

**2022年春季学期  
计算学部《软件构造》课程**

**Lab 2实验报告**

|  |  |
| --- | --- |
| 姓名 | 瞿久尧 |
| 学号 | 120L022314 |
| 班号 | 2003012 |
| 电子邮件 | 2668136485@qq.com |
| 手机号码 | 15390311738 |

**目录**

[1 实验目标概述 1](#_Toc72247573)

[2 实验环境配置 1](#_Toc72247574)

[3 实验过程 1](#_Toc72247575)

[3.1 Poetic Walks 1](#_Toc72247576)

[3.1.1 Get the code and prepare Git repository 1](#_Toc72247577)

[3.1.2 Problem 1: Test Graph <String> 1](#_Toc72247578)

[3.1.3 Problem 2: Implement Graph <String> 1](#_Toc72247579)

[3.1.3.1 Implement ConcreteEdgesGraph 2](#_Toc72247580)

[3.1.3.2 Implement ConcreteVerticesGraph 2](#_Toc72247581)

[3.1.4 Problem 3: Implement generic Graph<L> 2](#_Toc72247582)

[3.1.4.1 Make the implementations generic 2](#_Toc72247583)

[3.1.4.2 Implement Graph.empty() 2](#_Toc72247584)

[3.1.5 Problem 4: Poetic walks 2](#_Toc72247585)

[3.1.5.1 Test GraphPoet 2](#_Toc72247586)

[3.1.5.2 Implement GraphPoet 2](#_Toc72247587)

[3.1.5.3 Graph poetry slam 2](#_Toc72247588)

[3.1.6 Before you’re done 2](#_Toc72247589)

[3.2 Re-implement the Social Network in Lab1 2](#_Toc72247590)

[3.2.1 FriendshipGraph类 2](#_Toc72247591)

[3.2.2 Person类 3](#_Toc72247592)

[3.2.3 客户端main() 3](#_Toc72247593)

[3.2.4 测试用例 3](#_Toc72247594)

[3.2.5 提交至Git仓库 3](#_Toc72247595)

[4 实验进度记录 3](#_Toc72247596)

[5 实验过程中遇到的困难与解决途径 3](#_Toc72247597)

[6 实验过程中收获的经验、教训、感想 4](#_Toc72247598)

[6.1 实验过程中收获的经验和教训 4](#_Toc72247599)

[6.2 针对以下方面的感受 4](#_Toc72247600)

# 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

[ComputerScienceHIT/HIT-Lab2-120L022314: HIT-Lab2-120L022314 created by GitHub Classroom](https://github.com/ComputerScienceHIT/HIT-Lab2-120L022314)

# 实验过程

请仔细对照实验手册，针对两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

此问题集的目的是练习设计、测试和实现抽象数据类型。此问题集侧重于实现提供规范的可变类型。

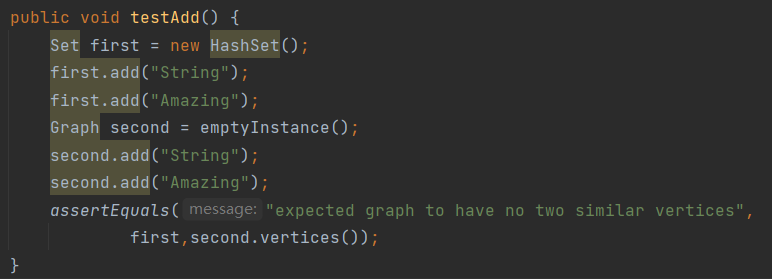
### Get the code and prepare Git repository

* 在Get the code步骤中，无法连接MIT的Athena服务器，从以下地址获取初始代码：  
  <https://github.com/rainywang/Spring2022_HITCS_SC_Lab2/tree/master/P1>
* 在作业描述中遇到“commit and push”的要求，将代码push到GitHub Lab2仓库中。
* MIT作业页面提及的文件路径，按照下表的目录结构进行调整。例如“test/poet”应为“test/P1/poet”，“src/poet”应为“src/P1/poet”。

