

**2018年春季学期  
计算机学院大二软件构造课程**

**Lab 2实验报告**

|  |  |
| --- | --- |
| 姓名 | 胡钦 |
| 学号 | 116370103 |
| 班号 | 1637101 |
| 电子邮件 | a591027895@qq.com |
| 手机号码 | 18846782550 |

**目录**

[1 实验目标概述 1](#_Toc507927440)

[2 实验环境配置 1](#_Toc507927441)

[3 实验过程 1](#_Toc507927442)

[3.1 Poetic Walks 1](#_Toc507927443)

[3.1.1 Get the code and prepare Git repository 1](#_Toc507927444)

[3.1.2 Problem 1: Test Graph <String> 1](#_Toc507927445)

[3.1.3 Problem 2: Implement Graph <String> 1](#_Toc507927446)

[3.1.3.1 Implement ConcreteEdgesGraph 2](#_Toc507927447)

[3.1.3.2 Implement ConcreteVerticesGraph 2](#_Toc507927448)

[3.1.4 Problem 3: Implement generic Graph<L> 2](#_Toc507927449)

[3.1.4.1 Make the implementations generic 2](#_Toc507927450)

[3.1.4.2 Implement Graph.empty() 2](#_Toc507927451)

[3.1.5 Problem 4: Poetic walks 2](#_Toc507927452)

[3.1.5.1 Test GraphPoet 2](#_Toc507927453)

[3.1.5.2 Implement GraphPoet 2](#_Toc507927454)

[3.1.5.3 Graph poetry slam 2](#_Toc507927455)

[3.1.6 Before you’re done 2](#_Toc507927456)

[3.2 Re-implement the Social Network in Lab1 2](#_Toc507927457)

[3.2.1 FriendshipGraph类 2](#_Toc507927458)

[3.2.2 Person类 3](#_Toc507927459)

[3.2.3 客户端main() 3](#_Toc507927460)

[3.2.4 测试用例 3](#_Toc507927461)

[3.2.5 提交至Git仓库 3](#_Toc507927462)

[3.3 The Transit Route Planner（选做，额外给分） 3](#_Toc507927463)

[4 实验进度记录 3](#_Toc507927464)

[5 实验过程中遇到的困难与解决途径 3](#_Toc507927465)

[6 实验过程中收获的经验、教训、感想 4](#_Toc507927466)

# 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 示外泄（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

实验环境设置请参见 Lab-0 实验指南。

除此之外，本次实验需要你在 Eclipse IDE 中安装配置 EclEmma（一个用于 统计 JUnit 测试用例的代码覆盖度的 plugin）。请访问 http://www.eclemma.org，了 解 EclEmma 并学习其安装、配置和使用。

本次实验在 GitHub Classroom 中的 URL 地址为： https://classroom.github.com/a/z9utaaos 请访问该 URL，按照提示建立自己的 Lab2 仓库并关联至自己的学号。

本地开发时，本次实验只需建立一个项目，统一向 GitHub 仓库提交。实验包 含的 2(+1)个任务分别在不同的目录内开发，具体目录组织方式参见各任务最后 一部分的说明。请务必遵循目录结构，以便于教师/TA 进行测试。

在这里给出你的GitHub Lab2仓库的URL地址（Lab2-学号）。

https://github.com/ComputerScienceHIT/Lab2-1163710103-3

# 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

在这里简要概述你对该任务的理解。

通过语料库生成亲和图。根据输入的字符串来生成一首诗。

### Get the code and prepare Git repository

如何从GitHub获取该任务的代码、在本地创建git仓库、使用git管理本地开发。实验手册上有，不再叙述。

### 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。

设计思路：这是一个接口，所以我们需要用类去实现。

过程：完成边和角类对它进行实现。

结果：这个接口中不需要做什么操作。

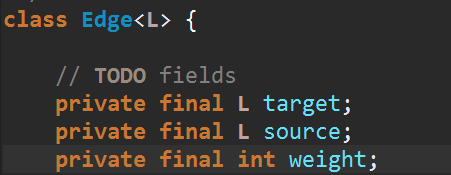
### Problem 2: Implement Graph <String>

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

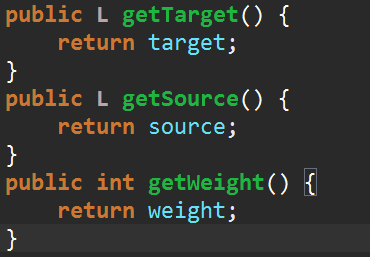
设计思路：先实现String类型的这个类，再改为通用的泛型。完成Edge类，自己填充属性和方法。从而能够借助这个类来实现graph的各个方法。

