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	林搏海
学号	1190202128
班号	1903002
电子邮件	1172798125@qq.com
手机号码	13452053282

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。

基本的 Java OO 编程

基于 Eclipse IDE 进行 Java 编程

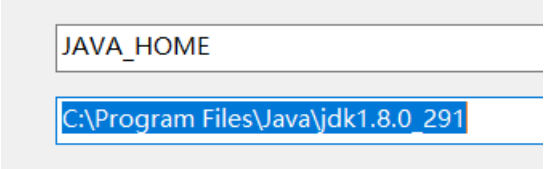
基于 JUnit 的测试

基于 Git 的代码配置管理。

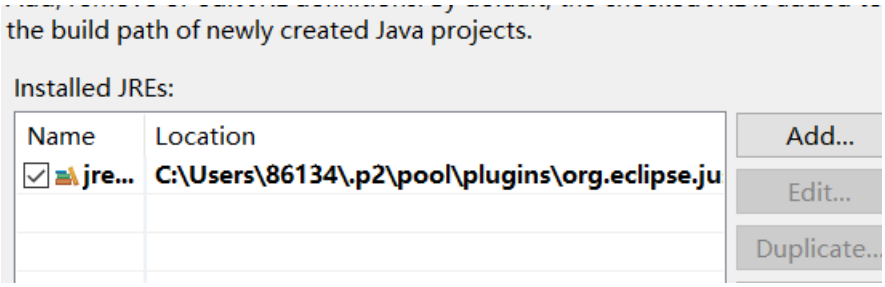
2 实验环境配置

下载和配置 JDK：

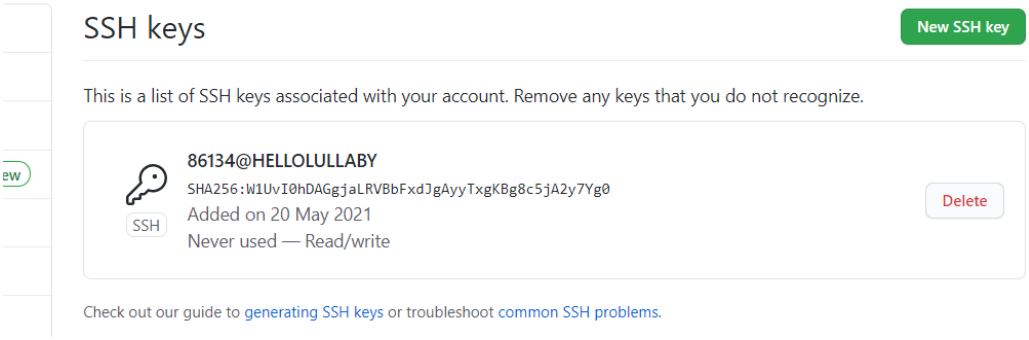
我们首先需要在官网下载 JDK(Java Development Kits)，然后根据网上的教程配置环境变量（如下图）



然后我们需要下载 IDE，我们使用 eclipse，同样需要完成环境变量的配置（如下图）



配置完成后，我们继续配置 Git 的部分，我们通过 ssh 链接将本地 Git 与 github 进行连接（如下图）



我们在 github 中新建仓库，然后附上 Lab1 的地址如下：

https://github.com/microsurge/Lab1-1190202128.git

按照老师的要求，我们需要将分支名改为 master，具体方法如下：

首先通过 ssh 和 git clone 指令将远程仓库下载到本地：

```
86134@HELLOLULLABY MINGW64 /d/GitRepository
$ git clone git@github.com:microsurge/Lab1-1190202128.git
Cloning into 'Lab1-1190202128'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

其次，打开仓库文件，利用 git branch -a 查看所有分支

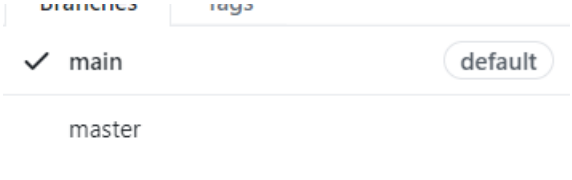
```
$ git branch -a
* main
  remotes/origin/HEAD -> origin/main
  remotes/origin/main
```

再使用 git branch -m main master 将本地分支名替换为 master，并且 push 到远程仓库

