

**2021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**

2 实验环境配置

3 实验过程

3.1 Poetic Walks

3.1.1 Get the code and prepare Git repository

3.1.2 Problem 1: Test Graph <String>

3.1.3 Problem 2: Implement Graph <String>

3.1.3.1 Implement ConcreteEdgesGraph

3.1.3.2 Implement ConcreteVerticesGraph

3.1.4 Problem 3: Implement generic Graph<L>

3.1.4.1 Make the implementations generic

3.1.4.2 Implement Graph.empty()

3.1.5 Problem 4: Poetic walks

3.1.5.1 Test GraphPoet

3.1.5.2 Implement GraphPoet

3.1.5.3 Graph poetry slam

3.1.6 使用Eclemma检查测试的代码覆盖度

3.1.7 Before you’re done

3.2 Re-implement the Social Network in Lab1

3.2.1 FriendshipGraph类

3.2.2 Person类

3.2.3 客户端main()

3.2.4 测试用例

3.2.5 提交至Git仓库

4 实验进度记录

5 实验过程中遇到的困难与解决途径

6 实验过程中收获的经验、教训、感想

6.1 实验过程中收获的经验和教训

6.2 针对以下方面的感受

# 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

# 实验环境配置

本次实验需要在 Eclipse IDE 中安装配置 EclEmma，下载网站如下： [http://www.eclemma.org](http://www.eclemma.org。)

了解 EclEmma 并学习其安装、配置和使用。具体学习博客如下：https://blog.csdn.net/aotian16/article/details/7812751

我的仓库如下：

https://github.com/ComputerScienceHIT/HIT-Lab2-1190301610

# 实验过程

## Poetic Walks

读入一个文本，转化为有向加权图，点——单词；边——相邻，边上的权重——相邻次数。设计ADT，用Graph<L>,L为泛型，L代替任何一个具体的数据类型； Graph<L>，接口，interface；spec已经提供了，没有任何实现；

第一步：写Graph<String>的测试用例，静态方法，实例方法测试

第二步：实现 Graph<String>具体的类，内部数据表示，方法实现，给出两种实现思路，写两个函数

第三步：替换String为L，写一个抽象的ADT

第四步：Graph<L>实现，读入一个文本，转化为有向加权图，点，单词；边，相邻，边上的权重，相邻次数。

注：表示泄露，check，toString

做一个图的数据结构，图为有向，边有权（int），节点为immutable类型数据的图，可以完成给定spec的操作，并自己设计测试。

### Get the code and prepare Git repository

从这个网址上下载仓库：

<https://github.com/rainywang/Spring2021_HITCS_SC_Lab2/tree/master/P1>

### Problem 1: Test Graph <String>

写graph<String>的测试用例。

要求写一个有向图的测试用例。根据方法的功能，我想要做如下测试：

1、测试是否可以加入、删除点 public void testAddOrRemovevertices()

2、测试是否可以加入、删除边public void testAddorRemoveedge()

3、测试是否可以正确返回以某点作为起点、终点的所有边 public void testTarget() 、 public void testSource()

4、写了一个环状图，测试 public void testTargetVerticesGraph()

### Problem 2: Implement Graph <String>

#### Implement ConcreteEdgesGraph

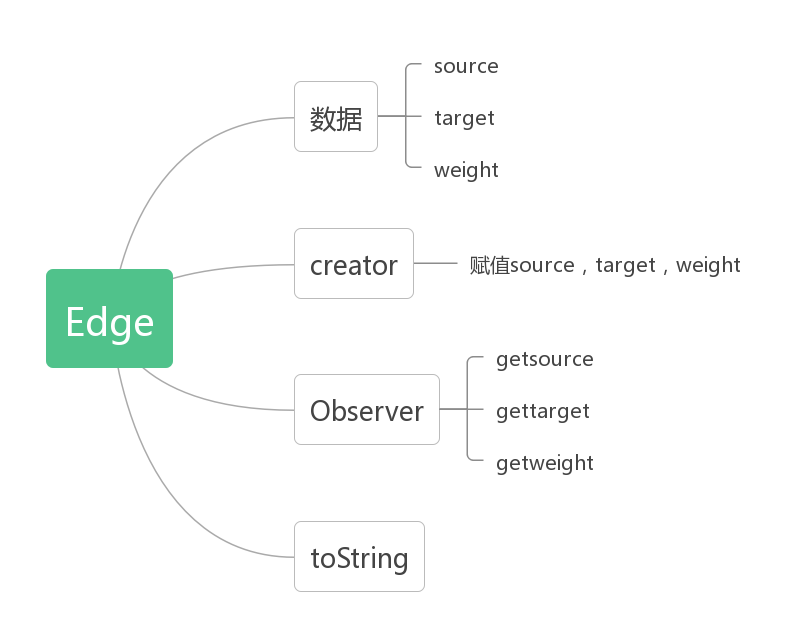
本任务要求根据Graph的接口里面的spec，写一个具体的Graph实现类，顶点的类型不要求泛型，直接当做String来处理即可。

