

**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 | 罗瑞欣 |
| 学号 | 1170300821 |
| 班号 | 1703008 |
| 电子邮件 | 3102595709@qq.com |
| 手机号码 | 15048260039 |

**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

Windos10专业版，eclipse，jdk-9.01，Junit5.

GitHub Lab2仓库的URL地址：

git@github.com:ComputerScienceHIT/Lab2-1170300821.git

# 实验过程

## Poetic Walks

分别新建两个类ConcreteEdgesGraph，ConcreteVerticesGraph 实现Graph接口。

Graph接口要求实现add（添加新节点），set（添加新边），remove（移除节点），vertices（获得所有的节点集合），sources（target）获得以target为目标节点的边的起始节点，targes(source)获得以source为起始节点的边的目标节点。

Poet：给定一组单词（文件输入），对于两个相邻的单词a和b，认为存在一条由a到b的有向边，通过Graph接口构造有向图。再给定一由单词组成的句子，如果句子中两个相邻单词之间在Graph图中有一个中间单词则将中间单词插入到两单词之间（如果有多个则插入权重最大的那个）。

### Get the code and prepare Git repository

$ Git init

$ Git remote add origin git@github.com:ComputerScienceHIT/Lab2-1170300821.git

$ Git pull origin master

$ Git add .

$ Git commit -m “init”

$ Git push origin master

### Problem 1: Test Graph <String>

修改empty为：

Public Graph<String> empty() {

return new ConcreteEdgesGraph();

}。

### Problem 2: Implement Graph <String>

#### Implement ConcreteEdgesGraph

|  |  |
| --- | --- |
| 函数 | 实现思路 |
| public boolean add(L vertex) | 调用List.add，添加成功返回true，list中已有返回false |
| public int set(L source, L target, int weight) | 检查输入满足weight>=0。当weight<0时，如果当前没有该edge，则直接添加，返回值为0，如果已经有该edge，则记录旧值，修改为新值。如果weight==0，当删除的边不存在时返回0，当存在时，删除该节点，该边，如果此时两端点之中没有边与之相连了则删除节点。 |
| public boolean remove(L vertex) | 首先检查vertex是否在vertices内。若存在，则遍历检查有没有与之相连的边，全部删除。 |
| public Set<L> vertices() | 返回vertices |
| public Map<L, Integer> sources(L target) | 在edges中寻找所有目标点是target的初始节点 |
| public Map<L, Integer> targets(L source) | 在edges中寻找所有目标点是sourse的初始节点 |

Edge<L>类

|  |  |
| --- | --- |
| 函数 | 实现思路 |
| private void checkRep() | 检查weight>0 |
| public String toString() |  |

#### Implement ConcreteVerticesGraph

ConcreteVerticesGraph<L>类：

|  |  |
| --- | --- |
| 函数 | 实现思路 |
| public boolean add(L vertex) | 调用List.add，添加成功返回true，list中已有返回false |
| public int set(L source, L target, int weight) | 检查输入满足weight>=0。当weight<0时，如果当前没有该edge，则直接添加，返回值为0，如果已经有该edge，则记录旧值，修改为新值。如果weight==0，当删除的边不存在时返回0，当存在时，删除该节点，该边，如果此时两端点之中没有边与之相连了则删除节点。 |
| public boolean remove(L vertex) | 去除节点和相连的边。节点存在返回true，不存在返回false |
| public Set<L> vertices() | 返回每个vertex的lable |
| public Map<L, Integer> sources(L target) | 调用result.put(temp.getLabel(), temp.getTargetmap().get(target)) |
| public Map<L, Integer> targets(L source) | 调用result.putAll(temp.getTargetmap()) |

Vertex<L>类

|  |  |
| --- | --- |
| 函数 | 实现思路 |
| public boolean equals(Object obj) | 判断相等 |
| public int hashCode() | 生成hash值 |
| public void addanedge(L target, int weight) | Setter |
| public void removeanedge(L target) | Setter |
| private void setLabel(L label) | Setter |

