

**2021年春季学期  
计算学部《软件构造》课程**

**Lab 2实验报告**

|  |  |
| --- | --- |
| 姓名 | 陈一帆 |
| 学号 | 1191000606 |
| 班号 | 1903003 |
| 电子邮件 | 1191000606@stu.hit.edu.cn |
| 手机号码 | 13972934654 |

**目录**

[1 实验目标概述 1](#_Toc72249750)

[2 实验环境配置 1](#_Toc72249751)

[3 实验过程 1](#_Toc72249752)

[3.1 Poetic Walks 2](#_Toc72249753)

[3.1.1 Get the code and prepare Git repository 2](#_Toc72249754)

[3.1.2 Problem 1: Test Graph <String> 2](#_Toc72249755)

[3.1.3 Problem 2: Implement Graph <String> 4](#_Toc72249756)

[3.1.3.1 Implement ConcreteEdgesGraph 4](#_Toc72249757)

[3.1.3.2 Implement ConcreteVerticesGraph 5](#_Toc72249758)

[3.1.4 Problem 3: Implement generic Graph<L> 6](#_Toc72249759)

[3.1.4.1 Make the implementations generic 6](#_Toc72249760)

[3.1.4.2 Implement Graph.empty() 6](#_Toc72249761)

[3.1.5 Problem 4: Poetic walks 6](#_Toc72249762)

[3.1.5.1 Test GraphPoet 6](#_Toc72249763)

[3.1.5.2 Implement GraphPoet 7](#_Toc72249764)

[3.1.5.3 Graph poetry slam 7](#_Toc72249765)

[3.1.6 使用Eclemma检查测试的代码覆盖度 7](#_Toc72249766)

[3.1.7 Before you’re done 9](#_Toc72249767)

[3.2 Re-implement the Social Network in Lab1 10](#_Toc72249768)

[3.2.1 FriendshipGraph类 10](#_Toc72249769)

[3.2.2 Person类 10](#_Toc72249770)

[3.2.3 客户端main() 10](#_Toc72249771)

[3.2.4 测试用例 10](#_Toc72249772)

[3.2.5 提交至Git仓库 11](#_Toc72249773)

[4 实验进度记录 11](#_Toc72249774)

[5 实验过程中遇到的困难与解决途径 12](#_Toc72249775)

[6 实验过程中收获的经验、教训、感想 12](#_Toc72249776)

[6.1 实验过程中收获的经验和教训 12](#_Toc72249777)

[6.2 针对以下方面的感受 12](#_Toc72249778)

# 实验目标概述

1. 掌握ADT的设计，理解OOP
2. 学会ADT规约，测试，AF，RI，防止表示泄露
3. 初步使用泛型

# 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。

1. 硬件环境：X64CPU；1.6GHz；8G RAM；512G SSD
2. 软件环境：Ubuntu 18.04.5 LTS 64位
3. 开发工具：IDEA
4. 配置过程中遇到的困难

将github账户的用户名username改为1191000606，然后连接不上远程仓库了，每一次push都要重新连接。最后还是每一次上传的时候都使用token。

1. GitHub Lab2仓库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

对问题的理解：首先要根据给出的接口graph实现两个实现类，分别是基于边的类和基于点的类，然后使用实现的类完成Poet Graph中的两个函数。

实验过程如下：

### Get the code and prepare Git repository

下载即可。由于实验一的经历，这一次拉取远程代码到本地比较顺利，调正目录结构也并无问题。

### Problem 1: Test Graph <String>

问题的理解

首先阅读MIT网站上的要求，即测试静态测试，同时书写动态测试。

问题的求解

首先阅读代码，理解整个代码的结构，继承关系等。重点阅读test文件中Graph部分，其中共有四个test文件，阅读其中已有代码和Todo了解到：

1. GraphStaticTest文件是用来测试接口中的静态方法的，即为empty()。
2. GraphhInstanceTest是用来测试接口中的非静态方法的，即那些需要实例化之后才可以调用的方法，其中包括接口中的所有非静态方法，同时它也是ConcreteVerticesGraphTest，ConcreteEdgesGraphTest的父类。
3. ConcreteVerticesGraphTest是针对实现类ConcreteVerticesGraph的测试，它继承了GraphhInstanceTest中所有的方法，因此接口中的方法的测试不用在两个实现类的测试中出现两次，代码的复用。同时它也测试一些GraphhVertexGraph中特有的方法，比如toString，以及Vertex中的方法。注意在这个测试文件中要使用GraphhVertexGraph实现来重写emptyInstance()，使得测试用例中的图均由ConcreteVertexGraph实现类实现。
4. ConcreteEdgeVertexTest，同上。

