

**2022年春季学期  
计算学部《软件构造》课程**

**Lab 1实验报告**

|  |  |
| --- | --- |
| 姓名 | 曹煜浩 |
| 学号 | 120L021501 |
| 班号 | 2003012 |
| 电子邮件 | 1072311998@qq.com |
| 手机号码 | 15101283257 |

**目录**

[1 实验目标概述 1](#_Toc97648154)

[2 实验环境配置 1](#_Toc97648155)

[3 实验过程 1](#_Toc97648156)

[3.1 Magic Squares 1](#_Toc97648157)

[3.1.1 isLegalMagicSquare() 1](#_Toc97648158)

[3.1.2 generateMagicSquare() 1](#_Toc97648159)

[3.2 Turtle Graphics 1](#_Toc97648160)

[3.2.1 Problem 1: Clone and import 2](#_Toc97648161)

[3.2.2 Problem 3: Turtle graphics and drawSquare 2](#_Toc97648162)

[3.2.3 Problem 5: Drawing polygons 2](#_Toc97648163)

[3.2.4 Problem 6: Calculating Bearings 2](#_Toc97648164)

[3.2.5 Problem 7: Convex Hulls 2](#_Toc97648165)

[3.2.6 Problem 8: Personal art 2](#_Toc97648166)

[3.2.7 Submitting 2](#_Toc97648167)

[3.3 Social Network 2](#_Toc97648168)

[3.3.1 设计/实现FriendshipGraph类 2](#_Toc97648169)

[3.3.2 设计/实现Person类 2](#_Toc97648170)

[3.3.3 设计/实现客户端代码main() 2](#_Toc97648171)

[3.3.4 设计/实现测试用例 3](#_Toc97648172)

[4 实验进度记录 3](#_Toc97648173)

[5 实验过程中遇到的困难与解决途径 3](#_Toc97648174)

[6 实验过程中收获的经验、教训、感想 3](#_Toc97648175)

[6.1 实验过程中收获的经验和教训（必答） 3](#_Toc97648176)

[6.2 针对以下方面的感受（必答） 3](#_Toc97648177)

# 实验目标概述

本次实验通过求解四个问题，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。

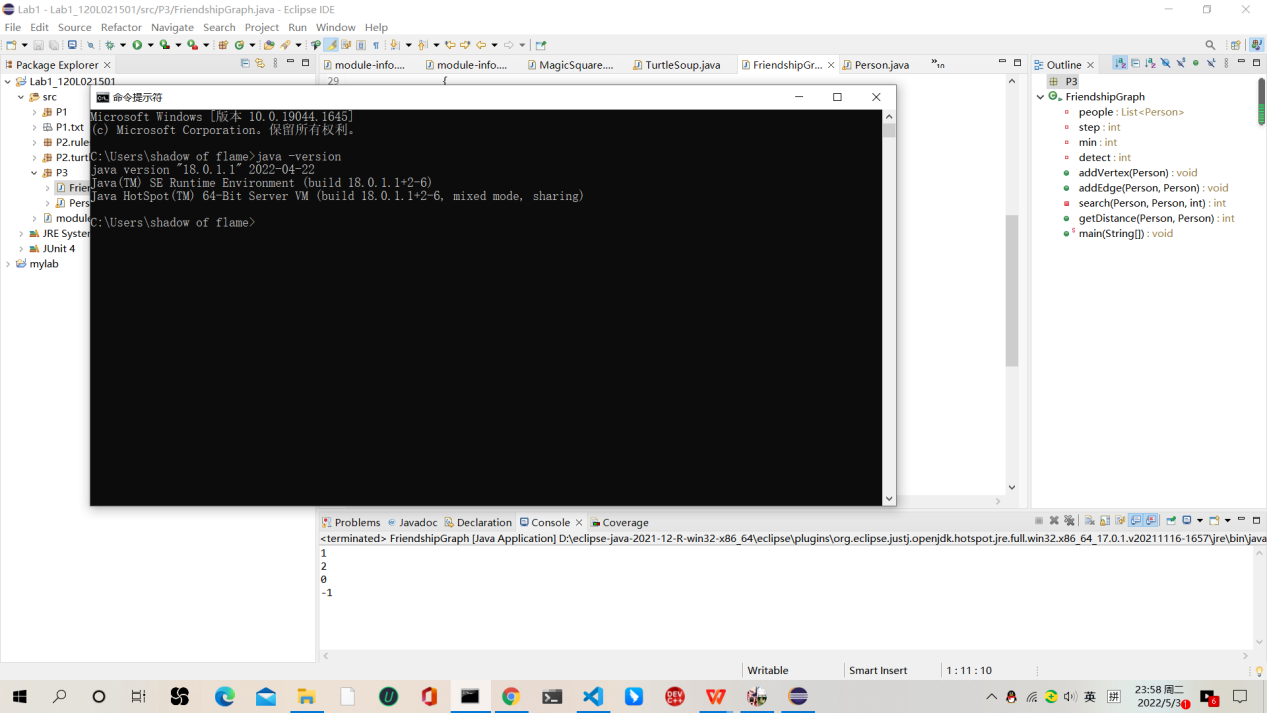
* 基本的Java OO编程
* 基于Eclipse IDE进行Java编程
* 基于JUnit的测试
* 基于Git的代码配置管理

