

**2022年春季学期  
计算学部《软件构造》课程**

**Lab 2实验报告**

|  |  |
| --- | --- |
| 姓名 | - |
| 学号 | - |
| 班号 | - |
| 电子邮件 | - |
| 手机号码 | - |

**目录**

[1 实验目标概述 1](#_Toc72249750)

[2 实验环境配置 1](#_Toc72249751)

[3 实验过程 1](#_Toc72249752)

[3.1 Poetic Walks 1](#_Toc72249753)

[3.1.1 Get the code and prepare Git repository 1](#_Toc72249754)

[3.1.2 Problem 1: Test Graph <String> 1](#_Toc72249755)

[3.1.3 Problem 2: Implement Graph <String> 1](#_Toc72249756)

[3.1.3.1 Implement ConcreteEdgesGraph 2](#_Toc72249757)

[3.1.3.2 Implement ConcreteVerticesGraph 2](#_Toc72249758)

[3.1.4 Problem 3: Implement generic Graph<L> 2](#_Toc72249759)

[3.1.4.1 Make the implementations generic 2](#_Toc72249760)

[3.1.4.2 Implement Graph.empty() 2](#_Toc72249761)

[3.1.5 Problem 4: Poetic walks 2](#_Toc72249762)

[3.1.5.1 Test GraphPoet 2](#_Toc72249763)

[3.1.5.2 Implement GraphPoet 2](#_Toc72249764)

[3.1.5.3 Graph poetry slam 2](#_Toc72249765)

[3.1.6 使用Eclemma检查测试的代码覆盖度 2](#_Toc72249766)

[3.1.7 Before you’re done 2](#_Toc72249767)

[3.2 Re-implement the Social Network in Lab1 2](#_Toc72249768)

[3.2.1 FriendshipGraph类 2](#_Toc72249769)

[3.2.2 Person类 3](#_Toc72249770)

[3.2.3 客户端main() 3](#_Toc72249771)

[3.2.4 测试用例 3](#_Toc72249772)

[3.2.5 提交至Git仓库 3](#_Toc72249773)

[4 实验进度记录 3](#_Toc72249774)

[5 实验过程中遇到的困难与解决途径 3](#_Toc72249775)

[6 实验过程中收获的经验、教训、感想 4](#_Toc72249776)

[6.1 实验过程中收获的经验和教训 4](#_Toc72249777)

[6.2 针对以下方面的感受 4](#_Toc72249778)

# 实验目标概述

1. 掌握ADT的设计，理解OOP

2. 学会ADT规约，测试，AF，RI，防止表示泄露

3. 初步使用泛型

# 实验环境配置

简要陈述你配置本次实验所需环境的过程，必要时可以给出屏幕截图。

特别是要记录配置过程中遇到的问题和困难，以及如何解决的。

1. 硬件环境：X64CPU；1.6GHz；8G RAM；512G SSD

2. 软件环境：Ubuntu 18.04.5 LTS 64位

3. 开发工具：IDEA

4. 配置过程中遇到的困难

在这里给出你的GitHub Lab2仓库的URL地址（Lab2-学号）。

# 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

首先要根据给出的接口graph实现两个实现类，分别是基于边的类和基于点的类，然后使用实现的类完成Poet Graph中的两个函数。

### Get the code and prepare Git repository

在仓库里面下载到本地即可。

### Problem 1: Test Graph <String>

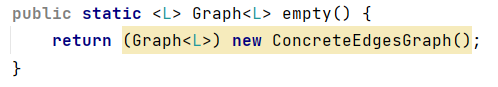
以下各部分，请按照MIT页面上相应部分的要求，逐项列出你的设计和实现思路/过程/结果。

