

**2021年春季学期  
计算学部《软件构造》课程**

**Lab 1实验报告**

|  |  |
| --- | --- |
| 姓名 | 陈一帆 |
| 学号 | 1191000606 |
| 班号 | 1903003 |
| 电子邮件 | 1191000606@stu.hit.edu.cn |
| 手机号码 | 13972934654 |

**目录**

[1 实验目标概述 1](#_Toc29325521)

[2 实验环境配置 1](#_Toc29325522)

[3 实验过程 1](#_Toc29325523)

[3.1 Magic Squares 1](#_Toc29325524)

[3.1.1 isLegalMagicSquare() 2](#_Toc29325525)

[3.1.2 generateMagicSquare() 4](#_Toc29325526)

[3.2 Turtle Graphics 5](#_Toc29325527)

[3.2.1 Problem 1: Clone and import 6](#_Toc29325528)

[3.2.2 Problem 3: Turtle graphics and drawSquare 6](#_Toc29325529)

[3.2.3 Problem 5: Drawing polygons 7](#_Toc29325530)

[3.2.4 Problem 6: Calculating Bearings 7](#_Toc29325531)

[3.2.5 Problem 7: Convex Hulls 7](#_Toc29325532)

[3.2.6 Problem 8: Personal art 8](#_Toc29325533)

[3.2.7 Submitting 8](#_Toc29325534)

[3.3 Social Network 9](#_Toc29325535)

[3.3.1 设计/实现FriendshipGraph类 9](#_Toc29325536)

[3.3.2 设计/实现Person类 10](#_Toc29325537)

[3.3.3 设计/实现客户端代码main() 15](#_Toc29325538)

[3.3.4 设计/实现测试用例 15](#_Toc29325539)

[4 实验进度记录 16](#_Toc29325540)

[5 实验过程中遇到的困难与解决途径 17](#_Toc29325541)

[6 实验过程中收获的经验、教训、感想 18](#_Toc29325542)

[6.1 实验过程中收获的经验和教训 18](#_Toc29325543)

[6.2 针对以下方面的感受 18](#_Toc29325544)

# 实验目标概述

学会基本的Java OO编程

学会IDE的基本使用方法

学会书写JUnit测试

学会使用Git管理代码

# 实验环境配置

许多配置工作在寒假完成，故遇到的问题，困难并不完整。

开发为了兼容实验，采用IDEA提供的JDK8，具体为corretto-1.8。

首先克隆github上代码到本地。原计划直接将寒假写好的代码上传到指定私有仓库，结果因为网络原因无法登陆github账号，导致上传失败。多次尝试后使用Token进行认证，先将代码克隆到本地，链接上指定仓库，然后再上传代码。

测试采用JUnit-4.13.1.jar。

在这里给出你的GitHub Lab1仓库的URL地址。

https://github.com/ComputerScienceHIT/HIT-Lab1-1191000606

# 实验过程

## Magic Squares

要求构造MagicSquare.java来检验幻方，即n\*n大小的正整数矩阵，是否每一行，每一列，对角线之和相等。主要考察：文件读写，exceptions处理，类型转换。

### isLegalMagicSquare()

思路：读取数据，逐行，列，对角线求和，判断是否相等，满足正确性。同时考虑各种exceptions，保证代码的健壮性。

设计过程与步骤：

首先根据输入参数filename，打开文件（注意加上./src/P1/txt/）。注意处理FileNotFoundException，IOException等各种exceptions。如图1。

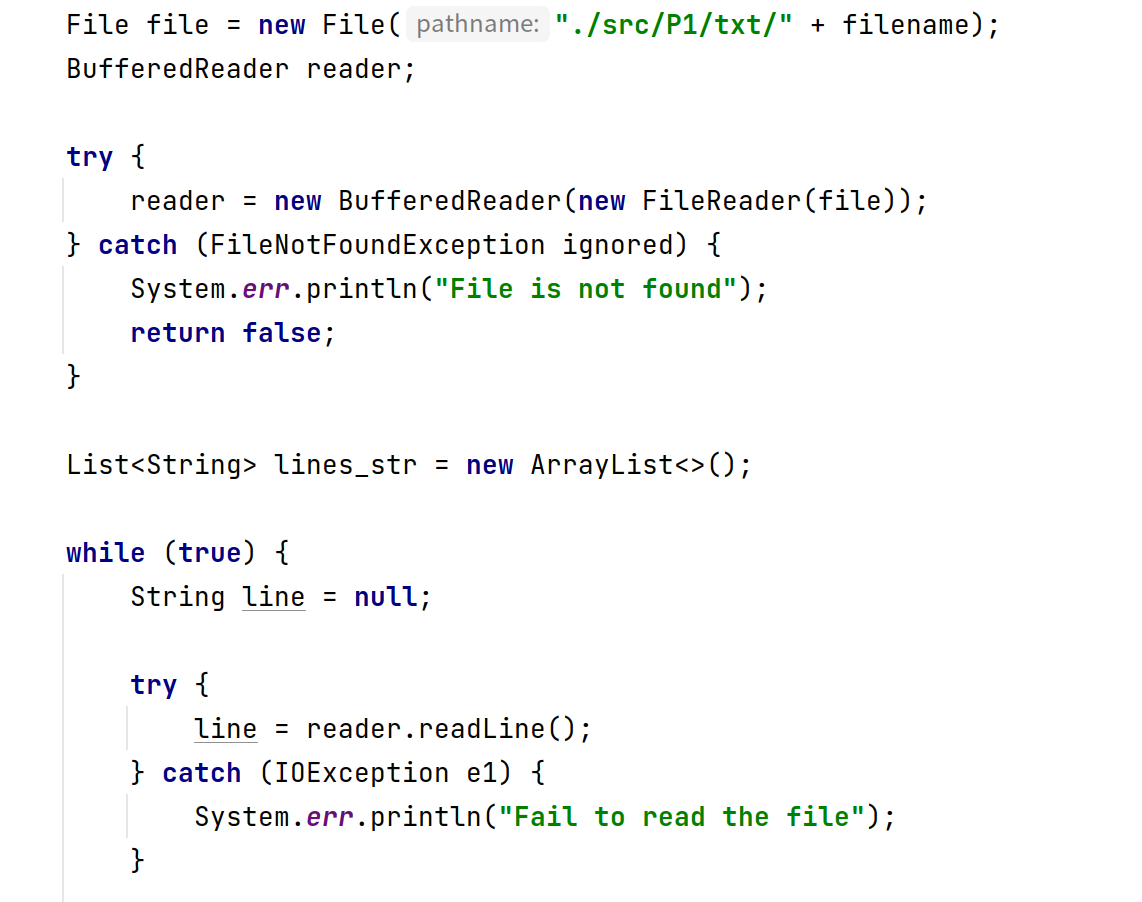


图1 File I/O

然后读取文件每一行，以String的形式，储存在一个ArrayList变量lines\_str中。

每一行String以“\t”作为分割符，然后对分割出的String进行判断，首先看是否能化为整数，能则判断正负，不能则判断是否可化为浮点数。能说明该数为浮点数，不能则说明出现了其他分割符。根据该String的情况，选择继续，或则打印错误信息后退出。如图2。

