

**2021年春季学期  
计算学部《软件构造》课程**

**Lab 3实验报告**

|  |  |
| --- | --- |
| 姓名 | 王子奕 |
| 学号 | 1190200121 |
| 班号 | 1903001 |
| 电子邮件 | 654483577@qq.com |
| 手机号码 | 15531580911 |

**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)

[3.2.2 局部共性特征的设计方案 2](#_Toc74666204)

[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 4-11 讲的内容，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

⚫ 语法驱动的编程、正则表达式

⚫ 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

前两次实验已经配置完

https://github.com/ComputerScienceHIT/HIT-Lab3-1190200121

# 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。

三个应用的共性：

1. 三者都是在一段时间上进行的活动。
2. 这段时间内进行的活动一定是连续的不可中断的
3. 每项label在每时刻的状态是确定的
4. 每个lable关联的信息不唯一例如（课表的课程信息、授课教师、授课教室、课时数等）。

三个应用的不同：

1. 首次安排的不确定性：duty与process的首次安排设计是不确定的，不一定何时轮到何人上班。course的安排是以周为周期性循环的。

②每次调度持续时间的单位长度不同：值班以天为单位，上课以两个课时为单位等。

1. 与第一条类似每次安排被中断后下次再进行的时间是未知的，可能要利用random随机得到，也可能是按照最近优先原则进行匹配。·但在course中授课教师知道下次上课时间。

## 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

在**IntervalSet**中，需要的操作方法有如下所示：

|  |  |  |
| --- | --- | --- |
| 函数名 | 方法 | 参数 |
| empty | 返回一个空的intervalset | 无 |
| insert (long start, long end, L label) | 向intervalset中插入新的interval | start的开始时间点  end的结束时间点  label的标签  返回true为插入成功  False为失败 |
| Remove (L label) | 在intervalset中移除lable对应的interval | label-所需移除interval的标签  true 为移除成功，false为失败 |
| Labels | 得到所有lables的一个集合，此处类型为<L>因为标签的格式不确定 | 无 |
| getStart | 返回指定lable对应地interval的开始时间 | label-所需要得到返回开始时间的interval对应的标签 |
| getEnd | 返回指定lable对应地interval的结束时间 | label-所需要得到返回结束时间的interval对应的标签 |
| Whole | 返回所有interval，构建一个list来存储 | 无 |

对AF，RI以及Safety from repexposure的论述为：

/\*\*

\*

\* Abstract Function:

\* Represents a map of interval, which has start time and end time. And

\* every interval has its own label to distinguish from each other

\* this is a mutable interval set with labeled intervals.

\*

\* Represent Invariant:

\* Each interval has its own start time and end time.

\* Also guarantee the start>=0 and end>start

\* And notice the label of each element is different

\* all Intervals have distinct labels of an immutable type.

\* and they must have a non-negative start and end at the same time end must later than start.

\*

\* Safety from rep exposure:

\* Each one of fields is modified by key word final and private so they can't be

\* accessed by outside of the class

\* And also do some defensive copying

\* @param <L> all types of labels in this interval set is immutable

\*/

### 局部共性特征的设计方案

局部共性的设计方案实现为**CommonIntervalSet**。**AF, RI**以及**Safety from rep exposure**的撰写同上。为了实现接口IntervalSet我们设计了这里的**CommonIntervalSet**的，实现的方法：

插入（insert），返回标签集合（labels），移除（remove），得到开始时间（getStart），得到结束时间（getEnd），得到所有interval（可排序）的whole等操作。

部分具体方法及规约实现如下：

以下是进行插入的操作，包括将时间段的开始时间，结束时间以及标签进行插入。

|  |
| --- |
| /\*\*  \* add a new interval to the interval set.  \*  \* @param start the start time of the interval(assume non-negative)  \* @param end the end time of the interval(assume non-negative) later than start time  \* @param label the label of the interval  \* @return true if this interval set has succeed added the interval;  \* false otherwise  \*/ |

**public boolean insert(long start, long end, L label)**

以下操作是返回labels集合

|  |
| --- |
| /\*\*  \* Get all intervals in this interval set.  \*  \* @return the set of intervals which in the interval set  \*/ |

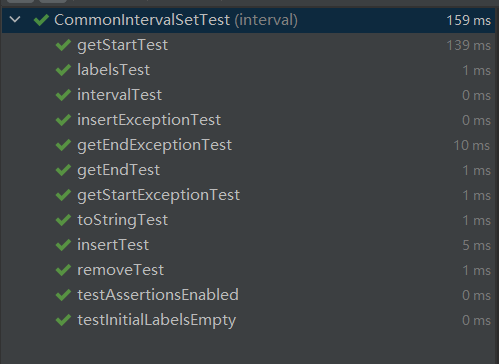
**public Set<L> labels()**

移除操作

|  |
| --- |
| /\*\*  \* Remove an interval from this interval set.  \*  \* @param label label of the interval to remove  \* @return true if this interval set included an interval with the given label;  \* otherwise false (and this interval set is not modified)  \*/ |

**public boolean remove(L label)**

进行test测试结果：



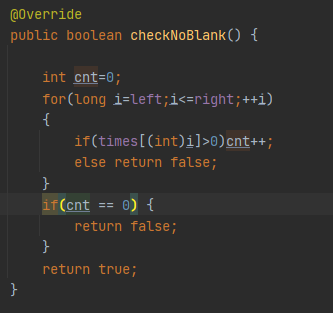
### 面向各应用的IntervalSet子类型设计（个性化特征的设计方案）

1、个性化的设计中主要强调的就是各个时间段之间是否存在blank（比如值班表应用中，就不能出现某天无人值班的情况，也就是不允许空白的出现）。

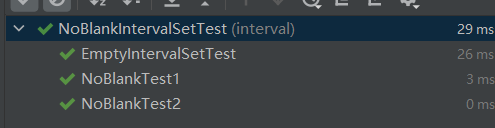
具体的操作设计就是，针对每个时间创建一个散列表，对每个人遍历他工作的时间，从start到end每一个位置上的散列表都加一。最后从最早的start扫描到最晚的end，若中间有某个位置为零则该处出现了Blank。

主要实现函数中对应的规约如下：

|  |
| --- |
| /\*\*  \* Abstraction function:  \* multiIntervals represents the set of all of interval  \* and every interval has it start time and end time  \* the numbers represents the times of the interval shows  \*  \* Representation invariant:  \* There must be no blank space between any intervals  \*  \* Safety from rep exposure:  \* All fields are private final, so the outside can't access them  \* The Set and Map objects in the rep are made immutable by unmodifiable wrappers.  \*/ |



测试结果：



2、个性化设计中再来就是是否允许不同的interval之间存在overlap（比如在值班表应用中，就不能出现某天多人值班的情况，而在进程执行过程中，某一时间段内也只能是一个进程来占据操作资源，不能出现多个进程“抢占”的情况）。

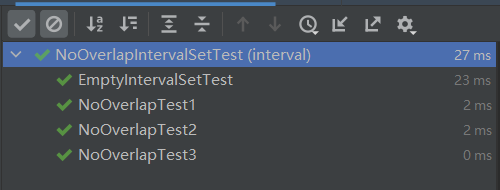
具体的操作设计就是，针对每个时间创建一个散列表，对每个人遍历他工作的时间，从start到end每一个位置上的散列表都加一。最后从最早的start扫描到最晚的end，若中间有某个位置大于一则该处出现了Overlap。

部分主要实现函数的规约如下：

|  |
| --- |
| /\*\*  \* Abstraction function:  \* multiIntervals represents the set of all of interval  \* and every interval has it start time and end time  \* the numbers represents the times of the interval shows  \*  \* Representation invariant:  \* There must not be overlap between the interval  \*  \* Safety from rep exposure:  \* All fields are private final, so the outside can't access them  \* The Set and Map objects in the rep are made immutable by unmodifiable wrappers.  \*/ |



