

**2020年春季学期  
计算机学院《软件构造》课程**

**Lab 2实验报告**

|  |  |
| --- | --- |
| 姓名 | 张义辞 |
| 学号 | 1180300314 |
| 班号 | 0300314 |
| 电子邮件 | [1307686612@qq.com](mailto:1307686612@qq.com) |
| 手机号码 | 19845178026 |

**目录**

[1 实验目标概述 1](#_Toc29325498)

[2 实验环境配置 1](#_Toc29325499)

[3 实验过程 1](#_Toc29325500)

[3.1 Poetic Walks 1](#_Toc29325501)

[3.1.1 Get the code and prepare Git repository 1](#_Toc29325502)

[3.1.2 Problem 1: Test Graph <String> 1](#_Toc29325503)

[3.1.3 Problem 2: Implement Graph <String> 1](#_Toc29325504)

[3.1.3.1 Implement ConcreteEdgesGraph 2](#_Toc29325505)

[3.1.3.2 Implement ConcreteVerticesGraph 2](#_Toc29325506)

[3.1.4 Problem 3: Implement generic Graph<L> 2](#_Toc29325507)

[3.1.4.1 Make the implementations generic 2](#_Toc29325508)

[3.1.4.2 Implement Graph.empty() 2](#_Toc29325509)

[3.1.5 Problem 4: Poetic walks 2](#_Toc29325510)

[3.1.5.1 Test GraphPoet 2](#_Toc29325511)

[3.1.5.2 Implement GraphPoet 2](#_Toc29325512)

[3.1.5.3 Graph poetry slam 2](#_Toc29325513)

[3.1.6 Before you’re done 2](#_Toc29325514)

[3.2 Re-implement the Social Network in Lab1 2](#_Toc29325515)

[3.2.1 FriendshipGraph类 2](#_Toc29325516)

[3.2.2 Person类 3](#_Toc29325517)

[3.2.3 客户端main() 3](#_Toc29325518)

[3.2.4 测试用例 3](#_Toc29325519)

[3.2.5 提交至Git仓库 3](#_Toc29325520)

[3.3 Playing Chess 3](#_Toc29325521)

[3.3.1 ADT设计/实现方案 3](#_Toc29325522)

[3.3.2 主程序MyChessAndGoGame设计/实现方案 3](#_Toc29325523)

[3.3.3 ADT和主程序的测试方案 3](#_Toc29325524)

[4 实验进度记录 4](#_Toc29325525)

[5 实验过程中遇到的困难与解决途径 4](#_Toc29325526)

[6 实验过程中收获的经验、教训、感想 4](#_Toc29325527)

[6.1 实验过程中收获的经验和教训 4](#_Toc29325528)

[6.2 针对以下方面的感受 4](#_Toc29325529)

# 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：

1. 针对给定的应用问题，从问题描述中识别所需的 ADT
2. 设计 ADT 规约（pre-condition、post-condition）并评估规约的质量；
3. 根据 ADT 的规约设计测试用例；
4. ADT 的泛型化；
5. 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）
6. 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）
7. 测试 ADT 的实现并评估测试的覆盖度；
8. 使用 ADT 及其实现，为应用问题开发程序
9. 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

测试开发环境为Eclipse和JAVA JDK8，在GitHub网站上下载git进行代码提交测试，在Eclipse使用team进行commit和push。

配置的测试Junit版本为1.4

java path环境为：

新建系统环境变量JAVA\_HOME,对应的变量值为你jdk安装的目录。

新建系统环境变量classpath

变量值.;%JAVA\_HOME%\lib;%JAVA\_HOME%\lib\tools.jar

在Path变量中新建变量为 %JAVA\_HOME%\bin

GitHub Lab2仓库的URL地址: https://github.com/ComputerScienceHIT/Lab2-1180300314

# 实验过程

## Poetic Walks

需要构造一个graph，以边集和点集实现接口，并且将ADT泛型化，以便支持所有的数据类型。

### Get the code and prepare Git repository

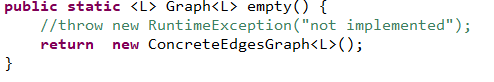
Git init

Git remote add origin <>

Git clone

### Problem 1: Test Graph <String>

将原有的Graph empty()方法修改为



适用ConcreteEdgesGraph作为Graph empty实现的实例类

### Problem 2: Implement Graph <String>

#### Implement ConcreteEdgesGraph

Class Edge

1. rep

private int weight;//边的权值

private String source;//起点

private String target;//终点

1. constructor

public Edge(L source\_1, L target\_1, int weight\_1) {

if (source\_1.equals(target\_1)) {

throw new RuntimeException("source and target should not be equal");

