

**2021年春季学期  
计算学部《软件构造》课程**

**Lab 3实验报告**

|  |  |
| --- | --- |
| 姓名 | 彭明翼 |
| 学号 | 1190201927 |
| 班号 | 1903008 |
| 电子邮件 | [1601690791@qq.com](mailto:1601690791@qq.com) |
| 手机号码 | 15298225260 |

**目录**

[1 实验目标概述 1](#_Toc74666198)

[2 实验环境配置 1](#_Toc74666199)

[3 实验过程 1](#_Toc74666200)

[3.1 待开发的三个应用场景 1](#_Toc74666201)

[3.2 面向可复用性和可维护性的设计：IntervalSet<L> 1](#_Toc74666202)

[3.2.1 IntervalSet<L>的共性操作 1](#_Toc74666203)

时

[3.2.3 面向各应用的IntervalSet子类型设计（个性化特征的设计方案） 2](#_Toc74666205)

[3.3 面向可复用性和可维护性的设计：MultiIntervalSet<L> 2](#_Toc74666206)

[3.3.1 MultiIntervalSet<L>的共性操作 2](#_Toc74666207)

[3.3.2 局部共性特征的设计方案 2](#_Toc74666208)

[3.3.3 面向各应用的MultiIntervalSet子类型设计（个性化特征的设计方案） 2](#_Toc74666209)

[3.4 面向复用的设计：L 2](#_Toc74666210)

[3.5 可复用API设计 2](#_Toc74666211)

[3.5.1 计算相似度 2](#_Toc74666212)

[3.5.2 计算时间冲突比例 2](#_Toc74666213)

[3.5.3 计算空闲时间比例 2](#_Toc74666214)

[3.6 应用设计与开发 2](#_Toc74666215)

[3.6.1 排班管理系统 3](#_Toc74666216)

[3.6.2 操作系统的进程调度管理系统 3](#_Toc74666217)

[3.6.3 课表管理系统 3](#_Toc74666218)

[3.7 基于语法的数据读入 3](#_Toc74666219)

[3.8 应对面临的新变化 3](#_Toc74666220)

[3.8.1 变化1 3](#_Toc74666221)

[3.8.2 变化2 3](#_Toc74666222)

[3.9 Git仓库结构 3](#_Toc74666223)

[4 实验进度记录 3](#_Toc74666224)

[5 实验过程中遇到的困难与解决途径 4](#_Toc74666225)

[6 实验过程中收获的经验、教训、感想 4](#_Toc74666226)

[6.1 实验过程中收获的经验和教训 4](#_Toc74666227)

[6.2 针对以下方面的感受 4](#_Toc74666228)

# 实验目标概述

本次实验覆盖课程第前两次课的内容，目标是编写具有可复用性和可维护性的软件，主要使用以下软件构造技术：

* 子类型、泛型、多态、重写、重载
* 继承、代理、组合
* 常见的OO设计模式
* 语法驱动的编程、正则表达式
* 基于状态的编程
* API设计、API复用

本次实验给定了三个具体应用（值班表管理、操作系统进程调度管理、大学课表管理），学生不是直接针对每个应用分别编程实现，而是通过ADT和泛型等抽象技术，开发一套可复用的ADT及其实现，充分考虑这些应用之间的相似性和差异性，使ADT有更大程度的复用（可复用性）和更容易面向各种变化（可维护性）。

# 实验环境配置

使用Eclipse即可。

https://github.com/hit-pengmignyi/HIT-lab3-1190201927.git

# 实验过程

请仔细对照实验手册，针对每一项任务，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！）。

## 待开发的三个应用场景

简要介绍三个应用。

分析三个应用场景的异同，理解需求：它们在哪些方面有共性、哪些方面有差异。

排班表需要在指定日期范围内给每一天都安排上值班人，值班人只在连续的K天值班，并且不会在一个值班任务阶段结束后再次值班。

进程调度是模拟CPU随机调度后台进程，这里的CPU一个时段只能处理一个进程的分片，可以有CPU闲置时段。

课程表是设置一个星期内的课程任务，每一个课程都可以出现多次，不同的课程也可以在同一个时段出现。

应用场景一是**排班表**：要求是在时间轴上没有周期的，没有空隙的，不多次使用同一标签的，同一时段只能有一个标签的任务，。

应用场景二是**进程调度**：要求是在时间轴上没有周期的，可以有空隙的，可以多次使用同一标签的，同一时段只能有一个标签的任务。

应用场景三是**课程表**：要求是在时间轴上有周期的（周期为7天），可以有空隙的，可以多次使用同一标签的，同一时段可以有多个标签的任务。

排班表需要时间轴的起始终止时间；进程调度起始时间为0，终止时间在所有进程终止时终止（运行时获得）；课程表需要起始终止时间。在时间轴起始终止时间上的设计需要在各自app完成。

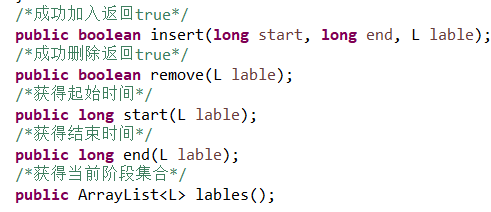
三者都是对一个<start\_time,Lable,end\_time>的序列的实现，这个可以集合成IntervalSet<L>。

进程调度与课程表都可以有空隙，重点是可以多次使用同一标签，这二者可以集合成MultiIntervalSet<L>。

## 面向可复用性和可维护性的设计：IntervalSet<L>

该节是本实验的核心部分。

### IntervalSet<L>的共性操作



IntervalSet<L>需要实现的功能包括<start,Lable,end>（即一个时段period）三元组插入，可以根据lable返回对应的起止时间（当时间轴上有多个lable时不可用，需要在multiIntervalSet<L>中实现），另外就是返回当前轴上的所有标签。

### 局部共性特征的设计方案

Insert操作：查找实现类中的period，根据需要是否有重复lable来确定能否插入，其他的情况可以直接插入，对于排班表与进程调度，可以按照时间先后插入，这样有利于显示。

