



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

# 2021 年春季学期 计算学部《软件构造》课程

## Lab 1 实验报告

姓名	沈城有
学号	1190200526
班号	
电子邮件	
手机号码	

## 目录

1 实验目标概述 .....	1
2 实验环境配置 .....	1
3 实验过程 .....	2
3.1 Magic Squares .....	2
3.1.1 isLegalMagicSquare() .....	2
3.1.2 generateMagicSquare() .....	3
3.2 Turtle Graphics .....	5
3.2.1 Problem 1: Clone and import .....	5
3.2.2 Problem 3: Turtle graphics and drawSquare .....	6
3.2.3 Problem 5: Drawing polygons .....	7
3.2.4 Problem 6: Calculating Bearings .....	8
3.2.5 Problem 7: Convex Hulls .....	9
3.2.6 Problem 8: Personal art .....	9
3.2.7 Submitting .....	10
3.3 Social Network .....	10
3.3.1 设计/实现 FriendshipGraph 类 .....	11
3.3.2 设计/实现 Person 类 .....	12
3.3.3 设计/实现客户端代码 main() .....	12
3.3.4 设计/实现测试用例 .....	13
4 实验进度记录 .....	13
5 实验过程中遇到的困难与解决途径 .....	14
6 实验过程中收获的经验、教训、感想 .....	14
6.1 实验过程中收获的经验教训 .....	14
6.2 针对以下方面的感受 .....	14

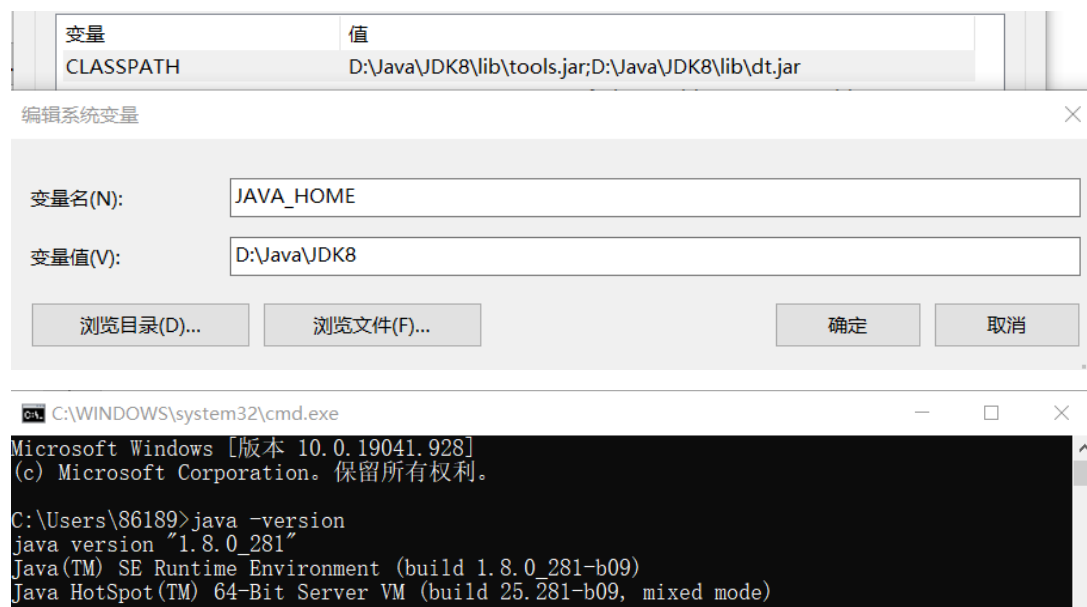
## 1 实验目标概述

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。

## 2 实验环境配置

寒假自学期间按照我校计算学部团委 QQ 公众号发布的《Java 设计预习第一弹:教你如何下载配置 Java 编译环境》进行了 Java 环境及 Eclipse 的安装及基本配置;从 Git 官网上下载了 Git 并简单学习了其使用方式;GitHub 使用之前注册的账号。

在配置 Java 环境结束后检查是否成功时,发现了环境变量错误问题,此问题导致在 cmd 中输入 `java -version` 报错。经检查为 `JAVA_HOME` 路径错误,修改后解决。下图为修改成功后的环境变量及 cmd 运行截图:



GitHub Lab1 仓库的 URL 地址:

<https://github.com/ComputerScienceHIT/HIT-Lab1-1190200526>

## 3 实验过程

### 3.1 Magic Squares

该任务要求我们首先了解什么是 Magic Square (幻方), 即按正方形排列的一组数, 其满足每行、列、对角线上的数的和均相等。任务主要分为两部分, 其一是编写 `isLegalMagicSquare()` 函数检查输入是否为 Magic Square, 二是理解并测试 `generateMagicSquare()` 函数。

#### 3.1.1 isLegalMagicSquare()

首先使用传入函数的文件路径创建一个 `FileReader` 对象, 利用此 `FileReader` 对象创建 `BufferedReader` 对象用于逐行读取文件内容, 如下图:

```
// 尝试读取文件, 检测文件是否存在
FileReader fr;
try {
    fr = new FileReader(fileName);
} catch (FileNotFoundException e) {
    System.out.println("Error:File not found!");
    e.printStackTrace();
    return false;
}
```