}

this.source = source\_1;

this.target = target\_1;

this.weight = weight\_1;

checkRep();

}

1. function

Edge(String,String,int)//构造函数生成一条边

checkRep()//确保weight>=0,source!=null,target!=null，target!=source

getSource()//一条边得到其源点

getTarget()//一条边得到其终点

getWright()//一条边得到其权值

toString()//输出一条边的表达形式

Class ConcreteEdgesGraph

1. rep

private final Set<String>vertices;//保存所有顶点

private final List<Edge<String>>edges;//保存所有边

1. function

checkRep();//检查表示不变性，需要满足所有边都有对应源点和终点，权值大于等于0

boolean add(String vertex);//加入一条边，存在则返回false，不存在则加入返回true.

int set(String source,String target,int weight);//遍历所有点，判断是否存在点，如不存在则先加入点，然后遍历所有边，判断边是否存在，如果存在且权值不为0则加入边，并修改权值，如果不存在则加入边。如果原有边权重值不为0且输入的权值为0则删除该边。

Boolean Remove(String vertex);先判断vertices中是否存在该点，不存在则返回false,如果存在则适用iterator来删除该边

Set<String> vertices();//先new一个HashSet适用防御式拷贝，复制一份vertices到HashSet中

Map<String,Integer> targets(String source) 建立一个空HashMap，遍历edges，找出以source为起点的edge集合加入hashmap中。

Map<String,Integer> source(String source) 建立一个空HashMap，遍历edges，找出以target为终点的edge集合加入hashmap中。

GraphInstanceTest.java

Public abstract Graph<String> emptyInstance();//抽象方法创建新的Graph

Public void testAdd();//测试对于已经存在的点和未存在的点能够正确加入

Public void testRemove();//测试对于存在的点能否正确移除，测试对为存在的点能够在移除时抛出异常

Public void testTargets();//测试能否正确返回目标终点集

Public void testSources();//测试能否正确返回目标原点集

Public void testSet();//测试边存在，边不存在，边为0的情况

ConcreteEdesGraphTest.java

Public Graph<String> emptyInstance();//将创建对应的EdgesGraph图

Public void testToString();//测试graph能否正确返回tostring函数

Public void testEdge();//测试能够正确构造出Edge

Public void testEdge\_1();//测试源点和终点相同时能否正确抛出异常

Public void testGetSource();测试能够正确得到原点

Public void testGetTarget();//测试能够正确得到终点

Public void testtoString();//测试能否正确返回边

#### Implement ConcreteVerticesGraph

**Class Vertex**

1. **rep**

Private L name;//当前节点名字

Private final Map<L,Integer>Sources = new HashMap<>();//以当前点为终点的边集和权值的HashMap

Private final Map<L,Integer>Targets = new HashMap<>();//以当前点为原点的边集和权值的HashMap

1. constructor

public Vertex(L name);//以名字来构造节点

1. function

checkRep();//检查所有对应的HashMap中的权值都大于0；

setTargetEdge(L,int);//设置到目标点的一条边

setSourcerTarget(L,int);设置到源点的一条边

removeTargetEdge();//移除到目标点的一条边

removeSourceEdge();//移除到源点的一条边

getSource();//得到Sources中所有的点

getTarget();//得到Targets中所有的点

Class concreteVertices

1. rep

private final List<Vertex<String>> vertices;//存储顶点列表

1. function

checkRep();//检查所有的边name不相同，且不会出现null

boolean add(L vertex);/.遍历vertices，如果存在name相同的边则返回false，否则创建新顶点加入list

int set(L source,L target,int weight)//如果source，target任何一个不存在则加入顶点，否则，如果两者都存在，则更新边权，如果权值等于0，则删除该边

