

**2021年春季学期  
计算学部《软件构造》课程**

**Lab 3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**

目录

[1 实验目标概述 1](#_Toc243157127)

[2 实验环境配置 1](#_Toc506560349)

[3 实验过程 2](#_Toc2135607738)

[3.1 待开发的三个应用场景 2](#_Toc1876831314)

[3.2 ADT设计 3](#_Toc1083864463)

[3.2.1 IntervalSet接口设计 4](#_Toc1237955772)

[3.2.2 CommenIntervalSet的实现 6](#_Toc1502034621)

[3.2.3 IntervalSetDecorator类 6](#_Toc371205394)

[3.2.4 NoBlankIntervalSet 7](#_Toc1295746545)

[3.2.5 UniqueIntervalSet 7](#_Toc1074791662)

[3.2.6 OverlapIntervalSet 8](#_Toc576240354)

[3.3 可复用API设计 8](#_Toc1309101255)

[3.3.1 计算相似度 8](#_Toc1173292664)

[3.3.2 计算时间冲突比例 8](#_Toc1941666972)

[3.3.3 计算空闲时间比例 8](#_Toc234306919)

[3.4 应用设计与开发 9](#_Toc1369984602)

[3.4.1 排班管理系统 9](#_Toc948733229)

[3.4.2 操作系统的进程调度管理系统 10](#_Toc82434876)

[3.4.3 课表管理系统 12](#_Toc168291645)

[3.5 基于语法的数据读入 13](#_Toc1462889259)

[3.6 应对面临的新变化 14](#_Toc1637421223)

[3.6.1 变化1 14](#_Toc33709955)

[3.6.2 变化2 15](#_Toc121352629)

[3.7 Git仓库结构 15](#_Toc482035309)

[4 实验进度记录 16](#_Toc1483325993)

[5 实验过程中遇到的困难与解决途径 17](#_Toc1908471203)

[6 实验过程中收获的经验、教训、感想 17](#_Toc620162885)

[6.1 实验过程中收获的经验和教训 17](#_Toc951482837)

[6.2 针对以下方面的感受 17](#_Toc1988082090)

# 实验目标概述

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

给定三个具体的应用，通过ADT和泛型等抽象技术，开发一套可服用的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

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

使用token作为github push的验证方式不稳定，后使用ssh

1. GitHub Lab3仓库URL地址（Lab3-学号）

https://github.com/ComputerScienceHIT/HIT-Lab3-1191000606

# 实验过程

## 待开发的三个应用场景

首先是应用一，设计值班表，有以下属性：

1. 要全部排满
2. 不运行一个人的排班时间段不连续
3. 每个时间段有且仅有一个人值班

应用二，操作系统：

1. 一个进程运行时间可以不连续
2. CPU可以空闲
3. 采用随机的方法

应用三，课表：

1. 一个时间段运行有多个课程，但是老师，上课地点不能重复。每周课表均相同。
2. 允许课程多次出现，甚至要求所有课程的学时都安排够。

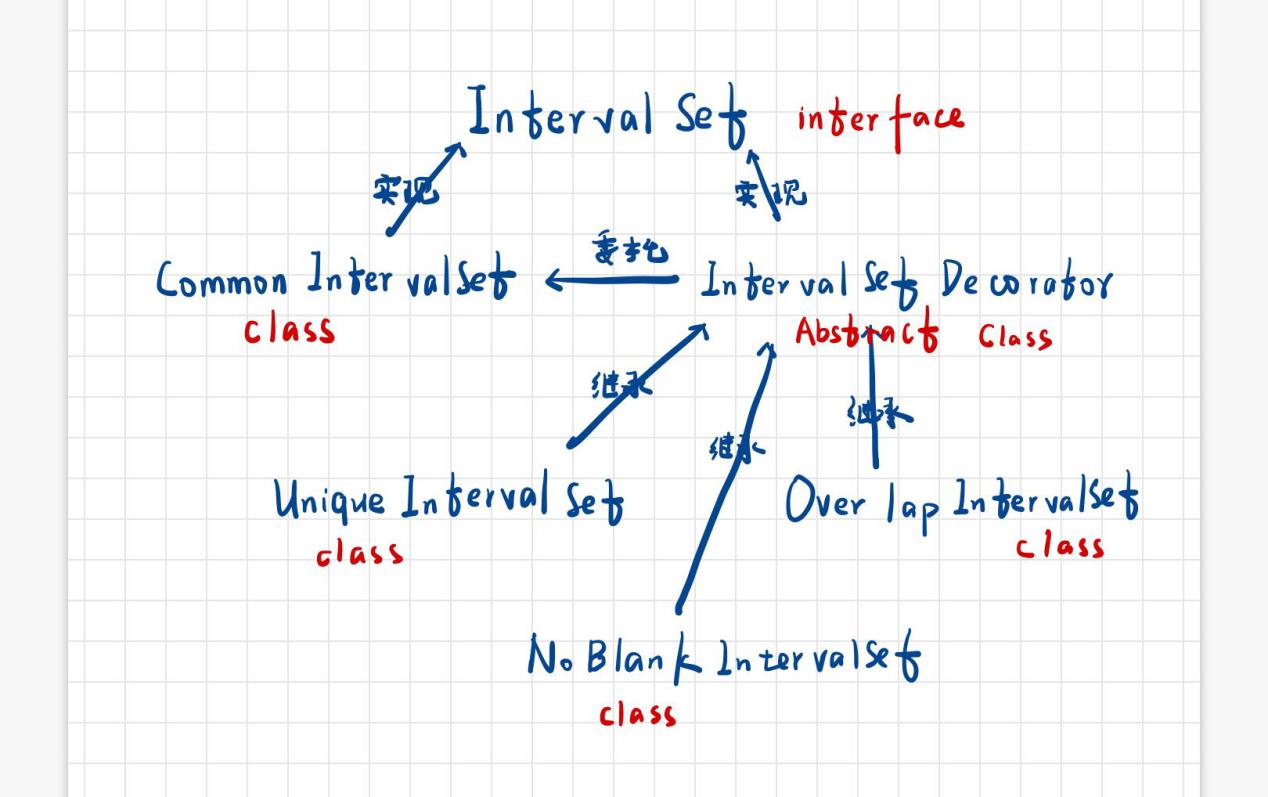
首先三个应用共性的方面，都有一个时间段，要填充相应的label。关于填充方式有很多种选择。总体而言有四个维度：