随后利用循环逐行读取文件中数据至二维数组, 过程与实验手册中提示的过程类似, 此处不再展示源代码。下图为读取过程中所声明的变量及类:

```
// 逐行读取文件, 整理数据
BufferedReader br = new BufferedReader(fr);
String line; // 暂存读取到的行
String[] proc_line; // 暂存一行按"\t"分割后的字符串数组
int line_cnt = 1; // 记录行数, 用于后续检测
int width = 0; // 记录一行宽度, 用于后续检测
int[][] square; // 保存读取到的Magic Square
```

在读取过程中及读取完成后要检查并处理一些特殊情况:

- a) 不符合定义情况 1 (行数大于列数);
- b) 不符合定义情况 2 (存在非正整数->输出 Error 提示, 或格式不正确输入->exception);
- c) 不符合定义情况 3 (并非矩阵);
- d) 不符合定义情况 4 (行数小于列数);
- e) 空文件情况。

以上情况中, 除 b) 以外, 均通过编程检测并输出对应提示信息, b) 中除检测到负整数外均利用异常输出相关的错误提示信息。遇到以上任何一种情况, 程序均会在输出错误提示信息后返回 `false`。

最后, 如果数据通过了以上检查, 则检查各行、列、对角线和是否相等。我

的思路是：首先单独计算一行的和，存入变量 `sum`，随后利用 `for` 循环逐行、列、对角线计算和并与 `sum` 比较，如果出现不相等则立即停止循环返回 `false`，否则返回 `true`。

按照以上设计思路及过程编写的代码运行结果如下图：

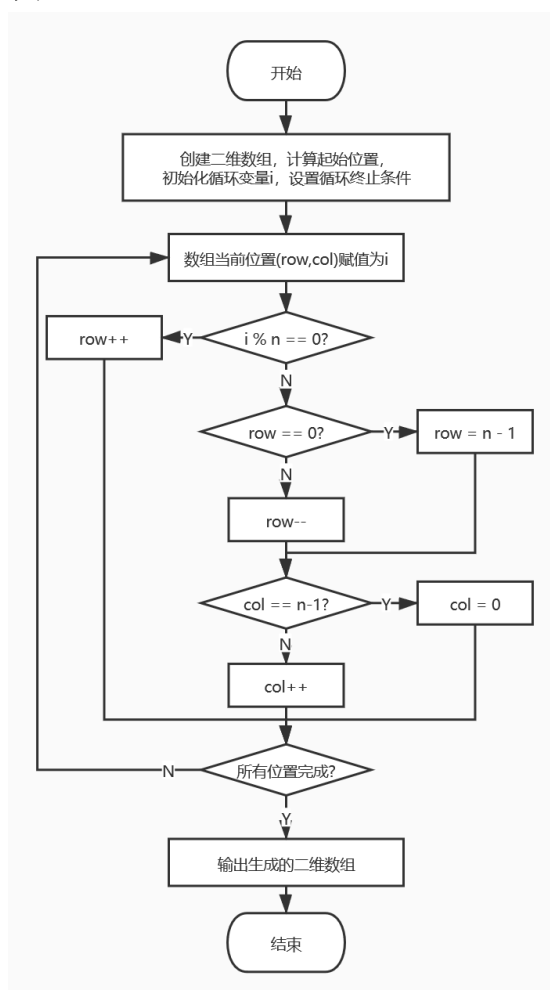
```

-----1.txt-----
true
-----2.txt-----
true
-----3.txt-----
Error:Input is not a matrix!
false
-----4.txt-----
Error:-35 is not a positive number!
false
-----5.txt-----
Error:Wrong input format or other errors!
java.lang.NumberFormatException: For input string: "12673 12796"
    at java.lang.NumberFormatException.forInputString(Unknown Source)
    at java.lang.Integer.parseInt(Unknown Source)
    at java.lang.Integer.valueOf(Unknown Source)
    at P1.MagicSquare.isLegalMagicSquare(MagicSquare.java:51)
    at P1.MagicSquare.main(MagicSquare.java:155)
false

```

### 3.1.2 generateMagicSquare()

下图为程序流程图：



函数使用奇数参数  $n$ ，首先计算初始位置  $(0, n/2)$ ，赋值为 1，之后每次取当前位置的右上的位置，同时考虑超出范围的处理：如果当前行是第一行，则下一行为最后一行，如果当前列是最右边的列，则下一次取左边第一列（超类似于循环的思想）设置的值每次加 1，如此重复  $n*n$  次，就完成了对整个矩阵的赋值，而且达到了每行、列及对角线之和均相同的效果。

如果输入的参数  $n$  为偶数，变量 `row` 的值会出现等于  $n$  的情况，造成数组越界访问，从而抛出 `ArrayIndexOutOfBoundsException` 异常，程序终止。

如果输入的参数  $n$  为负数，数组索引值会为负值，同样也是对数组的非法访问，从而抛出 `NegativeArraySizeException` 异常，程序终止。

此函数中文注释见源代码文件，此处不再展示。

函数扩展 1——写文件部分如下图（按格式逐个写入）：

```
// 写文件
File wfile = new File("src/P1/txt/6.txt");
try {
    wfile.createNewFile(); // 创建文件
    FileWriter writer = new FileWriter(wfile);
    BufferedWriter bwriter = new BufferedWriter(writer);
    // 按格式逐个写入
    for (int a = 0; a < n; ++a) {
        for (int b = 0; b < n; ++b) {
            bwriter.write(magic[a][b] + "\t");
        }
        bwriter.write("\n");
    }
    bwriter.flush(); // 缓冲区内内容写入
    bwriter.close();
} catch (IOException e) {
    e.printStackTrace();
    return false;
}
```

