

**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 | 安天 |
| 学号 | 1170300627 |
| 班号 | 1703006 |
| 电子邮件 | 2507770493@qq.com |
| 手机号码 | 14794410346 |

**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)

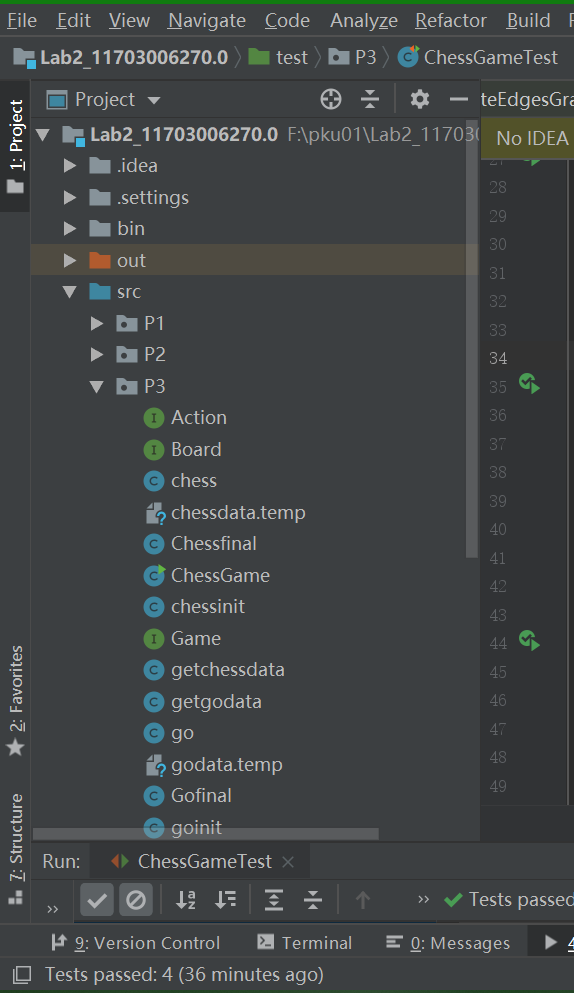
[6.2 针对以下方面的感受 4](#_Toc1988246)

# 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



https://github.com/ComputerScienceHIT/Lab2-1170300627

# 实验过程

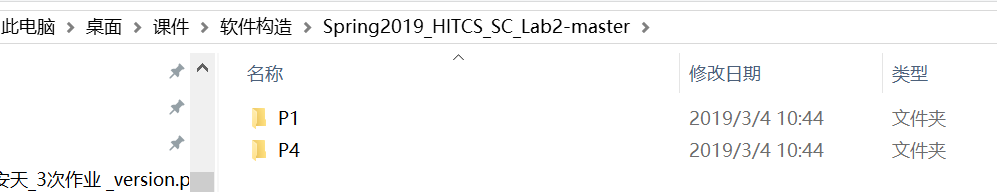
请仔细对照实验手册，针对三个问题中的每一项任务，在下面各节中记录你的实验过程、阐述你的设计思路和问题求解思路，可辅之以示意图或关键源代码加以说明（但千万不要把你的源代码全部粘贴过来！）。

## Poetic Walks

我觉得该实验分成了三个部分，第一部分是完成以边为主的图结构创建以及实现图结构对应的方法和边类对应的方法，第二部分是实现以顶点为基础的图结构的创建以及实现顶点中的方法的实现。在最后一部分的诗意漫步中需要完成的是在之前已经实现的图结构的基础上寻找关联度最紧密的顶点，从而完成诗歌的创建。

### Get the code and prepare Git repository

直接点击download下载就好了



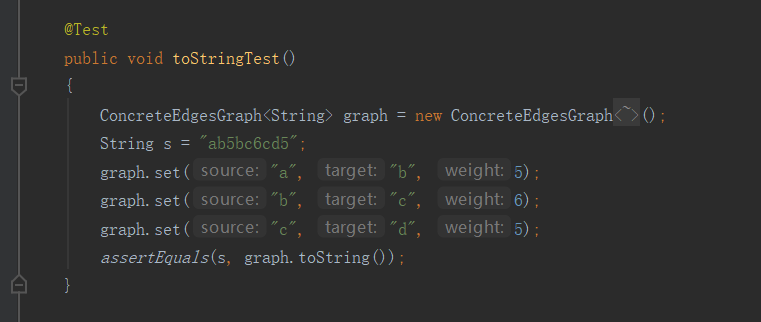
然后该文件夹就会被下载到本地

最后再解压一下，导入编译器

完成

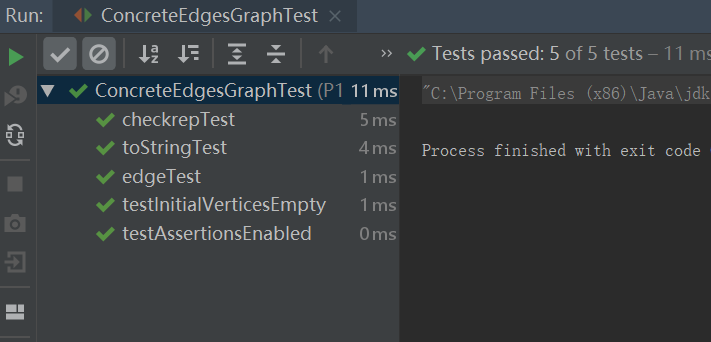
### Problem 1: Test Graph <String>

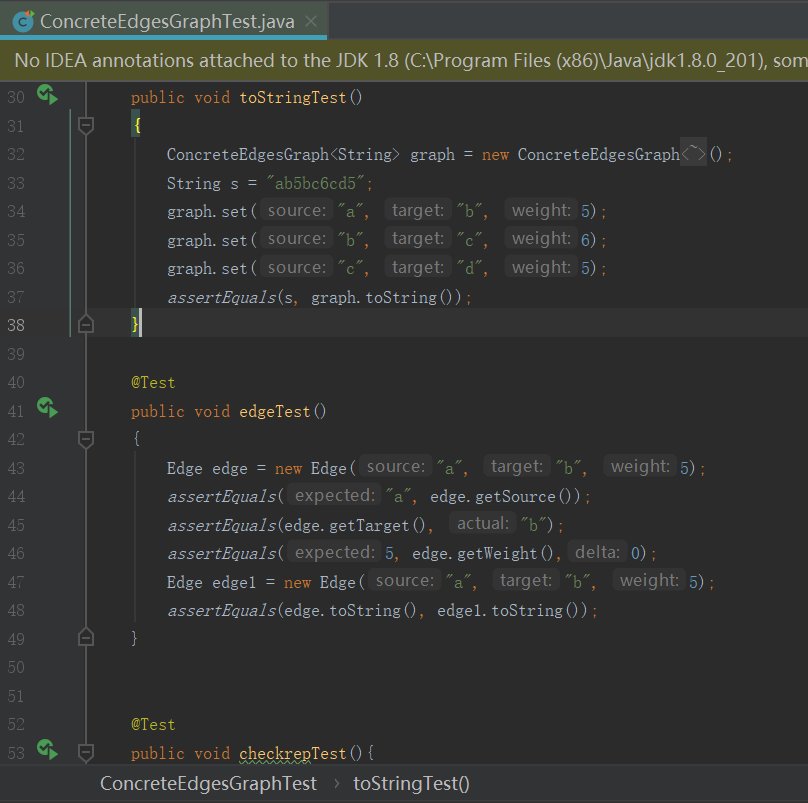




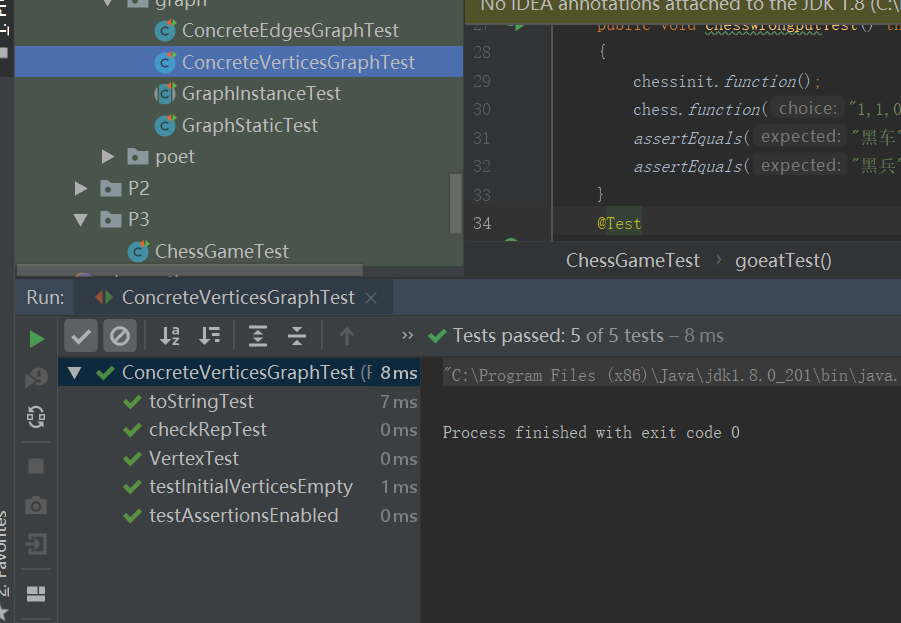
在针对第一个测试函数进行完成的时候我对是否能够添加边，以及不变量检查都进行了测试

