



哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

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	穆添愉
学号	1160301008
班号	1603010
电子邮件	1417553133@qq.com
手机号码	15636094072

目录

1 实验目标概述	1
2 实验环境配置	1
3 实验过程	1
3.1 Magic Squares	1
3.1.1 isLegalMagicSquare()	1
3.1.2 generateMagicSquare()	2
3.2 Turtle Graphics	3
3.2.1 Problem 1: Clone and import	3
3.2.2 Problem 3: Turtle graphics and drawSquare	4
3.2.3 Problem 5: Drawing polygons	5
3.2.4 Problem 6: Calculating headings	6
3.2.5 Problem 7: Personal art	7
3.2.6 Submitting	7
3.3 Social Network	7
3.3.1 设计/实现 FriendshipGraph 类	8
3.3.2 设计/实现 Person 类	8
3.3.3 设计/实现客户端代码 main()	8
3.3.4 设计/实现测试用例	10
3.4 Tweet Tweet (选作, 额外记分)	12
3.4.1 Extracting data from Tweets	12
3.4.2 Filtering lists from tweets	13
3.4.3 Inferring a social network	14
3.4.4 Get Smarter	14
4 实验进度记录	15
5 实验过程中遇到的困难与解决途径	16
6 实验过程中收获的经验、教训、感想	16

1 实验目标概述

本次试验通过求解四个问题, 训练基本 Java 技能, 能够利用 Java OO 开发基本的功能模块, 能够阅读理解基本的测试程序并完成测试, 初步保证所开发代码正确性。另一方面, 利用 Git 作为代码的配置管理工具, 学会 Git 的基本使用方法。

2 实验环境配置

首先是 Java 环境的配置, 由于我是 Ubuntu 系统, 所以配置起来可能会稍微麻烦一些, 首先将 JDK 安装并解压到/opt 目录下, 然后在/etc/profile 中加入环境变量就好。主要是 Git 的操作由于之前没有接触过所以不太熟悉, 尤其是秘钥 SSH 那里, 用了很长时间才大概弄懂。

我的仓库地址:

<https://github.com/ComputerScienceHIT/Lab1-1160301008>

3 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。

3.1 Magic Squares

幻方是横向、纵向、对角线上的数字相加都一样的正方形矩阵, 所以只需要一些判断条件来判断符合要求的矩阵就可以确定是幻方。

3.1.1 isLegalMagicSquare()

首先先判断是不是矩阵, 也就是判断行列数是否相等, 并判断每一行有没有写满, 这样初步先判断它是不是一个矩阵, 然后再判断是否含有小于等于零的数、

是不是全为整数、是不是用 tab 分割等等。

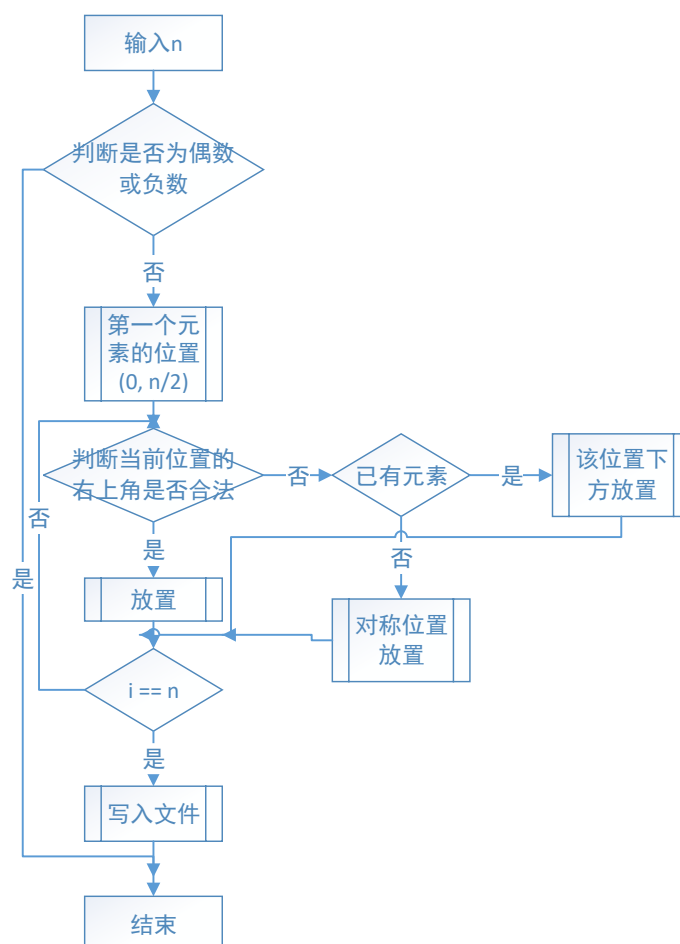
实验过程：从文件中读取，这里耗费了比较长的时间，因为对 Java 文件处理并不熟悉，所以现学现用，运用输入流和 BufferedReader 的知识来进行文件读取，然后再按照上述思路判断即可。

