

**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 | 夏铭远 |
| 学号 | 1173710209 |
| 班号 | 1737102 |
| 电子邮件 | 462272701@qq.com |
| 手机号码 | 13029878656 |

**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

在Eclipse中新建项目Lab2\_1173710209，首先搭建好目录结构，再将项目与Github仓库连接并push到仓库里。

<https://github.com/ComputerScienceHIT/Lab2-1173710209.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

该任务首先要设计Graph这个ADT，并且要给出两种设计方式，针对边和针对节点，同时还要实现泛型。之后再复用设计出的Graph去实现GraphPoet。

重在理解接口和泛型。

### Get the code and prepare Git repository

1. 首先从老师的仓库下载代码，导入已建好的项目的相应位置。
2. 右键项目🡪team🡪share project🡪建立本地仓库并绑定远程仓库。
3. Stage🡪commit and push

### Problem 1: Test Graph <String>

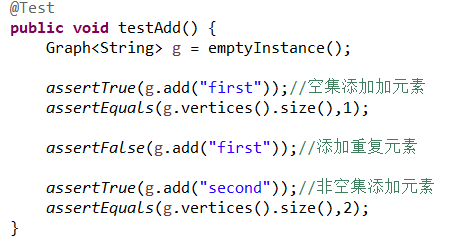
设计Graph接口的测试用例

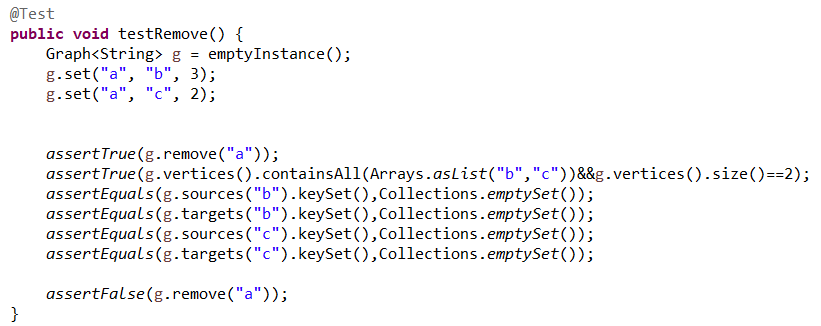
1. GraphStaticTest

先不动

1. GraphInstanceTest

主要测试add,set,remove方法，其它方法在set中带过，具体情况已在测试函数中讨论

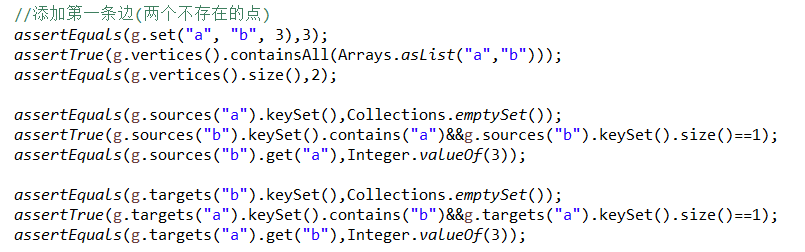
测试add方法



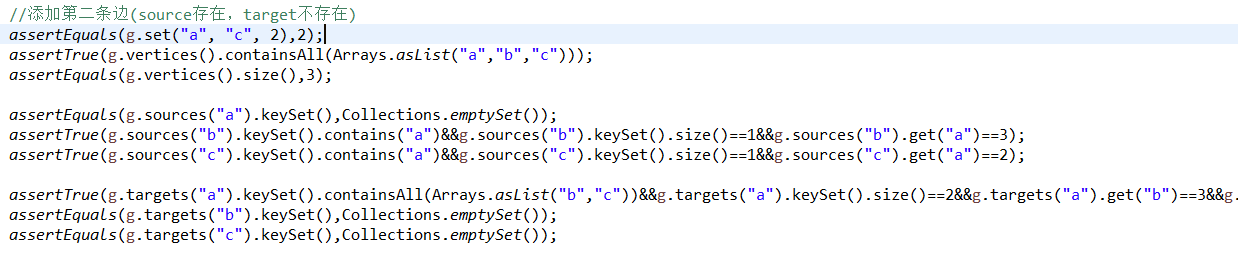
测试remove方法

测试Set方法时从5种情况入手

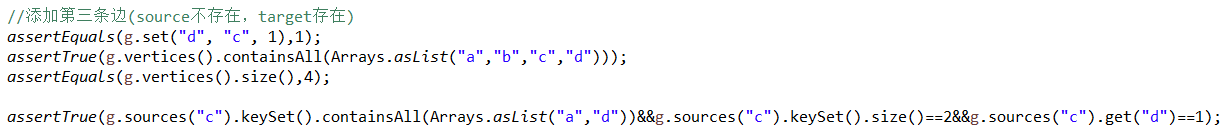
1. 参数是两个不存在的点