ConcreteEdgesGraph里面已经给了数据结构，

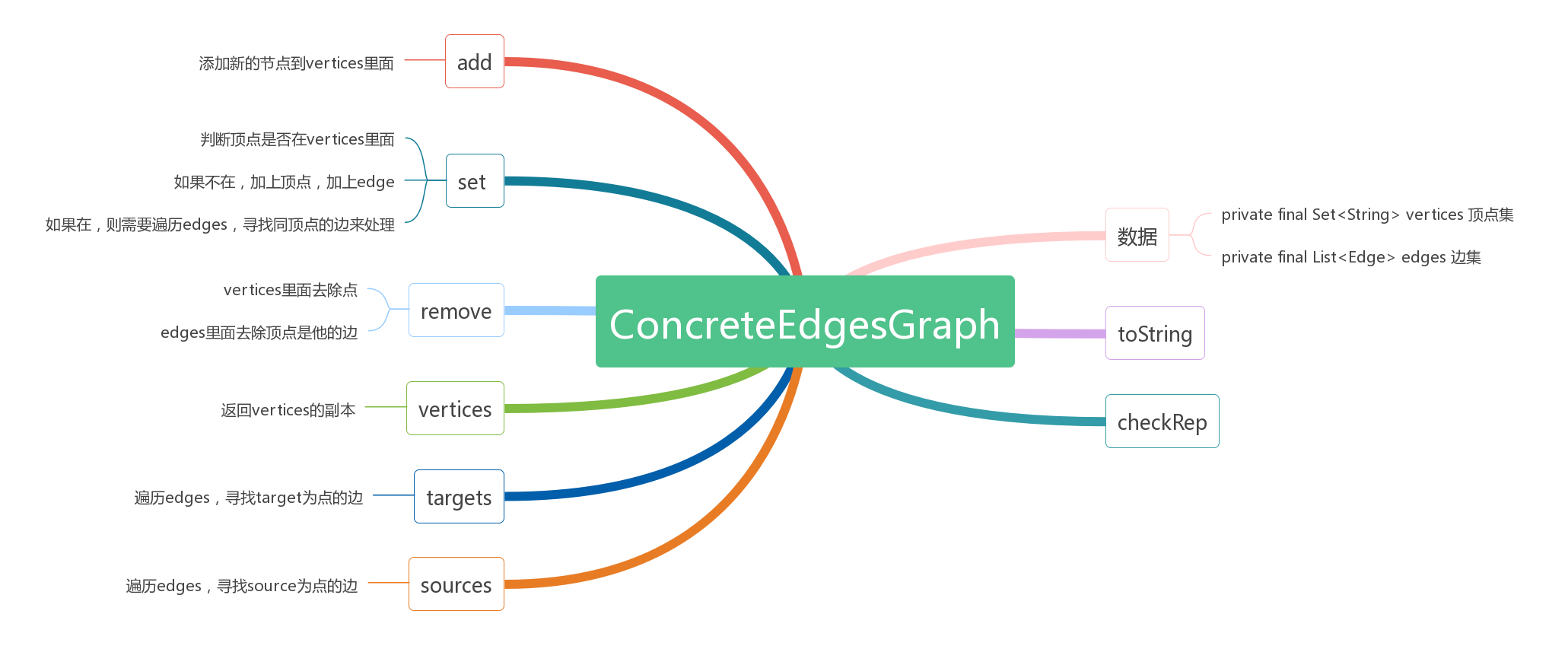
**private** **final** Set<String> vertices = **new** HashSet<>();