Remove操作：遍历实现类period中所有标签，不论是否允许lable重复，将所有与传入lable相同的lable都删除即可。

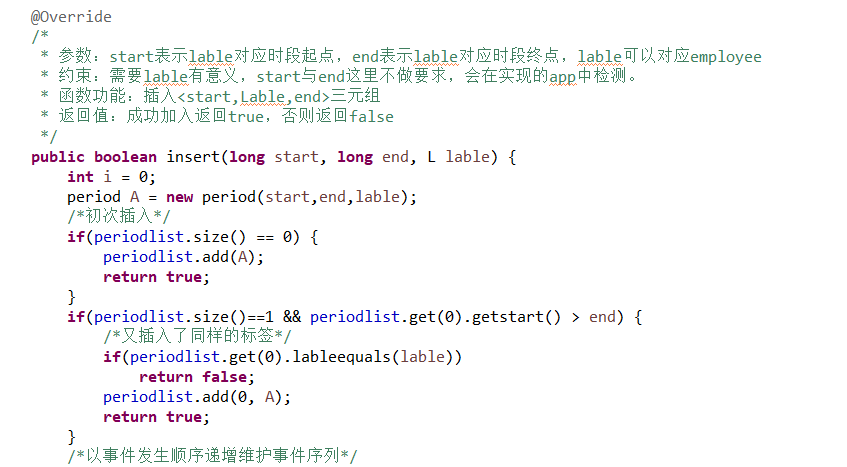
Start操作：遍历实现类所有的period，将第一个匹配的start返回。

End操作：遍历实现类所有的period，将第一个匹配的end返回。

Lables操作：申请一个ArrayList，将实现类period的所有标签加入后返回ArrayList。

### 面向各应用的IntervalSet子类型设计（个性化特征的设计方案）

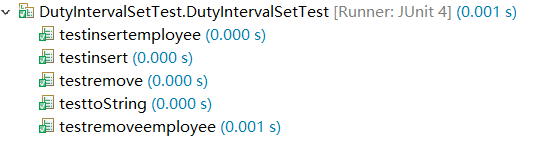
CommonIntervalSet<L>中：只能插入不相同标签的设计。



MultiIntervalSet<L>中：可以插入相同的标签



DultiIntervalSet测试：



## 面向可复用性和可维护性的设计：MultiIntervalSet<L>

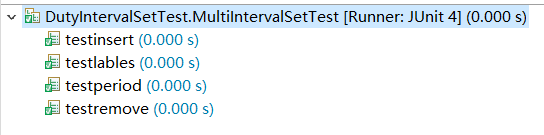
### MultiIntervalSet<L>的共性操作

我设计的MultiIntervalSet<L>是IntervalSet<L>的一个实现，可以实现IntervalSet<L>共性的功能，即共性功能基本与前述的IntervalSet<L>共性操作相同，但是没有根据lable获得起始终止时间的操作。

其他的共性操作：

Periods：根据lable查找时间轴，将其中以lable为标签的片段对应的所有<start,end>对返回。

MultiIntervalSet的测试：

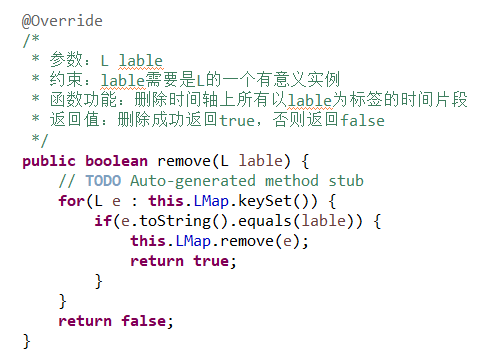


### 局部共性特征的设计方案

Insert：



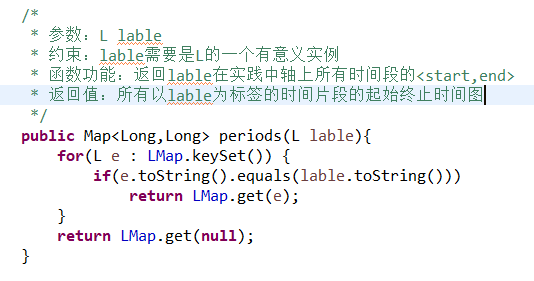
Remove：



Lables：



Periods：

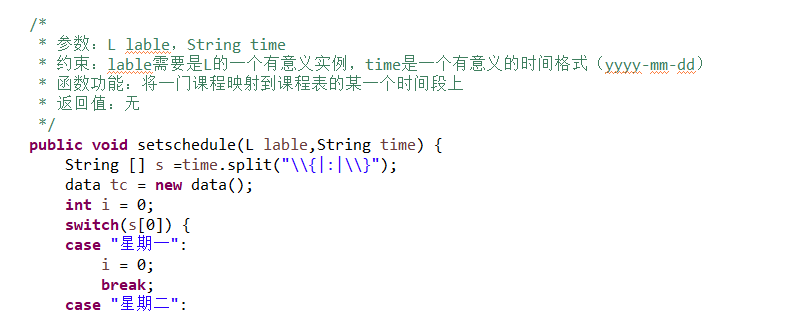


### 面向各应用的MultiIntervalSet子类型设计（个性化特征的设计方案）

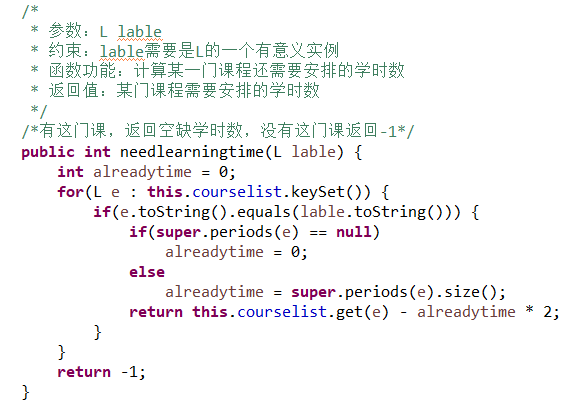
MultiIntercalSet<L>有两个子类型，一个是CourseScheduleApp<L>，一个是ProcessScheduleApp<L>。