测试结果：



3、个性化设计中还存在的一个方面是时间段是否存在周期性periodic

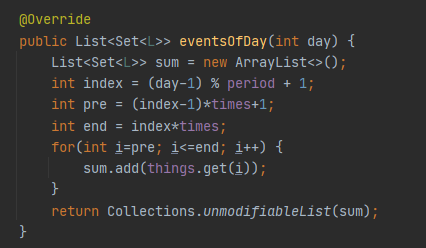
并且此设计对于courseapp中的一些数据计算也进行了实现

具体思想即把当前已有课程可未安排课程分别存储，针对不同需求进行操作，同时设计period和times两个参数维持周期的大小和单位时间的长短

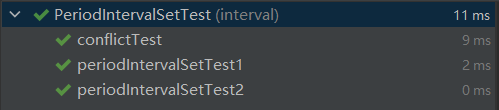
部分主要实现函数的规约如下：

|  |
| --- |
| /\*\*  \* AF:  \* period means the rotate  \* times means the number of course in a day  \* things contains all the events  \* RI:  \* collections all courses in a whole day to calculate whether it is periodic  \* Safety from rep exposure:  \* All fields are private final,  \* The Set and Map objects in the rep are made immutable by unmodifiable wrappers.  \*/ |

例如计算每一天的所有课程



测试结果如下：



## 面向可复用性和可维护性的设计：MultiIntervalSet<L>

### MultiIntervalSet<L>的共性操作

在**MultiIntervalSet**中，需要的操作方法有如下图所示：

|  |  |  |
| --- | --- | --- |
| 函数名 | 方法 | 参数 |
| empty | 返回一个空的intervalset | 无 |
| insert (long start, long end, L label) | 向intervalset中插入新的interval | start的开始时间点  end的结束时间点  label的标签  返回true为插入成功  False为失败 |
| Remove (L label) | 在intervalset中移除lable对应的interval | label-所需移除interval的标签  true 为移除成功，false为失败 |
| Labels | 得到所有lables的一个集合，此处类型为<L>因为标签的格式不确定 | 无 |
| getStart | 返回指定lable对应地interval的开始时间 | label-所需要得到返回开始时间的interval对应的标签 |
| getEnd | 返回指定lable对应地interval的结束时间 | label-所需要得到返回结束时间的interval对应的标签 |
| Whole | 返回所有interval，构建一个list来存储 | 无 |

对AF，RI以及Safety from rep exposure的论述为：

/\*\*

\*

\* Abstract Function:

\* Represents a map of multiinterval, which has start time and end time. And

\* every interval has its own label to distinguish from each other

\* this is a mutable interval set with labeled intervals.

\*

\* Represent Invariant:

\* Each interval has its own start time and end time.

\* Also guarantee the start>=0 and end>start

\* And notice the label of each element is different

\* all Intervals have distinct labels of an immutable type.

\* and they must have a non-negative start and end at the same time end must later than start.

\*

\* Safety from rep exposure:

\* Each one of fields is modified by key word final and private so they can't be

\* accessed by outside of the class

\* And also do some defensive copying

\* @param <L> all types of labels in this interval set is immutable

\*/

### 局部共性特征的设计方案

局部共性的设计方案实现为**CommonMultiIntervalSet**。**AF, RI**以及**Safety from rep exposure**的撰写同上。为了实现接口IntervalSet我们设计了这里的**CommonIntervalSet**的，实现的方法：

插入（insert），返回标签集合（labels），移除（remove），得到开始时间（getStart），得到结束时间（getEnd），得到所有interval（可排序）的whole等操作。

部分具体方法及规约实现如下：

以下是进行插入的操作，包括将时间段的开始时间，结束时间以及标签进行插入。

|  |
| --- |
| /\*\*  \* add a new interval to the interval set.  \*  \* @param start the start time of the interval(assume non-negative)  \* @param end the end time of the interval(assume non-negative) later than start time  \* @param label the label of the interval  \* @return true if this interval set has succeed added the interval;  \* false otherwise  \*/ |

**public boolean insert(long start, long end, L label)**

以下操作是返回labels集合

|  |
| --- |
| /\*\*  \* Get all intervals in this interval set.  \*  \* @return the set of intervals which in the interval set  \*/ |

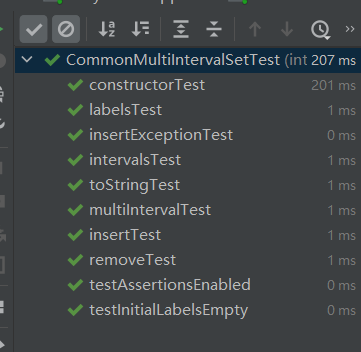
**public Set<L> labels()**

移除操作

|  |
| --- |
| /\*\*  \* Remove an interval from this interval set.  \*  \* @param label label of the interval to remove  \* @return true if this interval set included an interval with the given label;  \* otherwise false (and this interval set is not modified)  \*/ |

**public boolean remove(L label)**

进行test测试结果：



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

同3.2.3部分，也是实现的是是否允许空白，是否允许覆盖，是否有周期性

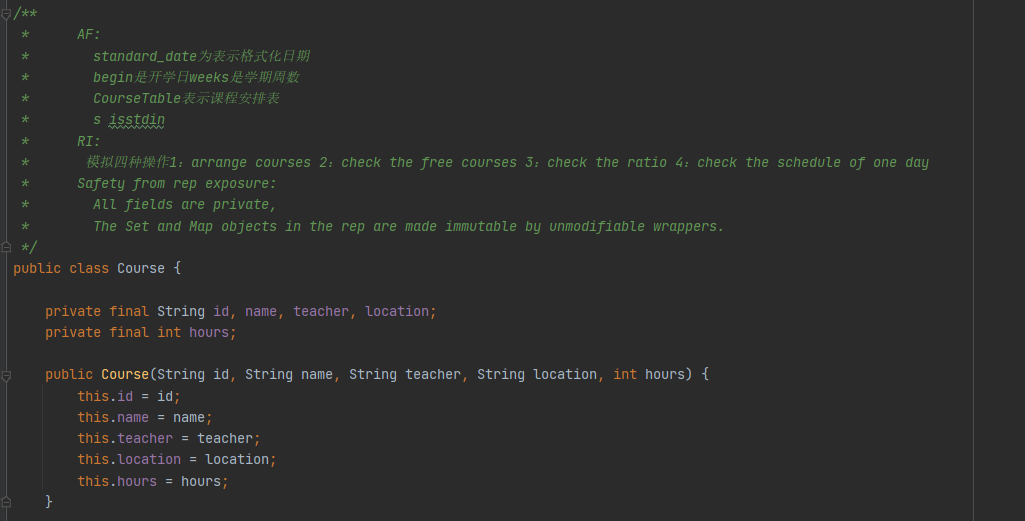
## 面向复用的设计：L

这里面L是一个泛型类型，含义是每个**interval**所用到的标签。在抽象层面的设计进行时用L进行抽象化，在具体类设计时将L赋为具体的类型例如，**Employee、Process、Course**分别为值班人员、进程、课程的标签。

