

**2019年春季学期  
计算机学院《软件构造》课程**

**Lab 2实验报告**

|  |  |
| --- | --- |
| 姓名 | 任永鹏 |
| 学号 | 1173710107 |
| 班号 | 1737101 |
| 电子邮件 | [805402160@qq.com](mailto:805402160@qq.com) |
| 手机号码 | 18800420869 |

**目录**

[1 实验目标概述 1](#_Toc1988214)

[2 实验环境配置 1](#_Toc1988215)

[3 实验过程 1](#_Toc1988216)

[3.1 Poetic Walks 1](#_Toc1988217)

[3.1.1 Get the code and prepare Git repository 1](#_Toc1988218)

[3.1.2 Problem 1: Test Graph <String> 1](#_Toc1988219)

[3.1.3 Problem 2: Implement Graph <String> 1](#_Toc1988220)

[3.1.3.1 Implement ConcreteEdgesGraph 2](#_Toc1988221)

[3.1.3.2 Implement ConcreteVerticesGraph 2](#_Toc1988222)

[3.1.4 Problem 3: Implement generic Graph<L> 2](#_Toc1988223)

[3.1.4.1 Make the implementations generic 2](#_Toc1988224)

[3.1.4.2 Implement Graph.empty() 2](#_Toc1988225)

[3.1.5 Problem 4: Poetic walks 2](#_Toc1988226)

[3.1.5.1 Test GraphPoet 2](#_Toc1988227)

[3.1.5.2 Implement GraphPoet 2](#_Toc1988228)

[3.1.5.3 Graph poetry slam 2](#_Toc1988229)

[3.1.6 Before you’re done 2](#_Toc1988230)

[3.2 Re-implement the Social Network in Lab1 2](#_Toc1988231)

[3.2.1 FriendshipGraph类 2](#_Toc1988232)

[3.2.2 Person类 3](#_Toc1988233)

[3.2.3 客户端main() 3](#_Toc1988234)

[3.2.4 测试用例 3](#_Toc1988235)

[3.2.5 提交至Git仓库 3](#_Toc1988236)

[3.3 Playing Chess 3](#_Toc1988237)

[3.3.1 ADT设计/实现方案 3](#_Toc1988238)

[3.3.2 主程序ChessGame设计/实现方案 3](#_Toc1988239)

[3.3.3 ADT和主程序的测试方案 3](#_Toc1988240)

[3.4 Multi-Startup Set (MIT) 4](#_Toc1988241)

[4 实验进度记录 4](#_Toc1988242)

[5 实验过程中遇到的困难与解决途径 4](#_Toc1988243)

[6 实验过程中收获的经验、教训、感想 4](#_Toc1988244)

[6.1 实验过程中收获的经验和教训 4](#_Toc1988245)

[6.2 针对以下方面的感受 4](#_Toc1988246)

# 实验目标概述

本次实验训练抽象数据类型（ADT）的设计、规约、测试，并使用面向对象 编程（OOP）技术实现 ADT。具体来说： 针对给定的应用问题，从问题描述中识别所需的 ADT；设计 ADT 规约（pre-condition、post-condition）并评估规约的质量；根据 ADT 的规约设计测试用例；ADT 的泛型化；根据规约设计 ADT 的多种不同的实现；针对每种实现，设计其表示 （representation）、表示不变性（rep invariant）、抽象过程（abstraction function） 使用 OOP 实现 ADT，并判定表示不变性是否违反、各实现是否存在表 示泄露（rep exposure）； 测试 ADT 的实现并评估测试的覆盖度； 使用 ADT 及其实现，为应用问题开发程序； 在测试代码中，能够写出 testing strategy 并据此设计测试用例。