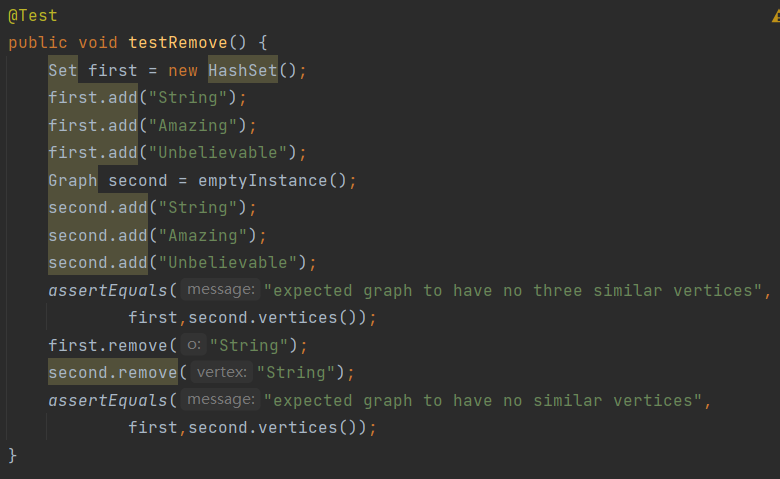
### Problem 1: Test Graph <String>

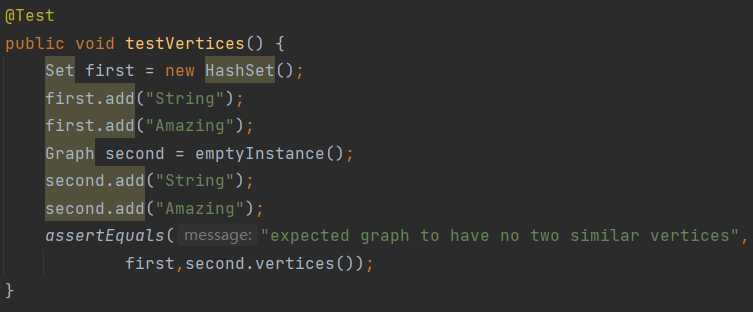
static测试已经写好，无需修改，只需要为instance写测试。

根据Graph中各函数的spec以及函数的参数和返回值，根据客户端需要的功能来进行编写测试











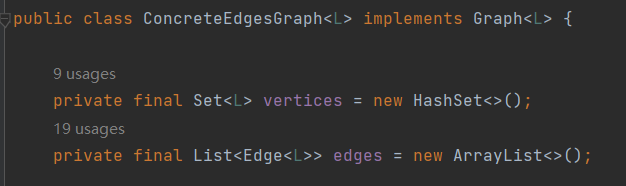


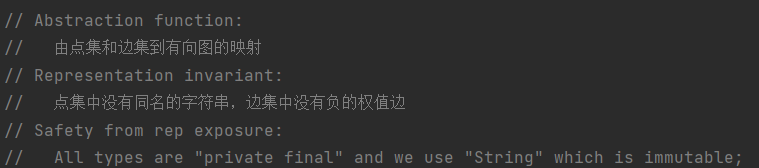
### Problem 2: Implement Graph <String>

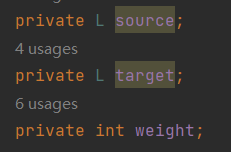
此问题要求对接口Graph中的各个方法进行两种不同的实现方式。

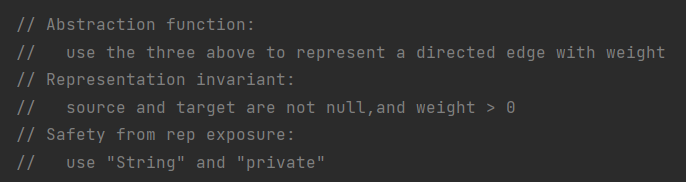
#### Implement ConcreteEdgesGraph

首先需要编写测试，由于该类实现Graph接口，所以Graph中已经写过的测试函数无需再编写测试，需要编写重写的toString方法以及新增加的Edge类。根据图的特点以及由rep到这个有向图的关系，可以完善抽象函数，表示不变量，最后保证rep无暴露。

