

**2021年春季学期  
计算学部《软件构造》课程**

**Lab 3实验报告**

|  |  |
| --- | --- |
| 姓名 | 宋保聚 |
| 学号 | 1190200203 |
| 班号 | 1903010 |
| 电子邮件 | 1916951109@qq.com |
| 手机号码 | 18617930402 |

**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

根据实验手册简要撰写。

目标是编写具有可复用性和可维护性的软件，主要使用一下软件构造技术：

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

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

在官网上下载Eclipse、JDK和Git并安装。在为JDK配置环境变量的时候，在系统环境变量中新建一个叫“JAVA\_HOME”的环境变量，变量值为JDK的安装路径。然后在Path的环境变量中添加一个“%JAVA\_HOME%bin”的变量值。省去很多其它的步骤。

在这里给出你的GitHub Lab3仓库的URL地址（Lab3-学号）。

https://github.com/ComputerScienceHIT/Lab3-HIT--hit1190200203

# 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

三个应用：

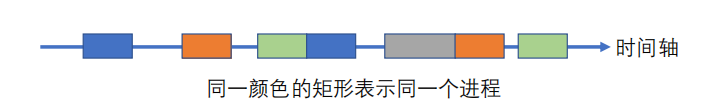
1.值班表管理（DutyRoster）：

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。



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”，以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，每周一上午8:00-10:00 和每周三上午8:00-10:00 在正心楼 11 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（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1-2 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。



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。

共性：对于时间的规划，安排管理，都要考虑到先后因素，并且对于时间都或多或少具有周期性，间隔性。

差异：1.对于员工的安排需要连续；2.更强调时间的先后顺序，并且有可能不发生事件；3.对于课表要考虑到老师和学生是否有时间，并且周期性较强。

## 面向可复用性和可维护性的设计：IntervalSet<L>

考虑上节给出的三个应用，其中都包含了具有不同特征的“时间段集合”对象，为了提高软件构造的可复用性和可维护性，可为其设计和构造一套统一的ADT。

### IntervalSet<L>的共性操作

**3.2.1.1：**IntervalSet<L>的实现

L代表每个时间段附着的一个特定的标签，且标签不重复

三个应用的共性操作包括：创建一个空对象，在当前对象中插入新的时间段和标签，去掉一个时间段获得当前对象中的标签集合，从当前对象中移除某个标签所关联的时间段，返回某个标签对应的时间段的开始时间，返回某个标签对应的时间段的结束时间：把这些操作都放入IntervalSet<L>接口中封装起来。

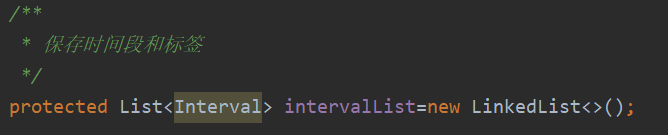
其方法：

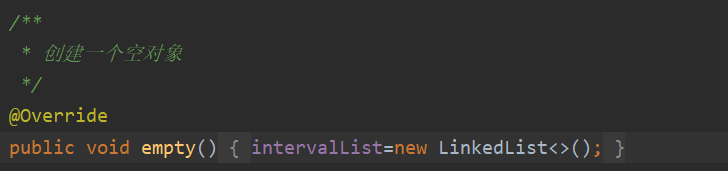
|  |  |
| --- | --- |
| empty() | 创建一个空对象 |
| void insert(long start, long end, L label) | 在当前对象中插入新的时间段和标签 |
| Set<L> labels() | 获得当前对象中的标签集合 |
| boolean remove(L label) | 从当前对象中移除某个标签所关联的时间段 |
| long start (L label) | 返回某个标签对应的时间段的开始时间 |
| long end (L label) | 返回某个标签对应的时间段的结束时间 |

