

**2024年春季学期  
计算学部《软件构造》课程**

**Lab 3实验报告**

|  |  |
| --- | --- |
| 姓名 | 翁浩男 |
| 学号 | 2022111171 |
| 班号 | 2237101 |
| 电子邮件 | 1970209297@qq.com |
| 手机号码 | 13821122246 |

**目录**

[1 实验目标概述 1](#_Toc164601170)

[2 实验环境配置 1](#_Toc164601171)

[3 实验过程 1](#_Toc164601172)

[3.1 待开发的三个应用场景 1](#_Toc164601173)

[3.2 面向可复用性和可维护性的设计：IntervalSet<L> 3](#_Toc164601174)

[3.2.1 IntervalSet<L>的共性操作 3](#_Toc164601175)

[3.2.2 局部共性特征的设计方案 3](#_Toc164601176)

[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） 4](#_Toc164601177)

[3.3 面向可复用性和可维护性的设计：MultiIntervalSet<L> 4](#_Toc164601178)

[3.3.1 MultiIntervalSet<L>的共性操作 4](#_Toc164601179)

[3.3.2 局部共性特征的设计方案 5](#_Toc164601180)

[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） 5](#_Toc164601181)

[3.4 面向复用的设计：L 6](#_Toc164601182)

[3.5 可复用API设计 6](#_Toc164601183)

[3.5.1 计算相似度 6](#_Toc164601184)

[3.5.2 计算时间冲突比例 6](#_Toc164601185)

[3.5.3 计算空闲时间比例 6](#_Toc164601186)

[3.6 应用设计与开发 7](#_Toc164601187)

[3.6.1 排班管理系统 8](#_Toc164601188)

[3.6.2 操作系统的进程调度管理系统/课表管理系统 11](#_Toc164601189)

[3.7 基于语法的数据读入 14](#_Toc164601190)

[4 实验进度记录 15](#_Toc164601191)

[5 实验过程中遇到的困难与解决途径 16](#_Toc164601192)

[6 实验过程中收获的经验、教训、感想 16](#_Toc164601193)

[6.1 实验过程中收获的经验和教训 16](#_Toc164601194)

[6.2 针对以下方面的感受 16](#_Toc164601195)

# 实验目标概述

本次实验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：

（1）子类型、泛型、多态、重写、重载

（2）继承、代理、组合

（3）语法驱动的编程、正则表达式

（4）API设计、API复用

跟据三个具体应用（值班表管理、操作系统进程调度管理、大学课表管理）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

IDE：IntelliJ IDEA Community Edition 2023.3.5

JDK：Java11

测试： Junit 4

开发系统：windows11

GitHub Lab3仓库的URL地址：[ComputerScienceHIT/HIT-Lab3-2022111171: HIT-Lab3-GJSJ7612 created by GitHub Classroom](https://github.com/ComputerScienceHIT/HIT-Lab3-2022111171)

## 待开发的三个应用场景

**值班表管理系统：**

要求在某个时间段内（例如寒假 1 月 10 日到 3 月 6 日期间），每天只能安排唯一一个员工在单位值班，且不能出现某天无人值班的情况；每个员工若被安排值班 m 天（m>1），那么需要安排在连续的 m 天内。值班表内需要记录员工的名字、职位、手机号码，以便于外界联系值班员。



图1 值班表管理系统示意图

**操作系统进程调度管理系统：**

要求考虑计算机上有一个单核CPU，多个进程被操作系统创建出来，它们被调度在 CPU 上执行，由操作系统决定在各个时段内执行哪个进程。操作系统可挂起某个正在执行的进程，在后续时刻可以恢复执行被挂起的进程。可知：每个时间只能有一个进程在执行，其他进程处于休眠状态；一个进程的执行被分为多个时间段；在特定时刻，CPU 可以“闲置”，意即操作系统没有调度执行任何进程；操作系统对进程的调度无规律，可看作是随机调度。

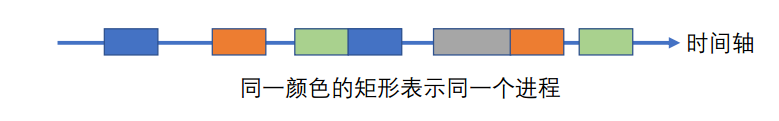


图2 值班表管理系统示意图

**大学课表管理系统：**

在本应用中，我们对实际的课表进行简化：针对某个班级，假 设其各周的课表都是完全一样的（意即同样的课程安排将以“周” 为单位进行周期性的重复，直到学期结束）；一门课程每周可以出 现 1 次，也可以安排多次（例如每周一和周三的“软件构造”课） 且由同一位教师承担并在同样的教室进行；允许课表中有空白时间 段（未安排任何课程）；考虑到不同学生的选课情况不同，同一个 时间段内可以安排不同的课程；一位教师也可以承担课表中的多门 课程。

**共性分析：**

每个系统都需要对有限的时间段进行分配。在第一个系统中是 员工的值班时间，在第二个系统中是CPU的执行时间，在第三个 系统中是教室和教师的时间。所有系统都需要解决如何有效地调度 资源的问题。值班表需要确保每天都有人值班，CPU调度需要决定 哪个进程在何时执行，课表安排需要决定哪些课程在何时何地进行。

**特性分析：**

值班表管理系统要求员工分配的时间段是连续的，而操作系统进 程调度管理系统，大学课表管理系统均可以分配多个时间段，在值班表 管理系统与操作系统进程调度管理系统中均要求分配的时间段不能产生 重叠，操作系统进程调度管理系统和大学课表管理系统均可以存在空白

## 面向可复用性和可维护性的设计：IntervalSet<L>

### IntervalSet<L>的共性操作

empty(long begin, long over) ：静态工厂方法，默认CommonIntervalSet为该接口的实现类

void insert(long start, long end, L label)：实例方法，该方法在当前对象中插入新的时间段和标签

Set<L> labels()：实例方法，获得当前对象中的标签集合

boolean remove(L label)：实例方法，从当前对象中移除某个标签所关联的时间段

long start(L label)：实例方法，返回某个标签对应的时间段的开始时间

long end(L label)：实例方法，返回某个标签对应的时间段的结束时间

long getBegin()：实例方法，获取限定时间段的起始时间

long getOver()：实例方法，获取限定时间段的终止时间

void setBegin(long begin)：实例方法，设定限定时间段的起始时间

void setOver(long over)：实例方法，设定限定时间段的终止时间

double accept(IntervalVisitor<L> v)：实例方法，扩展功能接口

(使用visitor设计模式）

### 局部共性特征的设计方案

CommonIntervalSet<L>：

IntervalSet<L>的实现类，默认为允许有空白、允许产生时间段重叠、不含有周期信息的IntervalSet，其中Rep包含begin：限定时间段的起始时间、over：限定时间段的终止时间、interval：标签到所关联时间段的映射，其中标签作为键、关联时间段作为值，保存限定时间段内的时间段关系。