实验结果如下：

```
<terminated> MagicSquare [Java Applicatio  
符合要求, Right  
符合要求, Right  
行列数不等, 不是方阵, False  
存在非正整数, False  
行列数不等, 不是方阵, False  
_ _
```

3.1.2 generateMagicSquare()

流程图如下：



根据实验要求, 要先判断是不是正奇数, 如果不符合要求就特判一下直接返回即可。所以这个实验的难点在于文件的写入, 同样利用 `BufferReader` 来进行文件写入, 这样, `MagicSquare` 就可以完成了。

实验结果如下:

```
False
False
符合要求, Right
```

3.2 Turtle Graphics

和 Python 中的海龟作图类似, 就是对海龟的各种操作来实现作图, 重点是角度的调整和长度的把握。

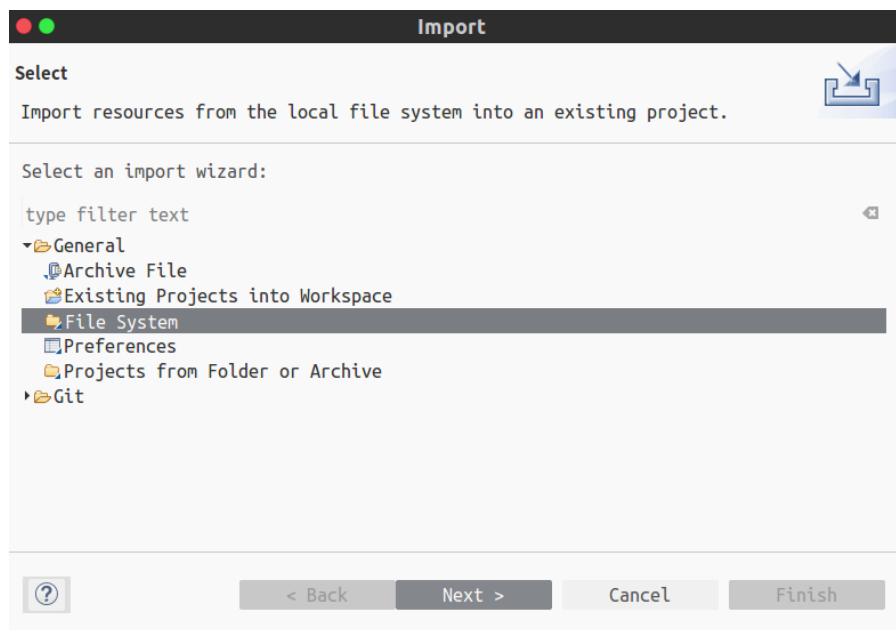
3.2.1 Problem 1: Clone and import

(1) 使用“git clone

https://github.com/rainywang/Spring2018_HITCS_SC_Lab1/tree/master/P2” 指令,

获取 github 上的源码到本地仓库。

(2) 首先, 在项目那里右键-> import -> General -> File System, 最后找到需要用到的文件即可。

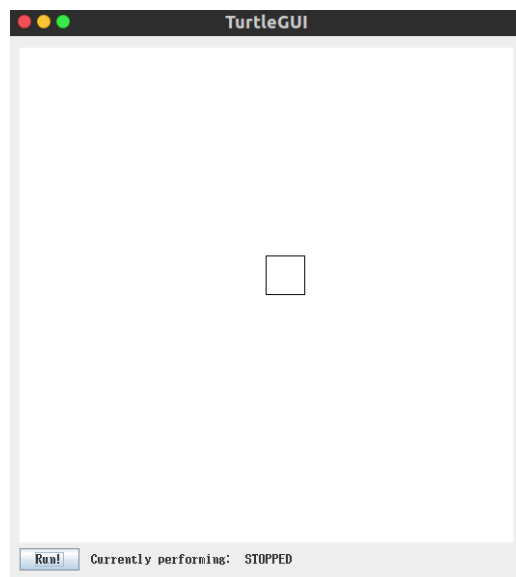


3.2.2 Problem 3: Turtle graphics and drawSquare

采用一个 for 循环，运用 forward () 方法和 turn () 方法，就可以实现画正方形的需求。代码如下：

```
public static void drawSquare(Turtle turtle, int sideLength) {  
    if (sideLength < 0)  
        throw new RuntimeException("implement me!");  
    int i;  
    for (i = 0; i < 4; ++i) {  
        turtle.forward(sideLength);  