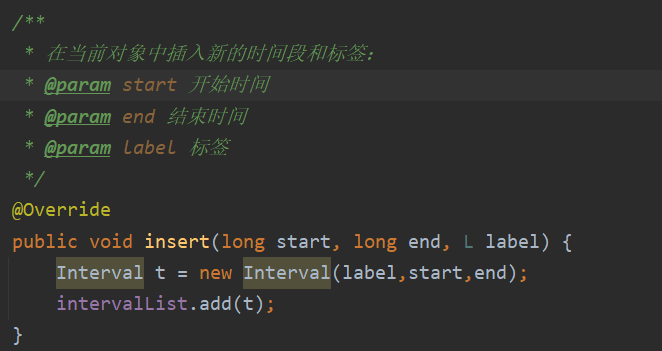
3.2.1.2CommonIntervalSet<L>的实现

IntervalSet的实现类，具体实现IntervalSet各种方法

具体方法如下：

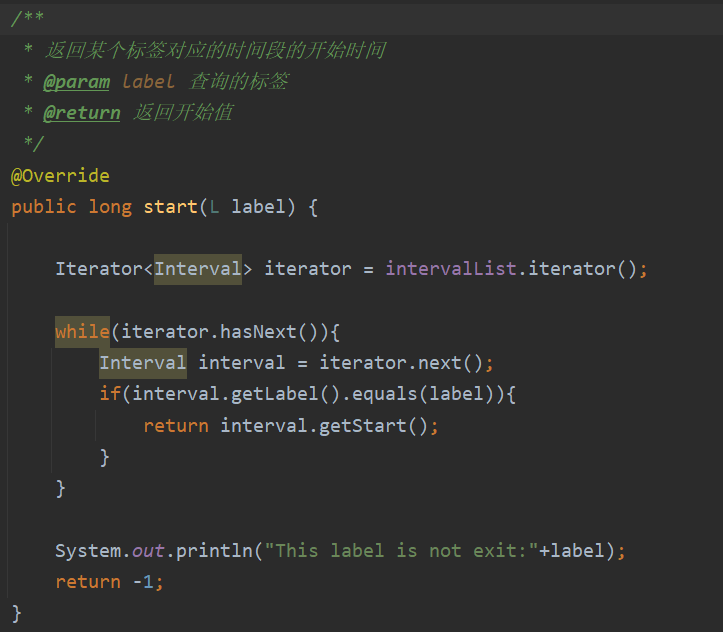


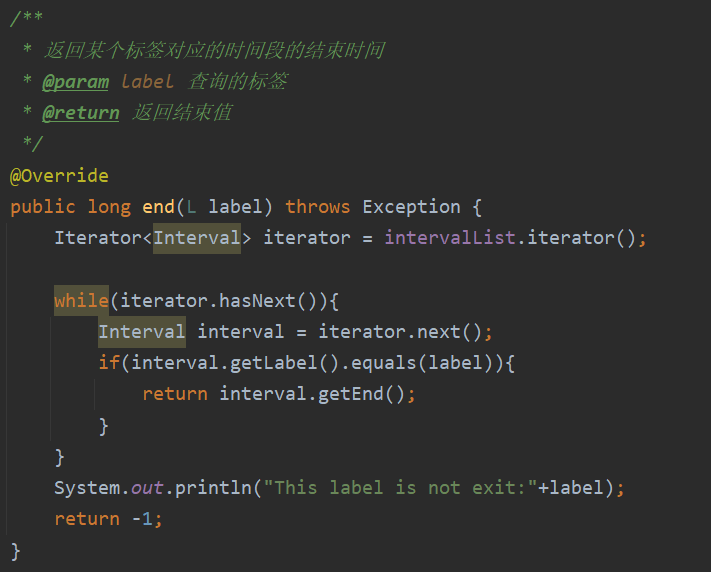


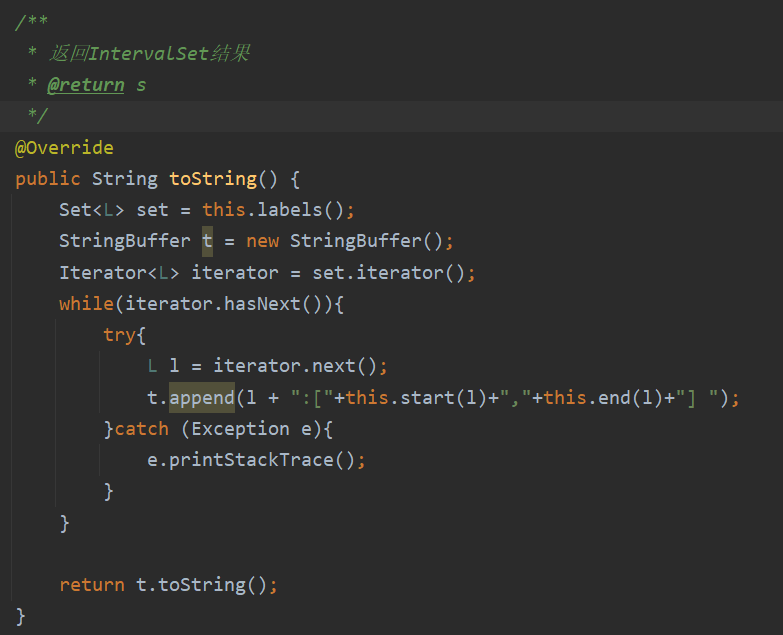




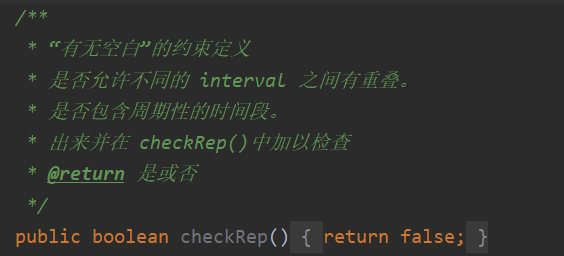








rep、RF、AI、Safety from Rep Exposure的设计



### 局部共性特征的设计方案

局部共性操作有初始化时间段集合，设置、删除时间段，更改该时间段的标签，获取时间段 安排内容，例如获取上课老师的各种信息，如手机号等。

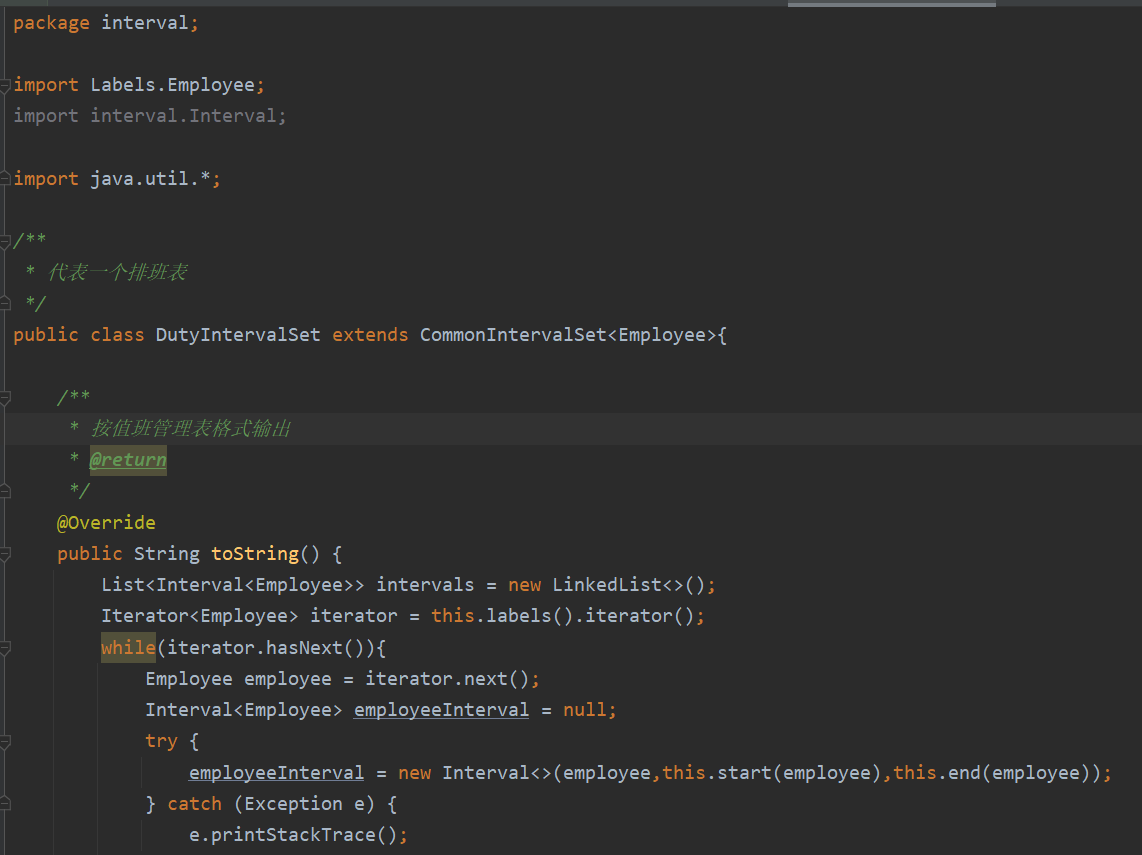
在方案二中，将各特殊操作分别放入底层的应用子类。例如：

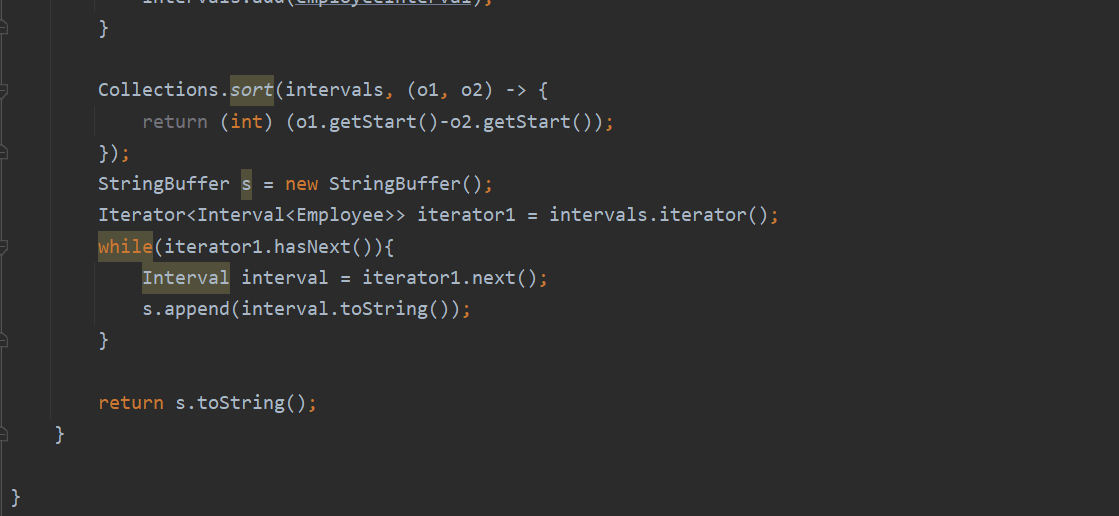
在实现排课和进程调度的子类中实现上述 insert 方法，其他应用的子类不需实 现该方法。

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

3.2.3.1值班表管理（DutyIntervalSet）

在 DutyIntervalSet 类中extends了 CommonIntervalSet ，使其按照值班表的格式输出。代码如下：





## 面向可复用性和可维护性的设计：MultiIntervalSet<L>

### MultiIntervalSet<L>的共性操作

**3.3.1.1：MultiIntervalSet<L>**的实现

这是一个 mutable 的 ADT，描述了一组在时间轴上分布的“时间段”（interval），每个时间段附着一个特定的标签，且标签不重复。  
每个标签包含多个时间段集合。

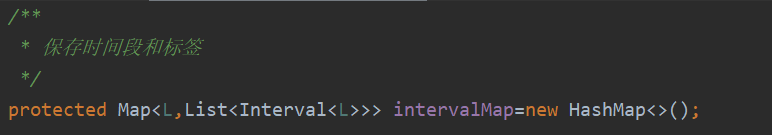
其方法为：

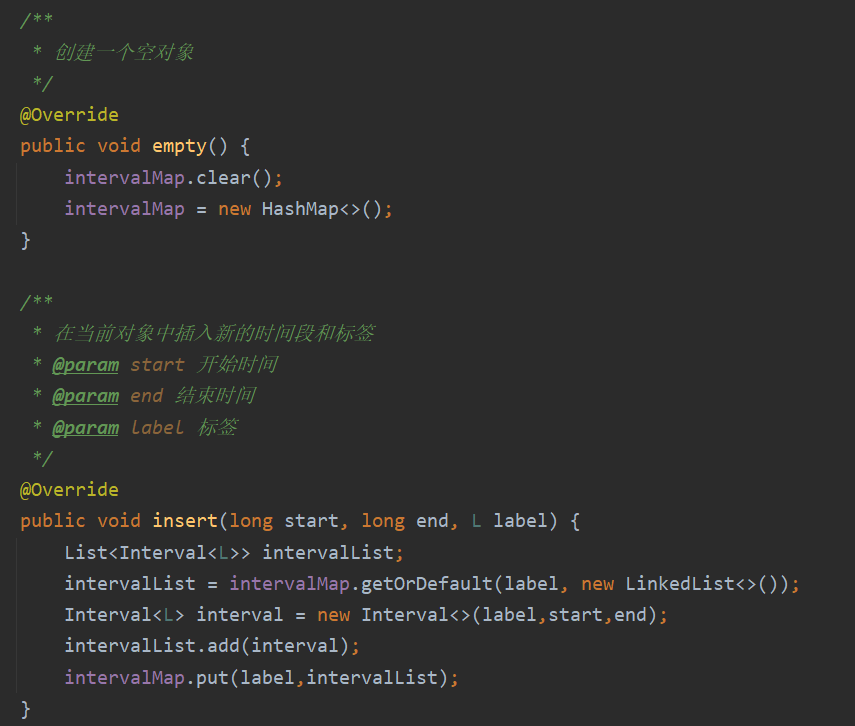
|  |  |
| --- | --- |
| empty() | 创建一个空对象 |
| void insert(long start, long end, L label) | 在当前对象中插入新的时间段和标签 |
| Set<L> labels() | 获得当前对象中的标签集合 |
| boolean remove(L label) | 从当前对象中移除某个标签所关联的时间段 |
| IntervalSet<Integer> intervals(L label); | 从当前对象中获取与某个标签所关联的所有时间段 |