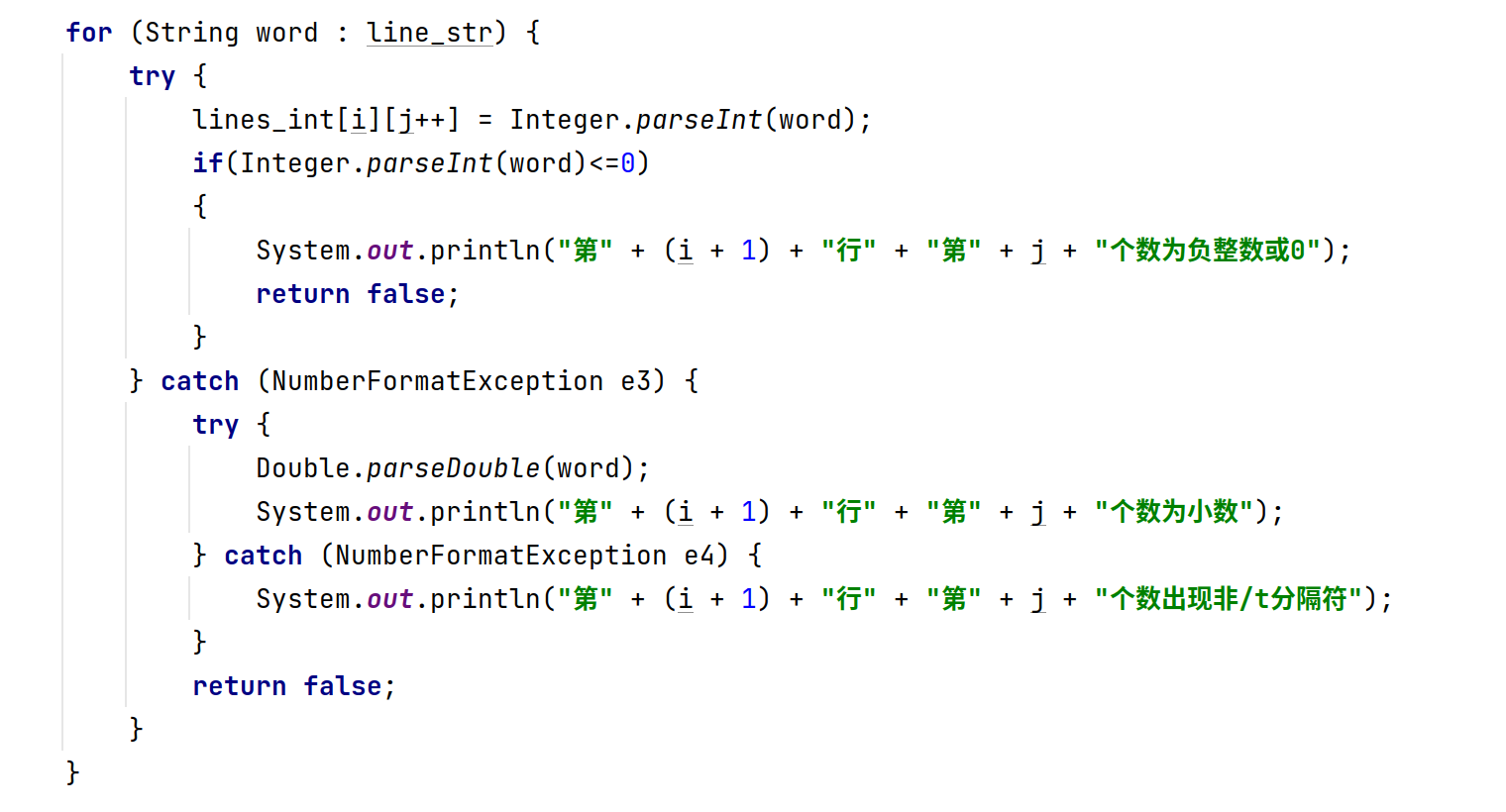


图2 Exceptions处理

如果程序顺利将所有String转为正整数，储存在lines\_int中，判断行数，列数是否相等后进入下一步。

采用循环，计算每一行，每一列，对角线的和。有不同则报错，全部相等则返回true。

结果：如图3所示。

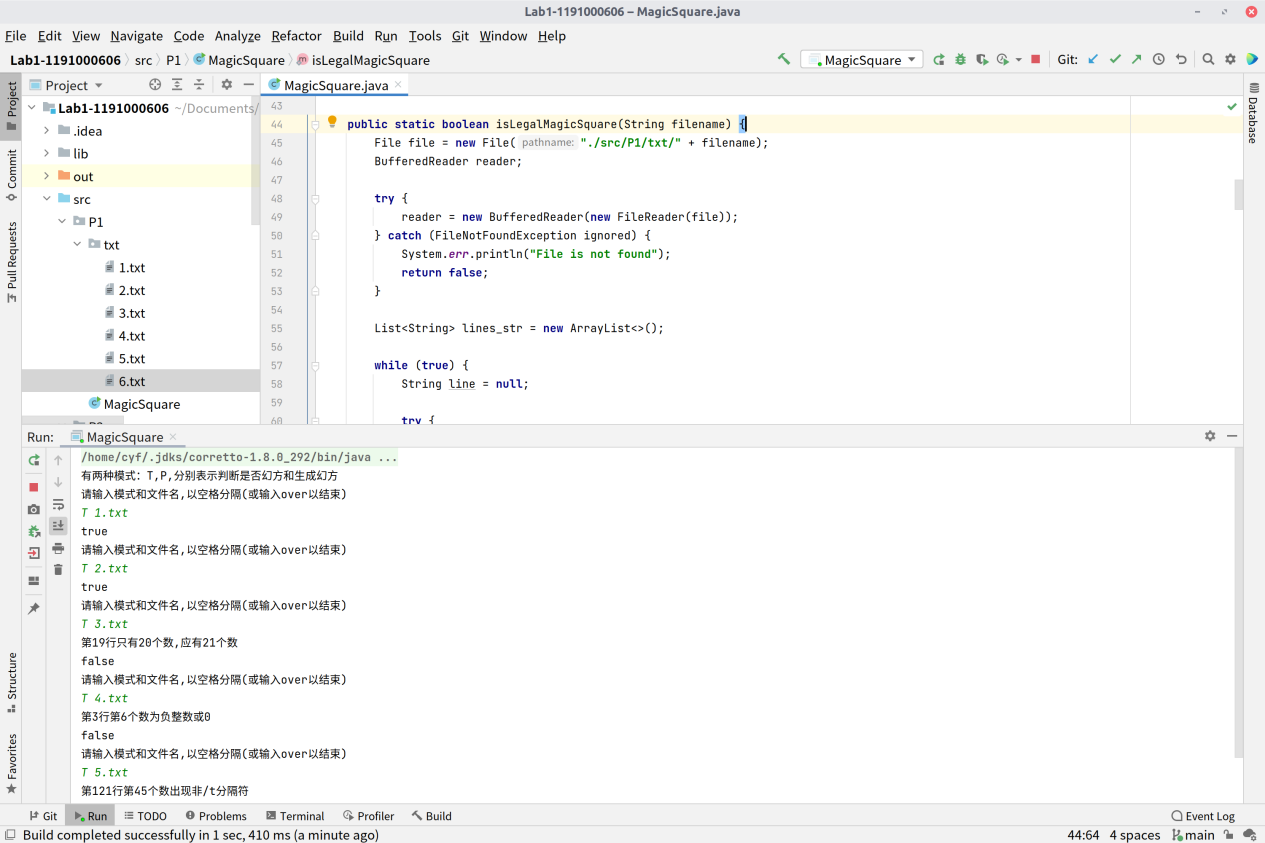
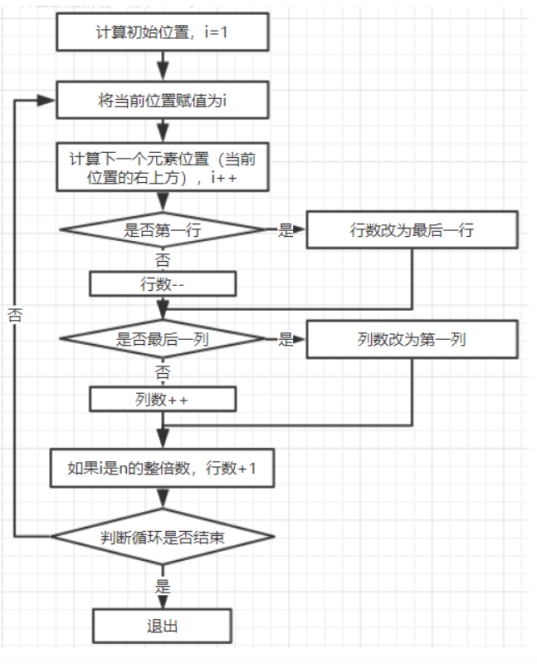


