

**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 | 王沐贤 |
| 学号 | 1160300922 |
| 班号 | 1636101 |
| 电子邮件 | Wangmuxian6319@163.com |
| 手机号码 | 17310500759 |

**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：  子类型、泛型、多态、重写、重载  继承、代理、组合  常见的 OO 设计模式  语法驱动的编程、正则表达式  基于状态的编程  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<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）

# 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 Lab3仓库的URL地址（Lab3-学号）。

https://github.com/ComputerScienceHIT/Lab3-1160300922

# 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

四个场景都是基于图结构的建立。Graphpoet是单重有向图、带权图、 单模图。Socialnetwork是多重有向图、带权图、单模图。Networktopology是简单图、带权图、多模图。Moviegraph是单重图、多模图。特殊之处在于，graphpoet带权，socialnetwork有多种类型边，networktopology对点之间的链接有限制，而moviegraph有超边

## 基于语法的图数据输入

这里存在两个内容：用正则表达式对txt进行解析以及初始图的构建。这里仅解释文字解析的部分。整个txt由6-7部分组成。第一部分是图的类型，直接读入引号间的数据即可。第二部分是图的名称，方法同上。第三部分是点的类型，这里的读入是判断之后读入的点类型是否有出入。第四部分是点的介绍，我将这部分拆成两个表达式：前面的两对引号中第一对中的内容是该点的label，后面的是该点的类型。之后的属性单独使用一个表达式。在属性中，存在多属性的点类型将被进一步拆分成某个独立的属性。第五部分边的类型与第三部分操作一致。第六部分是边的内容。基本操作与第四部分相同，但是存在更多的属性，最后一个属性“y/n”可以用来检验边类型的正确性。而构建可能出现的第七部分是moviegraph中特有的超边。属性的内容类似于表示一个集合。

## 面向复用的设计：Graph<L, E>

Graph是该场景的总接口。包含9个抽象函数

 public static <L,E> Graph<L,E> empty() 构造一张图的空实例 

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

public 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。 

public Set<L> vertices() 返回图的节点集合 

public Map<L, Double> sources(L target) 与Lab2中Graph接口的同名操作含义相同；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。 

public Map<L, 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。 

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

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

public Set<E> edges() 返回边的集合（包括超边）

之后的concretegraph是该接口下的实例。有一个private的字符串表示其名称，在其中实现接口的各个函数。这些函数的操作与lab2中graph基本一致。不过，我的equals函数使用的是通过比较label的一致性方法得出的结论，在该实验要求下避免了重写hashcode可能带来的麻烦。之后的各个图的实例继承自concretegraph，其中包含改变有权边的两个图：graphpoet和socialnetwork还多加一组函数负责权重的增加，改变和删除。

## 面向复用的设计：Vertex

Vertex是总抽象类，后面的各个点类型继承了这个类做成实例。有一个private的label，以及独有的构造器。

Vertex(String label) 构造函数 

abstract void fillVertexInfo (String[] args)

public String getLabel() 返回节点 label 的取值 

toString() 如果针对不同子类型有不同的 string 表示，则需要在各 个子类型中 override 该函数。 

equals() 类似于 toString() 

hashCode() 类似于 toString()

## 面向复用的设计：Edge

与vertex类似，edge是所有的总抽象类，下面分成有向边和无向边以及超边继承之。而各个实际边通过继承以上3个实例父类对各个边实现要求。

函数如下

Edge(String label, double weight) 构造函数 

abstract public boolean addVertices(List<Vertex> vertices) 如果是超边，vertices.size()>=2，该函数添加 vertices 中的所有 节点到该超边；如果是有向边，vertices.size()==2，该操作将 vertices 中的第一个元素作为 source，将第二个元素作为 target； 如果是无向边，vertices.size()==2，无需考虑次序； 

boolean containVertex(Vertex v) 该边中是否包含指定的点 v 

Set<Vertex> vertices() 边包含的点集 

abstract Set<Vertex> sourceVertices() 返回该边的源节点集合 

abstract Set<Vertex> targetVertices() 返回该边的目标节点集合 

toString() 不同类型的边，其字符串表示应有不同，需表征出端点、 方向、权重、label，具体形式自行设计，在子类型中 override； 

equals() 类似于 toString() 

hashCode() 类似于 toString()

edge的属性中除了label还有weight，二者都是private的，为保证其安全性的同时可以被改变。各个实例仅需在构造器中继承其父类即可。同时进行实例规范性的判断。

## 可复用API设计

## 图的可视化：第三方API的复用（选做）

## 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

### 使用factory method模式构造Edge对象

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

### 使用Strategy模式调用centrality度量算法

### 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）

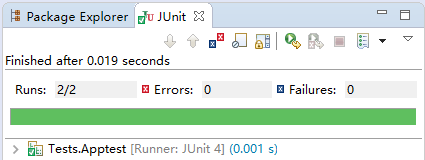
## 图操作指令的输入和处理（选做）

使用façade设计模式，完善ParseCommandHelper类

## 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。

这里4个图的app采用了相同的构建方法，所以我将图的构建函数直接写在了prasecommandhelper中。第一部分和第二部分调用建图函数，第三第四部分调用造点函数，第五第六第七部分调用造边函数。在调用过程中进行图的正确性的测试。测试结果如图



### 单词网络GraphPoet

### 微博社交网络SocialNetwork

### 网络拓扑图NetworkTopology

### 电影网络MovieGraph

## 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

### 网络拓扑图NetworkTopology

### 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 |
| 20180404 | 20180411 | 完成graph | 按时完成 |
| 20180411 | 20180418 | 完成prase解析 | 拖延一周 |
| 20180425 | 20180506 | App及中心度 | 未能彻底完成 |

# 实验过程中遇到的困难与解决途径

Set存储方式hashcode变化导致的存储偏移。解决方法是使用list代替。

# 实验过程中收获的经验、教训、感想

一定要提前学习课上还没有讲但是实验要求掌握的内容，这次的正则表达式学习了两周时间，造成实验未能最终完成。还有就是实验的要求一再变更，造成了一定的混乱和停滞不前。