哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2022 年春季学期

计算学部《软件构造》课程

Lab 1 实验报告

姓名	李世轩
学号	120L022109
班号	2003007
电子邮件	1146887979@qq.com
手机号码	15234117960

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

简要陈述你配置本次实验所需开发、测试、运行环境的过程，必要时可以给出屏幕截图。

开发环境：IntelliJ IDEA

运行环境：IntelliJ IDEA

测试环境：Junit 4.13.1

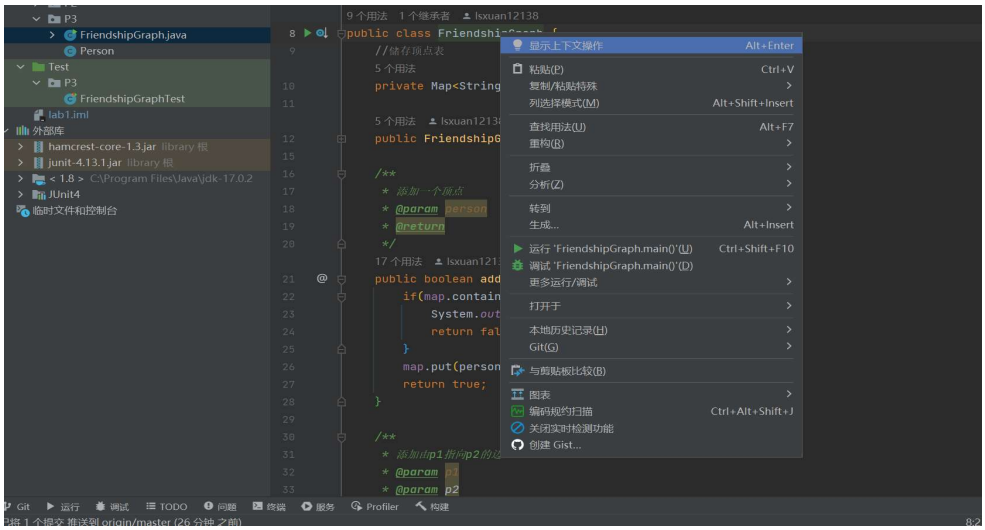
2.1 IntelliJ IDEA

