

**2021年春季学期**

**计算学部《软件构造》课程**

**Lab 2实验报告**

|  |  |
| --- | --- |
| 姓名 | 张瑞 |
| 学号 | 1190201421 |
| 班号 | 1936603 |
| 电子邮件 | 1190201421@stu.hit.edu.cn |
| 手机号码 | 15736059288 |

**目录**

[1 实验目标概述 3](#_Toc74519698)

[2 实验环境配置 3](#_Toc74519699)

[3 实验过程 3](#_Toc74519700)

[3.1 Poetic Walks 3](#_Toc74519701)

[3.1.1 Get the code and prepare Git repository 4](#_Toc74519702)

[3.1.2 Problem 1: Test Graph <String> 4](#_Toc74519703)

[3.1.3 Problem 2: Implement Graph <String> 4](#_Toc74519704)

[3.1.3.1 Implement ConcreteEdgesGraph 4](#_Toc74519705)

[3.1.3.2 Implement ConcreteVerticesGraph 5](#_Toc74519706)

[3.1.4 Problem 3: Implement generic Graph<L> 6](#_Toc74519707)

[3.1.4.1 Make the implementations generic 6](#_Toc74519708)

[3.1.4.2 Implement Graph.empty() 6](#_Toc74519709)

[3.1.5 Problem 4: Poetic walks 6](#_Toc74519710)

[3.1.5.1 Test GraphPoet 6](#_Toc74519711)

[3.1.5.2 Implement GraphPoet 7](#_Toc74519712)

[3.1.5.3 Graph poetry slam 7](#_Toc74519713)

[3.1.6 使用Eclemma检查测试的代码覆盖度 8](#_Toc74519714)

[3.1.7 Before you’re done 8](#_Toc74519715)

[3.2 Re-implement the Social Network in Lab1 9](#_Toc74519716)

[3.2.1 FriendshipGraph类 9](#_Toc74519717)

[3.2.2 Person类 9](#_Toc74519718)

[3.2.3 客户端main() 9](#_Toc74519719)

[3.2.4 测试用例 9](#_Toc74519720)

[3.2.5 提交至Git仓库 9](#_Toc74519721)

[4 实验进度记录 10](#_Toc74519722)

[5 实验过程中遇到的困难与解决途径 11](#_Toc74519723)

[6 实验过程中收获的经验、教训、感想 11](#_Toc74519724)

[6.1 实验过程中收获的经验和教训 11](#_Toc74519725)

[6.2 针对以下方面的感受 12](#_Toc74519726)

# 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

本次实验的环境之前已经搭建好，此处不再赘述。需要指出的是，本次实验需要在Eclipse IDE中安装配置EclEmma，按实验手册操作时发现Eclipse已经默认配置好了。

在这里给出你的GitHub Lab2仓库的URL地址（Lab2-学号）。

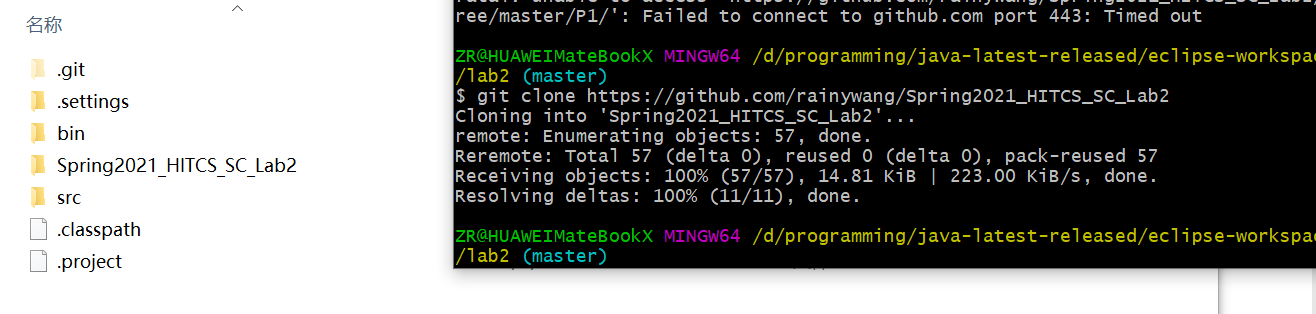
<https://github.com/ComputerScienceHIT/HIT-Lab2-1190201421>

