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	张瑞
学号	1190201421
班号	1936603
电子邮件	1190201421@stu.hit.edu.cn
手机号码	15736059288

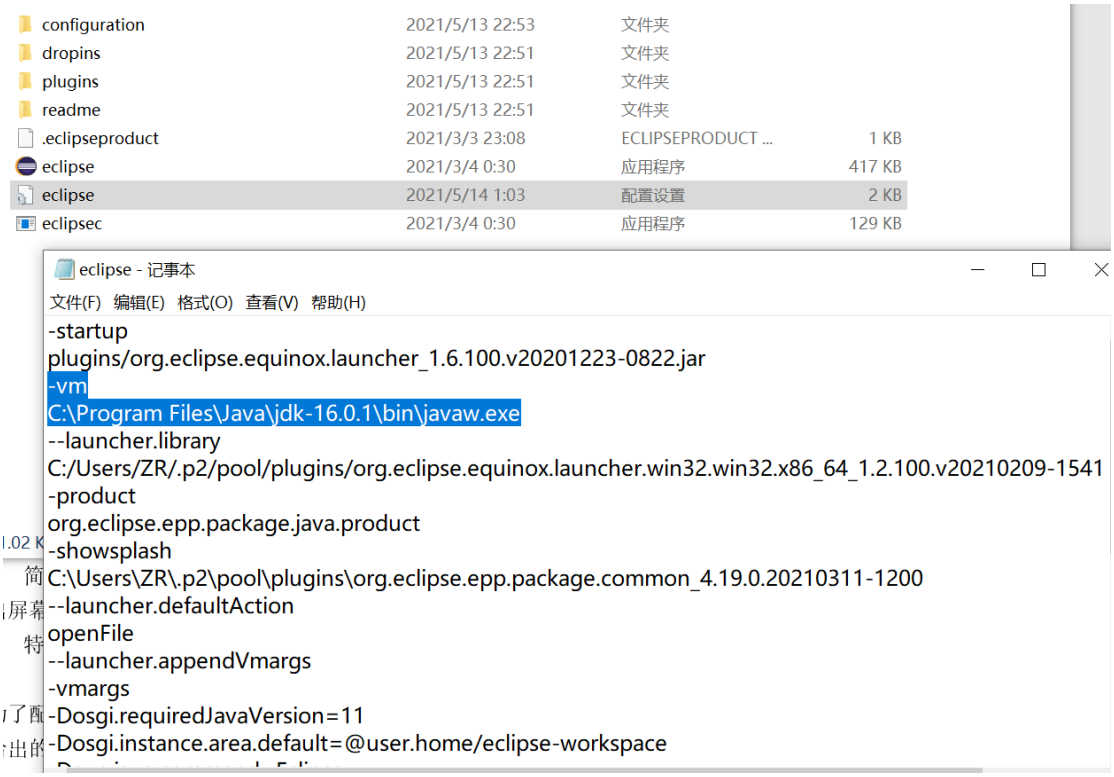
目录

1 实验目标概述

本次实验包含三个问题的求解。一方面，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

为了配置本次实验所需的环境，我参考了老师给出的 lab0 实验指南，按照里面给出的步骤进行，均十分顺利。但在启动 Eclipse 时发现无法打开，在上网查阅相关资料说需要添加环境变量，但仍未成功。最终找到了可行的解决方案，需要在 Eclipse 的配置文件中加入 jdk 的路径：



在这里给出你的 GitHub Lab1 仓库的 URL 地址。

https://github.com/ComputerScienceHIT/HIT-Lab1-1190201421

3 实验过程

3.1 Magic Squares

该任务分为两部分。第一部分是判断一个输入是不是幻方，涉及的操作有从文件中读取输入，判断各元素是否为满足要求的数，判断是否为矩阵，将矩阵按行求和，按列求和，按主对角线和副对角线求和等。第二部分对指导手册里生成一个幻方的代码进行测试，并且改进代码，将生成的幻方写入文件。

3.1.1 isLegalMagicSquare()

