

**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 | 冯运 |
| 学号 | 1160300524 |
| 班号 | 1603005 |
| 电子邮件 | 294889365@qq.com |
| 手机号码 | 18846188605 |

**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)

# 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
6. API 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），学生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<V,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）。

# 实验环境配置

实验需要配置elclipse,但是我之前已经习惯于使用jetbrain系列的开发ide,所以我这次使用了IDEA，同时下载安装了jdk，版本号是jdk1.8.0\_101，将jdk的安装路径加入了系统的环境变量。在IDEA里面下载安装了junit插件,配置好了java的测试环境。

Lab3的仓库url:

https://github.com/ComputerScienceHIT/Lab3-1160300524.git

# 实验过程

请仔细对照实验手册，针对每一项任务，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！）。

## 待开发的四个应用场景

四个应用场景大体比较相似，所有的需求都是基于一个图来进行的。而多模图可以兼容单模图。四个应用场景的边的类型有有向边，无向边，超边这几种。

这几种边的操作大体一致，可以将共性的操作放在同一个抽象父类中。然后对于超边，需要一些特殊的操作，可以将父类的方法重写，来实现独特的功能。然后超边中还有一些独特的方法，可以单独定义和实现。

点的类型比较多，而且点的内部属性各不相同，只能将公共属性比如labe放在公共父类中，然后将独有的属性放在子类中单独实现。

## 基于语法的图数据输入

我是将图的实现分了三个模块，第一个模块是graph模块，负责判断图的类型，然后将具体的工作派发给具体某种图的子类。第二个模块是vertex模块，这个模块负责点部分信息的解析和读入，然后将具体的点的类型的信息派发给具体某种类型的子类中。第三个模块是edge模块，这个模块负责边部分的信息解析和读入，然后将具体的边类型的信息派发给具体某种边的子类中，由子类负责创建相应的边对象。

这样各个模块的分工比较清晰，便于开发和维护

## 面向复用的设计：Graph<L, E>

addVertex

boolean addVertex(Vertex vertex)

向图中添加一个Vertex对象

参数:

vertex - Vertex对象,表示一个点

返回:

true: 向图中成功的添加了这个点 false: 图中已经包含了这个点

removeVertex

boolean removeVertex(Vertex vertex)

从图中移除这个点，任何包含这个点的边都会同时被删除

参数:

vertex - 需要移除的点对象

返回:

true ：移除成功 false: 图中不包含这个点，删除失败

vertices

java.util.Set<Vertex> vertices()

得到图中的所有点对象

返回:

图中所有点的一个集合

sources

java.util.Map<Vertex,java.util.List<java.lang.Double>> sources(Vertex target)

得到以传入点为终点的所有边的起点，以及所有边的权重值

参数:

target - 一个点的对象

返回:

一个点和一个权重值的数组形成的map, 点是边的起点，边的终点就是传入的点

targets

java.util.Map<Vertex,java.util.List<java.lang.Double>> targets(Vertex source)

Get the target vertices with directed edges from a source vertex and the weights of those edges. 得到以传入点为起点的所有边的终点，以及所有边的权重值

参数:

source - 一个点的对象

返回:

一个点和一个权重值的数组形成的map, 点是边的终点，边的起点就是传入的点

addEdge

boolean addEdge(Edge edge)

向图中添加一条边

参数:

edge - 一个边的对象，表示一条边

返回:

true: 图中成功添加了这条边 false: 图中已经有了这条边，添加失败

removeEdge

boolean removeEdge(Edge edge)

从图中移除指定的边对象

参数:

edge - 一个边的对象，表示一条边

返回:

true: 成功的从图中删除了这条边 false: 图中没有传入的边对象，删除失败

edges

java.util.Set<Edge> edges()

得到图中的所有边对象

返回:

图中所有边对象的集合

## 面向复用的设计：Vertex

具体的函数

public abstract void fillVertexInfo(java.lang.String[] args)

给该顶点添加附加的信息

参数:

args - 附加的数据项,一个String表示一个信息单元

addInEdge

public boolean addInEdge(Edge inEdge)

给这个点添加入边,添加的是一个Edge对象 如果这条边已经存在, 或者边的的终点不是这个点,都会返回false 如果添加成功,则会返回true

参数:

inEdge - 一个Edge对象,表示一条边

返回:

true: 这条入边添加成功, false: 这条入边添加失败

addOutEdge

public boolean addOutEdge(Edge outEdge)

给这个点添加出边,添加的是一个Edge对象 如果这条边已经存在, 或者边的的起点不是这个点,都会返回false 如果添加成功,则会返回true

参数:

outEdge - 一个Edge对象,表示一条边

返回:

true: 这条入边添加成功, false: 这条入边添加失败

removeEdge

public boolean removeEdge(Edge edge)

