

**2018年春季学期  
计算机学院大二软件构造课程**

**Lab 1实验报告**

|  |  |
| --- | --- |
| 姓名 | 刘胡超 |
| 学号 | 1163710222 |
| 班号 | 1603002 |
| 电子邮件 | 1249490233@qq.com |
| 手机号码 | 18846196338 |

**目录**

[1 实验目标概述 1](#_Toc506282558)

[2 实验环境配置 1](#_Toc506282559)

[3 实验过程 1](#_Toc506282560)

[3.1 Magic Squares 1](#_Toc506282561)

[3.1.1 isLegalMagicSquare() 1](#_Toc506282562)

[3.1.2 generateMagicSquare() 1](#_Toc506282563)

[3.2 Turtle Graphics 1](#_Toc506282564)

[3.2.1 Problem 1: Clone and import 2](#_Toc506282565)

[3.2.2 Problem 3: Turtle graphics and drawSquare 2](#_Toc506282566)

[3.2.3 Problem 5: Drawing polygons 2](#_Toc506282567)

[3.2.4 Problem 6: Calculating headings 2](#_Toc506282568)

[3.2.5 Problem 7: Personal art 2](#_Toc506282569)

[3.2.6 Submitting 2](#_Toc506282570)

[3.3 Social Network 2](#_Toc506282571)

[3.3.1 设计/实现FriendshipGraph类 2](#_Toc506282572)

[3.3.2 设计/实现Person类 2](#_Toc506282573)

[3.3.3 设计/实现客户端代码main() 2](#_Toc506282574)

[3.3.4 设计/实现测试用例 2](#_Toc506282575)

[3.4 Tweet Tweet（选作，额外记分） 3](#_Toc506282576)

[4 实验进度记录 3](#_Toc506282577)

[5 实验过程中遇到的困难与解决途径 3](#_Toc506282578)

[6 实验过程中收获的经验、教训、感想 3](#_Toc506282579)

# 实验目标概述

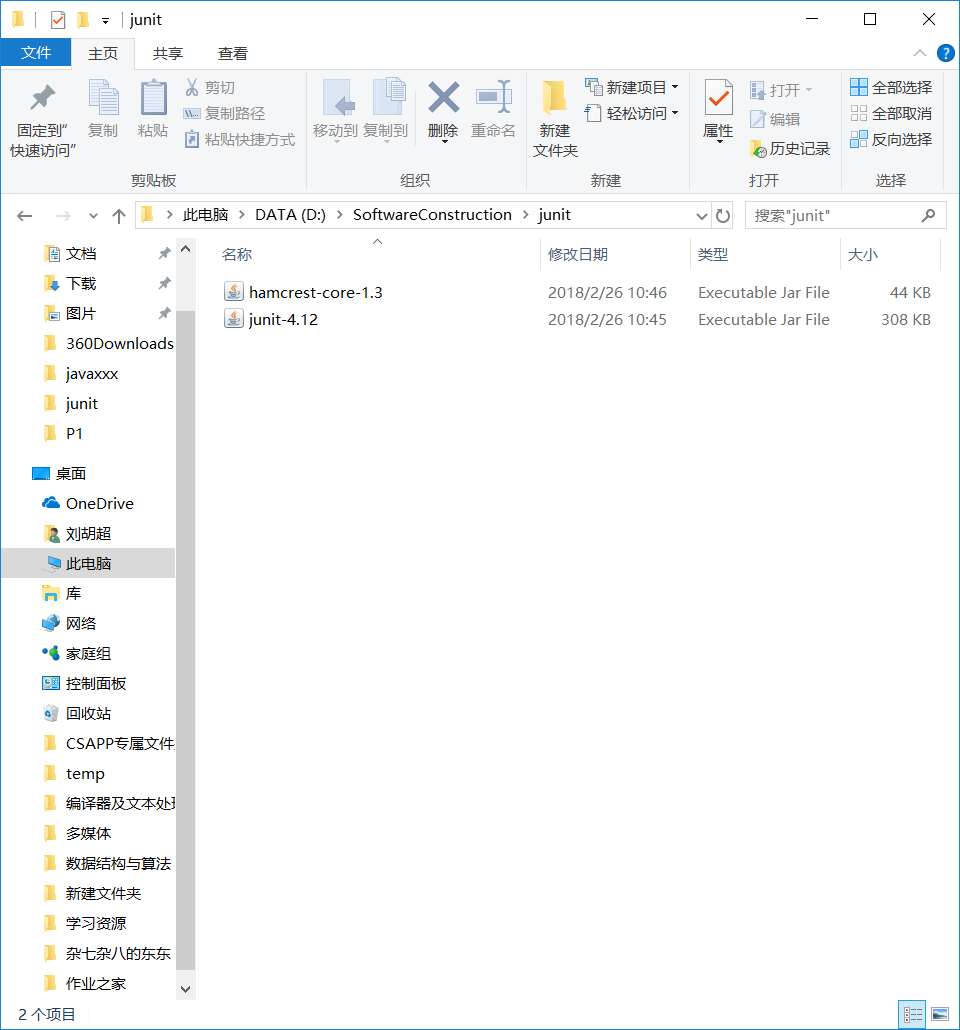
# 通过求解四个问题，训练基本 Java 编程技能，从而使得学生能够利用 Java OO 开发基本的功能模块，能够使用Eclipse IDE进行JAVA编程，能够阅读理解已有代码框架并根据功能需求补全代码，能够为所开发的代码编写基本的测试程序并完成测试，初步保证所开发代码的正确性。另一方面，利用 Git 作为代码配置管理的工具，学会 Git 的基本使用方法。

# 实验环境配置

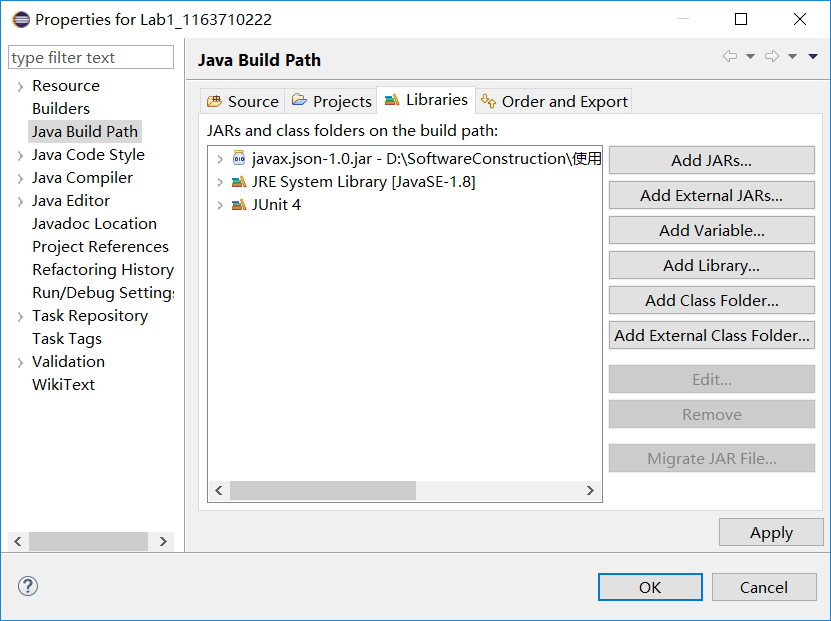
配置环境过程：

1.安装jdk和jre以及Eclipse。由于在此之前学习过JAVA，所以没有遇到太多的安装问题。而且jdk版本也恰好是实验要求的SE8。

2.安装junit，由于实验提供的链接下载的junit无法使用，



在与同学的讨论中得知Eclipse中自带junit4的jar,因此直接导入自带的junit4，成功完成配置。



3.在进行实验四时，需要导入json的jar,由于老师上课讲过P4中带有该jar因此比较顺利地完成了导入。

**Github Lab1仓库URL地址**: <https://github.com/ComputerScienceHIT/Lab1-1163710222> （Lab1-1163710222）

# 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（但千万不要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

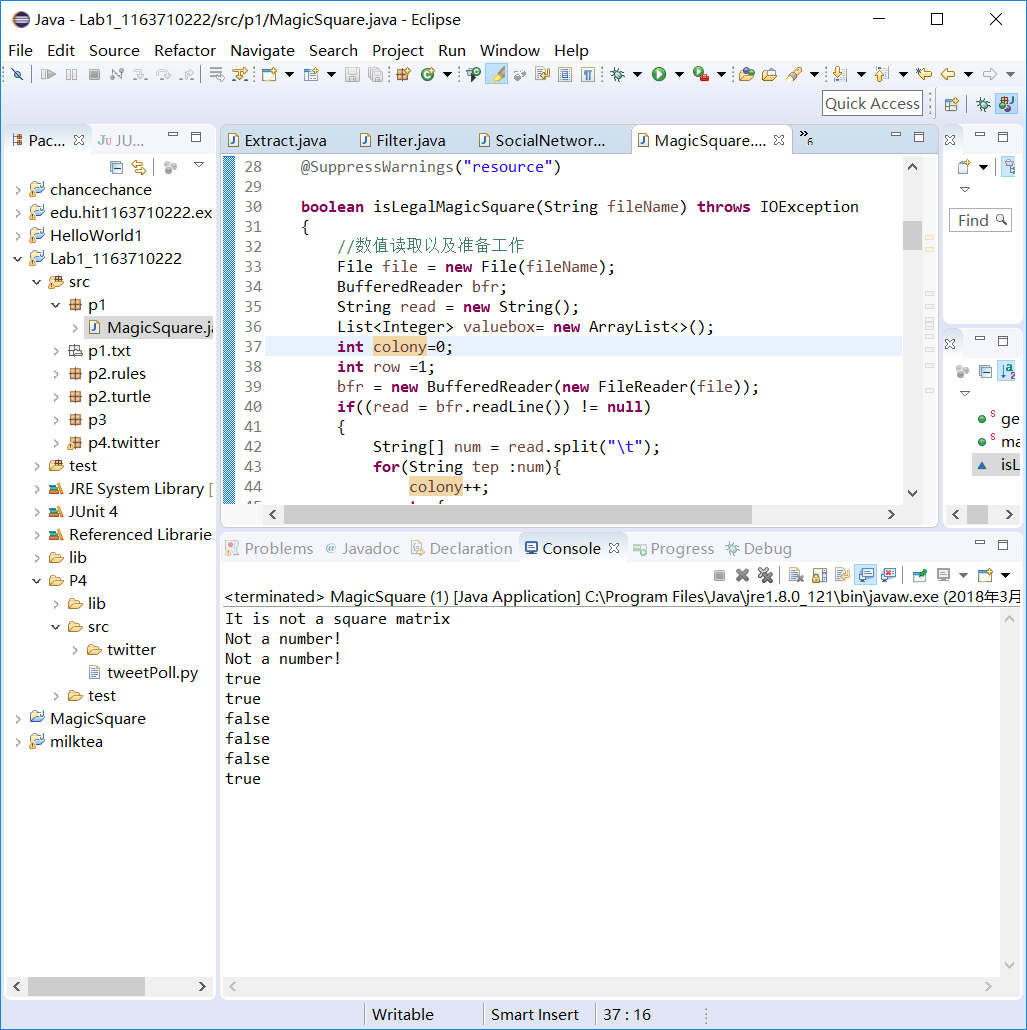
我认为该任务的目的是练习java之中比较复杂的文件输入输出流，字符串处理以及基本的数组定义，数组列表的使用，除此之外是最为基本的循环，判断等结构，从而使得我们能够熟悉最为基本的java知识。

### isLegalMagicSquare()

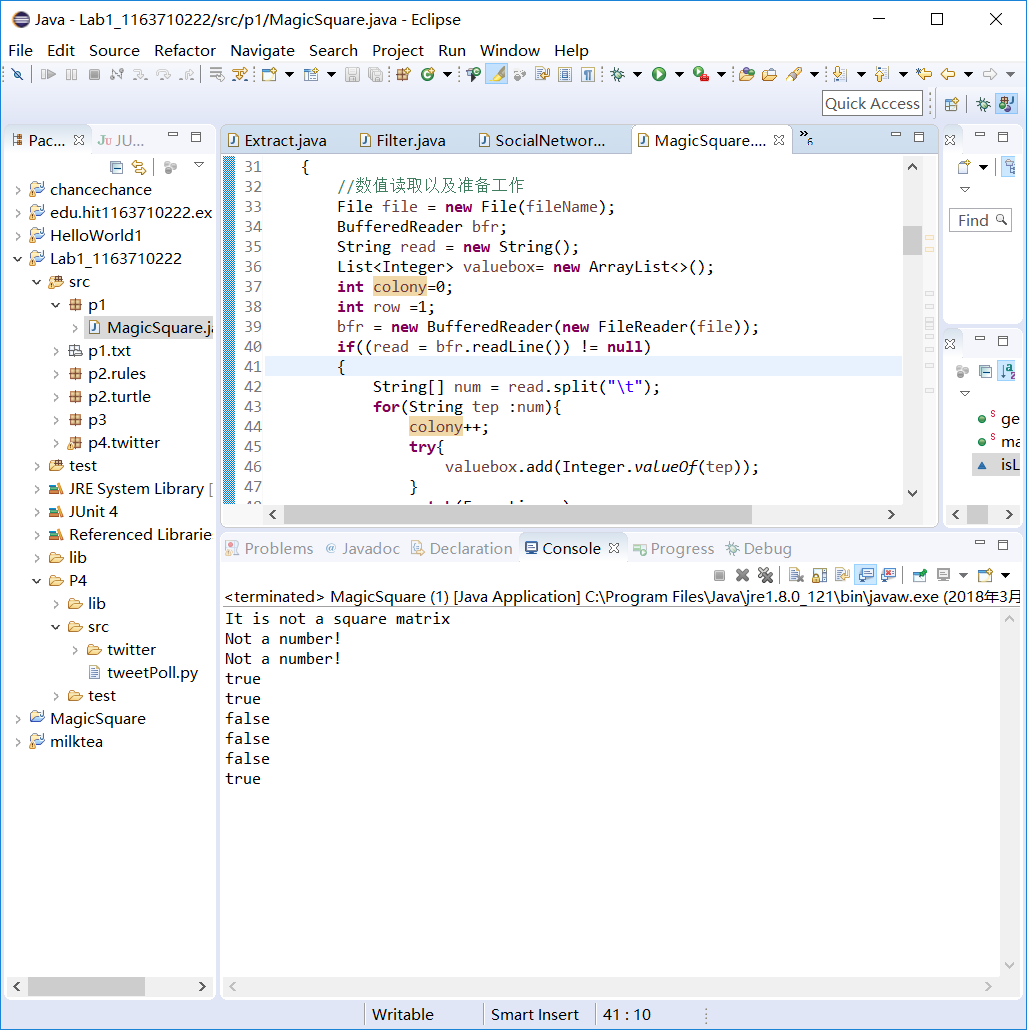
**实现思路**：首先需要从5个txt文件中读取数据，然后通过核对行列数，核对每行元素，核对分割字符是否为“\t”，核对每行每列每一对角线的数值是否相等，从而判断是否满足要求，若不满足要求，则输出错误原因，并且返回false。

**实现过程**：

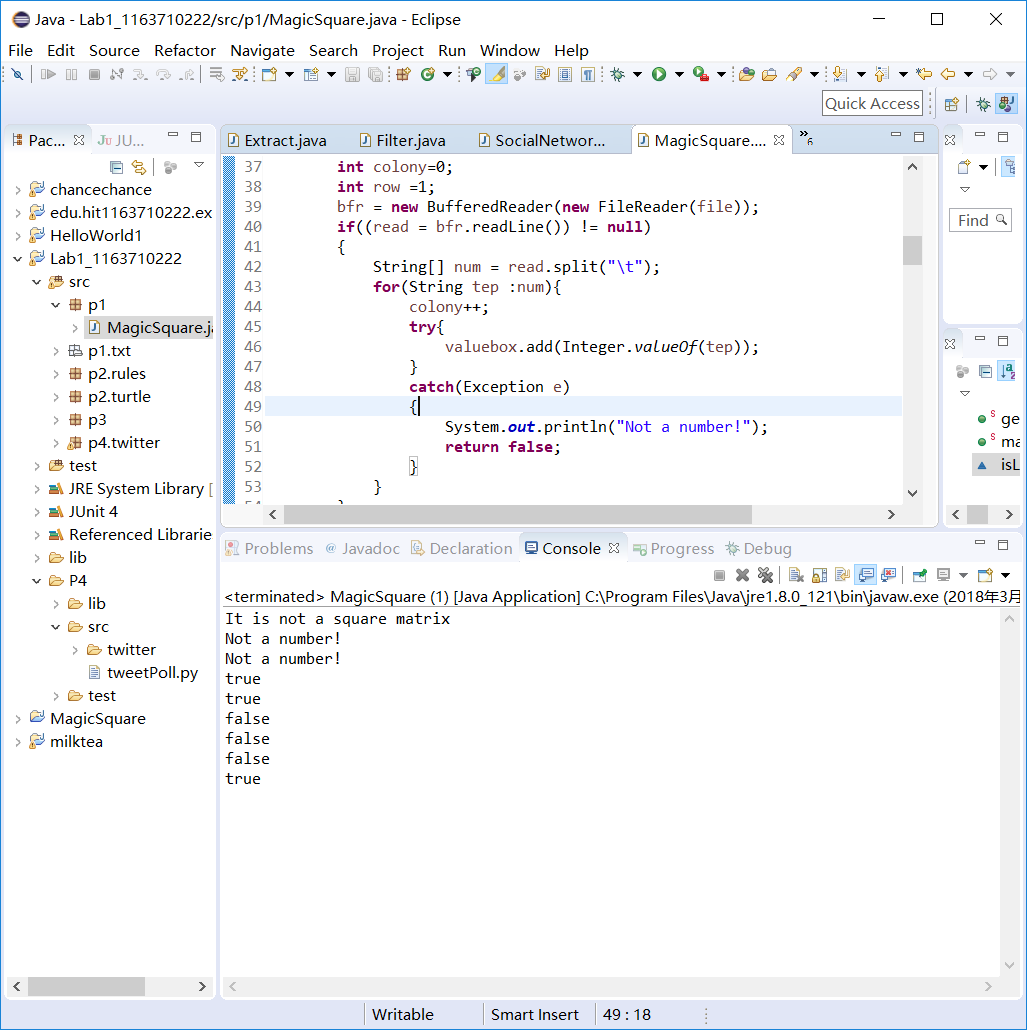
(1).读取文件中的数据，使用bufferedReader,利用文件内容进行初始化。