委派用接口：

1. NoBlankIntervalSet<L>：

实现保证IntervalSet中不含有空白的接口，包含boolean checkNoBlank(IntervalSet<L> intervalSet)方法，用于检验IntervalSet是否存在空白

1. NonOverlapIntervalSet<L>

实现保证 IntervalSet中时间段不重叠的接口，包含void checkNoOverlap(IntervalSet<L> intervalSet)方法，用于检查IntervalSet中是否存在重叠的时间段，当发生重叠时会抛出自定义IntervalConflictException的异常

1. Period

实现Interval的周期性，将Interval限定的时间段作为单元时间段，并为这个单元时间段附加一个周期的属性，包含int getTimes()、void setTimes(int times)两个方法，可以对周期重复的时间次数进行设定

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

IDutyIntervalSet<L>

实现方式主要是**基于委派方式**，该类分别对CommonIntervalSet、NoBlankIntervalSet、NonOverlapIntervalSet这三者进行委托实现。

static<L> DutyIntervalSet<L> empty(long begin, long over) 静态工厂方法，默认使用DutyIntervalSet<L>实现

void addEmployee(long start, long end, L employee) 实例方法，在当前排班表中增加新的值班人员

Set<L> Employees() 实例方法，获得当前排班表中已添加员工集合

boolean removeEmployee(L employee) 实例方法，从排版表中移除某个员工的排班记录

long[] checkEmployeeStartToEnd(L employee) 实例方法，返回某个员工的工作时间段

int checkEmployeeTime(L employee) 实例方法，返回某个员工的工作天数

boolean checkNoBlank() 实例方法，检查排班表中是否存在空白

long getBegin() 实例方法，获取限定时间段的起始时间

long getOver() 实例方法，设定限定时间段的终止时间

double calculateFreeTime() 实例方法，计算当前的空闲比率（使用CalcFreeTimeRatioInterval 的visitor实现方法）

## 面向可复用性和可维护性的设计：MultiIntervalSet<L>

### MultiIntervalSet<L>的共性操作

empty(long begin, long over) ：静态工厂方法，默认CommonIntervalSet为该接口的实现类

void insert(long start, long end, L label)：实例方法，该方法在当前对象中插入新的时间段和标签

Set<L> labels()：实例方法，获得当前对象中的标签集合

boolean remove(L label)：实例方法，从当前对象中移除某个标签所关联的时间段

IntervalSet<Integer> intervals(L label)：实例方法，从当前对象中获取与某个标签所关联的所有时间段

long getBegin()：实例方法，获取限定时间段的起始时间

long getOver()：实例方法，获取限定时间段的终止时间

void setBegin(long begin)：实例方法，设定限定时间段的起始时间

void setOver(long over)：实例方法，设定限定时间段的终止时间

double accept(MultiIntervalSetVisitor<L> v)：实例方法，扩展功能接口

### 局部共性特征的设计方案

CommonMultiIntervalSet<L> MultiIntervalSet的实现类，默认为允许有空白、允许产生时间段重叠、不含有周期信息的MultiIntervalSet，主要实现方式是基于对 IntervalSet的委派，其中rep包含标签到其所有关联时间段的映射，标签作为键，其关联的所有时间段使用List<long[]>保存为值

MultiIntervalSetDecorator是MultiIntervalSet的装饰器抽象类，其实现类有NonOverlapMultiIntervalSet，通过重写insert方法在CommonMultiIntervalSet的基础上实现了没有重叠时间段的功能。

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

IProcessIntervalSet是基于装饰器实现的，其中包含

empty() 静态工厂方法，默认使用ProcessIntervalSet<L>实现

void addProcess(long start, long end, L progress) 实例方法，在进程序列中添加一次进程活动

Set<L> processes() 实例方法，获取当前进程序列中的进程集合

long runTime(L progress) 实例方法，获取目标进程的当前已运行时间

IntervalSet<Integer> checkProgress(L progress) 实例方法，获取某个进程已运行时间段

void setOver(long over) 实例方法，设定限定时间段的终止时间

ICourseIntervalSet是基于继承+委派实现的，其中包含

empty(long begin, int Cycle) 静态工厂方法，默认使用CourseIntervalSet<L>实现

void addCourse(long start, L course) 实例方法，在课表中添加课程

boolean removeCourse(L course)实例方法，移除课表中某个课程

Set<L> courses() 实例方法，查看课表中已添加课程

IntervalSet<Integer> checkCourse(L course) 实例方法，检查课表中的某个课程已安排时间段

void setBegin(long begin) 实例方法，设定学期的开始时间

long getBegin() 实例方法，获取学期的开始时间

void setCycle(int cycle) 实例方法，设定学期周数

int getCycle() 实例方法，获取学期周数

## 面向复用的设计：L

Employee：

员工类，包含员工的姓名、职务、手机号码 ，其中要求姓名、职位不能为空，电话为一个11位数。包含String getName()、String getOffice()、String getPhoneNumber()三个获取其内部rep数值的方法

MyProcess：

进程类，包括进程的编号、名称、最小的运行时间与最大的运行时间，其中要求PID最高为5位数，进程名称不能为空，最大/最小运行时间必须为自然数,且最大运行时间要大于等于最小运行时间，包含int getPID()，String getname()，long getminTime()，long getmaxTime()四个获取内部rep数值的方法

Course：

课程类，包括课程的课程号、课程名称、教师姓名、地点与周学时数，其中要求课程号、课程名称、教师姓名、地点名称不能为空，周学时数必须为偶数，包含String getCourseID()、String getCourseName()、String getTeacherName()、String getLocation()、int getWeekHour()五个获取内部rep数值的方法

## 可复用API设计

### 计算相似度

采用API接口的方式实现，在API接口中包含double Similarity(MultiIntervalSet<L> s1, MultiIntervalSet<L> s2)的方法，按照时间轴从早到晚的次序，针对同一个时间段内两个对象里的interval，若它们标注的label等价，则二者相似度为1，否则为0；若同一时间段内只有一个对象有 interval 或二者都没有，则相似度为 0。将各interval的相似度与interval的长度相乘后求和，除以总长度，即得到二者的整体相似度。

### 计算时间冲突比例

采用visitor的设计模式，分别设计有IntervalVisitor和MultiIntervalSetVisitor两种对应的visitor接口，分别对应有CalcConflictRatioInterval、CalcFreeTimeRatioInterval

和CalcConflictRatioMultiIntervalSet、CalcFreeTimeRatioMultiIntervalSet四种实现，实现了计算IntervalSet、MultiIntervalSet的计算时间冲突比例和计算空闲时间比例