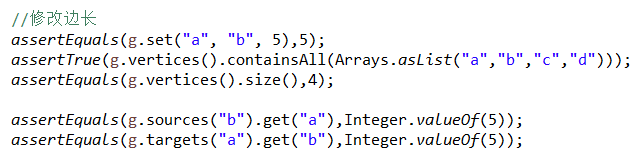
1. source存在，target不存在



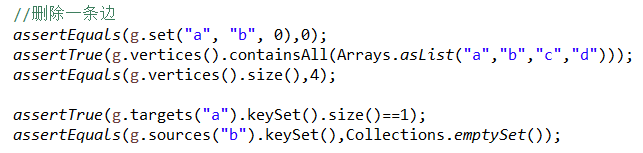
1. source不存在，target存在



1. 参数都存在



1. Weight为0

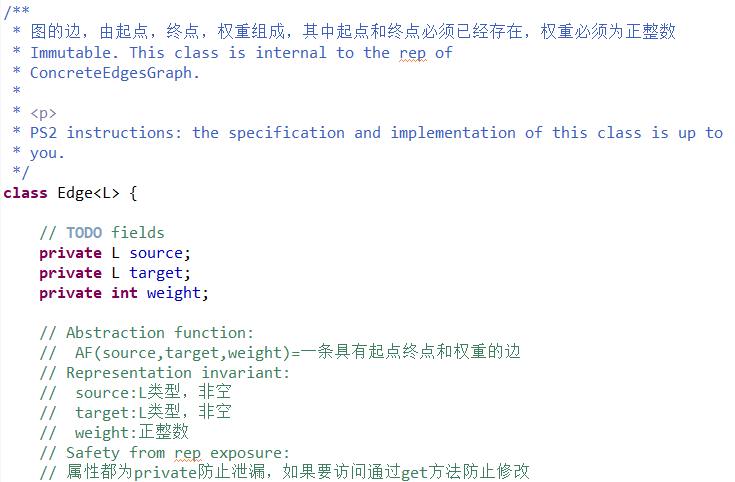


### Problem 2: Implement Graph <String>

#### Implement ConcreteEdgesGraph

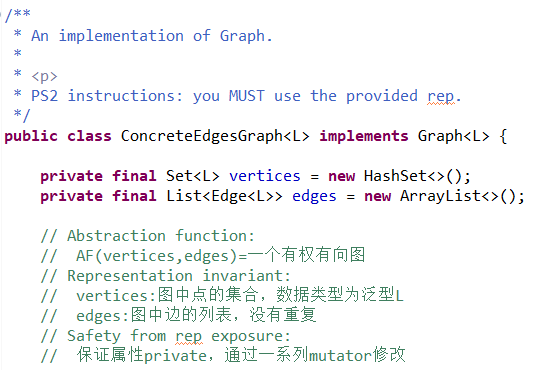
设计一个实现Graph接口的基于边的图。

首先设计边的类Edge



有get和toString等方法。

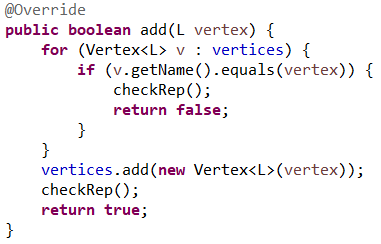
再基于此类设计图



按照给定的rep去重写Graph接口中的方法

1. Add

遍历vertices变量，如果不存在传入的参数则在vertices中添加此参数并返回true，反之什么都不做并返回false



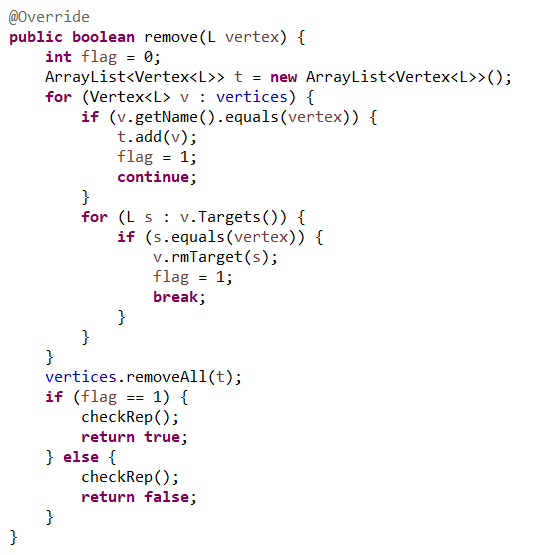
1. Set

复用上面的add方法，添加两个节点参数（直接过滤掉重复等情况）。然后遍历edges，找到输入的边就修改其值，如果weight是0就直接删除



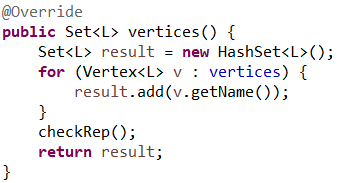
1. Remove

先判断参数点是否在图中，若不在返回false，若存在则遍历edges，找到所有含有参数节点的边并删除，随后删除该节点。



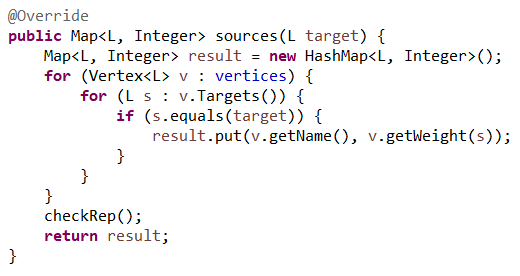
1. Vertices

建立一个Set保存节点，遍历vertices变量全部添加到Set中，最后返回Set



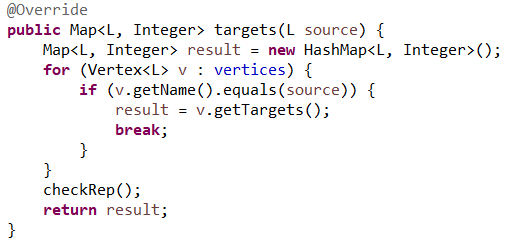
1. Sources

将所有以参数节点为target的边的source以key的形式加入map，并把边的weight以value加入，最后返回map



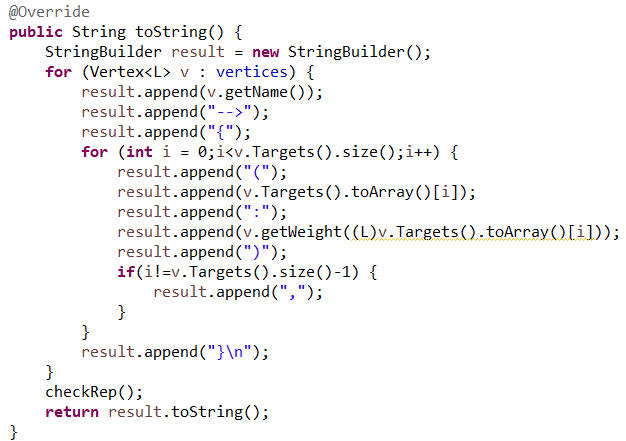
1. Targets

