

**2021年春季学期  
计算学部《软件构造》课程**

**Lab 2实验报告**

|  |  |
| --- | --- |
| 姓名 | 梁浩 |
| 学号 | 1190200717 |
| 班号 | 1903008 |
| 电子邮件 | 3235962608@qq.com |
| 手机号码 | 15961817952 |

**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 2](#_Toc72249754)

[3.1.2 Problem 1: Test Graph <String> 2](#_Toc72249755)

[3.1.3 Problem 2: Implement Graph <String> 3](#_Toc72249756)

[3.1.3.1 Implement ConcreteEdgesGraph 4](#_Toc72249757)

[3.1.3.2 Implement ConcreteVerticesGraph 6](#_Toc72249758)

[3.1.4 Problem 3: Implement generic Graph<L> 10](#_Toc72249759)

[3.1.4.1 Make the implementations generic 10](#_Toc72249760)

[3.1.4.2 Implement Graph.empty() 11](#_Toc72249761)

[3.1.5 Problem 4: Poetic walks 13](#_Toc72249762)

[3.1.5.1 Test GraphPoet 13](#_Toc72249763)

[3.1.5.2 Implement GraphPoet 13](#_Toc72249764)

[3.1.5.3 Graph poetry slam 17](#_Toc72249765)

[3.1.6 使用Eclemma检查测试的代码覆盖度 17](#_Toc72249766)

[3.1.7 Before you’re done 18](#_Toc72249767)

[3.2 Re-implement the Social Network in Lab1 19](#_Toc72249768)

[3.2.1 FriendshipGraph类 19](#_Toc72249769)

[3.2.2 Person类 20](#_Toc72249770)

[3.2.3 客户端main() 21](#_Toc72249771)

[3.2.4 测试用例 21](#_Toc72249772)

[3.2.5 提交至Git仓库 21](#_Toc72249773)

[4 实验进度记录 22](#_Toc72249774)

[5 实验过程中遇到的困难与解决途径 22](#_Toc72249775)

[6 实验过程中收获的经验、教训、感想 23](#_Toc72249776)

[6.1 实验过程中收获的经验和教训 23](#_Toc72249777)

[6.2 针对以下方面的感受 23](#_Toc72249778)

# 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（abstractionfunction）

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

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。

实验所需的环境在以前的学习中就已经配好。

在这里给出你的GitHub Lab2仓库的URL地址（Lab2-学号）。

<https://github.com/ComputerScienceHIT/HIT-Lab2-1190200717/tree/master>

# 实验过程

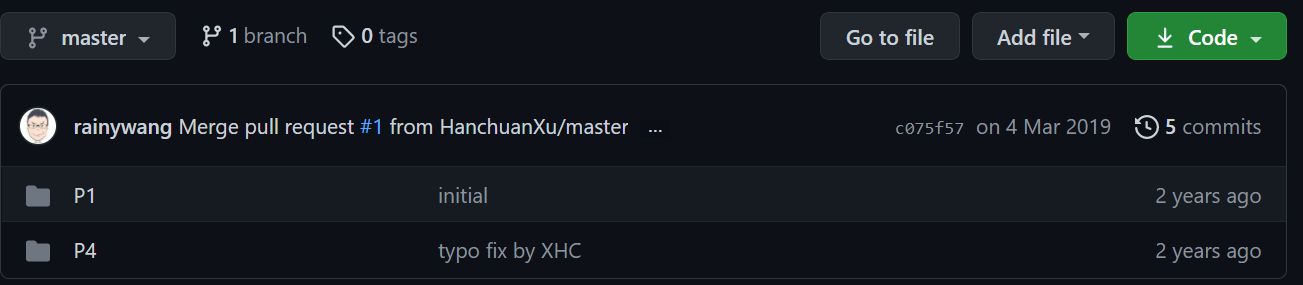
请仔细对照实验手册，针对三个问题中的每一项任务，在下面各节中记录你的实验过程、阐述你的设计思路和问题求解思路，可辅之以示意图或关键源代码加以说明（但千万不要把你的源代码全部粘贴过来！）。

## Poetic Walks

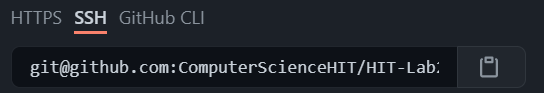
这个实验的主要目的是测试 ADT 的规约设计和 ADT 的多种不同的实现，并练习 TDD 测试优先编程的编程习，在后面练习 ADT 的泛型化。

### Get the code and prepare Git repository

从下图处获取所需代码，创建新的项目后按照要求建立目录。



git init初始化Git仓库，将已有的文件提交到本地仓库中，配置好远程仓库后，将本地仓库的内容push到远程仓库中。





### Problem 1: Test Graph <String>

设计测试Instance方法的testing strategy。主要对每一个需要测试的函数进行输入空间的划分，然后根据输入空间的划分以“最少一次覆盖”的策略进行测试。设计如下：