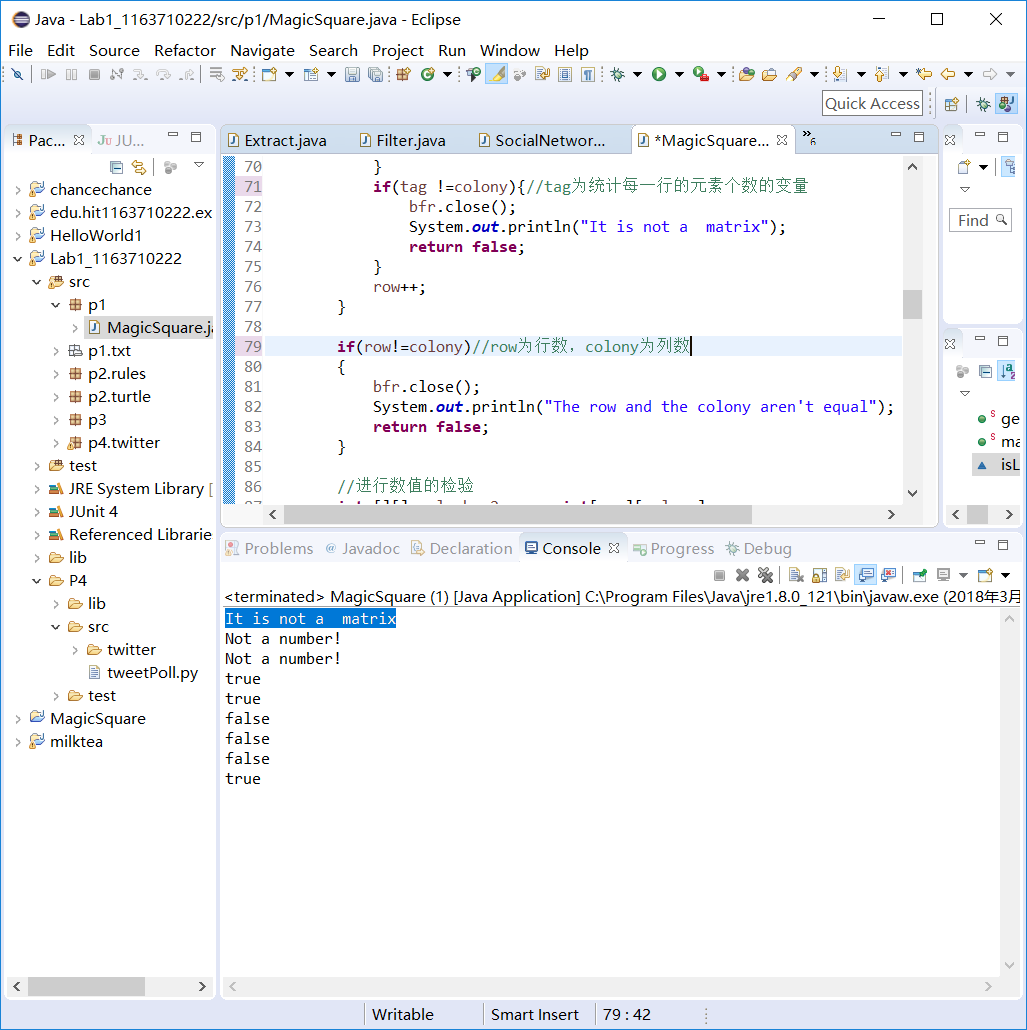
利用readLine以及while进行逐行循环读取，从而并且使用split方法对读取的字符串进行分割并且将其存放在字符串数组之中。



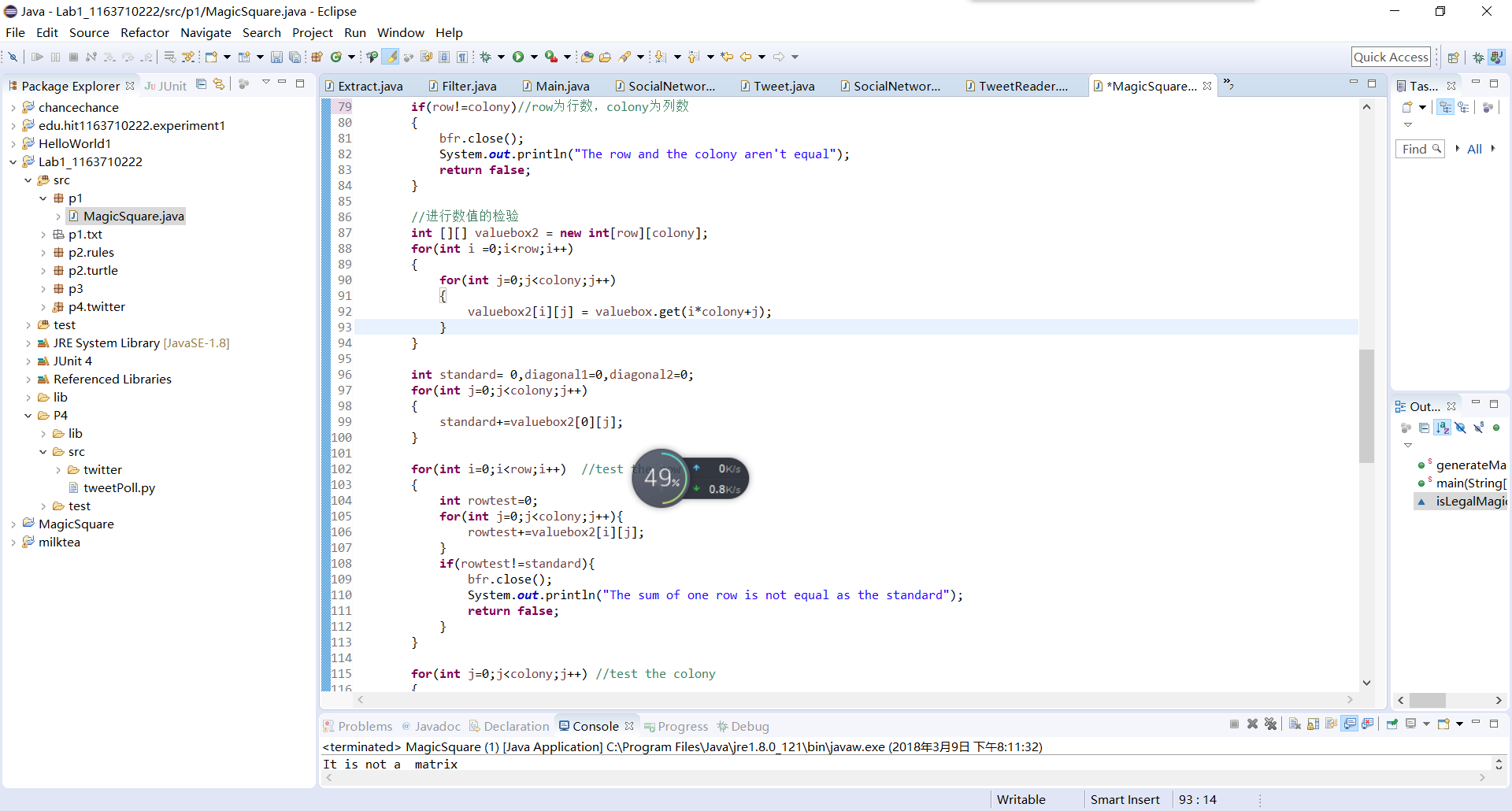
(2).在判断读取的数据是否为正整数时，使用try catch结构，利用非Integer元素使用valueof进行转化时抛出异常的特点进行捕捉，在控制台打印“Not a number!”的错误提示信息，从而完成对分割符是否为“\t”的判断。



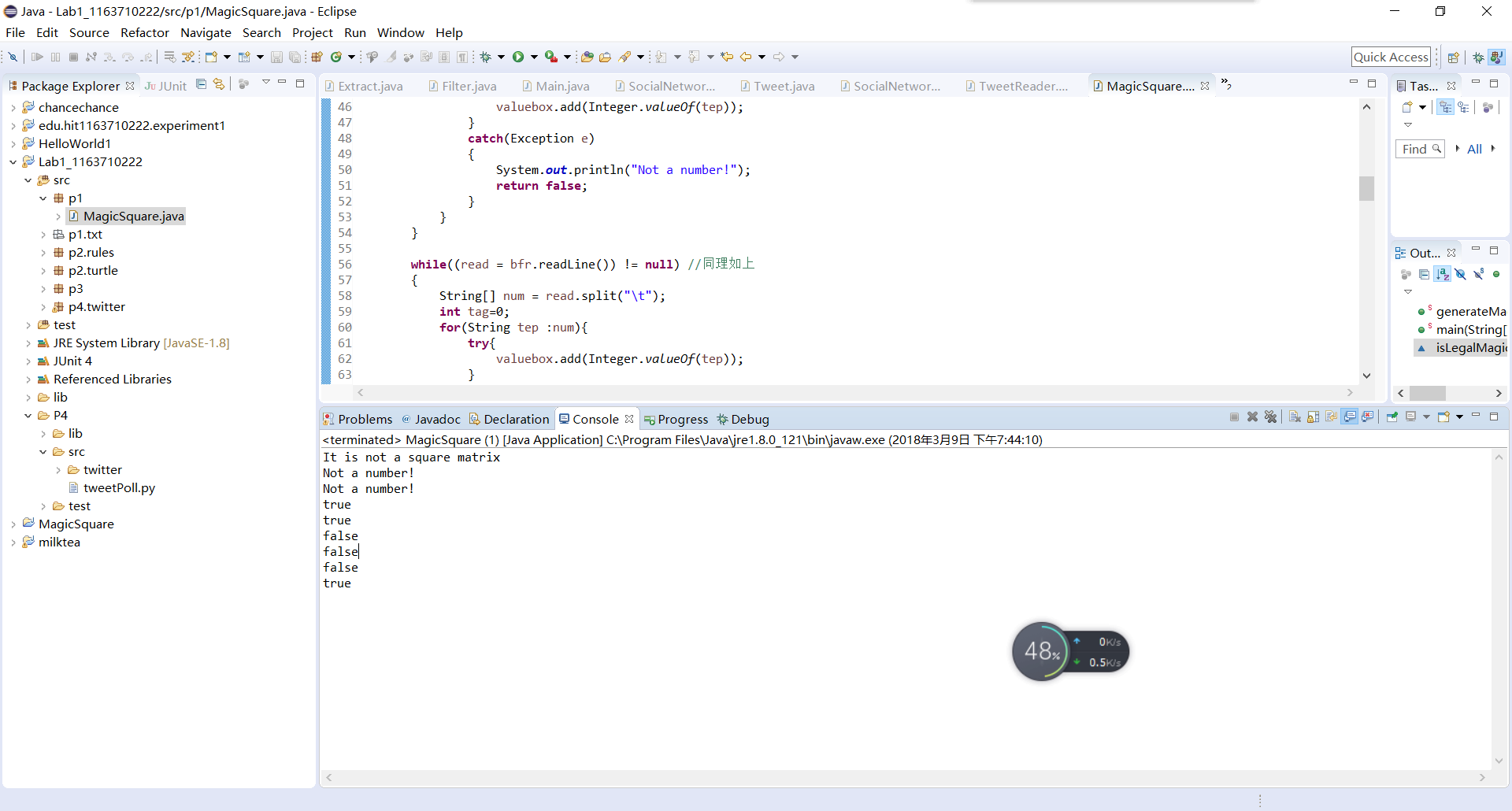
(3).在进行行列数比较，以及每行元素是否相等时，都使用第一行元素个数作为参照，从而得出结论，如果行列数不相同，则输出“It is not a square matrix”，若每行元素个数有不同，则输出“It is not a matrix”。



(4).在进行数值检验时将存储在数组列表中的元素存放到二维数组之中，使用二重循环即可进行判断，同时将第一行的数值作为standard，与其他数值进行比较，如果行不满足条件，则输出错误提示信息“The sum of one row does not equal to the standard ”，若列不满足，则输出“The sum of one colony does not equal to the standard”,若对角线不满足，则输出“The sum of one diagonal does not equal to the standard”。



**实现结果**：

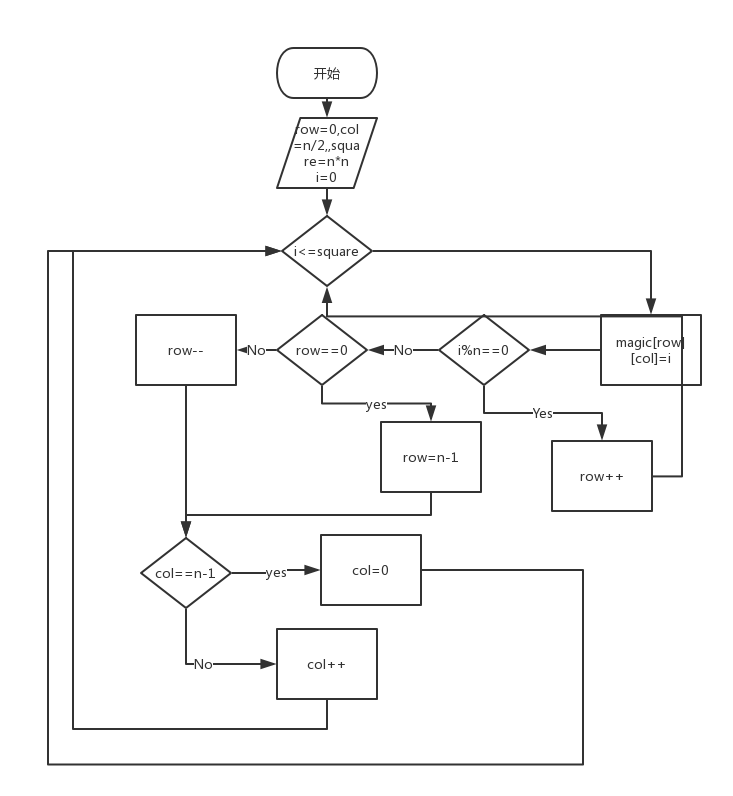


其中1.txt以及2.txt都是满足条件的MagicSquare,而3.txt不是满足正方形矩阵的条件，而4.txt以及5.txt均存在不以制表符分隔数据的情况。

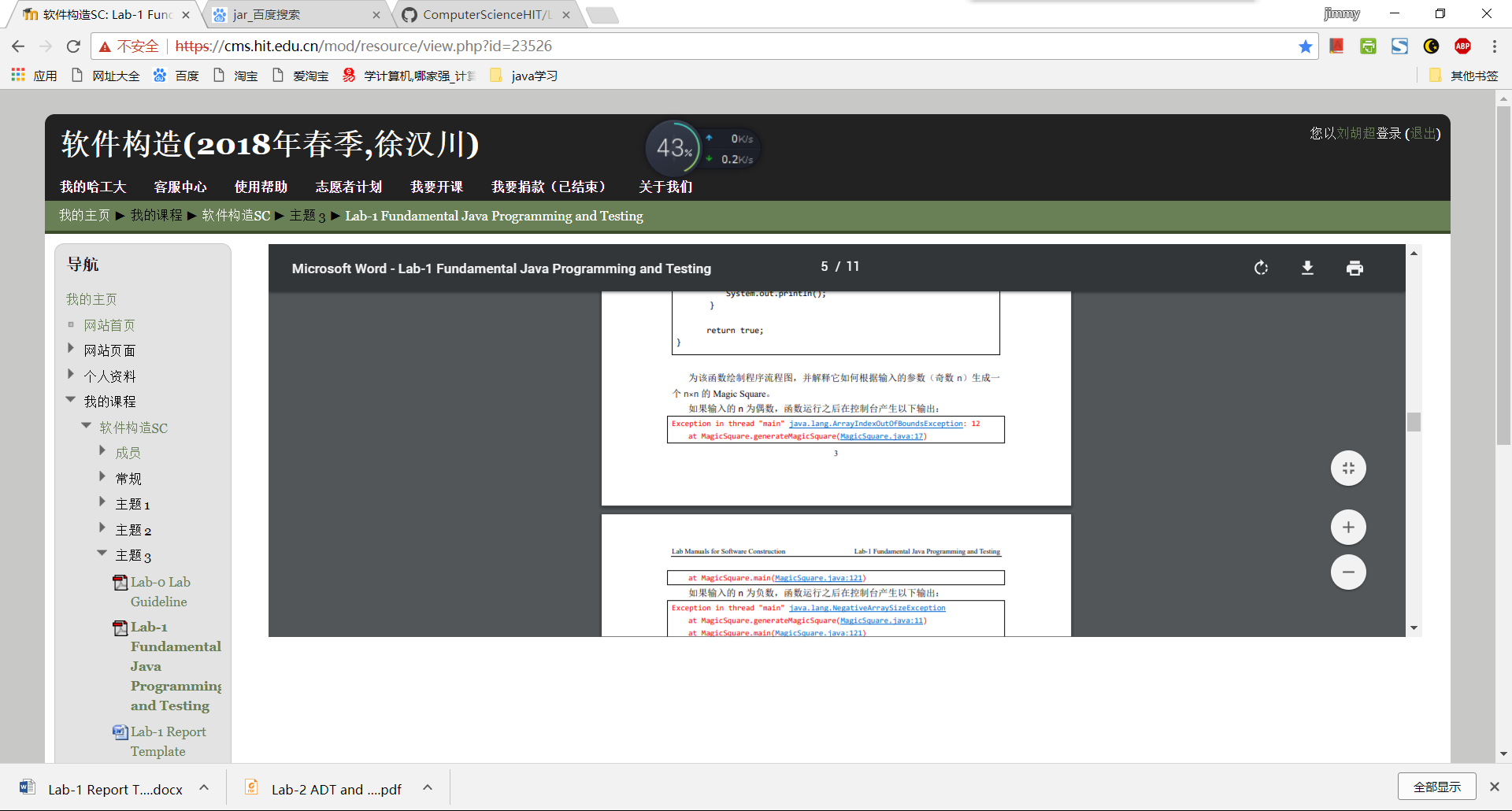
### generateMagicSquare()