按照提示安装即可

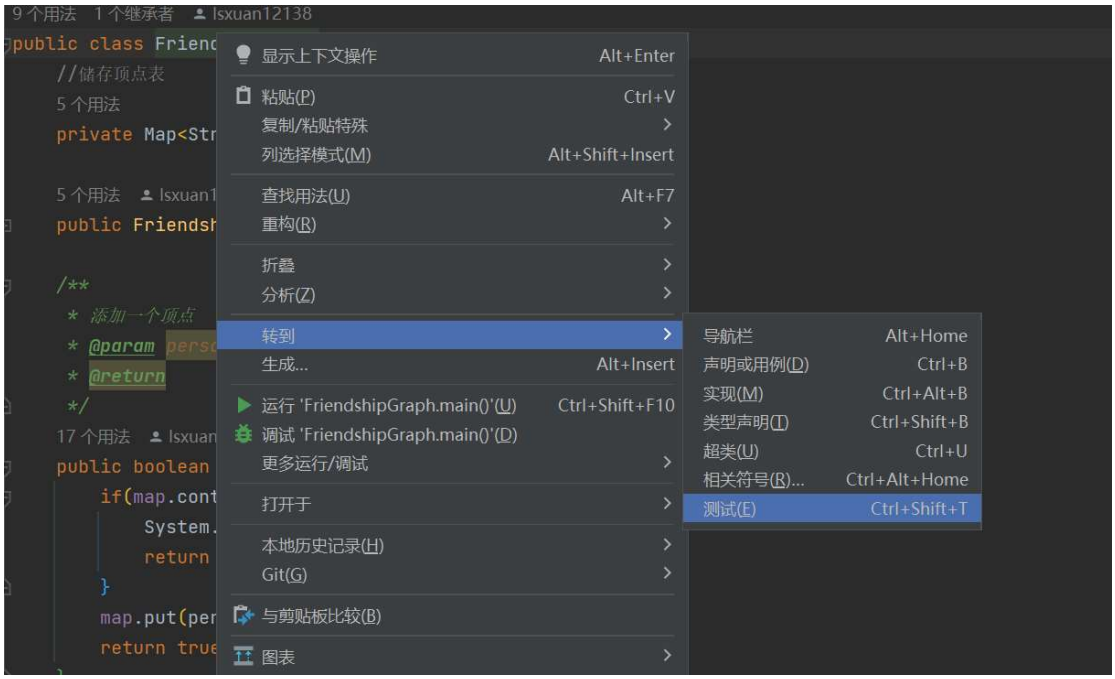
2.2 Junit 配置过程

因为 IntelliJ IDEA 内置了 Junit 的库，直接使用即可

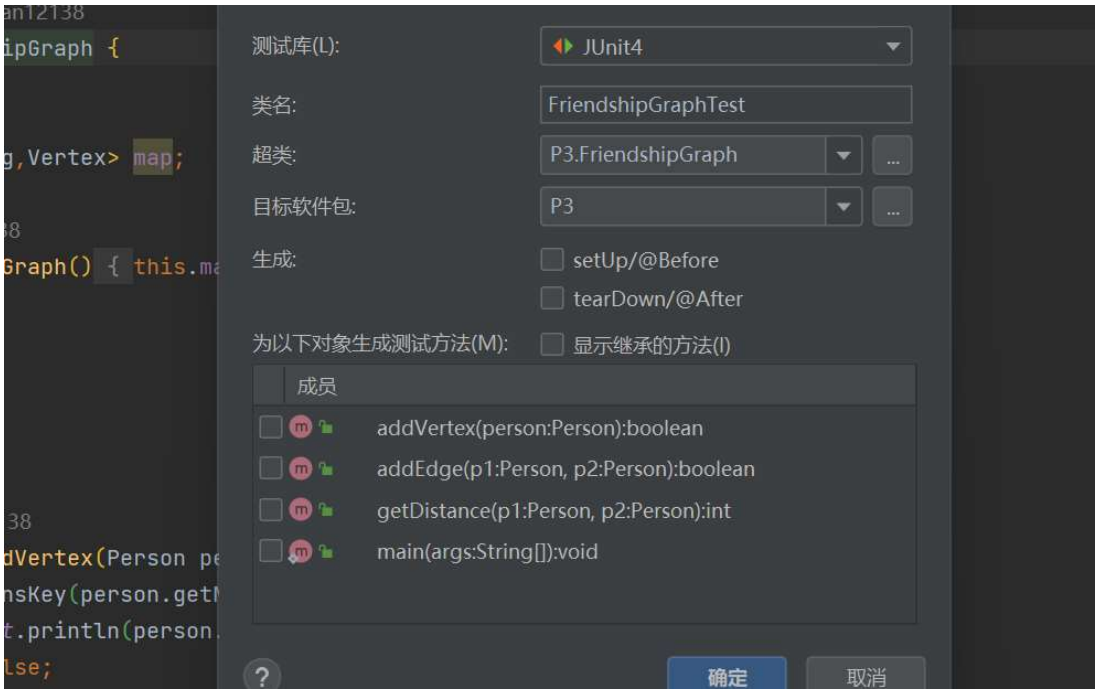
-先在某一个需要测试的类中，对类名右键



-在右键菜单中选择->转到->测试



-点击创建新测试



-在弹出的窗口中选择要测试的方法，即可生成测试与方法。

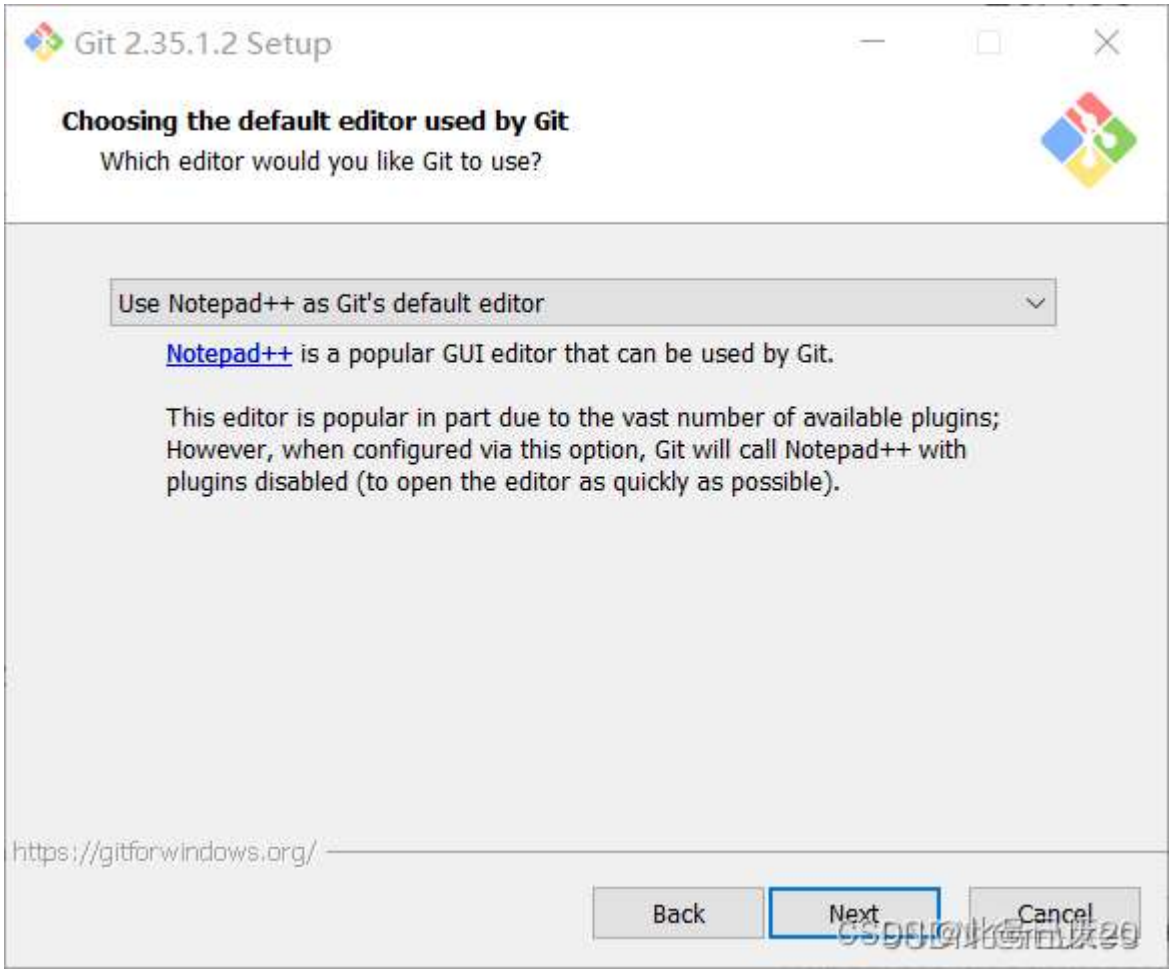
2.3 Git 的配置

安装过程中参考了一篇博客

https://blog.csdn.net/mukes/article/details/115693833

但是在安装过程中遇到一个问题

在其博客里 *2.2.5 选择 Git 默认编辑器*中，我选择 Sublime Textwe 默认浏览器后。但是在完成编辑器的安装后，且配置完成环境变量后 Git 的安装界面中的 Next 并没有如博主一样亮起，无法点击。



解决办法是，若 next 未亮起，再多按几次 Back，Next，返回前一界面再到这个界面，这个 Next 按钮就会亮起，然后就可以继续正常安装。

在这里给出你的 GitHub Lab1 仓库的 URL 地址。

https://github.com/ComputerScienceHIT/HIT-Lab1-120L022109

3 实验过程

3.1 Magic Squares

幻方为一个每个横行，每个纵行，每个对角线上元素的和都相同的二位矩阵
该任务主要涉及两个问题，幻方的判断和生成。

3.1.1 isLegalMagicSquare()