思路：这一节的任务是实现Graph<String>的测试设计，需要实现对其中的每个方法进行等价类划分的测试。

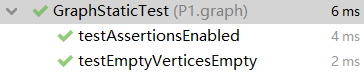
实现过程：

① GraphStaticTest的设计

这部分的设计基本已经完成，需要完善的是Graph中empty的设计，这个函数是要返回一个新的空图，使用new创建一个新的即可。



测试结果如下：

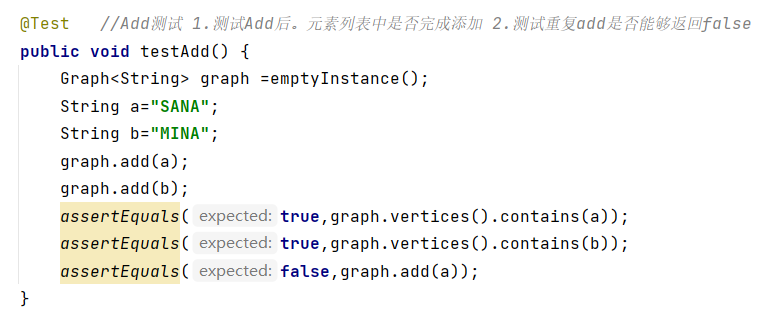


②GraphInstanceTest

这部分主要是针对Graph中的每个方法进行测试，具体的测试设计如下：

1.Add测试，代码如下：

测试战略：1.测试Add后。元素列表中是否完成添加 2.测试重复add是否能够返回false



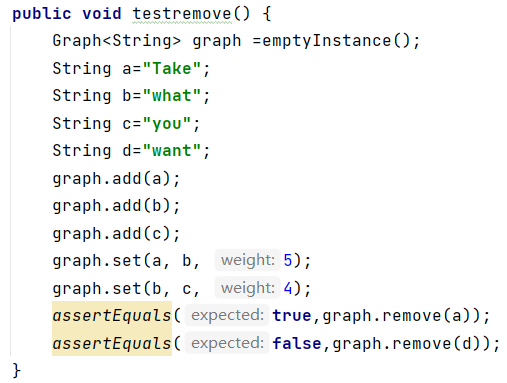
2.Set测试，代码如下：

测试战略：1.测试添加边时，若之前存在边，能否正常返回旧边的值 2.添加权重不符合输入要求的边，能否返回-1



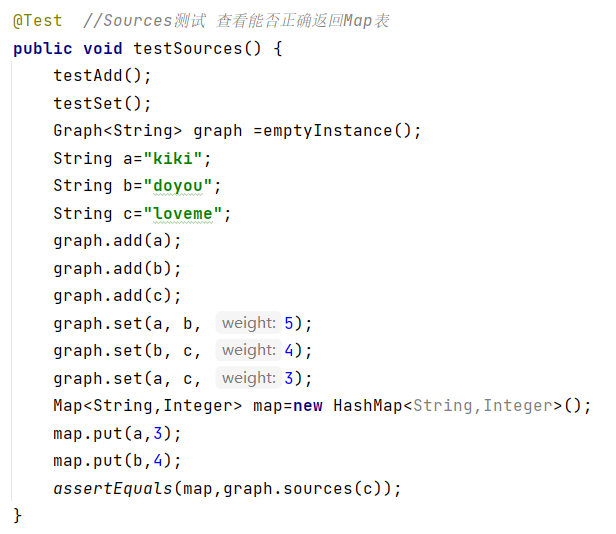
3.remove测试，代码如下：

测试战略：1.删除表中存在的一个元素，查看能否返回true 2.删除不存在的一个元素，查看能否返回false



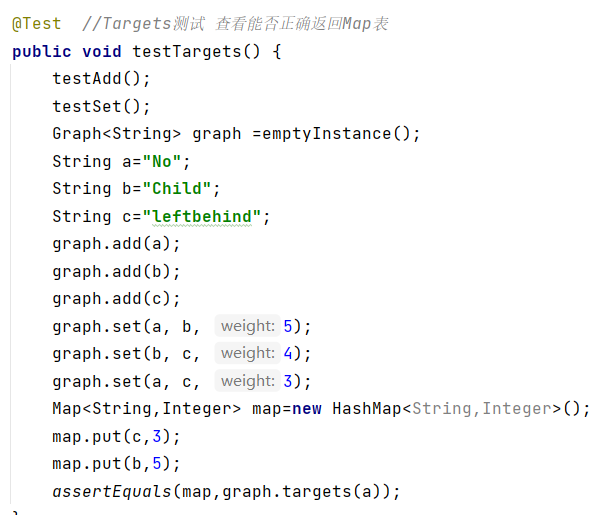
4.Sources测试，代码如下：

测试战略：查看能否正确返回Map表



5.Targets类测试，代码如下：

测试战略：查看能否正确返回Map表



### Problem 2: Implement Graph <String>

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。

#### Implement ConcreteEdgesGraph

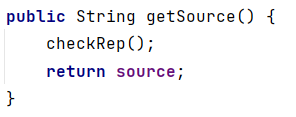
①Edge类的设计：

边的内容包含源点，终点，以及其权重，这里设计为immutable，防止被外界更改。

Edge类中设计了多个方法，以便进行后续的操作。

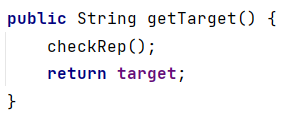
1.getSoure

返回边的源点的信息，直接return即可，代码如下：



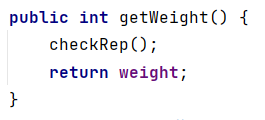
2.getTarget

返回边的终点信息，直接return即可，代码如下：