# 实验环境配置

我在运行时遇到过JDK版本过低的问题，解决的方法是设置了高版本的JDK。

在设置过程中无法切换到高版本的JDK，解决方法是重新配置了环境变量。

这是测试JDK版本时的截图



https://github.com/ComputerScienceHIT/Lab1-120L021501

# 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

Magic Squares，主要有两个函数，一个是isLegalMagicSquare判断已有的数字矩阵是否是一个幻方；另一个是利用已有的generateMagicSquare函数，来生成一个奇数行数的幻方。

这里主要面临的问题是文件读取方面遇到了一些问题。

### isLegalMagicSquare()

1. 判断输入的字符是否合法，从文件中读取数字矩阵，采用按行读取的方式，并另外设计了一个函数boolean isConSpeCharaters(String string) 来判断，读取的每一行是否含有除了0-9, ‘\t’, ‘\n’, ‘-‘, ‘.’之外的字符。其判断方式如下

**if** (string.replaceAll("\\d\*\\t\*\\n\*-\*.\*", "").length() == 0) {

**return** **true**;

}

**return** **false**;

其中’\t’是用来分隔数字矩阵所采用的分隔符，

1. 判断行数和列数是否相等，首先我需要进行输入数据的处理，我采用split函数来将空格符号给区分开来，方便后序处理。

templine=in.readLine();

String [] a=templine.split("\t");

String[][] ss;

**int** rows=0,cols;

cols=a.length;

ss = **new** String[cols][cols];

ss[rows]=a;

**while**((templine=in.readLine())!=**null**)

{

**if** (!*isConSpeCharacters*(templine)) {

System.***out***.print("Contains other delimiters! ");

**return** **false**;

}

a=templine.split("\t");

rows++;

ss[rows]=a;

**if**(a.length!=cols)

{

System.***out***.print("wrong\n");

**return** **false**;

}

}

rows++;

**if**(rows!=cols)

{

System.***out***.println("rows != cols\n");

