

**2021年春季学期  
计算学部《软件构造》课程**

**Lab 3实验报告**

|  |  |
| --- | --- |
| 姓名 | 肖泽强 |
| 学号 | 1191000814 |
| 班号 | 1903001 |
| 电子邮件 | [3543226403@qq.com](mailto:3543226403@qq.com) |
| 手机号码 | 16673479299 |

**目录**

[1 实验目标概述 1](#_Toc71115456)

[2 实验环境配置 1](#_Toc71115457)

[3 实验过程 1](#_Toc71115458)

[3.1 待开发的三个应用场景 1](#_Toc71115459)

[3.2 面向可复用性和可维护性的设计：IntervalSet<L> 1](#_Toc71115460)

[3.2.1 IntervalSet<L>的共性操作 1](#_Toc71115461)

[3.2.2 局部共性特征的设计方案 2](#_Toc71115462)

[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](#_Toc71115463)

[3.3 面向可复用性和可维护性的设计：MultiIntervalSet<L> 2](#_Toc71115464)

[3.3.1 MultiIntervalSet<L>的共性操作 2](#_Toc71115465)

[3.3.2 局部共性特征的设计方案 2](#_Toc71115466)

[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](#_Toc71115467)

[3.4 面向复用的设计：L 2](#_Toc71115468)

[3.5 可复用API设计 2](#_Toc71115469)

[3.5.1 计算相似度 2](#_Toc71115470)

[3.5.2 计算时间冲突比例 2](#_Toc71115471)

[3.5.3 计算空闲时间比例 2](#_Toc71115472)

[3.6 应用设计与开发 2](#_Toc71115473)

[3.6.1 排班管理系统 3](#_Toc71115474)

[3.6.2 操作系统的进程调度管理系统 3](#_Toc71115475)

[3.6.3 课表管理系统 3](#_Toc71115476)

[3.7 基于语法的数据读入 3](#_Toc71115477)

[3.8 应对面临的新变化 3](#_Toc71115478)

[3.8.1 变化1 3](#_Toc71115479)

[3.8.2 变化2 3](#_Toc71115480)

[3.9 Git仓库结构 3](#_Toc71115481)

[4 实验进度记录 3](#_Toc71115482)

[5 实验过程中遇到的困难与解决途径 4](#_Toc71115483)

[6 实验过程中收获的经验、教训、感想 4](#_Toc71115484)

[6.1 实验过程中收获的经验和教训 4](#_Toc71115485)

[6.2 针对以下方面的感受 4](#_Toc71115486)

# 实验目标概述

本次实验覆盖课程第 2、3 章的内容，目标是编写具有可复用性和可维护性 的软件，主要使用以下软件构造技术： l 子类型、泛型、多态、重写、重载 l 继承、代理、组合 l 语法驱动的编程、正则表达式 l 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

实验环境: JDK 1.8 + IDEA 2021

在这里给出你的GitHub Lab3仓库的URL地址（Lab3-学号）。

https://github.com/ComputerScienceHIT/lab3-reusability-and-maintainability-programming-2021-1191000814.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

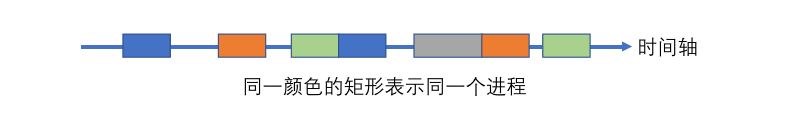
简要介绍三个应用。

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。

1. 值班表管理（DutyRoster）：一个单位有 n 个员工，在某个时间段内（例 如寒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。



1. 操作系统进程调度管理（ProcessSchedule）：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线程。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。



1. 大学课表管理（CourseSchedule）：看一下你自己的课表，每一上午 10:00-12:00 和每周三上午 8:00-10:00 在正心楼 42 教室上“软件构造”课程。课程需要特定的教室和特定的教师。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（例如周一上午 3-4 节的计算方法和软件 构造）；一位教师也可以承担课表中的多门课程。



## 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

(1) 创建一个IntervalSet的实现类,这里创建的是CommonIntervalSet

public static <L> IntervalSet<L> empty(){  
 return new CommonIntervalSet<>();  
}

(2) 插入一个时间段  
public void insert(Interval<L> interval) throws

IntervalConflictException, LabelRepeatException;

(3) 获取全部的标签集合  
public Set<L> labels();

(4) 移除一个标签及其时间段  
public boolean remove(L label);  
  
(5) 获取该标签的的时间段列表  
public List<Interval<L>> getIntervals(L label) throws

NoSuchElementException;

### 局部共性特征的设计方案

采用了方案6设计方法：使用 decorator 设计模式 将 CommonIntervalSet 看作是原始的、未被装饰的对象，将这三个维度看作是三种“装饰”（每个维度的不同特征取值可以产生不同的“装饰”效果）。 请参照讲义上关于该设计模式的说明，设计相应的子类型继承关系树，然后在具体应用中通过为一个对象逐层装饰三个不同特征，即可实现应用所需的组合特征。

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

(1) CommonIntervalSet: 直接实现IntervalSet接口,限制性最少的类,可以有重叠时间段,重复标签,以及可以有空白

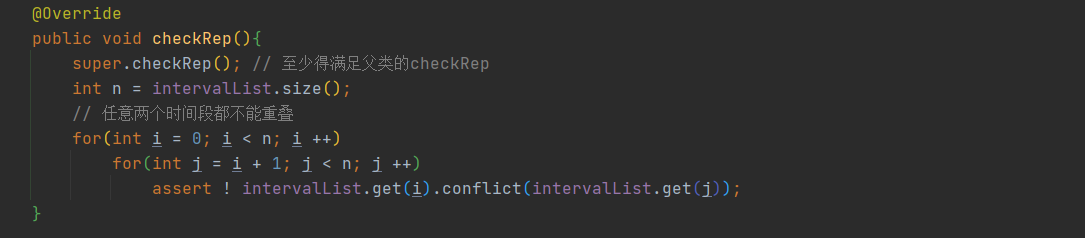
(2) MultiIntervalSet: 继承CommonIntervalSet,不能有重叠时间段,可以有重复标签,不能有空白时间段

(3) LinearIntervalSet: 继承MultiIntervalSet,不能有重叠时间段,不能有重复标签,不能有空白时间段

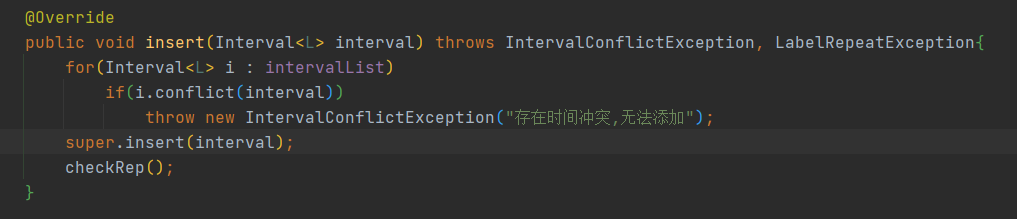
## 面向可复用性和可维护性的设计：MultiIntervalSet<L>

### MultiIntervalSet<L>的共性操作

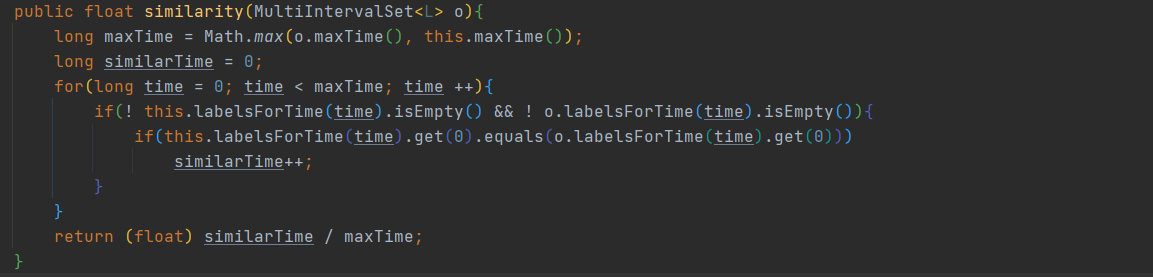
MultiIntervalSet继承了CommonIntervalSet,并且继承了它的大部分方法,很大程序上复用了CommomIntervalSet和IntervalSet的代码,除了继承父类的方法外,还重写或添加了了以下函数:

(1) checkRep方法,MultiIntervalSet首先得满足其父类的性质,所以先使用父类的checkRep方法,再根据自己的特性(标签不能重复),设计方法  


(2) insert方法,先判断插入的时间段是否存在时间段冲突,如果有则抛出异常,如果没有使用父类的insert方法

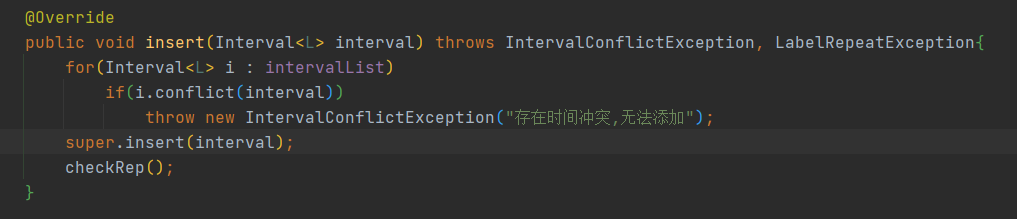


(3) 相对于父类添加了similarity方法,因为MultiIntervalSet不会有时间段冲突,所以可以计算相似度



### 局部共性特征的设计方案

MultiIntervalSet的特征是不能有重叠时间段,且不能有空白时间段,具体是在重写Insert函数中,先判断先判断插入的时间段是否存在时间段冲突,如果有则抛出异常,如果没有使用父类的insert方法

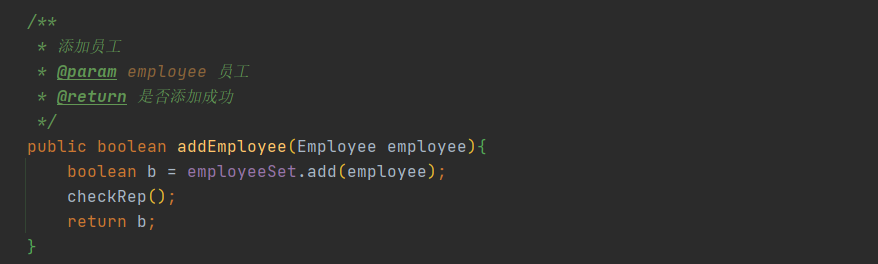


### 面向各应用的MultiIntervalSet子类型设计（个性化特征的设计方案）

**(1) 排班IntervalSet : DutyIntervalSet**

实现:委托了LinearIntervalSet类

添加员工:



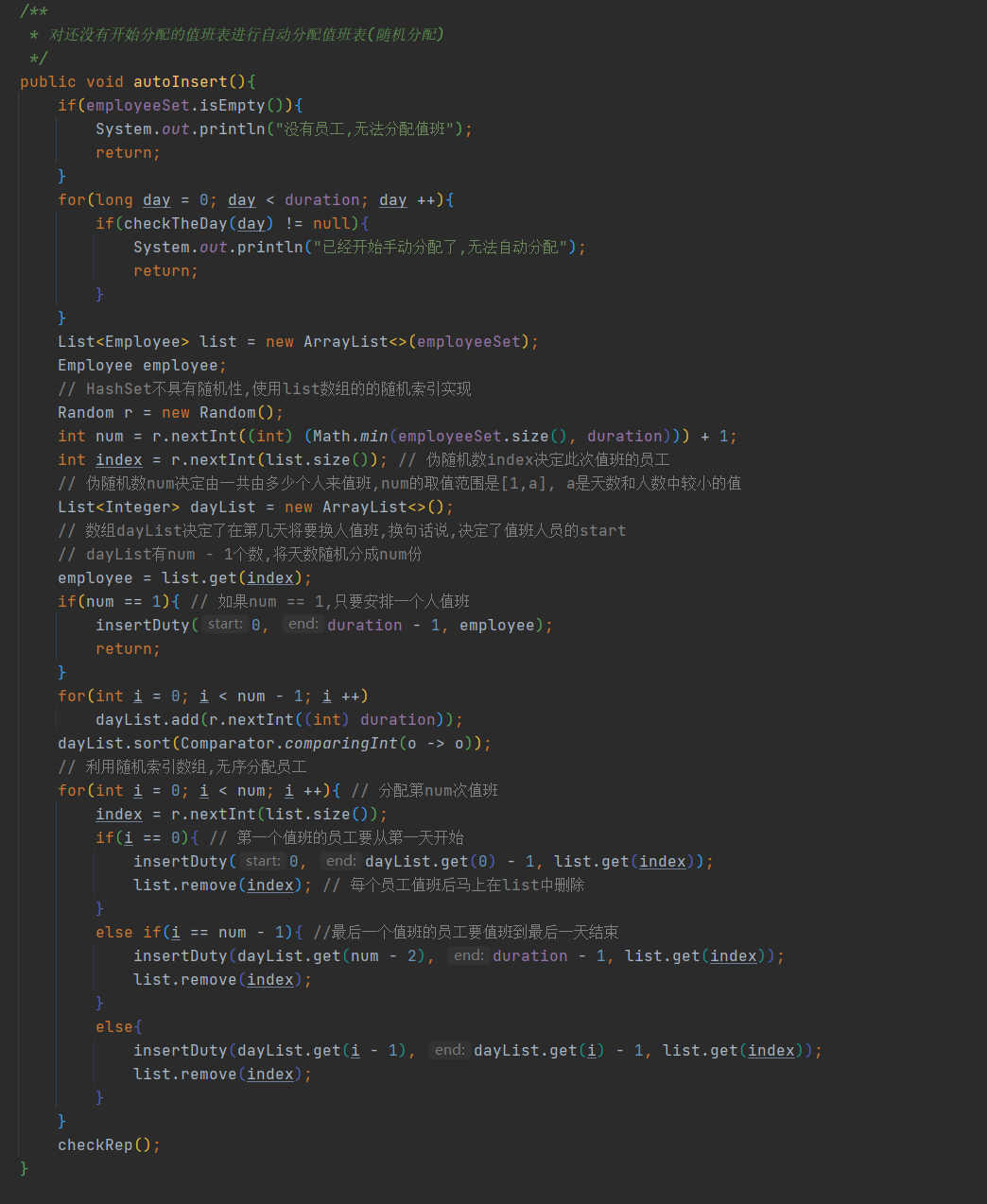
删除员工:



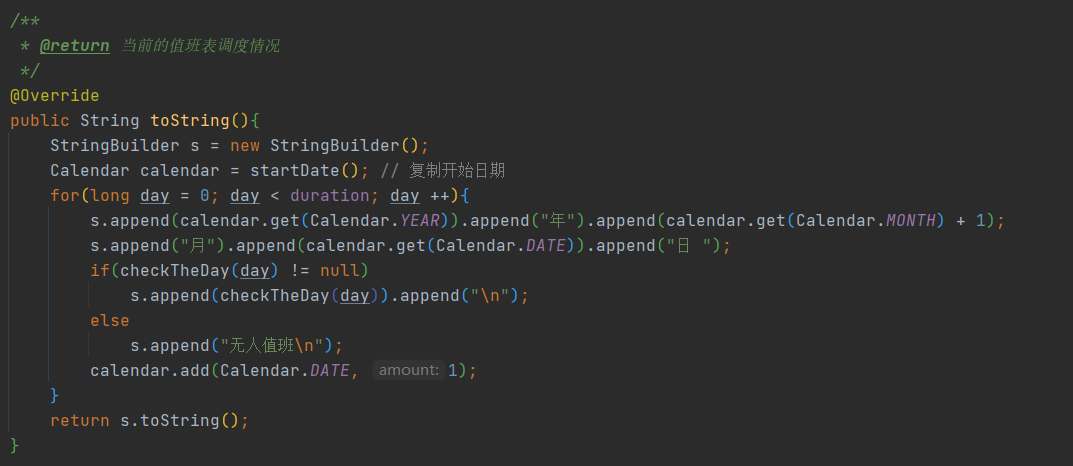
插入排班:



自动排班:



展示当前排班进度:



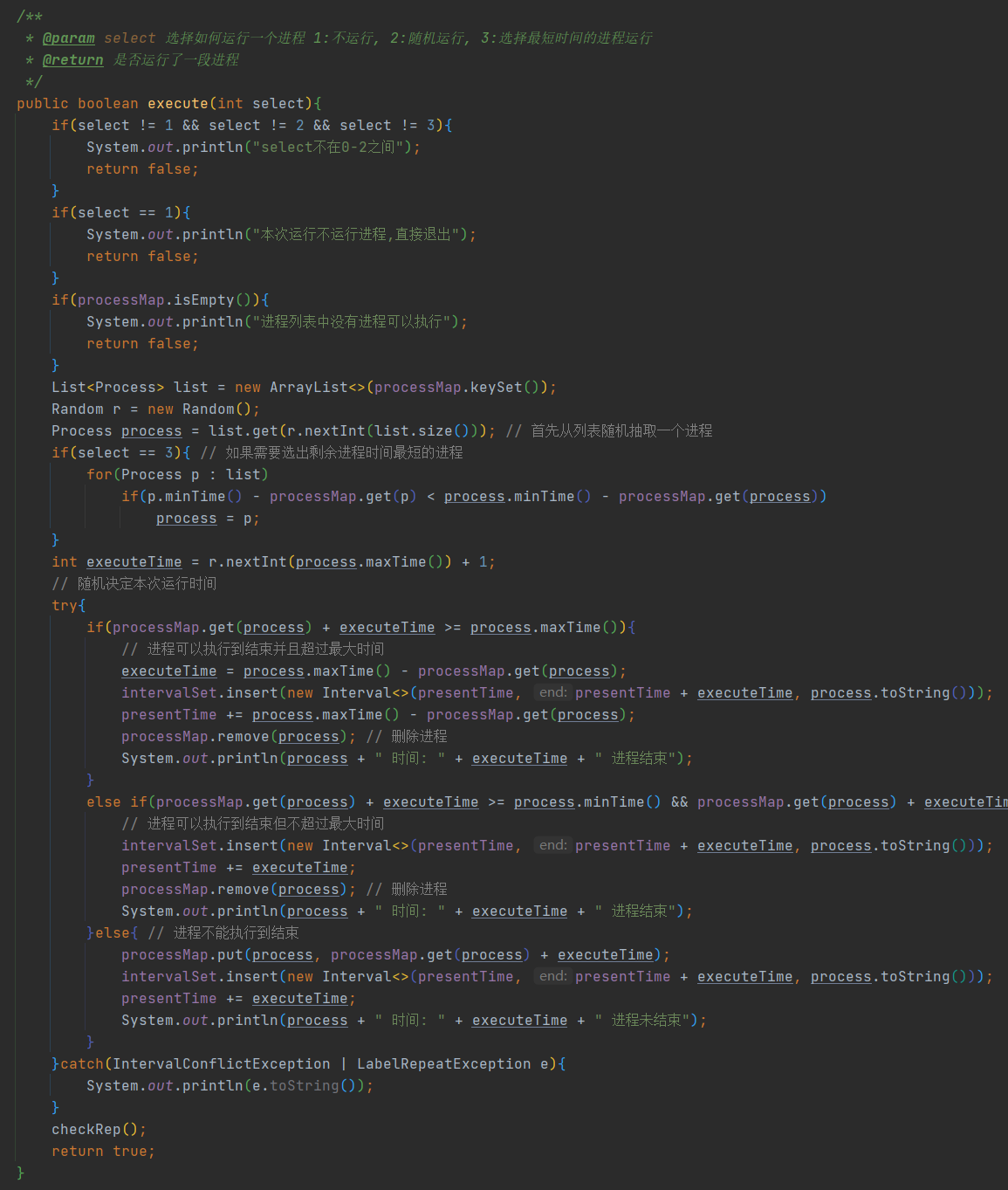
**(2) 进程管理IntervalSet :ProcessIntervalSet**

实现方法: 委托了MultiIntervalSet

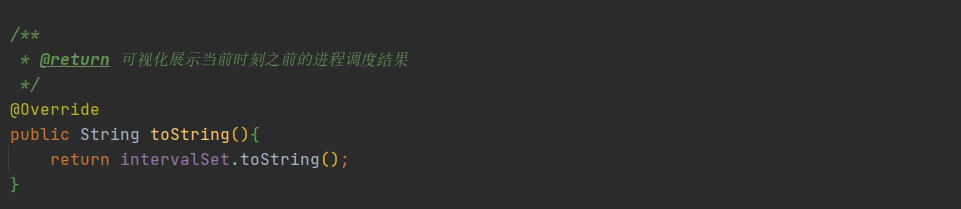
添加进程:



执行进程(两种方式:随机选择和最短优先)



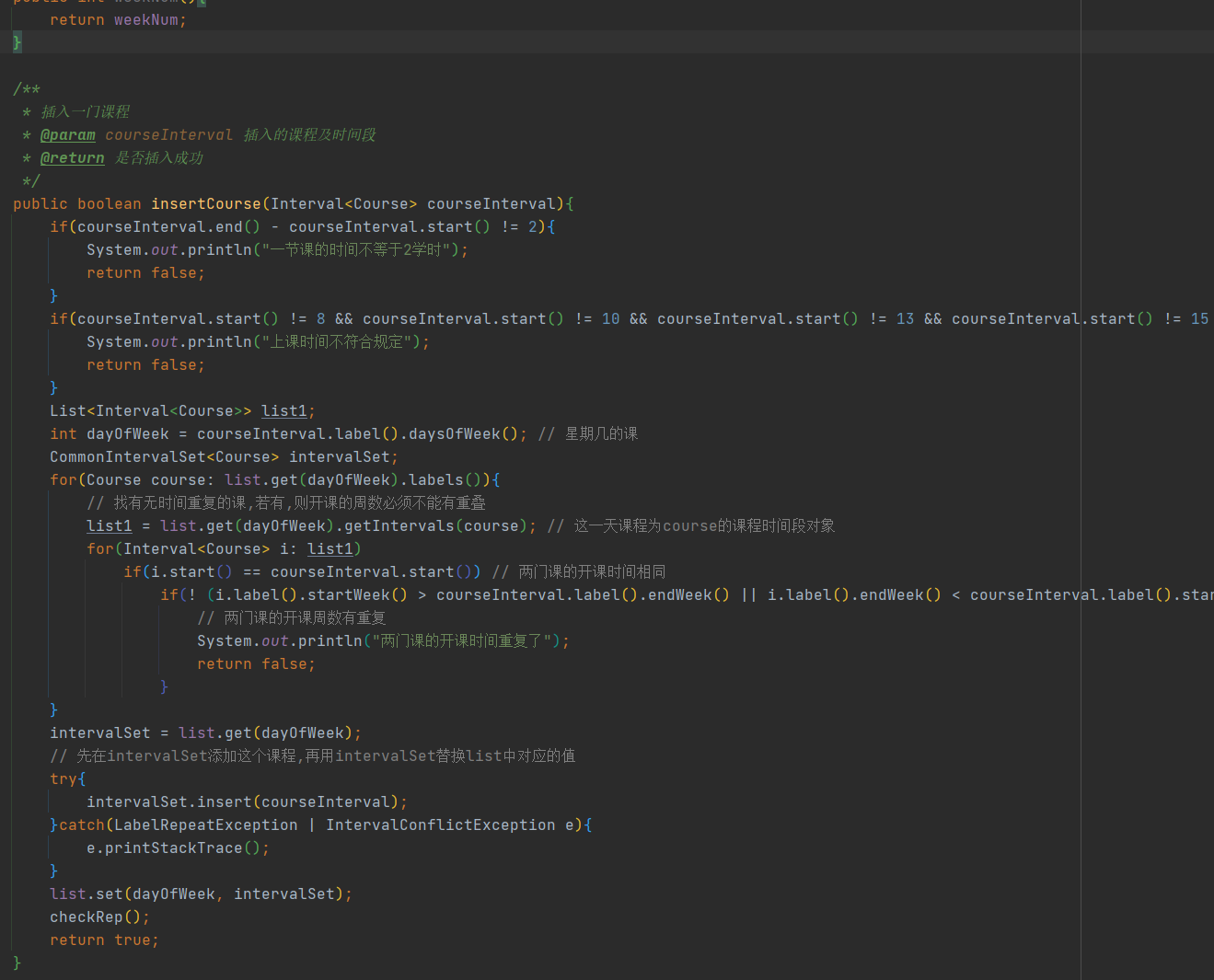
可视化当前进程:



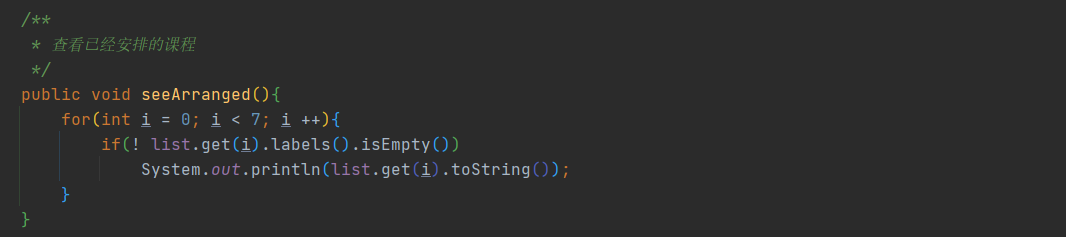
**(3) 排课管理 IntervalSet : CourseIntervalSet**

实现方法:委托了CommonIntervalSet

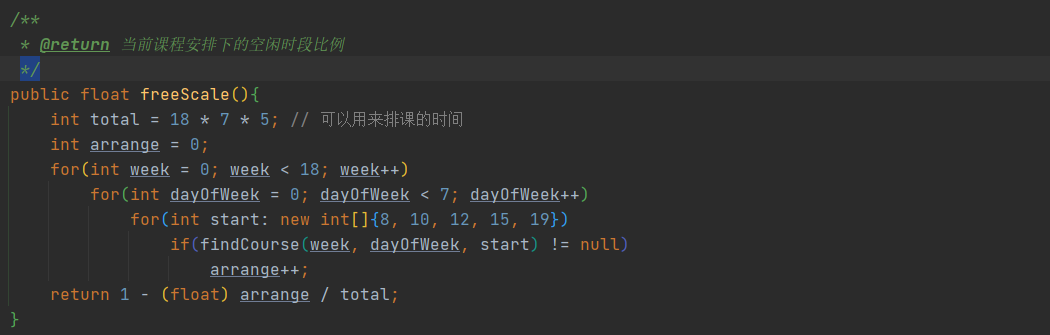
插入课程:



查看已经安排的课程:



当前排课下的时间空闲比例:



查看某一周安排的课程:



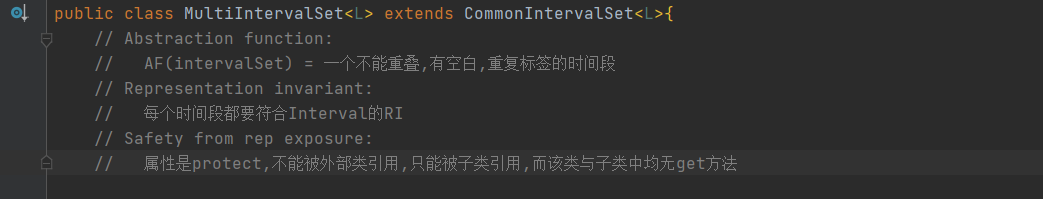
## 面向复用的设计：L

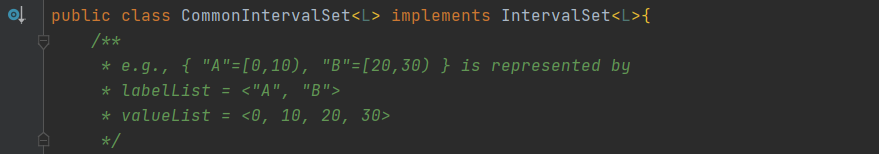
IntervalSet<L>和 MultiIntervalSet<L>中的泛型参数 L，可以是你所设计的任何 immutable 的类。 对三个应用来说，其 L 分别应为“员工”（Employee）、“进程”（Process）、“课程”（Course），所需关注的属性分别为：

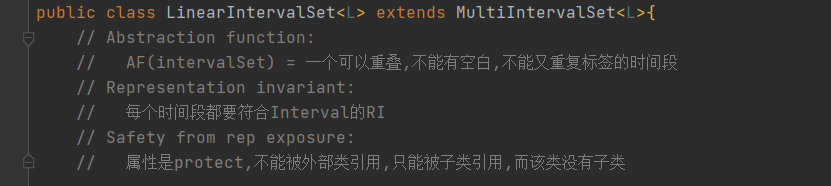
Employee：姓名、职务、手机号码

Process：进程 ID、进程名称、最短执行时间、最长执行时间

Course：课程 ID、课程名称、教师名字、地点分别实现这些 ADT。



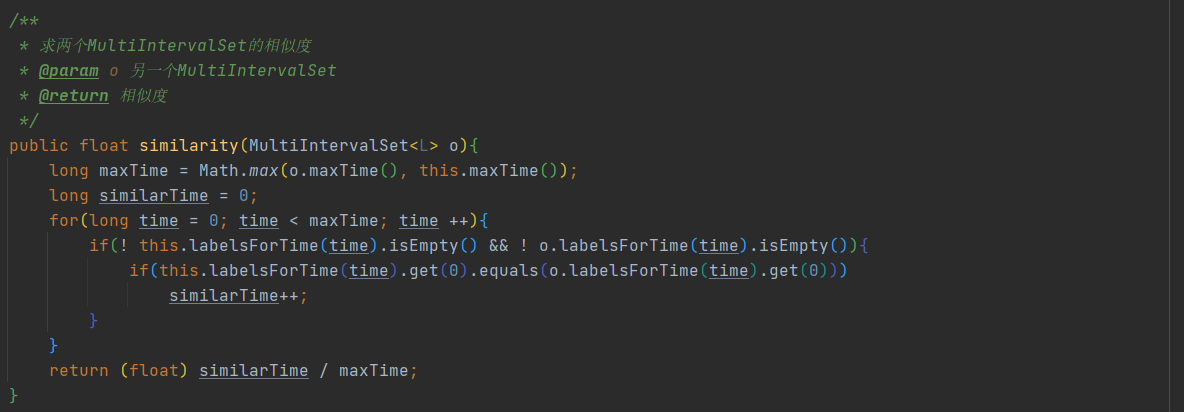




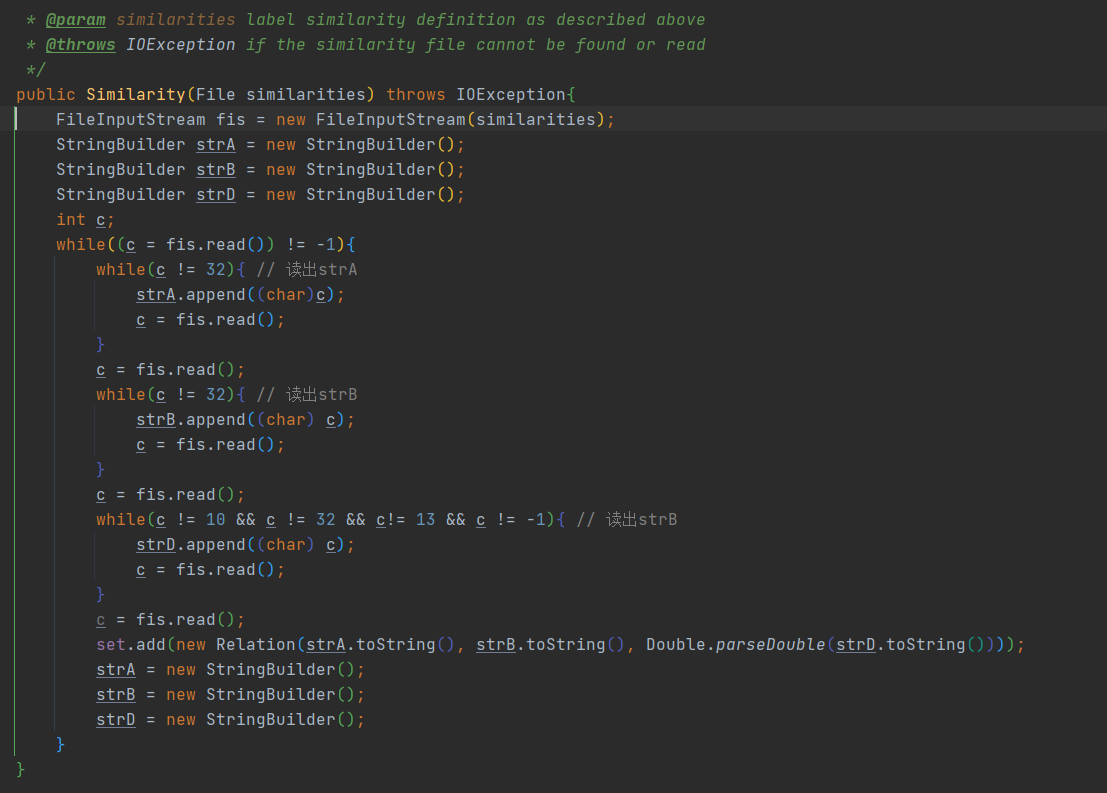
## 可复用API设计

### 计算相似度

double Similarity(MultiIntervalSet<L> s1, MultiInterval Set<L> s2) 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。