**private** **final** List<Edge> edges = **new** ArrayList<>();

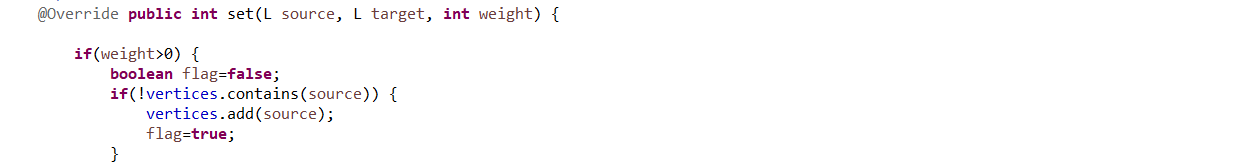
首先先定义Edge类，他的的结构和实现方法大致如下：



ConcreteEdgesGraph各部分代码的大致实现思路如下：



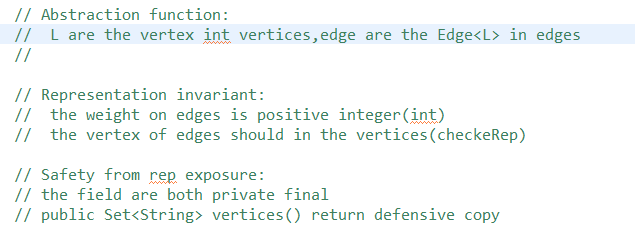
其中比较复杂的函数set实现如下，分为weight>0和weight=0两种情况，对每种情况进行了讨论：