**return** **false**;

}

我先读取一行的数据，将第一行的数据作为列数。然后逐行读取，判断每一行的数据个数是不是和第一行的列数相等。

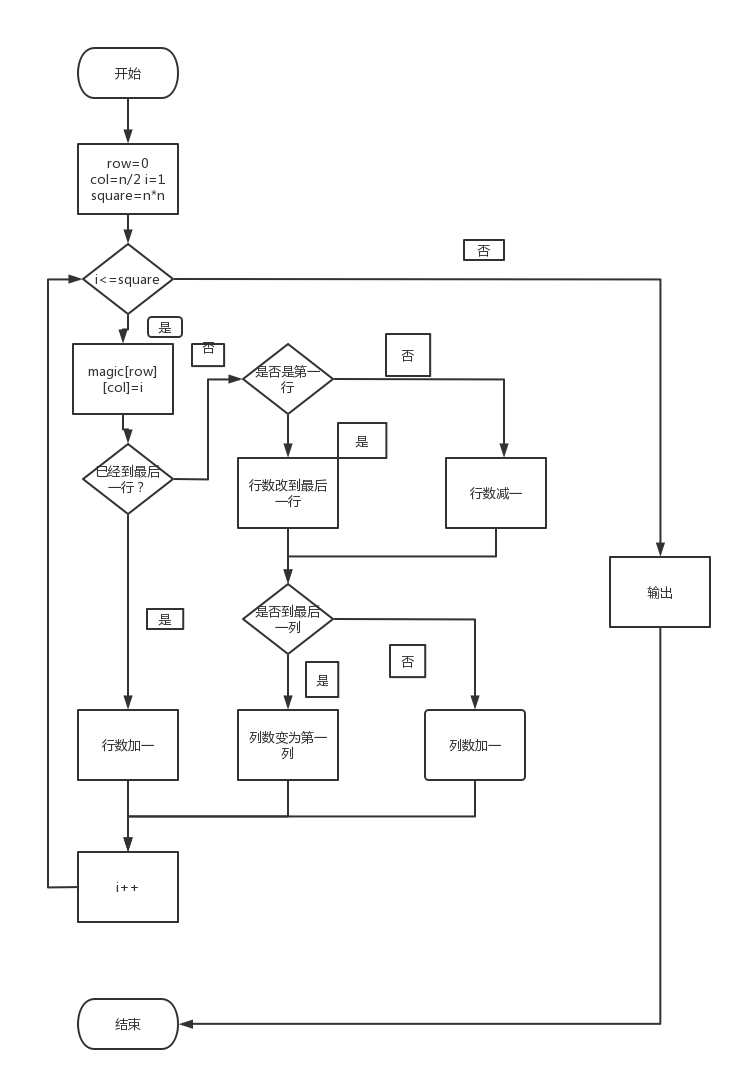
最后判断所有的行数是否和列数相等，以此来去除行列数不符合要求的数据。

3、在保证数字矩阵为方阵后，将分隔的字符串转为整数。如果有异常抛出，则存在浮点数；如没有，判断是否有负数。如均无，则将第一行数字的和作为基准，分别将每一行、每一列、正负对角线上的数字求和与基准比较，出现不一致则证明不是幻方；否则，可以判定该数字矩阵为幻方。

### generateMagicSquare()

按步骤给出你的设计和实现思路/过程/结果。

以下是该函数的流程图



1、java.lang.ArrayIndexOutOfBoundsException是常见的数组越界造成的异常，即在代码中使用了不合法的索引访问数组导致数组越界。在使用偶数n的情况下，每一次循环运行到第(n/2 \* n)次的时候，都会执行row++，然而此时已经在整个数组的第(n-1)列，所以下一次访问就出现了越界。

2、java.lang.NegativeArraySizeException是由于程序尝试建立大小为负的数组时抛出的异常。原因是n取负数时，建立二维数组的时候会将行数和列数取成负数，导致该异常抛出。

## Turtle Graphics

这个实验主要是对于turtle库的理解，其中，涉及了找凸包的算法，更好的加深了我对算法的理解以及编写软件的能力。

### Problem 1: Clone and import

首先在本地新建一个文件夹，然后git init .