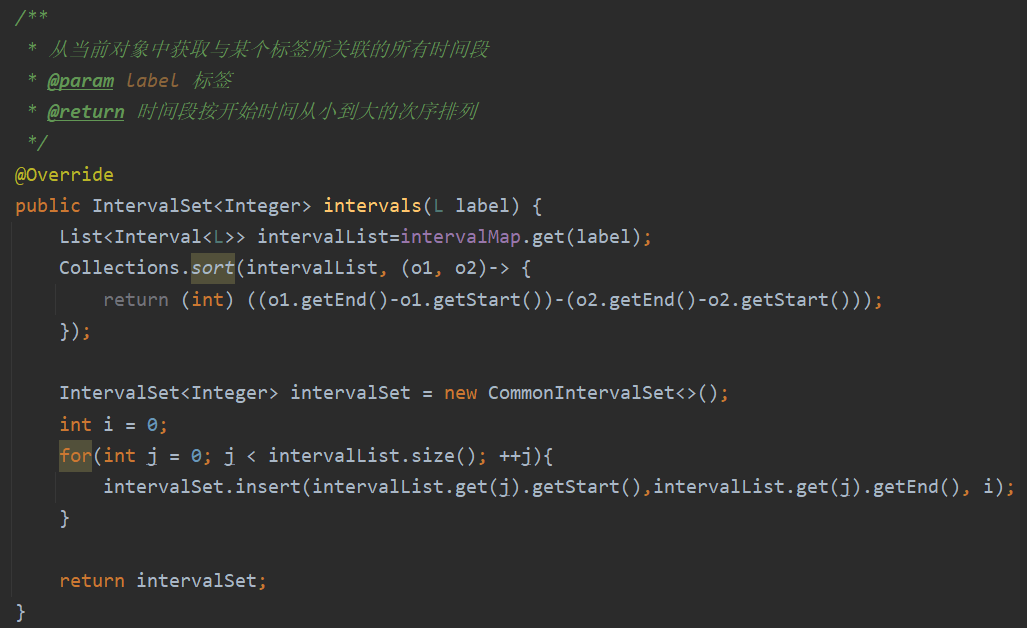
**3.3.1.2：CommonMultiIntervalSet<L>**的实现

这是一个 mutable 的 ADT，描述了一组在时间轴上分布的“时间段”（interval），每个时间段附着一个特定的标签，且标签不重复。每个标签包含多个时间段集合,MultiIntervalSet实现类。

具体实现如下：

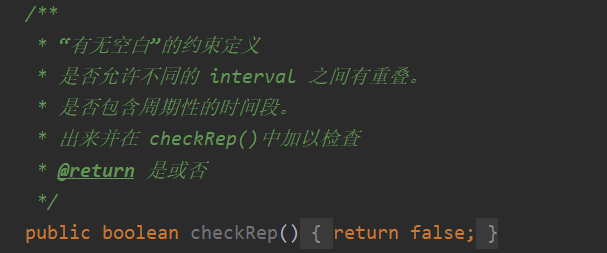








rep、RF、AI、Safety from Rep Exposure的设计



### 局部共性特征的设计方案

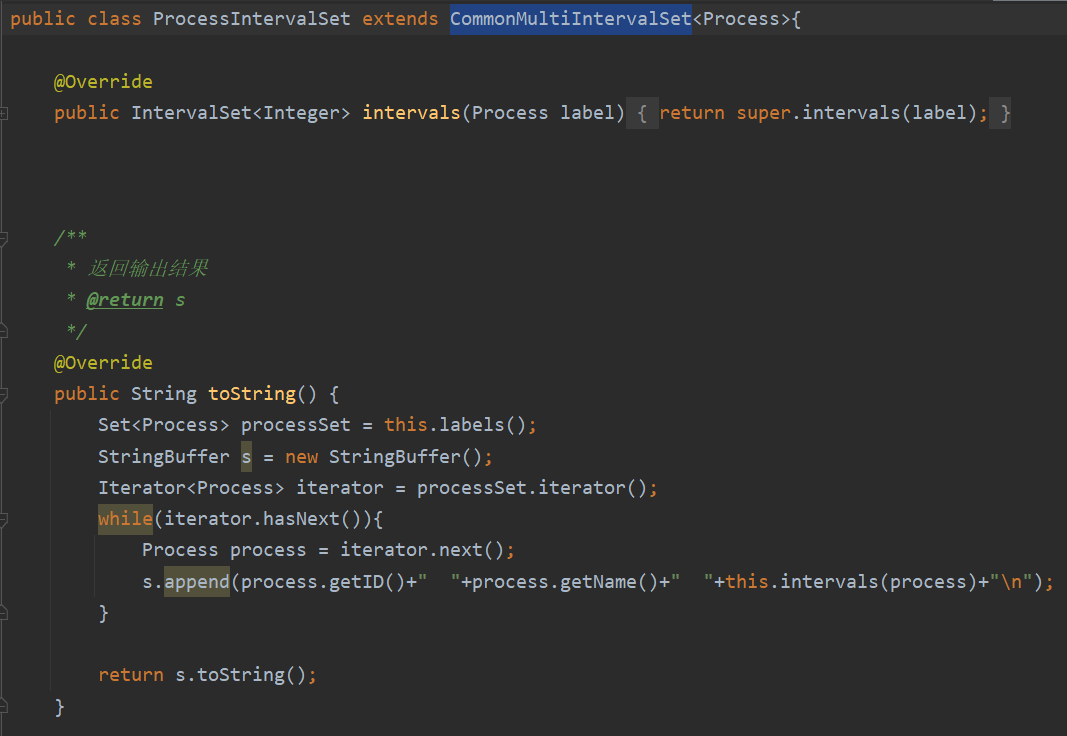
局部共性操作有初始化时间段集合，设置、删除时间段，更改该时间段的标签，获取时间段 安排内容，例如获取进程中的最低开始时间等等。

一个标签可以有多个时间序列，一个标签的时间序列不可重叠，但对于方案二，不同标签的的时间序列可以重叠。

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

3.3.3.1进程表管理（ProcessIntervalSet）

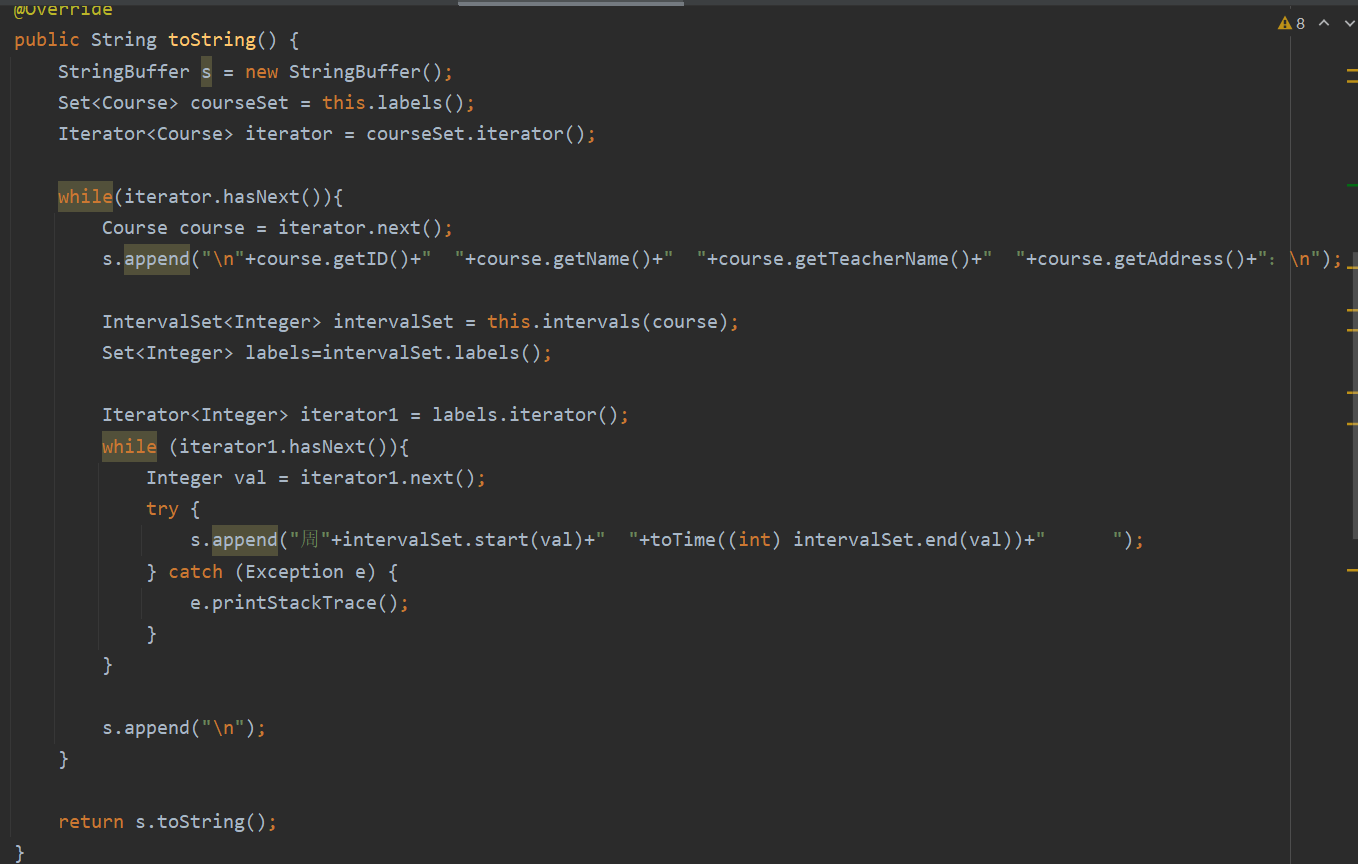
在 ProcessIntervalSet 类中extends了 CommonMultiIntervalSet ，代表一个操作系统对进程的调度记录，并且允许出现空白。代码如下：