```
86134@HELLOLULLABY MINGW64 /d/GitRepository/Lab1-1190202128 (main)
$ git branch -m main master

86134@HELLOLULLABY MINGW64 /d/GitRepository/Lab1-1190202128 (master)
$ git push origin master
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/microsurge/Lab1-1190202128/pull/new/master
remote:
To github.com:microsurge/Lab1-1190202128.git
 * [new branch]      master -> master
```

我们发现此时远程仓库有了两个分支：



将 master 设置为 default 分支再将 main 分支删除即可

3 实验过程

3.1 Magic Squares

该任务的难点主要由以下几个方面：

如何从文件读入二维数组并完成对于其是否为方阵或者数据是否为正整数的检测。

如何检测二维数组是否为 MagicSquare。

如何修改和扩展 generateMagicSquare()。

3.1.1 isLegalMagicSquare()

在进行将文件中的矩阵读入之前，我希望能够确保起已经为正整数了，因此我选择构造 isValidNumber 函数，直接在文件中进行扫描是否是正整数，具体的检测方法为：逐行读入字符串缓冲区，使用 replaceAll 函数将串中的整数和\t 部分替换为空白，再进行对串长进行检测，若串长为 0 则该行数据符合规范。

具体实现方法如下：

```
while((bufLine = buf.readLine())!=null) {
    bufLine = bufLine.replaceAll("[0-9]+\\s*", "");
    if(bufLine.length() != 0) {
        System.out.println("Invalid numbers !");
```

为了区分正整数和小数以及负数，选择合适的正则表达式十分重要，经过查阅资料和不断试错，我构造了[0-9]+\\s*去替换，能够起到准确的识别作用。

此外，若数字合法，isValidNumber 函数的返回值则是矩阵的行数，我们基于行数 row 确定一个 row*row 的二维数组，接下来我们再次扫描文件，逐行读入的结果按照\t 进行拆分，将拆分得到的结果存入一个 Sring 类的数组中，数组的每一个元素都是一个拆分得到的串。

我们比较该 String 数组的元素个数和 row 的值，若不相等则直接返回 false，输出非方阵等提示信息，若相等则利用 String.valueof()将每个字符串转换成对应的正整数，重复 row 次之后我们就将文件中的数据转移到了二维数组中。

```
for(i = 0; i < row ; i++) {
    bufLine = buf.readLine();
    //将每一行按\t分割，拆分得到的字符串放入字符串数组中
    tempLine = bufLine.split("\t");
    if(row != tempLine.length) {
        //若每一行的元素数不等于行数说明不是方阵
        System.out.println("Not the martrix required!");
        buf.close();
        return false;
    }
    for(j = 0 ;j < row; j++)
        matrix[i][j] = Integer.valueOf(tempLine[j]);
}
buf.close();
```

对于 MagicSquare 的检测需要检测每行每列和两个对角线，我们只需要创建一个长度为 2*row+2 的数组，再遍历二维数组，将每个元素累加到对应的一维数组元素上，完成循环之后再检测一维数组的值是否相等即可。

```
for(i = 0; i < row ; i++)
    for(j = 0 ;j < row; j++){
        check[i] += matrix[i][j];
        check[row+j] += matrix[i][j];
        if(j == i)
            check[row*2] += matrix[i][j];
        if(i + j == row - 1)
            check[row*2+1] += matrix[i][j];
    }
for(i = 1; i < row*2 + 2 ; i++) {
    if(check[0]!=check[i])
        return false;
```

运行结果如下图：

```
case 1: src/1.txt
true
case 2: src/2.txt
true
case 3: src/3.txt
Not the martrix required!
false
case 4: src/4.txt
Invalid numbers !
false
case 5: src/5.txt
Not the martrix required!
false
```

3.1.2 generateMagicSquare()

对于题目中提到的两处错误如下：

第一个错误是由于，当 n 等于偶数时，在执行 n 次循环后，此时标记位有可能来到了最后一行，此时执行 row--就会导致数组标识溢出。

第二个错误是由于 n 为负数作为数组下标，而数组不允许这种行为，因此返回数组下标为负数的错误提示。

如果输入的 n 为偶数，函数运行之后在控制台产生以下输出：

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 12
    at MagicSquare.generateMagicSquare(MagicSquare.java:17)
    at MagicSquare.main(MagicSquare.java:121)
```