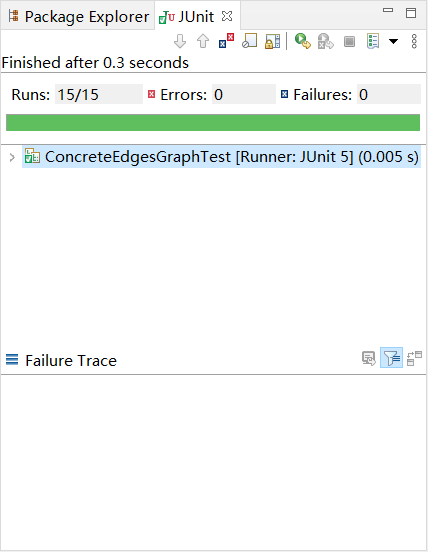




AF，RI，数据表示保护措施如下：

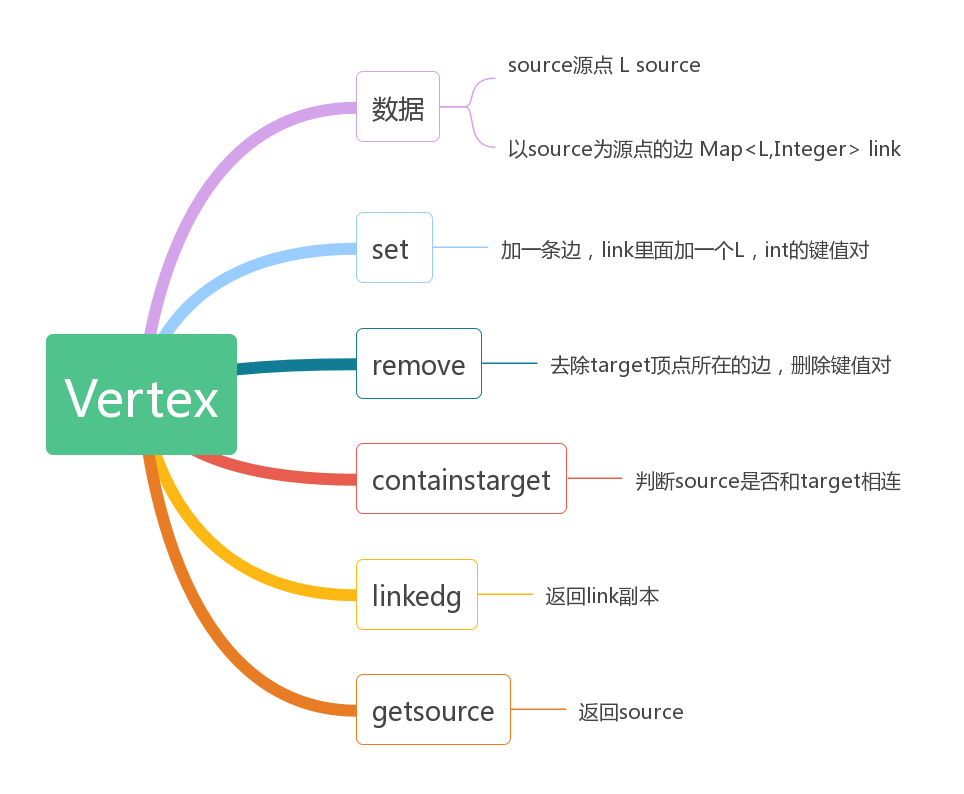


Junit测试结果如下：

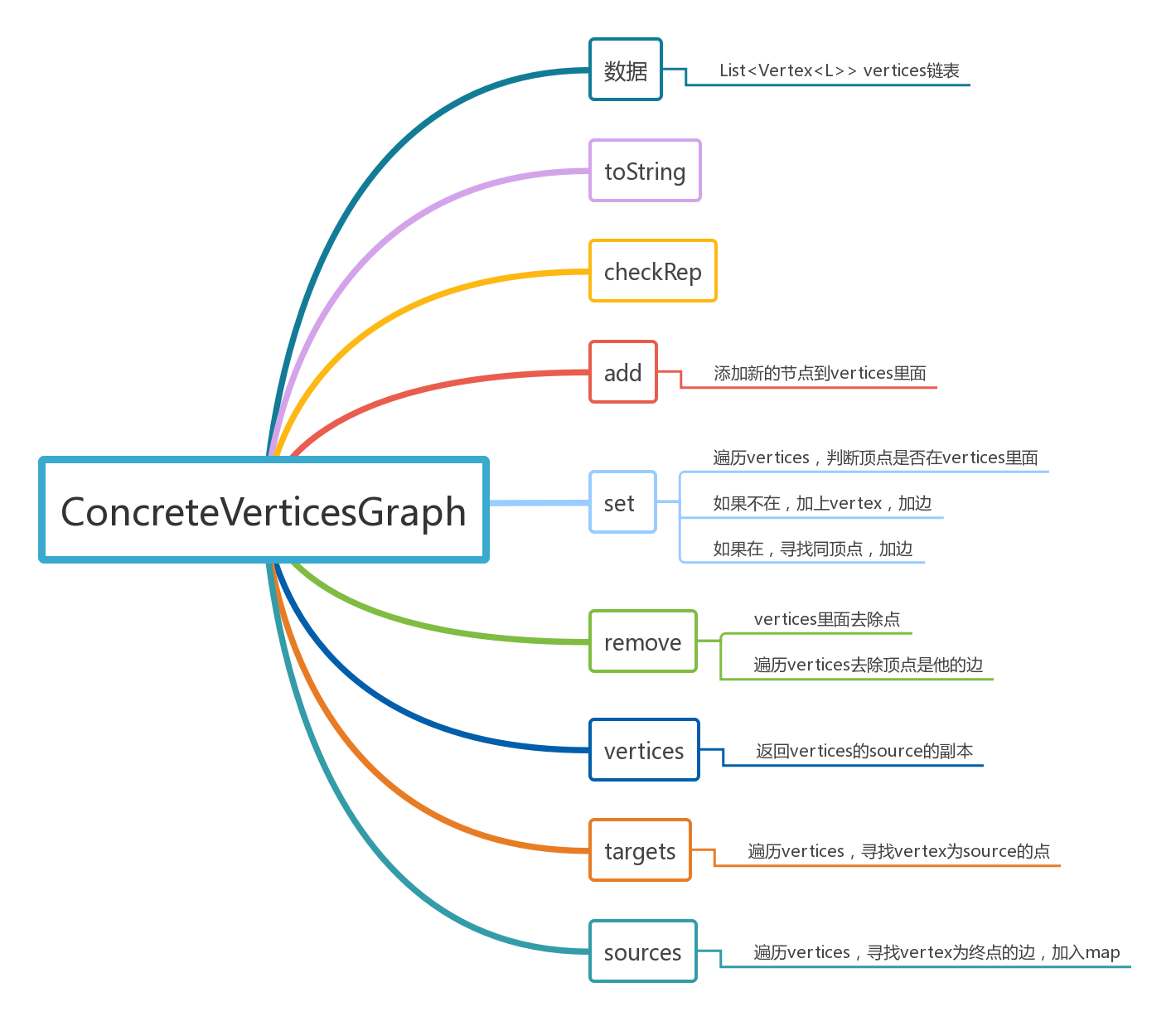


#### Implement ConcreteVerticesGraph

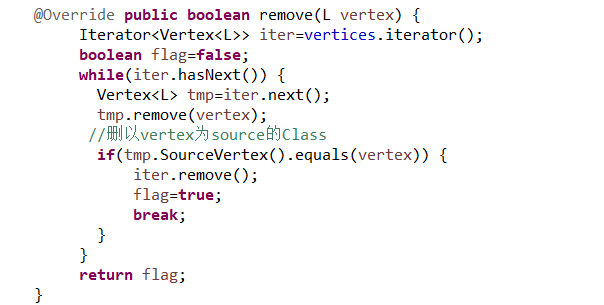
Vertex的结构和实现思路大致如下：

****

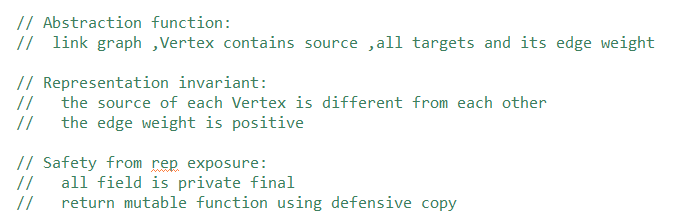
ConcreteVerticesGraph的各函数的实现思路大致如下：

****

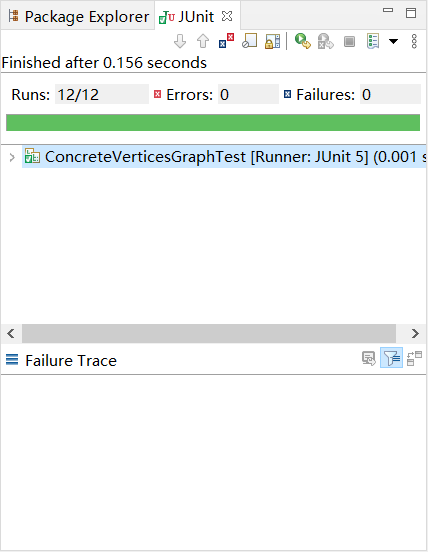
Remove函数的实现代码如下，遍历vertices，删除target或者source为vertex的边：



AF RI 表示泄露保护措施如下：



Junit测试

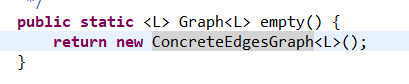


### Problem 3: Implement generic Graph<L>

#### Make the implementations generic

将String全部替换为L即可，记得在Class的声明里面加上L。