1. 是否允许不连续
2. 是否允许重叠
3. 是否允许空白
4. 是否有循环

上述三个应用中主要的ADT主要是以上四个维度的差异。

## ADT设计

实验指导书中提到，可以不受本节要求的限制,使用一个 ADT 来替代这两个 ADT。因此我的ADT主要结构如下：



很显然，这是装饰模式。

基础数据类型CommenIntervalSet是这样的：

1. 可以有空白
2. 不可以重叠
3. 可以不连续
4. 不存在周期

UniqueIntervalSet可以将之装饰为必须连续。NoBlankSet可以将之装饰为必须无空白，OverlapIntervalSet可以将之装饰为可以重叠。

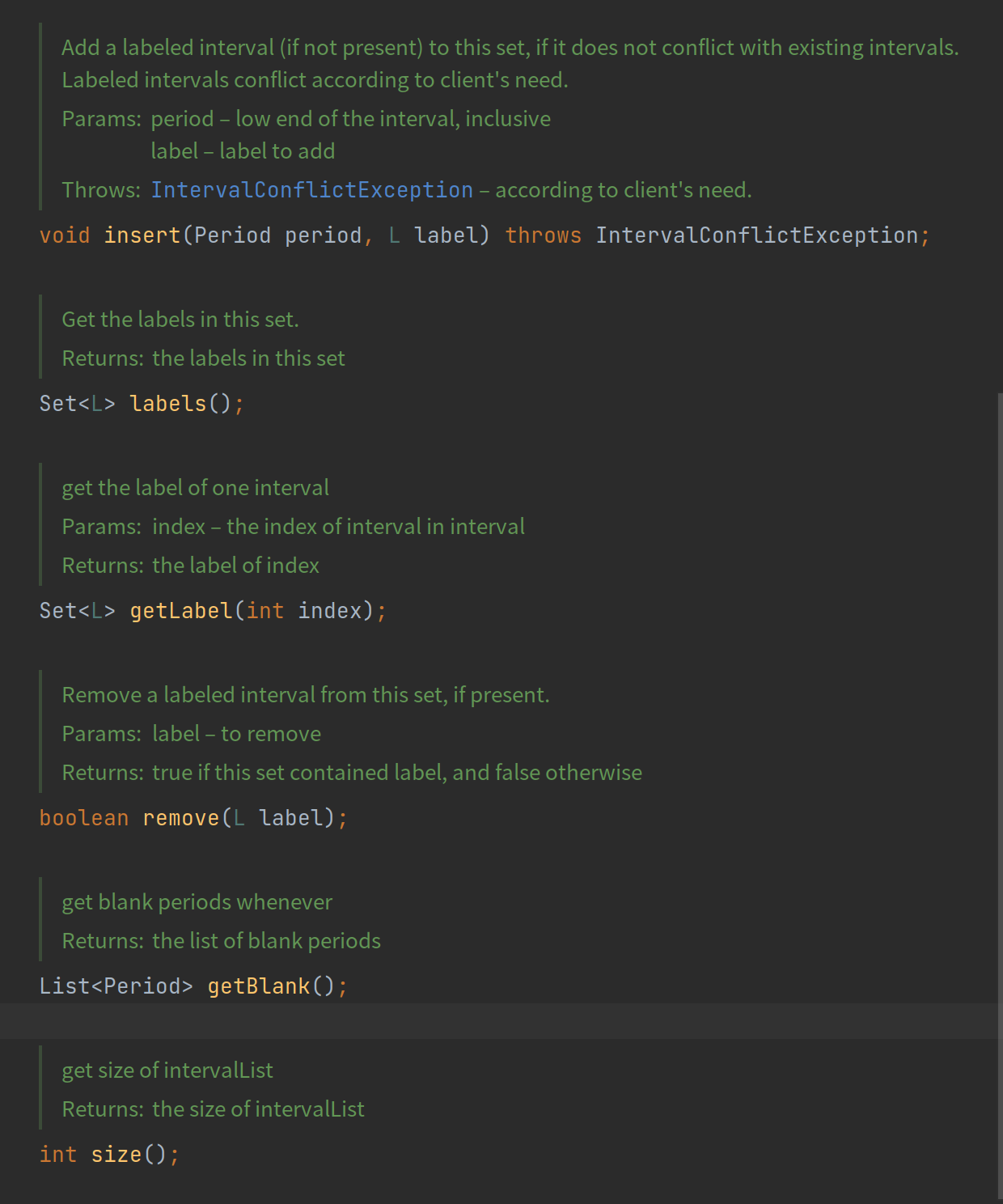
至于周期性，考虑到只有一个应用用它（课程安排），因此并未单独创建一个周期性的装饰。

