

**2018年春季学期  
计算机学院大二软件构造课程**

**Lab 3实验报告**

|  |  |
| --- | --- |
| 姓名 | Arcwoc |
| 学号 | xxxxx |
| 班号 | xxxxxx |
| 电子邮件 | xxxxx |
| 手机号码 | xxxxx |

**目录**

[1 实验目标概述 1](#_Toc508910344)

[2 实验环境配置 1](#_Toc508910345)

[3 实验过程 1](#_Toc508910346)

[3.1 待开发的四个应用场景 1](#_Toc508910347)

[3.2 面向复用的设计：Graph<L, E> 1](#_Toc508910348)

[3.3 面向复用的设计：Vertex 1](#_Toc508910349)

[3.4 面向复用的设计：Edge 1](#_Toc508910350)

[3.5 可复用API设计 2](#_Toc508910351)

[3.6 图的可视化：第三方API的复用（选做） 2](#_Toc508910352)

[3.7 设计模式应用 2](#_Toc508910353)

[3.7.1 使用State/Memento模式进行Vertex的状态管理（选做） 2](#_Toc508910354)

[3.7.2 使用factory method模式构造Vertex对象 2](#_Toc508910355)

[3.7.3 使用factory method模式构造Edge对象 2](#_Toc508910356)

[3.7.4 使用abstract factory或builder模式构造Graph对象 2](#_Toc508910357)

[3.7.5 使用Strategy模式调用centrality度量算法 2](#_Toc508910358)

[3.7.6 使用Composite模式设计超边对象（选做） 2](#_Toc508910359)

[3.7.7 使用decorator模式构造不同特征的Edge对象（选做） 2](#_Toc508910360)

[3.7.8 使用其他设计模式（选做） 2](#_Toc508910361)

[3.8 读取基于语法的文件并进行语法解析 2](#_Toc508910362)

[3.9 图操作指令的输入和处理 2](#_Toc508910363)

[3.10 应用设计与开发 3](#_Toc508910364)

[3.10.1 单词网络GraphPoet 3](#_Toc508910365)

[3.10.2 微博社交网络SocialNetwork 3](#_Toc508910366)

[3.10.3 网络拓扑图NetworkTopology 3](#_Toc508910367)

[3.10.4 电影网络MovieGraph 3](#_Toc508910368)

[4 实验进度记录 3](#_Toc508910369)

[5 实验过程中遇到的困难与解决途径 3](#_Toc508910370)

[6 实验过程中收获的经验、教训、感想 3](#_Toc508910371)

# 1 实验目标概述

本次实验覆盖课程第 3、5、6 章的内容,目标是编写具有可复用性和可维护

性的软件,主要使用以下软件构造技术:

①子类型、泛型、多态、重写、重载②继承、代理、组合③常见的 OO 设计模式

④语法驱动的编程、正则表达式⑤基于状态的编程⑥API 设计

本 次 实 验 给 定 了 四 个 具 体 应 用 ( Lab 2 中 的 GraphPoet 、 Lab 1 中 的SocialNetwork、网络拓扑结构 NetworkTopology、电影网络 MovieGraph),不是直接针对四个应用分别编程实现,而是通过 ADT 和泛型等抽象技术,开发一套可复用的 ADT 及其实现,在 Lab 2 所完成的抽象数据类型 Graph<L>的基础上,进一步扩展至 Graph<L,E>,充分考虑这些应用之间的相似性和差异性,使ADT 有更大程度的复用和更容易面向各种变化(可维护性)。

# 2 实验环境配置

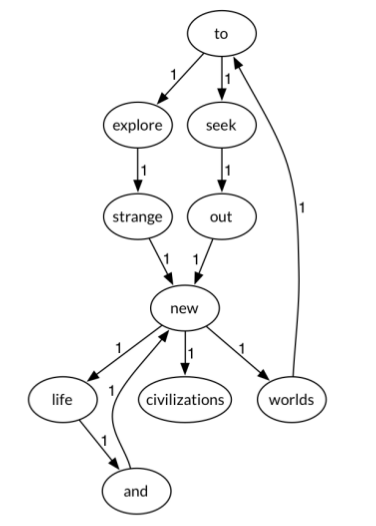
配置同前两个实验。

<https://github.com/ComputerScienceHIT/Lab3-1160301012>

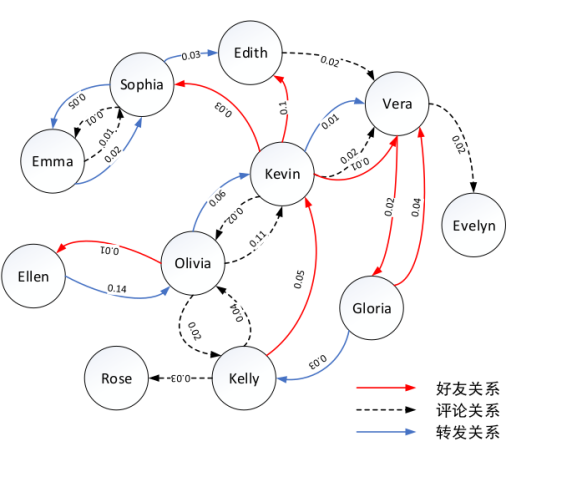
# ３ 实验过程

## 3.1 待开发的四个应用场景

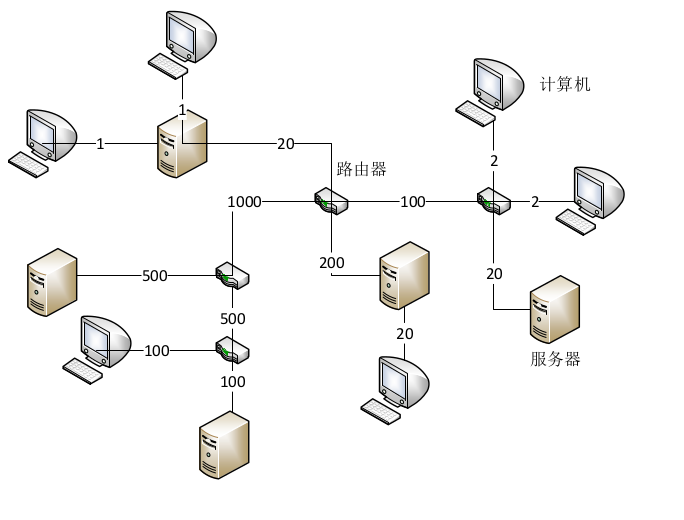
1. 单词网络GraphPoet:具体参见 Lab 2 中 3.1 节的说明,节点为“单词”(label 为该单词的文本字符串,无其他属性),边为两个单词在文本中的相邻关系,边的权重是相邻出现的次数(值域为正整数)。图中可以出现loop,即一条边的起点和终点为同一个节点。它是单重有向图、带权图、单模图。



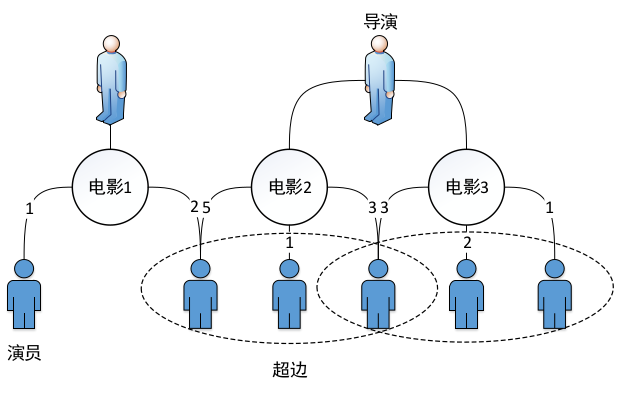
1. 微博社交网络 SocialNetwork:节点为“用户”(label为用户姓名,其他属性包括:性别、年龄),边为两个用户的社交关系。两个用户之间最多可存在3种类型的社交关系边,分别表征“好友关系”(A关注了B,边的方向为A->B)、评论关系(A曾评论过B的微博,边的方向为 A->B)、转发关系(A曾转发过B的微博,边的方向为A->B)。边的权重表示二者通过特定社交关系类型进行交互的频度,取值范围为(0,1],图中所有边的权值之和=1。当有新边加入图、从图中删除已有边、调整已有边的权值时,需重新调整其他各边的权值,以满足所有边权值之和=1的约束条件



1. 网络拓扑图 NetworkTopology:节点为“计算机”、“服务器”、“路由器”(label 为主机名,属性包括:IP 地址),边为它们之间的网络连接关系(但计算机之间不能直接相连、服务器之间不能直接相连),权重为网络连接的带宽(例如 1、20、100)。图中不能出现 loop,即一条边的起点和终点为不能为同一个节点。它是简单图、带权图、多模图。