Git clone

然后将考下来的代码导入到P2底下。

### Problem 3: Turtle graphics and drawSquare

在turtlesoup.java文件下编写drawSqure其中主要是利用turtle库中的forward函数和turn函数，函数的代码如下：

turtle.forward(sideLength);

turtle.turn(90);

turtle.forward(sideLength);

turtle.turn(90);

turtle.forward(sideLength);

turtle.turn(90);

turtle.forward(sideLength);

turtle.turn(90);

### Problem 5: Drawing polygons

画多边形的代码也和画正方形差不错，只需更改每一次旋转的角度就可以了，代码如下：

**public** **static** **void** drawRegularPolygon(Turtle turtle, **int** sides, **int** sideLength) {

//throw new RuntimeException("implement me!");

turtle.forward(sideLength);

**double** degrees = 360.00 / sides;

**for** (**int** i = 1; i < sides; i++) {

turtle.turn(degrees);

turtle.forward(sideLength);

}

}

画正多边形的方法其实与画正方形一样，只是循环次数和每一次右转的角度更改一下即可。在实现drawRegularPolygon之前，已经实现了根据边数计算正n边形calculateRegularPolygonAngle函数，右转的角度就是正n边形的外角度数。

### Problem 6: Calculating Bearings

**public** **static** **double** calculateBearingToPoint(**double** currentBearing, **double** d, **double** e,

**double** f, **double** g) {

//throw new RuntimeException("implement me!");

**double** vectorX = f - d;

**double** vectorY = g - e;

**double** degree = Math.*toDegrees*(Math.*atan*(vectorY/vectorX));

**double** cbtp =90- degree - currentBearing;

**if** (cbtp < 0)

**return** cbtp+360.0;

**else**

**return** cbtp;

}

calculateBearingToPoint方法实现了从初始角度，初始点到目标点的旋转角，因此只需按照算法反复调用这个方法即实现了最小凸点集问题的求解。

### Problem 7: Convex Hulls

**public** **static** Set<Point> convexHull(Set<Point> points) {

//throw new RuntimeException("implement me!");

ArrayList<Point> convex=**new** ArrayList<>();

ArrayList<Point> Point\_container=**new** ArrayList<>();

Point\_container.addAll(points);

**if**(points.size()<3) **return** points;

Point first=Point\_container.get(0);

**int** temp=0;

**for**(**int** i=0;i<points.size();i++)

{

**if**(first.x()>Point\_container.get(i).x())

{

first=Point\_container.get(i);

temp=i;

}

**else** **if**(first.x()==Point\_container.get(i).x()&&first.y()<Point\_container.get(i).y())

{

first=Point\_container.get(i);

temp=i;

}

}

convex.add(first);

Point\_container.remove(temp);

Point min=first;

**int** flag=0,tag=0;

**do**

{

tag=0;

**if**(flag==1) Point\_container.add(first);

**double** degree=360;

Point w=min;

**for**(**int** i=0;i<Point\_container.size();i++)

{

**int** px=(**int**)Point\_container.get(i).x();

**int** py=(**int**)Point\_container.get(i).y();

**int** qx=(**int**)Point\_container.get(temp).x();

**int** qy=(**int**)Point\_container.get(temp).y();

**double** degree1=*calculateBearingToPoint*(0, min.x(), min.y(),px,py);

**if**(degree>degree1)

{

degree=degree1;

temp=i;

}

**else** **if**(degree==degree1)

{

**double** dx=(px-(**int**)min.x()\*(px-(**int**)min.x())+(py-(**int**)min.y())\*(py-(**int**)min.y()));

**double** dy=(qy-(**int**)min.y()\*(qy-(**int**)min.y())+(qx-(**int**)min.x())\*(qx-(**int**)min.x()));

**if**(dx>dy)

{

w=Point\_container.get(temp);

temp=i;

}

}

}

flag++;

**if**(w!=min) tag=1;

min=Point\_container.get(temp);

convex.add(min);

Point\_container.remove(temp);

**if**(tag==1)

{

**for**(**int** i=0;i<Point\_container.size();i++)

{

**if**(Point\_container.get(i)==w) Point\_container.remove(i);

}

}

degree=360;

