

**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 | 郭潇 |
| 学号 | 1160300521 |
| 班号 | 03005 |
| 电子邮件 | wqsnzszw@163.com |
| 手机号码 | 18523735573 |

**目录**

[1 实验目标概述 1](#_Toc513404127)

[2 实验环境配置 1](#_Toc513404128)

[3 实验过程 1](#_Toc513404129)

[3.1 待开发的四个应用场景 1](#_Toc513404130)

[3.2 基于语法的图数据输入 1](#_Toc513404131)

[3.3 面向复用的设计：Graph<L, E> 1](#_Toc513404132)

[3.4 面向复用的设计：Vertex 1](#_Toc513404133)

[3.5 面向复用的设计：Edge 1](#_Toc513404134)

[3.6 可复用API设计 1](#_Toc513404135)

[3.7 图的可视化：第三方API的复用（选做） 1](#_Toc513404136)

[3.8 设计模式应用 1](#_Toc513404137)

[3.8.1 使用State/Memento模式进行Vertex的状态管理（选做） 1](#_Toc513404138)

[3.8.2 使用factory method模式构造Vertex对象 1](#_Toc513404139)

[3.8.3 使用factory method模式构造Edge对象 1](#_Toc513404140)

[3.8.4 使用abstract factory或builder模式构造Graph对象 1](#_Toc513404141)

[3.8.5 使用Strategy模式调用centrality度量算法 1](#_Toc513404142)

[3.8.6 使用Composite模式设计超边对象（选做） 1](#_Toc513404143)

[3.8.7 使用decorator模式构造不同特征的Edge对象（选做） 1](#_Toc513404144)

[3.8.8 使用其他设计模式（选做） 1](#_Toc513404145)

[3.9 图操作指令的输入和处理（选做） 1](#_Toc513404146)

[3.10 应用设计与开发 1](#_Toc513404147)

[3.10.1 单词网络GraphPoet 1](#_Toc513404148)

[3.10.2 微博社交网络SocialNetwork 1](#_Toc513404149)

[3.10.3 网络拓扑图NetworkTopology 1](#_Toc513404150)

[3.10.4 电影网络MovieGraph 1](#_Toc513404151)

[3.11 应对四个应用面临的新变化（任选两个） 1](#_Toc513404152)

[3.11.1 单词网络GraphPoet 1](#_Toc513404153)

[3.11.2 微博社交网络SocialNetwork 1](#_Toc513404154)

[3.11.3 网络拓扑图NetworkTopology 1](#_Toc513404155)

[3.11.4 电影网络MovieGraph 1](#_Toc513404156)

[4 实验进度记录 1](#_Toc513404157)

[5 实验过程中遇到的困难与解决途径 1](#_Toc513404158)

[6 实验过程中收获的经验、教训、感想 1](#_Toc513404159)

# 实验目标概述

根据实验手册简要撰写。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
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

# 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。

利用和实验1相似的方法从github上利用clone操作获取实验项目

在这里给出你的GitHub Lab3仓库的URL地址（Lab3-学号）。

https://github.com/ComputerScienceHIT/Lab3-1160300521

# 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

分析四个应用场景的异同，理解需求。

1. GraphPoet

特点：

GraphPoet是单重有向图、带权图、单模图。即Vertex只能为Word，Edge只能为WordNeighborhood。权重为单词出现的次数，为非负数。

需求：

读入相应文本生成Graph；根据单词间的关系利用poet函数添加适当的bridge；

1. SocialNetwork

特点：

SocailNetwork是一个多重有向图、带权图、单模图，由此可知任意两个节点之间的边的总数最多为6。即Vertex只能为Persob，Edge可以为FriendTie、CommentTie、ForwardTie。

需求：

读入相应文本生成Graph；根据单词间的关系利用poet函数添加适当的bridge；

1. NetworkTopology

特点：

NetworkTopology是一个简单图、带权图、多模图。Vertex类型可以为Computer、Router、Server；Edge只能为NetworkConnection。

需求：

确保计算机和计算机之间不能形成边；服务器和服务器之间不能形成边；不出现边的起始点相同情况；权值为正数。

1. MovieGraph

特点：

MovieGraph是一个单重图、多模图，具有超边。Vertex类型可以为Movie、Actor、Director；Edge可以为MovieActorRelation、MovieDirectorRelation。

需求：

MovieActorRelation的边有权值，表示角色的次序；MovieDirectorRelation的边无权值；两种类型的边均不可出现环路。

## 基于语法的图数据输入

设计：

1. 从文本中逐行读入数据，判断文本的信息类型（图、点、边）。
2. 根据读入的信息构建对应的图，点和边并加入对应的信息。

实现：

1. 从文件路径中逐行读取相应文件。
2. 利用正则表达式判断读入信息是图、点、还是边。
3. 对读取的信息用string的spilt和replace方法进行提取。
4. 根据读取的类型构建相应结构并加入从文本中提取的信息。