**interval**进行过程中可以依据相应的指定标签进行插入项、移除项、获取项的开始时间与结束时间。这样可以在指定类型下不断增加相应的个性化信息例如课程的地点、授课老师、课时等等，从而丰富代码。

这种泛型的实现就体现了可复用性的设计思想，抽象层面的类可以用来复用，只需要将其中的泛型L进行相应的修改即可。

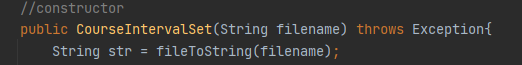
1、课程的复用设计：



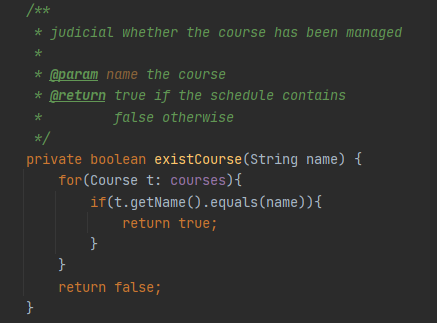
设计的内容包括了课程编码、上课的教室、授课教师、课程名字以及上课时长等信息。

部分涉及到的具体方法：

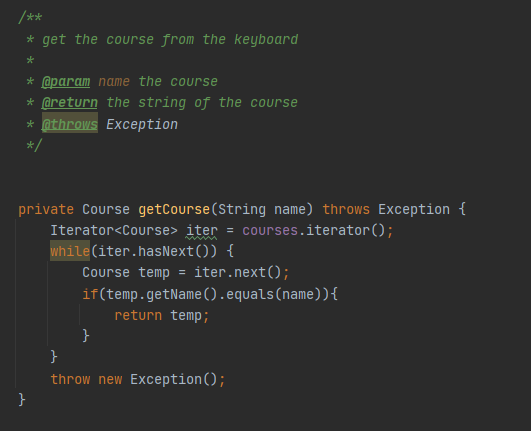
1. 从文件读入初始的数据集（正则语法）



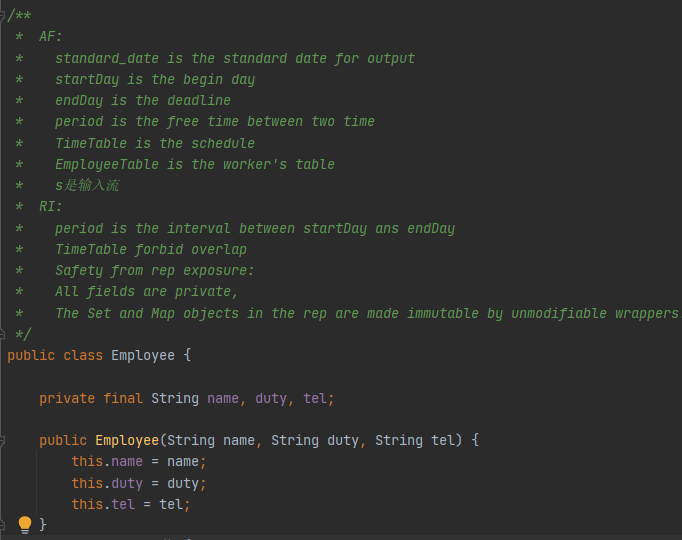
1. 判断课程是否已经加入



1. 获取课程的信息等getname之类的函数



2、值班人员的复用设计：



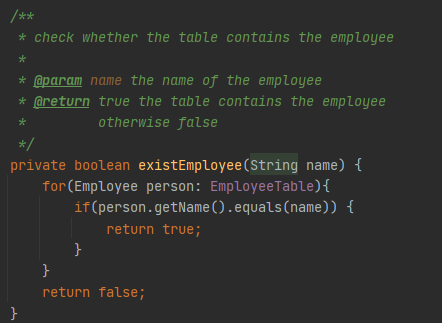
设计的内容包括了值班人员的姓名、职务电话等信息

部分具体方法包括：

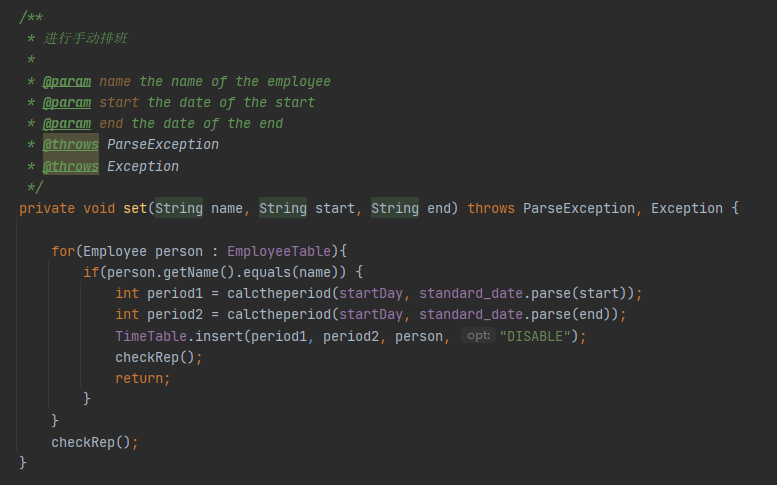
1 从文件读入初始的数据集（正则语法）



2 检查该员工是否存在于已排好的表中

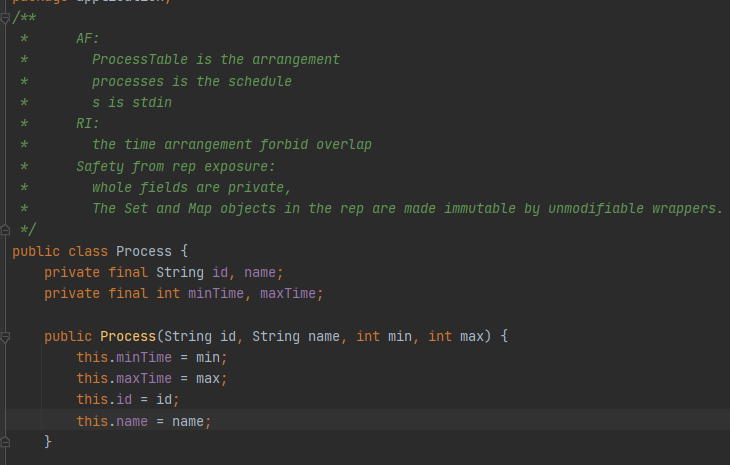


3 手动排班包装一个人的工作信息



4获取雇员相应的信息详见APP部分

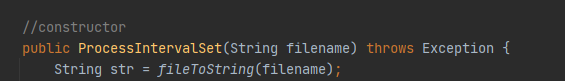
3、进程的复用设计：



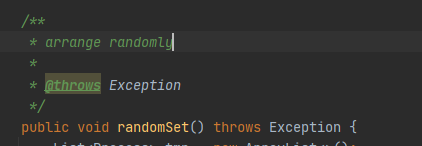
设计的内容包括进程的ID，名字，最短执行时间以及最长执行时间等信息。

部分具体操作有：

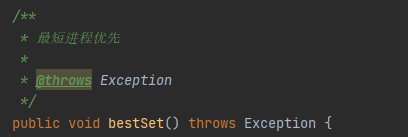
1 从文件读入初始数据集（正则语法）



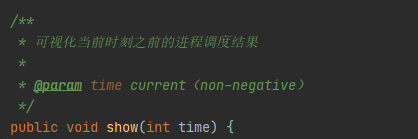
2 设置随机排班，得到随机进程调度表



3 最短进程优先的模拟策略：每次选择进程的时候，优先选择距离其最大执行时间差距最小的进程。



4 可视化当前时刻之前的进程调度结果 注意时间必须为非负数

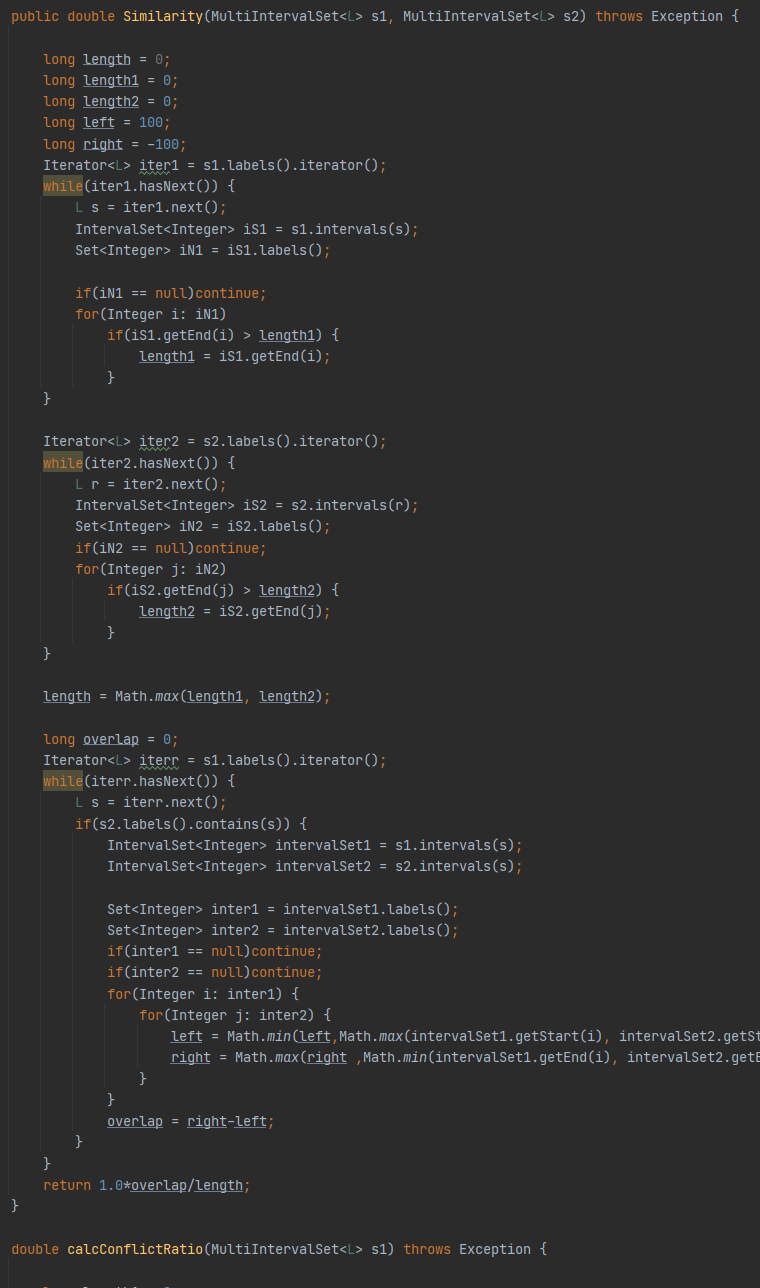


## 可复用API设计

### 计算相似度

下述相似度的计算方法：具体计算方法：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。

思想：先在两个set进行遍历中找到所需要的最长的时间段，作为答案的分母，分别得到其总的时间段，之后再使用max函数得到最长的长度。



