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	
学号	
班号	
电子邮件	
手机号码	

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。

2 实验环境配置

首先下载 Eclipse 安装包，同时下载 JDK11，进行简单的环境配置之后完成了对 Eclipse 的配置，从而可以开始进行 java 编程。对于 git 的配置主要是生成了一个 ssh，使用 git remote 命令连接实验代码仓库之后每次 push 使用 git push origin master 命令将代码 push 到仓库中的 master 分支上。

3 实验过程

3.1 Magic Squares

任务的第一部分是设计一个函数检测 5 个文本文件中的矩阵是否是幻方（幻方定义是首先为方阵，其次行列，对角线之和均相等）其中涉及对于非法输入的处理等；第二部分是使用给定的一个函数生成幻方，并检测，这一部分同样涉及对于非法输入的处理。

3.1.1 isLegalMagicSquare()

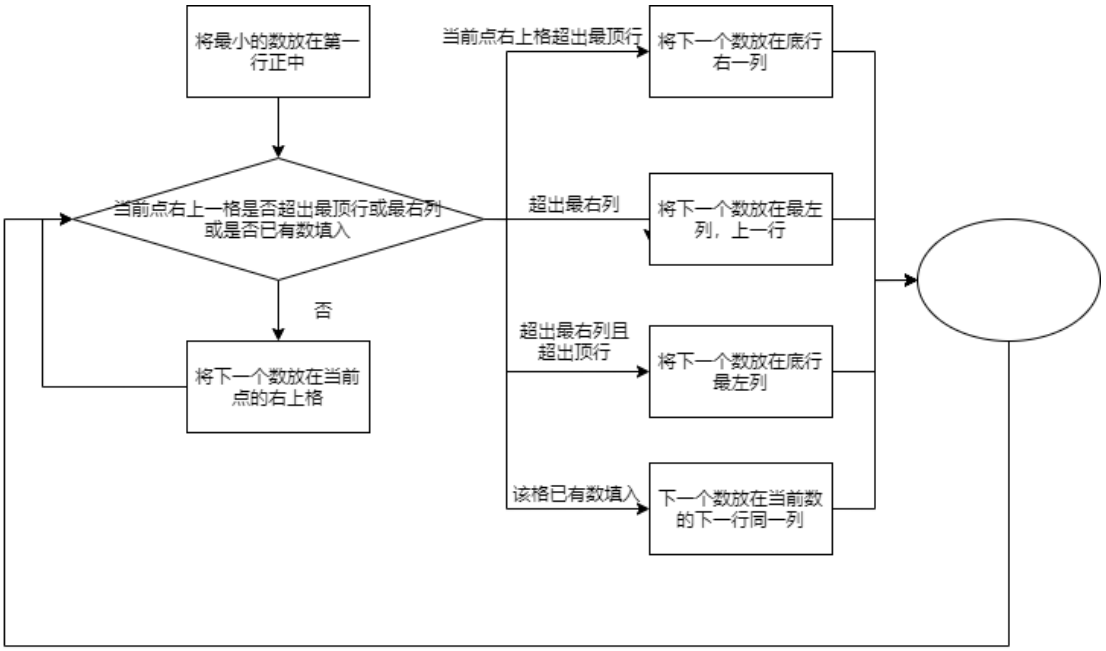
首先，实验的第一步是读取 txt 文件中的内容部分，本代码中采用的是 java.io.BufferedReader 包中的函数实现这一功能的。当读取到信息后首先进行非法数据检测，如果出现非数字，非整数等非法字符或出现不是方阵或是数据没有以\t 分割，分别输出错误信息并退出程序。在这一遍历过程中将 txt 中读到的数据填到一个二维数组中作为判断使用。

接下来判断是否是幻方，采用的方法是首先计算主对角线上的和作为比较标准，接着遍历计算每一行、每一列的和，如果与主对角线和不同，输出 false，如果遍历结束发现都符合条件那么输出 true。代码片段如下：

```
for (int i = 0; i < col; i++)
    sum += square[i][i];
for (int i = 0; i < row; i++) {
    int rowSum = 0;
    int colSum = 0;
    for (int j = 0; j < col; j++) {
        rowSum += square[i][j];
        colSum += square[j][i];
    }
    if (rowSum != sum || colSum != sum) return false;
```