设计思路与求解过程

GraphhStaticTest已经写好，重点在GraphhInstanceTest。

首先设计addTest，使用emptyInstance()获得一个空的图，然后向其中加入顶点。然后划分等价类。顶点中的label可以为null，空字符串，中文字符；加入时图中可分为有/没有这个顶点。然后根据这两组等价类的划分书写测试用例。

setTest等价类的划分有：源点label为null，空字符串，中文字符；终点label可以为null，空字符串，中文字符；源点存在/不存在于图中；终点存在/不存在与图中；权重设为0/正整数；根据这几组等价类，书写测试。

removeTest等价类划分有：顶点label为/不为null；存在/不存在于图中；存在的时候，与其相连的边的多少。

vertexTest等价类的划分有：顶点集的大小；其中特殊的label(如null，空字符串，中文字符)能否顺利打印。

sourceTest，targetTest：等价类的划分较为简单，均为相连顶点数的多少；是否存在；相连的顶点中label特殊情况时。

至此完成了InstanceTest，其两个子类的测试文件待到设计时再书写。

### Problem 2: Implement Graph <String>

#### Implement ConcreteEdgesGraph

Edge类的设计

包含源点，终点的label，以及权重，设计为immutable，不存在任何mutable方法，同时设计了hashCode，Equals方法。toString返回的是源点，终点，以及权重的字符串。

由于Edge是immutable类，因此为了后续比较的方便，重写了hashCode，equals方法，考虑到label可能为null,因此当label为null的时候返回-1。equals方法，判断相等的条件是三个属性均相等。

ConcreteGraph的属性的设计：

已经给出，有点集和边集。

ConcreteGraph的各个方法的设计：

1. add,首先判断这个label是否存在于点集中，是的话返回否，并不改变graph。不是的话加入其中。
2. set，首先判断要设置为weight是否等于0。等于0的话，如果源点，终点有一个点不包含在点集中，那么这一条边肯定也不在图中，返回即可。如果两者都在，那么取其中某一点的所有相连的顶点和与他们的距离。

由于后期经常使用这个函数，因此设计一个私有方法findLinkedVertex，给定一个label，如果这个label在点集中，那么返回所有与其相连的顶点和距离，以Map的形式。不在的话返回null。

取出之后看另外一个点是否在返回的Map中，在的话则返回原来的距离长度（这个距离是正的），并将原来的边从边集中删除掉。注意删除掉之后就算点没有链接的边了也不能删掉。

如果weight不为0的话，由于规约加强了，weight可以为负，表示将输入的参数source，target颠倒一下，然后如果两点有一点不存在于点集中，那么说明边集中也不存在要删除边，直接加入即可，然后为方便，将两点都add到点集中。如果有边的话，则删除原边，加入新边，返回原边的权重。

1. remove，首先判断有没有，然后在边集中删除相连的边，点集中删除点。
2. vertices，记得defensive copy。
3. source,直接调用之前的方法即可。
4. target，直接调用之前的方法即可。
5. toString，打印所有的边集和点集。

#### Implement ConcreteVerticesGraph

Vertex类的设计

关于vertex类属性的设计，为了避免重复，我初步设计是在其中加入一个static字段，记录所有用来实例化vertex对象的String对象。后来发现，有不同的图同时使用这个类，如果全部放在vertex类中会区分不开，因此放弃了这个想法。

于是属性只有label，还有一个与之相连的键值对，键为Vertex，值为Integer，表示所有与这个顶点相连的点，值可以为正/负整数，表示边的方向，显然一条边会有两个顶点保存其信息，这也是checkRep要检查的点。

我认为这里还有值得商讨的方法，比如将键值对中的键换为String会如何，限于时间原因留待日后探讨。

Vertex方法的设计，显然Vertex设计是一个mutable的方法，按照课上PPT，建议使用行为等价性，但是后续会将其放入Hash类的Collections中，因此我们仍然重写了hashCode，equals方法。关于hashcode的设计，其取决于Vertex中一个不变的属性，那么对象发生一定的改变导致hash类Collections无法找到这个对象的问题不会存在。equals则判断其全部属性是否相同。