实现：通过获取IntervalSet/MultiIntervalSet中每个时间段并对其进行排序，再次遍历排序后的时间段，当发生冲突时获取冲突的时间段，当发生多重冲突时，将新产生的冲突时间与上一个冲突时间段进行比较，若有交集则将两个时间段合并成一个新的冲突时间段，最终遍历整个冲突时间段集合来计算总冲突时间段时间，并与总时间段作比，得到时间冲突比例。

### 计算空闲时间比例

设计思路过程同上

实现：通过获取IntervalSet/MultiIntervalSet中每个时间段并对其进行排序，再次遍历排序后的时间段，当前时间段若与上一时间段产生交集则合并生成新的时间段，最后遍历限定的时间段内存在的时间段，用总时间段减去已有时间段得到空白时间段并与总时间段作比，得到空闲时间比例。

## 应用设计与开发

图3 系统结构设计类图

### 排班管理系统

整体实现思路采用黑盒框架的思路实现。整体框架设计样式如图所示

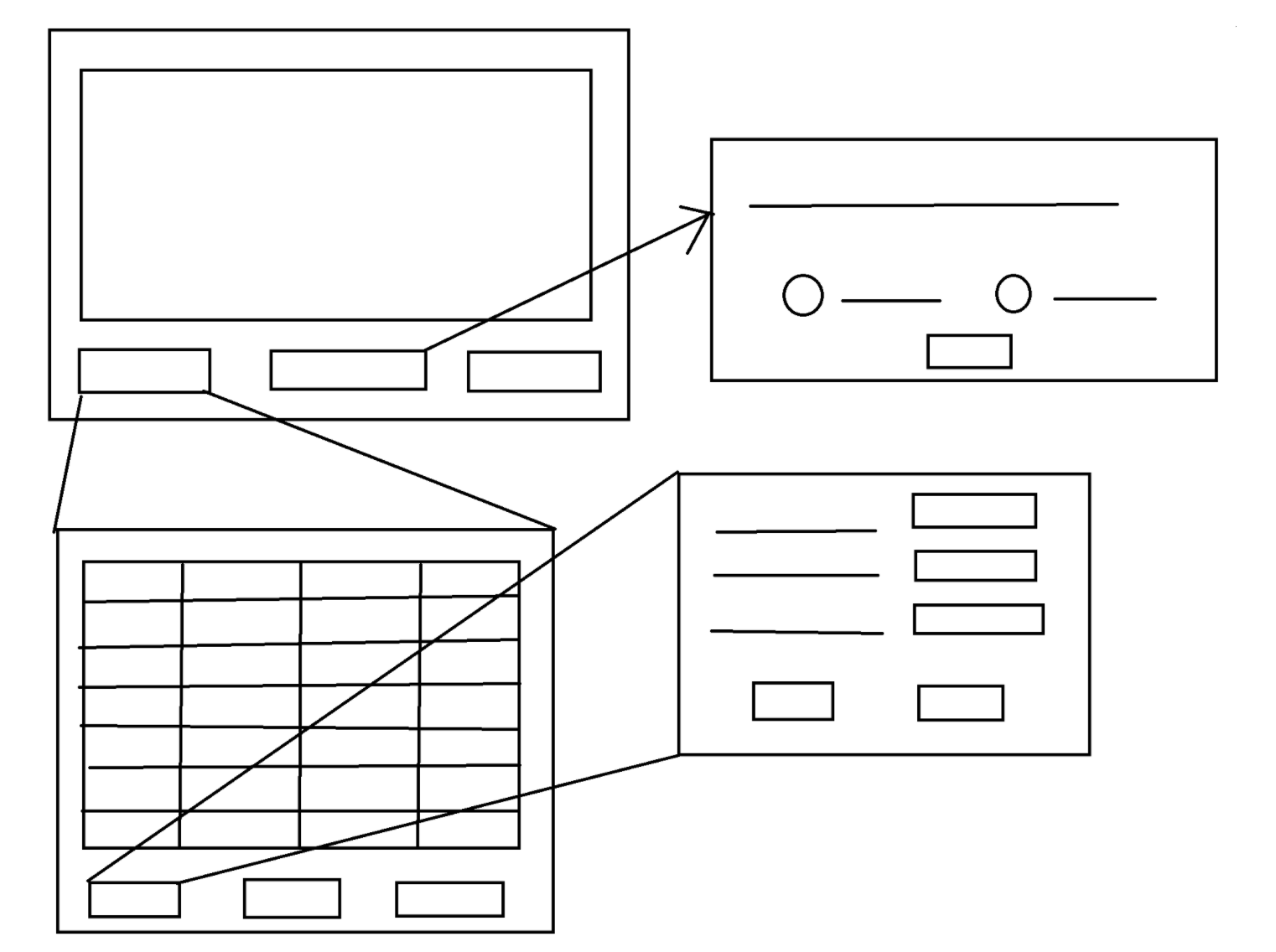


图4 黑盒框架样式设计示意图

接口设计如下：

Component special() 在主界面空白区域添加特殊的组件（如表格、绘图）

String setTitle() 设定主窗口标题

String setButton1()设定按钮1名称

String setButton2()设定按钮2名称

String setButton3()设定按钮3名称

String ShowMessage() 设定主界面显示信息

String setAddElementTitle() 设定按钮1子窗口标题

boolean initWindow() 创建初始化窗口

boolean checkElement(JTextField[] textFields) 检验添加元素（员工、进程等）是否符合类本身规范

void addElement(JTextField[] textFields) 添加元素到内部数据结构

Object[][] ElementData() 将内部元素数据结构规整到便于输出的格式

String[] setElementColumn() 设定展示元素信息的列名

boolean controlDelete(String Element) 检查移除元素时是否合规

void DeleteElement(String Element) 从内部数据结构中删除元素

String setStrategy1() 设定按钮2子窗口左选项名

String setStrategy2() 设定按钮2子窗口右选项名

void strategy(int state, JLabel message) 选择不同的执行策略

void button3(JLabel message) 设定按钮3运行逻辑（留给系统个性实现）

为辅助该程序的完成，设计有适配器接口Adaptor辅助系统编写

Adaptor包含两个方法translatorStringToLong和translatorLongToString，他们分别将形如YYYY-MM-DD的字符串日期转化成对应的long值和将long值转化成对应的YYYY-MM-DD的字符串日期

**系统实现：**

Step 1 设定排班开始日期、结束日期，具体到年月日即可。

针对第一步设定了初始化界面

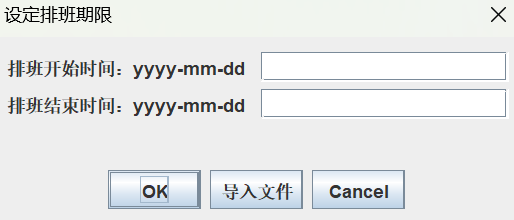


图5 初始化界面图

用户可选择手动输入排班的开始时间或选择导入已有文件（详情将在基于语法的数据读入中解释），当用户进行手动输入排班开始日期与结束日期时，要求用户输入的日期格式为YYYY-MM-DD（如开始时间2022-01-01，结束时间2022-01-03），当用户输入非法的格式时会弹出警告窗（图6），当用户输入开始时间晚于结束时间时同样会弹出警告窗（图7）