3.1.2 generateMagicSquare()

这一函数是按照劳伯法生成奇数阶幻方的。劳伯法主要思想是把 1（或最小的数）放在第一行正中；每一个数放在前一个数的右上一格；若该数所要放的格已经超出了顶行那么就把它放在底行，仍然要放在右一列；若该数数所要放的格已经超出了最右列那么就把它放在最左列，上一行；若该数所要放的格已经超出了顶行且超出了最右列，则放在底行左列；若该数所要放的格已经有数填入，那么就把它放在前一个数的下一行同一列的格内，代码流程图如下：



这一函数如果输入的参数为偶数将会出现报错，主要原因的如果参数是偶数那么第一步找正中的过程将会找不到一个满足的格，那么将报错。如果是负数的话由于方阵不可能元素个数不可能是负数因此报错。对于这两种情况实验代码都进行输出错误原因后退出程序。

3.2 Turtle Graphics

这一部分一共八个任务，首先需要获取源代码，然后创建和管理本地仓库。

画正方形、据内角求边数、据边数求内角，画正多边形，计算方位，计算凸包，个性创作。

3.2.1 Problem 1: Clone and import

首先需要获取代码，采用的方法是将本地仓库与代码仓库连接之后使用 git pull 命令获取需要的代码文件。在本地使用 git 管理时需要先对需要管理的文件夹使用 git init 命令初始化之后，使用 git add，git commit 等命令进行管理。最后使用 git push origin master 命令将代码上传到 github 仓库中。

3.2.2 Problem 3: Turtle graphics and drawSquare

```
    public static void drawSquare(Turtle turtle, int sideLength) {
        for(int i=0;i<4;i++)
        {
            turtle.forward(sideLength);
            turtle.turn(90);
        }
    }
```

主要思路是进行四次循环，每次执行前进一次，转 90° 角一次，就可以画出一个正方形。

3.2.3 Problem 5: Drawing polygons

首先需要计算正多边形的每一个内角的大小，假设是正 n 边形，那么每一个内角的大小是（n-2）*180/n，计算出每一个内角大小之后，需要进行的是 n 次循环，每次循环执行前进与转角任务，其中每次转角的度数为 180- n-2）*180/n。根据这一思路就可以画出正 n 边形。

3.2.4 Problem 6: Calculating Bearings

首先需要处理的是计算两个点之间的转角，这一点可以先忽略现有转角，先计算从当前点到目标点需要的转角，这一转角使用 Math 包中的 atan2 函数实现，假设这一转角为 degree，则最终的 degree 为 degree = (90 - degree) – currentBearing，其中 currenBearing 为在当前点的转角，如果 degree 小于 0 则将其加上 360° 。代码片段如下：

```
public static double calculateBearingToPoint(double currentBearing, int current
                                             int targetX, int targetY) {
    double degree=Math.toDegrees(Math.atan2(targetY-currentY, targetX-currentX)
    degree = (90 - degree) - currentBearing;
    if(degree<0)
    {
        degree+=360;
    }
    return degree;
}
```