至于如何来表示时间段，用什么样的数据结构，这是一很重要的问题。我选择直接采用List的形式表达，也就是一个时间单元的label是一个元素。应用一，有55天，也就是一个大小为55的employee的List。应用三，每天只能排5节课，一周7天，List大小为35。

这样做有一个问题，如果时间跨度较大，并且有很多连续的情况，只储存开始时间，结束时间，一个label就比较好。而不是储存(end-start)个label的指针，在第二个应用进程管理中就要面对这个问题。但是仔细阅读进程管理要求可以发现，这个并不需要存储数据，只需要模拟，并且采用随机化/确定的策略来执行，因此可以不考虑应用二的使用。

### IntervalSet接口设计

一共6个方法，其中去除了很多给定的不需要的方法，为后面应用实现方便加上了很多需要的方法。如下：



insert：给定一个Period（包含start,end）和label，将start，end]里所有的元素设置为label。数组初始化为null。

labels:获得List中现存的所有的labels的一个集合，如果某个时间段未分配，会含有null。

getLebal:给定一个下标，返回这个下标对应的label。考虑到后续会有重叠的情况，这里的返回类型为Set<L>，如果这个时间点未分配，返回集合包含null。

remove：将所有出现过这个label的地方设置为null

getBlank:返回所有为null的时间段的集合，每一个时间段包含起点终点

size：返回这个List的大小。

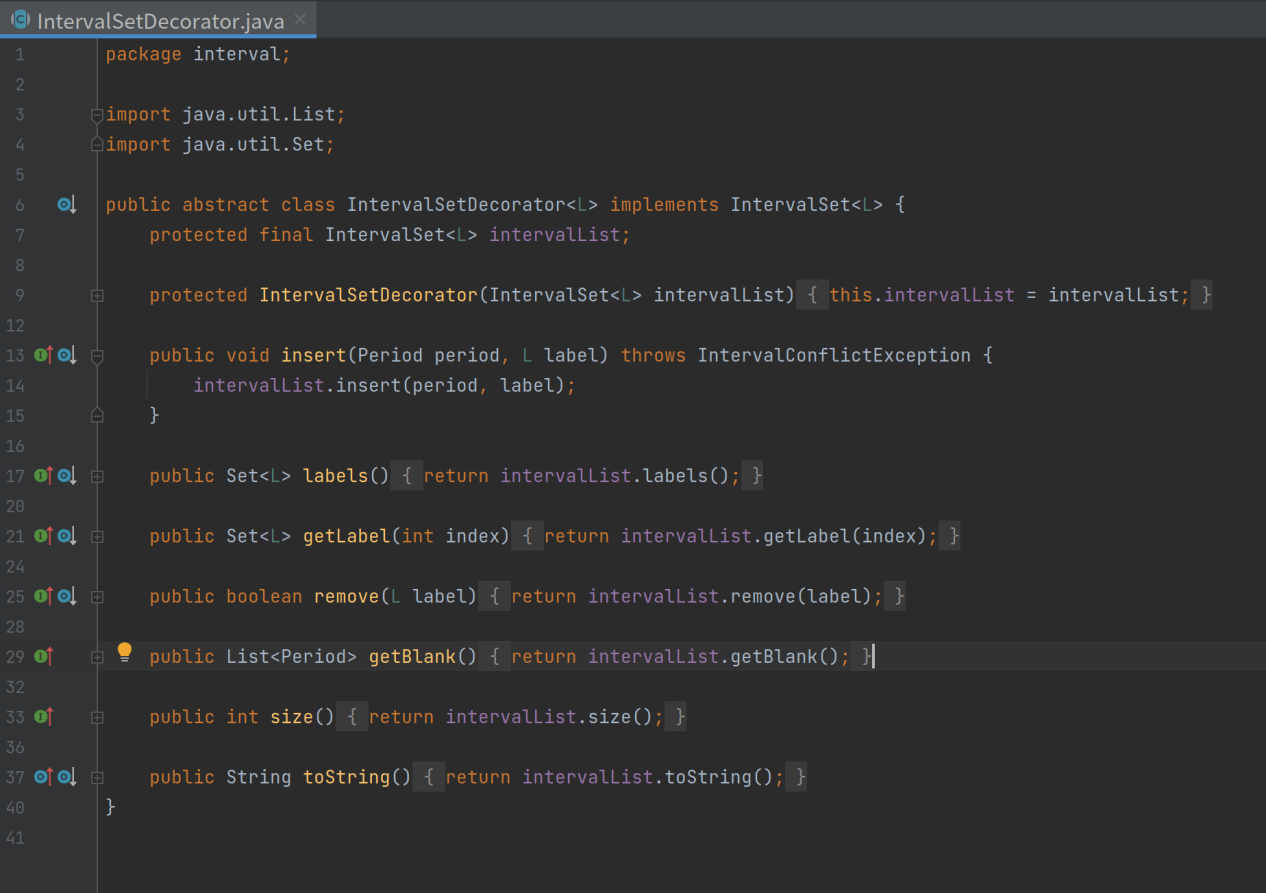
### CommenIntervalSet的实现

1. 构造函数，给定总的时间段长度，其中元素初始化为null。
2. insert，首先检测是否有冲突，没有冲突的话便插入。有冲突的话，则返回一个异常，由于CommonIntervalSet的特点，这时候的冲突只可能是发生重叠，因此，返回一个OverlapException的异常。
3. labels，直接返回即可
4. getLabel，直接返回即可
5. remove，使用遍历的方法，remove实质是将出现label的地方替换为null。
6. getBlank，while循环的方法，找到所有的空闲段，空闲段就是连续的null段。注意：本项目中所有的段均为闭区间。
7. size直接返回即可
8. toString返回每一个时间单元的label，null也打印为null。