过程：

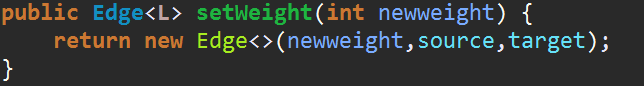
1.确定Edge的属性：

只需要记录下目标点，起始点和权值即可

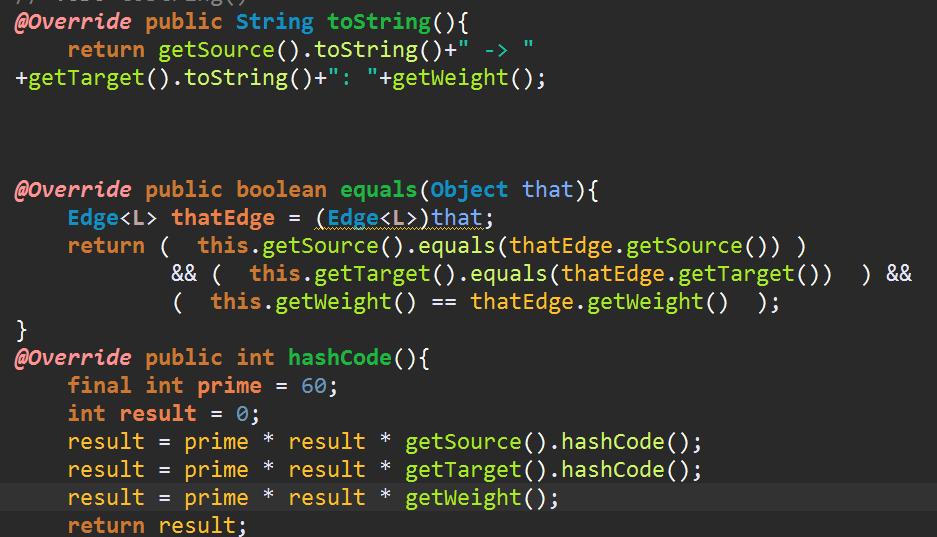
1. 完成get方法，可以用来获取属性：



1. 完成设置权值方法：

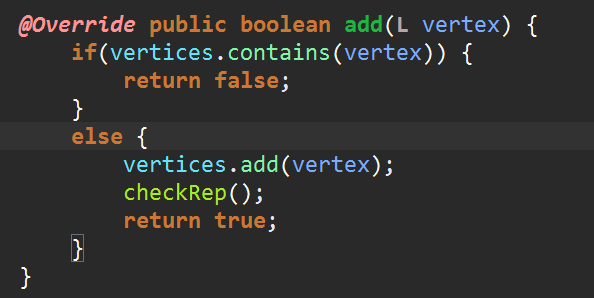


1. 书写toString和equals方法，hashcode方法：



1. 接下来就可以用Edge实现graph中的操作了：
2. add:

先判断接收到的参数是否已经在vertices中存在，不存在则添加进去：



1. set:

先判断是否已经有这条边，如果已经有了，则在原来的基础上进行修改权值即可，如果没有则增添这条边。记得添加pre变量来记录原来的权值进行返回：



1. remove：

先判断点是否在图中，如果在，那么直接remove，并且对边集合进行清理：将所有与这个点有关系的边都清理掉。如果不在，那么返回false说明不存在这个点。



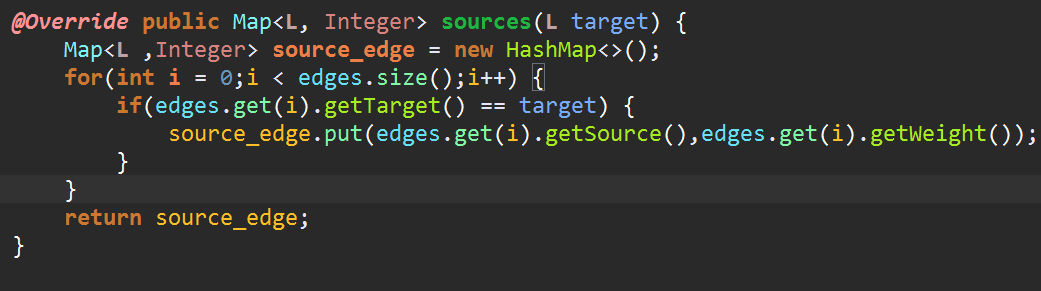
1. vertices：

直接返回已经建立的集合vertices，此处比较简单，但是后来用点来进行实现的时候比较难。

1. sources：

实现得到一个点，输出所有它的起始点并且输出这条连接他们的边的权值：

新建一个Map在遍历的同时往里面加元素即可。

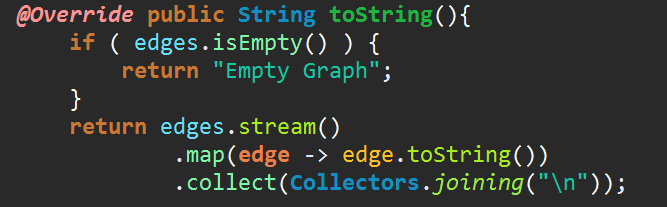


1. targets：

实现和上面的差不多,得到一个起始点，输出它的终结点和对应权值。

1. toString方法：

在保证不是空的图的基础上返回它的实现方式和内存地址。

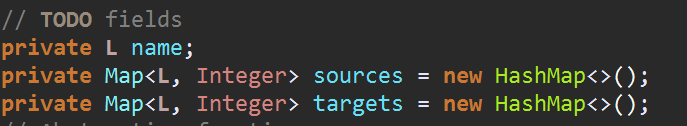


实现结果：书写测试函数之后，都通过了测试。

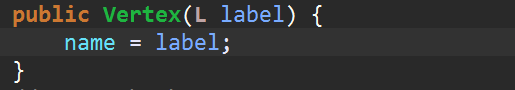
#### Implement ConcreteVerticesGraph

实现思路：通过点来进行实现graph的所有功能。

1. 先实现vertex类：
2. 确定属性：只需要名字，起始点对应权值，终结点对应权值，就可以完整的表示一个图

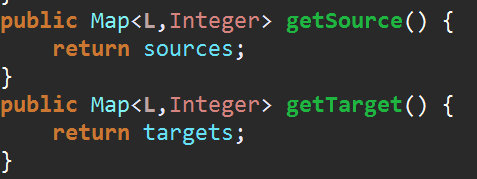


1. 构造方法：



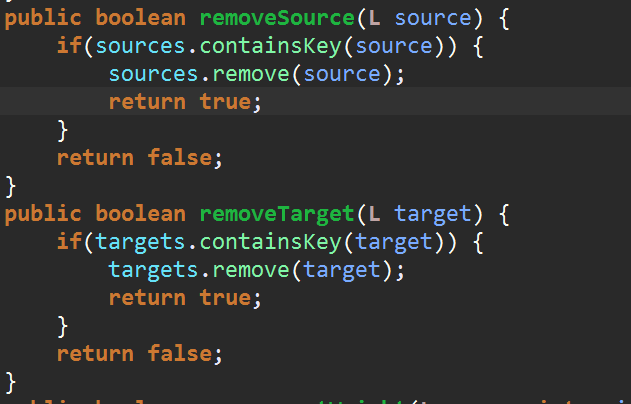
一个名字参数即可创建一个点类

1. get方法，能够得到所有起始点和对应边权值，所有终结点和对应边权值，直接返回属性中的两个map即可：



1. remove：

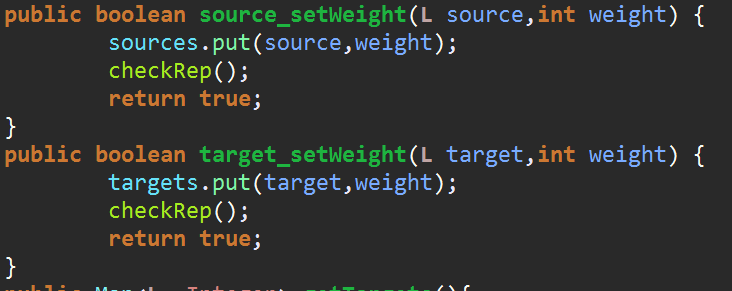
比起用边进行实现，用点进行实现的时候因为内部属性比较麻烦，在外面对其进行操作不太方便，于是在内部多写一些方法能更方便的实现graph接口中的方法.于是这里比起用边来实现，用点实现多加入了remove方法。



