

**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 | 熊峰 |
| 学号 | 1190200708 |
| 班号 | 1903008 |
| 电子邮件 | xiong257246@outlook.com |
| 手机号码 | 13114991114 |

**目录**

[1 实验目标概述 1](#_Toc72698434)

[2 实验环境配置 1](#_Toc72698435)

[3 实验过程 2](#_Toc72698436)

[3.1 Magic Squares 2](#_Toc72698437)

[3.1.1 isLegalMagicSquare() 3](#_Toc72698438)

[3.1.2 generateMagicSquare() 5](#_Toc72698439)

[3.2 Turtle Graphics 7](#_Toc72698440)

[3.2.1 Problem 1: Clone and import 8](#_Toc72698441)

[3.2.2 Problem 3: Turtle graphics and drawSquare 8](#_Toc72698442)

[3.2.3 Problem 5: Drawing polygons 9](#_Toc72698443)

[3.2.4 Problem 6: Calculating Bearings 10](#_Toc72698444)

[3.2.5 Problem 7: Convex Hulls 13](#_Toc72698445)

[3.2.6 Problem 8: Personal art 15](#_Toc72698446)

[3.2.7 Submitting 16](#_Toc72698447)

[3.3 Social Network 17](#_Toc72698448)

[3.3.1 设计/实现FriendshipGraph类 17](#_Toc72698449)

[3.3.2 设计/实现Person类 18](#_Toc72698450)

[3.3.3 设计/实现客户端代码main() 18](#_Toc72698451)

[3.3.4 设计/实现测试用例 19](#_Toc72698452)

[4 实验进度记录 21](#_Toc72698453)

[5 实验过程中遇到的困难与解决途径 22](#_Toc72698454)

[6 实验过程中收获的经验、教训、感想 22](#_Toc72698455)

[6.1 实验过程中收获的经验和教训 22](#_Toc72698456)

[6.2 针对以下方面的感受 22](#_Toc72698457)

# 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

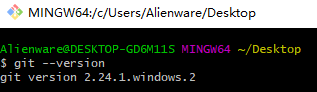
环境配置：

IntelliJ IDEA 2020.3.1 (JDK 1.8 Junit 4.12(下载到lib目录中))



Git:

Git version 2.24.1.windows.2



GtiHub Lab1 URL：https://github.com/ComputerScienceHIT/HIT-Lab1-1190200708.git

# 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

MagicSquare主要实现了以下功能：

1. 读取所给的五个txt文件，并用数组保存；
2. 检验所给的五个txt文件是否为幻方矩阵

2.1检验文件是否为空

2.2检验每行是否存在小数

2.3检验每行是否存在负数

2.4检验每行是否存在非法字符

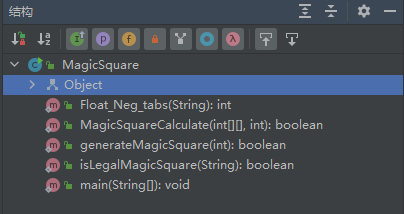
2.5检验行的数量和列的数量是否相同

2.6检验每行元素个数是否相同

2.7检验行、列、对角线元素和是否相同

1. 若是幻方矩阵则返回true，否则抛出并打印错误；
2. 检验要求创建的幻方矩阵的数是否合法，考虑奇数或偶数时的异常，若不存在异常，则生成n\*n的矩阵，并写入6.txt文件

函数结构如下：



### isLegalMagicSquare()

3.1.1.1 设计思路：

首先通过System.getProperty("user.dir") + "\\src\\P1\\txt\\" + fileName和Scanner读取文件，每次读取按行读取，读取后，将每行元素作为参数，调用Float\_Neg\_tabs()函数检验每行文本是否存在小数、负数、除\t以外的其他字符。若读取的数都为正数，且每行元素个数相同，且行数与列数相同，则调用MagicSquareCalculate()函数计算读取的矩阵是否为幻方矩阵。

3.1.1.2 所调用的函数

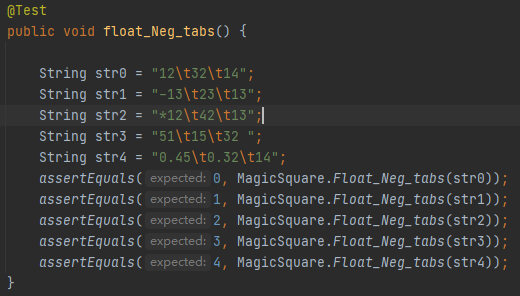
public static int Float\_Neg\_tabs(String Number)

功能：用于检验读取到的每行元素是否存在非法数据，若字符串合法则返回0，若字符串中含有负数，则返回1，若字符串中含有非’\t’的字符，则返回2，若字符串结尾为非’\t’字符，则返回3，若字符串中含有小数，则返回4.

实现过程：对读取的字符串分析，若字符串最后一个字符为非’0’-‘9’且非’\t’，则返回3，表明，当前字符串的结尾为非’\t’字符；对字符串每个字符检查，若存在字符大于’9’或者小于’0’，则检查是否为’.’或‘\t’或’-‘，若存在’-’符号，则表明存在负数，函数返回1；若存在’.‘符号，则表明存在小数，函数返回4；若该符不为以上三种符号，则存在其他非法字符，返回2；对以上检查完毕后，若无异常则，返回0；

运行结果：通过Junit4.12测试，结果如下。





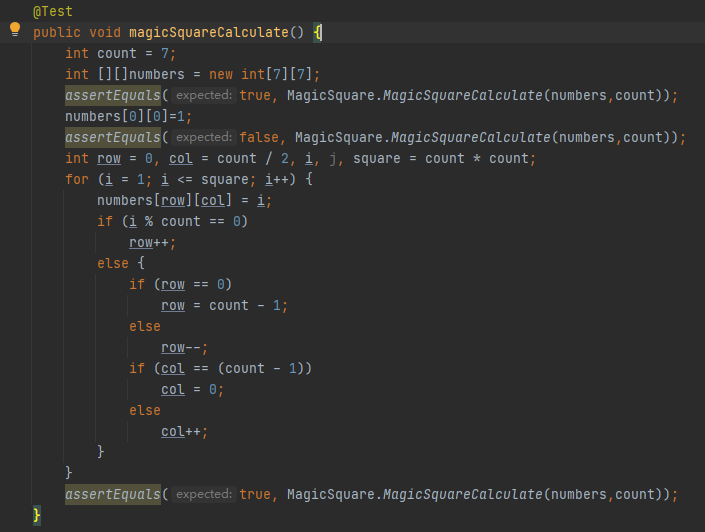
public static boolean MagicSquareCalculate(int[][] Num, int count)