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

更改类名：

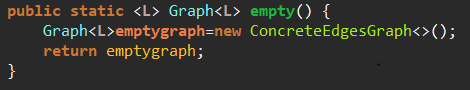
public class ConcreteEdgesGraph<L> implements Graph<L>

public class ConcreteVerticesGraph<L> implements Graph<L>

并修改相关参数为泛型L

#### Make the implementations generic

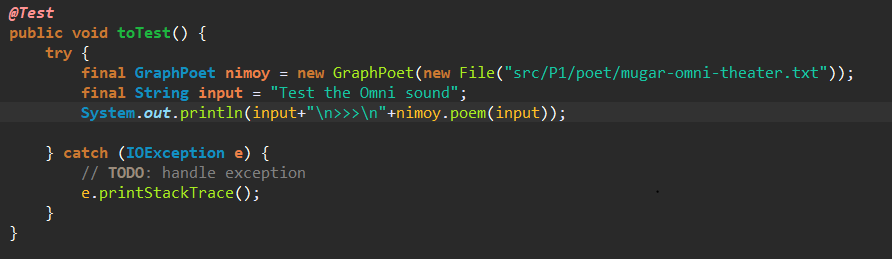
#### Implement Graph.empty()



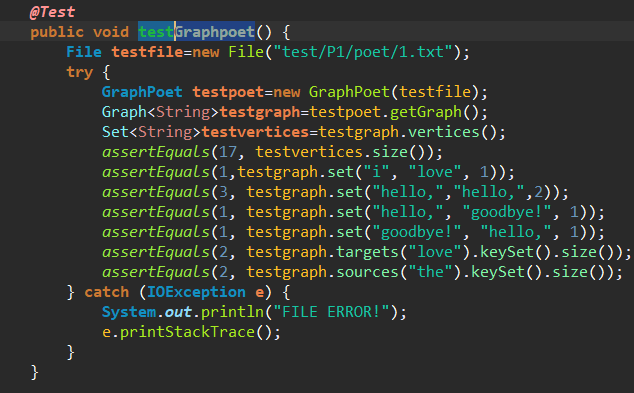
### Problem 4: Poetic walks

#### Test GraphPoet

测试样例：



测试Graphpoet类：



测试poet类：



#### Implement GraphPoet

利用empty构建新的graph

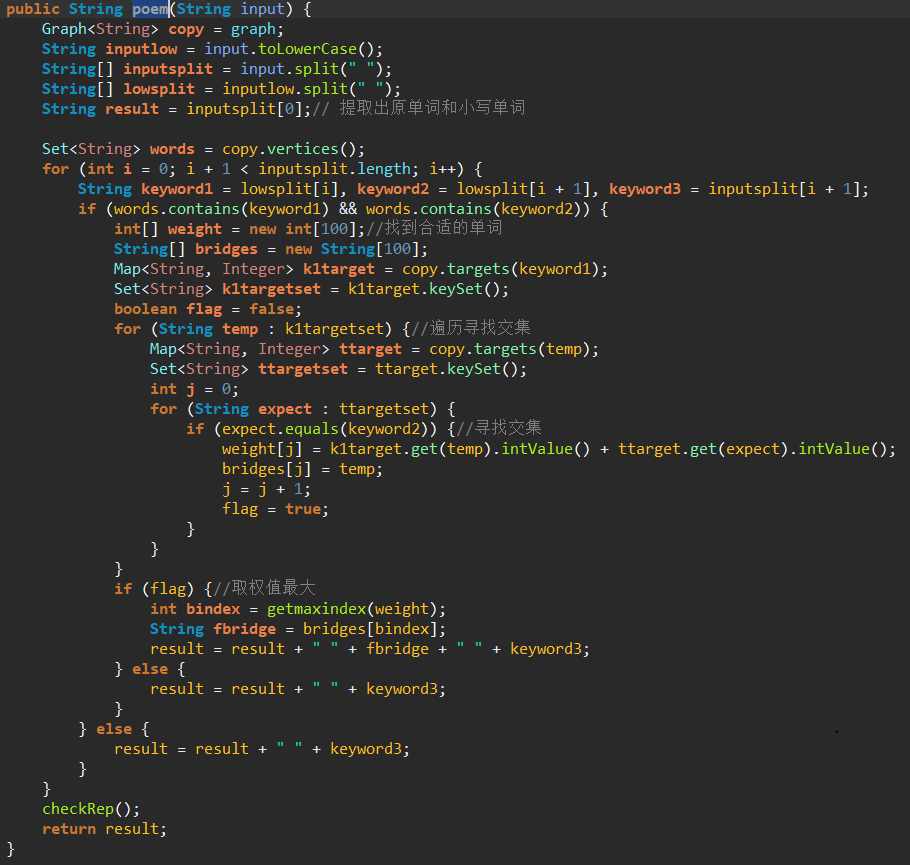


利用split按照” ”提取出每个单词，根据单词的重复次数设置weight，添加到graph中。

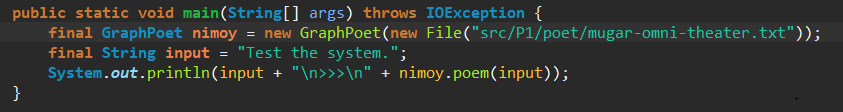


Poem类：

首先将input切割为单词序列inputWords，对于每两个相邻的节点v1和v2，如果v1.targets()与v2.sources()存在交集，则说明两个单词之间可以添加一个bridge，自交集中选择去权重最大的一个添加到两单词之间。



