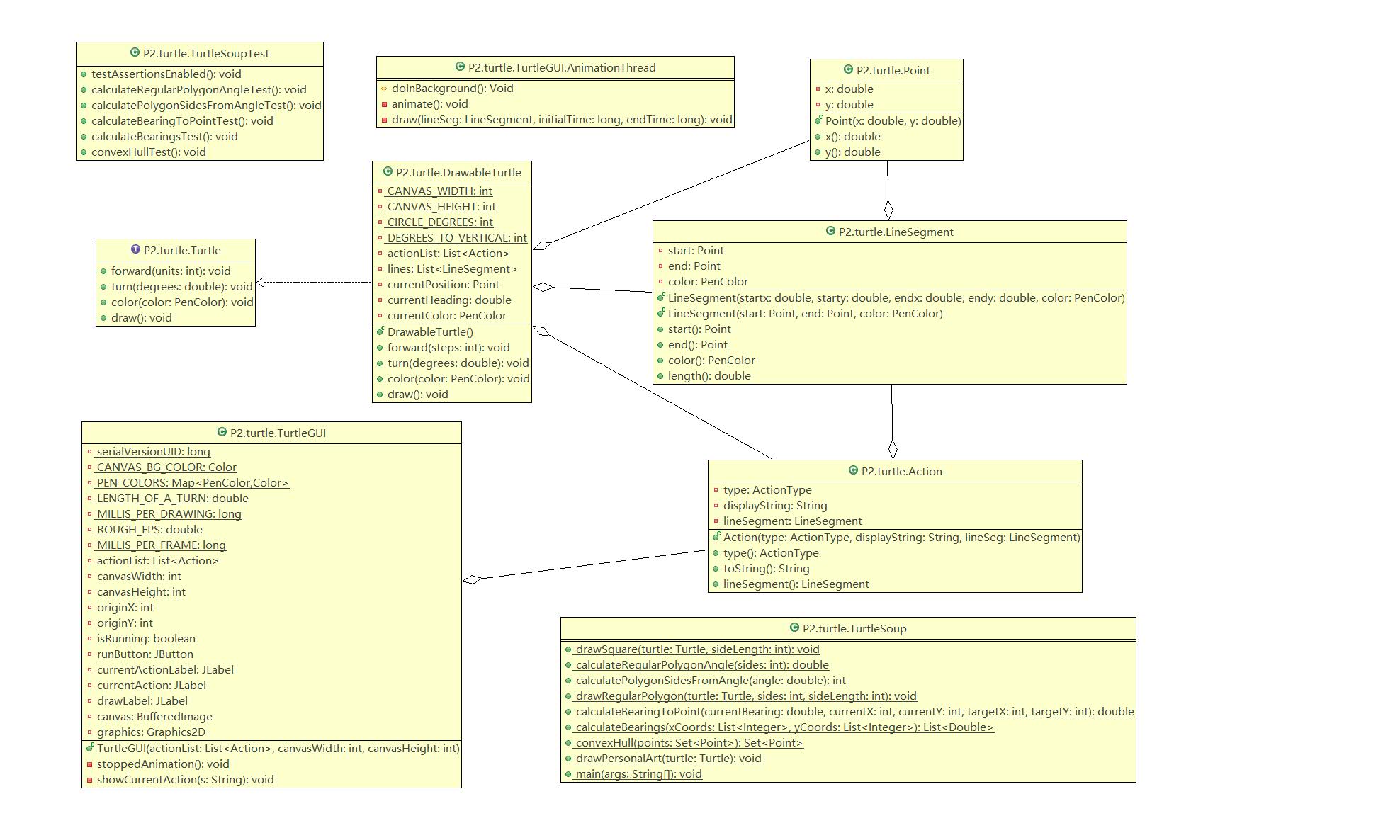
该函数要实现生成一个边长为奇数的幻方。

1. 初始化
   * 1. 生成空矩阵
     2. Row=0，Col=n/2
2. 循环n\*n次填充矩阵
   * 1. 将矩阵的[row, col]位置填充为i
     2. 在保证坐标在矩阵范围内的情况下使Row--，Col++
3. 打开文件，打印结果

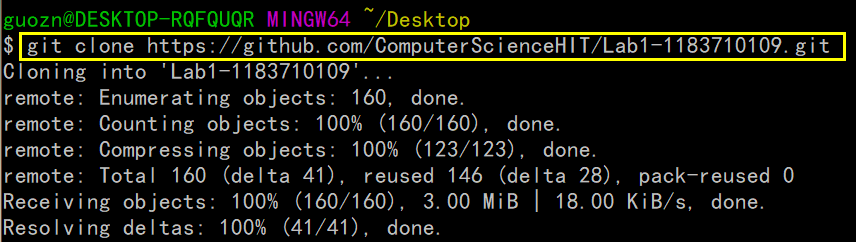


## Turtle Graphics

该任务需要我们clone已有的程序后，利用turtle按照要求画图，其中需要利用几何知识设计一些函数简化编程，最后可以发挥想象力进行Personal Art。首先分析turtle的package组成，了解类成员。



