

**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 | 王世川 |
| 学号 | 1163710227 |
| 班号 | 1637102 |
| 电子邮件 | admin@mail.kurokawa.club |
| 手机号码 | 17766520412 |

**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

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

实验环境配置没有遇到困难。

<https://github.com/ComputerScienceHIT/Lab2-1163710227>

# 实验过程

## Poetic Walks

在这里简要概述你对该任务的理解。

### Get the code and prepare Git repository

打开实验说给的仓库地址，下载zip在本地解压，重新处理目录结构以适应本次实验。

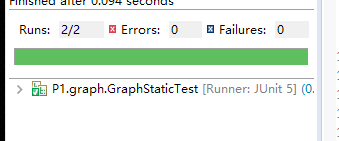
### Problem 1: Test Graph <String>

步骤一：按要求，尝试运行Junit测试，graph类中，返回空图的方法没有实现所以测试失败，修改使之返回边图或者点图任意一个。我选择了点图。

步骤二：按题目要求，为了降低难度所以可以用固定类型String，所以讲graph类中所有的L都先改成Sting。

步骤三：修改完后再运行测试知道因为点图中对应方法没有实现，所以爆出代码中原本填好的not implemented错误，将对应函数补全。

步骤四：再测试：



### Problem 2: Implement Graph <String>

分析和思路：按照指导将代码中todo部分补充完整即可

#### Implement ConcreteEdgesGraph

步骤一：先编写测试文件，此时还不需要考虑equal和hashcode方法，需要测试的地方主要是tostring，此处分三类情况需要考虑的：

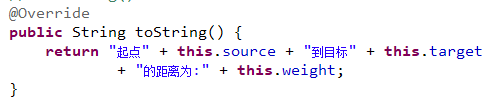
所有边为空或者所有点为空，所有边和点都不为空，有的点是空的，或者有些边是空的

其他函数每个函数编写一个测试即可

步骤二：实现构造函数，参考其他比如String，起码应当有两种构造的方法，无参数的用于构造一个空的图，有参数的应当能根据参数构造一个新图，这样的构造方法可以有很多，我只实现了导入点的set和边的list的一个方法，其他可能固定方法可以以此类推。

步骤三：先实现Edge子类，包括构造方法，cheakrep，tostring，getsource和gettarget

其中tostring的定义是自由的，我定义为：



变量为source 和 target 还有 a int number weight

source 和 target都不能为空，权值应当大于0

所有的变量都是私有、final的，weight是固定类型int

步骤四：实现整个类，最难的地方是cheakrep和tostring，这个类用存储边来记录一个图，点应当为集合来防止重复，边记录两个点：source和target，还有两者之间的距离，应当防止边重复，即source和target不能同时相同，两个字段（顶点和边）用final和private修饰

Cheackrep时将List导入set，比较大小来检查重复，确保没有一个边指向不存在的点，并且所有边的长度都是大于0的，在每次增加和修改之后都要吊用这个函数来检查。

Add函数应注意检查该顶点是否是已经存在的。

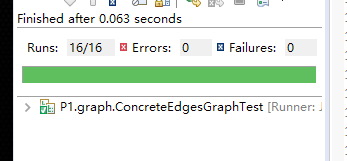
Set函数先检查对应的两个顶点是存在的，如果两个点有一个不存在，先加入点再新建边,注意两个点不能一样，如果是已经存在的边，则更新这条边，否则新建，注意传入的weight应该大于0。

Remove函数先检查该点是否存在，存在再移除，删除顶点的同时要遍历查找所有相关的边，也要同时删除。

Set函数返回一个新建的set。

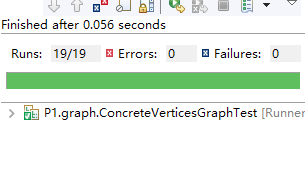
Tostring函数可以自由发挥

步骤五：设置了15个测试都通过了：



#### Implement ConcreteVerticesGraph

实现此步骤的过程基本同上一部分。测试结果如下：



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

分析和思路：先将原来所有的String都修改成<L>，为子类添加标签<L>，然后大体上修复各个细小的报错即可。

#### Make the implementations generic

编写完接口以后，要在实体类中实现，补全接口中定义的函数。只需要把之前的变量类型修改成泛型即可。

#### Implement Graph.empty()

**此处返回一个新建的实体类，边图和点图都可以。**

### Problem 4: Poetic walks

#### Test GraphPoet

编写测试规范由于时间问题，没法仔细写了

#### Implement GraphPoet

大致上完成了GraphPoet，没有完成代码覆盖度测试和编写测试规范。

#### Graph poetry slam

### Before you’re done

请按照<http://web.mit.edu/6.031/www/sp17/psets/ps2/#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。

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

在这里简要概述你对该任务的理解。

### FriendshipGraph类

**已完成。**

### Person类

已完成。实际上写上一个类的过程中就经常需要在Person类中增加方法，所以二者是同步完成的。

### 客户端main()

未完成

### 测试用例

未完成

### 提交至Git仓库

如何通过Git提交当前版本到GitHub上你的Lab3仓库。

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 |
| 4.1 | 13.45-15.30 | 阅读实验报告，理解任务 | 按时完成 |
| 4.8 | 12.00-13.00 | 完成3.1中的边图 | 按时完成 |
| 4.8 | 20.00-23.00 | 做其他的实验部分 | 未能按时完成 |

# 实验过程中遇到的困难与解决途径

个人的事情有点多，这次的实验没能完成是计划之中的事情，尽力在各个科目中平摊欠缺的时间，以后的实验一定尽力。

# 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？

面向ADT编程时考虑更抽象，蛀牙考虑实现的作品有何特性，面向应用时考虑实现具体功能。前者更有套路可循。

1. 使用泛型和不使用泛型的编程，对你来说有何差异？

使用泛型时候因为无法知道对象类型，没法利用对象接口。如果不使用泛型可以利用已知的对象接口做一些事情。

1. 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？

优势是清楚自己需要完成的目标。能适应在有测试用例下编程，但是不适应一开始就自己写一个测试用例，因为这个时候并不知道如何评估自己的代码才是可靠的。

1. 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？

复用可以帮助我们理解泛型的意义，以及发现一些不足，并去修补。

1. 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？

防止变量暴露

一个好的ADT应当将他的内部信息、变量隐藏起来，用户只需要知道它的用法，黑箱操作即可。这些事情旨在让程序更加安全，无法从外部破坏一个对象。愿意。

1. 关于本实验的工作量、难度、deadline。

本实验的工作量不算上而外加分题实际上适当，但是由于我自己也有其他的事情觉得挺赶的。如果能够有中文的实验指导将会节省大量时间，因为用到了大量专业术语，谷歌翻译在翻译实验报告的时候非常糟糕，而如果自己阅读英文的指导，有时候极其难琢磨一些细微的意思，相当于额外附加了一篇英语阅读理解。