函数扩展 2——输入  $n$  不合法异常处理（主要分为  $n$  为负数、偶数等情况，我选择设置 try-catch 来处理，也可以直接检测退出避免异常）：

```
try {
    magic = new int[n][n];
} catch (NegativeArraySizeException e) {
    // 负数异常处理
    System.out.print("Error:Input n is negative!\n");
    e.printStackTrace();
    return false;
}

try {
    magic[row][col] = i; // 赋值
} catch (ArrayIndexOutOfBoundsException e) {
    // 偶数异常处理
    System.out.print("Error:Input n is even!\n");
    e.printStackTrace();
    return false;
}
```

主函数测试部分见源代码文件。

运行结果:

```
Input n for generateMagicSquare():19
generateMagicSquare() returned true
-----6.txt-----
true

Input n for generateMagicSquare():6
Error:Input n is even!
java.lang.ArrayIndexOutOfBoundsException: 6
    at P1.MagicSquare.generateMagicSquare(MagicSquare.java:134)
    at P1.MagicSquare.main(MagicSquare.java:201)
generateMagicSquare() returned false

Input n for generateMagicSquare():-5
Error:Input n is negative!
java.lang.NegativeArraySizeException
    at P1.MagicSquare.generateMagicSquare(MagicSquare.java:124)
    at P1.MagicSquare.main(MagicSquare.java:201)
generateMagicSquare() returned false
```

## 3.2 Turtle Graphics

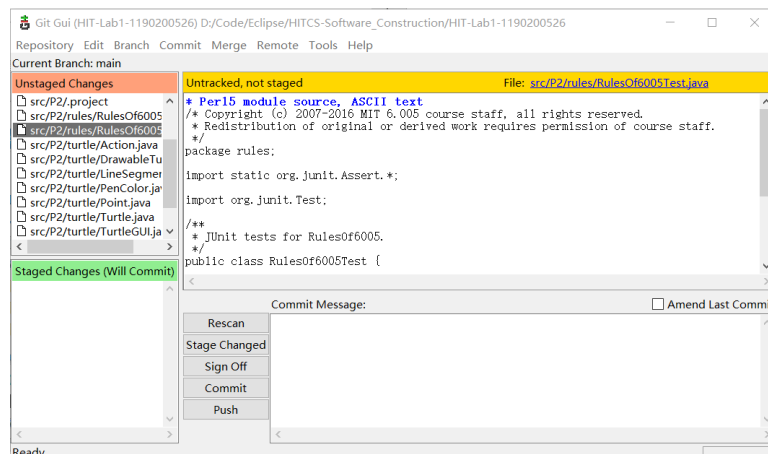
该任务的主要目标是练习使用 Git 的基本功能、学会在 Eclipse 中导入项目、熟悉 Turtle Graphics 中的一些函数接口及简单使用 Junit 进行程序测试。主要任务是实现简单的绘图及计算。

### 3.2.1 Problem 1: Clone and import

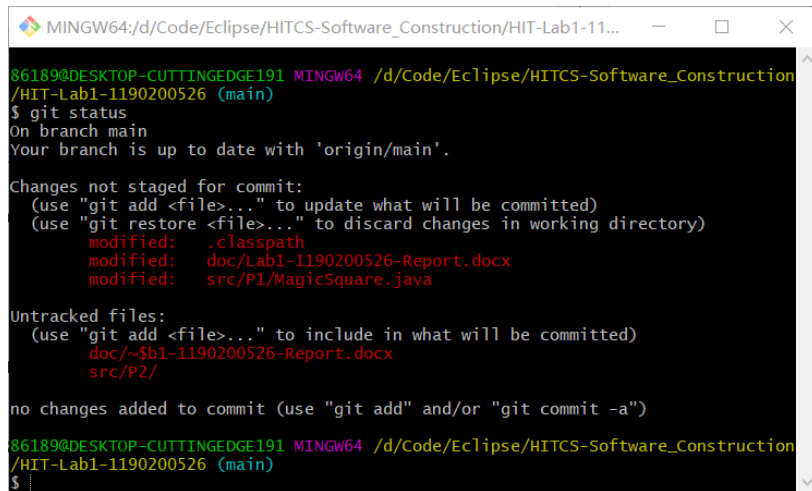
按实验手册要求,我从 GitHub 网页端下载了该任务代码的 ZIP 压缩包,并提取其中所需内容至之前已经创建好的本地 Git 仓库。

本地 Git 仓库通过 git clone 命令 (SSH 地址) 将 GitHub 上的 Lab1 库克隆至本地指定文件夹,克隆的过程也确定了本地库与远程库的关联关系。

打开本地库文件夹,右键菜单选择 Git Gui Here,可以可视化地查看库的状态并执行操作。界面如下图:



右键菜单中也有 Git Bash Here 的选项, 使用此工具我们可以直接通过键入命令执行对库的操作, 界面如下图:



```

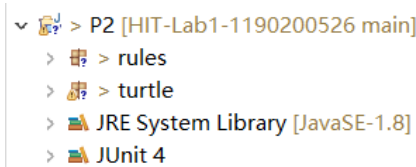
MINGW64:/d/Code/Eclipse/HITCS-Software_Construction/HIT-Lab1-11...
86189@DESKTOP-CUTTINGEDGE191 MINGW64 /d/Code/Eclipse/HITCS-Software_Construction
/HIT-Lab1-1190200526 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   .classpath
    modified:   doc/Lab1-1190200526-Report.docx
    modified:   src/P1/MagicSquare.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    doc/~$b1-1190200526-Report.docx
    src/P2/

no changes added to commit (use "git add" and/or "git commit -a")
86189@DESKTOP-CUTTINGEDGE191 MINGW64 /d/Code/Eclipse/HITCS-Software_Construction
/HIT-Lab1-1190200526 (main)
$
  
```

在开始 P2 的编程部分前, 在 Eclipse 中按 MIT 实验指导的步骤完成了项目的导入, 结果如下图:



```

P2 [HIT-Lab1-1190200526 main]
├── rules
├── turtle
├── JRE System Library [JavaSE-1.8]
└── JUnit 4
  
```

### 3.2.2 Problem 3: Turtle graphics and drawSquare

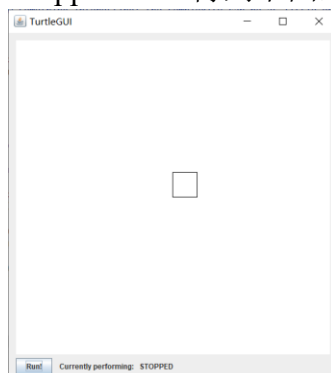
要求: 查看 TurtleSoup.java, 使用方法 forward 和 turn 实现 drawSquare() 函数来画出一个正方形。

执行四次 forward 和 turn 即可。代码如下:

```

public static void drawSquare(Turtle turtle, int sideLength) {
    //画出一个正方形
    for (int i = 1; i <= 4; ++i) {
        turtle.forward(sideLength);
        turtle.turn(90);
    }
}
  
```

右键选择 Run as -- 1 Java Application 得到下图结果:

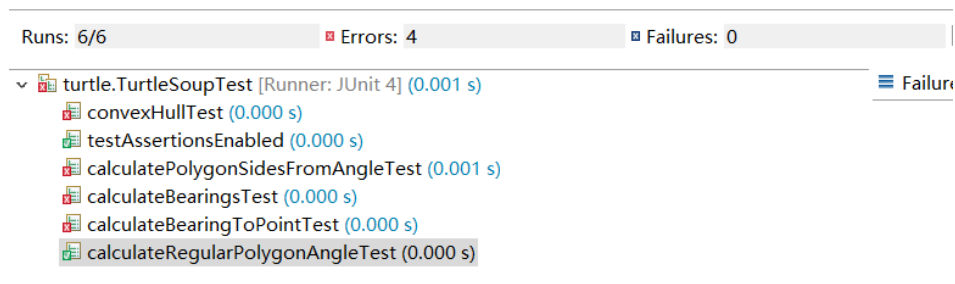




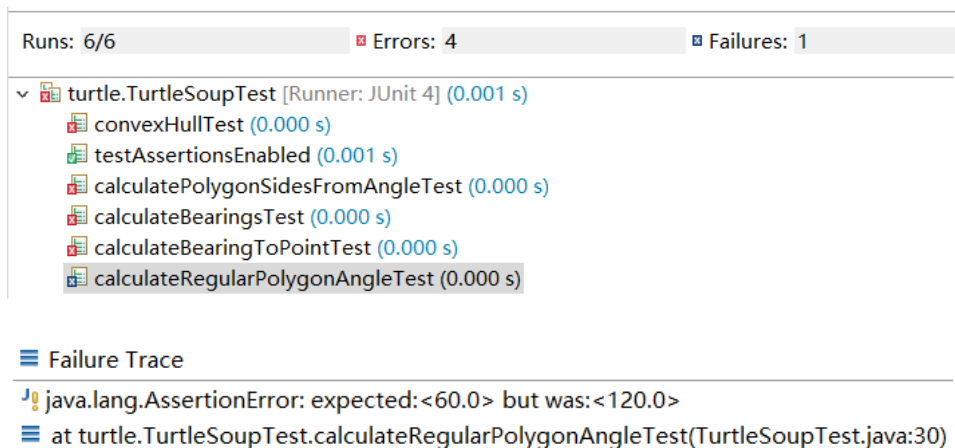
### 3.2.3 Problem 5: Drawing polygons

首先实现函数 `calculateRegularPolygonAngle()`，即计算正  $n$  边形一个内角的大小，可使用  $(sides - 2) * 180.0 / sides$  来计算。

随后使用 JUnit 测试，结果如下图：

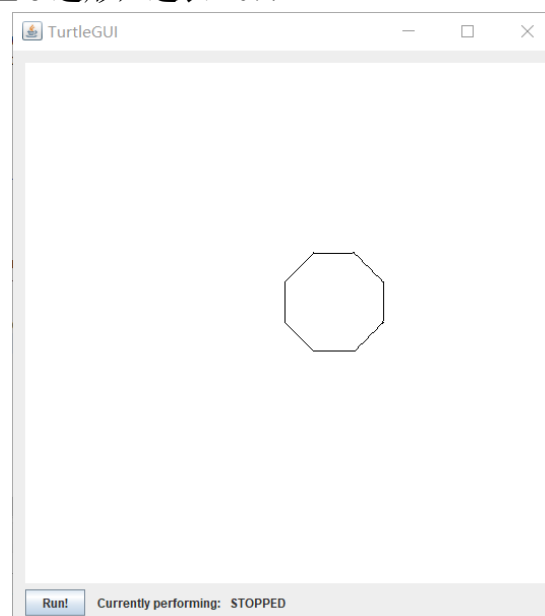


将实现的函数改为错误的，查看信息，了解 JUnit:



最后修复函数，实现 `drawRegularPolygon()`。我的思路是利用循环执行 `forward` 和 `turn`，循环边数次，每次转的角度为  $180.0^\circ - \text{calculateRegularPolygonAngle}()$  的计算值。代码此处略。

结果如下图（正 8 边形，边长 40）：



### 3.2.4 Problem 6: Calculating Bearings

首先实现 `calculateBearingToPoint()` 函数。此函数根据当前位置、方向及目标位置计算前往目标位置需调整的角度。思路为使用  $\arctan(x/y)$  的原理计算夹角，将夹角转换至所需范围，最后计算角度差，也要转换一次范围。代码及注释如下：

```
double res;
int dis_X = targetX - currentX;
int dis_Y = targetY - currentY;
res = Math.toDegrees(Math.atan2(dis_X, dis_Y)); // 计算与y轴正方向夹角
res -= currentBearing; // 计算角度差
if (res < 0)
    res += 360.0; // 处理负角度差情况
return res;
```

随后实现 `calculateBearings()` 函数。此函数调用 `calculateBearingToPoint()` 函数计算 List 中每相邻两点间要调整的角度，并将结果写入一个 `List<Double>` 对象输出。

主要实现思路为首先获取初始坐标、目标坐标及默认方向，通过函数 `calculateBearingToPoint()` 计算调整角度写入结果 List，并更新当前坐标、目标坐标及当前方向，如此重复直至遍历完所有坐标。

主体部分代码如下图：

```
// 获取初始坐标
cur_X = xCoords.get(0);
cur_Y = yCoords.get(0);
List<Double> res = new ArrayList<>(); // 保存结果
for (int p = 1; p < X_size; ++p) {
    // 获取目标坐标
    dst_X = xCoords.get(p);
    dst_Y = yCoords.get(p);
    // 计算调整角度、更新当前角度
    adjust = calculateBearingToPoint(cur_Degree, cur_X, cur_Y, dst_X, dst_Y);
    cur_Degree = (cur_Degree + adjust) % 360.0;
    // 写入结果
    res.add(adjust);
    // 更新当前坐标
    cur_X = dst_X;
    cur_Y = dst_Y;
}
```

最终测试结果如下图：

Finished after 0.022 seconds

Runs: 6/6

Errors: 2

Failures: 0

turtle.TurtleSoupTest [Runner: JUnit 4] (0.000 s)
 

- convexHullTest (0.000 s)
- testAssertionsEnabled (0.000 s)
- calculatePolygonSidesFromAngleTest (0.000 s)
- calculateBearingsTest (0.000 s)
- calculateBearingToPointTest (0.000 s)
- calculateRegularPolygonAngleTest (0.000 s)

Fail

### 3.2.5 Problem 7: Convex Hulls

要求我们实现 `convexHull()` 函数，此函数计算并输出构成凸包周长顶点的输入点的最小子集。由于对此问题的定义及解决算法完全不熟悉，我首先在网上查找了相关资料。

此问题最常用的算法为 Graham 扫描法和 Gift wrapping 算法，此处我采用较容易实现的 Gift-Wrapping 算法。具体实现步骤如下：

a) 首先在所有点中选取  $y$  坐标最小的一点当作基点。如果存在多个点的  $y$  坐标都为最小值，则选取  $x$  坐标最小的一点；

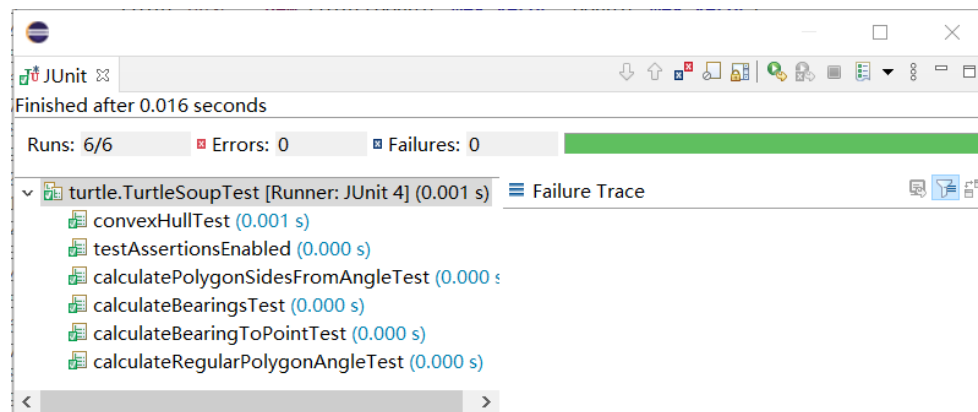
b) 然后遍历点集，计算与基点连接后相对于  $y$  轴正方向偏转最小的点  $P$ ，偏转均最小时选距离最大的，则  $P$  对应线段  $HP$  一定在凸包上，即点  $H$ 、 $P$  在结果集合中；