首先要将输入从文件中读取出来，判断是否都为正整数，是否含有“\t”以外的符号，是否满足矩阵的特点。以上条件均满足的情况下再求和，判断所有和是否相等。

注意，需要把非法符号的判断放在矩阵判断的前面，否则可能出现将出现非法符号判定为非矩阵的情况，如本题中的第五个输入。该部分代码截图如下：

```
List<ArrayList<Integer>> matrix = new ArrayList<>();
int len = lines.size();
for(int i = 0; i < len; i++) {
    ArrayList<Integer> lineofmatrix = new ArrayList<Integer>();
    String myline = lines.get(i);
    String[] split = myline.split("\t");

    for(int j = 0; j < split.length; j++) {
        try {
            int value = Integer.valueOf(split[j]);
            if(split[j].matches("[0-9]+") && value != 0)
                lineofmatrix.add(value);
            else {
                System.out.println("Illegal number.");
                return false;
            }
        }catch(NumberFormatException e) {
            System.out.println("Illegal symbol.");
            return false;
        }
    }
    matrix.add(lineofmatrix);

    if(len != split.length) {
        System.out.println("It is not a matrix.");
        return false;
    }
}
```

运行结果表明，在提供的五个矩阵里，前两个是幻方，第三个不是矩阵，第四个含有非法数字（负数及浮点数），第五个出现非法符号：

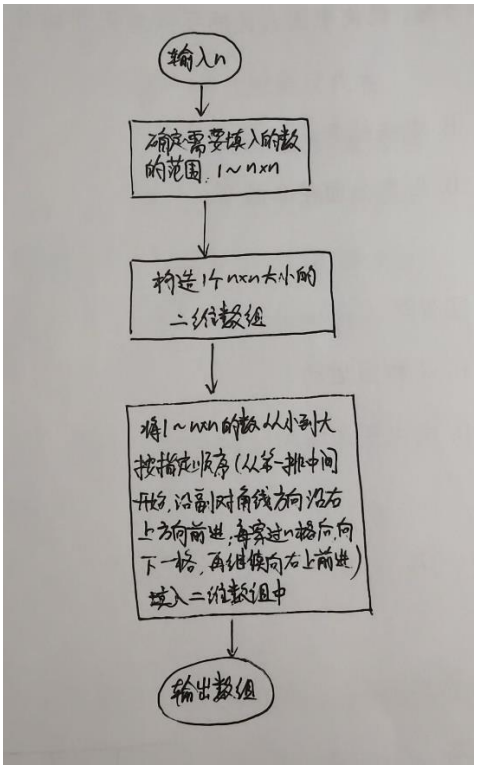
```
true

true

It is not a matrix.
false

Illegal number.
false

Illegal symbol.
false
```

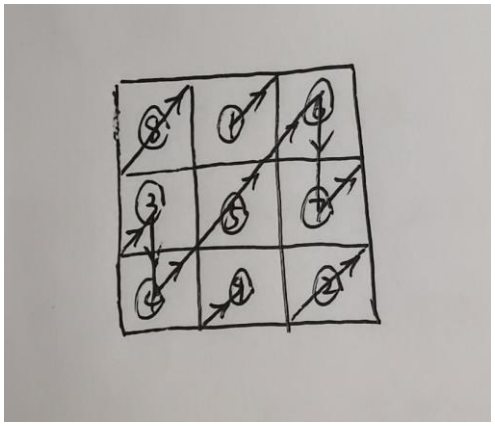
3.1.2 generateMagicSquare()

该方法为指导书给定，只需要加到 MagicSquare 类中即可。然后还要对代码进行修改，使其能写入指定文件，最后利用 isLegalMagicSquare 方法来验证其为幻方。

注意，指导书要求程序在输入为负数和偶数时，要能够“优雅的”退出，函数输出为 false。若不修改，程序会在输入为负数时，因为构造的二维数组下标为负而产生异常并强行退出；在输入为偶数时，会在填完某 n 个数之后，row++出现下标越界的情况，产生异常并强行退出。