# 实验环境配置

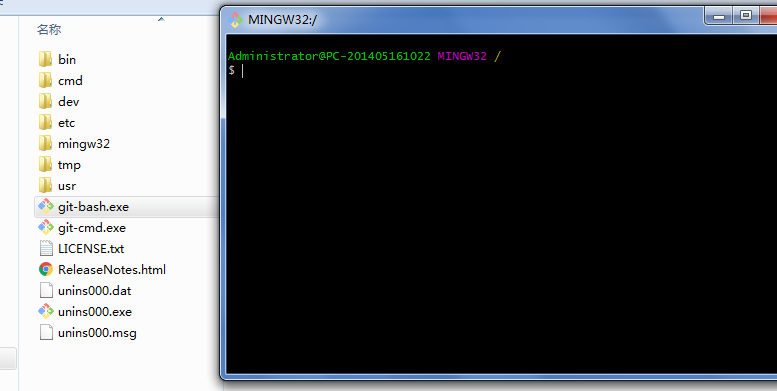
简要陈述你配置本次实验所需环境的过程，必要时可以给出屏幕截图。

特别是要记录配置过程中遇到的问题和困难，以及如何解决的。

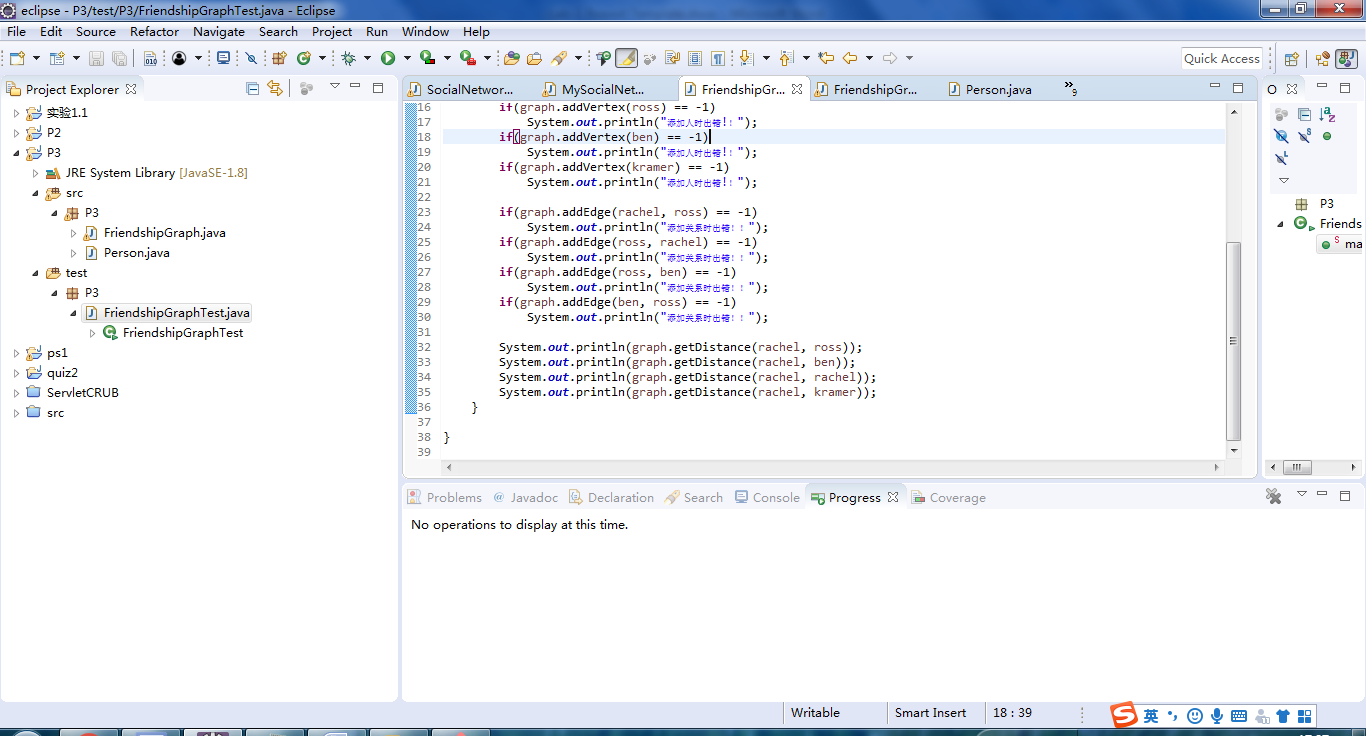
1安装Java

从Java Development Kit 10下载，下载Java SE Development Kit 10.0.2并运行安装程序。此处需要配置环境变量。

2安装Git



3安装eclipse



在这里给出你的GitHub Lab2仓库的URL地址（Lab2-学号）。

https://github.com/ComputerScienceHIT/Lab2-1173710107.git

# 实验过程

请仔细对照实验手册，针对三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！）。

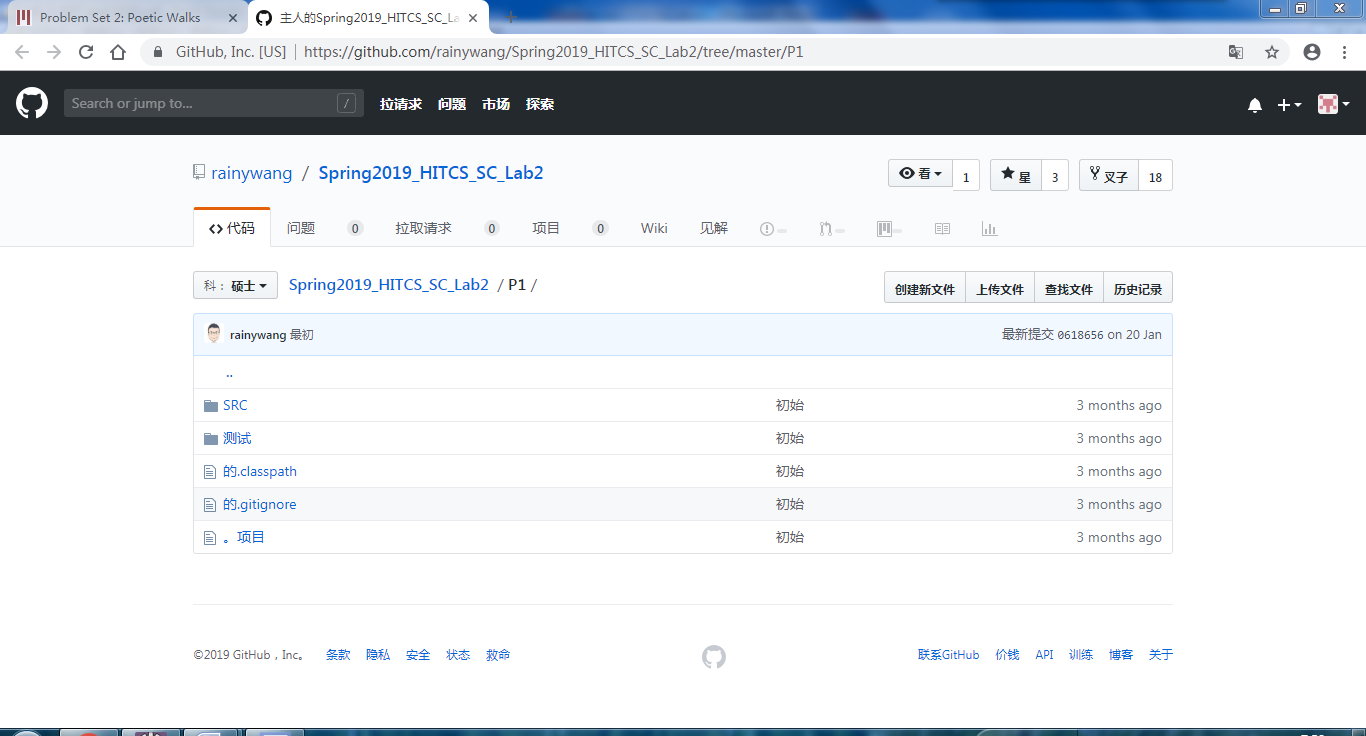
## Poetic Walks

在这里简要概述你对该任务的理解。

这部分实验主要是让我们实现ConcreteEdgesGraph以及ConcreteVerticesGraph两个部分，用不同的表示方法来实现有向图的处理。然后去实现Graph<L>，让上面的类可以得到通用。最后的一个部分是自己写一段代码，实现通过读入语料库的信息，增加字在语句中，可以让输入的语句变的更加有诗意。

### Get the code and prepare Git repository

如何从GitHub获取该任务的代码、在本地创建git仓库、使用git管理本地开发。



从上面获取代码

打开git bash

输入命令git clone +网址将github上面的仓库克隆到本地

下面是上传文件的命令

git add .        （注：别忘记后面的.，此操作是把文件夹下面的文件都添加进来）

git commit  -m  ”提交信息”

git push -u origin master   （注：此操作目的是把本地仓库push到github上面，此步骤需要输入帐号和密码）

### Problem 1: Test Graph <String>

以下各部分，请按照MIT页面上相应部分的要求，逐项列出你的设计和实现思路/过程/结果。

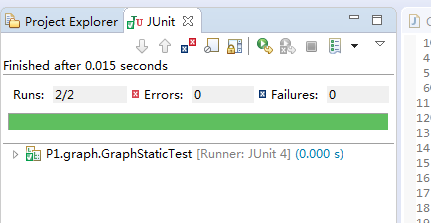
修改empty为：

Public Graph<String> empty() {

       return new ConcreteEdgesGraph();

}

然后在ConcreteEdgesGraph中的vertices方法直接返回vertices，则在调用test时候，返回一个空的set



此时可以运行GraphStaticTest进行测试。

### Problem 2: Implement Graph <String>

以下各部分，请按照MIT页面上相应部分的要求，逐项列出你的设计和实现思路/过程/结果。

这部分是按照接口实现两个类ConcreteEdgesGraph以及ConcreteVerticesGraph的方法，其中一个是以边为中心，一个是以点为中心。

下面依次介绍两个部分的实现：

#### Implement ConcreteEdgesGraph

这部分是以边为中心的，每条边是由起点，终点和权重组成。

然后定义两个集合

**private** **final** Set<String> vertices = **new** HashSet<>();

**private** **final** List<Edge<String>> edges = **new** ArrayList<>();

下面的操作通过对这两个集合来实现：