#### Implement Graph.empty()



### Problem 4: Poetic walks

给定一个语料库corpus，根据corpus中的文本生成一个单词图，然后给定一条语句输入，在图中搜索词之间的关系，自动补全语句中可能可以完善的部分。

图的构建规则是，在corpus中，对每一个不一样的单词看作一个顶点，相邻的单词之间，建立一条有向边，相邻单词对出现的次数，作为这条有向边的权值。在输入信息补全时，对相邻单词A和B做检查，如果存在一个单词C，在图中可以由前一个单词A通过这个单词C到达单词B，那么就在A和B之间补全C，补全的优先级按照权值越大者优先。

#### Test GraphPoet

#### 在基于预设的测试用例基础上，增加等价类划分的多种情况。

#### 等价类划分：两个单词之间不存在连接词，两个单词之间只有一个连接词，两个单词之间有多个连接词。此外还要注意句末的句号，测试当一个句子最后一个词是“桥”的一端。

#### 考虑多个文本，比如空文本，长的文本，网站给出的测试用例的文本，进行测试。

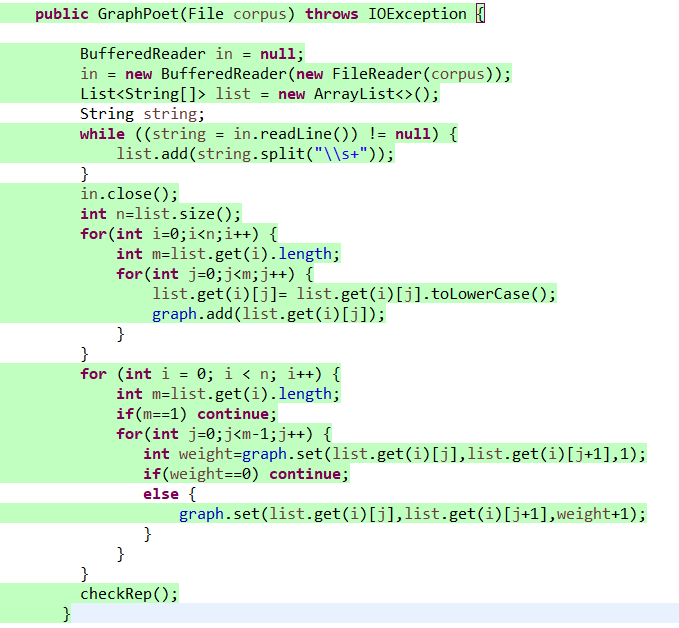
#### 

#### Implement GraphPoet

构造函数public GraphPoet(File corpus) throws IOException：

用文件输入单词，String.split()分割为数组，通过String.toLowerCase()转化为小写。

接下来构建图，首先要在加边前通过Graph.add()加点，相邻的单词加边。加边时要判断是否存在：由于Graph.set()能返回之前加的边的值，以此来判断是否存在，存在则在之前的值加一。



