

**2021年春季学期  
计算学部《软件构造》课程**

**Lab 2实验报告**

|  |  |
| --- | --- |
| 姓名 | 冯开来 |
| 学号 | 1190201215 |
| 班号 | 1903007 |
| 电子邮件 | [3223200086@qq.com](mailto:3223200086@qq.com) |
| 手机号码 | 13771513877 |

**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

根据实验手册简要撰写。

# 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。

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

* 实验给了一个Graph的接口，我们的任务就是分别建立一个边的类和一个点的类分别继承这个Graph的接口，并在里面实现一系列方法，并且能够实现抽象数据型。最后完成这个Poetic Walks诗意的漫步。
* 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。

### Get the code and prepare Git repository

得到实验classroom的网址<https://classroom.github.com/a/NL2TjK2z>

访问该URL，建立自己的Lab2仓库关联至自己的学号

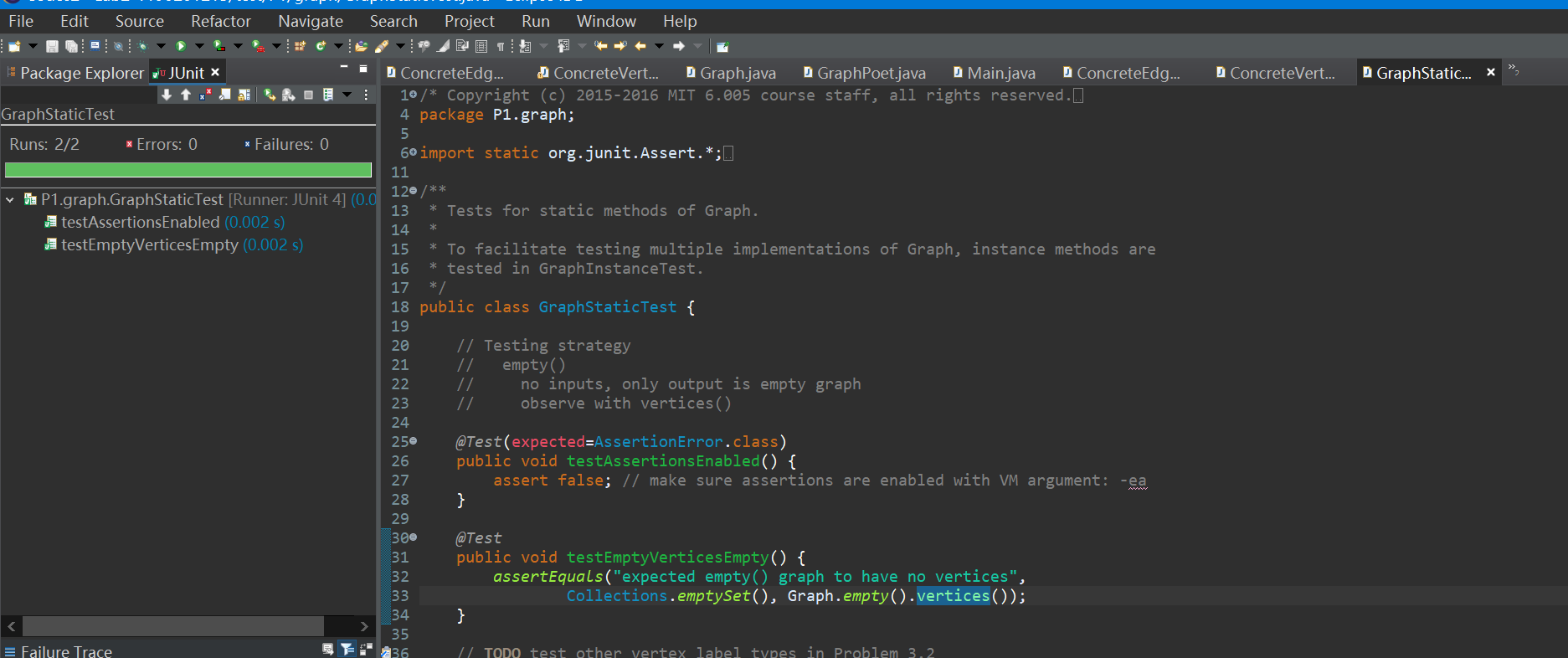
Gitbash上运行，下载工程文件

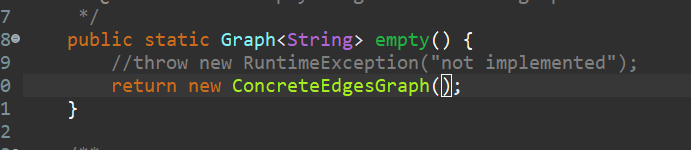
gitclone <https://github.com/rainywang/Spring2020_HITCS_SC_Lab2.git>

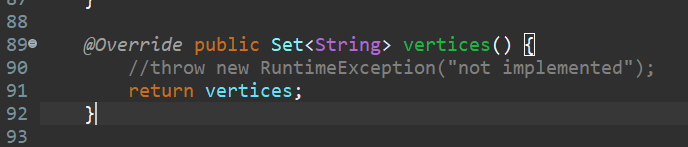
### Problem 1: Test Graph <String>

针对Graph<String>设计的测试策略，可以针对里面每一个方法进行测试。

不过这里我不是很懂，没有理解这第一题想要我们测试什么，什么都还没写。不过应该是添加一个ConcreteEdgesGraph里面vertices这个方法，然后进行junit的测试也是通过了。







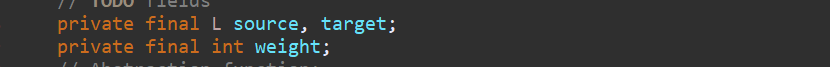
### Problem 2: Implement Graph <String>

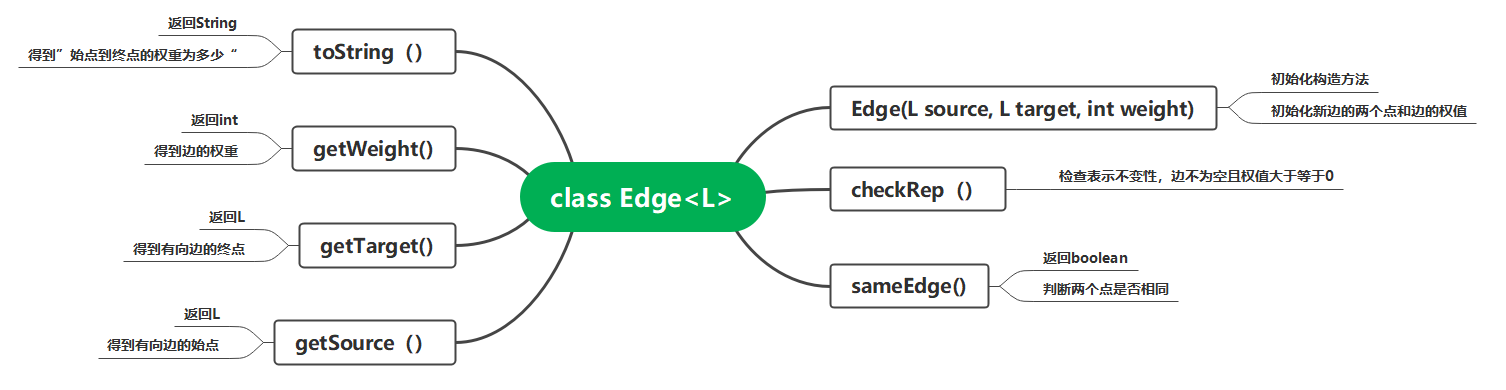
以下各部分，请按照MIT页面上相应部分的要求，逐项列出你的设计和实现思路/过程/结果。