Boolean remove(L vertex);//遍历查找是否有vertex在vertices中，如果没有则返回false，如果有则移除该点以及对应源点和终点上的边

Set<L>vertices 返回点集合的name

Sources(L vertex)//返回map，找到该点的所有源点及其权值，如果为空，则返回空的map

Targets(L vertex)//返回hashmap，找到该点的所有终点及其权值，如果为空，则返回空的map

ToString(L vertex)//格式化输出有向图

ConcreteVerticesGraphTest.java

Public void testToString()//调用Graph中的方法构建一张有向图，测试toString的返回值是否能正确规范地表示这张图。

Public void testVertexToString()//测试Vertex中方法能够格式化表示一个点及其终点和源点

Void testGetName()//测试Vertex类能够返回其name

Void testRemoveSource()//如果源点存在，测试Vertex能否删除其源点，如果不存在则抛出异常

Void testRemoveTarget()//如果终点存在，测试Vertex能否删除其终点，如果不存在则抛出异常

Void testSetTarget()//测试设置终点及其权值

Void testSetSource()//测试设置源点及其权值

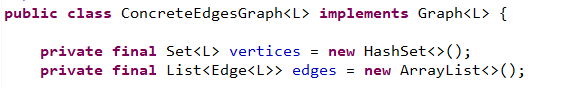
Void testGetSource()//测试得到hashmap包括源点和权值，如果没有源点就得到空的map

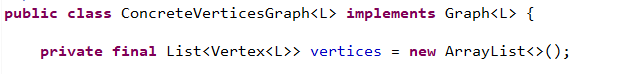
Void testGetTarget()//测试得到hashmap包括终点和权值，如果没有终点就得到空的map

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

#### Make the implementations generic

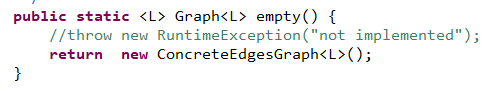
将两个EdgeGraph和VertexGraph类终得String参数替换为<L>得泛型



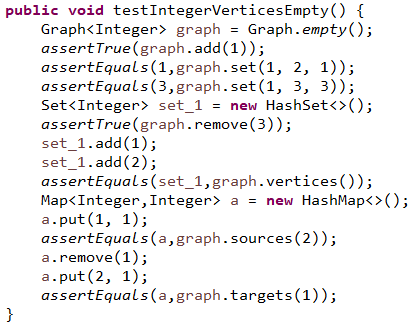


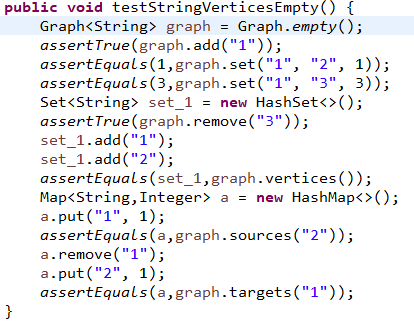
#### Implement Graph.empty()