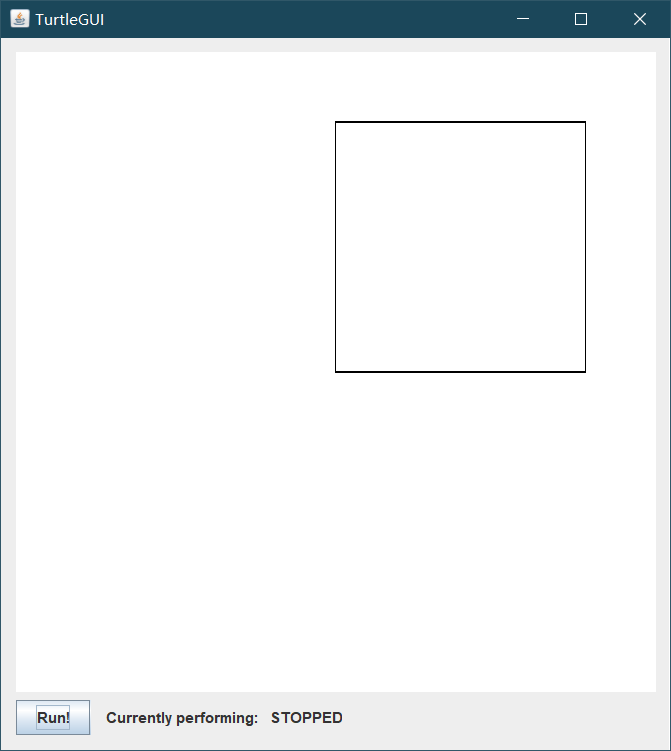
### Problem 1: Clone and import

* 打开目标存储文件夹
* 右键点击Git Bash
* 输入git clone https://github.com/ComputerScienceHIT/Lab1-1183710109.git
* 
* Done

### Problem 3: Turtle graphics and drawSquare

该函数需要实现：已知边长，画出边长为指定数值的正方形。参数是海龟对象turtle和编程sidelength。

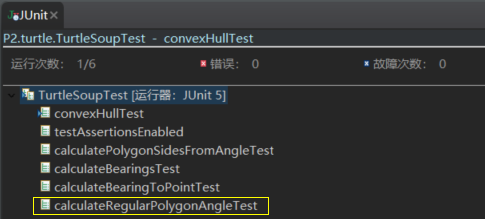
首先将海龟画笔设置为黑色。然后执行4次的前进sidelength长度、转完90度，即可完成一个边长为sidelength的正方形。下图是边长为200的正方形：



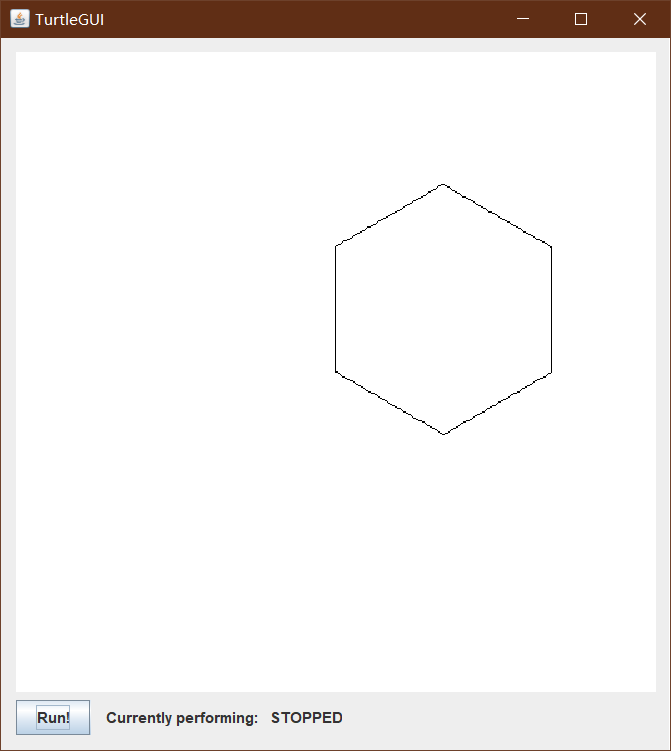
### Problem 5: Drawing polygons

1. 该问题首先希望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，通过运行中的Junit测试得：

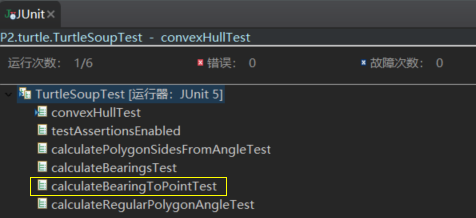


1. 该问题还希望已知正多变型得边数和边长画出一个正多边形。参照画正方形的方法，可以先前进sidelength，再使海龟旋转一个角度，执行“边数”次。其中，这个角度是正多边形内角的补角，利用的功能计算出多边形内角，再用180°减去这个值即可。边长为100的正六边形效果如下：

