

**2018年春季学期  
计算机学院大二软件构造课程**

**Lab 2实验报告**

|  |  |
| --- | --- |
| 姓名 | 路天帅 |
| 学号 | 1163710208 |
| 班号 | 1637102 |
| 电子邮件 | 1321743979@qq.com |
| 手机号码 | 18846178718 |

**目录**

[1 实验目标概述 1](#_Toc507927440)

[2 实验环境配置 1](#_Toc507927441)

[3 实验过程 1](#_Toc507927442)

[3.1 Poetic Walks 1](#_Toc507927443)

[3.1.1 Get the code and prepare Git repository 2](#_Toc507927444)

[3.1.2 Problem 1: Test Graph <String> 2](#_Toc507927445)

[3.1.3 Problem 2: Implement Graph <String> 2](#_Toc507927446)

[3.1.3.1 Implement ConcreteEdgesGraph 2](#_Toc507927447)

[3.1.3.2 Implement ConcreteVerticesGraph 3](#_Toc507927448)

[3.1.4 Problem 3: Implement generic Graph<L> 3](#_Toc507927449)

[3.1.4.1 Make the implementations generic 3](#_Toc507927450)

[3.1.4.2 Implement Graph.empty() 3](#_Toc507927451)

[3.1.5 Problem 4: Poetic walks 4](#_Toc507927452)

[3.1.5.1 Test GraphPoet 4](#_Toc507927453)

[3.1.5.2 Implement GraphPoet 4](#_Toc507927454)

[3.1.5.3 Graph poetry slam 4](#_Toc507927455)

[3.1.6 Before you’re done 4](#_Toc507927456)

[3.2 Re-implement the Social Network in Lab1 5](#_Toc507927457)

[3.2.1 FriendshipGraph类 5](#_Toc507927458)

[3.2.2 Person类 5](#_Toc507927459)

[3.2.3 客户端main() 5](#_Toc507927460)

[3.2.4 测试用例 6](#_Toc507927461)

[3.2.5 提交至Git仓库 6](#_Toc507927462)

[3.3 The Transit Route Planner（选做，额外给分） 7](#_Toc507927463)

[4 实验进度记录 7](#_Toc507927464)

[5 实验过程中遇到的困难与解决途径 8](#_Toc507927465)

[6 实验过程中收获的经验、教训、感想 8](#_Toc507927466)

# 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

示外泄（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 并据此设计测试用例。

# 实验环境配置

在lab1中已配置好eclipse和git等环境，在本次实验中又在eclipse中通过商店来安装EclEmma插件来统计Junit测试用例的代码覆盖度

Lab2-1163710208

# 实验过程

请仔细对照实验手册，针对三个问题中的每一项任务，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！）。

## Poetic Walks

通过编写graph类来一步步练习泛型类的编程方法。

### Get the code and prepare Git repository

在 Get the code 步骤中，你无法连接 MIT 的 Athena 服务器，使用 Git

指令从https://github.com/rainywang/Spring2018\_HITCS\_SC\_Lab2/tree/master/P1获取初始代码，然后按照以下步骤上传。

1. 在git Bash用命令行打开实验所在的目录
2. 使用git add .和git commit等命令来将实验内容加入仓库
3. 最后使用git remote add origin [git@github.com:ComputerScienceHIT/Lab2-1163710208.git](mailto:git@github.com:ComputerScienceHIT/Lab2-1163710208.git)命令将项目加入到实验的Lab2-1163710208

### Problem 1: Test Graph <String>

按照实验要求在对应位置的TODO替换为abstraction function 和representation invariant以及testing strategy等文件注释信息。