### IntervalSetDecorator类

采用标准的装饰类的写法，包含一个IntevalSet类型的参数，直接采用委托的方法调用其函数，实现各个方法。

这个类是一个抽象类，所有具体的装饰类作为它的子类继承之。



### NoBlankIntervalSet

由于设计上，应用输出排班表，打印课表，全部是基于toString的，因此这个装饰只要判断一下，列表中是否有null元素存在，有的话toString返回排班未完成的提示信息，没有的话表示排班完成，允许输出。

### UniqueIntervalSet

只需要修改一下insert函数，当label已经分配到某个时间节点上时便报出异常。

很自然想到这样一个问题，如果[0,100]上分配了Label A，那么在[20,30]上分配B会不会将A分割为两段，导致A处于不连续的时间段中？答案是不会，如果允许重叠，那么A便没有被分隔开，如果不允许重叠，那么B便不能分配到这一段中。

那能不能用remove, 或者insert null label的方法使之不连续呢？首先remove并不能只移除一部分，insert设计的时候也并不能插入null来置空。

### OverlapIntervalSet

总体设计，加上一个additionalSet的字段，这是一个<Integer,Set<L>>的Map，键为索引，值为这个索引处重叠的label，当intervalSet中安排label之后，同一位置上多余的label就放在AdditionalSet中。

要修改的地方有insert，labels，getLabel，tostring，remove。后面四个字只需要考虑到AdditionalSet的存在，并对其执行相应的操作就好了。

insert方法，首先委托调用原有的insert，如果无overlapException便完成，有的话便将多余的重叠的放在AdditionalSet中，但如果某个时间没有，则还是优先放在intervalSet中。

## 可复用API设计

### 计算相似度

首先读取文件，储存于Map中，str1×str2->int，可以将<str1+str2,int>，和<str2+st1,int>放入Map，这样就完成了对应，后续只要遍历，然后求比例即可。

### 计算时间冲突比例

同样采用遍历的方法

### 计算空闲时间比例

同样遍历，空闲即为null。

## 应用设计与开发

### 排班管理系统

ADT:new UniqueIntervalSet (new NoBlankIntervalSet<>(new CommonIntervalSet <> (length)))