这个方法分为两个阶段
一个是从文件中读入幻方，一个是对读入的幻方进行判断。
首先对幻方进行读入，在这个过程中涉及对各种非法情况的处理
大致思路如下
先读入第一行，若其中不含有”\t”，说明它不符合对幻方的要求则直接返回，且输出报错信息。
将读入的字符串 split("\t")在第一行中我们可以得到该幻方的列数，以及正确的列数信息。
接下来按照行列数进行循环，每次读入一行，并将其处理放入准备的整数矩阵中，其中对一些非法信息进行处理
若在循环中读到 line 为 null 而循环还没有结束，说明文件中的行数比列数少

```
line = bufferedReader.readLine();
//若文件中还有元素行数比列数少
if (line == null) {
    System.out.println("请保持输入行列数相等");
    return false;
}
array = line.split( regex: "\t");
```

若分割后的数组长度与第一行元素个数不同，结束

```
if (cnt != array.length) {
    System.out.println("请保持每行元素个数相等，出现问题的为第"+(i+1)+"行，应以\\t分割元素");
    return false;
}
```

若处理后得到非正数，则结束

```
if (temp <= 0) {
    System.out.println("请输入正整数，零和负数都是不合法的,出现问题的为第 "+(i+1)+" 行,第 "+(j+1)+" 列");
    return false;
}
```

若 parseInt 抛出 NumberFormatException 异常，说明输入的元素格式不符合整数

```
} catch (NumberFormatException e) {
    //e.printStackTrace();
    System.out.println("输入格式错误：出现问题的为第 "+(i+1)+" 行,第 "+(j+1)+" 列"+e.getMessage());
    return false;
}
```

接下来对已经读入的幻方进行判断

先求得第一行元素的和

分别求取每行每列及对角线元素的和，若其与之前的值不同，返回 false

3.1.2 generateMagicSquare()

方法思路：

先初始化变量 row = 0, col = n / 2, i, j, square = n * n;

/共向矩阵中填写 n*n 个数，从 1 到 n*n

让 row 从 n-1 到 0 从大到小不断循环，且初值为 0，让 col 从 0 到 n-1 从小到大不断循环，且初值为 n/2；

每放置 n 个数，row+1，防止覆盖前面已放置的数

对该函数做扩展：

(1) 将产生的 magic square 写入文件\src\P1\txt\6.txt 中；

使用文件输出流，将产生的幻方写入文件

```
try {
    fileWriter= new BufferedWriter(new FileWriter( fileName: "src/P1/txt/6.txt"));
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            fileWriter.write( str: magic[i][j] + "\t");
        }
        fileWriter.write( c: '\n');
    }
```

(2) 当输入的 n 不合法时（n 为偶数、n 为负数等），不要该函数抛出异常并非法退出，而是提示错误并“优雅的”退出——函数输出 false 结束

将产生幻方的部分用 try{}catch(){}包裹

Catch 对应的异常

```
}catch (NegativeArraySizeException e){
    System.out.println("数组尺寸不应为负数，请确认输入n为正整数");
    return false;
}
catch (ArrayIndexOutOfBoundsException e){
    System.out.println("请确认输入n为奇数");
    return false;
}
```

3.2 Turtle Graphics

使用 Java 简单实现 Logo 语言中的 Turtle 图形。

3.2.1 Problem 1: Clone and import

如何从 GitHub 获取该任务的代码

在 Git Bash 中输入

git clone "仓库地址"

即可从 GitHub

在本地创建 git 仓库

进入某一目录后，使用

git init

即可在本地创建 git 仓库

使用 git 管理本地开发。

添加到缓存区

git add <filename>

git add *

提交

git commit -m “代码提交信息”

3.2.2 Problem 3: Turtle graphics and `drawSquare`

该题为画一个正方形

向前 sideLength 后应该转 90 度

循环 4 次

```
public static void drawSquare(Turtle turtle, int sideLength) {
    turtle.forward(sideLength);
    turtle.turn( degrees: 90);
    turtle.forward(sideLength);
    turtle.turn( degrees: 90);
    turtle.forward(sideLength);
    turtle.turn( degrees: 90);
    turtle.forward(sideLength);
}
```