}**while**(first!=min);

Set<Point> final\_point=**new** HashSet<>();

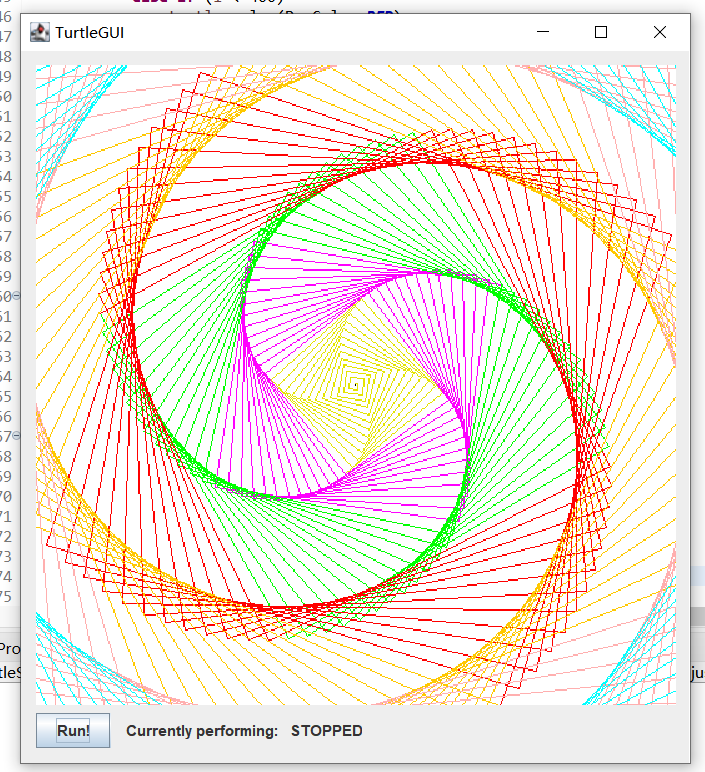
final\_point.addAll(convex);

**return** final\_point;

}

首先，寻找出最左下角的点，这个点作为凸包的第一个点，然后，按照极坐标系下的大小找出旋转角最小的点，这些点一块组成了凸包。

### Problem 8: Personal art



**public** **static** **void** drawPersonalArt(Turtle turtle) {

//throw new RuntimeException("implement me!");

**for**(**int** i=1;i<800;i=i+2)

{

turtle.forward(i);

**if** (i < 100)

turtle.color(PenColor.***YELLOW***);

**else** **if** (i < 200)

turtle.color(PenColor.***MAGENTA***);

**else** **if** (i < 300)

turtle.color(PenColor.***GREEN***);

**else** **if** (i < 400)

turtle.color(PenColor.***RED***);

**else** **if** (i < 500)

turtle.color(PenColor.***ORANGE***);

**else** **if** (i < 600)

turtle.color(PenColor.***PINK***);

**else** **if** (i < 700)

turtle.color(PenColor.***CYAN***);

**else** {

turtle.color(PenColor.***BLUE***);

}

turtle.turn(91);

}

}

### Submitting

如何通过Git提交当前版本到GitHub上你的Lab1仓库。

## Social Network

在这里简要概述你对该任务的理解。

### 设计/实现FriendshipGraph类

在处理这种社交圈关系的问题时，我采用了由两个链表来储存邻接表的办法来处理数据，这种数据处理办法的优势是节省了空间，以及方便便利。

**public** **class** FriendshipGraph {

**public** List<Person> people=**new** ArrayList<>();

**private** **int** step=0,min=100,detect=0;

**private** **boolean** checksame(Person jia)

{

**for**(**int** i=0;i<people.size();i++)

{

**if**(jia.name==people.get(i).name)

{

System.***out***.println("in put the same name!");

System.*exit*(0);

//return false;

}

}

**return** **true**;

}

**public** **void** addVertex(Person jia)

{

**if**(checksame(jia)==**true**)

{

people.add(jia);