如果输入的 n 为负数，函数运行之后在控制台产生以下输出：

```
Exception in thread "main" java.lang.NegativeArraySizeException
    at MagicSquare.generateMagicSquare(MagicSquare.java:11)
    at MagicSquare.main(MagicSquare.java:121)
```

我们依据题意，我们对于 n 为偶数和 n 为负数的情况，我们只需在程序开头完成检测，若为负数或偶数则直接返回即可。

我们还需要在生成完毕整个幻方数组之后，将数组写入文件，该过程可以有 FileWriter 类实现。

源代码如下：

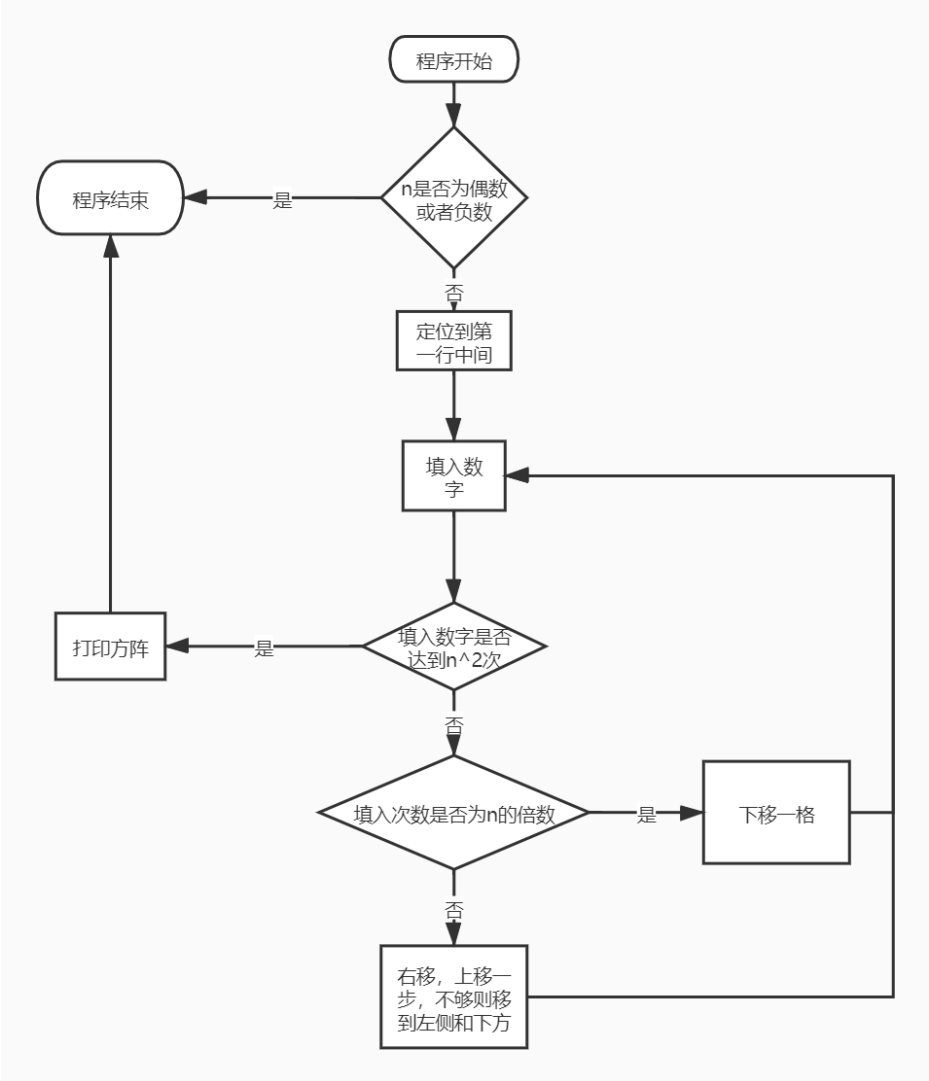
```
public static boolean generateMagicSquare(int n) throws IOException {
    //偶数和负数直接退出
    if(n % 2 == 0 || n < 0)
        return false;
    int magic[][] = new int[n][n];
    //初始化为第一排中间
    int row = 0, col = n / 2, i, j, square = n * n;
    for (i = 1; i <= square; i++) {
        magic[row][col] = i;
        if (i % n == 0)//填完n个数字就下移一行
            row++;
        else {
            if (row == 0)//若在行尾则转向第一行
                row = n - 1;
            else//不在行尾则去下一行
                row--;
            if (col == (n - 1))//若在列尾则转向第一列
                col = 0;
            else//否则去向下一列
                col++;
        }
    }
    FileWriter fw = new FileWriter("src/6.txt",false);
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++)
            fw.write(magic[i][j] + "\t");  //逐个读入数字
        fw.write("\n");//读完一行就换行
    }
    fw.close();
    return true;
}
```