功能：计算每行每列对角线元素和是否相同，若相同则返回true，否则返回false。

实现过程：首先计算第一行元素和，与每行、每列、写对角线元素和比较，若相等，则表明该矩阵为幻方矩阵，否则不为幻方矩阵。

运行结果：通过Junit4.12测试，与预期结果一致，表明函数实现所需功能。





3.1.1.3过程

读取文件后，调用函数判断每行内容是否合法，若不合法，则抛出相应异常，再检查每行元素个数是否相同，若不相同，则抛出异常；若读取的行数超过列数时，抛出异常；若读取文件完毕后，行数小于列数，抛出异常。

调用函数，计算读取的矩阵是否为幻方矩阵。

3.1.1.4运行结果

通过Junit4.12测试，发现运行结果与预期一致。





### generateMagicSquare()

3.1.2.1设计思路

修改后的程序：

首先判断参数是否为负数或偶数，并抛出相应异常。若检验出参数为正奇数，则此时可以通过所给代码生成幻方矩阵。并将幻方矩阵通过PrintWriter写入到6.txt中，并通过函数判断是否为幻方矩阵。

源程序生成幻方矩阵思路（Siamese方法）：

1.首先将1放置在第一行中间；

2.顺序将2，3，…等数依次放在右上方格中；

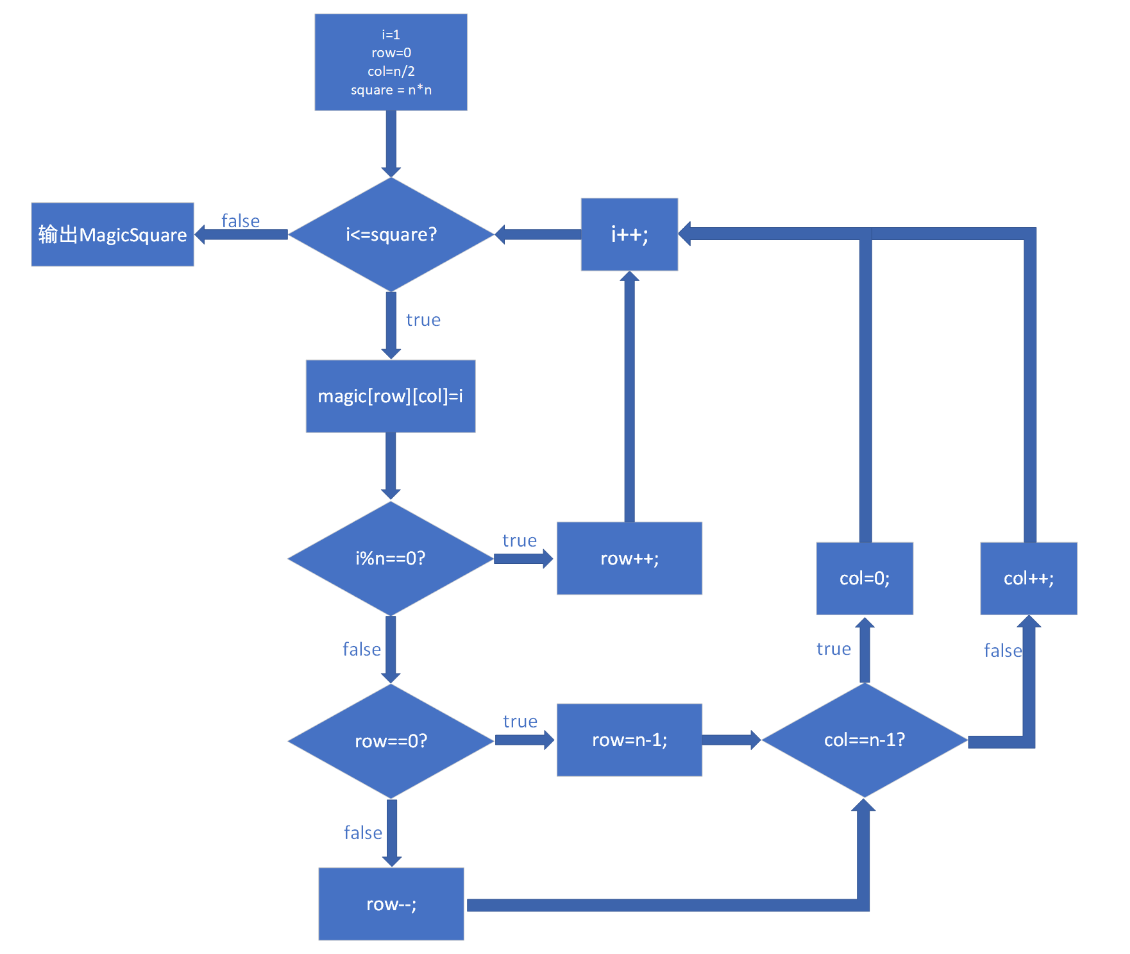
3.当右上方格出界的时候，由另一边进入；

4.当右上方格中已经填有数，则把数填入正下放的表格中；

5.按照以上步骤直到填写完所有N2个方格。

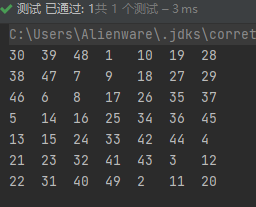
3.1.2.2流程图

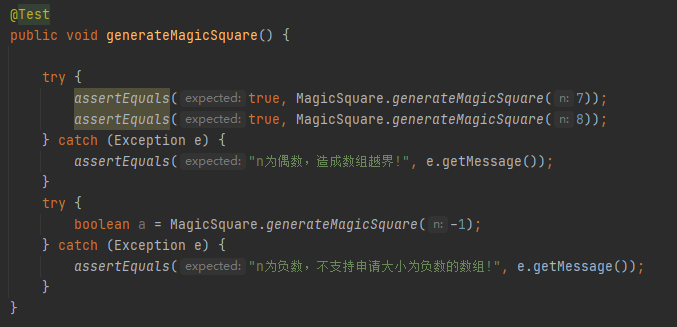
以下流程图为源代码生成幻方矩阵的流程。



3.1.2.3运行结果

通过Junit4.12对代码测试，在负数、偶数时候抛出异常，在正奇数时候，生成对应的幻方矩阵，符合预期要求。