CourseSchedule<L>中，共性操作不变，个性操作加入：

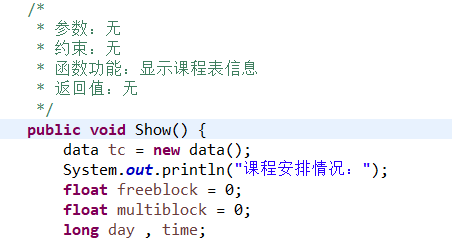
1. Setschedule



2.needlearningtime:用以辅助显示排课信息等函数

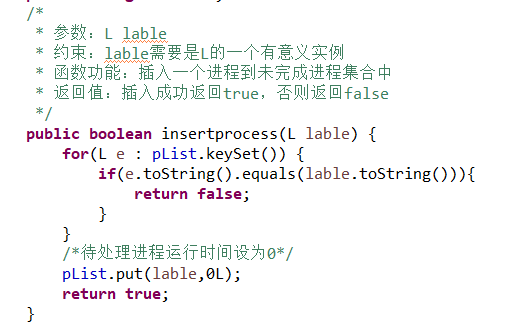


3.Show：显示排课信息：课程列表信息，当前空闲时间占比，当前重叠时间占比

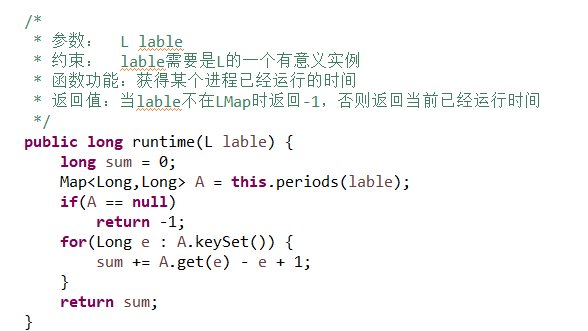


ProcessSchedule<L>中，共性操作不变，个性操作加入：

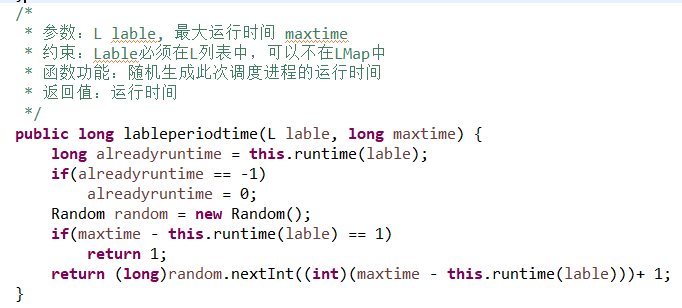
1.insertprocess:更新进程表



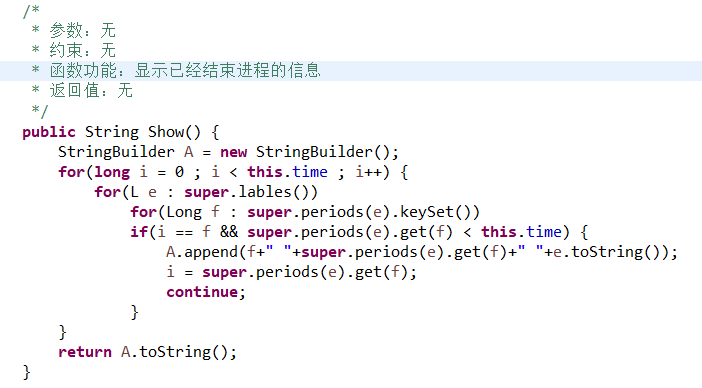
2.runtime：用于辅助确定进程是否应该结束以及随机生成进程运行时间



3.lableperiodtime

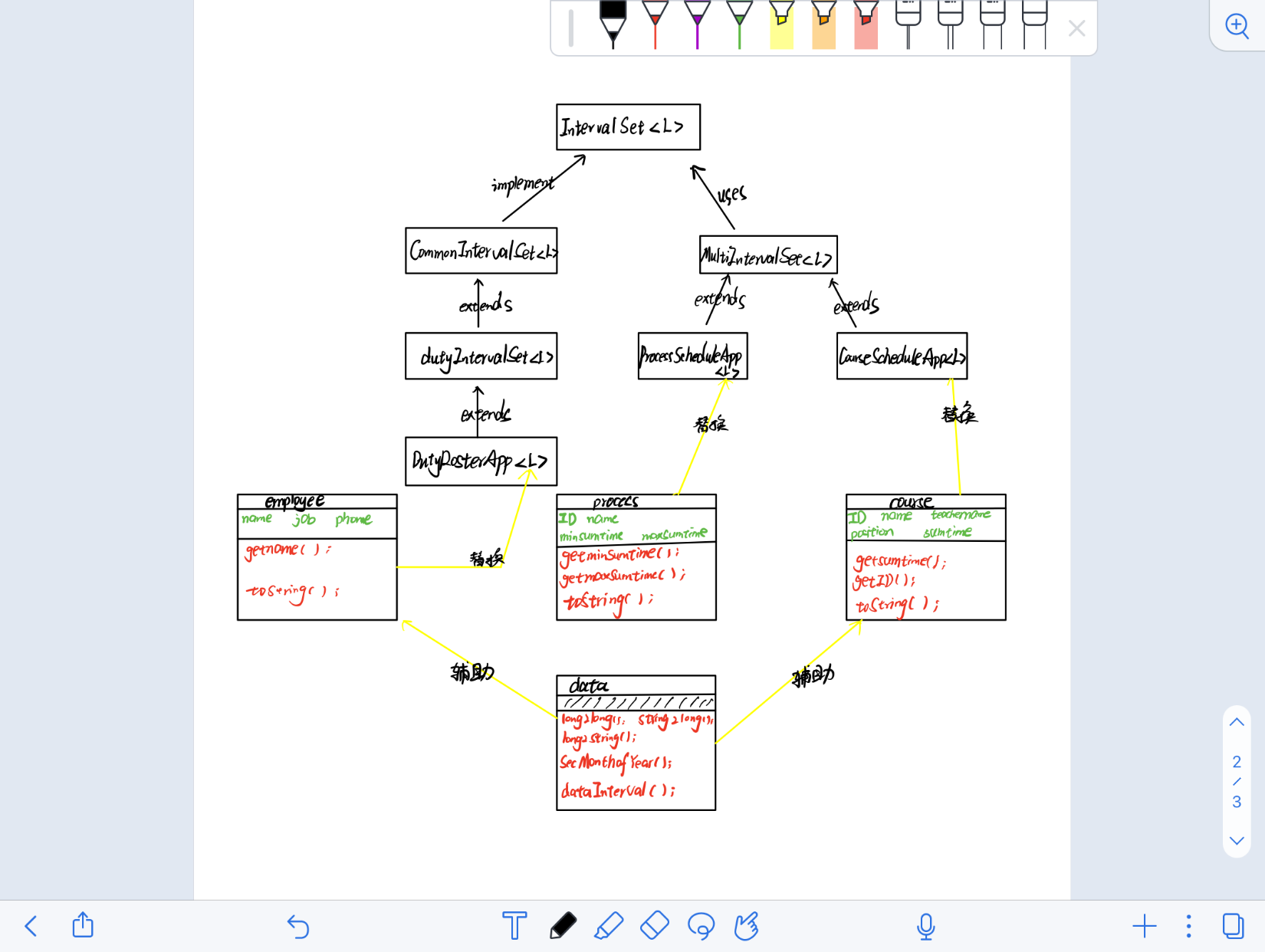


4.Show



## 面向复用的设计：L

**此次试验设计结构：**



L被替换为三个具体的类：employee process course

Employee类：

包括数据：name job phone

包括操作：getname(); toString();

由于存在用户输入员工名字删除相关排班的方法，所以使用getname()来获取员工名字实现。ToString()；展示员工信息。

Process类：

包括数据：ID name minsumtime maxsumtime

包括操作：getminsumtime()；用以确定进程允许最短时间，在判断进程是否应该结束时使用。Getmaxsumtime()；用以确定进程允许最长时间，在随机生成进程运行时间时使用。toString()；用于展示进程信息。

Course类：

包括数据：ID name teachername position sumtime

包括操作：getsumtime()；用于计算还有多少学时应当被安排。GetID()；用于唯一标识一个课程。toString()；用于展示课程信息。