}

}

**public** **void** addEdge(Person jia, Person yi)

{

jia.friends.add(yi);

//yi.friends.add(jia);

}

**private** **int** search(Person jia,Person yi,**int** path)

{

jia.step=path;

**for**(**int** i=0;i<jia.friends.size();i++)

{

**if**(jia.friends.get(i).visited==**false**)

{

jia.friends.get(i).visited=**true**;

**if**(jia.friends.get(i)==yi)

{

**if**(step<min) min=jia.step+1;

detect=1;

//step=0;

**return** 1;

}

search(jia.friends.get(i),yi,(jia.step+1));

}

}

//step=0;

**return** 0;

}

**public** **int** getDistance(Person jia, Person yi)

{

detect=0;

**int** path=0;

**if**(jia.name==yi.name) **return** 0;

**for**(**int** i=0;i<people.size();i++)

{

people.get(i).visited=**false**;

people.get(i).step=0;

}

search(jia,yi,path);

**if**(detect==0) **return** -1;

**return** min;

}

}

在处理重名问题时，我在addVertex函数中调用了checksame函数，调用了该函数后，对LIST中的每一个元素进行访问，如果有重名现象，立即退出程序。在getdistance函数中，我采用了广度优先搜索，并且记录了最短路径在min中，在结束遍历后返回min。

### 设计/实现Person类

**public** **class** Person {

**public** String name;

**public** **boolean** visited;

**public** **int** step;

**public** List<Person> friends=**new** ArrayList<>();

**public** Person(String s)

{

name=s;

}

}

这是我的person类，visited变量的作用是在深度优先搜索时判断该变量有没有被访问，如果被访问，标记为true，否则为false，这个在getdistance时会初始化为false。Name用来储存名字。

LIST类型的friends是用来记录朋友名字的。

### 设计/实现客户端代码main()

给出你的设计和实现思路/过程/结果。

**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));

//should print 1

System.***out***.println(graph.getDistance(rachel, ben));

//should print 2

System.***out***.println(graph.getDistance(rachel, rachel));

//should print 0

System.***out***.println(graph.getDistance(rachel, kramer));

//should print -1

}

将ross改为rachel后，程序正常终止。

### 设计/实现测试用例

给出你的设计和实现思路/过程/结果。

在测试的过程中，我用了更加复杂的网络来进行测试。测试代码如下：

import static org.junit.Assert.\*;

import org.junit.Test;

public class FriendshipGraphTest {

@Test

public void test()

{

FriendshipGraph graph = new FriendshipGraph();

Person A = new Person("A");

Person B = new Person("B");

Person C = new Person("C");

Person D = new Person("D");

Person E = new Person("E");

graph.addVertex(A);

graph.addVertex(B);

graph.addVertex(C);

graph.addVertex(D);

graph.addVertex(E);

graph.addEdge(A, B);

graph.addEdge(B, A);

graph.addEdge(B, C);

graph.addEdge(C, B);

graph.addEdge(C, D);

graph.addEdge(D, C);

graph.addEdge(D, E);

graph.addEdge(E, D);

graph.addEdge(E, A);

graph.addEdge(A, E);

assertEquals(1, graph.getDistance(A, B));

assertEquals(2, graph.getDistance(A, C));

assertEquals(2, graph.getDistance(A, D));

assertEquals(1, graph.getDistance(A, E));

}

@Test

public void test1()

{

FriendshipGraph graph = new FriendshipGraph();

Person A = new Person("A");

Person B = new Person("B");

Person C = new Person("C");

Person D = new Person("D");

