

**2023年春季学期  
计算学部《软件构造》课程**

**Lab 1实验报告**

|  |  |
| --- | --- |
| 姓名 | 唐轩 |
| 学号 | 2021111858 |
| 班号 | 2137101 |
| 电子邮件 | 15084551298@163.com |
| 手机号码 | 15084551298 |

**目录**

[1 实验目标概述 1](#_Toc130067488)

[2 实验环境配置 1](#_Toc130067489)

[3 实验过程 5](#_Toc130067490)

[3.1 Magic Squares 5](#_Toc130067491)

[3.1.1 isLegalMagicSquare() 5](#_Toc130067492)

[3.1.2 generateMagicSquare() 6](#_Toc130067493)

[Turtle Graphics 8](#_Toc130067494)

[3.1.3 Problem 1: Clone and import 10](#_Toc130067495)

[3.1.4 Problem 3: Turtle graphics and drawSquare 12](#_Toc130067496)

[3.1.5 Problem 5: Drawing polygons 12](#_Toc130067497)

[3.1.6 Problem 6: Calculating Bearings 14](#_Toc130067498)

[3.1.7 Problem 7: Convex Hulls 14](#_Toc130067499)

[3.1.8 Problem 8: Personal art 16](#_Toc130067500)

[3.1.9 Submitting 16](#_Toc130067501)

[3.2 Social Network 16](#_Toc130067502)

[3.2.1 设计/实现FriendshipGraph类 16](#_Toc130067503)

[3.2.2 设计/实现Person类 17](#_Toc130067504)

[3.2.3 设计/实现客户端代码main() 17](#_Toc130067505)

[3.2.4 设计/实现测试用例 18](#_Toc130067506)

[4 实验进度记录 19](#_Toc130067507)

[5 实验过程中遇到的困难与解决途径 20](#_Toc130067508)

[6 实验过程中收获的经验、教训、感想 21](#_Toc130067509)

[6.1 实验过程中收获的经验和教训（必答） 21](#_Toc130067510)

[6.2 针对以下方面的感受（必答） 21](#_Toc130067511)