@Override public boolean add(String vertex)：

该方法是增加点到有向图中，所以只需要在vertices中直接add就行了

@Override public int set(String source, String target, int weight)：

该方法是将source到target这条边的权重设置为weight，这里这里一个flag=0,然后遍历整个edges，如果能找到对应的边，将权重修改，flag=1，在结束之前判断flag的值，如果是1，证明被修改过了，直接返回，如果是0，证明没有找到这条边，需要添加这条边，然后返回。

@Override public boolean remove(String vertex)：

该方法是删除掉这个顶点，需要注意的是，如果一个无向图中没有这个顶点，那么这个顶点作为source或者target的边也就不存在了。所以需要先把vertices中的顶点去掉，然后遍历edges，判断如果这条边的source或者target和vertex相同，那么就删除掉这条边。

@Override public Set<String> vertices()

这个在上一个问题中就已经实现了，此处不再赘述。

@Override public Map<String, Integer> sources(String target)

这个方法是获取所有边中终点为target的边，并且把相关信息填充到map中，返回map.具体实现是首先定义一个map,然后遍历整个edges，如果target相同的话，将source和weight填到map中，遍历结束之后返回map就行了

@Override public Map<String, Integer> targets(String source)

这个方法是获取所有边中起点为source的边，并且把相关信息填充到map中，返回map.具体实现是首先定义一个map,然后遍历整个edges，如果source相同的话，将target和weight填到map中，遍历结束之后返回map就行了

#### Implement ConcreteVerticesGraph

这部分是以点为中心实现有向图，一个点类包括这个点，以这个点为起点的边构成的map,以这个点为终点的边构成的map,具体map是终点/起点+权重

private final List<Vertex<String>> vertices = new ArrayList<>();

下面是具体接口方法的实现：

@Override public boolean add(String vertex)

该方法是将顶点添加到有向图中，所以需要自己新建一个Vertex，将vertex添加进去，然后vertices.add就可以了

@Override public int set(String source, String target, int weight)

该方法是将source到target这条边的权重设置为weight，所以需要遍历整个vertices，提前设置flag和flag1，都等于零，用来查看是否找到对应的边，在遍历过程中，看顶点和source或者target是否相同，如果和source相同，那么在这个顶点的target的map中添加信息，flag=1，另一个同理对应，然后在结束时查看flag和flag1是否改变，如果没有改变说明该边没有找到，那么自己添加进去。

@Override public boolean remove(String vertex)

该方法是删除掉这个顶点，此处的这个数据结构不存在删除边的问题，所以直接遍历顶点集，如果vertex相同的话，删除顶点就可以了。

@Override public Set<String> vertices()