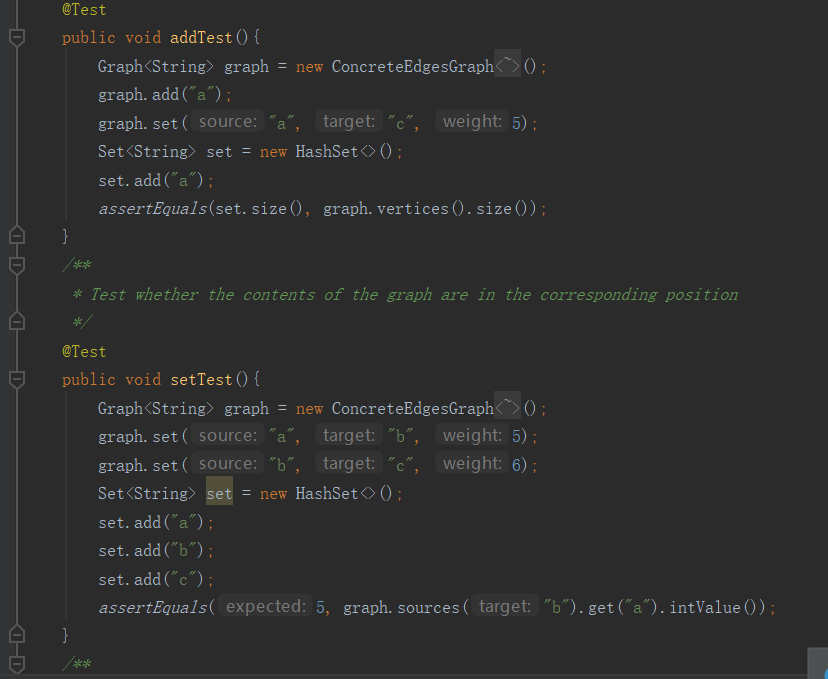
对于需要重写的tostring方法也针对我写的测试用例进行了测试，结果显示我的测试用例和我的程序能对应上

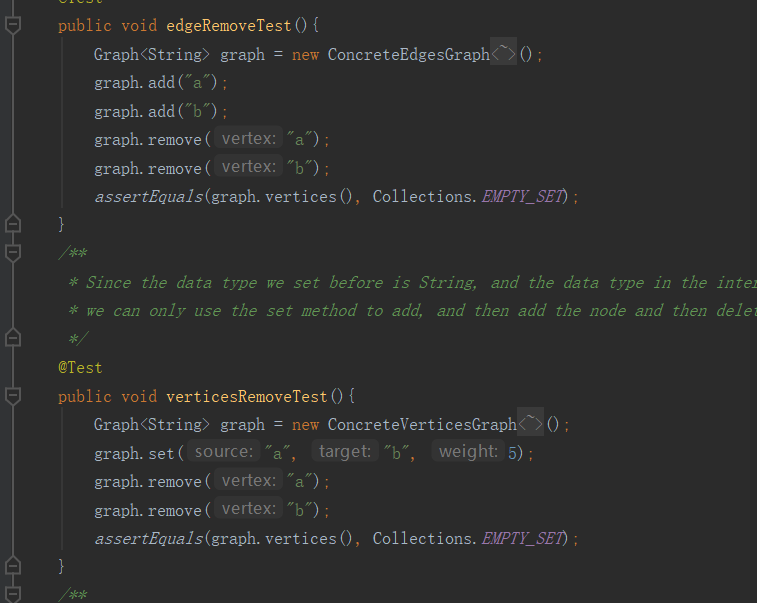




对于边的测试和上面顶点测试的测试内容相同，测试的结果也是相同我选择的测试用例我的函数都能够实现功能

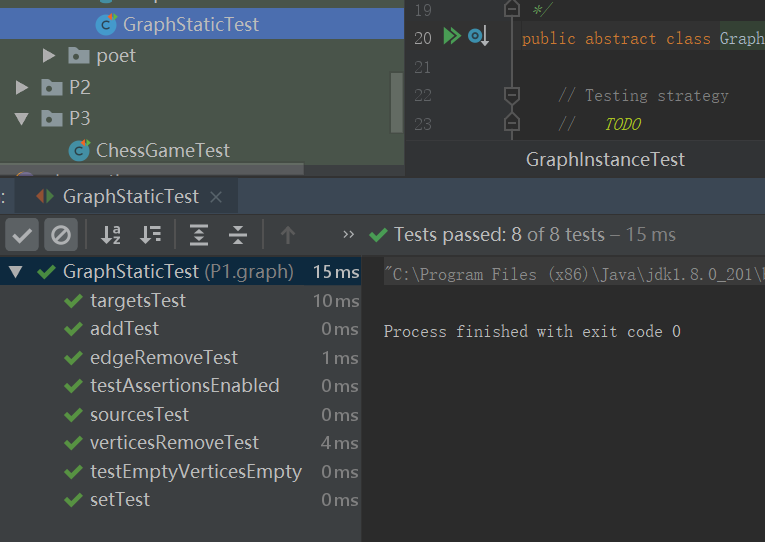






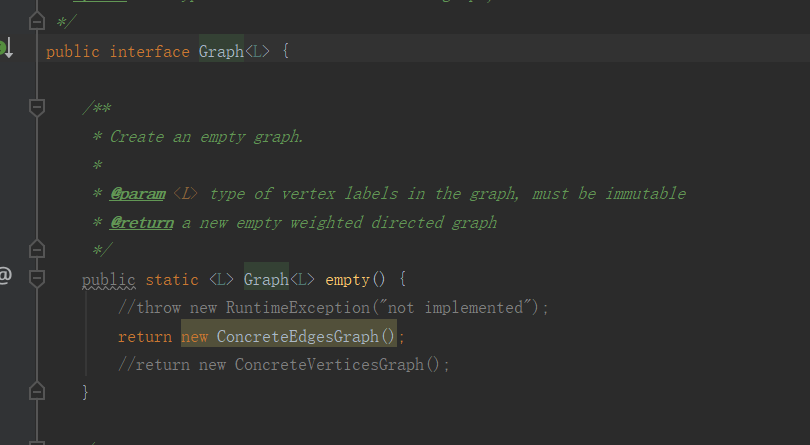


在static的测试中我对于函数中所有给定的方法也都设计了测试用例并进行测试。结果如下



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。

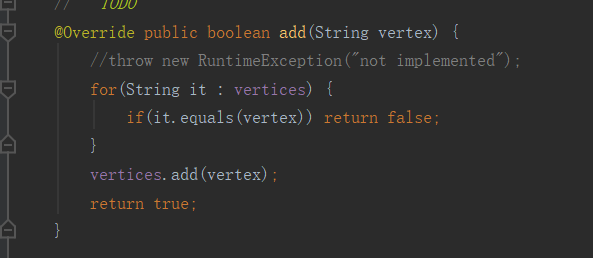
### Problem 2: Implement Graph <String>



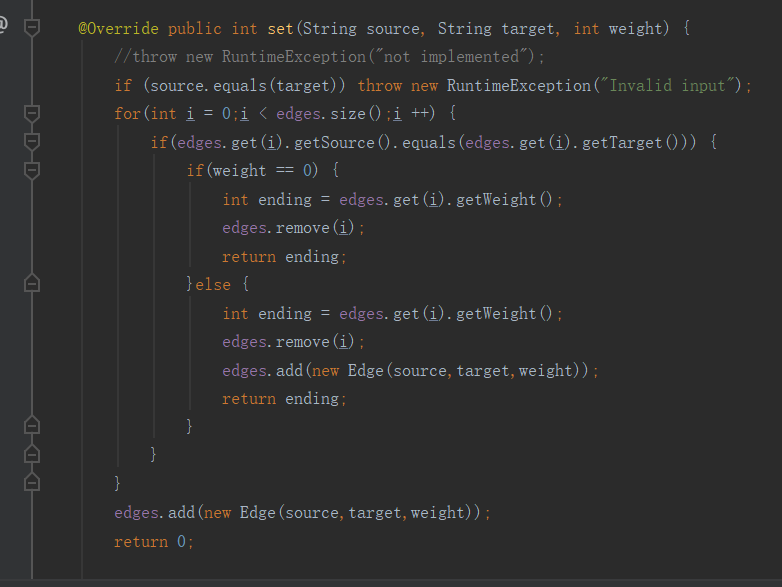
在进行该项功能的时候只需要把L都换成String即可，因为java在范围上可以向下兼容，但是不可以向上兼容。

#### Implement ConcreteEdgesGraph

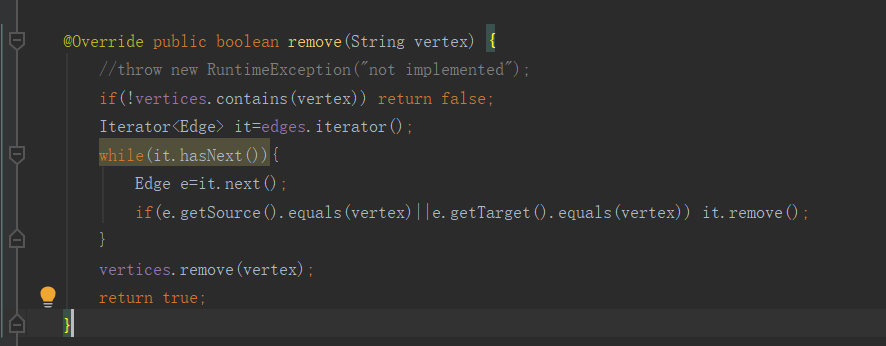
Add函数中只需要向vertices中addvertex就行了，重要的我觉得就是对于重复情况的判断，用来保证程序的健壮性



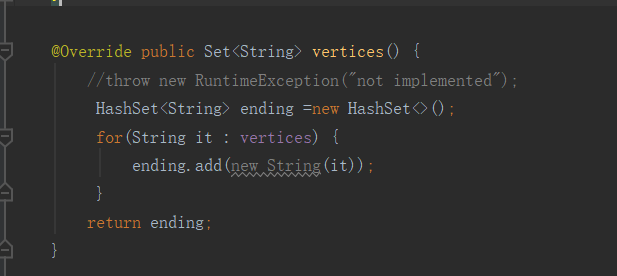
在set函数中和上一个问题一样，主要的还是对于是否有重复情况的判断，以及输入的weight情况的判断，在判断之后直接保存结果就好了