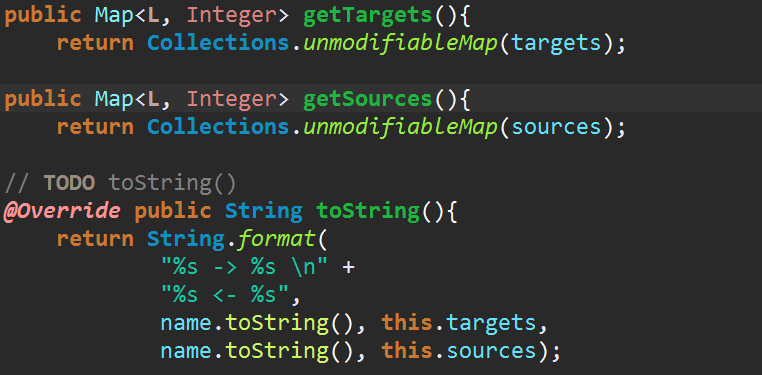
可以根据参数来清除这个起始点或者终结点。

1. set方法：

这里的set方法就比较麻烦了，因为根据用户输入的点的类型（起始点或终结点）要能够对对应的map进行添加和修改。

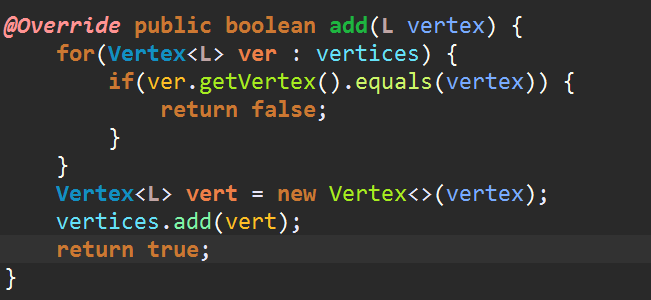


1. toString和对应的辅助函数



写完之后就可以实现所有的graph方法了：

1. 实现所有功能：
2. add：



在所有点中找到用户给的vertex就添加失败，只有不存在的条件下才添加成功

1. set：

（代码过长不粘贴了）

设置两个Vertex类型的变量，在图中进行遍历，当一个为source，一个为target时，判断weight，看是不存在的边（添加）还是已存在的边（修改），还是进行删除。