#### Graph poetry slam



### Before you’re done

请按照[http://web.mit.edu/6.031/www/sp17/psets/ps2/#before\_youre\_done](http://web.mit.edu/6.031/www/sp17/psets/ps2/" \l "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。

$ Git add .

$ Git commit -m “x”

$ Git 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

继承P1中ConcreteEdgesGraph<Person>或者ConcreteVerticesGraph<Person>类 实现FriendshipGraph，通过基本操作实现FriendshipGraph中addVertex，addEdge和getDistance三个接口。

### FriendshipGraph类

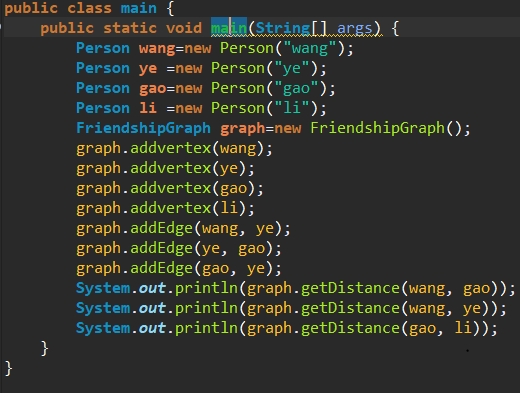
|  |  |
| --- | --- |
| 方法 | 实现思路 |
| public void addEdge(Person xperson, Person yperson) | 检查两个person是否存在，若存在则调用set |
| public void addvertex(Person vertex) | 调用add |
| public int getDistance(Person xperson, Person yperson) | 检查两个person是否存在，若存在使用BFS算法求得图中stPerson与edPerson之间的最短距离 |
| public int getDistance(Person xperson, Person yperson) | 调用targets遍历 |

### Person类

|  |  |
| --- | --- |
| private String name; | 人名 |
| Getter |  |
| Setter |  |

### 客户端main()

使用Lab1提供的样例



### 测试用例

同main。

### 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。

$ Git add .

$ Git commit -m “x”

$ Git 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

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。

MyExp类

自定义异常类，继承自Exception，提供的接口有：

|  |  |
| --- | --- |
| Public String getExpMsg | 获得expMsg |
| Public void setExpMsg | 设置expMsg |
| @Override public String toString | 根据expMsg构造错误信息 |
| Public static void assertTrue(Boolean cond,String msg) throws MyExp | 检查cond是否正确，如果为false则抛出MyExp |

Piece类

域：

|  |  |
| --- | --- |
| pieceState | 棋子状态，0为未放置，1为已经放置，2为放置之后被拿出棋盘且不可用 |
| pName | 棋子种类名称 |
| Px,py | 棋子在棋盘中所处的坐标 |

接口：

|  |  |
| --- | --- |
| Getter |  |
| Setter |  |
| public void removeFromBoard() | 将该棋子从棋盘中移出 |

Player类

域：

|  |  |
| --- | --- |
| private String playerName | 玩家名称 |
| private StringBuilder gameHistory | 玩家操作历史 |
| private Set<Piece> playerPieces | 玩家棋子集合 |

接口：

|  |  |
| --- | --- |
| Getter + setter |  |
| public Boolean addPiece(Piece piece) | 若不存在piece，则向playerPieces加入piece |
| public Boolean isContainPiece(Piece piece) | 判断该玩家是否包含指定棋子piece |
| public void addHistory(String gameStep) | 向操作历史中添加一步的操作（字符串形式） |
| public boolean movePiece(int stX, int stY, int edX, int edY) | 将已经处于棋盘上的位于st的棋子移动到空地址ed |
| public Piece getPiece(int x, int y) | 返回棋盘上(x,y)处的piece，若不存在则返回null |
| public Piece getAnyPieceByFilter(Predicate<? super Piece> predicate) | 获取该玩家所有满足功能函数predicate要求的棋子的任意一个。 |