图3 isLegalMagicSquare()结果

### generateMagicSquare()

思路：首先判断输入的n是否合法，合法则继续，不合法则作为exceptions处理后退出。然后进行新建文件，按给定的算法生成数组，将数组写入文件中。

算法分析：流程图如下，主要思路是：首先计算初始位置（0，n/2），赋值为1，



之后每次取当前位置的右上角的位置，设置的值每次加1，如果当前行是第一行，则下一行为最后一行，如果当前列是最右边的列，则下一次取左边第一列，如此重复平方次，就对整个矩阵赋完值了，且满足幻方定义。

设计过程与步骤：

考虑到作为参数的n必须为整数，因此在main中判断n是否为整数，在generateMagicSquare()中判断是否为奇数。

然后打开（新建）文件6.txt，然照所给出的算法得到矩阵，再将矩阵写出文件中，然后保存文件。

结果：

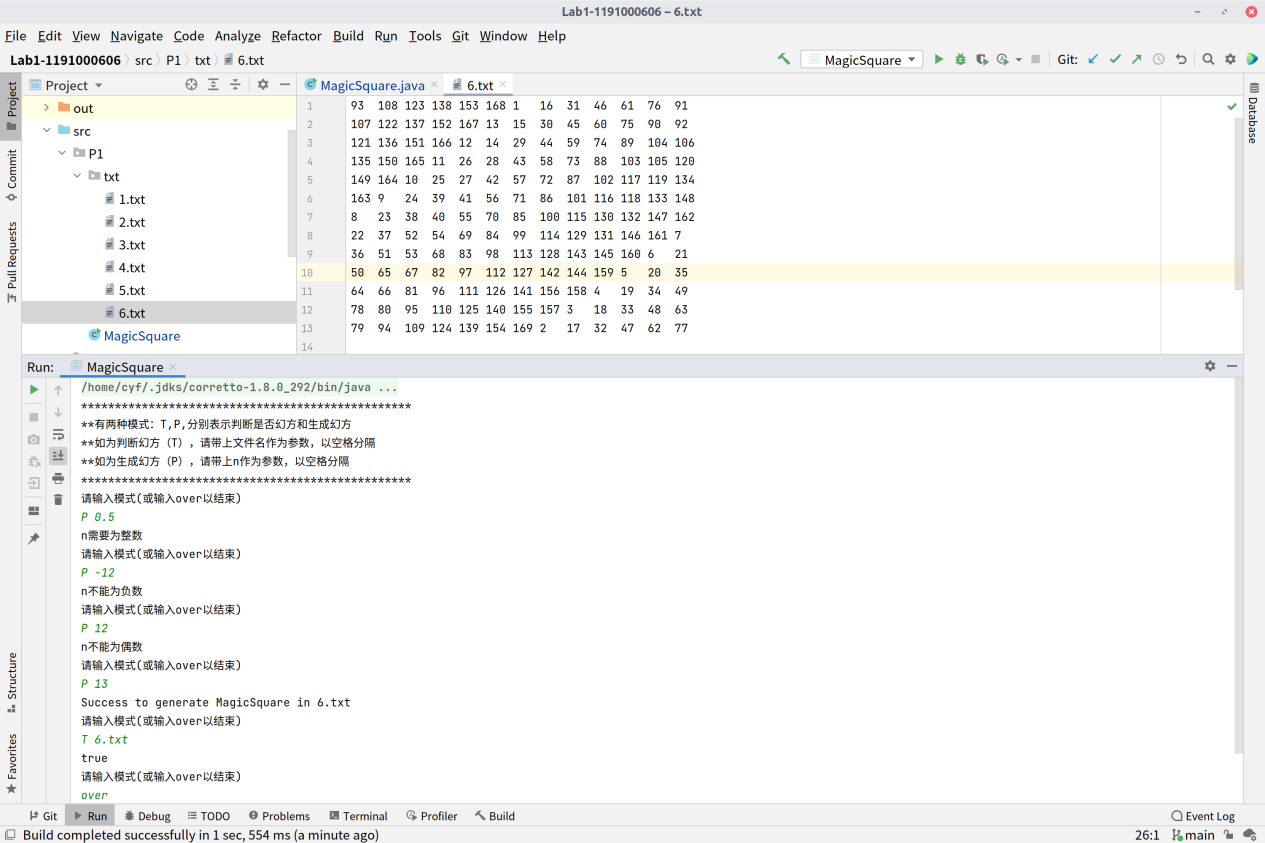


图4 generateMagicSquare()结果

## Turtle Graphics

问题一：要求配置环境，获取代码。

问题二：阅读并理解给出的《合作守则》，根据完成实验的情况，书写一个判断程序，判断各种情况下，是否可以将代码作为作业，并通过测试。

问题三，五，六，七，八：给出了一个绘图工具，Turtle graphics，阅读其中的源代码，使用提供的方法，根据注释，补全各个方法。其中问题八应该较为复杂，需要考虑诸多边际情况和几何关系。

问题四：git提交操作讲解。

### Problem 1: Clone and import

Clone的代码来自于某位老师的开源仓库，我下载了代码，解压到指定文件

夹，commit提交到本地，push到github即可。后续操作则为修改代码->commit->push。这个操作较为顺利。

### Problem 3: Turtle graphics and drawSquare

首先阅读Turtle文件夹下各个类，接口。

观察接口，有前进，转向（顺时针），设置颜色，开始绘制这几个方法。

Point.class中，点有横，纵坐标两个字段，为有private final修饰的double类型，一旦实例化则不可更改。故Point为immutable。

PenColor.enum，枚举类，给出了绘图时需要的颜色。所有的枚举值都是public static final的常量。也可以看作immutable。

LineSegement.class，用来表示线段，有三个字段，start(Point), end(Point), color(Pencolor)，均为private final。由于Point，Pencolor为immutable,所以LineSegement也是immutable。

DrawTurtle是对Turtle接口的一个实现。与TurtleGUI，Action一个构成了图形界面。

drawSquare画一个正方形，较为简单。

### Problem 5: Drawing polygons

calculateRegularPolygonAngle计算正多边形内角，（n-2）\*180/n即可。

drawRegularPolygon较为简单，但注意起始方向为0，即12点钟方向。转动为顺时针转动。

### Problem 6: Calculating Bearings

calculateBearingToPoint，注意几何关系即可。

calculateBearings，使用循环计算角度。

### Problem 7: Convex Hulls

这一题比较困难，尤其是算法中涉及几何的部分。较为繁琐。

理解：这个方法要求，给出一组点的集合。返回一组点的集合，使得这组点的连线构成的多边形，可以囊括所有的点。

当点的个数小于等于3个时，这组点即位所求。直接返回即可。

当点的个数大于3时，首先找到纵坐标最小的点，然后找到下一个点，使得两点连线相对于x轴正方向的夹角最小，这样可以覆盖尽可能多的点。然后以新找到的点为源点，找到下一个点，使得上一条边到这两点之间的连线逆时针转动的角度最小。这样一直循环直到首尾相同。

返回这组点。

### Problem 8: Personal art

我运用了迭代的思想绘制了一个图形。

首先绘制一个边长为10的正方形，然后以其对角线为一边，绘制一个新正方形（与原来的正方形相比逆时针旋转了45度）。然后以新正方形的一条对角线为边长，绘制下一个正方形。依次不断迭代。