# 实验过程

## Poetic Walks

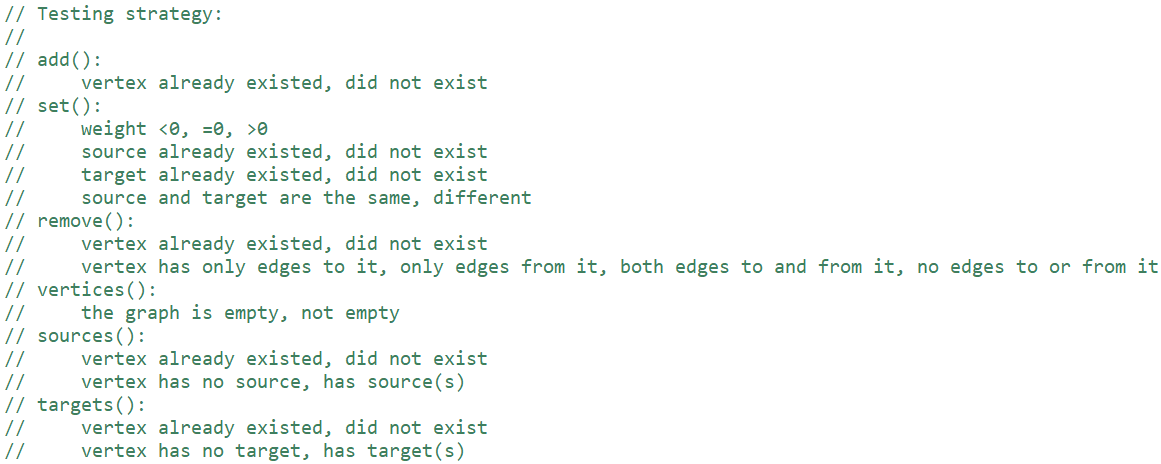
先用两个类ConcreteEdgesGraph和ConcreteVerticesGraph 分别实现Graph接口。然后将ADT泛型化，进行从String到L的转变。最后运用已经实现的Graph模型实现一个应用——Poetic walks。

### Get the code and prepare Git repository

先选好工作区的位置，然后在git bash中输入“git init”建立本地仓库，再“git clone <https://github.com/rainywang/Spring2021_HITCS_SC_Lab2>”即可获取远程仓库的代码：

### Problem 1: Test Graph <String>

本部分要求完成测试用例，其中GraphStaticTest的测试用例已经给出，需要完成的是GraphInstanceTest。我们需要对Graph中的各个实例方法都给出测试用例，首先根据规约进行等价类划分，写出测试策略：



此处的方法中，add()为creator，set()和remove()是mutator，vertices()、sources()和targets()是observer。

接下来按照“覆盖每个取值”的策略为每个方法构造测试用例，其中，对于creator和mutator，应该用observer来观察操作后的结果是否正确；对于observer，应该先用creator和mutator来改变对象再观察结果是否正确。

### Problem 2: Implement Graph <String>

#### Implement ConcreteEdgesGraph

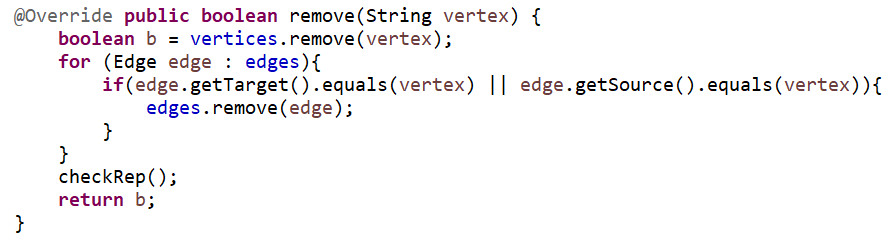
首先完成对Edge的设计，确定rep有source、target和weight，方法包含getSource()、getTarget()和getWeight()，并且重写toString()，给出Edge()的构造方法和checkRep方法。并完成spec、AF、RI、safety from Rep Exposure的撰写。