# 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。

l 基本的 Java OO 编程

l 基于 Eclipse IDE 进行 Java 编程

l 基于 JUnit 的测试

l 基于 Git 的代码配置管理

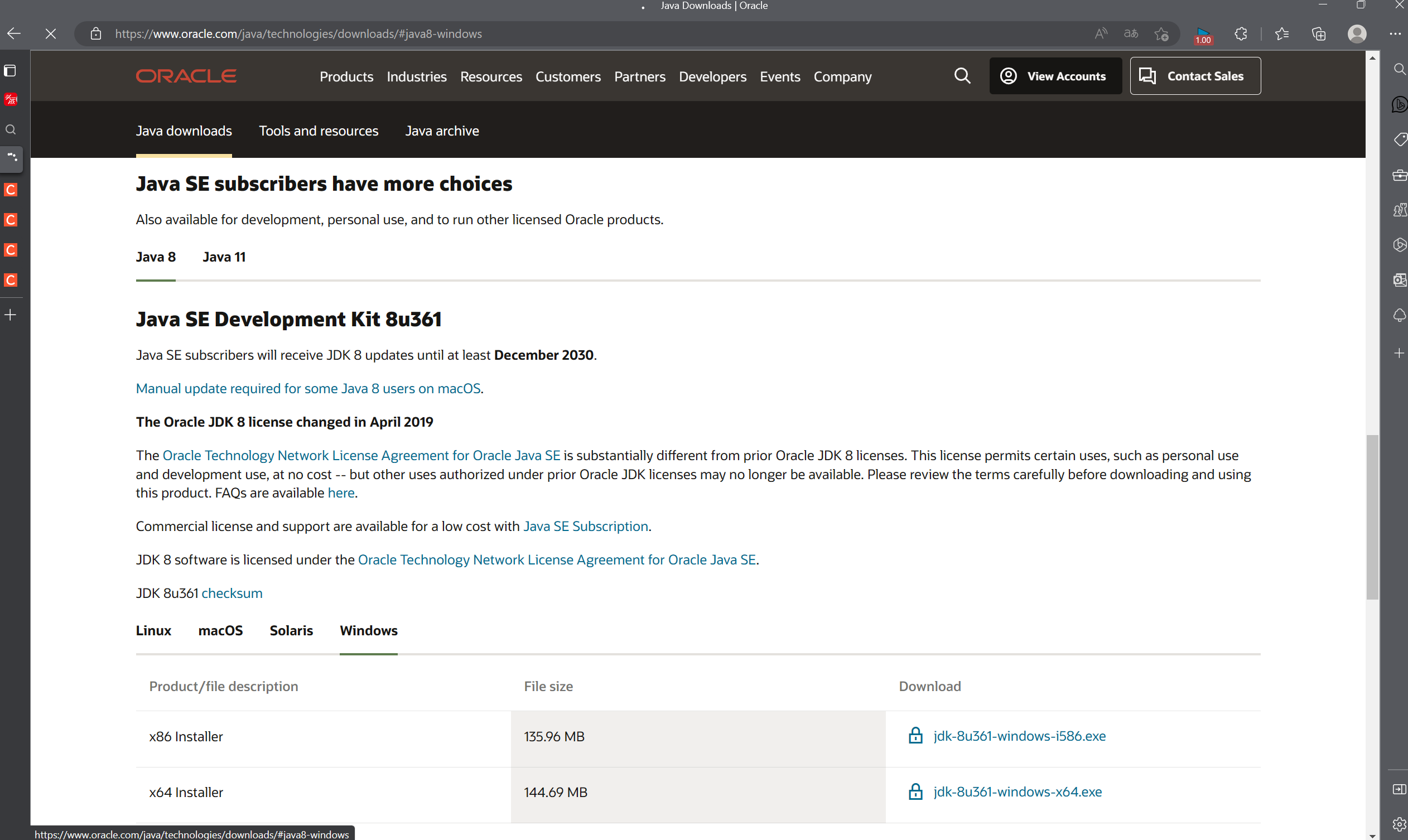
# 实验环境配置

/\* 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。

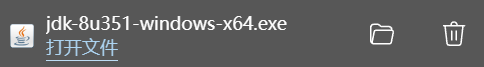
特别是要记录配置过程中遇到的问题和困难，以及如何解决的。\*/

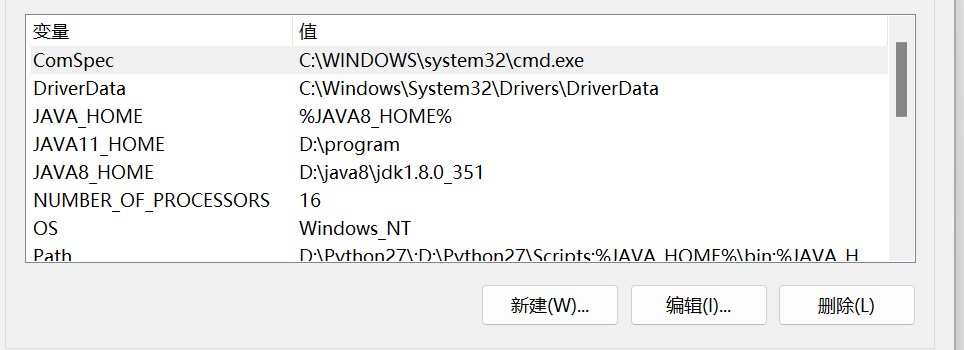
在这里给出你的GitHub Lab1仓库的URL地址。

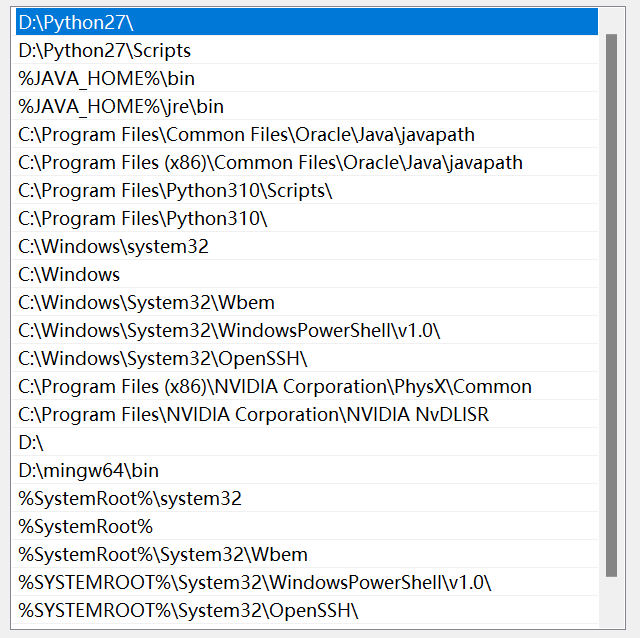
下载java8：



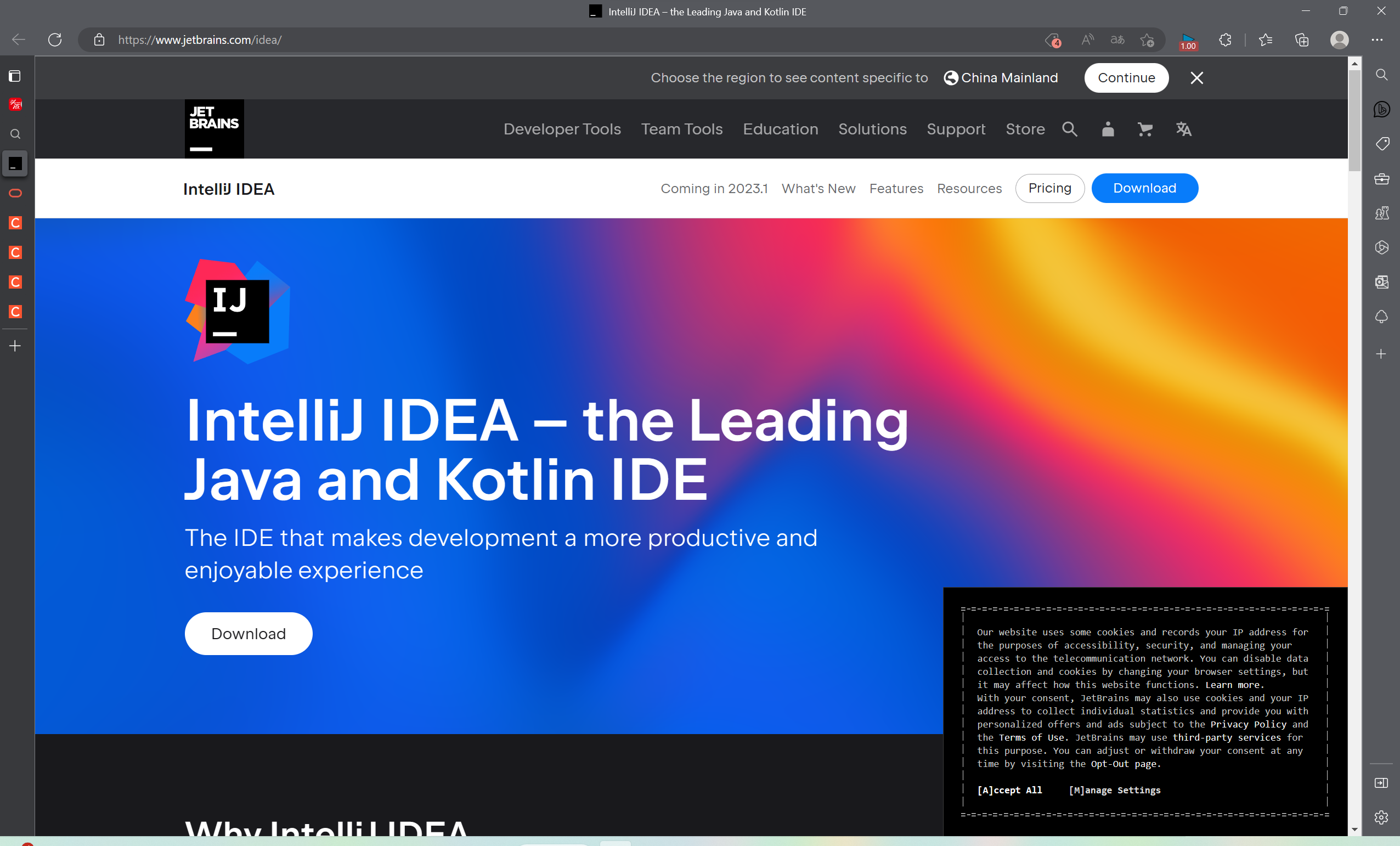
进入java官网。



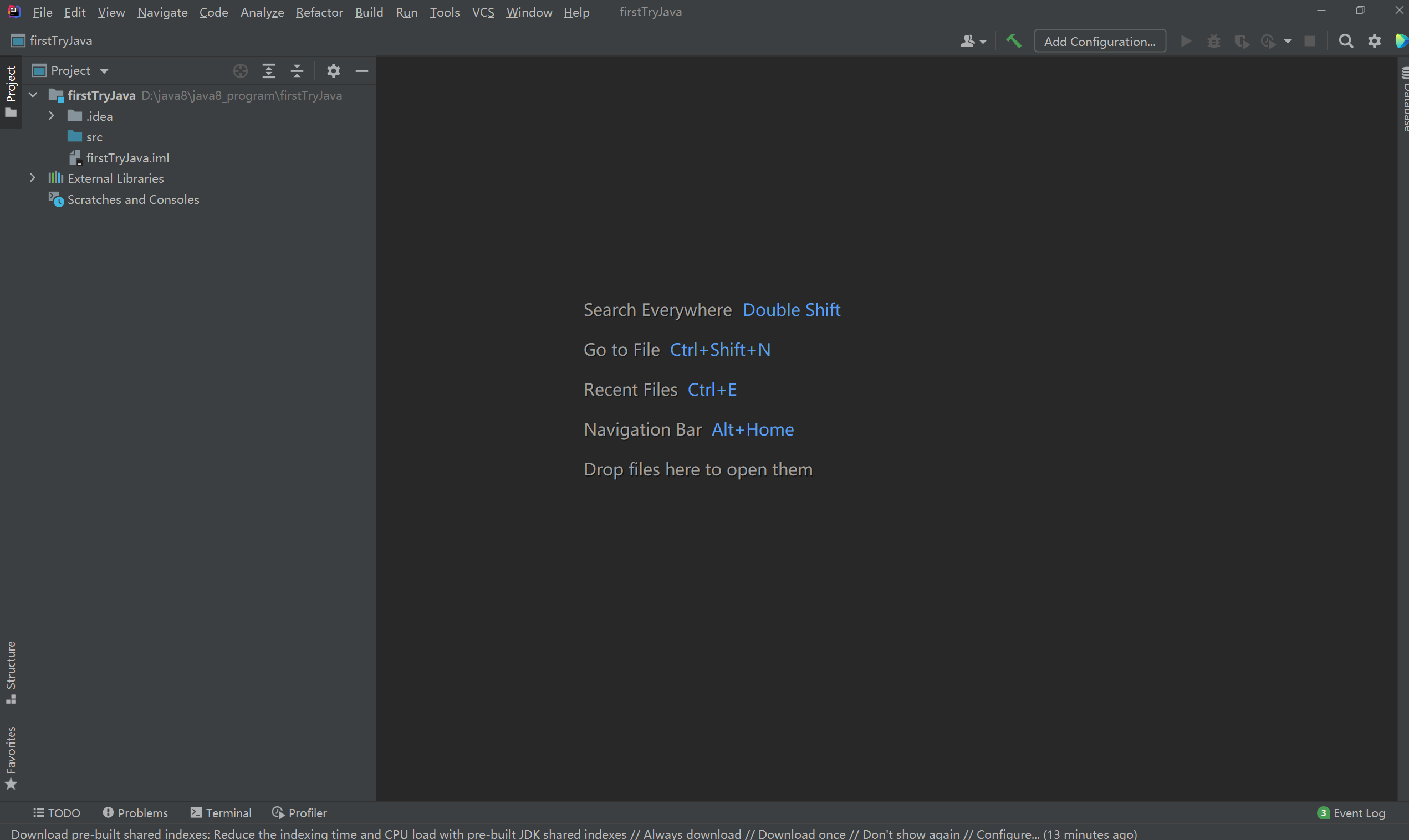




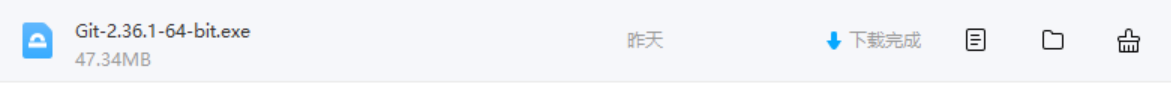
下载成功并安装，通过查阅资料了解配置步骤，配置环境变量。



进入idea官网下载idea并安装，此处下载的为2021.1.3 x64版本。

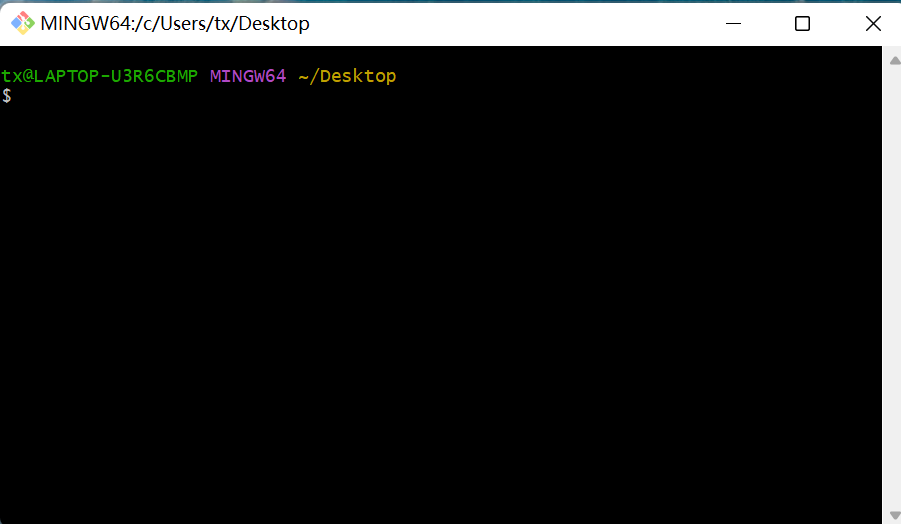


使用的为30天免费试用，通过上网查阅资料下载拓展包适当的延长了试用时间。



下载git，因为在官网下载速度过慢，在网盘中找到旧版本的git，下载时间较快。





参照网上资料中的步骤进行安，装可以正常运行。

# 实验过程

请仔细对照实验手册，针对三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

要编的第一个函数是判断的5个文件中的数字序列是否为幻方，返回true或false。在遇到错误时要判断错误类型，给出错误提示，并返回false。可能出现的错误有数字之间没有用“\t”分隔，数字序列不是矩阵，数字序列是矩阵但行列数不同，数字不为正整数，数字的行之和与列之和和对角线之和不相等。