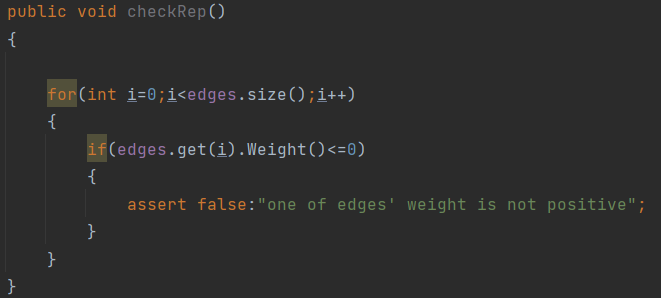


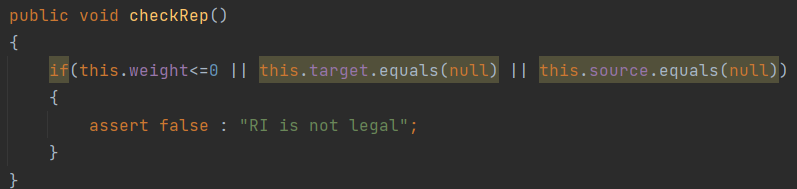






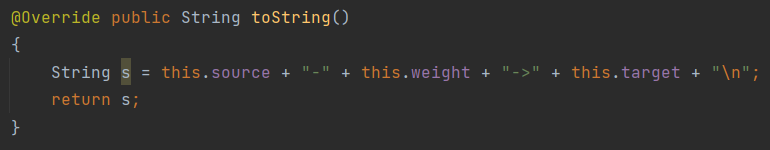
需要在这两个类中重写chekRep方法，确保无负权值的有向边，顶点非空





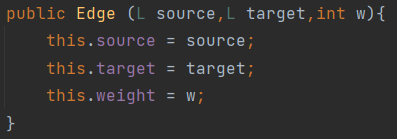
同时还需要重写toString方法进行描述图描述边





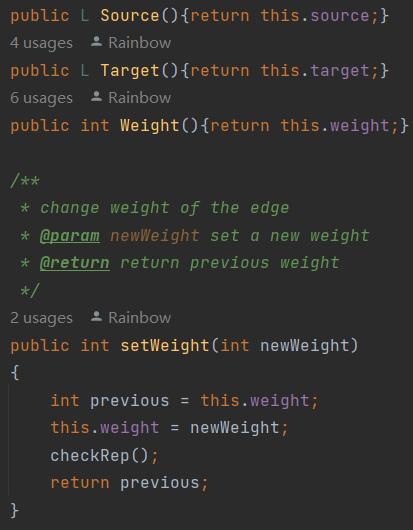
将Edge类完善，其中需要包含能对Edge的各种方法操作。

Constructor

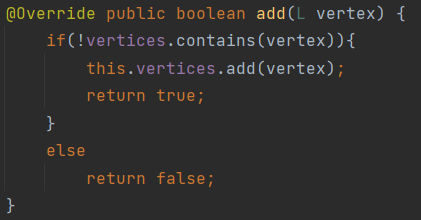


Observer来获取edge类中的private量

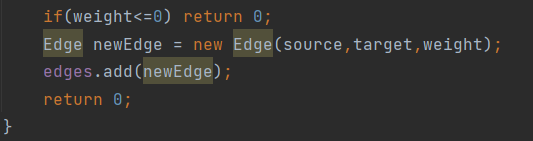
setWeight来修改weight值，方便主类中函数的调用

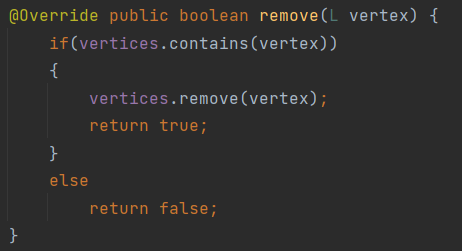


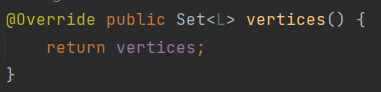
最终实现Graph接口中的各个方法

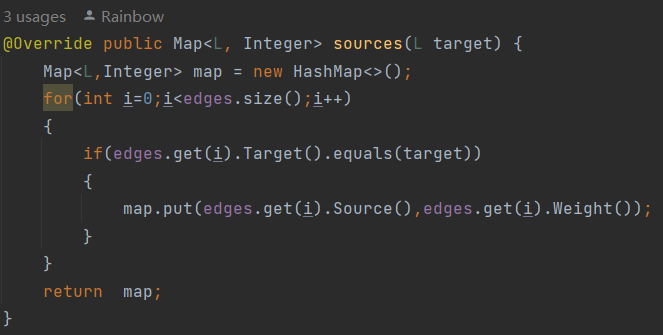


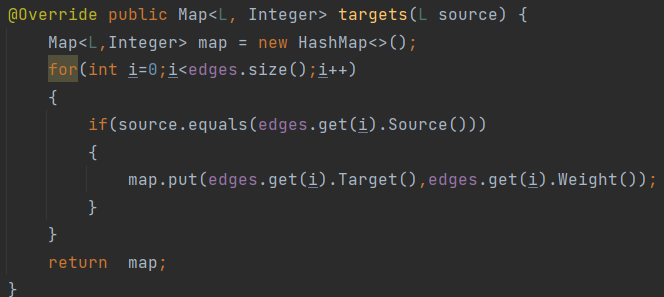










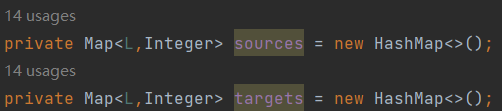


#### Implement ConcreteVerticesGraph

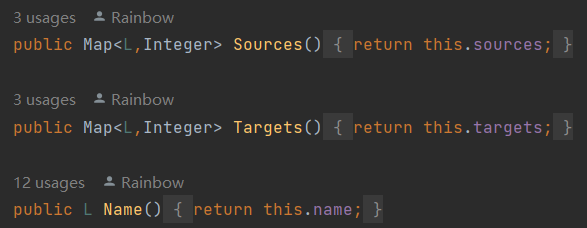
同3.1.3.1，按照顺序编写test测试，完善abstract function，RI，rep exposure