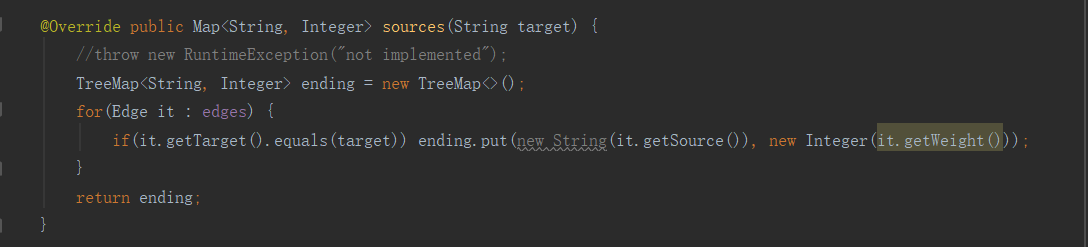
Remove在判断了是否存在之后直接删除就好了



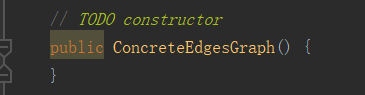
在set函数中注意防御性拷贝，把所有的元素都新弄一个然后再返回就好了，其实String并不需要如此操作，因为string是不可变类型，但是我想这写程序的时候规范一下，然后就做了一个无用的防御性拷贝来规范自己的编程习惯



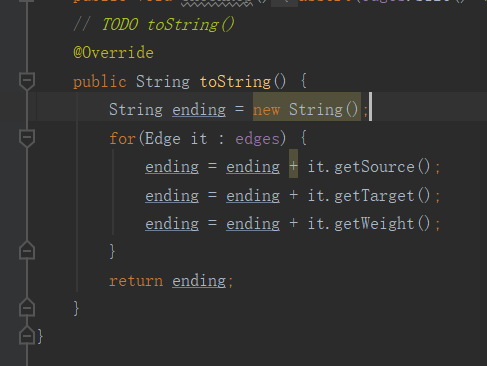
Map返回一个map在程序中便利一下所有的情况，装进去就好了



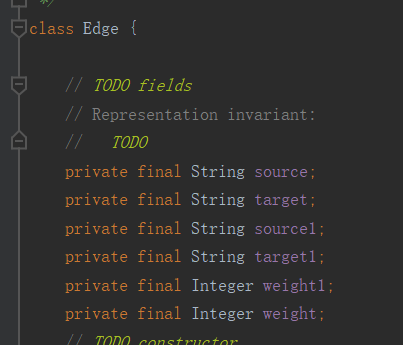
构造器，感觉没什么写的必要，因为默认的构造器就是下面我写的东西，但是老师让写我就把这个原型给写上吧权当加深自己对于java的理解了

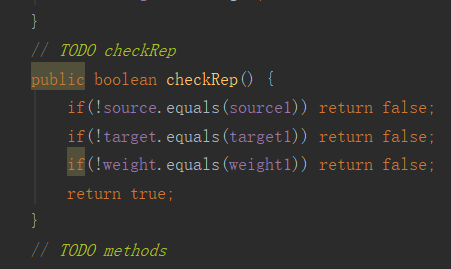


Tostring虽然编译器自带重写功能但是为了表示我写了作业我还是自己实现了这个函数，虽然在格式上不是很规整，但是所含信息都覆盖到了



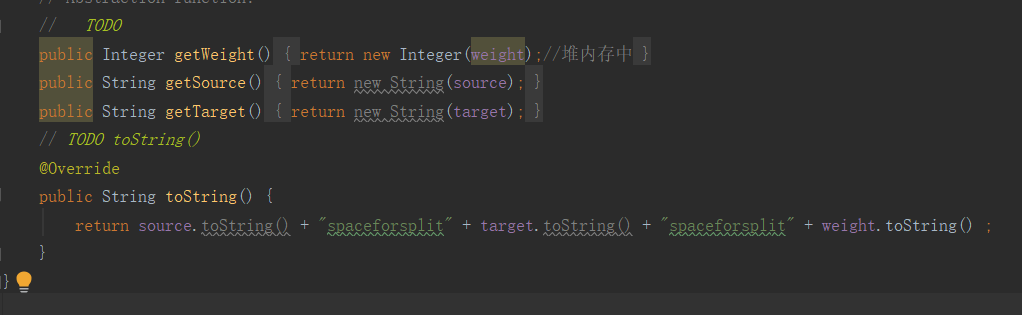
在边的类中我为了保证程序的不变性，所以我做的第一点改进就是把所有的变量都弄成了private final的形式，有一份是程序里有并且提供给用户用的，还有一份是我留在类中的，作用只是核对另一份是否正确，所以checkrep函数就是在这个基础上实现的，检查是否有字段被修改





虽然string并不需要防御性拷贝但是我依旧进行了一下防御性拷贝

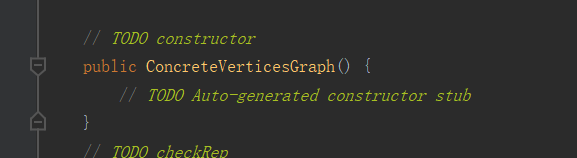
在tostring中我也是自己写了一个tostring



#### Implement ConcreteVerticesGraph

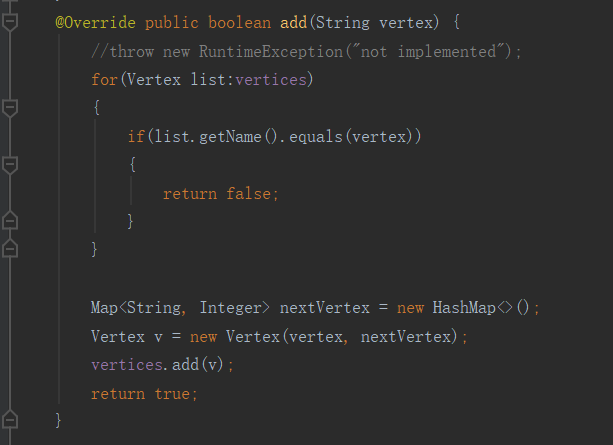
**图换一个方式实现**

**构造器，依旧是直接写出来了，虽然我也不知道老师设置这个问题到底有什么意义**



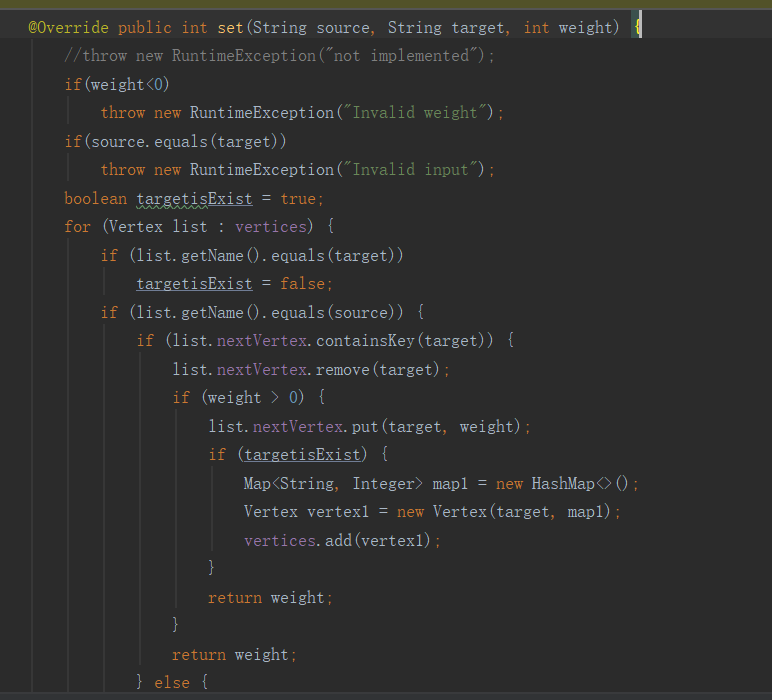
Add函数用于添加顶点，那就直接添加就好了

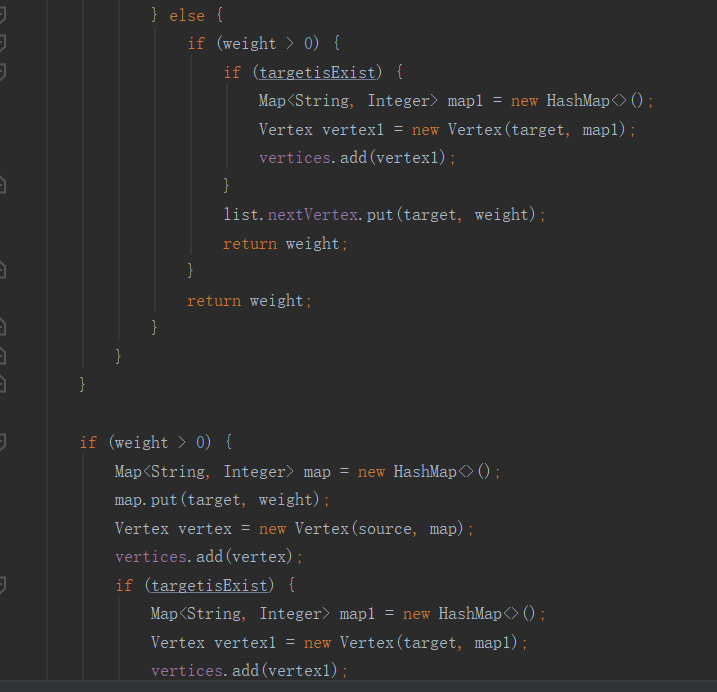
Map找顶点和权之间的对应关系，依旧是遍历



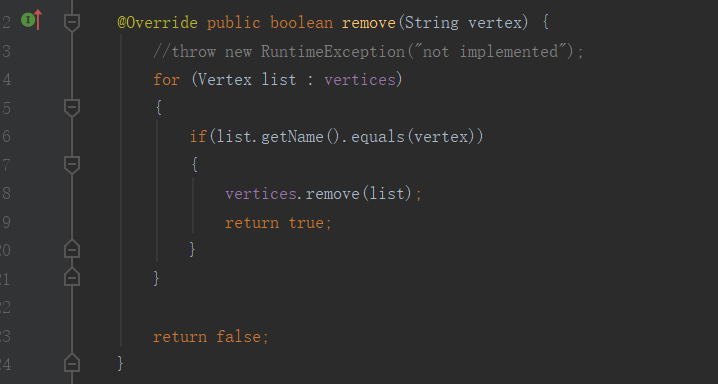
在这个问题中的set

有一些复杂因为所给的结构并不能简介的完成任务，对于wight的情况，以及顶点是否有重复的情况都需要进行判断。还得看看是否给的两个点在我的顶点集中并没有。只有在判断结束之后，所有的要求都符合了之后才能将所给的条件装入。或者如果输入的weight是0那就代表该条边不存在了，就得删除此条边