### isLegalMagicSquare()

按步骤给出你的设计和实现思路/过程/结果。

首先步骤为文件读写，此处采用BufferedReader字符流读入文件，一次读入一行，读文件和关闭文件流时会产生编译时错误，在函数后抛出异常即可处理。如下所示：

public static boolean isLegalMagicSquare(String fileName) throws IOException

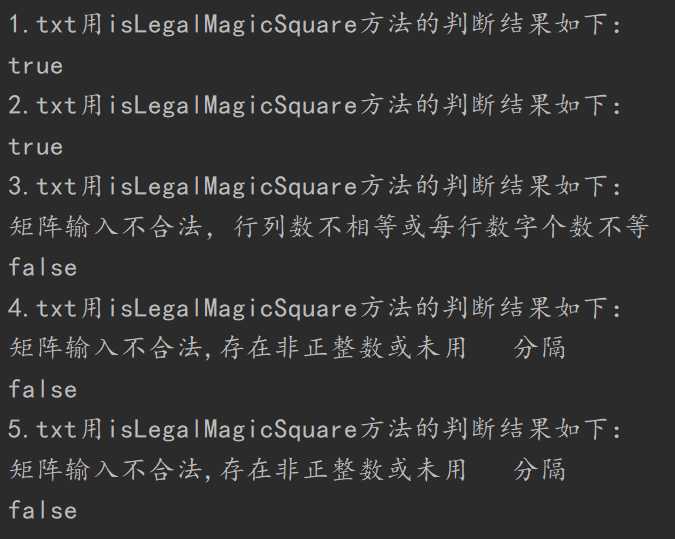
第二个关键点为判断输入是否合法，读文件时已经将txt文本中的每行以字符串读入了字符数组中，并已经用string.split(“\t”)分割完毕，如下：

while((s = bfr.readLine()) != null){  
 str\_matrix[count] = s.split("\t");  
 num[count] = str\_matrix[count].length;  
 count++;  
}

故可直接用正则表达式”\\d+”判断读入的字符串是否合法，即判断是否有非正整数的输入或者未用\t分隔。注意到幻方矩阵行列数一定相同，故可用一循环判断输入是否为矩阵或行列数是否相同。

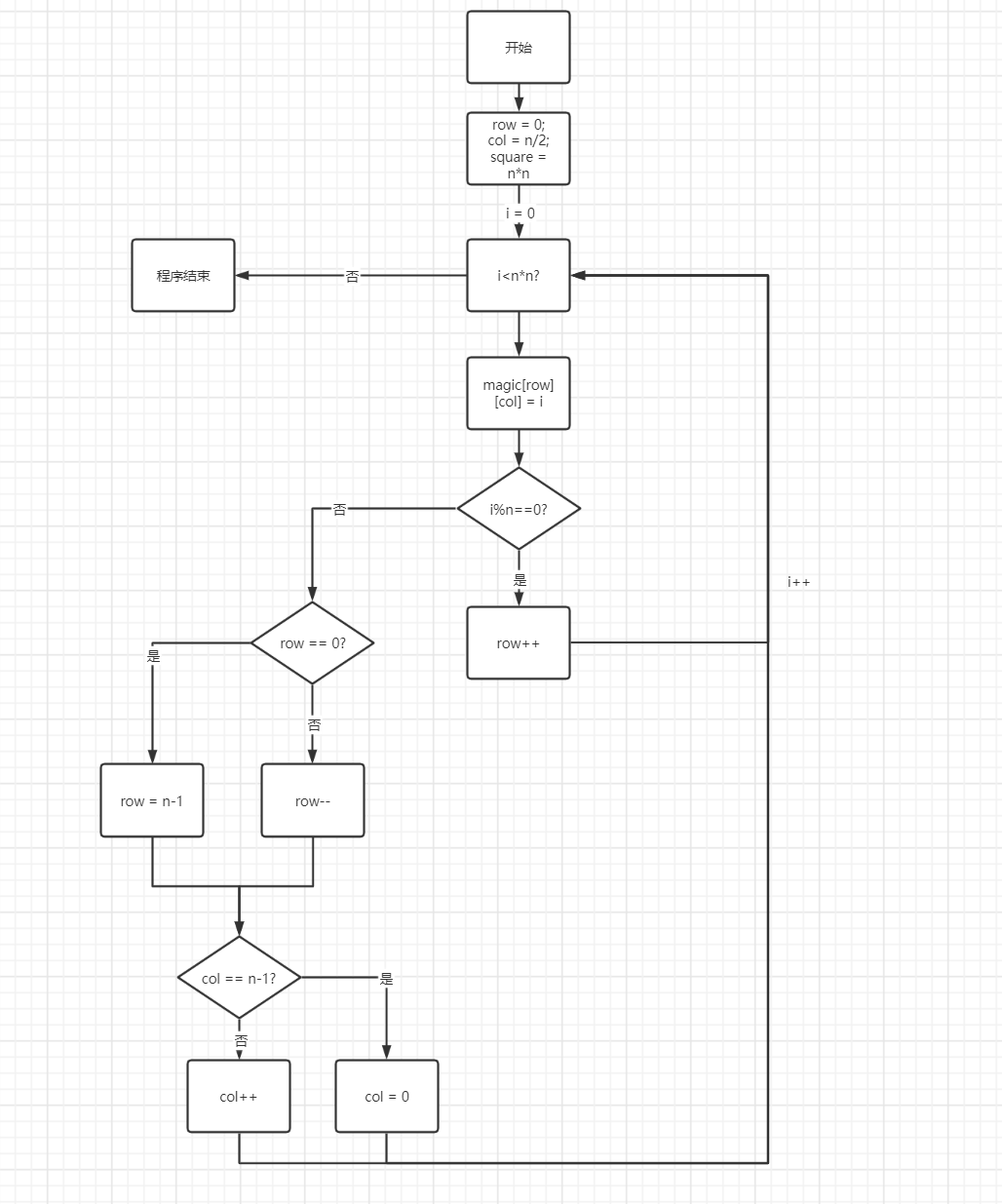
第三步相对简单，矩阵输入已经合法，用sum变量直接保存任意列（或者任意行）的数字和，分别与每行每列和两对角线的数字和对比，如果对比中任意一次不同，即可返回false。最后，如果通过了所有的比较，在函数末尾直接返回true即可。

在main中调用结果如下：



### generateMagicSquare()

n为奇数生成流程：首先在第一行的中间赋值为1，然后遵循如下规则：优先判断当前数字是否为n的倍数，如果是，就在该数字的下一行（正下方）赋值该数字+1；如果不是，就给当前数字右上角的数字赋值当前数字+1，如果该数字在上边界，就跳跃到正下方的下边界右面一个位置赋值，如果在右边界，就就跳跃到正左边的左边界上面一个位置赋值。遵循此规则，赋值n\*n次后结束，具体流程图如下：



当n输入为偶数时，矩阵赋值不能按照此规则赋值，因为在赋值的时候会在对角线越界赋值，从而会报错ArrayIndexOutOfBoundsException（越界访问错误）；当n输入为负数时必然是不合法的，因为不存在长度为负数的数组，所以会报错NegativeArraySizeException（负数大小错误）。

故可以采用try/catch语句对错误进行捕获，进行“优雅的退出”各个catch语句如下所示：

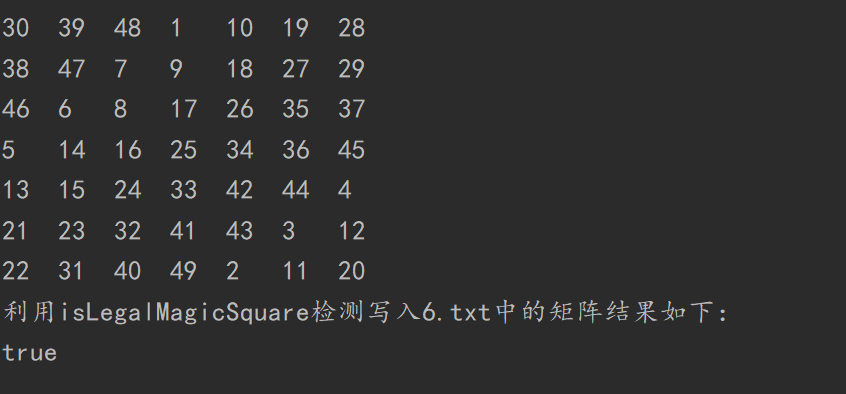
catch (ArrayIndexOutOfBoundsException e)

catch (NegativeArraySizeException e)

catch (FileNotFoundException e)

catch (IOException e)