3.3.3.2课程表管理（CourseIntervalSet）

在 CourseIntervalSet 类中extends了 CommonMultiIntervalSet ，代表一个班级的课表，并且允许出现空白和重叠。代码如下：





## 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：

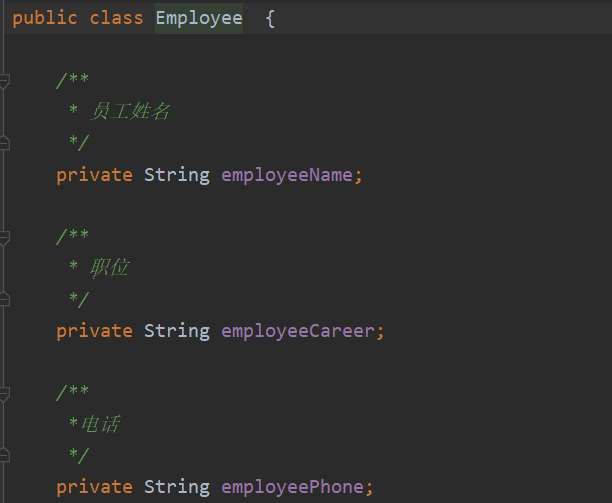
Employee：姓名、职务、手机号码

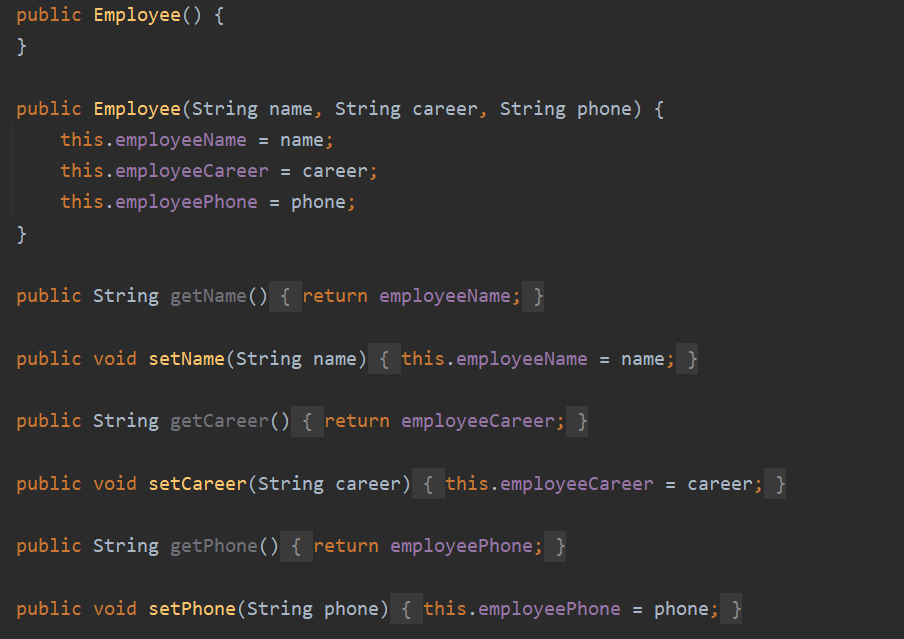
Process：进程 ID、进程名称、最短执行时间、最长执行时间

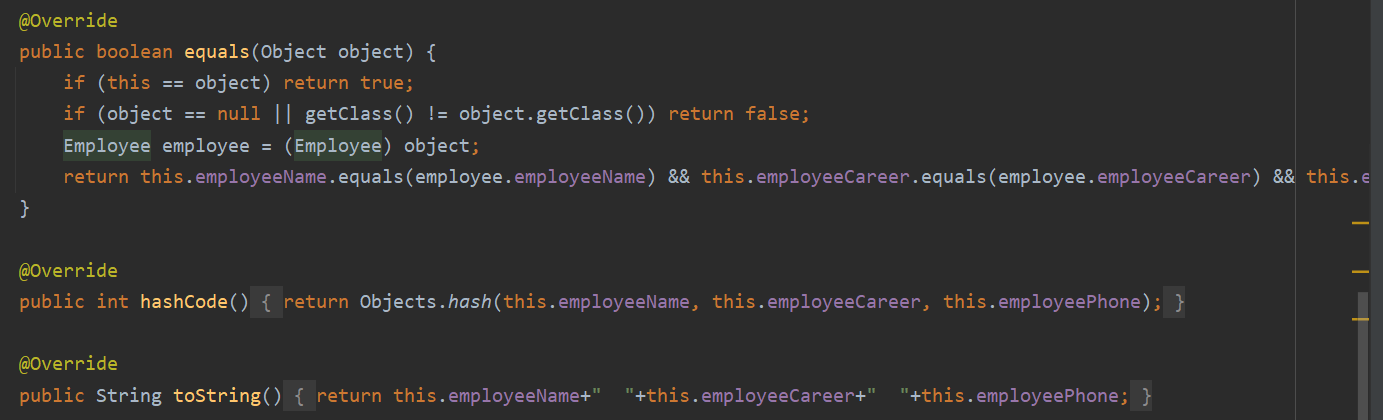
Course：课程 ID、课程名称、教师名字、地点、周学时数

分别实现这些 ADT。