        turtle.turn(90.0);  
    }  
}
```

实验结果如下：



3.2.3 Problem 5: Drawing polygons

(1)calculateRegularPolygonAngle()

首先，这个方法要计算正多边形的内角，所以现根据正多边形内角和 $(n - 2) * 180$ 来计算正多边形的内角和，最后除以边数就可以实现这个方法。注意采用地板取整。

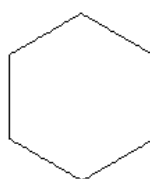
(2)calculatePolygonSidesFromAngle()

这个方法要求根据给出的角度（正多边形内角）来计算正多边形的边数，运用公式 $360 / n$ 即可进行计算。这里还是要注意浮点运算的时候取整问题。

(3)drawRegularPolygon()

这个方法要画出一个正多边形，可以采用上一个函数，只需要计算出每次需要转的角度即可画出一个正多边形。

结果如下：



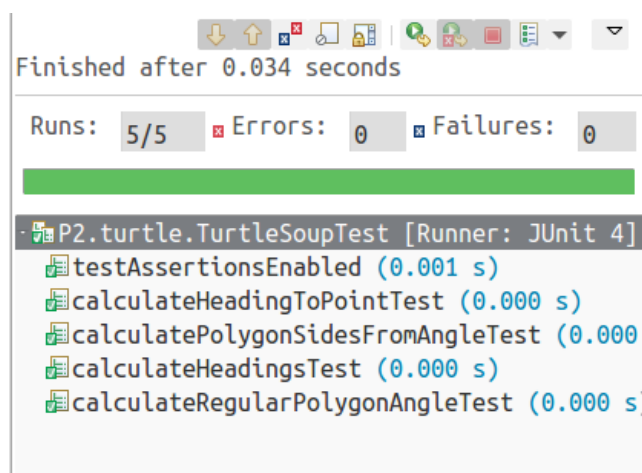
(4)calculateHeadingToPoint()

这个方法需要一点思考, 首先我采用转换坐标系的方法来进行思考这个问题, 首先, 以起点为坐标原点, 重新进行坐标系的构建, 然后根据目标点的相对坐标来进行情况的分类, 可以根据对于 $\text{deltaY} / \text{deltaX}$ 再进行反三角函数, 来求出一个角度, 再结合 `currentHeading` 和一些简单的几何知识, 就可以解出这道题, 但是在写这个方法的时候, 遇到了一些问题, 就是有时候我算的方法和预期会相差 180° 或者 360° , 经过不停的思考。发现自己犯的一个很低级的错误, 就是 `arctan` 求出的角度不是一个解, 而是多个, 所以可以根据 `if-else` 来进行判断, 或者采用题目中提示的 `arc2` 函数, 这样可以解决这个问题。

3.2.4 Problem 6: Calculating headings

(5)这个方法只需要循环调用 `calculateHeadingToPoint()`即可, 获取 `list` 的长度, 然后首先将 `list` 中第一个点作为起始点, 下一个点作为目标点, 传入 `calculateHeadingToPoint()`中, 求出角度, 加入返回的 `list` 中即可, 每次更新起始点和终止点还有初始朝向角度, 即可实现。