这个方法需要返回的是顶点的set，所以需要新声明一个Set，然后遍历将顶点加进Set中就可以了

@Override public Map<String, Integer> sources(String target)

这个方法是获取所有边中终点为target的边，并且把相关信息填充到map中，返回map.因为这个数据结构直接有map了，所以只需要遍历找到对应的边，然后返回sources的map就可以了

@Override public Map<String, Integer> targets(String source)

这个方法是获取所有边中起点为source的边，并且把相关信息填充到map中，返回map. 因为这个数据结构直接有map了，所以只需要遍历找到对应的边，然后返回这个顶点的target的map就可以了

### Problem 3: Implement generic Graph<L>

这个问题的主要作用就是在未来的使用过程中,将顶点的类型不仅仅局限于string类型，所以使用L替代string，可以将所写的代码支持多个数据类型

#### Make the implementations generic

public class ConcreteVerticesGraph<L> implements Graph<L>

public class ConcreteEdgesGraph<L> implements Graph<L>

将两个类的函数声明改掉，然后将两个类中的所有String都用L代替，包括上面定义的set,list都要改掉，就可以了。

#### Implement Graph.empty()

将上面的

Public Graph<String> empty() {

       return new ConcreteEdgesGraph();

}

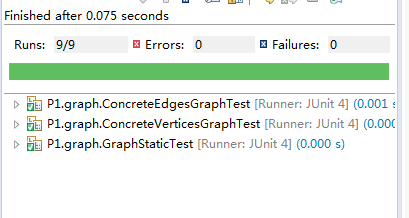
改成：

**public** **static** <L> Graph<L> empty() {

**return** **new** ConcreteEdgesGraph<L>();

}就可以了

下面测试整个P1.praph



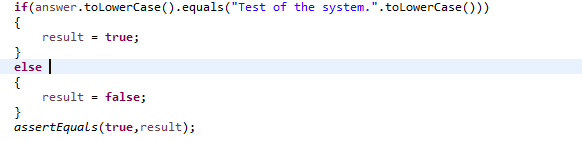
### Problem 4: Poetic walks

这一部分是通过他给的一步步操作，编写test，并实现通过读入语料库的信息，增加字在语句中，可以让输入的语句变的更加有诗意。

#### Test GraphPoet

这部分主要是先写测试用例，就是在提供语料库的前提下，将一个句子输入到即将要写的public String poem方法中，并获取他的返回值，然后将获得的句子和自己的预期得到的句子，将两个句子进行比较。

在这里具体实现我遇到的一个问题就是assertequals方法因为是判断两个参数的地址是否相同，所以无法直接用两个句子比较，这里我利用设置boolean类型判断不同的结果下给出不同值的办法解决这个问题。



#### Implement GraphPoet

具体实现我用了下面两个方法：

public GraphPoet(File corpus)

该方法是读入语料库的信息，将他们构成一个有向图。

这里的具体实现我借助了P1.graph的接口，将语料库读入之后，根据分隔符将库中的语句分成若干的词汇，然后将每一个词汇作为一个有向图的顶点，然后将每一个词和下一个词连接一条有向的边。最后构成一个有向图

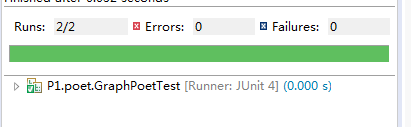
public String poem(String input)

该方法是将读入的语句返回一个新的语句，具体实现是将输入的语句利用分隔符划分成词汇，然后依次遍历这些词汇，将每一个词汇在有向图中查找，如果找到相同的词汇，将有向图广度优先搜索两个长度，判断是否和输入语句的下一个词汇相同，如果相同的话，将图中搜索一个长度的词插入到语句中，然后将下一次词在进行相同的操作，直到语句遍历完之后，返回整个句子。

#### Graph poetry slam

在main函数中读入文件中的信息，然后调用.poem方法来获取添加之后语句，将两个语句输出

下面是test的结果

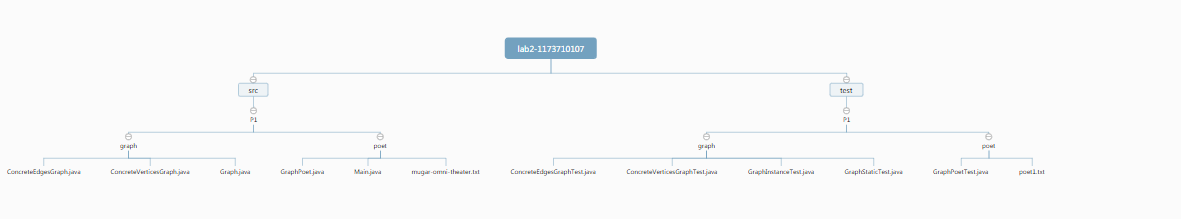


### Before you’re done

请按照<http://web.mit.edu/6.031/www/sp17/psets/ps2/#before_youre_done>的说明，检查你的程序。

如何通过Git提交当前版本到GitHub上你的Lab2仓库。

在这里给出你的项目的目录结构树状示意图。



下面是上传文件的命令

git add .        （注：别忘记后面的.，此操作是把文件夹下面的文件都添加进来）

git commit  -m  ”提交信息”