另外还有一个辅助类，实现时间转换与计算

Long2long()；计算从一个日起开始过length天是什么日期。例：long2long(2020-2-26,4)==2020-3-1。

String2long()；将时间字符串转换为对应的long值，例：”2020-02-16”转化为20200216。

Long2string()；将long转化为对应的时间字符串，例：20200312转为为“2020-3-12”。

SecMonthofYear()；用于计算某年的二月有多少天，比如SecMonthofYear(2020)=29。

dataInterval()；用于获得两个long之间的日期序列，例：dataInterval(20200227,20200305)={“2020-2-28\n”,”2020-2-29\n”,”2020-3-1\n”,”2020-3-2\n”,”2020-3-3\n”,”2020-3-4”}。

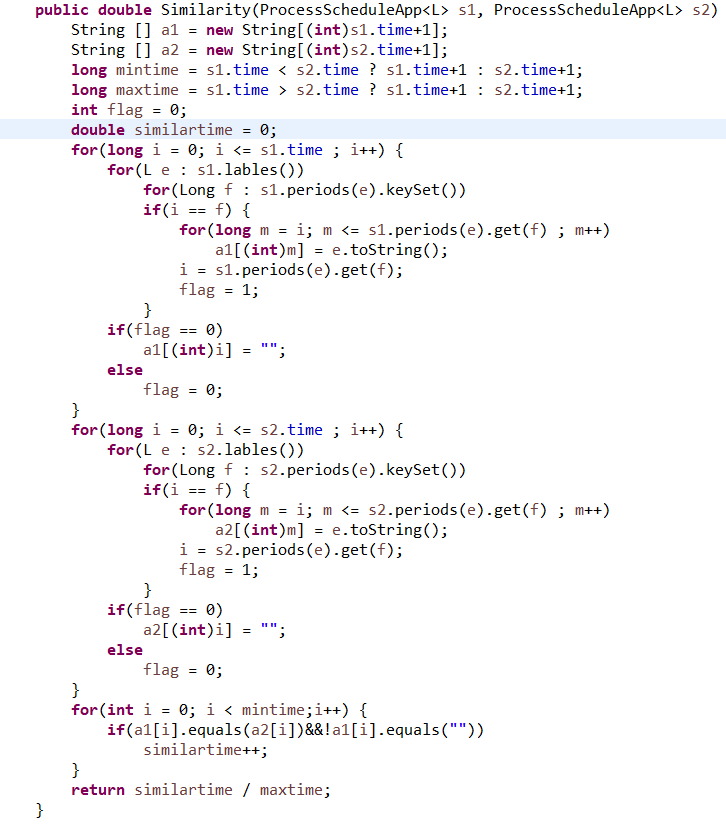
## 可复用API设计

### 计算相似度

计算两个时间轴的相似度，针对应用二

Double Similarity(ProcessSCheduleApp<L> s1,ProcessScheduleApp<L> s2)

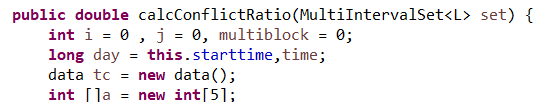
设计思路是先分别对两个进程的时间创建String数组标识各自某一时刻的标签，然后以较短的进程为基准，若它与另一个对应时刻的标识相同且不为空，那就把相似长度加1，最后用总的相似长度除以两进程最长时长即可。



### 计算时间冲突比例

double calcConflictRatio(MultiIntervalSet<L> set)

在应用三中实现，设计思路是按照时间顺序依次查找是否有多个课程安排在一个时间块上，如果是，那么multiblock就自增，一天有5个时间块（8:00-10:00 10:00-12:00 13:00-15:00 15:00-17:00 19:00-21:00），一周有5\*7=35个，由于周期性，统计一周即可，最后使用multiblock/35.0即可。