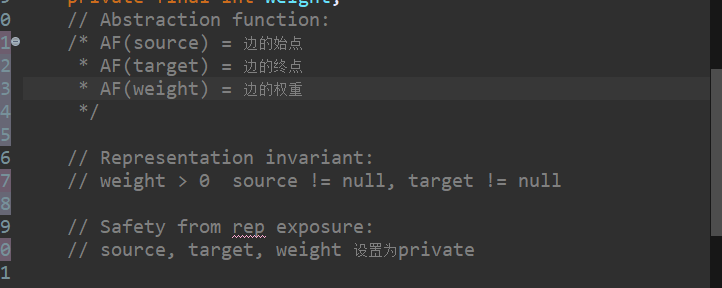
#### Implement ConcreteEdgesGraph

1. 实现EDGE类。
   1. **EDGE类字段定义**

包括边两点的命名，源点和终点。还有边的权重，int类型。Immutable类。



* 1. **EDGE类中需要实现的方法**
  2. **AF、RI、Safety from rep exposure**

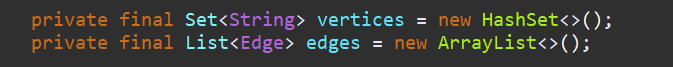


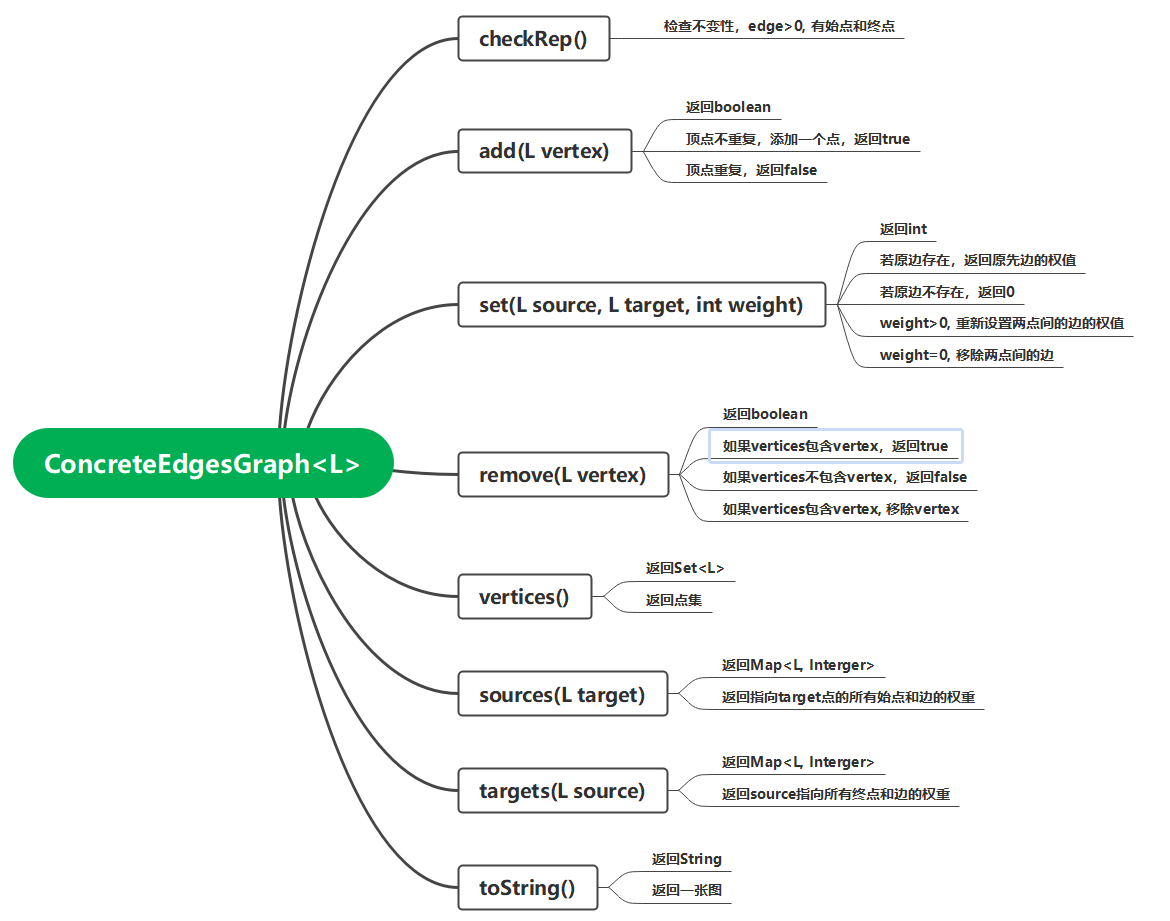
* 1. **Testing strategy**

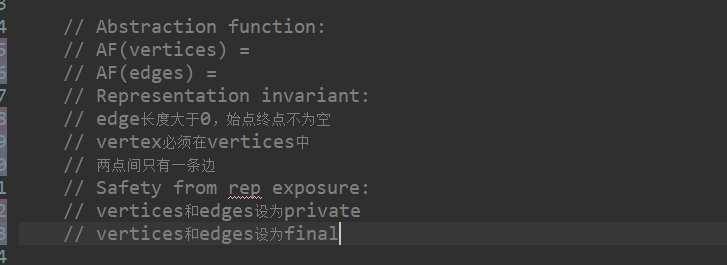


1. 实现ConcreteEdgesGraph类
   1. **ConcreteEdgesGraph类字段定义**

包括顶点集Set表和边集List表，定义私有类型。Immutable类

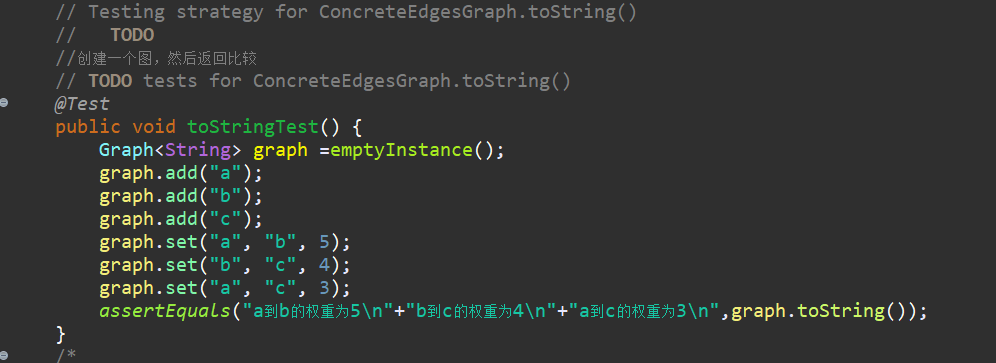


* 1. **ConcreteEdgesGraph类方法实现**
  2. **AF、RI、Safety from rep exposure**