将graph.empty()修改为默认以ConcreteEdgesGraph实现得实例类



在GraphStaticTest中测试两种泛型，以String和Int为例



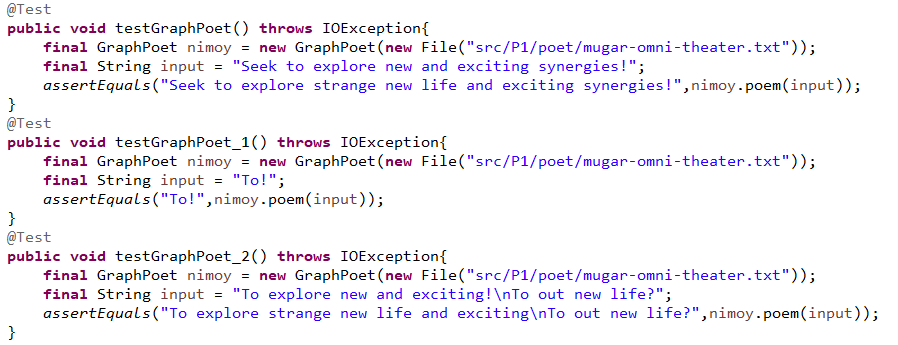


### Problem 4: Poetic walks

#### Test GraphPoet

测试单个单词，一行句子和多行句子的情况。

根据语料库的条件，输入相应句子



#### Implement GraphPoet

1. Rep

Private Graph<String> graph = Graph.empty();//适用empty建立有向图

1. Constructor

Public GraphPoet(File corpus)throws IOException//根据File路径读入文件，如果不能正常读入则抛出异常。

1. Function

Public void checkRep()//建立的语料库不能为空，否则无意义

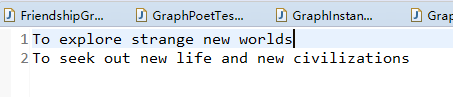
Public String poem(String input)//对input输入进行扩展，其中使用了自定义方法wordInsert来辅助插入单词。

Public void wordINsert(String1,String2) 如果两单词在语料库中有另单词作为第一个单词的终点和第二个单词的源点，且从源点到终点的权值（出现次数）最大,则返回该word

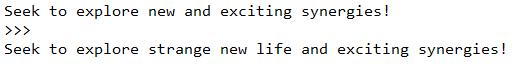
Public String toStirng()//重写方法，调用graph.toString来直观得到有向图语料库。

#### Graph poetry slam

语料库。



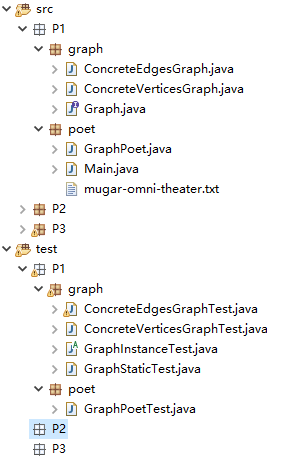
输出：



### Before you’re done

通过eclipse中的team，share进行commit 和push

在这里给出你的项目的目录结构树状示意图。



## Re-implement the Social Network in Lab1

通过任务1中实现的ConcreteEdgesGraph<L>，将其中的泛型替换为Person类实现

### FriendshipGraph类

1. Rep

Set<String> contain = new HashSet<>();//保存已经加入图的人的名字

1. constructor

private final Graph<Person> graph = Graph.empty();//通过Graph.empty()方法生成一张图

1. function

Public Person addVertex(Person person);//加入一个人到vertices中，如果这个人的名字已经存在，则抛出异常，否则，加入vertices

Public Boolean addEdge(Person ,Person)//在两个顶点之间加入一条边，边的权值未1

Public int getDistance(Person,Person)//计算两个人之间的距离，如果这两个人相同，则返回0，否则，适用BFS算法，先获得Person1的targets，然后入队，逐个出队获得后续的targets，当targets中的key值与person2相同时，结束运算，返回距离。如果遍历完所有点之后都没有边相连，则返回-1

### Person类

1. Rep

Private String name;//名字对应人

1. Function

Public String getName();//获得名字

CheckRep();//名字不为null

1. Constructor

Public Person(String name){

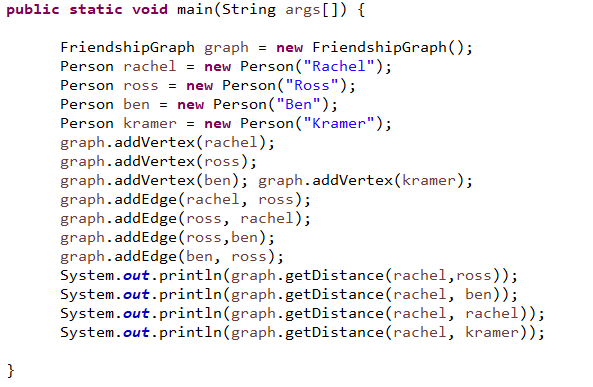
This.name = name;

}

### 客户端main()

与实验1中的main函数相同

先加入4个人，然后测试他们之间的距离。



### 测试用例

Public void testAddVertex();//测试重名，已经加入图和未加入图

Public void testaddEdge();//测试正确加入边，人际关系应该允许重边

Public void testGetDistance();//测试正确得到距离，测试联通点和相同点和不同点

Public void testComplexadjacencyList();//测试复杂图是否正确

### 提交至Git仓库

通过eclipse中的team 中的share进行commit和push

在这里给出你的项目的目录结构树状示意图。

## Playing Chess

### ADT设计/实现方案

1. Position(mutable)

// AF:x对应横坐标上的x点，y对应纵坐标上的y点

// RI x,y>=0

// secure way:using defensive copy and all fields are private

Rep:

Private int x;

Private int y;//横纵坐标

Constructor

Public Position(int x,int y){

This.x=x;

This.y=y;

}

Function:

Public int getX();//返回x坐标和y坐标

Public int getY();//

Public void setPosition(int x,int y)//修改坐标

1. Piece(mutable)

// AF:该棋子玩家对应gameUser，棋子类型对应type,棋子位置对应position

// RI gameType!=null,type!=null,position!=null

// secure way:using defensive copy and all fields are private

Rep:

Private final String User;//棋子拥有者

Private final String type;//棋子类型

Private Position piecePosition;//棋子位置

Constructor

Piece(String user,String type);//构造棋子

Function

Public String getType();//

Public String getUser();

Public int getX();

Public int getY();

Public void set(int x,int y);

1. Player(mutable)

// AF:该玩家m名字对应playerName，是否是当前玩家对应firstHand

// RI playerName!=null

// secure way:using defensive copy and all fields are private

Rep:

Private final String playerName;

Private Boolean firstHead;

Constructor

Player(String name){

This.playName = name;

}

Function:

String getName();//返回名字

Void setFirstHead();//设置先手，firsthand = true;

Void setSceondHead();//设置后手，firsthand = false

getFirstHead();//返回是否先手，是先手返回true，否则false

1. Game(interface)

Public static Game empty(String gameType)

public String getgameType();

public int getSize();

1. ChessGame(immutable)implements Game

// AF游戏类型对应gameType，大小对应size

// RI gameType!=null,size!=null

// secure way:using defensive copy and all fields are private

Private final int size=8;//国际象棋棋盘大小8\*8

Private final String gameType = “chess”;//国际象棋类型为chess

1. GoGame(immutable) implements Game

// AF游戏类型对应gameType，大小对应size

// RI gameType!=null,size!=null

// secure way:using defensive copy and all fields are private

Private final int size=19;//围棋棋盘大小19\*19

Private final String gameType = “go”;//为其象棋类型为go

1. Board(mutable)

//AF游戏类型对应gameType，大小对应size，pieces对应放置的棋子的二维数组，placed对应二维数组上是否有子

//RI gameType!=null,size!=null,Pieces!=null

//secure way:using defensive copy and all fields are private

Rep:

private String gameType;//游戏类型

private int size;//棋盘大小

private Piece[][] pieces;//pieces在棋盘上

private int[][] placed;//棋盘上的点是否有子

Constructor:

Board(String type,int size)//构造一个棋盘

Function

public void changePlaced(int x, int y, int type)//改变棋盘上是否有子

public void setPiece(Piece piece, int x, int y)//在Pieces[][]上放置Piece

public int[][] getPlaced()//返回Placed[][]是否有子的情况

public Piece[][] getPiece()//返回Piece[][]

public void removePiece(int x,int y)//在Pieces[][]上移除x,y点的piece

1. Action(interface)

Constructor

Public static Action empty(String game,String player\_1,String player\_2){

If(game.equals(“go”)){

Return new goAction();

}else{

Return new chessACtion();

}

}

Function:

public void setPlayer(String player\_1,String player\_2);//设置玩家

public int getSize();//获得大小

public void initalBoard();//初始化棋盘

public Board newBoard();//新建一个棋盘

public Player getFirstHeadPlayer();//得到当前的玩家

public void setsecondHead\_1();//设置后手

public void setfirstHead\_1();//设置先手

public boolean eatPiece(Player player\_1, Player player\_2, int x\_1, int y\_1, int x\_2, int y\_2);//吃子，如果不是自己的子或者自己用对方的子，输入越界都会抛出异常

public void changeFirstHead();//改变先手方

public boolean movePiece(Player player, int x\_1, int y\_1, int x\_2, int y\_2);//移动棋子，如果移动不是自己棋子，或者移动的地方已经有棋子，输入越界等都会抛出异常