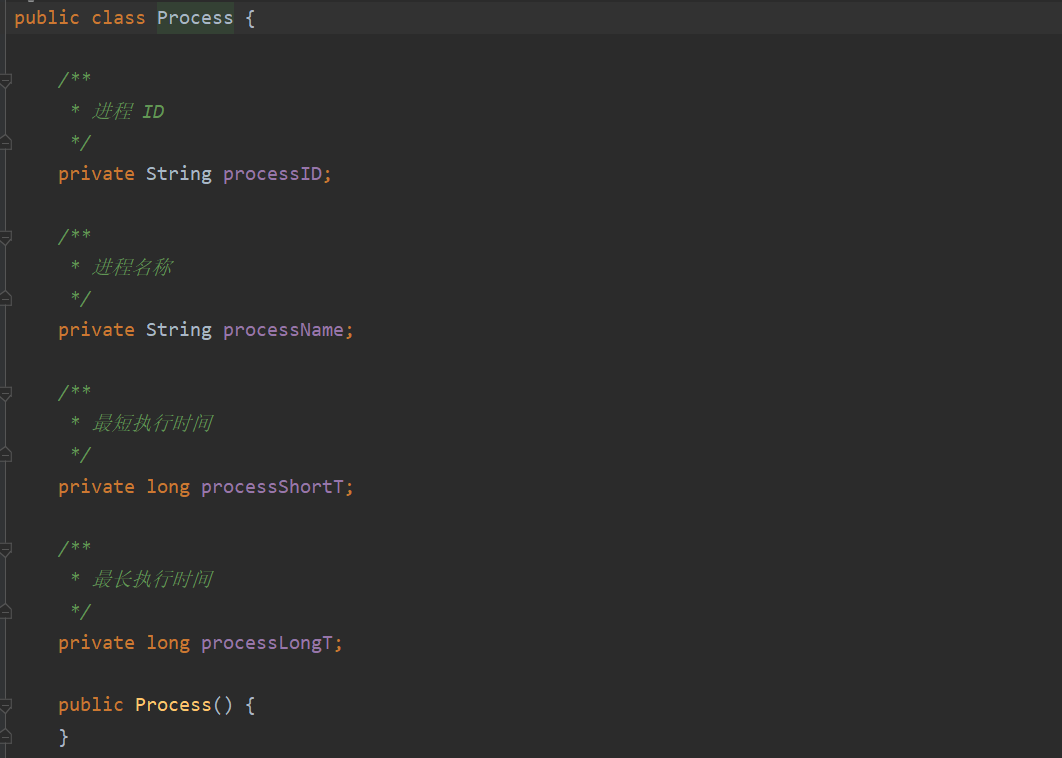
Employee 类：



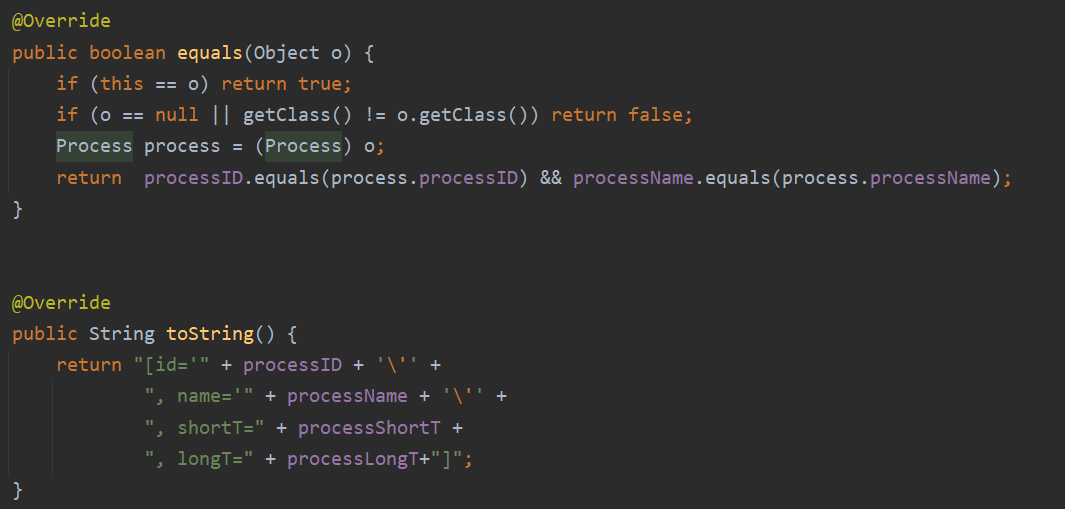




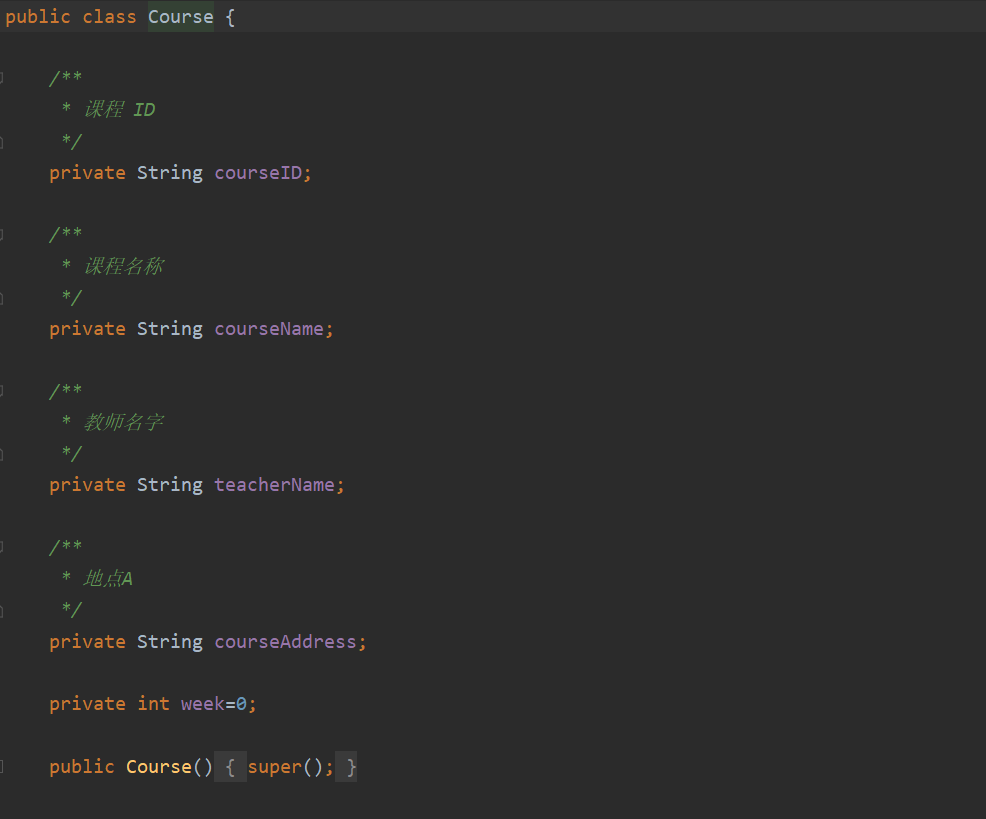
Process类：







Course类：







## 可复用API设计

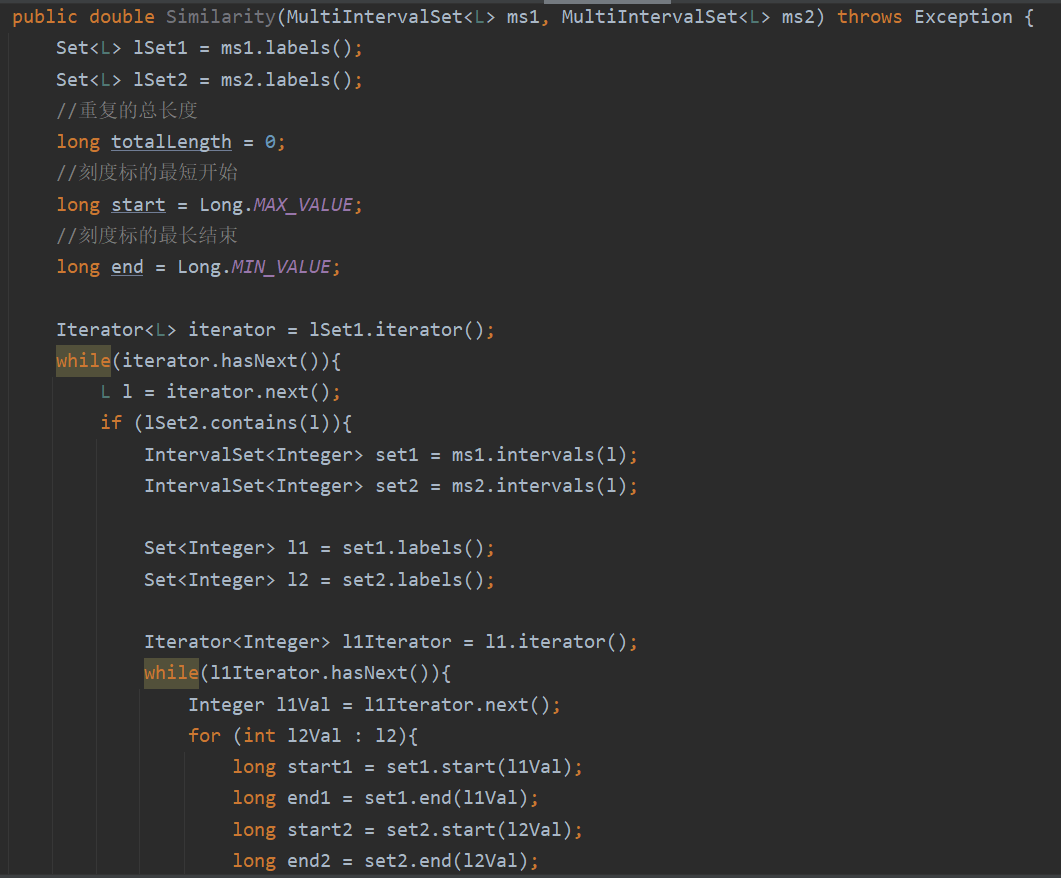
### 计算相似度

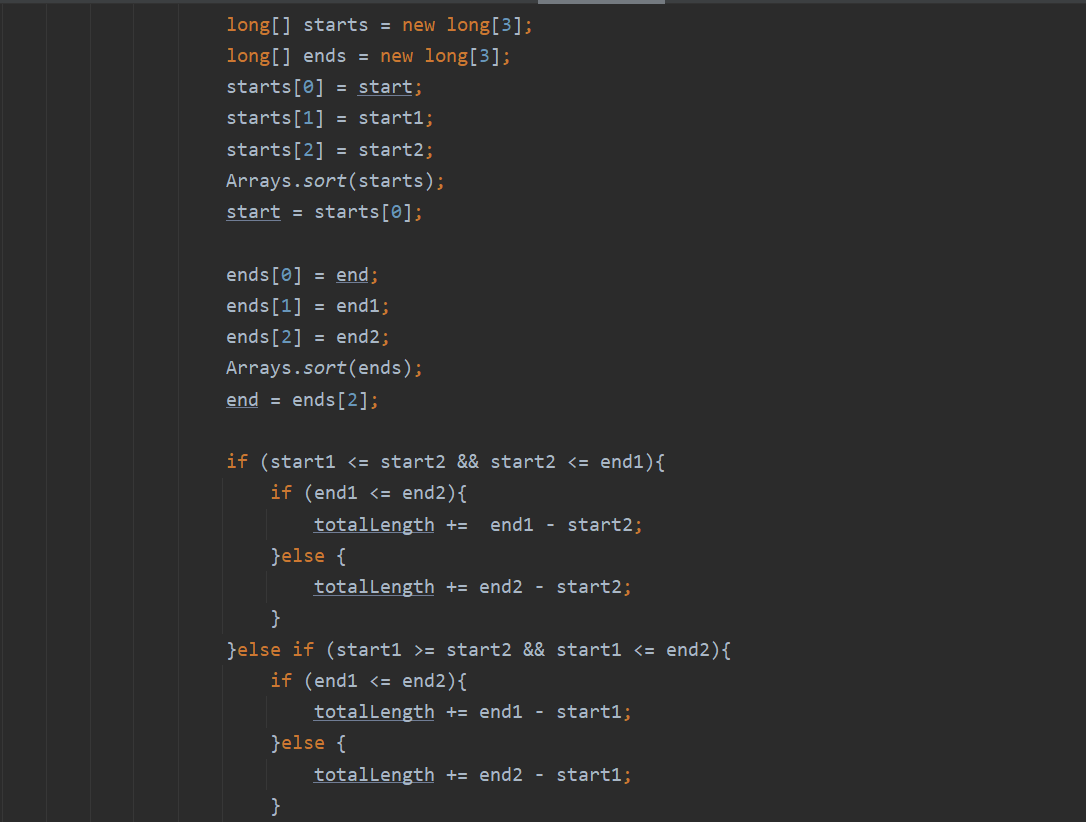
计算两个 MultiIntervalSet 对象的相似度：

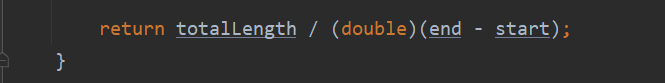
double Similarity(MultiIntervalSet<L> s1, MultiInterval Set<L>