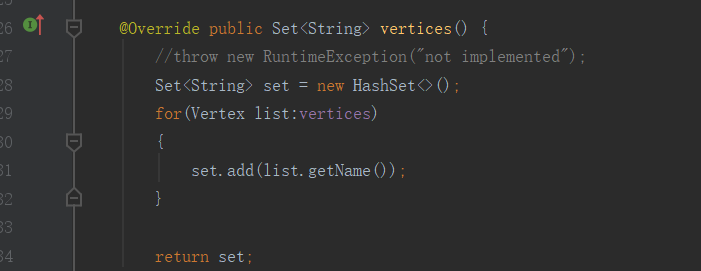




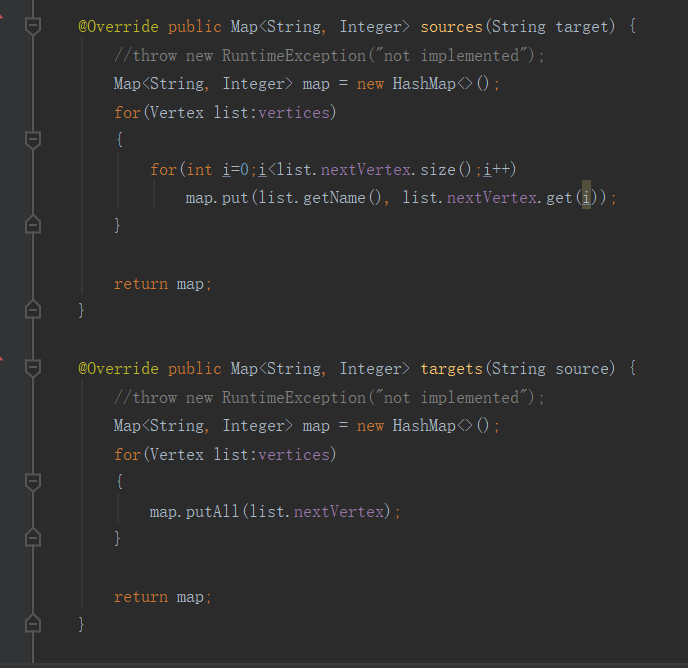
Remove直接删除就好了，如果没有对应的就返回一个false



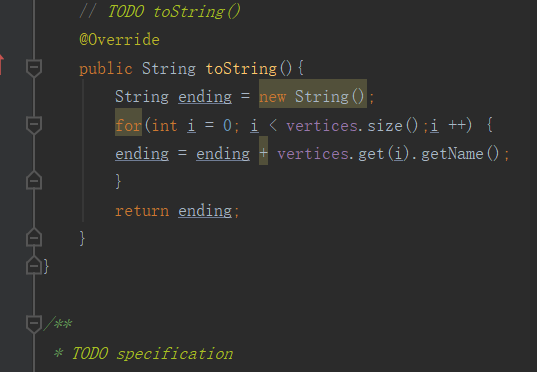
在set的时候返回一个集合就好了，遍历一下集合中的元素，否则，如果直接返回集合的话，用户就可能对集合进行修改，违背了我们不变量的初衷



下面的这两个函数其实在逻辑上是一样的，只不过一个是找出来起始顶点一个是找出来终止顶点罢了。只不过是遍历一下而已

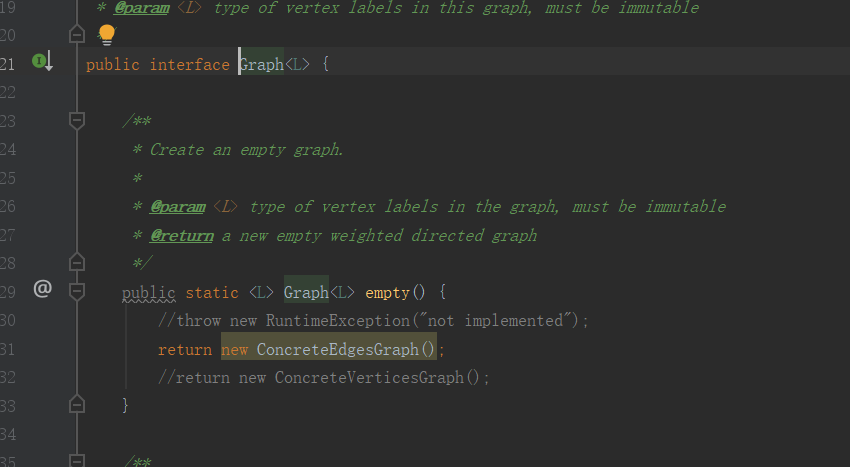


Tostring在string中返回一下所有的信息就好了



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

在Graph中只有这一个函数需要完成

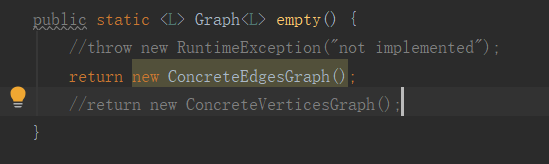


#### Make the implementations generic

只需要把带函数中部分带String的地方都给改成L就好了

#### Implement Graph.empty()

直接在里面新建一个对象就好了，有两个u第项可供选择，用什么随意。因为都是实现了接口的类

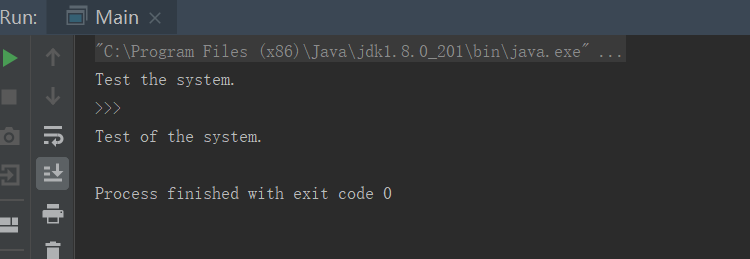


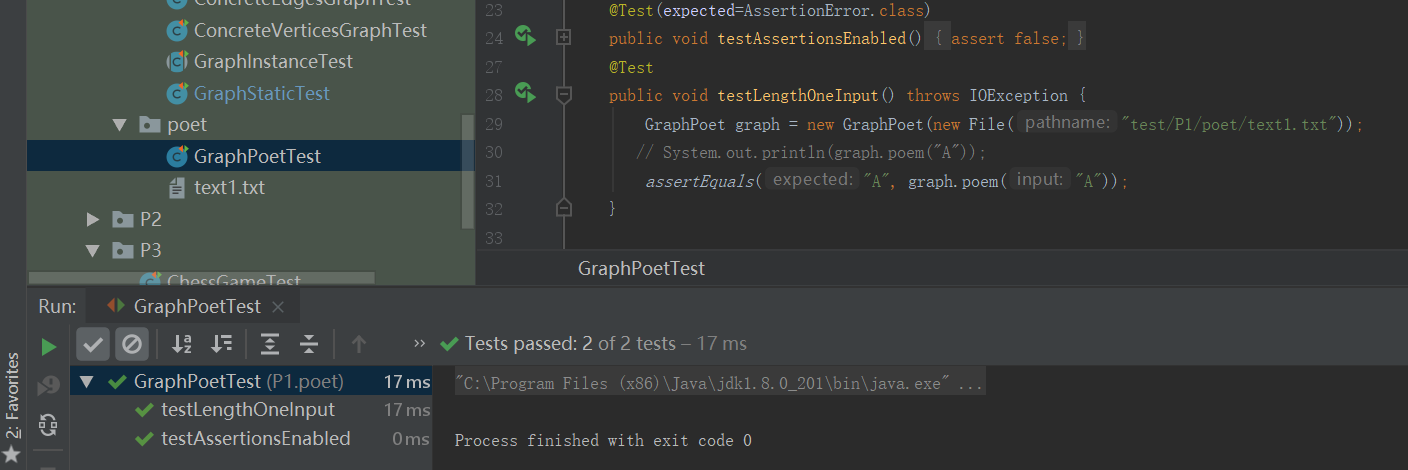
### Problem 4: Poetic walks

#### Test GraphPoet

老师给的文件中自带的Main测试已经被我完成了，在此基础上我觉得多点的测试已经完成了，剩下的应该就是单点的测试了，所以我就写了一个针对于单点测试的函数，看看能给出什么样的结果