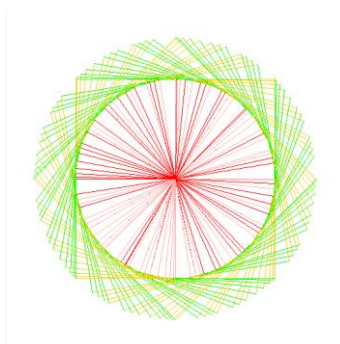
测试结果：



3.2.5 Problem 7: Personal art

这里采用了一个 0 到 360 的循环，并适当调整了画笔的颜色，画出个人艺术图形。

结果如下：



3.2.6 Submitting

如何通过 Git 提交当前版本到 GitHub 上你的 Lab1 仓库。

3.3 Social Network

这个实验首先要有 `Person` 类来存储每一个人的信息，然后在 `Friendship` 类中来实例化 `Person` 对象，并根据题目要求来新建一个图，采用动态二维 `List` 来存储并在此基础上来进行最短路计算，最短路可以采取广度优先搜索来进行求取。

3.3.1 设计/实现 FriendshipGraph 类

有一个 List 来存储所有顶点，运用队列来进行广度优先搜索，即可找到每个顶点到起始点的最短路径，直到搜到终止点，返回更新的终止点的 dist 属性，如果与之相连的所有顶点都搜索过还没有找到目标点，则返回 -1。

3.3.2 设计/实现 Person 类

代码如下：

```
public class Person {  
    protected String name;    //名字  
    protected boolean flag = false; //访问标志  
    protected List<Person> friends = new ArrayList<Person>();  
    protected int dist;  
    public Person(String s) {  
        this.name = s;  
        dist = 0;  
    }  
}
```

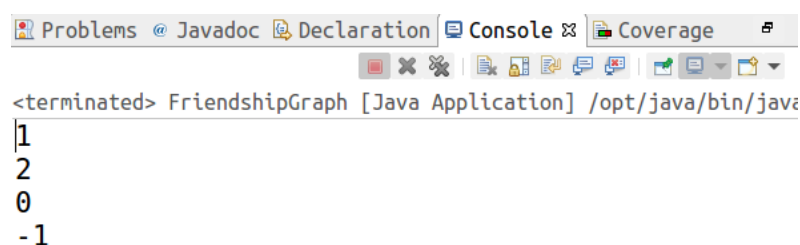
用一个 String 来存储每个人的名字，flag 标记来存储是否被访问过，在进行广度优先搜索的时候可以用到，避免成环，friends 列表来存储与其相连的人，这样可以更容易的进行遍历，也不用担心二维数组空间不够或者浪费空间，dist 来存储距离起始点的最短距离，如果为 -1，则说明不相连。

3.3.3 设计/实现客户端代码 main()

代码如下：

```
public static void main(String[] args) {  
    FriendshipGraph graph = new FriendshipGraph();  
    Person rachel = new Person("Rachel");  
    Person ross = new Person("Ross");  
    //Person ross = new Person("Rachel");  
    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));  
}
```

运行结果如下：

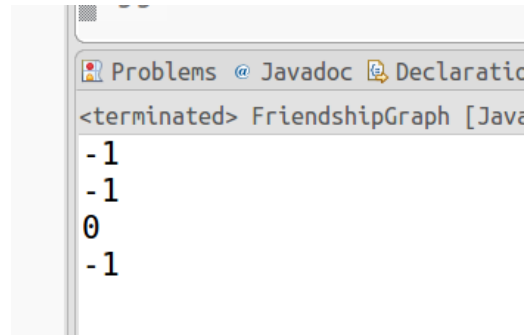


```
<terminated> FriendshipGraph [Java Application] /opt/java/bin/java  
1  
2  
0  
-1
```