其中每个正方形运用了不同的颜色。

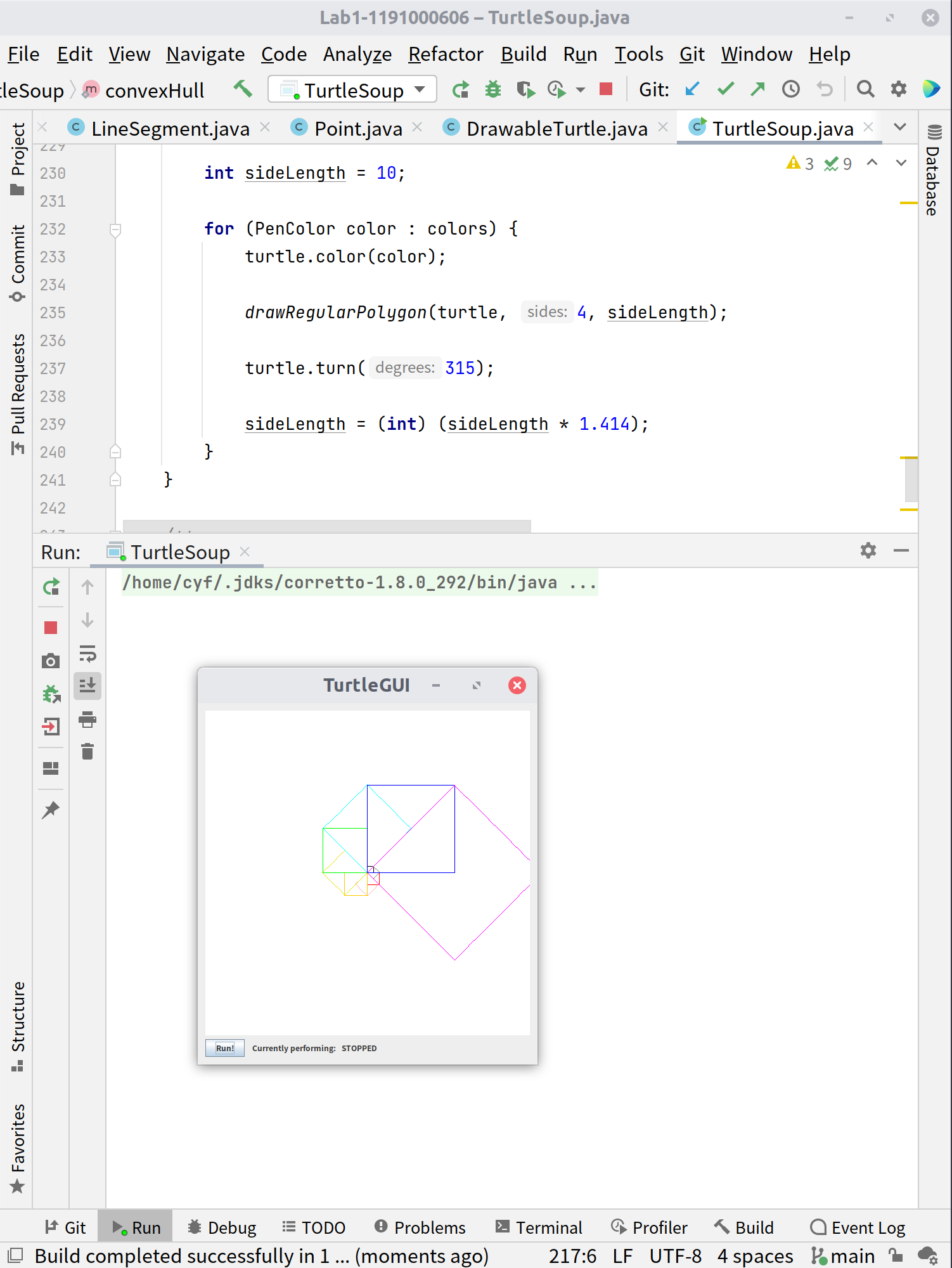


图5 Personal art

### Submitting

IDEA为git提供了友好的图形化界面，点击commit提交到本地，然后push到github即可。

## Social Network

首先要求构造两个类：Person，FriendshipGraph。然后以Person为节点，在FriendshipGraph中构造一个无向图，可以实现填加节点，添加边。并且要求实现一个方法，可以给出任意两点之间的最短距离，不存在即-1。

然后在FriendshipGraph.class的main中放入作业给出的client code，初步实现对Person，Friendshipgraph的测试。然后要求写出FriendshipGraphTest.class，进行测试。

### 设计/实现Person类

#### 属性

首先应该有一个字段name。根据实验指导中的要求：不能有两个Person对象有相同的名字，当用户试图实例化一个相同名字的变量时报错退出。因此在Person中增加一个静态字段：nameList，这个字段属于Person类，而不是某个Person对象。用来判断Person的名字是否重复。

#### 行为

对构造方法进行重写，当判断作为参数的String对象重复时，报错退出，不在则正常返回，并将这个String放入nameList。但这还有一个疑问，如果我们new出了两个String对象，含有相同的字符串，那么nameList是否能判断出已经存在。查阅资料发现，Java对于String类型，有一块特殊的内存区域，凡是已经new过的String，如果再创建一个相同的，那么新创建的String对象会其引用地址会指向已经存在的String对象。所以不用担心new出两个相同String对象后实例化问题。

#### 访问/非访问修饰符设计

Person类设计时应该只会被同Package的FriendshipGraph中调用，因此访问权限设为默认。

根据Java设计相关知识，类的访问权限设置为Public，则该类可以被不同包中的类访问。如为默认，则只可以被同包中的类访问。

包中字段nameList不需要被外界访问，设置为nameList，与不用提供getter, setter方法。该字段属于整个类，加上static，为了保护nameList，不让他引用其他List，为其加上final。

包中字段name，阅读实验指导，虽然未发现有对name的读写，但是还是写出了getter方法，返回name(String)，同时将String设置为private。同时为了防止篡改name字段，要考虑两点：1.有无可能让name引用其他String，理论上应该将name设置为final，但是设置为final，就需要初始化，而不能根据用户的输入设置，所以设置为private，不提供setter方法。同样保证name不会引用其他String。2.name引用的String是否会被修改，由于String是immutable类型，因此不会被修改。

因此，我们设计的Person类是immutable。

### 设计/实现FriendshipGraph类

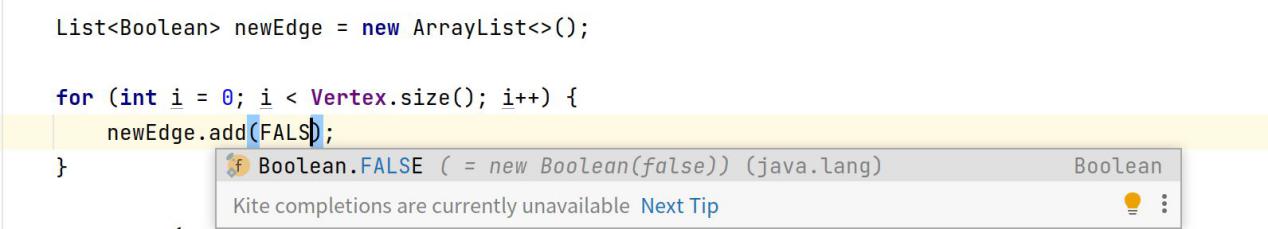
#### 数据结构设计

这是最简单的一部分，显然逻辑结构是一个无向图，存储结构选择邻接矩阵。矩阵的实现选择ArrayList<ArrayList<Boolean>>，ArrayList方便增加元素，由于两点之间只有连接与不连接两种情况，因此使用Boolean类型。