3.2.3 Problem 5: Drawing polygons

该题意为画一个正多边形，边数为 sides，每个边的长度为 sideLength

首先计算内角，使用方法

angle = calculateRegularPolygonAngle(sides)

其实现思路为

我们知道在转过 sides 个弯后，应该转过 360 度

所以内角的补角应该为 360/sides

所以内角大小为 180-360/sides

在得到内角角度后

开始画正多边形

循环 sides 次，每次转弯 angle 度，向前走 sideLength

```
lsxuan12138
public static void drawRegularPolygon(Turtle turtle, int sides, int sideLength) {
    double angle = calculateRegularPolygonAngle(sides);
    turtle.forward(sideLength);
    for(int i = 2;i<=sides;i++){
        turtle.turn( degrees: 180-angle);
        turtle.forward(sideLength);
    }
}
```

3.2.4 Problem 6: Calculating Bearings

public static List<Double> calculateBearings(List<Integer> xCoords, List<Integer> yCoords)

该题意为在若干个点之间按顺序移动，求每两个点之间应该转动的角度为多少

其中调用的子方法为 double calculateBearingToPoint(double currentBearing, int currentX, int currentY, int targetX, int targetY)

子方法实现的思路为

先计算 current 到 target 直接向量的方向角，相对于正北，以顺时针为正

然后减去当前角度 angle 即可

定义 angle 为当前角度，初始值为 0，delta 为变化角度

先获取点的个数，确定循环次数为个数-1，接下来开始循环，从第一个点开始计算

计算当前点到下一个点的角度，将其装入结果

angle+=deltaDegree; angle%=360;获取转过角度后的当前角度

循环完成后返回结果

```
1 个用法  lsxuan12138 *
public static List<Double> calculateBearings(List<Integer> xCoords, List<Integer> yCoords) {
    List<Double> result = new ArrayList<Double>();
    int cnt = xCoords.size();
    double angle = 0;
    double deltaDegree;
    for (int i = 0; i < cnt-1; i++) {
        deltaDegree=calculateBearingToPoint(angle,xCoords.get(i),yCoords.get(i),xCoords.get(i+1),yCoords.get(
        result.add(deltaDegree);
        angle+=deltaDegree;
        angle%=360;
    }
    return result;
}
```

3.2.5 Problem 7: Convex Hulls

该题为计算若干点的闭包

使用 Graham(格拉翰)扫描法

思路为

1 把所有点放在二维坐标系中，则纵坐标最小的点一定是凸包上的点，记为 P0。

2 把所有点的坐标平移一下，使 P0 作为原点。

3 计算各个点相对于 P0 的幅角 α ，按从小到大的顺序对各个点排序。当 α 相同时，距离 P0 比较近的排在前面。假设得到的结果为 P1，P2，P3，P4，P5，P6，P7，P8。我们由几何知识可以知道，结果中第一个点 P1 和最后一个点 P8 一定是凸包上的点。

4 以上，我们已经知道了凸包上的第一个点 P0 和第二个点 P1，我们把它们放在栈里面。现在从步骤 3 求得的那个结果里，把 P1 后面的那个点拿出来做当前点，即 P2。接下来开始找第三个点。