**设计：**首先是解决输入偶数以及输入负数时出现报错的含义以及原因

函数流程图：



输入为奇数时：由于函数在row==0时重置行号以及在i%n==0时进行行号的增加，使得行号的变化始终是与i相适应的，每次运算都进行行列调整，使得数字出现在正确的位置上。

偶数：

输入偶数会导致数组越界访问。

负数：

输入负数会使得出现负数的数组索引

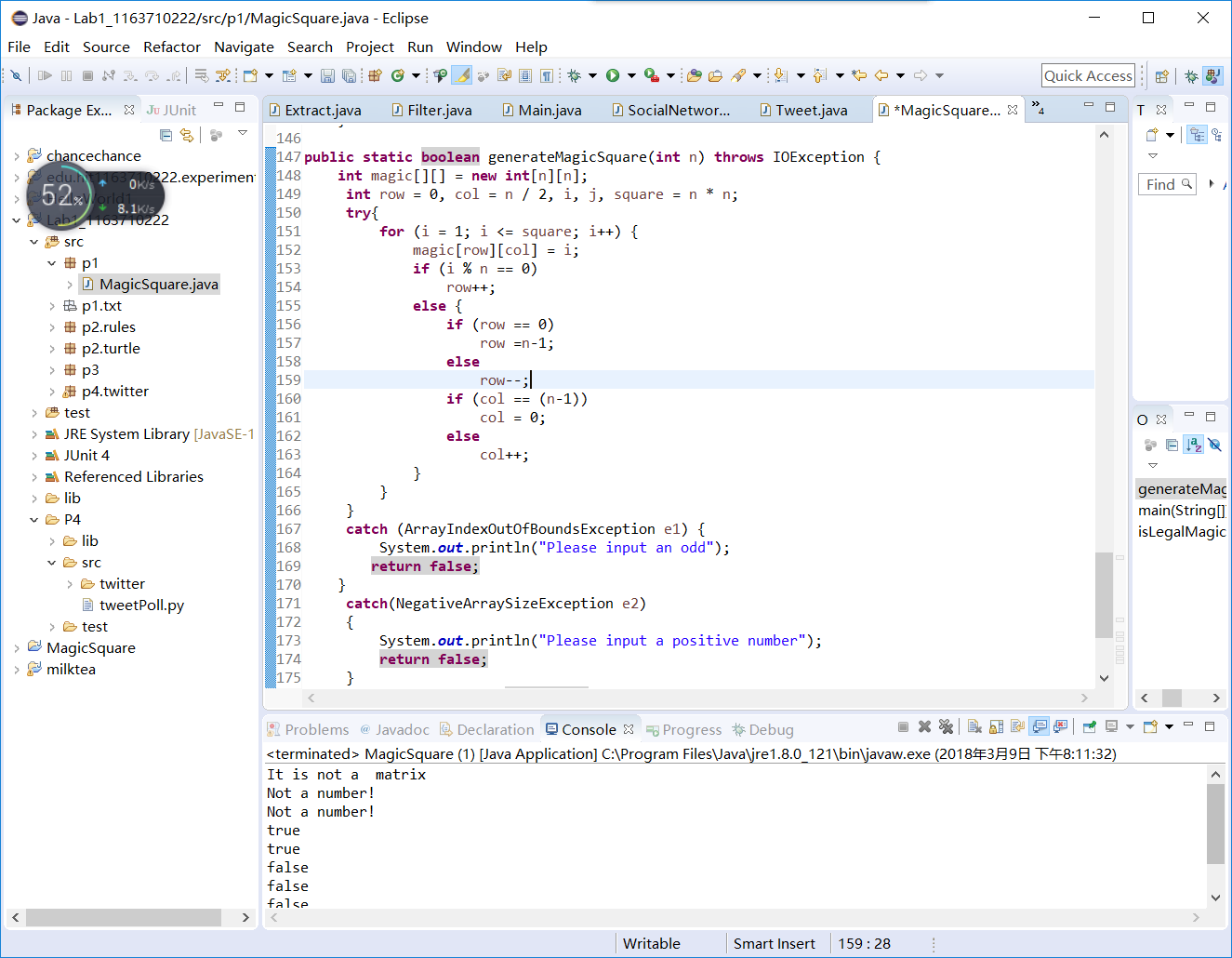
**实现思路：**

通过在函数中加入try catch语句块，当抛出ArrayIndexOutOfBound的错误时，输出”Please input an odd”.当抛出NegativeArraySizeException的错误时，输出“Please input a positive number”.

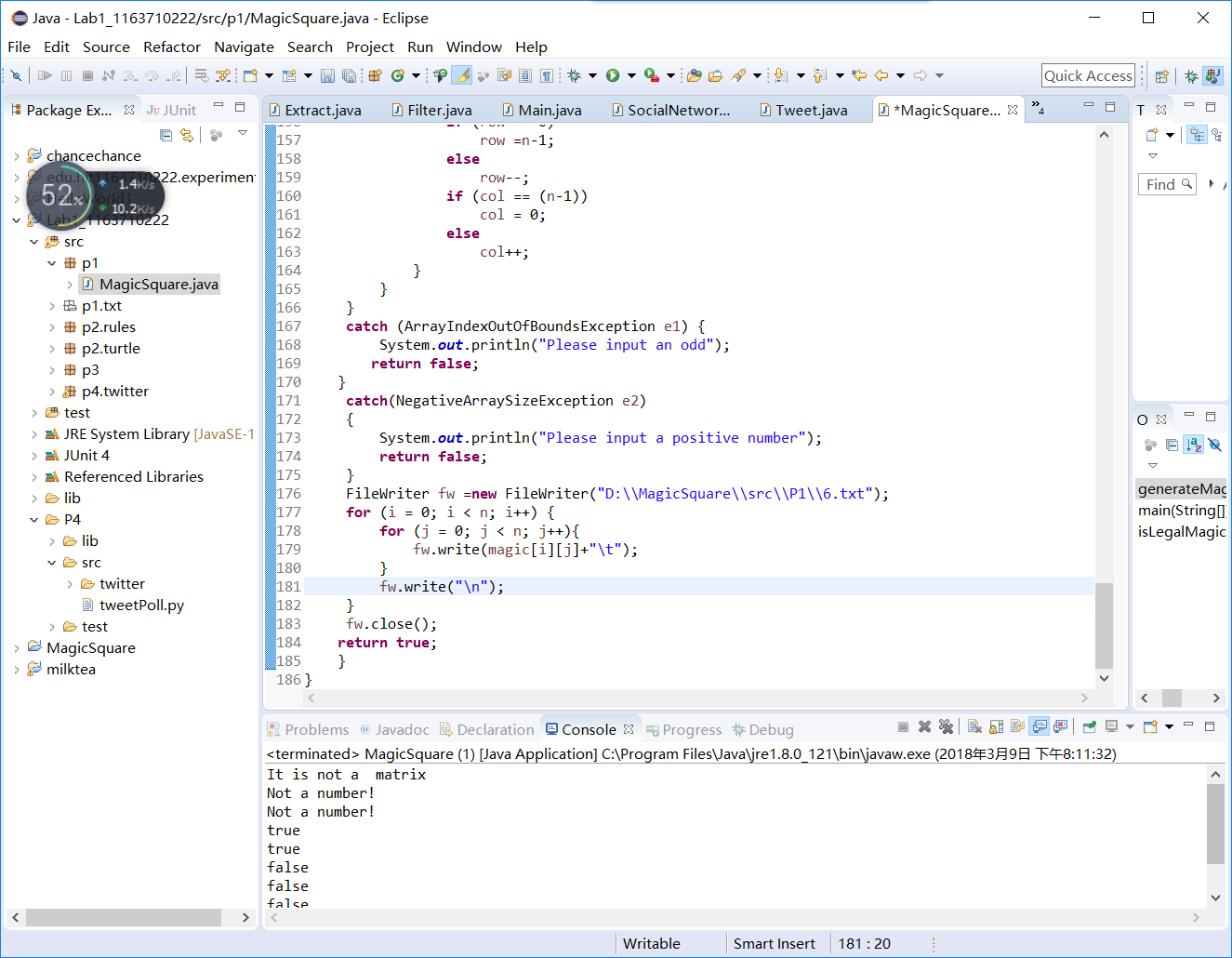
**实现过程：**

1.将实验报告提供的源码，进行理解

2.在进行MagicSquare生成阶段加入try catch语句块，根据抛出的错误进行输出。

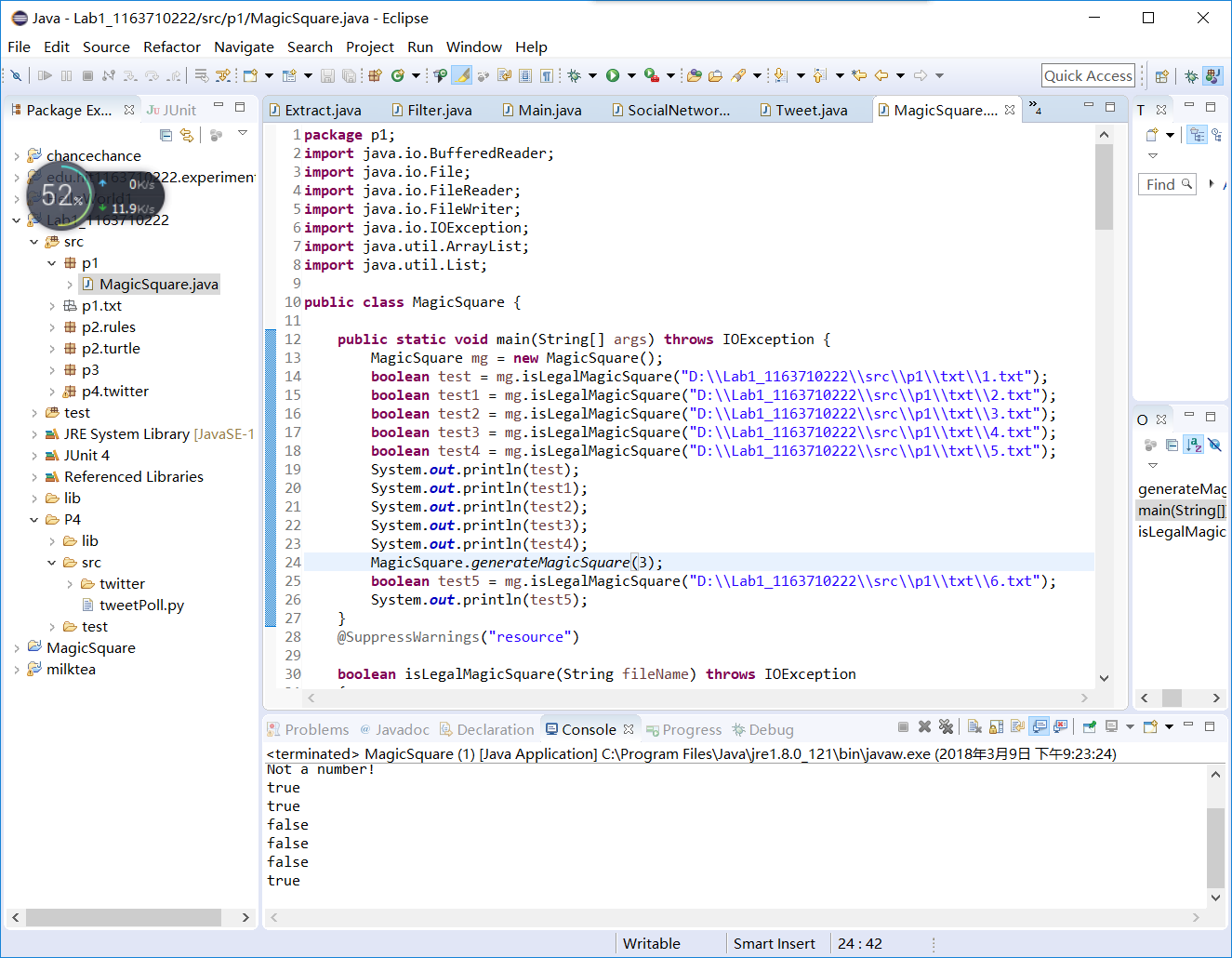


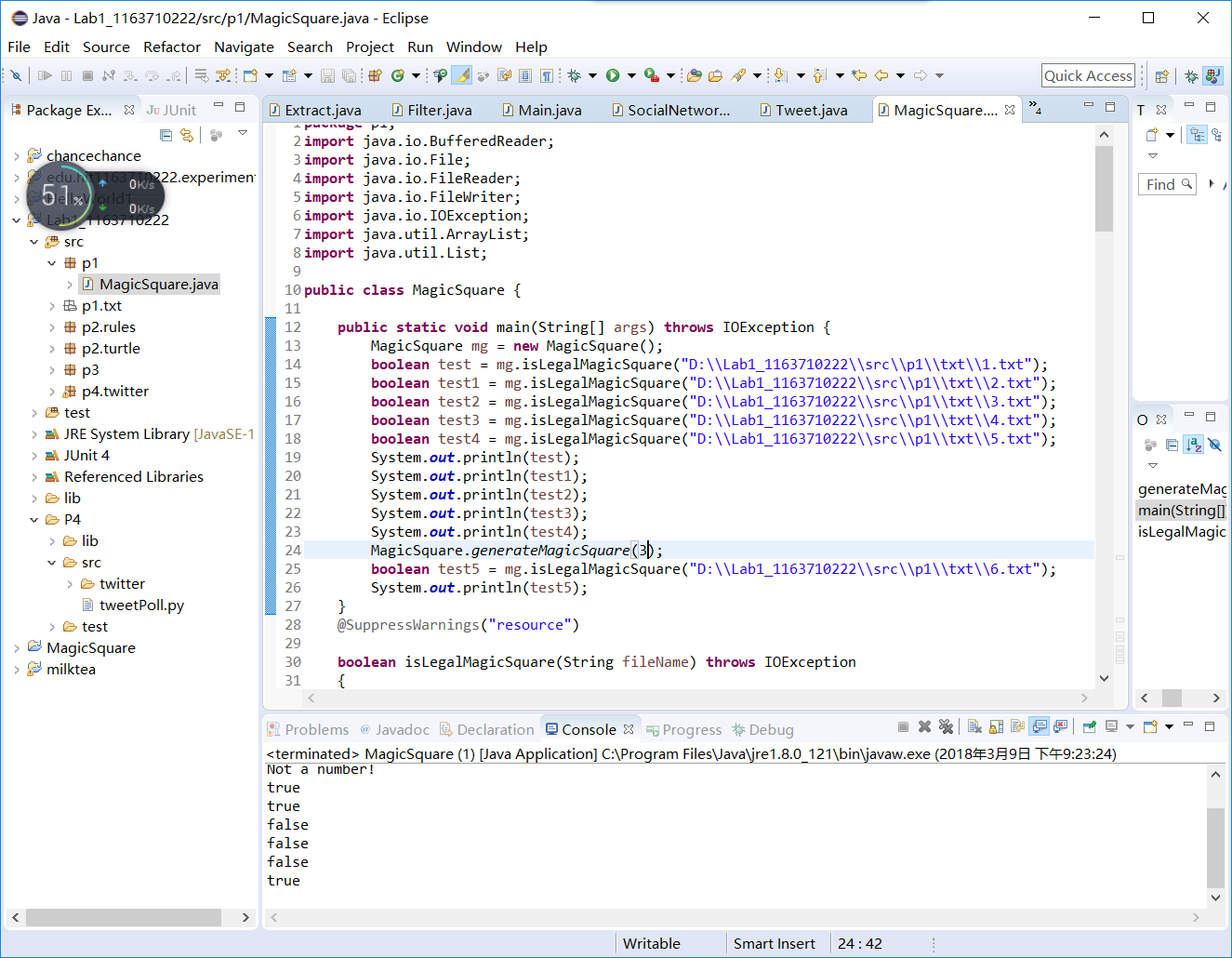
3.若输入的数据为奇数，则将数据写入文件，并且返回true.