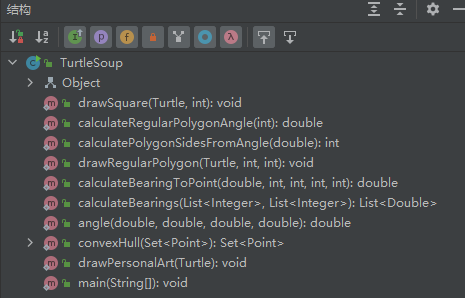




## Turtle Graphics

Turtle Graphics主要实现以下几个功能：

1. 画正方形
2. 计算正多边形内角角度
3. 通过正多边形内角，计算正多边形的边数
4. 画正多边形
5. 已知当前对y轴正半轴的偏角，求从点current到点target需要旋转的角度
6. 当前方向为y轴正半轴时，求从xCoords的点到yCoords的点需要旋转的角度的List
7. 当前方向为x轴正半轴时，求从点A到点B需要旋转的角度
8. 在给定一些点的坐标的时候，求这些点的凸包的集合
9. 绘画出个人作品
10. 测试以上功能



### Problem 1: Clone and import

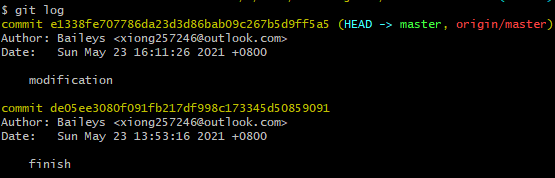
使用git命令 git clone [url] 下载一个项目和他的整个代码历史

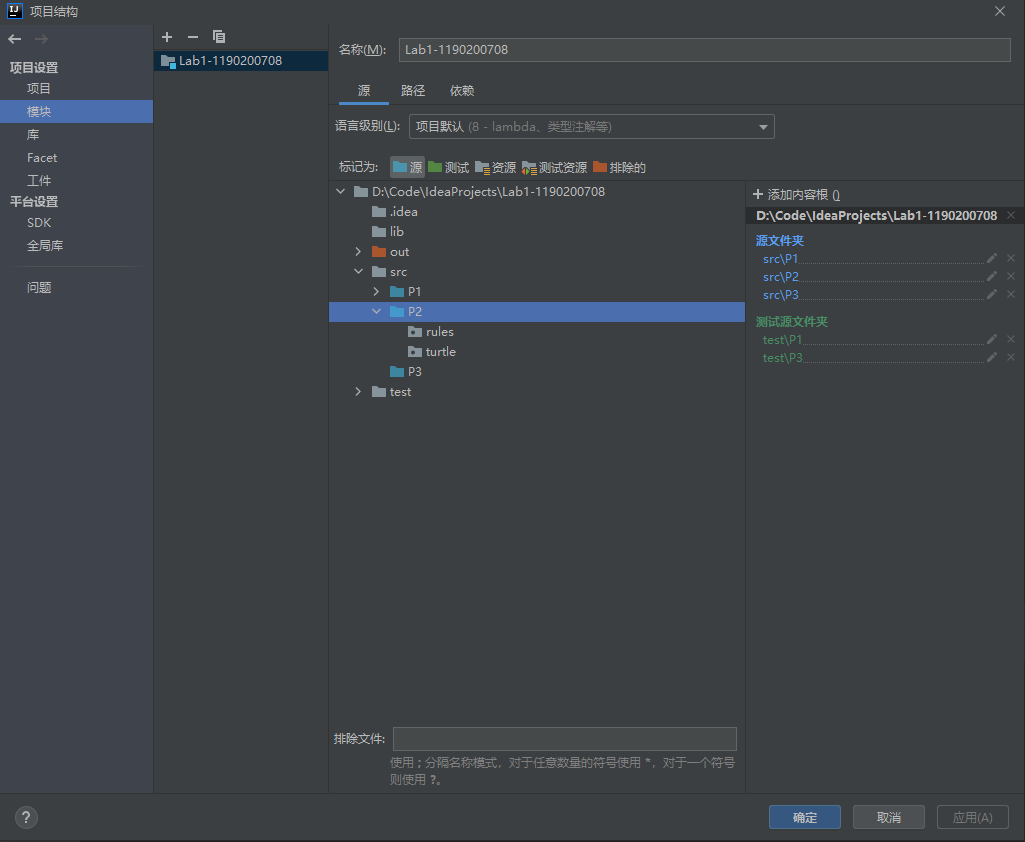
git clone <https://github.com/rainywang/Spring2021_HITCS_SC_Lab1>

在本地使用git init命令，即可新建一个Git代码库。

在本地使用git命令git log即可查看git日志详细信息。

在IntelliJ IDEA中打开项目结构，选择模块，在src中建立P2目录，并标记为源根，将即可正常运行。





### Problem 3: Turtle graphics and drawSquare

3.2.2.1设计思路

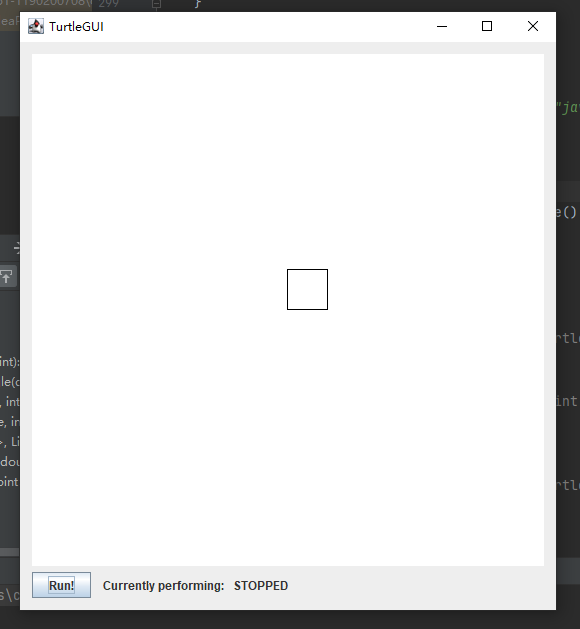
利用turtle的forward和turn方法，完成转动角度，向前移动等操作，画出边长，并确定多边形的内角。