将“Ross”替换为“Rachel”,结果如下

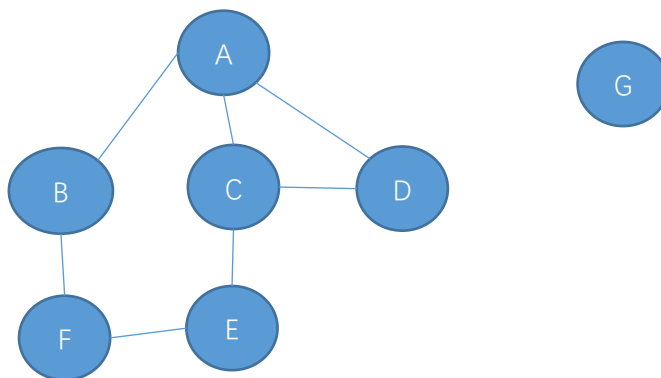
```
Wrong input, Rachel has been inputed!
```

将第十行代码注释掉，结果如下：



3.3.4 设计/实现测试用例

初始化一些人，并且做出一个简单的图，其中有孤立于其他的点，来检测不连通的情况。图如下



代码如下：

```
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");  
Person f = new Person("F");  
Person g = new Person("G");
```

```
graph.addVertex(a);  
graph.addVertex(b);  
graph.addVertex(c);  
graph.addVertex(d);  
graph.addVertex(e);  
graph.addVertex(f);  
graph.addVertex(g);
```

```
graph.addEdge(a, b);  
graph.addEdge(b, a);  
graph.addEdge(a, c);  
graph.addEdge(c, a);  
graph.addEdge(a, d);  
graph.addEdge(d, a);  
graph.addEdge(b, f);  
graph.addEdge(f, b);  
graph.addEdge(c, e);  
graph.addEdge(e, c);  
graph.addEdge(f, e);  
graph.addEdge(e, f);  
graph.addEdge(c, d);  
graph.addEdge(d, c);|
```

addVertexText()

```
assertEquals(a, graph.Vertex.get(0));  
assertEquals(b, graph.Vertex.get(1));  
assertEquals(c, graph.Vertex.get(2));  
assertEquals(d, graph.Vertex.get(3));  
assertEquals(e, graph.Vertex.get(4));  
assertEquals(f, graph.Vertex.get(5));  
assertEquals(g, graph.Vertex.get(6));
```

addEdgeText()

```
assertEquals(0, a.friends.indexOf(b));  
assertEquals(1, a.friends.indexOf(c));
```

getDistanceText()

```
assertEquals(1, graph.getDistance(a, b));  
assertEquals(2, graph.getDistance(a, f));  
assertEquals(0, graph.getDistance(a, a));  
assertEquals(-1, graph.getDistance(a, g));  
assertEquals(1, graph.getDistance(e, f));
```

结果如下：



3.4 Tweet Tweet（选作，额外记分）

这个题目主要是对文本处理的考察，比如涉及到的 `getUsersMentioned()`，（寻找一条推文中@到的用户）可以采用正则表达式来进行实现或者直接暴力匹配就可以完成，Problem 4 提取#标签也涉及文本处理，但是在 Problem 4 中我选择了另一个建议，也就是 Triadic closure，这其中涉及对 Set 的修改及遍历，但是在实验过程中，发现 Set 并不能一边修改一边遍历，因为 for each 遍历是采用迭代器的，在遍历过程中修改是会破坏迭代器的，所以这个难题我想了一下午，甚至想要放弃，直接放弃掉 Problem 4 去进行第二周的实验随堂检查，还好最后时间不够，没有检查上，这坚定了我要把 Problem 4 完成的信心，终于，在晚上把它写好了，我采用了一个新的 Map 来存储需要额外添加关系的人，然后在遍历好 Map 之后，去遍历这个辅助 Map 来修改之前的 Map，终于不再报错，这个错误不仅让我知道了 Set 不能一边遍历一边修改，还让我对迭代器有了一个初步的认识，我相信这对我今后的实验和工作是很有帮助的。同时这也锻炼了我查资料解决新问题的能力。

3.4.1 Extracting data from Tweets

这个问题的难点就在于时间戳之间不能直接进行比较，要转换成秒数来进行比较，这个秒数是相对同一时间之间的时间差，这样就可以获取最早时间的推文和最晚时间的推文，达到题目要求。