public boolean removePiece(Player player, int x, int y);//提子，如果输入地方没有子或者提的不是对方的子，输入越界，都会抛出异常

public boolean placePiece(Player player, int x, int y);//放子，如果输入越界，以及该地方已经有子，都会抛出异常

public String getGameType();//得到游戏类型

public Player newPlayer(String playerName);//新建玩家

public String getPieceType(int x,int y);//设置棋子类型

public int calPieces(String player\_1);//计算玩家的棋子数量

1. ChessAction

//AF:棋盘对应Board，玩家对应Player,游戏对应Game

//RI:player\_1!=null,player\_2!=null

//secure way:using defensive copy and all fields are private

Rep:

private Game game = Game.empty("chess");//新建游戏

private final int size = game.getSize();

private Player player\_1;//玩家1

private Player player\_2;//玩家2

private Board board;//棋盘

1. GoAction

//AF:棋盘对应Board，玩家对应Player,游戏对应Game

//RI:player\_1!=null,player\_2!=null

//secure way:using defensive copy and all fields are private

Rep:

private Game game = Game.empty("go");

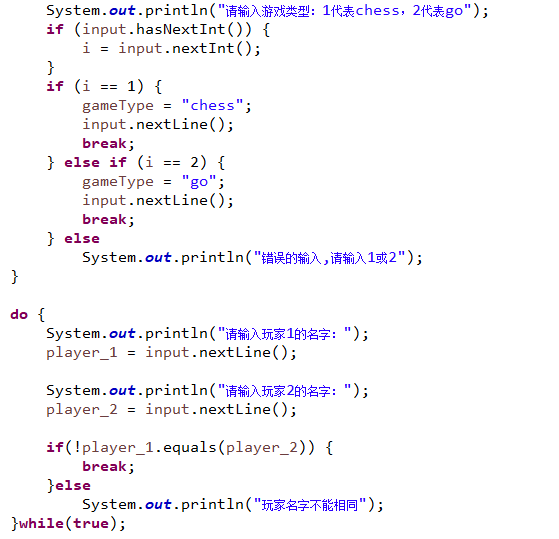
private Player player\_1;

private Player player\_2;

private Board board;