//boolean add(L vertex)

test add方法:

graph: 1、graph为空

2、graph非空

参数L vertex: 1、新的节点

2、已经存在graph中的点

//int set(L source,L target,int weight)

test set方法:

graph: 1、graph为空

2、graph非空

参数L source: 1、source是新的节点

2、source已经在graph中

参数L target: 1、source是新的节点

2、source已经在graph中

参数int weight: 1、weight = 0

2、weight > 0

//boolean remove(L vertex)

test remove方法:

参数L vertex: 1、vertex为新的节点

2、vertex已经在graph中,但没有节点和它相连

3、vertex已经在graph中，且有节点和它相连

//Set<L> vertices()

test vertices方法:

graph: 1、graph是空图

2、graph非空

//Map<L,Integer> sources(L target)

test sources方法：

graph: 1、graph为空

2、graph非空

参数L target: 1、target是新的节点

2、target是graph中的节点，但是没有边指向它

3、target是graph中的节点，有边指向它

//Map<L,Integer> targets(L source)

test targets方法:

graph: 1、graph为空

2、graph非空

参数L source: 1、source是新的节点

2、source是graph中的点，但是没有边以它为起点

3、source是graph中的点，且有边以它为起点

### 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。

这一部分主要是将 Graph<String>实现两次，分别基于边为主和点为主来实现图的存储和基本操作。

#### Implement ConcreteEdgesGraph

1）edge类的设计

edge类的成员变量如下图所示：

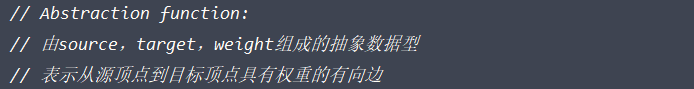


方法如下：

|  |  |
| --- | --- |
| Edge | 含参构造函数，利用传入的参数创建一条新的边 |
| checkRep | 检查不变量，边的权值>0，边的源顶点和目标顶点非空 |
| getSource | 返回边的源顶点 |
| getTarget | 返回边的目标顶点 |
| getWeight | 返回边的权值 |
| toString | 按照指定格式打印边，比如”a->b 权值为:1” |

对于Abstraction function:

由source，target，weight组成的抽象数据型表示从源顶点到目标顶点具有权重的有向边



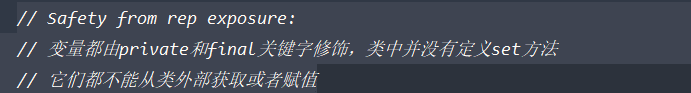
对于Representation invariant:

weight > 0 source和target不为空

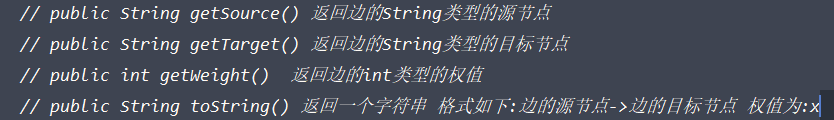


对于Safety from rep exposure:

变量都由private和final关键字修饰，类中并没有定义set方法，成员变量都不能从类外部获取或者赋值，保证了数组不会外泄。



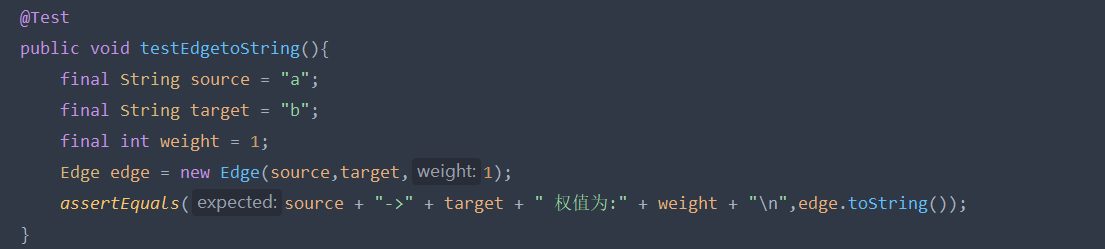
2）Testing strategy for Edge



3）实现Edge类和Edge类的测试

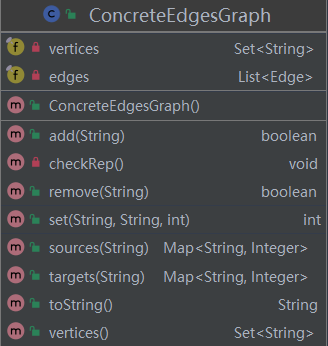
截取部分的代码如下：





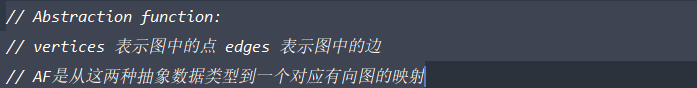
4）ConcreteEdgesGraph类的设计

成员变量和成员函数如图所示：



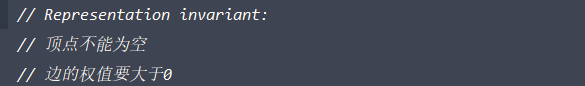
对于Abstraction function:

vertices 表示图中的点 edges 表示图中的边，AF是从这两种抽象数据类型到一个对应有向图的映射



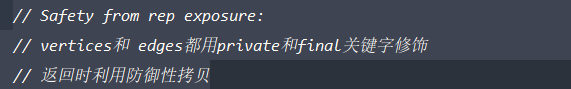
对于Representation invariant:

顶点不能为空，边的权值要大于0



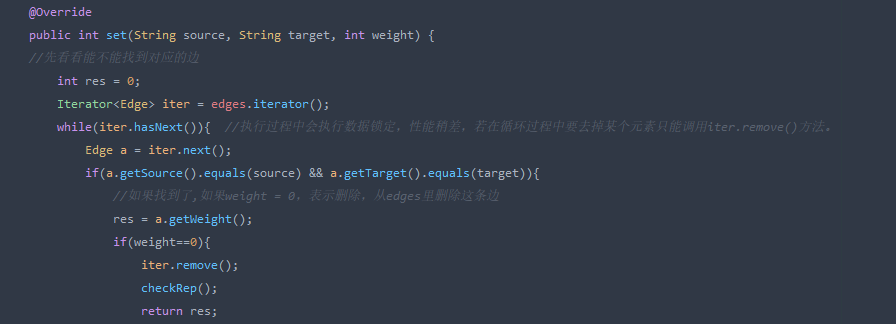
对于Safety from rep exposure:

vertices和 edges都用private和final关键字修饰，返回时利用防御性拷贝



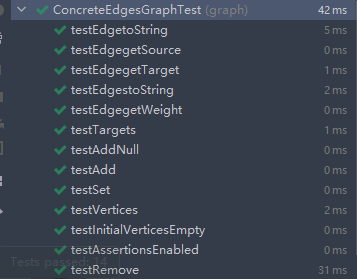
5）实现ConcreteEdgesGraph类和它的测试

截取部分的代码如下：

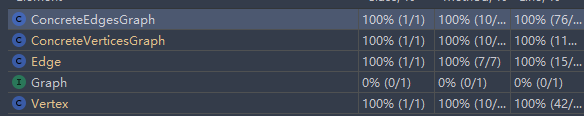


6）运行测试

全部通过



覆盖率测试如下图所示，覆盖率高达100% :



#### Implement ConcreteVerticesGraph

1）Vertex类的设计

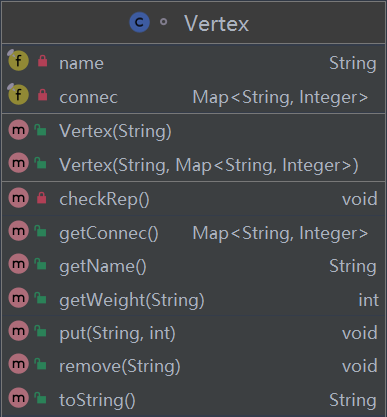
Vertex类的成员变量如图所示：



Vertex类的成员函数包括：

|  |  |
| --- | --- |
| Vertex | 含参构造函数，public Vertex(final String name) |
| Vertex | 含参构造函数，public Vertex(final String name, final Map<String,Integer> connec) |
| checkRep | 检查不变量，顶点的name不为空或者””，边的权值>0 |
| getName | 获取顶点名称 |
| getConnec | 获取顶点的边集 |
| remove | 删除这个源点的某条边 |
| put | 加边或者修改权值 |
| getWeight | 获取指定边的权值 |
| toString | 按指定格式打印 |

类的UML图如下图所示：



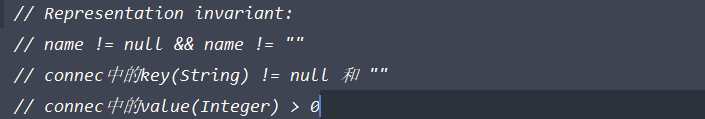
对于Abstraction function:

代表有权值的有向边中的源点



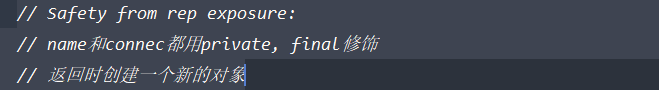
对于Representation invariant:

顶点名不为空或者””，边的权值大于0

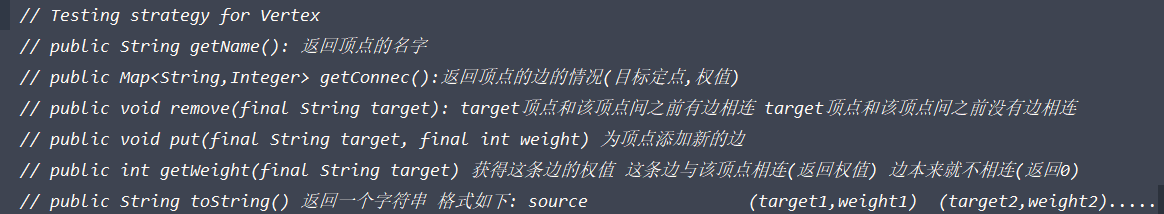


对于Safety from rep exposure:

name和connec都用private, final关键字修饰，函数返回时创建一个新的对象



2）Testing strategy for Vertex



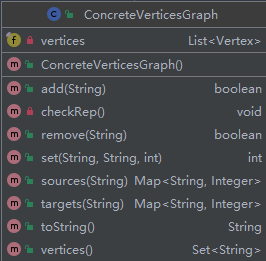
3）实现Vertex类和它的测试

部分代码如图所示：



4）ConcreteVerticesGraph类的设计

UML图如下：



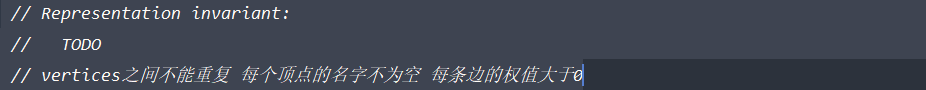
对于Abstraction function:

vertices到有向图的映射



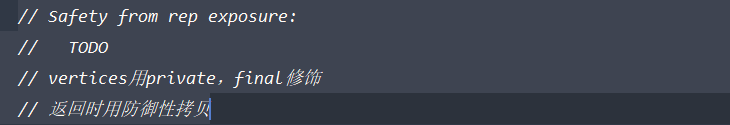
对于Representation invariant:

vertices之间不能重复 每个顶点的名字不为空 每条边的权值大于0



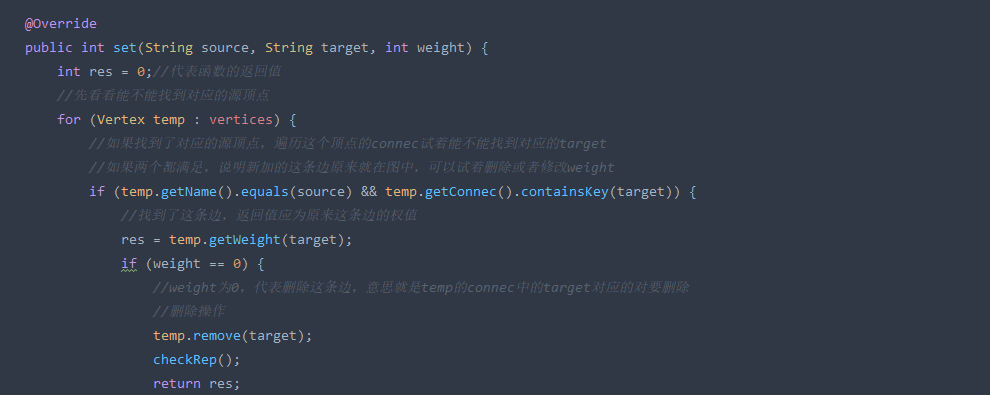
对于Safety from rep exposure:

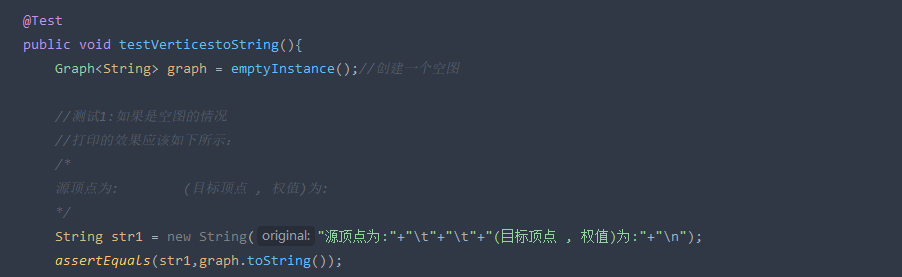
vertices用private，final关键字修饰，返回时用防御性拷贝



4）实现ConcreteVerticesGraph类和它的测试

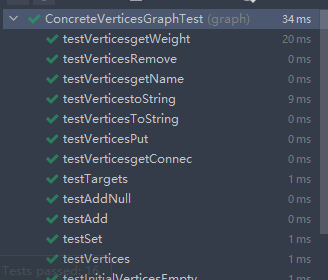
部分代码截图如下：



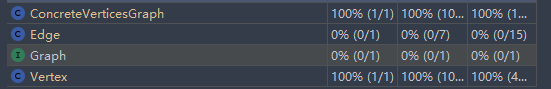


5）运行测试

测试全部通过



代码覆盖率高达100%

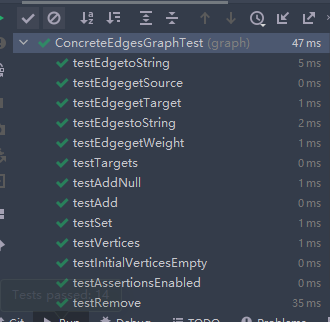


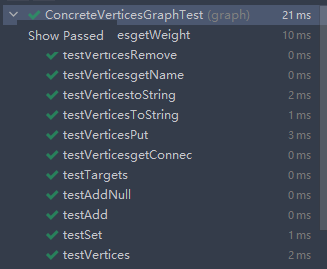
### Problem 3: Implement generic Graph<L>

