



哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2020 年春季学期
计算机学院 《软件构造》 课程

Lab 1 实验报告

姓名	郭茁宁
学号	1183710109
班号	1837101
电子邮件	847738428@qq.com
手机号码	13905082373

0 目录

0 目录	2
1 实验目标概述	1
2 实验环境配置	2
2.1 Java & Eclipse & Maven 使用配置方法	2
2.1.1 下载 Eclipse	2
2.1.2 下载 JDK13	2
2.1.3 配置 JDK	2
2.1.4 编辑器字体修改	2
2.1.5 修改默认编码	2
2.1.6 设置代码自动提示	3
2.1.7 配置 Maven	3
2.2 JUnit	4
2.2.1 新建 Java 项目	4
2.2.2 编写代码 & 打包	5
2.2.3 将 JUnit 加入 Library	5
2.2.4 创建 Junit Test Case	6
2.2.5 查看测试结果	7
3 实验过程	8
3.1 Magic Squares	8
3.1.1 isLegalMagicSquare()	8
3.1.2 generateMagicSquare()	9
3.2 Turtle Graphics	10
3.2.1 Problem 1: Clone and import	10
3.2.2 Problem 3: Turtle graphics and drawSquare	11
3.2.3 Problem 5: Drawing polygons	11
3.2.4 Problem 6: Calculating Bearings	12
3.2.5 Problem 7: Convex Hulls	13
3.2.5.1 凸包问题	13
3.2.5.2 算法描述	14
3.2.5.3 JUnit 测试结果	14
3.2.6 Problem 8: Personal art	14
3.2.6.1 多彩螺旋线	14

3.2.6.2	代码.....	15
3.2.6.3	效果.....	16
3.2.7	Submitting	16
3.2.7.1	初始化 git.....	16
3.2.7.2	添加远程仓库 URL	17
3.2.7.3	添加上传文件	17
3.2.7.4	添加修改日志	17
3.2.7.5	上传 push.....	18
3.2.7.6	查看上传情况	18
3.2.7.7	下载/同步.....	18
3.3	Social Network.....	18
3.3.1	设计/实现 FriendshipGraph 类	19
3.3.1.1	邻接表存储结构.....	19
3.3.1.2	Class Node.....	20
3.3.1.3	Method addVertex().....	21
3.3.1.4	Method addEdge().....	21
3.3.1.5	Method getDistance().....	22
3.3.2	设计/实现 Person 类	22
3.3.3	设计/实现客户端代码 main()	23
3.3.3.1	重复名字错误测试	23
3.3.3.2	简单图测试	23
3.3.3.3	复杂图测试	24
3.3.4	设计/实现测试用例	25
3.3.4.1	重复名字错误测试	25
3.3.4.2	简单图测试	26
3.3.4.3	复杂图测试	26
3.3.4.4	Junit 测试结果	26
4	实验进度记录.....	27
5	实验过程中遇到的困难与解决途径	28
6	实验过程中收获的经验、教训、感想	30
6.1	实验过程中收获的经验教训	30
6.2	针对以下方面的感受.....	30

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

2.1 Java & Eclipse & Maven 使用配置方法

2.1.1 下载 Eclipse

Download 64 bit

Select Another Mirror

China - University of Science and Technology of China

作者选择中科大 USTC 镜像

2.1.2 下载 JDK13

- 点击 Accept License Agreement 后, 点击灰条选项

Java SE Development Kit 13.0.2		
You must accept the Oracle Technology Network License Agreement for Oracle Java SE to download this software.		
<input type="radio"/> Accept License Agreement <input checked="" type="radio"/> Decline License Agreement		
Product / File Description	File Size	Download
Linux	155.72 MB	jdk-13.0.2_linux-x64_bin.deb
Linux	162.66 MB	jdk-13.0.2_linux-x64_bin.rpm
Linux	179.41 MB	jdk-13.0.2_linux-x64_bin.tar.gz
macOS	173.3 MB	jdk-13.0.2_osx-x64_bin.dmg
macOS	173.7 MB	jdk-13.0.2_osx-x64_bin.tar.gz
Windows	159.83 MB	jdk-13.0.2_windows-x64_bin.exe
Windows	178.99 MB	jdk-13.0.2_windows-x64_bin.zip

2.1.3 配置 JDK

- 点击 : Window -> Preferences -> Java -> Installed JREs -> Add -> Standard VM -> 选择 JDK 目录 D:\Program Files\jdk-13.0.2 -> Finish -> 在列表勾选刚刚添加的 JDK -> Apply

2.1.4 编辑器字体修改

- 点击: Window -> Preferences -> General -> Appearance -> Colors and Fonts -> Java -> Java Editor Text Font
- 推荐: Consolas+小四号字体

2.1.5 修改默认编码

- 点击 : Windows -> Preferences -> General -> Workspace , 在右侧的 Text file encoding 中选择 UTF-8
- 点击 : Windows -> Preferences -> General -> Content Types , 在右侧点开 Text , 选择 Java Source File , Java Properties File , JSP , 在下面的 Default encoding 输入 UTF-8 , 点击 Update
- 点击 : Windows -> Preferences -> Web -> JSP Files , 面板选择 ISO 10646/Unicode(UTF-8) 右键选择项目 -> Properties -> Resource -> 设

置编码为 UTF-8

2.1.6 设置代码自动提示

- 点击 Window -> Preferences -> Java -> Editor -> Content Assist , 在右侧的 Auto Activation 选项的 Auto activation triggers for Java 中输入 .abcdefghijklmnopqrstuvwxyz 即可

2.1.7 配置 Maven

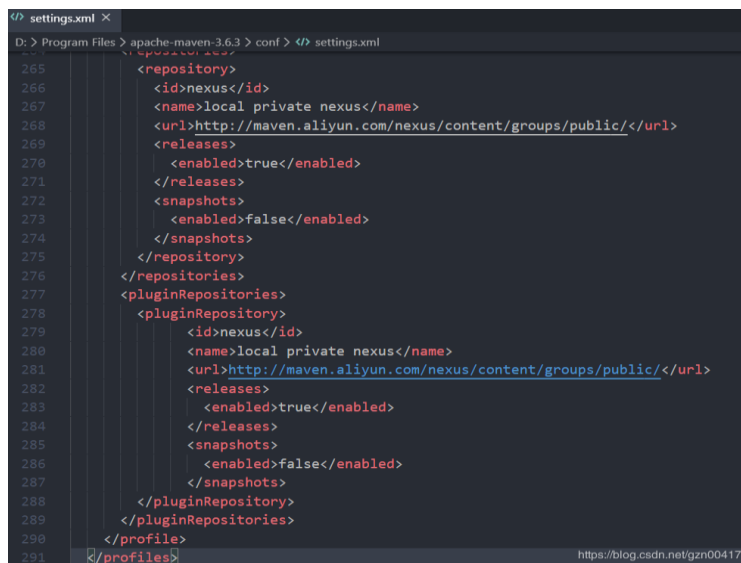
- 首先配置系统环境变量 :
- 右键计算机 -> 属性 -> 高级系统设置 -> 高级 -> 环境变量 , 在下方的系统变量 添加 MAVENHOME , 值是 Maven 文件夹路径 , 如 : C:\MyProgram\Maven , 修改 Path 变量 , 添加 %MAVENHOME%\bin; , 注意分号 ; 配置完成之后查看是否成功 : 运行 CMD 输入 mvn -v

```
C:\Users\guozn>mvn -v
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: D:\Program Files\apache-maven-3.6.3\bin\..
Java version: 13.0.2, vendor: Oracle Corporation, runtime: D:\Program Files\jdk-13.0.2
Default locale: zh_CN, platform encoding: GBK
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

- 然后配置 Eclipse :
- 打开 Eclipse -> Window -> Preferences -> Maven -> User Settings -> Global Settings: 和 User Settings: -> Browse... , 选择 Maven 目录下的 \conf\settings.xml 文件 , 然后点击 Update Settings 按钮即可
- 添加镜像 (国内阿里云镜像)
- conf/setting.xml 中插入代码
- 在 mirrors 中间插入

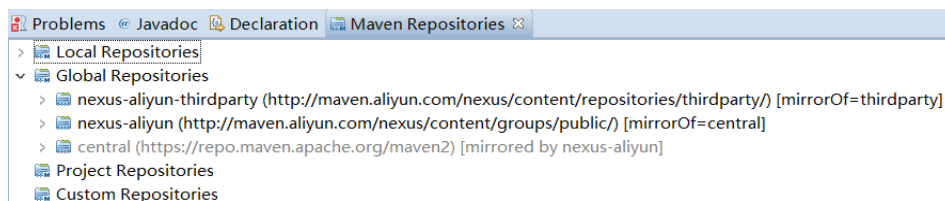
```
<mirror>
  <id>nexus-aliyun</id>
  <mirrorOf>central</mirrorOf>
  <name>Nexus aliyun</name>
  <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
</mirror>
<mirror>
  <id>nexus-aliyun-thirdparty</id>
  <mirrorOf>thirdparty</mirrorOf>
  <name>Nexus aliyun thirdparty</name>
  <url>http://maven.aliyun.com/nexus/content/repositories/thirdpa
rty/</url>
</mirror>
```

- 在 profiles 中间插入



