

**2019年春季学期  
计算机学院《软件构造》课程**

**Lab 2实验报告**

|  |  |
| --- | --- |
| 姓名 | 柳轶 |
| 学号 | 1170300411 |
| 班号 | 1703004 |
| 电子邮件 | 1323308672@qq.com |
| 手机号码 | 18145644352 |

**目录**

[1 实验目标概述 1](#_Toc1988214)

[2 实验环境配置 1](#_Toc1988215)

[3 实验过程 1](#_Toc1988216)

[3.1 Poetic Walks 1](#_Toc1988217)

[3.1.1 Get the code and prepare Git repository 1](#_Toc1988218)

[3.1.2 Problem 1: Test Graph <String> 1](#_Toc1988219)

[3.1.3 Problem 2: Implement Graph <String> 1](#_Toc1988220)

[3.1.3.1 Implement ConcreteEdgesGraph 2](#_Toc1988221)

[3.1.3.2 Implement ConcreteVerticesGraph 2](#_Toc1988222)

[3.1.4 Problem 3: Implement generic Graph<L> 2](#_Toc1988223)

[3.1.4.1 Make the implementations generic 2](#_Toc1988224)

[3.1.4.2 Implement Graph.empty() 2](#_Toc1988225)

[3.1.5 Problem 4: Poetic walks 2](#_Toc1988226)

[3.1.5.1 Test GraphPoet 2](#_Toc1988227)

[3.1.5.2 Implement GraphPoet 2](#_Toc1988228)

[3.1.5.3 Graph poetry slam 2](#_Toc1988229)

[3.1.6 Before you’re done 2](#_Toc1988230)

[3.2 Re-implement the Social Network in Lab1 2](#_Toc1988231)

[3.2.1 FriendshipGraph类 2](#_Toc1988232)

[3.2.2 Person类 3](#_Toc1988233)

[3.2.3 客户端main() 3](#_Toc1988234)

[3.2.4 测试用例 3](#_Toc1988235)

[3.2.5 提交至Git仓库 3](#_Toc1988236)

[3.3 Playing Chess 3](#_Toc1988237)

[3.3.1 ADT设计/实现方案 3](#_Toc1988238)

[3.3.2 主程序ChessGame设计/实现方案 3](#_Toc1988239)

[3.3.3 ADT和主程序的测试方案 3](#_Toc1988240)

[3.4 Multi-Startup Set (MIT) 4](#_Toc1988241)

[4 实验进度记录 4](#_Toc1988242)

[5 实验过程中遇到的困难与解决途径 4](#_Toc1988243)

[6 实验过程中收获的经验、教训、感想 4](#_Toc1988244)

[6.1 实验过程中收获的经验和教训 4](#_Toc1988245)

[6.2 针对以下方面的感受 4](#_Toc1988246)

# 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泄露（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

实验环境设置请参见Lab-0 实验指南。

除此之外，本次实验需要你在Eclipse IDE中安装配置EclEmma（一个用于统计JUnit测试用例的代码覆盖度的plugin）。请访问<http://www.eclemma.org>，了解EclEmma并学习其安装、配置和使用。

本次实验在GitHub Classroom中的URL地址为：

<https://classroom.github.com/a/z9utaaos>

请访问该URL，按照提示建立自己的Lab2仓库并关联至自己的学号。

<https://github.com/1170300411liuyi/lab2.git>

本地开发时，本次实验只需建立一个项目，统一向GitHub仓库提交。实验包含的4个任务分别在不同的目录内开发，具体目录组织方式参见各任务最后一部分的说明。请务必遵循目录结构，以便于教师/TA进行测试。

# 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

### Get the code and prepare Git repository

### 构造一个Graph，实现spec中的基本功能，并且以边和点两种方式实现接口，并且需要将ADT泛型化。编写测试文件的时候，需要从实现的基本功能出发，有良好的测试覆盖率。对于problem4，我们则需要根据文件中的输入，构造poet，再由input和图中的映射关系，得出最后的poem。

### Problem 1: Test Graph <String>

选用ConcreteEdgesGraph作为Graph<String>的具体实现

### Problem 2: Implement Graph <String>

#### 

#### public boolean add(String vertex)思路：如果在vertices的Set集合中成功添加了顶点string，则返回true

#### private int findEdge(String source, String target)思路：My method findEdge() ，为了找到list是否存在一条指定的边，该边的source和target必须与传入的参数相等，返回该边的index

#### public int set(String source, String target, int weight)思路：根据parameter找到指定边，并调用findEdge()返回index。当weight>0，如果index<0，没找到指定边，则添加顶点和边；如果index>0，找到了指定边，则update这条边。当weight>0且index<0时，找到了指定边，将其删除/

#### Implement ConcreteVerticesGraph

public ConcreteVerticesGraph() {

}

public void checkRep()