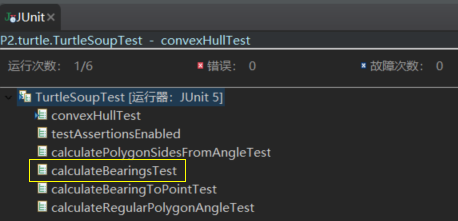


### Problem 6: Calculating Bearings

1. 该问题首先希望解决，已知起点和当前朝向角度，想知道到终点需要转动的角度。例如，如果海龟在（0，1）朝向 30 度，并且必须到达（0，0）它必须再转动 150 度。
   1. 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；
   2. 然后取相反数（海龟旋转的方向是顺时针，坐标轴角度的旋转角度的逆时针）；
   3. 再减去90°（海龟的0°线是向上，坐标轴的0°线是向右，向右到向上要逆时针旋转90°）；
   4. 最后调整为0-360°之间（可能大于360°或小于0°）。



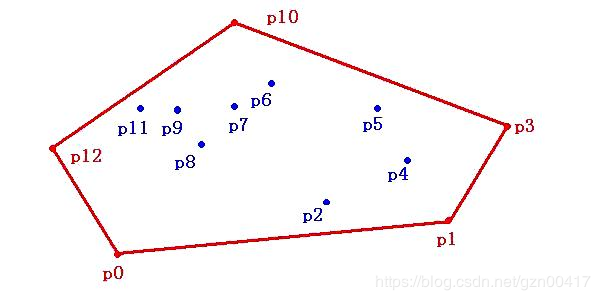
1. 基于上一个问题，此时有若干个点，想知道从第一个点开始到第二个点，再从第二个点到第三个点……以此类推每次转向的角度。
2. 将“起点”选为第一个点（坐标为）；
3. 循环n-1次（n为点的个数）
4. 每次将第i+1号点设置为“终点”，通过上一个函数计算旋转角度并存储到List中；
5. 将下一次的“起点”用当前“终点”更新，继续循环；
6. 退出循环后返回List。



### Problem 7: Convex Hulls

#### 凸包问题

给定平面上一堆点集，输出位于凸包上的点。如下所示：输入如下这么多点集，输入P0，P1，P3，P10，P12。



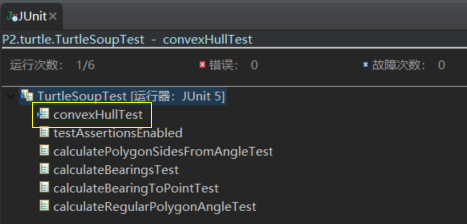
#### 算法描述

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……

#### JUnit测试结果



### Problem 8: Personal art

#### 多彩螺旋线

思路：在画正多边形的基础上，步长不是一直相同，而是越来越长，并且角度比画正多边形需要的角度多一些，每次拐弯变换颜色。

* Size是螺旋的大小
* Step的每一步的长度，每走一步拐弯一次
* Densi是密度，角度越小，螺旋越密
* ColorNum是色彩的数量，更改时要在switch里更改

#### 代码

    public static void drawPersonalArt(Turtle turtle) {

        int Size = 400, Step = 1, Densi = 1, ColorNum = 5;

        for (int i = 1; i <= Size; i++) {

            switch (i % ColorNum) {

                case 0:

                    turtle.color(PenColor.BLUE);

                    break;

                case 1:

                    turtle.color(PenColor.GREEN);

                    break;

                case 2:

                    turtle.color(PenColor.YELLOW);

                    break;

                case 3:

                    turtle.color(PenColor.RED);

                    break;

                case 4:

                    turtle.color(PenColor.MAGENTA);

                    break;

                case 5:

                    turtle.color(PenColor.ORANGE);

                    break;

            }

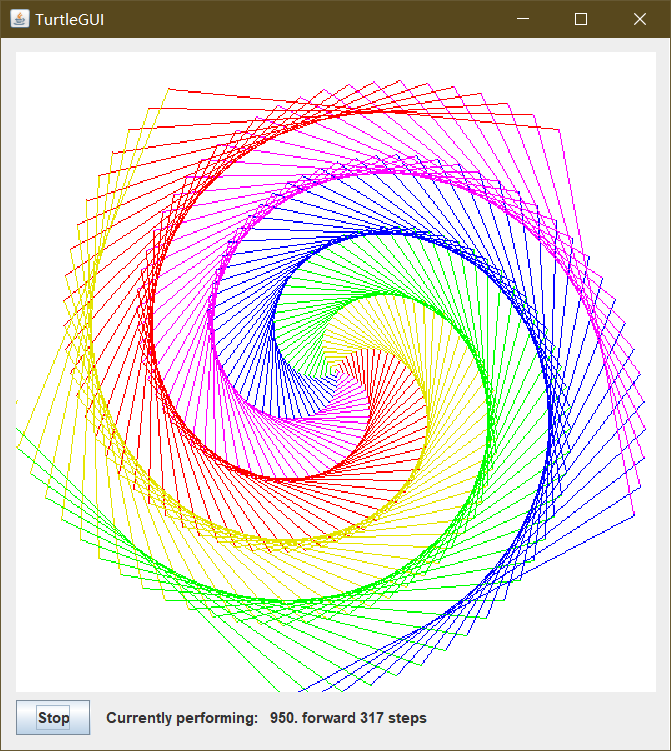
            turtle.forward(Step \* i);

            turtle.turn(360 / ColorNum + Densi);