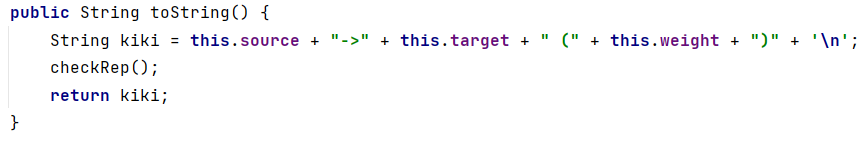
3.getWeight

返回边的权重信息，直接return即可，代码如下：



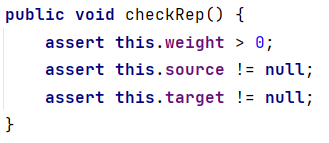
4.toString

打印边的信息，代码如下：



5.checkRep

检查内容是否符合要求，代码如下：



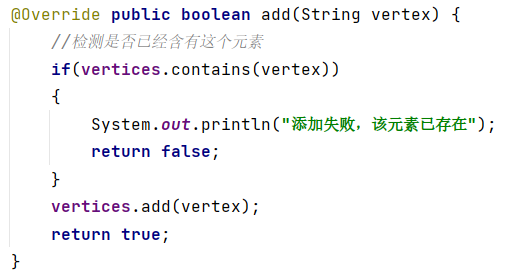
②ConcreteEdgesGraph的设计

类中各个方法的设计思路如下：

1.add

功能：实现节点的添加

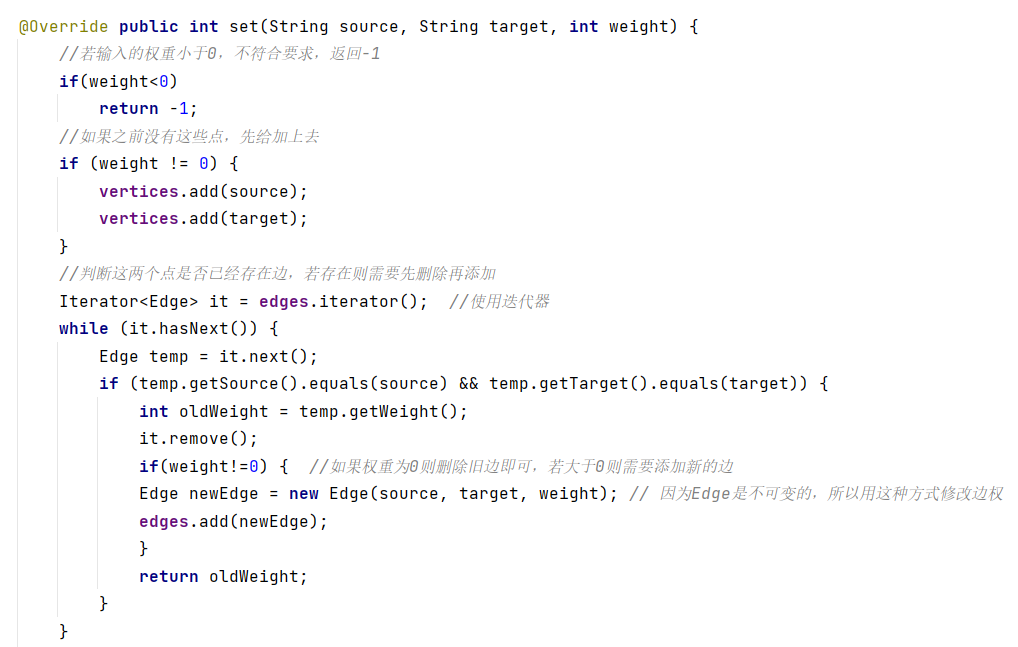
设计思路：首先判断是否已经存在同名节点，若存在则返回false，若不存在则可以添加并返回true



2.set

功能：添加一个边

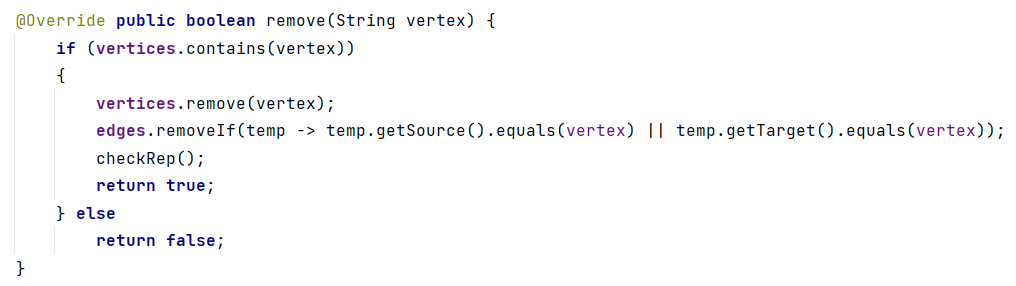
设计思路：先判断输入的权重是否小于0，若小于0则直接返回-1。若是输入的边的源点或结点在图中并没有，则先添加上。之后进行如下处理：若是这两点之前存在边，则需要先删除再添加新的边，有一点需要注意的是，若是输入的新的权重为0，则只删除不添加；若是这两点之前不存在边，如果输入的权重为0，则直接返回0，不添加边，若是不为0，则添加新边并返回0。



3.remove

功能：删除一个元素

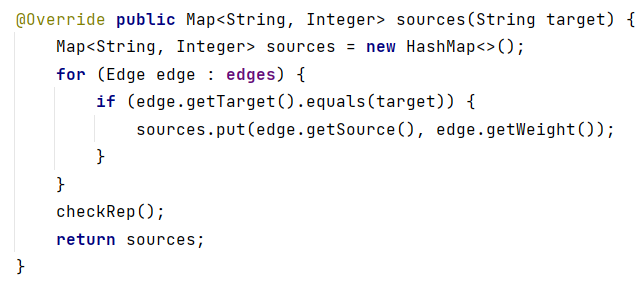
设计思路：首先检查表中是否有这个元素，如果有就删除然后返回true，否则返回false



4.source

功能：输入一个点，返回和这个点之间有边连接的源点

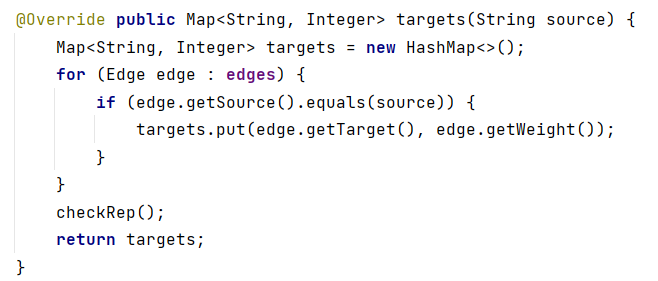
设计思路：遍历边表，逐个找出要求的点



5.target

功能：输入一个点，返回和这个点之间有边连接的终点

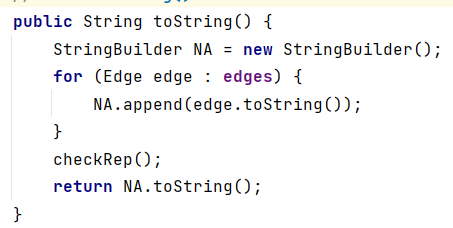
设计思路：遍历边表，逐个找出要求的点



6.toString

功能：返回所有边的信息

设计思路：遍历边表，使用append将信息收集



#### Implement ConcreteVerticesGraph

①Vertex类的设计

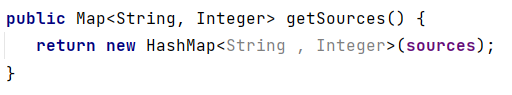
该类的内容包括：1.该元素的名称 2.包含与该点相连的源点及边的权重的map表3. 包含与该点相连的终点及边的权重的map表