git push -u origin master   （注：此操作目的是把本地仓库push到github上面，此步骤需要输入帐号和密码）

## Re-implement the Social Network in Lab1

在这里简要概述你对该任务的理解。

这部分实验主要是让我们复用上面自己完成的接口实现类来重写自己的lab1p3，两个实现类选择一个就可以了，然后实现自己之前写的功能。

### FriendshipGraph类

给出你的设计和实现思路/过程/结果。

这部分我借用的是ConcreteVerticesGraph<Person>

这个类主要实现三个方法：

public int addVertex(Person name)：

该方法主要用于向关系有向图中添加顶点，也就是人，首先判断该name是否可用，如果不可用，直接返回-1，可用的情况下，遍历整个图，寻找是否有和该人名相同的人，有的话说明该人已经被添加。否则使用add将该人添加到有向图中。

public int addEdge(Person person1, Person person2)：

该方法用于将两个人建立有向的关系。具体实现是判断是否这两个人可用，可用的话使用set将关系添加到有向图中。

public int getDistance(Person person1, Person person2)：

该方法主要用于获取两个人之间的距离。具体实现是先看这两个人是否可用，可用的话，令distance=0，然后判断两者的人名是否相同，相同的话直接返回distance，否则进行深度优先搜索算法去寻找person1的target，没找完一轮，distance++；找到之后返回distance,整个图找完还是没有的话，返回-1.

### Person类

给出你的设计和实现思路/过程/结果。

Person中包括name，visited，friends三个参数，并且撰写set,get方法，其中name是人名，visited表示这个人是否被访问过，friends表示这个人的朋友，用list存储。

### 客户端main()

给出你的设计和实现思路/过程/结果。

Main函数主要用于添加一些自己构建的人名，并构成一个有向图，然后测试他们的距离是否和预计的相同

### 测试用例

给出你的设计和实现思路/过程/结果。

这里主要测试主要的三个方法是否成功实现。

void testAddVertex()，添加一个人名，看看返回值和预期是否相同

void testAddEdge()，添加人名和边，看看返回值和预期是否相同

void testGetDistance()自己构建一个有向关系网，然后判断返回的距离和预期是否相同

### 提交至Git仓库

如何通过Git提交当前版本到GitHub上你的Lab3仓库。

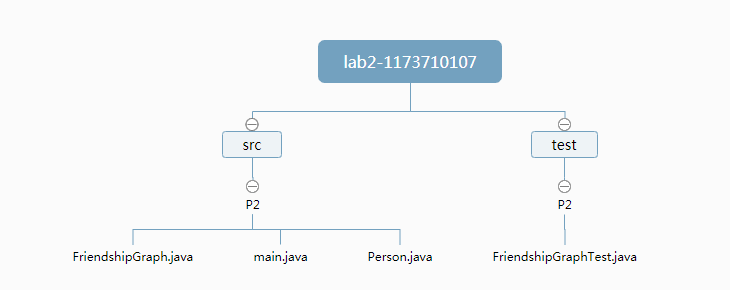
在这里给出你的项目的目录结构树状示意图。

下面是上传文件的命令

git add .        （注：别忘记后面的.，此操作是把文件夹下面的文件都添加进来）

git commit  -m  ”提交信息”

git push -u origin master   （注：此操作目的是把本地仓库push到github上面，此步骤需要输入帐号和密码）



## Playing Chess

### ADT设计/实现方案

设计了哪些ADT（接口、类），各自的rep和实现，各自的mutability/ immutability说明、AF、RI、safety from rep exposure。

必要时请使用UML class diagram（请自学）描述你设计的各ADT间的关系。

我使用了ACTION,PIECE,PLAYER,BOARD等多个接口和类，并用一个接口P3GAME将这些接口整合到一起。对于多个类，主要是一个game1类，下面包括piece1，board1,player1三个类。

下面依次介绍具体的实现：

1. public void movePiece(int num,int startx,int starty,int endx,int endy);

该方法是将该玩家的start位置的棋子移动到end位置

Num表示实现操作的玩家

Immutability AF：num=0表示玩家一，num=2表示玩家二

RI：x,y都>0且小于棋盘大小num只能是01，true

2. public void removePiece(int num,int endx,int endy);

该方法是将该玩家的end位置的棋子拿掉,参数为实现移除棋子操作的玩家以及移除棋子的位置，Num表示实现操作的玩家，num=0表示玩家一，num=2表示玩家二

Immutability AF：num=0表示玩家一，num=2表示玩家二

RI：x,y都>0且小于棋盘大小num只能是01，true

3. public void eatPiece(int num,int startx,int starty,int endx,int endy);

该方法是实现player玩家的吃子操作，参数为实现吃子操作的玩家以及棋子的起始位置start和结束位置end, Num表示实现操作的玩家，num=0表示玩家一，num=2表示玩家二

Immutability AF：num=0表示玩家一，num=2表示玩家二

RI：x,y都>0且小于棋盘大小num只能是01，true

4. public Piece1 getPiece(int x,int y);

该方法是获取xy对应位置的棋子的信息,参数为获取棋子的位置的x y坐标

返回该位置的棋子

mutability

RI：x,y都>0且小于棋盘大小，true

5. public void setPiece(int x,int y,Piece1 piece);

该方法是将棋子放在棋盘的对应位置，参数为棋子piece以及放置的位置x y

mutability RI：x,y都>0且小于棋盘大小，true

6. public boolean isavailable(int x,int y);

该方法用于检查给定的xy位置坐标是否可用，参数为棋盘上x y坐标 ，对于返回值，如果该位置是有意义的，那么返回true，否则返回false

Immutability RI：x,y都>0且小于棋盘大小，true

7. public boolean isinborad(Piece1 piece);

该方法检查该棋子是否在棋盘上，参数为需要检查的棋子piece，对于返回值，如果该棋子在棋盘上，返回true，否则返回false

mutability

true

8,。public int getplayernum(Player1 player);