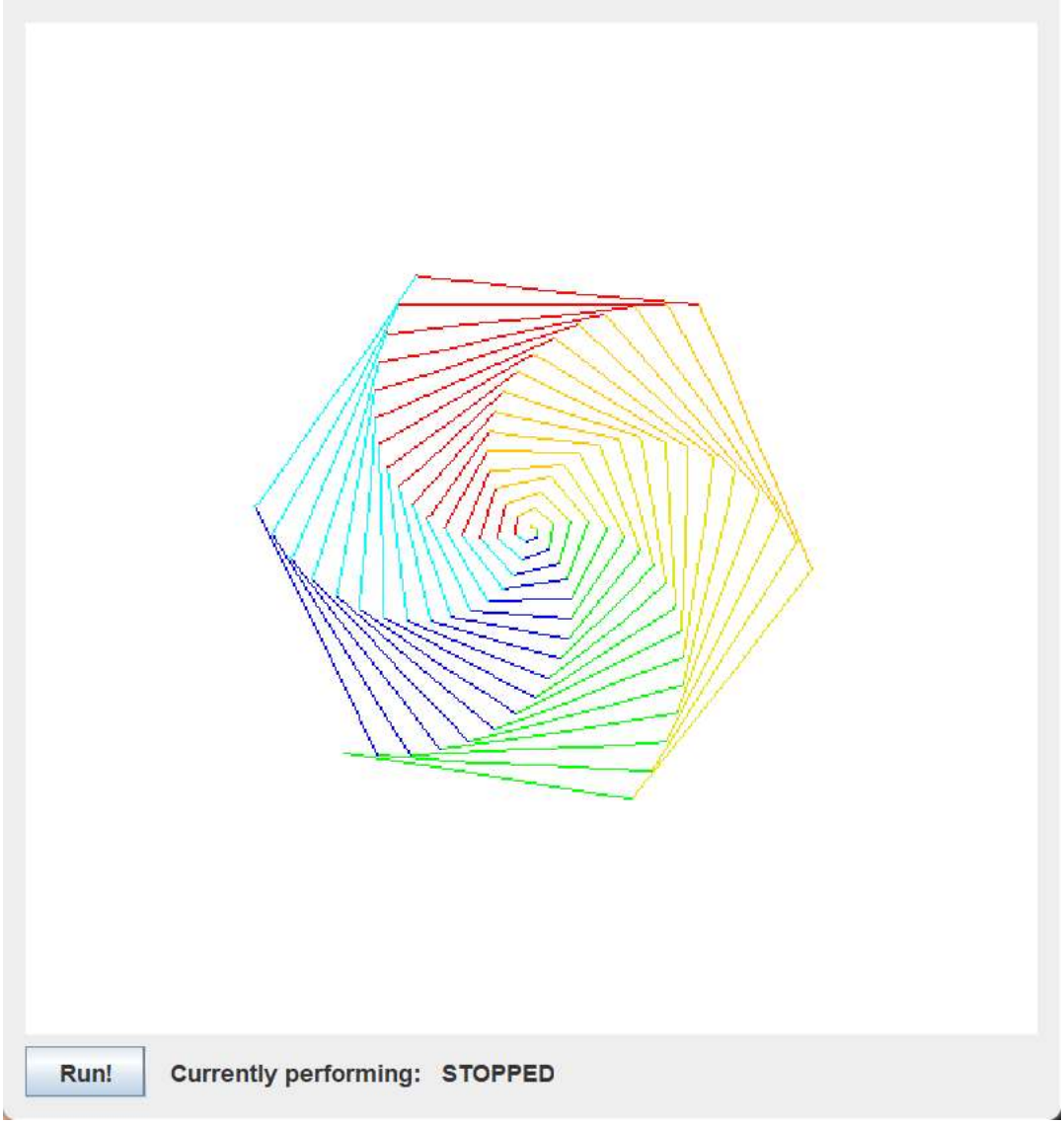
5 连接栈最上面的两个元素，得到直线 L。看当前点是在直线 L 的右边还是左边。如果在直线的右边就执行步骤 6；如果在直线上，或者在直线的左边就执行步骤 7。

6 如果在右边，则栈顶的那个元素不是凸包上的点，把栈顶元素出栈。执行步骤 5。

7 当前点是凸包上的点，把它压入栈，执行步骤 8。

8 检查当前的点是不是步骤 3 那个结果的最后一个元素。是最后一个元素的话就结束。如果不是的话就把当前点后面那个点做当前点，返回步骤 5。

3.2.6 Problem 8: Personal art




```
public static void drawPersonalArt(Turtle turtle) {
    String[] colors={"RED","ORANGE","YELLOW","GREEN","BLUE","CYAN"};
    int sides = 6;
    for(int i=0;i<100;i++){
        turtle.color(PenColor.valueOf(colors[i%sides]));
        turtle.forward( units: i*3/sides+i);
        turtle.turn( degrees: 360/sides+1);
    }

}
```