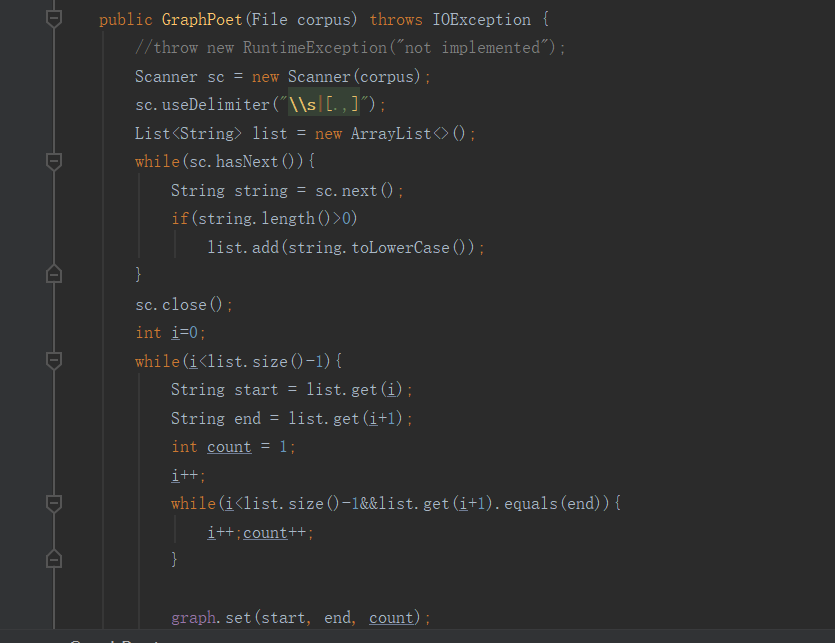
测试结果显示我的函数在单点和多点的情况下都唔那个完成任务



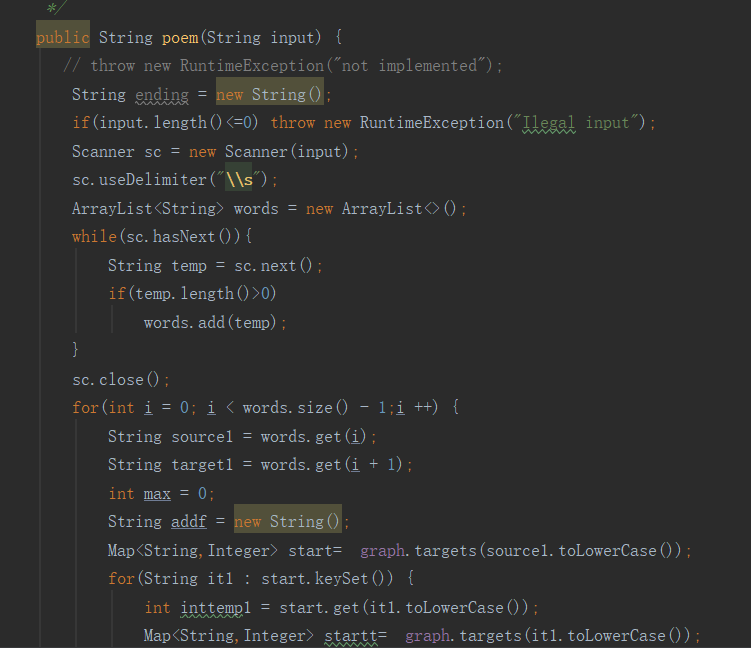


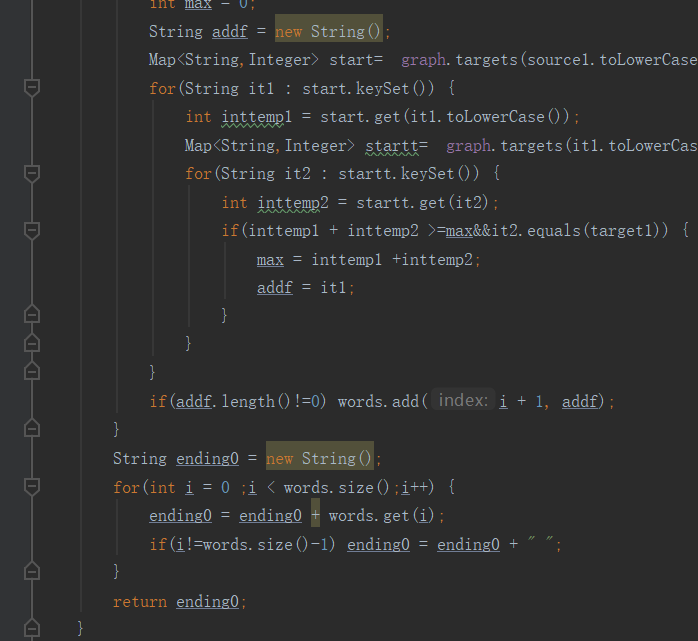
#### Implement GraphPoet

Graphpoet函数先把我的语料库给读进来，弄到之前我写的图里去，为下一步的检索做铺垫



我的语料库已经被我导入到内存中了，所以剩下我需要做的就是检索了，针对由于所有的点我都两两遍历一下，寻找之间的权值最大的情况，然后再输出就好了

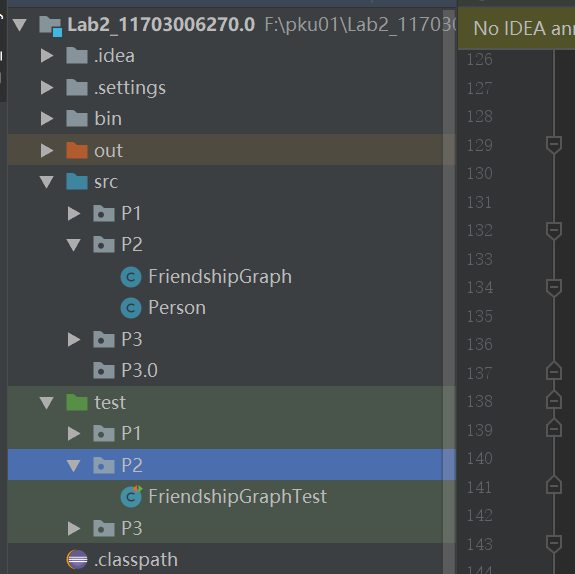




#### Graph poetry slam

上一问中说了

### Before you’re done

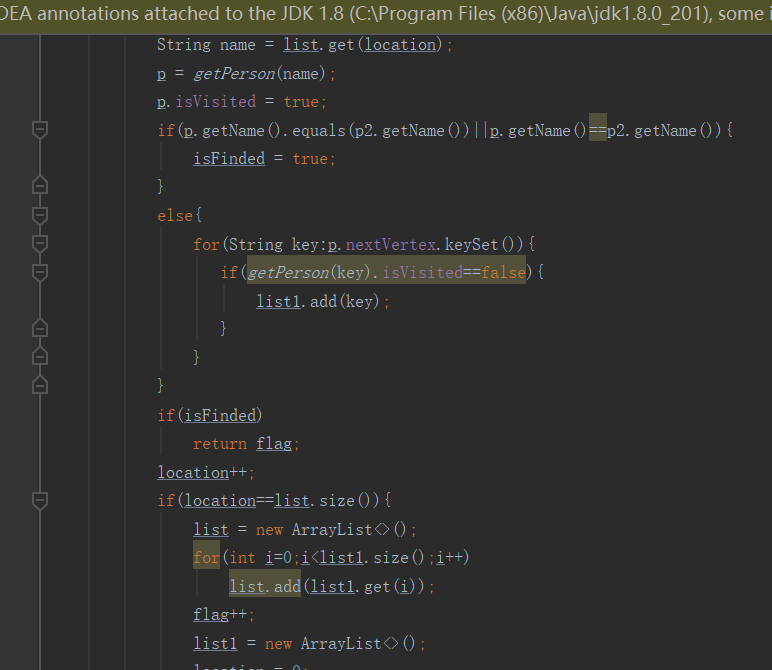
在这里给出你的项目的目录结构树状示意图。

## Re-implement the Social Network in Lab1

就是在实验1的代码的基础上对不变量进行一下补充就好了

### FriendshipGraph类

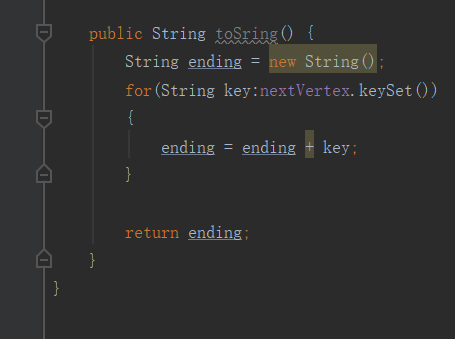
此次实验中在getdistance的功能上实现的更加谨慎，不改变原有数据是我在此次实验中的追求，也就是维持程序和变量的不变性