该方法用于获取该玩家棋盘上棋子的数量，参数为玩家player，返回值为该玩家棋盘上棋子的数量

mutability AF：player包括name,pieces ,true

9. public void initgameplayer(String name1,String name2,int num);

该方法主要用于初始化游戏，具体包括棋盘，玩家，参数为两个player的名字以及游戏类型num ，当num=0时，表示是象棋游戏，当num=1时，表示是围棋游戏

Immutability AF：num=0表示象棋，num=2表示围棋

RI：name不为空num只能是0,1，true

10. public Player1 getPositionplayer(Position1 x);

该方法用于获取该位置对应的玩家，参数为获取棋盘位置position，返回值为该位置的玩家

mutability AF：position对应一个x和y RI：position里面x,y都>0且小于棋盘大小，true

11. public Piece1 getPositionPiece(Position1 x);

该方法用于获取该位置的棋子，参数为获取棋子的位置position，返回值为该位置的棋子

mutability AF：position对应一个x和y RI：position里面x,y都>0且小于棋盘大小，true

12. public int getnumPiece(int num);

该方法用于获取该玩家棋盘上棋子的数量，参数为玩家num,num=0表示玩家一，num=1表示为玩家二，返回值为该玩家棋盘上棋子的数量

immutability AF：num=0对应玩家一，num=1对应玩家二 RI：num只能是01，true

13. public String getpname(int px,int py);

该方法主要用于获取当前棋子的name，参数为需要获取name的棋子piece的xy，返回值返回该棋子的name 类型为String

immutability RI： x,y都>0且小于棋盘大小，true

14. public String getpfrom(Piece1 piece);

该方法主要用于获取该棋子所属玩家的名字，参数为该棋子，返回值为玩家名字

immutability AF：piece包括name，from，xy RI：piece里面x,y都>0且小于棋盘大小,from必须是两个玩家中的一个，true

15. public void setpfrom(String from,Piece1 piece);

该方法主要是设置棋子的所属玩家，参数from为玩家的名字，piece为设置的棋子

mutability AF：piece包括name，from，xy RI：piece里面x,y都>0且小于棋盘大小，from必须是玩家名字，true

16. public int getpx(Piece1 piece);

该方法是获取棋子在棋盘上的y坐标，参数为棋子piece，返回值为棋盘上的坐标y

mutability AF：piece包括name，from，xy RI：piece里面x,y都>0且小于棋盘大小，from必须是玩家名字，true

17. public int getpy(Piece1 piece);

该方法用于获取棋子的状态，0表示未放置，1表示被放置在棋盘上，2表示被吃掉弃用，参数为棋子piece，返回值为该棋子的当前状态int类型

mutability AF：piece包括name，from，xy RI：piece里面x,y都>0且小于棋盘大小，from必须是玩家名字，true

18. public int getpstate(int px,int py);

该方法用于获取棋子的状态，0表示未放置，1表示被放置在棋盘上，2表示被吃掉弃用，参数为棋子piece，返回值为该棋子的当前状态int类型

immutability RI：x,y都>0且小于棋盘大小，返回只有012三种可能，true

19. public void setpname(String pname,Piece1 piece);

该方法用于设置棋子的name，参数为棋子piece以及name

mutability AF：piece包括name，from，xy RI：piece里面x,y都>0且小于棋盘大小，from必须是玩家名字，true

20. public void setpx(int px,Piece1 piece);

该方法用于设置棋子在棋盘上的x坐标，参数为棋子在棋盘上x位置px,棋子piece

mutability AF：piece包括name，from，xy RI：piece里面x,y都>0且小于棋盘大小，from必须是玩家名字，true

21. public void setpy(int py,Piece1 piece);

该方法用于设置棋子在棋盘上的y坐标，参数为棋子在棋盘上x位置py,棋子piece

mutability AF：piece包括name，from，xy RI：piece里面x,y都>0且小于棋盘大小，from必须是玩家名字，true

22. public void remove(Piece1 x,int num);

该方法用于删除棋盘上的棋子，num表示该棋子是哪个玩家的，num=0表示是玩家一的棋子num=1表示是玩家二的棋子，参数为需要删除的棋子piece以及玩家标志变量num

mutability AF：piece包括name，from，xy ，num=0表示玩家一，num=1表示玩家二RI：piece里面x,y都>0且小于棋盘大小，from必须是玩家名字num只能是01，true

23. public String getplayername(int num);

该方法用于获取玩家的name,其中num是玩家标志num=0表示是玩家一，num=1表示是玩家二，参数为玩家标志变量num，返回值为玩家name,String类型

immutability AF：num=0表示玩家一，num=1表示玩家二 RI：num必须是01，true

24. public Set<Piece1> getplayerPieces(int num);

该方法用于获取该玩家的棋子的集合， 其中num是玩家标志num=0表示是玩家一，num=1表示是玩家二，参数为玩家标志变量num，返回值为该玩家的棋子的集合set

mutability AF：num=0表示玩家一，num=1表示玩家二RI：num必须是01，true

25. public StringBuilder getplayerhistory(int num);

该方法用于获取该玩家的操作历史记录，其中num是玩家标志num=0表示是玩家一，num=1表示是玩家二，参数为玩家标志变量num，返回值为该玩家的操作历史StirngBuilder

immutability AF：num=0表示玩家一，num=1表示玩家二 RI：num必须是01，true

26. public boolean addPiece(Piece1 x,int num);

该方法用于添加棋子到玩家中，参数为添加棋子piece以及num其中num是玩家标志num=0表示是玩家一，num=1表示是玩家二，返回值为如果添加成功返回true,否则返回false

mutability AF：piece包括name，from，xy ：num=0表示玩家一，num=1表示玩家二RI：piece里面x,y都>0且小于棋盘大小,from必须是两个玩家中的一个，num必须是01，true

