

**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 | 黄柄峄 |
| 学号 | 1160300526 |
| 班号 | 1603005 |
| 电子邮件 | [1181049074@qq.com](mailto:1181049074@qq.com) |
| 手机号码 | 17766589109 |

**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 1](#_Toc507927444)

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

[3.1.3 Problem 2: Implement Graph <String> 1](#_Toc507927446)

[3.1.3.1 Implement ConcreteEdgesGraph 2](#_Toc507927447)

[3.1.3.2 Implement ConcreteVerticesGraph 2](#_Toc507927448)

[3.1.4 Problem 3: Implement generic Graph<L> 2](#_Toc507927449)

[3.1.4.1 Make the implementations generic 2](#_Toc507927450)

[3.1.4.2 Implement Graph.empty() 2](#_Toc507927451)

[3.1.5 Problem 4: Poetic walks 2](#_Toc507927452)

[3.1.5.1 Test GraphPoet 2](#_Toc507927453)

[3.1.5.2 Implement GraphPoet 2](#_Toc507927454)

[3.1.5.3 Graph poetry slam 2](#_Toc507927455)

[3.1.6 Before you’re done 2](#_Toc507927456)

[3.2 Re-implement the Social Network in Lab1 2](#_Toc507927457)

[3.2.1 FriendshipGraph类 2](#_Toc507927458)

[3.2.2 Person类 3](#_Toc507927459)

[3.2.3 客户端main() 3](#_Toc507927460)

[3.2.4 测试用例 3](#_Toc507927461)

[3.2.5 提交至Git仓库 3](#_Toc507927462)

[3.3 The Transit Route Planner（选做，额外给分） 3](#_Toc507927463)

[4 实验进度记录 3](#_Toc507927464)

[5 实验过程中遇到的困难与解决途径 3](#_Toc507927465)

[6 实验过程中收获的经验、教训、感想 4](#_Toc507927466)

# 实验目标概述

根据实验手册简要撰写。

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

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。

ANS:在Lab1中已经完成配置，在新实验中没有重新配置。

在这里给出你的GitHub Lab2仓库的URL地址（Lab2-学号）。

https://github.com/ComputerScienceHIT/Lab2-1160300526.git

# 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

在这里简要概述你对该任务的理解。

1. 构造一个有向带权图的抽象数据类型。图由结节，边构成。其中节点是immutable类型；

2. 图的基本操作如下：

（1）初始化一个空的图（没有结点也没有边）；

（2）增加或删除结点（删除结点时会同时删除与该结点有关的边）；

（3）增加或删除边；

（4）得到所有节点的集合；

（5）得到所有的有向边的源点或终点。

3. 有关细节：

（1）Graph <L>的规范中，对类型L的不变性有相同的要求；

（2）该任务要求使用两次不同的代码实现Graph<L>，练习选择抽象函数和代表不变量，并防止rep暴露。因为List的ArrayList和LinkedList实现满足具有不同性能需求的客户端；

（3）通过我们的图抽象数据类型，我们最终将在Problem 4中实现Graph Poet，可以将泛型L替换为String，最终利用该抽象数据类型来产生一个诗歌的类。

### Get the code and prepare Git repository

从GitHub获取该任务的代码、在本地创建git仓库、使用git管理本地开发步骤如下：

（1）建立本地仓库；

（2）fork至自己的远程仓库；

（3）clone至本地仓库；

（4）完成编程；

（5）add至暂存区并commit提交到本地仓库；

（6）push至远程仓库。

### Problem 1: Test Graph <String>

综述：本部分是设计并实施Graph <String>的测试。

在这一部分实验要求测试带有字符串顶点标签的图。设置如下：

（1）静态Graph.empty（）方法的测试策略和测试在GraphStaticTest.java中。由于该方法是静态的，只会有一个实现；

（2）在测试中，需要使用emptyInstance（）方法来获取空图；

### Problem 2: Implement Graph <String>

综述：任务该部分要求实现带有字符串标签的带权有向图。

在这一部分的设置如下：

（1）记录抽象函数和表示形式不变；

（2）随着代表不变量，记录类型如何防止rep暴露；

（3）实施checkRep()函数来检查表示不变量；

（4）用抽象值的有用可读表示来实现toString()；

#### Implement ConcreteEdgesGraph

综述：在ConcreteEdgesGraph.java中声明类Edge。

（1）首先定义Edge的属性：用邻接表表示两个顶点之间的关系（需要注意的是边为带权有向边），Edge是immutable类型；

（2）equals()与hashCode()的重写；

（3）确保在toString()上使用@Override并且确保正确覆盖了该方法，而非创建一个新的不同的方法；

（4）进行JUnit测试。

#### Implement ConcreteVerticesGraph

综述：在ConcreteVerticesGraph.java中声明类Vertice。

（1）定义Vertice的属性：用List来实现结点类；

（2）重写toString(),并且在在ConcreteVerticesGraphTest.java中为其编写测试以测试Graph规范；

（3）进行JUnit测试。

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

综述：结节标签是immutable，但是将Sting类型拓展到其它类型（用equals()方法来比较）。

#### Make the implementations generic

综述：实现支持任何类型的边与结点类： Edge<L>、Vertex<L>。

（1）将用Edge或List<Edge>声明的变量改为用Edge<L>与<Edge<L>> 声明；

（2）将的诸如new ConcreteEdgesGraph（）或 new Edge（）的声明改为ConcreteEdgesGraph <String>（）与Edge<L>()；