将所有以节点参数为source的边的target以key的形式加入map，并把边的weight以value加入，最后返回map



1. toString

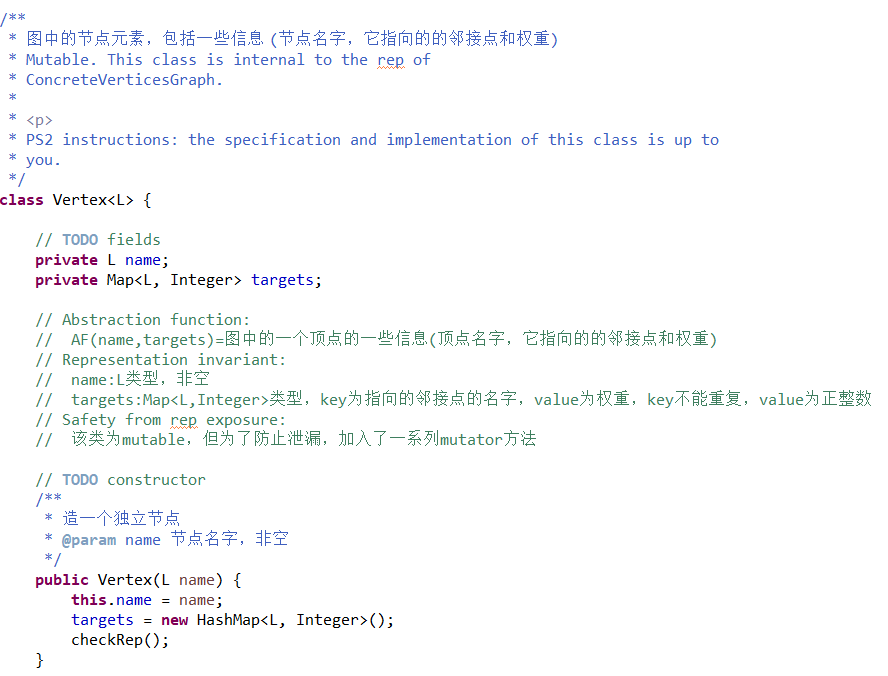
把图以邻接表的形式表示出来便于查看



#### Implement ConcreteVerticesGraph

和上一部分相同，这一部分基于节点设计图

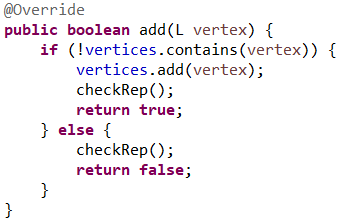
先设计节点类



按照给定的rep去重写Graph接口中的方法

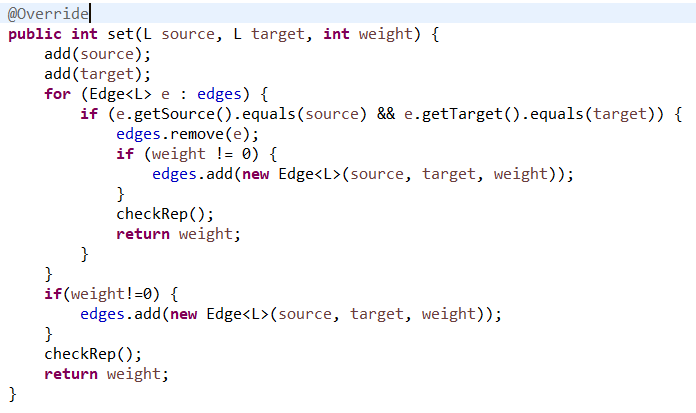
1.Add

遍历vertices变量，如果不存在传入的参数则在vertices中添加此参数并返回true，反之什么都不做并返回false



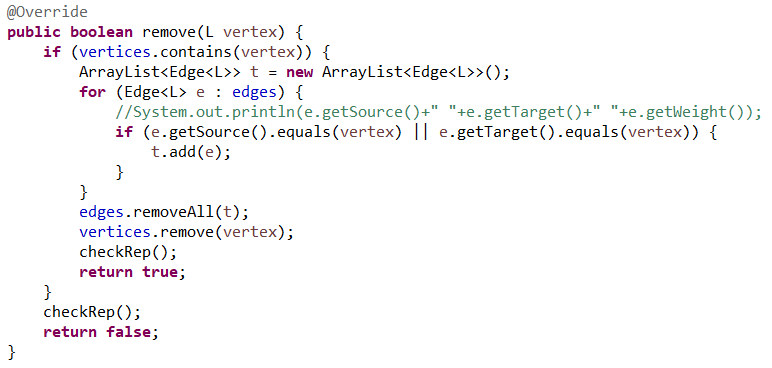
2.Set

复用上面的add方法，添加两个节点参数（直接过滤掉重复等情况）。然后遍历edges，找到输入的边就修改其值，如果weight是0就直接删除



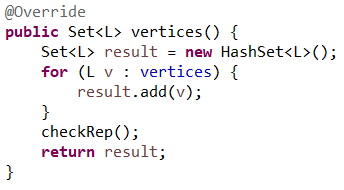
3.Remove

先判断参数点是否在图中，若不在返回false，若存在则遍历edges，找到所有含有参数节点的边并删除，随后删除该节点。



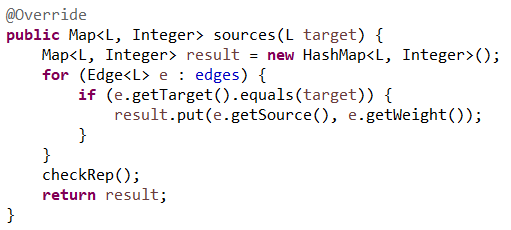
4.Vertices

建立一个Set保存节点，遍历vertices变量全部添加到Set中，最后返回Set



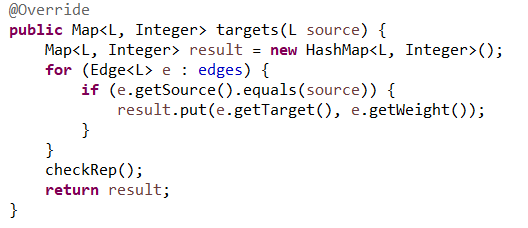
5.Sources

将所有以参数节点为target的边的source以key的形式加入map，并把边的weight以value加入，最后返回map



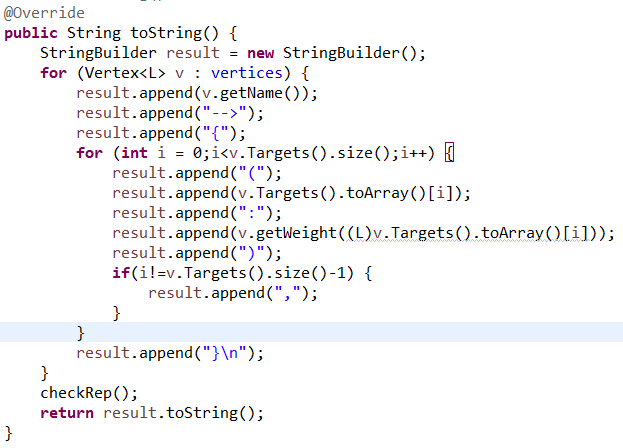
6.Targets

将所有以节点参数为source的边的target以key的形式加入map，并把边的weight以value加入，最后返回map



7.toString

把图以邻接表的形式表示出来便于查看



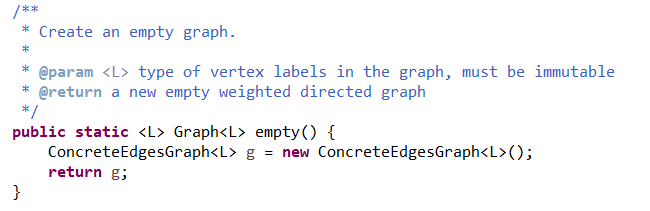
### Problem 3: Implement generic Graph<L>