下面是测试文件：

```
private static final Instant d1 = Instant.parse("2016-02-17T10:00:00Z");
private static final Instant d2 = Instant.parse("2016-02-17T11:00:00Z");
private static final Instant d3 = Instant.parse("2016-02-16T11:00:00Z");
private static final Instant d4 = Instant.parse("2016-02-17T11:00:01Z");

private static final Tweet tweet1 = new Tweet(1, "alyssa", "@is, it reasonable to talk ,@ggabout rivest so much?@tianyu,@what,@
private static final Tweet tweet2 = new Tweet(2, "bbitdiddle", "rivest talk in 30 minutes #hype", d2);
private static final Tweet tweet3 = new Tweet(3, "tianyu", "I like to study the Software Construction!", d3);
private static final Tweet tweet4 = new Tweet(4, "lalala", "why don't you love me?", d4);
```

测试函数：

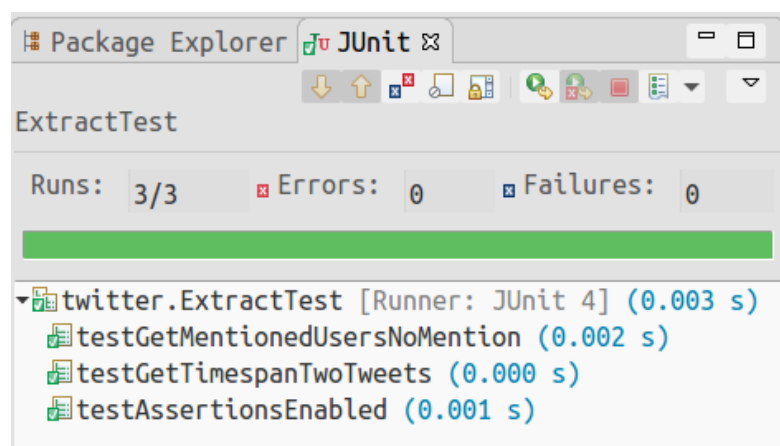
```
public void testGetTimespanTwoTweets() {
    Timespan timespan = Extract.getTimespan(Arrays.asList(tweet1, tweet2, tweet3, tweet4));

    assertEquals(d2, timespan.getStart());
    assertEquals(d4, timespan.getEnd());
}

@Test
public void testGetMentionedUsersNoMention() {
    Set<String> mentionedUsers = Extract.getMentionedUsers(Arrays.asList(tweet1));

    //assertTrue("expected empty set", mentionedUsers.isEmpty());
    assertFalse(mentionedUsers.contains("qq"));
    assertEquals(true, mentionedUsers.contains("is"));
    assertEquals(true, mentionedUsers.contains("tianyu"));
}
```