### Problem 2: Implement Graph <String>

#### Implement ConcreteEdgesGraph

1. 首先设计Edge类，其中定义一个String类型的source存源节点和一个String类型的target存终结点还有一个int类型的weight存储权。

在构造器中初始化source，target和weight。

1. 然后依次实现各个函数，add函数直接HashSet的add方法，结束后可以通过checkRep（）检测一下是否有重复元素。
2. Set函数过程比较复杂，首先判断weight是否为零，若不为零，然后判断source和target是否在图里，若不在，将其加入到图里，然后查看这个边是否在图里，若已存在则更新weight，若不在新建边。若weight为零，判断该边是否为存在，若存在，则删除。否则返回0.
3. Remove函数直接调用HashSet的remove方法。
4. Vertices直接返回HashSet类型的vertices。
5. Sources函数，遍历edges如果target存在且weight不为零，则把source和weight存入一个HashMap里，最后返回这个HasnMap。
6. Targets函数，遍历edges如果source存在且weight不为零，则把target和weight存入一个HashMap里，最后返回这个HasnMap。
7. toString函数调用Edge类的toString方法最后返回一个包含所有点和所有边和权值的String。

#### Implement ConcreteVerticesGraph

1. 首先设计Vertice类，其中定义一个String类型的source存源节点和一个HasnMap类型的edge存target终结点和int类型的weight权。

在构造器中初始化source。

1. 然后依次实现各个函数，add函数采用循环遍历ArrayList的方法，如果点不存在则加进去，存在就返回false。结束后可以通过checkRep（）检测一下是否有重复元素。
2. Set函数过程比较复杂，首先判断weight是否为零，若不为零，然后判断source和target是否在图里，若不在，将其加入到图里，然后查看这个边是否在图里，若已存在则更新weight，若不在新建边。若weight为零，判断该边是否为存在，若存在，则删除。否则返回0.
3. Remove函数采用循环遍历ArrayList的方法，如果点存在则删除，不存在就返回false。
4. Vertices直接返回HashSet类型的vertices。
5. Sources函数，遍历edges如果target存在且weight不为零，则把source和weight存入一个HashMap里，最后返回这个HasnMap。
6. Targets函数，遍历edges如果source存在且weight不为零，则把target和weight存入一个HashMap里，最后返回这个HasnMap。
7. toString函数调用Edge类的toString方法最后返回一个包含所有点和所有边和权值的String。

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

#### Make the implementations generic

1. 按照mit要求先将类名ConcreteVerticesGraph和ConcreteEdgesGraph改为<L>。
2. 将Graph接口也写为<L>，然后在类里逐一将String改为L，然后有些地方调用Edge类的和Vertex类的位置没有String但需要添加一个<L>说明是泛型。

#### Implement Graph.empty()

根据mit要求随便选一个，我选择了ConcreteEdgesGraph作为emptyGraph的实现。直接return一个new ConcreteEdgesGrap<L>.

### Problem 4: Poetic walks

#### Test GraphPoet

在测试中我又添加了一个Test the computer science system.的测试。

#### Implement GraphPoet

1. 首先完成GraphPoet函数，按行读取corpus然后按照空格换行来读取每个字符串，按照设计的原则来调用Graph类建立个有向带权图。
2. 然后完成poem函数，将输入input函数以空格分割然后存入一个ArrayList中，遍历该ArrayList若发现某两点中间有长度为二的边则加入该边上的bridge word。

#### Graph poetry slam

添加了一个Test the computer science system.的测试。

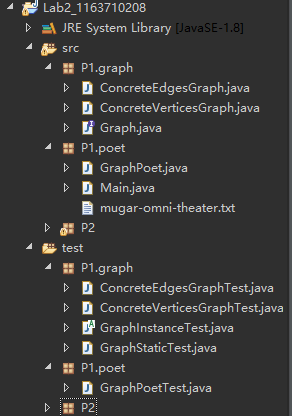
### Before you’re done

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/" \l "before_youre_done)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。

1. 在git Bash用命令行打开实验所在的目录
2. 使用git add .和git commit等命令来将实验内容加入仓库
3. 最后使用git remote add origin [git@github.com:ComputerScienceHIT/Lab2-1163710208.git](mailto:git@github.com:ComputerScienceHIT/Lab2-1163710208.git)命令将项目加入到实验的Lab2-1163710208