将已有的两个 Graph<String>的实现改为Graph<L>的泛型实现。

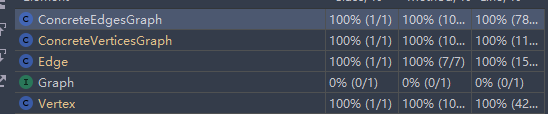
#### Make the implementations generic

将ConcreteEdgesGraph和ConcreteVerticesGraph都用泛型实现后，重新跑测试



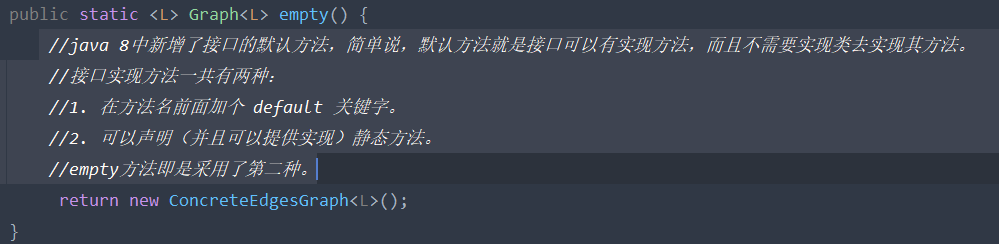


覆盖率依旧是100%



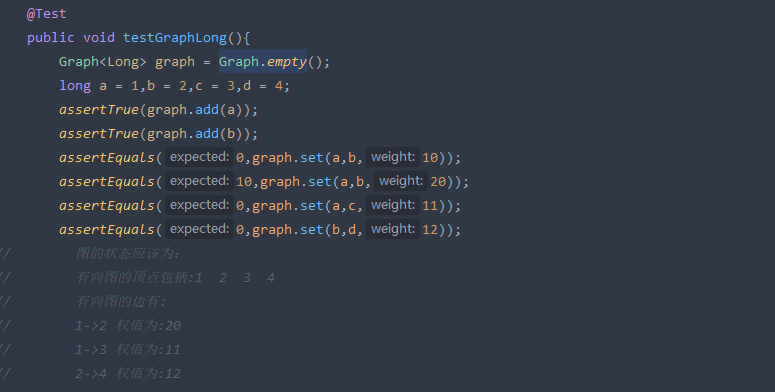
#### Implement Graph.empty()

1、选择ConcreteEdgesGraph 作为返回对象实现Graph.empty()函数:

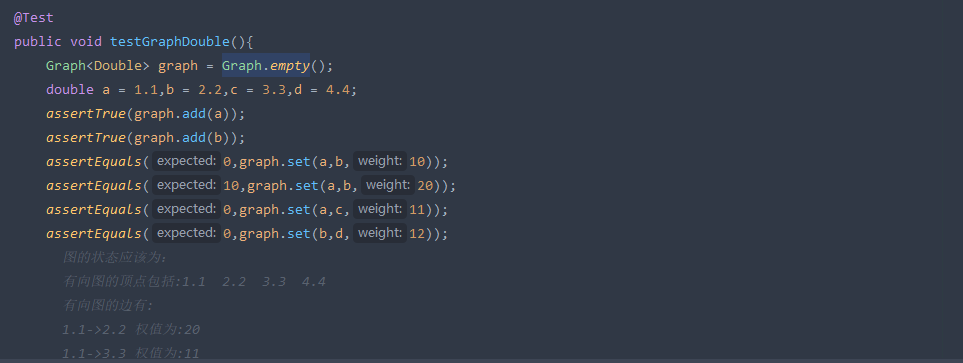


2、测试GraphStaticTest

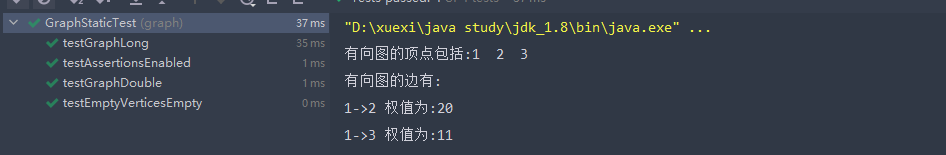
尝试label为long的情况：



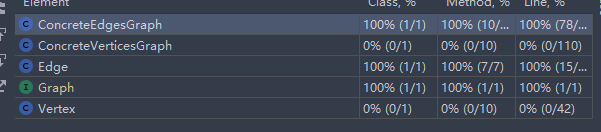
尝试label为double的情况：



写完后运行测试，运行成功



查看一下覆盖率，依旧是100%：



### Problem 4: Poetic walks

这个问题一部分是用给的语料生成图，相邻的单词间用一条有向边连接，另一部分是给定一个输入字符串，通过在图中判断它们之间是否有bridge来对字符串进行扩充。

#### Test GraphPoet

Testing strategy:

对于构造函数:

1、传入不存在的文件

2、传入的文件为空

3、传入的语料文件只有一行单词

4、传入具有多行的语料文件

对于poem函数:

1、传入的字符串为null

2、传入的字符串!=null，但只有一个单词

3、传入的字符串!=null，不止一个单词(还可以再设计根据权值选bridge的情形）

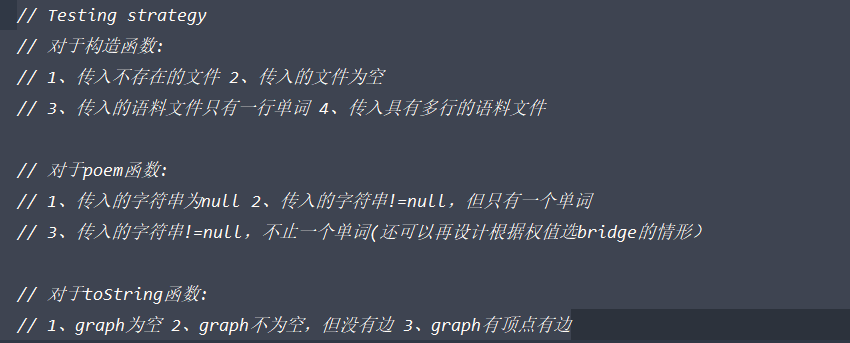
对于toString函数:

1、graph为空

2、graph不为空，但没有边

3、graph有顶点有边

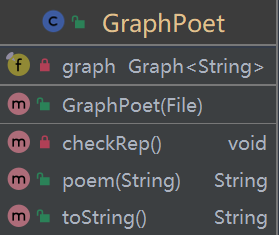
测试策略部分截图如下:



#### Implement GraphPoet

1）GraphPoet的结构分析

UML图如下:

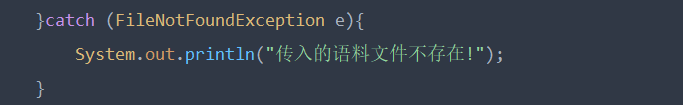


对于成员变量graph:



对于含参构造函数:

1、如果传入的文件不存在，catch异常，打印



2、如果文件存在，按行读入，根据要求生成对应的图



对于poem方法:

关键在于寻找bridge，当有多个符合要求的节点时，要选择权值大的。



对于toString方法:

沿用ConcreteEdgesGraph.java中的打印方法

2）关于AF,RI和rep exposure

对于Abstraction function:

将文本文件转化成有向图，图的顶点就是文件中的单词，相邻单词在图中对应的顶点间有有向边



对于Representation invariant:

图中的顶点不为空或者""，边的权值要大于0



对于Safety from rep exposure:

graph用private和final关键字修饰，返回参数时创建新的变量



3）类的实现

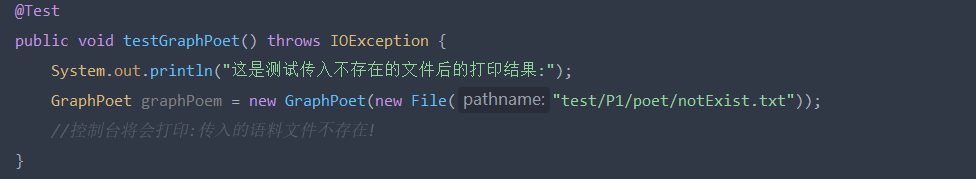
部分代码截图如下：



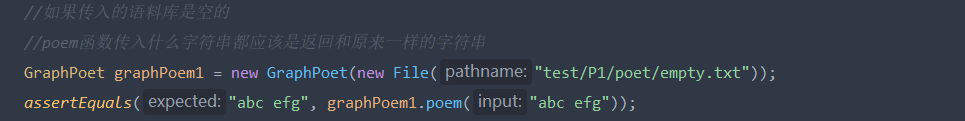
4）测试

对poem参数不同情况下的测试:

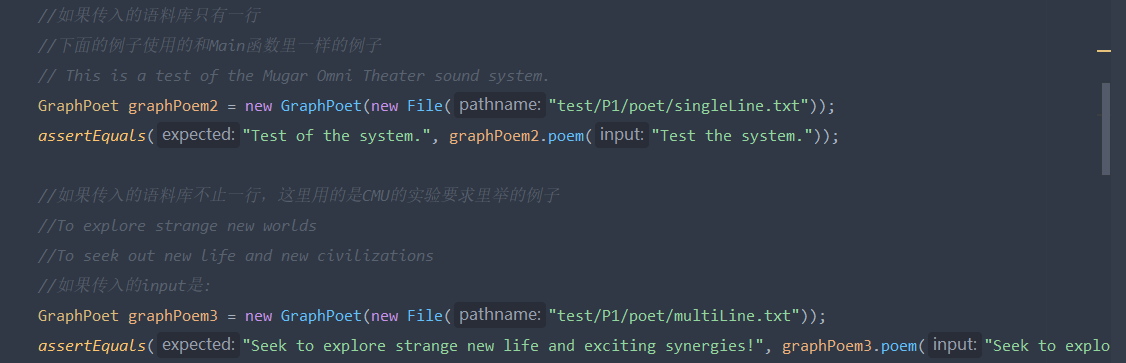
当传入的文件不存在时，控制台打印“传入的语料文件不存在!”