在这个点中删除某条边的记录,入边和出边都会删除 如果点的入边和出边中都不存在这条记录,则删除失败,返回false 如果删除成功,则返回true

参数:

edge - 一个Edge对象,表示一条边

返回:

true: 这条边删除成功, false: 这条边删除失败

getInEdges

public java.util.Set<Edge> getInEdges()

返回这个点包含的所有入边

返回:

这个点包含的所有入边

getOutEdges

public java.util.Set<Edge> getOutEdges()

返回这个点包含的所有出边

返回:

这个点包含的所有出边

getLabel

public java.lang.String getLabel()

获得这个点的 唯一标识 label 的值 这是一个 observer函数

返回:

String对象 label的值

toString

public java.lang.String toString()

覆盖:

toString 在类中 java.lang.Object

equals

public boolean equals(java.lang.Object obj)

覆盖:

equals 在类中 java.lang.Object

hashCode

public int hashCode()

覆盖:

hashCode 在类中 java.lang.Object

## 面向复用的设计：Edge

getLabel

public java.lang.String getLabel()

得到边的label属性值 是一个 observer函数

返回:

label的值, 是String类型

getWeight

public double getWeight()

得到边的权重值 是一个observer函数

返回:

weight的值，是double类型

addVertices

public abstract boolean addVertices(java.util.List<Vertex> vertices)

向边中添加点 如果这是一条普通边，List的长度应该为2 如果是有向边，第一个点是有向边的起点，第二个点是有向边的终点 如果是无向边，两个点没有顺序 如果是一条超边，List的长度 大于 2

参数:

vertices - list 形式的点的集合

返回:

true :添加成功 false:添加失败

vertices

public java.util.Set<Vertex> vertices()

获取边中的所有顶点

返回:

边中所有顶点的一个集合

sourceVertices

public abstract java.util.Set<Vertex> sourceVertices()

获取边的所有起点 如果是有向边，起点只有一个 如果是无向边，返回边的所有顶点

返回:

边的所有起点

targetVertices

public abstract java.util.Set<Vertex> targetVertices()

返回边的所有终点 如果是有向边，终点只有一个 如果是无向边，返回边的所有终点

返回:

边的所有终点

setWeight

public double setWeight(double weight)

设置或修改边的权重

参数:

weight - 权重的新值

返回:

返回边的权重的旧值

toString

public java.lang.String toString()

将边的信息转换为字符串

覆盖:

toString 在类中 java.lang.Object

返回:

转换成的字符串

equals

public boolean equals(java.lang.Object obj)

覆盖:

equals 在类中 java.lang.Object

hashCode

public int hashCode()

覆盖:

hashCode 在类中 java.lang.Object

## 可复用API设计

degreeCentrality

public static double degreeCentrality(Graph g)

计算整个图的degree, 计算公式是图中度最大的值-所有点的度，然后累加 ，再除以图中点的个数

参数:

g - 传入的图对象

返回:

返回整个图的degree

degreeCentrality

public static double degreeCentrality(Graph g,

Vertex v)

获得一个无向图中点的 点度中心性 如果图中没有这个点,将会返回0 如果传入的图是有向图，且点的入度和出度不相同 ，会throw UnsupportedOperationException

参数:

g - 传入所求点所在的图

v - 需要求度的点

返回:

所求点的度，如果传入的图中没有这个点，就会返回0

inDegreeCentrality

public static double inDegreeCentrality(Graph g,

Vertex v)

获得一个有向图中点的 点入度中心性 如果图中没有这个点,将会返回0

参数:

g - 传入所求点所在的图

v - 需要求度的点

返回:

所求点的入度值，如果传入的图中没有这个点，就会返回0

outDegreeCentrality

public static double outDegreeCentrality(Graph g,

Vertex v)

获得一个无向图中点的 点出度中心性 如果图中没有这个点,将会返回0

参数:

g - 传入所求点所在的图

v - 需要求度的点

返回:

所求点的出度值，如果传入的图中没有这个点，就会返回0

closenessCentrality

public static double closenessCentrality(Graph g,

Vertex v)