初期数据的读入只能使用文件读入的方法，文件中数据的格式与后文中给出的test测试文件格式相同。

共6个功能，手动排班，自动排班，查看排班情况，删除某个人员，打印排班表，退出。

1. 手动排班

输入排班人员编号，排班起止时间，然后调用insert方法，这里需要用到下标和日期相互转换的方法。

1. 自动排班

首先调用getBlank得到所有的空闲区，然后，将没有分配过的员工依次或者是随机的分配到空闲块中。

1. 查看情况

同样调用getBlank得到所有没分配过的块，然后打印，并且计算比例

1. 删除某个人员

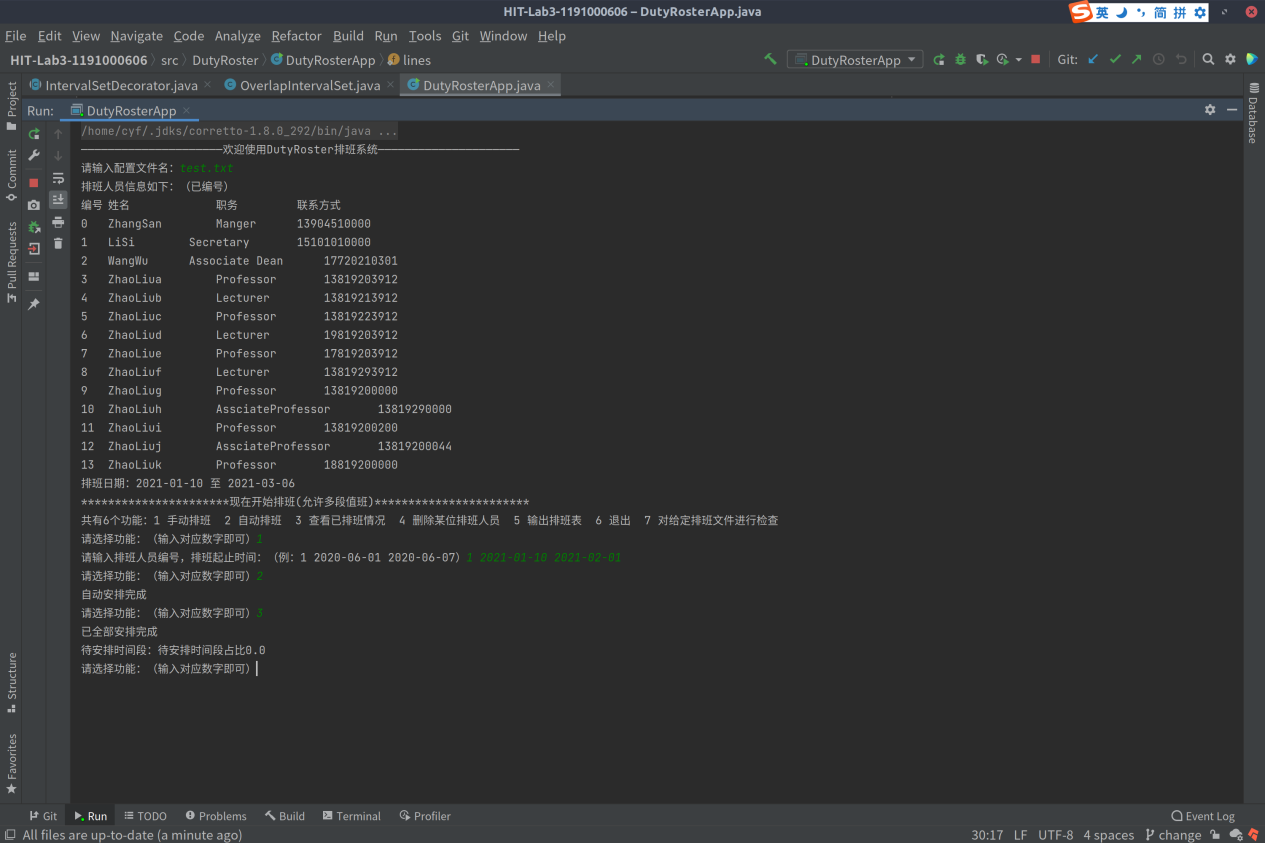
遍历，将所有出现过的时间段全部置为null。

1. 打印排班表

调用toString方法，如果返回的是未安排完成，则要求用户继续编排，完成的话，则分割返回字符串，进行一定的处理即可。

1. 退出

调用exit函数

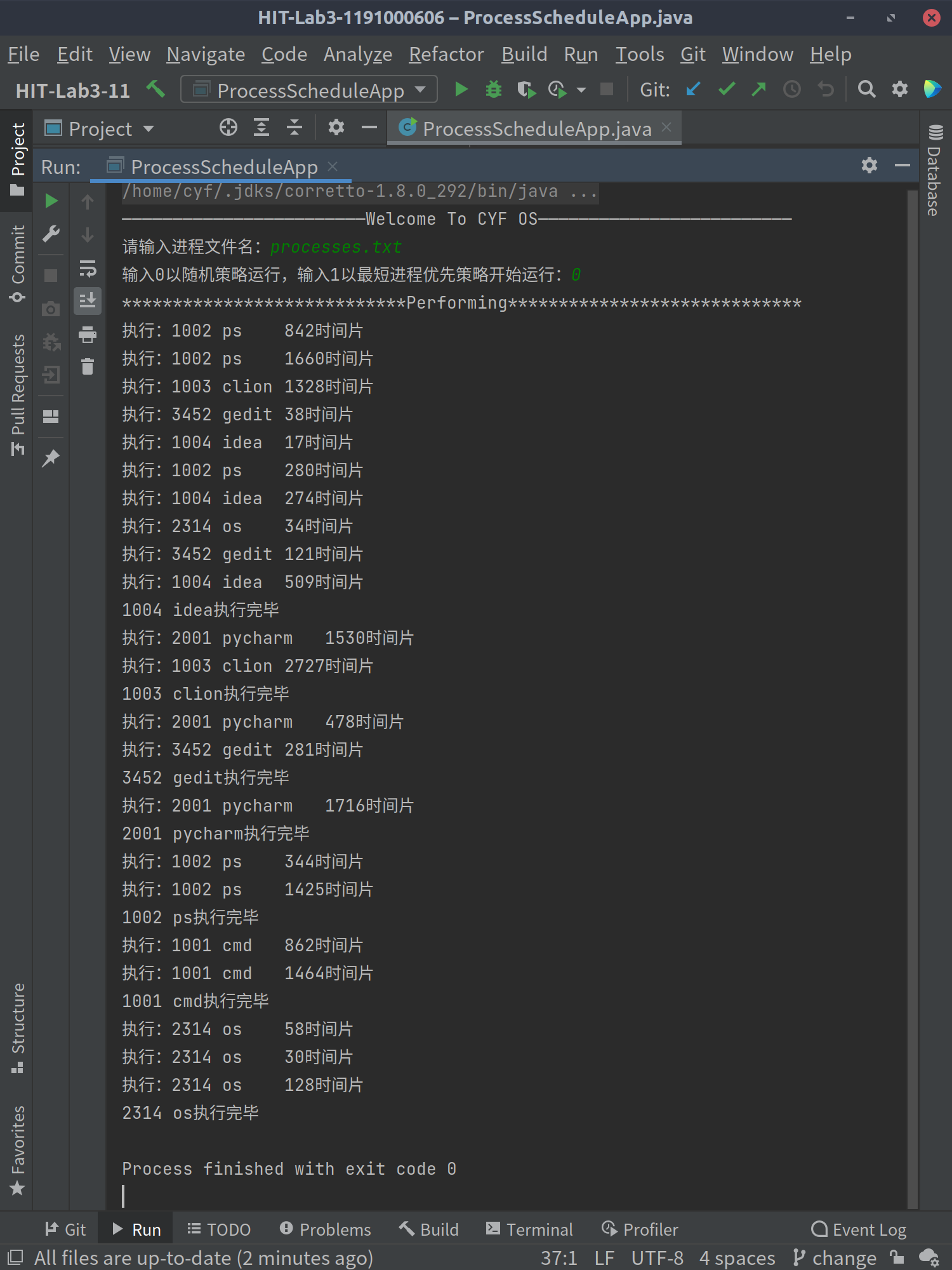