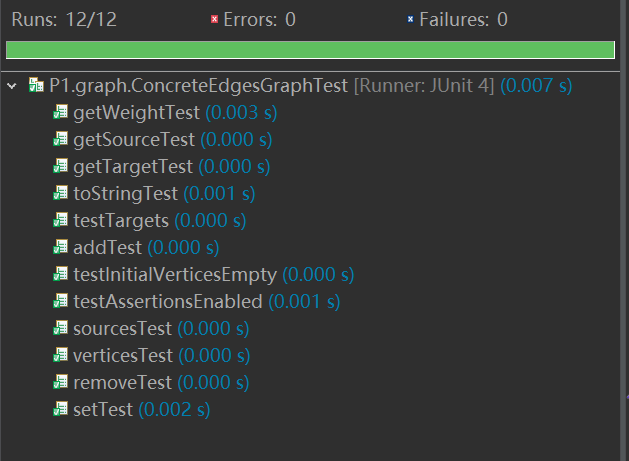


* 1. **Testing strategy**

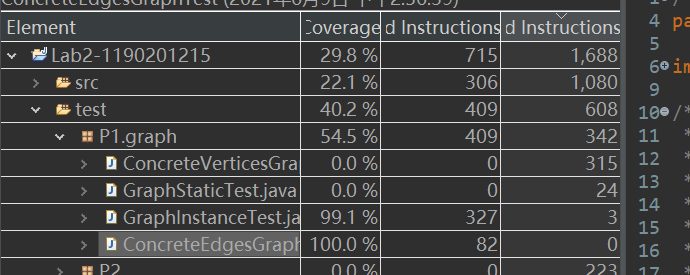
继承Graph的测试策略，增加对toString的测试即可。



测试结果：



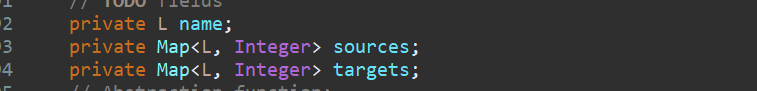
覆盖率：

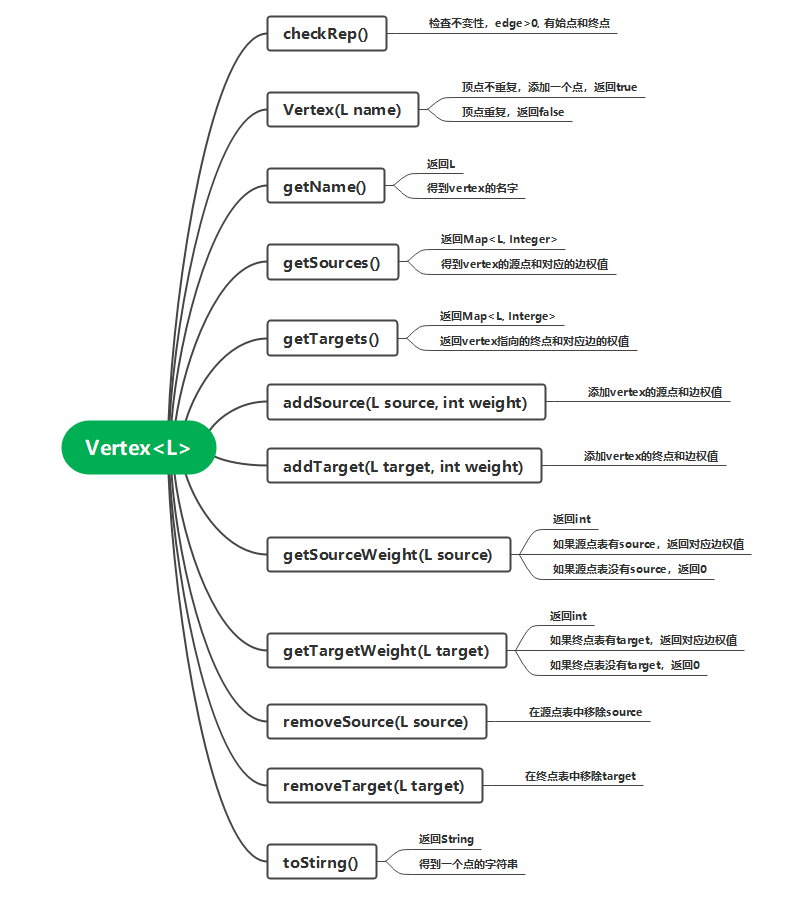


#### Implement ConcreteVerticesGraph

1. 实现Vertex类
   1. **Vertex类字段定义**

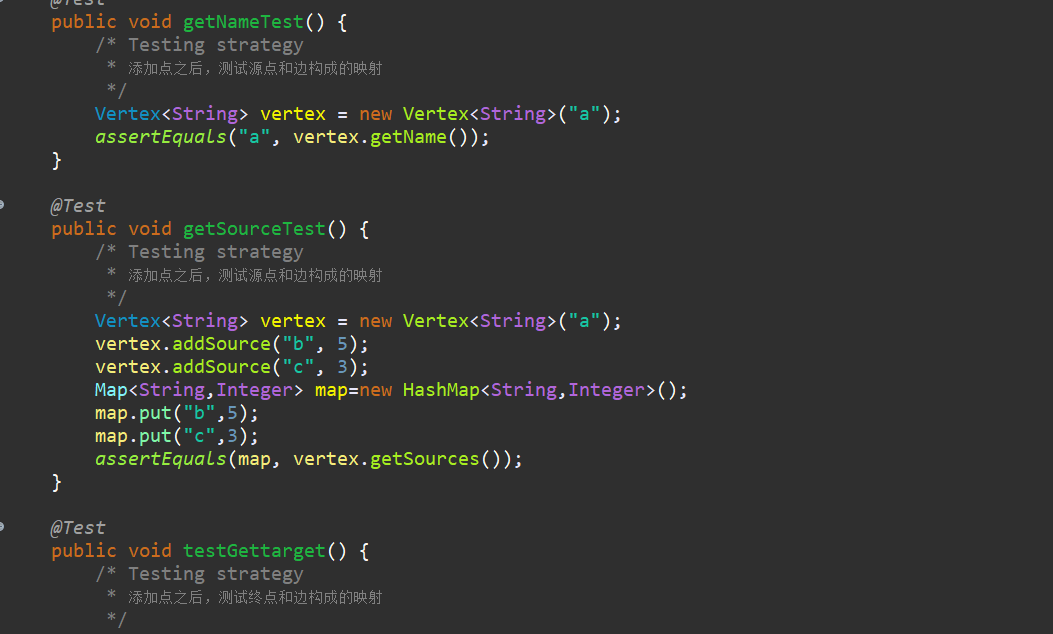
应该包括点的命名，指向该点的源点表Map，该点指向的终点表Map，定义私有类型。Immutab类

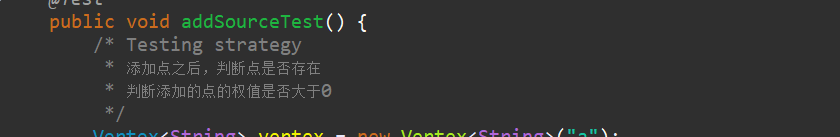


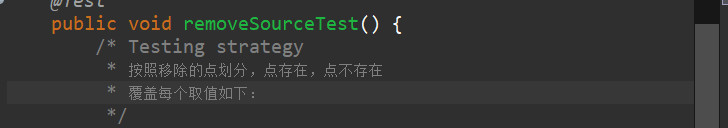
* 1. **Vertex类方法实现**
  2. **AF、RI、Safety from rep exposure**