```
265 <repository>
266   <id>nexus</id>
267   <name>local private nexus</name>
268   <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
269   <releases>
270     <enabled>true</enabled>
271   </releases>
272   <snapshots>
273     <enabled>false</enabled>
274   </snapshots>
275 </repository>
276 </repositories>
277 <pluginRepositories>
278   <pluginRepository>
279     <id>nexus</id>
280     <name>local private nexus</name>
281     <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
282     <releases>
283       <enabled>true</enabled>
284     </releases>
285     <snapshots>
286       <enabled>false</enabled>
287     </snapshots>
288   </pluginRepository>
289 </pluginRepositories>
290 </profile>
291 </profiles>
```

- 查看是否成功
 - Window -> Show View -> Other -> Maven -> Maven Repositories -> Open
 - 若打开后更改, 可以点击 Maven Repositories 查看视图右边的 Reload setting.xml



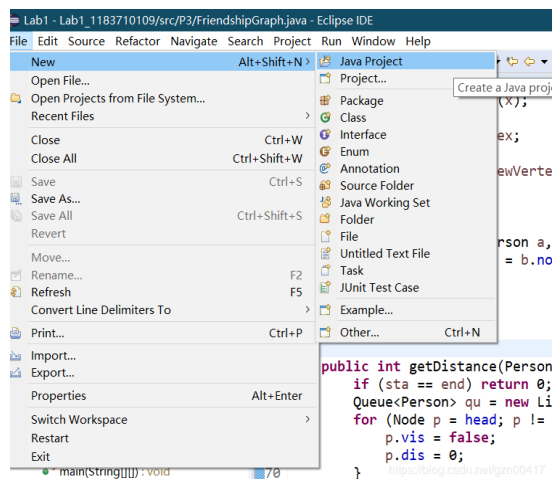
2.2 JUnit

- JUnit is a simple framework to write repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks.
- 使用 JUnit 为 Java 程序编写测试代码并执行测试

2.2.1 新建 Java 项目

- File -> New -> Java Project

- 根据自己的需求配置

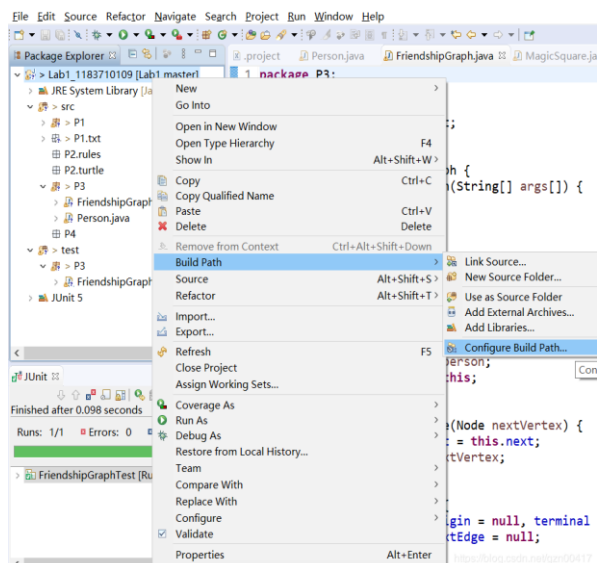


2.2.2 编写代码 & 打包

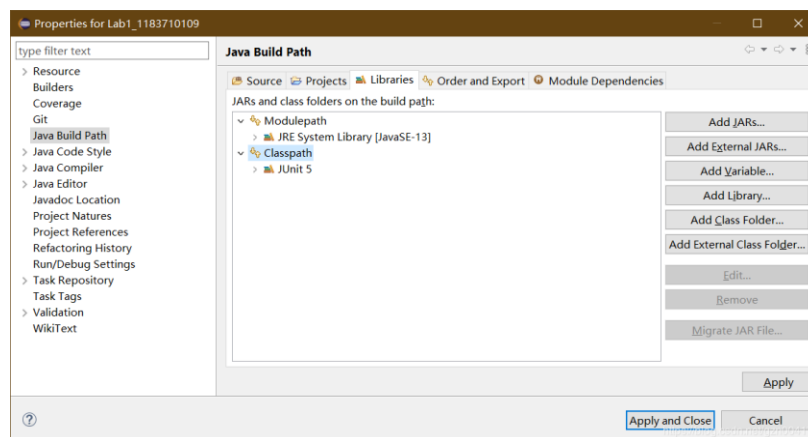
- 准备好 src 文件夹和 test 文件夹
- src 文件夹中
- package 名与 test 中相同（必须）

2.2.3 将 JUnit 加入 Library

- 项目名字（右键）-> Build Path -> Configure Build Path

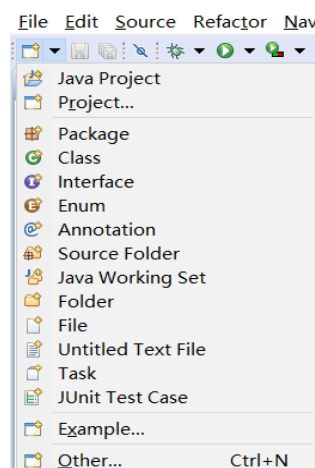


- Library（中间上面）-> classpath（左边）-> Add Libraries（右边）

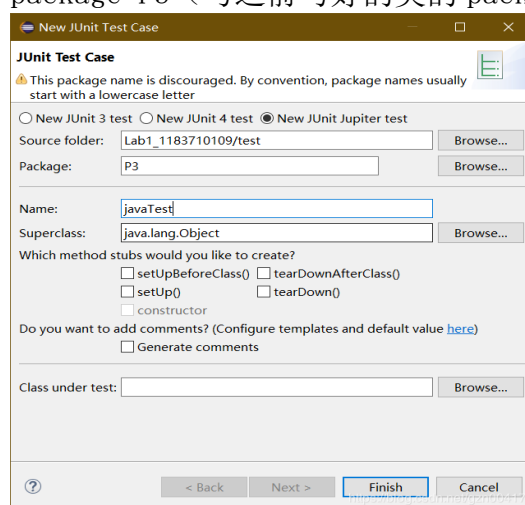


2.2.4 创建 Junit Test Case

- 和创建 project 类似的方法创建 JUnit Test Case



- 弹框选项
 - New Junit Jupiter Test
 - Source folder: ../../test
 - package P3 (与之前写好的类的 package 一样)



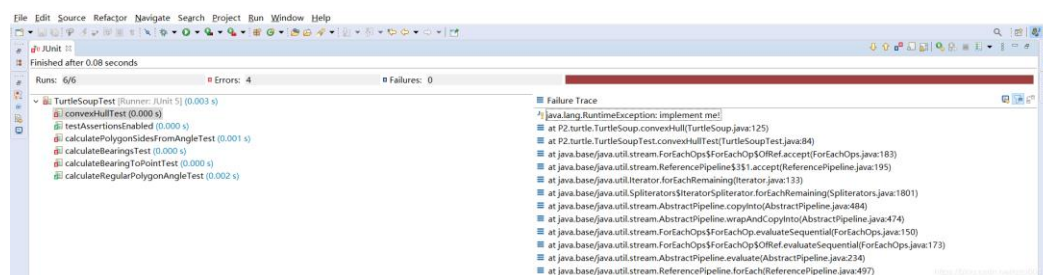
- 创建好的示例:

```
package P3;
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;
class Test1 {
    @Test
    void test() {
        assertEquals(2.0, Calc(1.0, 1.0), 0);
        fail("Not yet implemented");
    }
}
```

- 更改 void test() 中的内容即可!

2.2.5 查看测试结果

- 双击 JUnit 视图放大查看
- 每一个 @Test 为一组测试
- 每组测试会显示测试结果, 正确为绿色, 错误为红色, 还有其他错误提示



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（但无需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

幻方是一个有 $n \times n$ 个不同数字、且每行、每列和斜线上都有相同的和的方形结构。要求写出程序判断输入一个矩阵是否是幻方，并且构造幻方。

Main 函数有两个部分，分别是：读取 5 个矩阵并判断、生成一个矩阵判断后输出到文件中。在读取时，用 for 循环分别将字符 1 到 5 拼入地址中，输出同理。其中在生成幻方之前，判断 n 的合法性。

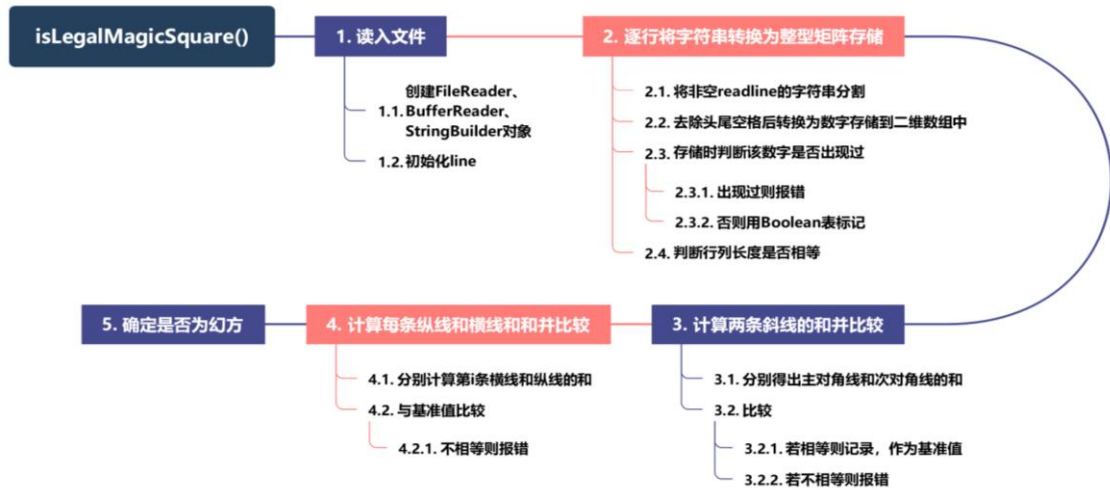
P1.MagicSquare	
△ N: int	
◦ square: int[][]	
◦ vis: boolean[]	
◦ main(args: String[]): void	
◦ isLegalMagicSquare(fileName: String): boolean	
◦ generateMagicSquare(n: int): boolean	

3.1.1 isLegalMagicSquare()

该函数要实现判断一个矩阵是否为幻方。

1. 读入文件
 - a) 创建 FileReader、BufferedReader、StringBuilder 对象
 - b) 初始化 line
2. 逐行将字符串转换为整型矩阵存储
 - a) 将非空 readline 的字符串分割
 - b) 去除头尾空格后转换为数字存储到二维数组中
 - c) 存储时判断该数字是否出现过
 - i. 出现过则报错
 - ii. 否则用 Boolean 表标记
 - d) 判断行列长度是否相等
3. 计算两条斜线的和并比较
 - a) 分别得出主对角线和次对角线的和
 - b) 比较
 - i. 若相等则记录，作为基准值
 - ii. 若不相等则报错

4. 计算每条纵线和横线之和并比较
 - a) 分别计算第 i 条横线和纵线的和
 - b) 与基准值比较
 - i. 不相等则报错
5. 确定是否为幻方



异常处理示例:

```

1 Non-Square false
2 Number false
3 Diff-Sum false
4 Negative false
5 true
  
```

3.1.2 generateMagicSquare()

该函数要实现生成一个边长为奇数的幻方。

1. 初始化
 - i. 生成空矩阵
 - ii. $Row=0, Col=n/2$
2. 循环 $n*n$ 次填充矩阵
 - i. 将矩阵的 $[row, col]$ 位置填充为 i
 - ii. 在保证坐标在矩阵范围内的情况下使 $Row--$, $Col++$
3. 打开文件, 打印结果



异常处理示例:

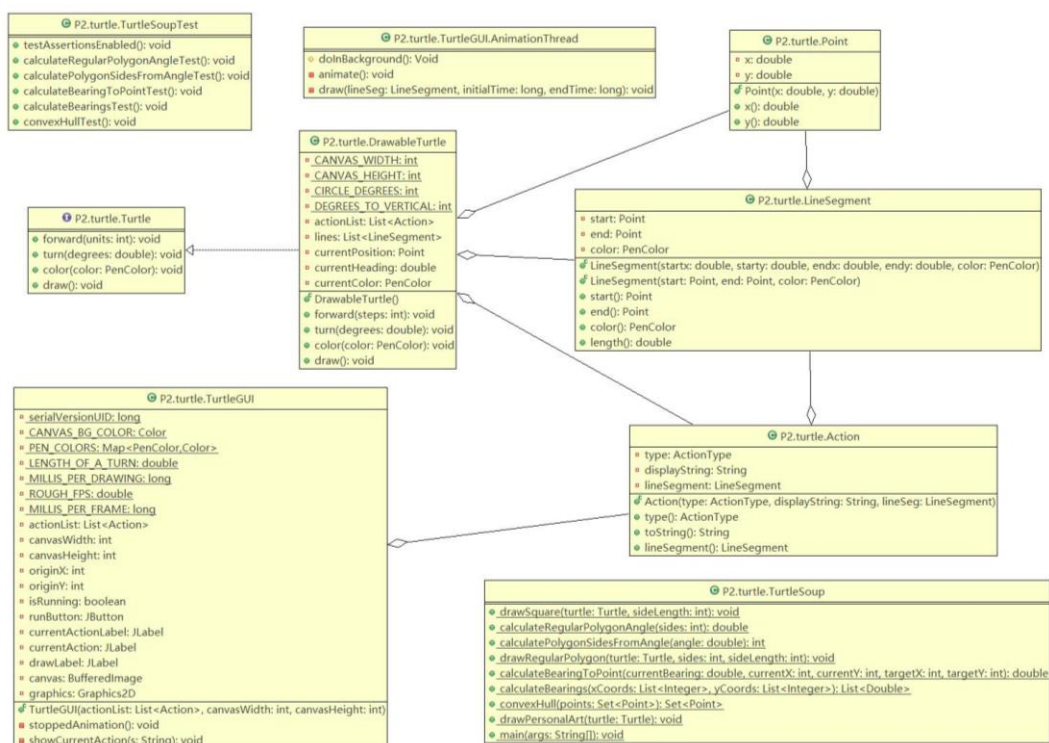
```