（3）对相应的方法改变为对应的类型；

（4）进行JUnit测试

#### Implement Graph.empty()

综述：实现Graph.empty()。

（1）支持不同类型的结点或边；

（2）在GraphStaticTest.java中添加对空图的测试，检验其是否可以创建并使用含有不同类型的结点的图实例；

（3）进行JUnit测试。

### Problem 4: Poetic walks

综述：利用上面实现的图来将诗歌知情个单词图。具体如下：

（1）顶点是不区分大小写的单词；

（2）边权重是按序相邻性计数；

（3）由于一个单词不具有多个最大权值的边，因此可以得到唯一的句子。

#### Test GraphPoet

综述：在GraphPoetTest.java中设计并实现GraphPoet的测试。

（1）将测试文件放入test/poet目录下；

（2）用java.File.io的方法读取文件；

（3）函数readLine()读取一行；

（4）对GraphPoet进行测试。Implement GraphPoet

#### Graph poetry slam

综述：加入其它的诗歌。

（1）将文件加入sre/poet目录下（与mugar-omni-theater.txt相同的目录中。）

### 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。

通过Git提交当前版本到GitHub上你的Lab2仓库：

（1）git add 文件

（2）git commit 提交到本地仓库

（3）git push到远程仓库

在这里给出你的项目的目录结构树状示意图。

ConcreteEdgesGraph.java

ConcreteVerticesGraph.java

graph

Graph.java

GraphPoet.java

src

Main.java

poet

mugar-omni-theater.txt

P1

ConcreteEdgesGraphTest.java

ConcreteVerticesGraphTest.java

graph

test

GraphInstanceTest.java

GraphStaticTest.java

GraphPoetTest.java

poet

## Re-implement the Social Network in Lab1

综述：在Graph<L>实现的基础上重构social work。

（1）图的结点是一个String类型的人名；

（2）结点之间存在有向关系；

（3）可以完成增加删除结点或边的操作。

### FriendshipGraph类

综述：关系图中的结点与结点之间的关系。

（1）Person[][]类型的二维数组persons，邻接矩阵用于记录人与人之间的关系；

（2）int[]类型的一维数组total，用于记录每个人有多少个好友；

（3）numPers记录整个关系图中有多少个人；

（4）函数功能：

addPerson(Person person)用于增加结点；addFriendship(Person person\_1, Person person\_2)用于增加边；getDistance(Person person\_1, Person person\_2)用于确定任意两个人的最短距离；isInGraph(Person person)用于判断某个人是否存在于该关系图中；Person[] getFriends(Person person)用于获得某人的所有朋友。

### Person类

有一个String类型的属性

### 客户端main()

如实验一，确定Rachel, Ross, Ben, Kramer四人，并确定关系图。

### 测试用例

调用FriendshipGraph类中的方法，检测是否准确。

### 提交至Git仓库

通过Git提交当前版本到GitHub上你的Lab2仓库：

（1）git add 文件

（2）git commit 提交到本地仓库

（3）git push到远程仓库

在这里给出你的项目的目录结构树状示意图。

FriendshipGraph.java

src

Person.java

P2

FriendshipGraphTest.java

test

## 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 |
| 3月19日 | 13:45~15:30 | 完成P1 | 完成 |
| 3月26日 | 13:45~15:30 | 完成P2 | 完成 |
| 4月2日 | 13:45~15:30 | 完成P3 | 完成 |

# 实验过程中遇到的困难与解决途径

（1）对英文实验手册理解存在困难

借助翻译工具与反复阅读来确定任务要求；

（2）在使用git时出现了一些问题

查阅博客与文档，尤其学习git remote add origin,git add, git clone,git fork,git push,git pull等

（3）java语法

阅读书本，并且在网络上寻找前人经验

# 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. 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？
2. 使用泛型和不使用泛型的编程，对你来说有何差异？
3. 在给出ADT的规约后就开始编写测试用例，优势是什么？你是否能够适应这种测试方式？
4. 本实验设计的ADT在三个应用场景下使用，这种复用带来什么好处？
5. 为ADT撰写specification, invariants, RI, AF，时刻注意ADT是否有rep exposure，这些工作的意义是什么？你是否愿意在以后编程中坚持这么做？
6. 关于本实验的工作量、难度、deadline。
7. 《软件构造》课程进展到目前，你对该课程有何体会和建议？

收获：

（1）在实验一的基础上，更加熟练了使用新的工具——git与GitHub，这使我的代码管理有了很大的便利；

（2）面向ADT编程会有更高的复用性，在本次实验中已经得到了验证，对Graph<L>的多次利用，这是面向ADT编程的优势所在；

（3）泛型接口可以面向所有的数据类型，使用方面更为广阔；

（4）复用的好处是避免了反复编写同一段代码，提高了编程效率；

（5）撰写specification, invariants, RI, AF有利于自己或他人阅读编写的代码，我会逐渐练习这方面的能力。