1. 电影网络 MovieGraph:节点为“电影”(label 为电影名,还有三个属性:上映年份、拍摄国家、IMDb 评分)、“演员”(label 为姓名,属性:年龄、性别)、“导演”(label 为姓名,属性:年龄、性别),有两种无向边:演员 A 参演了电影 M、导演 D 执导了电影 M。这是单重图、多模图,演员和电影之间的边有权值(表示 A 在 M 中的角色次序,用正整数表示),导演和电影之间的边无权值。图中不能出现 loop,即一条边的起点和终点不能为同一个节点。参演过同一部电影的所有演员形成一条超边,超边无权值。



注:在以上四个应用中,图中均不能出现具有相同 label的节点、具有相同label的边。

## 3.2基于语法的图数据输入

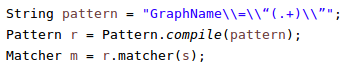
这一部分的功能是根据特定格式的输入文件，用正则表达式对其解析，并且从中抽取信息，构造图结构。

①GraphFactory

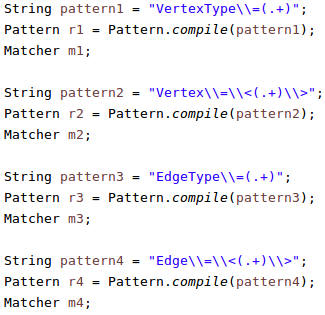
a1

用正则表达式提取GraphType的类型，然后在GraphFactory中选择要产生图的类型，以文件路径为参数调用相应的工厂方法。

②GraphPoet,SocialNetwork,NetworkTopology因为都是没有超边，在处理字符串上面类似，如下：

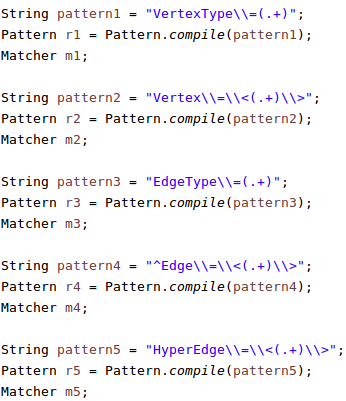


用正则匹配提取GraphName，用GraphName新建一个GraphName图



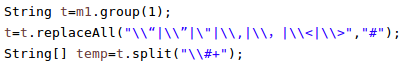
上面四个字符串分别用来匹配VertexType,Vertex,EdgeType和Edge，然后再通过字符串切割，可以获得要建图的有效信息，向图中加入顶点和边。

③MovieGraph因为有超边，所以正则表达式有一些变动



先是增加了HyperEdge类的解析，用来获得超边的信息，又因为它的存在影响了原有的普通Edge类的解析，所以也要修正Edge的正则表达式，在前面加上”^”，从开头开始匹配，否则会影响建图的正确性。

通过字符串切割，将字符串按照逗号，引号分隔开，再做一些细微的处理，就可以用来初始化一个图。

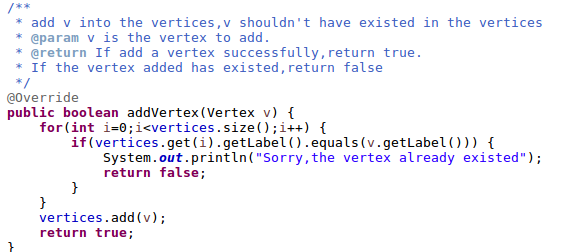


## 3.3面向复用的设计：Graph<L, E>

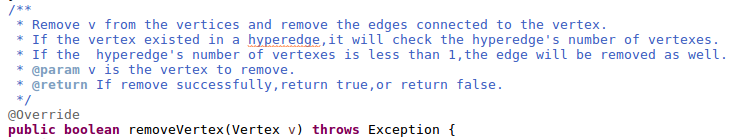
Ｌ和Ｅ分别代表节点和边的类型，该接口提供操作：

①public static <L,E> Graph<L,E> empty() 构造一张图的空实例。返回一个图的实例，里面没有节点和边。

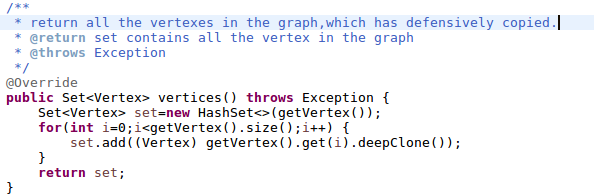
②public boolean addVertex(L v) 向图中增加一个节点



③public boolean removeVertex(L v) 从图中删除一个节点 v。如果v 是某条边的两端之一,则该边被删除;如果某节点属于某条超边,若该节点删除后该条超边仍可合法存在,则该超边继续保留,否则就删除之。

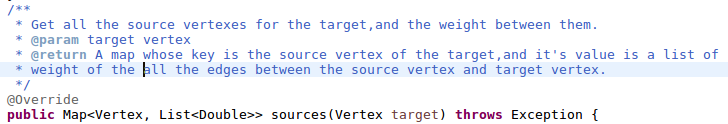


④public Set<L> vertices() 返回图的节点集合

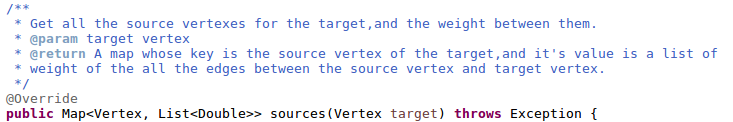


该方法利用深拷贝来做防御性拷贝防止表示暴露。

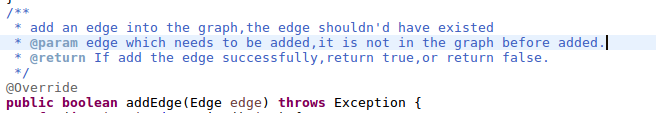
⑤public Map<L, List<Double>> sources(L target) 与Lab2中Graph接口的同名操作含义相同,返回的Map中Key为source节点,List<Double>为当前节点与该source节点之间的所有边的权值;如果与target相连的边包括无向边,则无向边的另一端节点也需包含在返回值Map中;不需考虑超边。



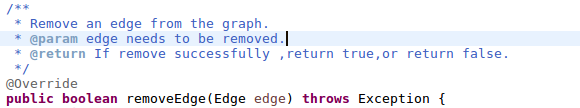
⑥public Map<L, List<Double>> targets(L source) 与Lab2中Graph接口的同名操作含义相同;如果与source相连的边包括无向边,则无向边的另一端节点也需包含在返回值Map中;不需考虑超边。



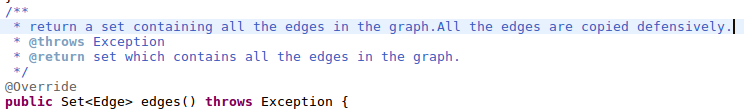
⑦public boolean addEdge(E edge)增加一条边(包括超边)



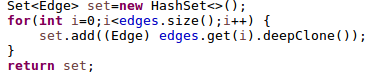
⑧public boolean removeEdge(E edge) 删除一条边(包括超边)



⑨public Set<E> edges() 返回边的集合(包括超边)



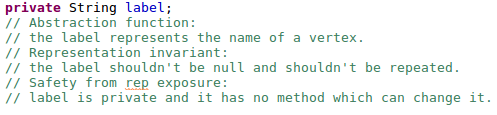
该方法用序列化深拷贝来达到防止表示暴露的作用。



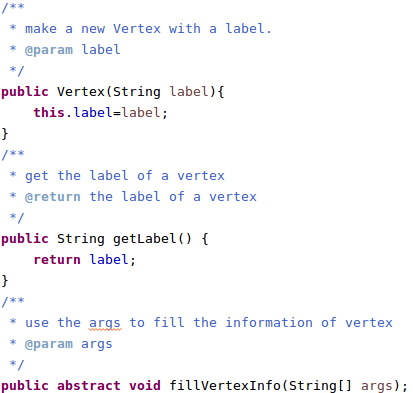
## 3.4面向复用的设计：Vertex

Vertex是抽象类，定义各种节点的共性数据以及在共性数据上的操作。包含了外部可以看到的节点的标签信息，以及可以获得标签的方法。除此之外，还有相应的equals()方法和hashcode()方法以及toString()方法。根据各个子类型的属性不同，提供了统一的抽象方法fillVertexInfo()可以为具体节点添加详细属性信息。子类可以通过override或者overload等方式，支持具体需求。

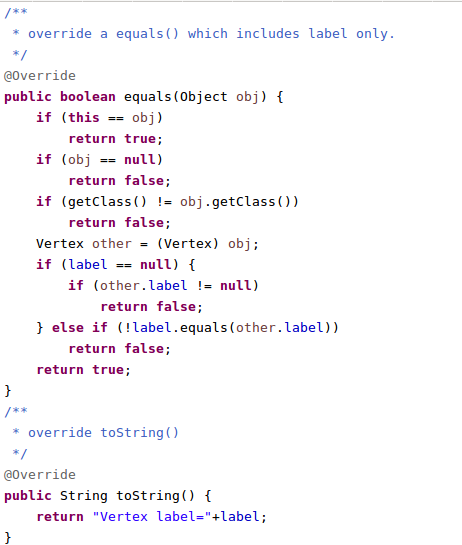
Vertex的属性和AF,RI,Savety from rep exposure.