绘制的流程图及填数顺序示意图（以 3 为例）如下：





初始时，尚未修改代码，对方法的中文注释如下：

```
public static boolean generateMagicSquare(int n) {//输入参数n，表示构造一个n*n的幻方
    int magic[][] = new int[n][n];
    int row = 0, col = n / 2, i, j, square = n * n;
    for (i = 1; i <= square; i++) {//将1~n*n按从小到大的顺序依次填入
    magic[row][col] = i;//填数时的起始位置为第一排中间格
    if (i % n == 0)//每填完n个数，向下一格继续填数
    row++;
    else {//尚未填完n个数时，沿着副对角线方向向右上进行填数
    if (row == 0)
    row = n - 1;
    else
    row--;
    if (col == (n - 1))
    col = 0;
    else
    col++;
    }
    }
    for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++)
    System.out.print(magic[i][j] + "\t");
    System.out.println();
    }
    return true;
}
```

当输入为奇数时，运行结果正确，成功写入文件，返回 true，并且能通过 isLegalMagicSquare 的验证；当输入为偶数（包含 0 这个特例）和负数时，“优雅的”退出，并返回 false：

```
156
157         Boolean f = MagicSquare.generateMagicSquare(3);
158         System.out.println(f+"\n");
159
160         Boolean g = MagicSquare.generateMagicSquare(-1);
161         System.out.println(g+"\n");
162
163         Boolean h = MagicSquare.generateMagicSquare(0);
164         System.out.println(h+"\n");
165
166         Boolean i = MagicSquare.generateMagicSquare(4);
167         System.out.println(i+"\n");
168
169         Boolean j = MagicSquare.isLegalMagicSquare("src\\P1\\txt\\6.txt");
170         System.out.println(j+"\n");
171
```

Problems Javadoc Declaration Console Coverage

<terminated> MagicSquare

```
true

Negative input.
false

Even input.
false

Even input.
false

true
```

6 - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

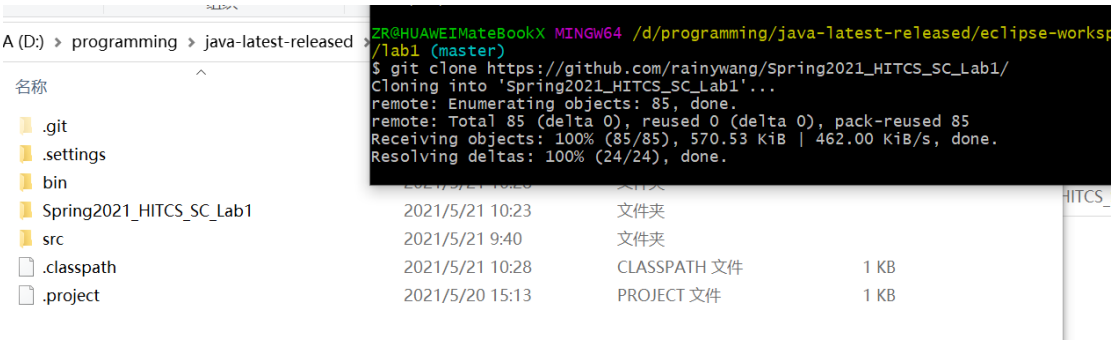
```
8       1       6
3       5       7
4       9       2
```

3.2 Turtle Graphics

完善已有代码，并利用已给出的方法，画出自己设计的图形。

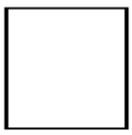
3.2.1 Problem 1: Clone and import

先选好工作区的位置，然后在 git bash 中输入“git init”建立本地仓库，再“git clone https://github.com/rainywang/Spring2021_HITCS_SC_Lab1”即可获取远程仓库的代码。



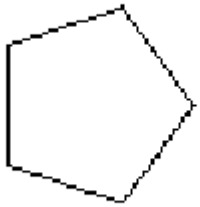
3.2.2 Problem 3: Turtle graphics and `drawSquare`

利用已给的代码中的方法 forward 和 turn，画出正方形。任务很简单，只需要让 turtle 每移动相同长度（sideLength）后向转动 90°，直至画完四条边即可。结果如下：