然后完成对Edge的测试用例的书写，先进行等价类划分，写出测试策略，然后按照“覆盖每个取值”的策略为每个方法构造测试用例。

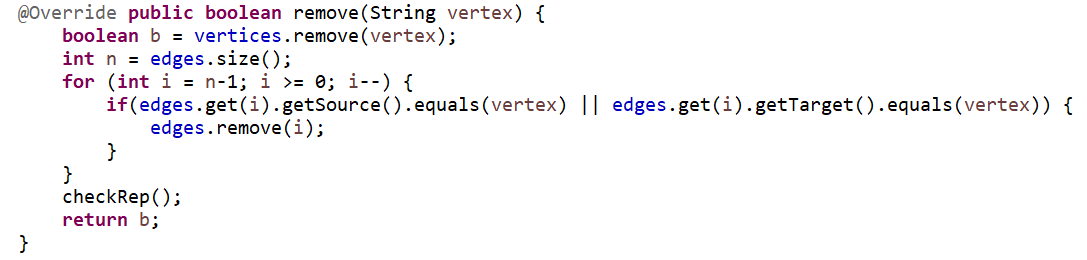
接下来完成对ConcreteEdgesGraph中重写的toString方法的测试用例。

最后完成ConcreteEdgesGraph的spec、AF、RI、safety from Rep Exposure，并实现Edge和ConcreteEdgesGraph。

值得注意的是，我在写代码过程中遇到了在for循环里进行删除操作进而引发错误的问题，该段错误代码如下：



错误的原因是删除了元素，但循环的计数会继续，从而导致遍历出现遗漏，没把该删的元素删完。最后我采用了按List内在顺序从大到小遍历的方法，避免删除操作带来遗漏：



#### Implement ConcreteVerticesGraph

首先完成对Vertex的设计，确定rep有label、sources和targets，方法包含setSource()、setTarget()、getLabel()、getSources()和getTargets()，并且重写toString()，给出Vertex()的构造方法和checkRep方法。并完成spec、AF、RI、safety from Rep Exposure的撰写。

然后完成对Vertex的测试用例的书写，先进行等价类划分，写出测试策略，然后按照“覆盖每个取值”的策略为每个方法构造测试用例。

接下来完成对ConcreteVerticesGraph中重写的toString方法的测试用例。

最后完成ConcreteVerticesGraph的spec、AF、RI、safety from Rep Exposure，并实现Vertex和ConcreteVerticesGraph。

有了实现ConcreteEdgesGraph和ConcreteEdgesGraphTest的经验，这一段任务完成得较为顺利，就是实现ConcreteVerticesGraph的方法的时候，发现好像更复杂了一点，还是花了不少时间。

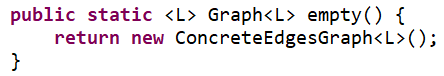
### Problem 3: Implement generic Graph<L>

#### Make the implementations generic

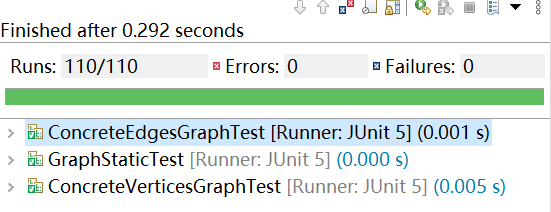
将前面已经实现的String类型改为泛型L，这一步十分简单，将String替换为L（注意不能替换toString部分输出时用到的String类型），将Vertex改为Vertex<L>，将Edge改为Edge<L>，ConcreteEdgesGraph变为ConcreteEdgesGraph<L>，ConcreteVerticesGraph变为ConcreteVerticesGraph<L>。

