哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2021 年春季学期

计算学部《软件构造》课程

Lab 1 实验报告

姓名	李昆泽
学号	1190201018
班号	1936603
电子邮件	3343033352@qq.com
手机号码	15720828552

目录

1 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。

- 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

2 实验环境配置

在本次实验之前，我的电脑里其实已经配置好了包含 JDK、Eclipse、Git 等软件的开发环境，但是我的 JDK 版本不太对，需要进行修改。另外，我还没有注册 Github 账号，关于 JUnit 的配置及使用也比较陌生。

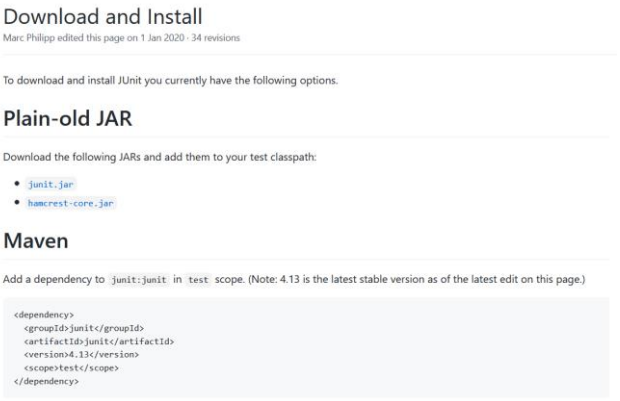
关于 JDK 的版本问题，实验要求是使用 JDK8、JDK9 或 JDK10，我目前的 JDK 版本是 JDK15。这个时候我并没有选择重新下载一个新的 JDK，而是将原来的工程转化为 Maven Project，可以通过创建.travis.yml 文件进行配置。例如上图就是使用 JDK10 进行工程的构建。

```
1  language: java
2  jdk:
3    - openjdk10
```

并且，我后来发现这样做对后面 JUnit 的配置、使用 Travis-CI 进行在线 build 都提供了便利。

关于 Github 账号的注册我在这里就不赘述了，按照流程一步步注册就好。关于 git 本地仓库与 github 仓库的关联，我遇到的问题是初始的 github 仓库是没有 master 分支的，现在的默认分支是 main 分支。通过参考这篇博客（https://blog.csdn.net/wankui/article/details/53328369）我的问题得到了解决。这里主要涉及到的是切换分支的问题。

然后是 JUnit 的配置，这个通过老师提供的链接发现有两种方法。



Download and Install

Marc Philipp edited this page on 1 Jan 2020 · 34 revisions

To download and install JUnit you currently have the following options.

Plain-old JAR

Download the following JARs and add them to your test classpath:

- `junit.jar`
- `hamcrest-core.jar`

Maven

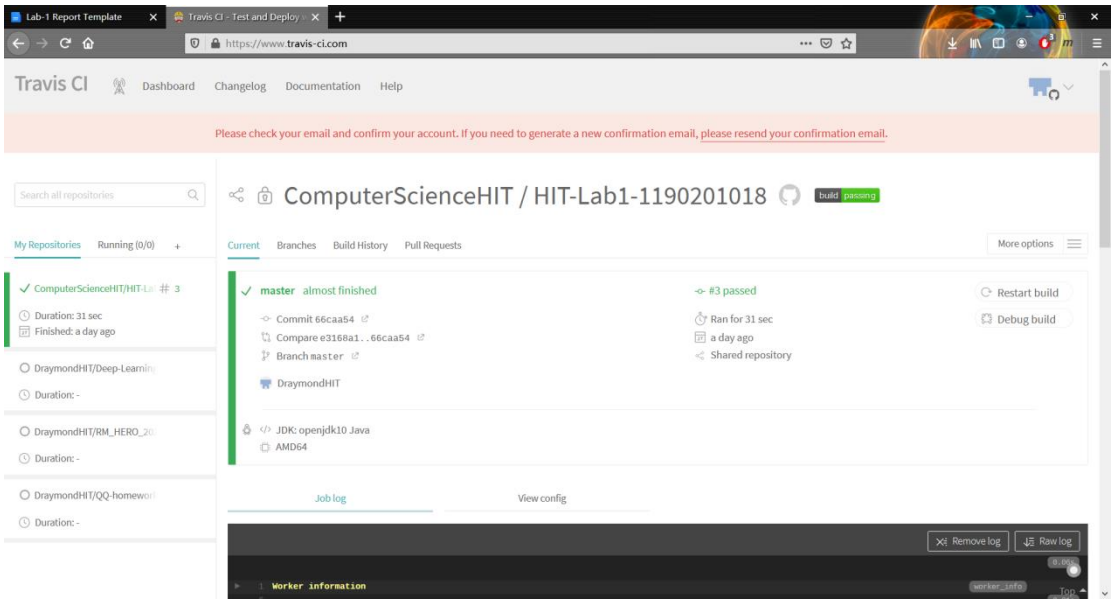
Add a dependency to `junit:junit` in `test` scope. (Note: 4.13 is the latest stable version as of the latest edit on this page.)