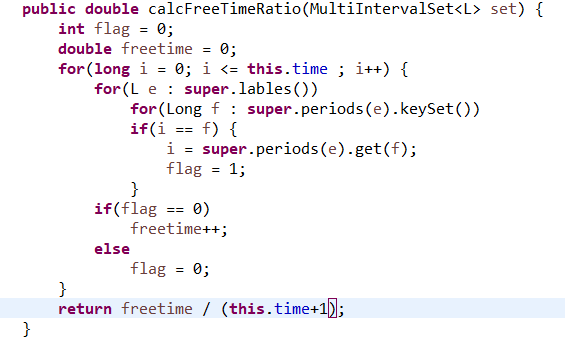


### 计算空闲时间比例

在processScheduleApp中：

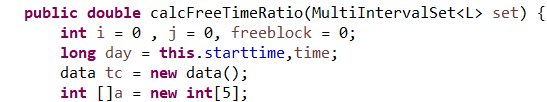
时间轴范围是[0,this.time]，长度是this.Time+1。

循环查找每一个时刻是否是一个进程调度的起始时刻，如果不是就让空闲时间自增，如果是就直接跳转到这个进程结束时的时刻，继续执行。



在CourseScheduleApp中：

与计算冲突时间比例思路完全一致，将某一个时间块无课程时记为freeblock，使其自增，最后再用freeblock/35.0即可。



## 应用设计与开发

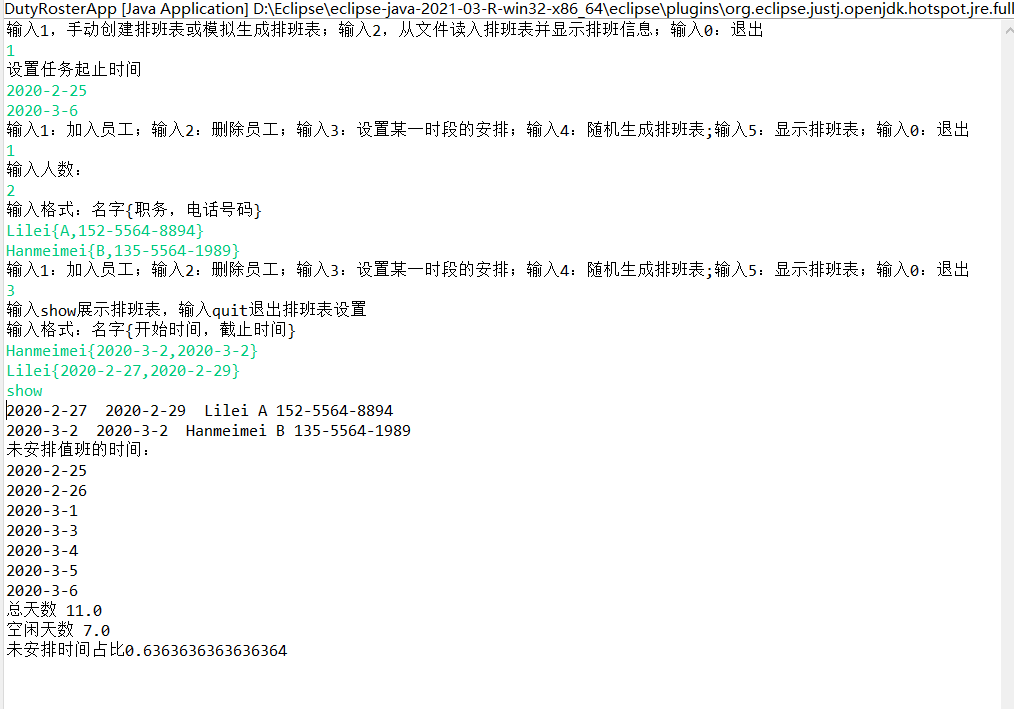
利用上述设计和实现的ADT，实现手册里要求的各项功能。

### 排班管理系统

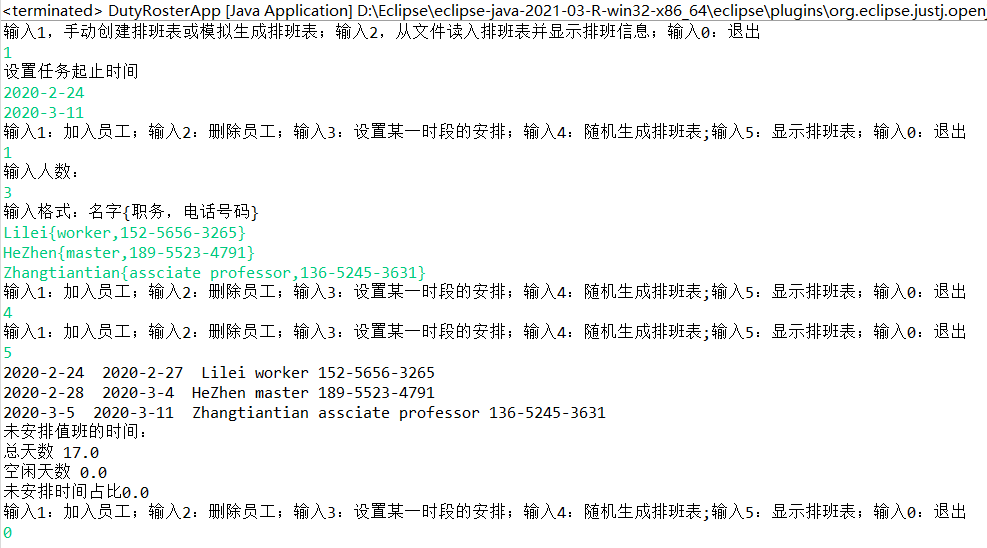
设计思路是有两个模式，可以选择手动输入，也可以选择模拟生成，也可以选择从文本中读取。

手动输入：

先确定排班任务的起始终止日期，然后提示用户输入员工信息，再之后开始setperiod给员工安排时间段值班，可以不让所有员工值班，在这个过程中可以输入show显示当前排班情况，输入quit终止安排。



随机生成排班表：

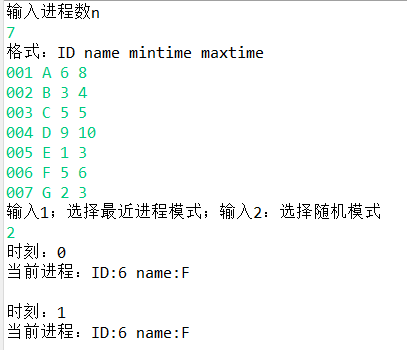
需要先添加员工，然后自动生成一个全安排的排班表，如果没有员工表会提示加入员工表  


### 操作系统的进程调度管理系统

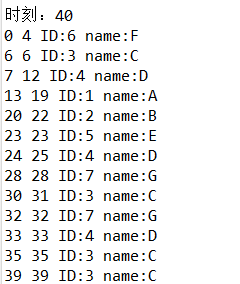
进程调度有模拟和最小时间进程优先两种模式

模拟模式：

首先要求输入一组进程的信息，然后时间轴默认从0开始，每一个非进程调用的时刻都随机选择是保持空闲还是选择一个进程，每当一个进程到达自己的最小时间与最大时间区间时就结束，最后当输入的每一个进程都结束时时间轴收束。