设计的方法如下：

1.getSource

功能：返回该Vertex中的source map表

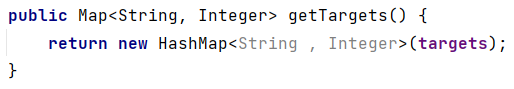
设计思路：返回即可



2.getTarget

功能：；返回该Vertex中的target map表

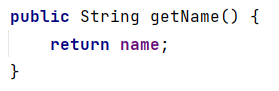
设计思路：返回即可



3.getName

功能：返回该Vertex中的name

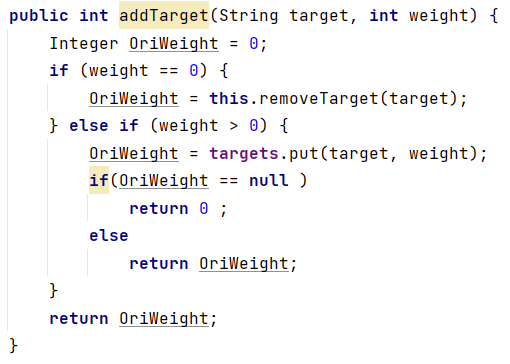
设计思路：返回即可



4.setTarget

功能：实现在Vertex中的Target表设置

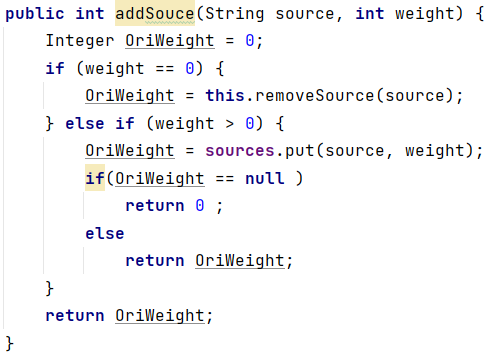
设计思路：若要添加边的权重为0，则只需要把原来的边删除即可，不再添加新的，否则就是put新边即可



5.setSource

功能：实现在Vertex中的Target表设置

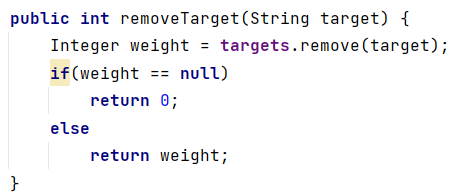
设计思路：与setTarget类似



6.removeTarget

功能：删除终点

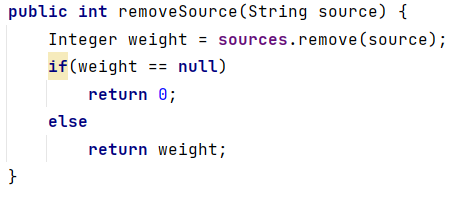
设计思路：若不存在，则返回0，否则正常删除



7.removeSource

功能：删除源点

设计思路：若不存在，则返回0，否则正常删除



8.