* 1. **Testing strategy**



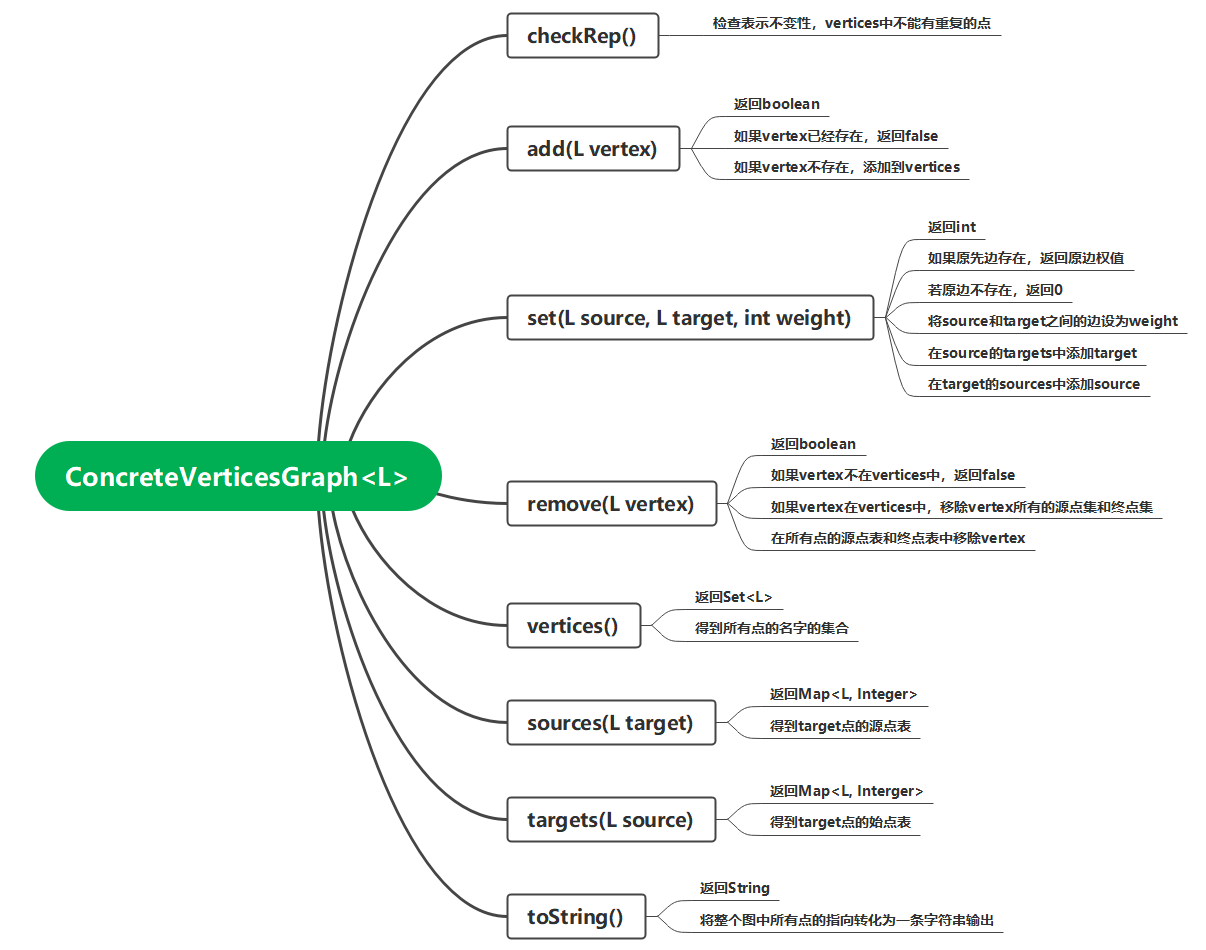
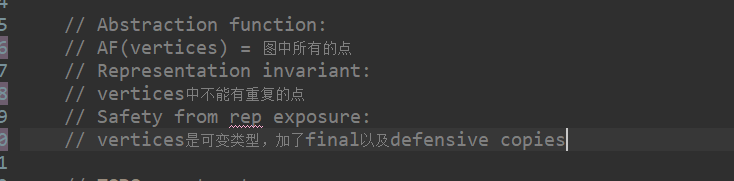


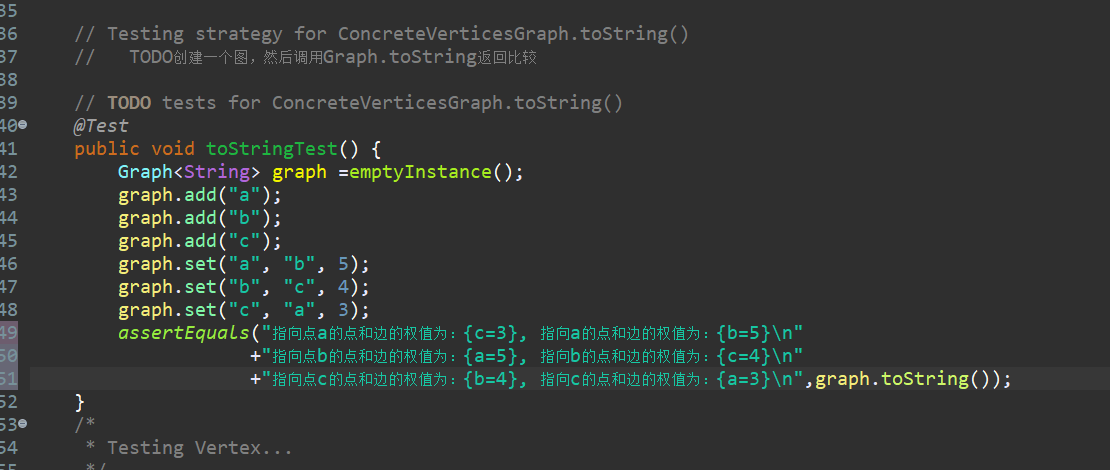


1. 实现ConcreteVerticesGraph类
   1. **ConcreteVerticesGraph类字段定义**

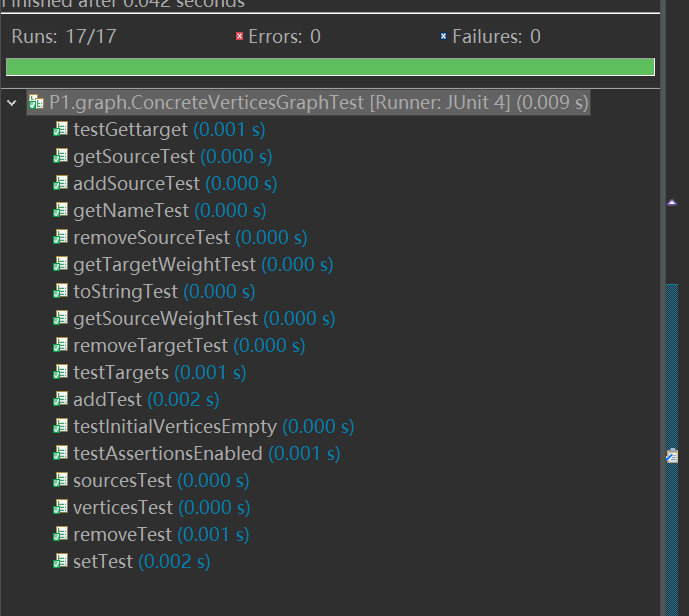
定义有Vertex构成的List，定义私有类型。Immuta类。

****

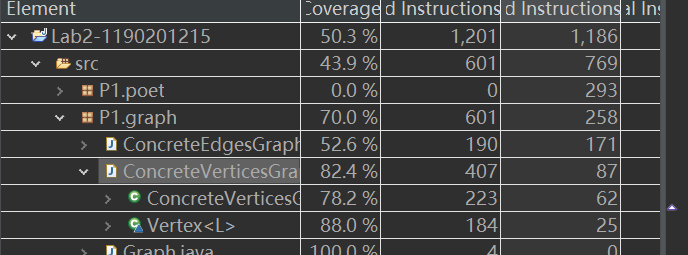
* 1. **ConcreteVerticesGraph类方法实现**
  2. **AF、RI、Safety from rep exposure**
  3. Testing strategy

继承Graph的测试策略，添加对toString的测试：

测试结果：



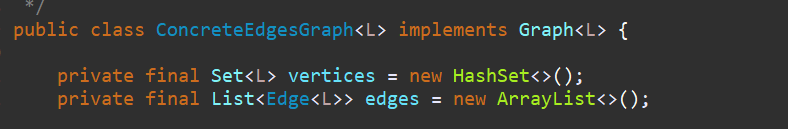
覆盖率：

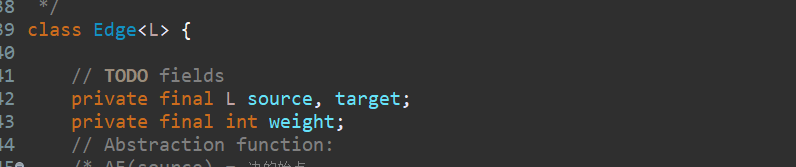


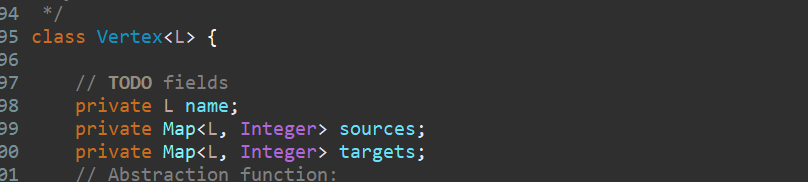
### Problem 3: Implement generic Graph<L>