8
Input Wrong
10
Input Wrong
-1
Input Wrong
9
6 true

```

3.2 Turtle Graphics

该任务需要我们 clone 已有的程序后, 利用 turtle 按照要求画图, 其中需要利用几何知识设计一些函数简化编程, 最后可以发挥想象力进行 Personal Art。首先分析 turtle 的 package 组成, 了解类成员。



3.2.1 Problem 1: Clone and import

打开目标存储文件夹

右键点击 Git Bash

输入 `git clone https://github.com/ComputerScienceHIT/Lab1-1183710109.git`

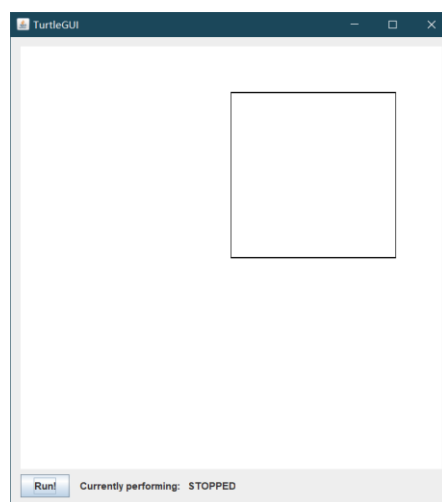
```
guozn@DESKTOP-RQFQUQR MINGW64 ~/Desktop
$ git clone https://github.com/ComputerScienceHIT/Lab1-1183710109.git
Cloning into 'Lab1-1183710109'...
remote: Enumerating objects: 160, done.
remote: Counting objects: 100% (160/160), done.
remote: Compressing objects: 100% (123/123), done.
remote: Total 160 (delta 41), reused 146 (delta 28), pack-reused 0
Receiving objects: 100% (160/160), 3.00 MiB | 18.00 KiB/s, done.
Resolving deltas: 100% (41/41), done.
```

Done

3.2.2 Problem 3: Turtle graphics and drawSquare

该函数需要实现：已知边长，画出边长为指定数值的正方形。参数是海龟对象 `turtle` 和编程 `sidelength`。

首先将海龟画笔设置为黑色。然后执行 4 次的前进 `sidelength` 长度、转完 90 度，即可完成一个边长为 `sidelength` 的正方形。下图是边长为 200 的正方形：

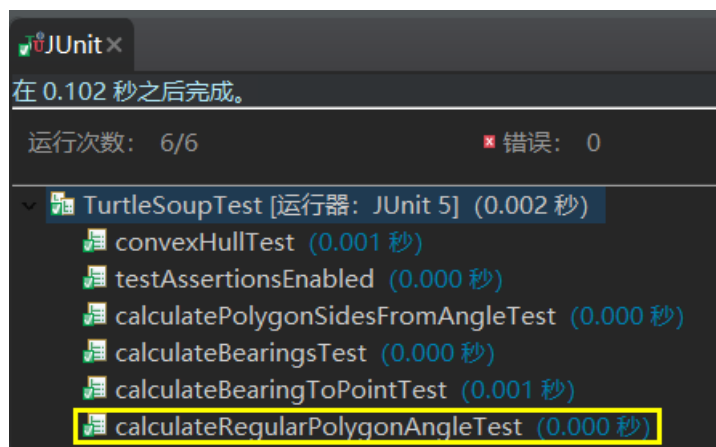


3.2.3 Problem 5: Drawing polygons

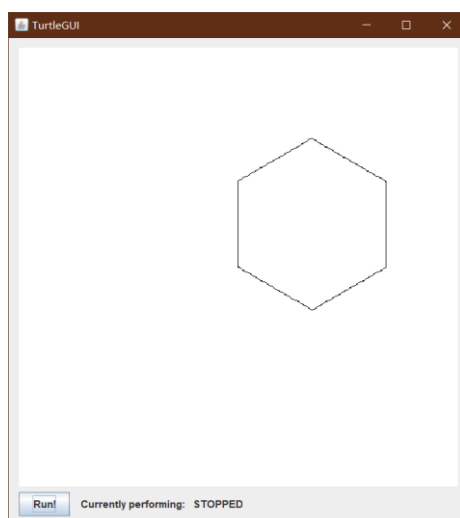
a. 该问题首先希望已知正多边形边数的情况下计算正多边形的内角度。根据几何知识可以推导得公式：

$$(\text{double}) 180.0 - (\text{double}) 360.0 / \text{sides}$$

使用该公式，实现 `calculateRegularPolygonAngle`，通过运行 `TurtleSoupTest` 中的 Junit 测试得：

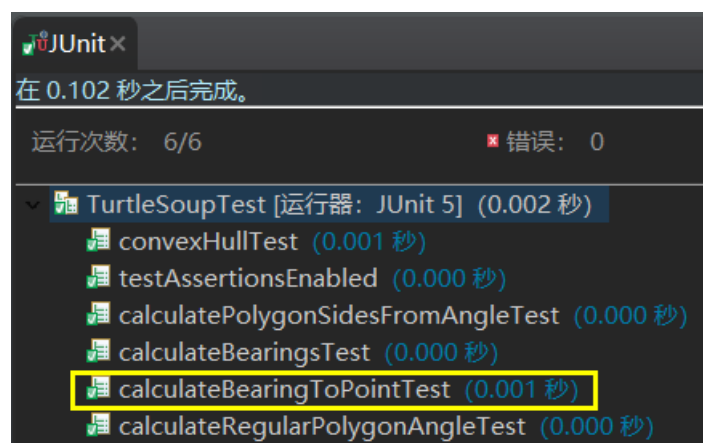


- b. 该问题还希望已知正多边形得边数和边长画出一个正多边形。参照画正方形的方法，可以先前进 `sidelength`，再使海龟旋转一个角度，执行“边数”次。其中，这个角度是正多边形内角的补角，利用 `calculateRegularPolygonAngle` 的功能计算出多边形内角，再用 180° 减去这个值即可。边长为 100 的正六边形效果如下：



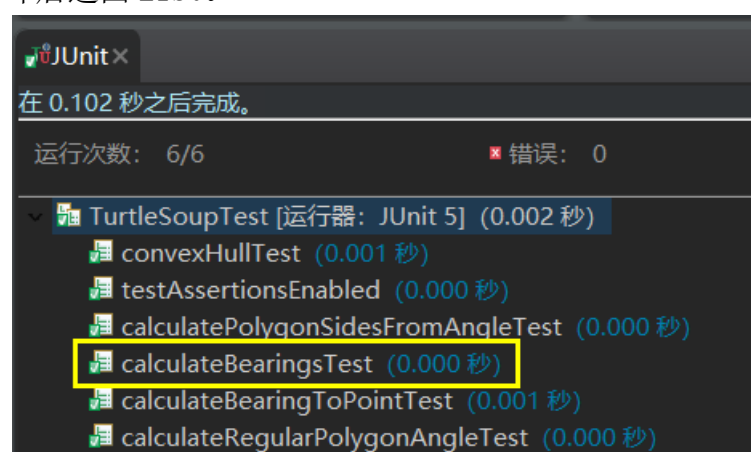
3.2.4 Problem 6: Calculating Bearings

- a. 该问题首先希望解决，已知起点和当前朝向角度，想知道到终点需要转动的角度。例如，如果海龟在 $(0, 1)$ 朝向 30° ，并且必须到达 $(0, 0)$ 它必须再转动 150° 。
1. 首先使用 `Math.atan2` 函数计算两点之间的边在坐标系的角度，减去当前朝向的角度；
 2. 然后取相反数（海龟旋转的方向是顺时针，坐标轴角度的旋转角度的逆时针）；
 3. 再减去 90° （海龟的 0° 线是向上，坐标轴的 0° 线是向右，向右到向上要逆时针旋转 90° ）；
 4. 最后调整为 $0-360^\circ$ 之间（可能大于 360° 或小于 0° ）。



b. 基于上一个问题，此时有若干个点，想知道从第一个点开始到第二个点，再从第二个点到第三个点……以此类推每次转向的角度。

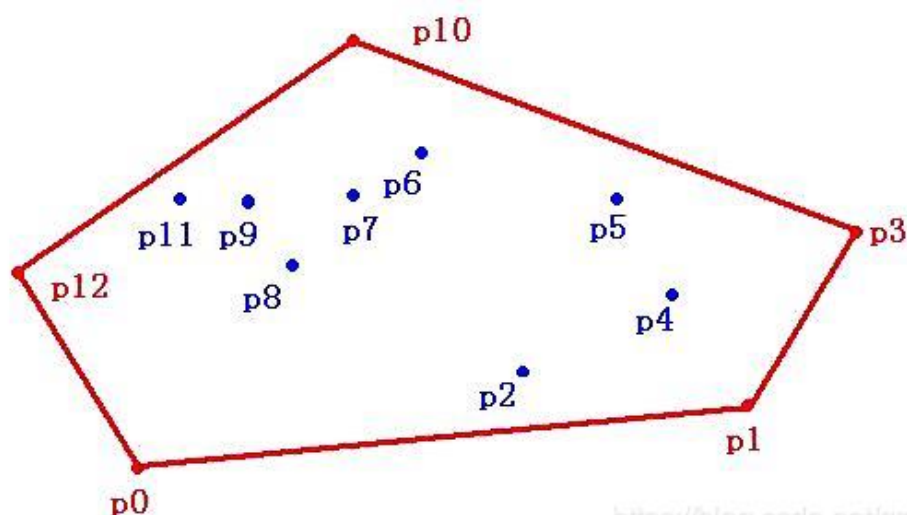
1. 将“起点”选为第一个点（坐标为 `(xCoords.get(0), yCoords.get(0))`）；
2. 循环 $n-1$ 次（ n 为点的个数）
3. 每次将第 $i+1$ 号点设置为“终点”，通过上一个函数计算旋转角度并存储到 List 中；
4. 将下一次的“起点”用当前“终点”更新，继续循环；
5. 退出循环后返回 List。



3.2.5 Problem 7: Convex Hulls

3.2.5.1 凸包问题

给定平面上一堆点集，输出位于凸包上的点。如下所示：输入如下这么多点集，输入 P0, P1, P3, P10, P12。



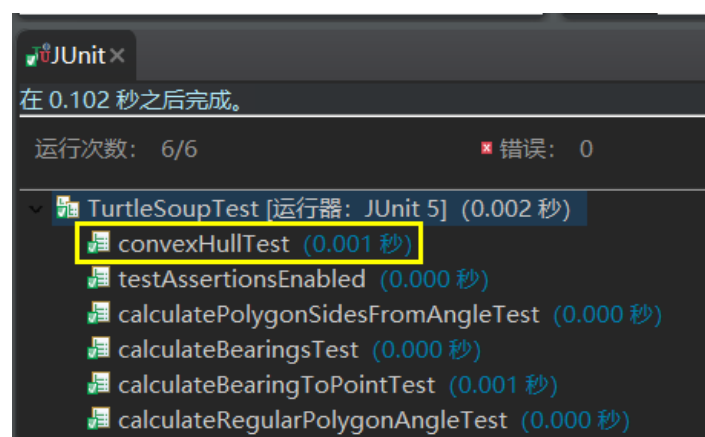
3.2.5.2 算法描述

这里使用 Gift-Wrapping 算法。

我们发现任意凸包上的点，你会发现以该点建立一个极角坐标系，该点连结其它所有点的极角中，该点逆时针方向的第一凸包点到该点极角最小，例如 P0，到所有点的极角中 P0P1 极角最小。

算法中首先找到最左边的点，这个点必然在凸包上，然后计算该点连接点极角最小的，这里计算有技巧，算法中进行试验，直到找到到最右端的点，找到 P1 后，就可以从 P1 开始，接着顺次找到 P2，又以 P2 为起点……

3.2.5.3 JUnit 测试结果



3.2.6 Problem 8: Personal art

3.2.6.1 多彩螺旋线

思路：在画正多边形的基础上，步长不是一直相同，而是越来越长，并且角度比画正多边形需要的角度多一些，每次拐弯变换颜色。

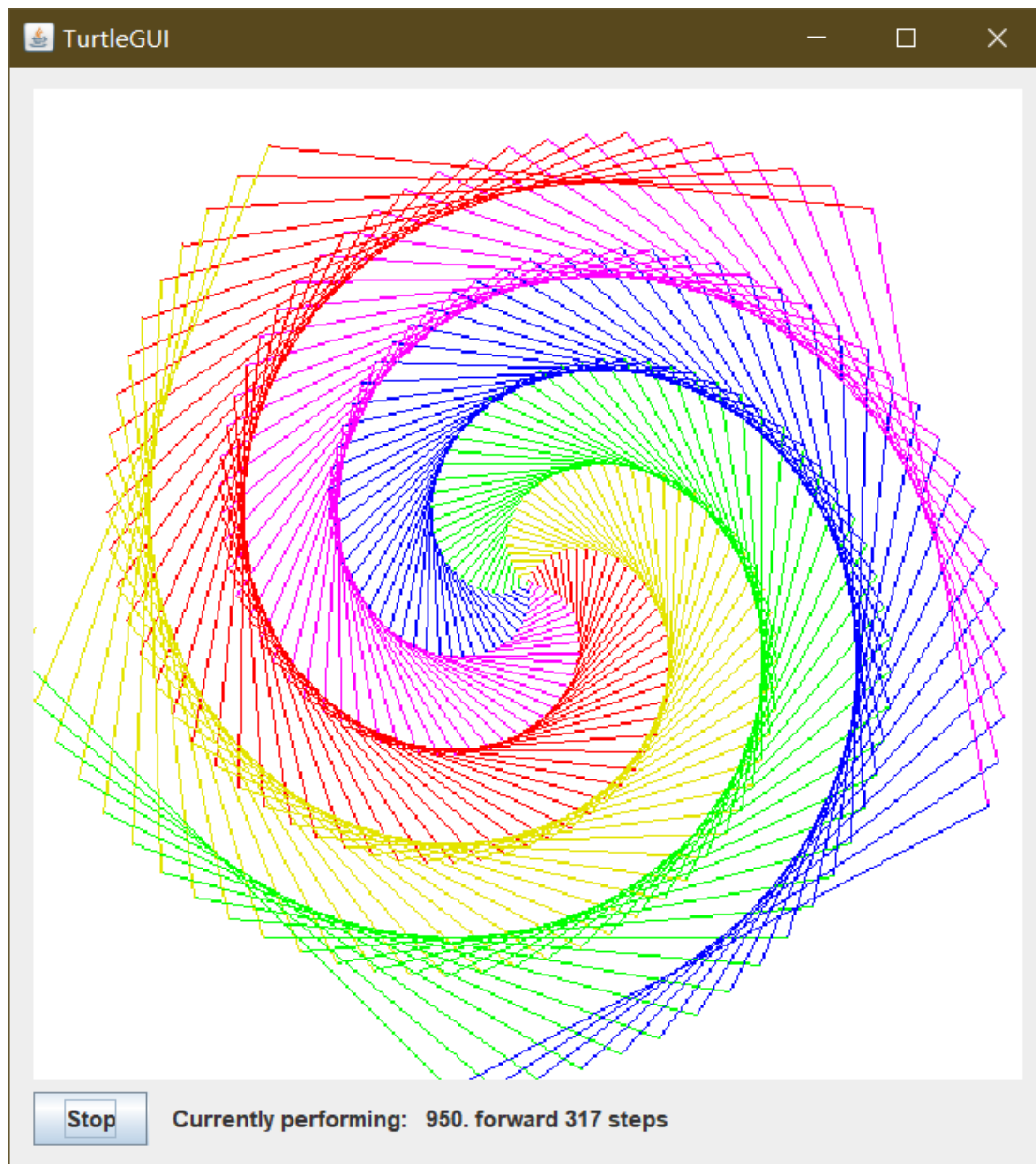
- Size 是螺旋的大小

- Step 的每一步的长度，每走一步拐弯一次
- Densi 是密度，角度越小，螺旋越密
- ColorNum 是色彩的数量，更改时要在 switch 里更改

3.2.6.2 代码