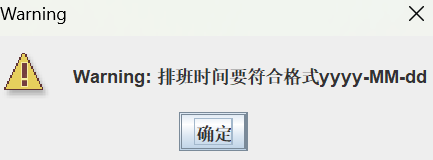
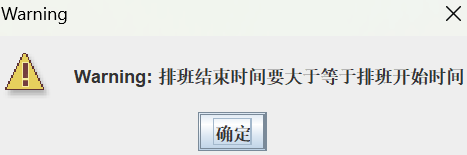
 

图6、7 非法时间格式弹窗图

当初始化完成后进入主界面，主页面会将从开始日期到结束日期的表格一一列出，空白行代表尚未排班表格下方显示有排班开始时间，结束时间与空白比率



图8 主界面窗口图

Step 2 通过点击按钮1来管理员工信息，点击添加按钮可以增加一组员工，输入他们各自的姓名、职务、手机号码，后点击OK即可添加成功，并可随时通过选中表格中某列并点击删除按钮来删除某些员工。但如果某个员工已经被编排进排班表，那么他不能被删除，并会弹出警告窗（图11），必须将其排班信息删掉之后才能删除该员工。

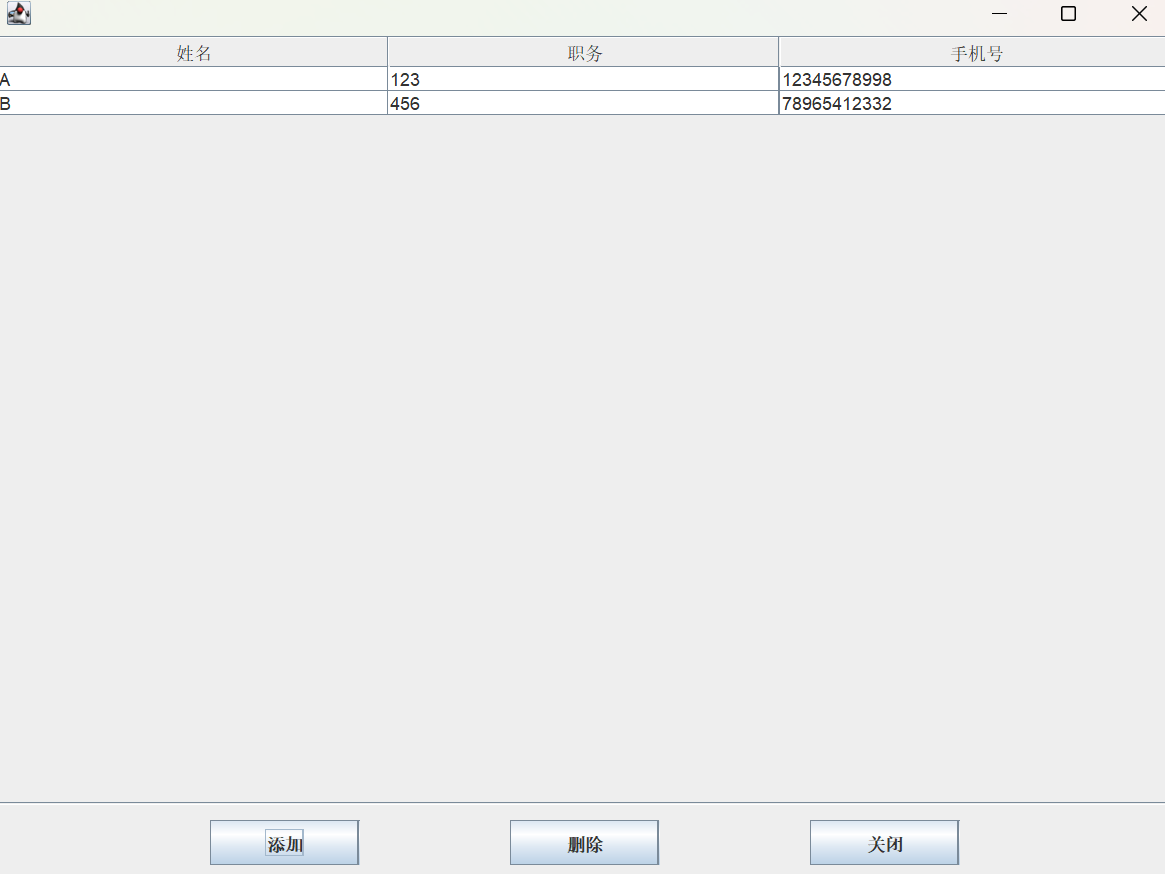


图9 管理员工信息窗口图

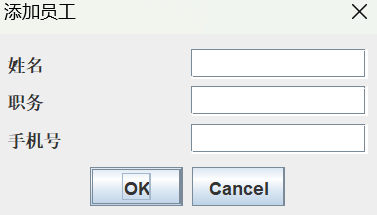


图10 添加员工信息窗口图

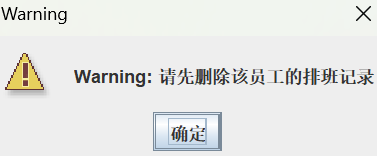


图11 删除失败警告窗

Step 3 点击按钮2可通过手工或自动的方式，向排班表增加排班记录。自动排班将无视已有排班表，使用目前员工表中员工进行随机排班，使其排满整个排班表 ；选择手动排班则要求用户输入员工工作的开始时间，结束时间与安排的员工姓名，并且当排班表不满时会要求用户继续输入，当输入日期格式不满足或姓名不存在与员工名单，则同样会产生警告窗（图6、14）