#### Implement Graph.empty()

我采用的是ConcreteEdgesGraph<L>来实现Graph<L>：



然后为了测试Graph<L>的正确性，我在GraphStaticTest中为IntegerGraph和DoubleGraph设计了几个简单的测试用例，最终所有测试用例都通过了：



### Problem 4: Poetic walks

#### Test GraphPoet

为了更方便测试，我在GraphPoet中添加了几个observer，分别是vertices()、sources()和targets()。

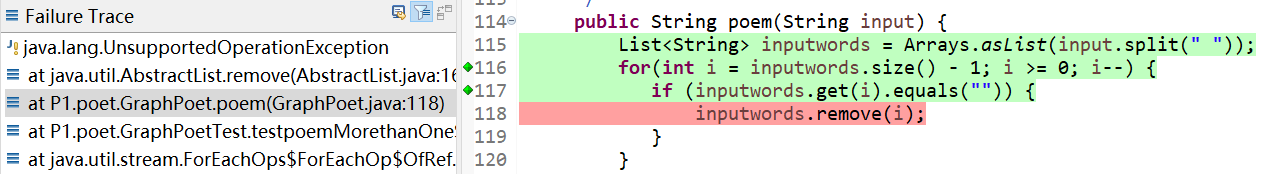
然后完成对GraphPoet的测试用例的书写，先进行等价类划分，写出测试策略，然后按照“覆盖每个取值”的策略为每个方法构造测试用例。

因为添加的vertices()、sources()和targets()，以及待实现的toString()我都准备直接采用GraphPoet中rep对应的graph内在的方法来实现，对于它们的正确性已在前面得到验证，故此处未再设计测试用例。

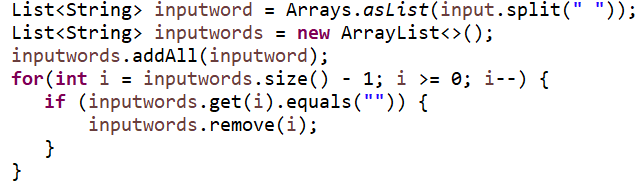
#### Implement GraphPoet

完成测试用例之后，接着完成GraphPoet的AF、RI、safety from Rep Exposure，并实现各个方法。

值得注意的是，我在写代码过程中遇到了下列报错：

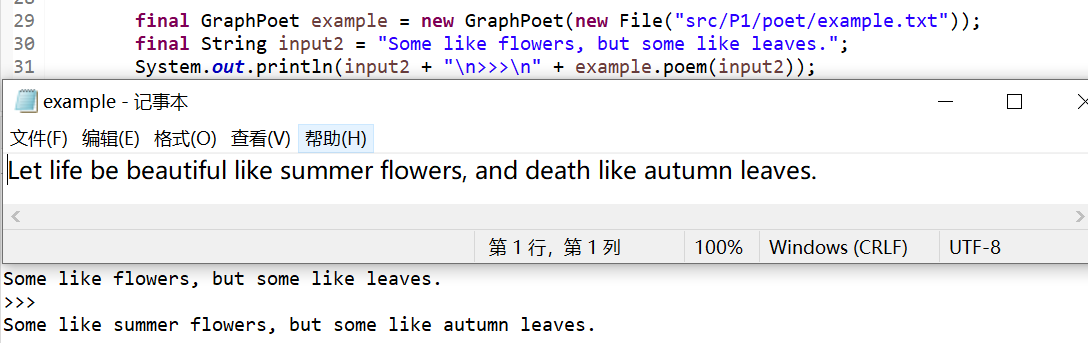


经查阅资料发现，Arrays.asList转换得到的ArrayList继承了AbstractList类，而这个类的set、remove、add方法都定义为了“throw new UnsupportedOperationException();”，故不能进行删除操作，然后我通过再创建一个List的办法解决了此问题：