27. public boolean iscontainPiece(int num,Piece1 x);

该方法用于判断该玩家是否有该棋子，其中num是玩家标志num=0表示是玩家一，num=1表示是玩家二，参数为棋子piece以及玩家标志变量num，返回值为如果该玩家包含该棋子，返回true，否则返回false

mutability AF：piece包括name，from，xy ：num=0表示玩家一，num=1表示玩家二RI：piece里面x,y都>0且小于棋盘大小,from必须是两个玩家中的一个，num必须是01，true

28. public void setplayername(String name,int num);

该方法用于设置玩家名字，参数为玩家名字name以及num,其中num是玩家标志num=0表示是玩家一，num=1表示是玩家二

immutability AF： num=0表示玩家一，num=1表示玩家二RI：num必须是01,name不为空，true

29. public void addhistory(int num,String step);

该方法主要用于记录玩家的操作，参数为step表示玩家的操作步骤，以及num，其中num是玩家标志num=0表示是玩家一，num=1表示是玩家二

immutability AF：step不为空 num=1表示玩家二RI： num必须是01，true

30. public Player1 getplayer(int num);

该方法用于获取玩家，其中num是玩家标志num=0表示是玩家一，num=1表示是玩家二，参数为 玩家标志num

mutability AF： num=0表示玩家一，num=1表示玩家二RI： num必须是01，true

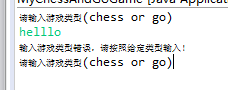
### 主程序MyChessAndGoGame设计/实现方案

辅之以执行过程的截图，介绍主程序的设计和实现方案，特别是如何将用户在命令行输入的指令映射到各ADT的具体方法的执行。

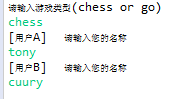
程序中有printMenu()方法，用于输出命令的菜单。

主程序的运行顺序如下：

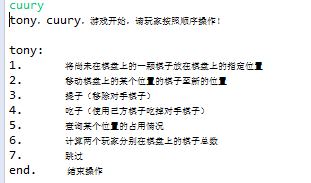
首先要求玩家选择棋牌的类型，当输入错误时，会出现下面的结果：



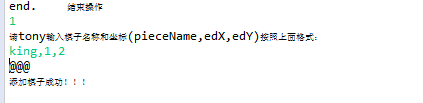
输入正确之后，会要求输入两个玩家的信息：



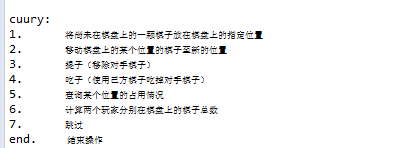
结束之后：



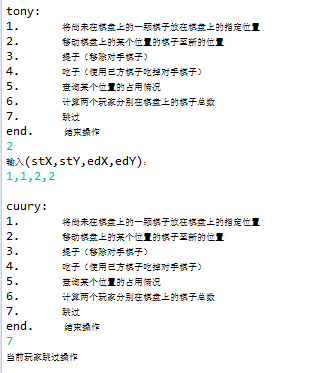
开始下棋，先输入名字的玩家执先手。



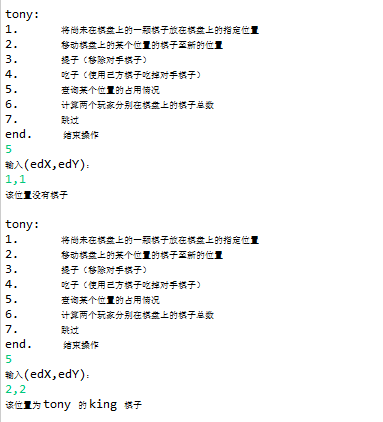
操作完成之后轮到另一位玩家：



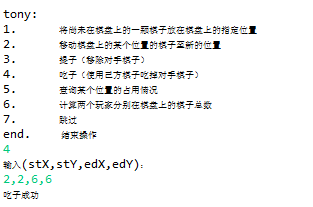
接下来测试各个功能：



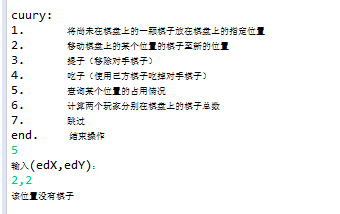
跳过操作直接跳过



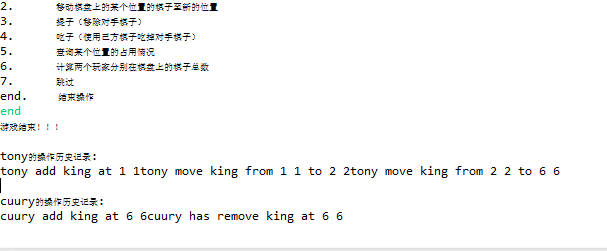
从上面可以看到棋子已经被移动，而且查询某个位置占用情况的功能也被实现



吃子之后



因为该位置棋子移动，所以此处2，2没有棋子了



最后可以看到玩家的两个操作记录

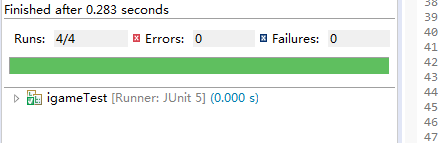
### ADT和主程序的测试方案

介绍针对各ADT的各方法的测试方案和testing strategy。

介绍你如何对该应用进行测试用例的设计，以及具体的测试过程。

因为该实验主要的操作就是包括移动棋子，删除棋子，吃子和获取棋子的数目，其他的实现方法都是可以通过上面的测试过程中去检测的，也就是说只要其他的方法出现了问题，上面的四个方法同样也不会通过，所以主要测上面的四个方法

