

Lab Manuals for  
***Software Construction***

**Lab-1  
Fundamental Java Programming and Testing**



School of Computer Science and Technology

Harbin Institute of Technology

Spring 2019

**目录**

[1 实验目标 1](#_Toc535824350)

[2 实验环境 1](#_Toc535824351)

[3 实验要求 1](#_Toc535824352)

[3.1 Magic Squares (MIT) 1](#_Toc535824353)

[3.2 Turtle Graphics (MIT) 4](#_Toc535824354)

[3.3 Social Network (CMU) 5](#_Toc535824355)

[3.4 Tweet Tweet（MIT） 7](#_Toc535824356)

[4 实验报告 8](#_Toc535824357)

[5 提交方式 8](#_Toc535824358)

[6 评分方式 9](#_Toc535824359)

# 实验目标

本次实验通过求解四个问题（其中一个可选），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

实验环境设置请参见Lab-0实验指南。

本次实验在GitHub Classroom中的URL地址为：

<https://classroom.github.com/a/qL3Yd1XX>

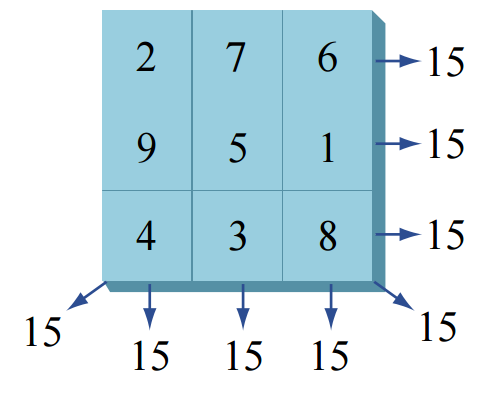
请访问该URL，按照提示建立自己的Lab1仓库并关联至自己的学号。

本地开发时，本次实验只需建立一个项目，统一向GitHub仓库提交。实验包含的3(+1)个任务分别在不同的目录内开发，具体目录组织方式参见各任务最后一部分的说明。请务必遵循目录结构，以便于教师/TA进行测试。

# 实验要求

## Magic Squares (MIT)

A magic square of order *n* is an arrangement of n×n numbers, usually distinct integers, in a square, such that the n numbers in all rows, all columns, and both diagonals sum to the same constant (see Wikipedia: [Magic Square](https://en.wikipedia.org/wiki/Magic_square)).



**要求1**

You are to write a Java program (MagicSquare.java) for checking the row/column/diagonal values of a matrix and then judging whether it is a magic squares.

* We give you five text files: 1.txt, 2.txt, ..., 5.txt. Download them from <https://github.com/rainywang/Spring2019_HITCS_SC_Lab1/tree/master/P1> and add them into your project directory \src\P1\txt\;
* For each file: open the file, and check that all rows indeed sum to the same constant.
* Check that the columns and the diagonal also sum to the same constant.
* Return a boolean result indicating whether the input is a magic square or not.
* 函数规约：boolean isLegalMagicSquare(String fileName)
* 在main()函数中调用五次isLegalMagicSquare()函数，将5个文本文件名分别作为参数输入进去，看其是否得到正确的输出（true, false）。
* 需要能够处理输入文件的各种特殊情况，例如：文件中的数据不符合Magic Square的定义（行列数不相等、并非矩阵等）、矩阵中的某些数字并非正整数、数字之间并非使用\t分割等。若遇到这些情况，终止程序执行（isLegalMagicSquare函数返回false），并在控制台输出错误提示信息。

**Some Hints**

Copy all five text files to the \src\P1\txt\ directory of your project. You can also use absolute paths to the files (c:\somedir\1.txt on Windows or /Users/myuser/1.txt on Mac). However, it is better to use the relative paths of these files.

You will need to handle or re-throw IOException.

Read the files line by line. Use ... = myLine.split("\t"); to break apart each line at the tab character, producing an array of String (String[]), each containing one value. Consult the Java API reference for [String.split](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#split-java.lang.String-)).

Finally, use ... = Integer.valueOf(substring); to transform each string value into an integer value.