c) 将点  $P$  设为当前点，遍历点集，按照 b) 重复执行；

d) 最终再次走到基点时结束。

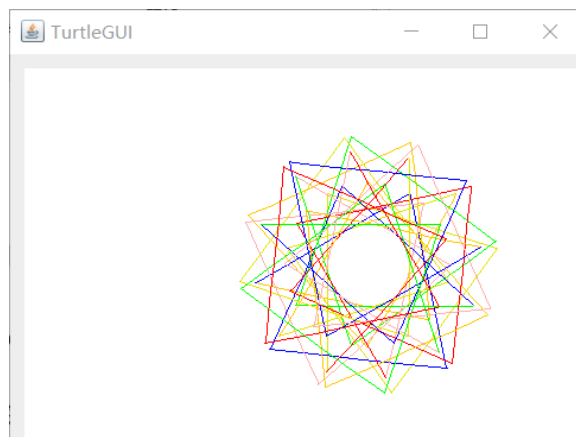
计算偏转角可使用重载的 `calculateBearingToPoint` 方法。具体代码实现较长，见源代码文件。

JUnit 测试结果如下图：



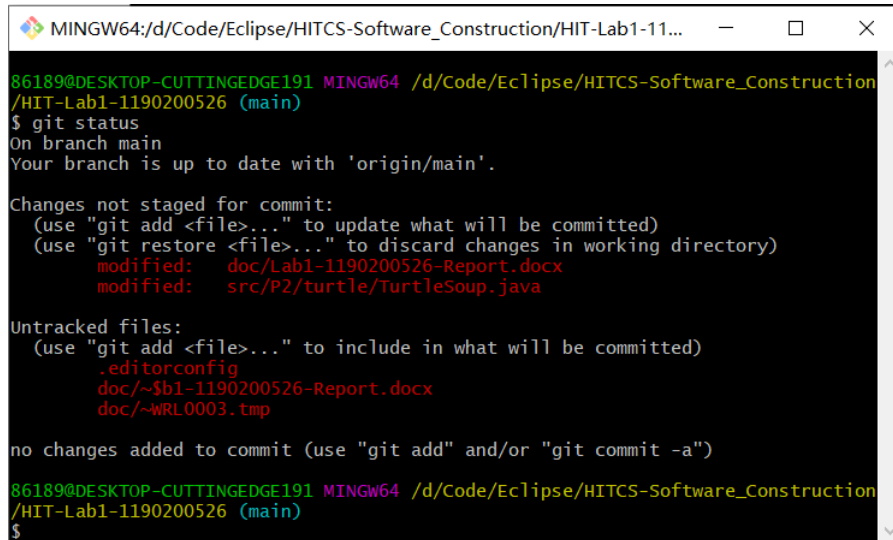
### 3.2.6 Problem 8: Personal art

要求使用 Turtle Graphics 自行设计绘图。我的设计主要利用了 Problem 5 开始时由于忘记对内角求补而得到的不规则图形，通过循环画出一组这样的图形，并在作图过程中不断修改颜色可得到下图：



### 3.2.7 Submitting

打开本地 Git 库所在文件夹，右键菜单选择 Git Bash Here，进入 Git Bash，使用 `git status` 命令，查看当前库状态，如下图：



```
86189@DESKTOP-CUTTINGEDGE191 MINGW64 /d/Code/Eclipse/HITCS-Software_Construction/HIT-Lab1-1190200526 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   doc/Lab1-1190200526-Report.docx
        modified:   src/P2/turtle/TurtleSoup.java

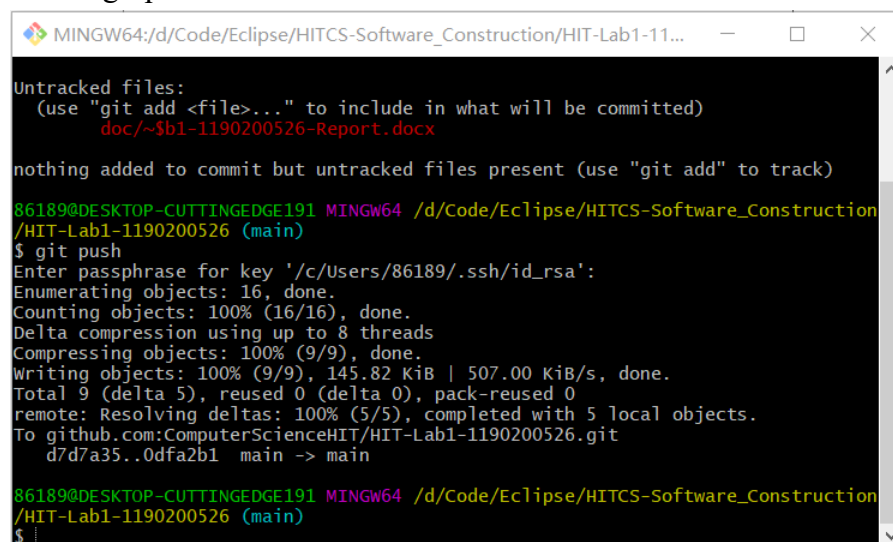
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .editorconfig
        doc/~$b1-1190200526-Report.docx
        doc/~WRL0003.tmp

no changes added to commit (use "git add" and/or "git commit -a")

86189@DESKTOP-CUTTINGEDGE191 MINGW64 /d/Code/Eclipse/HITCS-Software_Construction/HIT-Lab1-1190200526 (main)
$
```