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。 设计思想，遍历两个时间轴，得出重叠的部分，通过作差比较，得出两个 MultiIntervalSet 的 相似度，比较代码如下：



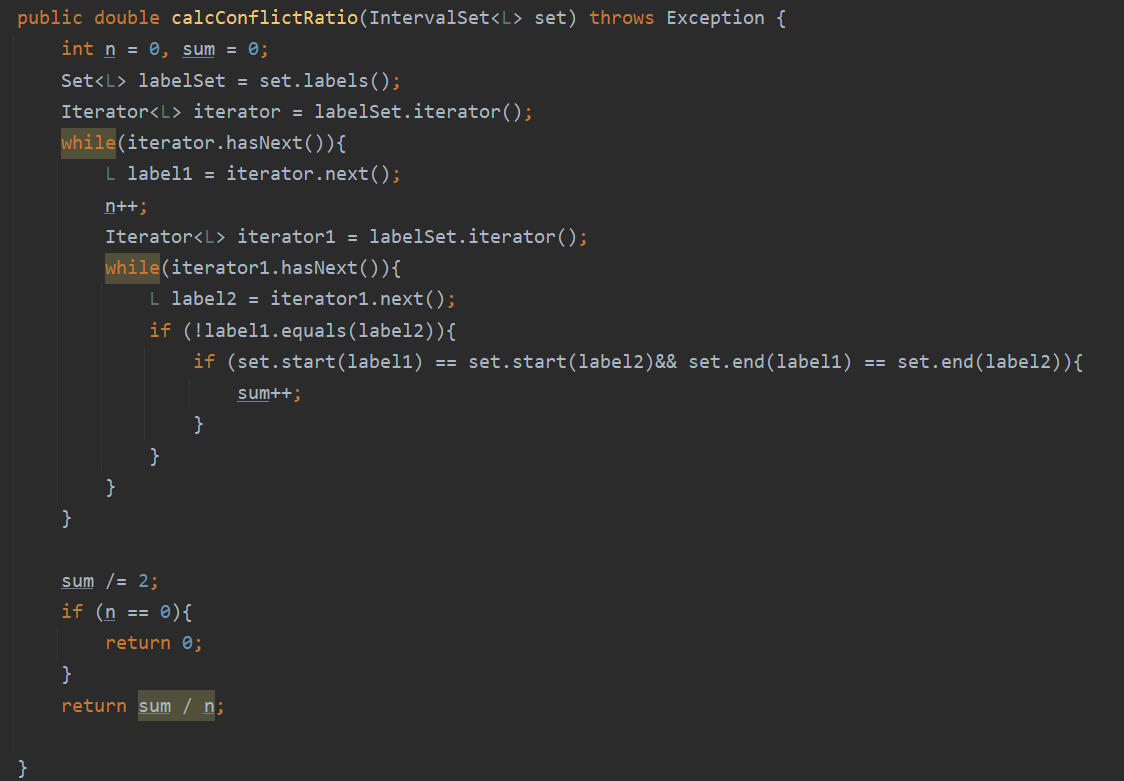


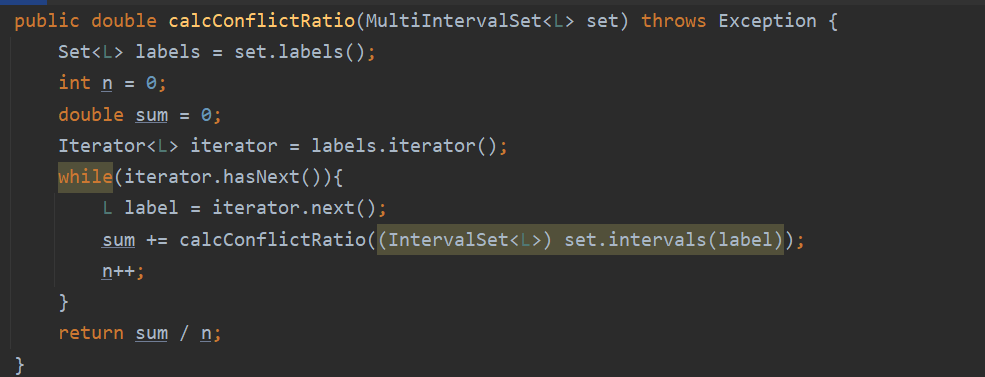


### 计算时间冲突比例

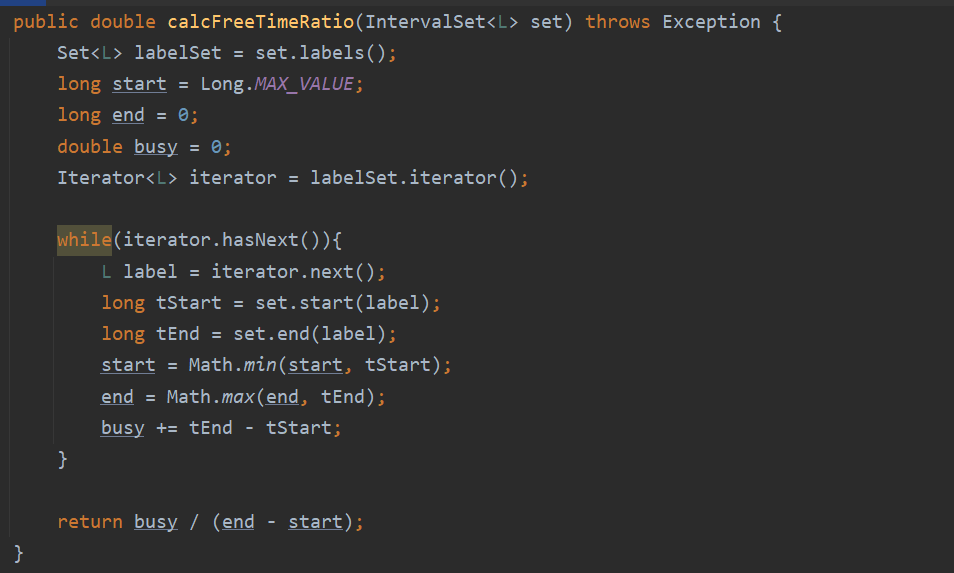
IntervalSet计算冲突方法所谓的“冲突”，是指同一个时间段内安排了两个不同的 interval 对象。用发生冲突的时间段总长度除于总长度，得到冲突比例，是一个[0,1]之间的值@param set IntervalSet @return MultiIntervalSet @throws Exception 异常捕获

具体实现如下：

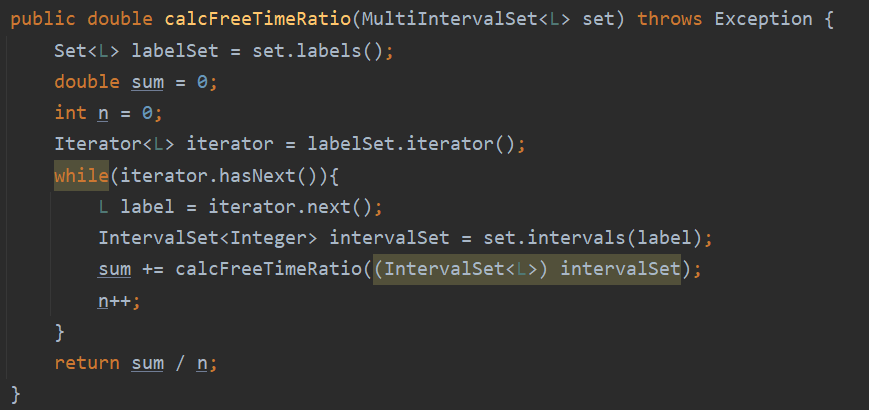


MultiIntervalSet计算冲突方法所谓的“冲突”，是指同一个时间段内安排了两个不同的 interval 对象。用  
发生冲突的时间段总长度除于总长度，得到冲突比例，是一个[0,1]之间的值  
@param set MultiIntervalSet  
@return 冲突比例  
@throws Exception 异常捕获  


### 计算空闲时间比例

计算一个 IntervalSet<L>对象中的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  
 @param set IntervalSet  
 @return 空闲比例  
 @throws Exception 异常捕获  


计算一个 IntervalSet<L>对象中的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  
 @param set MultiIntervalSet  
 @return 空闲比例  
 @throws Exception 异常捕获