在输入为 7 时，生成的 6.txt 文件如下：

30	39	48	1	10	19	28
38	47	7	9	18	27	29
46	6	8	17	26	35	37
5	14	16	25	34	36	45
13	15	24	33	42	44	4
21	23	32	41	43	3	12
22	31	40	49	2	11	20

因此我们的写入文件无误。

此外，依据函数的行为，我们可以给出 generateMagicSquare 函数的流程图入下所示：

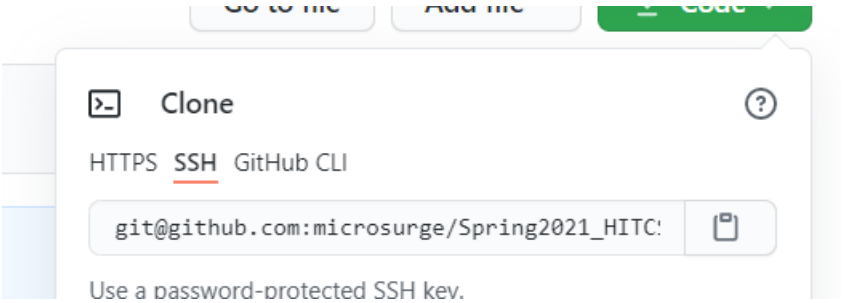


3.2 Turtle Graphics

我们需要根据文件的描述迅速学习关于 Turtle 类的调用，并且使用其提供的一些基本方法完成题目，在此过程中我们需要掌握关于一些数据结构例如 List 或者 Set 的基本用法和一些具体实现的思想。此外，还需要回顾一些三角函数的知识和学习关于 Math 库中对于一些数学运算的调用方法

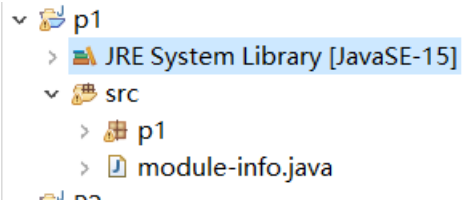
3.2.1 Problem 1: Clone and import

我们找到老师的 github 仓库，然后利用 ssh 进行 git clone：




```
86134@HELLOLULLABY MINGW64 /d/GitRepository
$ git clone git@github.com:microsurge/Spring2021_HITCS_SC_Lab1.git
Cloning into 'Spring2021_HITCS_SC_Lab1'...
remote: Enumerating objects: 89, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 89 (delta 0), reused 0 (delta 0), pack-reused 85
Receiving objects: 100% (89/89), 571.32 KiB | 589.00 KiB/s, done.
Resolving deltas: 100% (24/24), done.
```

然后进入 ecli 中导入对应的文件路径即可。



3.2.2 Problem 3: Turtle graphics and drawSquare

绘制一个正方形的工作相对简单，只需要利用 forward 函数行走然后利用 turn 函数旋转 90 度，循环四次即可，代码如下：

```
public static void drawSquare(Turtle turtle, int sideLength) {
    //throw new RuntimeException("implement me!");
    turtle.forward(sideLength);
    turtle.turn(90);
    turtle.forward(sideLength);
    turtle.turn(90);
    turtle.forward(sideLength);
    turtle.turn(90);
    turtle.forward(sideLength);
    turtle.turn(90);
}
```