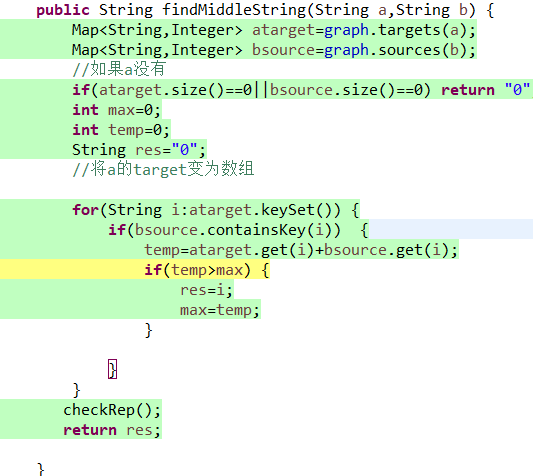
#### Graph poetry slam

public String poem(String input)，根据规则来返回String

为了简化implement，写了一个函数public String findMiddleString(String a,String b)，这个函数的功能是给定两个String单词，如果两个单词之间有桥，则返回这个String。

具体实现如下：

当相邻两个单词任意一个不在之前创建的图里，无桥，由于Bridge长度只能为2，所以：分别求两个单词的sources和targets，对他们的key值求交集，若交集为空，则无桥，若交集不空，则在交集中找weight最大的桥（可以在Map的value中查询weight）。



### 使用Eclemma检查测试的代码覆盖度

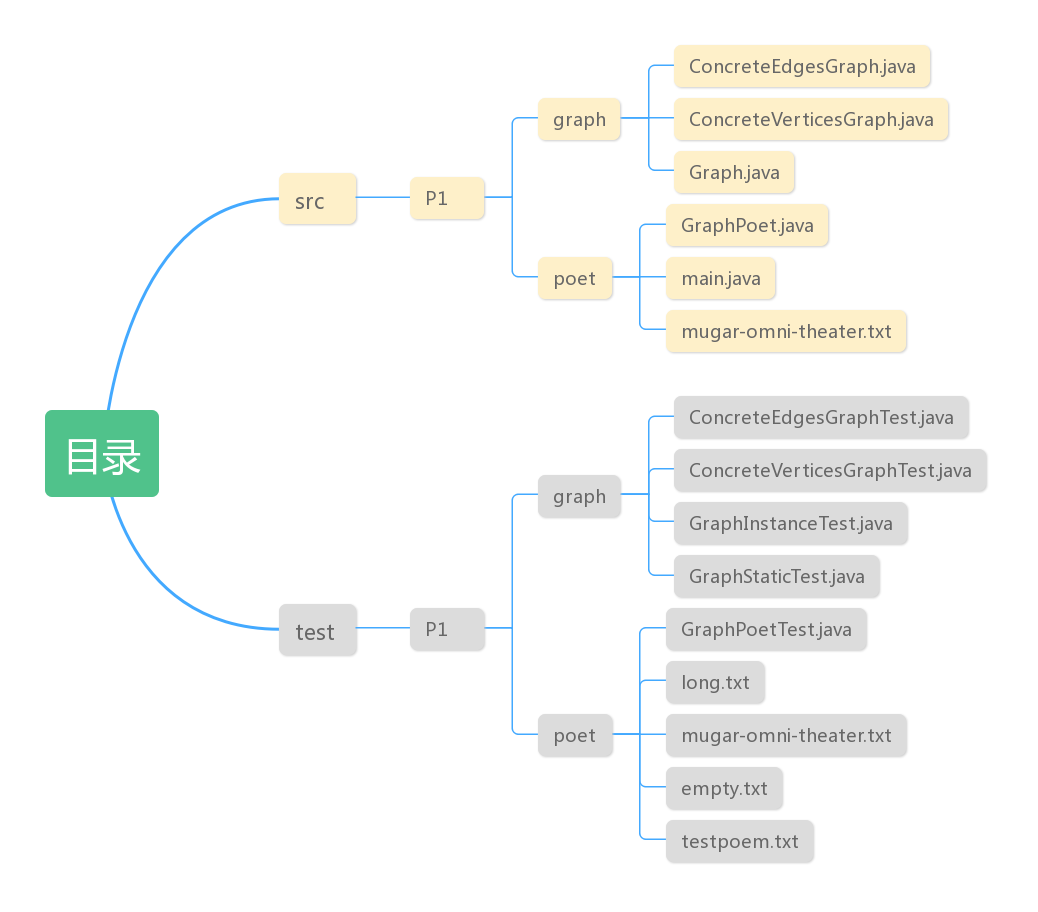
根据Eclemma测试，Graphpoet的代码绝大部分都是绿色，只有极少数判断语句和函数名是黄色，没有红色。