## 面向复用的设计：Graph<L, E>

Graph<L, E>的理解：

1. Graph<L,E>是一个接口，L为点的泛型，E为边的泛型。
2. 编写点的抽象类Vertex，边的抽象类Edge，根据相应的应用从Vetex何Edge中继承子类满足需求。
3. 四种不同的应用均从ConcreteGraph中继承相应子类，通过重写、重载、加入新方法实现需求。

ConcreteGraph的设计：

1. RI为满足图中顶点不为null，边不形成自环
2. AF为不形成自环边和不为null的顶点
3. Safety from Rep Expouser为所有变量为private；所有类型是mutable的返回值，均创建新的变量返回防止泄露

ConcreteGraph的实现：

1. 用两个Set分别存放所有顶点和边。
2. checkRep检查顶点是否符合RI的要求
3. add操作依靠Set的add方法实现，判断顶点是否存在依靠自己写的方法且若加入的顶点已存在则返回false
4. set操作需检查要加入的边是否存在，若存在且权值为0则删除已有的边，权值大于0则修改权值并返回以前的权值；若加入的边不存在且不为空则加入新边。
5. vertices、sources、targets方法均创建新的变量返回。

ConcreteGraph的测试：

1. addVertex操作按照graph的大小为0、1、n；vertex是否在图中划分
2. addEdge操作按照graph的大小为0、1、n；weight的大小是否为0；target，source是否在graph中；加入的edge是否是new edge划分
3. removeVertex操作按照graph的大小为0、1、n；vertex是否在图中;
4. removeEdge操作按照edge的数量为0、1、n；edge是否在图中;
5. vertices操作按照graph的大小为0、1、n划分
6. Edge操作按照edge的数量为0、1、n划分
7. sources操作按照graph的大小为0、1、n；edge的数量为0、1、n划分
8. targets操作按照graph的大小为0、1、n；edge的数量为0、1、n划分

## 面向复用的设计：Vertex

Vertex<L>的理解

1. Vertex是待实现的四个应用中各节点的父类，其他节点需要通过重写、重载、加入新方法实现要求。
2. 根据不同的应用的节点，结合题意重写toString、equals、Hashcode方法。
3. 对于fillVertexInfo方法，不同节点信息的添加顺序和类型不一样，需对其进行重写。
4. 重写checkRep方法对每个节点信息的合法性进行检查。

Vertex的设计

1. RI为满足lable不为null且相应信息合法的点
2. AF为符合上述要求的所有点
3. Safety from Rep Expouser为所有变量为private；所有的变量通过get方法观察；remove方法只返回true或false；add方法返回

Vertex的实现：

1. 在构建方法中调用父类方法加入label，其他新添加的属性若不赋值则设为合法的默认值。
2. equals方法只有在点的类型和label均相同时才返回true。
3. 不同的点有不同的checkRep以保证符合题目的要求。
4. 重写hashcode方法会利用所有的属性产生新的hashcode值。

Vertex的测试：

1.测试equals方法，策略为：点的类型是相同还是不同；label是否相同

2.测试CheckRep方法，策略为：单个属性是否合法；

## 面向复用的设计：Edge

Edge<E>的理解

1. Edge是待实现的四个应用中各节点的父类，其他边需要通过重写、重载、加入新方法实现要求。
2. 根据不同的应用的边，结合题意重写equals、Hashcode方法。
3. 四个应用的边直接的父类应该为DirectedEdge、UndirectedEdge、HyperEdge。
4. 重写checkRep方法对每个节点信息的合法性进行检查。

Edge的设计

1. 相应边RI为满足该类型边约束条件。
2. AF为符合上述要求的所有边
3. Safety from Rep Expouser为所有变量为private；所有的变量通过get方法观察；remove方法只返回true或false；add方法返回

Edge的实现：

1. 在构建方法中调用父类方法加入label，其他新添加的属性若不赋值则设为合法的默认值。
2. 有向边有正数的权值，无向边的权值无限制。
3. equals方法只有在点的类型和label均相同时才返回true。
4. 不同的边有不同的checkRep以保证符合题目的要求，例如超边中点的类型必须是actor，Network边不能存person类型的顶点。
5. 重写hashcode方法会利用所有的属性产生新的hashcode值。