3.2.3 Problem 5: Drawing polygons

利用已给的方法画出任意正多边形，这里要运用几何知识，先完善多边形的内角计算，公式为 180°-360°/边数，再计算 turtle 每次转弯时角度为 180°-内角度数。实现时，获取边数，然后计算每次转弯的角度，在移动相同长度后转动该角度即可。我以正五边形为例进行了初步测试：



3.2.4 Problem 6: Calculating Bearings

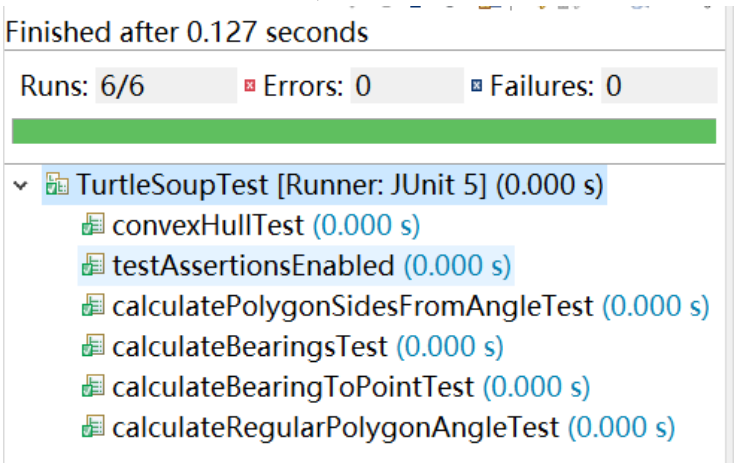
利用已给的方法，计算从起点到终点需要转过的角度。思路详见代码及注释：

```
public static double calculateBearingToPoint(double currentBearing, int currentX, int currentY,
                                             int targetX, int targetY) {
    //运用反三角函数计算目标点与当前点所成斜线与X轴正向的夹角
    double angle = Math.atan2(targetY - currentY, targetX - currentX) * 180.0 / Math.PI;
    //角度为负的，调整为正值
    if(angle < 0)
        angle += 360.0;
    //计算斜线与y轴正向之间的夹角（y轴为始边，顺时针方向为正向），再减去当前偏移角度
    double bearing = (360 - angle + 90 >= 360 ? 90 - angle : 360 - angle + 90) - currentBearing;
    //最后调整为0-360°之间
    return bearing < 0 ? 360.0 + bearing : bearing;
}
```

3.2.5 Problem 7: Convex Hulls

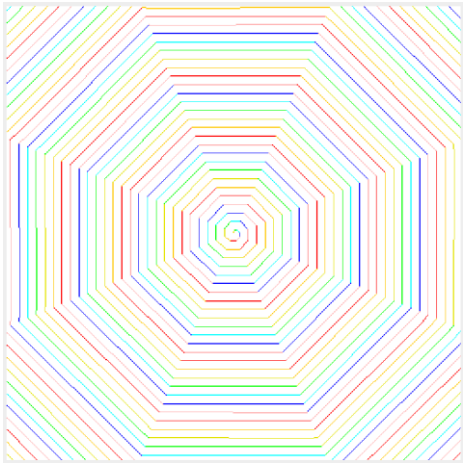
思考易得，当点的个数小于等于 3 时，所有点都应该在凸包内；点的个数大于 3 时，所有点中最左下的点必定在凸包内，此时查找凸包中的其他点，即查找点集中最外侧一层的所有点，若将这些点顺序连线，会发现每次转弯的时候，到路线上下一个点的转弯角度相比于其他所有点是最小的，若出现使得旋转角度同样小的点，则取离当前点更远的点，由此便找到算法的思路。具体实现见代码和注释。