但是与上一个实现不同的是，在此实现中，Rep只提供了一个与Vertex有关的列表，因此需要更完善Vertex，以便实现各种需求。

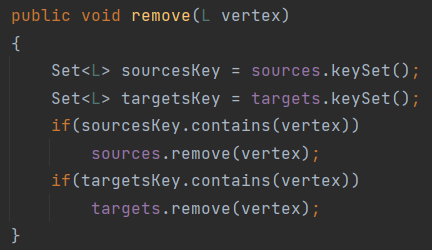
通过Map来实现与顶点有关的边

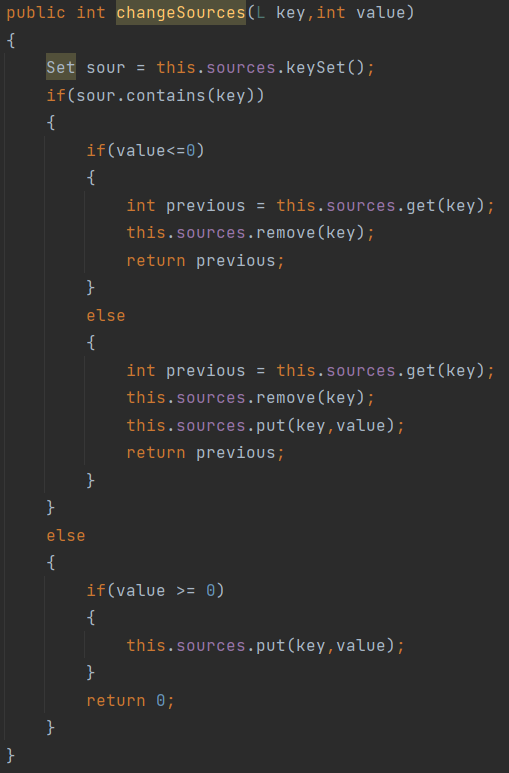


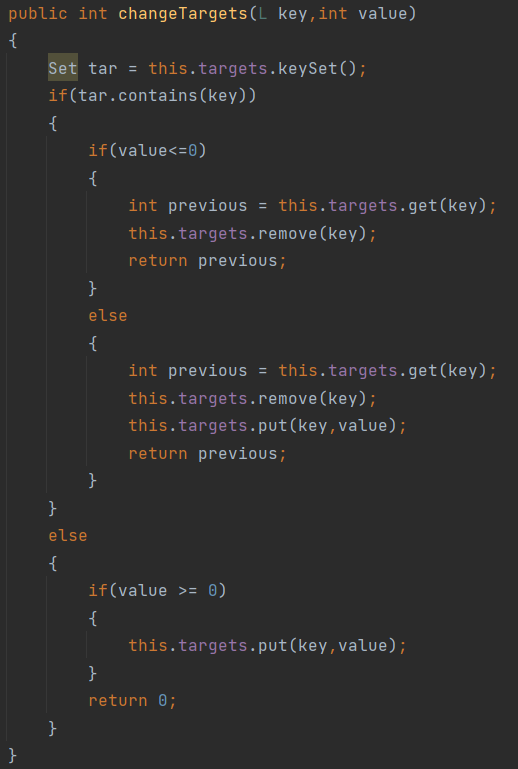
Observer



与Vertex相关的需求







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

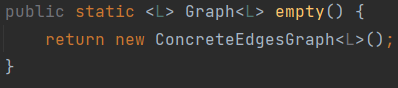
由于在上一个问题中，我们都是使用String来进行实现，因此现在改成通用类型后，只需要将大部分String修改成<L>即可完成要求