而处理多个点之间的转角时只需要进行遍历并累加即可（注意当累加的角大于 360° 时需要对 360° 取模）。

3.2.5 Problem 7: Convex Hulls

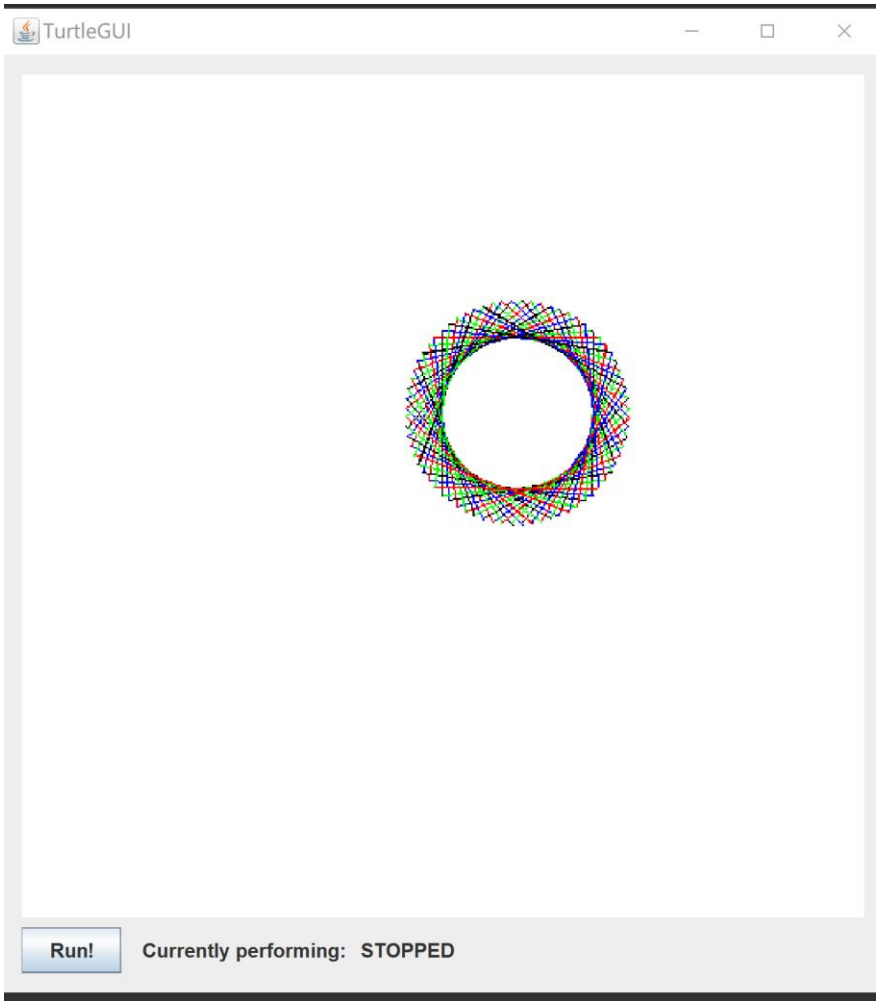
这一部分需要对给定的点求凸包，也就是最少的点组成的多边形能够包含所有的点。思路是找到所有的点中左下角的那个点，这个点一定是需要的点，从这个点开始遍历所有的点，在遍历每个点的时候需要求该点与未选定点之间的距离与需要的转角，选择最小的那个转角的点作为下一个点，如果有多个最小转角的点，那么选择与当前点距离最远的那个点作为下一个点，选定下一个点之后将这个点从未选点集合中删除，加入已选点中。在实现的过程中使用的是对 List 的二重遍历，同时使用了上述步骤中求两点之间需要的转角的函数实现的。部分代码如下：

```
double distance=0;
for(Point temp:freePoint)
{
    double tempDegree=calculateBearingToPoint(0, (int) prePoint.x(), (i
            (int) temp.y());//计算转向角
    double tempDis=Math.pow(prePoint.x() - temp.x(), 2) + Math.pow(preP
    if(tempDegree==minDegree&&tempDis>distance)
    {
        nextPoint=temp;
        distance=tempDis;
    }
    else if(tempDegree<minDegree)
    {
        nextPoint=temp;
        distance=tempDis;
        minDegree=tempDegree;
    }
```

3.2.6 Problem 8: Personal art

这一部分需要做的是自行设计并绘制一个图案，设计代码与画出的结果如下：

```java
public static void drawPersonalArt(Turtle turtle) {
    PenColor[] color = new PenColor[4];
    color[0]=PenColor.BLACK;
    color[1]=PenColor.BLUE;
    color[2]=PenColor.RED;
    color[3]=PenColor.GREEN;
    for(int i=0;i<100;i++)
    {
        turtle.color(color[i%4]);
        turtle.forward(100);
        turtle.turn(95);
    }
}
```



3.2.7 Submitting

首先使用 git remote 命令连接实验代码仓库，之后将 commit 后的结果使用 git push origin master 提交到代码仓库中的 master 分支。