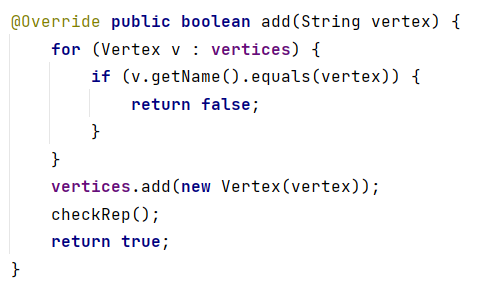
② ConcreteVerticesGraph设计

方法的设计如下：

1.add

功能：添加节点

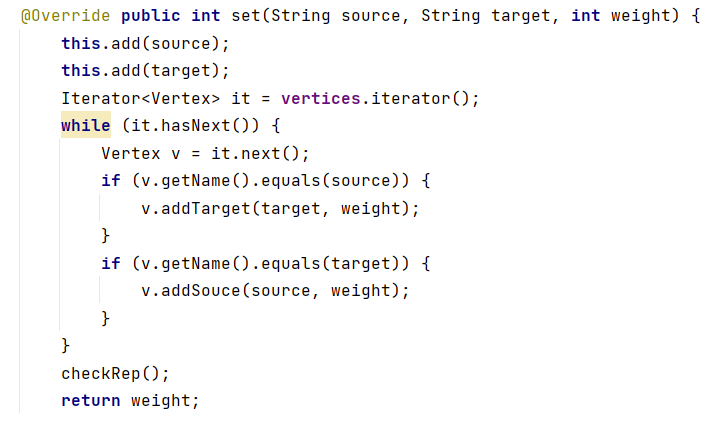
设计思路：首先检查是否有同名的，若没有则添加



2.set

功能:添加边

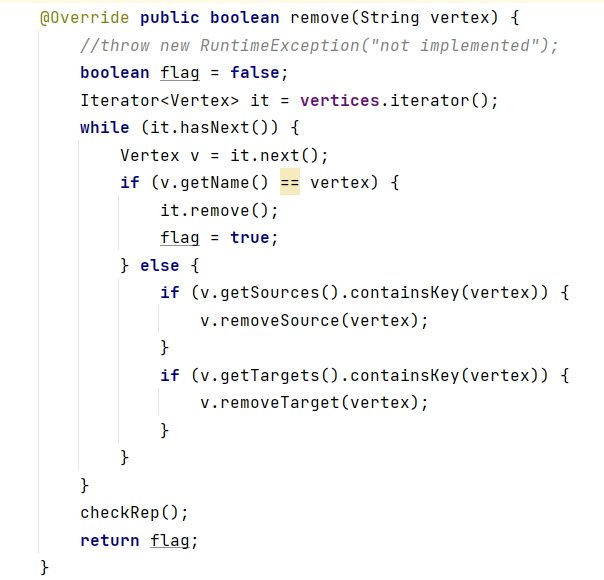
设计思路：在集合中找到两点然后添加信息



3.remove

功能：删除某个元素

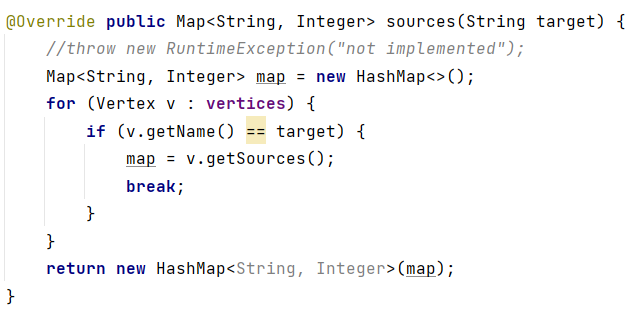
设计思路：将与该点相关的边全部删除



4.sources

功能：调用vertex类的sources

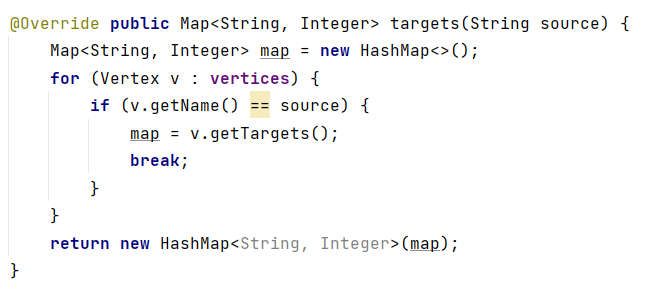
设计思路：new一个新表然后复制一下信息



5.targets

功能：调用vertex类的targets

设计思路：new一个新表然后复制一下信息



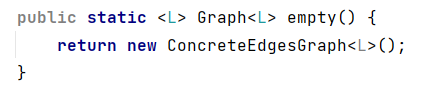
### Problem 3: Implement generic Graph<L>