Board类：

Board为从左下角（1,1）开始的矩阵

域：

|  |  |
| --- | --- |
| Private int boardType | 棋盘类型，0为放在格子里，1为放在交点上 |
| Private int boardSize | 棋盘大小，指的是棋盘上行或列所有的格子数目 |
| Private Piece boardPieces[][] | 存放棋盘上对应位置所放的棋子 |

接口：

|  |  |
| --- | --- |
| Getter + setter |  |
| public Piece getPieceAtCord(int px, int py) | 获取处于(px,py)位置的棋子，如果位置不合法抛出MyExp |
| public void setPieceAtCord(int px, int py, Piece piece) | 将棋子piece放置在棋盘的(px,py)位置处。如果位置不合法则抛出MyExp |
| public boolean isCordAvailable(int cx, int cy) | 判断坐标(cx,cy)是否是一个合法坐标 |
| public boolean isPieceInBoard(Piece piece) | 判断棋子piece是否处于棋盘之内。 |
| public void removePiece(int x, int y) | 去除(x,y)处的点 |

Action类：

域：

|  |  |
| --- | --- |
| private Board gameBoard | 游戏中棋盘对象的引用 |
| private Player playerA, playerB | 游戏中玩家对象AB的引用 |

接口：

|  |  |
| --- | --- |
| public void putPiece(Player player, Piece piece, int x, int y) | 将玩家player的未处于棋盘的piece棋子落到(x,y)处 |
| public void movePiece(Player player, int stX, int stY, int edX, int edY) | 将玩家player的已经处于棋盘上的位于st的棋子移动到空地址ed |
| public void removePiece(Player player, int x, int y) | 将用户player的位于棋盘上（x，y）的棋子移出棋盘 |
| public void eatPiece(Player player, int stX, int stY, int edX, int edY) | 使用用户player的位于棋盘st位置的棋子吃掉到对手的ed位置的棋子 |

Game类：

域：

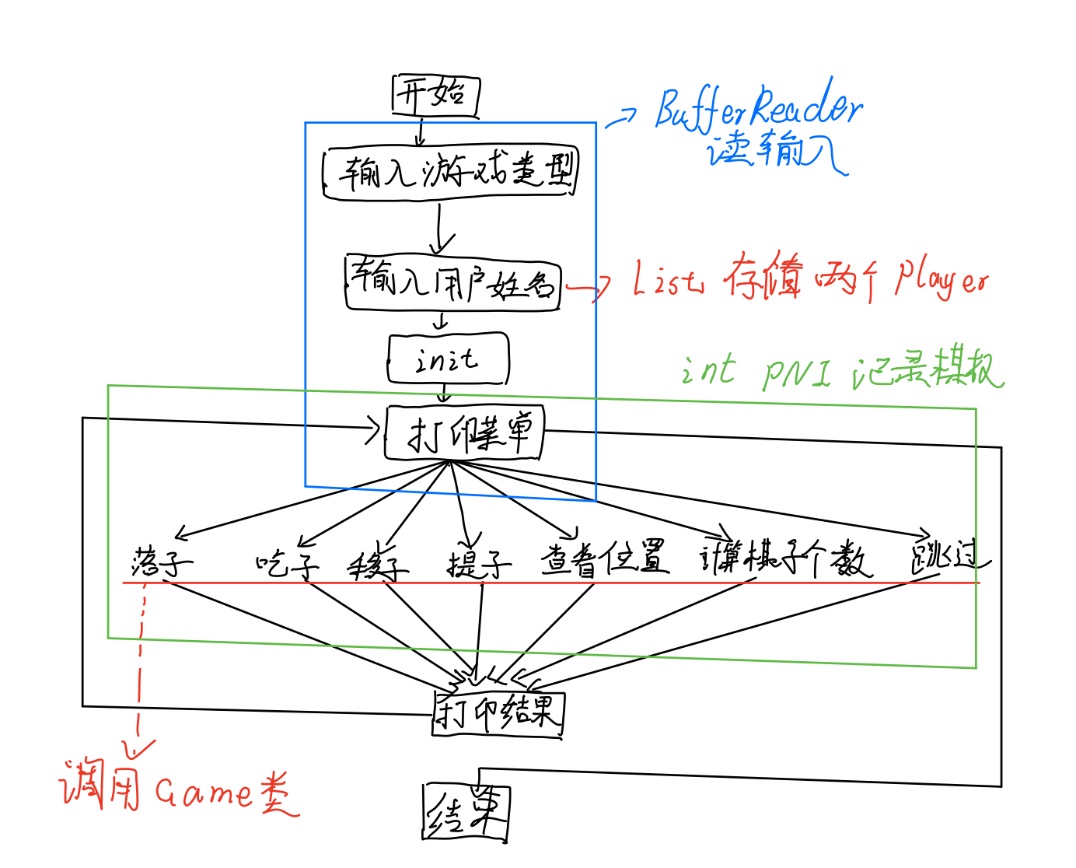
|  |  |
| --- | --- |
| private String gameType | 游戏类型 |
| private Board gameBoard | 游戏棋盘 |
| private Player PlayerA, PlayerB | 游戏玩家，playerA为先手 |
| private Action gameAction | 游戏动作 |