### 主程序MyChessAndGoGame设计/实现方案

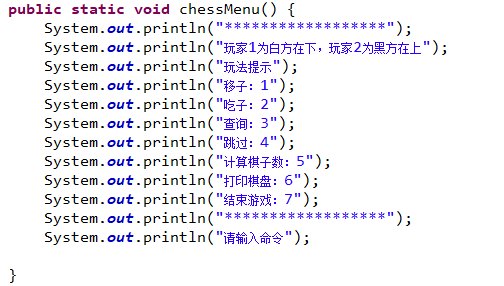
首先让用户输入1或2选择chess或者go，然后输入用户名，两个用户名不能相同



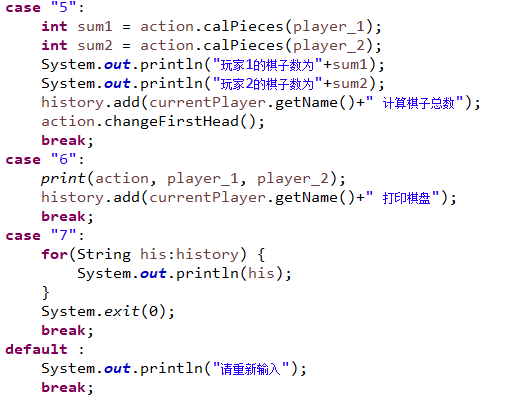
通过输入的game类型，构造相应的GameAction

在Action中有着Board棋盘变量，在Board上进行相应的操作

打印提示内容以chess为例



然后提示用户输入相应的选项，根据相应的选项选择对应的函数



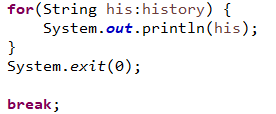
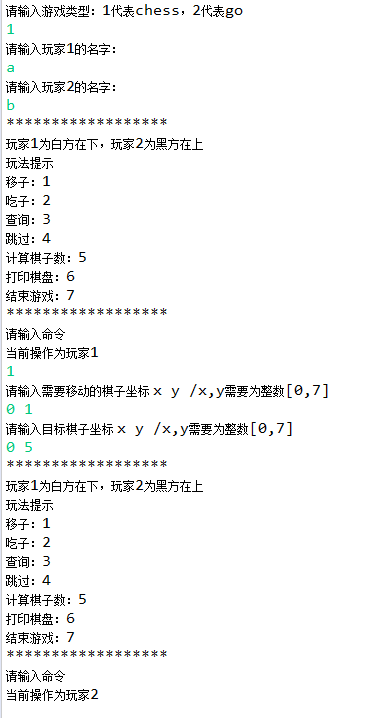
通过list<Stirng>来记录用户的操作内容

如果用户选择打印棋盘



通过双重循环来得到棋子的类型同时转化为对应汉字

最后用户输入7(end)时退出程序，并打印所有操作



以下为操作国际象棋示意

输入游戏类型

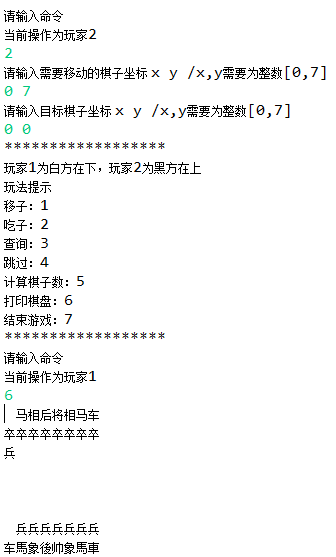
输入玩家1名字

输入玩家2名字

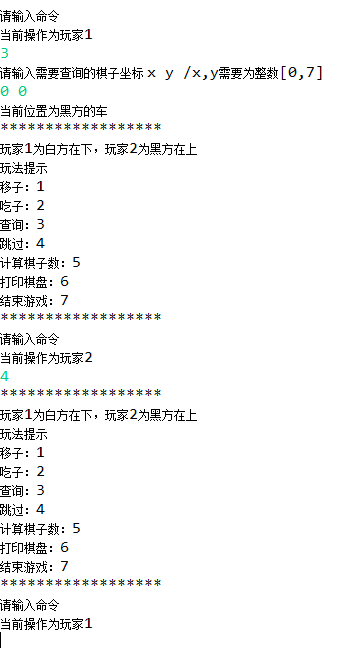
选择1进行移子操作

如果范围超过了所规定的输入，则会

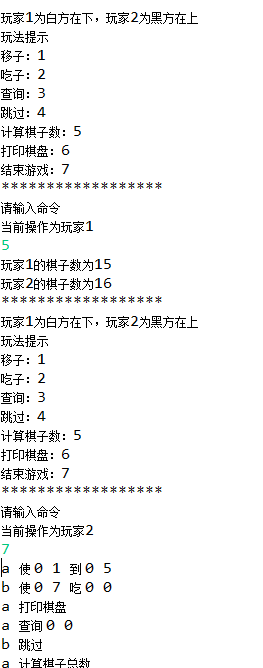
抛出异常



输入 2进行吃子操作，选择可操作的棋子



玩家2选择跳过则到玩家1进行操作



玩家2选择结束，则打印对局历史

### ADT和主程序的测试方案

1. board

testBoard()//测试构造函数

testchangePlaced()//测试能够改变二维数组place是否有子

testsetPiece();//测试设置二维数组Piece[][]，能否赋值

testgetPlaced();//测试x，y坐标能得到place上是否有子

testgetPiece();//测试得到piece[][]

testremovePiece();//测试能移除Pieces[][]x，y坐标上的piece

1. chessAction

getSize()//返回棋盘大小

testsetPlayer();//设置玩家

testgetFirstHeadPlayer();//得到当前先手的玩家

testsetsecondHead\_1();//设置玩家1为后手

testsetFirstHead\_1();//设置玩家1为先手

testEatpiece();//能否成功吃子，如果被吃的子不属于对方，输入的坐标无子，使用的不是自己的子，都会抛出异常

testchangeFirstHead();//测试改变先手

testmovePiece();//测试移动棋子，移动的不是自己的棋子，移动的棋子目标点已经有子，都会抛出异常

testremovepiece();//测试提子始终为false

testplacePiece();//测试放子始终未false

testgetGameType();//测试返回游戏类型

testgetPieceType();//测试返回x，y处的棋子类型

testcalPieces();测试该玩家的棋子总数。

1. ChessGameTest

testgetGameType();//测试chessgame的游戏类型是否为”chess”，是则返回true，否则返回false

testgetSize();//测试getsize(),能否返回8.