3.2.3 Problem 5: Drawing polygons

先实现 calculateRegularPolygonAngle 方法，然后利用次法实现画正多边形，只需循环 sides 次类似于画正方形的过程即可。

```
public static double calculateRegularPolygonAngle(int sides) {
    //throw new RuntimeException("implement me!");
    return (sides - 2)*180/(double)sides;
}
```

```
public static void drawRegularPolygon(Turtle turtle, int sides, int sideLength) {
    //throw new RuntimeException("implement me!");
    for(int i = 0; i < sides ; i++) {
        turtle.forward(sideLength);
        turtle.turn(calculateRegularPolygonAngle(sides));
    }
}
```

3.2.4 Problem 6: Calculating Bearings

我们可以把当前点（currentX，currentY）到（targetX，targetY）看做一个向量（target-currentX，targetY-currentY），Math 类提供了将向量通过反三角函数转换为弧度制角度（-pi~pi）的函数 Math.atan2。

然后我们将其转换为以 x 轴正坐标为轴的极坐标角度（0~360），由此得到了目标角度，用该目标角度减去起始角度，若结果为负数，再加上 360 即为所求，代码如下：

```
public static double calculateBearingToPoint(double currentBearing, int currentX, int currentY,
        int targetX, int targetY) {
        //throw new RuntimeException("implement me!");
        if(currentX == targetX && currentY == targetY)
            return 0;
        //计算最终角度
        double tarang =90 - Math.atan2(targetY - currentY, targetX - currentX)*180/Math.PI;
        if(tarang < 0)
            tarang += 360;
        //最终角度和起始角度相减即为调整量
        double adjustment = tarang -currentBearing;
        if(adjustment < 0)
            adjustment += 360;
        return adjustment;
}
```

3.2.5 Problem 7: Convex Hulls

题目要求寻找到一个凸包，我们可以借助上一问的方法。

显然，当点集的元素个数小于 4 时则该集合就是一个凸包。

```
if(points.size() < 4)
    return points;
```

除此之外，首先，我们寻找到点集中最左的点（x 坐标最小的点），并将其加入凸包集中。

```
for(Point p : points)//遍历集合
    if(p.x() < leftmost.x() || p.x() == leftmost.x() && p.y() < leftmost.y())
        leftmost = p;
//将得到的点加入凸包集合并设为当前到达的点
ans.add(leftmost);
current = leftmost;
```

然后设置初始角度为 0，currentPoint 设置为该点，遍历除当前点外所有的点，寻找使偏转角最小（即 calculateBearingPoint 输出最小）的点，将其加入凸包集，再以新找到的点为 currentPoint，重复上述步骤，直到重新找到第一个点，此时输出凸包集即为所求。

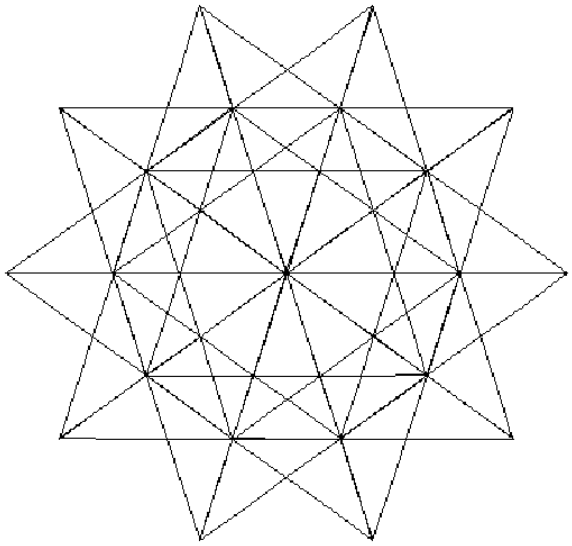
对于其中涉及到的相同转角的取舍，每次遇到一个和当前标记的最小偏转角角度相同的点，我们就选择两个点中距离 currentPoint 中最远的那个点即可。