### 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/)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。

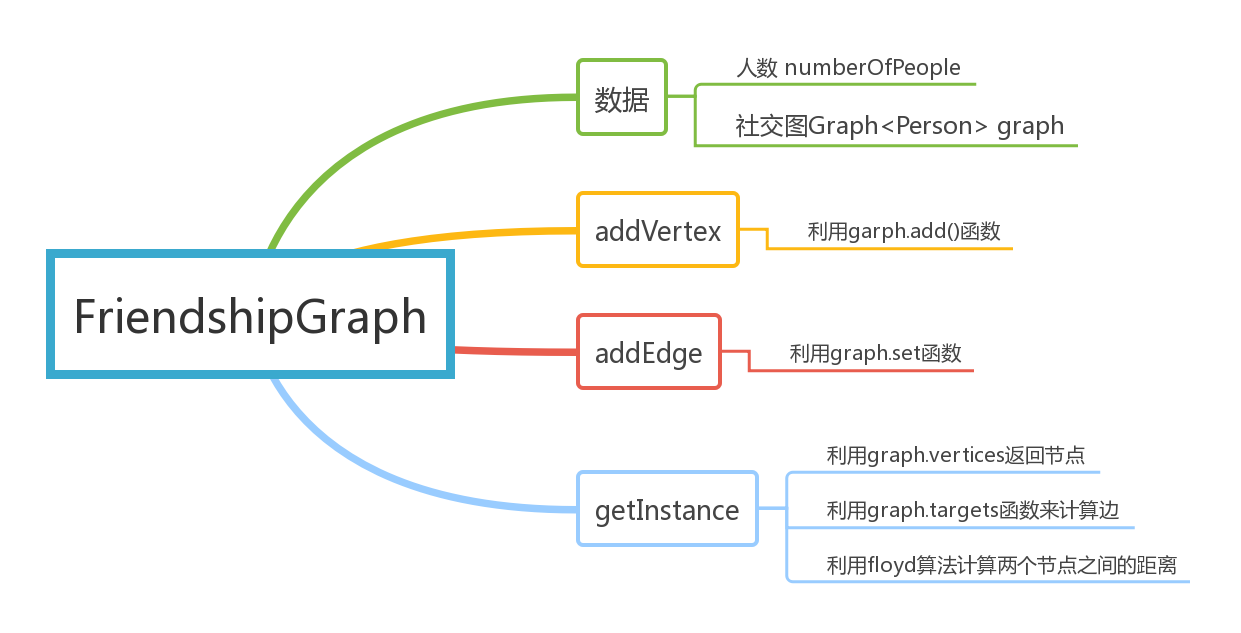
在这里给出你的项目的目录结构树状示意图。



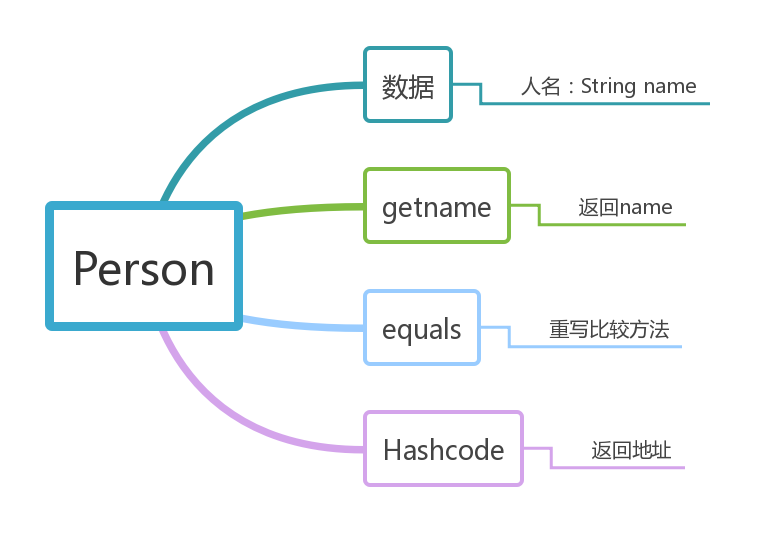
## Re-implement the Social Network in Lab1