        }

    }

#### 效果



### Submitting

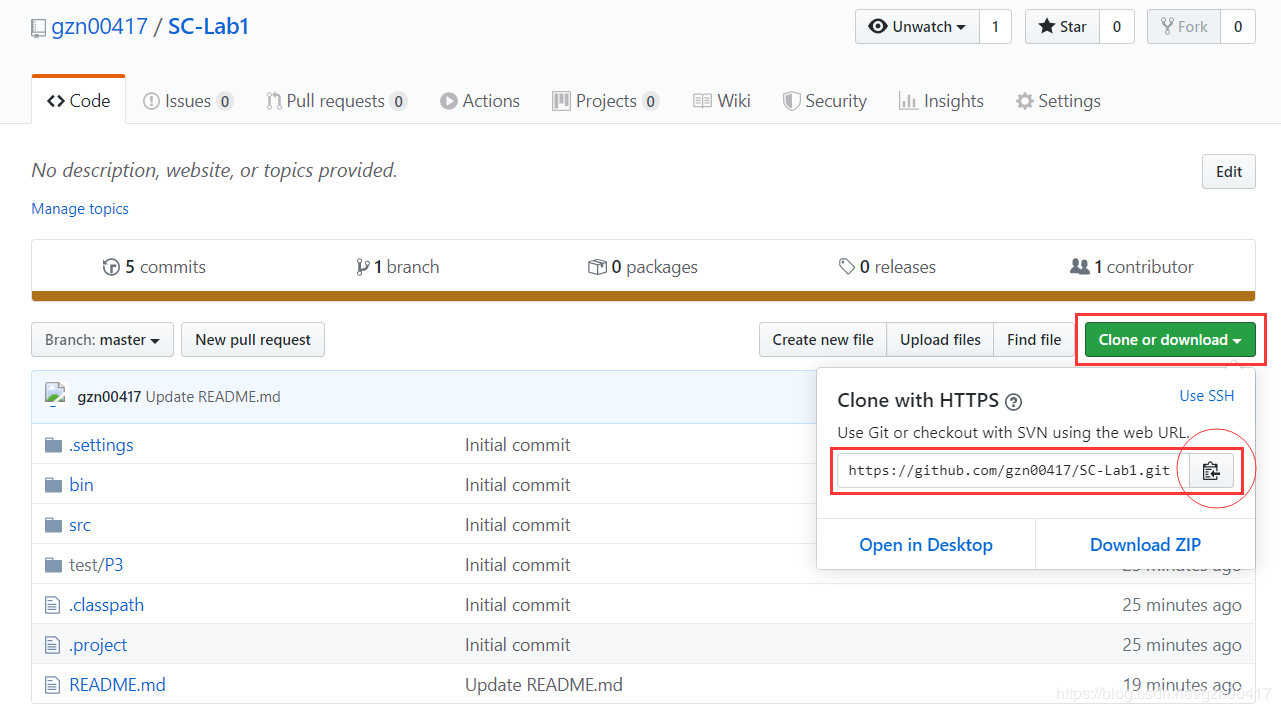
通过git提交项目

#### 初始化git

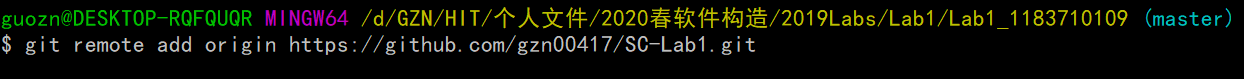
打开 Git Bash

输入cd /d/.../

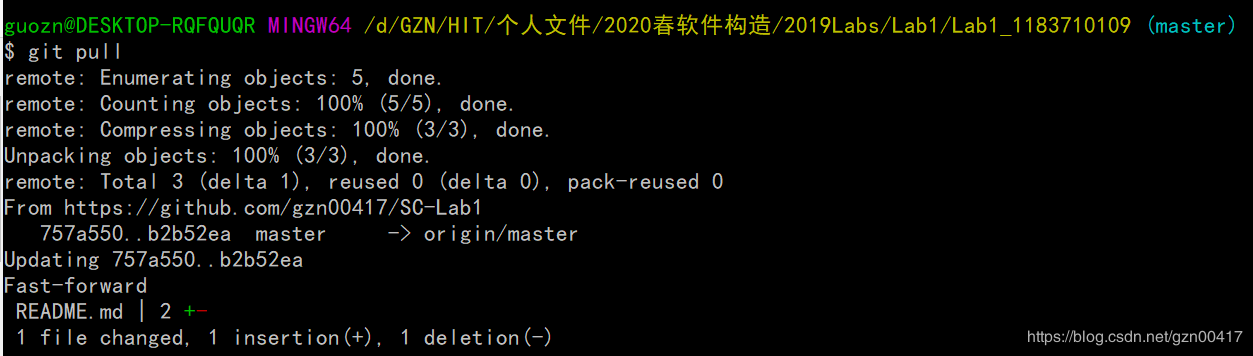
#### 添加远程仓库URL

* 查找URL，图示位置（绿色按键-->复制URL）  
  

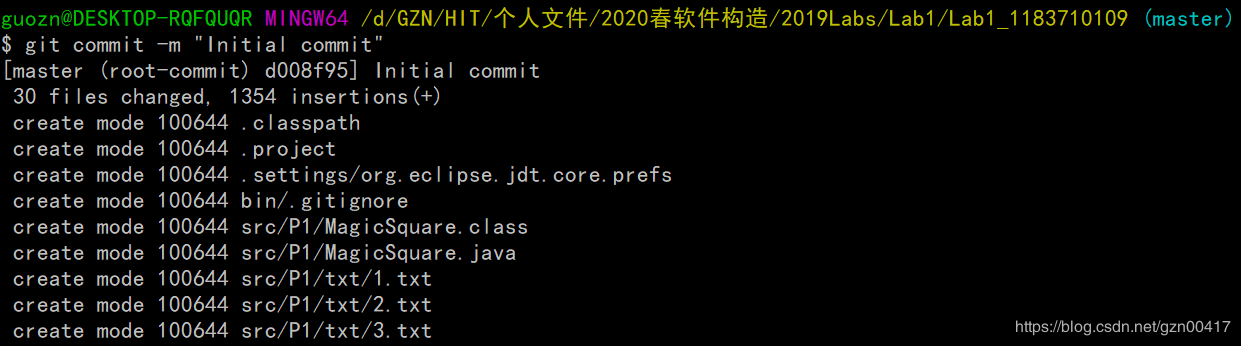
添加远程仓库链接，命令：git remote add origin https://github.com/.../x...x.git

最后的链接就是复制来的URL  