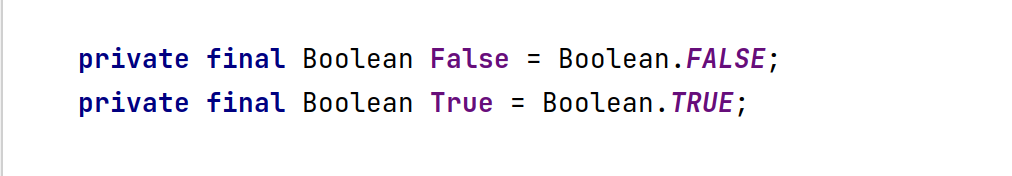
#### 属性设计

G=（V，E）所以肯定有顶点集Vertex，边集Edges。权限设置为private，为防止引用地址被篡改，使用final。考虑到方便增加其中元素，采用ArraryList。

False，True：实验早期我采用这种循环，根据IDEA的提示，他会进行new，因此怀疑会new出多个Boolean。



于是单独new出了False，True两个Boolean对象，并且用final，private，由于Boolean这个类是immutale的，因此这样就保证了实例化出的False，True不会被篡改。也减少了内存的使用。



但询问老师，上网查询了解到，Boolean以及其他由原始数据类型封装出的类，除了Float，Double外，都使用了常量池技术。即在一定的取值范围内，如果检测到试图重复实例化某个对象，便只会进行一下浅复制，新new出的对象实际上与原有对象引用的是同一个地址空间。因此其实上述循环没有消耗过多内存。

经过实际验证，使用上述方式（认为会new很多Boolean对象的方式），确是均引用的是同一个地址空间，如下图所示。

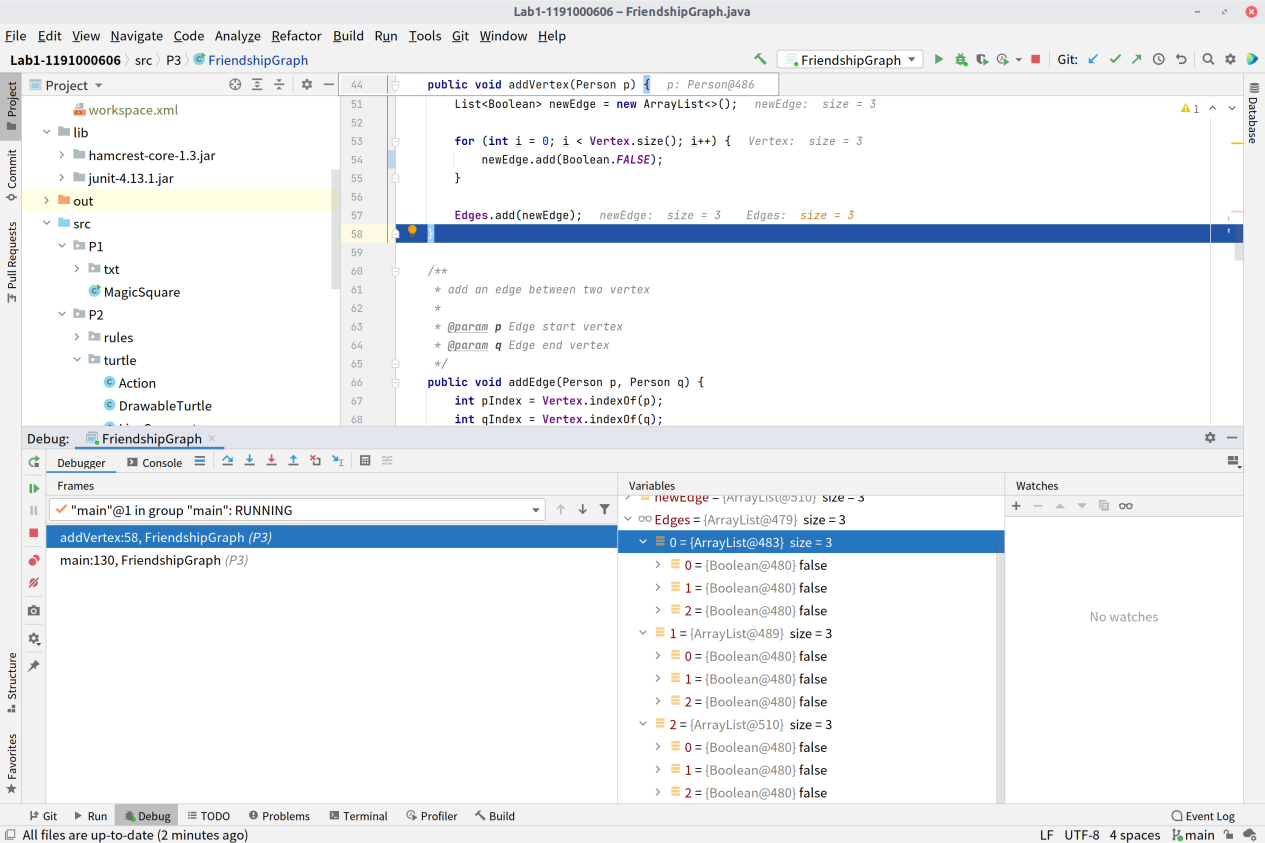
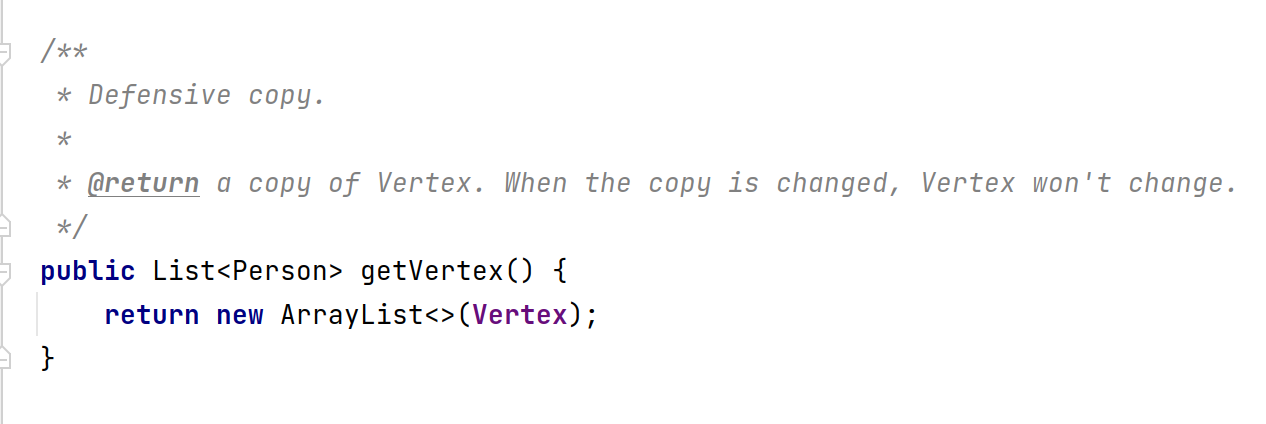