用于计算一个点到图中其他点的开销 因数 closenessCentrality

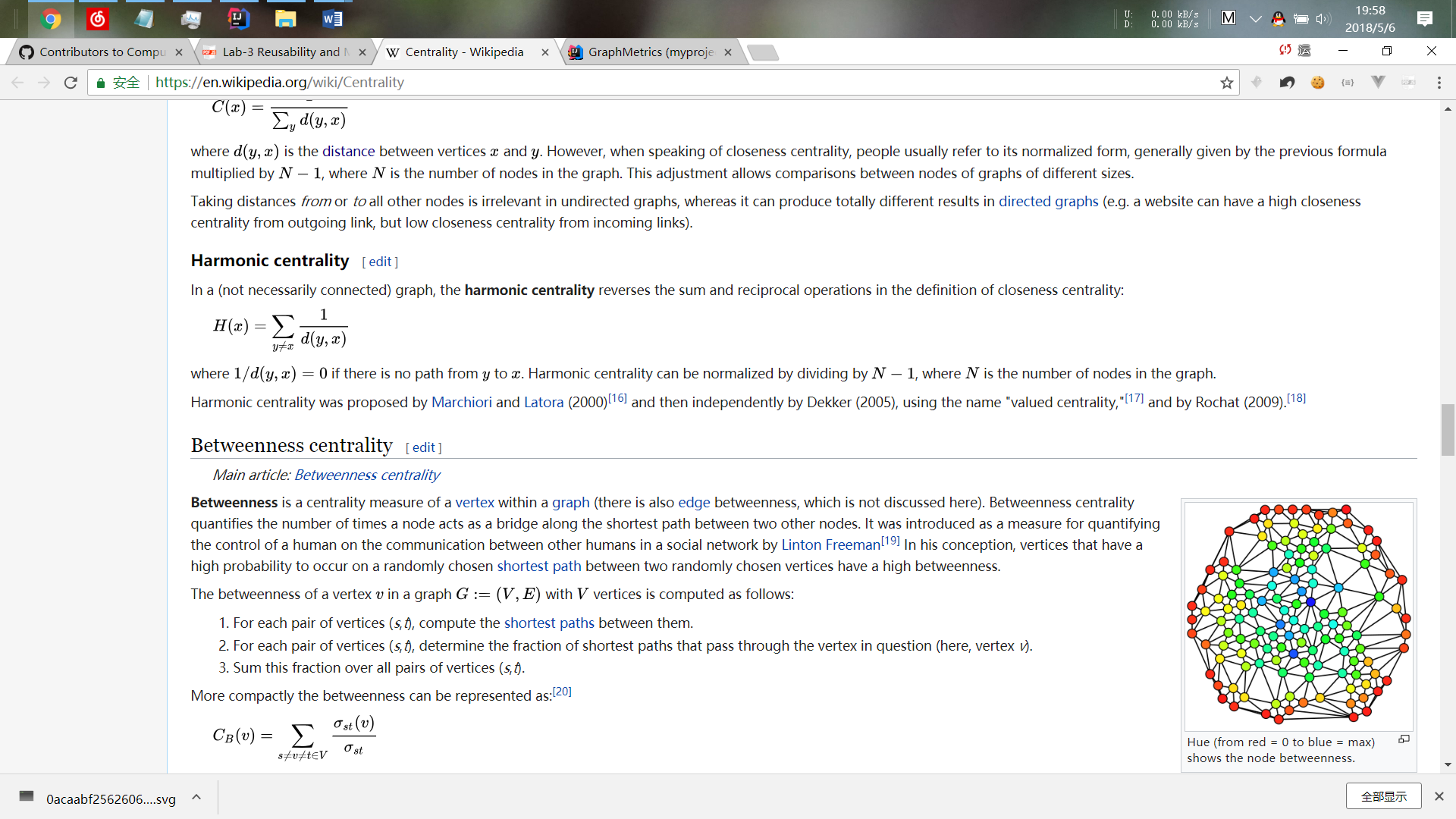
参数:

g - 图对象

v - 点对象

返回:

closenessCentrality 的值



主体使用floyd算法，求出该点到其他所有点的最短路径，然后根据上图的公式就可以求得值

betweennessCentrality

public static double betweennessCentrality(Graph g,

Vertex v)

求某个点的 BetweennessCentrality 值，这个值表示这个点在图中促进其他节点进行通信的能力

参数:

g - 图对象

v - 点对象

返回:

BetweennessCentrality 的值

这个方法使用的是jung第三方库提供的函数，将自己的图转换为jung提供的图， 然后直接调用jung中的相应函数就可以求得值。

distance

public static double distance(Graph g,

Vertex start,

Vertex end)

用于求图中两个点之间的最短路径的边的个数

参数:

g - 传入目标的图对象

start - 起始的顶点

end - 终止的顶点

返回:

两个点在图g中的最短路径的边的个数

实现思路是先使用Dijistra算法求出一条最短路径，然后得到最短路径上边的个数，就是distance的值

eccentricity

public static double eccentricity(Graph g,

Vertex v)