1. GoAction

getSize()//返回棋盘大小

testsetPlayer();//设置玩家

testgetFirstHeadPlayer();//得到当前先手的玩家

testsetsecondHead\_1();//设置玩家1为后手

testsetFirstHead\_1();//设置玩家1为先手

testEatpiece();//能否成功吃子，如果被吃的子不属于对方，输入的坐标无子，使用的不是自己的子，都会抛出异常

testchangeFirstHead();//测试始终为false

testmovePiece();//测试始终为false

testremovepiece();//测试提子，提子的子不是对方的子，目标点无子，都会抛出异常

testplacePiece();//测试放子，目标点有子，会抛出异常

testgetGameType();//测试返回游戏类型

testgetPieceType();//测试返回x，y处的棋子类型

testcalPieces();测试该玩家的棋子总数。

1. GoGame

testgetgameType()//测试能够返回游戏类型为“go”

testgetSize()//测试游戏棋盘大小是否为19

1. PieceTest.java

testPiece()//测试构造函数

testgetUser()//测试返回name

testgetX()//测试得到x坐标

testGetY()//测试得到y坐标

testSet()//测试成功设置x坐标和y坐标

1. PlayerTest.java

testPlayer();//测试构造函数是否成功

testgetName()//测试返回Name

testsetFirstHead()//测试成功设置先手

testsetSceondHead()//测试成功设置后手

testgetFirsthead()//测试是否是先手，如果是先手返回true，否则返回false

1. PositionTest.java

testPosition()//测试构造函数是否成功

testGetX()//返回position的Y坐标

testGetY()//返回position的x坐标

testSetPosition()//设置position的x和y坐标

testEquals()//重写，判断两个position是否x相等y也相等,，测试相等和不想等

# 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 | 计划任务 | 实际完成情况 |
| 4.1 | 20:00-23:00 | 完成P1的Edge图 | 延迟一小时完成 |
| 4.2 | 19:00-24:00 | 完成P1的vertices图 | 准时完成 |
| 4.4 | 18:00-23:00 | 完成Poet | 未完成 |
| 4.6 | 18:.00-24:00 | 接着完成poet和P2 | 完成 |
| 4.7 | 20:00-22:00 | 完成P3的类position | 完成 |
| 4.8 | 20:00-23:00 | 完成P3的类game | 完成 |
| 4.9 | 19:30-22:00 | 完成P3的类piece,action,board | 准时完成 |
| 4.10 | 20:00-24:00 | 完成P3的主函数 | 延迟一小时完成 |
| 4.12 | 7:00-20:00 | 完成实验报告 | 准时完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| Input.nextln()在输入时会吞行 | 查询CSDN，知道了要将\n清楚 |
| 遍历删除 | 查询百度学会了三种遍历删除的方法 |
| 抽象类的实现 | 查询百度，学会了抽象类如何实现 |

# 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

## 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？

面向ADT编程是将各种类型看作抽象类和抽象方法，然后实例化进行完成，面向过程编程是直接一步一步完成编程，没有抽象的思想。

1. 使用泛型和不使用泛型的编程，对你来说有何差异？

适用泛型编程，能够在其他代码里复用，降低了书写难度

1. 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？

能够有效约束客户端的输入，在渐渐适应过程中

1. 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？

能够减少书写的代码量，同时容易复用，减少错误发生

1. 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？

感觉有写ADT之间的内容重复了，没有增加ADT数量的必要

1. 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？

这些工作的意义保证了封装的安全性，不会出现表示泄露的问题，以后坚持这么做

1. 关于本实验的工作量、难度、deadline。

本实验的工作量巨大，难度适中，deadline非常紧，写spec，AF，RI占用了很多时间

1. 《软件构造》课程进展到目前，你对该课程有何体会和建议？

该课程需要多写代码来熟练掌握老师所的内容