考虑到Vertex会保存边的改变，为其设计了一个方法对内部的边进行操作。

ConcreteVertexGraph的设计

类似于ConcreteEdgesGraph。

仅说明其中不一样的部分，关于set的设计，先看两个label在不在，如果一个不在且weight为0，那么什么都不用做直接返回，不为0的话加进去。后续略。

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

#### Make the implementations generic

将所有的String改为L,除了toString部分。注意此时Vertex，Edge也是基于L的类，也要变成Vertex<L>，Edge<L>。

#### Implement Graph.empty()

使用不同的immutable类作为label，如Integer，Character。

### Problem 4: Poetic walks

#### Test GraphPoet

首先书写Test，其中有三个方法的测试，构造方法，poem方法，观察方法toString。

构造方法，等价类的划分：输入文件为空，一行，多行。

poem方法：等价类的划分：简单和复杂

toString方法：等价类的划分：输入文件空，一行，多行。

#### Implement GraphPoet

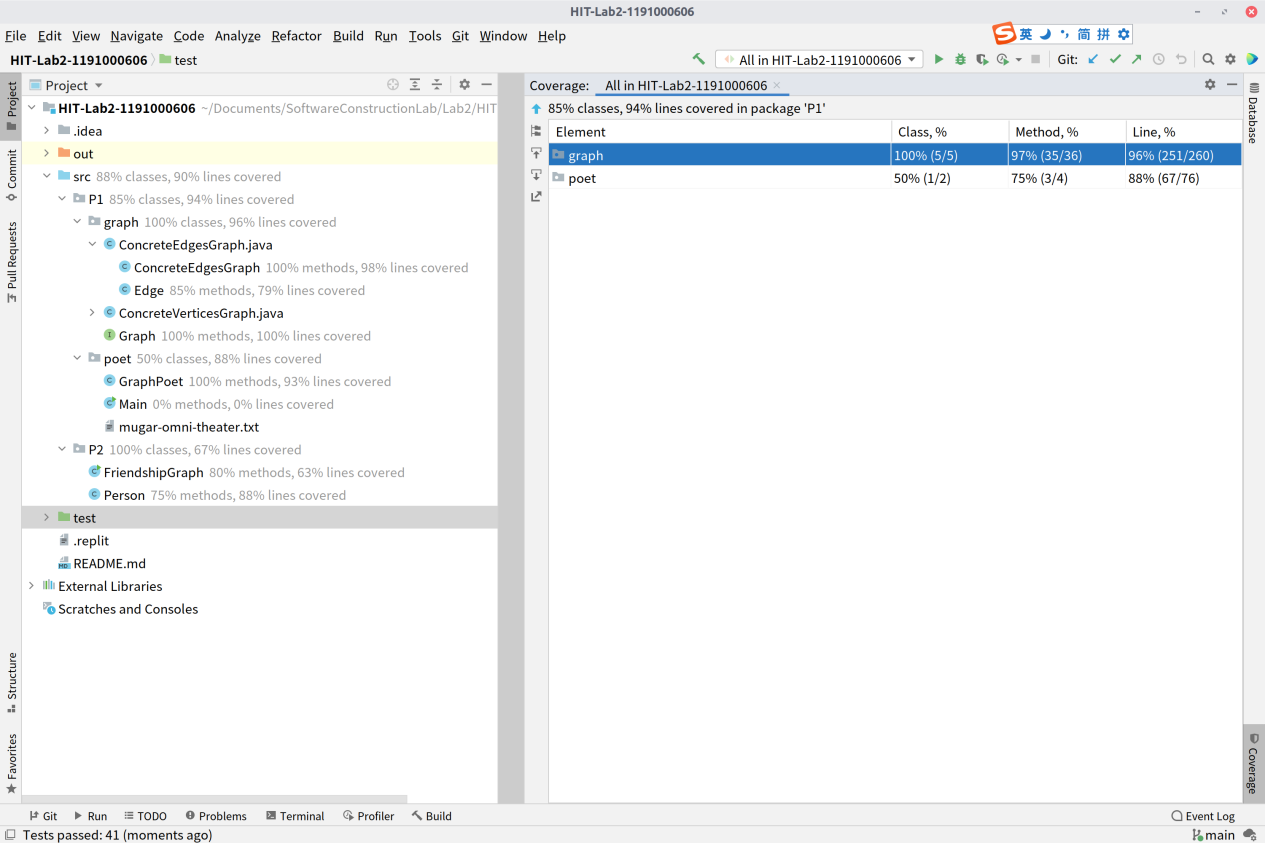
设计思路：打开文件，每次读取一行，如果长度为0，则读取下一行，为1，则将唯一的一个放进去，读取下一行。