把矩阵写入文件采用了OutputStreamWriter流，需要注意的是在关闭时需要先关闭OutputStreamWriter流，再关闭FileOutputStream流（即先打开的后关闭），否则会报错提示关闭失败。写入成功后调用isLegalMagicSquare测试，会显示矩阵符合要求（此处以n = 7为例）：



## Turtle Graphics

该任务需要我们clone已有的程序后，利用turtle按照要求画图，其中需要利用几何知识设计一些函数简化编程，最后可以发挥想象力进行Personal Art。

初次导入项目时，每个文件中的代码均会报出不同数量的错误，经分析后这些错误主要来源于导入包名错误和JUnit未在idea中导入，解决办法如下：

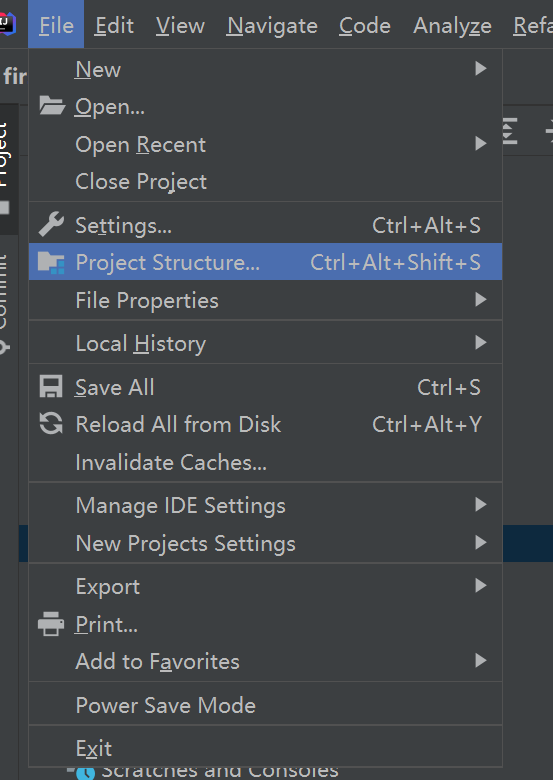
1.将每处包名导入错误补写完整，即加上P2.，示例如下：

package P2.turtle;

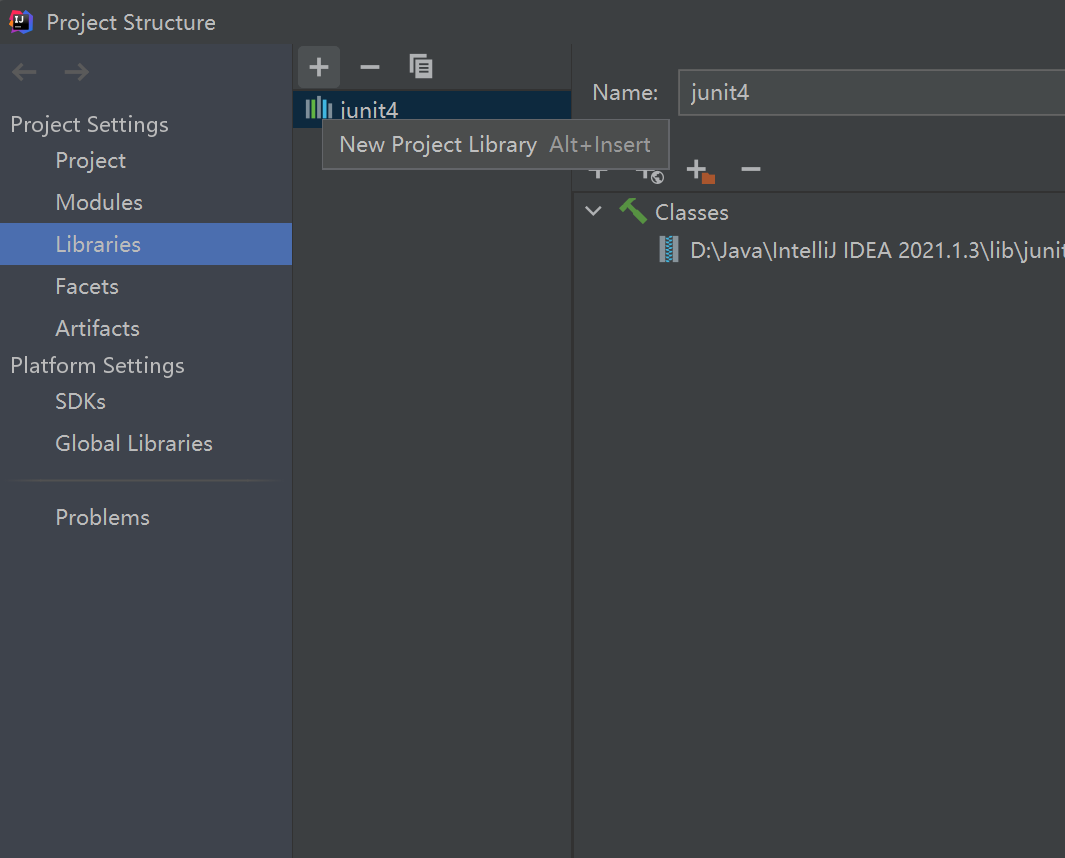
因为所有文件和包都在P2下，故需要加上P2.。除了开头引用部分，各个方法处或方法内部也有同样情况出现，补写完整即可。