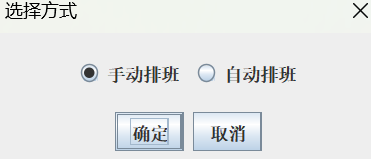


图12 选择排班方式窗口

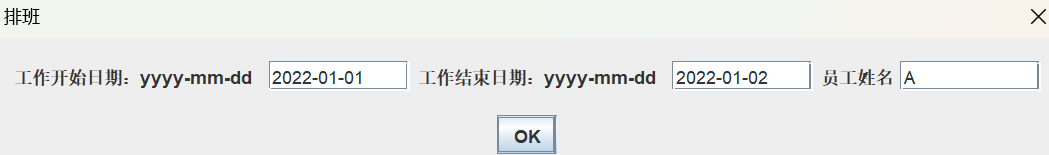


图13 手动排班方式窗口

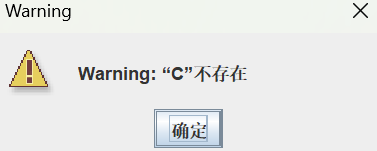


图14 排班人员不存在警告窗

Step 4 点击按钮3后，可以通过输入想要删除排班记录对应的员工姓名来删除该员工所有的排班记录，同样当姓名不存在于排班表中时会弹出警告窗（图16）

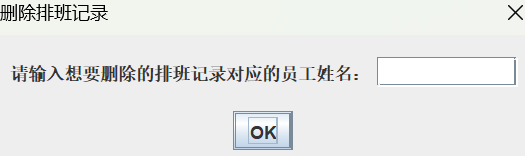


图15 删除排班记录窗口

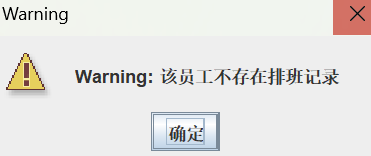
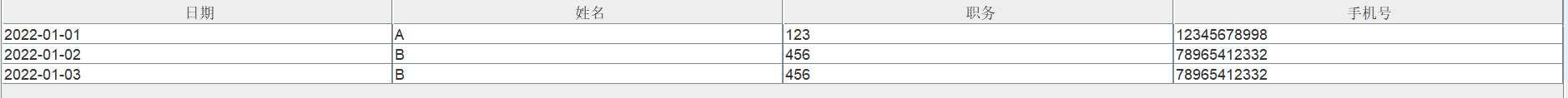


图16 排班员工不存在警告窗



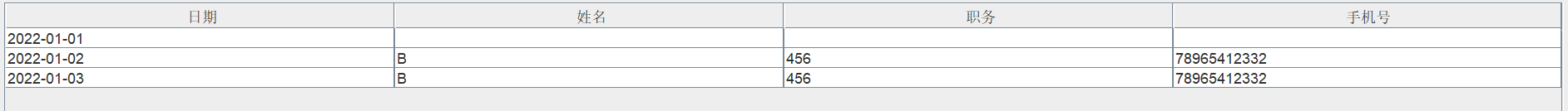


图17 删除A员工排班记录效果图

### 操作系统的进程调度管理系统

相较于排班管理系统，操作系统的进程调度管理系统（下简称进程系统）整体流程较为相近，特殊之处在于该系统将主界面的空白窗口中的内容换成了画板，并且在按钮2的方案选择的实现上采取了策略模式的设计模式，StrategyProcess接口中包含有SelectProcess一个方法，对应着有两种实现方式StrategyProcessPriority和StrategyProcessRandom，分别实现了按照“最短进程优先”的策略选取的进程和随机选取进程的策略。

ProcessDiagram类是针对画板进行进程可视化的类，通过读取内部元素数据结构来对进程进行绘制，每次内部元素数据结构发生变化时，将会清空画板上的所有绘制并重新绘制之前元素与新增元素。从而达到画板内容动态更新的效果

**系统实现：**

由于进程系统不需要输入起始时间与终止时间，故打开程序即为主界面，且空白区域不存在任何绘制

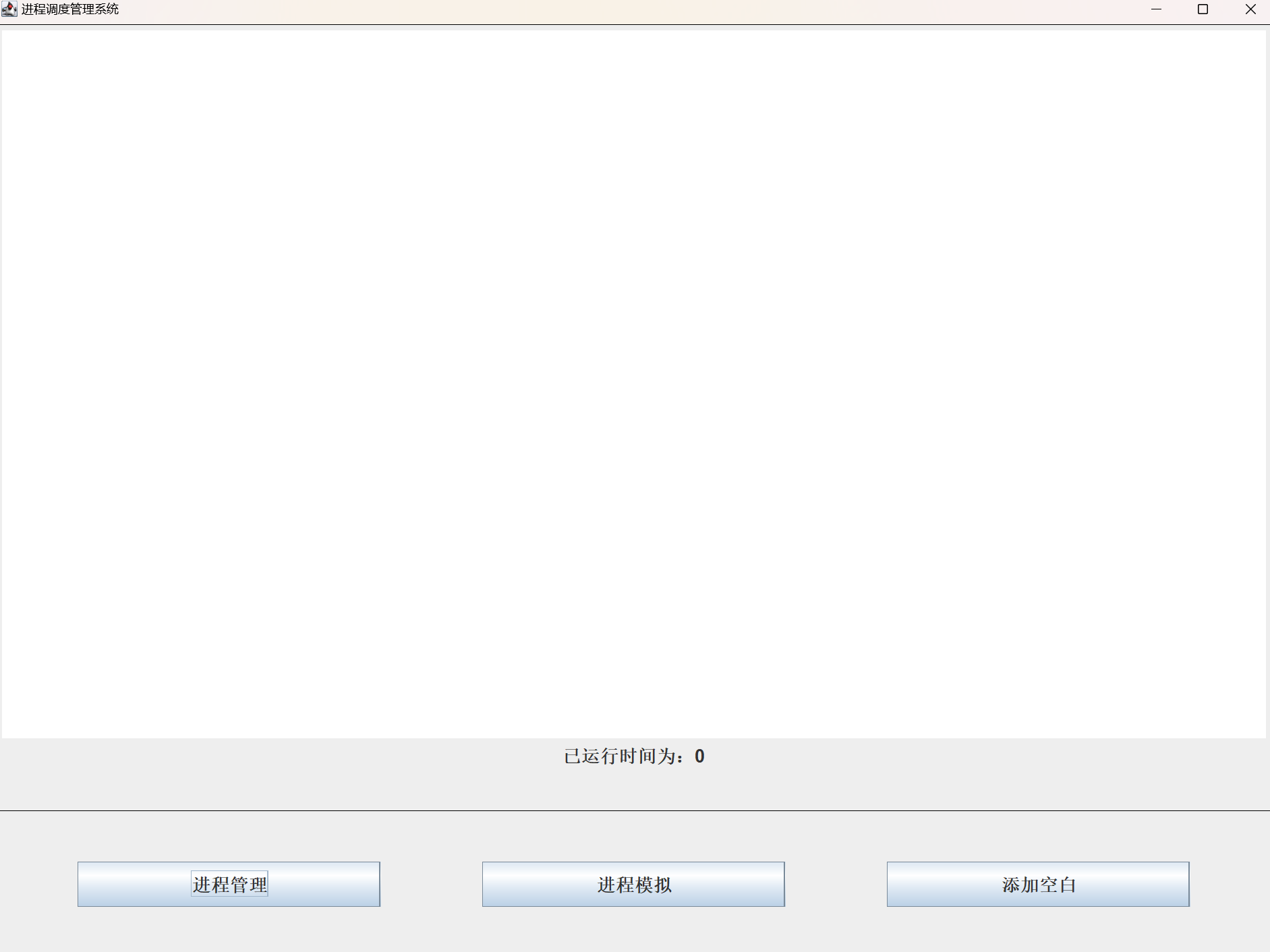


图18 进程系统主页面窗口图

Step 1同排班表系统近似，可类比添加员工的方式添加进程信息，其中要求PID最高为5位数，进程名称不能为空，最大/最小运行时间必须为自然数,且最大运行时间要大于等于最小运行时间，当不满足以上条件时会产生警告窗