**要求2**

阅读以下代码，将其加入你的MagicSquare类中作为一个静态函数，并试着在main()中测试它。

|  |
| --- |
| public static boolean generateMagicSquare(int n) {  int magic[][] = new int[n][n];  int row = 0, col = n / 2, i, j, square = n \* n;  for (i = 1; i <= square; i++) {  magic[row][col] = i;  if (i % n == 0)  row++;  else {  if (row == 0)  row = n - 1;  else  row--;  if (col == (n - 1))  col = 0;  else  col++;  }  }  for (i = 0; i < n; i++) {  for (j = 0; j < n; j++)  System.out.print(magic[i][j] + "\t");  System.out.println();  }  return true;  } |

为该函数绘制程序流程图，并解释它如何根据输入的参数（奇数n）生成一个n×n的Magic Square。据此为上述代码添加中文注释。

如果输入的n为偶数，函数运行之后在控制台产生以下输出：

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 12

at MagicSquare.generateMagicSquare(MagicSquare.java:17)

at MagicSquare.main(MagicSquare.java:121)

如果输入的n为负数，函数运行之后在控制台产生以下输出：

Exception in thread "main" java.lang.NegativeArraySizeException

at MagicSquare.generateMagicSquare(MagicSquare.java:11)

at MagicSquare.main(MagicSquare.java:121)

请查阅JDK了解上述异常的含义，并分析该函数为何会产生这些异常（注：我们将在Lab4中训练异常处理机制）。

对该函数做扩展：(1) 将产生的magic square写入文件\src\P1\txt\6.txt中；(2) 当输入的n不合法时（n为偶数、n为负数等），不要该函数抛出异常并非法退出，而是提示错误并“优雅的”退出——函数输出false结束。

利用你前面已经写好的isLegalMagicSquare()函数，在main()函数判断该函数新生成的文本文件6.txt是否符合magic square的定义。

**项目的目录结构**

项目： Lab1\_学号

目录： src

子目录： P1

文件： MagicSquare.java

函数： boolean isLegalMagicSquare(String fileName)

void generateMagicSquare(int n)

void main(String[] args)

子目录： txt

文件： 1.txt

...

6.txt

请使用git指令将符合上述结构的代码push到你的GitHub Lab1仓库中。

## Turtle Graphics (MIT)

请阅读<http://web.mit.edu/6.031/www/sp18/psets/ps0/>，遵循该页面内的要求完成编程任务。

* 在Problem 1: Clone and import中你无法连接MIT的didit服务器，请从<https://github.com/rainywang/Spring2019_HITCS_SC_Lab1/tree/master/P2>获取代码。
* 忽略Problem 2: Collaboration policy**。**
* 在Problem 4: Commit and push your work so far步骤的第g步，忽略页面中涉及Athena和didit的内容，请使用git指令将代码push到你的GitHub仓库中。
* 在页面最后的Submitting步骤中，请同样将你的代码push到你的GitHub Lab1仓库上。
* 其他步骤请遵循MIT作业页面的要求。

**项目的目录结构：**

项目名称： Lab1\_学号

目录： src

子目录： P2

（从GitHub上clone的目录/文件不应改变，直接放置在P2目录下）

rules

….java

turtle

….java

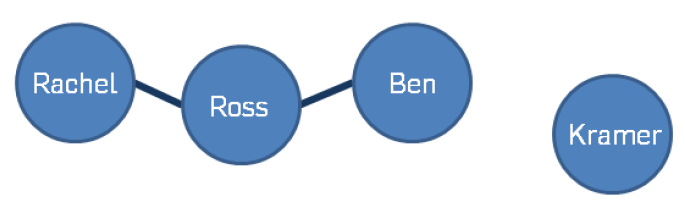
## Social Network (CMU)

本作业来自于CMU 15-214软件构造课。

Implement and test a FriendshipGraph class that represents friendships in a social network and can compute the distance between two people in the graph. An auxiliary class Person is also required to be implemented.

You should model the social network as an **undirected** graph where each person is connected to zero or more people, but your underlying graph implementation should be **directed**. 注：本问题拟刻画的社交网络是无向图，但你的类设计要能够支持未来扩展到有向图。正因为此，如果要在两个Person对象A和B之间增加一条社交关系，那么需要同时调用addEdge(A,B)和addEdge(B,A)两条语句。

For example, suppose you have the following social network:



Your solution must work with the following client implementation.

1. FriendshipGraph graph = new FriendshipGraph();
2. Person rachel = new Person("Rachel");
3. Person ross = new Person("Ross");
4. Person ben = new Person("Ben");
5. Person kramer = new Person("Kramer");
6. graph.addVertex(rachel);
7. graph.addVertex(ross);
8. graph.addVertex(ben);
9. graph.addVertex(kramer);
10. graph.addEdge(rachel, ross);
11. graph.addEdge(ross, rachel);
12. graph.addEdge(ross, ben);
13. graph.addEdge(ben, ross);
14. System.out.println(graph.getDistance(rachel, ross));

//should print 1

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

//should print 2

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

//should print 0

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

//should print -1

Your solution should work with the client code above. The getDistance method should take two people (as Person) as arguments and return the shortest distance (an int) between the people, or -1 if the two people are not connected (or in other words, there are no any paths that could reach the second people from the first one).