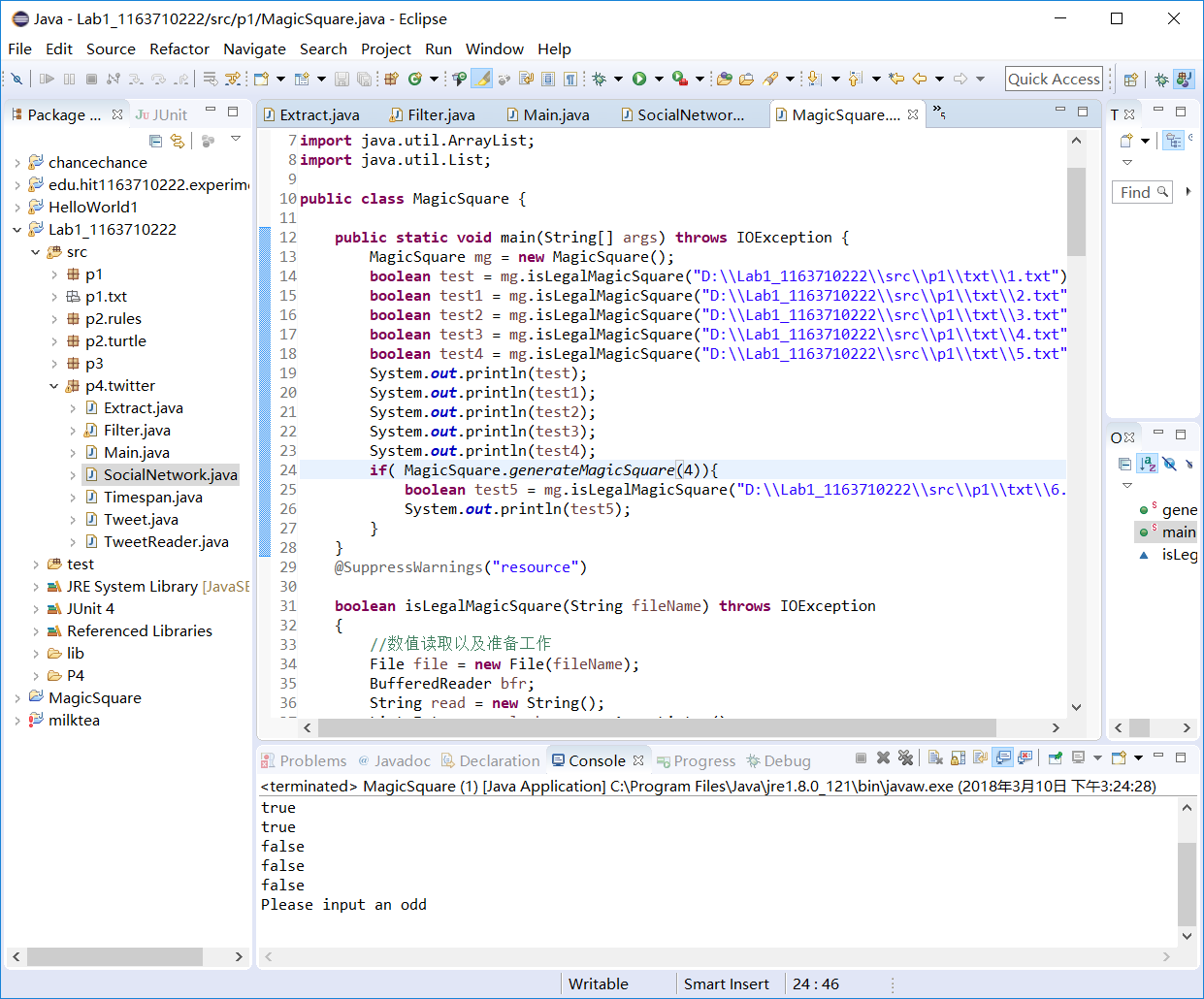
**实现结果：**

输入为3：

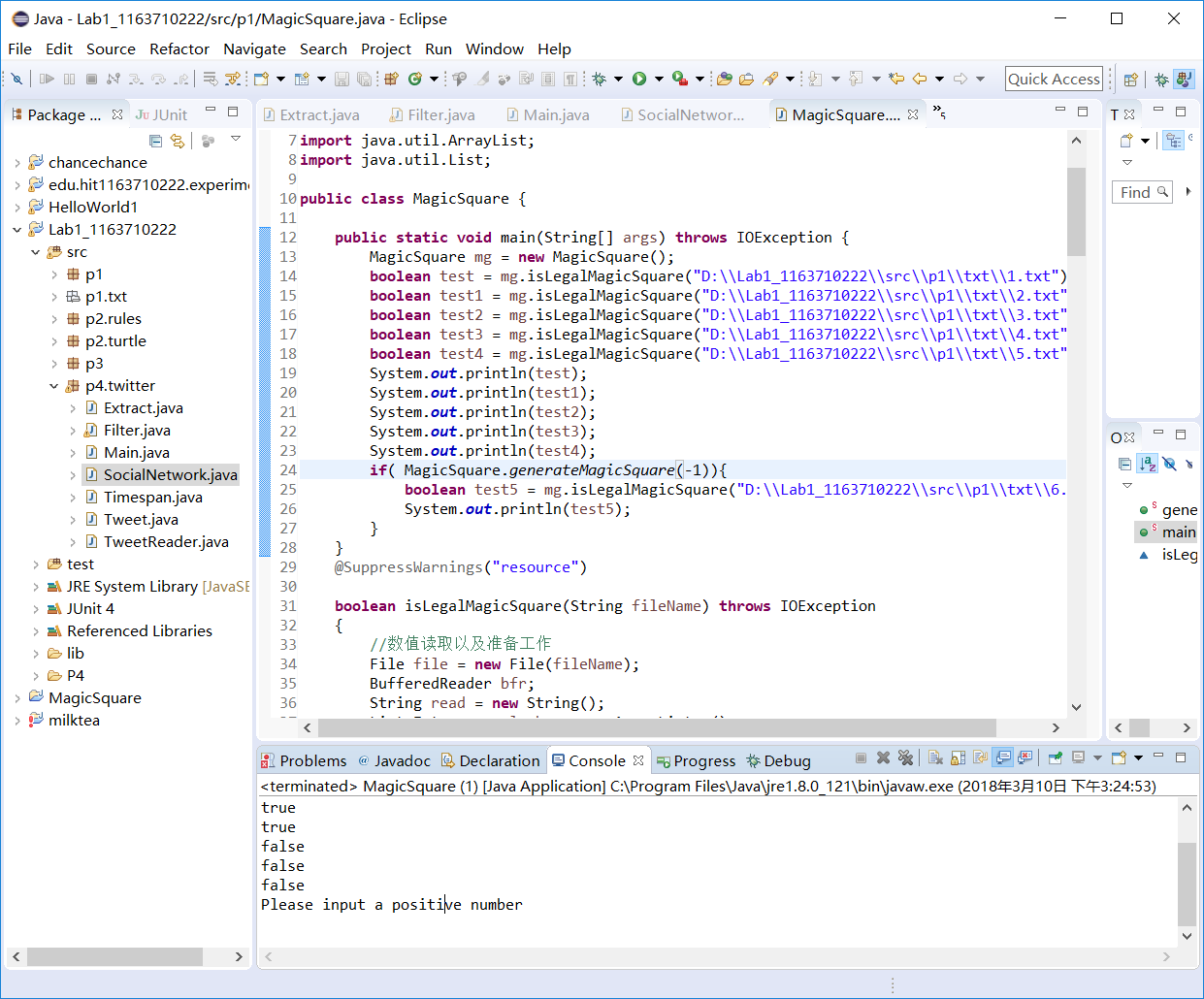




输入为4



输入为-1



## Turtle Graphics

在这里简要概述你对该任务的理解。

我认为这个任务的目的是让我们能够解除并且熟悉junit的单元测试以及java中的一些基本的API和方法的使用.

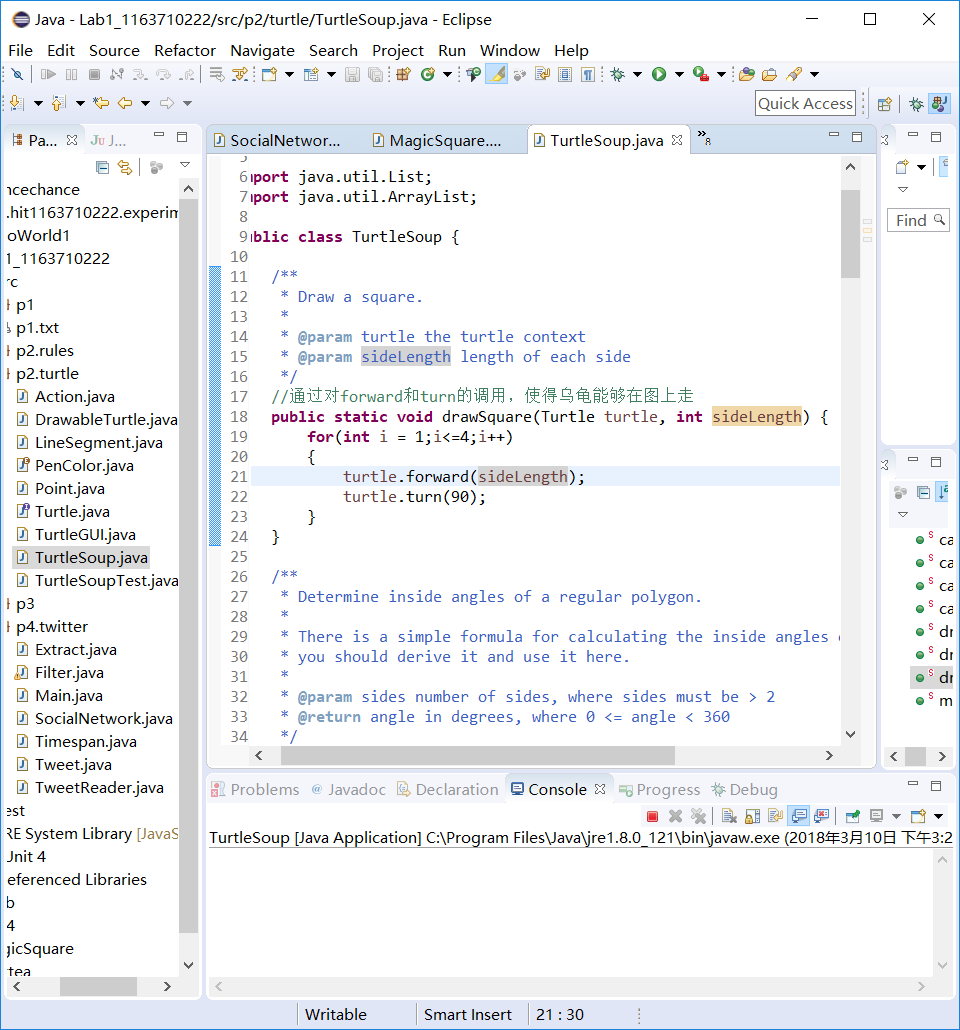
### Problem 1: Clone and import

下载git-64bit,建立本地repo,lab1-1163710222在命令行中输入git clone URL,将文件克隆到本机,每当完成一次修改时,用git add加入到暂存区,在使用git commit 将文件存储到本地仓库

### Problem 3: Turtle graphics and drawSquare

**思路**:简单的数学问题,循环加上函数调用即可解决

**过程**:



**结果**:

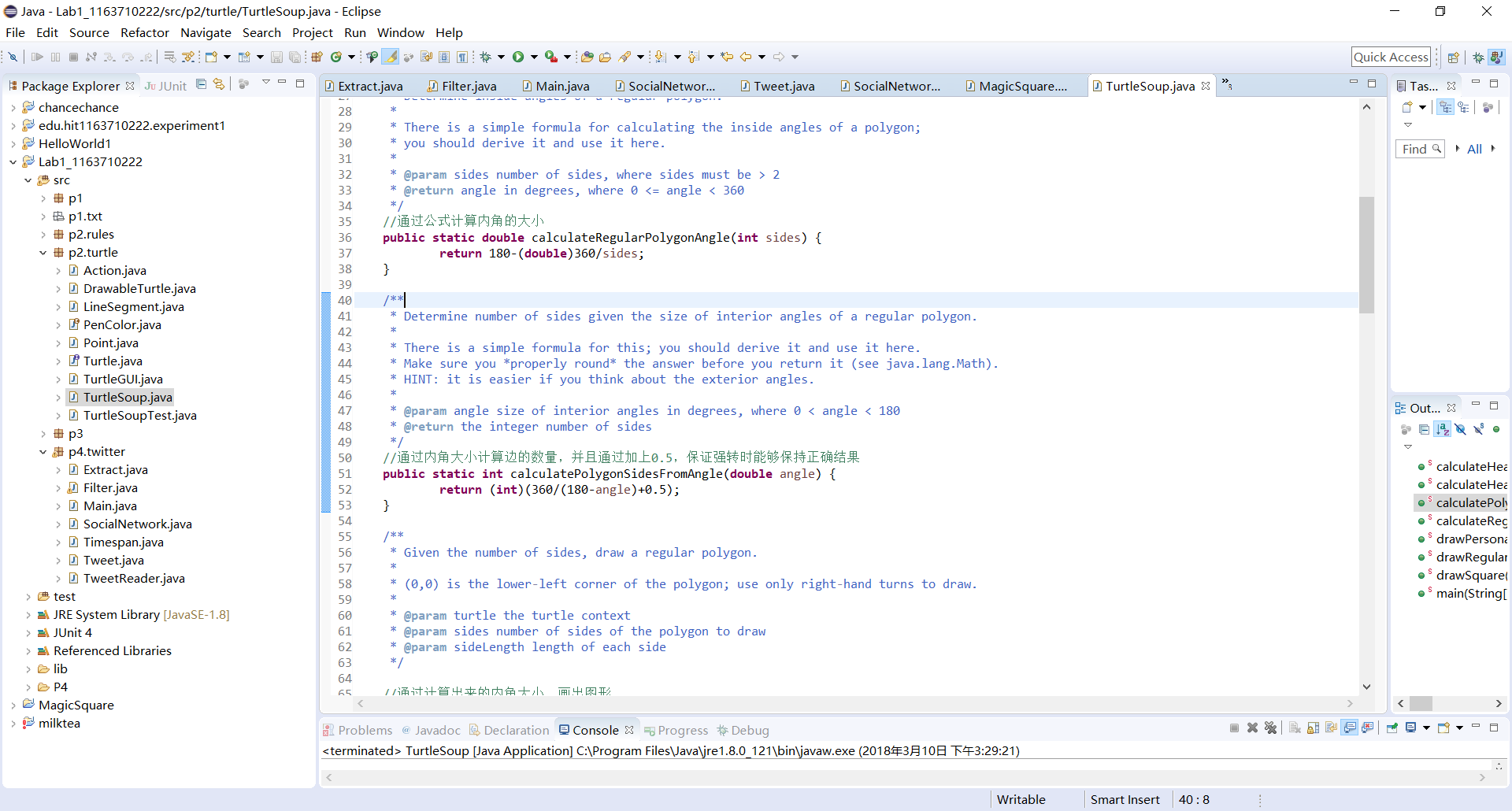


### Problem 5: Drawing polygons

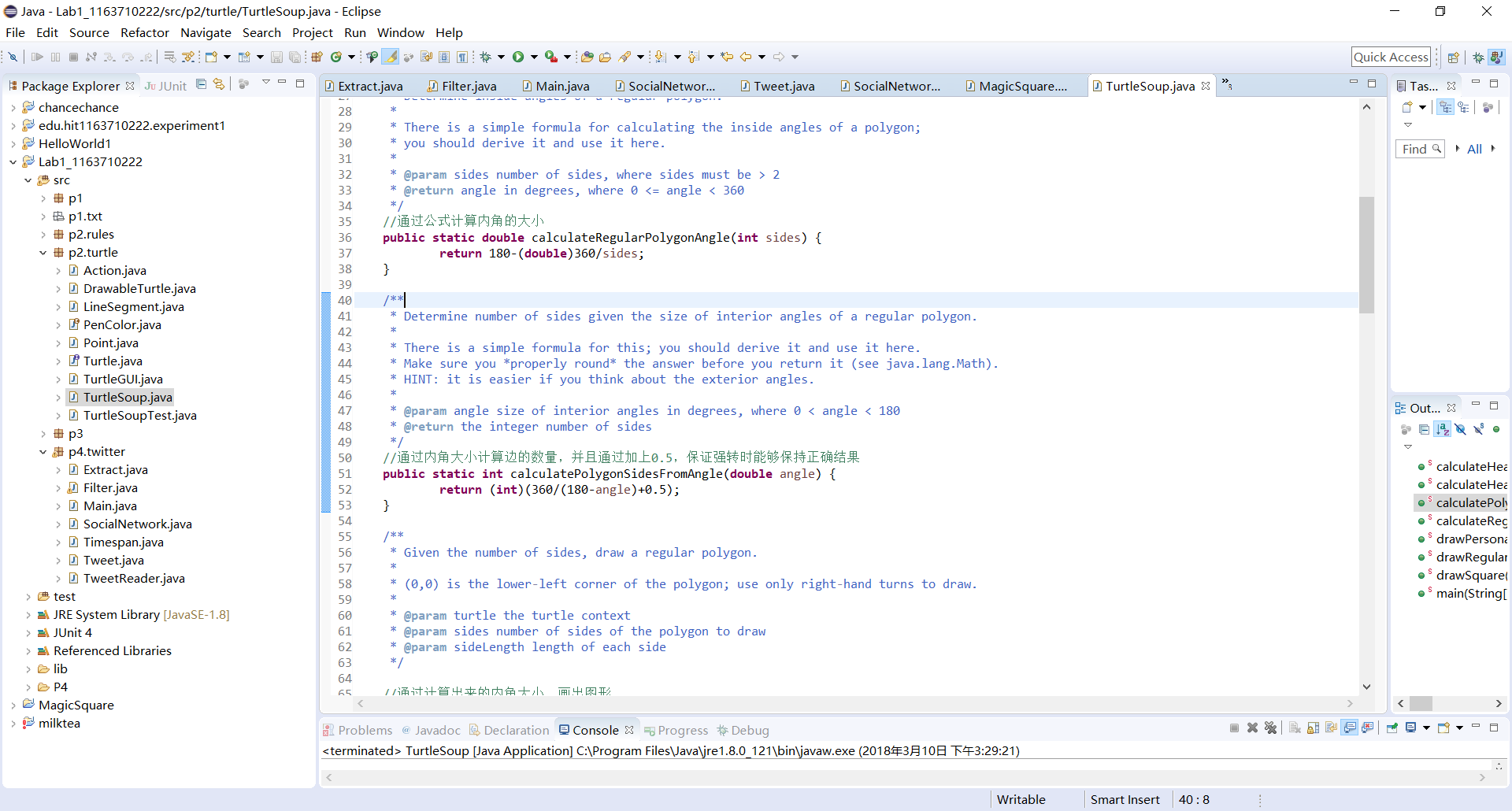
**思路**:弄清楚多边形内角大小,及时调整方向即可完成,调用计算角度的函数calculateRegularPolygonAngle以及计算边的数量的函数calculatePolygonSidesFromAngle即可.