计算给点在给定图中的eccentricity值(这个点到其他所有点的distance的最大值)

参数:

g - 给定的图

v - 给定的点

返回:

点v在图g中的eccentricity值

radius

public static double radius(Graph g)

求给定图的radius值

参数:

g - 给定的图对象

返回:

图g的radius值

diameter

public static double diameter(Graph g)

求给定图的diameter值

参数:

g - 给定的图对象

返回:

图g的diameter值

## 图的可视化：第三方API的复用（选做）

visualize

public static void visualize(Graph g)

这个函数用于将传入的图可视化，图形化的输出

参数:

g - 需要图形化的参数

transferGraph

public static edu.uci.ics.jung.graph.AbstractGraph<Vertex,Edge> transferGraph(Graph g)

将自己设计的图的ADT转换成jung2 提供的图结构，从而能够调用Jung2的可视化api

参数:

g - 需要转换的自己的图

返回:

jung2提供的图的结构

这个任务需要使用adapter的设计模式，将自己的图的ADT使用adapter适配jung2提供的图可视化API

我的 transgerGraph就是实现这个功能的函数。

CircleLayout<Vertex, Edge> layout = *new* CircleLayout<>(*transferGraph*(g));  
layout.setSize(*new* Dimension(300, 300)); *// sets the initial size of the space // The BasicVisualizationServer<V,E> is parameterized by the edge types*BasicVisualizationServer<Vertex, Edge> vv = *new* BasicVisualizationServer<>(layout);  
vv.setPreferredSize(*new* Dimension(600, 600)); *//Sets the viewing area size*JFrame frame = *new* JFrame("Simple Graph View");  
*float* dash[] = {10.0f};  
*final Stroke* edgeStroke = *new* BasicStroke(1.0f, BasicStroke.CAP\_BUTT,  
 BasicStroke.JOIN\_MITER, 10.0f, dash, 0.0f);  
vv.getRenderContext().setVertexFillPaintTransformer(Vertex -> Color.BLUE);  
vv.getRenderContext().setEdgeStrokeTransformer(Edge -> edgeStroke);  
vv.getRenderContext().setVertexLabelTransformer(*new* ToStringLabeller<>());  
vv.getRenderContext().setEdgeLabelTransformer(*new* ToStringLabeller<>());  
vv.getRenderer().getVertexLabelRenderer().setPosition(*Renderer*.*VertexLabel*.Position.CNTR);  
  
frame.setDefaultCloseOperation(*WindowConstants*.HIDE\_ON\_CLOSE);  
frame.getContentPane().add(vv);  
frame.pack();  
frame.setVisible(*true*);

这些代码是官方文档上的使用示例，将自己转换好的jung2的graph对象传入，即可实现图的可视化。

## 设计模式应用

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

### 使用factory method模式构造Vertex对象

ActorVertexFactory

ComputerVertexFactory

DirectorVertexFactory

MovieVertexFactory

PersonVertexFactory

RouterVertexFactory

ServerVertexFactory

VertexFactory

WirelessRouterFactory

WordVertexFactory

点的Factory有以上这些类，将一个点的所有信息传入到父类 VertexFactory中，然后由它去根据点的具体类型，调用具体类型点的工厂函数，最后返回一个生成好的Vertex对象。

每个点独特的属性赋值都可以通过具体的点工厂函数实现。

### 使用factory method模式构造Edge对象

commentEdgeFactory

EdgeFactory

forwardEdgeFactory

friendEdgeFactory

MovieActorEdgeFactory

MovieDirectorEdgeFactory

networkEdgeFactory

poetEdgeFactory

SameMovieHyperEdgeFactory

边的Factory有以上这些类，将一个边的所有信息传入到父类 EdgeFactory中，然后由它去根据边的具体类型，调用具体类型边的工厂函数，最后返回一个生成好的Edge对象。

每个边独特的属性赋值都可以通过具体的边工厂函数实现。

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

GraphFactory

GraphMovieFactory

GraphPoetFactory

GraphSocialFactory

GraphTopologyFactory

将一个图传入GraphFactory中，然后它根据图的具体类型，由具体的子类生成一个图对象，这里的图工厂就是将点和边的工厂函数进行组合，从而产生一个具体的图对象。

### 使用Strategy模式调用centrality度量算法

定义centrality接口Centrality，对每一种计算centrality的方法只需实现接口中的calculate方法即可。计算时实例化一个具体的对象，并将其传入GraphMetrics.vertexCentrality()中即可。