#### Make the implementations generic

使用泛型。根据我的理解，其实可以很简单的把String改成L，然后剩下地方哪里出错改哪里，一般就是加个L。

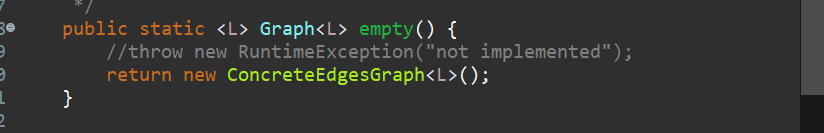




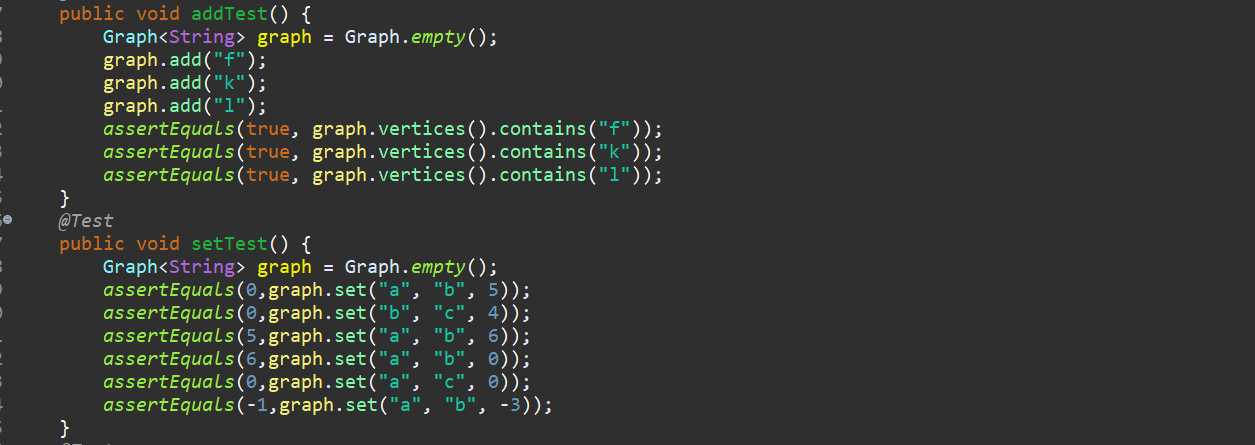


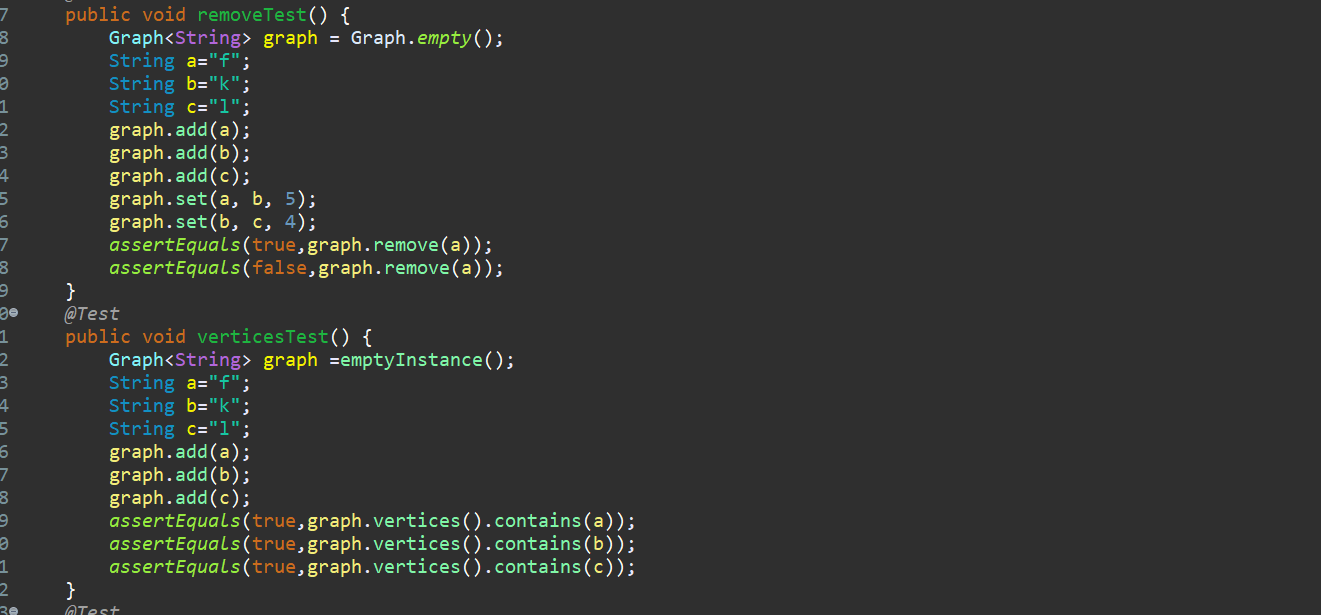
#### Implement Graph.empty()

**调用一个具体的实现即可：**



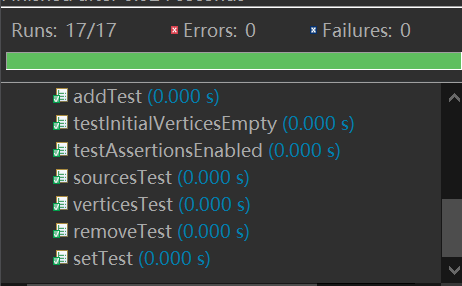
**最后进行具体化之后所有方法的测试，包括add（）、set（）、remove（）等：**