源代码实现如下：

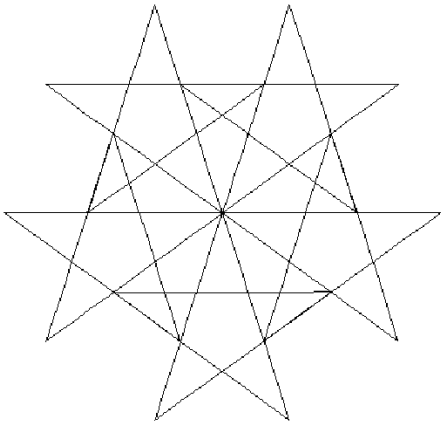
```
double angle = 0;
do {
    double min_adjustment = Double.MAX_VALUE;
    for(Point p : points) {
        //跳过当前到达的点
        if(p.equals(current))
            continue;
        //寻找最小偏转角的点存放在temp中
        double adjustment = calculateBearingToPoint(angle,(int)current.x(), (int)current.y(), (int)p.x(), (int)p.y());
        if(adjustment <min_adjustment ) {
            min_adjustment = adjustment;
            temp = p;
        }
        //若有两个及以上最小偏转角的点则选择离当前到达的点最远的那个
        else if(adjustment == min_adjustment && (p.x()<temp.x()&&temp.x()<current.x()||current.x()<temp.x()&&temp.x()<p.x()))
                temp = p;
    }
    current = temp;
    //调整当前角度
    angle = (angle + min_adjustment)%360;
    ans.add(current);
}while(!current.equals(leftmost));
return ans;
```