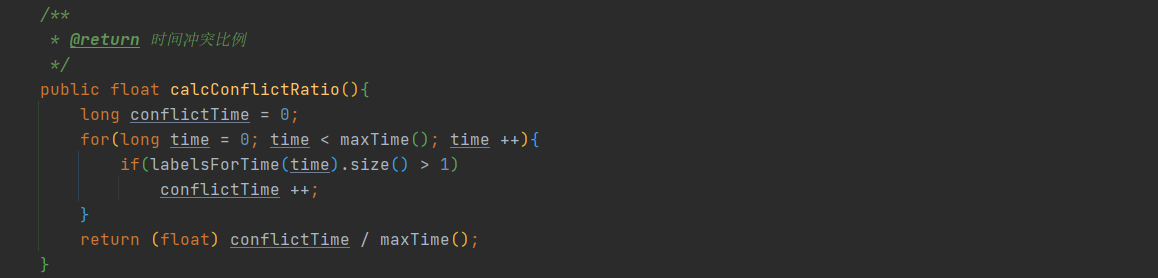


另一种相似度是根据不同标签之间已经确定的相似度比例,通过每个时间节点的两个标签的相似度比例乘积之和,来确定两个MultiIntervalSet的相似度



### 计算时间冲突比例

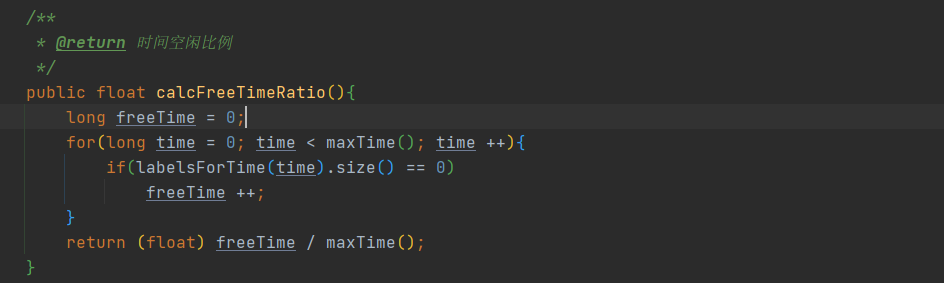
所谓的“冲突”，是指同一个时间段内安排了两个不同的 interval 对象。用 发生冲突的时间段总长度除于总长度，得到冲突比例，是一个[0,1]之间的值。



### 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，是一个[0,1]之间的值。



## 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。

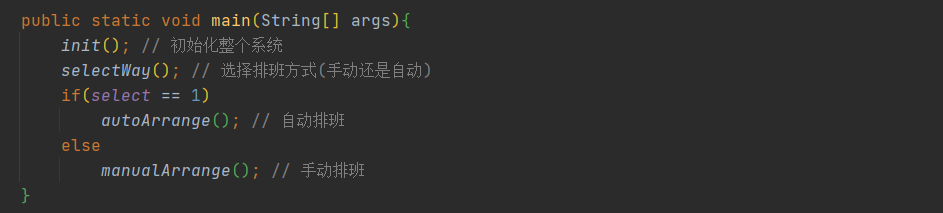
### 排班管理系统

针对排班管理系统，所需完成的功能为：

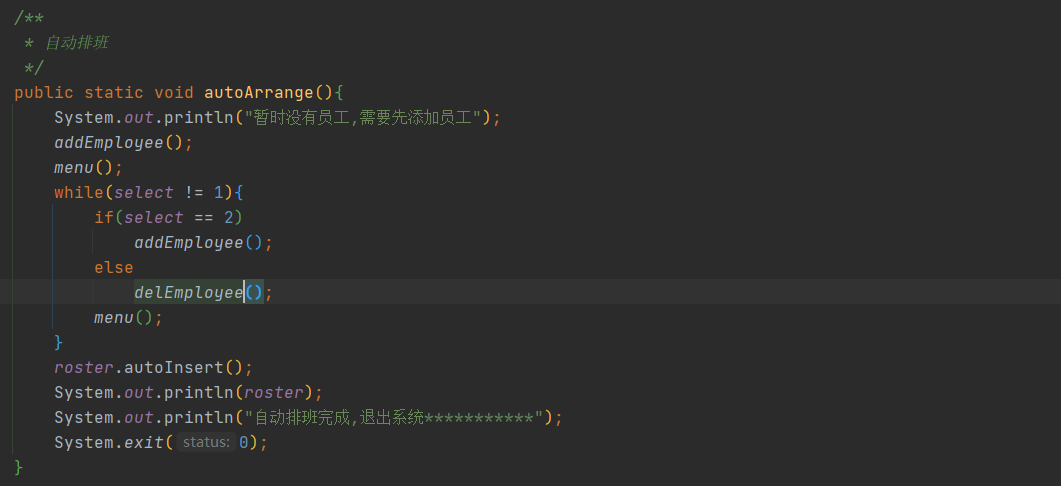
Step 1 设定排班开始日期、结束日期，具体到年月日即可。  
Step 2 增加一组员工，包括他们各自的名字、职务、手机号码，并可随时删除某些员工。如果某个员工已经被编排进排班表，那么他不能被删除，必须将其排班信息删掉之后才能删除该员工。员工信息一旦设定则无法修改。  
Step 3 可手工选择某个员工、某个时间段（以“日”为单位，最小1天，可以是多天），向排班表增加一条排班记录，该步骤可重复执行多次。在该过程中，用户可随时检查当前排班是否已满（即所有时间段都已被安排了特定员工值班）、若未满，则展示给用户哪些时间段未安排、未安排的时间段占总时间段的比例。  
Step 4 除了上一步骤中手工安排，也可采用自动编排的方法，随机生成排班表。  
Step 5 可视化展示任意时刻的排班表。可视化要直观明了，可自行设计。