测试方法就是在测试方法里面自己构建一个简单的棋盘，然后进行上面的操作，使用assertequals函数判断对应得到的结果和期望结果是否相同！下面是测试结果：



## Multi-Startup Set (MIT)

请自行设计目录结构。

注意：该任务为选做，不评判，不计分。

# 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 | 计划任务 | 实际完成情况 |
| 03-18 | 实验课 | 完成P1问题一 | 未完成，还是没明白让我干什么。。。。 |
| 03-19 | 21:00-23:00 | 完成P1问题一 | 完成 |
| 03-21 | 15:00-22:00 | 完成P1问题二 | 完成一部分，有一些东西不是很明白什么意思，没有写 |
| 03-23 | 08:00-18:00 | 完成p1全部 | 除了问题二一些还是不太明白，剩下的写完了 |
| 03-24 | 08:00-15:00 | 完成P2 | 提前完成 |
| 03-25 | 实验课 | 设计P3 | 完成 |
| 03-30 | 08:00-22:00 | 设计接口和类，写SPEC，写测试用例 | 测试用例没有写 |
| 03-31 | 08:00-12:00 | 完成测试用例和具体的实现 | 具体实现未完成 |
| 04-01 | 实验课 | 继续写P3 | 未完成 |
| 04-02 | 14:00-22:00 | 完成实验P3 | 完成 |
| 04-05 | 整天 | 写实验报告完善之前没写的 | 未完成报告，未完成实验内容 |
| 04-06 | 整天 | 同上 | 完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 在做P1的Poetic walks的时候，输出的字符串开头总带有null，比如应该输出test of the system.，结果输出的是null test of the system.， | 经过自己的检查，并不是自己的写的方法不对，而是在我把list转化成string的过程中，一开始定义string s =null，然后遍历list累加string这部分的问题，我把一开始的s初始化变成answer.get(0)，之后就没有问题了！ |
| 同样在做P1的测试部分的时候，因为之前遇到过问题，我知道那个assertEquals函数比较的是两个变量的地址是否相同，而不是单单是值是否相同，此时新产生的字符串和自己的正确答案就不在一个地址上，这怎么比较 | 这个在网上看到了一个方法，就是因为字符串的比较.equals是可以只比较值，所以自己比较字符串，根据结果给出对应的boolean，最后在assertEquals的函数中比较boolean值 |
| 在做P3的整体代码结构的设计的时候，因为第一次设计，有的功能没有设计上，有的功能完全是多余的，最后根本没有用上 | 在实际写代码的时候把少的功能自己加上了，多余的功能也实现了，我觉得毕竟作为一个接口，这些功能总会有在其他时候去使用的价值，在以后设计代码的时候可能自己会更有经验 |
|  |  |
|  |  |

# 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

这次实验感觉实现代码还是次要的，主要还是要养成写注释写测试用例以及spec的习惯，同时感觉以后做实验感觉需要做一部分实验之后马上就写这部分的实验报告，否则的话，之后写实验报告感觉并没有当时刚做完实验那么有感触，一些细节的地方之后都不太记得了

## 针对以下方面的感受

1. 面向ADT的编程和直接面向应用场景编程，你体会到二者有何差异？
2. 使用泛型和不使用泛型的编程，对你来说有何差异？
3. 在给出ADT的规约后就开始编写测试用例，优势是什么？你是否能够适应这种测试方式？
4. P1设计的ADT在多个应用场景下使用，这种复用带来什么好处？
5. P3要求你从0开始设计ADT并使用它们完成一个具体应用，你是否已适应从具体应用场景到ADT的“抽象映射”？相比起P1给出了ADT非常明确的rep和方法、ADT之间的逻辑关系，P3要求你自主设计这些内容，你的感受如何？
6. 为ADT撰写specification, invariants, RI, AF，时刻注意ADT是否有rep exposure，这些工作的意义是什么？你是否愿意在以后编程中坚持这么做？
7. 关于本实验的工作量、难度、deadline。
8. 《软件构造》课程进展到目前，你对该课程有何体会和建议？

ADT编程需要考虑到安全性的问题，使用泛型的话，可以在其他的地方复用自己之前已经写好的代码，代码可用性更高，首先编写测试用例可以更明白自己的这些方法需要具体实现那些功能，可以在具体写代码之前首先对自己的整个代码框架有更好的了解。自己还不是很适应这个模式，写测试因为自己的代码还没有实现，写起来就感觉很难受，以后可能写多了会习惯的。复用的好处是因为泛型可以支持多个数据结构，相同功能的代码就可以不用自己重复写了，写P3我的感受很深，因为是从0开始写，所以自己一开始设计整个代码框架的时候有时候会考虑不全面，忽略到某个部分的问题，直到自己真正开始写的时候才会发现，感觉自己从0开始的能力还是很弱，以后需要多加强这方面的练习。撰写这些在于可以帮助我们的代码在面对不同的情况都可以稳定的执行，对于个人的良好的变成习惯的培养很有帮助，愿意以后坚持这种习惯。这个实验难度还行，有部分实验在一开始理解题意方面有很大问题，比如P1刚拿到题，我一直很懵，不知道要干什么，现在做完回过头来看，就比较容易了。这门课程感觉对自己编程的习惯的提升很大，希望老师在布置雨课堂的时候把截止时间延长一点，毕竟下课之后还有其他课，上楼去其他教室也是需要时间的。