2.在idea中导入JUnit，具体步骤如下：

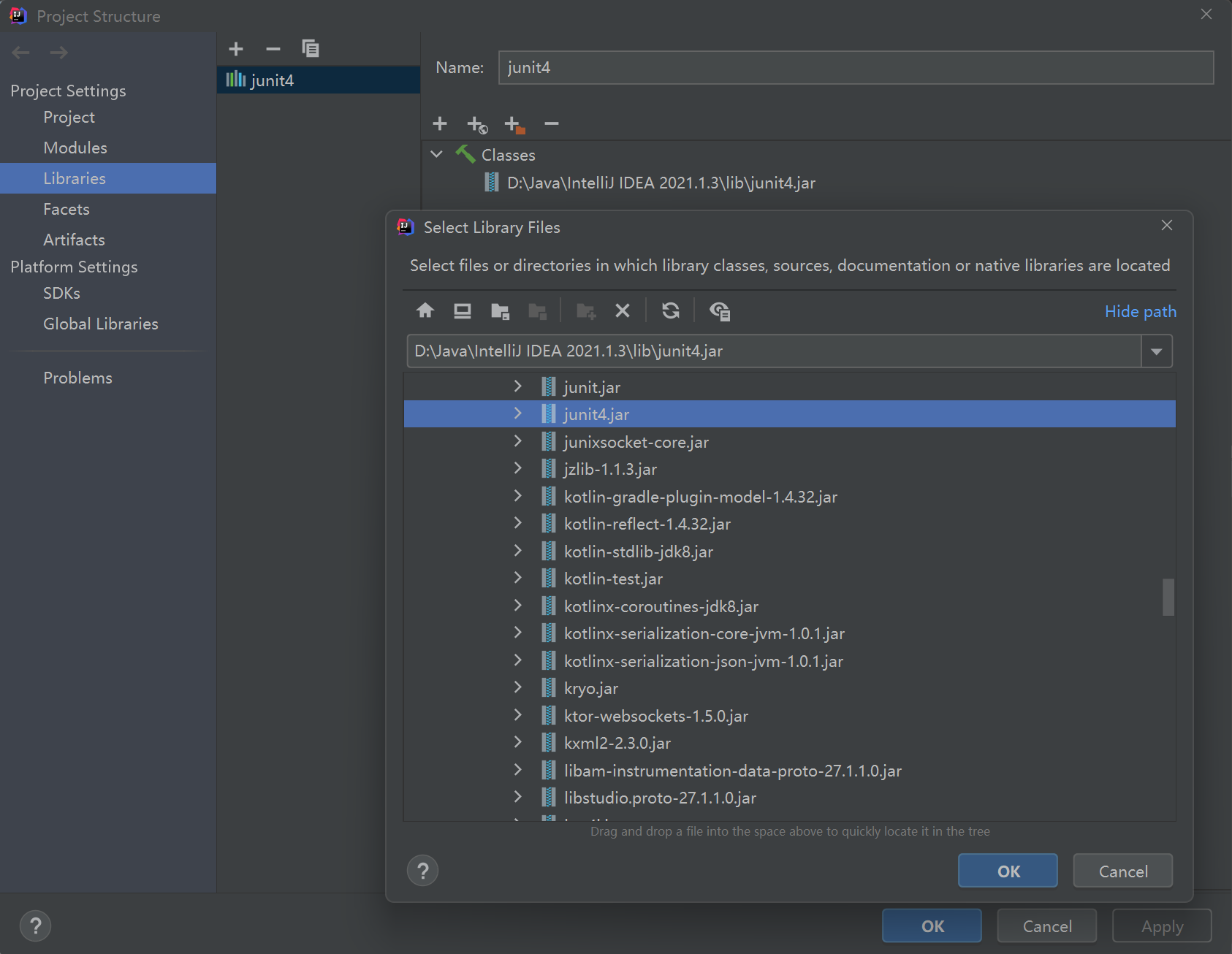
打开File下的这一栏：



在其中的Libraries点击左边的加号（截图时间为导入完成后）



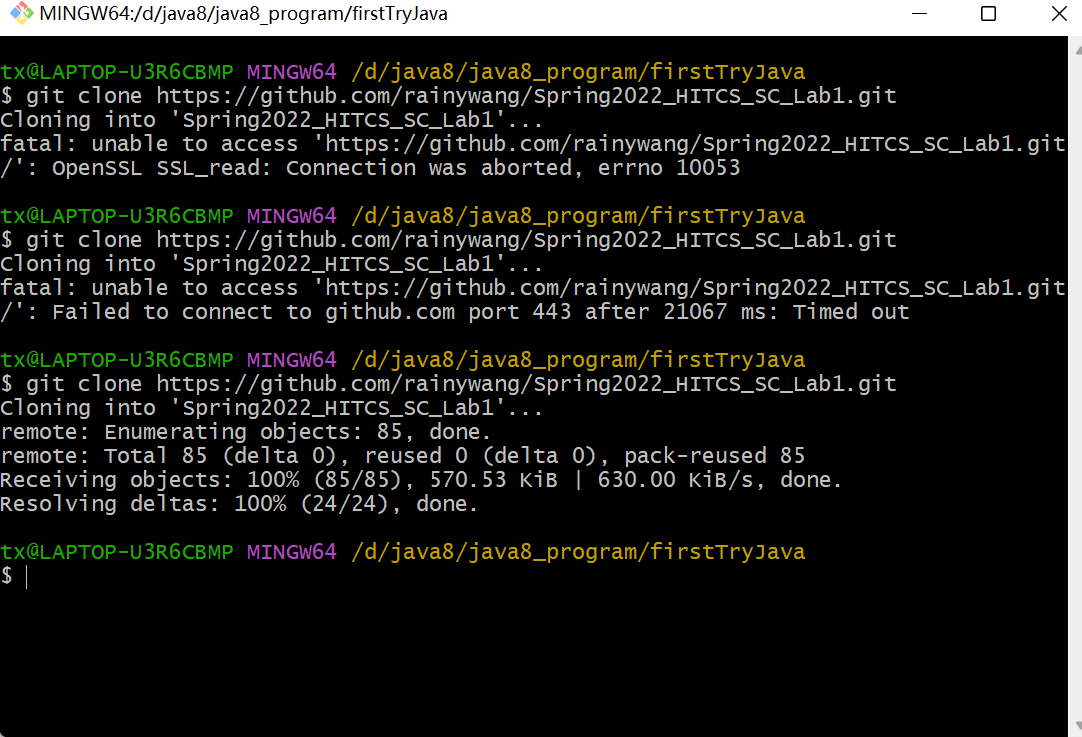
点击Java一栏，在idea的安装目录下找到lib文件夹，在其中找到junit4包



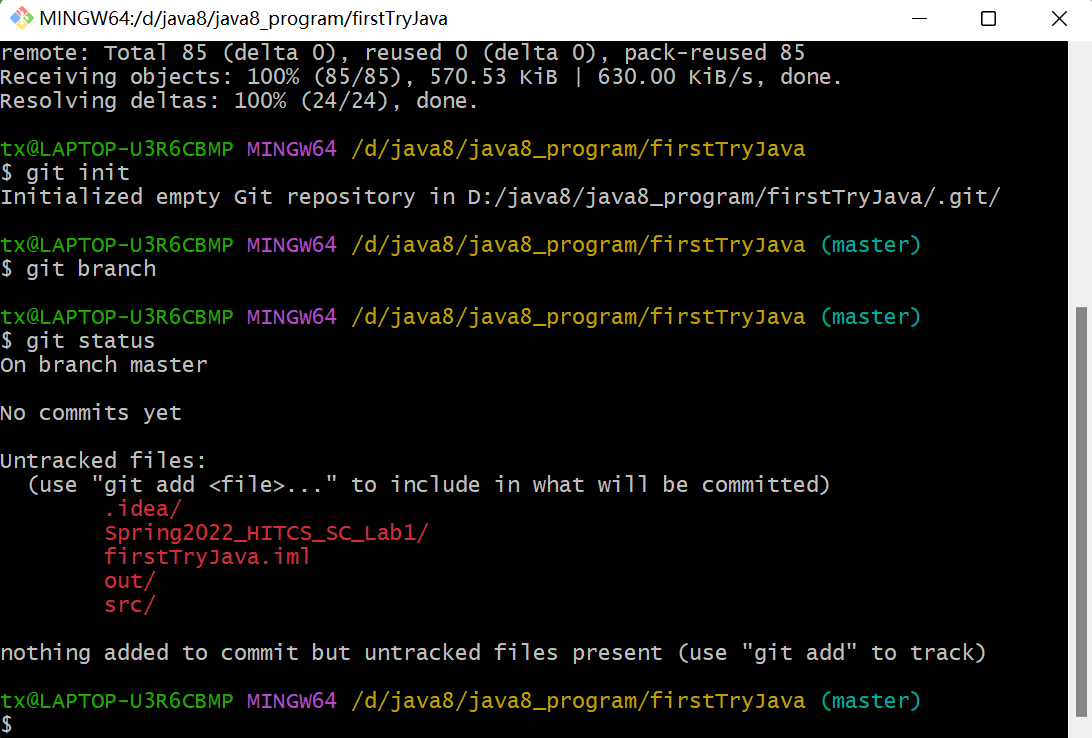
之后点击OK保存后即可，导入成功后Test文件中不再提示” Cannot resolve symbol 'junit' ”。

### Problem 1: Clone and import

进入到对应目录中，打开git，利用在GitHub上复制的地址克隆到本地，如下：

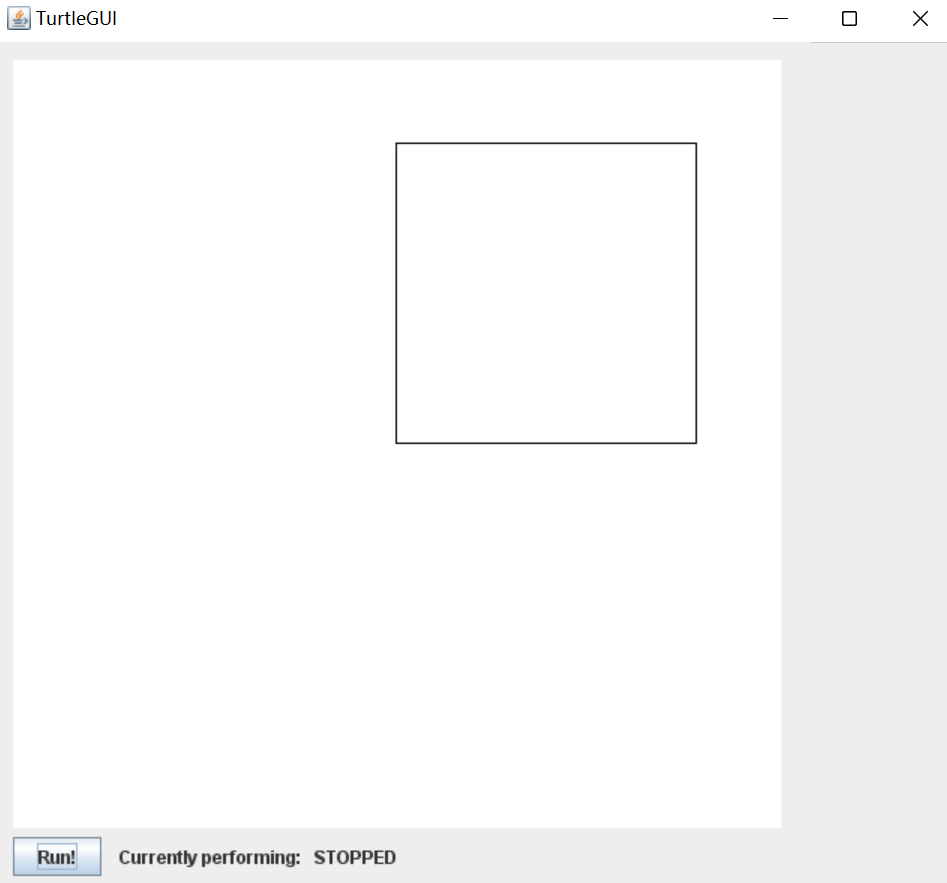


在使用git init指令初始化本地库之后，就可以用各种git指令管理本地开发了。



### Problem 3: Turtle graphics and drawSquare

该部分要实现drawSquare函数画一个正方形，利用forward函数和turn函数可以十分容易的实现，只需要循环四次前进画线和转弯90°即可，200单位的正方形如下：