Person E = new Person("E");

graph.addVertex(A);

graph.addVertex(B);

graph.addVertex(C);

graph.addVertex(D);

graph.addVertex(E);

graph.addEdge(A, B);

graph.addEdge(B, A);

graph.addEdge(B, C);

graph.addEdge(C, B);

graph.addEdge(C, D);

graph.addEdge(D, C);

graph.addEdge(D, E);

graph.addEdge(E, D);

graph.addEdge(E, A);

graph.addEdge(A, E);

assertEquals(2, graph.getDistance(A, C));

assertEquals(2, graph.getDistance(A, D));

assertEquals(1, graph.getDistance(A, E));

}

@Test

public void test2()

{

FriendshipGraph graph = new FriendshipGraph();

Person A = new Person("A");

Person B = new Person("B");

Person C = new Person("C");

Person D = new Person("D");

Person E = new Person("E");

graph.addVertex(A);

graph.addVertex(B);

graph.addVertex(C);

graph.addVertex(D);

graph.addVertex(E);

graph.addEdge(A, B);

graph.addEdge(B, A);

graph.addEdge(B, C);

graph.addEdge(C, B);

graph.addEdge(C, D);

graph.addEdge(D, C);

graph.addEdge(D, E);

graph.addEdge(E, D);

graph.addEdge(E, A);

graph.addEdge(A, E);

assertEquals(2, graph.getDistance(A, D));

}

@Test

public void test3()

{

FriendshipGraph graph = new FriendshipGraph();

Person A = new Person("A");

Person B = new Person("B");

Person C = new Person("C");

Person D = new Person("D");

Person E = new Person("E");

graph.addVertex(A);

graph.addVertex(B);

graph.addVertex(C);

graph.addVertex(D);

graph.addVertex(E);

graph.addEdge(A, B);

graph.addEdge(B, A);

graph.addEdge(B, C);

graph.addEdge(C, B);

graph.addEdge(C, D);

graph.addEdge(D, C);

graph.addEdge(D, E);

graph.addEdge(E, D);

graph.addEdge(E, A);

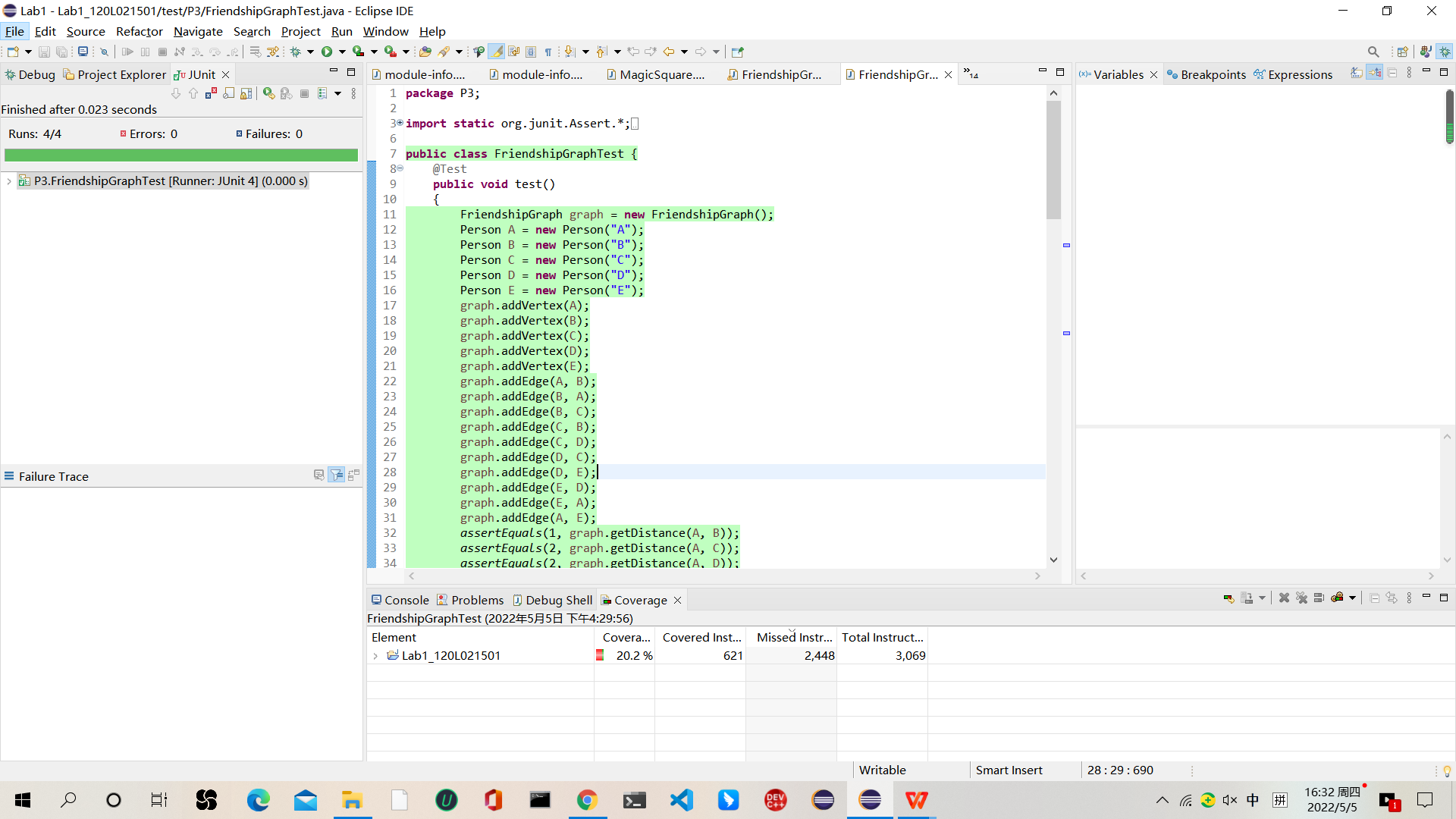
graph.addEdge(A, E);