#### Make the implementations generic

将对应的类型换为L即可

#### Implement Graph.empty()

调用一个具体的实现即可：



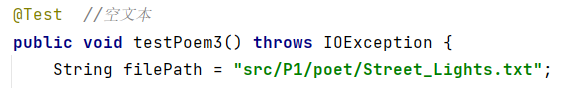
### Problem 4: Poetic walks

#### Test GraphPoet

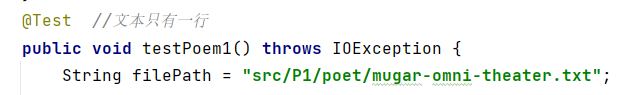
测试策略：

①读入文件的种类：

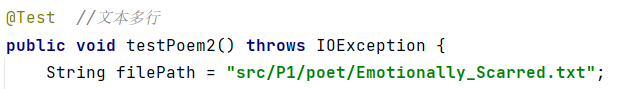
1. 空文件



1. 一行文本

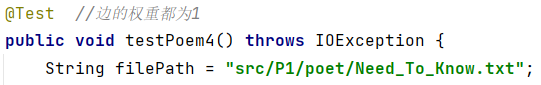


1. 多行文本

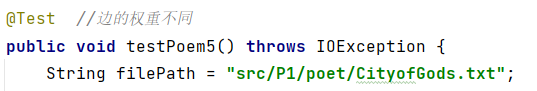


②边的权重测试：

1. 权重都为1

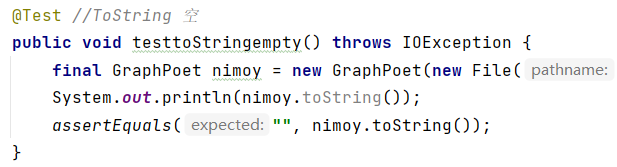


1. 权重有多种情况

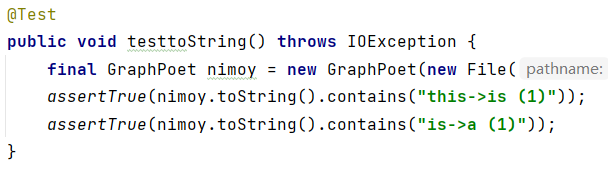


③toString测试：

1. 空文件



1. 非空



#### Implement GraphPoet

函数功能：建立诗对应的有向图

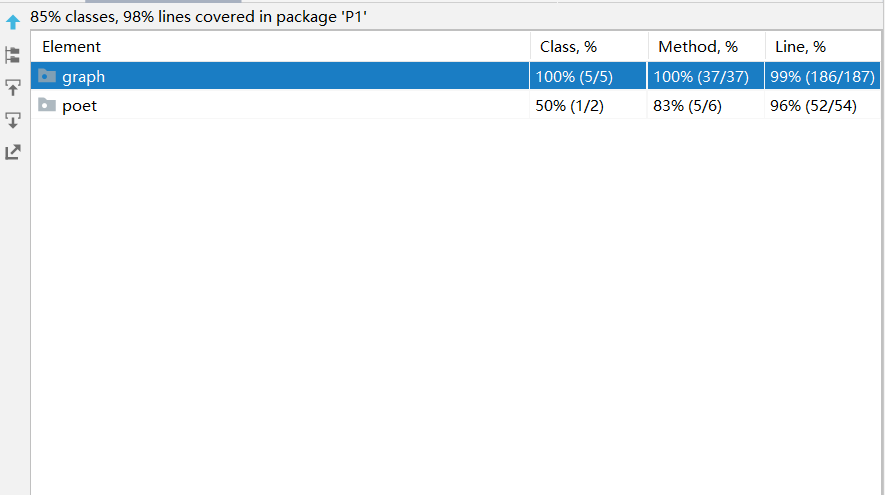
设计思路：打开文件，每次读取一行，如果长度为0，则读取下一行，为1，则将唯一的一个放进去，读取下一行。如果一行长度大于1，那么首先以“ ”为分割符，分割为多个String，然后将第一个放入其中。从第二个开始循环，首先将当前String放入其中，如果之前不存在这个顶点，便加入一条边（使用set）,从前一个字符到当前字符。如果之前存在，便判断一下上一个点与当前点之前是否有边，有的话若方向相同则权重加一，不同设为1。没有的话则直接设置。

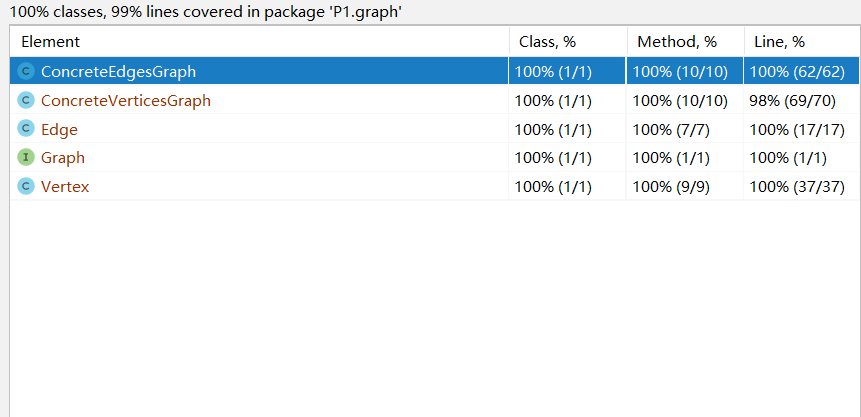


#### Graph poetry slam

同样采用某一个到下一个的办法，不过判断当前这个不在之后，便直接到下一个。当前这个在，下一个不在，那下标直接加2。

### 使用Eclemma检查测试的代码覆盖度





### Before you’re done

请按照<http://web.mit.edu/6.031/www/sp17/psets/ps2/#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。

HIT-Lab2-120L031815

doc

Lab2-120L031815-Report.docx

lib

hamcrest-core-1.3.jar

junit-4.13.1.jar

src

P1

graph

ConcreteEdgeGraph.java

ConcreteVerticesGraph.java

Graph.java

poet

GraphPoet.java

Main.java

mugar-omni-theater.txt

P2

FriendshipGraph.java

Person.java

test

P1

graph

ConcreteEdgeGraphTest.java

ConcreteVerticesGraphTest.java

GraphInstanceTest.java

GraphStaticTest.java

poet

GraphPoetTest.java

empty.txt

oneLine.txt

manyLines.txt

complexLines.txt

P2

FriendshipGraphTest.java

## Re-implement the Social Network in Lab1

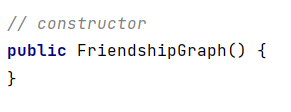
与Lab1相比，这个产生了变化，因为这里给出的Graph.java接口是有向有权图，而原来的图是无向图，并且没有权重。因此将之修改为有向图，权重全部设置为1。