3.2.2.2过程

使turtle首先向当前方向移动sideLength个单位，然后使用turn方法，向右转90度，再向当前方向移动sideLength个单位，再使用turn方法，向右转90度，再向当前方向移动sideLength个单位，再向右转90度，再向当前方向移动sideLength个单位，即可得到一个边长为sideLength的正方形。

3.2.2.3运行结果

该图形的运行结果如下，成功绘制出边长为40个单位的正方形。



### Problem 5: Drawing polygons

3.2.3.1设计思路

1. calculateRegularPolygonAngle

利用公式即可求正多边形内角。

2. drawRegularPolygon

使用已经实现的正多边形内角公式，即可计算出要绘制的正多边形的内角，进行绘制。

3.2.3.2过程

1. calculateRegularPolygonAngle

利用公式很容易计算出所求的内角。

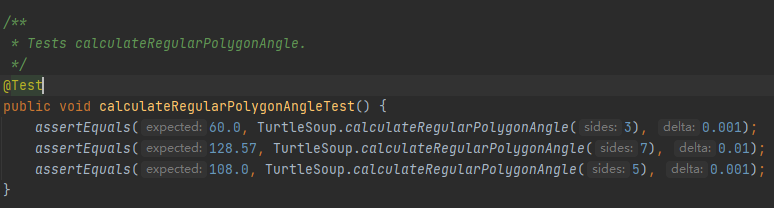
2. drawRegularPolygon

利用已经实现的计算正多边形内角的公式，即很容易计算出内角，通过turtle的turn方法，即可实现角度的转换，在每绘制一条边的时候，转动角度180-angle。再进行绘制。如此循环，即可画出正多边形。

3.2.3.3运行结果

通过Junit4.12对calculateRegularPolygonAngleTest检验，与预期结果一致。

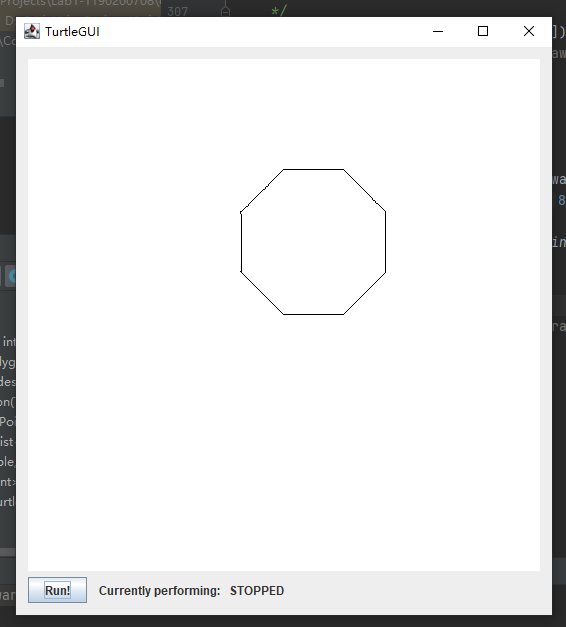




对drawRegularPolygon的功能检验：

例如：绘制一个边长为60的八边形，即调用drawRegularPolygon函数，drawRegularPolygon(turtle1,8,60);

运行结果如下，符合预期结果。



### Problem 6: Calculating Bearings

3.2.4.1设计思路

1. calculateBearingToPoint

通过斜率计算角度，即可计算出要旋转的角度

2. calculateBearings

通过斜率计算角度，将所得到的角度保存到List中

3.2.4.1过程

1. calculateBearingToPoint

比较currentX与targetX的值

(1)若currentX与targetX的值相等，则说明两点在直线x=currentX上。

若currentY>targetY，则说明此时需要将角度调整为y的负半轴，因此此时转动的角度为((540 - currentBearing) % 360 + 360) % 360

若currentY<targrtY，则说明此时需要将角度调整为y的正半轴，因此此时转动的角度为((360 - currentBearing) % 360 + 360) % 360;

由于turtle方法只能向右转动，通过加360并取余得到为正的角度。

(2)若此时currentX<targetX，则说明此时current在target左侧，即可通过斜率计算公式，和Math库函数中的atan函数，计算角度。

与x轴正半轴角度为angle = (Math.atan((double) (targetY - currentY) / (double) (targetX - currentX))) \* 180 / Math.PI;

如图所示，即可得到当前的与x轴的夹角，以靠近y的正半轴为正，靠近y的负半轴为负，不难得出，实际需要转动的角度为

((450 - angle - currentBearing) % 360 + 360) % 360

同理，通过加360并取余得到为正的角度。

currentBearing

current

angle

target

(3)若此时currentX>targetX，则说明此时current在target右侧，即可通过斜率计算公式，和Math库函数中的atan函数，计算角度。

与x轴正半轴角度为angle = (Math.atan((double) (targetY - currentY) / (double) (targetX - currentX))) \* 180 / Math.PI;

如图所示，即可得到当前的与x轴的夹角，以靠近y的正半轴为正，靠近y的负半轴为负，不难得出，实际需要转动的角度为

((630 - angle - currentBearing) % 360 + 360) % 360

同理，通过加360并取余得到为正的角度。

currentBearing

current

angle

target

2. calculateBearings

新建变量currentBearing，保存当前与y轴正半轴的角度

比较xCoords.get(i)与xCoords.get(i + 1)的值(i从0开始到n-2)

1. 若xCoords.get(i)=xCoords.get(i + 1)

判断yCoords.get(i) 与 yCoords.get(i + 1)的关系

此时与1中(1)情况类似(详细分析在1中)，

若yCoords.get(i)> yCoords.get(i + 1)

element = ((540 - currentBearing) % 360 + 360) % 360;

若yCoords.get(i)< yCoords.get(i + 1)

element = ((360 - currentBearing) % 360 + 360) % 360;

1. 若xCoords.get(i)<xCoords.get(i + 1)

此时与1中(2)情况类似(详细分析在1中)，

element = ((450 - angle - currentBearing) % 360 + 360) % 360;

1. 若xCoords.get(i)>xCoords.get(i + 1)

此时与1中(3)情况类似(详细分析在1中)，

element = ((630 - angle - currentBearing) % 360 + 360) % 360;