```
public static void drawPersonalArt(Turtle turtle) {  
    int Size = 400, Step = 1, Densi = 1, ColorNum = 5;  
    for (int i = 1; i <= Size; i++) {  
        switch (i % ColorNum) {  
            case 0:  
                turtle.color(PenColor.BLUE);  
                break;  
            case 1:  
                turtle.color(PenColor.GREEN);  
                break;  
            case 2:  
                turtle.color(PenColor.YELLOW);  
                break;  
            case 3:  
                turtle.color(PenColor.RED);  
                break;  
            case 4:  
                turtle.color(PenColor.MAGENTA);  
                break;  
            case 5:  
                turtle.color(PenColor.ORANGE);  
                break;  
        }  
        turtle.forward(Step * i);  
        turtle.turn(360 / ColorNum + Densi);  
    }  
}
```

3.2.6.3 效果



3.2.7 Submitting

通过 git 提交项目

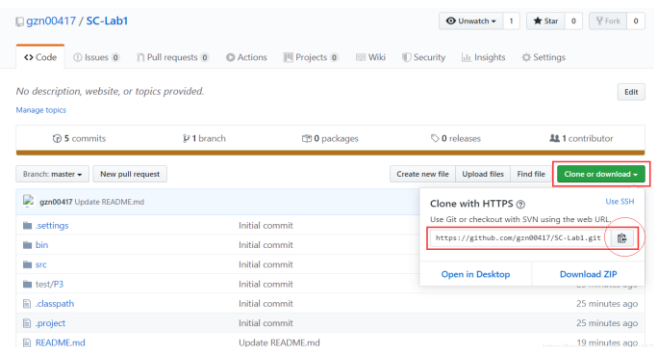
3.2.7.1 初始化 git

打开 Git Bash

输入 `cd /d/.../`

3.2.7.2 添加远程仓库 URL

- 查找 URL，图示位置（绿色按键-->复制 URL）



添加远程仓库链接，命令：`git remote add origin`

`https://github.com/.../x...x.git`

最后的链接就是复制来的 URL

```
guzn@DESKTOP-RQFQUQR MINGW64 /d/GZN/HIT/个人文件/2020春软件构造/2019Labs/Lab1/Lab1_1183710109 (master)
$ git remote add origin https://github.com/gzn00417/SC-Lab1.git
```

3.2.7.3 添加上传文件

- 注意：若新建仓库有 README.md，需要先在本地同步（下载）
- 命令：`git pull`
- 若无效则尝试：`git pull --rebase origin master`

```
guzn@DESKTOP-RQFQUQR MINGW64 /d/GZN/HIT/个人文件/2020春软件构造/2019Labs/Lab1/Lab1_1183710109 (master)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
Unpacking objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
From https://github.com/gzn00417/SC-Lab1
   757a550..b2b52ea  master    -> origin/master
Updating 757a550..b2b52ea
Fast-forward
 README.md | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

- 命令：`git add .`（后面是前后有空格的点，表示所有文件）
- 命令：`git status` 可以查看状态

```
guzn@DESKTOP-RQFQUQR MINGW64 /d/GZN/HIT/个人文件/2020春软件构造/2019Labs/Lab1/Lab1_1183710109 (master)
$ git add .

guzn@DESKTOP-RQFQUQR MINGW64 /d/GZN/HIT/个人文件/2020春软件构造/2019Labs/Lab1/Lab1_1183710109 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   .classpath
    new file:   .project
    new file:   .settings/org.eclipse.jdt.core.prefs
    new file:   bin/.gitignore
```

3.2.7.4 添加修改日志

命令：`git commit -m "Initial commit"`

引号内内容可以随意修改；

引号内的内容只会称为有修改过的文件的新日志；

命令（在-m前加入-a）：`git commit -a -m "Update ..."`

- 任何是否被 `git add` 的文件都将被 `commit`

```
guozn@DESKTOP-RQFQUQR MINGW64 /d/GZN/HIT/个人文件/2020春软件构造/2019Labs/Lab1/Lab1_1183710109 (master)
$ git commit -m "Initial commit"
[master (root-commit) d008f95] Initial commit
30 files changed, 1354 insertions(+)
create mode 100644 .classpath
create mode 100644 .project
create mode 100644 .settings/org.eclipse.jdt.core.prefs
create mode 100644 bin/.gitignore
create mode 100644 src/P1/MagicSquare.class
```

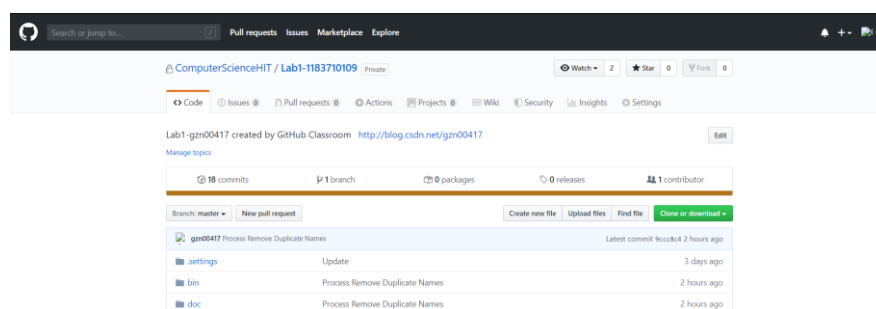
3.2.7.5 上传 push

命令: `git push -u origin master`

多次使用后可以省略后面参数, 只用 `git push`

```
guozn@DESKTOP-RQFQUQR MINGW64 /d/GZN/HIT/个人文件/2020春软件构造/2019Labs/Lab1/Lab1_1183710109 (master)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 305 bytes | 305.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/gzn00417/SC-Lab1.git
373684d..757a550 master -> master
```