#### Make the implementations generic

之前实现的是<String>，实现泛型只需将所有的String类型的标签改写为L即可

#### Implement Graph.empty()

创建一个空的子类对象即可，这里用的是基于边的图



### Problem 4: Poetic walks

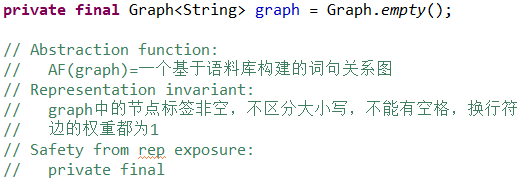
#### Test GraphPoet

测试了几个例子



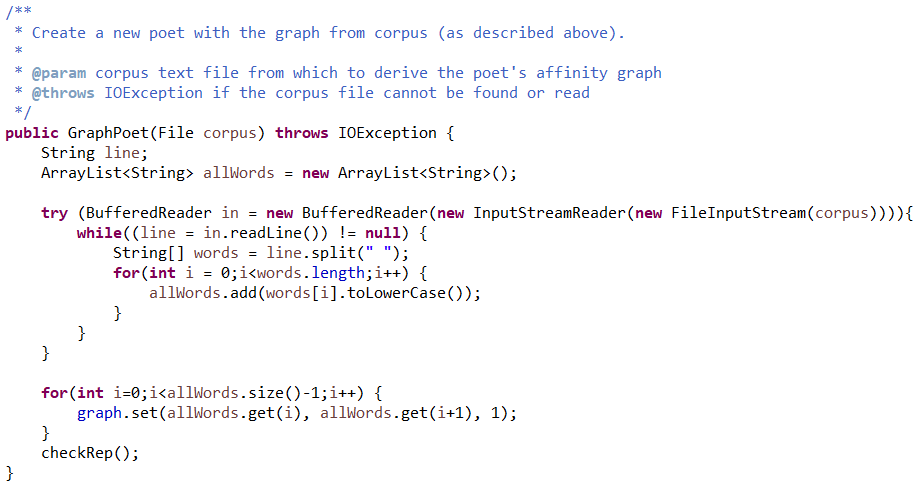
#### Implement GraphPoet

只有一个graph属性，是一个Graph对象



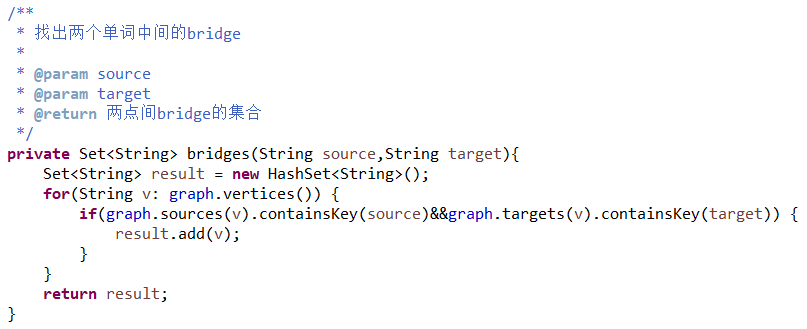
构造器

从文件参数corpus中读取单词加入graph中，按照规则形成GraphPoet

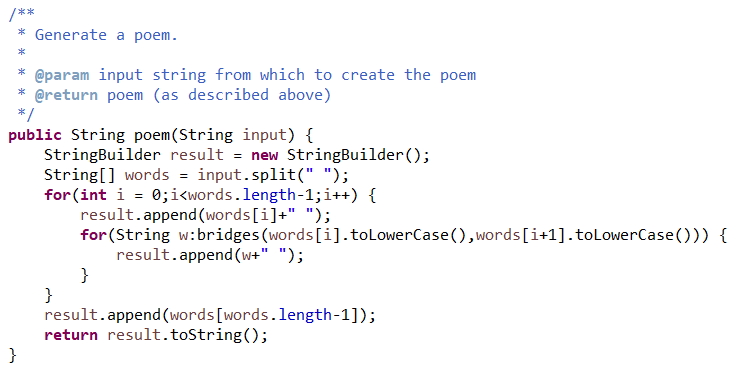




遍历graph中的节点，如果其sources中包含参数source并且targets中包含参数target,则在result中添加这个节点，最后返回result

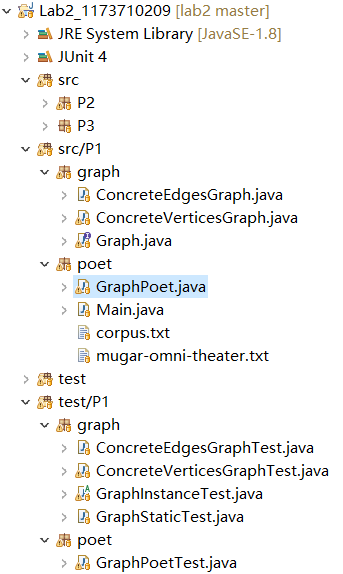


判断input中相邻的单词在graph中是否有bridge，如果有就全部加在中间，返回修改后的字符串



#### Graph poetry slam

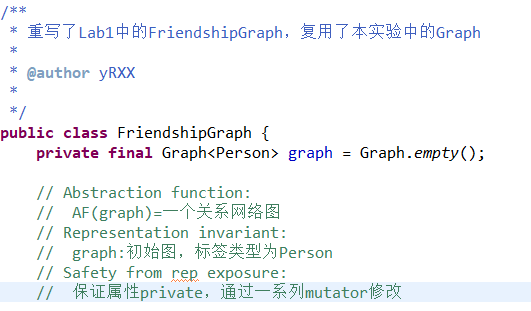