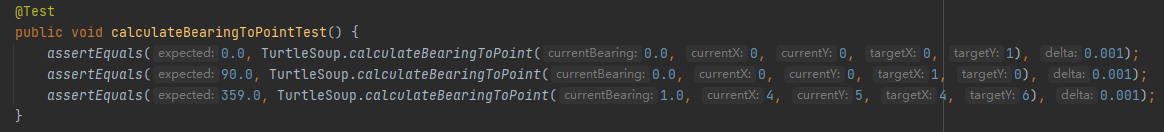
此时currentBearing+=element并对360求余，保存旋转后的角度。并将element元素添加到List中。

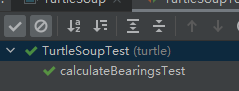
当xCoords与yCoords中的元素遍历后，返回List。

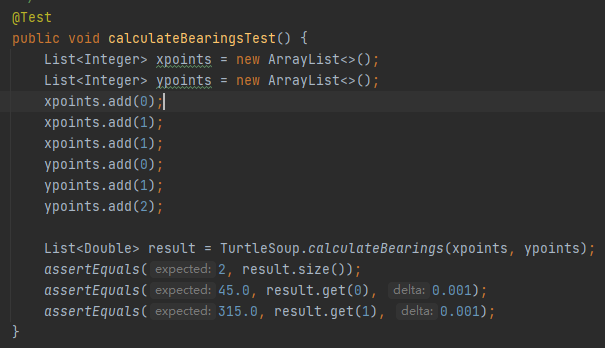
3.2.4.3运行结果

使用Junit4.12测试，与预期结果一致。









### Problem 7: Convex Hulls

3.2.5.1设计思路

利用graham凸包扫描算法，实现求凸包的功能。Graham扫描的思想是先找到凸包上的一个点，然后从那个点开始按逆时针方向逐个找凸包上的点，实际上就是进行极角排序，然后对其查询使用。

3.2.5.2过程

1.将points中的所有点，放在坐标系中，以最下面的点为坐标原点，对所有点进行平移；

2.求每个点与O点的连线与x轴正半轴所形成的夹角，并对其进行排序，若角度相同，则取与O点较近的元素，认为其较小，此时认为O点最小；

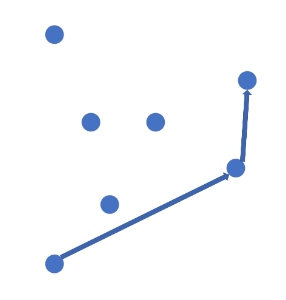
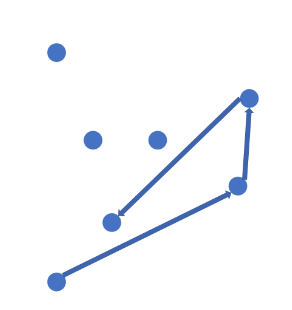
3.取步骤2中，所排列的集合中，取出最小的3个点，假设当前的方向一致为X正半轴，分别计算point1到point2、point2到point3，需要转动的角度angle1、angle2。由于第一次取出的是O点与最下角的点，因此第一次一定有angle1<angle2。此时将point1、point2、point3也存入栈内。

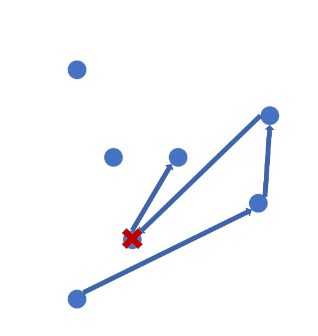
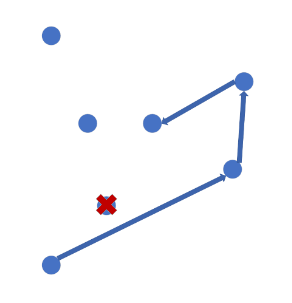
4.继续从排列好的集合中取出新的元素point3，取出栈顶两个元素作为point2和point1，再将这两个元素压栈，若满足angle1<angle2，则将point3存在栈中。若angle1>=angle2，则进入步骤5中。

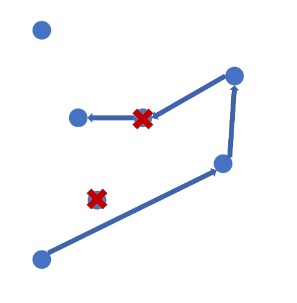
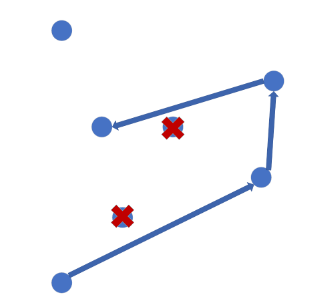
5.若angle1>=angle2，则先弹一次栈，该点不满足凸包的定义，再弹两次栈分别作为point2和point1，此时计算angle1和angle2，若满足angle1<angle2，则将point3压栈，并进入步骤4，否则继续进行步骤5.

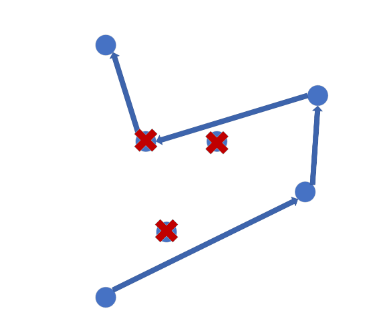
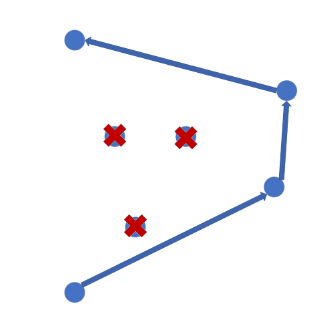
6.重复以上操作，直到points中所有元素都被遍历后，栈中的元素即为所求的凸包。