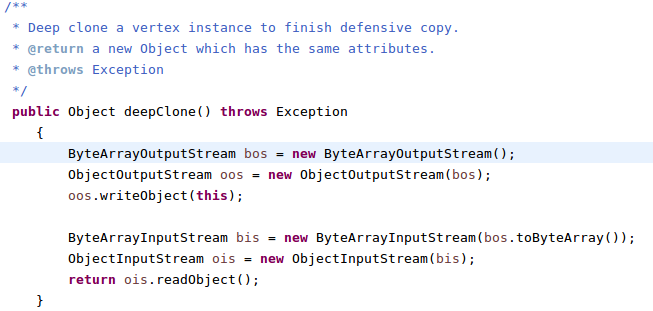
重要方法：



因为label是每个顶点都不可以重复的属性，所以可以用它来重写equals()方法和hashCode()方法



**另外，为了防止在图中或者边中表示暴露，Vertex实现了Serializable接口，通过序列化和反序列化，实现了一个深拷贝方法。**

****

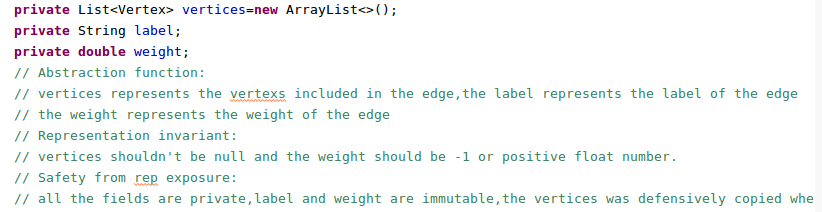
## 3.5 面向复用的设计：Edge

Edge是抽象类，定义各种边的共性数据以及在共性数据上的操作。

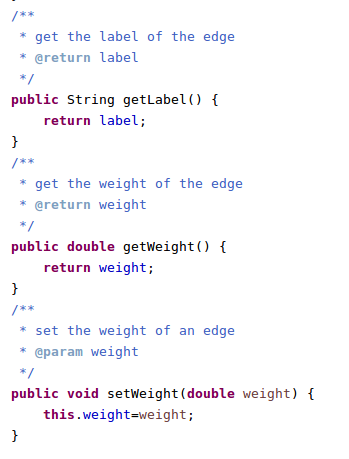
属性包含了外部可以看到的标签信息，以及内部存储了改变所包含的所有节点，每个节点的都是Vertex的子类的实例，此处使用的是ArrayList作为Edge的Rep，还有weight属性记录了边的权重，如果是带权边，则为非负数，否则属性为－１．

方法包含了构造函数，为边赋予了标签和权重，containVertex(Vertex v)判断边中是否包含制定的节点ｖ，vertices()返回边包含的点集，此外还有相应的还有相应的equals()方法和hashcode()方法以及toString()方法。根据子类的类型不同，提供了抽象函数addVertices(List<Vertex> vertices)向边中加入顶点，containVertex(Vertex v)判断边中是否包含指定的顶点ｖ，sourceVertices()返回该边源节点的集合，targetVertices()返回该边目标节点的集合。

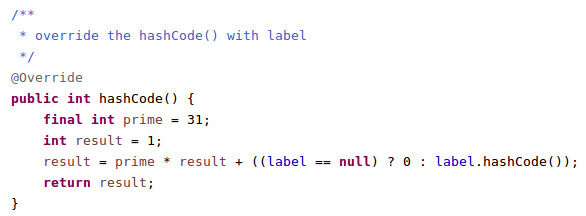
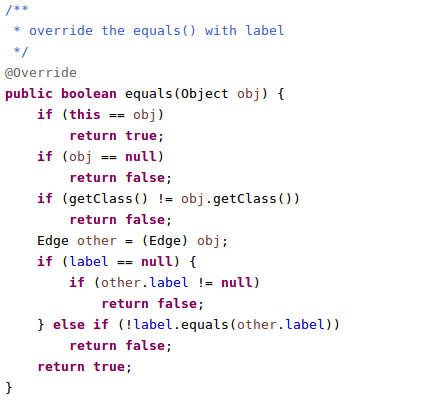
Edge的属性和AF,RI,Savety from rep exposure



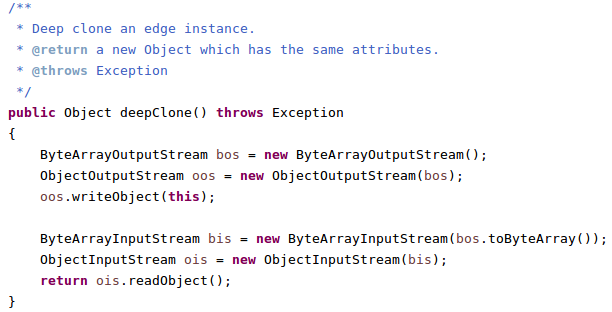
重要方法：



因为label是每条边都不可以重复的属性，所以可以用它来重写equals()方法和hashCode()方法



**另外，为了防止在图中或者边中表示暴露，Edge实现了Serializable接口，通过序列化和反序列化，实现了一个深拷贝方法。**

****

## 3.6 可复用API设计

遵循facade模式，完成一系列关于图的计算的方法，把所有的API放置在helper包中的GraphMetrics类中。

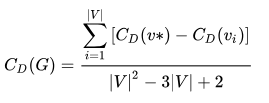
求最短路径的时候我使用的算法是Floyd算法，将图转化为邻接矩阵，找出所有最短路径使用的方法是每次遇到扩展后长度与不扩展长度相等的点时，将该点扩展和不扩展分成两条路径存储，最后在返回的时候检查结果里面是否有重复路径，如果有重复路径则删除，最终返回所有的最短路径。**在求解各种中心度的时候，按照老师答疑，我采取的方法是把所有的边权设置为1，当然如果要设置成图中本来的边的weight的话也很简单，直接把根据图构造邻接矩阵的代码里的1改成weight就可以了。后来老师答疑的时候表示可以调用第三方库jung里的API，所以部分中心度的求解调用了第三方库，但是之前的代码也在GraphMetrics2.java里有实现。**

①static double degreeCentrality(Graph<L,E> g, L v)计算图g中节点v的degree centrality

计算图g中顶点v的中心度，根据wiki介绍，可以知道计算方法为：若g是有向图，则计算的是v顶点的入度和出度之和。若g为无向图，则计算的是顶点v的入度。

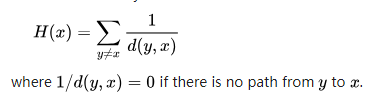
②static double degreeCentrality(Graph<L,E> g)计算图g的总体degree centrality

计算图g整体的中心度，根据wiki可以知道：



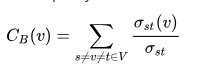
③static double closenessCentrality(Graph<L,E>g,L v)计算图g中节v的 closeness centrality

根据wiki可以知道：



另外，还要通过除以N-1来标准化（N是图中顶点的数目）。

④static double betweennessCentrality(Graph<L,E> g, L v)计算图g中节点v的 betweenness centrality。根据wiki可以知道：



另外，有向图通过除以(n-1)(n-2)来标准化，无向图通过除以(n-1)(n-2)/2来标准化。

⑤static double inDegreeCentrality(Graph<L,E> g, L v)计算图g中节点v的 indegree centrality

根据wiki可以知道，有向图连接到顶点v的边的条数称为v的入度。

⑥static double outDegreeCentrality(Graph<L,E> g, L v)计算图g中节点v的 outdegree centrality

根据wiki可以知道，有向图从v连接出去的边的条数称为v的出度。

⑦static double distance(Graph<L,E>g,L start,L end)计算两个结点之间的距离。计算start和end两节点之间的最短距离。