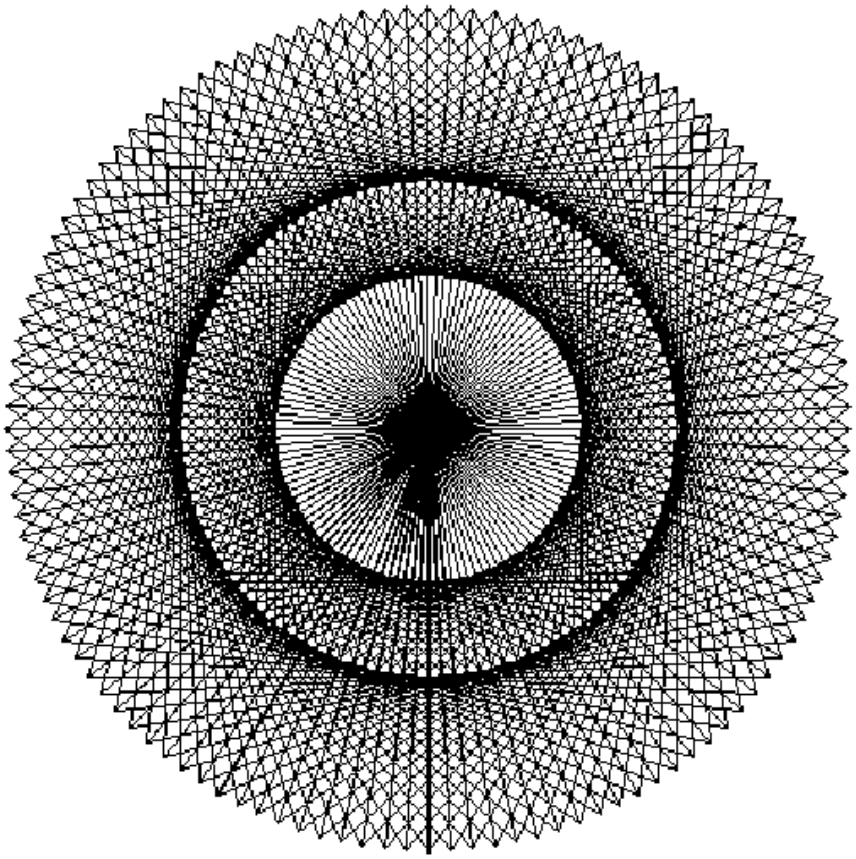
3.2.6 Problem 8: Personal art

首先，我实现了一个简单的五角星，然后将旋转 36 度，循环 10 次实现就可以得到下面这个图形：



选取不同的间隔就能得到不同的效果：



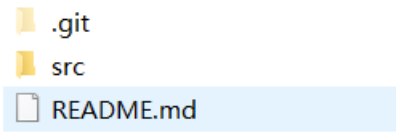


源代码如下：

```java
public static void drawPersonalArt(Turtle turtle) {
    //throw new RuntimeException("implement me!");
    turtle.turn(90);
    for(int j = 1; j <=5 ;j++) {
        for(int i = 0; i < 5 ; i++) {
            turtle.forward(200);
            turtle.turn(144);
        }
        turtle.turn(72);
    }
    return;
}
```

3.2.7 Submitting

首先，我们按照老师要求的结构将代码存放到我们 git clone 的文件夹



其次，使用 git add . 将当前目录的所有文件加入缓冲提交区，然后，我们使用 git commit -m “commit_P2”将缓冲区的文件提交至本地仓库并添加修改注释“commit_P2”

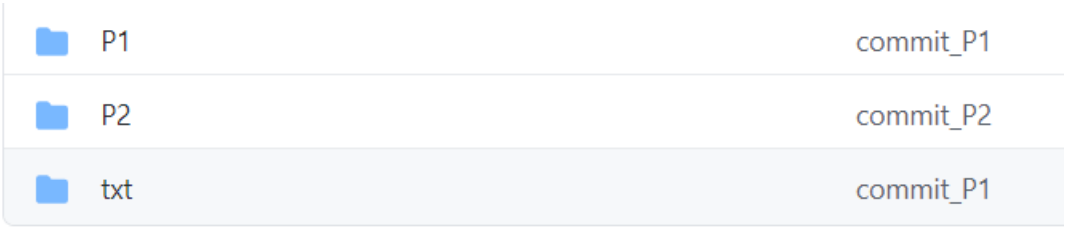
```
$ git add .
```

```
$ git commit -m "commit_P2"
[master e665816] commit_P2
```

然后使用 git push origin master 将本地仓库的文件提交至远程仓库

```
$ git push origin master
```

此时，在 github 看发现 P2 文件夹已经在其中。



3.3 Social Network

我们需要设计一个数据结构 graph 来表示 Person 之间的关系和记录所有的 Person，此外，在计算两个人的社交距离时，我们可以采用 Floyd 算法进行实现。

3.3.1 设计/实现 FriendshipGraph 类

使用一个二维数组存放 person 之间的关系，使用 List<Person>存放所有的 person 成员。

```
final int arrSize = 100;
private int [][] relationShip = new int [arrSize][arrSize];
private List<Person> nameTable= new ArrayList<Person>();
```

对于 addVertex 模块，我们需要检测是否是曾经出现过的 Person，因此我们在添加进 List 之前需要先遍历 List 以寻找是否有同名成员，如果有则直接退出。

```
public boolean addVertex(Person guy) {
    //检测是否已经存在
    for(int i = 0; i < nameTable.size() ; i ++)
        if(guy.getPersonName() == nameTable.get(i).getPersonName()) {
            System.out.println("Repeated person!");
            System.exit(0);
        }
    nameTable.add(guy);
    return true;
}
```

AddEdge 模块较为简单，直接对数组进行修改即可。

```
public void addEdge(Person x, Person y) {
    int x_index = nameTable.indexOf(x);
    int y_index = nameTable.indexOf(y);
    relationShip[x_index][y_index]=1;
}
```