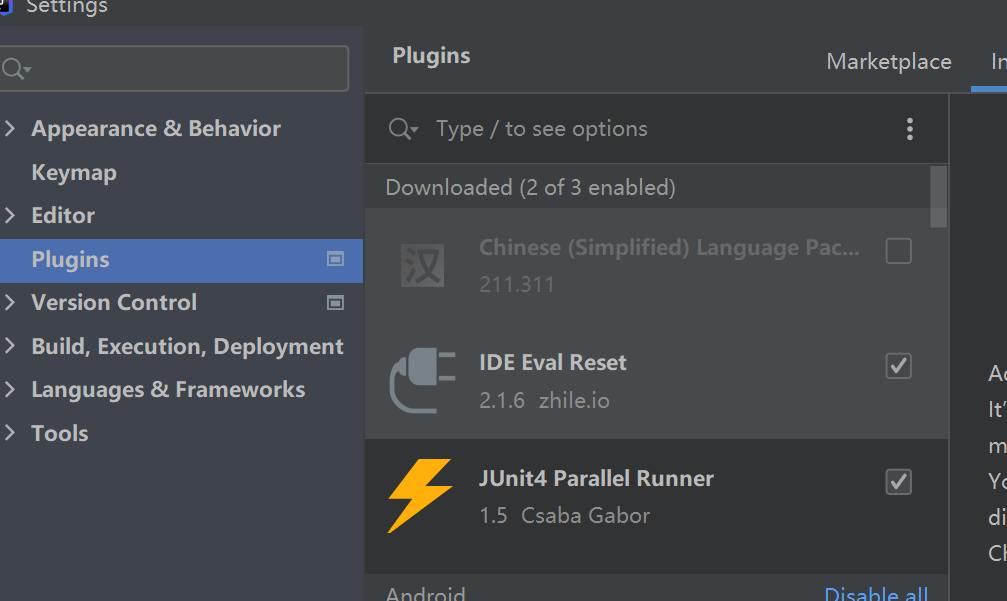
### Problem 5: Drawing polygons

该方法要求实现计算多边形内角的函数，注意合法性检验，当输入sides小于2时应打印错误语句，当sides合法时，由几何关系可得角度公式如下：

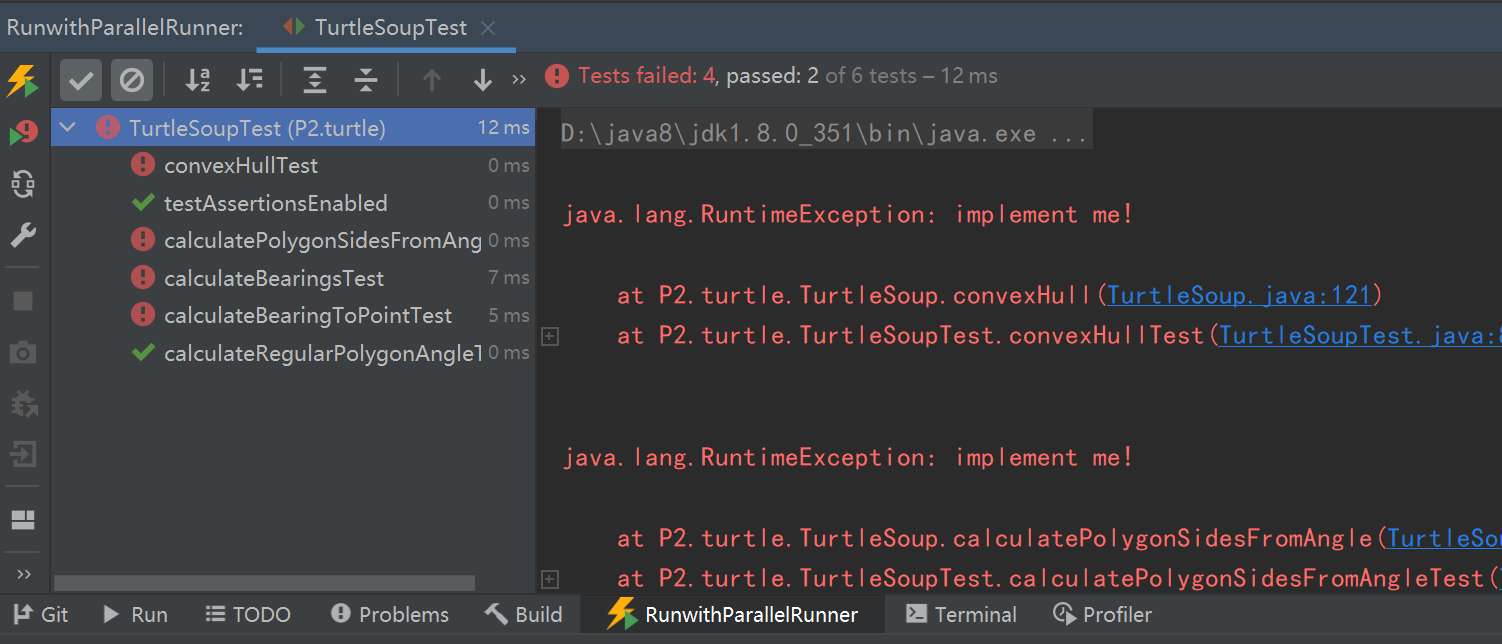
double angle = (double)180.0 - (double) 360.0 / (double) sides;

返回angle即可。

在这之后对TurtleSoupTest进行junit测试，这里使用了idea中的junit4 Parallel Runner插件进行测试



测试结果如下：

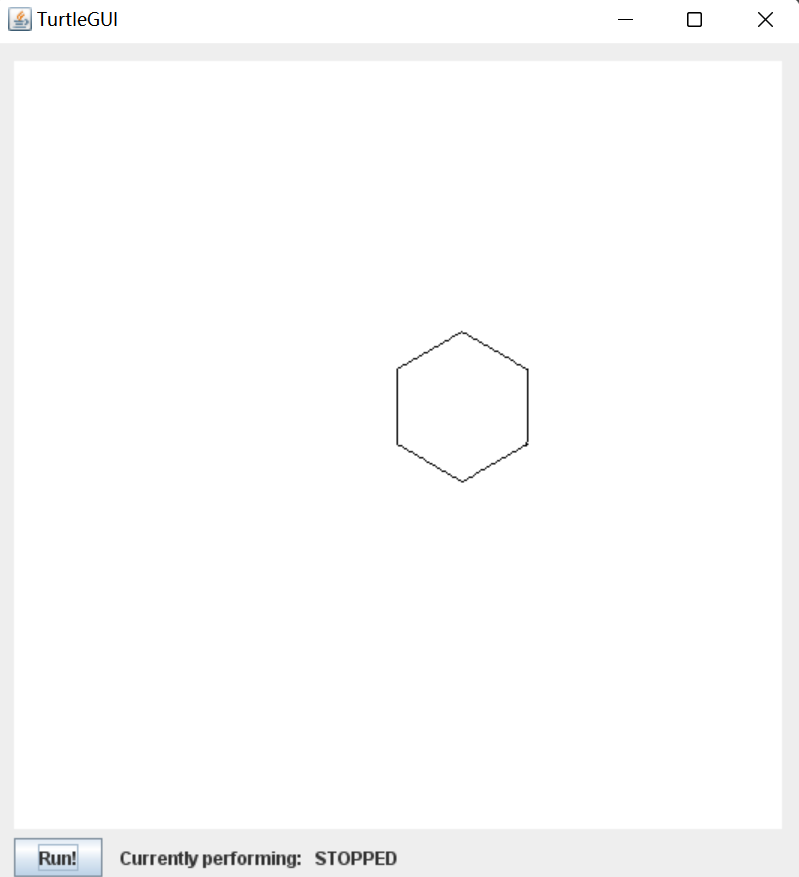


可以看到testAssertionsEnabled和calculateRegularPolygonAngleTest通过了测试，符合预期。

利用已经获得的calculateRegularPolygonAngle方法，我们可以计算出正多边形的内角，所以在drawRegularPolygon方法中只要给定正多边形的边数和边长就可以画出，只需要循环sides次，每次转过的角度为180.0 – 内角，循环代码如下：

for (int i = 0; i < sides; i++) {  
 turtle.forward(sideLength);  
 turtle.turn(180.0 - angle);  
}

单独测试drawRegularPolygon函数如下：



### Problem 6: Calculating Bearings

首先我们要计算turn从当前点到目标点所需的参数，以当前方向作为附加参数。例如，如果海龟在 (0,1) 处面向 30 度，并且必须到达 (0,0)，它必须再转 150 度，所以calculateBearingToPoint(30, 0, 1, 0, 0)会返回150.0。利用Math.atan2函数直接返回角度值，由数学关系得：

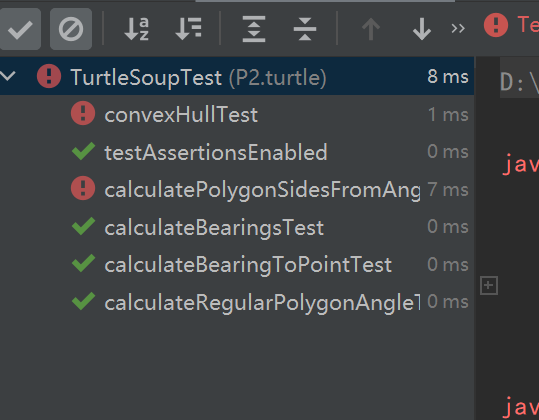
double angle = Math.*atan2*(targetY - currentY,targetX - currentX) \* 180.0 /Math.*PI*;  
angle = angle < 0 ? -angle : 360.0 - angle;  
return (90.0 - currentBearing + angle) % 360.0;