最终结果：

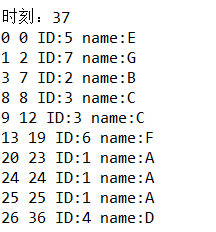


可以看到，在5 26 27 29 34 36 37 38时刻是无进程调度的，并且同一进程可以出现多次。

选择最近进程模式：

每一次进程调度都选择（可以理解为调度不会闲置了）其中已运行时间离最大运行时间最小的进程。

和上面同样的输入：

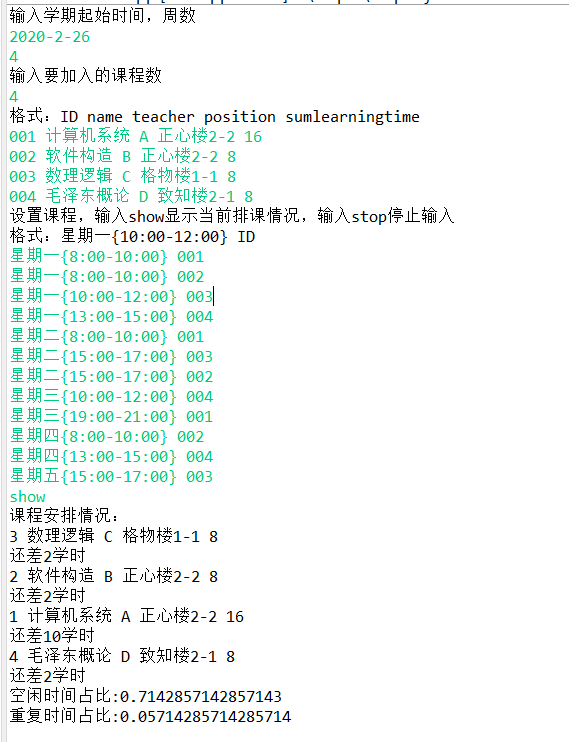


### 课表管理系统

设计思路是首先读取用户输入，确定本学期开始时间和结束时间，确定周数，为了防止对不上时间长度，此App只要求输入开学时间和周数，自动生成开学时间、放假时间和周数。

然后读取课程信息存入课程表中，再之后可以输入 “星期X{8:00-10:00} ID” 的方式将ID唯一指定的课程放入对应时间，在此过程中，可以输入show展示当前排课情况，输入stop表示设置课程结束。

最后，用户可以输入指定一天来显示那一天的课表，要是不是学期范围内的日期，就不会输出当日课表。



## 基于语法的数据读入

修改“排班管理”应用以扩展该功能。

在排班表的前面两个模式上加入新的读取文件的模式，设计思路就是以文本中的一行为输入，先提取出命令，是加入员工、设置起始终止时间还是安排员工，然后再使用另外的信息录入命令的目标存储取余即可。分析某一行使用正则表达式进行字符串切割，值得注意的是‘{’需要加\\转义才能有效果，另外多个分隔符用|连接。

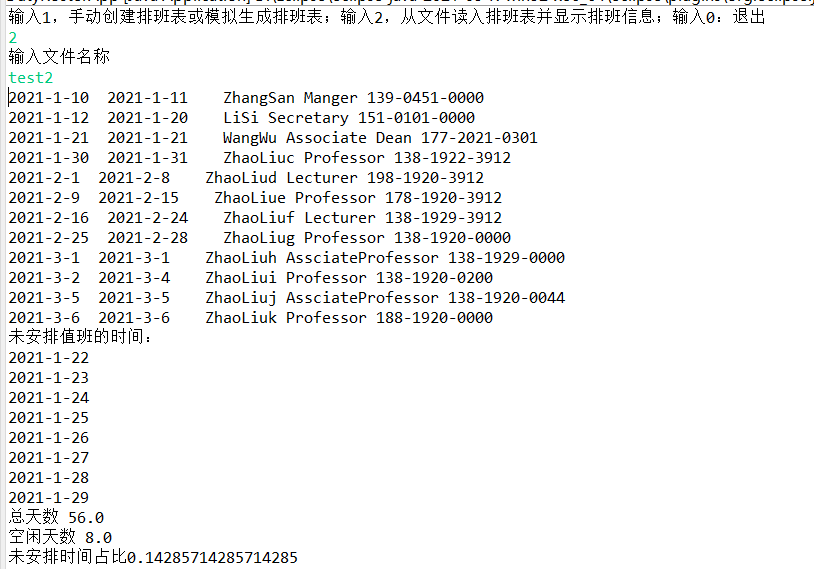
读取文件：

打开外部文件并读入，如果条目是Employee就添加员工，如果条目是Period就设置时间，如果条目是Roster就安排值班，注意的一点是：Roster是建立在Employee和Period都有的基础上的，所以Roster必须在最后。