委托了DutyIntervalSet,具体实现依赖于DutyIntervalSet的方法

main函数:先初始化,再选择排班方式(自动还是手动),最后按照方式排课



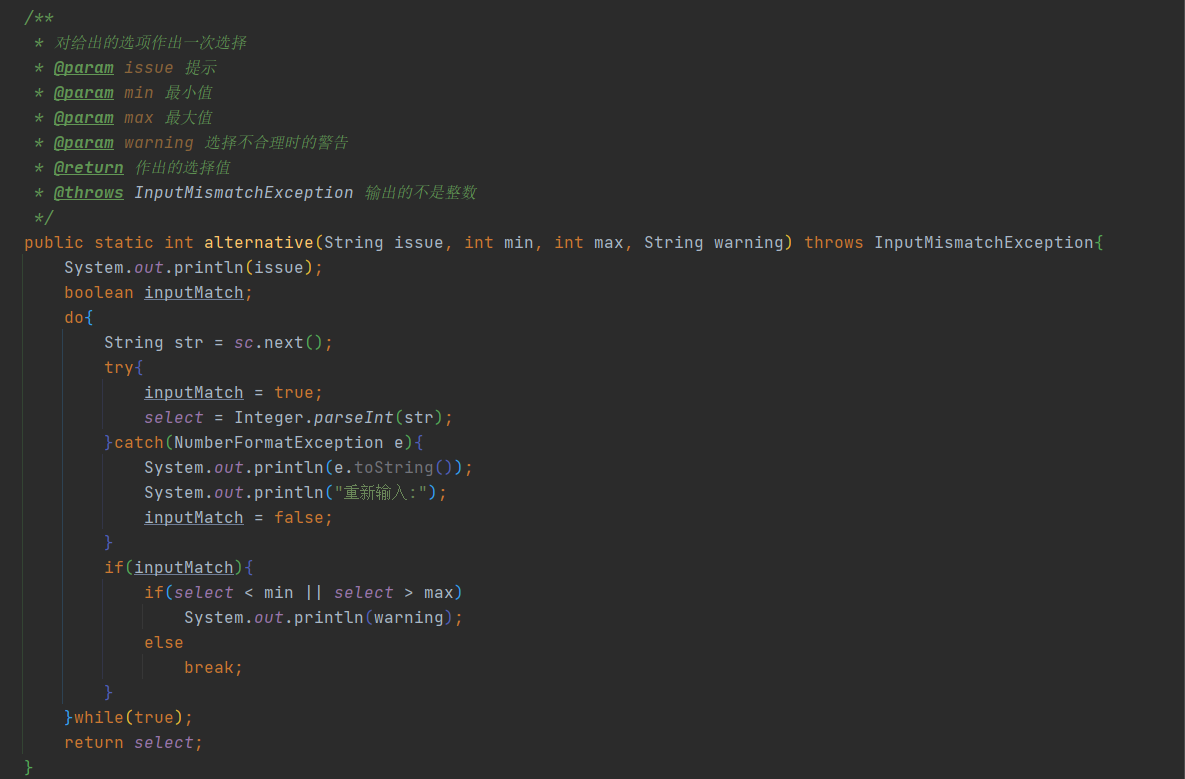
自动排班:



手动排班:



最重要的函数:alternative,极大的复用了代码,使整个app只有200行不到,并且在代码层面上也复用到了其他两个app的设计上。



### 操作系统的进程调度管理系统

针对操作系统进程管理系统，所需完成的功能为：

Step 1 增加一组进程，输入每个进程的ID、名称、最短执行时间、最长执行时间；进程一旦设定无法再修改其信息。  
Step 2 当前时刻（设定为0）启动模拟调度，随机选择某个尚未执行结束的进程在CPU上执行（执行过程中其他进程不能被执行），并在该进程最大时间之前的任意时刻停止执行，如果本次及其之前的累积执行时间已落到[最短执行时间，最长执行时间]的区间内，则该进程被设定为“执行结束”。重复上述过程，直到所有进程都达到“执行结束”状态。在每次选择时，也可“不执行任何进程”，并在后续随机选定的时间点再次进行进程选择。  
Step 3 上一步骤是“随机选择进程”的模拟策略，还可以实现“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。  
Step 4 可视化展示当前时刻之前的进程调度结果，以及当前时刻正在执行的进程。可视化的形式要直观明了，可自行设计。

委托了ProcessIntervalSet,具体实现依赖于ProcessIntervalSet的方法

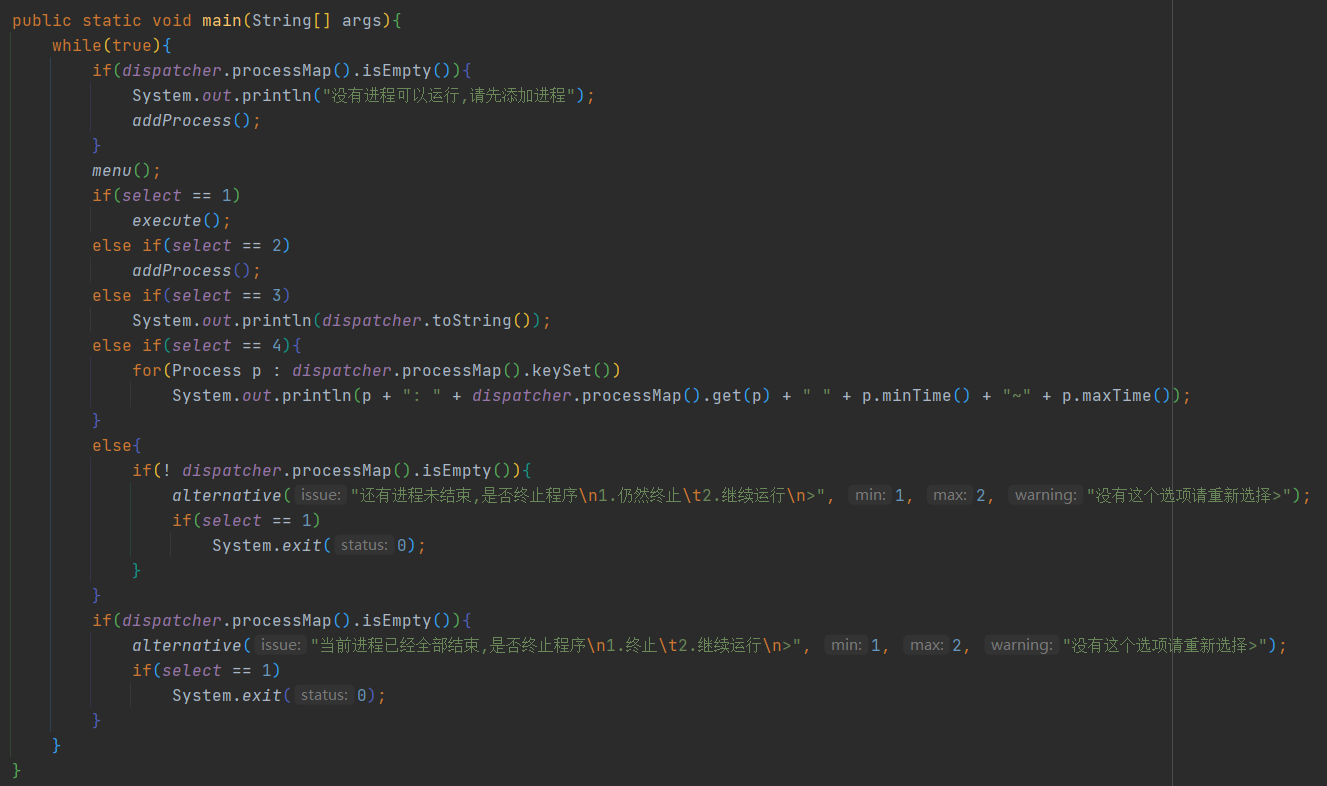
main函数:执行循环直到全部进程运行结束:

(1) 若无进程可以运行,则必须先添加程序

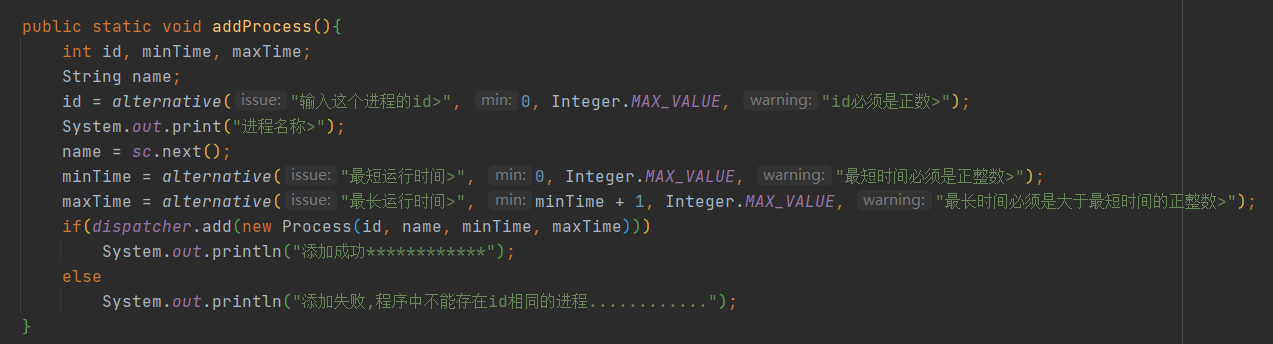
(2) 弹出菜单并进行选择, 有五个选项:1.运行进程2.添加进程3.查看已运行的进程4.查看未运行的进程5.终止程序

(3) 如果运行进程,则选择以何种方式运行(不运行,随机运行,运行最短时间的程序)