3.3 Social Network

需要实现的是一个社交网络的无向图，其中节点是每一个人，需要实现的函数功能是增加节点、增加两人之间的关系、求两人之间的距离，分别对应三个函数。

3.3.1 设计/实现 FriendshipGraph 类

该类具有两个属性与三个方法。首先两个属性分别是 People 列表与已经存在的人名的列表 nameList，后一个列表主要是为了处理两个人重名的不合法输入情况。在处理完输入不合法情况之后需要对三个方法进行处理。

addVertex 方法只需要向 People 中添加需要加入的 Person，并向 nameList 中添加该人的名字即可

addEdge 方法只需要使用 Person 类中的 addFriend 方法实现即可。

getDistance 方法使用了 java 中的 Map，并使用 BFS，在 BFS 每一次执行中都比对当前的节点是不是所求节点，如果是直接输出距离，如果不是且该节点未访问过，那么使用 Map 将其与一个当前距离+1 联系起来，直到寻找到需要的那个节点或是寻找不到输出-1，代码片段如下：

```
Queue<Person> personQueue=new LinkedList<Person>();
Map<Person,Integer>distanceMap=new HashMap<>();
distanceMap.put(p1, 0);
personQueue.add(p1);
while(!personQueue.isEmpty())
{
    Person topPerson=personQueue.poll();
    int tempDis=distanceMap.get(topPerson);
    List<Person> friends=topPerson.getFriends();
    for(Person tempP:friends)
    {
        if(!distanceMap.containsKey(tempP))
        {
            distanceMap.put(tempP, tempDis+1);
            personQueue.add(tempP);
            if(tempP==p2)return distanceMap.get(tempP);
        }
    }
}
```

3.3.2 设计/实现 Person 类

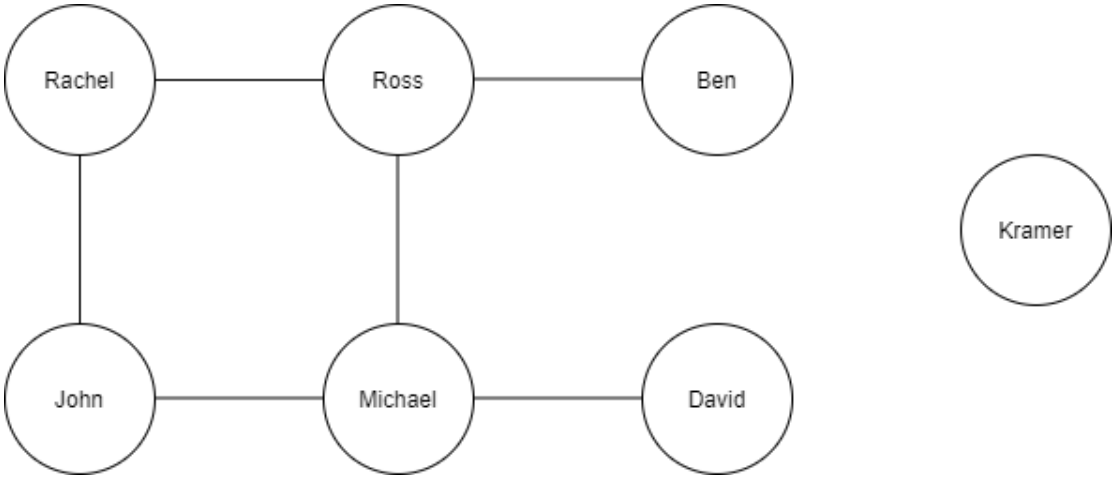
首先 Person 类中有两个属性，分别是名字与朋友列表，分别为 String 与 List。在定义了构造函数的同时，还有三个方法，分别是 addFriend，getName，getFriends，功能分别是增加朋友，获得 Person 的 name 与获得 Person 的 friend 列表。

3.3.3 设计/实现客户端代码 main()