3.2.7 Submitting

使用指令

git push origin master

其中的 master 可以换为任何其他分支

3.3 Social Network

使用有向图模拟社交网络，其中结点为人。

实现对有向图添加节点，添加边，获取两个节点间最短路径长度的方法

3.3.1 设计/实现 `FriendshipGraph` 类

构建两个子类

```
class Vertex{
    4 个用法
    private Person person;
    3 个用法
    private Edge firstEdge;
```

```
class Edge{
    3 个用法
    private Person data;
    3 个用法
    private Edge next;
```

使用 HashMap<String,Vertex>来存储图。

3.3.1.1 addVertex(Person person)方法实现

向 map 中添加键值对< person.getName(), new Vertex(person)>

3.3.1.2 addEdge(Person p1,Person p2)方法实现

先检查顶点的 firstEdge 是否为 null，若是，直接设置 vertex.setFirstEdge(new Edge(p2));

若不是，沿着链表向后找到末尾 edge.setNext(new Edge(p2));

3.3.1.3 getDistance(Person p1,Person p2)方法实现

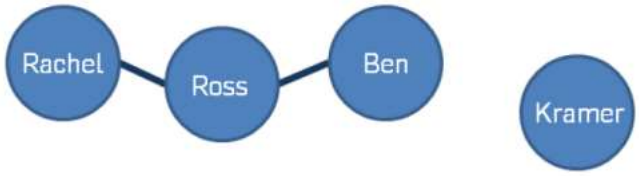
以图的广度优先遍历为基础
以 cnt 记录访问的深度，即最短路径长度
在遍历过程加入空节点作为标志，表示访问完一层的节点，使 cnt 加一
若遍历过程未返回
说明 p2 与 p1 未连接，返回-1

3.3.2 设计/实现 Person 类

因为当前需要的唯一变量为名字
定义成员变量 private String name;
实现相应构造方法，Setter and Getter

3.3.3 设计/实现客户端代码 main()

使用使用指导书中的代码。构造这样的图



并根据其中代码进行测试

3.3.4 设计/实现测试用例

3.3.4.1 测试 testAddVertex

测试正常添加节点的返回值是否正确
测试当节点重复添加时的反应

```
assertEquals( expected: true,graph.addVertex(rachel));
assertEquals( expected: true,graph.addVertex(ross));
assertEquals( expected: true,graph.addVertex(ben));
assertEquals( expected: true,graph.addVertex(kramer));

//当节点重复的情况下，应返回false，并输出提示信息
assertEquals( expected: false,graph.addVertex(kramer));
assertEquals( expected: kramer.getName()+"已存在", bytes.toString().trim());
```

3.3.4.2 测试 testAddEdge

测试正常添加边的返回值是否正确
测试当 p1，p2 相同时，程序的反应
测试重复添加边时的反应

```
assertEquals( expected: true,graph.addEdge(rachel, ross));
assertEquals( expected: true,graph.addEdge(ross, rachel));
assertEquals( expected: true,graph.addEdge(ross, ben));
assertEquals( expected: true,graph.addEdge(ben, ross));

assertEquals( expected: false,graph.addEdge(rachel,rachel));
assertEquals( expected: "你不能添加自己为好友",bytes.toString().split( regex: "\r\n")[0]);

assertEquals( expected: false,graph.addEdge(rachel,ross));
assertEquals( expected: "你们已经是好友了，无需再次添加",bytes.toString().split( regex: "\r\n")[1]);

addEdgeException.expect(IllegalArgumentException.class);
graph.addEdge(null,new Person());
```

3.3.4.3 测试 testGetDistance

测试当两个点间只有一条有向边的反应
测试两个点间有正常的两条边的结果是否正确

```
//rachel和ross只有单向的关系
graph.addEdge(ross, rachel);
graph.addEdge(ross, ben);
graph.addEdge(ben, ross);

assertEquals( expected: -1,graph.getDistance(rachel, ross));
//should print -1
assertEquals( expected: -1,graph.getDistance(rachel, ben));
//should print -1
assertEquals( expected: 0,graph.getDistance(rachel, rachel));
//should print 0
assertEquals( expected: -1,graph.getDistance(rachel, kramer));
//should print -1

graph.addEdge(rachel, ross);
assertEquals( expected: 1,graph.getDistance(rachel, ross));
//should print 1
assertEquals( expected: 2,graph.getDistance(rachel, ben));
//should print 2
assertEquals( expected: 0,graph.getDistance(rachel, rachel));
//should print 0
assertEquals( expected: -1,graph.getDistance(rachel, kramer));
//should print -1
```