在找到总时间段之后，以overlap的时间段作为分子，对标签重合的部分相似度为一，那么就需要找到overlap中最晚的起始时间与最早的结束时间作为左右端点，两者的差值作为重合度：

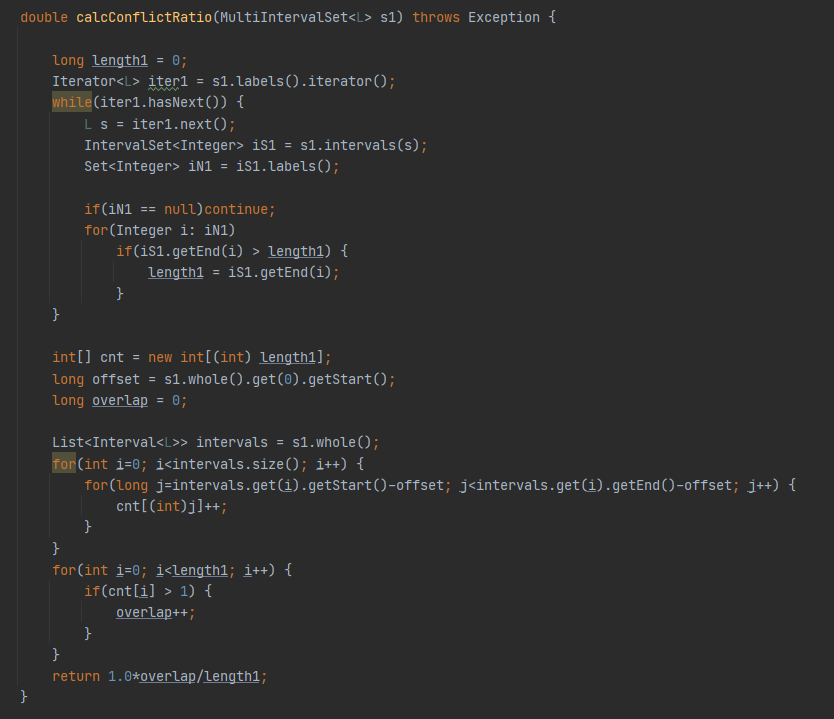


这样重合度除以找到的总时间段就是我们需要的两个段的相似度。

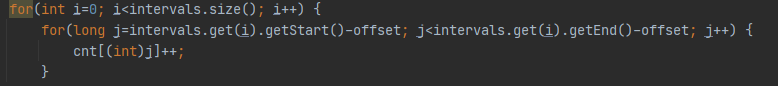
### 计算时间冲突比例

所谓的“冲突”，是指同一个时间段内安排了两个不同的 interval 对象。用发生冲突的时间段总长度除于总长度，得到冲突比例，介于[0,1]之间。

思想：同上找到总的时间段长度：



在找到总的时间长度之后，需要寻找冲突的时间段长度，采用散列表方法来进行计数，



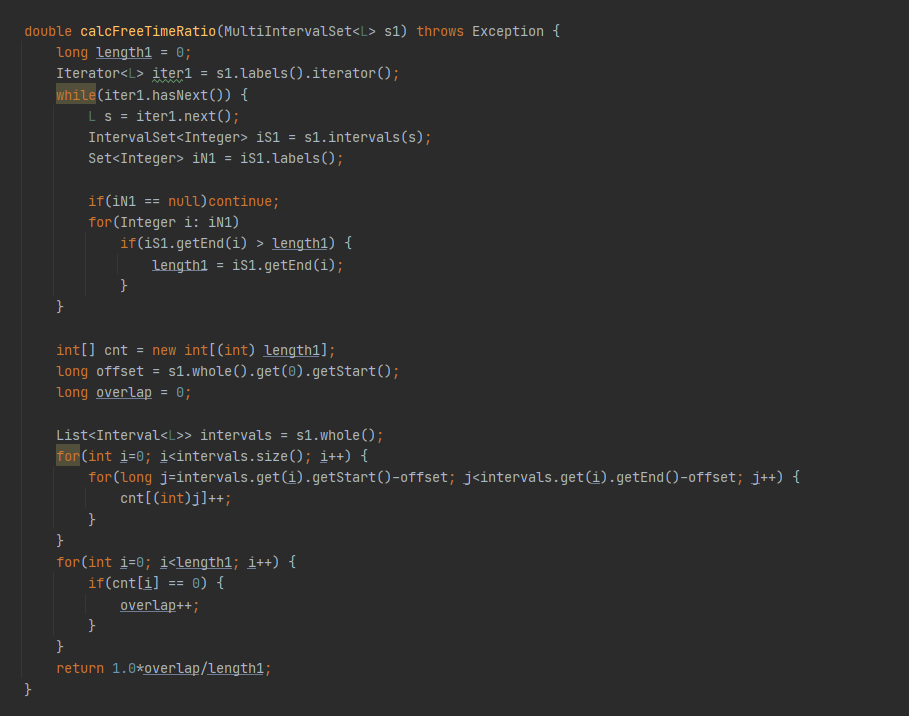
在冲突的地方就进行加一统计操作。

### 计算空闲时间比例

所谓的“空闲”，是指某时间段内没有安排任何 interval 对象。用空闲的时间段总长度除于总长度，得到空闲比例，介于[0,1]之间。

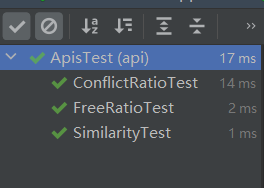
思想：同上计算总的时间段长度：

在找到总的时间长度之后，需要寻找冲突的时间段长度，继续采用“散列表”方法来进行计数，



在空闲的地方就进行加一操作。

进行测试结果如下



## 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

在初始时的文本中我们可以进行相应的排班的开始日期、结束日期的设置，并且给出值班人员的信息：姓名，职位，手机号码。

设计思想

/\*\*

\* AF:

\* standard\_date is the standard date for output

\* startDay is the begin day

\* endDay is the deadline

\* period is the free time between two time

\* TimeTable is the schedule

\* EmployeeTable is the worker's table

\* s是输入流

\* RI:

\* period is the interval between startDay ans endDay

\* TimeTable forbid overlap

\* Safety from rep exposure:

\* All fields are private,

\* The Set and Map objects in the rep are made immutable by unmodifiable wrappers.