## 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

排班管理系统中，需要实现的功能为：

1. 设定排班日期。

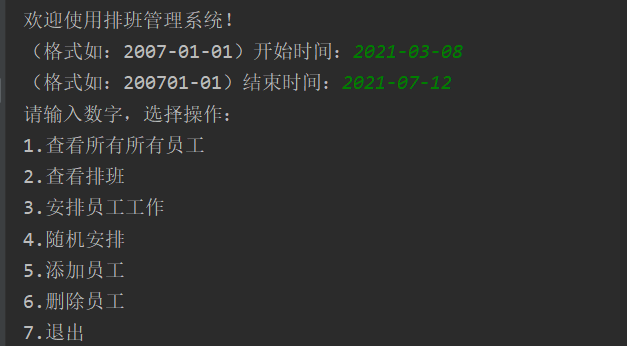
2. 增加一组员工

3. 手工选择员工的排班

4. 随机生成排班表

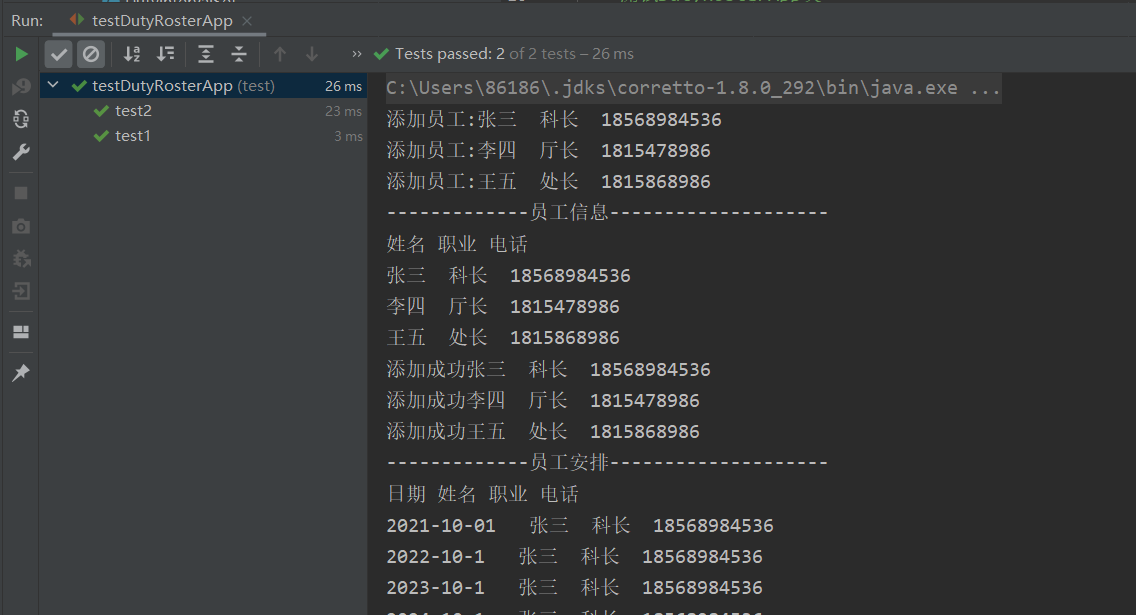
5. 可视化排班表。

APP 设计为命令行 APP，用户根据控制台提示输入相关信息，以实现值班表的安排。首次进入 APP 后，需输入值班安排的起始时间与终止时间，输入格式已给出：



功能设计：在 DutyRosterApp中使用之前设计好的 DutyIntervalSet 这一 ADT，并定义一个Employee的List以存储员工的信息，所有员工的信息都存在这个List中，包括已分配的员工信息与未分配的员工信息，而DutyIntervalSet中只存有已分配好值班时间的员工。

测试如下：



### 操作系统的进程调度管理系统

操作系统进程调度管理系统需要实现的功能为：

1. 增加进程，使用addProcess方法，可以向 ProcessScheduleApp 中添加新的进程。

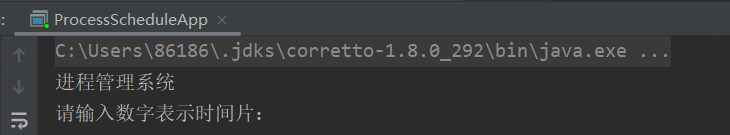
2. 随机调度：当前时刻（设定为 0）启动模拟调度，随机选择某个尚未执行结束的进程在CPU 上执行（执行过程中其他进程不能被执行），并在该进程最大时间之前的 任意时刻停止执行，如果本次及其之前的累积执行时间已落到[最短执行时间，最长 执行时间]的区间内，则该进程被设定为“执行结束”。重复上述过程，直到所有进程都达到“执行结束”状态。在每次选择时，也可“不执行任何进程”，并在后续 随机选定的时间点再次进行进程选择。