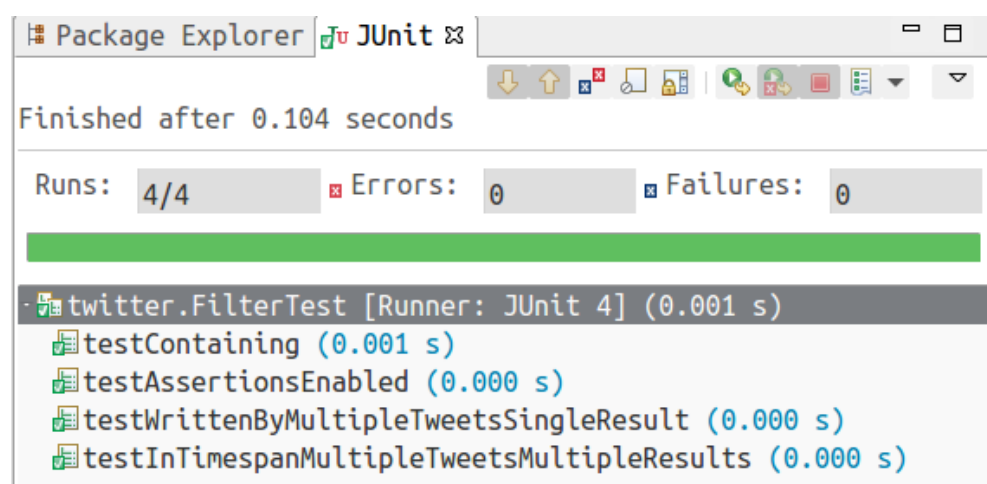
测试结果：



3.4.2 Filtering lists from tweets

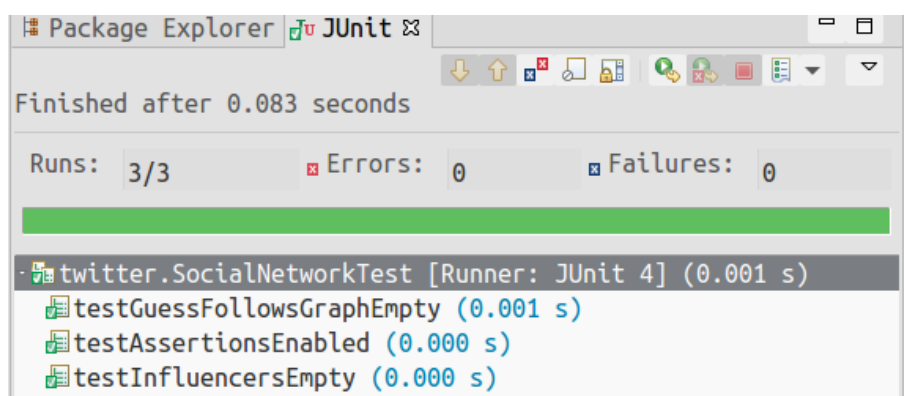
这个方法还是比较容易写的，只需要按要求返回作者、返回在规定时间内 tweets、返回包含特殊单词的 tweets 就可以实现，难度并不是很大。

测试结果：



3.4.3 Inferring a social network

这个方法中最难的一部分就是提取被提及的用户名，一开始我打算用正则表达式来进行提取，但是看了一晚上还是不太理解正则表达式，也不太会用，所以我放弃了这种方法，采取最暴力的方法，直接暴力匹配，提取 @ 后面的字符串来作为用户名，并存入一个 list 返回，这样我就得到了被提及用户的列表，然后根据列表去建立关系图，后面的步骤就不是很麻烦了，然后统计每个人的 followers，按照每个人的 followers 数量来进行排序就完成了。



3.4.4 Get Smarter

这里我选择了 Triadic closure 方法，也就是如果两个人都互相和第三个人互相@，那么他们两个人就也应该互相关注。这里遇到最大的难题就是 Set 的遍历问题，只能采取迭代器来进行遍历，就不能在一边遍历一边修改，所以采取一个辅助 list 来进行在 Problem 3 基础上返回的 Map 进行修改，最终实现要求。

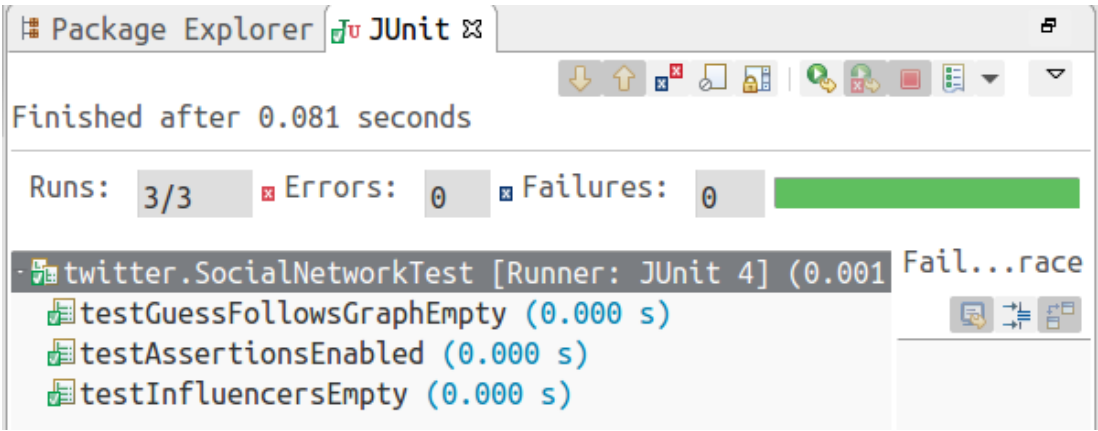
测试文件如下：


```
Instant d1 = Instant.parse("2016-02-17T10:00:00Z");
Instant d2 = Instant.parse("2016-02-17T11:00:00Z");
Instant d3 = Instant.parse("2016-02-16T11:00:00Z");
Instant d4 = Instant.parse("2016-02-17T11:00:01Z");
Tweet tweet1 = new Tweet(1, "alyssa", "is it reasonable to talk about rivest so much?@tianyu.@bbk", d1);
Tweet tweet2 = new Tweet(2, "bbitdiddle", "rivest talk in 30 minutes #hype", d2);
Tweet tweet3 = new Tweet(3, "tianyu", "I like to study the Software Construction!@alyssa", d3);
Tweet tweet4 = new Tweet(4, "tianyu", "why don't you love me?@bbk, @bbitdiddle", d4);
@Test(expected=AssertionError.class)
public void testAssertionsEnabled() {
    assert false; // make sure assertions are enabled with VM argument: -ea
}

@Test
public void testGuessFollowsGraphEmpty() {
    Map<String, Set<String>> followsGraph = SocialNetwork.guessFollowsGraph(Arrays.asList(tweet1, tweet2, tweet3, tweet4));

    assertFalse("expected empty graph", followsGraph.isEmpty());
    assertTrue(followsGraph.get("tianyu").contains("alyssa"));
    assertTrue(followsGraph.get("alyssa").contains("tianyu"));
    assertFalse(followsGraph.get("alyssa").contains("bbitdiddle"));
}
```