⑧static double eccentricity(Graph<L,E> g,L v)计算图g中v点的偏心度。

顶点v的偏心度是顶点v和其他顶点之间最短距离的最大值。

⑨static double radius(Graph<L,E> g)计算图的半径。偏心度最小的顶点的偏心度是图的半径。

⑩static double diameter(Graph<L,E>g)计算图的直径。偏心度最大的顶点的偏心度是图的直径。

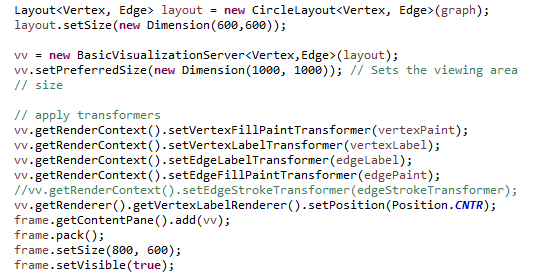
## 3.7 图的可视化：第三方API的复用（选做）

使用JUNG包中所提供API,为四个应用添加可视化功能 。在GraphVisualizationHelper 类中实现以下静态方法:

public static void visualize(Graph<L,E> g)

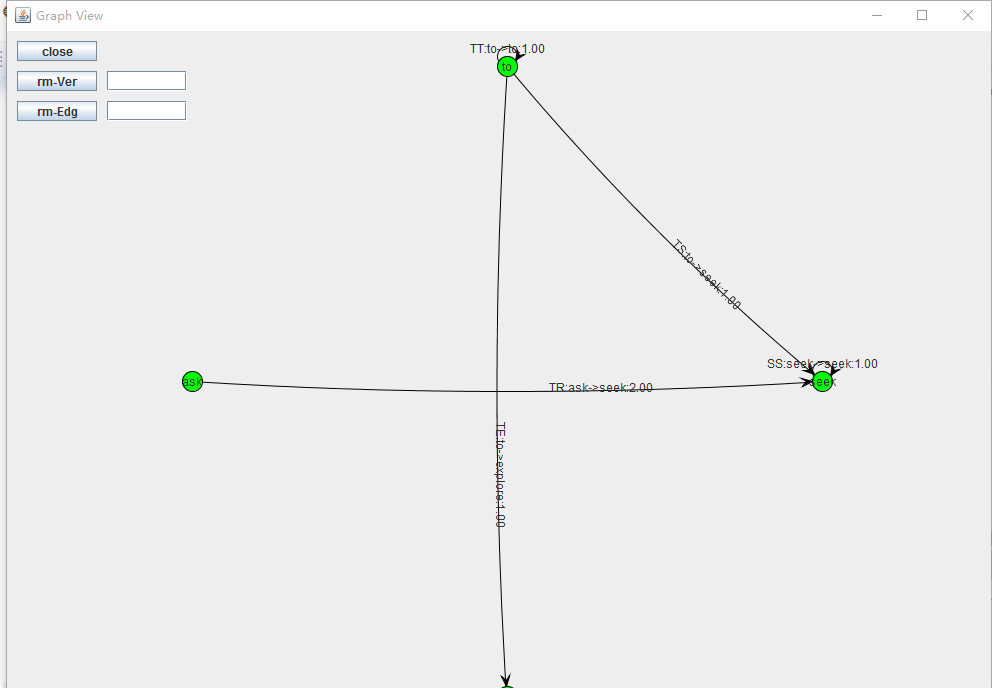
运行效果：

实现过程中，标签上显示了边的信息，如label，方向，权值等属性。在表示超边的时候，统一用红色的边来表示超边。同时在SocialNetwork中也可以表示多重边，边上打印了边的有关信息。



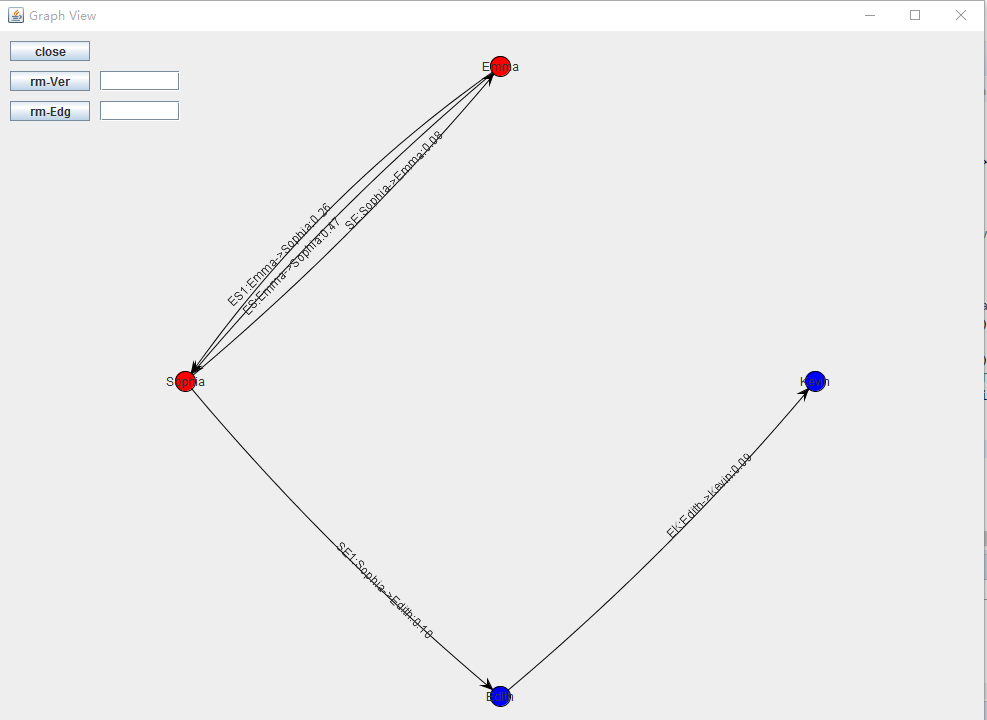
GraphPoet:

图中的结点上标有结点的label，边上标有边的方向和权值。**左上角的第一个按钮可以关闭窗口而不终止程序(如果从右上角关闭则终止程序),**左上角第二个按钮可以删除顶点以及与之关联的边，需要在右边输入要删除顶点的label(大小写敏感)。用户在第二个文本框中输入边的label(大小写敏感)然后点击左上角第三个按钮后，可以删除图中的边。



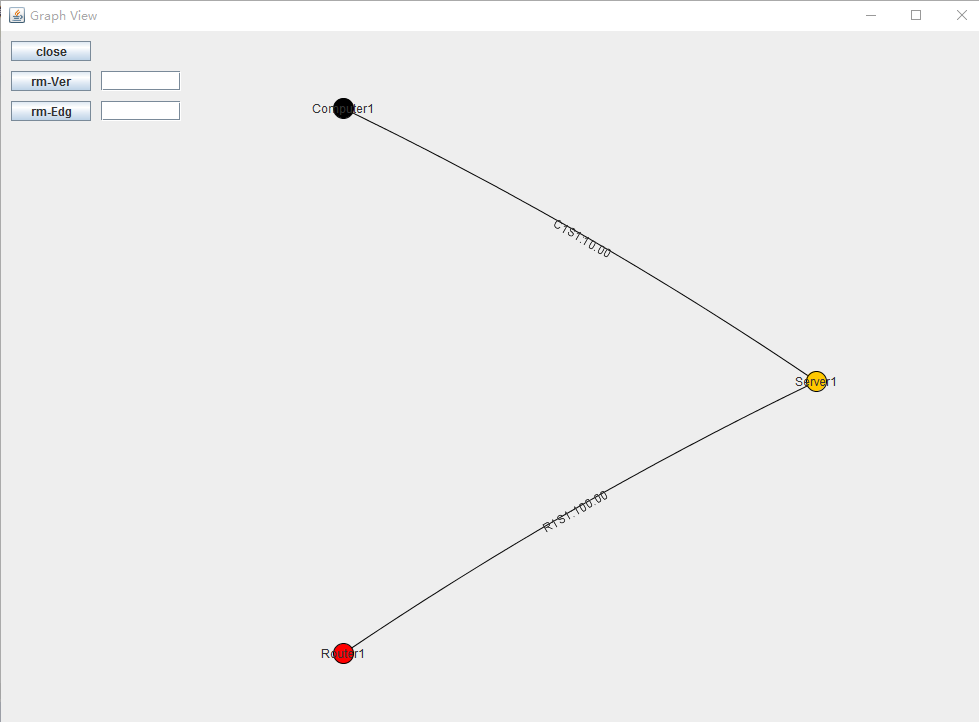
SocialNetwork:

图中的结点上标有结点的label，边上标有边的方向和权值。**左上角的第一个按钮可以关闭窗口而不终止程序(如果从右上角关闭则终止程序),**左上角第二个按钮可以删除顶点以及与之关联的边，需要在右边输入要删除顶点的label(大小写敏感)。用户在第二个文本框中输入边的label(大小写敏感)然后点击左上角第三个按钮后，可以删除图中的边。图中的结点根据性别把颜色涂成红色和蓝色。