至此，Junit 的测试全部通过：



3.2.6 Problem 8: Personal art

建立一个循环，利用 switch 语句，让 7 种颜色循环出现，每次转向角度为 45°。运行程序，得到一个类似彩虹色的八边形形状的图案：



3.2.7 Submitting

先进入工作区的位置，然后在 git bash 中输入“git init”建立本地仓库，再输入“git remote add lab1 https://github.com/ComputerScienceHIT/HIT-Lab1-1190201421”添加远程仓库，然后输入“git add <filename>”,“git commit -m “***””,“git push -u lab1 master”即可向远程仓库提交代码。

3.3 Social Network

利用图的相关知识，建立一个社交关系网络图。任务分为两部分，一个是按要求构造相关方法，一个是写出各方法对应测试用例。

3.3.1 设计/实现 FriendshipGraph 类

该类包含两个成员变量，一个是 people，存储加入社交网络图的人，另一个是 nameset，存储这些人的名字。该类还有三个方法，一个是 addVertex，用于添加人到社交关系网中，一个是 addEdge，用于添加朋友关系，还有一个是 getDistance，用于获取两人间最短距离。注意，addVertex 的时候需要将待加入的人的名字与 nameset 进行比对，不允许重名出现，若出现将输出提示信息并退出程序。计算两人间最短距离的时候运用了 BFS 的思想。

3.3.2 设计/实现 Person 类

该类包含两个成员变量，name 用于存储人名，friend 用于存储朋友名单。该类还有四个方法，Person 用于对象的初始化，getname 用于获取人名，addfriend

用于添加朋友，getfriend 用于获取朋友名单。

3.3.3 设计/实现客户端代码 main()

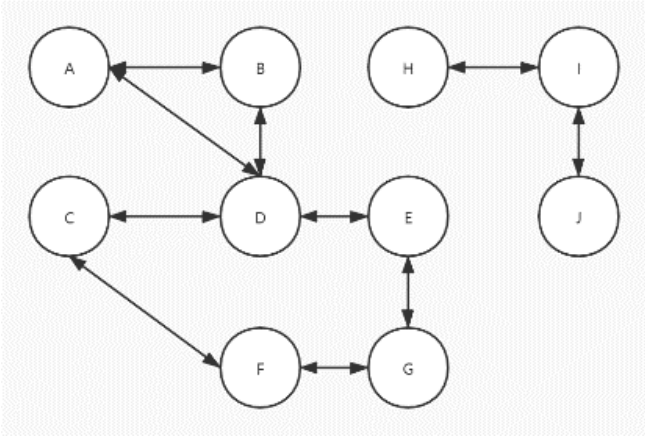
此部分实验指导书已给出，将其移植到 main 中，测试结果正确：

```
1
2
0
-1
```

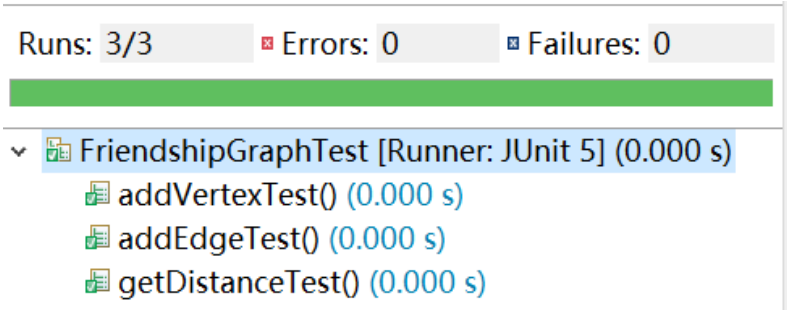
3.3.4 设计/实现测试用例

给出你的设计和实现思路/过程/结果。

测试 addVertex，向社交网络中加入一人，用 assertEquals 判断其已成功加入；测试 addEdge，向社交网络中加入两人，再添加其朋友关系（双向），两次运用 assertEquals 判断其关系已成功加入；测试 getDisdance，仿照 main 中样例，构造一个较为复杂的社交网络图，再调用 assertEquals 判断其最短距离计算准确。自行构造的社交网络图示意如下：