#### Make the implementations generic





#### Implement Graph.empty()

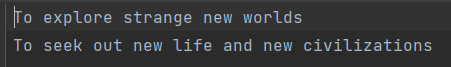


### Problem 4: Poetic walks

#### Test GraphPoet

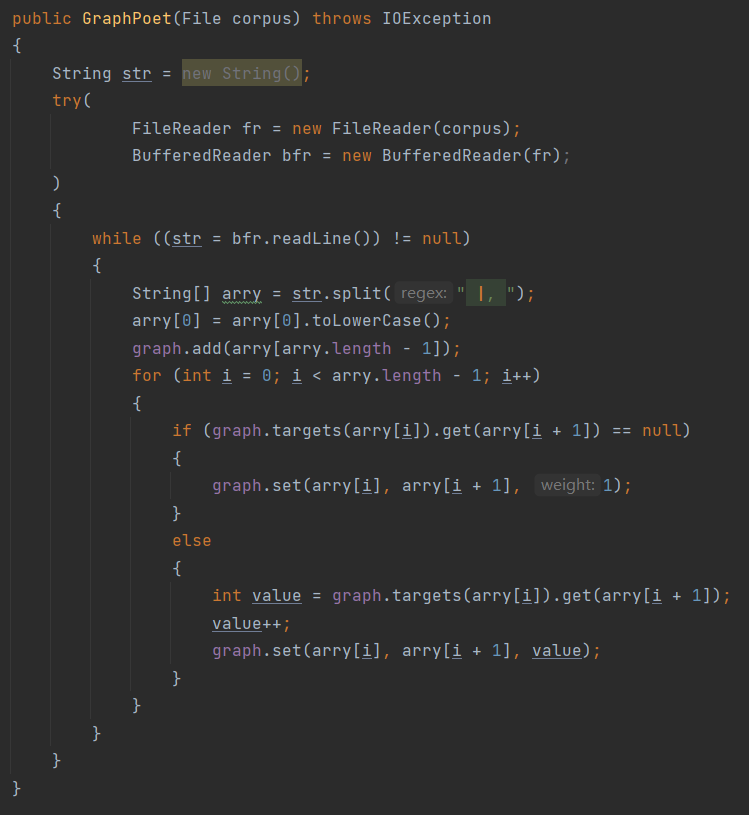
直接使用问题要求中所给的示例直接测试，较为准确





#### Implement GraphPoet

首先实现GraphPoet方法，打开文件后逐行读取并分割字符串，然后将字符串直接加入图中，并根据相关关系加入边。



然后实现poem方法，也就是根据我们的输入，然后在图中找到有关联的bridge进行尽可能的扩展

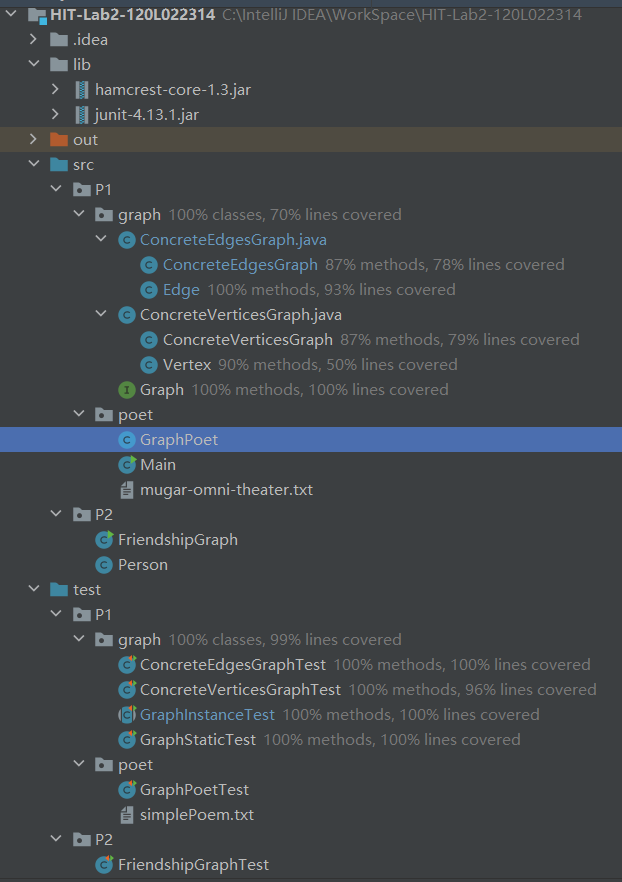


### Before you’re done

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。