**过程**:

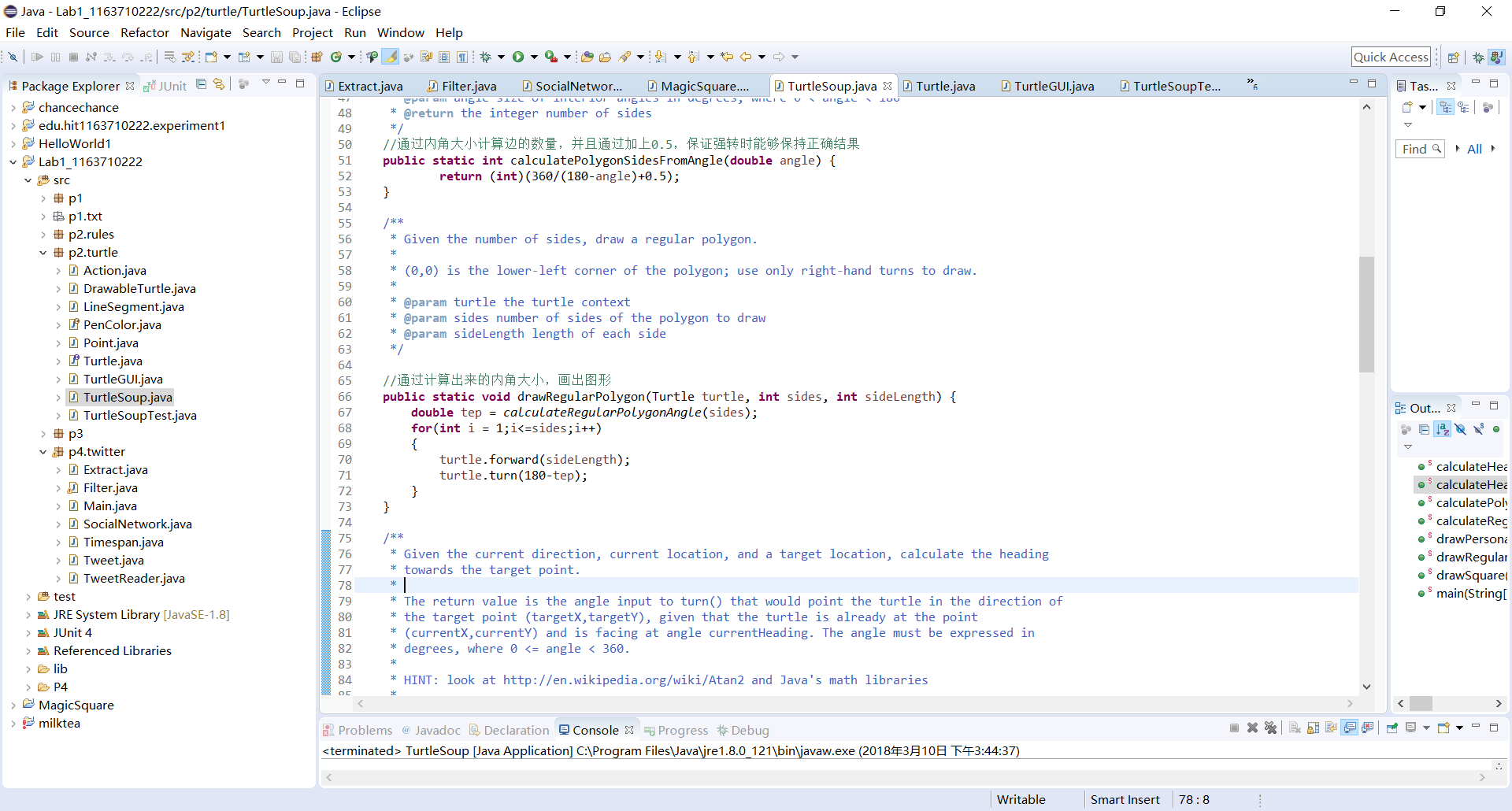
(1).根据公式,计算内角大小



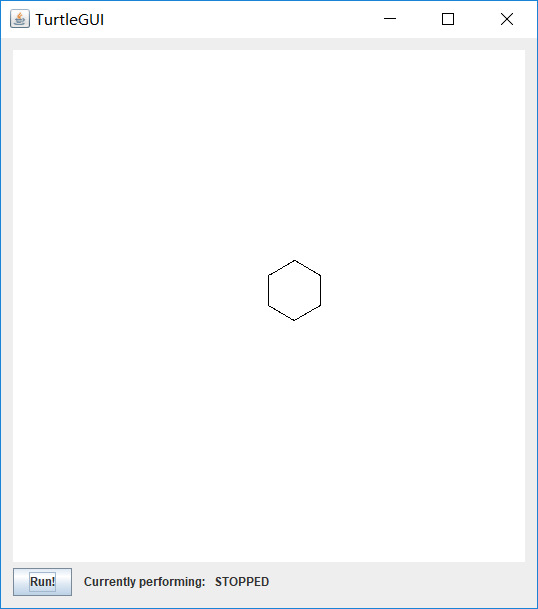
(2).根据给出的内角大小计算边的数量:



(3)通过计算的数据得出目标多边形:



**实验结果:**



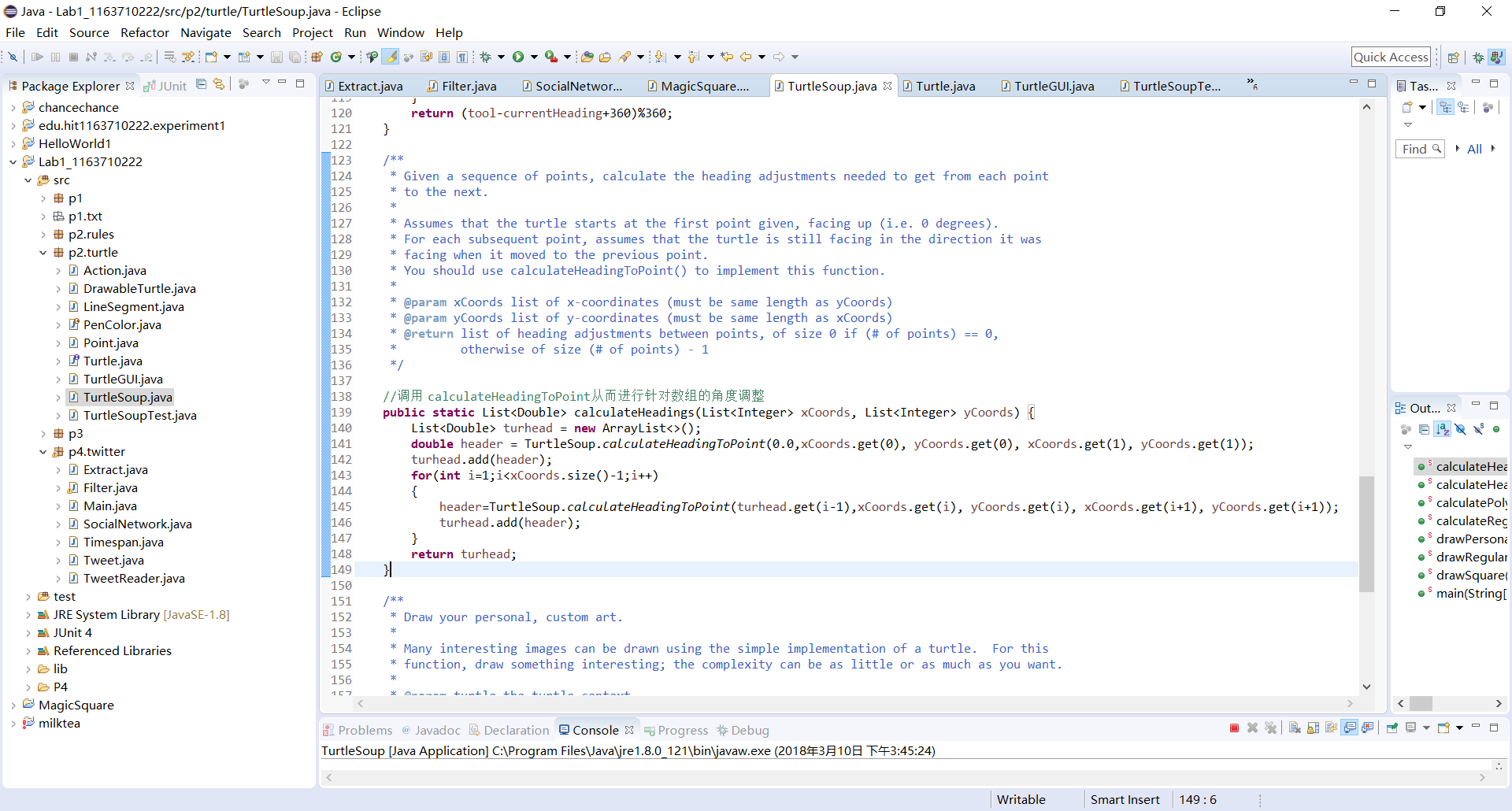
### Problem 6: Calculating headings

**设计思路:**首先使用java中所有的atan2函数,并且进行分类讨论,计算出要转向的角度.,然后利用java中的数组列表来完成连续转向的目的

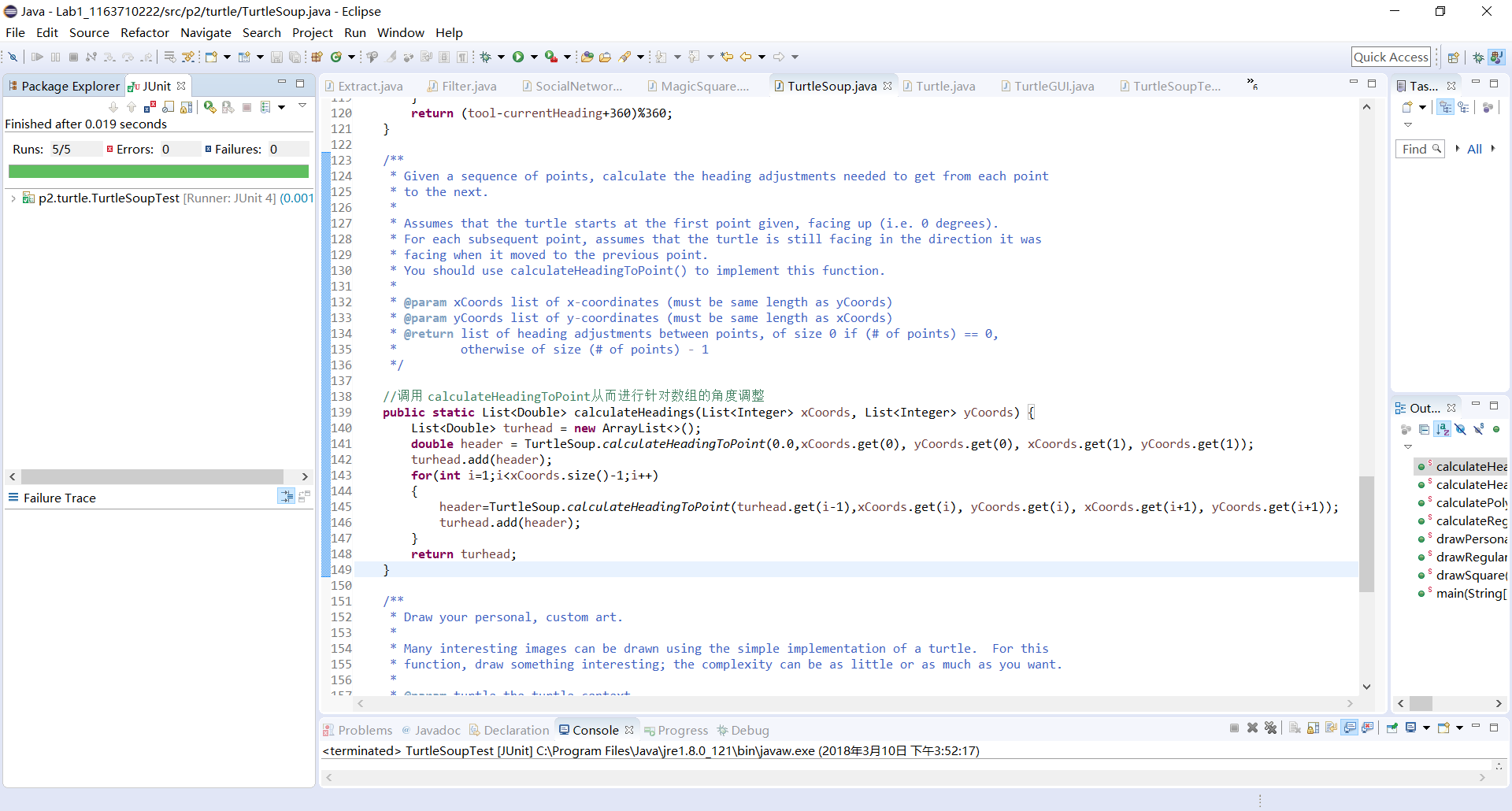
**过程**:

(1).计算目标指向的正切值,使用反三角函数得出角度,在根据角度的大小进行分类讨论,使得得出的角度始终在0-180.

(2).通过反复调用calculateHeadingToPoint()函数,从而使得turtle能够反复完成转换方向.



**结果:**

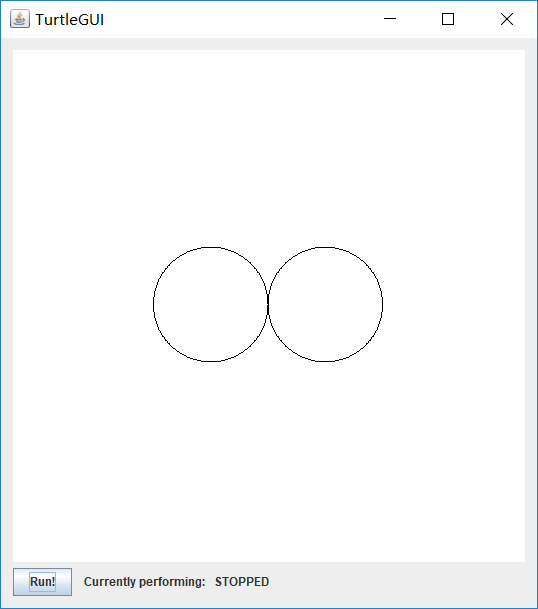


### Problem 7: Personal art

**思路:**简单地使用turtle画了两个圆,每次转动一个小的角度从而达到化直为曲的目的

**过程:** 

**结果**:



### Submitting

使用git push命令将本地仓库的最新文件复制到github中的repo中

## Social Network

我认为这一道题目的目的是让我们能够练习使用java完成广度优先搜索以及图的邻接表表示方法.。

### 设计/实现FriendshipGraph类

**设计:**定义数组列表FG,由于存储person,并且在FriendshipGraph中定义加入边,加入顶点的函数,在实现计算指定两人之间的距离时使用改良的广度优先搜索.

**实现**:

(1).定义名为FG的数组列表,为存储person做准备

(2)加入名为addVertex的函数,用于加入未曾出现的Person,若任命已经出现,则输出”Each person has a unique name”.

(3)加入名为addEdge的函数,用于加入人与人之间的关系,为无向图中加入元素.

(4)使用队列结构完成广度优先搜索,对每一层进行统计,若没有联系,则输出-1,否则输出距离.