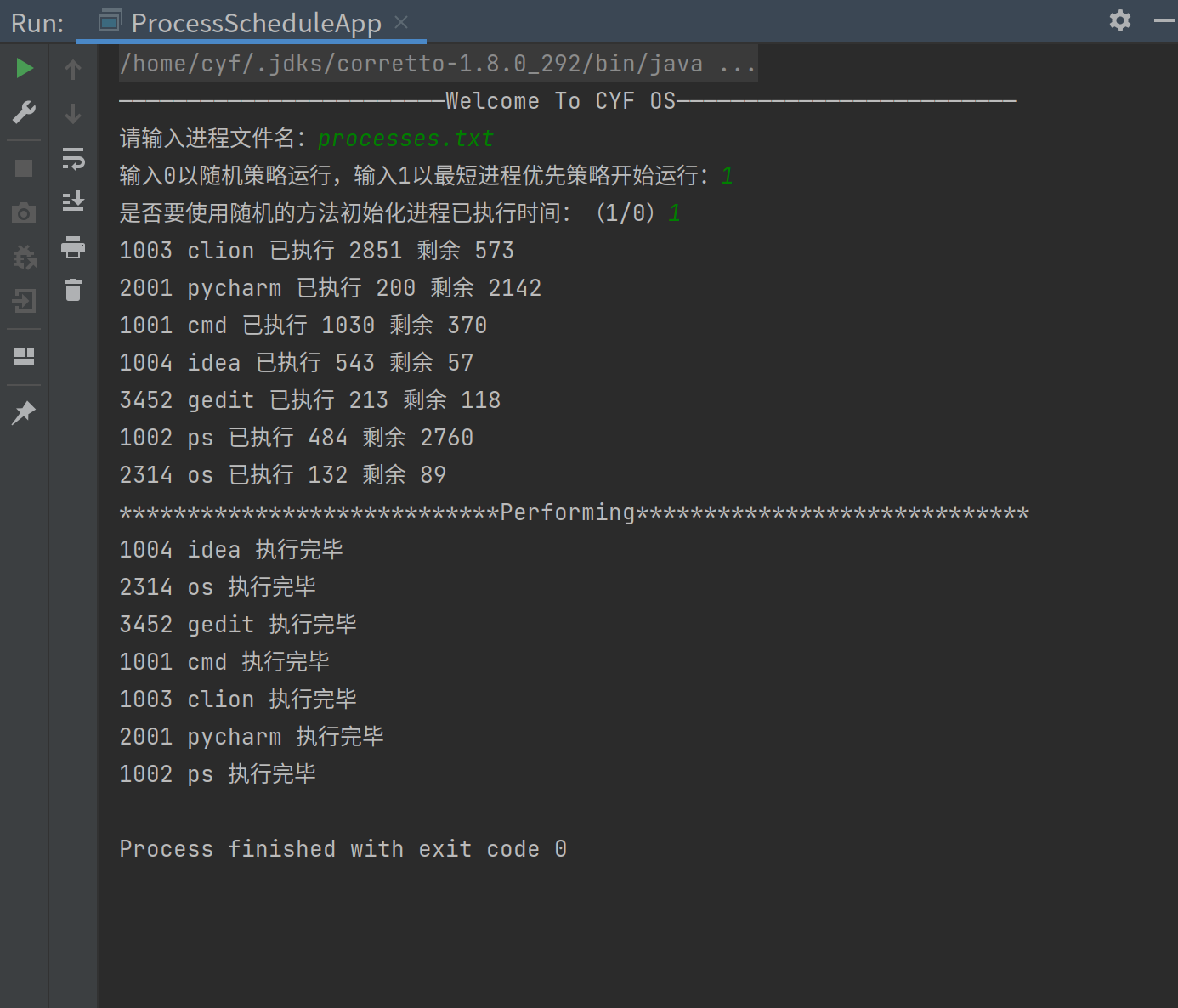
### 操作系统的进程调度管理系统

同样使用文件读入的方法。仔细阅读实验指导书，这一应用的设计并不需要存储数据，只需要模拟操作系统即可，因此没有使用ADT。此时有两个模式，随机，最短进程优先。

随机策略，随机选择下标，执行时间为最大执行时间乘以一个随机的比例，累计到超过最小执行时间则结束运行。运行和等待时间均采用sleep的方法进行。

最短进程优先的原则，首先最短执行时间乘以一个随机比例作为已执行时间，然后通过离最长执行时间的距离排序，依次执行，执行时间同样使用sleep模仿。





### 课表管理系统

和排班管理系统类似。

使用的ADT为new OverlapIntervalSet<>(new CommonIntervalSet<>(35))。同时用一个Map<Course,Integer>记录课程，和课程已安排了多少学时。