以下为实例：

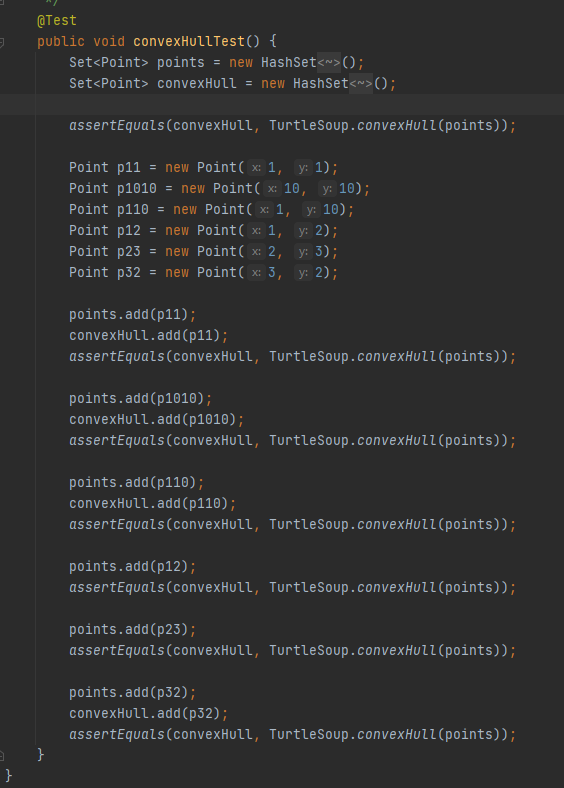
 

3.2.5.3运行结果

运行结果如下，与预期结果一致。





### Problem 8: Personal art

3.2.6.1设计思路

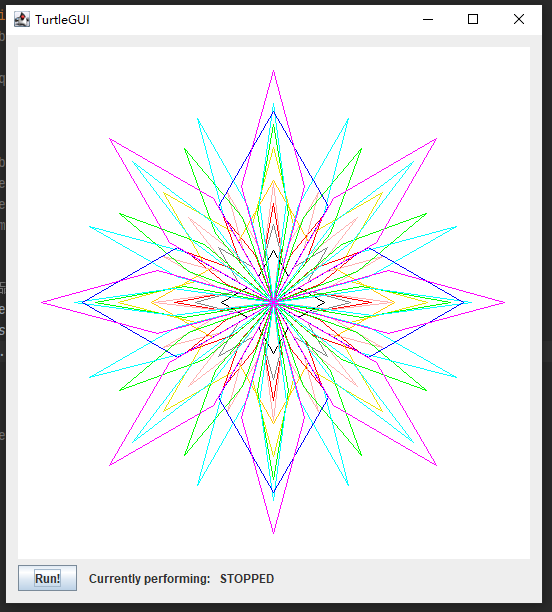
利用循环，foreach等操作，变换颜色，旋转角的度数，画出的图形的个数，完成一副具有个人色彩的绘画。

3.2.6.2过程

首先利用foreach语句，取出pencolor中的所有颜色，用每个颜色的笔画一些团，利用循环取余等操作，完成在循环数不同时，绘画出不同图形的操作。

3.2.6.3运行结果

最终绘制的图形如图所示：



### Submitting

通过git命令

git add . 添加文件到本地库

git commit -m “TurtleSoup” 提交文件到本地库

git remote add origin [git@github.com:ComputerScienceHIT/HIT-Lab1-1190200708.git](mailto:git@github.com:ComputerScienceHIT/HIT-Lab1-1190200708.git) 关联到远程库

git push origin master 推送到github

## Social Network

Social Network主要实现了以下功能：

1. 建立java版的数据结构，有向图。
2. 并完成增加边、节点的操作。
3. 广度搜索计算两个人之间最短的距离。
4. 对输入进行检查，若不符合条件抛出相应的异常。

### 设计/实现FriendshipGraph类

3.3.1.1设计思路

用一个HashSet保存有向图的节点，在每一个Person的对象中，设私有变量，保存每个人认识的人。通过HashSet实现类似邻接表的存储结构，并且能够避免添加重复等问题，查找效率较高。人与人之间距离应该用广度搜索，方便记录距离。

3.3.1.2过程

设置两个私有变量，private HashSet<Person> personArrayList保存有向图的每一个节点，即Person；count保存当前已经存入的人数。

设置一个构造函数，对FriendshipGraph初始化。

addVertex函数用于添加顶点，将图中的每个人都保存到personArrayList中。

addEdge函数用来添加边，若需要添加一条边，则调用person1的方法，将person2的信息保存到person1的私有变量中。

getDistance函数通过广度搜索获得person1与person2的之间的距离。

广度搜索步骤：

1. 将person1保存到HashSet<Person> Search中；
2. 将person1认识的人保存到HashSet<Person> Search\_Next中；
3. 将Search和Search\_Next都保存到新的HashSet<Person> Search\_assist中；
4. 初始化distance = 1；
5. 若Search\_Next中含有person2，返回distance，即为所求距离。否则将Search清空，将Search\_Next保存到Search中，在将Search中所有的人认识的人的HashSet中所有的元素添加到Search\_Next，再将Search\_Next中所有在Search\_assist中保存的元素消除掉，再将所有Search\_Next元素添加到Search\_assist中。此时distance++；
6. 重复步骤5，若Search\_Next最终为空，则不存在距离，返回值为-1；否则，person1与person2之间存在路径，且 person1与person2距离为distance。