\*/

用户可在命令行中进行可视化操作 /\*\*

\* according to the input to choose the module(must 0-4)

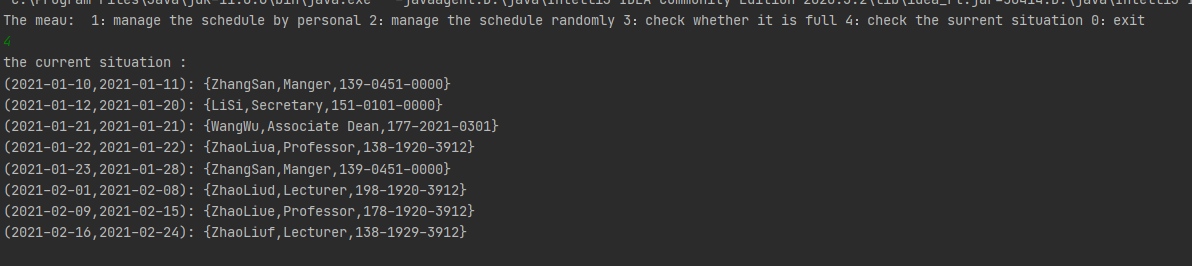
\* 1：manage the schedule by personal 2：manage the schedule randomly 3：check whether it is full 4：check the surrent situation 0：exit

\*

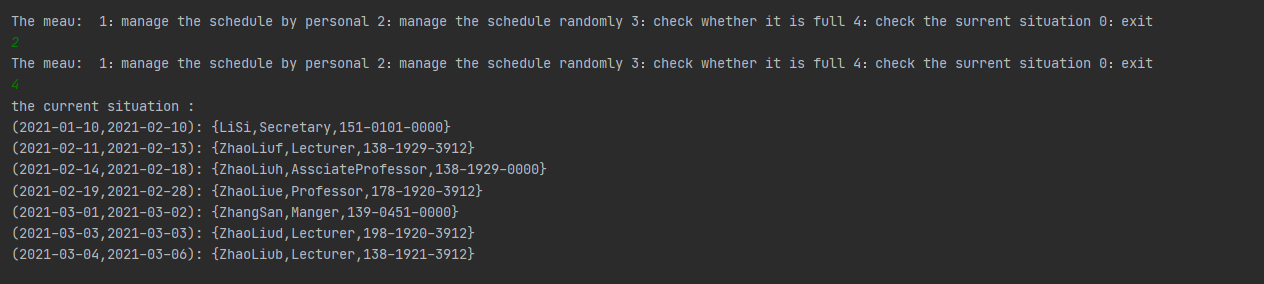
\* @throws Exception

\*/

第一步查看当前排班情况

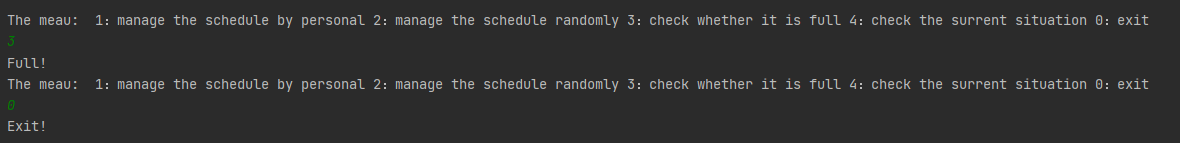


第二步随即排班并查看结果

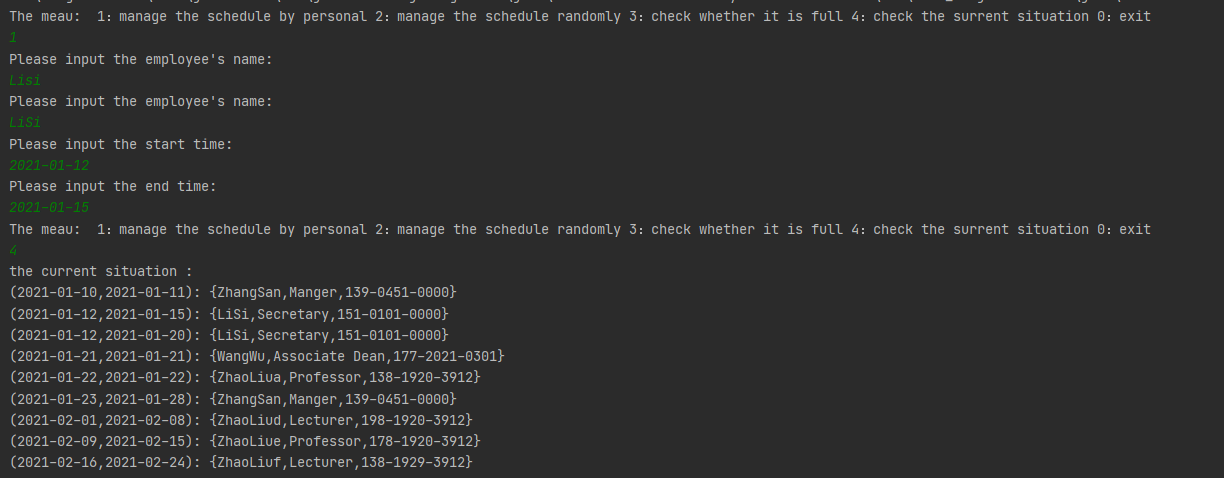


观察两步的不同

输入3检查是否班满了，0退出



此后，我们可以进行手动排班：输入1，即进行手动排班，让李四去上班（1-12到1-15）



可以看到程序对于输入的健壮性也进行了保护，即输入名字不合法时会提示重新输入

观察输入4后的结果可以发现增加了如下一段排班



### 操作系统的进程调度管理系统

同样地，本次设置进程调度时也是从文本文件读入初始的数据，当然也可以在文本文件中修改相应的进程信息（正则语法）。

设计的具体思路：

/\*\*

\* AF:

\* ProcessTable is the arrangement

\* processes is the schedule

\* s is stdin

\* RI:

\* the time arrangement forbid overlap

\* Safety from rep exposure:

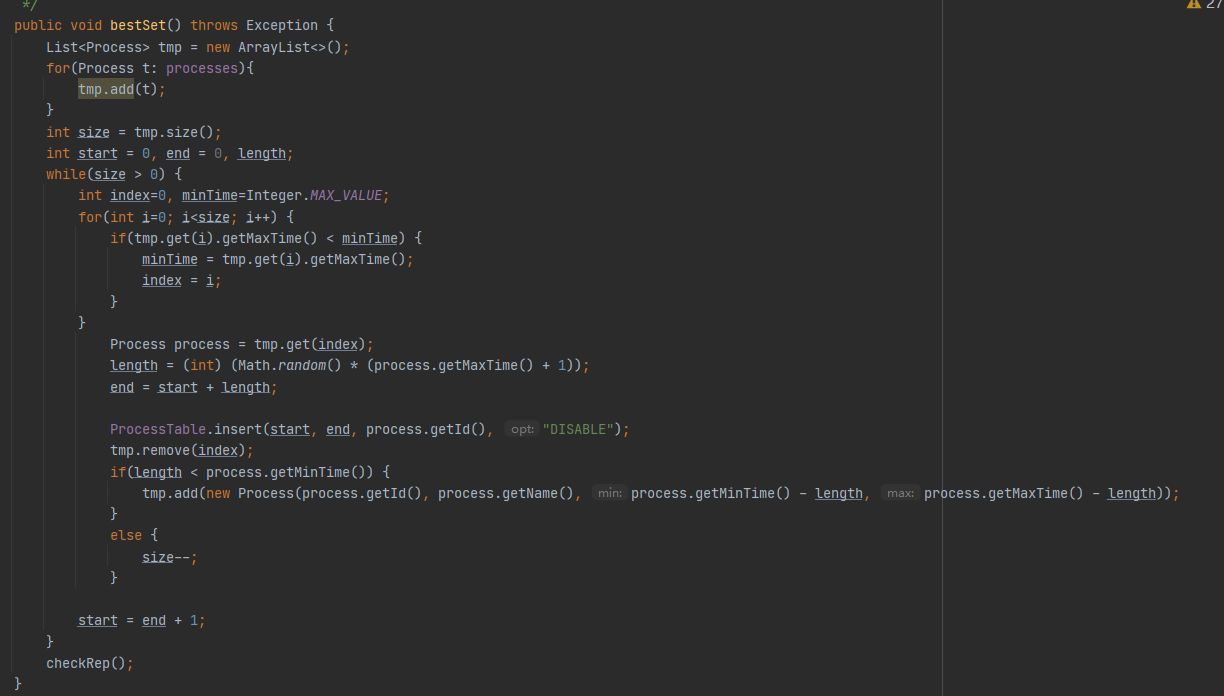
\* whole fields are private,

\* The Set and Map objects in the rep are made immutable by unmodifiable wrappers.