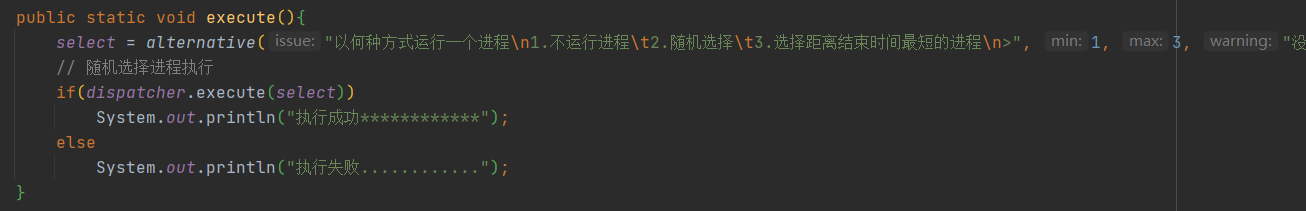
(4) 没次执行完选项中的一个,如已经没有进程可以运行,选择是否要退出程序



添加进程:



执行进程:



### 课表管理系统

针对课表管理系统，所需完成的功能为：

Step 1 设定学期开始日期（年月日）和总周数（例如 18）；

Step 2 增加一组课程，每门课程的信息包括：课程 ID、课程名称、教师名

字、地点、周学时数（偶数）；

Step 3 手工选择某个课程、上课时间（只能是 8-10 时、10-12 时、13-15 时、

15-17 时、19-21 时），为其安排一次课，每次课的时间长度为 2 小时；可重复安 排，直到达到周学时数目时该课程不能再安排；

Step 4 上步骤过程中，随时可查看哪些课程没安排、当前每周的空闲时间比 例、重复时间比例；

Step 5 因为课程是周期性的，用户可查看本学期内任意一天的课表结果。

委托了CourseIntervalSet,具体实现依赖于CourseIntervalSet的方法

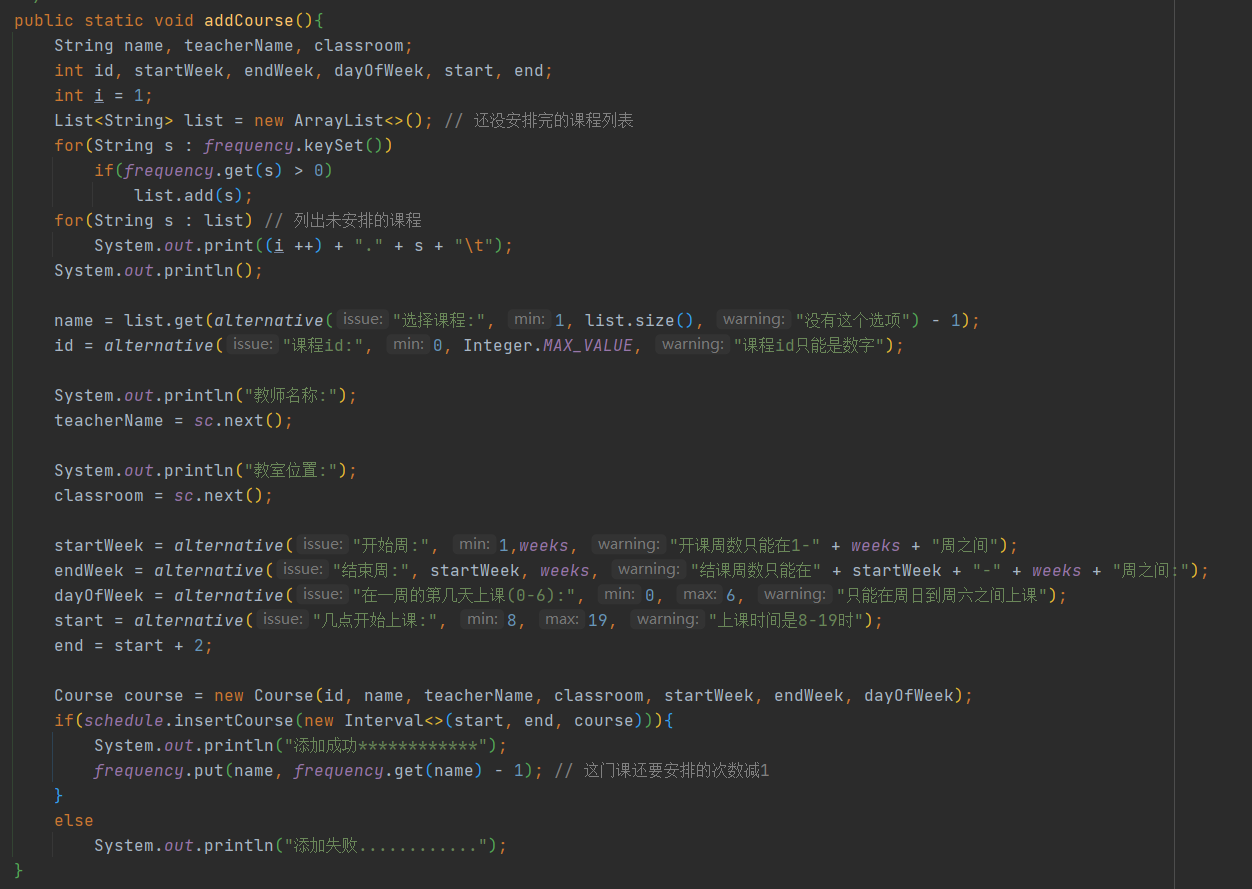
main函数:先初始化变量,再安排课程(课程和需要安排的次数是已经确定的)



执行循环直到课程被安排完成,每次都可以选择下列一项操作完成:

1.安排课程2.查看排课进度3.查询课表

安排课程:



查看排课进度:



查询某一周的课表:



## 基于语法的数据读入

针对 3.10 节中开发的“值班表管理”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，从中抽取信息，构造值班表对象。输入文件的语法示例如下所示，示例文件中不带有下划线的文字部分为固定说明部分，带有下划线的文字部分为可变化信息。“//”后续文字是解释说明，并非语法的构成部分。 文件里描述的 Employee、Period、Roster 的次序是不确定的，违反该次序不能被看作非法。在读取文件过程中，若发现有违反本例中解释说明部分给出的规则，则该文件不符合语法，需结束读取并提示用户选择其他合法的文件。另外，请忽略文件中的所有缩进，它并非语法的组成部分。

重写了ParserRoster类和PatserMain程序以实现正则表达式读取与解析文件内容,并将其写入值班表中。

用ParserRoster委托一个DutyIntervalSet类,先将所有内容读到一个字符串test中,然后用一个正则表达式stringMatch去匹配它,先匹配Period内容,找到Period所在的位置,用

Pattern.compile("Period\\{[0-9,-]\*}");

匹配出开始和结束日期,实现初始化。再用

Pattern.compile("([a-zA-Z])+\\{([a-z A-Z])+,[0-9-]+}");

匹配员工列表,最后用

Pattern.compile("[a-zA-Z]+\\{[0-9,-]+,[0-9,-]+}");

匹配值班安排。

对test 文件1-8分别进行测试得到:

test1 : 正确

test2 : 没有ZhaoLiua和ZhaoLiub这两个员工

test3 : ZhaoLiue的电话号码格式不对

test4 : 正确

test5 : 存在时间冲突

test6 : 日期格式不匹配

test7 : 没有ZhaoLiue和ZhaoLiui这两个员工

test8 : 没有ZhaoLiue和ZhaoLiuk这两个员工

## 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，例如张三的值班 日期为(2021-01-01, 2021-01-10), (2021-02-01, 2021-02-06)；

之前的应用不能应对该变化,重新设计了DutyIntervalSet2和DutyRosterApp2,做了以下改动:

(1) DutyIntervalSet2委托了MultiIntervalSet而不是LinearIntervalSet

(2) DutyIntervalSet2改写了autoInsert方法:迭代地对每天都随机选择一个员工,可以重复选择

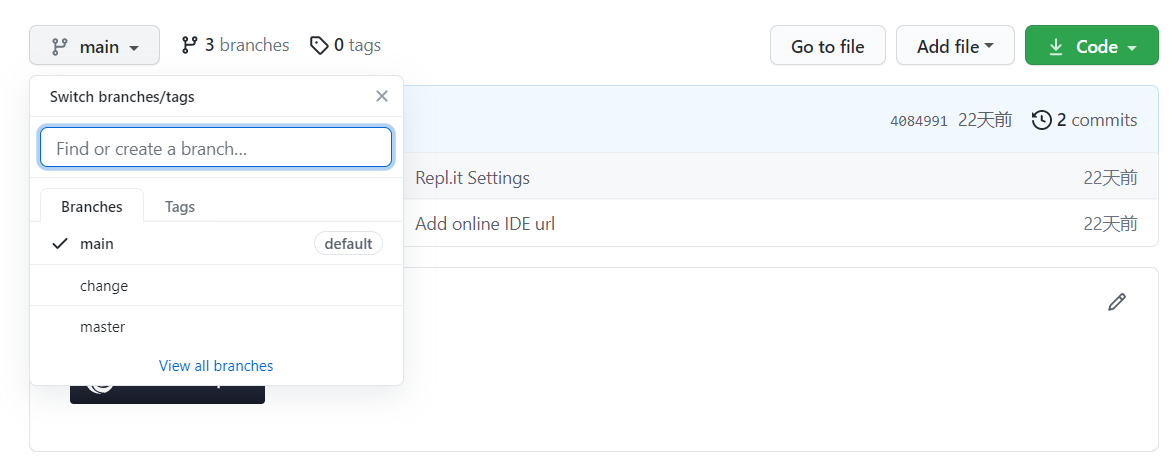
(3) DutyRosterSet2中InsertDuty方法不从列表中删除员工,因为员工可以多次值班

### 变化2

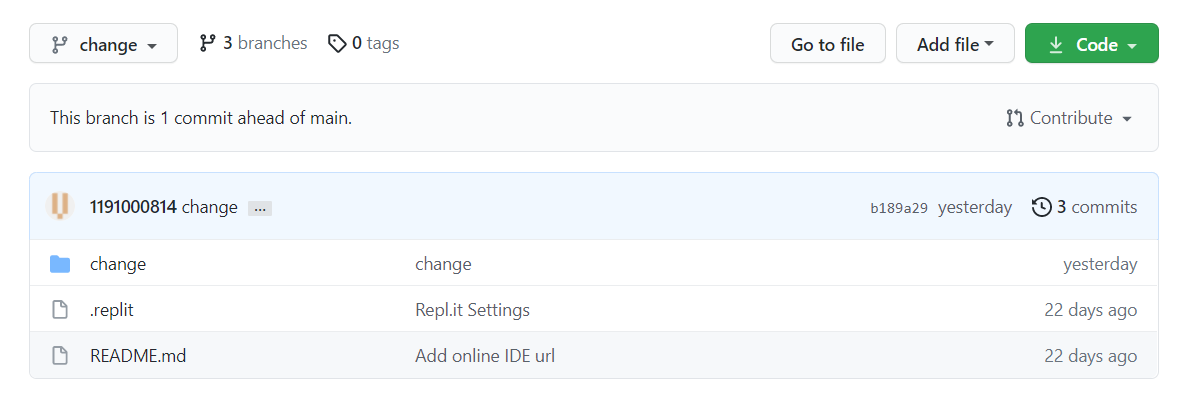
之前的程序CourseScheduleApp可以满足此变化, 因为在CourseSchedule中考虑了周数的问题,所以不可能会有同一时间课程的情况,故在change中没有重写CourseIntervalSet和CourseScheduleApp

## 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 |
| 6.11 | 12:30 | 阅读理解清楚题目和要求 | 按计划完成 |
| 6.16 | 18:00 | 发现自己写错了,删掉重写,并改写了部分模板 | 按计划完成 |
| 6.17 | 15:30 | 重新写好了三个IntervalSet类: CommomIntervalSet,MultiIntervalSet,  LinearIntervalset | 按计划完成 |
| 6.24 | 17:00 | 完成了第一个命令行客户端DutyRosterApp的基本代码 | 按计划完成 |
| 6.27 | 21:30 | 基本完善好了DutyRosterApp程序 | 按计划完成 |
| 6.29 | 8:30 | 做完了剩下两个App程序 | 按计划完成 |
| 6.29 | 10:00 | 完成了change分支 | 按计划完成 |
| 6.30 | 17:00 | 完成了正则表达式匹配 | 按计划完成 |
| 6.30 | 21:00 | 完成了Similarity类 | 按计划完成 |
| 7.1 | 8:00 | 完成后了所有的Test测试类 | 按计划完成 |
| 7.2 | 21:00 | 完成了实验报告 | 按计划完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| Scanner类输入nextInt时,不能try catch 输入匹配错误 | InputMatchException是IOException不能try catch,可以通过将字符串转化成int,try catch转化异常 |
| 正则表达式匹配失败多次 | 转义符写错,运算符周围多写了空格,大括号方向打错 |

# 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) 写之前要看清楚要求,不要还不知道有模板就直接写,结果还是的删掉重写

(2) 要弄懂子类与父类.以及委托之间的关系,不然写了好多发现写错了又要删掉重写

(3) 正则表达式要看清楚再写,转义符要弄清楚

## 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？

答: 复用节省了很多代码和工作时间,体会到了。

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？

答: 时刻防止数据泄露,如果有一个地方有数据泄露,造成的影响可能会很大,当然要坚持这么做。

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？

答: 难处体会到了,模板水太深,把持不住,最后直接把它给改了,乐处嘛,就是少写了点代码吗……

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？

答: Java的正则表达式也太low了,不建议用Java来写解析器……

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？

答: 难在乱七八糟的东西要整的好多,各种检查,规约,注释,AF,RI之类的,差不多得了……,还能怎么克服,硬着头皮写下去呗。

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？

答: 经验教训是,首先必须得弄清楚各个接口和实现类,子类的关系,整明白再写,否则写到半路发现半年前就翻车了……

1. 关于本实验的工作量、难度、deadline。

答: deadline太恐怖了,直接就在考试前几天,难度还好,但是工作量不是一般的大。

1. 到目前为止你对《软件构造》课程的评价。

1.这门讲的特别快,我一点都跟不上,刘铭老师不管是讲课还是讲话,速度都非常快,完全跟不上。

2.软件构造的ppt完全照抄MIT,为什么不能多翻译一点中文,多删除几页ppt,ppt上好多页都是废话连篇,真正的干货,需要我们认真看的根本没几页,老师下学期能不能辛苦一下,精简一下ppt,几乎全英文的ppt,真的很难看懂,更别说理解了,老师您不会认为我们还会靠这ppt学习英语吧。

3.老师在ppt上注释了很多,十分辛苦,给老师提个建议,能不能不要用**其他颜色背景+白色字体**来注释,因为我们很多学生都要打印ppt(哪有钱买平板啊),这样的字体打印出来完全就是一堆乱码,很难看清楚。

4.实验时间安排能不能再合理一点,还有一周就考试了,基本上还没人能做完lab3,相当一部分同学还没正式开始lab3,lab3的代码量之大也超出了我的想象……毕竟前几周也都是考试,或者老师可以延长至考试结束之后几天。

5.课程群没有匿名,几乎没有同学回答问题,老师说是学校政策,而且QQ群里老师也基本不回答学生的问题,但是类似于我们计算机系统,形式语言与自动机课程群,都是允许匿名,同学们讨论问题十分积极,学习热情和对老师的评价比软构好不知道要好多少倍。

6.这门课虽说是培养软件构造能力,但是实际上就还是像数据结构一样写写代码,并没有涉及到项目之类的,不过是换了种语言又学了一门类似数据结构的课程(学的知识哪有数据结构实用啊),类似于JavaEE课程但是还没把Java讲全讲透彻(甚至不如我寒假看的黑马内容充实干货多)

7.另外,报告的内容可以再精简一些,报告只需提炼出编程思想,不需要把每个函数都挨个查到,不然很多地方纯属是粘贴代码或图片。

8.综上,软构不是一门难课,相对于CS甚至可以说很简单(因为在上述条件下我仍然没有崩掉),只是老师们设置这门课时增加了难度,如何增加难度?通过在讲课时让学生们难以听懂,通过提供难以理解的学习资料,通过减少答疑和提问等方式抑制学生们的学习热情……最后,确实使这门课难度增加了。