### FriendshipGraph类

给出你的设计和实现思路/过程/结果。

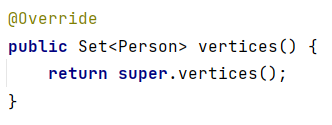
FriendshipGraph类实现了如下的方法：

1.构造方法



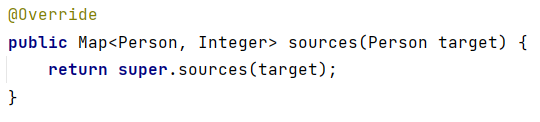
2.vertices

重写了该方法，让其返回super类的对应返回值



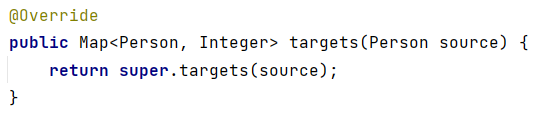
3.sources

重写了该方法，让其返回super类的对应返回值



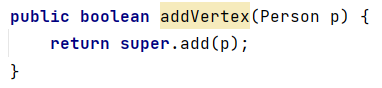
4.targets

重写了该方法，让其返回super类的对应返回值



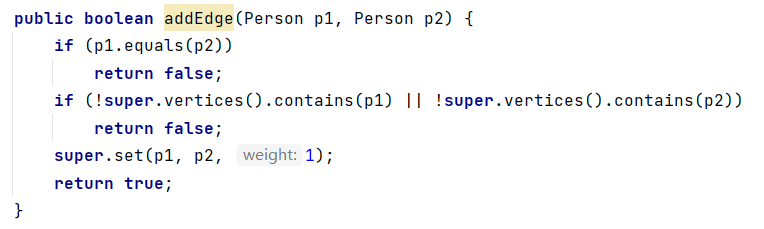
5.addVertex

调用add方法



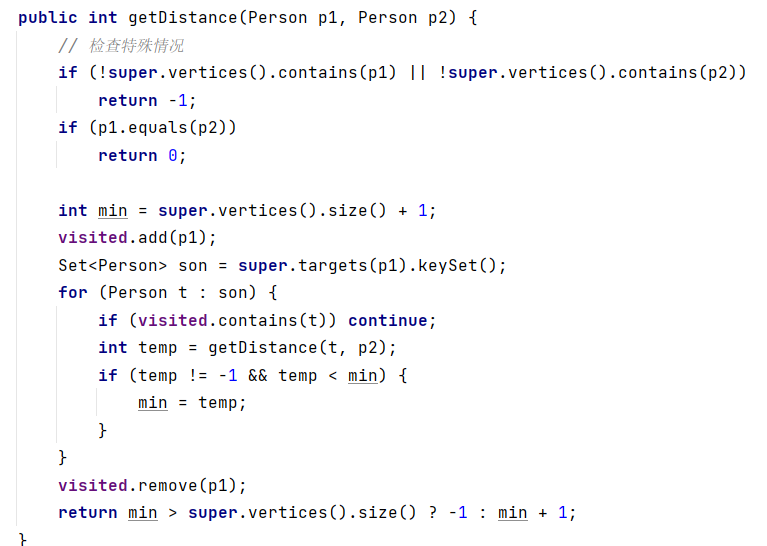
6.addEdge

判断是否为同一个点，若不是则使用set添加边



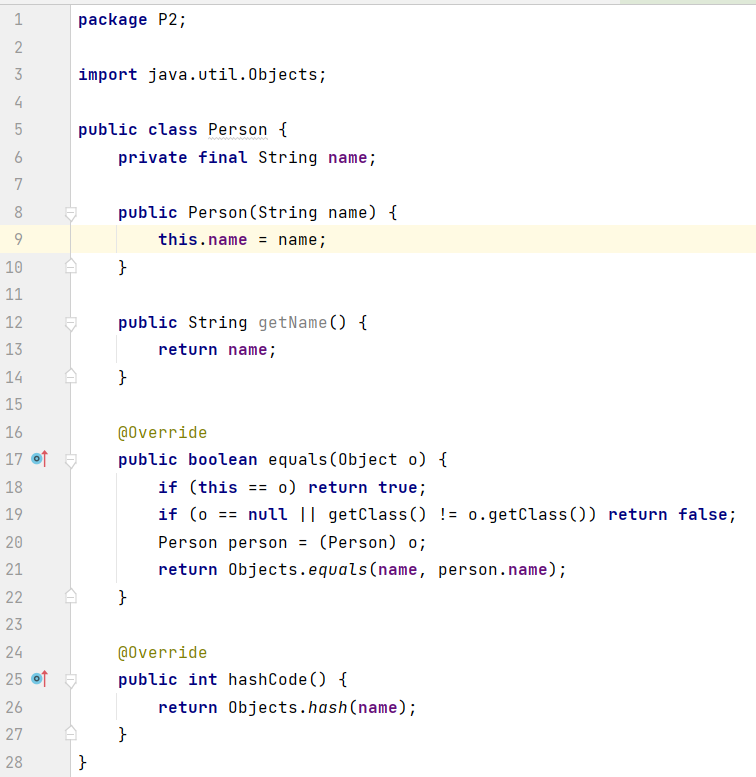
7.getDistance

首先对异常情况进行判断，若图中不包含参数顶点或两参数顶点相同，则直接返回，之后开始递归判断。先令最短距离min为顶点数+1（保证若存在通路，则实际的最短距离一定比其小），取p1顶点的子顶点，调用函数本身递归地寻找子顶点到p2的最短距离，记为temp，若temp非-1且比min小，则令min=temp，最后根据min的值判断函数的返回值。



### Person类

Person类设置了一个成员变量String name表示其名字，并重写了equals与hashcode方法方便对不同的Person类进行比较。与Lab1不同的是，Lab1中的Person类还具有一个表示其在关系矩阵中位置的变量。由于Lab2实现时避开了关系矩阵，故删掉了该成员变量。