在这里给出你的项目的目录结构树状示意图。



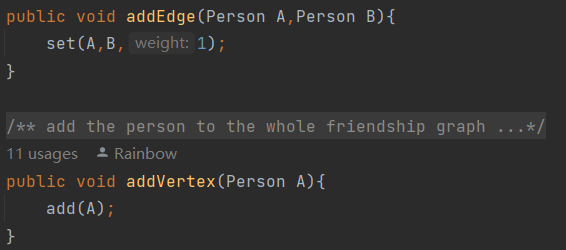
## Re-implement the Social Network in Lab1

在LAB1中我们已经完成了对一个friendship的图的构建，此任务就是想让我们利用我们已经实现的类和方法对这个friendship的实现进行改进和重写

### FriendshipGraph类

FriendshipGraph类继承自ConcreteEdgeGraph类，其中L用Person代替。

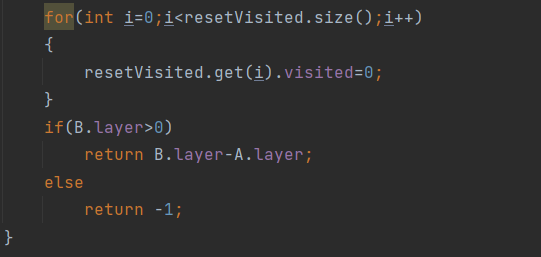
因此可以直接使用ConcreteEdgeGraph中的变量和方法进行复用



增加边和顶点的方法可以直接简单调用我们已经写好的add和set方法

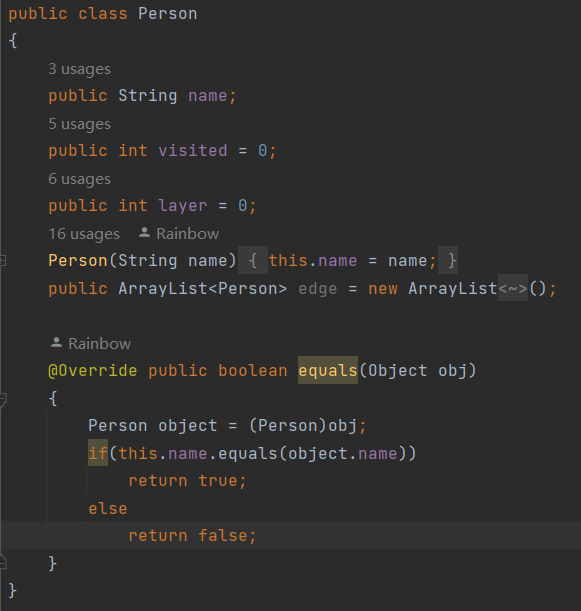
getDistance方法则采用广搜的方式，利用队列进行实现，其中用graph.targets来获取图中的顶点的边，进而获取距离