#### 添加上传文件

* 注意：若新建仓库有README.md，需要先在本地同步（下载）
* 命令：git pull
* 若无效果则尝试：git pull --rebase origin master  
  
* 命令：git add .（后面是前后有空格的点，表示所有文件）
* 命令：git status可以查看状态  
  

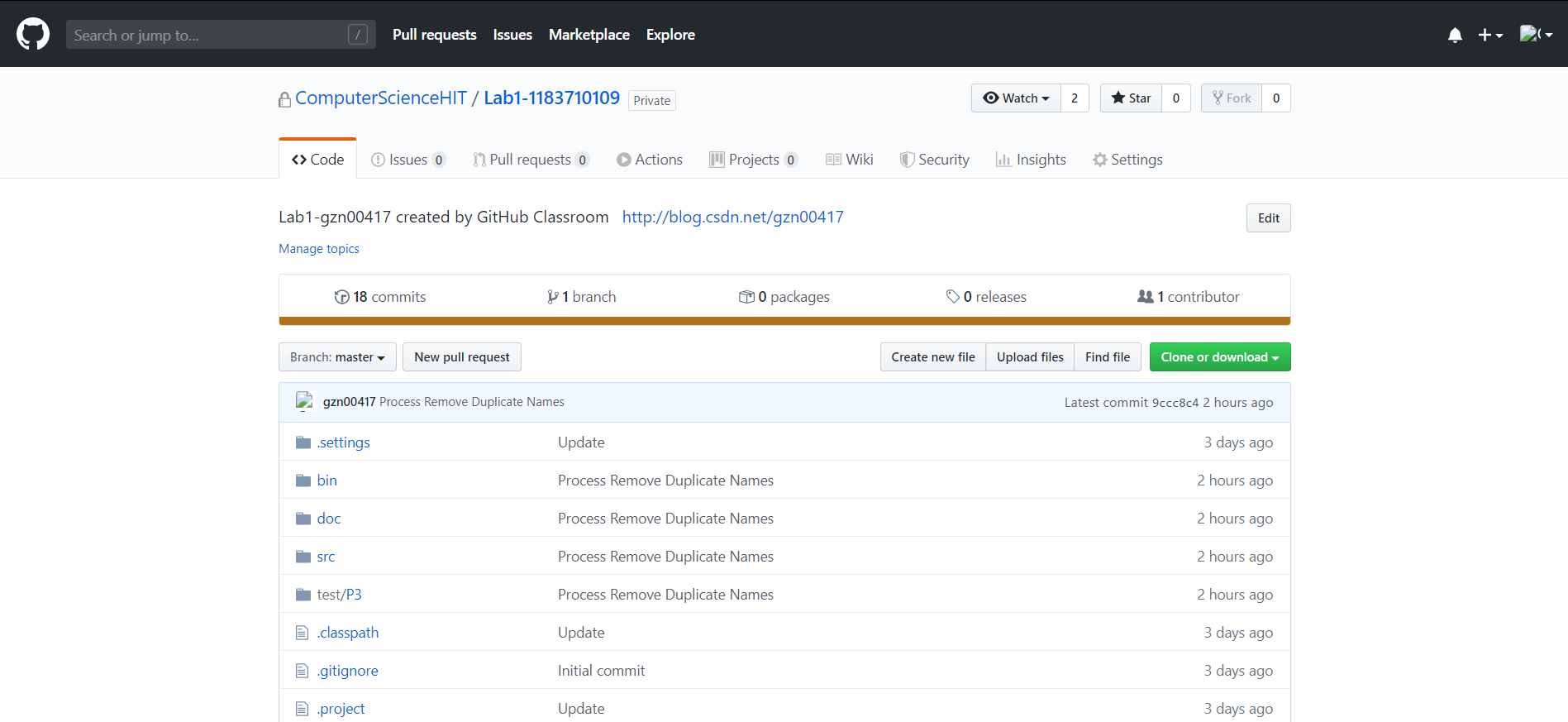
#### 添加修改日志

* 命令：git commit -m "Initial commit"
* 引号内内容可以随意修改；
* 引号内的内容只会称为有修改过的文件的新日志；
* 命令（在-m前加入-a）：git commit -a -m "Update …"
  + 任何是否被git add的文件都将被commit  
    