不同于排班表，进程可随时删除，不受进程进度的影响

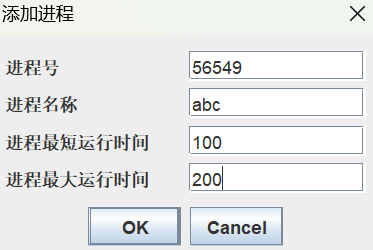


图19 添加进程页面窗口图

现向进程列表中添加如下进程：

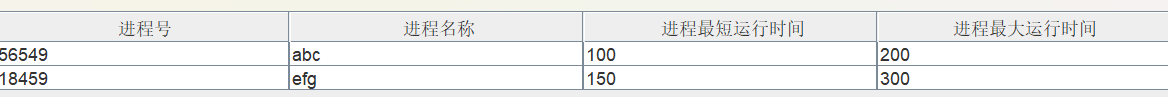


图20 已添加进程表格

Step 2点击按钮2后可以选择选择进程的方式，可以通过随机或选择距离最长运行时间最短的进程，当用户选择随机选择的方式时，系统会从未结束进程中随机选取一个进程来让用户输入运行时间，用户在输入时可以看到当前所选进程的全部信息，输入确定后即可在画布上看到代表该进程的矩形与其对应的标签，其中矩形的颜色是由进程的标签的哈希值转换成对应的rgb来实现的，即相同标签的进程会具有相同的颜色，提高了系统的直观程度



图21 随机选择进程窗口图

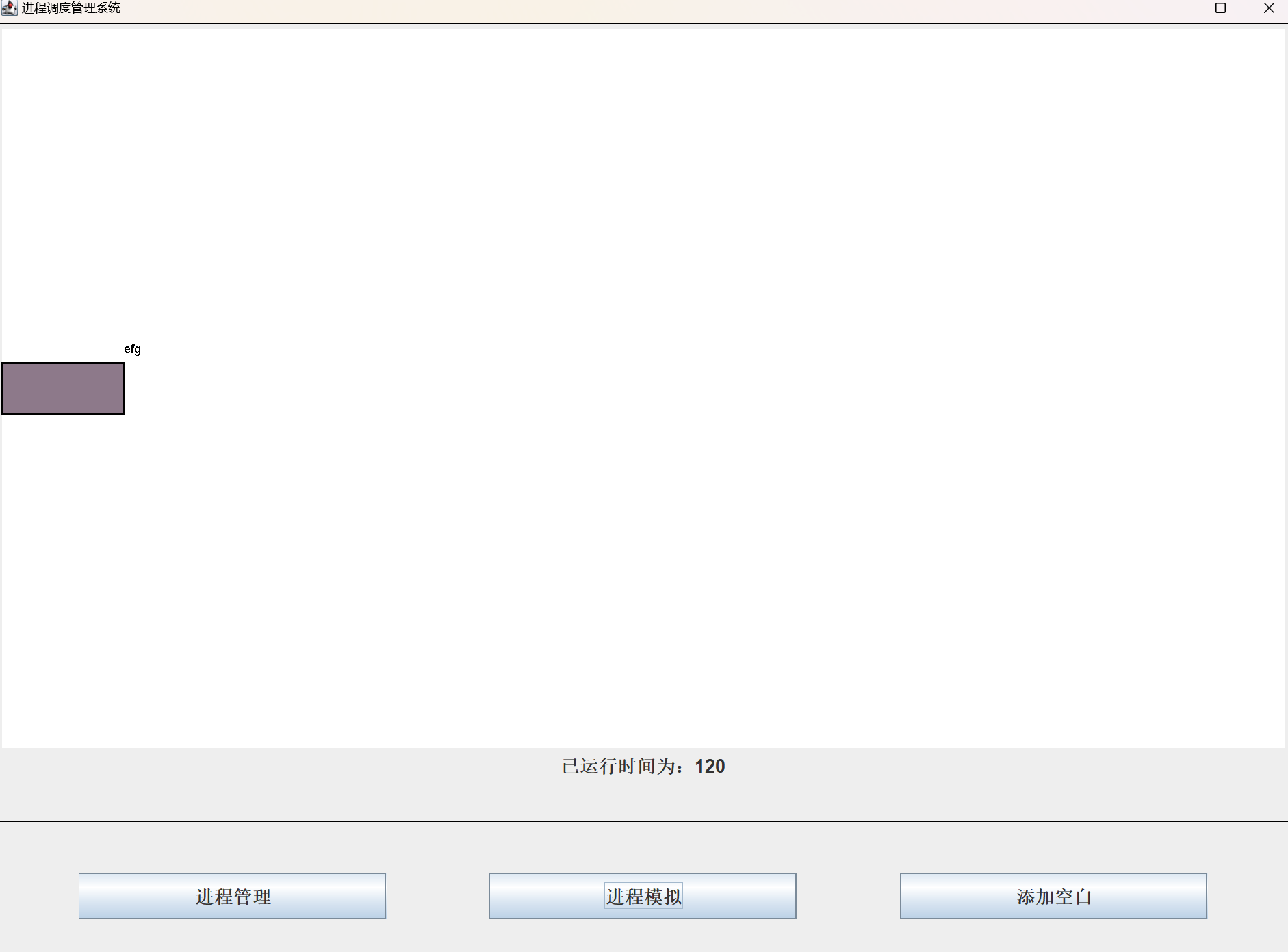


图22 添加进程效果图

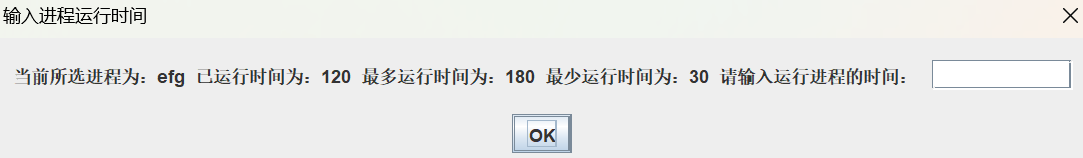
当用户选择距离最长运行时间最短的进程的策略时，由于当前进程已经运行120距离最大运行时间相差180小于abc进程的200，因此选择该策略时会再次选择efg进程

图23 选择距离最长运行时间最短的进程的策略效果图

Step 3在点击按钮3后，可以在当前时刻添加一段时间的空白进程，用户可以选择输入空白进程等待的时间，在输入了空白时间后进程图中将会出现一条横线代表进程轴，即此时无进程运行。