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

学习通过调用泛型类来实现一些具体问题。

### FriendshipGraph类

1. AddVertex函数通过调用graph的add方法直接添加。
2. AddEdge函数通过调用graph的set方法直接加入从A到B的边。
3. GetDistence函数采用广度优先遍历的方法来获取某两点的最短距离。

### Person类

1. 首先定义一个String类型的name存每一个人的名字，和boolean的tag用于判断广度优先遍历的点是否被访问。
2. 然后构造器内初始化名字name。tag函数来设置该人是否被访问，getTag来返回该信息。

### 客户端main()

首先定义一个FriendshipGraph类型的graph然后定义四个Person类型的人，然后调用graph的addVertex方法来将四个人加入graph，然后再调用graph的addEdge来将好友关系图加入graph，最后调用getDistance方法来求出所测试的人之间的距离。

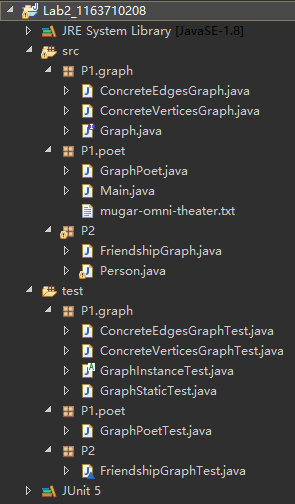
### 测试用例

首先定义一个FriendshipGraph类型的graph然后定义四个Person类型的人，然后通过语句assertEquals(rachel,graph.person.get(0)); 来测试人物是否加入到graph中。然后再调用graph的addEdge来将好友关系图加入graph，然后调用graph的addVertex方法来将四个人加入graph，然后通过语句assertEquals(1,graph.g[0][1]); 来测试图是否创建成功。最后调用getDistance方法来求出所测试的人之间的距离。再通过语句assertEquals(2,graph.getDistance(PersonA, PersonB)); 来测试A和B的距离。

### 提交至Git仓库

1. 在git Bash用命令行打开实验所在的目录
2. 使用git add .和git commit等命令来将实验内容加入仓库
3. 最后使用git remote add origin [git@github.com:ComputerScienceHIT/Lab2-1163710208.git](mailto:git@github.com:ComputerScienceHIT/Lab2-1163710208.git)命令将项目加入到实验的Lab2-1163710208

在这里给出你的项目的目录结构树状示意图。



## The Transit Route Planner（选做，额外给分）

请自行组织本节内的目录结构。

# 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 |
| 03-19-03-25 | 18:30-21:30 | 学习基本知识 | 基本完成 |
| 2018-03-26 | 18:30-22:00 | 编写ConcreteEdgesGraph类 | 延期一个半小时 |
| 2018-03-28 | 18:30-22:00 | 编写ConcreteVerticesGraph类 | 按时完成 |
| 2018-03-29 | 18:30-22:00 | 写graph的Junit测试改泛型 | 按时完成 |
| 2018-03-30 | 18:30-22:00 | 写poetic walk相关函数及测试 | 延迟两小时 |
| 2018-03-01 | 18:30-22:00 | 写P2的两个类 | 延迟一个半小时 |
| 2018-04-02 | 18:30-22:00 | 写P2的Junit测试 | 基本按时完成 |

# 实验过程中遇到的困难与解决途径

实验中遇到的困难主要就是对于题目的理解有障碍，尤其是poetic walk的把语料库变为图的要求，过于晦涩，而且还是英文，反反复复读了十多遍才完全理解题意，而且觉得还有许多模棱两可的地方，对于实验的进行造成极大障碍，最后就按照mit的字面意思来编写的实验。

# 实验过程中收获的经验、教训、感想

通过本次实验我对Java语言的理解更加深入，也熟悉了Java泛型的一些操作。教训就是以后要好好学英语。觉得这种泛型编程方法很好，模块化很强且利于扩展和复用。工作量还是不少的， CMU和MIT的作业很有挑战性，很有针对性。对于Java没学好的我来说，作业完成起来还是很有难度的。