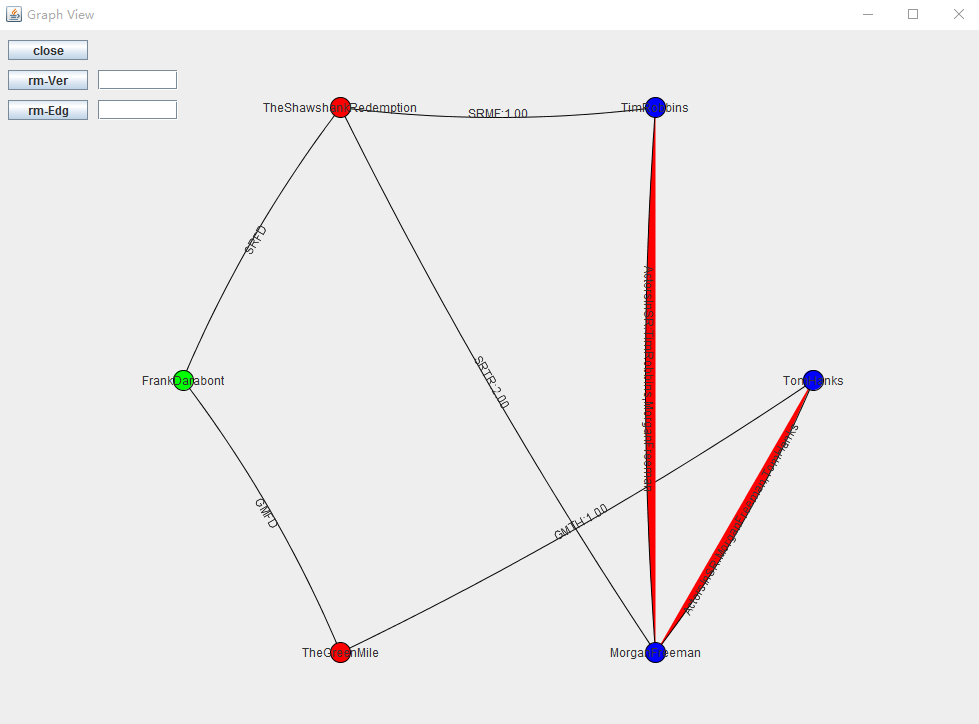
NetworkTopology:

图中的结点上标有结点的label，边上标有边的方向和权值。**左上角的第一个按钮可以关闭窗口而不终止程序(如果从右上角关闭则终止程序),**左上角第二个按钮可以删除顶点以及与之关联的边，需要在右边输入要删除顶点的label(大小写敏感)。用户在第二个文本框中输入边的label(大小写敏感)然后点击左上角第三个按钮后，可以删除图中的边。图中computer结点被染成黑色，server被染成橘黄色，router被染成红色。



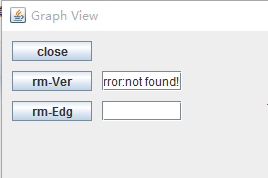
MovieGraph:（红色的是超边）

图中的结点上标有结点的label，边上标有边的方向和权值。**左上角的第一个按钮可以关闭窗口而不终止程序(如果从右上角关闭则终止程序),**左上角第二个按钮可以删除顶点以及与之关联的边，需要在右边输入要删除顶点的label(大小写敏感)。用户在第二个文本框中输入边的label(大小写敏感)然后点击左上角第三个按钮后，可以删除图中的边。图中根据结点类别的不同被染成红色，蓝色和绿色。另外，超边被染成红色。

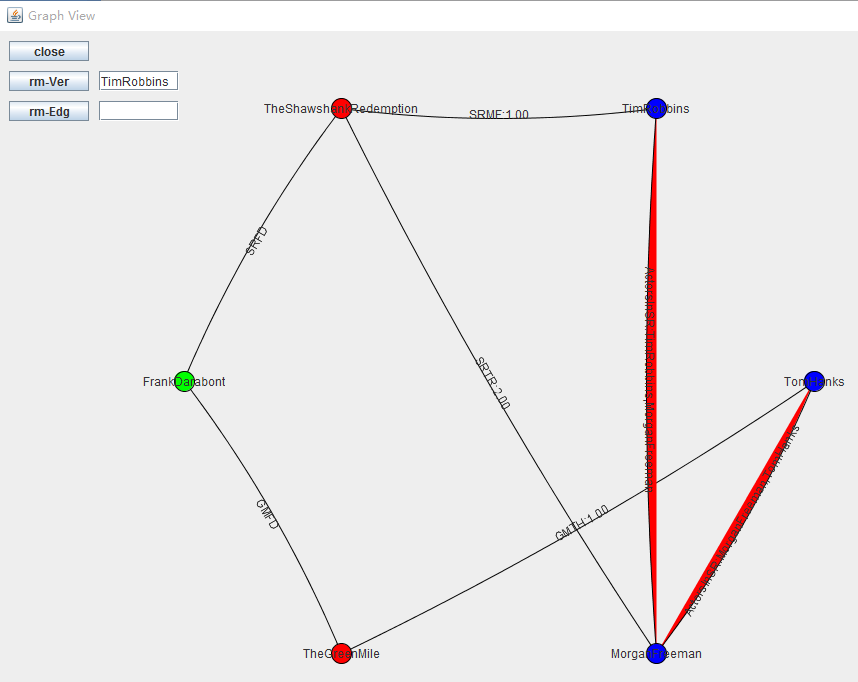


GUI删除结点：

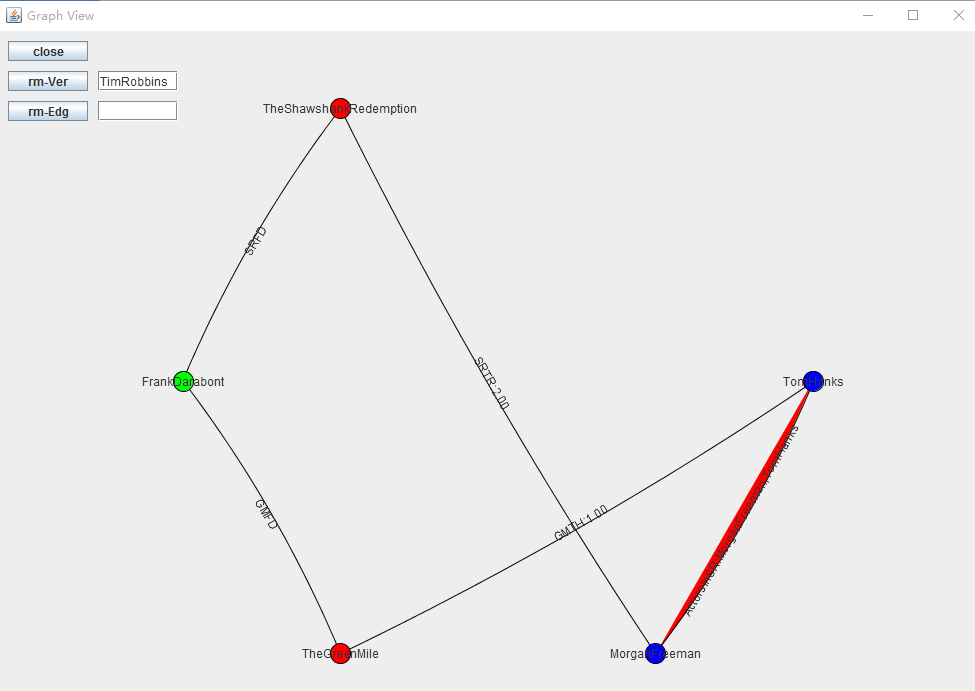
①如果在右边框中填入图中没有的结点，点击rm-Ver，框中显示”error:not found!”