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.13</version>
  <scope>test</scope>
</dependency>
```

由于我之前就是用 Maven 构建的工程，所以这里我就把上面的两个包对应的代码添加到了 pom.xml 中，截图如下。

```
11        <excludes>
12          <exclude>**/*.java</exclude>
13        </excludes>
14      </resource>
15    </resources>
16    <plugins>
17      <plugin>
18        <artifactId>maven-compiler-plugin</artifactId>
19        <version>3.8.1</version>
20        <configuration>
21          <release>10</release>
22        </configuration>
23      </plugin>
24    </plugins>
25  </build>
26  <dependencies>
27    <dependency>
28      <groupId>junit</groupId>
29      <artifactId>junit</artifactId>
30      <version>4.13.2</version>
31    </dependency>
32    <dependency>
33      <groupId>org.hamcrest</groupId>
34      <artifactId>hamcrest-core</artifactId>
35      <version>1.3</version>
36    </dependency>
37  </dependencies>
38 </project>
```

这样就配置好 JUnit 了，后面的在线 build 主要是跟前面的.travis.yml 文件有关，按照我之前的写法，是按照 JDK10 进行 build，符合实验要求。下面是在线 build 成功的截图。



GitHub Lab1 仓库的 URL 地址：https://github.com/ComputerScienceHIT/HIT-

3 实验过程

3.1 Magic Squares

在这里简要概述你对该任务的理解。

该任务主要是判断输入的二维数组是否是幻方（Magic Squares）。一个幻方需要满足：

（1）幻方必须是方阵

（2）n*n 的幻方里的元素必须是从 1 到 n*n

（3）每行、每列和每条对角线的和必须相同

除此之外，还需要对一些特殊情况进行处理（例如存在非法字符）。

解决这道题的办法就是对幻方需要满足的条件一一判断，只要有条件不符合就不是幻方，如果条件全部满足就是幻方。

3.1.1 isLegalMagicSquare()

我的设计思路其实很简单，就是先把输入的内容保存在一个二维数组中，然后按照上述的条件一一判断，只要条件全部满足就返回 true，否则就返回 false。

首先是加载文件与内容的读取。可以使用如下的语句按行进行读取。