### Before you’re done



## Re-implement the Social Network in Lab1

该任务复用P1中的Graph，实现Lab1中的SocialNetwork，重在理解实现一个接口的优点。

### FriendshipGraph类



addVertex

复用graph中的add方法，再加入一些条件判断



addEdge

复用graph中的Set方法，再加入一些条件判断



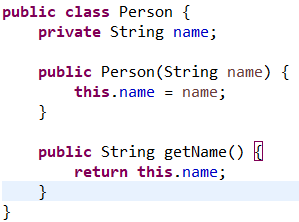
getDistance

同样用BFS



### Person类

与Lab1不同，这里Person类只有name一个属性，想要的功能用graph都可以实现了

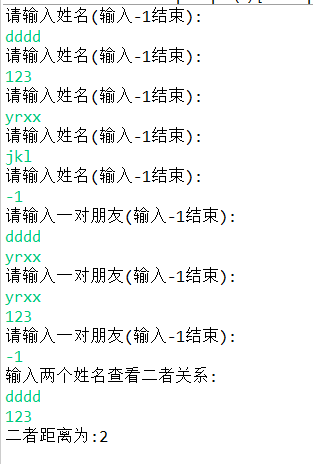


### 客户端main()

用户先循环输入节点名字，如果输入-1则结束

然后输入朋友关系，输入-1结束

然后可以查询朋友关系



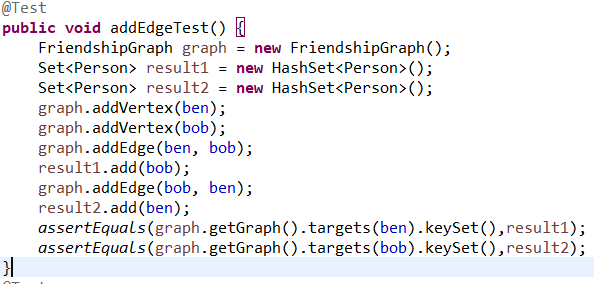
### 测试用例

testAddVertex

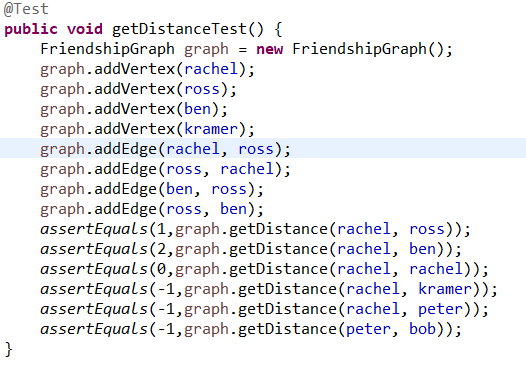
直接比较结果



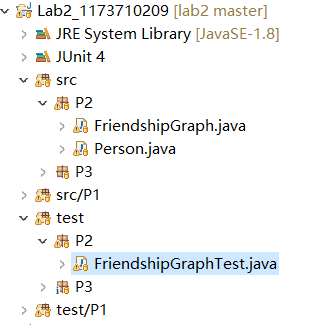
testAddEdge



testGetDistance

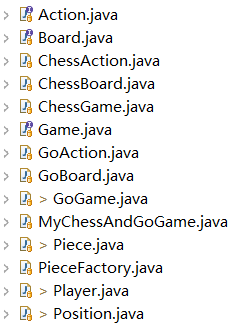


### 提交至Git仓库

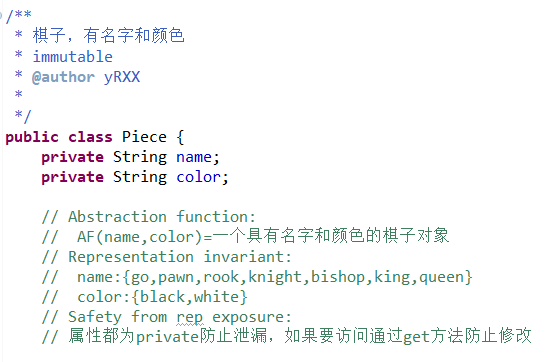
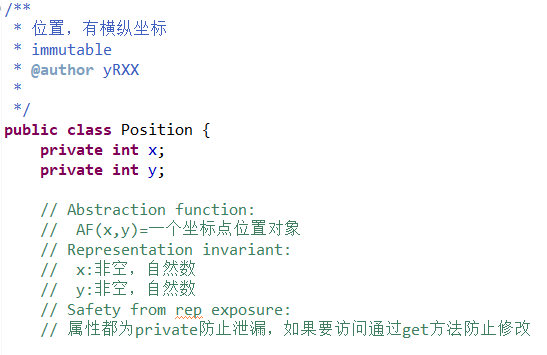