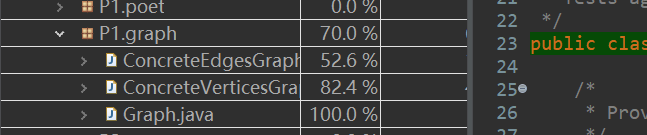




**测试结果：**



**覆盖率：**

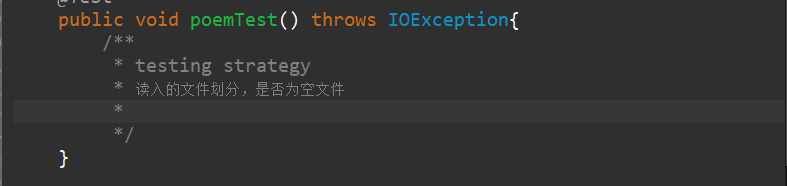


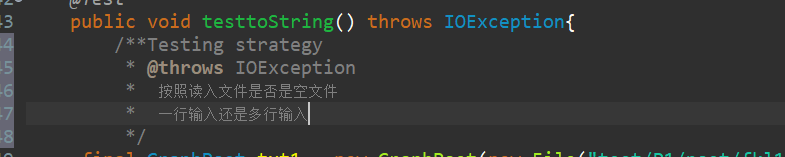
### Problem 4: Poetic walks

要求我们实现一个类，需要利用之前实现的图结构，能够将语料库转化为该种图结构，并且在图中搜索，完成对输入的诗句的句子进行扩充。

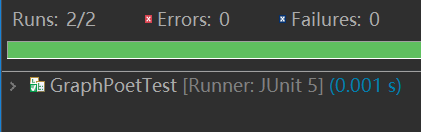
#### Test GraphPoet

**Testing strategy：**

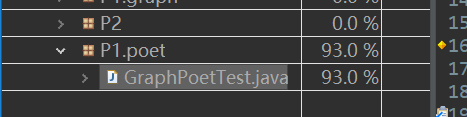




测试结果：



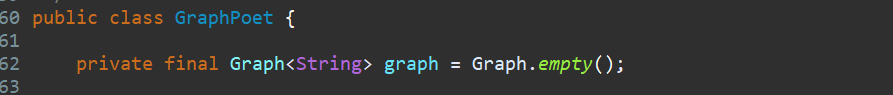
覆盖率：



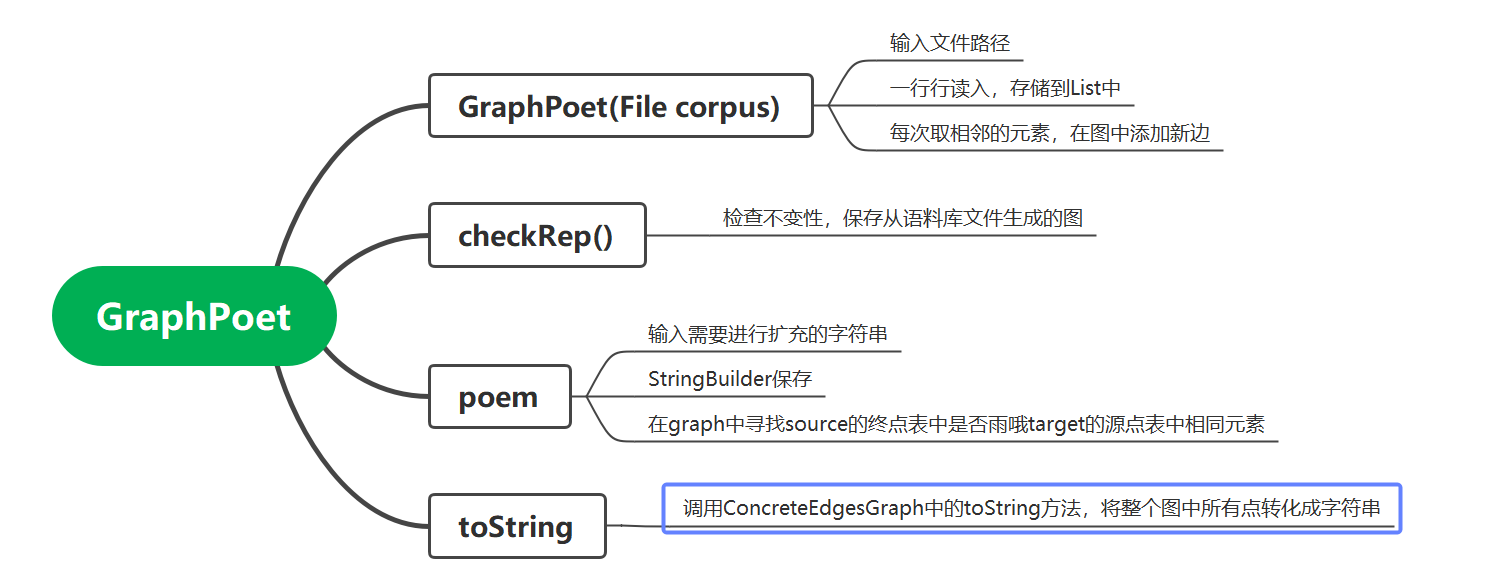
#### Implement GraphPoet

1. GraphPoet类字段定义

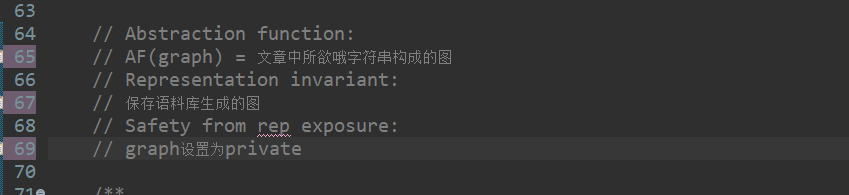
String构成的Graph，定义私有类型。Immuta类



1. GraphPoet类中实现方法

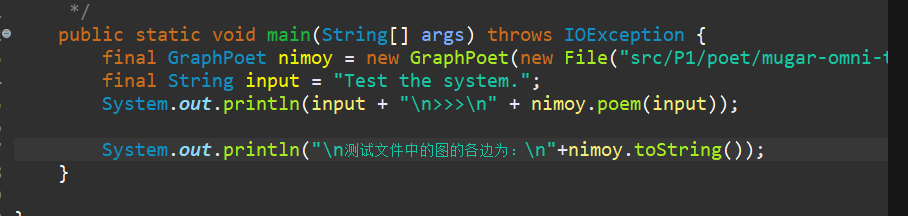


1. AF、RI、Safety from rep exposure

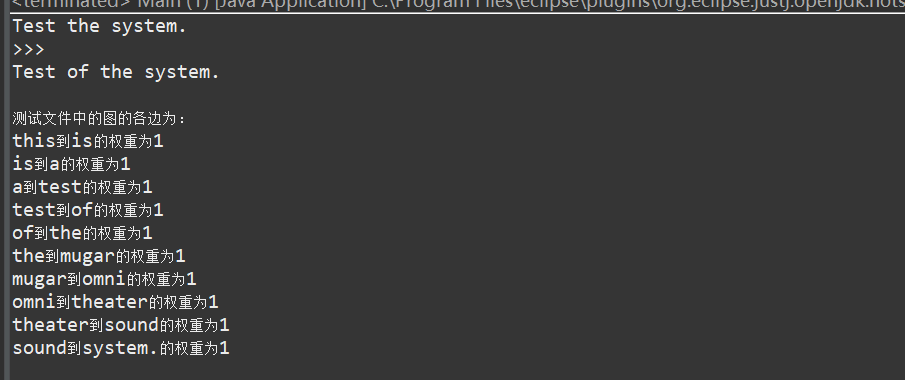