#### 上传push

* 命令：git push -u origin master
* 多次使用后可以省略后面参数，只用git push

#### 查看上传情况

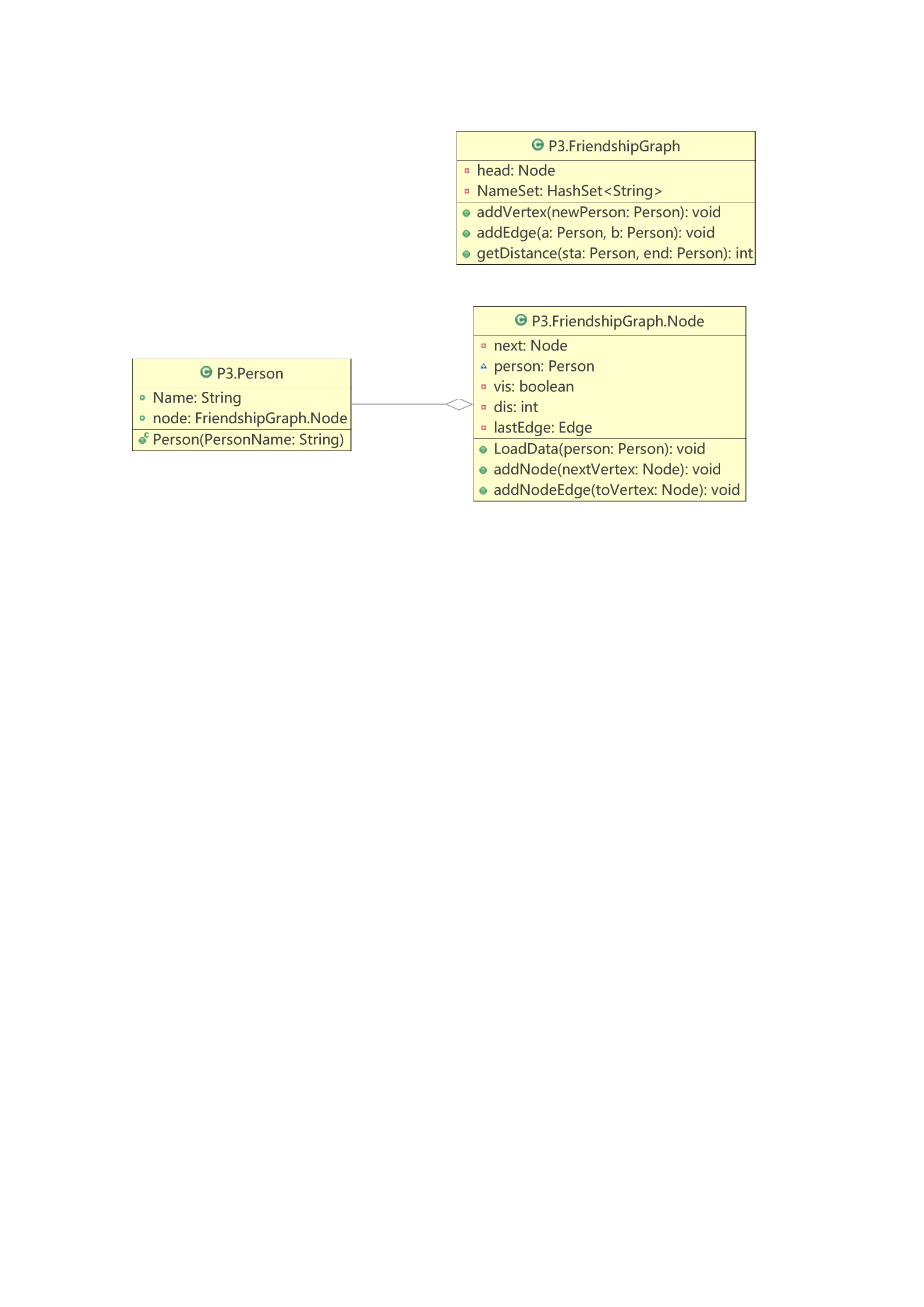


#### 下载/同步

* 命令：git pull 或 git pull --rebase origin master

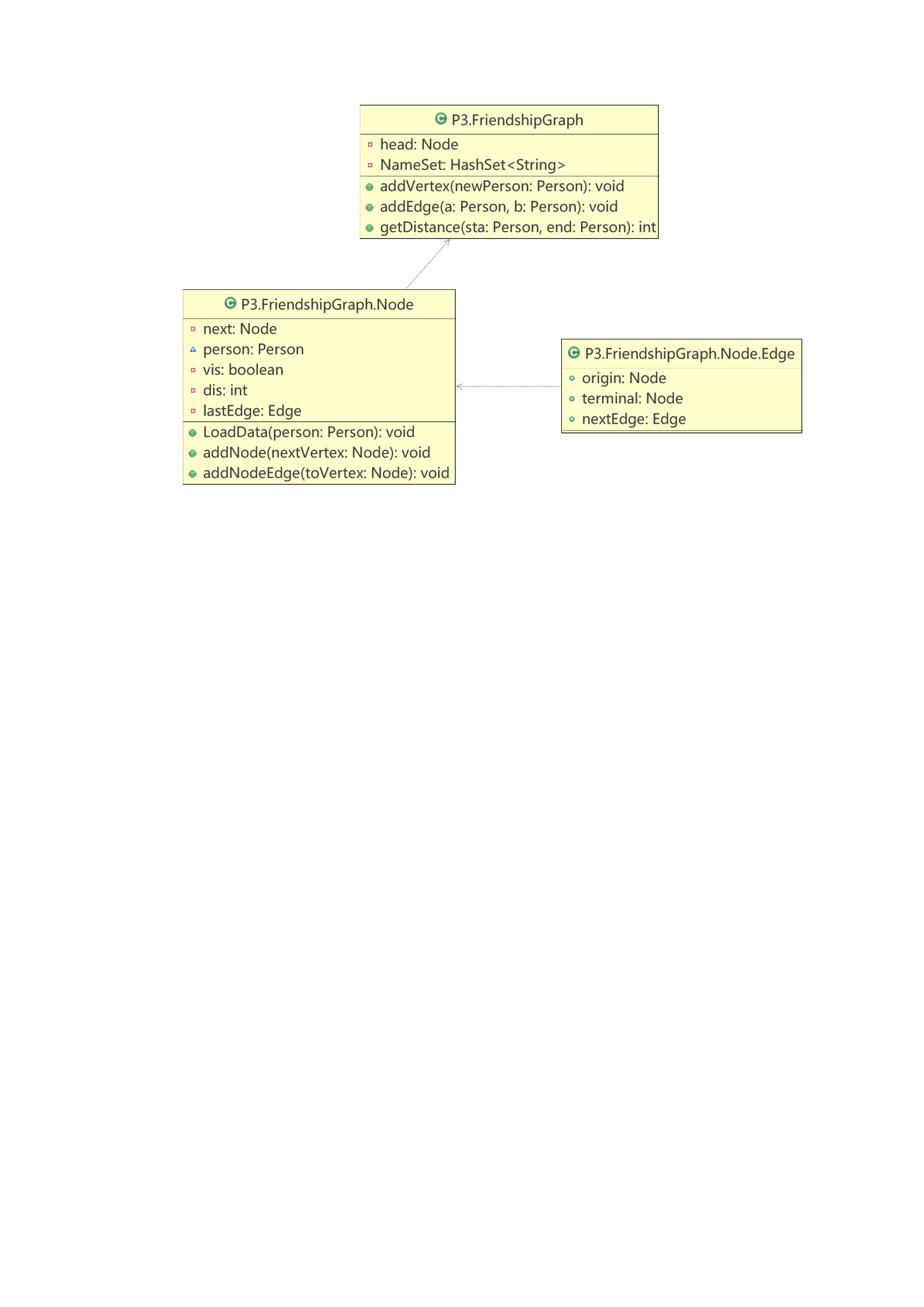
## Social Network

该任务要求设计一张社交网络图，基于连接人与人，并且能计算任意两人之间的联系情况。网络图基于两个类，分别是FriendshipGraph类和Person类。



### 设计/实现FriendshipGraph类

该类的实际意义是一张社交网络图，包括了代表每个Person的点、代表每两个Person之间联系的边、以及建立点和联系和计算距离的方法。



#### 邻接表存储结构

在存储社交网络时，我使用了邻接表。所有的Node被连在一起，方便查找，并补充了一个head变量用来标记首个Node。假定一个社交网络为



则该图转换为邻接表的示意图为：



#### Class Node

Node类要实现的是将一个Person转换为邻接表里的点，所以一个Node有邻接表中点的重要成员变量：下一个Node为next，对应的Person对象person，直接连接的边lastEdge，以及实现邻接表的相关方法。此外，为了能实现方法getDistance，我另外增设了vis和dis两个变量用来记录是否访问过以及与当前起点的最近距离。



* ***Class Node.Edge***

该类是邻接表中的边，每个Edge对象存储了邻接表中的下一条边，以及对应的边的两个Person所对应的Node。

* ***Method Node.LoadData***

该方法将Person对象导入到Node中进行存储，需要的时候可以直接调用。

* ***Method Node.addNode***

该方法将相应的Node加入到Node的链表中（即邻接表图中的纵向链表）。

* ***Method Node.addNodeEdge***

该方法将新的边加入到对应的Node中，更新每个Node后的Edge链表。

#### Method addVertex()

该方法目标是在社交网络图中增加一个新的节点，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。



#### Method addEdge()

该方法目标是将两个Person之间进行联系，在邻接表中，用有向边来代表“有社交关系”，由于题目设定是社交默认为双向，则需要在函数中两次调用Node中的addNodeEdge方法加两个方向的边。考虑到可扩展性和可复用性，程序考虑到了“单向社交的情况”，仅需将双向加边中的“B->A”删除即可。