**结果:**

|  |
| --- |
| **public** **class** FriendshipGraph {    **public** List<Person> FG;  **public** FriendshipGraph() {  FG = **new** ArrayList<>();  }  **public** **void** addVertex(Person p)  {  **for**(Person tep : FG)  {  **if**(tep.name.equals(p.name))  {  System.***out***.println("Each person has an unique name");  System.*exit*(0);  }  }  FG.add(p);  }  **public** **void** addEdge(Person p1,Person p2)  {  **for**(Person xx:p1.relation)  {  **if**(xx.equals(p2)&&!xx.equals(xx)){  System.***out***.println("The wrong relations");  System.*exit*(0);  }  }  p1.relation.add(p2);  p2.relation.add(p1);  }  **public** **int** getDistance(Person p1,Person p2)  {  Queue<Person> personBox1 = **new** LinkedList<>();  personBox1.add(p1);  **int** box = -1; ;  **if**(p1.equals(p2))  **return** 0;  **else**{  Person first = personBox1.peek();  Circle:**while**(personBox1!=**null**)  {  **if**(first.relation.size()==0)  **break** Circle;  **else**{  first = personBox1.peek();  **if**(first==**null**)  **break** Circle;  **else**  {  **for**(Person temp:first.relation)  {  temp.distance=first.distance+1;  **if**(temp.equals(p2)){  box = temp.distance;  **break** Circle;  }  **if**(temp.tag==**true**){  personBox1.add(temp);  temp.tag=**false**;  }  }  personBox1.remove(first);  }  }  }  **for**(Person abc : personBox1)  {  abc.distance=0;  abc.tag = **true**;  }  **return** box;  }  } |

### 设计/实现Person类

**设计**:

为person定义relation的数组列表,用于加入直接有关系的person.同时加入name, distance以及tag用于记录广度优先搜索中是否已经被遍历过.于此同时,改写函数的equals以及hashcode.

**过程:**

(1)定义name, tag, distance, relation,用于完成任务,在初始化时同时初始化所有属性

(2)改写equals方法,用于在判断时避免单纯地比较字符串内存地址.

**结果:**

|  |
| --- |
| **package** p3;  **import** java.util.ArrayList;  **import** java.util.List;  **public** **class** Person {    **public** String name;  **public** List<Person> relation;  **public** **boolean** tag ;  **public** **int** distance;//用于记录目标长度  Person(String name)  {  relation = **new** ArrayList<>();  **this**.name = name;  tag = **true**;  **this**.distance = 0;  }  @Override  **public** **int** hashCode() {  **final** **int** prime = 31;  **int** result = 1;  result = prime \* result + ((name == **null**) ? 0 : name.hashCode());  result = prime \* result + ((relation == **null**) ? 0 : relation.hashCode());  **return** result;  }  @Override  **public** **boolean** equals(Object obj) {  **if** (**this** == obj)  **return** **true**;  **if** (obj == **null**)  **return** **false**;  **if** (getClass() != obj.getClass())  **return** **false**;  Person other = (Person) obj;  **if** (name == **null**) {  **if** (other.name != **null**)  **return** **false**;  } **else** **if** (!name.equals(other.name))  **return** **false**;  **if** (relation == **null**) {  **if** (other.relation != **null**)  **return** **false**;  } **else** **if** (!relation.equals(other.relation))  **return** **false**;  **return** **true**;  }    }//建立了人的信息,以及初始化关系图 |

### :设计/实现客户端代码main()

**设计**:需要用户实例化FriendshipGraph,并且手动加入person,relation.

**结果**:

|  |
| --- |
| 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)); |

### 设计/实现测试用例

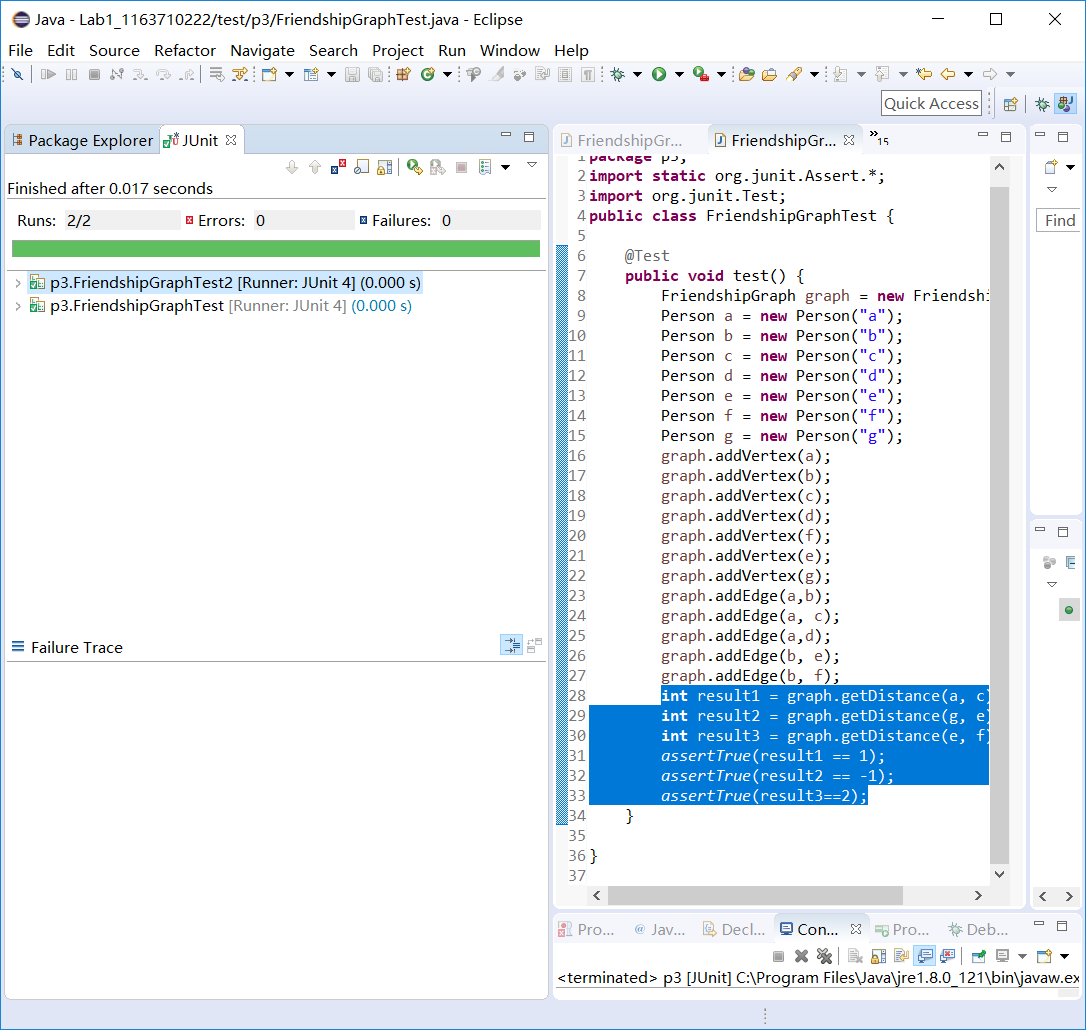
**思路**:与main基本相同,但是需要加入特定的junit判断语句

**过程**:

在main函数基础上加入如下的代码即可

|  |
| --- |
| **int** result1 = graph.getDistance(a, c);  **int** result2 = graph.getDistance(g, e);  **int** result3 = graph.getDistance(e, f);  *assertTrue*(result1 == 1);  *assertTrue*(result2 == -1);  *assertTrue*(result3==2); |

**结果**:



## Tweet Tweet（选作，额外记分）

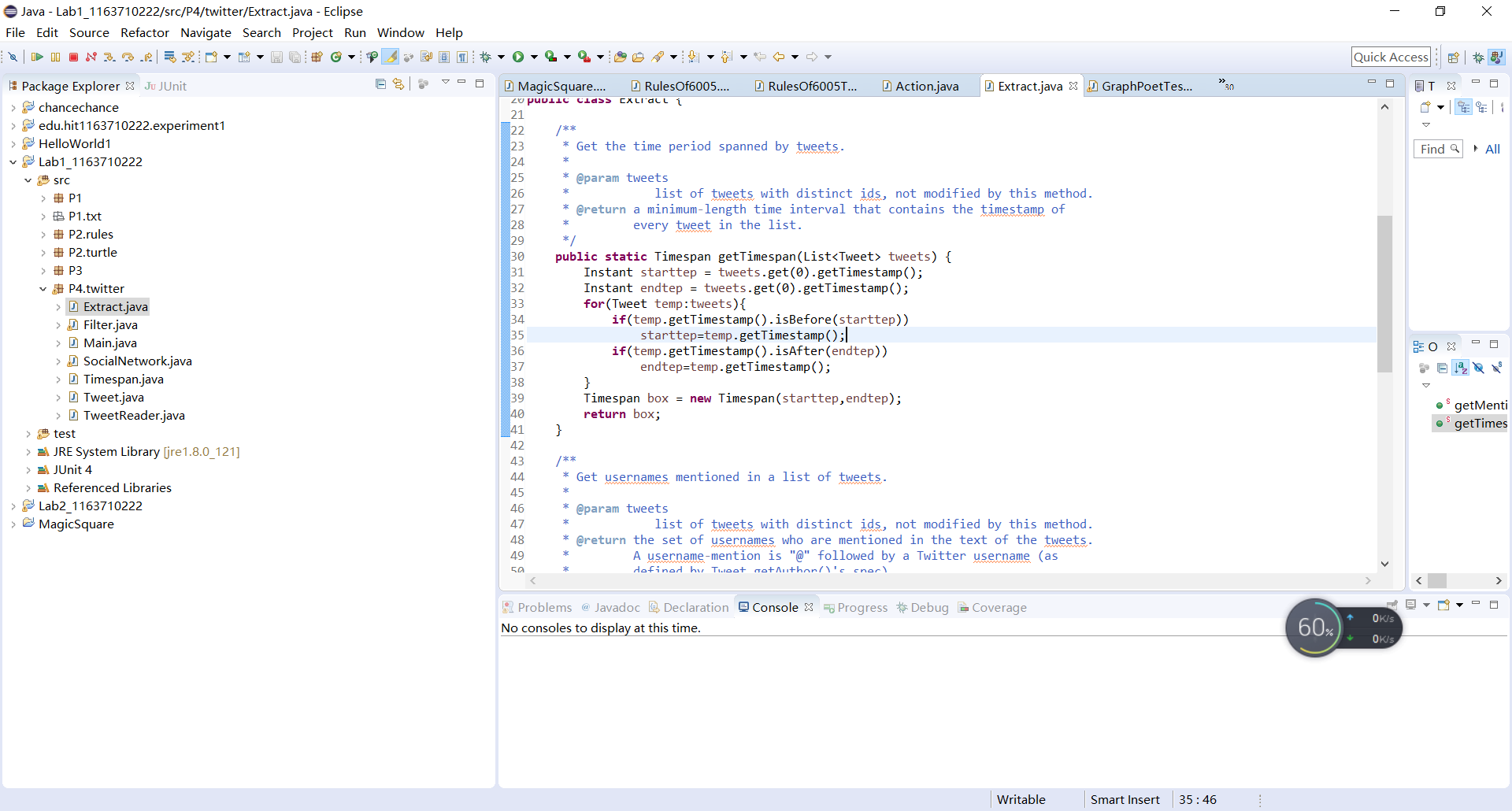
**3.4.1 实现extract**

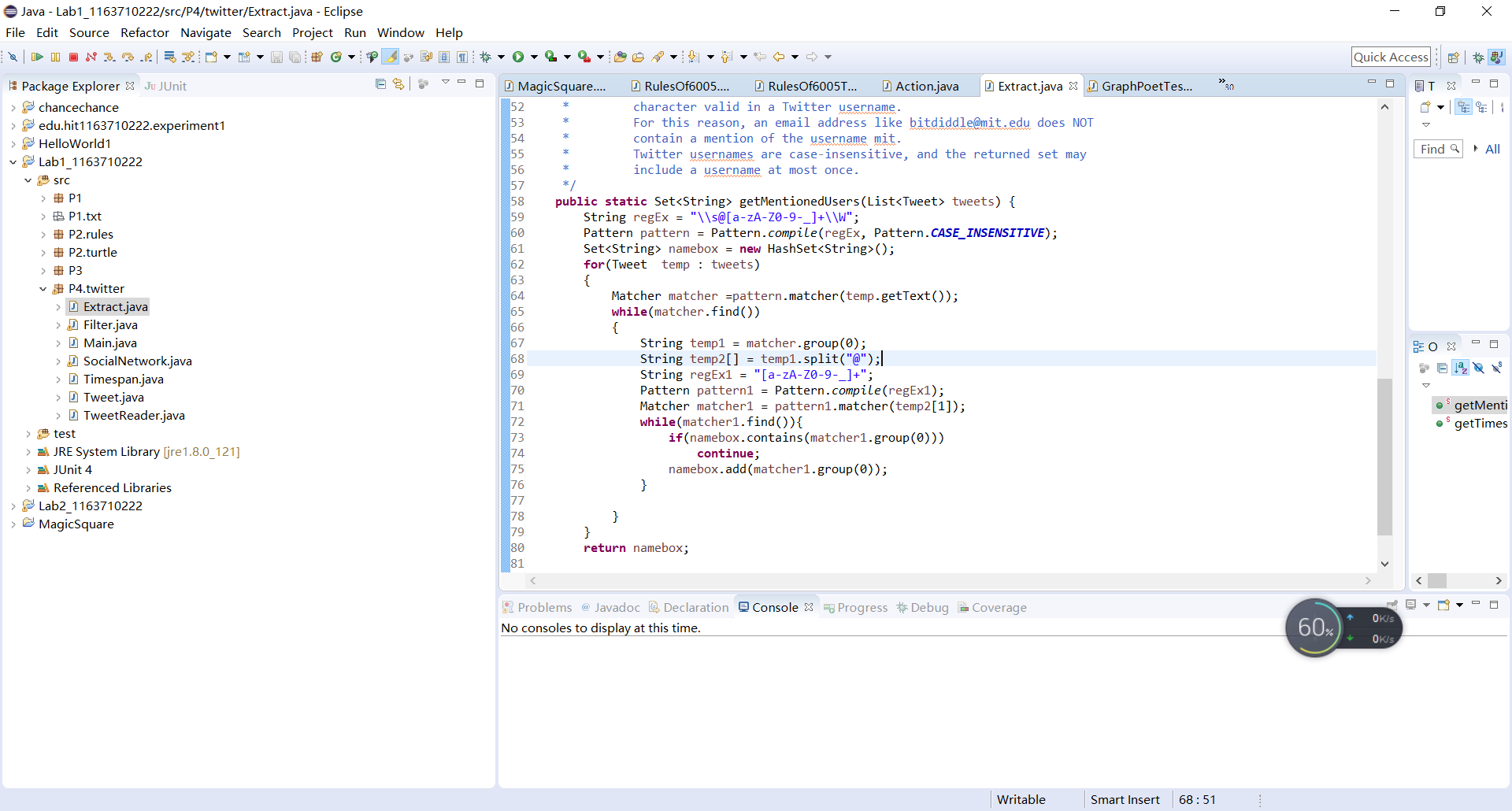
**设计思路:** 由于是从文本中提取信息,所以任务要求针对时间范围以及姓名进行信息提取,实现的主要难点是正则表达式

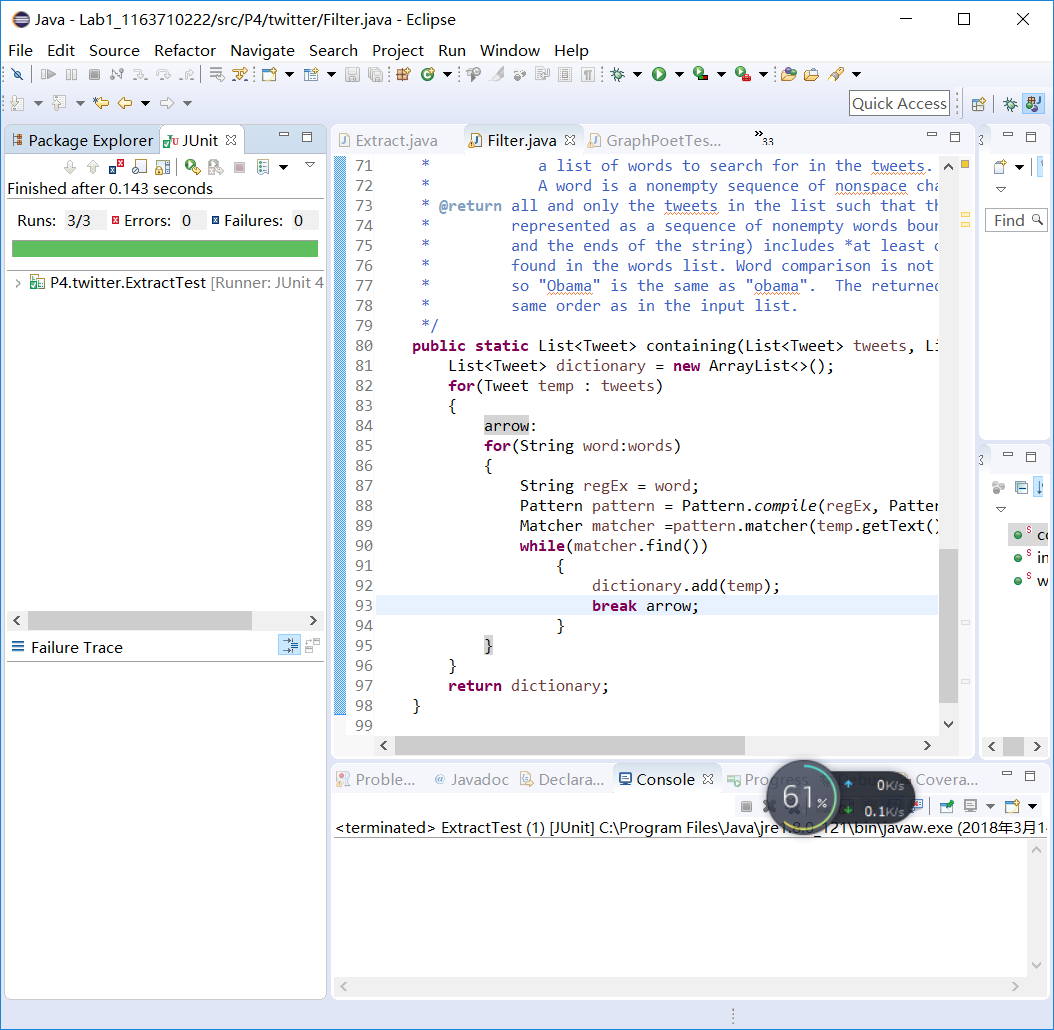
**实现过程:** 1.getTimespan,只需要进行简单的排序,找出最早的开始时间和最晚的结束时间即可得到涵盖所有推文的时间范围,难点主要是Instant类的方法

2.getMentionedUsers**,**需要提出匹配正则表达式的姓名而且还要对提取出来的字符串进行处理,把”@”,以及其他空字符剔除,然后存在指定的数组列表中.