图5 debug查看Boolean对象情况

因此，上述实例化两个对象的意义也仅仅只是保护。防止new出的Boolean对象被篡改，导致ArraryList中所有所有对象都改变。

#### 方法设计

getVertex返回顶点集，如下图。



虽然老师说new全部是浅复制，但是我在JUnit测试中发现，返回的顶点集和图中的顶点集确实拥有不同的内存空间，对一方的改变不会影响另一方。

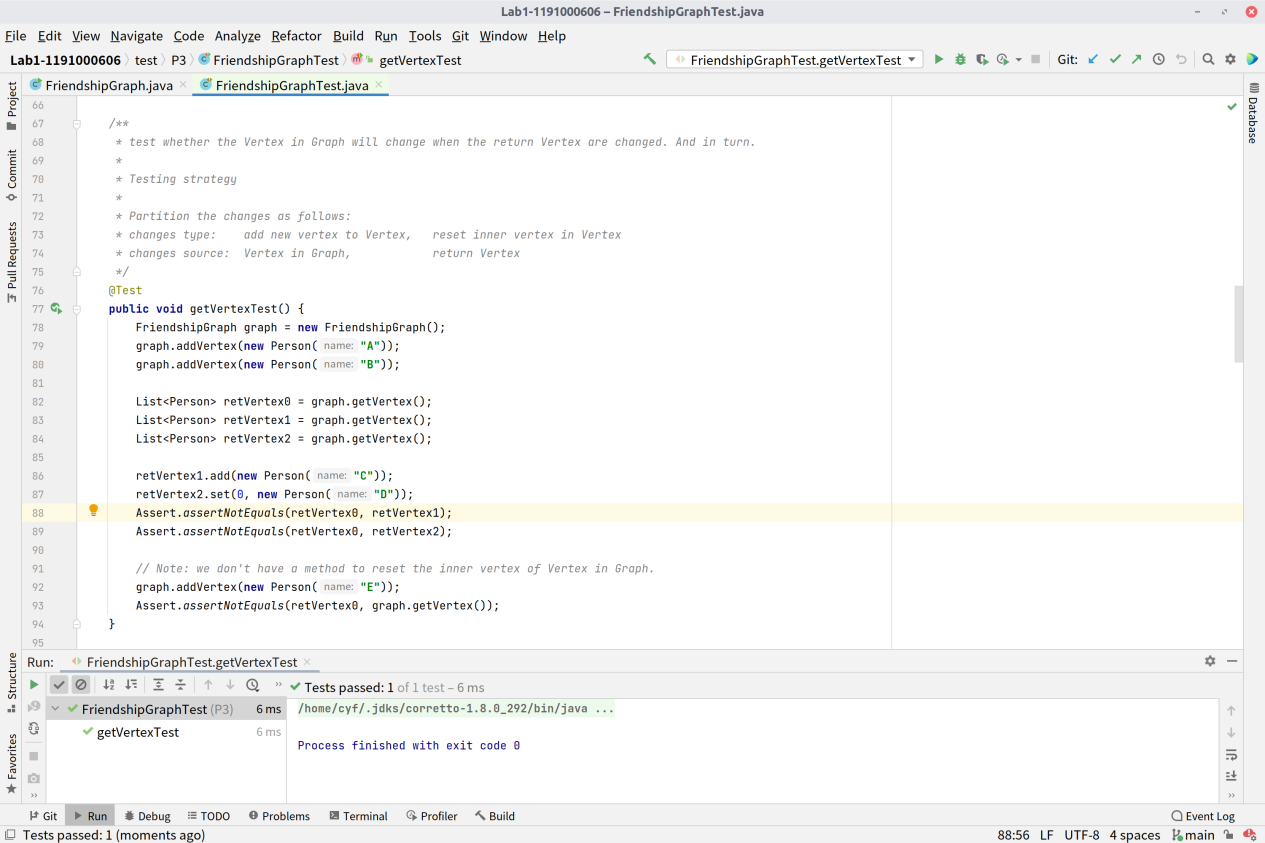


图6 getVertexTest测试

如上图，改变返回的顶点集，与Graph中顶点集比较，不同。改变Graph中顶点集，与之前返回的顶点集作对比，发现不同。证明确实做到了defensive copy。

getEdges(),返回边集。源代码如下图。

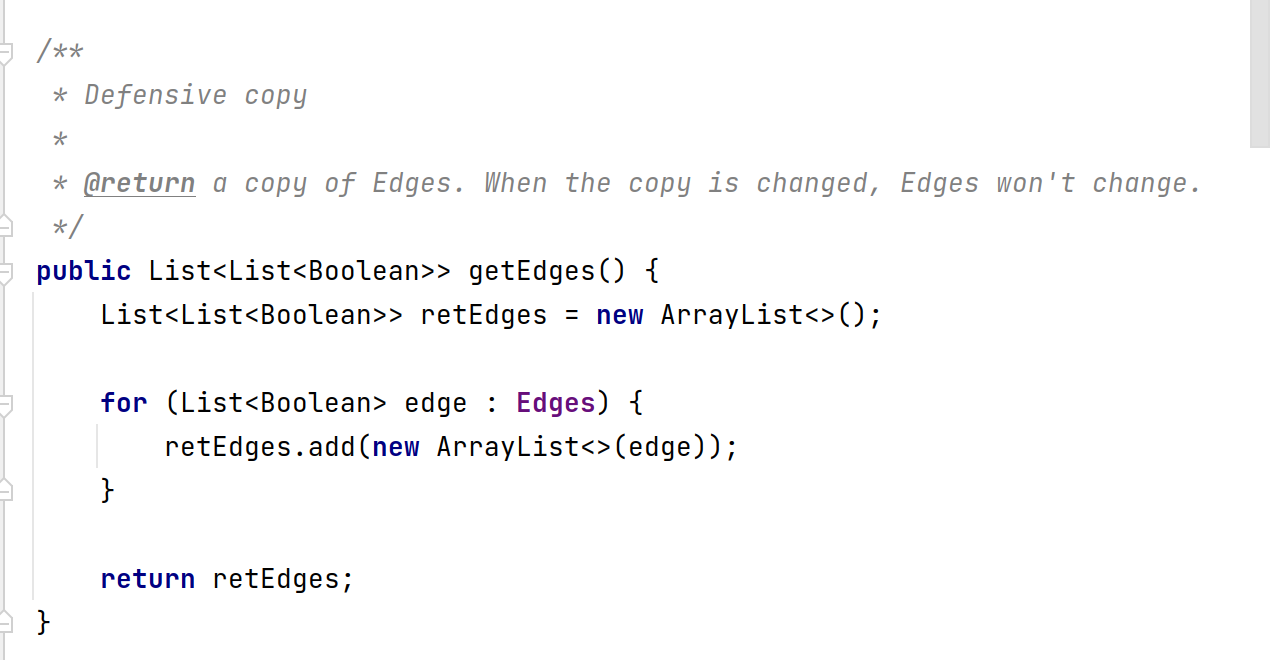


图7 getEdegs()源代码

与getVertex不同，如果同样采用上述的方法，向retEgdes增加一行List <Boolean>，不会影响Graph中的边集，但如果改变边集中某一个的Boolean对象，会影响Graph中的边集。

于是采用逐一List<Boolean>复制的方法，再将List<Boolean>加入到List <List<Boolean>>。经过测试，这样可以做到defensive copy。