\*/

最短进程优先：设计一个tmp存放可运行进程，每次扫描集合取出优先选择距离其最大执行时间差距最小的进程，根据当前剩余时间来选择，若剩余时间不够落入最短进程周期则将其还回到原tmp，若可以落入就把他在tmp删除并更新size和当前start，end

随机算法：每次随机一个进程，再在它能执行的剩余时间内rand一个长度，判断和之前最短进程优先一样，这里不再赘述



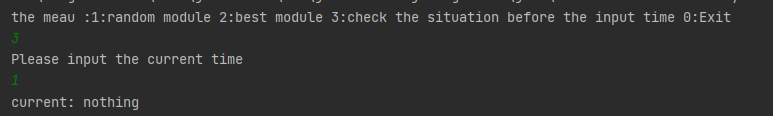
共设计三个选项提供给用户

the meau :1:random module 2:best module 3:check the situation before the input time 0:Exit

1 随机安排 2 最优安排 3 查看给定时间前进程情况 0 退出

可视化运行结果如下

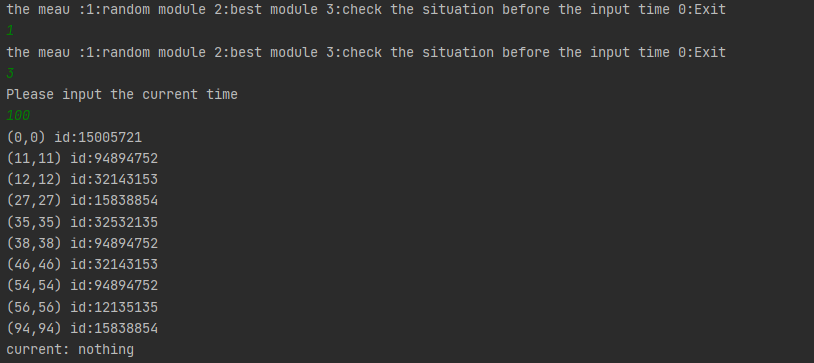
下面开始进行执行：

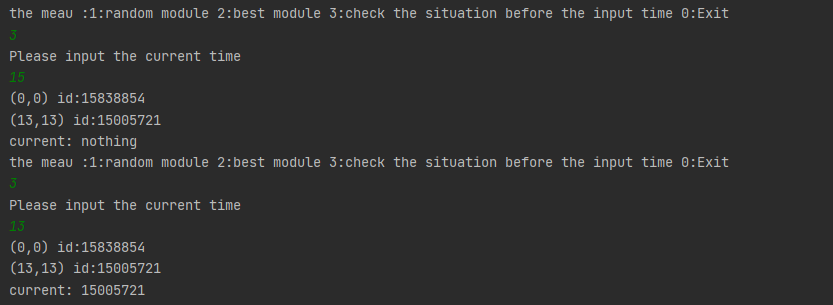


观察到此时刻前没有进程执行

当我们随机安排进程的时候，进程开始随机进行。

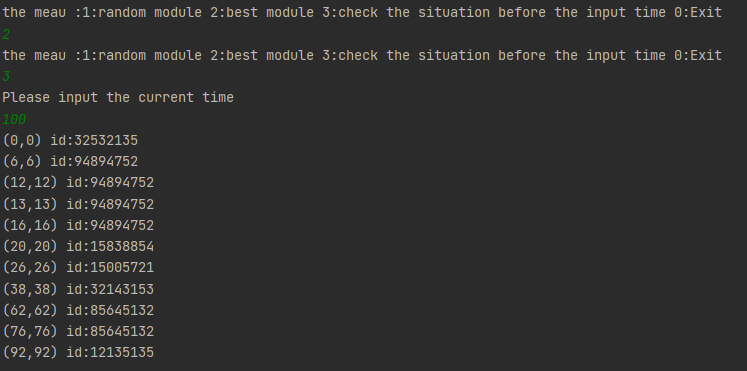
进行1操作并利用3检查

若时刻数落在一个区间内，还会输出当前进程pid



“最短进程优先”的模拟策略：每次选择进程的时候，优先选择距离其最大执行时间差距最小的进程

输入2并利用3检查，和之前对比发现结果正确



### 课表管理系统

同样地，本次设置课表时也是从文本文件读入初始的数据，当然也可以在文本文件中修改相应的课程信息（正则语法）。初始执行程序时，得到的页面如下图所示：即查看了当前未安排的课程的情况。在此处课程ID由5个数字组成的，课程名是由英文字母组成的（中间无空格），之后的授课教师、教室以及课时数都可以给出。

设计思想

/\*\*

\* AF:

\* standard\_date为表示格式化日期

\* begin是开学日weeks是学期周数

\* CourseTable表示课程安排表

\* s isstdin

\* RI:

\* 模拟四种操作1：arrange courses 2：check the free courses 3：check the ratio 4：check the schedule of one day

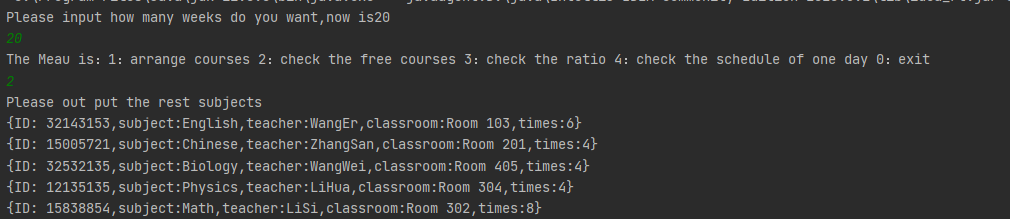
\* Safety from rep exposure:

\* All fields are private,

\* The Set and Map objects in the rep are made immutable by unmodifiable wrappers.