3.2.7.6 查看上传情况

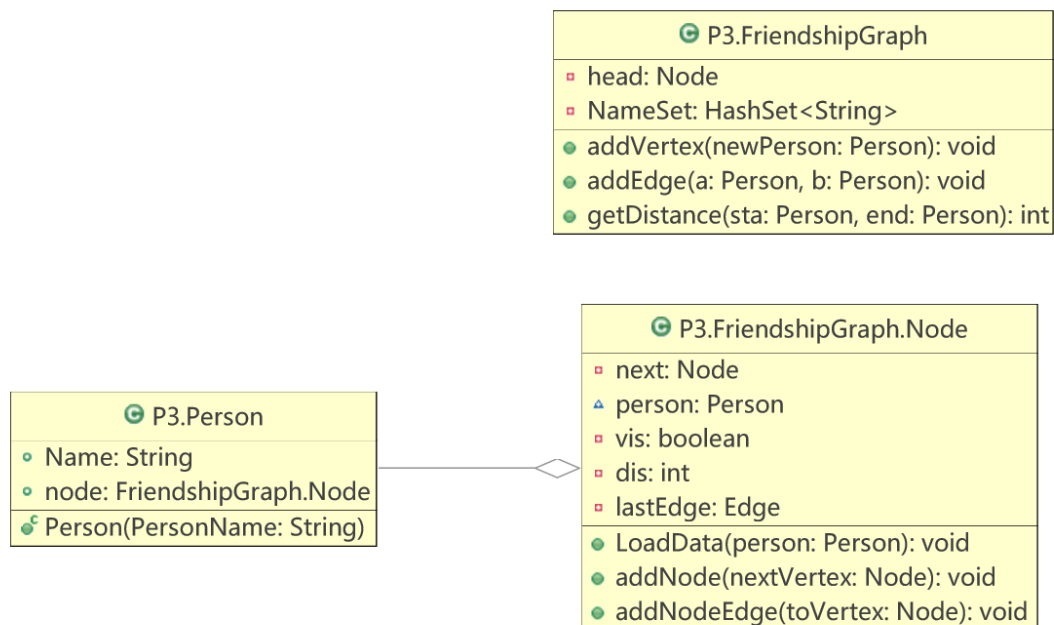


3.2.7.7 下载/同步

- 命令: `git pull` 或 `git pull --rebase origin master`

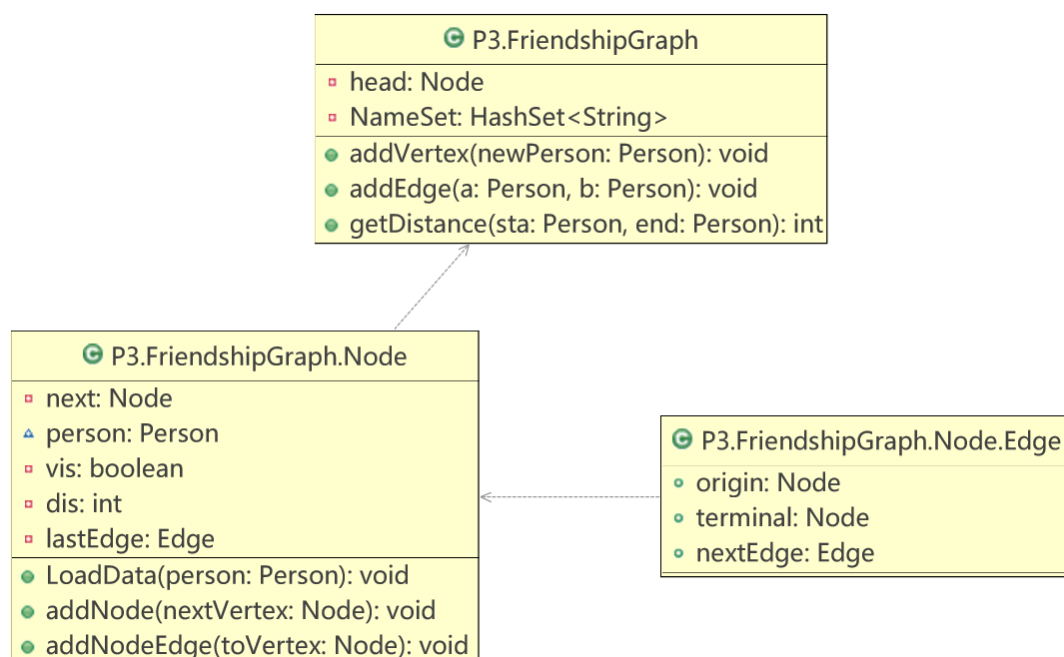
3.3 Social Network

该任务要求设计一张社交网络图, 基于连接人与人, 并且能计算任意两人之间的联系情况。网络图基于两个类, 分别是 `FriendshipGraph` 类和 `Person` 类。



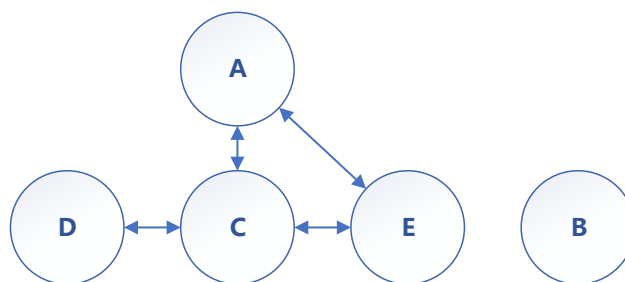
3.3.1 设计/实现 FriendshipGraph 类

该类的实际意义是一张社交网络图，包括了代表每个 Person 的点、代表每两个 Person 之间联系的边、以及建立点和联系和计算距离的方法。

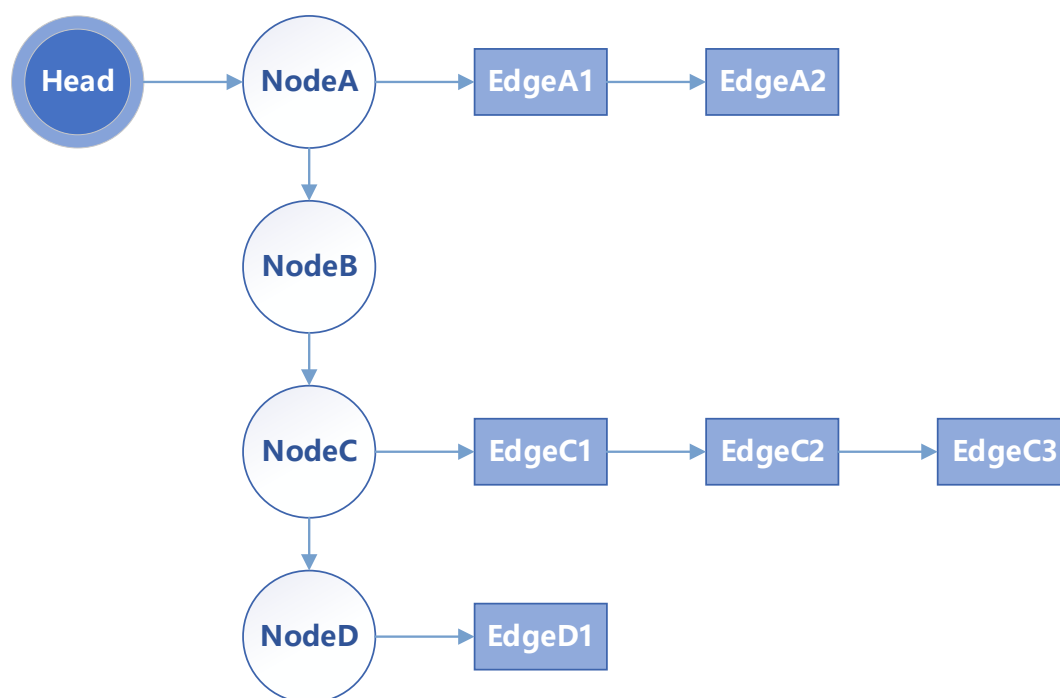


3.3.1.1 邻接表存储结构

在存储社交网络时，我使用了邻接表。所有的 Node 被连在一起，方便查找，并补充了一个 head 变量用来标记首个 Node。假定一个社交网络为

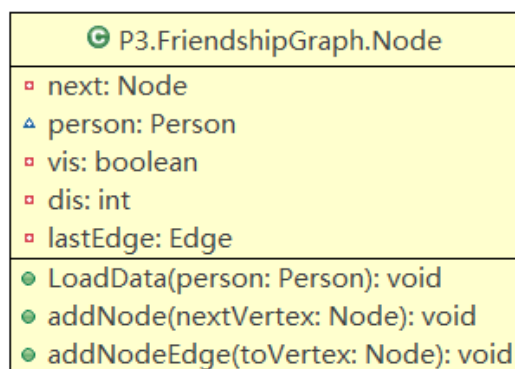


则该图转换为邻接表的示意图为:



3.3.1.2 Class Node

Node 类要实现的是将一个 Person 转换为邻接表里的点，所以一个 Node 有邻接表中点的重要成员变量：下一个 Node 为 next，对应的 Person 对象 person，直接连接的边 lastEdge，以及实现邻接表的相关方法。此外，为了实现方法 getDistance，我另外增设了 vis 和 dis 两个变量用来记录是否访问过以及与当前起点的最近距离。



- *Class Node. Edge*

该类是邻接表中的边，每个 Edge 对象存储了邻接表中的下一条边，以及对

应的边的两个 Person 所对应的 Node。

- **Method Node.LoadData**

该方法将 Person 对象导入到 Node 中进行存储，需要的时候可以直接调用。

- **Method Node.addNode**

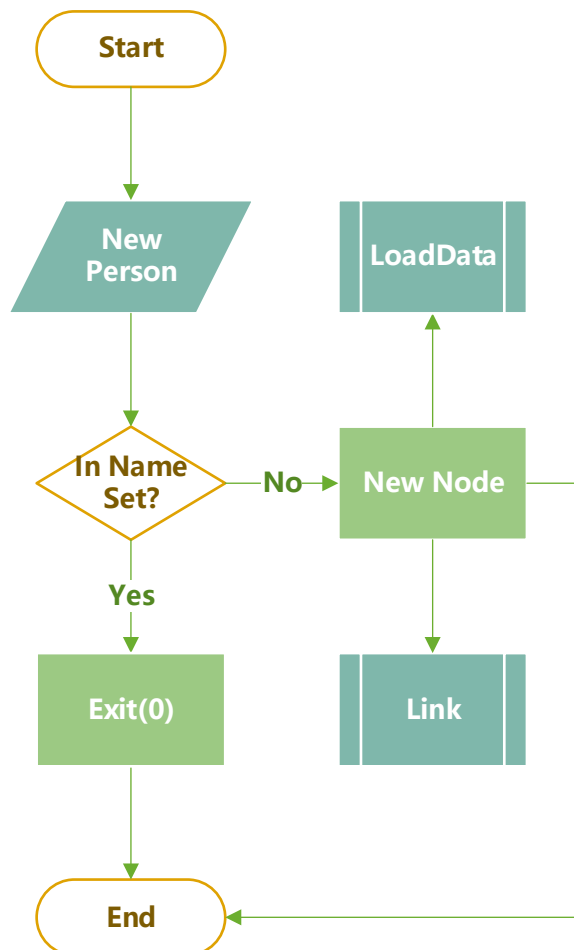
该方法将相应的 Node 加入到 Node 的链表中（即邻接表图中的纵向链表）。