#### Graph poetry slam

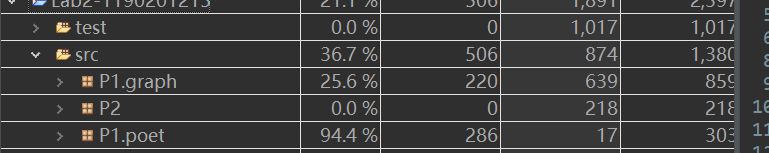
在继承原来的代码上添加一个toString输出



结果：

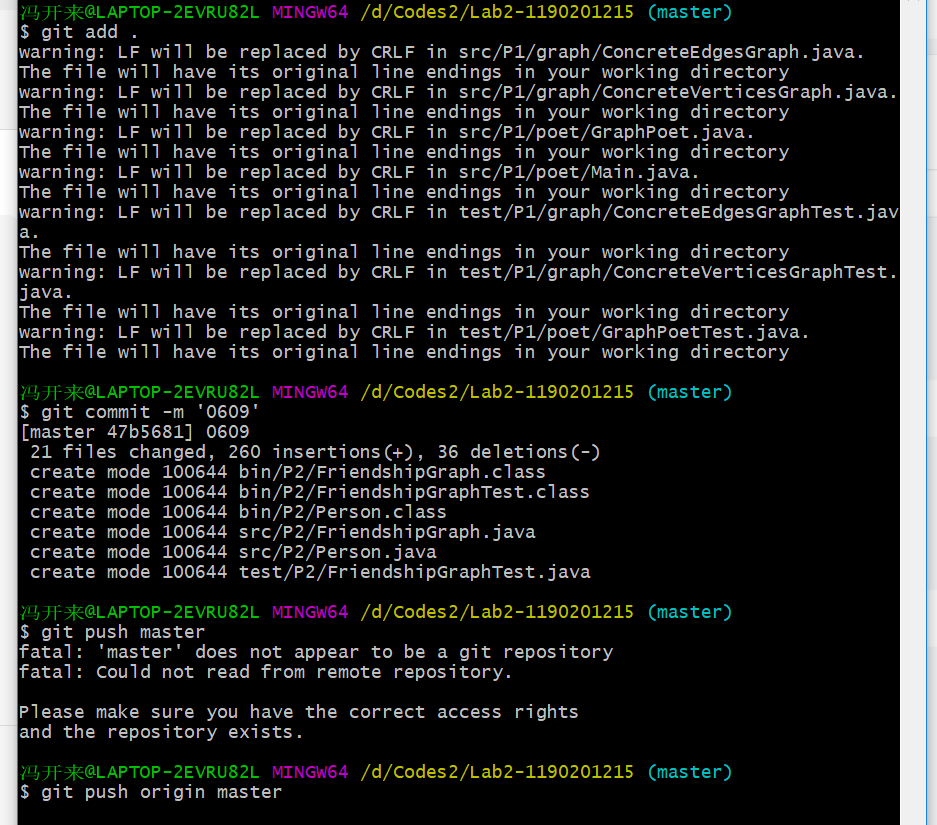


### 使用Eclemma检查测试的代码覆盖度

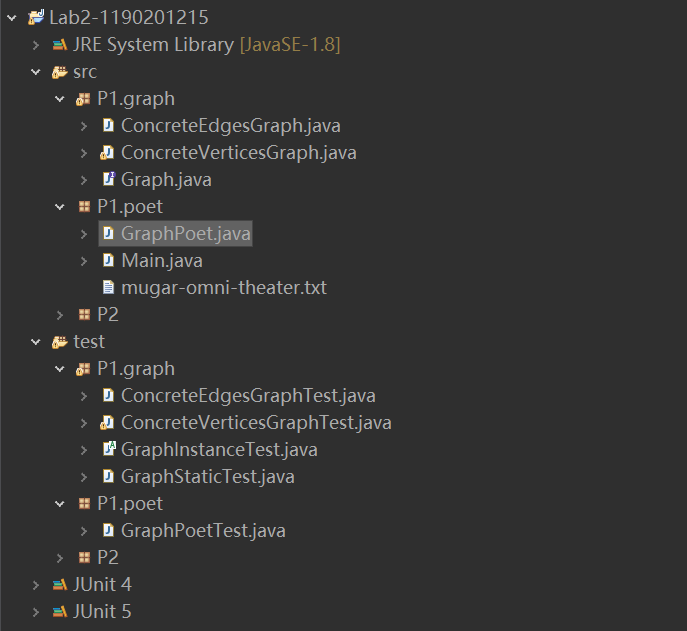


### Before you’re done

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。



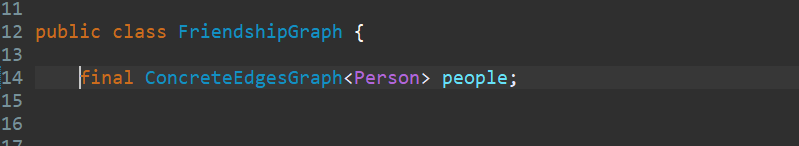
## Re-implement the Social Network in Lab1

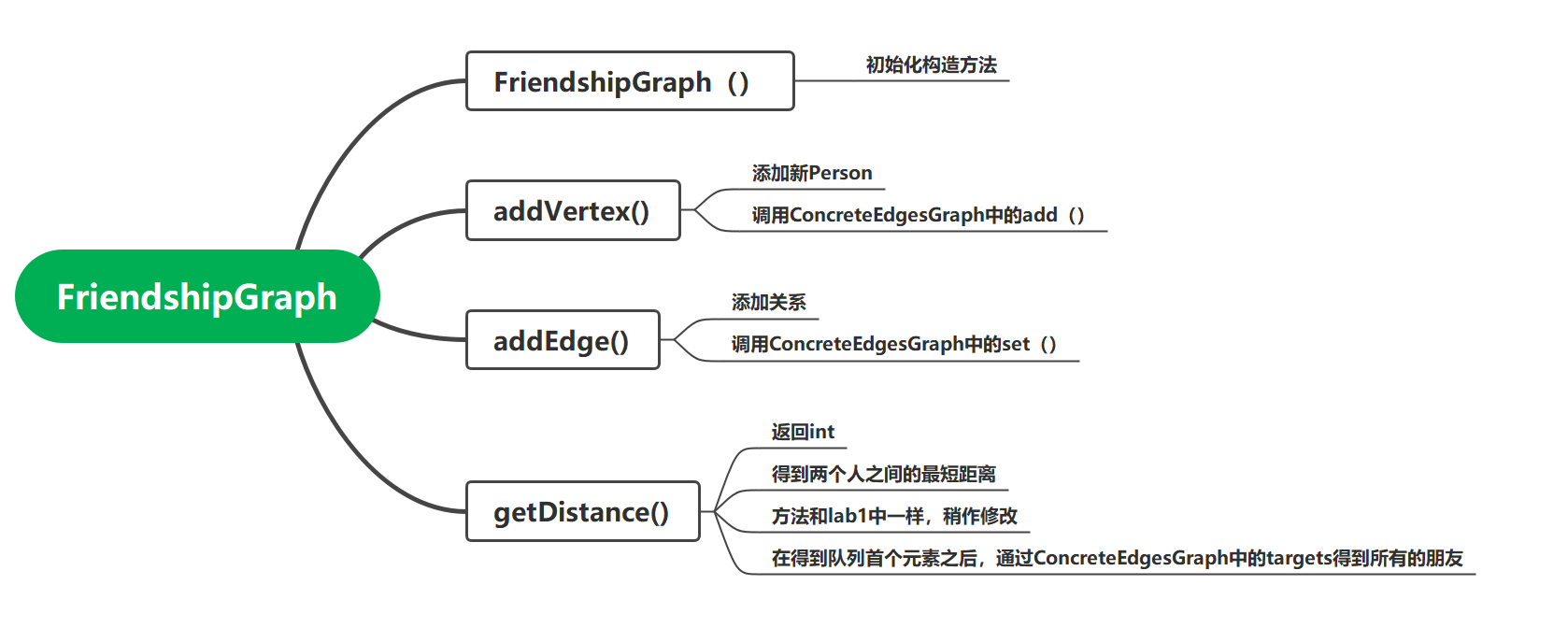
要求我们基于Poetic Walks中定义的Graph<L>及其两种实现（本人使用的是ConcreteVerticesGraph<L>），实现Lab1中Social NetWorek中的各种功能，并且尽可能复用ConcreteVerticesGraph<L>中已经实现的方法，然后运行提供的main()和执行Lab1中的Junit测试用例，使之正常运行。

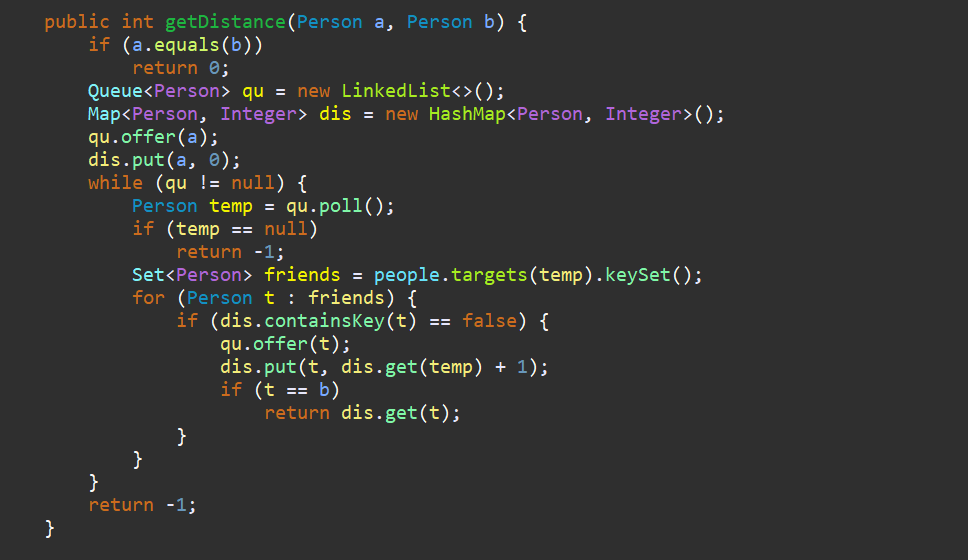
### FriendshipGraph类

1. **FriendshipGraph字段定义**

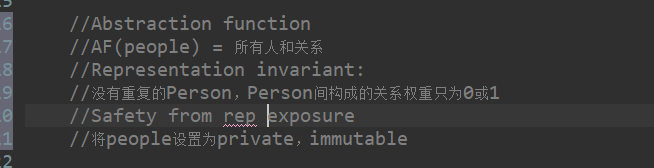
FriendshipGraph的字段为Person构成的ConcreteEdgesGraph，定义私有类型。Immutable类。