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
2022-04-27	18:30-19:30	编写问题 1 的 `isLegalMagicSquare` 函数并进行测试	按计划完成
2022-04-27	17:30-20:00	完成问题 2	按计划完成
2022-04-27	20:30-21:00	完成问题 2 中 1，3，5	完成
2022-04-27	21:30-22:30	完成问题 2 中 6	完成
2022-04-28	14:30-15:30	完成问题 2 中 7	未完成，延长
	15:30-17:30	完成问题 2 中 7	未完成，延长
	18:00-19:00:	完成问题 2 中 7	完成
2022-4-28	20:00-20:30	完成问题 3 中两个类基本构建	完成
2022-4-28	20:30-21:00	完成问题 3 中 addVertex	完成
2022-4-28	21:00-21:30	完成问题 3 中 addEdge	完成
2022-4-28	21:30-22:30	完成问题 3 中 getDistance	完成
2022-4-29	14:00-15:30	完成问题 3 中 Test 部分，并修改 bug	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
求凸包的算法 gift-wrapping algorithm 没看懂	改用 Graham(格拉翰)扫描法
Graham(格拉翰)扫描法实现过程中涉及到求点的极坐标并排序	先用 Map 存储点 Point 对应的极角 在根据冒泡排序去排序并放到一个 List 中
修改你的 FriendshipGraph 类和 Person 类，使该约束能够始终被满足（意即：一旦该条件被违反，提示出错并结束程序运行）。	在 Person 中添加一个静态成员变量 public static Set<String> namePool 来记录所有出现过的人名，当再次 new Person 对象时，先进行坚持，若已经出现过，直接结束程序运行

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（必答）

对一个方法的测试一定要做的全面，否则在之后其他方法中调用它出问题时，debug 会非常困难

6.2 针对以下方面的感受（必答）

(1) Java 编程语言是否对你的口味？与你熟悉的其他编程语言相比，Java 有何优势和不足？

因为我在高考结束后自学 java，所以其实 Java 是我接触的第一个编程语言，所以很对我口味

与其他编程语言相比，Java 运行在 JVM 上，所以 java 程序的可移植性非常强，可以说是即插即用。Java 本身提供一个非常大的类库，我们常用的方法不需要再去找第三方库。Java 完全规避了指针，虽然想要深入理解还是得明白指针的用法，但是程序员不再需要担心内存泄漏等问题

Java 的缺点就是，相对其他语言，因为它在虚拟机上运行，所以处理速度相等慢很多

(2) 关于 Eclipse 或 IntelliJ IDEA，它们作为 IDE 的优势和不足；

IntelliJ IDEA 是一个非常强大的 IDE，他除了基本的代码补全等功能，还可以开发 maven 项目等等大型项目。其调试功能也很强大。还可以在其中配置 Git 等等方便开发的功能

不足：正是由于它的强大，如果过分依赖它，非常不利于程序员的进步

(3) 关于 Git 和 GitHub，是否感受到了它在版本控制方面的价值；

Git 和 Github 可以让我们很方便的看到各个版本的代码变化，以及开发进度

(4) 关于 CMU 和 MIT 的作业，你有何感受；

国外顶尖大学的作业不仅难度较大，而且非常贴近实际，能很好的考察和锻炼程序员的能力

(5) 关于本实验的工作量、难度、deadline；

工作量比较大，难度也比较大，有些需要查阅资料，并且细致的思考

Deadline 比较合理

(6) 关于初接触“软件构造”课程；

软件构造课程让我对软件开发有了一个跟全面的认识，从过去完全在做数学题，或是模仿数据结构到了更高的领域