思路：每个顶点只能存在一个实例，因此顶点个数vertices()的大小相等

public boolean add(String vertex)

思路：若vertices()中已包含vertex，返回false，否则新建一个顶点将其加入vertices即可。

private int findVertex(String str)

思路：My method findVertex , 找到对应str的vertex所在的位置，返回index。

public int set(String source, String target, int weight)

思路：如果存在vertices()中找到source，使用findVertex返回对应index，从而找到源点，否则以source为string创建一个新的源点，并将它添加进vertices。对于target操作同理。

得到源点和目标点后，分别对目标点调用setSource，源点调用setTarget即可。

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

#### Make the implementations generic

#### Implement Graph.empty()

### Problem 4: Poetic walks

#### Test GraphPoet

#### Implement GraphPoet

#### Graph poetry slam

### 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。

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

### 基于在Poetic Walks中定义的Graph及其两种实现，重新实现Lab1中的 FriendshipGraph类。我们需要尽可能复用ConcreteEdgesGraph或 ConcreteVerticesGraph中已经实现的add()和set()方法，而不是从零开始。另外基于所选定的 ConcreteEdgesGraph 或 ConcreteVerticesGraph的rep来实现，而不能修改父类的rep

### FriendshipGraph类

实验poet的concertedges存在bug，无法实现lab2 的相关继承

### Person类

实验poet的concertedges存在bug，无法实现lab2 的相关继承

### 客户端main()

实验poet的concertedges存在bug，无法实现lab2 的相关继承

### 测试用例

实验poet的concertedges存在bug，无法实现lab2 的相关继承

### 提交至Git仓库

## Playing Chess

### ADT设计/实现方案

Piece，board，player，action，position，game

Piece定义棋子private String pieceName; //棋子名称

private int pieceState; //0未放置，1已放置。-1被remove

private int pieceX; //横坐标

private int pieceY; //纵坐标

board定义棋盘

private int boardSize; //棋盘大小

private Piece[][] boardPosition; //棋子在棋盘的位置

player定义选手名

private String playerName; //棋手名

private Set<Piece> remaining = new HashSet<Piece>(); //玩家剩余棋子

private String history = new String(); //操作历史

action类记录动作

private Board chessBoard = new Board();

position类记录棋子坐标

private int x; //横坐标

private int y; //纵坐标

Game

private String gameName; //游戏名

private Board gameBoard = new Board(); //棋盘

private Action gameAction = new Action(); //动作

private Player player1; //棋手1

private Player player2; //棋手2

### 主程序MyChessAndGoGame设计/实现方案

辅之以执行过程的截图，介绍主程序的设计和实现方案，特别是如何将用户在命令行输入的指令映射到各ADT的具体方法的执行。

### ADT和主程序的测试方案

介绍针对各ADT的各方法的测试方案和testing strategy。

介绍你如何对该应用进行测试用例的设计，以及具体的测试过程。

## Multi-Startup Set (MIT)

请自行设计目录结构。

注意：该任务为选做，不评判，不计分。

# 实验进度记录

请使用表格方式记录你的进度情况，以超过半小时的连续编程时间为一行。

每次结束编程时，请向该表格中增加一行。不要事后胡乱填写。

不要嫌烦，该表格可帮助你汇总你在每个任务上付出的时间和精力，发现自己不擅长的任务，后续有意识的弥补。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 3.24 | 下午3.00 | github下载程序包并使用Google逐条翻译MIT实验要求 | 圆满完成 |
| 3.30 | 晚上6.00 | 实验一完成graph和concertedges的设计 | 基本完成 |
| 4.3 | 下午2.00 | 解决之前出现的concertedge中迭代问题 | 无法完成 |
| 4.4 | 晚上5.10 | 实验二的dijskatra算法，发现需要直接继承实验一的接口，完善person和实验一debug | Debug失败 |
| 4.6 | 早晨至4.7凌晨6点 | 实验三 | 基本完成 |
| 4.7 | 下午1.30到4.00 | 实验报告 |  |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 接口的继承，  一个包中不同类和函数的相互调用 |  |
| Equal 不同的指向以及重写并向下转型 |  |
| Field的基本用法 |  |

# 实验过程中收获的经验、教训、感想

## 实验过程中收获的经验和教训

## 针对以下方面的感受

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. P1设计的ADT在多个应用场景下使用，这种复用带来什么好处？
5. P3要求你从0开始设计ADT并使用它们完成一个具体应用，你是否已适应从具体应用场景到ADT的“抽象映射”？相比起P1给出了ADT非常明确的rep和方法、ADT之间的逻辑关系，P3要求你自主设计这些内容，你的感受如何？
6. 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？
7. 关于本实验的工作量、难度、deadline。
8. 《软件构造》课程进展到目前，你对该课程有何体会和建议？