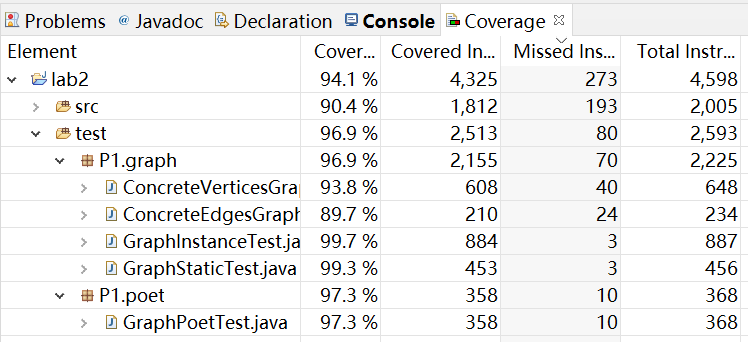


#### Graph poetry slam

简单引用了泰戈尔的一句话构造了一个句子：



### 使用Eclemma检查测试的代码覆盖度



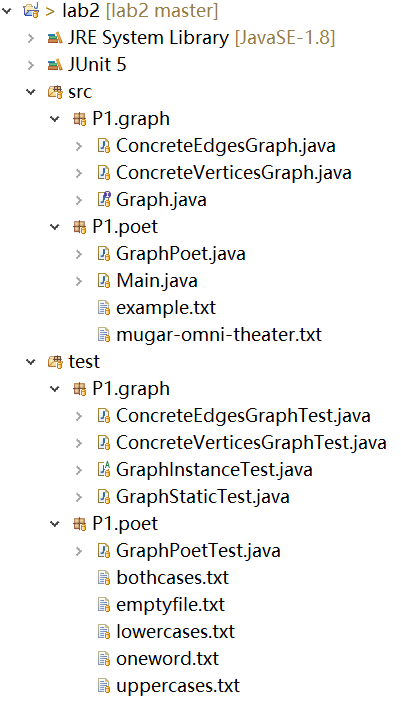
### Before you’re done

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/)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。

先进入工作区的位置，然后在git bash中输入“git init”建立本地仓库，再输入“git remote add lab2 <https://github.com/ComputerScienceHIT/HIT-Lab2-1190201421>”添加远程仓库，然后输入“git add <filename>”,“git commit -m “\*\*\*””,“git push -u lab2 master”即可向远程仓库提交代码。

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

运用已经实现的Graph模型重新实现Lab1中的FriendshipGraph。

### FriendshipGraph类

Lab1中已经有了FriendshipGraph的一个实现，但这里需要复用已经设计出来的Graph<L>，故进行了一定修改。Rep直接使用Graph<L>，这里我的Graph<L>选择的是ConcreteEdgesGraph<L>的实现方式。至于方法，除了要求的addVertex、addEdge和getDistance，我还加入了checkRep、getPeople和getFriends，其中getPeople用于查看graph中的Person，getFriends用于查看Person a的朋友（单向关系）。

然后按照ADT的设计方法，完善了Lab1中的FriendshipGraph，写出所有的spec、AF、RI、safety from Rep Exposure。

### Person类

同样的，Lab1中也已经有了Person的一个实现，此处继续使用这个类。但由于朋友关系能交给graph去记录，故略去了Person中和friend相关的内容。同时，由于Person是一个immutable的类，我重写了equals、hashCode和toString，还写了spec、AF、RI、safety from Rep Exposure，实现了checkRep方法。

### 客户端main()

此处按实验手册照搬Lab1中的代码。

### 测试用例

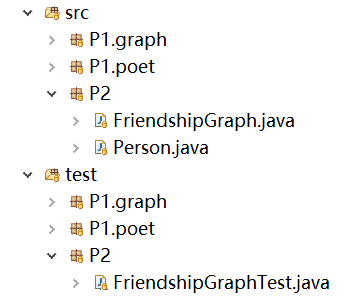
我先是按实验手册要求照搬了Lab1中的测试用例，后来发现当时设计的测试用例覆盖度不够，就对FriendshipGraph中各个方法进行等价类划分，写出测试策略，然后按照“覆盖每个取值”的策略为每个方法构造测试用例。