周一到周日分别用1—7代替，每天最有5个排课时间，分别为1-2节，3-4节，5-6节，7-8节，9-10节。每次排课只能为一门课程安排2个学时。

功能为：手动排课，查看排课情况，打印课表，查询课表，退出

手动排课时，需要注意：相同的时间，可以重叠开课，但是地点，老师不能重复，否则报错。排课时间不能超过上限，否则报错。

查看排课情况，即为打印已安排的课程，使用遍历的方法，未安排完的课程全部打印，同时记录有多少空闲时间段，占比多少。

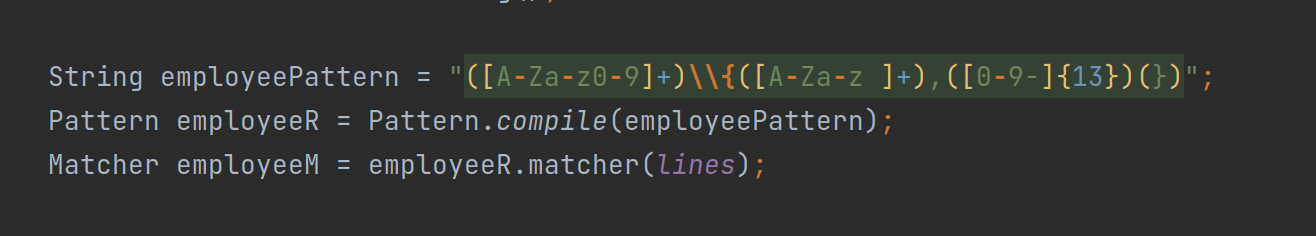
查询课表，即将给定的日期转化为星期几，注意不在本学期外的课程要提醒。

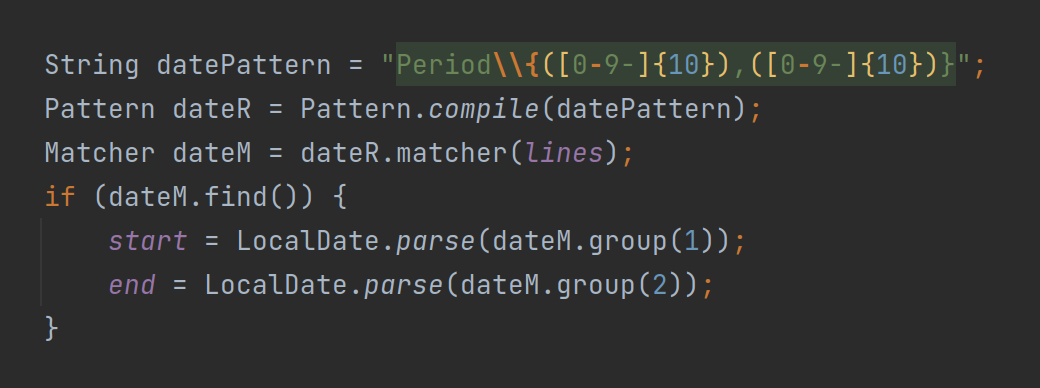


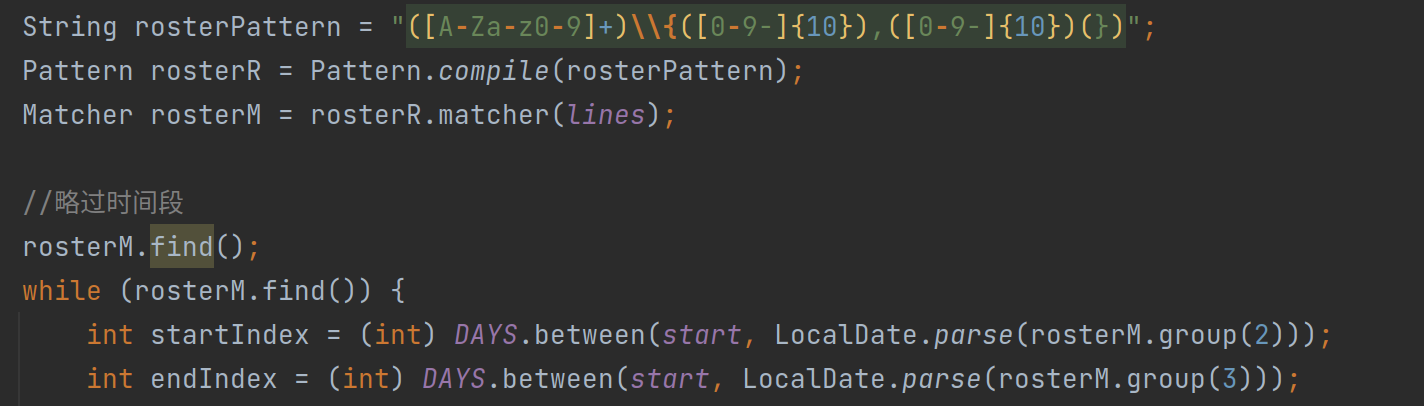
## 基于语法的数据读入

由于之前主要后后续要求，因此之前读入数据便采用的是正则语言读入。首先Employee，period，Roster中三者包含的字符串，构成并不相同，因此可以构造三个正则表达式，分别捕获，匹配，然后得到相应的数据后，模拟排班，没有异常，没有空白则排班完成。