因为考虑到结果需要在0-360.0区间内，所以结果应该%360.0。

calculateBearings函数较为简单，直接利用上面的calculateBearingToPoint函数即可，注意每次记录转过角度，并与已经转过角度相加：

for (int i = 0; i < xCoords.size()-1; i++) {  
 temp = *calculateBearingToPoint*(angle,xCoords.get(i),yCoords.get(i),xCoords.get(i+1),yCoords.get(i+1));  
 list\_angle.add(temp);  
 angle += temp;  
 temp = 0;  
}

之后可以看到利用junit测试符合预期：



### Problem 7: Convex Hulls

该函数要求从给定点中求出在点集凸包边界上的点，这里采用的是礼品包装算法。顾名思义，礼品包装算法就是先找到一个一定在边界上的点，然后逐渐的找到其他在边界上的点，将所有点像包装礼品一样包起来。

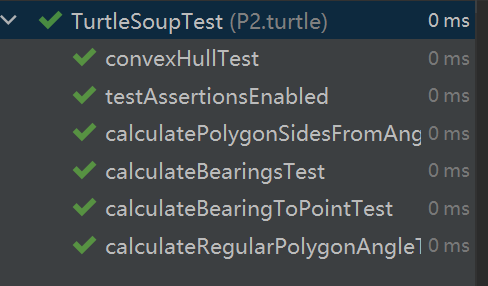
先找到在最左侧（左下角）的点（即x最小，x一样情况下找y最小），该点一定在边界上，然后顺时针寻找与当前朝向夹角最小的点加入集合，并更新朝向（当前朝向偏转角加旋转角），一直如此找下去，直到找到第一个点。此处可以利用之前写过的calculateBearingToPoint函数，该函数只是传入参与我们的目标不符，但功能上两者一致，无需修改内部代码，所以对该函数进行拓展如下：

public static double calculateBearingToPoint(double currentBearing, double currentX, double currentY,  
 double targetX, double targetY) {  
 double angle = Math.*atan2*(targetY - currentY, targetX - currentX) \* 180.0 / Math.*PI*;  
 angle = angle < 0 ? -angle : 360.0 - angle;  
 return (90.0 - currentBearing + angle) % 360.0;  
}

如此一来求凸包的工作会简单许多。先找到最左下角的点：

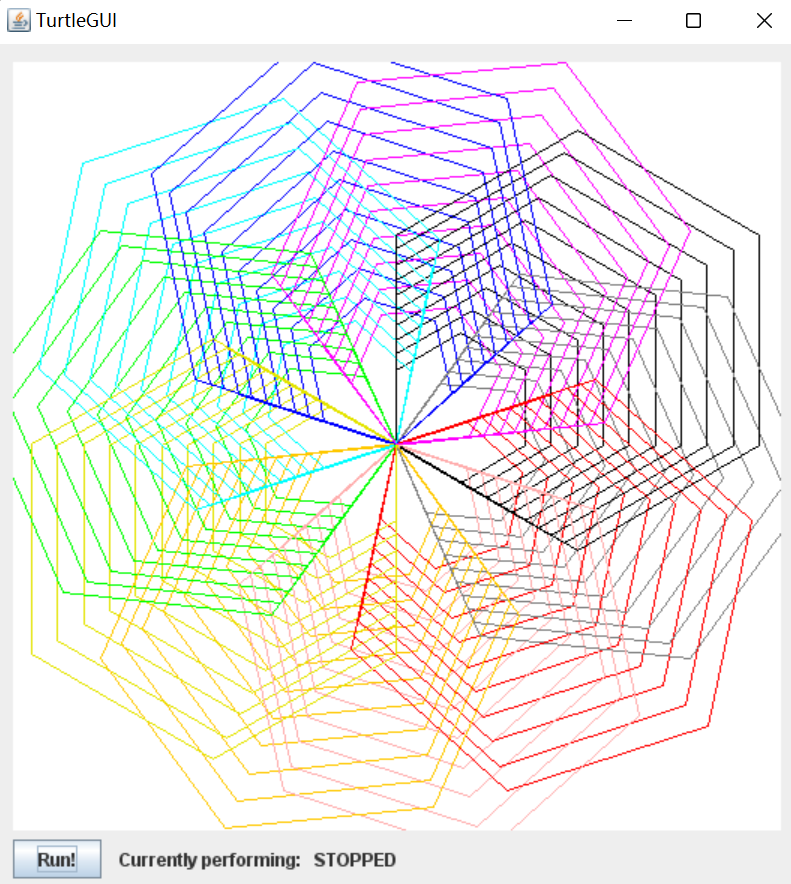
while (ie.hasNext()) {  
 Point a = ie.next();  
 if ((Double.*compare*(temp.y(), a.y()) == 1) || ((Double.*compare*(temp.x(), a.x()) == 0) && (Double.*compare*(temp.y(), a.y()) == 1))) {  
 temp = a;  
 }  
}

然后遍历寻找即可。



完成后测试全部符合预期。

### Problem 8: Personal art



该函数中可自由发挥创造力，绘制出自己喜欢的图形，上图是我的个人艺术绘出的图形。

### Submitting

将Lab1仓库的地址复制下来，在commit完成后利用git push完成提交。

## Social Network

该任务要求构建一个关系网络图，需要人作为图的顶点，顶点之间需要存在有向边，因为这个任务要构建的是一个无向图，所以当确定两个人之间是朋友关系时，需要添加双向有向边。

这个社交网络图需要可以加入顶点、添加朋友关系、计算两个人在社交网络上的最短距离，同时还要能检测是否有重名的人，Person这个类里面至少要有名字这个属性。

### 设计/实现FriendshipGraph类

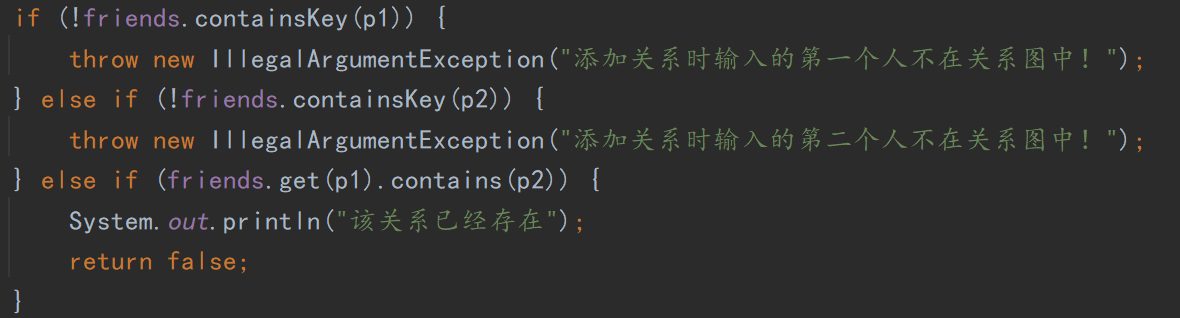
该类中要实现三个函数：添加点，添加单向边和计算两个点最短距离。故采用HashMap来存储整个关系图：

private final HashMap<Person, HashSet<Person>> friends;

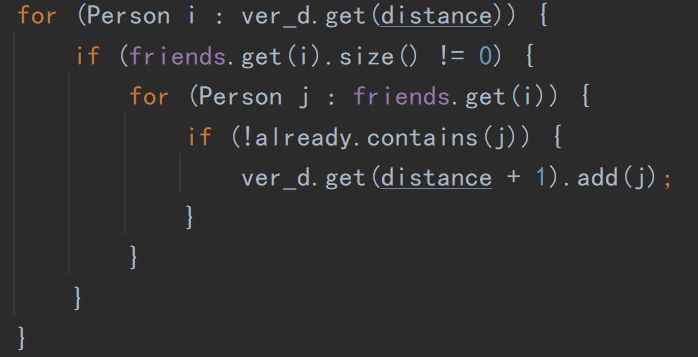
确定了合适的实现方式，下面实现起来就比较容易了。添加点的函数需要检测是否输入了同一个人（每个人名字唯一），如果重复就进行报错：

while(ie.hasNext()){  
 Person a = ie.next();  
 if(per.getName().equals(a.getName())){  
 throw new IllegalArgumentException("输入名字重复!");  
 }  
}