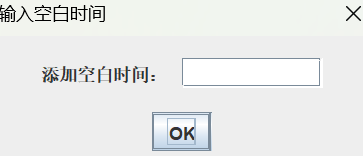


图24 输入空白进程窗口图

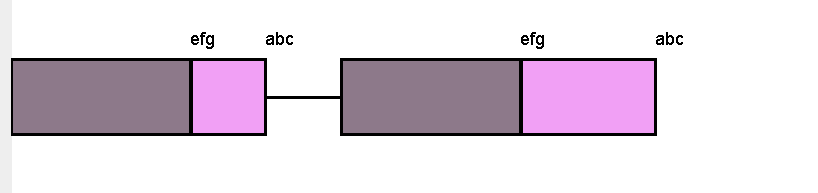


图25 添加空白进程效果图

## 基于语法的数据读入

该设计主要是针对文本的读取进行的，如图5，用户可在排班表系统的初始化窗口中选择导入文件，排班表系统会跟据正则表达式逐行读取文本中的内容，跟据文件设定好的格式，将排班起止时间，员工信息，排班记录等信息录入系统中当文件内容不合规时会产生一个警告窗（示例文件路径为src/File/working schedule.txt）

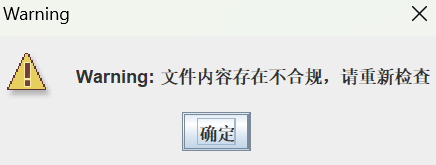


图26 文件不合规警告图

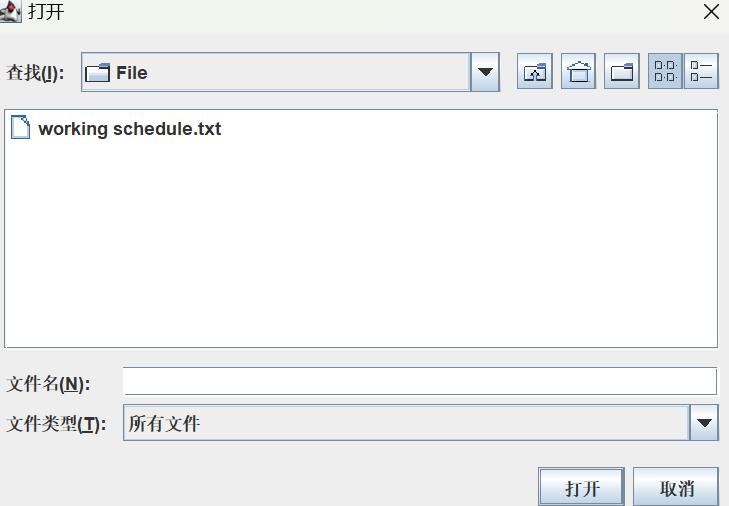


图27 导入文件窗口图

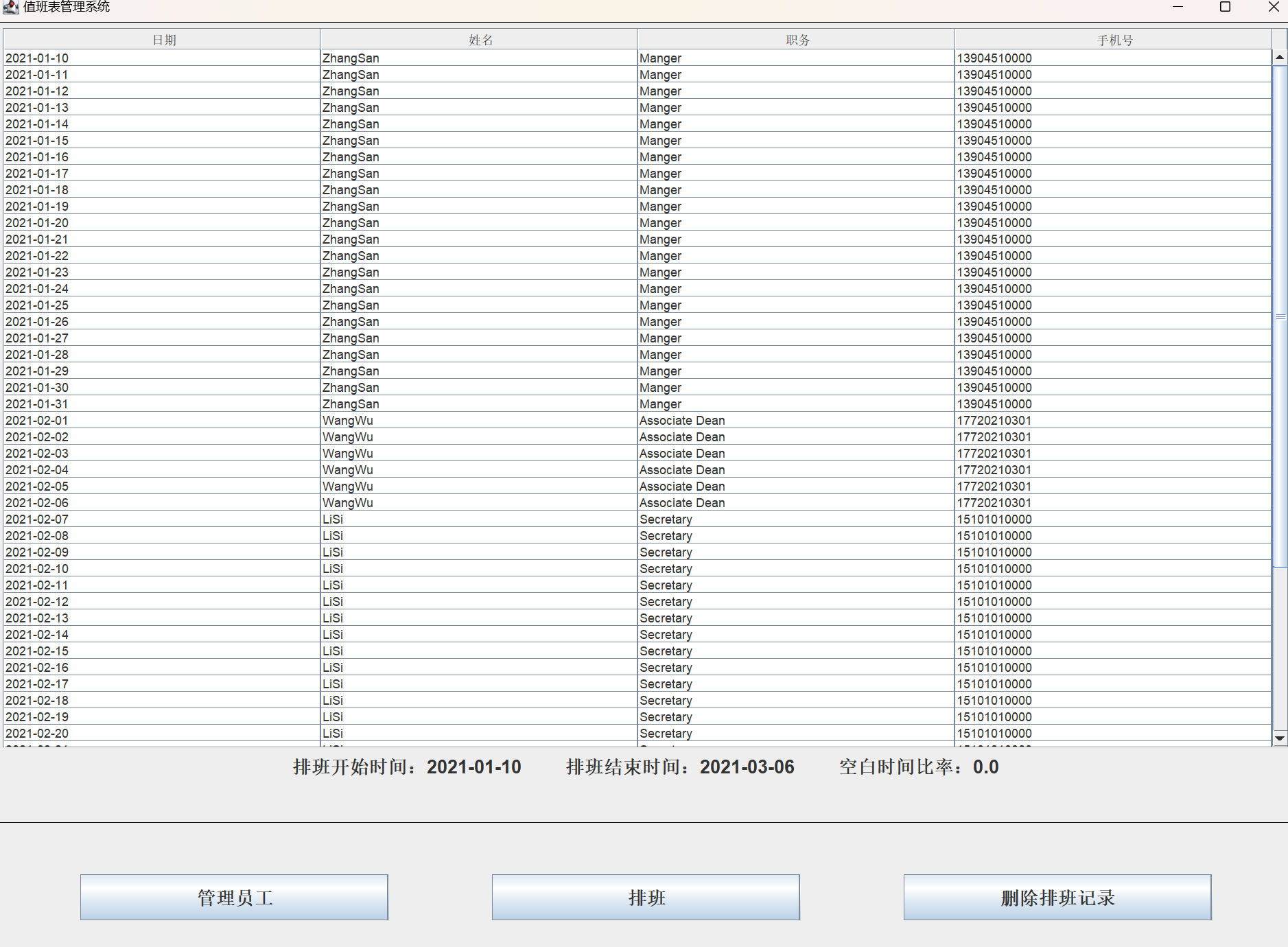


图27 导入文件结果窗口图

# 实验进度记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 2024.4.23 | 21：00-23：00 | 结合课件分析问题 | 完成 |
| 2024.4.24 | 14：00-17：00 | 编写Interval框架及测试 | 完成 |
| 2024.4.24 | 19：00-00：00 | 实现Interval和MultiInterval框架测试 | 提前完成 |
| 2024.4.25 | 19：00-00：00 | MultiInterval、 NoBlankIntervalSet、NonOverlapIntervalSet实现 | 推迟1h完成 |
| 2024.4.28 | 20：00-0：00 | PeriodicIntervalSet、IDutyIntervalSet部分实现、修改Interval等 | 修改原先部分rep，需要整体修改 |
| 2024.4.29 | 10:00-11:00 | 更新完成Interval等 | 按时完成 |
| 2024.4.29 | 21:30-23:00 | 编写装饰器并测试 | 推迟1h完成 |
| 2024.4.30 | 15:00-17:00 | 编写ProcessIntervalSet类 | 推迟1h完成 |
| 2024.5.2 | 10:00-12:00 | 编写ProcessIntervalSet类测试 | 按时完成 |
| 2024.5.2 | 14:00-18:00 | 编写CourselntervalSet、ICourselntervalSet | 按时完成 |
| 2024.5.2 | 20:00-00:00 | 编写Employee、Course、Proces | 未完成 |
| 2024.5.3 | 10:00-12:00 | 继续完成Employee、Course、Proces的编写并测试 | 按时完成 |
| 2024.5.3 | 14:00-19:00 | 编写IntervalVisitor、API并实现、测试 | 按时完成 |
| 2024.5.3 | 20:30-24:00 | 编写MultilntervalSetVisitor并实现、测试 | 按时完成 |
| 2024.5.4 | 10:00-12:00 | 编写framework | 未完成 |
| 2024.5.4 | 14:00-19:00 | 编写framework、operation | 未完成 |
| 2024.5.4 | 20:00-24:00 | 编写framework、operation | 按时完成 |
| 2024.5.5 | 10:00-12:00 | 编写DutyRosterApp | 未完成 |
| 2024.5.5 | 14:00-19:00 | 编写Adaptor、Adaptorlmpl、并测试，继续编写DutyRosterApp | Adaptor等按时完成  DutyRosterApp未完成 |
| 2024.5.5 | 20:00-24:00 | 编写DutyRosterApp | 按时完成 |
| 2024.5.6 | 8:00-12:00 | 编写ProcessScheduleApp  StrategyProcess及实现类 | StrategyProcess完成，  ProcessScheduleApp未完成 |
| 2024.5.6 | 21:30-24:00 | 编写ProcessScheduleApp | 按时完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 设计结构的困难，对上课讲的设计模式不能完全的理解与体会 | 通过反复实践、问GPT、反复钻研课件、查取资料、与同学讨论 |