三个正则表达式如图。







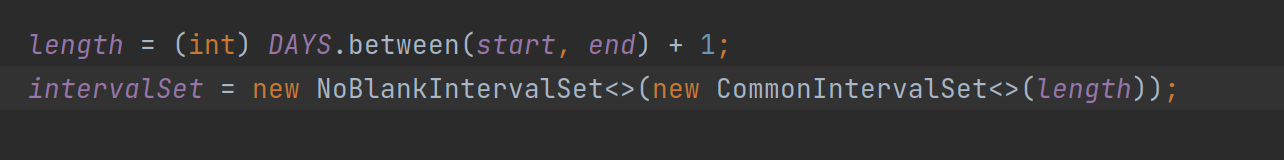
8个给定的测试文件中：

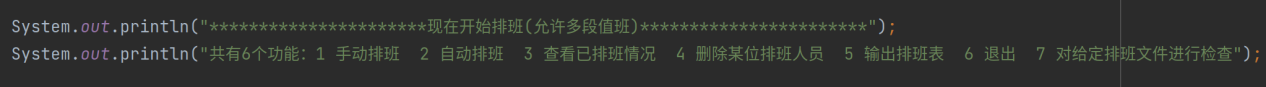
1. 正确
2. zhaoliua,b与1，2混用
3. 出现不合法电话178-192-03912，13-81920-0000
4. 正确，employee和period顺序互换
5. 1月19，20号未排
6. 不合法日期2021-2-25
7. employee中缺少员工zhaoliui
8. employee中缺少员工zhaoliue

## 应对面临的新变化

### 变化1

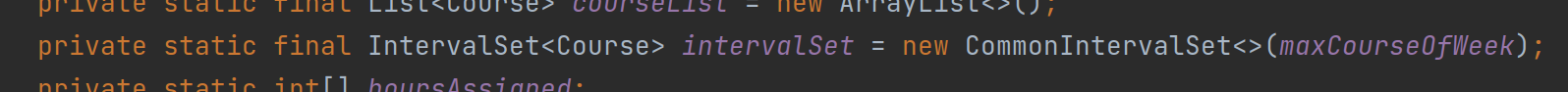
由于事先阅读过变化，因此变化极小，仅改便两行代码，一行为ADT，去掉了uniqueIntervalSet的修饰，一行为输出提示信息。

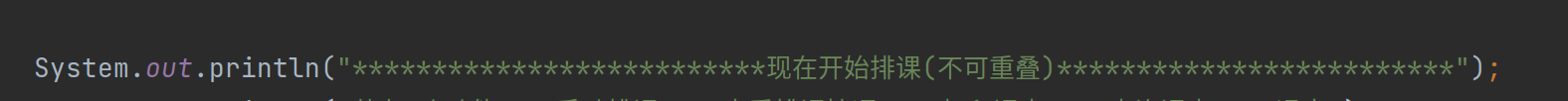




### 变化2

同样变化较小，修改了ADT的装饰，去掉了overlapIntervalSet，修改了一行注释代码，对inset中一段代码做了改动，去掉了处理重叠的情况。







## 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 |
| 6月30号 | 全天 | 设计实现ADT | 完成 |
| 7月1号 | 全天 | 修改ADT，开始APP1 | 完成 |
| 7月2号 | 全天 | 三个APP | 完成 |
| 7月3号 | 全天 | test，AF，RI，safe from exposure，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 |
| 正则语言不会 | 上网查 |

# 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

学会使用正则表达式，对设计模式尤其是装饰模式有了很好的理解

## 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在五个不同的应用场景下使用，你是否体会到复用的好处？

好像只有三个场景，不过复用确实大大减少了代码量，一天写完ADT（主要是在设计思考怎么能更好的复用，实现倒是简单），半天写APP1，半天根据APP1中出现的情况调整ADT，最后一天实现剩下两个应用。最后一天补上测试，AF，RI，API等等。

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？

这次实验中没有实际意义，ADT只有我一个人使用，并不会有其他人使用，所以书写规约并不是面向其他用户，而仅仅作为自己的提示。表示不变性也并没有实际意义，因为所有的应用都是我一个人写的，防止表示泄露也仅仅是让我自己看着更舒服一点，不会有其他人调用。以后的编程工作中我会按照实际情况来选择是否书写规约，AF，RI，不能过于迂腐，犯了教条主义错误，要理论实践相结合来看。

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？

这次实验中我开发的API是与实验高度结合的，是为了实现复用，面向三个应用以及后续需求设计的，并不是为了提供给他人使用。

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？

非常高效，有用，摆脱了以前使用各种字符串处理函数的繁琐。

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？

ADT最主要的问题是设计出来容易脱离实际应用，导致低效重复。所以设计ADT一定要考虑实际需求，先充分阅读各个应用的需求，然后设计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？

重复的功能要抽象出来，结构设计上要做到清晰美观。

1. 关于本实验的工作量、难度、deadline。

工作量还好，难度也不大。唯一的难点是正则表达式部分，课堂所讲知识完全没用。主要的问题是安排在了考试周，并且我因为转专业的原因，需要有一周的时间复习之前的课程备考，所以时间非常紧凑，程序只保证了正确性，对健壮性考虑较少。

1. 到目前为止你对《软件构造》课程的评价。

课程安排不是很不合理，没有Java基础，课时短，主要任务集中于考试周。实验设计照搬国外，比较脱离实际，缺少团队合作的部分。