Your graph implementation should be reasonably scalable. We will test your graph with several hundred or thousand vertices and edges.

Use proper access modifiers (public, private, etc.) for your fields and methods. If a field/method can be private, it should be private.

Do not use static fields or methods except for the main method(s) and constant

Follow the Java code conventions, especially for naming and commenting. Hint: use Ctrl + Shift + F to auto-format your code!

Add short descriptive comments (/\*\* ... \*/) to all public methods.

**Additional hints/assumptions**

For your implementation of getDistance, you may want to review [breadth-first search](https://en.wikipedia.org/wiki/Breadth-first_search).

You may use the standard Java libraries, including classes from java.util, but no third-party libraries.

You may assume that each person has a unique name.

You may handle incorrect inputs however you want (printing to standard out/error, silently failing, crashing, throwing a special exception, etc.)

You should write additional samples to test your graph, similar to our main method.

To print something to standard out, use System.out.println. For example:

System.out.println("DON'T PANIC");

You should also write JUnit test code to test the methods addVertex(), addEdge(), and getDistance() of the class FriendshipGraph. All the test cases should b e included in FriendshipGraphTest.java in the directory \test\P3. Test cases should be sufficient enough.

如果将上述代码的第10行注释掉（意即rachel和ross之间只存在单向的社交关系ross->rachel），请人工判断第14-17行的代码应输出什么结果？让程序执行，看其实际输出结果是否与你的期望一致？

如果将第3行引号中的“Ross”替换为“Rachel”，你的程序会发生什么？这其实违反了“Each person has a unique name”的约束条件。修改你的FriendshipGraph类和Person类，使该约束能够始终被满足（意即：一旦该条件被违反，提示出错并结束程序运行）。

**项目的目录结构**

项目名称： Lab1\_学号

src

P3

FriendshipGraph.java

Person.java

test

P3

FriendshipGraphTest.java

请使用git指令将符合上述结构的代码push到你的GitHub Lab1仓库中。

## Tweet Tweet（MIT）

请阅读<http://web.mit.edu/6.005/www/sp16/psets/ps1/>，遵循该页面内的要求完成编程任务。

* 在页面的Get the code步骤中，你无法连接MIT的didit服务器，请从<https://github.com/rainywang/Spring2019_HITCS_SC_Lab1/tree/master/P4>获取代码。
* 在页面最后的Submitting步骤中，请同样将你的代码push到你的GitHub仓库上。
* 其他步骤请遵循MIT作业页面的要求。

**项目的目录结构：**

项目名称： Lab1\_学号

src

P4

twitter

….java

test

P4

twitter

…Test.java

请使用git指令将符合上述结构的代码push到你的GitHub Lab1仓库中。

# 实验报告

针对上述四个编程题目，请遵循CMS上Lab1页面给出的**报告模板**，撰写简明扼要的实验报告。

**实验报告的目的是记录你的实验过程，尤其是遇到的困难与解决的途径。**不需要长篇累牍，记录关键要点即可，但需确保报告覆盖了本次实验的所有开发任务（4个问题，每个问题下有一系列任务）。

注意：

* 实验报告不需要包含所有源代码，请根据上述目的有选择的加入关键源代码，作为辅助说明。
* 请确保报告格式清晰、一致，故请遵循目前模板里设置的字体、字号、行间距、缩进；
* 实验报告提交前，请“目录”上右击，然后选择“更新域”，以确保你的目录标题/页码与正文相对应。

# 提交方式

**截止日期：**第3周周日夜间23:55。截止时间之后通过Email等其他渠道提交实验报告和代码，均无效，教师和TA不接收，学生本次实验无资格。

**源代码：**从本地Git仓库推送至个人GitHub的Lab1仓库内。

**实验报告：**除了随代码仓库(doc目录)提交至GitHub之外，还需手工提交至CMS实验1页面下。

# 评分方式

TA在第1-2周实验课上现场验收：学生做完实验之后，向TA提出验收申请，TA根据实验要求考核学生的程序运行结果并打分。现场验收并非必需，由学生主动向TA提出申请。

Deadline之后，教师和TA对学生在GitHub上的代码进行测试，阅读代码和实验报告，做出相应评分。