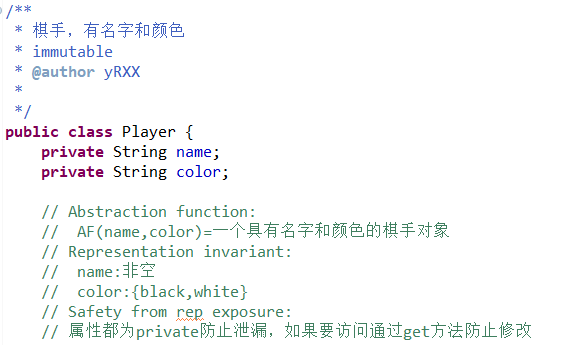
## Playing Chess

### ADT设计/实现方案



最基本的类：Piece，Position，Player

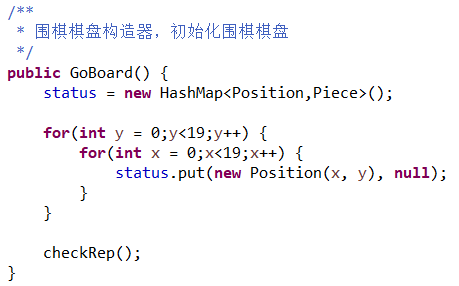


其它类都建立在此基础上

棋盘：GoBoard和ChessBoard都实现了Board接口



在二者的构造器中针对棋盘不同的规格进行初始化，并只有一个Map类型的变量status保存棋盘上的棋子分布情况





棋手动作：GoAction和ChessAction实现Action接口





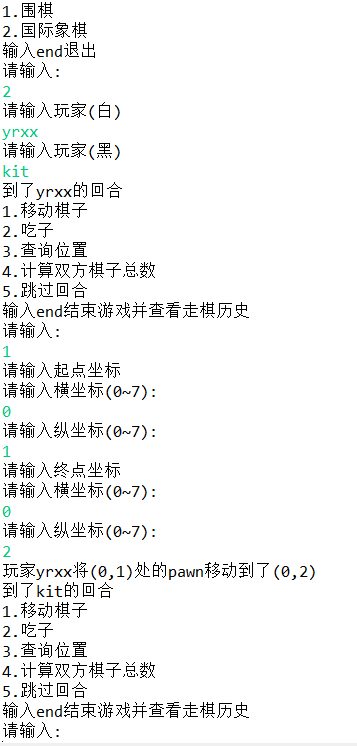
二者都以棋盘为参数，重写各自的方法

游戏：GoGame和ChessGame实现Game接口，包含游戏的主流程



### 主程序MyChessAndGoGame设计/实现方案

主程序通过调用Start方法启动游戏，Start方法中会根据用户输入选择性启动GoGame或ChessGame



### ADT和主程序的测试方案

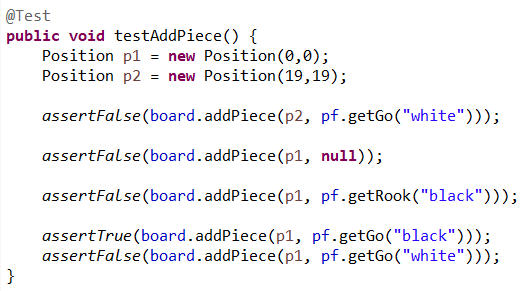
对于Piece,Position,Player的测试

主要测试它们在构造时的非法参数问题

GoBoardTest

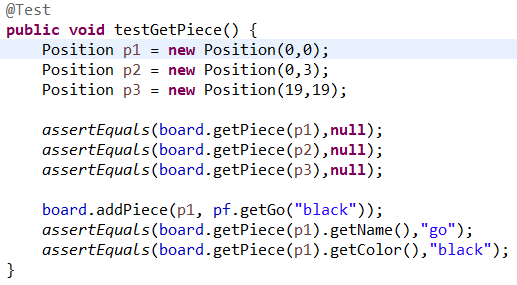
1. testAddPiece

测试在棋盘上的点，不在棋盘上的点，位置上已有棋子的几种情况



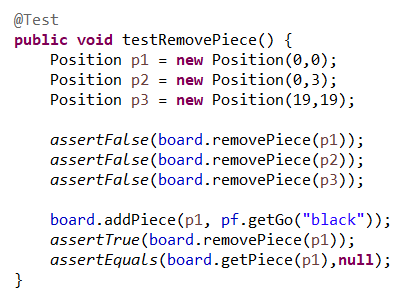
1. testGetPiece

测试在棋盘上的点，不在棋盘上的点，位置上有无棋子几种情况



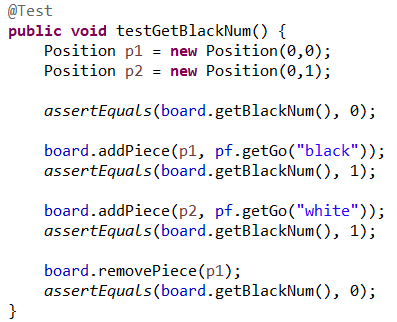
1. testRemovePiece

测试在棋盘上的点，不在棋盘上的点，位置上有无棋子几种情况



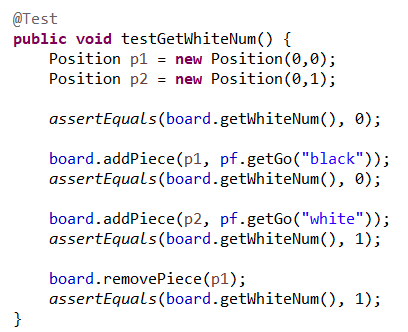
1. testGetBlackNum

做出一组动作，判断棋盘上的棋子数是否符合预期



1. testGetWhiteNum

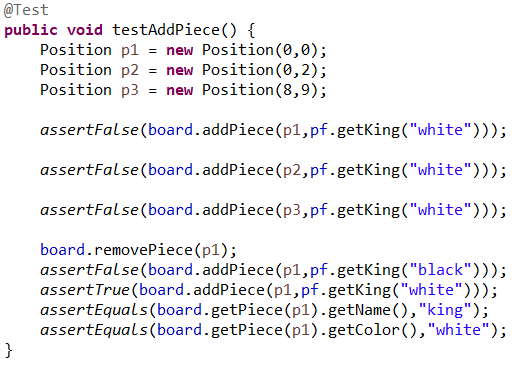