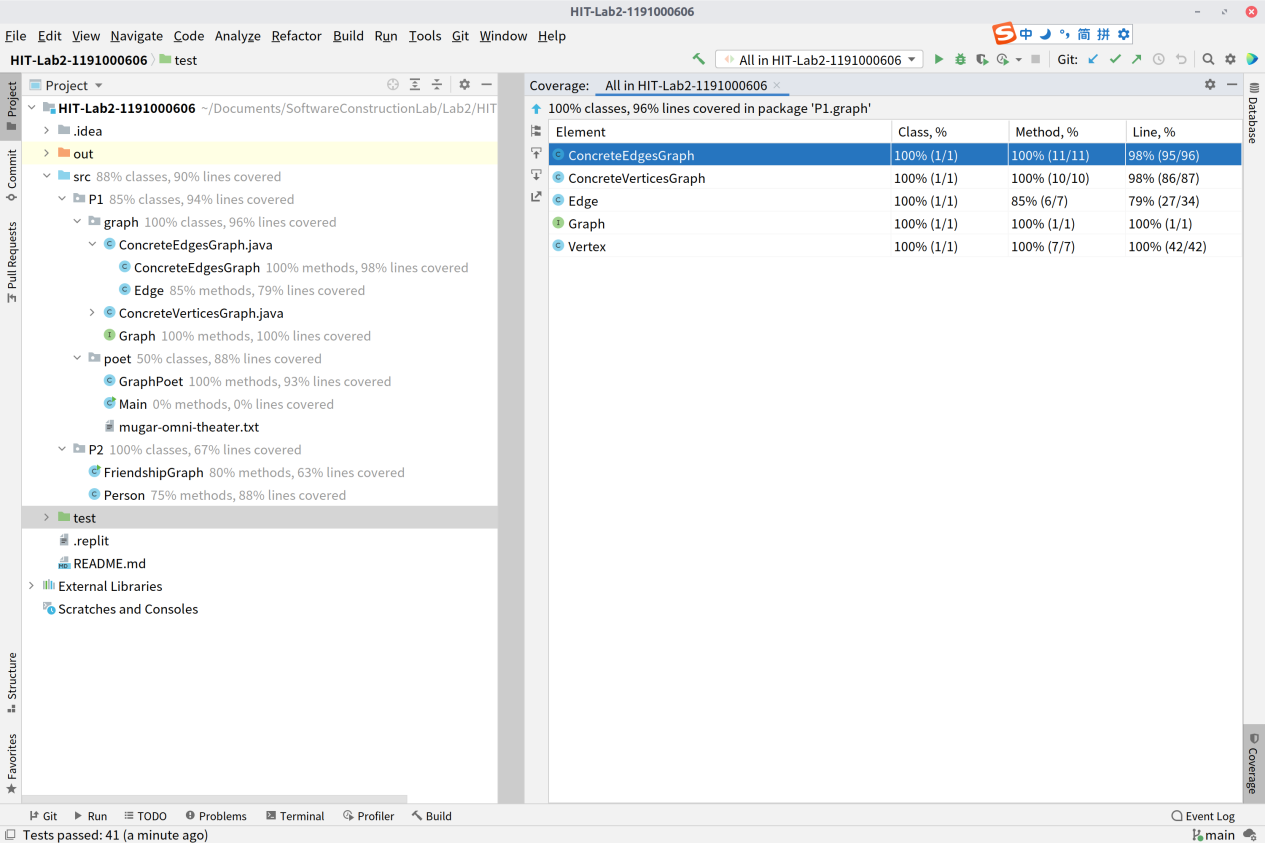
如果一行长度大于1，那么首先以“ ”为分割符，分割为多个String，然后将第一个放入其中。从第二个开始循环，首先将当前String放入其中，如果之前不存在这个顶点，便加入一条边（使用set）,从前一个字符到当前字符。如果之前存在，便判断一下上一个点与当前点之前是否有边，有的话若方向相同则权重加一，不同设为1。没有的话则直接设置。