接口：

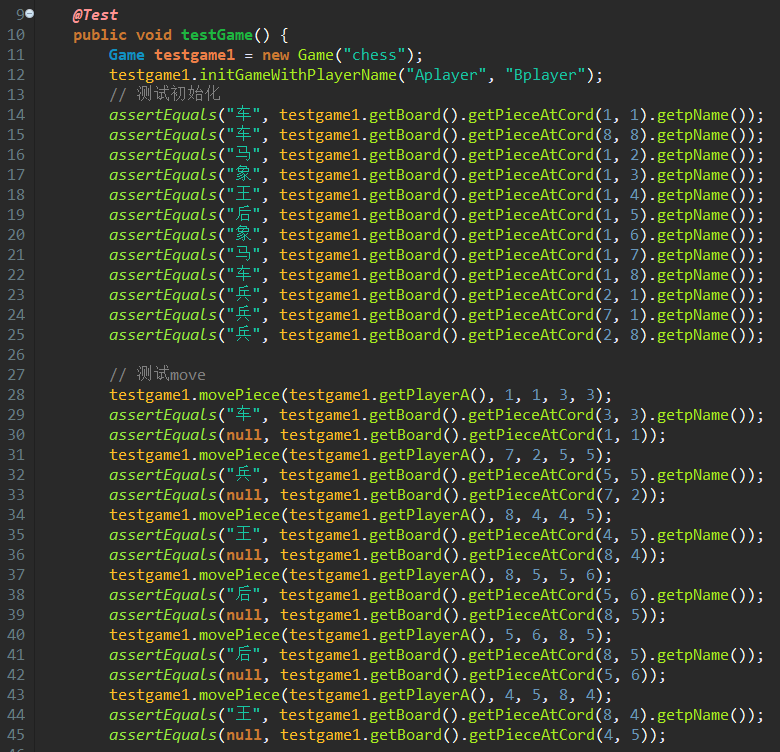
|  |  |
| --- | --- |
| public void iniGameWithPlayerName(String paName, String pbName) | 通过传入的两个玩家的名字初始化Game中的各类对象 |
| public void putPiece(Player player, Piece piece, int x, int y) | 将玩家player的未处于棋盘的piece棋子落到(x,y)处 |
| public void movePiece(Player player, int stX, int stY, int edX, int edY) | 将玩家player的已经处于棋盘上的位于st的棋子移动到空地址ed |
| public void removePiece(Player player, int x, int y) | 将用户player的位于棋盘上(x,y)的棋子移出棋盘 |
| public void eatPiece(Player player, int stX, int stY, int edX, int edY) | 使用用户player的位于棋盘st位置的棋子吃掉到对手的ed位置的棋子 |
| public Player getOwnerAtCord(int x, int y) | 获得处于(x,y)位置的棋子的所有者 |
| public Piece getPieceAtCord(int x, int y) | 获取处于(x,y)位置的棋子piece，如果没有棋子则返回null |
| public int getNumOfPlayerPiecesInBoard(Player player) | 获取用户在棋盘上的所有棋子数目 |