测试结果如下：



对于实验指南中提到的问题：

（1）若 rachel 和 ross 之间只存在单向社交关系 ross→rachel，则判断输出为：-1 -1 0 -1，运行验证确实如此：

```
    public static void main(String[] args) {
        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));
        System.out.println(graph.getDistance(rachel, ben));
        System.out.println(graph.getDistance(rachel, rachel));
        System.out.println(graph.getDistance(rachel, kramer));
    }
}
```

Problems Javadoc Declaration Console Coverage

```
<terminated> FriendshipGraph (1) [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javav
-1
-1
0
-1
```

（2）若将 100 行处的“Ross”换为“Rachel”，违反了不重名这一约束条件，修改程序使得提示出错并结束程序运行：

```
    public static void main(String[] args) {
        FriendshipGraph graph = new FriendshipGraph();
        Person rachel = new Person("Rachel");
        Person ross = new Person("Rachel");
        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));
        System.out.println(graph.getDistance(rachel, ben));
        System.out.println(graph.getDistance(rachel, rachel));
        System.out.println(graph.getDistance(rachel, kramer));
    }
}
```

Problems Javadoc Declaration Console Coverage

```
<terminated> FriendshipGraph (1) [Java Application] C:\Program Files\Java\jdk-16.0.1\
Person Rachel already existed.
Each person must have a unique name.
```

4 实验进度记录

日期	时间段	任务	实际完成情况
2021-05-18	13:45-15:30	配置环境，编写并测试问题 1 的 isLegalMagicSquare 函数	因配置时遇到问题，延期一小时完成
2021-05-20	18:30-20:30	将问题 1 全部完成并写完问题 1 的实验报告	按计划完成
2021-05-21	18:00-18:30	clone 代码，将问题 2 的 problem1-4 完成	按计划完成
2021-05-21	19:30-23:00	完成问题 2 并写完实验报告	完成，但事后发现凸包问题有漏洞需修改
2021-05-22	10:00-10:30	完成凸包问题漏洞	按计划完成
2021-05-22	14:30-18:00	完成问题 3 并写完实验报告	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
不会环境配置	参照指导书及给出的相应网址，遇到问题上 CSDN 查阅他人经验
不熟悉 Java	查看老师已给代码，实在不会的上网搜索
Git 老是打不开，push 也不好使	查了好多方法，没有总是有效的，最后发现硬着头皮多试几次（或许是几十次），突然就行了

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

网络是个好东西，上面能查到很多实用的资料，受到别人的帮助也进一步坚定了我在今后写博客分享经验的想法。有困难还可以多问问身边的人，也许一个困扰了自己许久的问题，别人一句话就能启发到自己，为自己指明方向，节约不少时间。

6.2 针对以下方面的感受

(1) Java 编程语言是否对你的口味？

感觉很不错。在本次实验之前虽然学过 Java 相关课程，但当时的重点在于介绍面向对象这一思想，对语法并不了解，在本次实验中查阅了大量资料和已有代码，也算是进一步学习了 Java 语言，在实践中还巩固了面向对象编程的思想。

(2) 关于 Eclipse IDE；

相当喜欢，页面什么的都很对胃口，对项目的管理也比较方便。而且能清晰的看到每个类的成员变量和方法，这在以前常用的 CodeBlocks 里面是没有的，这个功能我相当满意。

(3) 关于 Git 和 GitHub；

不太稳定，有时需要多次尝试才能打开。我也是第一次接触 github，感觉版本控制的作用还是相当不错的。

(4) 关于 CMU 和 MIT 的作业；

很有趣，第一次尝试全英文的编程背景，看起来难做起来却并不难，而且完成后会有成就感，惊讶自己也能做出世界顶级大学的题目，信心倍增！

(5) 关于本实验的工作量、难度、deadline；

工作量和难度适中，一开始确实有些迷茫，但上手之后就很快，deadline 也比较合适，时间是足够的。

(6) 关于初接触“软件构造”课程；

感受到了和大一编程课的不同，有了模块、对象的意识，开始注意自己程序的健壮性等，也初步有了测试的概念，还感受了版本控制的魅力。总的来说，虽然有一定挑战性，但对本门课程还是很有兴趣的，感觉能有很大的收获。