| GUI框架的不熟悉 | 通过GPT，生成了简单的功能模块，通过对功能模块的学习，尽可能快速掌握GUI的框架思路 |
| 时间较为紧张 | 由于本次实验恰好赶在考试较多的时候，因此只能尽可能在五一期间挤时间 |

# 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：

1. 学会从0到1的系统搭建思路

2. 学会了如何将课堂上讲的设计模式落到实处

教训：

1.必须在构建系统之前想好这样设计能带来那些好处和哪些缺点

2.不能随意的使用设计模式，不能为了用而用，要用在适当的场合

## 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？

面向ADT编程尽管可能会在前期付出较多的努力，但这些努力可以为日后的开发打下坚实的基础，通过复用以前的开发成果可以尽可能的减少开发的难度与成本，而直接面向编程，是每次编程都要从零开始，这对每次的开发都是一个挑战，而当众多的任务堆叠起来，带来的后果是难以估量的。

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？

这些工作的作用是一层一层的保护自己的ADT的安全性与稳健性，在实际的复杂开发中，ADT是一个很复杂的树形结构，而specification, invariants, RI, AF等可以很有效的保护每个编写的节点的安全，当再开发新的节点出现了问题时可以不会危害到已经开发的结构，并且也能让新节点与已开发部分变得更加适配，我认为这样做尽管需要付出较多的努力，但我还是会选择坚持这样去做。

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？

一方面来说，开发API既要考虑自己的作为开发者的实现方式，另一方面来说，也要考虑用户使用的方便程度，这既是本身既是一种挑战又是一种服务于他人的乐趣，在看到你的API能够被大家接受和使用后，那种愉快与自豪是不可比拟的

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？

语法驱动是一个非常快捷、非常便利、非常高效的一种方式，能快速的将大量的数据导入系统中，同时也方便系统的数据进行保存与备份，这一点是手动输入远远无法企及的，同时，在导入数据的过程中也进行了内容的合法性检验，这也进一步提高了系统的稳定性。

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？

设计ADT困难的地方有很多，首先就是对于问题的抽象，如何从几个到多个问题中抽象出来那些最本质，最底层的方法，这是非常困难的一件事，尽管这次实验提供了思路，但这个问题仍是以后要更加关注的点和着重训练的目标。其次是实现方式的选择，在设计模式中为我们开发ADT提供了很多很好的思路，但如何正确、高效的选择实现的方法、结构等仍是需要我们需要长时间的积累，最后到框架的搭建等问题，如何搭建一个足够通用的框架、如何尽可能复用已有的代码与结构这是也是需要我们不断的去进行抽象的训练才能达到的水平，因此，为了克服这些困难还是要从不断的训练自己练习抽象的能力入手，尝试着多去分析问题，提取底层的框架，只有把这种能力内化于心才算真正了解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？

组合、继承将设计的复杂度与功能性都提升到了很高的水平，在进行继承关系时，本身要承受父类众多方法的覆写与自己新方法的实现，这对于一个实现类来说会变得越来越庞大、越来越复杂，也越来越难以复用，

而通过委派能很好的将功能组合起来，但委派的方式又有很多种，当一个类里进行了过多的委派后也会将整个功能变得十分紊乱与复杂，因此如何适度的使用继承与委派是一个持久性的课题

1. 关于本实验的工作量、难度、deadline。

本实验的难度相较于Lab2来说又是一个质的飞跃，由于是从0开始开发，经常会产生那种没有把握的感觉，在开发过程中显得就不如以往的实验来的简单与顺利，工作量来说也是进行了十几个接口与类的开发，对与每个模块都需要花很长的时间去进行摸索与实践，又恰巧赶上考试较多的时候，因此在本次实验中感到时间上并不充裕

1. 到目前为止你对《软件构造》课程的评价。

软件构造真的是一门非常有用的学科，这不仅对我的代码能力上有了质的飞跃，另一方面，也让我对如何设计、开发软件有了进一步的理解与体会，这是在上这门课之前是难以体会的。