②删除TimRobbins之前：

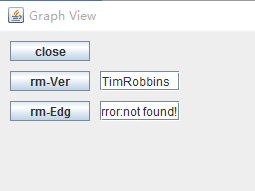


删除TimRobbins之后：

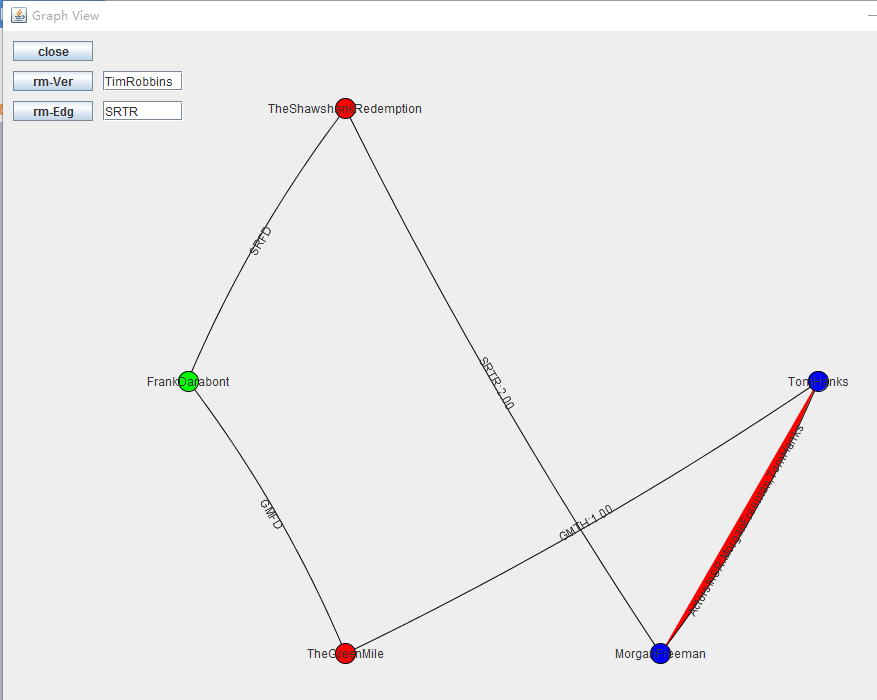


GUI删除边：

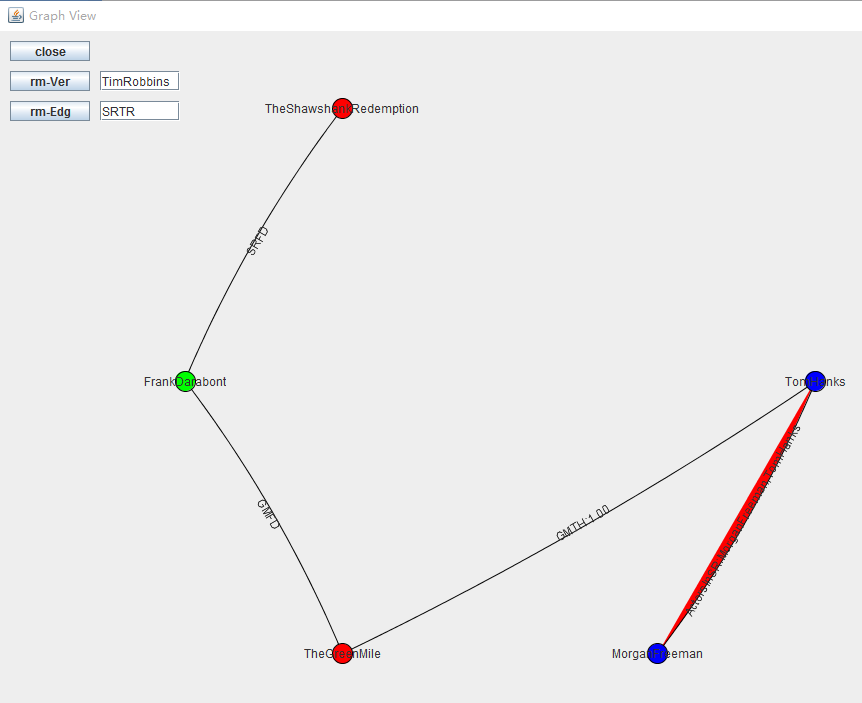
①如果在右边框中填入图中没有的边，点击rm-Edg，框中显示”error:not found!”



②删除SRTR之前:



删除SRTR之后：

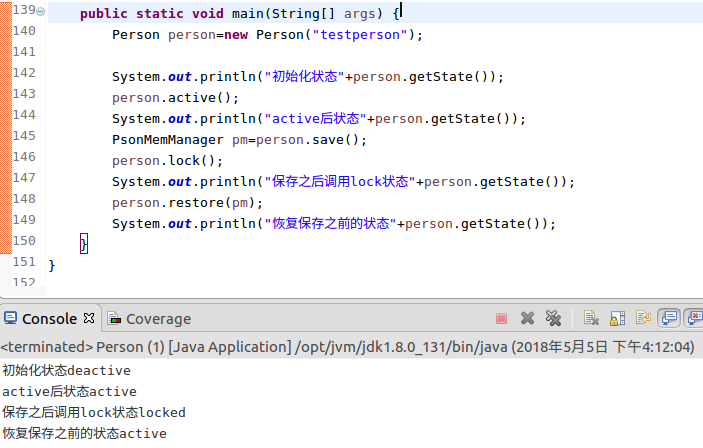


## 3.8设计模式应用

### 3.8.1 使用State/Memento模式进行Vertex的状态管理（选做）

ｓtatememento包中有四个类，分别是四种节点Computer,Router,Server和Person的备忘录，里面可以存储节点的状态。Mementomanager包中有四个类，分别是四种节点的备忘录管理器。在这四种节点中，有一个private域state表示此时顶点的状态，有save()方法可以产生一个新的备忘录，并且产生一个新的备忘录管理器，将备忘录传给备忘录管理器，返回备忘录管理器。restore()方法以备忘录管理器作为参数，将现在的状态重置为之前传入备忘录中的状态。getstate()方法可以获得Vertex目前的状态。

测试结果：



### 3.8.2使用factory method模式构造Vertex对象

VertexFactory提供抽象方法createVertex()，子类的工厂方法继承父类，实现createVertex()方法。用label参数新建一个节点,用，args参数作为fillVertexInfo的参数。最终返回一个相应类型的节点。

### 3.8.3使用factory method模式构造Edge对象

的EdgeFactory提供抽象方法createEdge()，子类的工厂方法继承父类，实现createEdge()方法，。用label和weight方法创建一个新的Edge，然后调用addVertices()方法，最后返回edge。

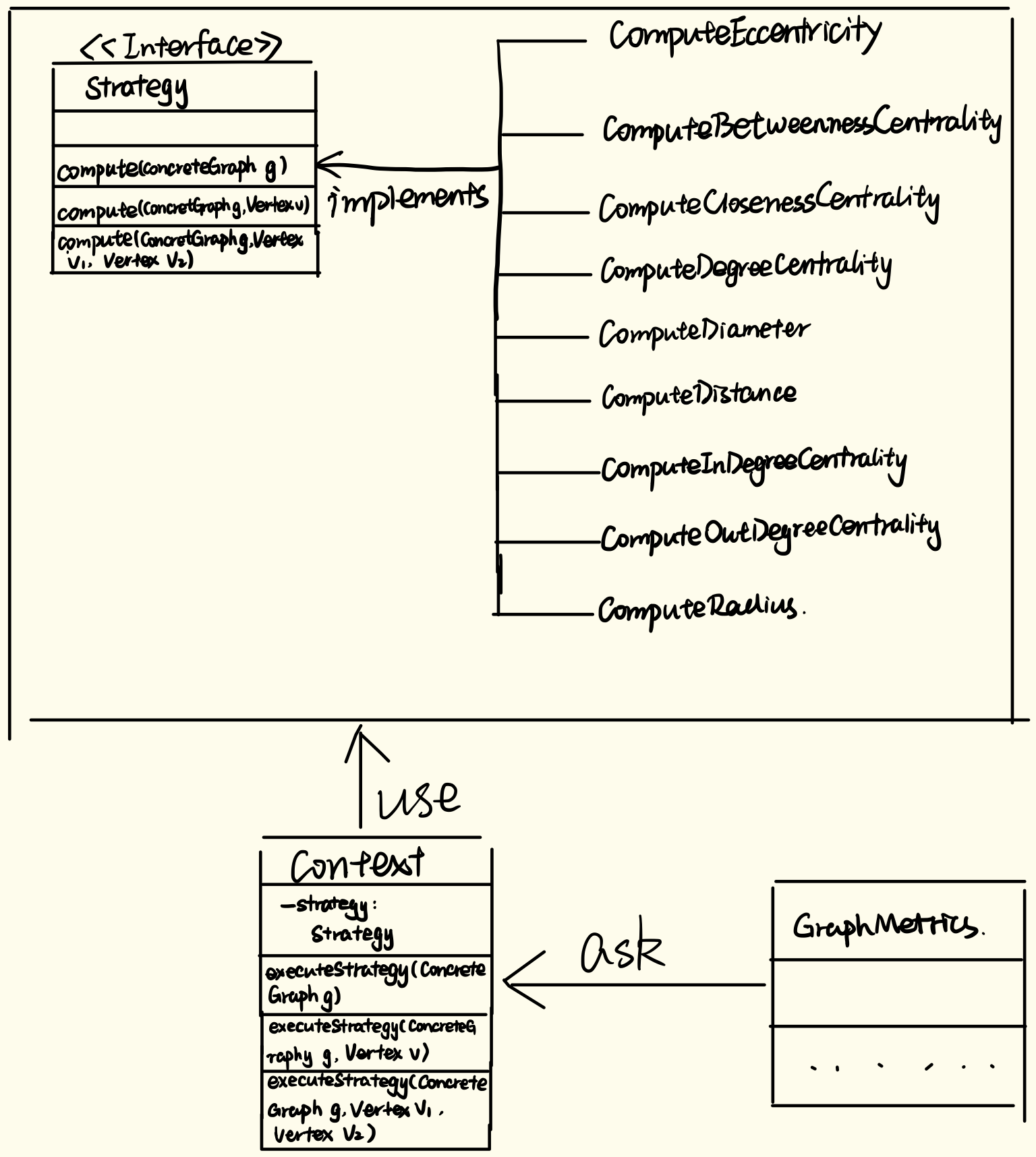
### 3.8.4使用abstract factory或builder模式构造Graph对象