```
//加载文件
FileReader file = new FileReader(fileName);
BufferedReader br = new BufferedReader(file);
```

然后可以定义一个 String 类型的变量 line，首先使用 br.readLine()的方法实现按行读取然后把读取的字符串赋值给 line。

接下来我将从几个判断条件出发依次给出实现方法。

（1）所有数字必须是正整数

我的想法是先定义一个集合，这个集合中只有 0 到 9，以及“\t”“\n”这些字符，如果出现其他字符，那么要么是出现了非法字符，要么是输入的数值不是正整数。这样做唯一无法判断的是 0，这里可以在后面的代码中进行实现。

（2）必须是矩阵

记录下第一行元素的个数，后面所有行、所有列上元素的个数必须和第一行元素的个数相等。

（3）每行、每列和每条对角线的和必须相同

和上一个条件的判断类似，可以先求出第一行元素的和，然后以此为比较的基准，后续所有行、列和对角线上元素的和必须与这个相等。在求和的过程中，需要使用 Integer.valueOf 将字符串转化为整数，此时一方面可以进行求和，另一方面可以对是否出现重复元素进行判断，也可以对是否出现 0 进行判断。

（4）幻方里的元素必须是从 1 到 n*n

首先初始化一个 boolean 类型的数组 used，所有元素全部初始化为 false，在对条件（3）进行判断时，对出现的整数值可以把它在 used 中下标对应的元素赋值为 true。最后对于一个 n*n 的矩阵，如果从 used[1]到 used[n]均为 true，则满足该条件，否则不满足。

另外，一旦有条件不满足可直接返回 false 退出函数，但在返回前要关闭 FileReader 和 BufferedReader 对象。

3.1.2 generateMagicSquare()

实现思路：

（1）初始化

初始化矩阵，确定填充的初始位置（row=0，col=n/2）

（2）循环 n*n 次填充矩阵

a. 将矩阵的[row, col]位置填充为 i；

b. 在保证坐标在矩阵范围内的情况下使 row－，col++

3.2 Turtle Graphics

该任务是基于现有项目（一个可以在屏幕上移动并进行画图的 turtle）进行更高级功能的设计（即设计相关的方法并进行封装）。其实总的来看，这是一个较为复杂的项目，但是我们需要做的不是从头编写这个工程，我们只需要在原来工程的基础上实现几个新的方法即可。

3.2.1 Problem 1: Clone and import

首先打开终端，选择 Git Bash，在终端中输入 git clone <<url>>，这里的 url 是仓库 clone 的地址，可以在 GitHub 上找到。执行完这条语句后便可以将该任务代码 clone 到本地。

在本地创建 git 仓库其实很简单，只需要选择（或新建）一个空的文件夹，然后在文件夹中打开 Git Bash，输入 git init 即可初始化该文件夹为 git 仓库。

使用 git 管理本地仓库，有几个比较基本的命令。第一，可以用命令 git add <file>告诉 Git，把文件添加到仓库；第二，用命令 git commit-m <message>告诉 Git，把文件提交到仓库。除此之外，可以通过 git log 查看历史记录，也可以通过 git reset 等命令回退至之前版本或撤销操作。

3.2.2 Problem 3: Turtle graphics and drawSquare

这部分非常简单，turtle 的 forward(units)方法实现前进 units 个单位长度功能，而 turn(degrees)实现的是转向功能。drawSquare 需要画一个正方形，只要连续 4 次前进相同的长度然后旋转 90 度即可。

3.2.3 Problem 5: Drawing polygons

这部分需要画一个正多边形，需要解决的核心问题是根据边的数量推算每次要转向多少度。这个在数学上是有相关公式的，即 degrees = 360 - 180/sides，这个也就是 calculateRegularPolygonAngle 这个方法要实现的功能，要注意转化为 double 类型进行运算。

另外，这样算出来的是正多边形每个内角的大小，还需要用 180 度减去这个角度以得到 turtle 每次需要转向的角度大小。

3.2.4 Problem 6: Calculating Bearings

首先计算前进方向和 y 轴正向的夹角，

```
alpha = Math.atan((double)(targetX-currentX)/(targetY-currentY))*180/Math.PI;
```

这时我发现在 targetY < currentY 时对于由上面公式算得的结果需要加上 180 度。另外还要考虑 atan 不存在的情况，即 currentY == targetY 的时候，这时候可以自己根据定义规定 alpha 是等于 90 度还是-90 度，最后用 alpha-currentBearings 就是最后要求的 bearings，只是要把值限制在 0 到 360 之间，只需加上以下的代码段即可。

```
if(ans < 0) {
    ans += 360;
}else if(ans > 360) {
    ans -= 360;
}
```

到这里为止，我们其实实现了 calculateBearingToPoint 方法，计算出给定 currentBearings 和两个点的坐标时的 bearings，下面我们要基于这个方法把对 bearings 的计算拓展到 List 上。

```
double currentBearing = 0;

for (int i=0; i<xCoords.size()-1; i++) {
    angles.add(calculateBearingToPoint(currentBearing, xCoords.get(i), yCoords.get(i), xCoords.get(i+1), yCoords.get(i+1)));
    currentBearing = Math.atan((double)(xCoords.get(i+1)-xCoords.get(i)/(yCoords.get(i+1)-yCoords.get(i))))*180/Math.PI;
    if(yCoords.get(i+1)<yCoords.get(i)) currentBearing += 180;
}
```

在调用 calculateBearingToPoint 方法时，我们要明确五个参数的对应关系，并且在每次循环中需要不断更新 currentBearings，如上图所示，是根据相邻两点的位置关系对 currentBearings 进行更新，并对特殊情况进行处理。

3.2.5 Problem 7: Convex Hulls

这个方法应该是这道题里最复杂的了。这题涉及到的主要算法是一个叫做 gift-wrapping algorithm 的一个凸包算法，在实现这个算法的过程中可能要调用前面我们编写的那些方法。

我的总体思路是：首先找到一个位于边界上的点，将它放入集合；然后再找到与它相邻的、也位于边界上的一个点，也把它放入集合；直至找到的点已经存在于集合中，这表示我们已经通过集合中的点把图上的所有点“包围”了起来，即这个点集就是我们要找的凸包。

下面是具体代码的实现思路。