- **Method Node.addNodeEdge**

该方法将新的边加入到对应的 Node 中，更新每个 Node 后的 Edge 链表。

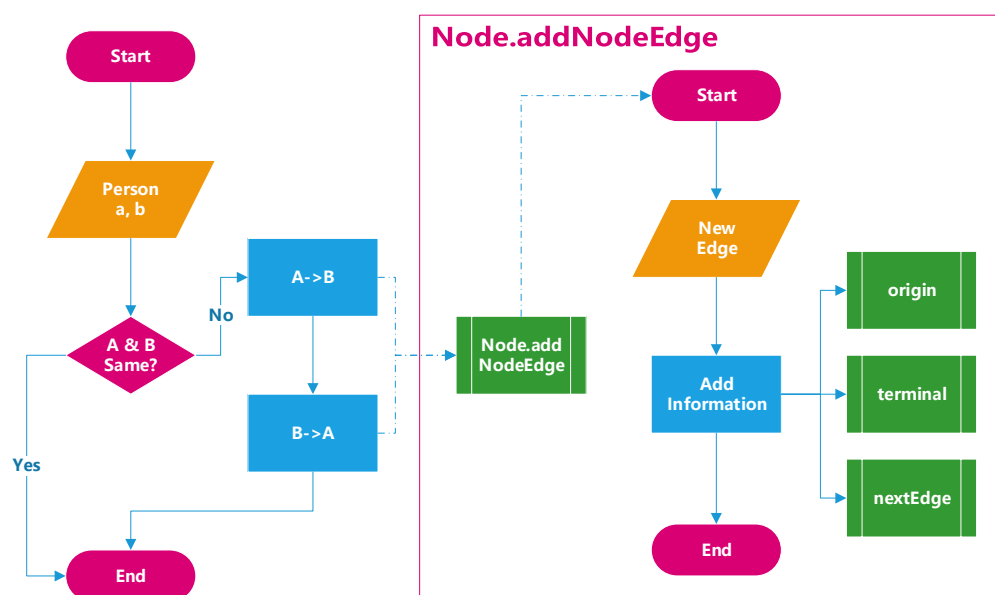
3.3.1.3 Method addVertex()

该方法目标是在社交网络图中增加一个新的节点，参数是要加入的 Person 类。首先，方法要对 Person 的名字进行判重：用哈希集合 HashSet 记录下已加入的所有 Person 的名字，每当新加入一个 Person 则进行判断是否在集合中；然后则新建一个 Node 类，使每一个 Person 与一个 Node 对应起来。



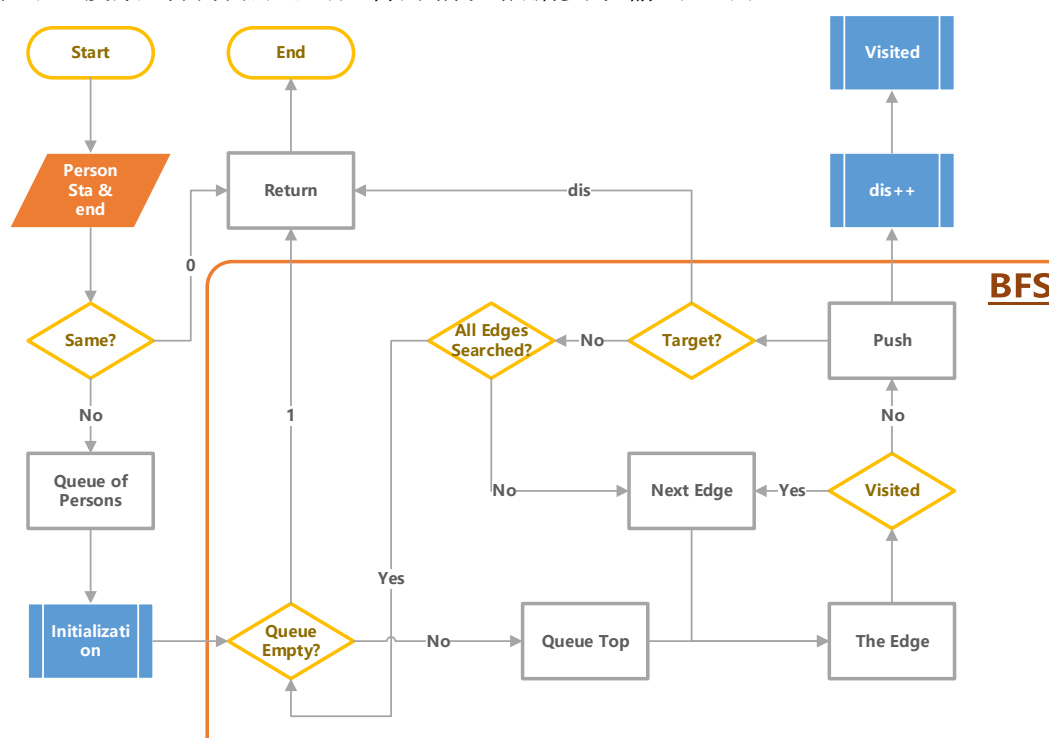
3.3.1.4 Method addEdge()

该方法目标是将两个 Person 之间进行联系，在邻接表中，用有向边来代表“有社交关系”，由于题目设定是社交默认为双向，则需要在函数中两次调用 Node 中的 `addNodeEdge` 方法加两个方向的边。考虑到可扩展性和可复用性，程序考虑到了“单向社交的情况”，仅需将双向加边中的“B→A”删除即可。



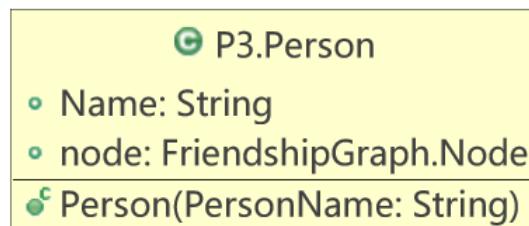
3.3.1.5 Method getDistance()

该方法要计算任意两个 Person 之间的“距离”，若没有任何社交关系则输出“-1”。两个 Person 之间计算使用 BFS，默认边权为 1，则在搜索到边时加 1 即可，搜索到目标点退出；特殊情况根据要求输出 0 或-1。



3.3.2 设计/实现 Person 类

该类的目标是将每一个人对对应到一个 Person 对象，并存储名字的信息。此外，在我的设计中，为了方便起见，我将每个 Person 对象在 FriendshipGraph 中对应的 Node 存储在对应 Person 的成员变量中。



3.3.3 设计/实现客户端代码 main()

3.3.3.1 重复名字错误测试

```

/**
 * TODO to test program exit when graph has the same names
 *
 */
@Test
void ExceptionProcess() {
    FriendshipGraph graph = new FriendshipGraph();

    Person a = new Person("a");
    graph.addVertex(a);

    Person b = new Person("b");
    graph.addVertex(b);

    // Person c = new Person("a");
    // graph.addVertex(c);
}
  
```

3.3.3.2 简单图测试

```

/**
 * Basic Network Test
 *
 */
@Test
void GraphTest1() {
    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));// 1
 * System.out.println(graph.getDistance(rachel, ben));// 2
 * System.out.println(graph.getDistance(rachel, rachel));// 0
 * System.out.println(graph.getDistance(rachel, kramer));// -1
 */
assertEquals(1, graph.getDistance(rachel, ross));
assertEquals(2, graph.getDistance(rachel, ben));
assertEquals(0, graph.getDistance(rachel, rachel));
assertEquals(-1, graph.getDistance(rachel, kramer));
}
```

3.3.3.3 复杂图测试

```
/**
 * Further Test
 */
@Test
void GrpahTest2() {
    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");
    Person h = new Person("H");
    Person i = new Person("I");
    Person j = new Person("J");

    graph.addVertex(a);
    graph.addVertex(b);
```

```
graph.addVertex(c);
graph.addVertex(d);
graph.addVertex(e);
graph.addVertex(f);
graph.addVertex(g);
graph.addVertex(h);
graph.addVertex(i);
graph.addVertex(j);

graph.addEdge(a, b);
graph.addEdge(a, d);
graph.addEdge(b, d);
graph.addEdge(c, d);
graph.addEdge(d, e);
graph.addEdge(c, f);
graph.addEdge(e, g);
graph.addEdge(f, g);
graph.addEdge(h, i);
graph.addEdge(i, j);

assertEquals(2, graph.getDistance(a, e));
assertEquals(1, graph.getDistance(a, d));
assertEquals(3, graph.getDistance(a, g));
assertEquals(3, graph.getDistance(b, f));
assertEquals(2, graph.getDistance(d, f));
assertEquals(2, graph.getDistance(h, j));
assertEquals(0, graph.getDistance(i, i));
assertEquals(-1, graph.getDistance(d, j));
assertEquals(-1, graph.getDistance(c, i));
assertEquals(-1, graph.getDistance(f, h));
}
```