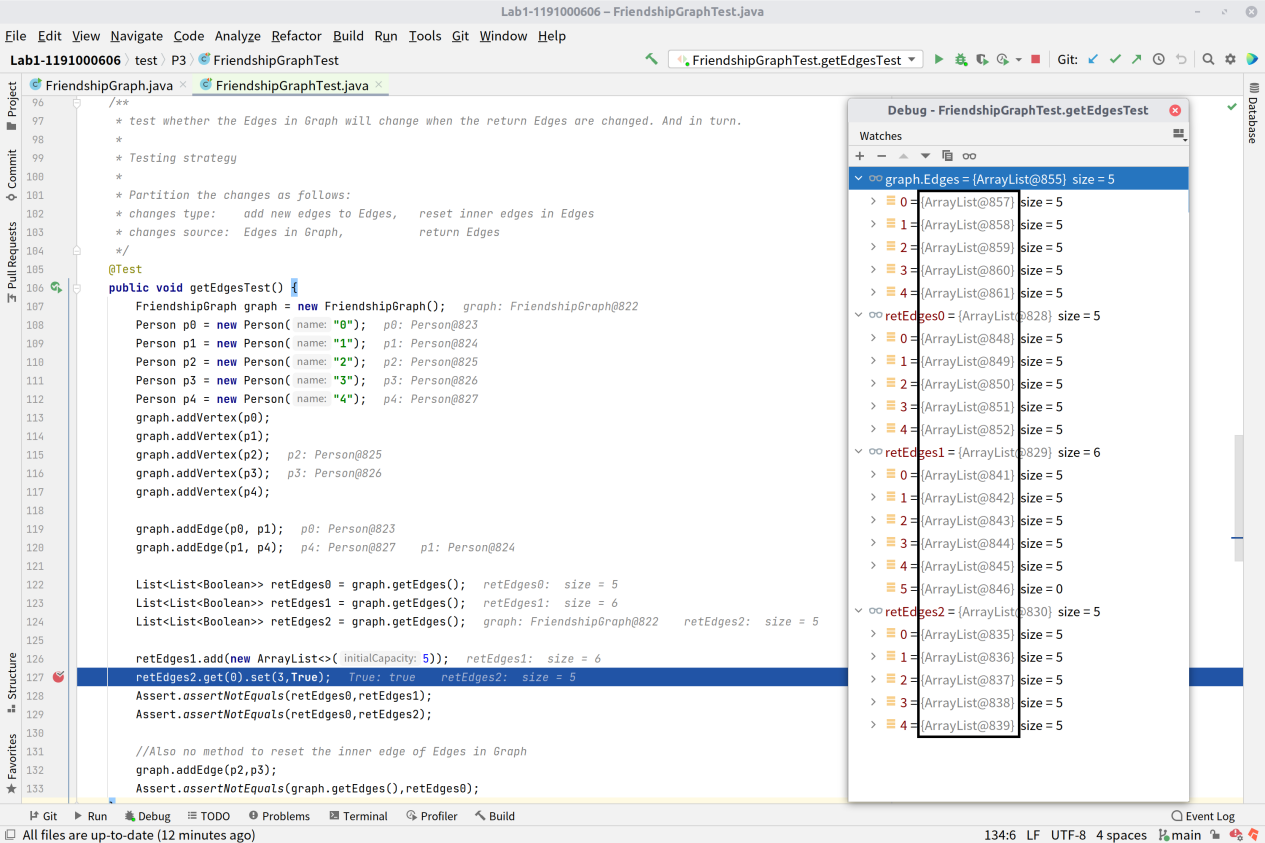


图8 FriendshipTest

图中观察可得，返回的边集内部的ArrayList<Boolean>编号是不同的，根据对IDEA的了解，这表示他们引用的地址空间不同。

addVertex，addEdge均较为简单。

getDistance使用深搜。构造距离数组，初始化为-1，起始点设为0。将起始点加入队列，然后将所有与之相连的节点加入队列，距离设为1。然后弹出队首。然后对队首进行操作，找出与之相连，且从距离为-1的节点加入队列，并将这些节点距离设为队首节点距离值加一，然后弹出队首。当结束节点成为队首，或者队列中没有节点时，结束循环。

### 设计/实现客户端代码main()

由于实验手册已经规定了client代码，因此复制即可。运行结果也符合要求。

### 设计/实现测试用例

#### addVertexTest

等价类划分：添加的结点几种特殊情况，空字符串，字符数字混合字符串，中文字符串。每个等价类取一个测试用例进行测试。

#### addEdgeTest

等价类划分：无，直接测试。

#### getVertexTest

测试是否做到了defensive code。返回的顶点集改变后，是否会影响Graph中的顶点集，或者Graph中的变化后，是否会影响过去返回的顶点集。

等价类的划分：

改变的方式：添加新的顶点，替换顶点集中的顶点。

改变的源头：返回的顶点集，Graph中的顶点集。

由此构造了三个测试用例，因为没有提供直接改变Graph中顶点集的方法。

#### getEdgesTest

同样测试是否做到了defensive code。

对等价类的划分采用了与getVertex相同的思想。如下图所示。

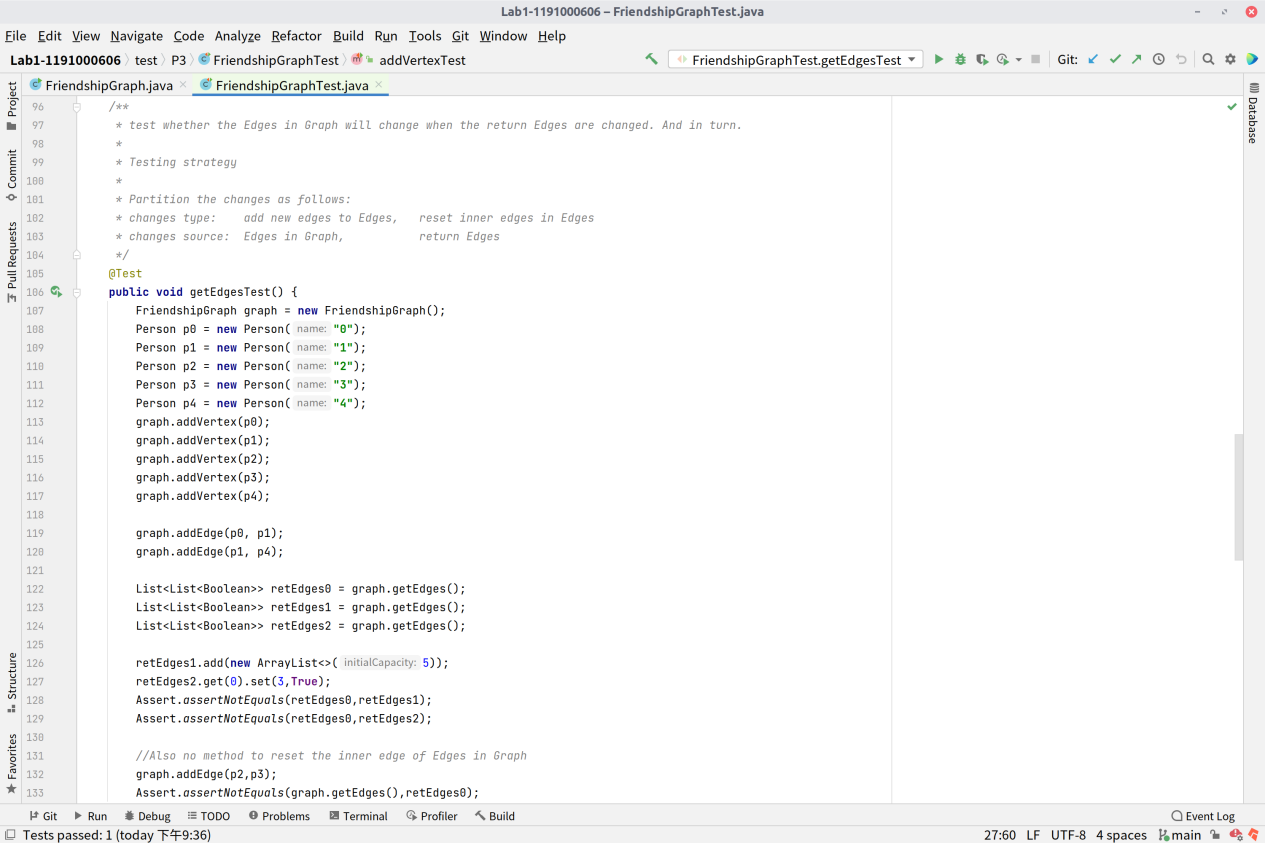


图8 getEdgesTest

#### getDistanceTest

等价类的划分：顶点集的大小：少量，较多

边集的大小：零，少，多。

由此构造一些测试用例组合。

# 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 |
| 寒假期间，约一周 | 不固定 | 学习基础Java编程，完成环境配置，git使用，P1，P2代码内容 | 按计划完成 |
| 2020-5-11 | 16:00-17:30 | 连接远程仓库，将代码打包为jar文件 | 遇到困难，未完成 |
| 2020-5-11 | 20:00-24:00 | 连接远程仓库，采用Token验证 | 按计划完成 |
| 2020-5-12 | 00:00-00:30 | 对P1，P2按照实验指导修改 | 按计划完成 |
| 2020-5-12 | 10:00-11:03 | 规范System.err.println和System.out. println的使用 | 按计划完成 |
| 2020-5-14 | 18:00-18:30 | 修改Personal art, 并按照实验指导和网站要求再次修改P2 | 按计划完成 |
| 2020-5-16 | 13:00-22:21 | 初步完成src/P3，test/P3 | 按计划完成 |
| 2020-5-17 | 20:30-21:00 | 初步实现P3中defensive copy | 有隐患 |
| 2020-5-18 | 15:45-17:30 | 完成P3中的defensive copy，对src/P3进行代码优化 | 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 |
| 无法连接git远程私有仓库 | 使用Token的方法进行验证，而不是使用github账号密码 |
| .java文件不能运行 | 在main()中添加参数String[] args |
| 不熟悉Java标准库（？）各种类以及类中的方法 | 在IDEA查看其规约，上网查询各种API的说明 |