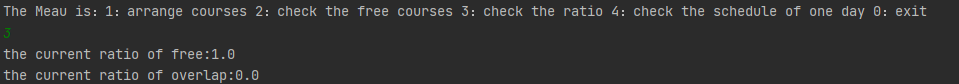
\*/

没什么好说的模拟即可，每次计算剩余课时，按照计算方法调用process方法处理ratio的计算



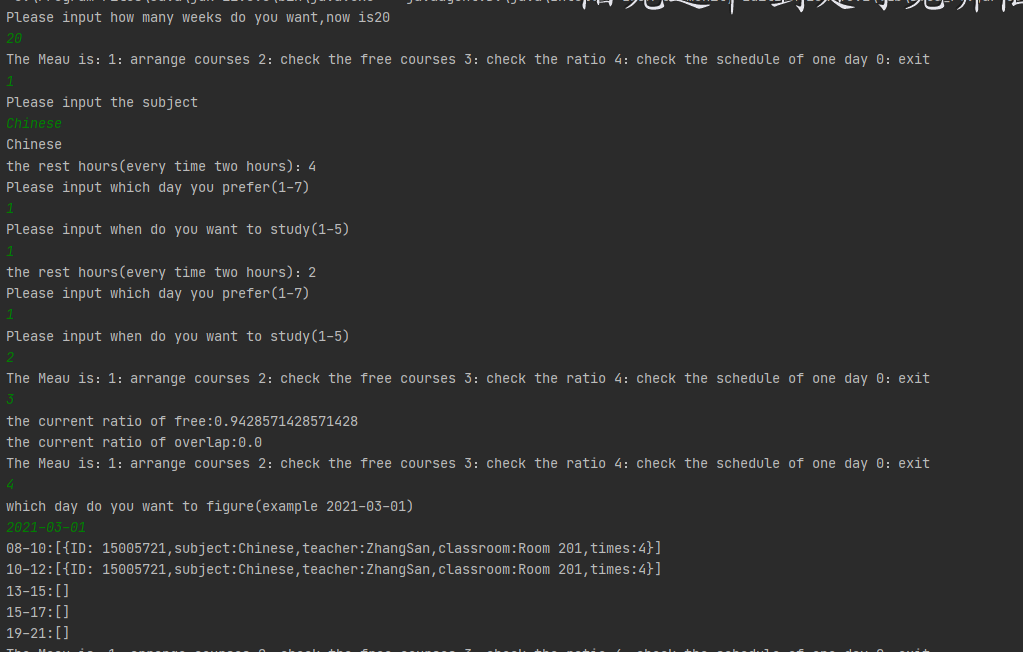
首先输入计划的学期周数，同时给出默认的学期周数

选择2查看现在可以选的课的信息

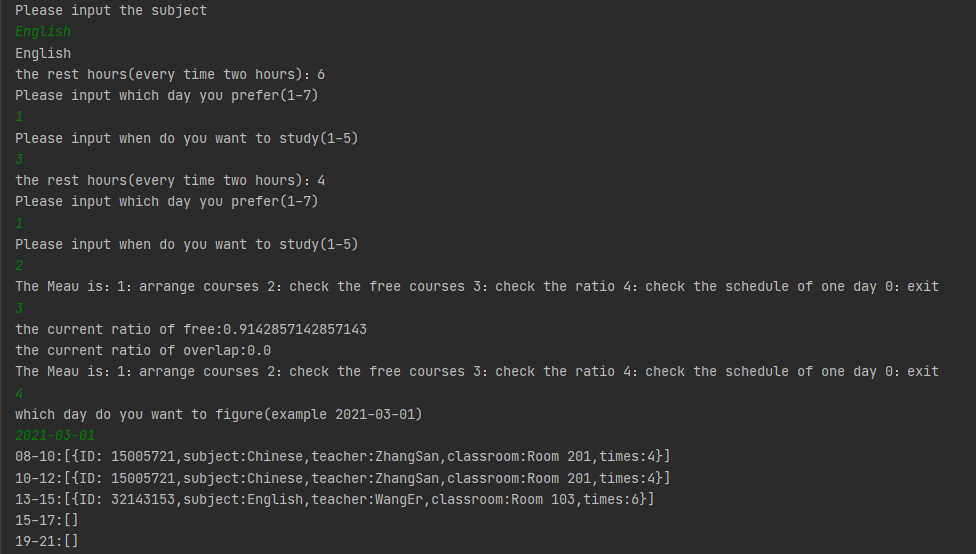


检查当前空闲率为1，冲突率为0

手动安排一门课



安排一节Chinese把四课时的课分两次安排进去，最后查询课程时间，发现课表和空闲率都有所更改。



添加一门新的English原理如上

## 基于语法的数据读入

针对3.6节中开发的“排班管理”应用，为其扩展一个功能从一个外部文本文件读入数据并使用正则表达式 parser对其进行解析，从中可抽取到的是雇员、课程、进程的信息。

对于此模块专门设计ParseInput特殊处理八个test。

主要利用matcher和pattern两个类，在java.util.regex包中下面是对二者的介绍

Pattern 类：

pattern 对象是一个正则表达式的编译表示。Pattern 类没有公共构造方法，要创建一个 Pattern 对象，要利用其公共静态编译方法，同时其返回一个 Pattern 类对象。pattern() 返回正则表达式的字符串形式,其实就是返回Pattern.complile(String regex)的regex参数

Matcher 类：

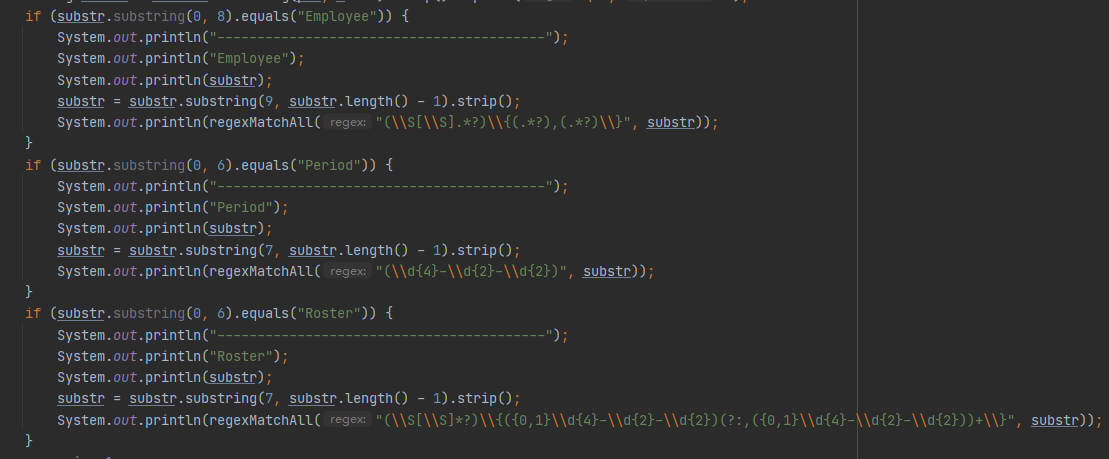
Matcher 对象是对输入字符串进行解释和匹配操作的引擎。与Pattern 类一样，Matcher 也没有公共构造方法。Pattern.matcher(CharSequence input)返回一个Matcher对象。Matcher类的构造方法也是私有的,不能随意创建,只能通过Pattern.matcher(CharSequence input)方法得到该类的实例.

Pattern类只能做一些简单的匹配操作,要想得到更强更便捷的正则匹配操作,那就需要将Pattern与Matcher一起合作.Matcher类提供了对正则表达式的分组支持,以及对正则表达式的多次匹配支持。

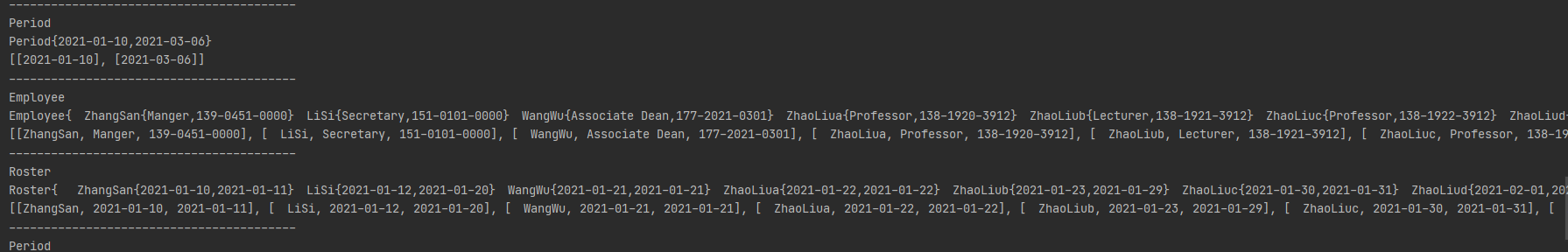
程序处理的关键在于首先处理中英文全半角空格符区别处理



针对三类读入分别处理



以下为部分运行结果：



## 应对面临的新变化

### 变化1

排班应用：可以出现一个员工被安排多段值班的情况。可以选用MultiIntervalSet的方式，即同一个laibel可以在多个时间段上出现。只需要将涉及到的IntervalSet申明增加改为MultiIntervalSet申明就可实现。