当传入的语料库文件为空时，poem后的字符串和以前一样



下面是语料库有一行或多行字符串时的情况

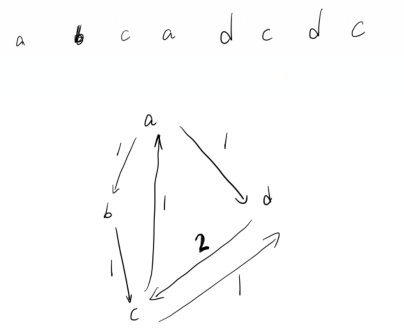


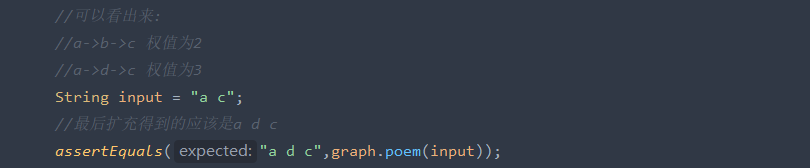
对ToString在空图、图非空（只有顶点）、图非空（既有顶点，也有边）的情况下的测试:



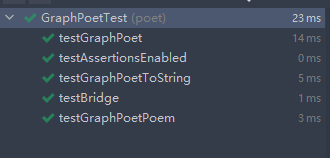
对bridge按照权值大小选取的测试:

如下图所示，构造一个特殊的图，涉及到bridge的选择





所有的测试结果:



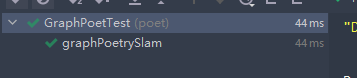
覆盖率测试如下:



#### Graph poetry slam

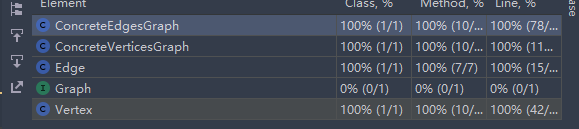
选取了WILLIAM BUTLER YEATS的*When You Are Old*

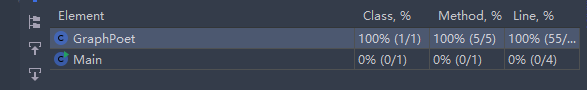
输入字符串为”you old”，扩充后的结果为”you are old”



### 使用Eclemma检查测试的代码覆盖度

代码覆盖度都是100%

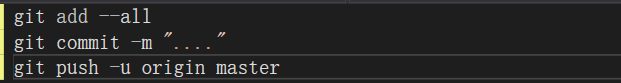




### Before you’re done

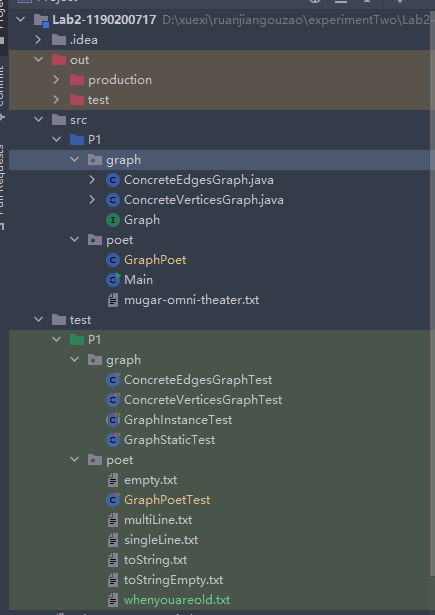
请按照[http://web.mit.edu/6.031/www/sp17/psets/ps2/#before\_youre\_done](http://web.mit.edu/6.031/www/sp17/psets/ps2/" \l "before_youre_done)的说明，检查你的程序。

如何通过Git提交当前版本到GitHub上你的Lab2仓库。



直接通过idea来提交会更加方便

项目的目录结构树状示意图如下：

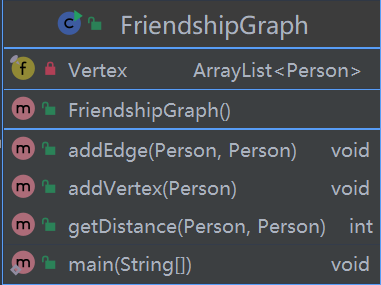


## Re-implement the Social Network in Lab1

这个任务主要是重新实现 Lab1 中的 Social Network，利用在P1中已经写好的 Graph<L>接口来实现，尽量重用已有的函数。

### FriendshipGraph类

这里给出FriendshipGraph类的UML图



包含成员变量Vertex:



用来保存图中的各顶点。

对于addVertex方法:

充分使用 Graph<L>里面提供的函数 add，即可实现增加顶点的功能。如果顶点名重复，打印错误信息，程序直接退出。

对于addEdge方法:

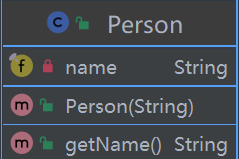
充分利用 Graph<L>里面提供的函数 set，即可实现增加顶点的功能，不过要先判断两个顶点是否相同。

对于getDistance方法:

利用BFS算法来求两个顶点之间的最短距离。

### Person类

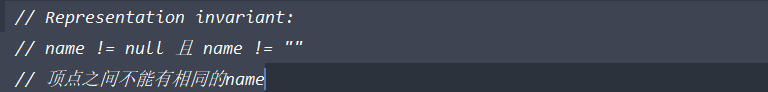
下面给出Person类的UML图:



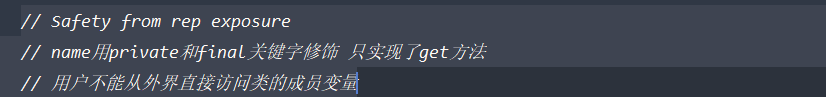
对于Abstraction function:



对于Representation invariant:



对于Safety from rep exposure:



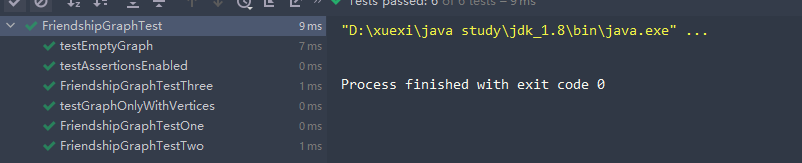
### 客户端main()

利用 Lab1 中已有的 main 客户端即可。

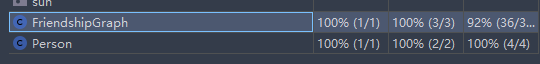
### 测试用例

测试策略主要是根据 FriendshipGraph 中的图的类型进行测试，主要分为空图的测试、仅有顶点的图的测试、复杂图的测试。

测试结果如下图：

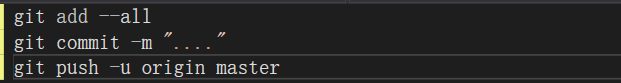


覆盖率测试如下图所示:



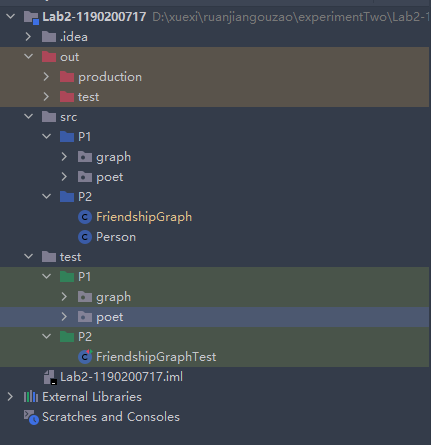
### 提交至Git仓库

如何通过Git提交当前版本到GitHub上你的Lab3仓库。



用idea的git工具来提交会更快。

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。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 2021/6/7 | 19:30-22:30 | Implement ConcreteEdgesGraph | 按时完成 |
| 2021/6/8 | 19:00-0:00 | 编写ConcreteVerticesGraph的方法 | 按时完成 |
| 2021/6/9 | 8:30-9:30 | 初步完成ConcreteVerticesGraph.java | 按时完成 |
| 2021/6/10 | 14:30-19:30 | ConcreteEdgesTest的编写 | 按时完成 |
| 2021/6/11 | 19:30-22:30 | poetic walk | 按时完成 |
| 2021/6/12 | 19:30-23:30 | 完成P2 | 按时完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 英文题目看不懂 | Google翻译，逐字逐句揣摩 |
| Java的泛型和接口概念不懂 | 菜鸟教程自学 |

# 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

代码的性能可以再优化，对于成员变量的封闭性平时考虑的比较少。

## 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？

在面向 ADT 编程的时候更多的需要考虑代码的可复用性，需要使整个的编程更加具有普适性；而面向应用场景编程，更加直接和方便，但是适用的范围明显更小。

1. 使用泛型和不使用泛型的编程，对你来说有何差异？

使用泛型编程的话，写起来不太习惯，而不使用泛型编程的话，写起来更加顺手。

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？

可以尽早发现错误，可以逐渐适应。

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？

可以将已有的代码进行复用来尽量节省时间。

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？

没做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？

提前发现错误，避免数据外泄，愿意。

1. 关于本实验的工作量、难度、deadline。

工作量比较大，时间紧凑。

1. 《软件构造》课程进展到目前，你对该课程有何体会和建议？

和计算机系统一起同时做实验，有点累人。