

**2020年春季学期  
计算机学院《软件构造》课程**

**Lab 1实验报告**

|  |  |
| --- | --- |
| 姓名 | 郭茁宁 |
| 学号 | 1183710109 |
| 班号 | 1837101 |
| 电子邮件 | gzn00417@foxmail.com |
| 手机号码 | 13905082373 |

# 目录

[1 实验目标概述 1](#_Toc34813848)

[2 实验环境配置 2](#_Toc34813849)

[3 实验过程 3](#_Toc34813850)

[3.1 Poetic Walks 3](#_Toc34813851)

[3.1.1 Get the code and prepare Git repository 3](#_Toc34813852)

[3.1.2 Problem 1: Test Graph <String> 3](#_Toc34813853)

[3.1.3 Problem 2: Implement Graph <String> 3](#_Toc34813854)

[3.1.3.1 Implement ConcreteEdgesGraph 4](#_Toc34813855)

[3.1.3.2 Implement ConcreteVerticesGraph 4](#_Toc34813856)

[3.1.4 Problem 3: Implement generic Graph<L> 4](#_Toc34813857)

[3.1.4.1 Make the implementations generic 4](#_Toc34813858)

[3.1.4.2 Implement Graph.empty() 4](#_Toc34813859)

[3.1.5 Problem 4: Poetic walks 4](#_Toc34813860)

[3.1.5.1 Test GraphPoet 4](#_Toc34813861)

[3.1.5.2 Implement GraphPoet 4](#_Toc34813862)

[3.1.5.3 Graph poetry slam 4](#_Toc34813863)

[3.1.6 Before you’re done 4](#_Toc34813864)

[3.2 Re-implement the Social Network in Lab1 4](#_Toc34813865)

[3.2.1 FriendshipGraph类 4](#_Toc34813866)

[3.2.2 Person类 4](#_Toc34813867)

[3.2.3 客户端main() 5](#_Toc34813868)

[3.2.4 测试用例 5](#_Toc34813869)

[3.2.5 提交至Git仓库 5](#_Toc34813870)

[3.3 Playing Chess 5](#_Toc34813871)

[3.3.1 ADT设计/实现方案 5](#_Toc34813872)

[3.3.2 主程序MyChessAndGoGame设计/实现方案 5](#_Toc34813873)

[3.3.3 ADT和主程序的测试方案 5](#_Toc34813874)

[4 实验进度记录 6](#_Toc34813875)

[5 实验过程中遇到的困难与解决途径 7](#_Toc34813876)

[6 实验过程中收获的经验、教训、感想 8](#_Toc34813877)

[6.1 实验过程中收获的经验和教训 8](#_Toc34813878)

[6.2 针对以下方面的感受 8](#_Toc34813879)

# 实验目标概述

根据实验手册简要撰写。

# 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。

在这里给出你的GitHub Lab2仓库的URL地址（Lab2-学号）。

# 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。

### Get the code and prepare Git repository

如何从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。

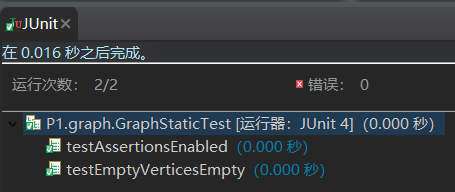
### Problem 1: Test Graph <String>

以下各部分，请按照MIT页面上相应部分的要求，逐项列出你的设计和实现思路/过程/结果。

    public static Graph<String> empty() {

        return new ConcreteEdgesGraph();

    }



### Problem 2: Implement Graph <String>

以下各部分，请按照MIT页面上相应部分的要求，逐项列出你的设计和实现思路/过程/结果。

#### Implement ConcreteEdgesGraph

#### Implement ConcreteVerticesGraph

### 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

在这里简要概述你对该任务的理解。

### FriendshipGraph类

给出你的设计和实现思路/过程/结果。

### Person类

给出你的设计和实现思路/过程/结果。

### 客户端main()

给出你的设计和实现思路/过程/结果。

### 测试用例

给出你的设计和实现思路/过程/结果。

### 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。

## Playing Chess

### ADT设计/实现方案

设计了哪些ADT（接口、类），各自的rep和实现，各自的mutability/ immutability说明、AF、RI、safety from rep exposure。

必要时请使用UML class diagram（请自学）描述你设计的各ADT间的关系。

### 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。

# 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 |
| 2020-03-09 | 晚上 | 初始化项目 | 完成 |
| 2020-03-10 | 中午 | Problem1 3.1.1-3.1.2 | 完成 |
| 2020-03-10 | 晚上 | Problem1 3.1.3.1 Edge Graph | 完成 |
| 2020-03-11 | 晚上 | Problem1 3.1.3.2 Vertex Graph | 完成 |
| 2020-03-12 | 下午 | 通过Graph Instance Test | 完成 |
| 2020-03-12 | 晚上 | Problem1 3.1.5 Graph Poet | 完成 |
| 2020-03-13 | 上午 | Problem1 3.1.5 Test | 完成 |
| 2020-03-13 | 上午 | Problem2 3.2整体完成 | 完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
|  |  |
|  |  |
|  |  |

# 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？