测试结果如下：



4 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
2018-02-26	15:45-17:30	编写问题 1 的 isLegalMagicSquare 函数并进行测试	按计划完成
2018-02-26	18:00-21:30	完成问题 1 剩余部分	按计划完成
2018-02-27	14:00-22:00	完成问题 2	延期 1 小时，因为一些细节上的小问题造成了延期、耽误了时间
2018-02-28	19:00-23:00	完成问题 3	按计划完成
2018-03-01	15:00-22:00	完成选做的 Problem 1 和 Problem 2	按计划完成
2018-03-02	断断续续	断断续续的进行对于 Problem 3 的思考	按计划完成

2018-03-03	14:00-21:00	进行正则表达式的学习, 没学懂, 放弃, 换方法实现 Problem 3	按计划完成
2018-03-05	14:00-18:00	实现 Problem 4	没有完成
2018-03-05	23:00-24:30	实现 Problem 4	按计划完成

5 实验过程中遇到的困难与解决途径

实验中首先遇到的第一个难题就是英语的问题, 英语阅读能力并不强, 所以只能通过翻译软件和问同学还有自己多阅读才能很好的理解问题要求, 这方面花费了很多时间。

其次, 就是在 `turtle` 的实验中遇到的 `arctan` 函数, 涉及高中知识, 没有怎么认真思考, 就造成了没有按计划完成的后果, 延期了大概一个小时。

最后是 `tweet` 实验中涉及到字符串匹配的方法, 花了大概三个小时去看正则表达式, 可是没有太看懂, 于是就放弃了正则表达式, 采用了暴力匹配的方法。应该课后把正则表达式看懂的。

6 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?
- (2) 关于 Eclipse IDE
- (3) 关于 Git
- (4) 关于 CMU 和 MIT 的作业
- (5) 关于本实验的工作量、难度、deadline
- (6) 关于初接触“软件构造”课程

建议老师可以适当的添加一些中文翻译, 尤其是关键函数的思路, 就拿这次试验为例, 直到昨天才发现有两个函数是对英文的理解上出了偏差, 导致推翻重来, 还好这两个方法对后续及整体的影响不是很大, 否则真的会很要命! 希望老师可以采纳。

其次, 关于 IDE, 虽然 Eclipse 是开源免费软件, 但是希望老师可以让大家用 IDEA IntelliJ, 首先, 这个 IDE 是可以通过学生认证来进行免费使用的, 其次, 在一些细节方面比如说自动补全等做的更加周到, 也更加易于上手, Eclipse 也不是没有自动补全, 只是要自己设置, 并且某种情况下及其难用, 不如 IDEA, 希望老师可以考虑。