```
Point left = new Point(p[0].x(),p[0].y());
int index = 0;

for(int i=1;i<p.length;i++) {
    if(p[i].x()<left.x()) {
        left = p[i];
        index = i;
    }
}
```

首先通过上述代码寻找位于最左边的点，这个点一定是在边界上的，也一定是凸包上的点，我们先把它加入集合中。

```
tempBearing = calculateBearingToPoint(currentBearing, (int)current.x(), (int)current.y(), (int)p[j].x(), (int)p[j].y());
tempDistance = Math.sqrt(Math.pow(p[j].x()-current.x(),2)+Math.pow(p[j].y()-current.y(),2));
if(tempBearing < angle) {
    angle = tempBearing;
    k = j;
    distance = tempDistance;
}else if(tempBearing == angle && tempDistance > distance) {
    distance = tempDistance;
    k = j;
}
```

这部分其实考虑的是在现有的点集上怎么选取下一个边界上的点加入点集。我首先将原来的 Set 类型的 points 转化为了 Point 类型的数组 p，这样比较方便循环及后续操作。我们主要针对两个指标——一是偏转角，偏转角越小说明这个点越靠近边界；而是与当前点的距离，这里主要考虑的是多个点位于同一条直线上的情况，这时应该选取与当前点距离最远的点放入集合。

后续其实就是更新 currentBearings 以及标记当前点已经加入集合。由于每次考虑的都是针对当前点的情况，所以这些实现都是在一个循环里的，循环最多执行 N 次，N 是点的数量，因为最多 N 个点都在边界上。

3.2.6 Problem 8: Personal art

这部分是比较开放的，我选择是是画一个正多角星。核心是计算出每次的偏转角。这个其实是有相关的数学公式进行计算的。