### Person类

直接使用Lab1的Person类即可，只是涉及到使用L.equals()方法，则需要为Person重写equals

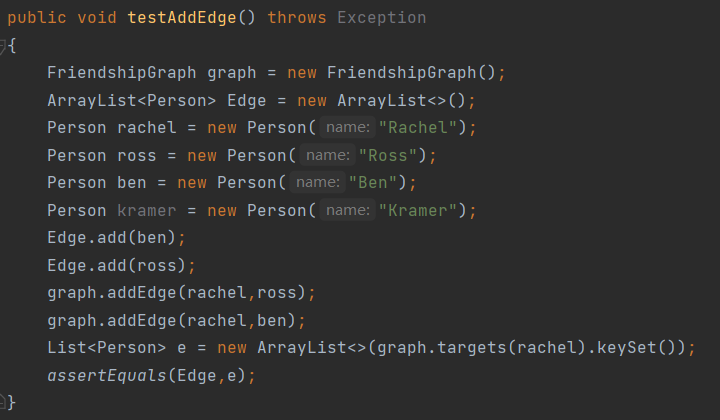


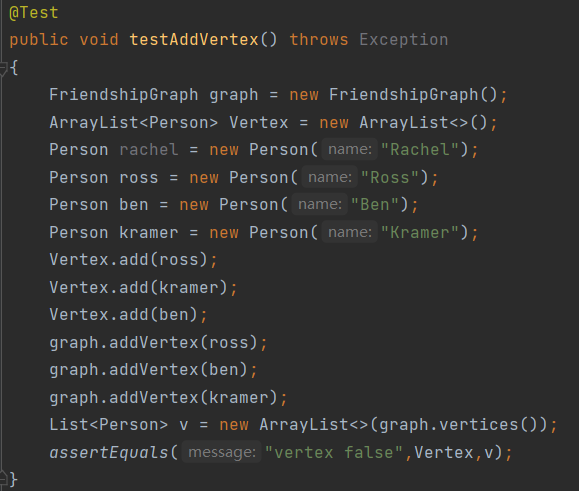
### 客户端main()