### 客户端main()

此处的客户端与Lab1中的客户端保存一致

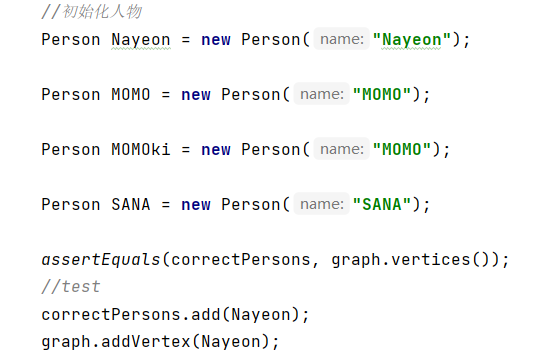
### 测试用例

给出你的设计和实现思路/过程/结果。

对FriendGraph中的三个函数，测试用例设计如下：

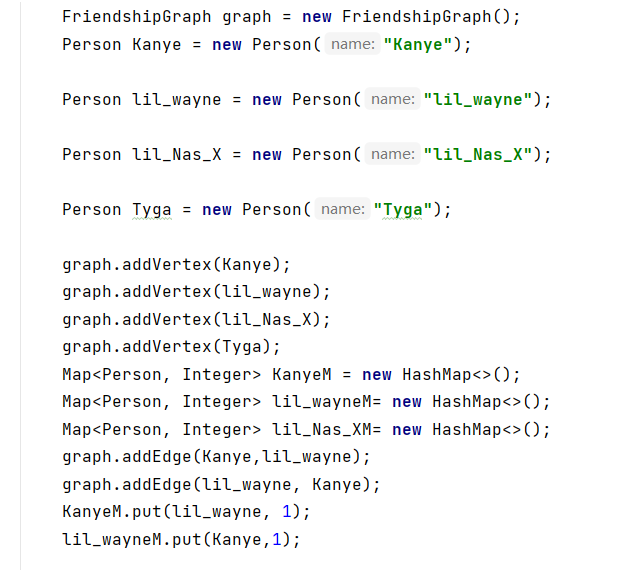
①addVertex的测试：

主要测试addVertex能否正确添加人物元素，设置一个同名人物，判断函数能否返回提示信息。



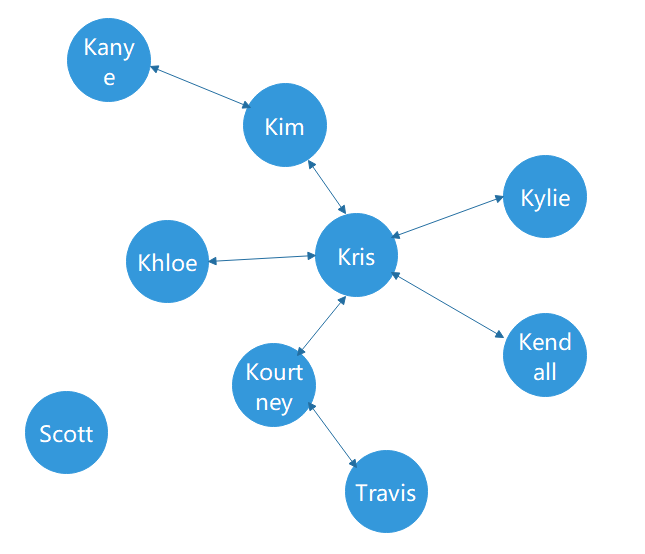
②addEdge的测试：

设计了addEdgeTest。验证添加边后图的邻接矩阵是否符合预期。并在向图中添加新的顶点后再次验证，以确保程序正确无误。



③getDistance的测试：

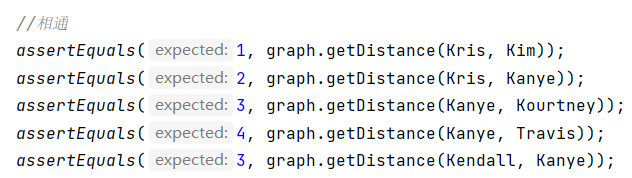
测试如下的关系网：



网络中共有9个人物，下面对以下几种情况的getDistance进行测试：

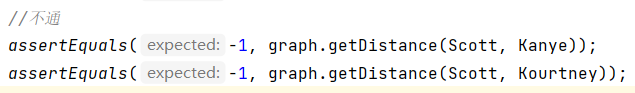
①相通的两点

对以下五组点进行测试，检查其是否能正确返回距离



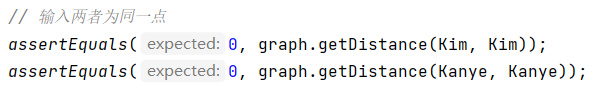
②不相通的两点

对以下两组点进行测试，检查其是否能返回-1



③输入同一点

测试两个Kim，两个Kanye的距离



④输入不存在的点



### 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。

在这里给出你的项目的目录结构树状示意图。

上面已经有了。

# 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月21号 | 10：00-12：00 | 阅读网页，理解任务，构思 | 完成 |
| 5月21号 | 18：00-21：00 | 编写ConcreteVertexGraph | 完成Vertex |
| 5月22号 | 14：00-16：00 | 编写ConcreteVertexGraph | 完成小部分 |
| 5月23号 | 13：45-23：59 | 编写ConcreteVertexGraph | 完成大部分 |
| 5月24号 | 13：45-23:59 | 编写ConcreteVertexGraph,  ConcreteEdgeGraph | 完成 |
| 5月25号 | 9：45-17：30 | 修改完善实现类 | 完成 |
| 5月26号 | 13：45-17：00 | 撰写Poet部分 | 完成 |
| 5月27号 | 13：45-23：59 | 撰写P2部分 | 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 |
| 设计ADT时有很多概念不是很清楚，对于设计的整体思路不是特别清晰 | 回顾课程PPT及上网查找资料，进而对ADT设计有了更深的认识 |

# 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

编写ADT时遇到了很多困难，但同时在这实践的过程中，将课堂上学到的理论知识，在实践中进行了检验，更加巩固了这一部分的设计知识，很大程度上提高了自身软件设计的水平，更意识到尤其是对于计算机系的学生，知识绝不能只停留在课本，而是应该在实现中，在代码中实践，这样才能够学习到真正的计算机知识。

## 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编程比较繁琐，将属性方法打包放在一起，编程更具模块化。

1. 使用泛型和不使用泛型的编程，对你来说有何差异？

泛型编程更加具有普适性，但同时，离开了具体类型，使设计更加抽象。

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？

测试十分方便，测试用例优先使编程更方便快捷，需求也更明确。

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？

代码更加简洁，模块化更高，提高编程效率

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？

。。。好像没有P3。

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？

意义在于，编程时思路更加清晰，使代码的整体思路保持流畅，愿意。

1. 关于本实验的工作量、难度、deadline。

工作量大，难度适中，时间安排也较为合理，能够按时完成。

1. 《软件构造》课程进展到目前，你对该课程有何体会和建议？

感觉到这门课程对于以后的就业帮助十分大，一定要好好学习，无建议。