### 设计/实现Person类

3.3.2.1设计思路

Person中存在私有变量HashSet<Person> friend，保存当前Person的朋友，并提供一个方法，用于添加当前Person的friend。

3.3.2.2过程

用一个私有变量HashSet<Person> friend，保存当前Person的朋友；再用一个私有变量保存当前Person的姓名。

设置一个构造函数，用所给字符串命名当前Person。

设置一个方法，用于给当前Person添加friend。

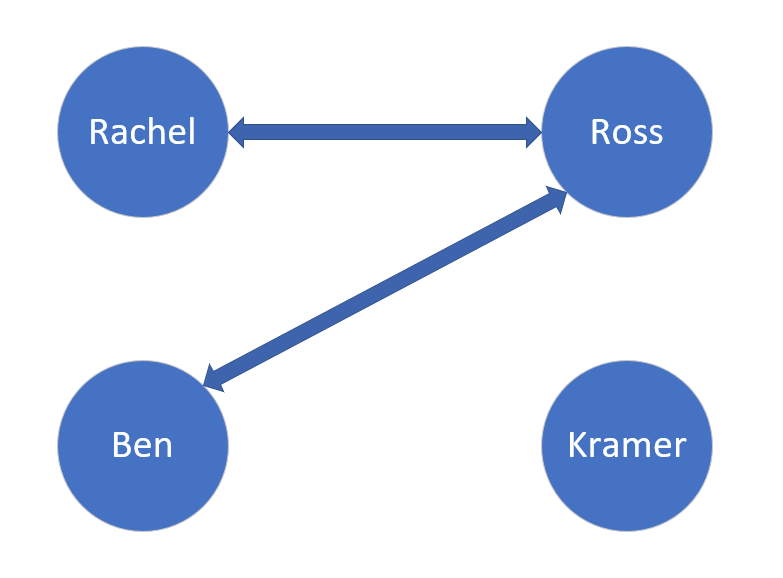
即可完成对每个人，及其好友的设定。

### 设计/实现客户端代码main()

main函数中的代码采用实验指导提供的代码，并对其检验。

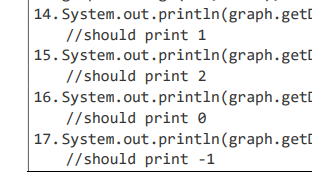
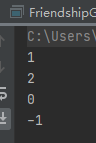
设计思路：首先构造四个Person对象，并添加四条边，通过检查距离是否相同验证函数编写是否正确。

过程：按照P3所给测试代码，四人的关系如下。



运行结果：

运行结果与预期一致。



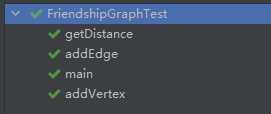
### 设计/实现测试用例

设计思路：

分别检验各个函数是否检测出非法数据，并抛出异常；并对P3所给代码修改后的结果作出预期，并用Test检验。

过程与运行结果：

所有测试成功运行。

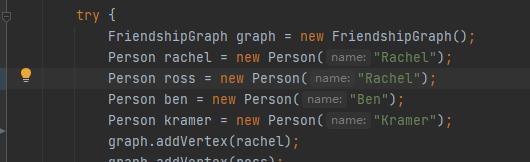


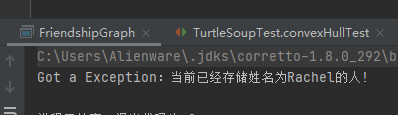
1. 检验addVertex函数时候，检验添加重复的人产生的结果。

当添加两个kramer时候，抛出异常。

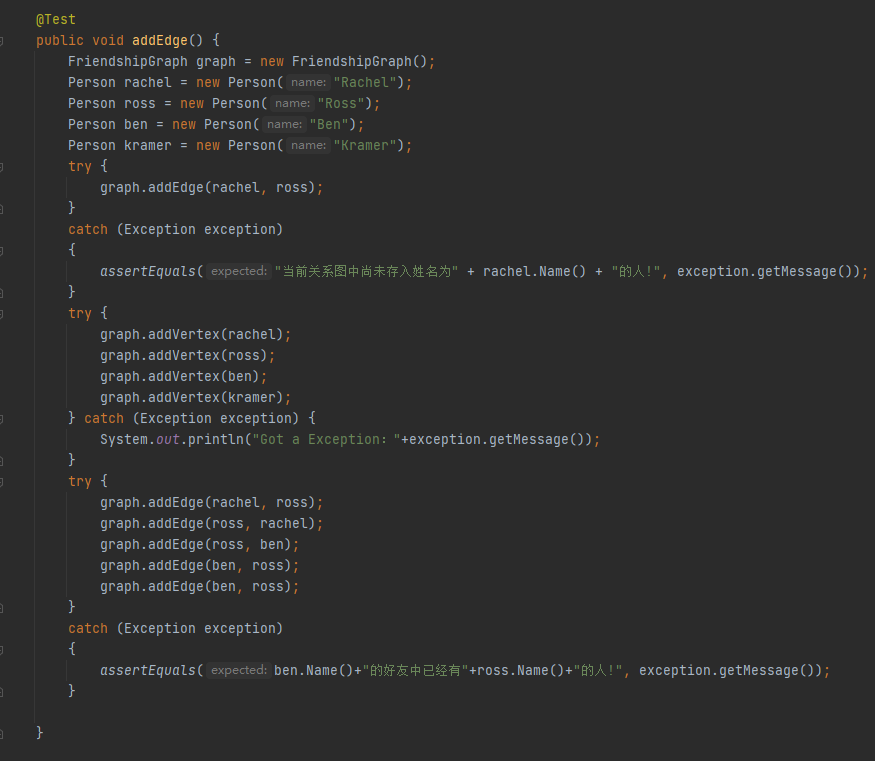


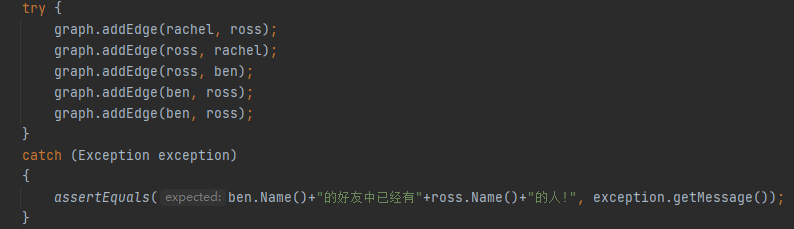
检验P3题目中的，将Ross改为Rachel。当人名存在重复时，抛出异常。这样使每个人都有独一无二的名字