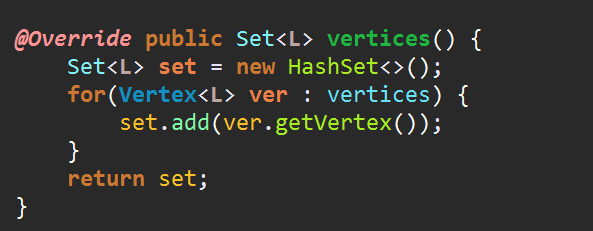
1. remove：

先把图中名字为vertex的点删除，再遍历所有的点中的map寻找与之有关的边，删除这对键值对。

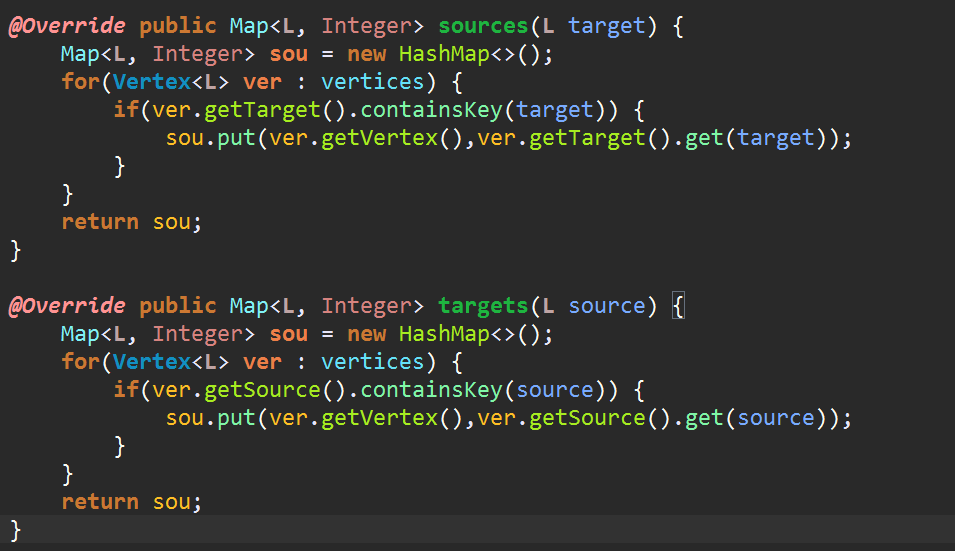


1. Vertices

将原来的List变为Set返回即可：



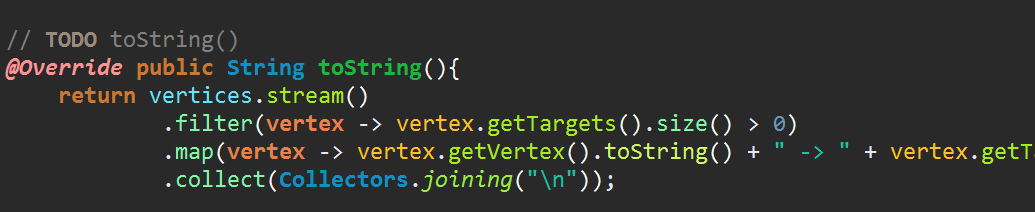
1. sources和targets



去vertices中的map键值对中找到所有要的键值对并且存起来返回。

1. toString

和edge的实现有些类似：



实验结果：测试用例全部通过。

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

#### Make the implementations generic

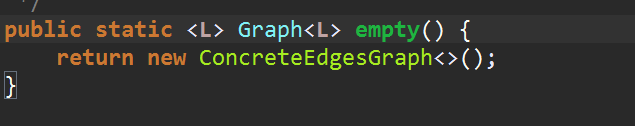
实验思路：之前实现了String的泛型，现在改为通用泛型L，只要把原来所有的String都改为L，再稍加修改即可。

实验过程：查找出所有的String，加以修改为L，让它适用于泛型。

同时修改测试函数，让它以String的泛型实例进行测试。

实验结果：完美。

#### Implement Graph.empty()



### Problem 4: Poetic walks

#### Test GraphPoet

在这里自己编写一个资料库，“test.txt”之后成其亲和图，以此来实例化一个GraphPeop对象，然后调用其所有的方法，然后判断结果与预期是否一样。

#### Implement GraphPoet

先定义语料库

#### Graph poetry slam

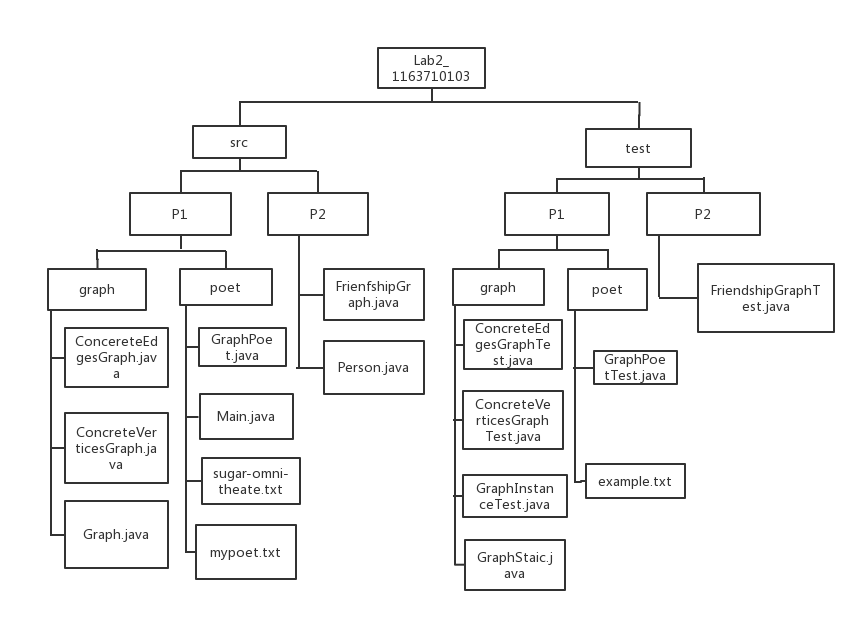
输入代码之后比较即可看见结果

### 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

在这里简要概述你对该任务的理解。

### FriendshipGraph类

给出你的设计和实现思路/过程/结果。

### Person类

给出你的设计和实现思路/过程/结果。

### 客户端main()

给出你的设计和实现思路/过程/结果。

### 测试用例

给出你的设计和实现思路/过程/结果。

### 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。

## The Transit Route Planner（选做，额外给分）

请自行组织本节内的目录结构。

# 实验进度记录

请尽可能详细的记录你的进度情况。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 4/5 | 晚上 | 完成P1 | 失败 |
| 4/6 | 晚上 | 完成P1 | 差测试用例 |
| 4/7 | 晚上 | 完成P1 | 成功 |
| 4/8 | 全天 | 完成P2 | 失败 |

# 实验过程中遇到的困难与解决途径

一开始觉得时间应该够，结果清明节有事导致差点不能按时完成。非常遗憾。

以后一定要提前写实验。

# 实验过程中收获的经验、教训、感想

本节除了总结你在实验过程中收获的经验和教训，也可就以下方面谈谈你的感受（非必须）：

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？
2. 使用泛型和不使用泛型的编程，对你来说有何差异？
3. 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？
4. 本实验设计的ADT在三个应用场景下使用，这种复用带来什么好处？
5. 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？
6. 关于本实验的工作量、难度、deadline。