GraphFactory的createGraph()通过读入文件第一行字符串确定图的类型，然后调用相应的工厂方法，产生相应的图。

GraphPoetFactory,SocialNetworkFactory,NetworkTopologyFactory,MovieGraphFactory是各个子类图的工厂方法，通过正则匹配获取文件里的有效信息，并且用这些有效信息建立相应的图并返回。

### 3.8.5使用Strategy模式调用centrality度量算法

在子类中实现接口，以达到不同的功能，然后用GraphMetrics来调用Context中的方法，以达到调用不同函数的效果。基本结构如下图。



## 3.9 图操作指令的输入和处理（选做）

使用façade设计模式，完善ParseCommandHelper类

先用正则表达式识别匹配指令，然后从中提取有效信息，并作出相应的操作。



关于图的指令的详细信息，见README.md

在工程中也有示例指令，见examplecmds.txt

## 3.10 应用设计与开发

关于图的指令的详细信息，见README.md

在每个应用中我的功能的实现都是依赖指令来处理的。下面演示具体测试过程。

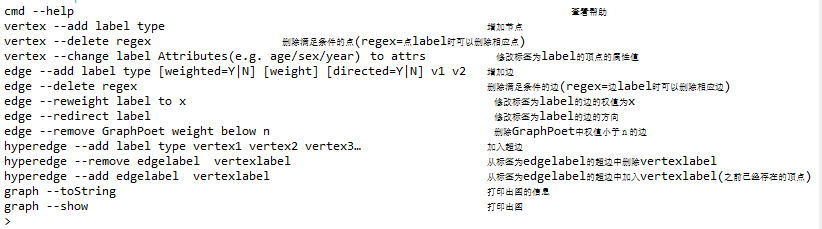
### 3.10.1单词网络GraphPoet

首先要求用户输入测试文件的路径，比如：

（在输入指令的时候，可以用空格，但是不要用任何引号，大小写敏感。）

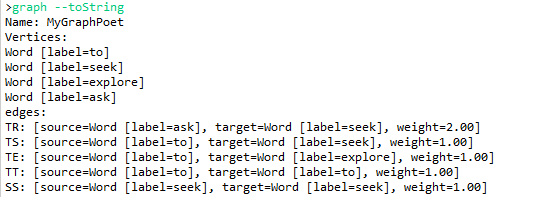


输入”cmd --help”可以查看命令详细信息：



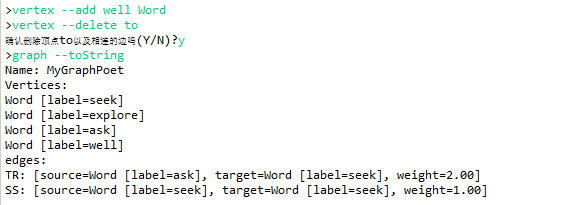
以下是各个指令的演示结果：

①graph --toString 打印出图的信息

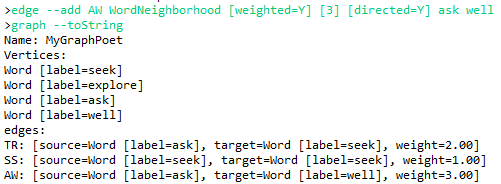


②加入和删除：

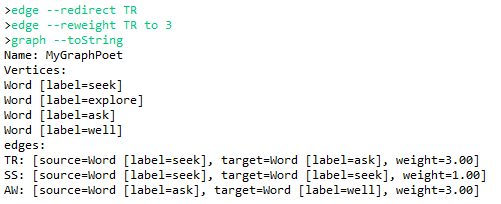
如图显示了用户要加入一个label为”well”,类型为”Word”的顶点，然后要删除label为”to”的顶点以及与其相关联的边。并且，在删除之前向用户确认是否要删除顶点”to”以及与之相关联的边。然后输入打印图信息的指令，打印信息同上一条相比息符合预期：



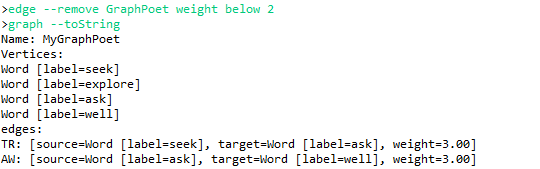
③加入一个类型为WordNeighborhood类型，label为”AW”的从”ask”到”well”的权重为3的边，打印信息同上一条相比息符合预期：



④将label为”TR”的边转换方向之后重置权值为3之后，打印信息同上一条相比息符合预期：



⑤用户确定要删除权值小于2的边，删除后打印信息同上一条相比息符合预期：



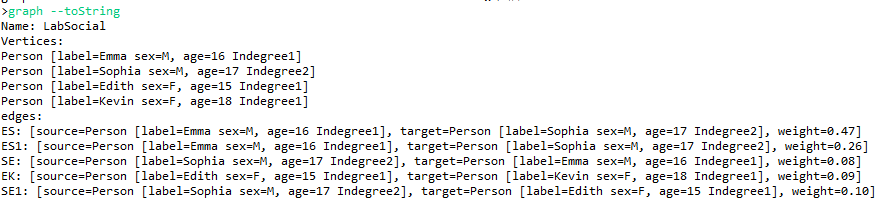
⑥最后，用GUI表现图的结构，与上一条graph --toString命令所展现的图的信息相符合。