Test1



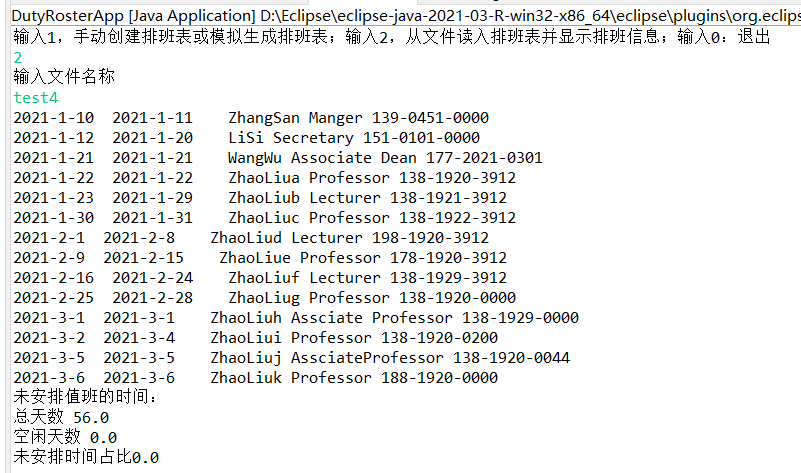
Test2



Test3



Test4



Test5

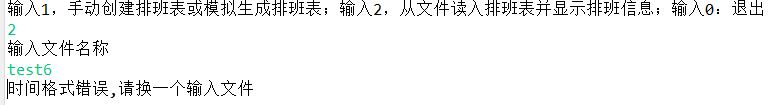




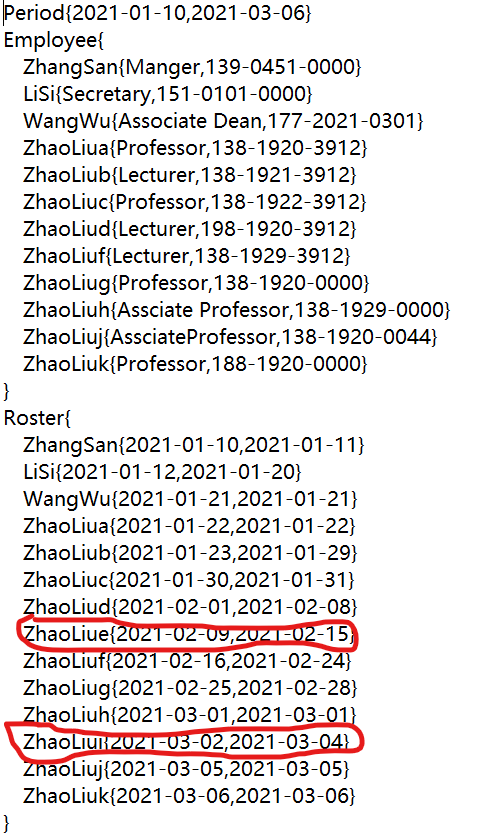
Test6



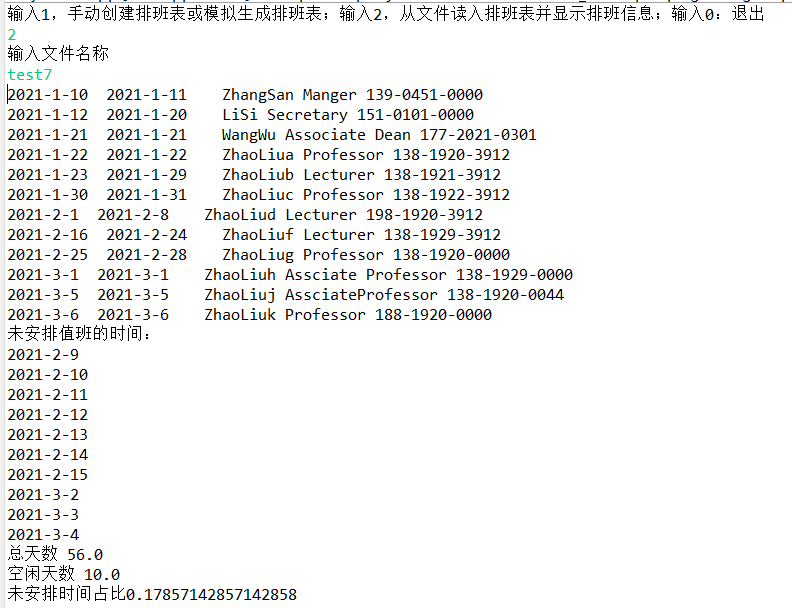
有非法时间，视为非法文件



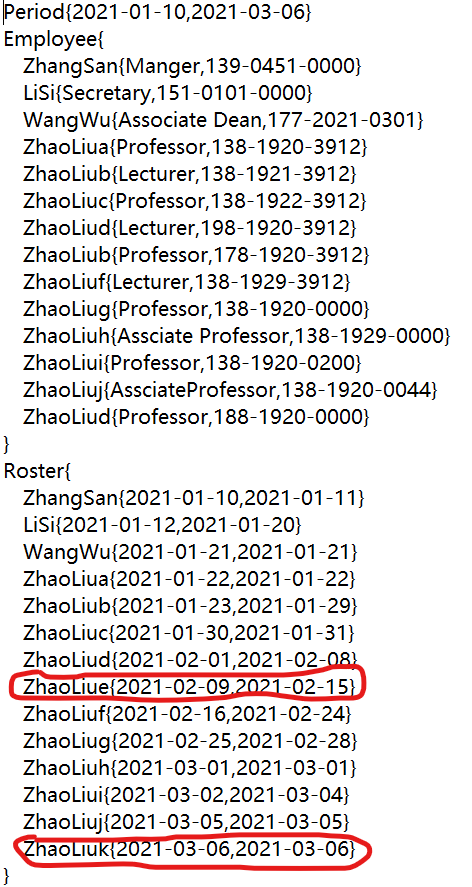
Test7



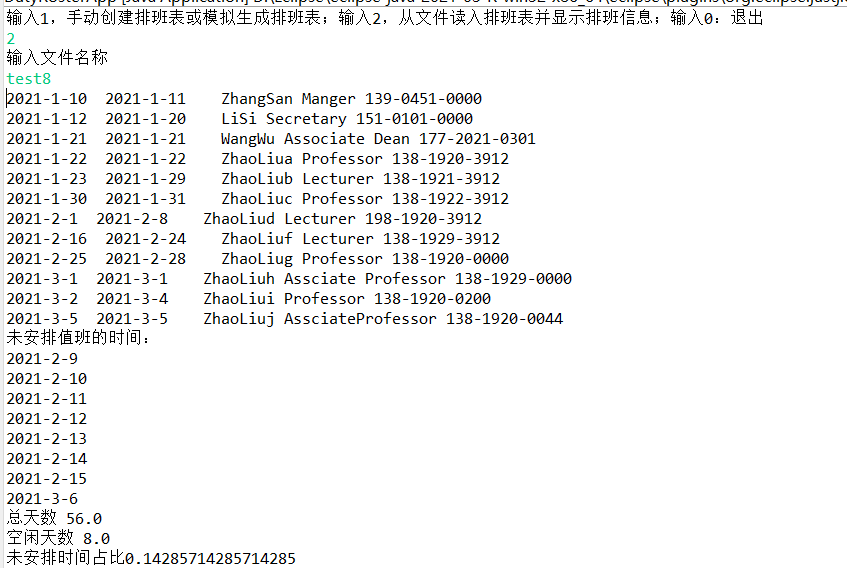
有两个员工不存在



Test8



这两个员工同样没有



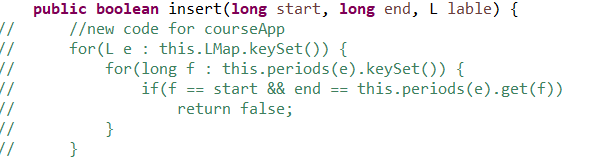
## 应对面临的新变化

### 变化1

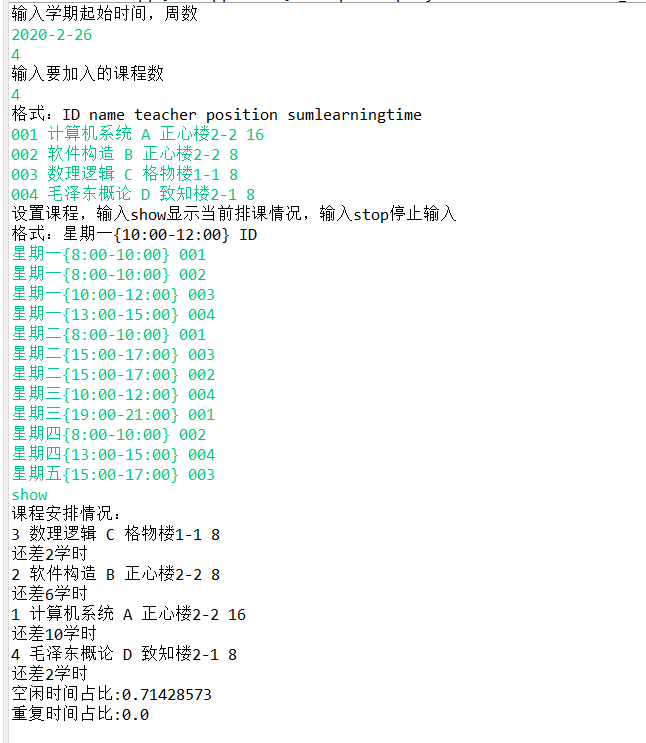
之前的设计不足以应对变化，所以实现能够插入相同标签的方法，应当更改insert，remove这两个函数，还有许多细节要调整，修改比较复杂，修改所用时间也很大。

### 变化2

可以直接在MultiIntervalSet<L>中，插入一段代码，检测是否有重叠，当欲加入的L Lable的起止时间与当前的课程表有重叠时，返回即可，其他函数功能不受影响。



同样的输入的结果：可以看到重复时间占比降到了0



代码修改较简单，修改时间较短。

## Git仓库结构

请在完成全部实验要求之后，利用Git log指令或Git图形化客户端或GitHub上项目仓库的Insight页面，给出你的仓库到目前为止的Object Graph，尤其是区分清楚change分支和master分支所指向的位置。

# 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 |
| 2021-7-1 | 14:00-22:00 | 完成基础类型的书写 | 完成 |
| 2020-7-2 | 9:00-23:00 | 完成排班表App | 完成 |
| 2020-7-3 | 9:00-22:00 | 完成进程App、课程App的书写 | 完成 |
| 2020-7-4 | 10:00-17:00 | 完成报告的整理 | 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 |
| StringBuilder只要创立，即使什么也不append，也会返回“”，这让我根据length判断的方法总是有边界错误 | 按照出错可能原因一一排查，最后发现了症结所在 |