做出一组动作，判断棋盘上的棋子数是否符合预期



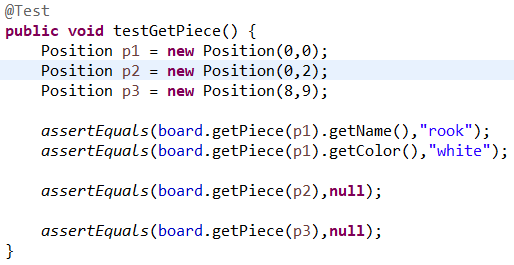
ChessBoardTest

1. testAddPiece

位置上有无棋子，是否是棋盘上的位置，棋子是否超过上限几种情况

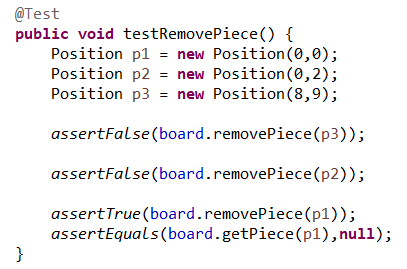


1. testGetPiece
2. 位置上有无棋子，是否是棋盘上的位置几种情况



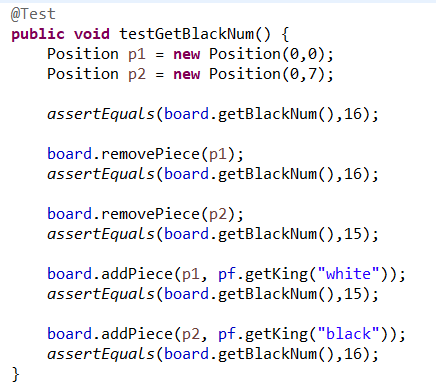
1. testRemovePiece

位置上有无棋子，是否是棋盘上的位置几种情况



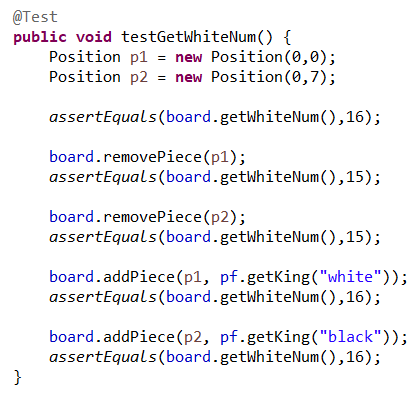
1. testGetBlackNum

给出一组动作，判断棋子数量是否符合预期



1. testGetWhiteNum

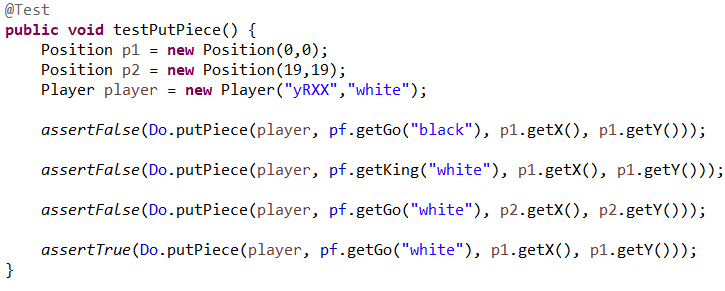
给出一组动作，判断棋子数量是否符合预期



GoActionTest

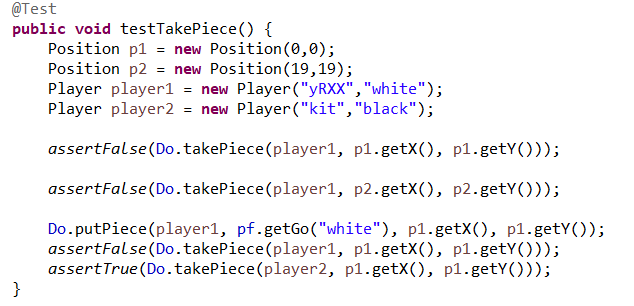
1. testPutPiece

位置是否在棋盘上，是否是己方棋子，位置上是否已经有棋子



1. testTakePiece

是否在棋盘上，是否是对方棋子，是否有棋子



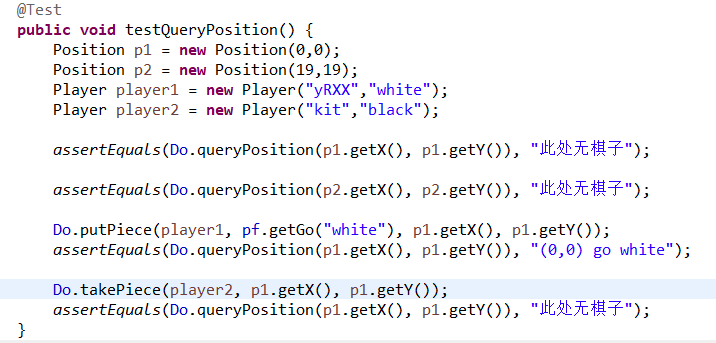
1. testGetHistory

给出一系列动作，判断历史记录是否符合预期



1. testQueryPosition

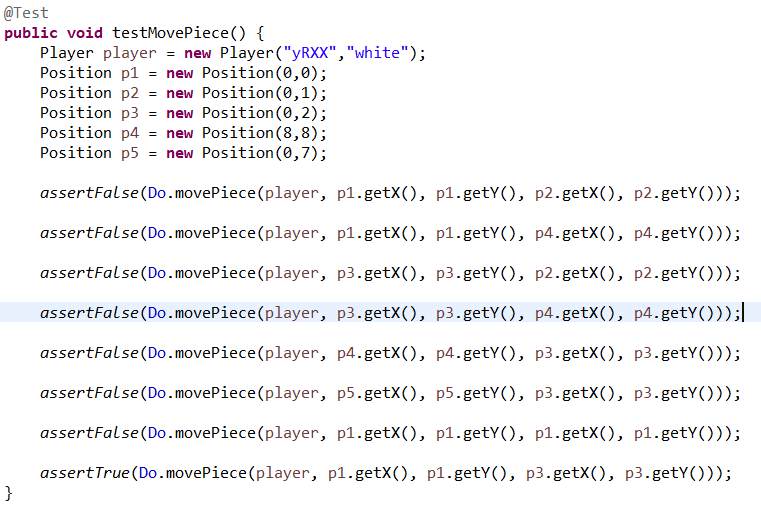
位置是否在棋盘上，是否有棋子，输出格式是否正确



ChessActionTest

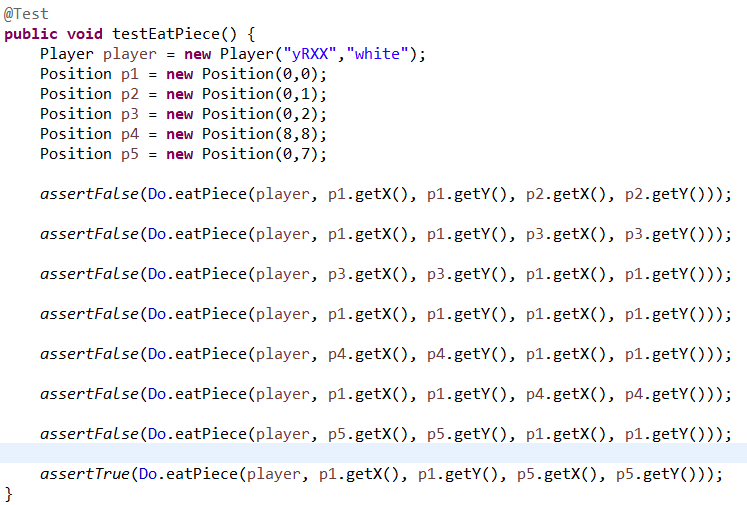
1. testMovePiece

起点和终点（是否在棋盘上，是否有棋子，颜色是否正确）



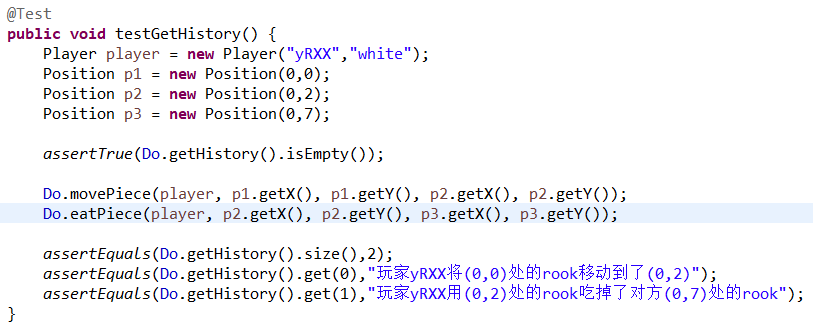
1. testEatPiece

起点和终点（是否在棋盘上，是否有棋子，颜色是否正确）



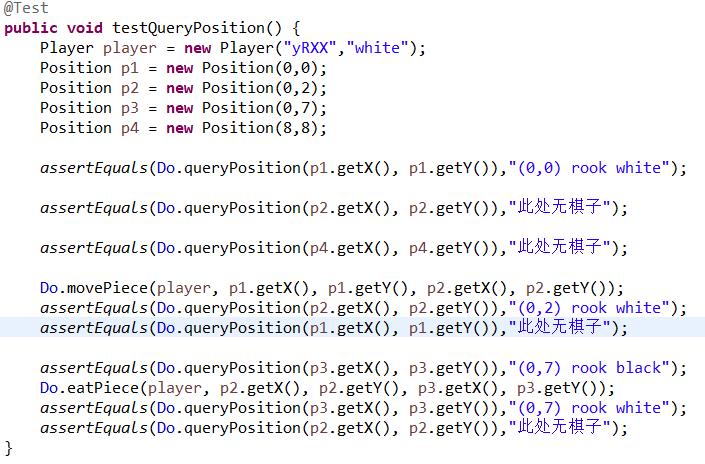