例如再noblank中

public NoBlankIntervalSetImpl(MultiIntervalSet<L> initial) throws Exception {

multiIntervals = initial;

Iterator<L> iter = multiIntervals.labels().iterator();

while(iter.hasNext()) {

L label = iter.next();

List<Interval<Integer>> tempIntervals = multiIntervals.intervals(label).whole();

for(int i=0; i<tempIntervals.size(); i++) {

for(long j=tempIntervals.get(i).getStart(); j<=tempIntervals.get(i).getEnd(); j++) {

numbers.add(j);

}

}

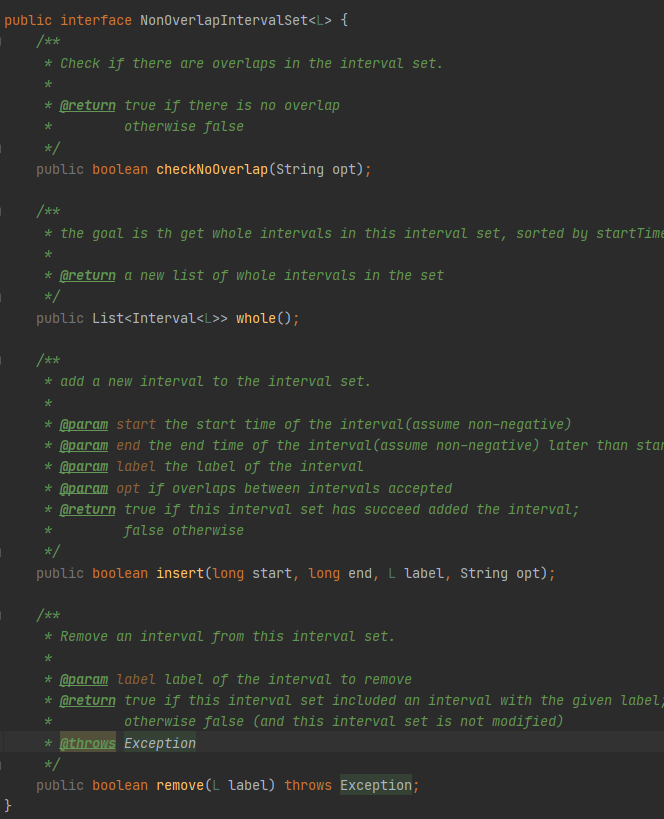
}

}

### 变化2

课表应用：不管学生选课状况如何，不能够出现两门课排在同一时间的 情况（即“无重叠”）

即NonOverlap模块中的情况，通过对NonOverlapIntervalSet接口的调用，可轻松处理

。

## 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。

以下给出了 Git 的相关操作指南。

…最初在 master 上工作…

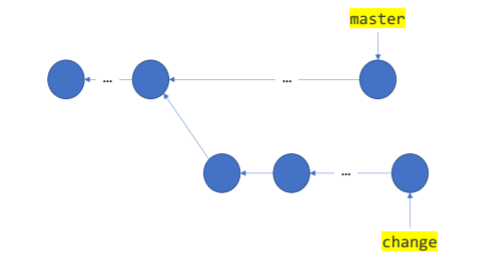
git checkout -b change 创建新分支

…按上面的要求进行代码修改…

git add \*

git commit -m "change"在该分支上提交

git checkout master 切换回 master 分支



# 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 |
| 2021-06-20 | 18:00-21:00 | 阅读实验手册 | 完成 |
| 2021-06-21 | 18:00-21:00 | 请教同学查阅资料归纳设计所有类以及他们的作用 | 完成 |
| 2021-06-23 | 18:00-21:00 | 建立并设计两个接口（interval\multiinterval）和两个类 | 完成 |
| 2021-06-24 | 14:00-17:00 | 对CommonIntervalSet的设计完成 | 完成 |
| 2021-06-26 | 18:00-21:00 | 对CommonMultiIntervalSet的设计完成 | 完成 |
| 2021-06-28 | 18:00-20:00 | 规划三个维度上的差异和共同点 | 延期 |
| 2021-06-30 | 18:00-20：00 | Tongs | 完成 |
| 2021-07-01 | 19:00-23:00 | 完成对Duty的ADT设计 | 完成 |
| 2021-07-02 | 18:00-21:00 | 完成对Course的ADT设计 | 完成 |
| 2021-07-02 | 21:00-24:00 | 完成对Process的ADT设计 | 完成 |
| 2021-07-03 | 12:00-21:00 | 可视化实现 | 完成 |
| 2021-07-04 | 12:00-15:00 | Parse正则表达式设计 | 完成 |
| 2021-07-04 | 15:00-21:00 | 撰写实验报告 | 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 |
| 设计模式的选取与理解，对于可复用性好的设计模式，一旦选取就不能改变否则将整个颠覆程序，因此构思好出发点与思路十分重要 | 阅读完实验手册后先查询了笔记和书籍 ，通过咨询同学和老师，浏览博客积累了之前的知识。理清思路了再进行代码设计。同时对每个模块进行test测试后再往后进行，避免了大规模的漏洞与错误。 |
| 命令行提示信息中文出现乱码 | 读入信息和输出提示信息编码出现混乱，最终全篇设计采用英文 |
| 上层设计和底层设计出现矛盾 | 冷静思考，绘制流程逻辑图，对接口、抽象类、类三层体系合理把控，理清思路从头修改 |
| 不理解如何可视化 | 查询资料询问同学，利用命令行实现人机交汇 |
| 针对存入列表等数据结构的信息出现丢失 | 仔细和同学debug发现自身设计时出现漏洞 |

# 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

1.本次实验主要利用课程所授3、4、5章的内容，目标是编写具有可复用性和可维护性的软件。主要面向可复用性进行了一次独立的项目设计与开发，核心是课上所讲的各种设计模式。而这种设计模式的思想在前后贯穿实验始终，经典的CRP模式构成了此次试验的核心骨架，通过前后各个特性的接口的模拟组合实现成一个通用模板的各个特殊性，是利用可复用性的一个典范。

另外，完成这些设计模式的一个重要手段是合理利用委托的思想，通过在底层实现的共性功能代码，得以在上层功能设计时可以直接委托利用底层类的实例，这样一来就能大大减少重复的代码量，从而使我切实体会到了复用性的设计的高效。

## 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？

二者在思维上存在较大的区别，需要有更好的前瞻性，对全局代码的把控能力。在复用层面，如果能把握好各模块的共性问题，就可以极大的减少代码的复杂程度。更高效的进行软件框架的设计。

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？

这些工作的意义很大，一方面便于自己清晰的了解代码的搭建结构思想，同时便于团队其他成员轻松理解你的思想，对程序的安全性也有很大保护。我愿意在未来的工作中坚持。

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？

无论是我们设计还是利用别人的api都是为了维护程序的稳定和安全，所以拥有好的api及其重要，而设计好的api也是为人民服务的一个体现，我自然乐在其中。

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？

语法驱动编程是为了让机器能自动去针对语法进行IO操作，本次实验我学到了正则如何处理，对工作有了极大的帮助。

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？

本次实验难在我要从头到尾进行自己设计，一旦在后期遇到什么问题就要反过头来从新处理，经常出现底层架构和自己后期的想法相悖，因此得出经验要考虑好再写。

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？

抽象的思想十分重要：接口>抽象类>具体类这一个观点贯穿了我们这个课程，我们所讲的各种设计模式但是其中共性的东西就是这个继承与委托，通过继承与委托可极大利用代码的可复用性。

1. 关于本实验的工作量、难度、deadline。

本次实验工作量和难度都超出前两次之和，时间比较紧，而且在期末。希望未来能提前。

1. 到目前为止你对《软件构造》课程的评价。

本课程作为本科阶段首个大规模编程设计实现的课，起初我单单理解为码农行为，但在后期的学习中我渐渐发现老师为我们提供的更多的是思想，是对代码设计的一些框架。授人以鱼不如授人以渔，通过学习这些体系架构和一次次实验的磨练，我能感到自己的实力在不断提升，同时也渐渐喜欢上了写代码！感谢这门课程的所有老师和助教！