Edge的测试：

1.测试equals方法，策略为：边的类型是相同还是不同；label是否相同

2.测试CheckRep方法，策略为：单个属性是否合法；权值是否为非负；点的类型是否与边相关。

## 可复用API设计

设计：

1. degreecentrality：度中心性，用节点的度数除以图中其他顶点的数量。
2. betweenness centrality：介数中心性，无向图中用节点的度数除以最大边数量的一半。有向图中用节点入读和出度之和除以最大边数量。
3. closeness centrality：紧密中心性，用节点与其他连通节点距离之和除以顶点数量。
4. inDegreeCentrality：入度中心性，用节点的入度除以图中其他顶点的数量。
5. outDegreeCentrality：出度中心性，用节点的出度除以图中其他顶点的数量。

实现：

1. degreecentrality：调用graph的targe和source方法，计算获得顶点数之和再除以图中点的数量减一。
2. betweenness centrality：调用graph的vertices方法获取图中顶点个数，计算边的最大数量。然后调用graph的targe和source方法，计算获得顶点数之和除以边的最大数量。
3. closeness centrality：调用graph的vertices方法获取图中顶点个数，计算边的最大数量。调用distanc方法计算顶点与其他顶点的距离之和，用距离之和除以顶点个数减一的值即为结果。
4. inDegreeCentrality：调用graph的source方法，计算获得顶点数之和再除以图中点的数量减一。
5. outDegreeCentrality：调用graph的targe方法，计算获得顶点数之和再除以图中点的数量减一。

## 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

设计：编写一个抽象类为VertexFactory含抽象方法createVertex(String label, String[] args)，对每个节点均编写一个对应的factory继承VertexFactory。利用new VertexFactory().createVertex(label, args)形式输入信息返回相应的Vertex。

### 使用factory method模式构造Edge对象

设计：编写一个抽象类为EdgeFactory含抽象方法createEdge(String label, List<Vertex> vertices, double weight)，对每种类型的边均编写一个对应的factory继承EdgeFactory。利用new EdgeFctory().createEdge(label, vertices, weight)形式输入信息返回相应的Edge。

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

设计：编写一个抽象类为GraphFactory含抽象方法createGraph(String filePath)，对每种类型的拖均编写一个对应的factory继承GraphFactory。利用new GraphPoetFactory().createGraph(String filePath)

形式输入信息返回相应的Graph。

### 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。

### 单词网络GraphPoet

Vertex设计：

与父类Vertex相比，Word只是重新了equals方法，要求Word与同类型之间才能等价。

Edge设计：

WordNeighborhood是DirectedEdge子类，重新了equals方法，要求其与同类型之间才能等价。

GraphPoet设计：

GraphPoet继承了ConcreteGraph，对Lab2的代码进行了适当修改然后加入到其中即实现需求。

### 微博社交网络SocialNetwork

Vertex设计：

与父类Vertex相比，Person加入int类型的年龄和stirng类型的性别。重写checkRep以保证年龄为正数，性别为F或M;重写了equals方法，要求Person与同类型之间才能等价;重写了Hashcode和tostirng方法使得其适应需求。

Edge设计：

Edge具有FriendTie、CommentTie、ForwardTie三种类型，为了使其能够顺利加入到图中便重新了equals方法，要求其与同类型之间才能等价；根据实验手册要求，重载了边的Checkrep方法以使其权值为（0，1]。

SocialNetwork设计：

SocialNetwork继承了ConcreteGraph，根据实验手册要求当加入新的边，边的权值被修改或边被删除对已有边的权值进行修改，利用graph的removedge和addege操作在判断边类型后生成相应边重写加入到图中以实现需求。

### 网络拓扑图NetworkTopology

Vertex设计：

具有Computer、Router、Server三种类型并加入String类型的ip地址。于是重载checkRep以保证ip地址符合要求;重写了equals方法，要求节点与同类型之间才能等价;重写了Hashcode和tostirng方法使得其适应需求。

Edge设计：

NetworkConnection是UndirectedEdge的子类，为了使其能够顺利加入到图中便重新了equals方法，要求其与同类型之间才能等价；根据实验手册要求，重载了边的Checkrep方法使得权值为正数且计算机和计算机之间，服务器和服务器之间不能直接相连。

NetworkTopology设计：

NetworkTopology继承了ConcreteGraph，只需重写addVertex和addEdge判断点是否加入到图中和加入的顶点是否符合规范即可实现需求。

### 电影网络MovieGraph

Vertex设计：

Actor和Director是Person的子类则只需重写了equals方法，要求Person与同类型之间才能等价。Movie是Vertex的子类，加入了int类型的上映年份；string类型的拍摄国家和double类型的socre。利用四舍五入将score保留两位并重写equals、Hashcode、tostring。重载了checkrep方法以满足需求。

Edge设计：

MovieActorRelation继承了DirectedEdge为了使其能够顺利加入到图中便重新了equals方法，要求其与同类型之间才能等价并写了边的Checkrep方法以使两个顶点必须是Actor和Movie类型。MovieDirectorRelation继承了UndirectedEdge为了使其能够顺利加入到图中便重新了equals方法，要求其与同类型之间才能等价并写了边的Checkrep方法以使两个顶点必须是Director和Movie类型。