| 不会灵活使用Split | 百度了前辈的经验，结合本学期的形式语言的知识，学会了按指定方式分解字符串 |
| 随机模拟函数的误用 | 随机模拟函数（int a），那么生成的只是[0,a)区间的整数值，再多次此时不符合预期之后，百度解决了问题 |

# 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

经验：

应当先想好结构，类与类的关系，结构的功能，以及继承、实现等方式。

教训：

以前犯过的错还会犯，没有总结记录，另外就是时间有些紧张。

## 针对以下方面的感受

1. 重新思考Lab2中的问题：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？本实验设计的ADT在三个不同的应用场景下使用，你是否体会到复用的好处？

应用场景更复杂，并且会有更多的前后照应的考量。复用很好。

1. 重新思考Lab2中的问题：为ADT撰写复杂的specification, invariants, RI, AF，时刻注意ADT是否有rep exposure，这些工作的意义是什么？你是否愿意在以后的编程中坚持这么做？

提高程序可读性，也提醒开发者注意程序安全性，很好，我愿意。

1. 之前你将别人提供的API用于自己的程序开发中，本次实验你尝试着开发给别人使用的API，是否能够体会到其中的难处和乐趣？

能体会到

1. 你之前在使用其他软件时，应该体会过输入各种命令向系统发出指令。本次实验你开发了一个解析器，使用语法和正则表达式去解析输入文件并据此构造对象。你对语法驱动编程有何感受？

很奇妙，很有作用。

1. Lab1和Lab2的大部分工作都不是从0开始，而是基于他人给出的设计方案和初始代码。本次实验是你完全从0开始进行ADT的设计并用OOP实现，经过五周之后，你感觉“设计ADT”的难度主要体现在哪些地方？你是如何克服的？

主要难在不知道写多少操作够用，而使用ADT时发现功能不够再加，很容易造成模块上的藕断丝连，我提前深思熟虑设计ADT

1. “抽象”是计算机科学的核心概念之一，也是ADT和OOP的精髓所在。本实验的五个应用既不能完全抽象为同一个ADT，也不是完全个性化，如何利用“接口、抽象类、类”三层体系以及接口的组合、类的继承、设计模式等技术完成最大程度的抽象和复用，你有什么经验教训？

一定要想好每一层作用是什么，每一个类表示什么对象。

1. 关于本实验的工作量、难度、deadline。

工作量大、难、deadline比较紧。

1. 到目前为止你对《软件构造》课程的评价。

软件构造还行。