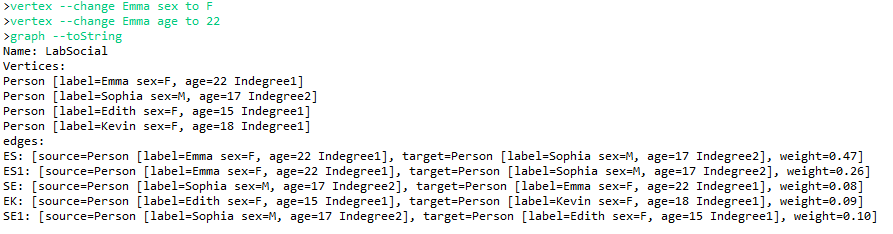
### 3.10.2微博社交网络SocialNetwork

APP总体实现与前一个相同，有一些细微的不同的地方如下：

①图初始化后显示信息如下：

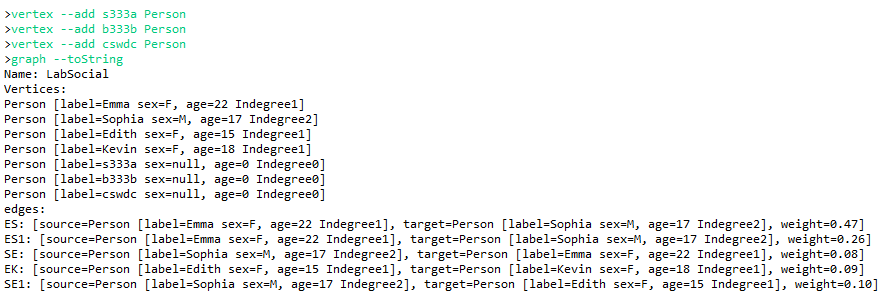


②修改label为”Emma”的结点的属性：把性别改为F，年龄改为22,与上一条相比，结果符合预期：

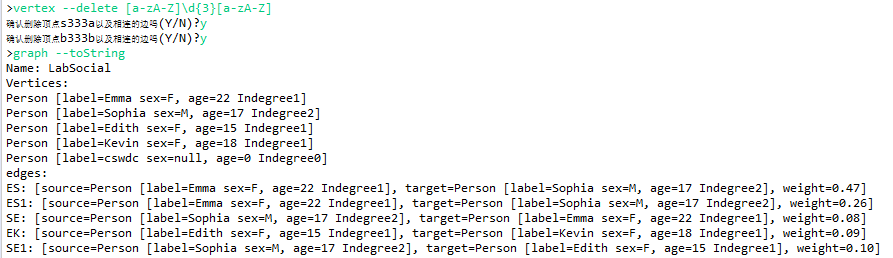


③测试正则表达式删除：

先向图中加入三个结点，有两个满足正则表达式[a-zA-Z]\d{3}[a-zA-Z]，加入后结果如下 ：



然后用指令删除后结果如下，符合预期：（并且在删除时向用户确认是否删除）

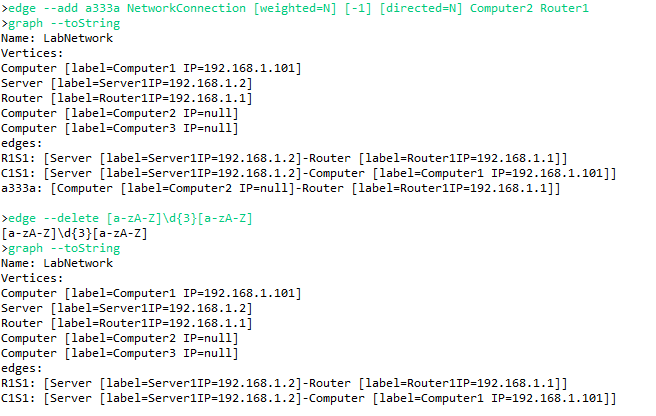


最后在计算图的各种度的时候，我们的测试可以得到

### 3.10.3网络拓扑图NetworkTopology

实现与前两个类似，先让用户输入测试文件路径，然后初始化图，然后用户可以用指令对图进行操作。

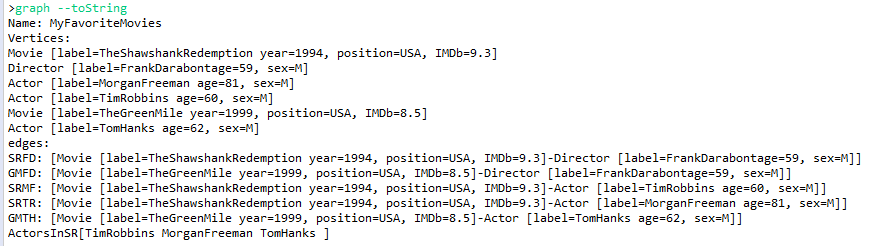
比如，依靠正则表达式去删除边的情况。



### 3.10.4电影网络MovieGraph

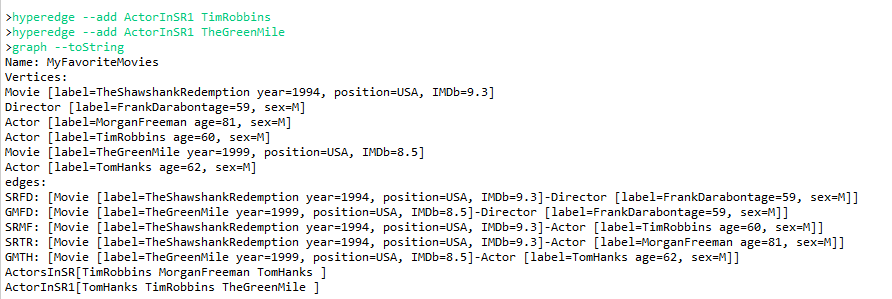
实现与前两个类似，先让用户输入测试文件路径，然后初始化图，然后用户可以用指令对图进行操作。

初始化：

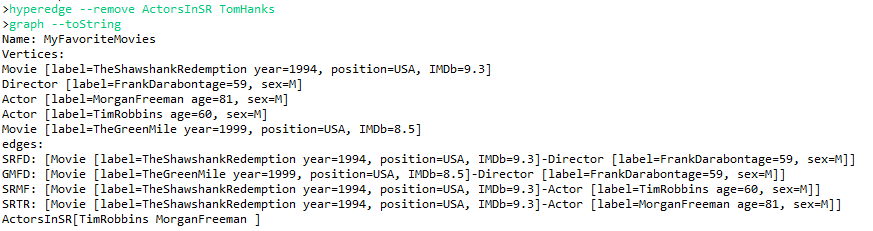


通过指令修改：

加边加点：



删点：

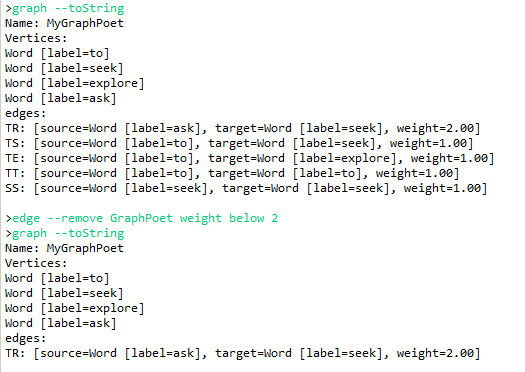


## 3.11应对四个应用面临的新变化（任选两个）

### 3.11.1单词网络GraphPoet

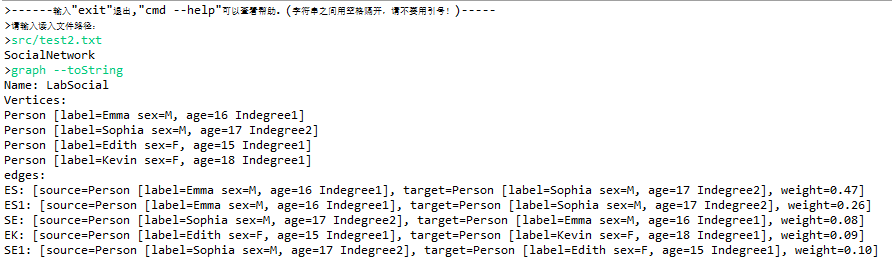
凡是权重小于n的有向边，不应该出现在图中，n为用户确定的参数。可以通过用户输入的参数确定要从图中删除哪些边。

测试结果：



### 3.11.2微博社交网络SocialNetwork

为结点添加属性indegree centrality，有多少有向边连接到改结点，改结点的indegree centrality就是多少，并且该属性随toString()方法打印出来。



# 4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 |
| 4.10-4.20 | 空闲时间 | 完成基本框架和实验前半部分内容 | 基本完成 |
| 4.20-4.30 | 空闲时间 | 完成实验后半部分内容 | 基本完成 |
| 5.1-5.６ | 空闲时间 | 修改需求，完善报告 | 基本完成 |

# 5实验过程中遇到的困难与解决途径

遇到的困难是对各种设计模式不是很熟悉，然后通过查阅资料，阅读类似的例子使困难得到解决。对第三方库jung的接口不是很熟悉，通过阅读官方文档对其有所了解然后完成任务。通过查找wiki来了解要计算的各种度的定义和注意事项。通过查阅资料了解了GUI的相关信息。

# 6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. 重新思考Lab2中的问题：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？本实验设计的ADT在四个图应用场景下使用，你是否体会到复用的好处？

面向ADT编程提高了代码的可复用性，复用使得代码更易于理解和维护。

1. 重新思考Lab2中的问题：为ADT撰写复杂的specification, invariants, RI, AF，时刻注意ADT是否有rep exposure，这些工作的意义是什么？你是否愿意在以后的编程中坚持这么做？

这些工作对以后代码的维护提供了方便，并且减少了bug发生的概率，看似降低了效率，但是长远来看还是益处更多的。我还是比较愿意在以后的编程中坚持这么做的。

1. 之前你将别人提供的API用于自己的程序开发中，本次实验你尝试着开发给别人使用的API，是否能够体会到其中的难处和乐趣？

开发给别人使用的API确实很有难度，需要考虑全面，要防止表示暴露，要选择合适的设计模式，等等。但是看到这样设计的代码的质量得到提高，还是比较有乐趣的。

1. 在编程中使用设计模式，增加了很多类，但在复用和可维护性方面带来了收益。你如何看待设计模式？

设计模式提高了工程的可维护性和重用性，虽然类变多了，但是工程变得更加有条理，更加易于维护了。

1. 你之前在使用其他软件时，应该体会过输入各种命令向系统发出指令。本次实验你开发了一系列命令行指令，使用语法和正则表达式去解析它们并映射到对后台程序的调用。你对语法驱动编程有何感受？

语法驱动编程比较灵活方便，但是我在处理的时候有点冗杂，希望以后能够学到更好的处理方式。

1. 关于本实验的工作量、难度、deadline。

工作量有点大，难度适中，deadline也是合适的，但是实验文档有些地方不是很妥当。

1. 到目前为止你对《软件构造》课程的评价。

这门课我学到很多东西，第一次用Java，学习面向对象，使用设计模式，去了解和使用各种工具，提高代码的可复用性和可维护性，了解了java里面很多的细节的东西和很强大的面向对象的设计思维。从老师对于实验的设计也可以看出有很用心很巧妙的idea，但是还是存在一些漏洞，当然，瑕不掩瑜。