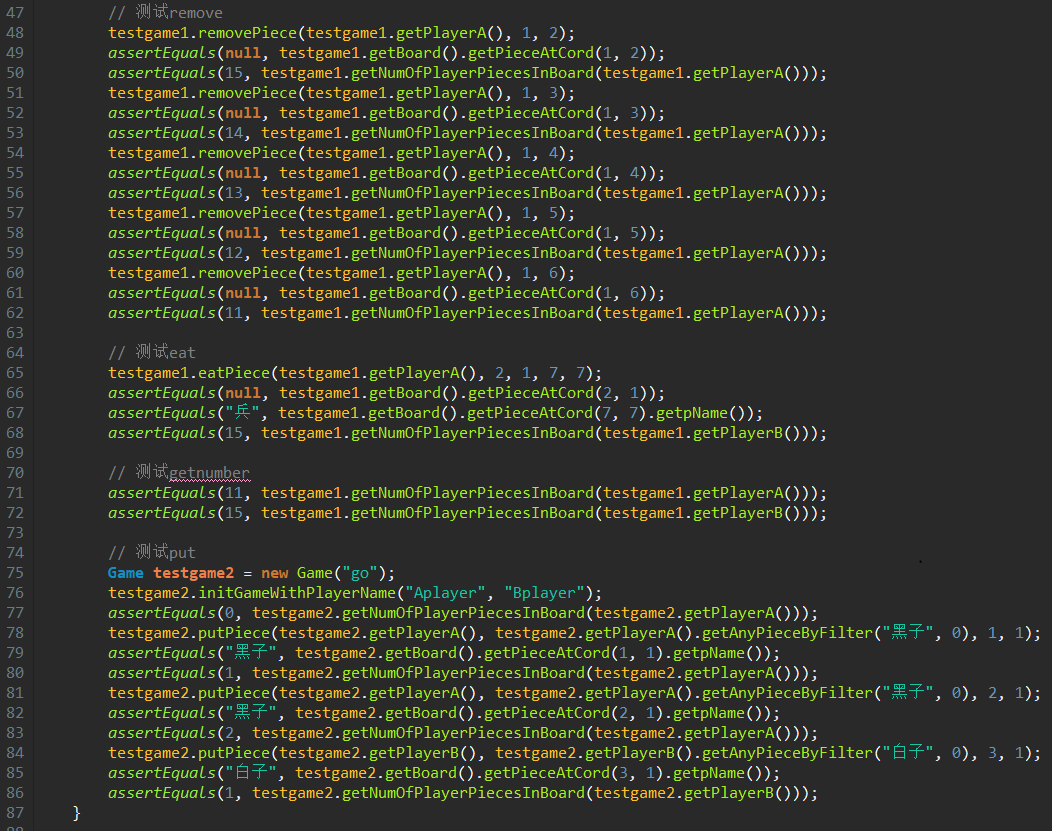
### 主程序MyChessAndGoGame设计/实现方案

如图所示：



### ADT和主程序的测试方案





包含测试board、action和player



## 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 |
| 3.18 | 15:30-22:30 | P1的test | 写完 |
| 3.19 | 19:00-22:30 | 写完ConcreteEdgesGraph | 写完 |
| 3.20 | 19:00-22:30 | 写完ConcreteVerticesGraph | 完成 |
| 3.24-3.29 |  | 电脑dcin充电口损坏，淘宝邮寄原厂二手件 | （居然只要一家卖） |
| 3.29 | 15:00-16:00 | 拆电脑，拆屏线，换配件 | Xps的排线真好。。 |
| 3.30 | 13:00-22:00 | P2+P3的myexp、piece和board类 | 完成 |
| 3.31 | 10:00-17:00 | P3的player和action类，以及部分game类 | 完成 |
| 3.31 | 16:00-24:30 | Eclipse崩了，P3之前的四个类直接变成初始化状态。重写myexp、piece、board、player、action和game类 | 写完了，但是很蛋疼 |
| 4.1 | 19:00-22:30 | 改进之前的六个类，重写才发现原来写的真烂。。。 | 改的不那么烂了 |
| 4.2 | 19:00-22:30 | 主函数 | 基本写完 |
| 4.3 | 19:00-21:30 | 调试主函数，写一部分test | 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 |
| 关于自定义异常类型的使用 | Google+问舍友 |
| 关于stream的使用 | Google+问舍友 |
| 关于泛型的具体使用规则 | Google，会用一点点，但是不是非常理解原理 |

# 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

每次写完一个类就push一下，想起了就push一下，eclipse崩了重写太坑爹了

## 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？

面向ADT，出错几率更小，更容易测试debug，编程更具有目的性

1. 使用泛型和不使用泛型的编程，对你来说有何差异？

代码更容易改进

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？

更具有目的性，不易出错

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？

方便，节省了一定的工作量

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？

非常非常有层次感，用层次上的“空间”换得了任务复杂度的降低。对于go难度变大，对于chess更简单，而且可以有很大的拓展性

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？

规范类的调用和继承关系

1. 关于本实验的工作量、难度、deadline。

可能是我比较有毒吧，做完P1电脑坏了，维修一周，P3做完一半eclipse崩了，重写P3。

1. 《软件构造》课程进展到目前，你对该课程有何体会和建议？