1. testGetHistory

给出一系列动作，判断历史记录是否符合预期



1. testQueryPosition

位置是否在棋盘上，是否有棋子，输出格式是否正确



## 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 |
| 2019-3-31 | 不敢数 | P1 | 未完成 |
| 2019-4-01 | 不敢数 | P1 | 未完成 |
| 2019-4-02 | 不敢数 | P1，P2 | 完成 |
| 2019-4-03 | 不敢数 | P3 | 未完成 |
| 2019-4-06 | 不敢数 | P3 | 完成 |
| 2019-4-07 | 不敢数 | 写报告 | 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 |
|  |  |
|  |  |
|  |  |

# 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

菜和懒是原罪，下个实验必须抓紧写

## 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？

很多地方可以复用，更能做到一劳永逸

1. 使用泛型和不使用泛型的编程，对你来说有何差异？

目前还没有感觉到差异，以后做大的系统就会有很大差异了

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？

可以避免自己写代码时遗漏的问题，如果先写代码再写注释难免会被自己的代码误导

适应不过来啊

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？

以后再写和图有关的东西我就可以直接用了

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？

头秃

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？

尽量完善安全性和健壮性

尽力而为

1. 关于本实验的工作量、难度、deadline。

适中

1. 《软件构造》课程进展到目前，你对该课程有何体会和建议？

要养成新的编程习惯确实很难，如果放到大一去上这门课就好了