使用 `git add *` 命令，将所作所有更改暂存，之后使用 `git commit -m + 描述` 将更改提交至本地库（图略）。

最后使用 `git push` 命令，将 commit 推送至 GitHub。如下图：



```
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        doc/~$b1-1190200526-Report.docx

nothing added to commit but untracked files present (use "git add" to track)

86189@DESKTOP-CUTTINGEDGE191 MINGW64 /d/Code/Eclipse/HITCS-Software_Construction/HIT-Lab1-1190200526 (main)
$ git push
Enter passphrase for key '/c/Users/86189/.ssh/id_rsa':
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 145.82 KiB | 507.00 KiB/s, done.
Total 9 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 5 local objects.
To github.com:ComputerScienceHIT/HIT-Lab1-1190200526.git
   d7d7a35..0dfa2b1  main -> main

86189@DESKTOP-CUTTINGEDGE191 MINGW64 /d/Code/Eclipse/HITCS-Software_Construction/HIT-Lab1-1190200526 (main)
$
```

## 3.3 Social Network

该任务要求利用一些数据结构实现和测试 `FriendshipGraph` 类和 `Person` 类，这两个类将以无向图形式刻画简单的社交网络模型，同时也要求设计的类支持未来扩展到有向图。

### 3.3.1 设计/实现 FriendshipGraph 类

根据实验手册要求,我定义了两个 private 对象:

```
private List<Person> personList;
private List<List<Integer>> friendGraph;
```

第一个用于保存关系网中的所有 Person 对象,第二个用于保存社交关系,原理类似于邻接矩阵。这些 private 对象将在构造时被初始化。

按照要求,公共方法主要有三个: addVertex, addEdge 及 getDistance。

对于 addVertex 方法,我的思路为:正常情况下,程序将新顶点加入 personList,并调用私有方法扩展 friendGraph ( $N \times N \rightarrow (N + 1) \times (N + 1)$ ); 如果发生姓名重复,则抛出一个异常。具体实现如下:

```
public void addVertex(Person p) {
    for (Person q : personList) { // 检测是否重复
        if (p.getName().equals(q.getName()))
            throw new RuntimeException("Error:" + p.getName() + " is already added!");
    }
    personList.add(p);
    extendfriendGraph();
}
```

对于 addEdge 方法,首先检测传入的参数是否都在 personList 中,如果是则设置 friendGraph 中对应位置为 1,否则抛出一个异常。具体实现如下:

```
public void addEdge(Person p1, Person p2) {
    int index1, index2;
    index1 = personList.indexOf(p1);
    index2 = personList.indexOf(p2);
    if (index1 < 0 || index2 < 0) {
        throw new RuntimeException("Error:At least one vertex is not in the graph!");
    }
    friendGraph.get(index1).set(index2, 1); // 设置边
}
```

对于 getDistance 方法,开始时先判断两个参数是否相等,相等则返回 0,若不相等,检测传入的参数是否都在 personList 中,如果不是,抛出异常;通过以上两个检测后,使用广度优先搜索的思想求出最短路径返回。

此方法核心在于 BFS,此部分代码如下:

```
Queue<Integer> queue = new LinkedList<Integer>(); // 队列用于广度优先搜索
Map<Integer, Integer> disMap = new HashMap<>(); // 保存中间结果并用于顶点判断是否访问过
queue.add(curIndex);
disMap.put(curIndex, 0);
while (!queue.isEmpty()) {
    curIndex = queue.poll();
    int curDis = disMap.get(curIndex);
    for (int i = 0; i < personList.size(); ++i) {
        if (friendGraph.get(curIndex).get(i) == 1 && !disMap.containsKey(i)) {
            disMap.put(i, curDis + 1);
            if (i == dstIndex) {
                queue.clear();
                break;
            }
            queue.add(i);
        }
    }
}
```

实现过程与 C++ 类似。通过使用 Java 中与 C++ 相似的 Queue 类及 Map 类中的 HashMap,可以较为方便地实现 BFS。

此类中私有方法 extendfriendGraph 较为简单,此处略去。

### 3.3.2 设计/实现 Person 类

Person 类作为实现的社交网络中人的代表，只需要简单的属性、构造函数及公共方法用于设置、保存和获取人的姓名。具体实现如下：

```
/* 私有属性 */
private String name;

/* 构造函数 */
public Person(String name) {
    this.name = name;
}

/**
 * 获取姓名
 *
 * @return 姓名拷贝
 */
public String getName() {
    String nameCopy = name;
    return nameCopy;
}
```

由于在实验手册客户端代码示例中没有有关删除、修改个人信息的操作，故 Person 类的实现较简单。

### 3.3.3 设计/实现客户端代码 main()

将手册中的代码复制后运行，程序输出与要求一致，如下图：

```
1
2
0
-1
```

如果将 `graph.addEdge(rachel, ross);` 这条语句注释掉，则 rachel 和 ross 之间只存在单向的社交关系 `ross→rachel`，且 rachel 与 ben 之间变为不可达，故第 14-17 行应输出 -1, -1, 0, -1。实际结果如下，与预期一致：