### 使用Composite模式设计超边对象（选做）

### 使用decorator模式构造不同特征的Edge对象（选做）

### 使用其他设计模式（选做）

## 图操作指令的输入和处理（选做）

使用façade设计模式，完善ParseCommandHelper类

单独设计了一个抽象类，用于集合对于点，普通边，超边，图的各种指令操作

下面是这个抽象类的所有方法

add

public abstract void add(java.lang.String[] args)

向图中添加输入的新对象

参数:

args - 用户输入的新对象的属性

delete

public abstract void delete(java.lang.String[] args)

从图中删除用户指定的对象

参数:

args - 用户输入的指定对象的属性

update

public abstract void update(java.lang.String[] args)

更新图中指定对象的属性

参数:

args - 用户输入的指定对象的属性

show

public abstract void show(java.lang.String[] args)

展示图中指定对象的属性

参数:

args - 用户输入的指定对象属性

confirm

public static boolean confirm()

在删除指定对象时，向用户确认是否删除

返回:

true: 用户确认删除 false:用户撤销了删除的操作

然后由主类GraphVisualizationHelper负责对外的façade

处理各种各样的操作。

由各个具体的子类实现具体的指令功能

## 应用设计与开发

利用上述设计和实现的ADT，实现手册里要求的各项功能。

我将所有的功能都放入了命令行，具体功能对应的命令如下所示

### 单词网络GraphPoet

要求的功能：

读取包含图数据的磁盘文件，验证其合法性，并解析为图结构；

在GUI上可视化展示图结构

增加节点、删除节点，修改节点的信息（Label、其他属性）；

增加边、删除边、修改边的label、权重、方向；

向超边中增加节点、从超边中去除节点；

选定某一节点，计算其三种中心度、eccentricity；

如果为有向图中的节点，

计算indegree和outdegree中心度；

计算图degree centrality、radius和diameter；

选定两个节点，计算它们之间的distance

具体功能对应的命令：

点的命令

vertex --add label type 向图中添加名为label，类型为type的点

vertex --delete regex 删除所有label符合regex规则的点

vertex --update label

* label= 改变点的label值
* argument= 改变点的argument值，即点的属性

vertex --show

* eccentricity 求点的eccentricity值
* degree 求点的degree值
* indegree 求点的indegree值
* outdegree求点的outdegree值
* closenessCentrality 求点的closenessCentrality值
* betweennessCentrality 求点的 betweennessCentrality值

边的命令

edge --add label type [weighted=Y|N] [weight] [directed=Y|N] v1, v2

edge --delete regex 删除所有label符合regex的边

超边的命令

hyperedge --add

* label type vertex1, ..., vertexn 向图中添加一个新的超边
* label vertex1, ..., vertexn 向label 为 label的超边添加点

hyperedge --delete label regex 删除超边中label符合regex规则的点

图的命令

graph --show

* degreeCentrality 输出图的degreeCentrality值
* radius 输出图的radius值
* diameter 输出图的diameter值
* visiable 将图的可视化结果输出
* distance="vertex1""vertex2" 输出vertex1 与 vertex2 在图中的 distance

### 微博社交网络SocialNetwork

具体的实现指令同上

### 网络拓扑图NetworkTopology

具体的实现指令同上

### 电影网络MovieGraph

具体的实现指令同上

## 应对四个应用面临的新变化（任选两个）

### 单词网络GraphPoet

### 微博社交网络SocialNetwork

给它对应的点对象中添加了 weight属性

然后提供了它对应的 observer 方法

在每次添加和删除边时更新它的值

更改的成本：

只需要在它自己的点的类中添加相应的属性即可，修改成本不大

### 网络拓扑图NetworkTopology

直接添加一个新的wireless点类，然后继承 Vertex类，里面具体的实现都和其他network类相同。

然后在工厂函数中添加对这个新类型的处理

更改的成本很小

### 电影网络MovieGraph

# 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 |
| 4．2--4．11 | ----------------- | ------------------------------------------------ | 完成Edge抽象类 |
| 4.11 | ----------------- | ------------------------------------------------ | 完成Vertex抽象类 |
| 4.11—4.14 | ----------------- | ------------------------------------------------ | 完成Graph抽象类 |

# 实验过程中遇到的困难与解决途径

到处都是困难，从头到尾，都异常困难

解决途径，花了大量时间慢慢啃

# 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？
2. 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？
3. 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？
4. 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？
5. 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？
6. 关于本实验的工作量、难度、deadline。

即使时间这么久，依然感觉时间不够用。

1. 到目前为止你对《软件构造》课程的评价。

这个lab3太太太变态了，让我萌生了转专业退学的想法。