### 提交至Git仓库

如何通过Git提交当前版本到GitHub上你的Lab2仓库。

先进入工作区的位置，然后在git bash中输入“git init”建立本地仓库，再输入“git remote add lab2 <https://github.com/ComputerScienceHIT/HIT-Lab2-1190201421>”添加远程仓库，然后输入“git add <filename>”,“git commit -m “\*\*\*””,“git push -u lab2 master”即可向远程仓库提交代码。

在这里给出你的项目的目录结构树状示意图。



# 实验进度记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 2021-5-25 | 14:00-15:30 | 完成P1的Problem1 | lab2相关内容尚未讲授，没太弄明白，未能完成 |
| 2021-6-1 | 13:45-15:30 | 继续完成P1的Problem1 | 按计划完成 |
| 2021-6-8 | 13:45-15:30 | 重新修改已完成的P1的Problem1，完成ConcreteEdgesGraph的Edge的设计与测试用例 | 按计划完成 |
| 2021-6-9 | 10:00-12:00 | 完成ConcreteEdgesGraph | 基本完成，有部分测试用例未通过 |
| 2021-6-9 | 15:00-18:00 | 修复ConcreteEdgesGraph中的问题，完成ConcreteVerticesGraph中的Vertex的设计与测试用例 | 按计划完成 |
| 2021-6-10 | 13:00-16:00 | 完成ConcreteVerticesGraph | 基本完成，有部分测试用例未通过 |
| 2021-6-10 | 16:00-18:00 | 修复ConcreteVerticesGraph中的问题，完成Problem3 | 按计划完成 |
| 2021-6-11 | 10:00-11:45 | 完成GraphPoet的设计与测试用例 | 按计划完成 |
| 2021-6-11 | 16:00-19:00 | 完成Poetic walks | 完成，但是实现GraphPoet后发现还需进一步完善测试用例 |
| 2021-6-11 | 20:00-22:00 | 完成P1 | 按计划完成 |
| 2021-6-12 | 19:00-20:00 | 完成P2 | 完成了实验手册上的任务，但添加测试用例额外花了时间 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 对于一些Java的类不熟悉 | 模仿老师已给代码，还不会的就上网搜索 |
| Git的加载是个问题，不太稳定 | 反复加载，坚持不懈 |
| 不会build | 问天问地，抱大佬大腿 |

# 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

在本次实验中对Java更加的熟悉了，代码能力有一定提升，也学会了更多关于Java类的细节。而且我还是把在循环遍历中删除元素的坑踩了一遍，对这个点的印象更加深刻了。此外，我还学习到了Arrays.asList转换得到的ArrayList继承了AbstractList类，而这个类的set、remove、add方法都定义为了“throw new UnsupportedOperationException();”，故不能进行增加和删除操最后，我还是没克服喜欢拖延，把事情堆到ddl快到了再开始赶的恶习，下次一定提前安排好进度。

## 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？

前者更为抽象，设计出来的成果能在多个应用场景下复用，二次开发的成本低；后者虽然更具体，但一旦有类似场景需要再设计时，重复工作较多。

1. 使用泛型和不使用泛型的编程，对你来说有何差异？

泛型理解起来更抽象一点，但代入一个具体的类型还是很好理解的。而且使用泛型编程方便以后的复用。

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？

能考虑到更全面的测试策略，思路更清晰。我还比较适应，就是写测试用例写得有点崩溃。

1. 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？

能利用已有成果，迅速完成多个场景的应用。一次开发，多次复用，效率更高。

1. 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？

什么P3？本次实验无P3。（但是下次就有了，希望我能适应……）

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？

既能使他人更容易读懂自己的代码，也能使自己以后能快速回想起设计与实现的思路。时刻注意程序的正确性，同时保证安全性，不会被客户端恶意破坏。

1. 关于本实验的工作量、难度、deadline。

比起Lab1工作量更大、难度有所提升，但也还在可承受范围内，deadline较为合理，虽然我懒，又拖延了。

1. 《软件构造》课程进展到目前，你对该课程有何体会和建议？

感觉还是能提升不少编程能力，让自己能编写出更高质量的程序。问题就是学时太少，导致实验时间较紧张，而且把课程和考试月排在一起压力较大。