1. **FriendshipGraph方法实现**

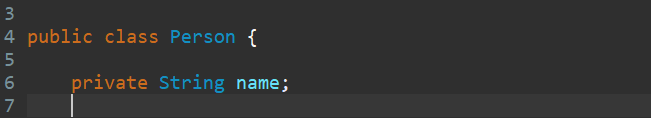


1. **AF、RI、Safety from rep exposure**



### Person类

1. **Person字段定义**

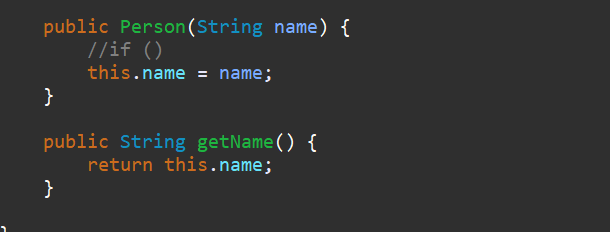


1. **Person方法实现**

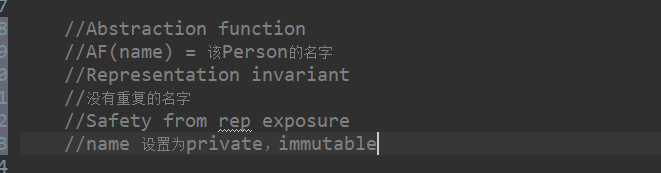
两个方法

Person(String name) 初始化构造方法

String getName() 返回Person的姓名



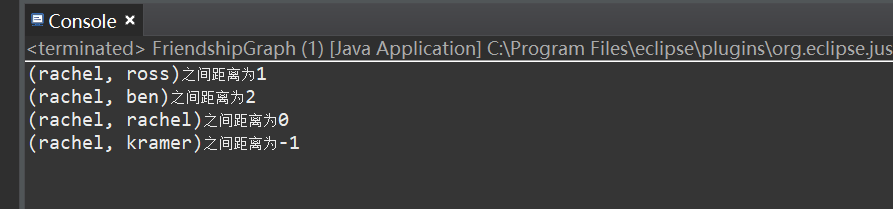
1. **AF、RI、Safety from rep exposure**



### 客户端main()

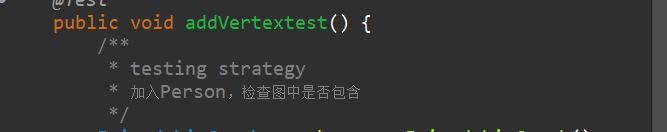
和lab1相同即可：

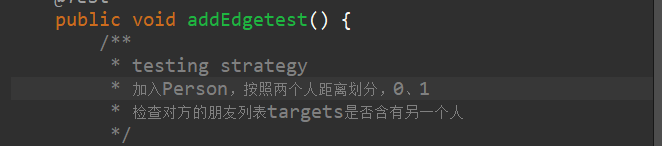
**测试结果：**

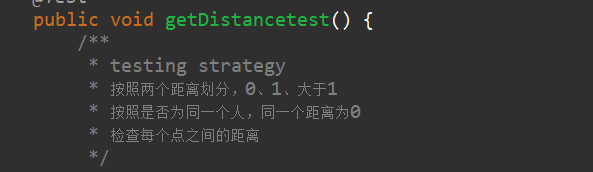
****

### 测试用例

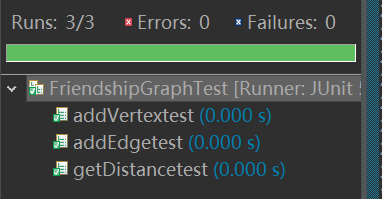
和lab1相同即可：



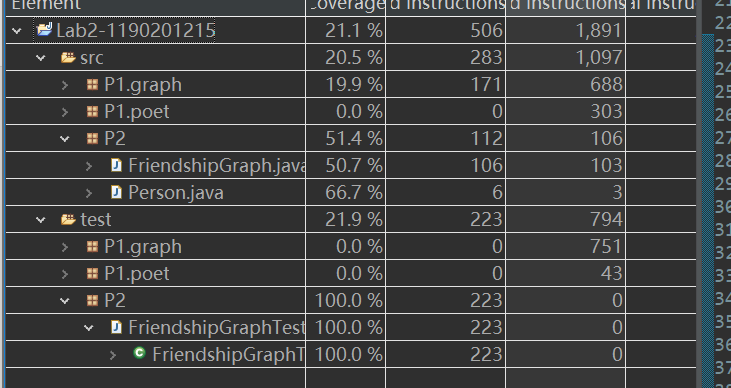




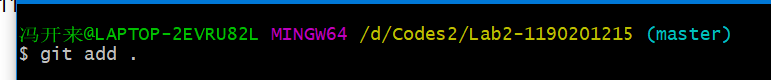
**测试结果：**

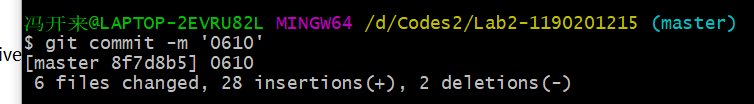


**覆盖率：**

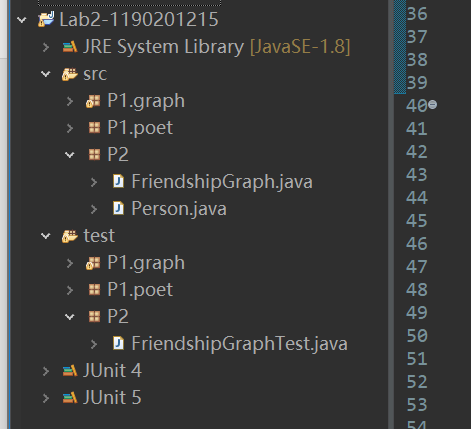


### 提交至Git仓库









# 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 |
| 5.2 | 13：00-15：00 | Poetic walks ConcreteEdgesGraph Edges类 | 完成 |
| 5.3 | 13：00-15：00 | Poetic walks ConcreteEdgesGraph类 | 完成 |
| 5.4 | 14：00-17：00 | Poetic walks ConcreteVerticessGraph Vertex类 | 完成 |
| 5.5 | 14：00-19：00 | Poetic walks ConcreteVerticesGraph类 | 完成 |
| 5.6 | 20：00-23：00 | Poetic walks poet类 | 完成 |
| 5.8 | 13：00-22：00 | 上面的test和friendshipGraph部分 | 完成 |
| 5.9 | 1：30-18：00 | 实验报告poetic walks部分 | 完成 |
| 5.10 | 3：45-4：30 | 实验报告friendshipGraph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 |
| 时间不够，任务量太多了！ | 熬夜 |
| 不理解泛型是什么 | Csdn等查阅资料 |
| 和计算机系统实验混合双打，任务量太大了 | 和同学多讨论 |

# 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

## 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？

面向对象能够每次完成对某个对象的编程，与面向过程的编程的思路完全不一样。

1. 使用泛型和不使用泛型的编程，对你来说有何差异？

泛型能够适应更多的变化，更加灵活

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？

能够保证代码的正确性，规格更加严格，及时修改。不适应。

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？

提高代码覆盖率，节省工作量

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？

慢慢分析对每个类进行了实现，但是还是很多方法用的很繁琐，并且很多类很重复，一些功能没有用到。

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？

防止内部变量被外部恶意修改。不想坚持。

1. 关于本实验的工作量、难度、deadline。

说实话只有一个这个实验还好，但是中间有两个计算机系统实验，不知道计算学部教学组怎么安排的，三个礼拜没有一天是12点前睡得，感觉都要猝死了。

1. 《软件构造》课程进展到目前，你对该课程有何体会和建议？

老师多一点讲解。多一点中文。少一点实验作业。