**实现结果:**







**3.4.2 实现filter**

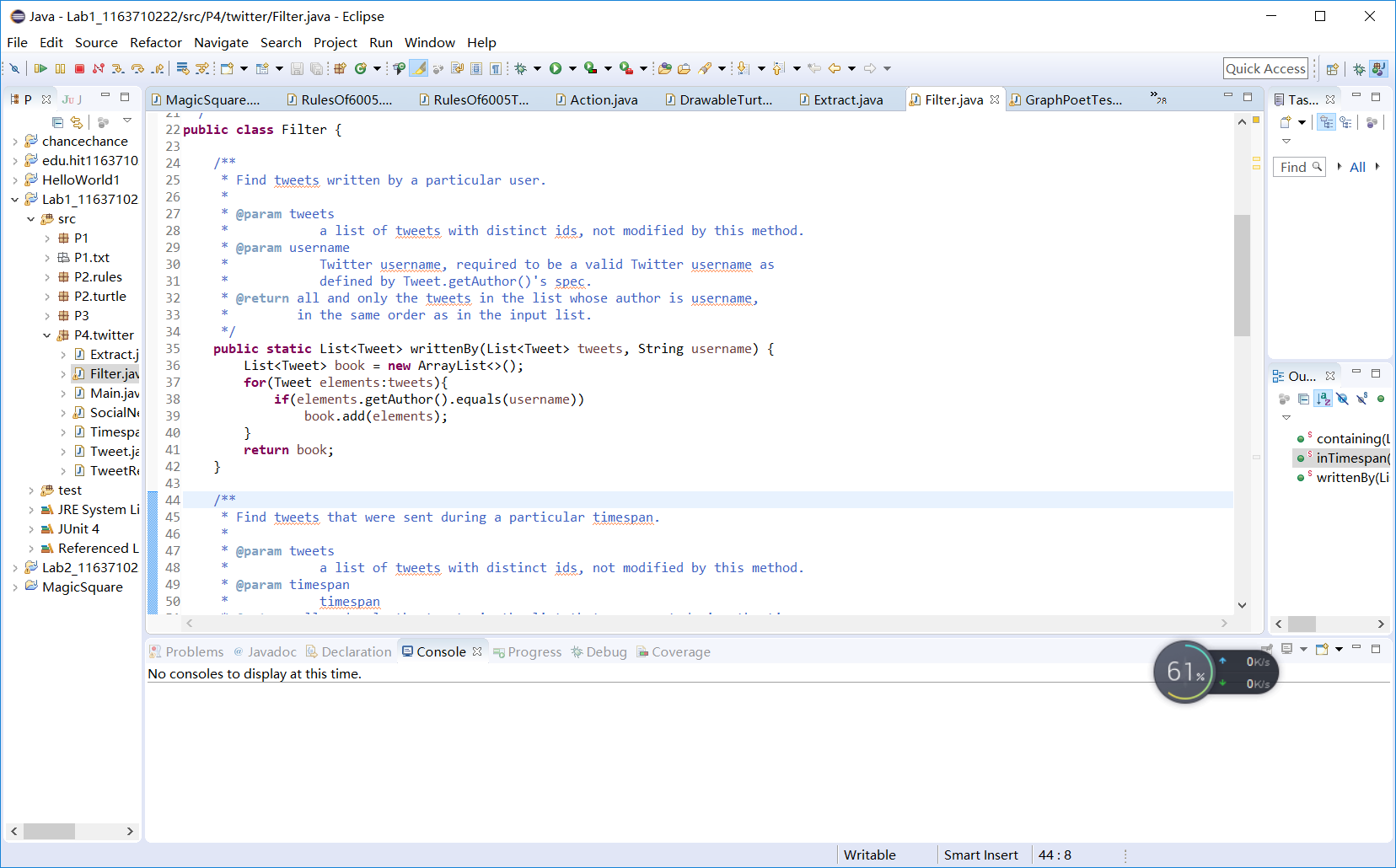
**设计思路:**统计指定作者的tweet,只需要将指定姓名与tweet的作者进行比较即可,统计是否在指定的时间范围内也只需要将时间戳与给定范围的起点与终点进行比较即可.,统计是否有指定的字符,只需要进行pattern与matcher的使用,非常简单.

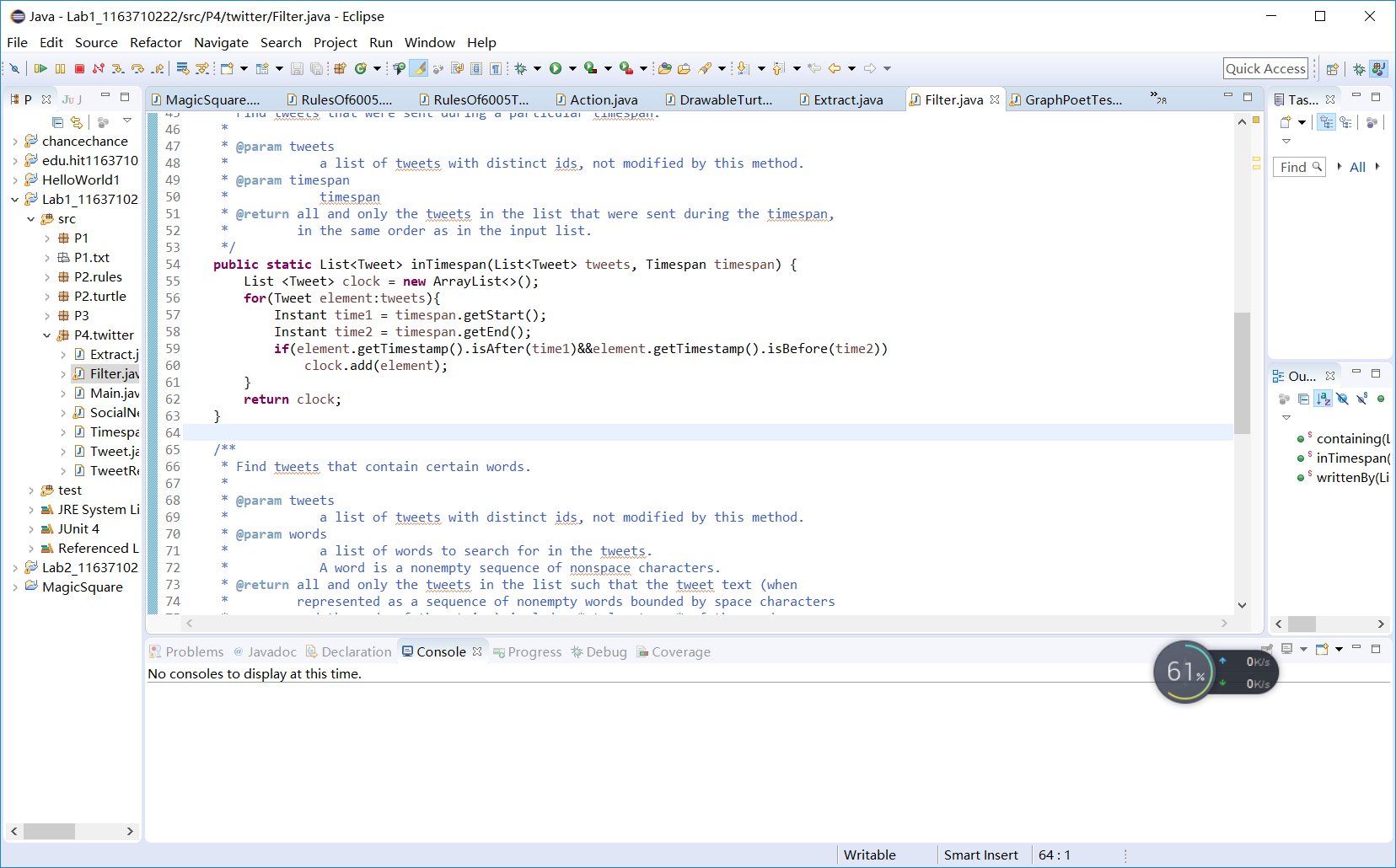
**实现过程:** 1. writtenBy 将指定字符与文章进行匹配,若有,则存入指定的数组列表中

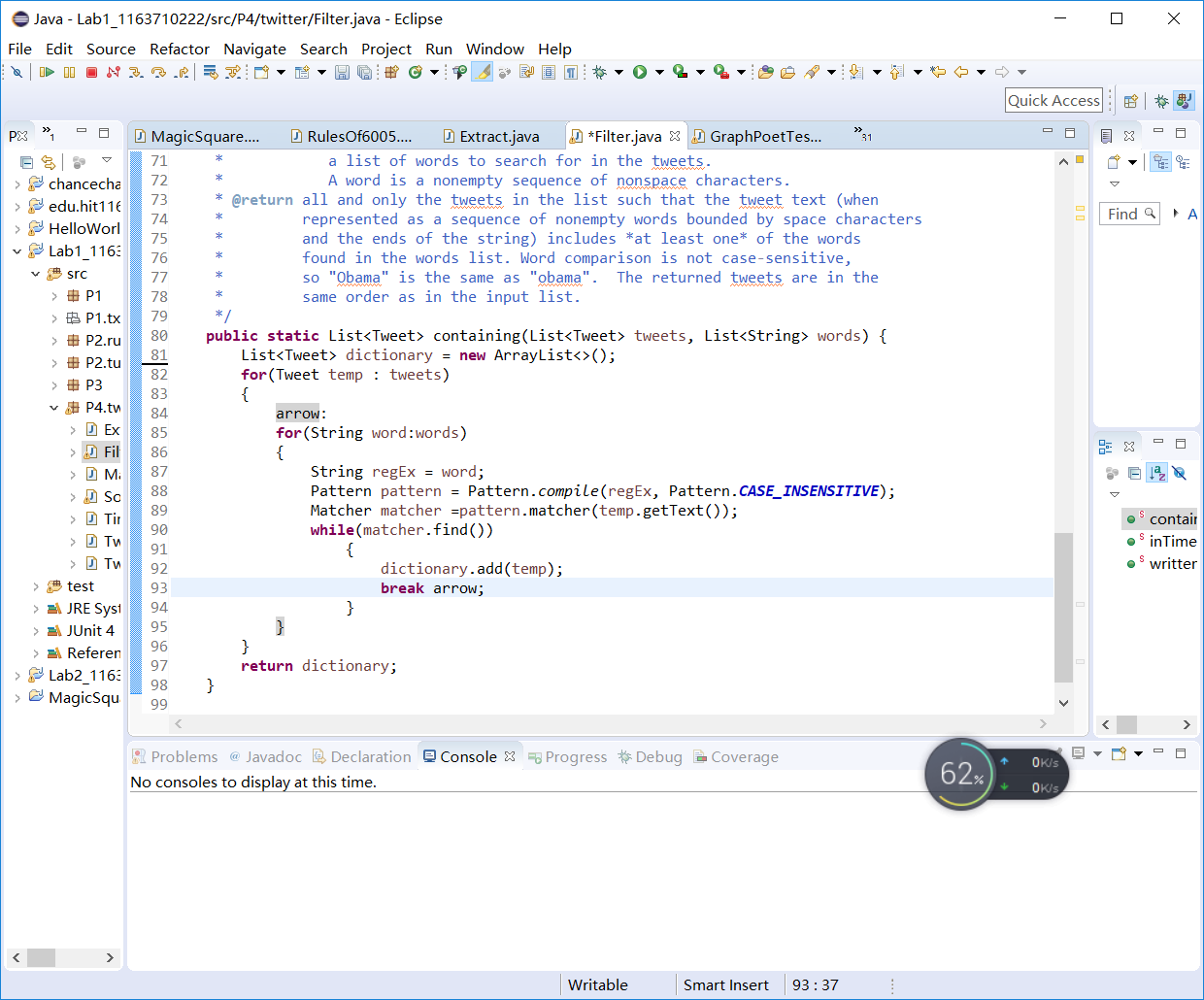
2.inTimespan 如果推文的时间在指定范围之中,那么存入指定的数组列表之中,并且执行结束后进行返回

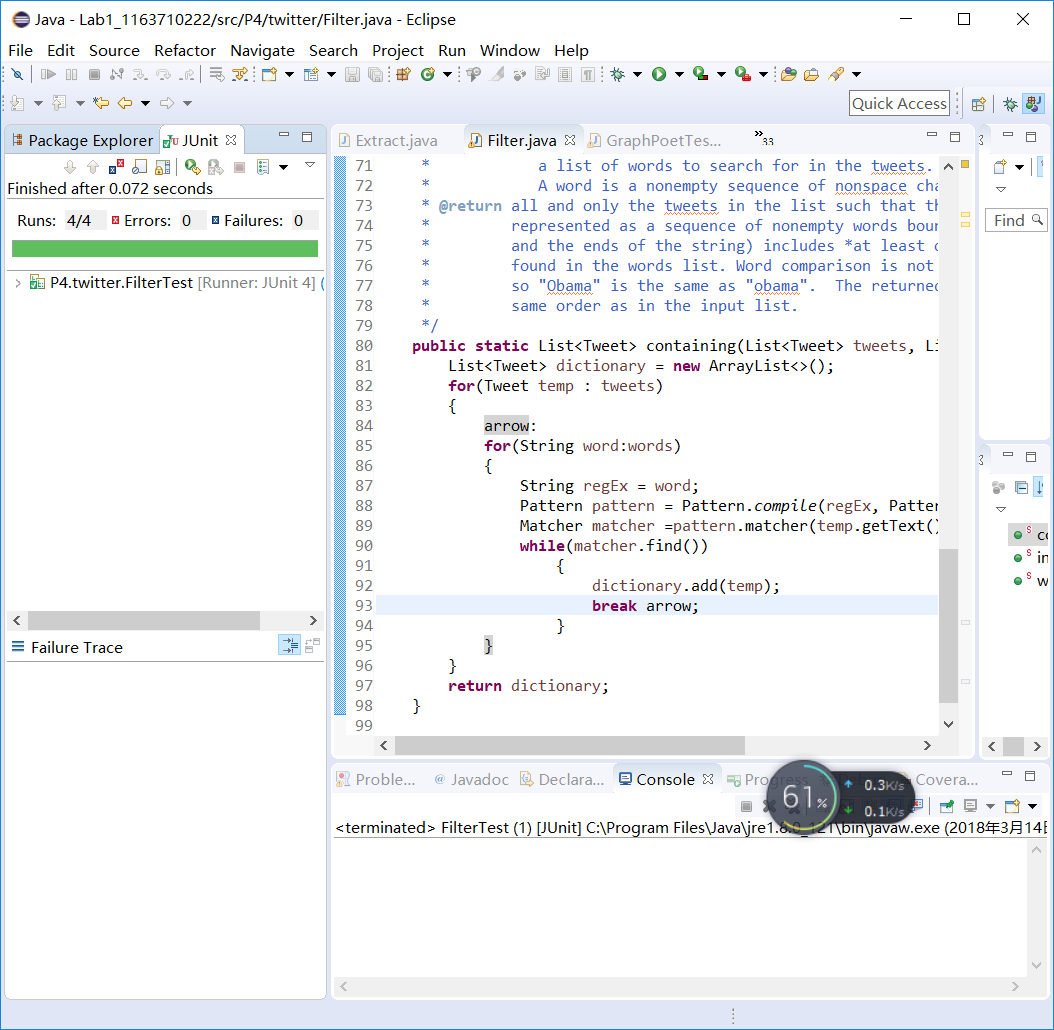
3. containing 如果文章发现指定的字符串,就直接存入数组,并且进行下一篇文章的搜索.