添加边的函数需要检测需要添加边的点是否在图内，以及需要添加的边是否已经存在：

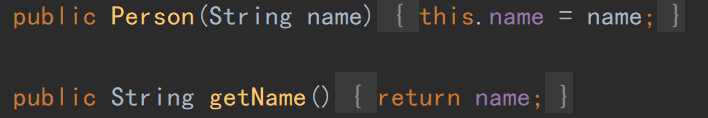


寻找最短距离使用广度优先搜索，核心代码如下：



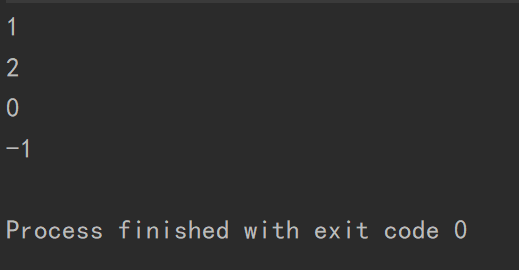
### 设计/实现Person类

Person类测试比较简单，只需要拥有赋值和获取名字的功能：

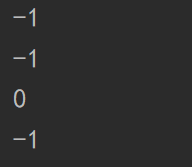


### 设计/实现客户端代码main()

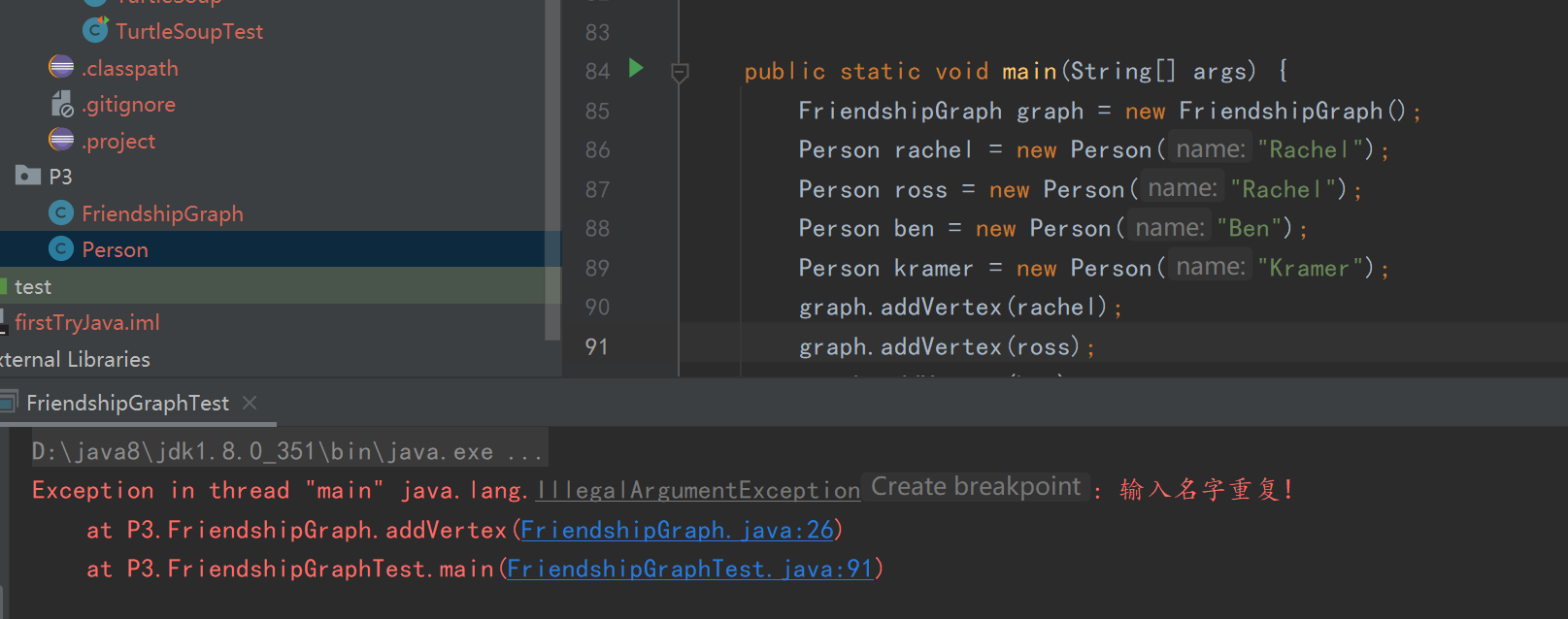
我将main代码放入了FriendshipGraphTest文件中，正常运行结果如下：



可以看出符合预期。如果将main中第10行注释掉，rachel便不再与别人连接，得出结果为：



符合预期。如果将main中第3行引号里替换为Rachel，违反人名不能重复的原则，会有如下结果：

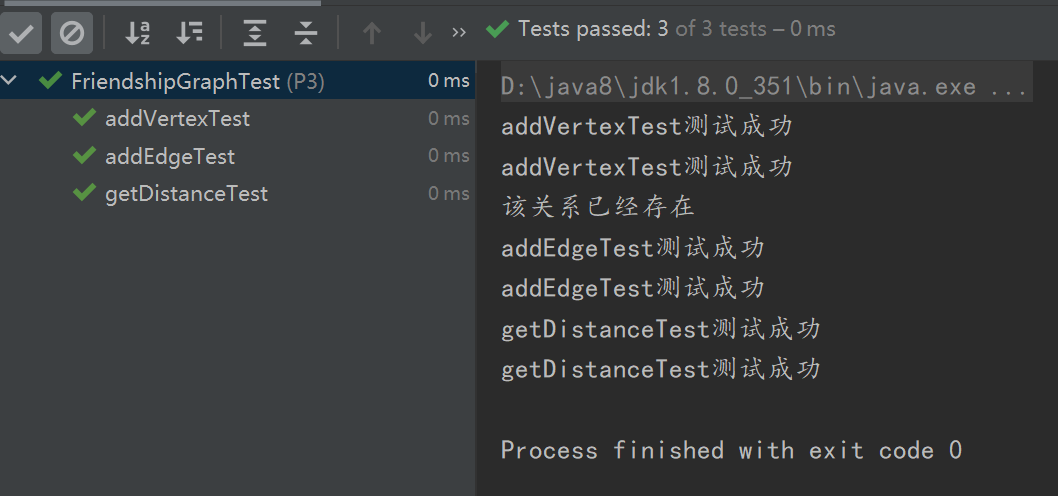


也符合预期。

### 设计/实现测试用例

给出你的设计和实现思路/过程/结果。

addVertexTest函数测试正常输入情况下函数是否返回true，以及重名情况下是否能catch到异常；addEdgeTest测试正常输入情况下是否返回true，以及点不存在是否能catch到异常，还有在边已经存在时是否返回false；getDistanceTest函数测试两个点之间有路径时的距离，无路径时是否返回-1，自己与自己是否返回0，以及查找的点不存在时是否能catch到异常。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。

每次结束编程时，请向该表格中增加一行。不要事后胡乱填写。

不要嫌烦，该表格可帮助你汇总你在每个任务上付出的时间和精力，发现自己不擅长的任务，后续有意识的弥补。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 任务 | 实际完成情况 |
| 2022-03-10 | 19:00-20:30 | 编写问题1的isLegalMagicSquare函数并进行测试 | 按计划完成 |
| 2022-03-10 | 20:30-22:30 | 编写问题1的generateMagicSquare函数并测试 | 正常完成 |
| 2022-03-12 | 19:00-20:00 | 尝试修改P2中代码中的错误（因为包的问题） | 遇到困难，未完成 |
| 2022-03-12 | 20:00-21:30 | 经过查阅资料后直到Test文件中的报错是因为未导入junit4 | 终于完成 |
| 2022-03-13 | 19:00-20:00 | 编写问题2中的drawSquare、calculateRegularPolygonAngle、drawRegularPolygon函数 | 正常完成 |
| 2022-03-13 | 20:00-21:00 | 计算calculateBearingToPoint函数中需要的角的关系，进行编写和测试 | 持续失败 |
| 2022-03-13 | 21:00-22:10 | 经上网查阅资料并利用atan2函数完成calculateBearingToPoint的编写和测试 | 正常完成 |
| 2022-03-14 | 19:00-20:00 | 编写和测试问题2中calculateBearings函数和convexHull | convexHull函数未能如期完成 |
| 2022-03-14 | 20:00-22:00 | 在网上查阅资料理清思路后，再次编写convexHull函数，但仍未成功，最后发现是每次循环时未将angle\_min变量初始化导致死循环 | 终于完成 |
| 2022-03-15 | 19:00-21:00 | 查阅资料后编写问题3中FriendshipGraph类和Person类 | 正常完成 |
| 2022-03-15 | 21:30-22:00 | 发现函数不符合预期，不能检测同名Person，稍作修改 | 正常完成 |
| 2022-03-18 | 19:00-20:00 | 编写问题3中FriendshipGraphTest类，并完成测试 | 正常完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的困难 | 解决途径 |
| isLegalMagicSquare函数文件读入和处理不熟悉 | 在编写前先进行资料查阅和网课学习 |
| P2中未修改的代码疯狂报错 | 在每个缺少包名的位置前面加上P2.，查阅资料后知道要导入junit4包 |
| convexHull函数一直死循环 | 检查两小时后发现每次循环时未将angle\_min变量初始化导致死循环 |

# 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）

本次实验我学会了java的基本使用和idea的基本使用，熟悉了java的编程过程，收获颇丰。也知道了遇到不会的问题一定要及时上网查阅资料，有很大可能会拓宽自己的思路，不要钻牛角尖。

## 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？

比较对口味，面向对象的编程和众多功能强大的类是java的优势，但代码可能相对某些语言来说不够简洁。

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

Idea的优势是强大的功能和简介美观的界面，但付费是它的不足。

1. 关于Git和GitHub，是否感受到了它在版本控制方面的价值；

深切的感受到了git在版本控制方面的强大。

1. 关于CMU和MIT的作业，你有何感受；

MIT的作业涉及的java知识更加丰富，注重考验对java知识的掌握，趣味性很浓；CMU作业偏向考验算法能力，趣味性稍差。

1. 关于本实验的工作量、难度、deadline；

对于刚接触或未接触过java的同学工作量偏大，因为需要画大量时间在网上学习java基础，难度适中，时间较为充足。

1. 关于初接触“软件构造”课程；

很好，很棒，很有意义。