计算distance

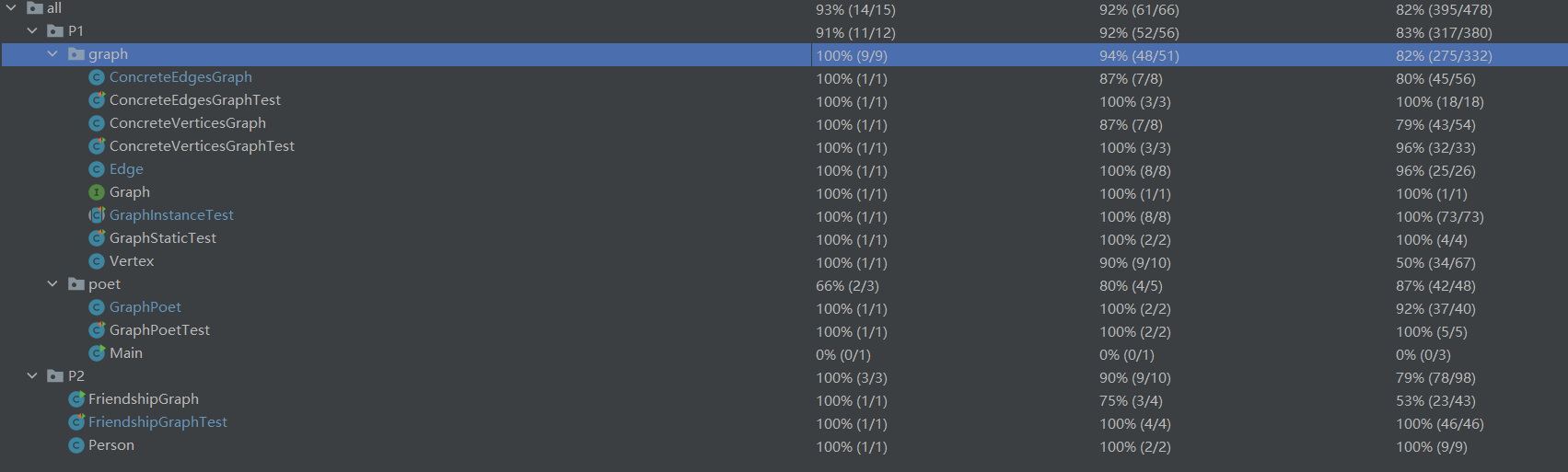


### 测试用例





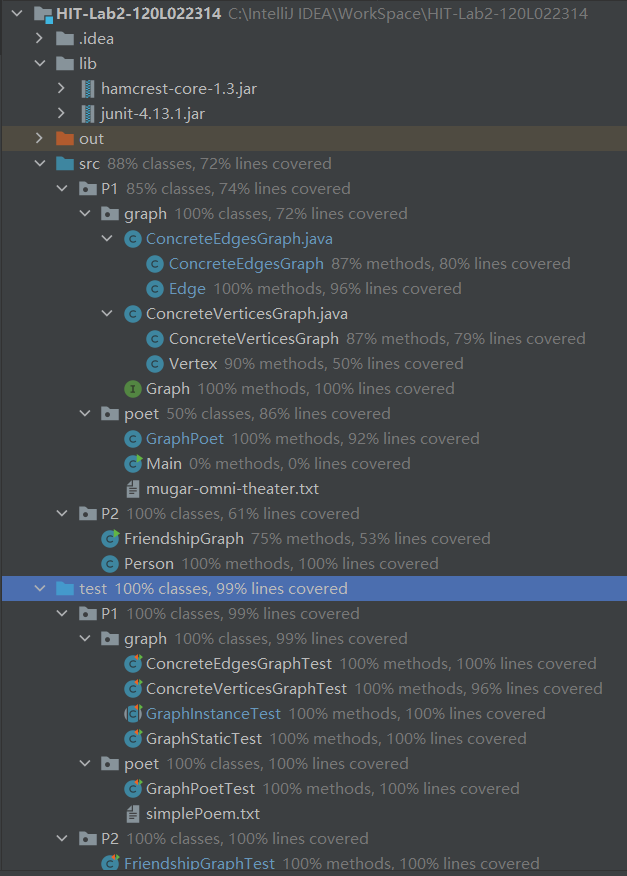




### 提交至Git仓库

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。



# 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 |
| 5.22 | 21：00-22：00 | 完成problem1 | 完成 |
| 5.23 | 20：00-22：00 | 完成problem2 and problem3 | 延期2小时 |
| 5.24 | 21：00-23：00 | 完成problem4 | 按时完成 |
| 5.25 | 20：00-23：00 | 完成3.2 | 延期1小时完成 |
| 5.26 | 10：00-12：00 | 完成实验报告 | 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 |
| 第二个实现时给的rep太少 | 在Vertex类中尽可能完善方法等，以满足rep中变量少而无法达到的要求 |
| 对poem walks的理解 | 仔细理解bridge的含义，修改方法 |
| 需要重写已经写过的类 | 继承方法后，利用继承来的方法简化操作 |

# 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

需要尽可能完善spec以方便我们在编写测试时对用户以及编程者的想法。在面对继承时，用到不同的类型，我们可能需要重写方法来满足我们的需求

## 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的时候更能从用户方面考虑，将用户能看见的东西定住，内部无所谓如何实现

1. 使用泛型和不使用泛型的编程，对你来说有何差异？

泛型需要考虑到多种因素的影响，而不使用泛型能够具体化来实现

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？

能够从用户的角度来看我们编写的代码，想要满足用户的需求，就必须尽可能满足spec；最初不是很适应这种方法

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？

能节省时间提高效率，减少工作量

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？

为了提高程序的安全性；但我个人并不喜欢这种操作

1. 关于本实验的工作量、难度、deadline。

工作量大，难度适中，时间适中

1. 《软件构造》课程进展到目前，你对该课程有何体会和建议？

教会我们面向对象开发，以及软件开发中的合适的开发形式以及一些细节处理