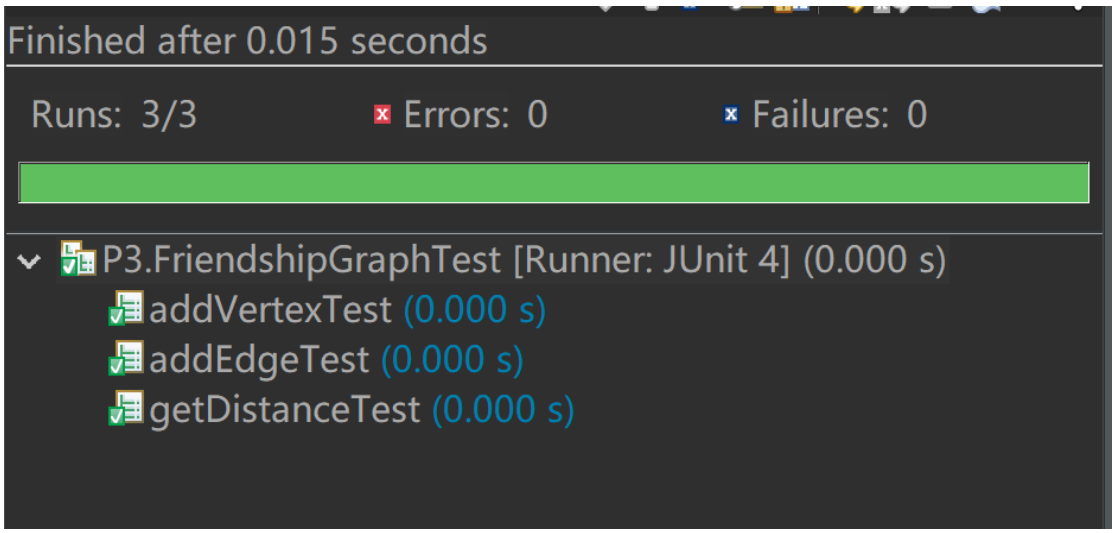
使用的是 pdf 中给出的测试样例，首先定义 Person，接着向生成的 graph 中添加节点、添加关系并测试两个人之间的关系是不是符合我们预先设定的值。

3.3.4 设计/实现测试用例

设计的测试样例的图示如下：



测试结果如下：



4 实验进度记录

日期	时间段	任务	实际完成情况

2021-05-11	13:45-15:30	编写 P1 的 isLegalMagicSquare	完成
2021-05-15	18:30-20:30	编写 P1 剩余部分并完成 P1 测试	完成
2021-05-18	13:45-15:30	编写 P2 的前五个问题	完成
2021-05-20	16:00-20:00	编写结束 P2 与 P3，并完成测试	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
部分时候 github 无法连接	使用加速器或等待网络情况正常均可解决
不知道如何读取 txt 文件内容	参考网上代码，使用 bufferedread 类进行文件流操作，先读入一行，之后利用 split 将 String 转为数字

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

第一次实验对于 java 这种语言还不是很熟悉，正在处于边写实验边学习 java 的过程中。在编写实验一的过程中发现如果使用 java 的 oop 特性可以比较方便的完成一些程序的开发，同时看起来编程的思路也更加的清晰。同时在实验一中感受最深的就是对于异常的处理。在程序中可能会出现各种非法输入，作为代码开发者需要对于非法输入定义一种行为，可以是输出错误信息后退出程序，这也是代码的健壮性所要求的。同时通过实验一认识到了使用 git 管理文件的方便性。

6.2 针对以下方面的感受

(1) Java 编程语言是否对你的口味？

感觉 java 在一些地方和 c++相似，但是显得更加方便一些。现在还处于初学阶段，感觉 java 是一门很优秀的编程语言。

(2) 关于 Eclipse IDE；

私以为 Eclipse 在很多方面和 IDEA 还是存在较大差距，比较明显的一点就是代码补全功能，Eclipse 的代码补全功能还是偏弱。但是 Eclipse 还是可以算是比较好用的。

(3) 关于 Git 和 GitHub；

Git 管理文件十分方便。

(4) 关于 CMU 和 MIT 的作业；

从这次难度来看难度与工作量都适中。

(5) 关于本实验的工作量、难度、deadline；

实验一工作量与难度都适中

(6) 关于初接触“软件构造”课程；

学习到了之前没有接触到的很多编程方面的知识，对于软件构造有了初步的了解。