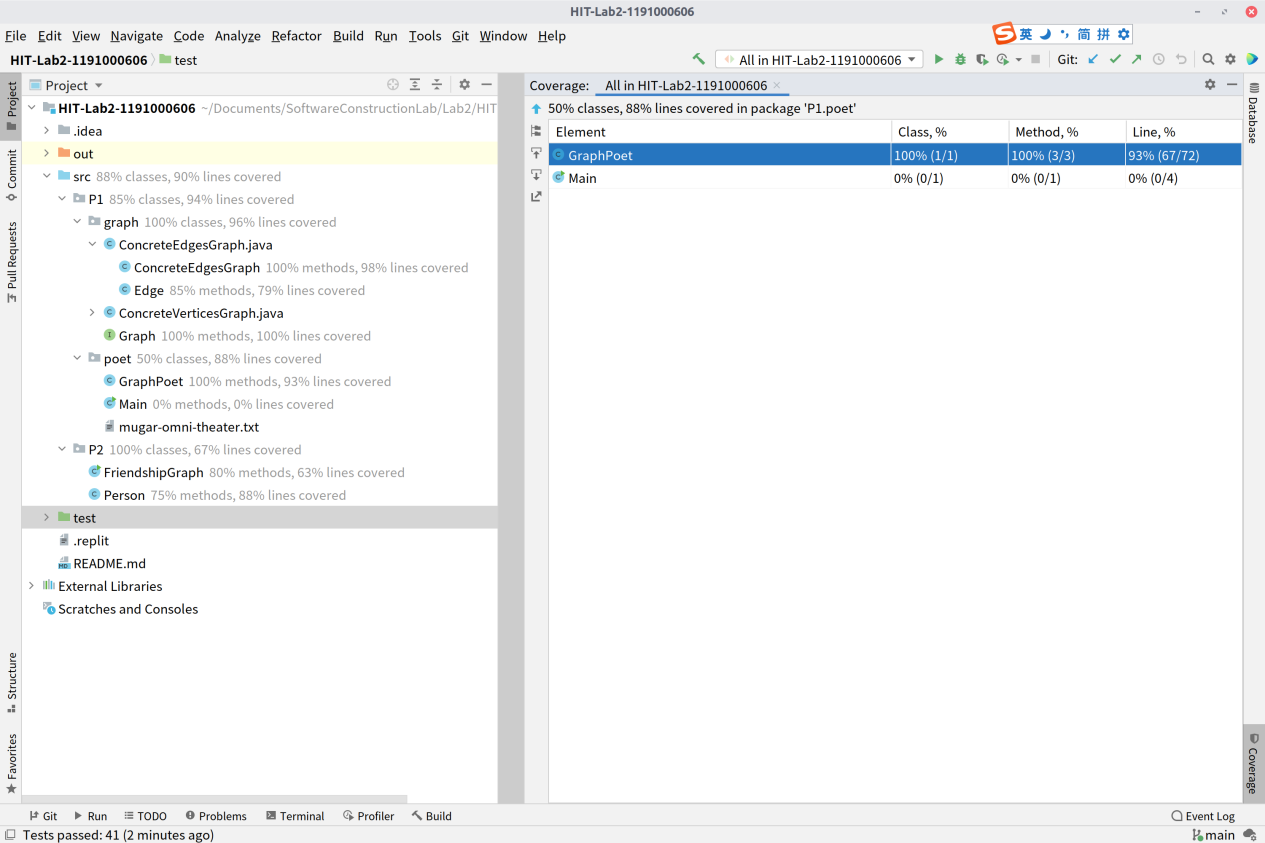
#### Graph poetry slam

同样采用某一个到下一个的办法，不过判断当前这个不在之后，便直接到下一个。当前这个在，下一个不在，那下标直接加2。仅此而已。具体见源代码。

### 使用Eclemma检查测试的代码覆盖度







### Before you’re done

检查程序：No error, no warning.