# 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

经验：要学会使用Java提供的便捷的类和方法。

使用好的IDE和git来提高编程效率

合理运用修饰符

## 针对以下方面的感受

1. Java编程语言是否对你的口味？

非常喜欢Java编程。

1）面向对象编程

在这一方面，我认为Java比C/Python做的好，体会到了一切皆可以是对象，还有一些不可实例化的类存在。同样还有接口的概念，因为没接触过C++，因此接触到这些概念后感觉非常神奇，对自己的编程世界观有了很大的影响。

1. 便捷的方法

与上学期的C语言相比，Java提供了很多方便的方法，为很多数据结构创建了各式各样的类，不再需要从0开始构造。同时，阅读函数规约的方式来了解一个方法，也让我摆脱了用法全靠百度或者mooc的情况。

同样提供了很多方法的还有Python，但是我感觉与Python同样便捷的方法相比，Java方法的使用更加克制严谨。

1. 信息隐藏，信息保护

学习Java，我学会使用各种修饰符，也在P3中尝试使用来让自己的类是immutable的。

如果要让一个类是immutable的，那么对于其中的属性，如果是基本数据类型，可以加上fianl的。如果是引用数据类型，不仅要保证引用的数据类型的是immutable的，还要保证变量引用的地址不会被改变，加上final即可。同时对于所有的属性，建议全部设置为private，然后依据情况选择是否提供getter，setter方法。

关于访问权限的设计，我非常认同以下观点。

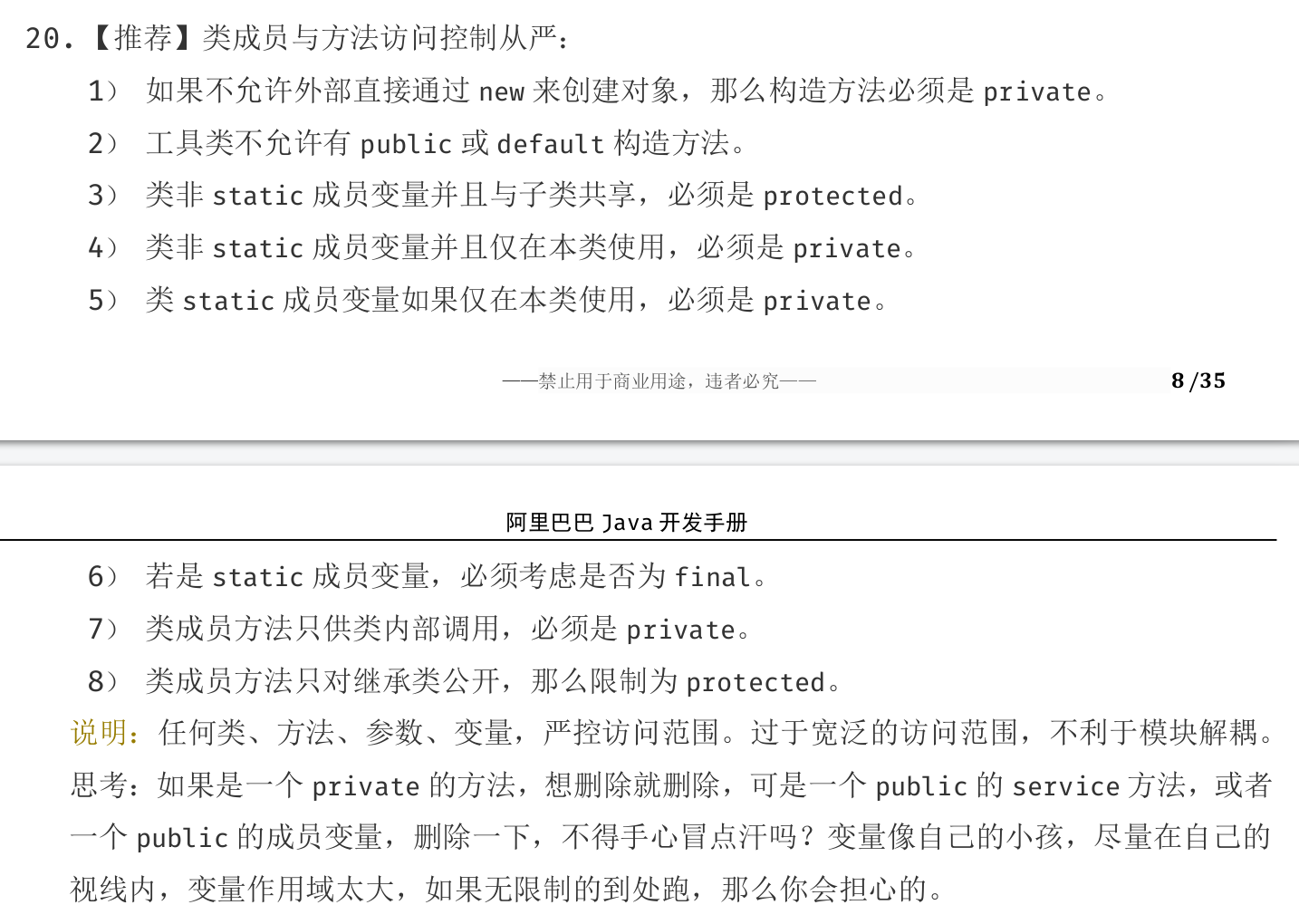


图8 关于修饰符的选择

出自《阿里巴巴 Java 开发手册》，访问权限的控制从严，既是对自己负责，也是对调用自己的类的人负责。

1. Junit测试

提高了代码开发效率。

1. 关于IntelliJ IDEA；

非常友好，美观，功能强大的IDEA。

快速，便捷自动补全，可以一键下载，选择，配置jdk环境。非常便捷的添加第三方包。

经常给出warning，不仅仅是可能出错的地方，还可以纠正一些不良的编程习惯，例如将C-style code转化为Java-style，添加一些修饰符，可以复用的代码。

debug模式中直接在源代码页面标注一些重要信息。

自带git工具，方便commit，push，还有git提交的图形化界面。

可以点击某个提供类，查看其中的属性，方法规约。（本人Java相当多的知识来自于阅读这些方法规约以及API文档）。

自动生成方法规约模板，列出所有的参数@param，@return。

1. 关于Git和GitHub；

方便的代码管理工具，就是经常连接不上去。Github上还有许多有意思的资料。

1. 关于CMU和MIT的作业；

代码难度应该是一样的，但在网页上看到：可以求助于助教，我认为这个是CMU和MIT的作业设计中比较优秀的地方。

1. 关于本实验的工作量、难度、deadline；

我认为是个工作量比较大的，如果之前没接触过Java语言。很多语法需要重新学习。同时构建开发环境，连接连接不上的github，对P2中Turtle包中路径的修改，我认为比较繁琐。并且近期有考试，和其他课程的实验，我认为时间比较紧。

1. 关于初接触“软件构造”课程；

不适合根据PPT自学。我认为学这门课最好的方法是多敲代码，我在本次实验中运用到的Junit，collections类，设计类为immutable，对理解课堂上的概念有很大帮助。