```
public static double calculateRegularStarAngle(int sides) {
    if(sides % 2 == 1 && sides >= 5) {
        return (double)180/sides;
    }else if(sides % 2 == 0 && sides >= 8) {
        return (double)360/sides;
    }else {
        System.out.println("不存在当前角数的正多角星");
        return -1;
    }
}
```

上面的这个方法就实现了多角星每个内角的角度计算。最后用 180 减去这个角度就是每次的偏转角。

3.2.7 Submitting

如何通过 Git 提交当前版本到 GitHub 上你的 Lab1 仓库。

我先将我在 GitHub 上的 Lab1 仓库 clone 到本地，然后将这个项目的当前版本复制到该文件夹。在该文件夹中打开 Git Bash，依次执行 git add .，git commit -m “Turtle Graphics”，git push origin master 即可将当前版本提交到 Lab1 仓库。前提是必须配置好 SSH Key 等信息。

3.3 Social Network

这道题其实主要是构建两个类，一个 Person 类，一个 FriendshipGraph 类。Person 类主要保存人的信息，而在 FriendshipGraph 这个类中，构建了一个以人为节点的图。然后便是在图上进行的操作，诸如：插入节点，插入边，计算两个节点间距离等等。

3.3.1 设计/实现 FriendshipGraph 类

在这个类中，我们要构建一个以 Person 作为节点的图，然后对这个图实现添加节点、添加边、计算两点间的距离等操作。

如果直接把 Person 看做图的节点可能有点复杂，我的想法是对于每个输入的 name，按输入次序得到一个唯一的索引，即顶点的编号。然后其实就转化为了一个普通的无向图，可以用一个邻接表（可变长的二维矩阵）进行存储。后面的添加顶点和添加边其实都比较容易，而计算顶点间的距离可以用 BFS 进行实现。

而针对一些特殊的输入，可以在代码里加入一些特殊处理。例如题目要求不能出现同名的两个人，我们可以将所有名字保存到一个集合里，然后借助集合进行去重。

3.3.2 设计/实现 Person 类

Person 类最基本的属性是姓名，最基本的功能是获取 Person 对象的姓名。这些都非常容易实现。除此之外，由于我在把 Person 加入图的时候给了一个唯一的顶点编号，我也把这个属性写入了 Person 类，同时还有 getIndex 和 setIndex 方法。

3.3.3 设计/实现客户端代码 main()

main()其实很简单，只要在 main()中调用之前编写的那些方法即可。我直接把实验指导书上的测试代码作为 main()的代码，也能实现测试方法的功能。

3.3.4 设计/实现测试用例

给出你的设计和实现思路/过程/结果。

在使用 JUnit 进行单元测试时，我们要对每个方法编写测试样例。对于 addVertex 的测试非常简单，在这里不再赘述。对于 addEdge 方法，我设计了一个辅助函数 isDirectlyConnected，判断两个顶点是否直接相连。在对 addEdge 方法进行测试时，借助这个辅助函数进行测试。对于添加边的两个顶点，辅助函数应该返回 true，否则应该返回 false，这也是 assertEquals 里的内容。

对于 getDistance 方法，可以先自己创建一个无向图，然后人工计算出某几对顶点间的距离，然后在测试时依据这个距离对 getDistance 进行测试。

4 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。

日期	时间段	任务	实际完成情况
2021-04-30	10:00-11:40	编写问题 1 的 isLegalMagicSquare 函数并进行测试	基本完成（可能存在漏洞）
2021-05-01	13:30-16:00	编写问题 2 的 Problem1-6，并上传至 Github	按计划完成
2021-05-02	14:00-17:30	编写问题 2 的 Problem7-8，编写问题 3 的主要功能代码，并上传至 Github	基本按计划完成，问题 3 的 JUnit 测试暂未完成
2021-05-03	14:00-16:00	完成问题 3 的 JUint 测试，完善问题 1isLegalMagicSquare 函数	按计划完成

5 实验过程中遇到的困难与解决途径

遇到的困难	解决途径
在问题 2 中，最后求解“凸包”的算法比较陌生，前面几个辅	通过在网上进行相关算法的搜索，我找到了 gift-wrapping algorithm 的主要思想，了解到这个算法的大致思路如下：先

助求解偏转角的算法也不是很容易实现。	找一个边界上的点，放入一个点集，然后寻找下一个在最“外围”的点（偏转角最小），加入点集，直至加入点集后出现重复的点，原来的点集即为“凸包”。
实验要求需要对 eclipse 构建的工程重新进行 build，我在使用 Travis-CI 进行在线 build 的时候遇到了困难。	通过参考一些 csdn 的博客，我成功的实现了在线 build。首先要把 eclipse 的工程转化为 Maven 工程，然后在文件中添加一个.travis.yml 文件，这里保存的实际上是 build 时的一些配置信息（例如 jdk 版本等）。然后在 Travis-CI 上实现与 GitHub 的关联便可以实现在线 build。

6 实验过程中收获的经验、教训、感想

6.1 实验过程中收获的经验和教训

通过这次实验，我熟悉了 Java 的基本编程语法，也熟悉了 eclipse、git 等相关工具的使用，Java 的编程能力得到了锻炼。

6.2 针对以下方面的感受

(1) Java 编程语言是否对你的口味？

我觉得 java 还是和 C++有相似之处的，无论是语法还是类的操作等等，而且 java 可能还有比 C++更丰富的库，调用起来可能会方便一些。我个人还是挺喜欢用 java 编程的，尽管还不是特别熟悉。

(2) 关于 Eclipse IDE；

感觉这个 IDE 在自动补全方面做的不太好，在对报错的提示和修改方面做的不错。

(3) 关于 Git 和 GitHub；

这两个都是非常好用的项目管理工具。

(4) 关于 CMU 和 MIT 的作业；

我还挺喜欢国外大学的作业风格的，尤其喜欢 MIT 这种循序渐进的题目。

(5) 关于本实验的工作量、难度、deadline；

其实有一部分工作量是花在环境的搭建与配置上的。真正花在代码上的工作量没有想象中那么大，三道题目的难度比较适中，能较为有效地锻炼编程能力。对 deadline，我觉得时间是很充裕的，因为我在“五一”假期就基本完成了实验代码的编写，所以相对而言这次实验的时间比较充裕。

(6) 关于初接触“软件构造”课程；

还挺喜欢这门课的，感觉已经很长时间没有一门以编程为主的课程了。