**实现结果:**







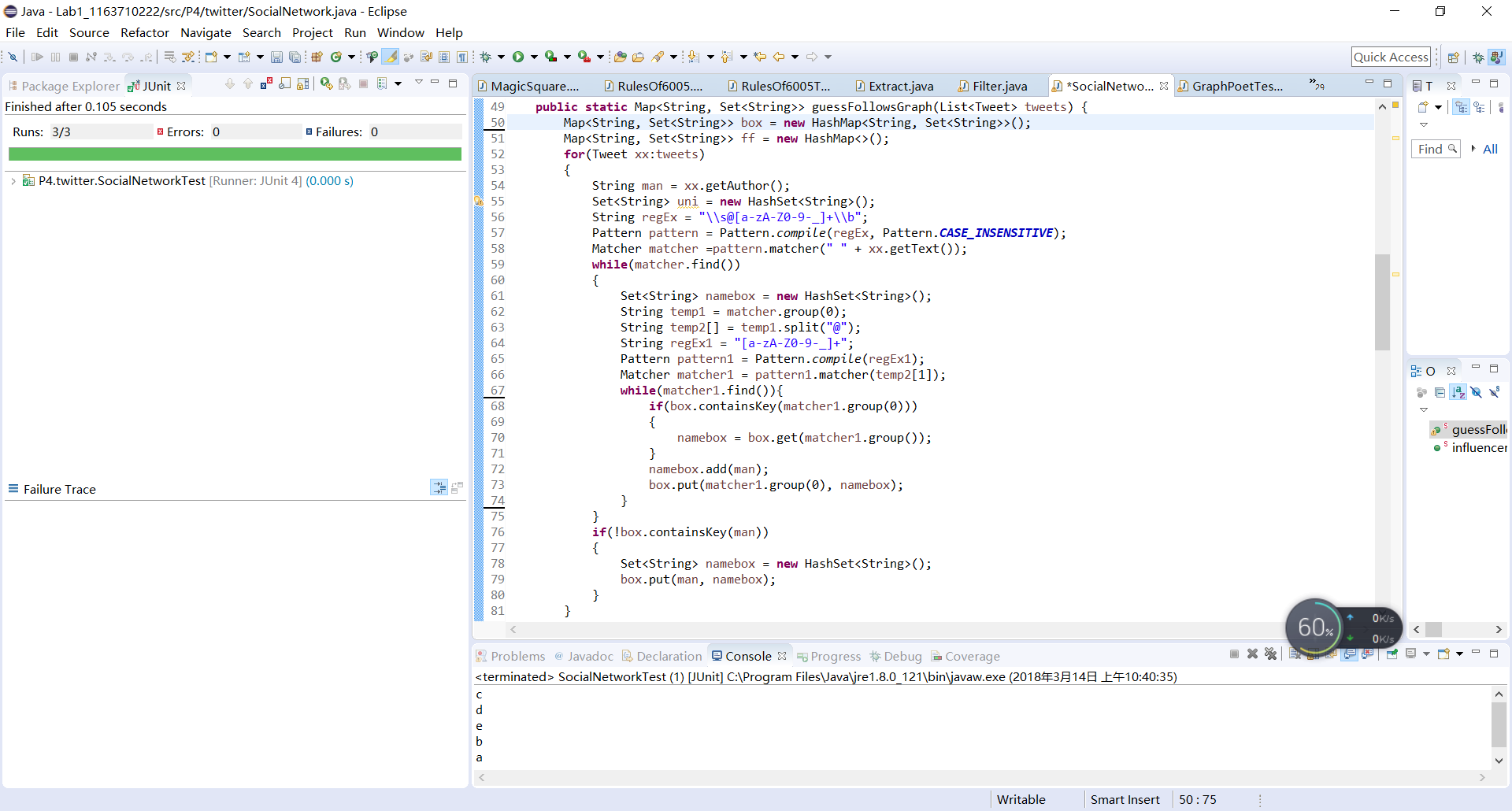


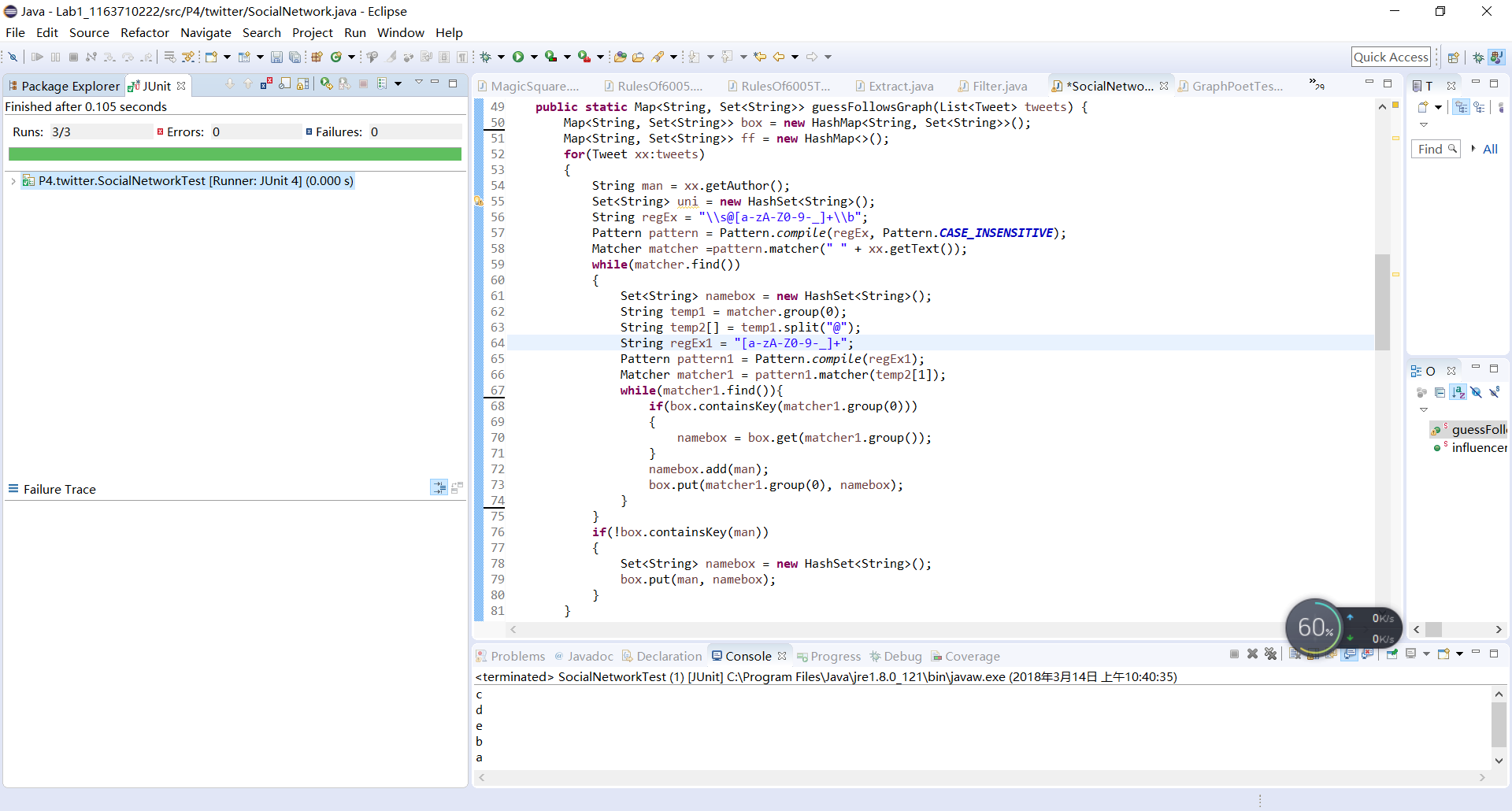
**3.4.3 SocialNetwork实现**

**实现思路:**比之前两个任务难了不少,但是最基本的还是字符串匹配,使用正则表达式以及正确的字符串处理就可以解决问题.

**实现过程:**使用指定的正则表达式[**\\s@[a-zA-Z0-9\_-]+\\b**](file:///\\s@[a-zA-Z0-9_-]+\\b)进行匹配,找到指定的字符串后,把姓名提取,然后将作者姓名加入到@的对象的数组列表中

**实现结果:**



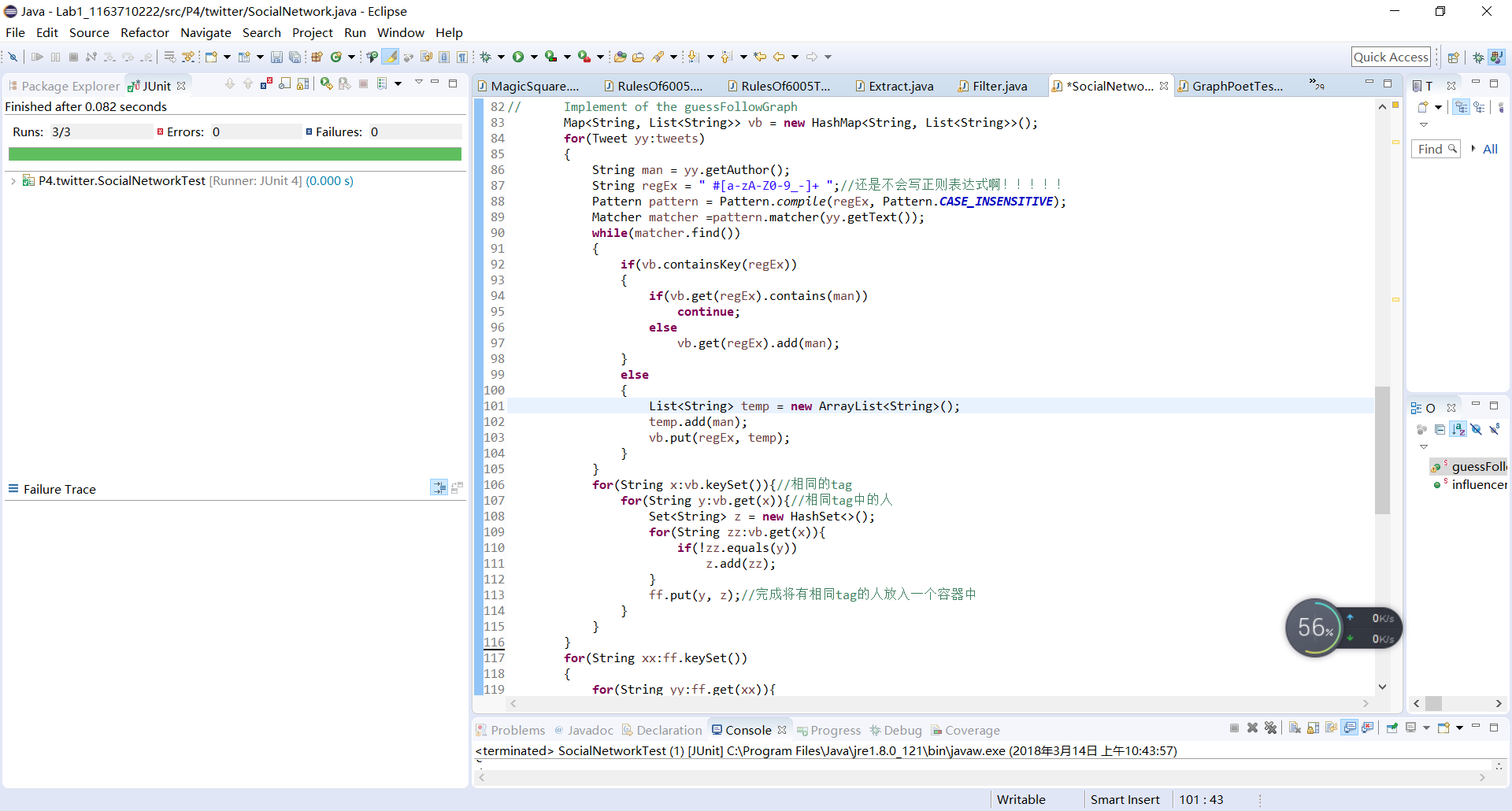


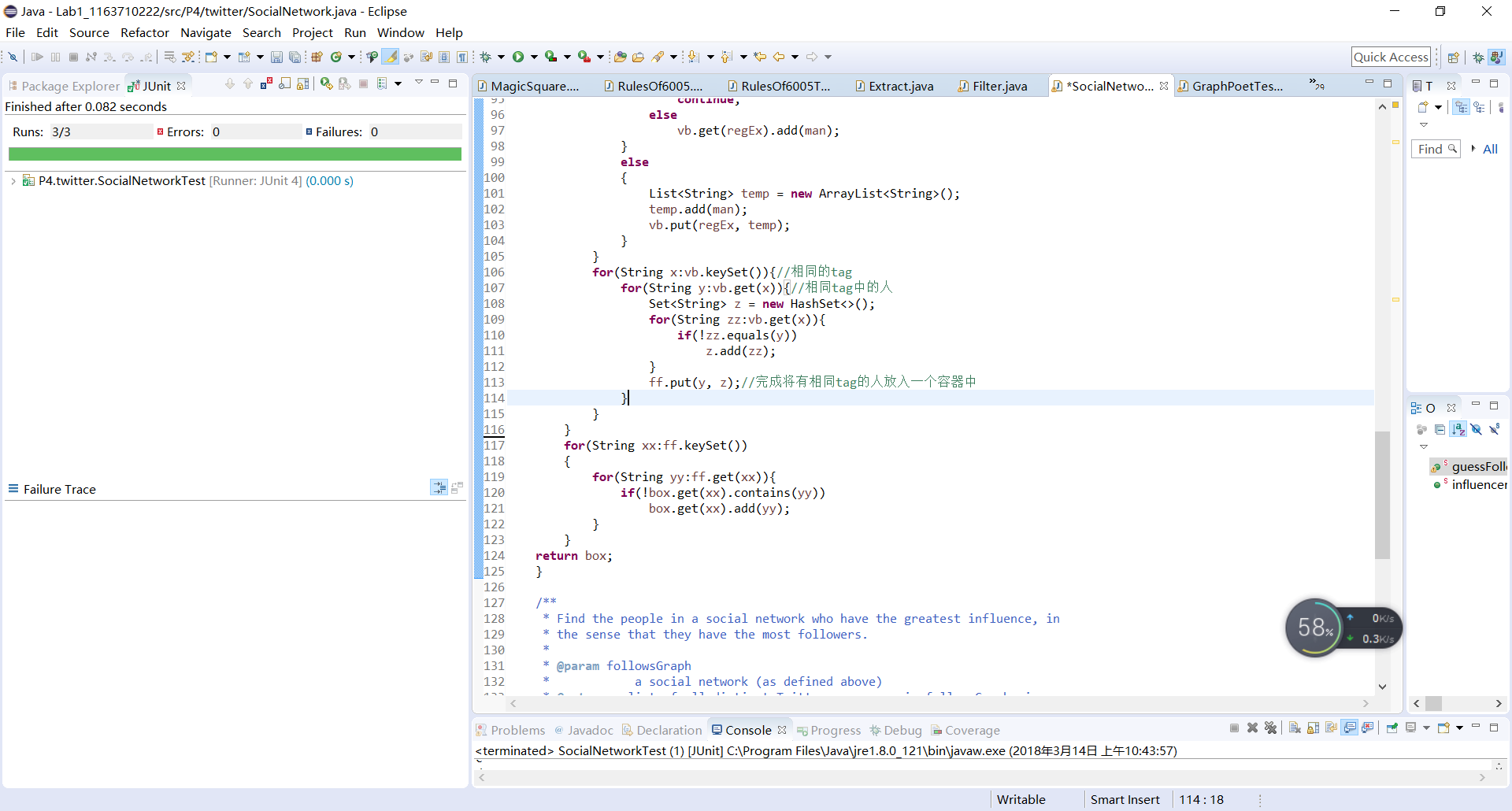
**3.4.4 对socialNetwork的补充**

**1.实现思路:** 在文章中找到有相同hashtag的推文的文章,将作者们加入指定的集合,让他们相互关注

**2.实现过程:**匹配推文中的hashtag,建立数组列表,将所有相关的人加入其中,最后对hashMap进行遍历,将所有拥有相同标签的人进行相互关注.

**3. 实现结果:**





# 实验进度记录

请尽可能详细的记录你的进度情况。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 2018-02-28 | 14:00-15:30 | 编写问题1的isLegalMagicSquare函数并进行测试 | 按计划完成 |
| 2018-03-01 | 18:00-22:30 | 编写问题1,问题2,问题3以及测试用例 | 延期1小时完成 |
| 2018-03-07 | 18:00-21:30 | 完成问题四1,2的编写,但是3出现问题 | 遇到困难，未完成 |
| 2018-03-08 | 19:20-22:30 | 完成问题四3,4的编写,但没有进行测试 | 按计划完成 |
| 2018-03-09 | 19:20-20:50 | 完成测试以及修改 | 延期一小时完成 |

# 实验过程中遇到的困难与解决途径

1. junit的导入,由于最开始是最新版本,使得出现问题,在查找资料后解决

2.编程时忽略set的特性,在遍历时出现错误,通过阅读<<java核心技术>>解决

3.在编写junit测试时不理解语法,通过浏览他人博客,掌握junit的原理以及基本的测试用例语法

# 实验过程中收获的经验、教训、感想

本节除了总结你在实验过程中收获的经验和教训，也可就以下方面谈谈你的感受（非必须）：

1. Java编程语言是否对你的口味？

我个人感觉java很强大

1. 关于Eclipse IDE

非常方便,但是插件太多

1. 关于Git

知识太多,希望能够有简化版教材

1. 关于CMU和MIT的作业

难度略大,但是还是能够基本完成

1. 关于本实验的工作量、难度、deadline

适中,略难,截止日期很合理

1. 关于初接触“软件构造”课程

有点不知所云的感觉,但是看了课件和推荐书目后还是可以较好地接受知识