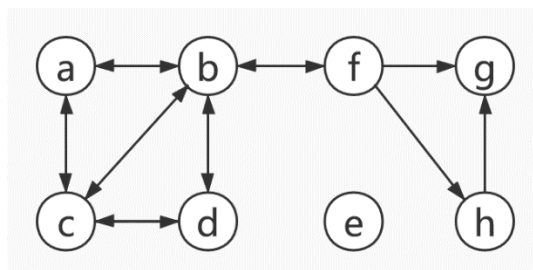
```
-1
-1
0
-1
```

如果将第 3 行引号中的“Ross”替换为“Rachel”，则违背了每个人名字不同的前置条件，我的程序会抛出异常，效果如下：

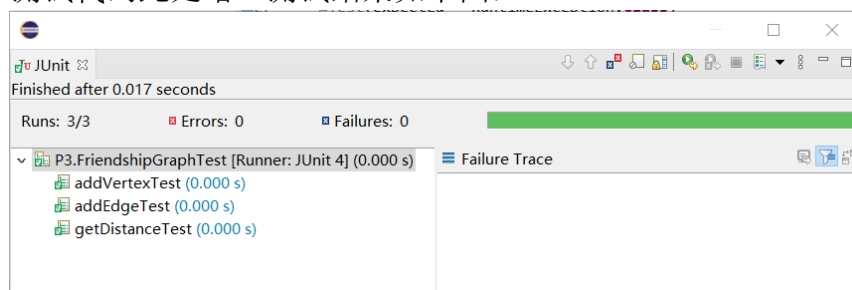
```
Exception in thread "main" java.lang.RuntimeException: Error:Rachel is already added!
    at P3.FriendshipGraph.addVertex(FriendshipGraph.java:24)
    at P3.FriendshipGraph.main(FriendshipGraph.java:108)
```

### 3.3.4 设计/实现测试用例

测试用例覆盖了各公共方法涉及到的非法输入处理（抛出异常），并创建了一个较实验手册稍复杂一些的社交网络图（见下图）来检测 `getDistance()` 方法在各种情况下的正确性。



具体测试代码此处略。测试结果如下图：



## 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。

日期	时间段	任务	实际完成情况
2021-05-13	14:00-15:30	学习 Java 相关语法知识，尝试编写问题 1 的 <code>isLegalMagicSquare()</code> 函数	基本完成，但不完善
2021-05-13	18:30-21:00	完成 <code>isLegalMagicSquare()</code> 函数的编写和测试，撰写对应部分的报告	按计划完成
2021-05-15	16:00-18:30	完成 P1 及对应部分的报告	按计划完成
2021-05-16	15:00-17:30	完成 P2 Problem1 - 4 及对应部分的报告	按计划完成
2021-05-16	18:00-20:00	完成 P2 Problem5、6 及对应部分报告	按计划完成
2021-05-17	20:00-22:40	完成 P2 剩下的编程任务	基本完成
2021-05-18	16:00-17:00	完成 P2 对应报告内容，完善程序代码	按计划完成
2021-05-18	19:00-20:30	学习 Java 类的相关知识，初步尝试设计 P3 的两个类	按计划完成
2021-05-19	16:00-17:00	继续进行 P3 编程	部分方法需完善
2021-05-19	18:30-22:00	完成 P3 除测试用例以外部分的编程及对应报告内容	按计划完成
2021-05-20	14:00-15:30	完成 P3 测试用例的设计	按计划完成
2021-05-20	18:30-21:00	完成实验全部任务	按计划完成



## 5 实验过程中遇到的困难与解决途径

遇到的困难	解决途径
不熟悉 Java 的输入输出流，特别是读取文件操作	阅读网络上相关文章： <a href="https://blog.csdn.net/caidewei121/article/details/89426032">https://blog.csdn.net/caidewei121/article/details/89426032</a> 借阅书籍：《Java 程序设计》、《Java 核心技术卷 II》
Eclipse 项目导入与配置问题	通过尝试和求助其他同学最终成功导入工程，完成相关配置，也对 Eclipse 项目结构有了一定的了解。
不了解凸包问题及其算法	查看维基百科上的关于凸包的内容： <a href="https://zh.wikipedia.org/wiki/%E5%87%B8%E5%8C%85">https://zh.wikipedia.org/wiki/%E5%87%B8%E5%8C%85</a>
不熟悉如何编写 JUnit 测试	参考 P2 的 Junit 测试代码--TurtleSoupTest.java

## 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教训

- 学会使用 Eclipse 进行 Java 简单编程；
- 能够设计简单的测试用例并使用 Junit 进行程序测试；
- 深化了课堂上所学的关于规约、可变性与不可变性等知识。

### 6.2 针对以下方面的感受

- (1) Java 编程语言是否对你的口味？

我对 Java 编程很感兴趣。Java 提供了大量功能强大的类（类似于 C++ 的 STL），在具备 C\C++ 语言的基础后能较快熟悉 Java 的基本使用，应用也十分广泛。

- (2) 关于 Eclipse IDE；

开始使用时遇到了一些困难，但经过不断尝试和探索，现在使用已经没有什么障碍。

- (3) 关于 Git 和 GitHub；

学习此课程前我已经使用了一段时间，已经掌握了基本的使用方法。

- (4) 关于 CMU 和 MIT 的作业；

作业要求具体，步骤明确，完成后很有收获。

- (5) 关于本实验的工作量、难度、deadline；

此次实验工作量较大，难度也不小，但 deadline 较为宽松。

- (6) 关于初接触“软件构造”课程；

课程注重实践，与实际应用紧密结合，我也有一定的兴趣。