给出你的设计和实现思路/过程/结果。

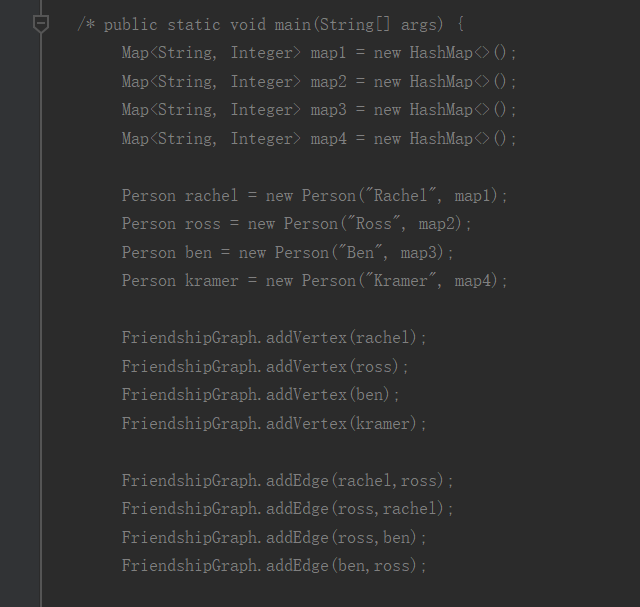
### Person类

在上次的基础上加上了tostring的功能函数

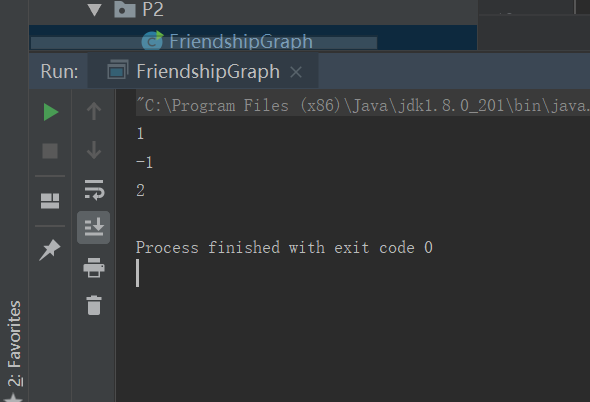


### 客户端main()

客户端也是上次实验中直接给出来的。。。

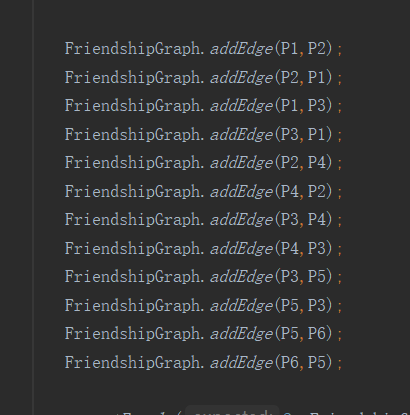


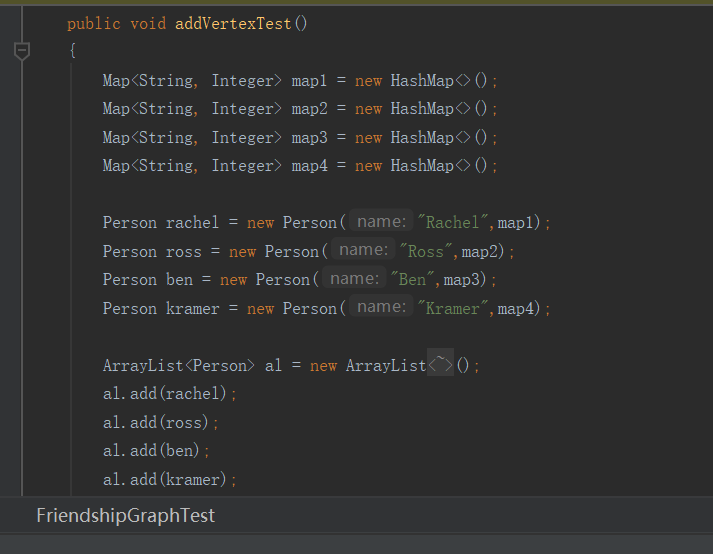
运行结果如下

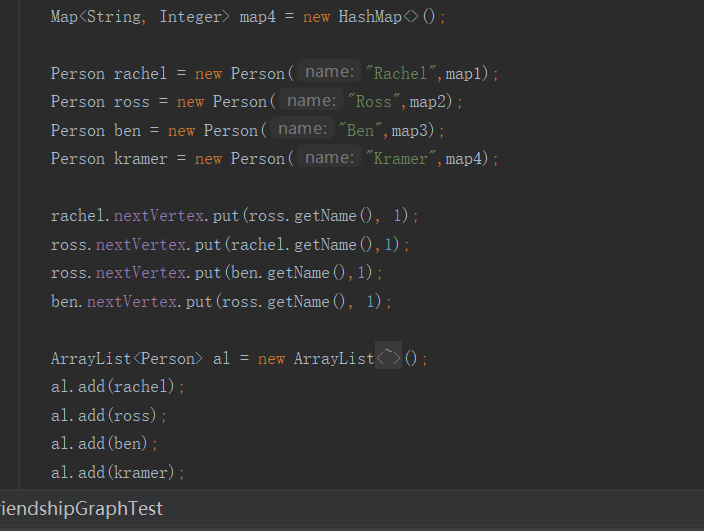


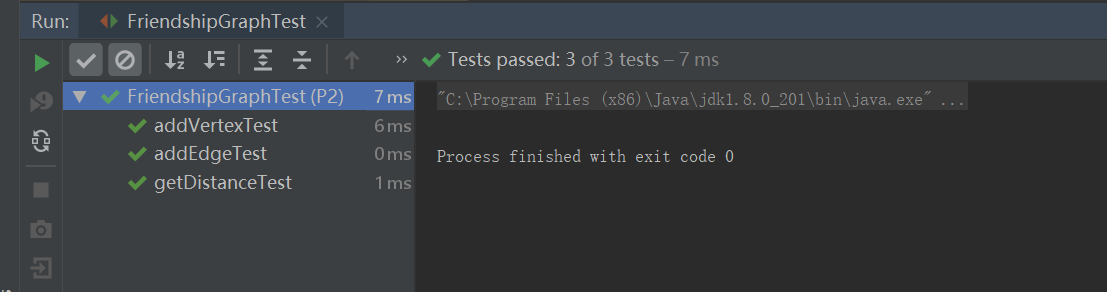
### 测试用例

在测试测试用例的时候我的思路就是先弄出来几个我能算出来距离的测试用例，再看看我的程序给出来的结果和实际的结果是否能对应上

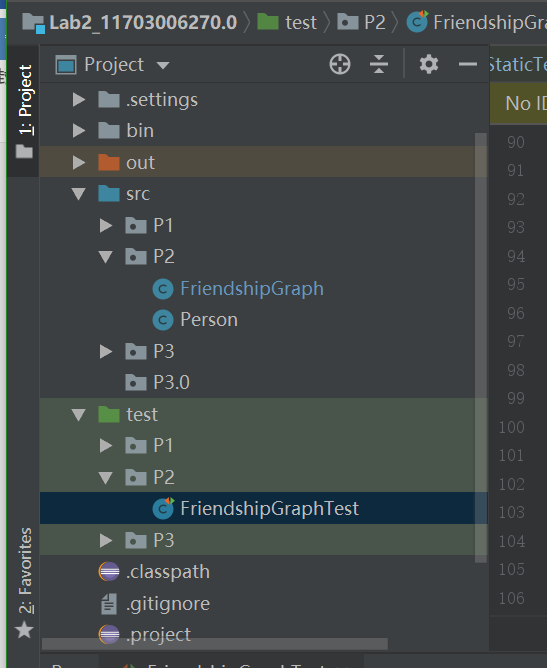






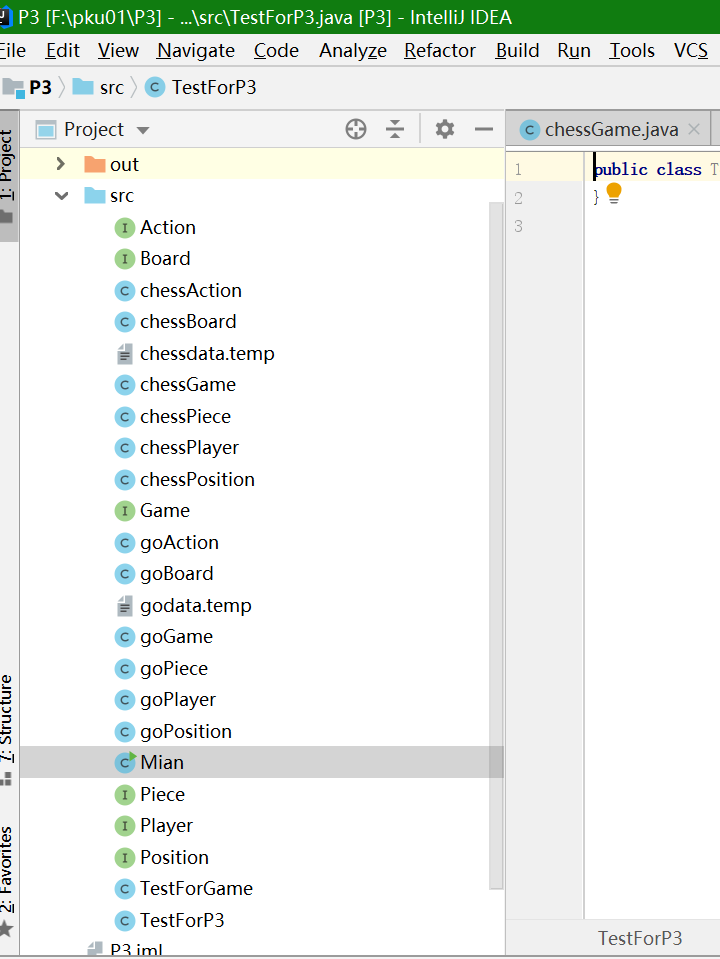


### 提交至Git仓库

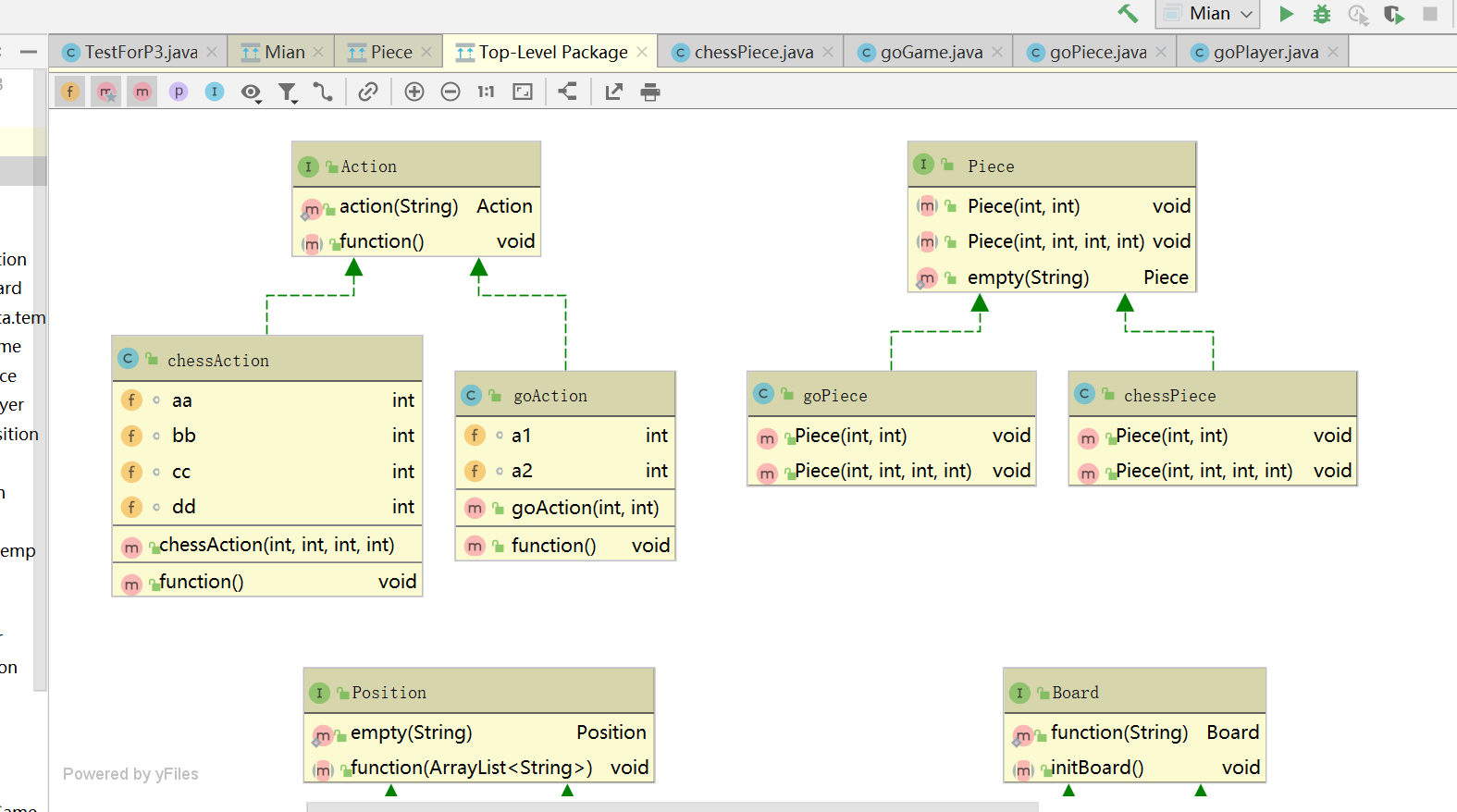


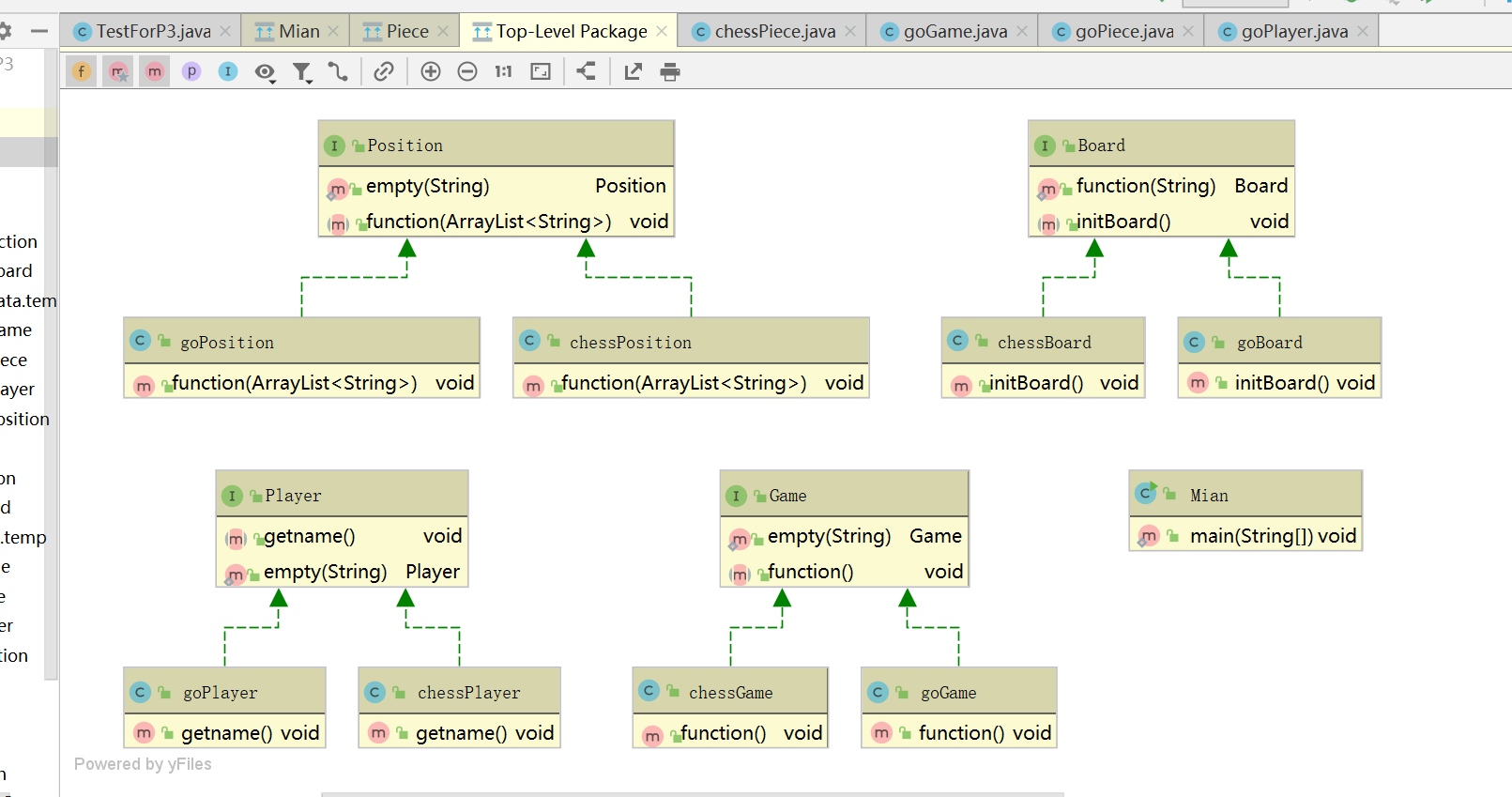
## Playing Chess

### ADT设计/实现方案



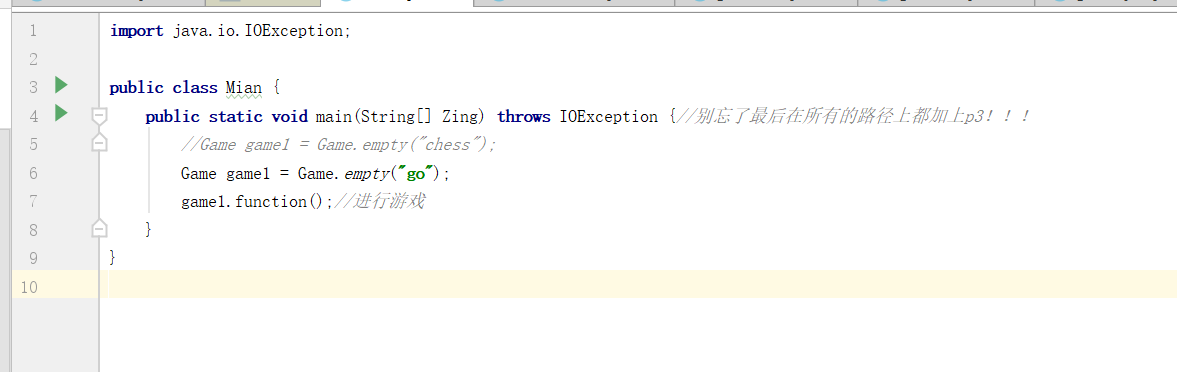
下面的图中每个类都由两个接口实现。具体的依赖关系和函数请见下图



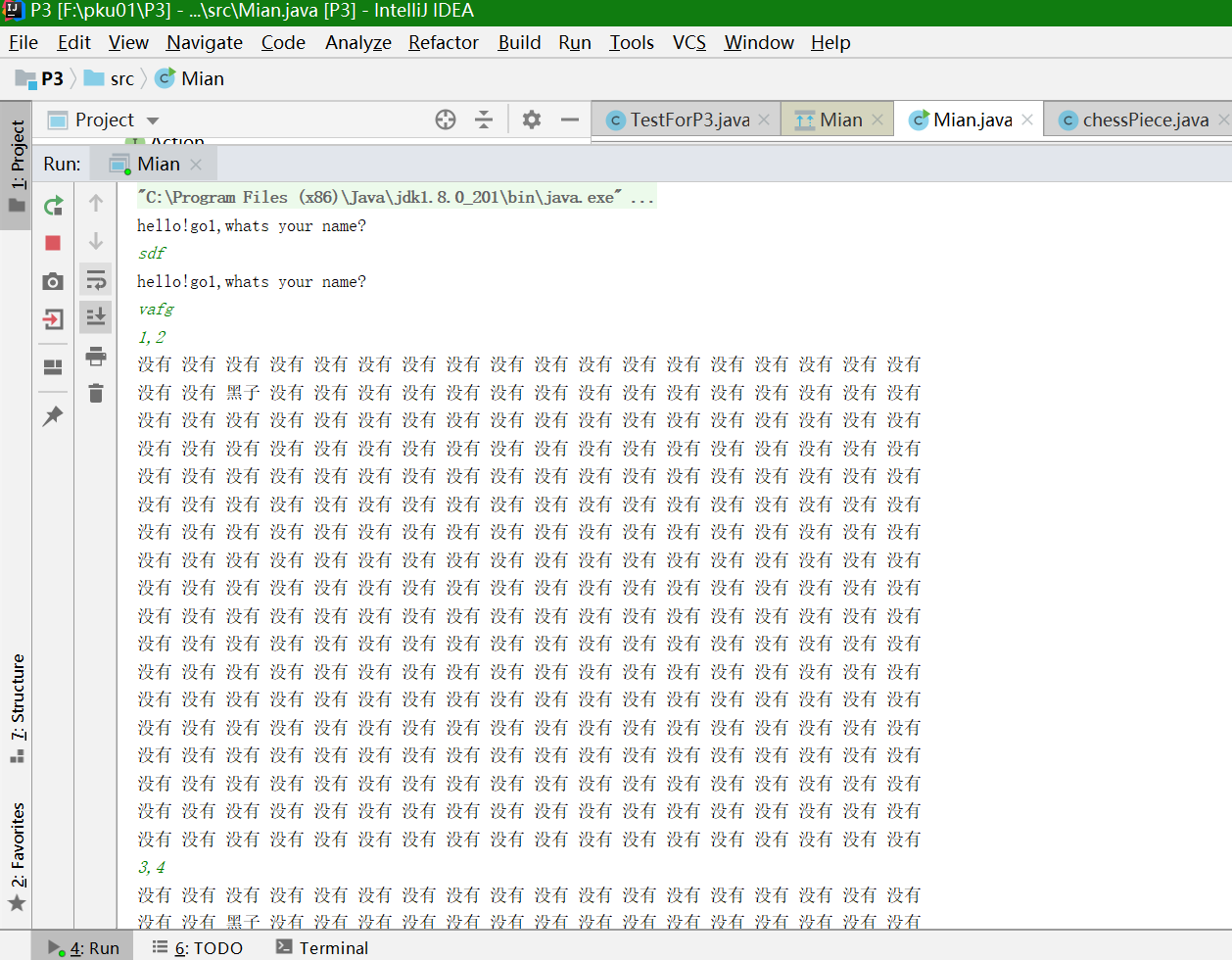