3.3.4 设计/实现测试用例

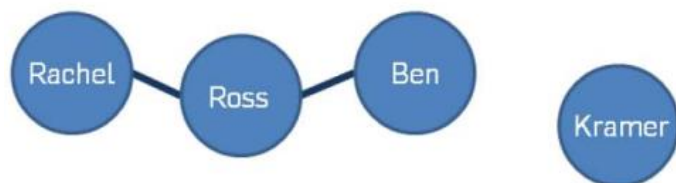
3.3.4.1 重复名字错误测试

测试当有重复名字“a”时，程序是否能终止

```
Person a = new Person("a");
graph.addVertex(a);
Person b = new Person("b");
graph.addVertex(b);
Person c = new Person("a");
graph.addVertex(c);
```

3.3.4.2 简单图测试

根据题目中的社交网络图：

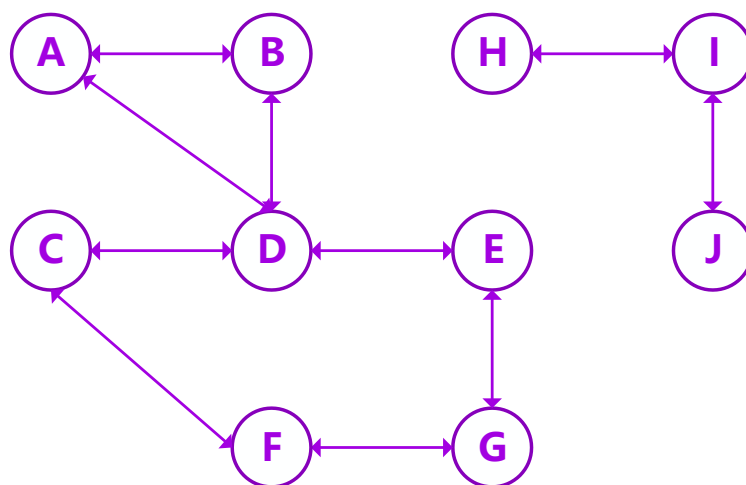


分别测试：

1. Rachel 和 Ross 距离是 1，Rachel 和 Ben 距离是 2
2. Rachel 和 Rachel 距离是 0
3. Rachel 和 Kramer 距离是-1

3.3.4.3 复杂图测试

设计 10 个点、10 条边的社交网络图：



分别测试：

1. AE 距离 2，AD 距离 1，AG 距离 3，BF 距离 3，DF 距离 2，HJ 距离 2
2. II 距离 0
3. DJ 距离-1，CI 距离-1，FH 距离-1

3.3.4.4 Junit 测试结果

全部正确。

```
在 0.169 秒之后完成。
运行次数: 3/3      ✖ 错误: 0      ✖ 故障次数: 0
▼ [Icon] FriendshipGraphTest [运行器: JUnit 5] (0.029 秒)
  [Icon] ExceptionProcess() (0.013 秒)
  [Icon] GrpahTest2() (0.005 秒)
  [Icon] GraphTest1() (0.011 秒)
```

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。

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
2020-02-03	下午	Magicsquare	按计划完成
2020-02-04	下午	建立本地 git 仓库并链接	遇到困难，但按计划完成
2020-02-04	晚上	配置、生成项目	按计划完成
2020-02-05	早上	Turtle 问题第二个小问	按计划完成
2020-02-05	下午	Socialnetwork 的两个类	按计划完成
2020-02-06	晚上	解决 3.2.5	按计划完成
2020-02-08	晚上	TurtleSoupTest 调试	按计划完成，时间较长
2020-02-24	下午	迁移项目，调整工作目录和路径	完成
2020-02-26	上午	写报告，配置环境部分和问题 1 概述	完成
2020-02-26	下午	完成 Personal Draw	完成
2020-02-27	中午	解决 Social Network 重名冲突问题	克服困难完成
2020-02-27	下午	写报告关于 Git Clone/Push 部分	完成
2020-02-28	上午	写报告关于 Turtle 函数实现部分	完成
2020-02-29	早晨	解决 word 标题编号奇葩问题	完成
2020-02-29	上午	Problem1 两个函数设计流程图	完成
2020-02-29	下午	Problem3 函数设计思路	完成
2020-03-01	上午	解决 AmaterasUML 安装问题	遇到困难，解决
2020-03-01	下午	画 UML 图	完成
2020-03-02	下午	FriendshipGraph 详细思路和流程图	完成
2020-03-02	晚上	报告中 Problem3 测试测试部分	完成
2020-03-06	上午	Extra: Update Junit5 Lib	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
Java 使用 Eclipse 出现 editor does not contain a main type	<p>原因: 当前源代码没有整合进 package, 即没有 build path</p> <p>部分已经 build path, 但新写的代码没有解决方案</p> <p>若已经有 build path (有红褐色标记), 则先 Remove; 没有 (全白) 则忽略</p> <p>src (右键) -> build path -> remove from build path</p> <p>此时, src 文件全部变白</p> <p>然后点击: src (右键) -> build path -> use as source folder</p>
Exception in thread "main" java.lang.Error: Unresolved compilation problem, 编译器/package 不一致	<p>检查编译器</p> <p>打开 cmd 检查版本</p> <p>项目名 (右键) -> propriety -> Java Compiler Enable... 打勾, Compiler...level 与 java 和 javac 一致</p> <p>可以把 Use... 勾去掉进行修改</p> <p>若无误, 则非编译器问题</p> <p>发现 package 名字有红线** (重要特征)**</p> <p>发现在 build path 之后不一致, 统一 package 完成 build path 请见解决</p> <p>修改代码中 package 名字** (每一个包中的代码)**</p> <p>再次运行 check</p> <p>注意!! 还需要更改同一个 package 下所有代码的部分 import (旧 package 名)</p>
Exception in thread "main" java.io.FileNotFoundException: xxx.xxx (系统找不到指定的路径。)	<p>该情况为读取文件路径错误</p> <p>src 里的代码, 在我们的直观感觉可以用相对路径 ./xxx</p> <p>然而有时候行, 有时候不行</p> <p>作者确实都经历过</p> <p>因此要加上以 src 为起始的路径</p> <p>例如代码在 src/P1 里, 读取的文件在 src/P1/a 里</p> <p>写 ./a/xxx 要碰运气</p> <p>写 src/P1/a/xxx 才是最稳的</p>

Eclipse 使用 AmaterasUML 自动生成 Java 程序 UML 图 及问题解决

如果 eclipse 是 4.0 以下在安装 AmaterasUML 之前, 须要先安装另外一个插件 GEF

(Graphical Editing Framework) 4.0 之后不要安装已自带。

作者尝试过一些 GEF 下载链接不能使用

可以使用 Eclipse Graphical Editing Framework (GEF)找到下面的链接

截至本文发布可用的链接是

<http://download.eclipse.org/tools/gef/updates/releases>

帮助->安装新软件->添加

选择 GEF common 里的第一个

重启 eclipse

接下来是重点——安装 AmaterasUML

下载链接

GitHub 下载链接

找到 AmaterasUML_1.3.4

下载后有 3 个文件

把它们移动到 Eclipse 安装目录中的 plugins 文件夹中

重启 Eclipse, 点击新建->其他, 找到

AmaterasUML, 选择 Class Diagram, 把

xxx.java 拖入即可

此时, 作者遇到的问题是, 新建里看不到

AmaterasUML

发现此文终于解决, 鸣谢!!!

新建->其他里没有 Amateras 解决方法:

将 Eclipse 安装目录下的

/configuration/org.eclipse.update 删除

原文说要用 cmd 什么的, 可是我 restart 就好了, 我觉得不需要

重启 Eclipse 即可

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教训

该总结的都在博客里，少说有 10 几篇：<https://blog.csdn.net/gzn00417>

6.2 针对以下方面的感受

- (1) Java 编程语言是否对你的口味？
 - a) 符合，Java 的 OOP 非常顺手，又相近于之前熟悉的 C++
- (2) 关于 Eclipse IDE；
 - a) 不够美观
 - b) 经过一定装饰后还是不够美观
 - c) 调试、JUnit 都很好用
 - d) 相对 vscode 太丑
- (3) 关于 Git 和 GitHub；
 - a) 能掌握 Git 基本功能
 - b) GitHub 真香
- (4) 关于 CMU 和 MIT 的作业；
 - a) 语言问题，有些地方理解比较困难
- (5) 关于本实验的工作量、难度、deadline；
 - a) 工作量非常大
 - b) 难度很大
 - c) Deadline 适中
- (6) 关于初接触“软件构造”课程；
 - a) 非常有用
 - b) 这肯定是一门编程课，但是我实在不知道怎么考试
- (7) 疫情期间，只能远程授课，个人在家里完成实验任务，你对该学习方式有什么想法？
 - a) 虽然我学习不好，但对我来说没有太大影响
 - b) 甚至网课还提高了我的学习效率