assertEquals(1, graph.getDistance(A, E));

}

}

测试结果如下：



# 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 |
| 2022-05-01 | 18:30-21:30 | 完成问题一 | 按计划完成 |
| 2022-05-02 | 17：30-18：30 | 学习了JAVA有关的知识 | 提前20分钟完成 |
| 2022-05-03 | 8：00-20：00 | 完成了问题二三 | 按计划完成 |
| 2022-05-04 | 18：00-21：00 | 实验报告 | 按计划完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的困难 | 解决途径 |
| 不熟悉list和queue的用法 | 查资料解决 |
| 不熟悉git指令 | csdn解决 |
| 代码出现问题 | 从深度优先搜索换为广度优先搜索解决 |

# 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（必答）

这次实验，极大的提高了我处理问题和解决问题的能力，教训是配置环境时需要谨慎，一不小心就会有意想不到的结果，甚至可能需要重装系统。

## 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？

java编程对我来说比较对胃，和我常用的C++相比，JAVA的语法较为简单，报错相对较少，不足之处是运行速度慢，且很多函数过于冗长。

1. 关于Eclipse或IntelliJ IDEA，它们作为IDE的优势和不足；

Eclipse作为编译器，优点是可以自动import需要的包，缺点是对于目录的分级比较麻烦。

1. 关于Git和GitHub，是否感受到了它在版本控制方面的价值；

Git的优势在于方便多人协同工作，以及由于他的树状结构，塔在版本控制上有明显的优势。

1. 关于CMU和MIT的作业，你有何感受；

我的感受是利用较为简单的题来让我们熟悉了JAVA，题目非常好，但我希望HIT能够出自己的题，这样才符合规格严格，功夫到家的校风。

1. 关于本实验的工作量、难度、deadline；

本次作业存在一定的工作量，难度适中，因为这周有考试，deadline比较让人难受。

1. 关于初接触“软件构造”课程；

这门课给我的感觉是侧重实践，以及实验量大，通过对于这门课的学习能够促使我们更好的提高自己的动手实践能力，并且帮助我成为一名合格的程序员。