### 主程序MyChessAndGoGame设计/实现方案

为了体现面向对象的思想，我把所有的函数都封装起来了，在用户端只要调用一下function就可以完成所有的功能了



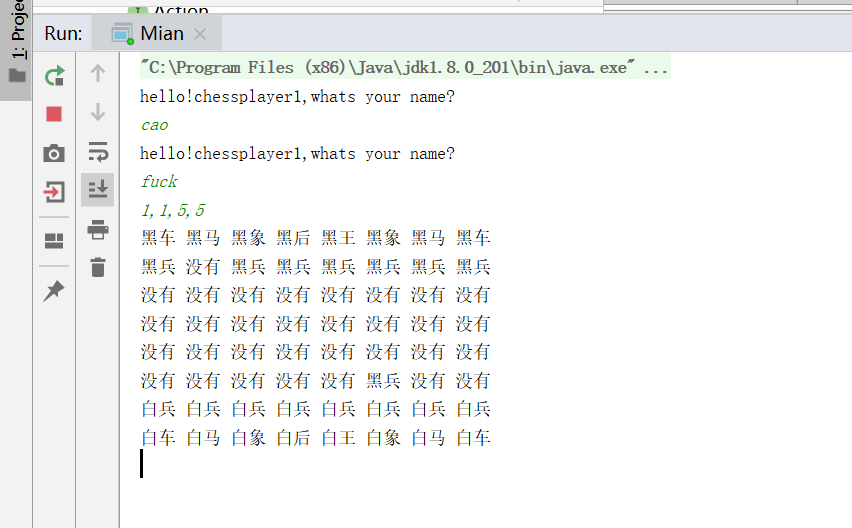
围棋运行过程如下



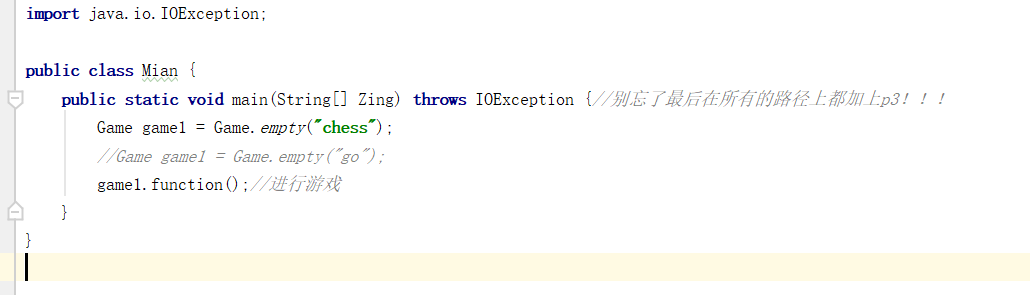
在下错了子的情况下会有如下的提示



象棋的程序演示如下

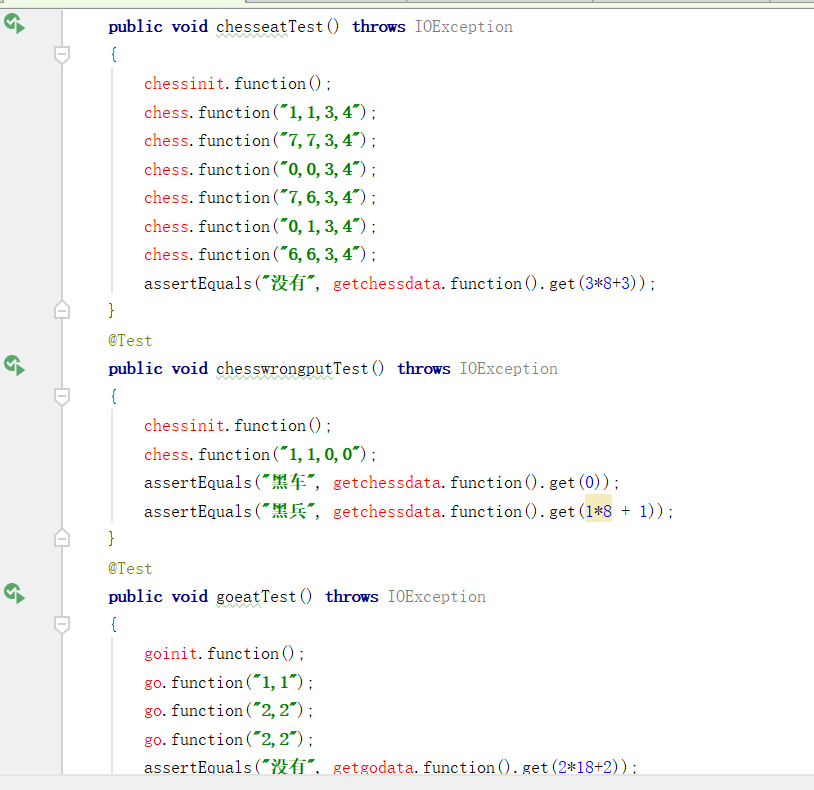


此时的主程序如下

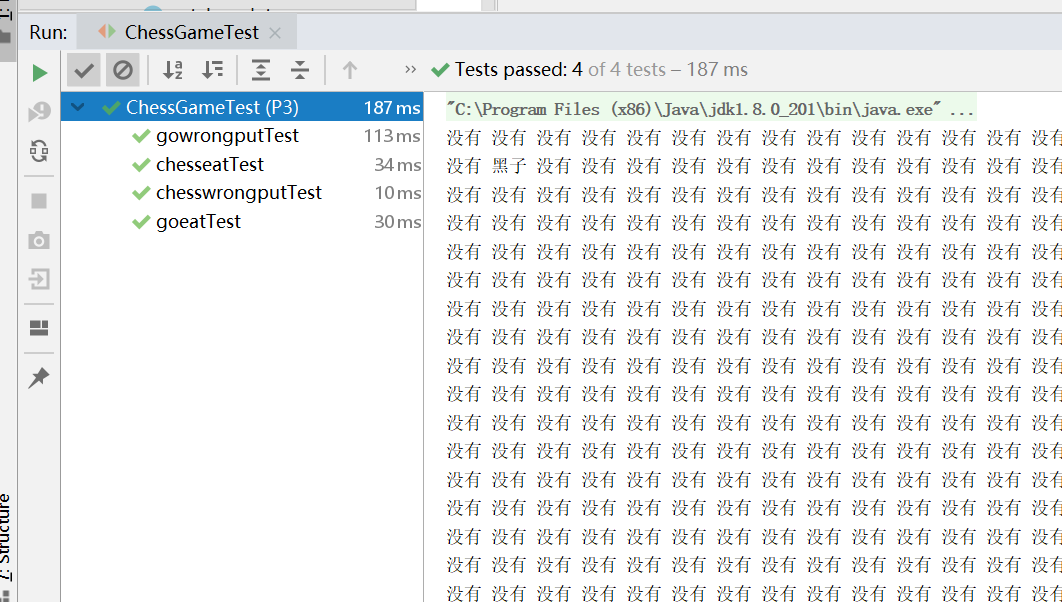


### ADT和主程序的测试方案

往里面放几组测试用例就好啦，看看得到的输出是不是我想要的输出







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。

## 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 |
| 3.21 | 14-19 | 实验1 | 完成 |
| 3.22 | 14-19 | 实验2 | 完成 |
| 3.33 | 14-19 | 实验3 | 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 |
| Empty函数没法实现。。。题错了，得修改题目。。。 | 那咋整啊。。。我想了两周也没想明白。。。感觉按照MIT的课件老师给的题目是不可能实现的。老师说题错了我才改题目。。。希望以后出题谨慎点吧。别出那种无解的题了，别坑下一届学弟学妹了！！！ |
|  |  |
|  |  |

# 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

以后拖到最后两天再写软件构造作业。这样题目中的错误就会被别人发现了。在软件构造作业上赶进度是不可能的。

## 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？

Adt编程对于程序的服用更加方便

1. 使用泛型和不使用泛型的编程，对你来说有何差异？

泛型可以兼容多种类型的对象，使用起来更加方便

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？

直接啊，不用等到设计好程序签名之后再写测试用例

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？

直接提供了规范，有一个统一化的规格。使用者使用起来更加方便，直接按着说明书操作就好了

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？

感觉对于我的程序设计能力有着很好的提升作用。对于adt的理解更加深刻了

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？

因为我们作为程序员我们需要保护类中的边类不被外界用户修改，如果里面的数据大家都能修改的话，那还怎么保护数据。

愿意，因为用户的操作总是不能符合程序员的思想

1. 关于本实验的工作量、难度、deadline。

工作量不大，没啥难度，如果说有难度的话就应该是那道错题了

1. 《软件构造》课程进展到目前，你对该课程有何体会和建议？

换个老师出题吧，希望以后的题能没有错误