利用之前实现的graph的接口，重新实现一个lab1里面的社交网络的结构。

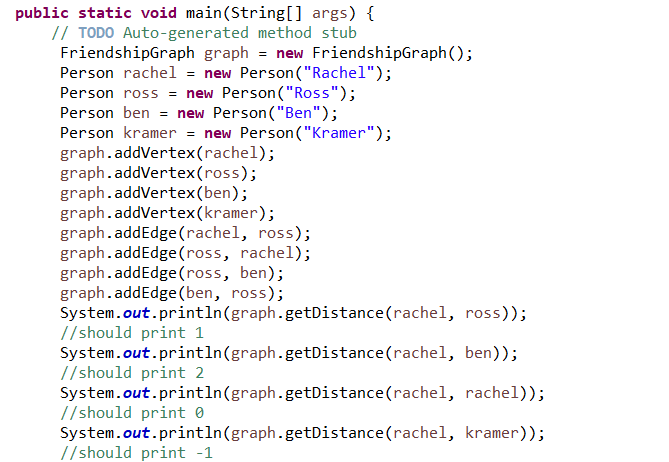
### FriendshipGraph类



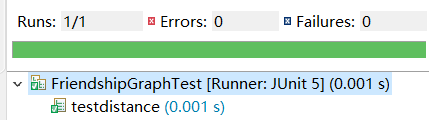
### Person类



### 客户端main()



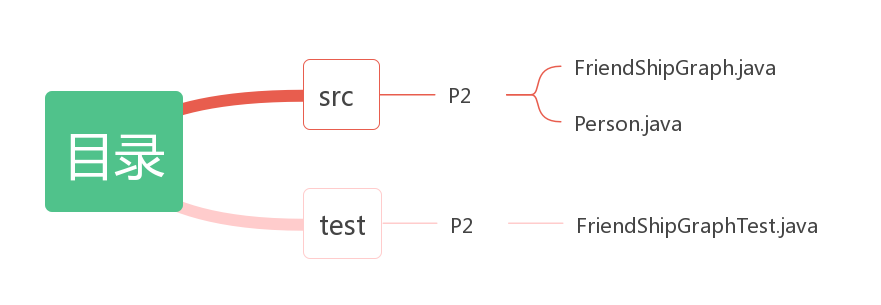
### 测试用例



### 提交至Git仓库

如何通过Git提交当前版本到GitHub上你的Lab3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。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 2021.5.25 | 5.6节实验课 | 完成ConcretEdgesGraph | 基本完成 |
| 2021.5.29 | 晚上 | 完成ConcreVerticesGraph | 基本完成 |
| 2021.6.1 | 5.6节实验课 | 完成Graphpoet | 未完成，发现bug |
| 2021.6.2 | 晚上 | 完成Graphpoet | 调试成功 |
| 2021.6.3 | 晚上 | 完成P2 | 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 |
| 看不太懂他让我干啥 | 反复阅读MIT官网的要求，graph里面的spec，查找相关资料，和同学进行讨论 |
| 老是觉得代码写的不漂亮，向重写 | 克制自己，比如可能一个遍历，可以用for（i；i<n；i++），或者用for（Edge i：edges），也可以用迭代器，然后我就想用迭代器，但是还有一些问题，最后还是改了第一个，虽然丑，但是没啥bug |
| String的比较 | 发现了==和equals的区别 |
| 对于AF,RI,CheckRep的概念理解的不清楚 | 阅读PPT，根据资料自己填写、修改 |

# 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

明白了程序员一定要心平气和，不要太暴躁，一个代码可能要改好几遍，不仅仅是因为bug，也可能是因为没有看懂spec或者实验要求，这个时候一定要耐心。

对于不熟悉的一些接口或者数据集合的调用、比较，要及时查明原理，起码要了解大概，不然容易出无名bug。

## 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？

ADT注重数据结构，实现方法，面向应用场景注重功能有没有准确实现。

1. 使用泛型和不使用泛型的编程，对你来说有何差异？

感觉没啥大差异，泛型可能反而更抽象简单一点，就是注意一下不要用某个类型特定的方法即可，

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？

可以比较好的明确各个方法的spec，已经可能出现的极端情况，这样可以提高代码的准确性和健壮性，不至于迷失方向。

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？

很方便，再设计别的代码的时候不用重写了。

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？

更进一步明确ADT的结构和规则，不要自己脑子不清楚，出现逻辑上的bug，这样是很难发现的，可能造成巨大损害。

并且注重ADT的健壮性，可以面对极端环境工作。

1. 关于本实验的工作量、难度、deadline。

其实工作量不是很大，关键是要明白他让你干啥，我理解让我干啥这个花了很多时间，也走了很多弯路

1. 《软件构造》课程进展到目前，你对该课程有何体会和建议？

觉得课程还是非常具有实用性的，目的性和逻辑性很强，老师讲的也很有激情，但是希望老师可以再把实验的要求什么的更强化一下，比如给大家讲讲每个实验都是要你干啥，注意什么之类的。