MovieGraph设计：

MovieGraph继承了ConcreteGraph，根据实验手册要求重写addVertex和addEdge判断点是否加入到图中和加入的顶点是否符合规范即可实现需求；加入新的方法用于加入和输出超边以实现需求。

## 应对四个应用面临的新变化（任选两个）

### 单词网络GraphPoet

设计：利用graph的remove和edges方法，获取graph的所有边，判断每条边的权值是否符合要求，若不符合要求则利用remove方法将去删去。

该变化可以用已有的方法实现，修改代价小。

### 微博社交网络SocialNetwork

### 网络拓扑图NetworkTopology

设计：添加一个新类型节点WirelessRouter为Router的子类，重写其equals方法即可实现需求。

该变化可以继承已有的顶点类型实现，修改代价小。

### 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 |
| 2018-04-25 | 18:00-22:00 | 完成WordPoet 的编写及测试 | 完成 |
| 2018-04-26 | 14:00-17:00 | 完成Socialnetwork的编写 | 遇到问题，未完成 |
| 2018-04-28 | 15:00-18:00 | 完成Socialnetwork的编写和测试 | 完成 |
| 2018-04-29 | 13:30-16:00 | 完成NetworTopology的编写及测试 | 完成 |
| 2018-04-29 | 19:30-21:00 | 完成MovieGraph的编写 | 基本完成 |
| 2018-05-01 | 14:00-17:00 | 完成MovieGraph的测试和各种类型点和边的测试 | 大部分玩成 |
| 2018-05-01 | 1900-21:00 | 完成MovieGraph的测试和各种类型点和边的测试 | 完成 |
| 2018-05-02 | 19:00-21:00 | 完成基于语法的图像输入部分 | 完成 |
| 2018-05-03 | 16:00-17:30 | 完成可复用api设计 | 完成 |
| 2018-05-04 | 14:30-17:00 | 加入设计模式 | 完成 |
| 2018-05-04 | 18:30-21:00 | 写实验报告 | 在写实验报告时发现很多问题 |
| 2018-05-05 | 13:00-17:00 | 继续写实验报告并完善不足 | 大部分完成 |
| 2018-05-05 | 19:00-22:00 | 完成写实验报告并完善不足 | 完成 |
| 2018-05-06 | 10:00-12;00 | spec和IR、AF、Safety from Rep Expouser的编写 | 完成 |
|  |  |  |  |
|  |  |  |  |

# 实验过程中遇到的困难与解决途径

1. 在做Socialnetwork过程中，当某条边的权值变化时，对其他的边进行修改后再加入图中出现重复情况。

解决：addedge方法在判断edge是否在图中已经存在时，使用了Set结构的contain方法，而权值修改的边显然会被误认为是新的边，于是不使用contain方法，通过遍历删除和添加改写了新的addedge方法以实现需求。

1. 不知道怎样测试checkrep是否抛出异常

解决：查阅教程，知道了junit4中测试抛出异常的方法，判断是否抛出AssertionError类的异常确保了checkrep方法的合理性。

1. 在做基于语法的图形输入时由于对正则表达式不熟悉，前期并未尝试去完成。

解决：在课上讲完正则表达式后，对正则表达式有了更多的了解，结合ppt中的例子，完成了该部分。

1. 在编写Vertex和Edge时未注意实验手册的新变化导致自己编写的程序偏离实验要求

解决：由于改写代价过大，只能在现有基础上加入新的要求，对新的要求进行单独测试。也牢记以后留意实验手册的更 新。

# 实验过程中收获的经验、教训、感想

1. 理解了重写和重载的区别，知道了继承关系的结构和使用。
2. 意识到测试的重要性，通过测试既能检验新加入的功能是否正确，又能发现以前编写的方法是否能实现新应用的功能。
3. 进一步了解了checkrep的作用：在构建点和边时便对其合法性进行检查可以防止在加入图时出现错误，减小了graph的代码量和考虑的情况数量。
4. 通过语法驱动编程，进一步掌握了正则表达式的知识，同时意识到语法驱动输入在解析数据合法性和大量数据上的优越性。
5. 我认为面向ADT的编程与直接面向应用场景编程相比，编写的基础方法能够被复用的程序直接使用或是用于实现新的方法，如果充分考虑多个场景的共同点，便可以提高效率。
6. 与前两次实验相比，在本次实验初期因为太多知识不了解，所有有一定畏惧心理，直到截止前两周才开始编写，导致本次实验与以前相比质量下降。所有意识到在以后的实验中，遇到不会的知识应该积极去学习掌握，根据实验的各项要求尽早完成实验。