对于 getDistance 模块，可以计算 Floyd 矩阵然后求解

```
public int getDistance(Person x, Person y) {
    if(x.getPersonName() == y.getPersonName())
        return 0;
    int x_index = nameTable.indexOf(x);
    int y_index = nameTable.indexOf(y);
    int[][] temp = new int [nameTable.size()][nameTable.size()];
    //初始化数组
    for(int i = 0; i < nameTable.size(); i++) {
        for(int j = 0; j < nameTable.size(); j++) {
            if(relationShip[i][j]==0)
                temp[i][j] = Integer.MAX_VALUE;
            else
                temp[i][j] = 1;
        }
    }
    //Floyd
    for(int k = 0; k < nameTable.size(); k++)
        for(int i = 0; i < nameTable.size(); i++)
            for(int j = 0; j < nameTable.size(); j++)
            {
                if(temp[i][k] < Integer.MAX_VALUE && temp[k][j] < Integer.MAX_VALUE)
                    if(temp[i][j] == Integer.MAX_VALUE)
                        temp[i][j] = temp[i][k] + temp[k][j];
                    else if(temp[i][j] > temp[i][k] + temp[k][j])
                        temp[i][j] = temp[i][k] + temp[k][j];
            }


    if(temp[x_index][y_index]==Integer.MAX_VALUE)
        return -1;
    return temp[x_index][y_index];
}
```

3.3.2 设计/实现 Person 类

Person 类较为简单，只需要存储姓名即可，注意将姓名设为 Private，并且提供一个返回姓名的方法。

```
public class Person {
    private String personName;
    public Person (String name) {
        personName = name;
    }

    public String getPersonName() {
        return personName;
    }
```

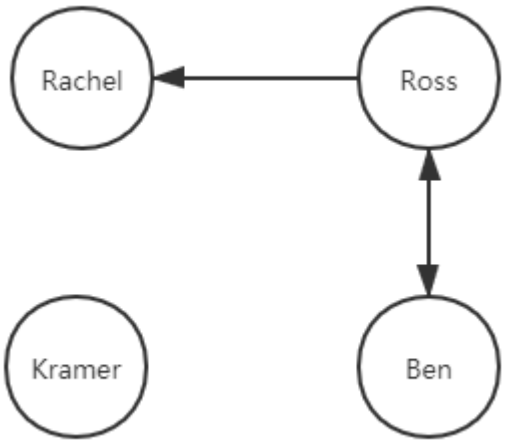
3.3.3 设计/实现客户端代码 main()

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    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));
    //should print 0
    System.out.println(graph.getDistance(Rachel, Kramer));
    //should print -1
}
```

若我们删除 Rachel->Ross,那么他们之间的关系图变为：



此时，函数会输出

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

若重复加入 Ross，会输出：

```
Repeated person!
```

3.3.4 设计/实现测试用例

利用 Junit 模块进行测试，注意 ClassPath 的添加，出现问题需要删除 module-info.java 模块。

```
class FriendshipGraphTest {

    @Test
    void test() {
        fail("Not yet implemented");
    }
    @Test
    public void TestFriendshipGraph() {
        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);

        assertEquals(true,graph.addVertex(Rachel));
        assertEquals(true,graph.addVertex(Ross));
        assertEquals(true,graph.addVertex(Ben));
        assertEquals(true,graph.addVertex(Kramer));
        assertEquals(1,graph.getDistance(Rachel, Ross));
        assertEquals(1,graph.getDistance(Rachel, Ben));
        assertEquals(1,graph.getDistance(Rachel, Rachel));
        assertEquals(1,graph.getDistance(Rachel, Kramer));
    }

}
```

4 实验进度记录

日期	时间段	任务	实际完成情况
2021-05-14	19:00-22:00	学习 git 使用，java 基本语法等知识	按计划完成
2021-05-16	18:30-21:00	学习文件操作，实现 P1	按时完成
2021-05-17	18:00-22:00	完成 P2，P3	按时完成
2021-05-22	18:50-21:30	完成实验报告	按时完成

2021-05-23	19:40-22:00	学习 Junit 等内容	按时完成

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
Git 无法完成 push	未将分支从 main 改为 master，改变后即可
无法实现凸包算法	查阅资料发现对 Set 操作不当，改变后即可实现

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

了解了初步的 java 工作的原理，学习了关于 git 的版本控制的知识。

我对于如何进行模块化编程有了更深的体会，最鲜明的例子是 P2 的凸包算法，如果直接写凸包算法也许我会难以完成，但是在从小模块开始实现之后我就可以独自实现凸包算法，这其中的思想令我受益匪浅。

6.2 针对以下方面的感受

我初步理解了软件构造的流程，以及如何进行多人协作开发，激发我的代码兴趣。