使用Token验证提交到Github。

项目的目录结构树状示意图。

HIT-Lab2-1191000606

doc

Lab2-1191000606-Report.docx

lib

hamcrest-core-1.3.jar

junit-4.13.1.jar

src

P1

graph

ConcreteEdgeGraph.java

ConcreteVerticesGraph.java

Graph.java

poet

GraphPoet.java

Main.java

mugar-omni-theater.txt

P2

FriendshipGraph.java

Person.java

test

P1

graph

ConcreteEdgeGraphTest.java

ConcreteVerticesGraphTest.java

GraphInstanceTest.java

GraphStaticTest.java

poet

GraphPoetTest.java

empty.txt

oneLine.txt

manyLines.txt

complexLines.txt

P2

FriendshipGraphTest.java

## Re-implement the Social Network in Lab1

与Lab1相比，这个产生了变化，因为这里给出的Graph.java接口是有向有权图，而原来的图是无向图，并且没有权重。因此将之修改为有向图，权重全部设置为1。

### FriendshipGraph类

直接使用Graph接口，使用任意一个实现类，对于addVertex方法，直接调用add方法，如果已经存在，那么就按照实验指导要求直接报错退出。addEdge方法，直接调用set方法。getDistance方法，仍然采用广度优先搜索，使用队列结构。然后加一个观察方法用于测试。

### Person类

只有一个属性label。重写hashCode，equals方法，提供getter方法。

### 客户端main()

使用实验指导已经给出的客户端代码。

### 测试用例

addVertexTest：等价类的划分：给定的label为空，混合字符，中文字符

addEdgeTest：等价类的划分：无。

getDistanceTest：等价类的划分：顶点集的大小：空，较少，多。边集的大小：空，较少，多。

getGraphTest：测试是否做到defensive copy。

### 提交至Git仓库

使用Token提交至github。

项目目录结构示意图见上文。

# 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 |
| 6月2号 | 10：00-12：00 | 阅读网页，理解任务，构思 | 完成 |
| 6月2号 | 18：00-21：00 | 撰写ConcreteVertexGraph，复习概念 | 完成Vertex |
| 6月3号 | 14：00-16：00 | 撰写ConcreteVertexGraph，复习概念 | 完成小部分 |
| 6月4号 | 13：45-23：59 | 撰写ConcreteVertexGraph，复习概念 | 完成大部分 |
| 6月5号 | 13：45-23:59 | 撰写ConcreteVertexGraph,  ConcreteEdgeGraph | 完成 |
| 6月6号 | 9：45-17：30 | 修改完善实现类 | 完成 |
| 6月7号 | 13：45-4：00 | 撰写Poet部分 | 完成 |
| 6月9号 | 13：45-23：59 | 撰写P2部分 | 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 |
| 修改username之后，github验证仍然经常出错 | commit到本地，在网络较好的时候，使用Token验证一次Push。 |
| hashCode与equals的理解 | 查阅资料，询问求助，课堂学习。 |
| ADT设计时的取舍 | 与人交流。 |

# 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

写代码时碰到问题要查资料，查资料有所得就写博客。这样的话理解能深刻不少。

想要知道方法好不好，是不是使用了更多的内存，写一段代码试一试就好了。然后再去学其中的原理。

学习Java底层很有必要，并且最好看书，更有体系。

## 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？

面向ADT编程比较繁琐，但如果做好复用也尚能接受。将属性方法打包放在一起，有助于日后复用理解。

1. 使用泛型和不使用泛型的编程，对你来说有何差异？

使用泛型编程更加具有普适性，离开了具体类型更抽象了，更能锻炼抽象思维的能里。

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？

方便测试，debug，先写测试用例再编程能能方便快捷，需求也更明确。比较适应这种测试方法。

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？

代码更加简洁，减少无用劳动，提高编程效率。

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？

没有P3。

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？

提醒自己注意不要表示泄露等，同时方便自己日后，他人阅读。

愿意。

1. 关于本实验的工作量、难度、deadline。

工作量大，难度低，过多的注释极其繁琐，感觉杀鸡用牛刀。时间安排还算合理

1. 《软件构造》课程进展到目前，你对该课程有何体会和建议？

建议书写大规模的软件代码练手，在实际中体会课上所学，而不是对着一百多行的代码写各种注释。