#### Method getDistance()

该方法要计算任意两个Person之间的“距离”，若没有任何社交关系则输出“-1”。两个Person之间计算使用BFS，默认边权为1，则在搜索到边时加1即可，搜索到目标点退出；特殊情况根据要求输出0或-1。



### 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。此外，在我的设计中，为了方便起见，我将每个Person对象在FriendshipGraph中对应的Node存储在对应Person的成员变量中。



### 设计/实现客户端代码main()

给出你的设计和实现思路/过程/结果。

### 设计/实现测试用例

给出你的设计和实现思路/过程/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 |
| 2020-02-03 | 下午 | Magicsquare | 按计划完成 |
| 2020-02-04 | 下午 | 建立本地git仓库并链接 | 遇到困难，但按计划完成 |
| 2020-02-04 | 晚上 | 配置、生成项目 | 按计划完成 |
| 2020-02-05 | 早上 | Turtle问题第二个小问 | 按计划完成 |
| 2020-02-05 | 下午 | Socialnetwork的两个类 | 按计划完成 |
| 2020-02-06 | 晚上 | 解决3.2.5 | 按计划完成 |
| 2020-02-08 | 晚上 | TurtleSoupTest调试 | 按计划完成，时间较长 |
| 2020-02-24 | 下午 | 迁移项目，调整工作目录和路径 | 完成 |
| 2020-02-26 | 上午 | 写报告，配置环境部分和问题1概述 | 完成 |
| 2020-02-26 | 下午 | 完成Personal Draw | 完成 |
| 2020-02-27 | 中午 | 解决Social Network重名冲突问题 | 克服困难完成 |
| 2020-02-27 | 下午 | 写报告关于Git Clone/Push部分 | 完成 |
| 2020-02-28 | 上午 | 写报告关于Turtle函数实现部分 | 完成 |
| 2020-02-29 | 早晨 | 解决word标题编号奇葩问题 | 完成 |
| 2020-02-29 | 上午 | Problem1两个函数设计流程图 | 完成 |
| 2020-02-29 | 下午 | Problem3函数设计思路 | 完成 |
| 2020-03-01 | 上午 | 解决AmaterasUML安装问题 | 遇到困难，解决 |
| 2020-03-01 | 下午 | 画UML图 | 完成 |
| 2020-03-02 | 下午 | FriendshipGraph详细思路和流程图 | 完成 |
|  |  |  | 遇到困难，未完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的困难 | 解决途径 |
| Java使用Eclipse出现editor does not contain a main type | 原因：当前源代码没有整合进package，即没有build path  部分已经build path，但新写的代码没有  解决方案  若已经有build path（有红褐色标记），则先Remove；没有（全白）则忽略  src（右键）-> build path -> remove from build path  此时，src文件全部变白  然后点击：src（右键）-> build path -> use as source folder |
| Exception in thread "main" java.lang.Error: Unresolved compilation problem，编译器/package不一致 | 检查编译器  打开cmd检查版本  项目名（右键）-> propriety -> Java Compiler  Enable…打勾，Compiler…level 与java和javac一致  可以把Use…勾去掉进行修改  若无误，则非编译器问题  发现package名字有红线\*\*（重要特征）\*\*  发现在build path之后不一致，统一package完成build path请见  解决  修改代码中package名字\*\*（每一个包中的代码）\*\*  再次运行check  注意！！还需要更改同一个package下所有代码的部分import（旧package名） |
| Exception in thread "main" java.io.FileNotFoundException: xxx.xxx (系统找不到指定的路径。) | 该情况为读取文件路径错误  src里的代码，在我们的直观感觉可以用相对路径./xxx  然而有时候行，有时候不行  作者确实都经历过  因此要加上以src为起始的路径  例如代码在src/P1里，读取的文件在src/P1/a里  写./a/xxx要碰运气  写src/P1/a/xxx才是最稳的 |

# 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

## 针对以下方面的感受

1. Java编程语言是否对你的口味？
   1. 符合，Java的OOP非常顺手，又相近于之前熟悉的C++
2. 关于Eclipse IDE；
   1. 不够美观
   2. 经过一定装饰后还是不够美观
   3. 调试、Junit都很好用
   4. 相对vscode太丑
3. 关于Git和GitHub；
   1. 能掌握Git基本功能
   2. GitHub真香
4. 关于CMU和MIT的作业；
   1. 语言问题，有些地方理解比较困难
5. 关于本实验的工作量、难度、deadline；
   1. 工作量非常大
   2. 难度很大
   3. Deadline适中
6. 关于初接触“软件构造”课程；
7. 疫情期间，只能远程授课，个人在家里完成实验任务，你对该学习方式有什么想法？