1. 检验addEdge函数时候，检验添加尚未添加的人与其他人的边产生的结果，成功抛出当前未存入Rachel和当前Ben的好友已经有Ross好友的异常。

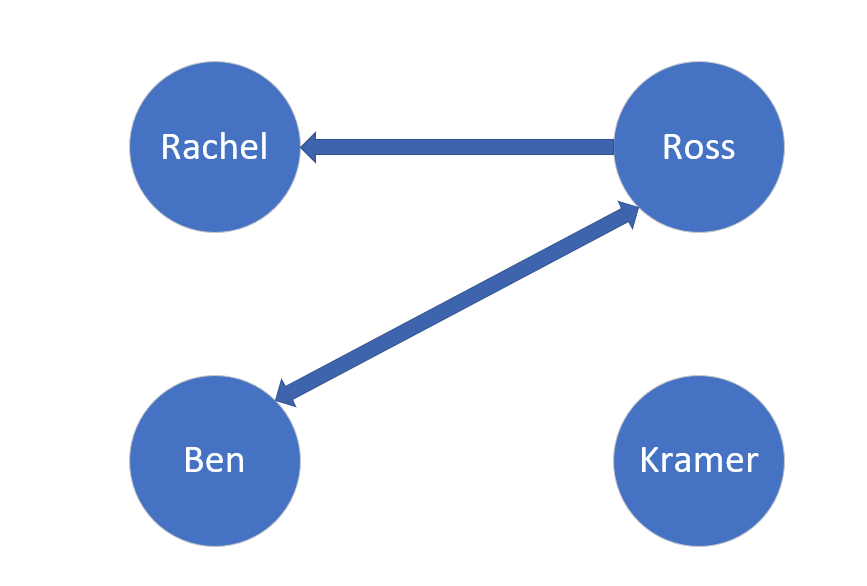


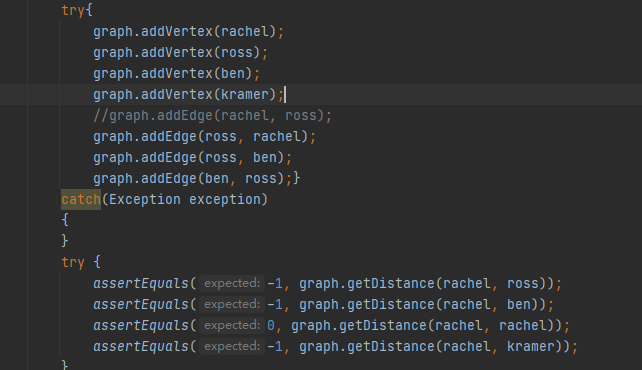


1. 检验getDistance函数时候，检验参数若为当前尚未添加的人产生的结果。



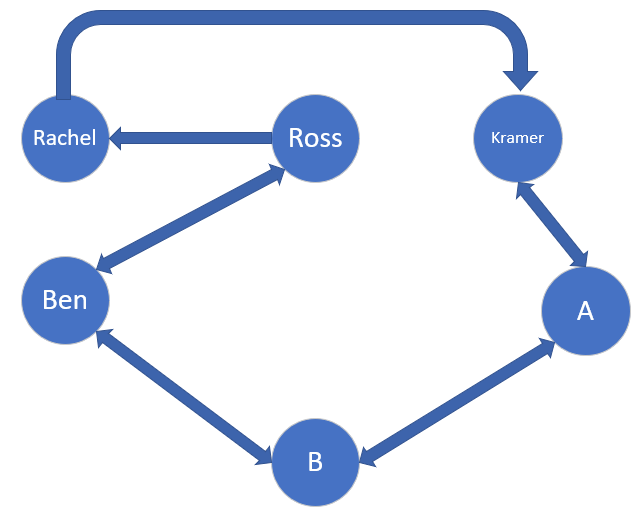
1. main函数中的检验为，将P3中给出的代码第十行注释掉之后的运行结果。在注释掉第十行后，四人的关系如下，预测结果为-1、-1、0、1。

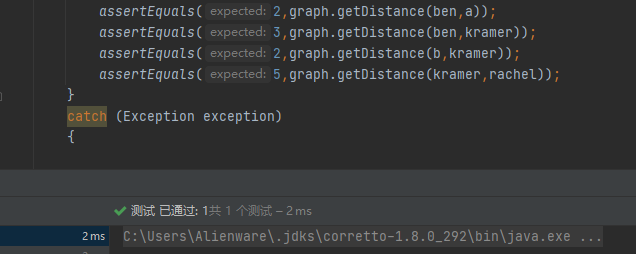




1. main函数的其他检验，由于题设给的例子中人数较少，因此此处自己加了一点检验。六人的关系如下：、

由图可得，ben与a的距离为2，ben与kramer的距离为3，b与kramer的距离为2，kramer与rachel的距离为5。





# 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 |
| 2021-05-18 | 20:00-22:30 | 编写问题1的isLegalMagicSquare函数并进行测试 | 按计划完成 |
| 2021-05-19 | 15:00-20:00 | 编写问题1剩余部分 | 按计划完成 |
| 2021-05-20 | 15:45-18:00 | 测试问题1中是否存在问题，处理问题1中需要抛异常的部分。 | 按计划完成 |
| 2021-05-20 | 19:00-22:00 | 编写问题2 | 按计划完成 |
| 2021-05-21 | 17:00-22:00 | 编写问题2剩余部分及问题3，并进行测试 | 按计划完成 |
| 2021-05-22 | 16:00-22:00 | 测试问题3剩余部分，并完善注释 | 按计划完成 |
| 2021-05-23 | 10:00-12:00 | 测试P1、P2、P3所有功能，并完善注释，完成报告 | 按计划完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 在问题1中遇到，文件读取遇到问题。 | 通过查阅相关资料解决问题。 |
| 在问题2中的凸包算法遇到问题。 | 通过查阅相关资料并上互联网搜索资料结局问题。 |
| 对于路径的设定有问题。 | 通过与同学讨论并查阅资料解决。 |

# 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

1. 在写代码时，应该注意添加版本控制，一开始没有注意这点，在使用eclipse打开idea的文件后，尝试编译后，发现idea的源码无法运行了。
2. 对内存的了解不够深刻，在P2求凸包处，发现仅仅构造出在数值上相等的点无法通过测试。
3. 应该注意在写代码时候就完成报告，在写完代码后在写报告没有写代码时即刻书写的想法丰富。

## 针对以下方面的感受

1. Java编程语言是否对你的口味？
2. 关于Eclipse IDE
3. 关于Git和GitHub
4. 关于CMU和MIT的作业
5. 关于本实验的工作量、难度、deadline
6. 关于初接触“软件构造”课程
7. Java语言的语法与C语言和C++语言很接近，使得学习起来比较容易。并且对C++来说进行了简化和一定的提高，提供了丰富的类库和API文档，以及第三方开发包工具包，因此实际体验非常好。
8. Eclipse在外观设计方面较IDEA略有差距，但eclipse总体体验还不错。实际体验上可能IDEA的自动修复与智能提示等功能更加强大。
9. GitHub让我可以将自己的代码保存到网上，使得在任意终端都可以查看自己的代码，Git使本地回退版本更加容易，总体体验很棒。
10. CMU与MIT的作业设计非常精妙，略有难度，但总体不错。
11. 工作量略大，但是感觉报告内容过多，希望可以略微削减报告内容。Deadline提前规划后还好。
12. 了解了软件构造一些基本概念，对我写程序有很大益处。