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

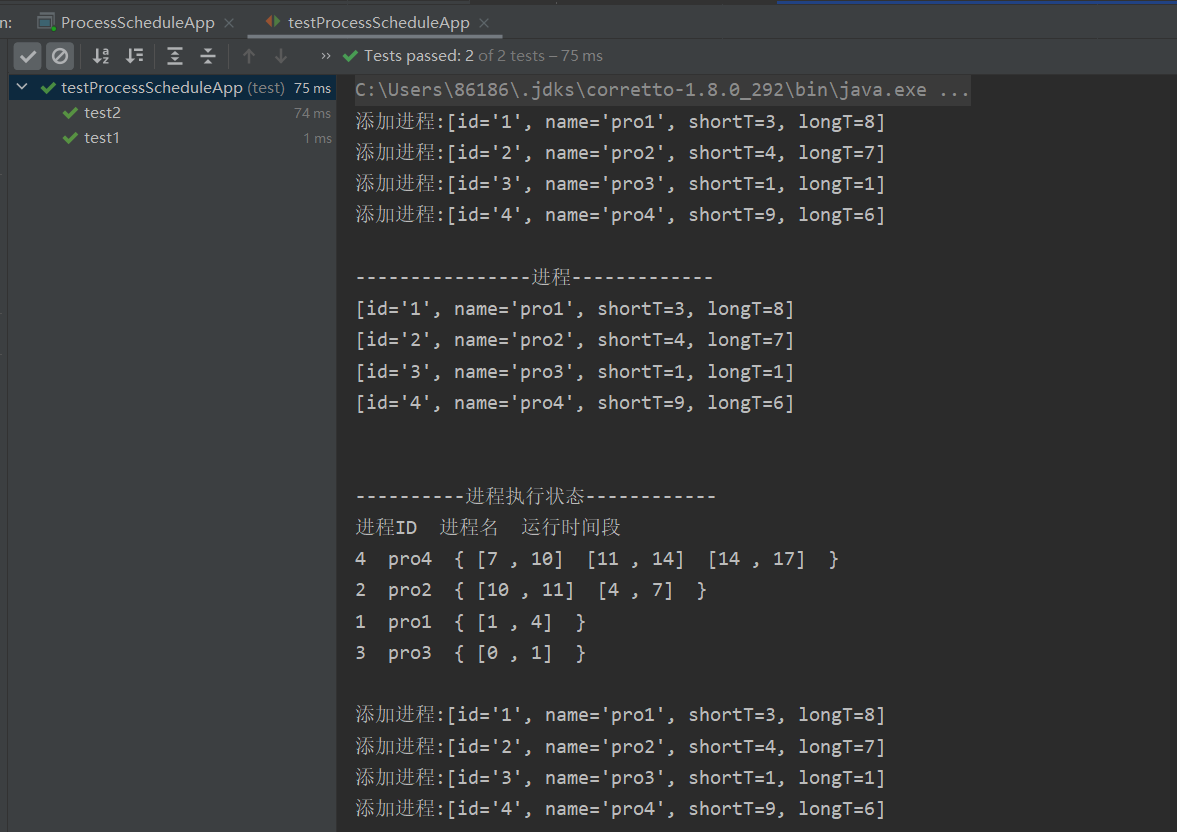
的进程。

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。



测试如下：



### 课表管理系统

针对课表管理系统，所需完成的功能为：

开始时设定学期开始日期（年月日）和总周数（例如 18）；

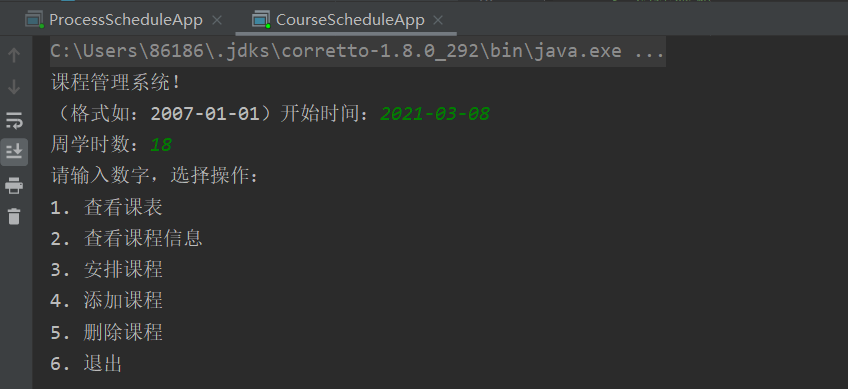
1. 查看课表

2. 查看课程信息

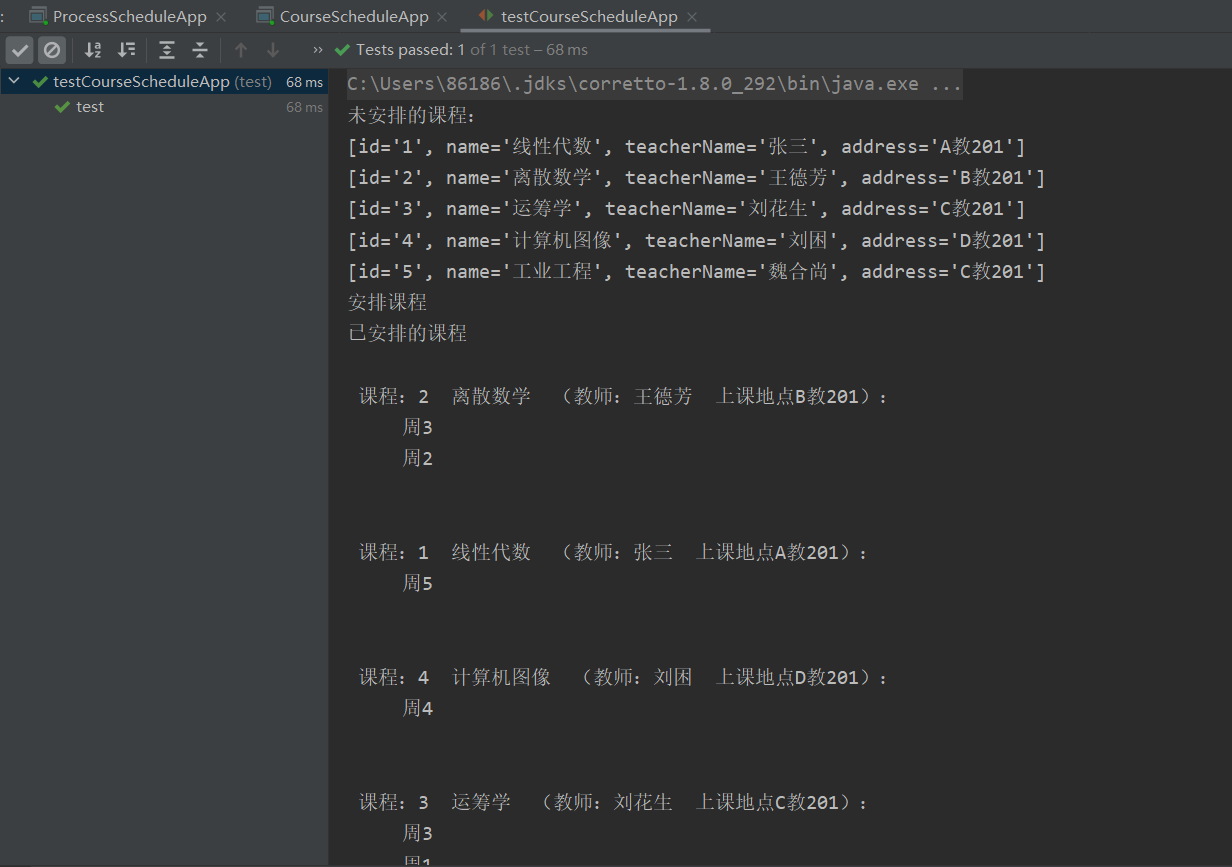
3. 安排课程

4. 添加课程

5. 删除课程



测试如下：

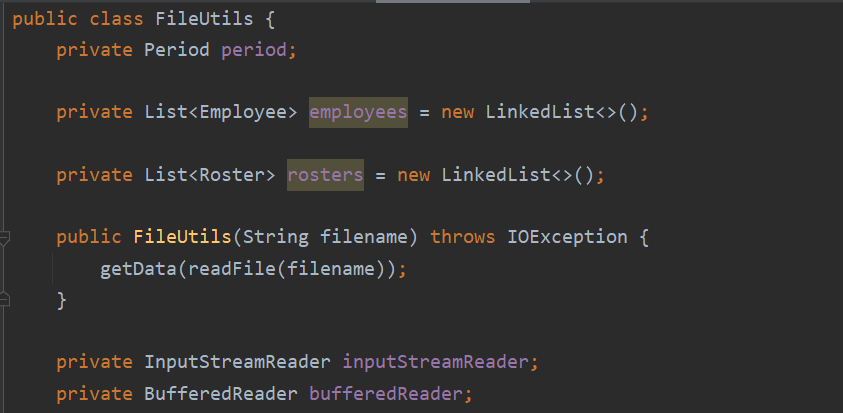


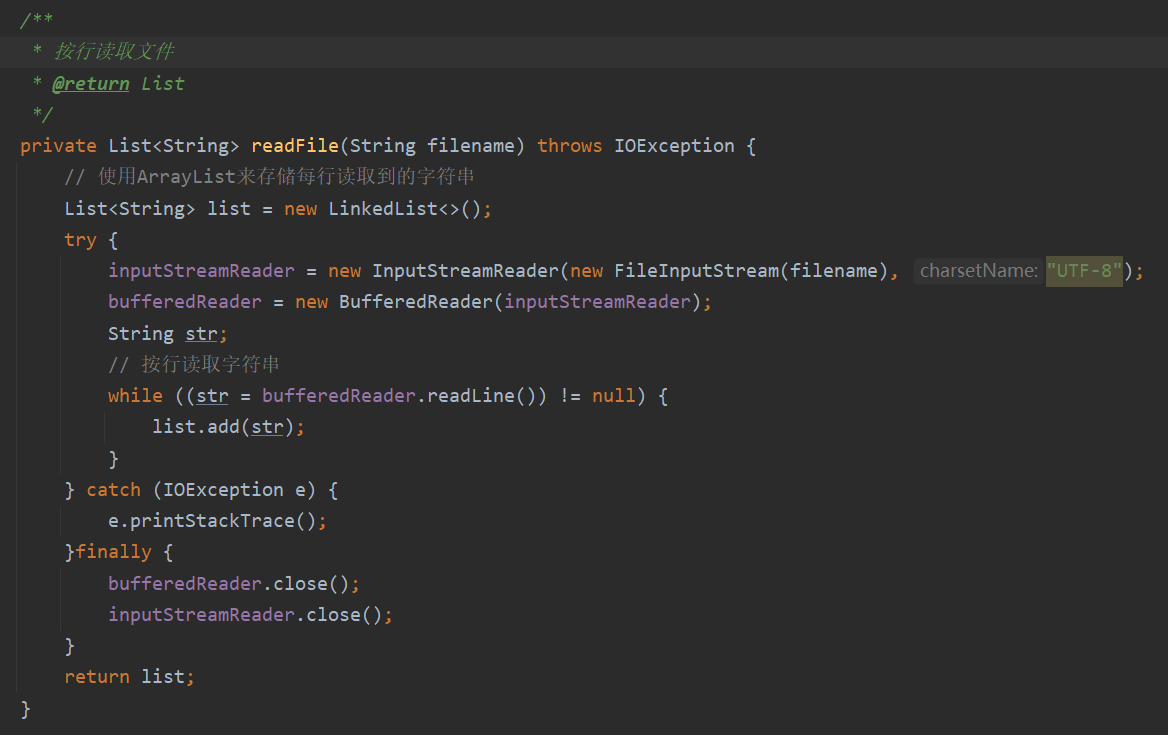
## 基于语法的数据读入

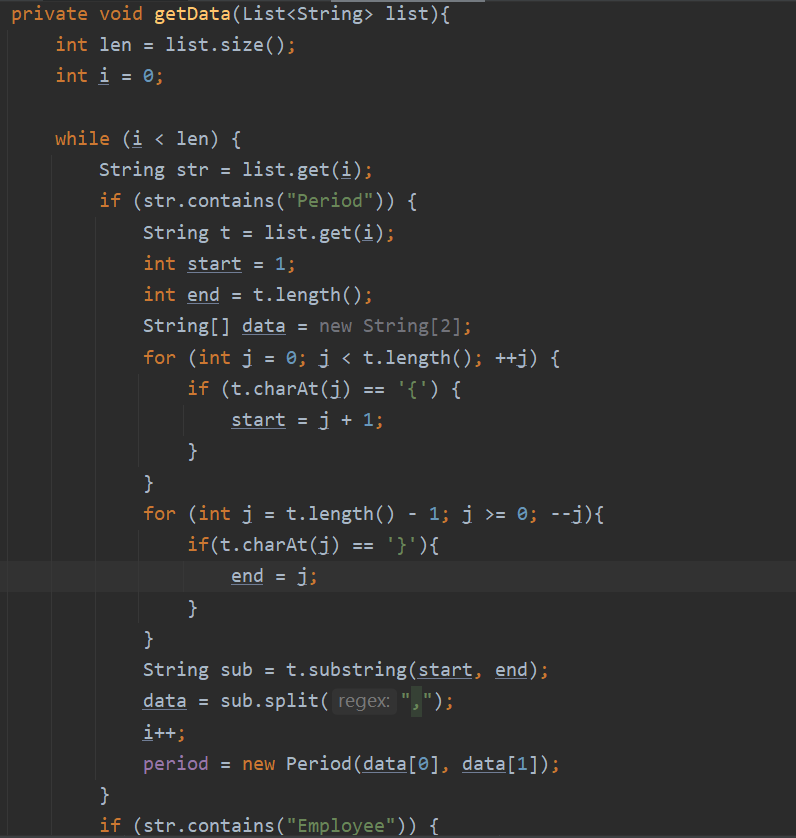
实验指导书中给出的文件中的字符串数据为，以此来读取文件内容并使用正则表达式来解析该文件。

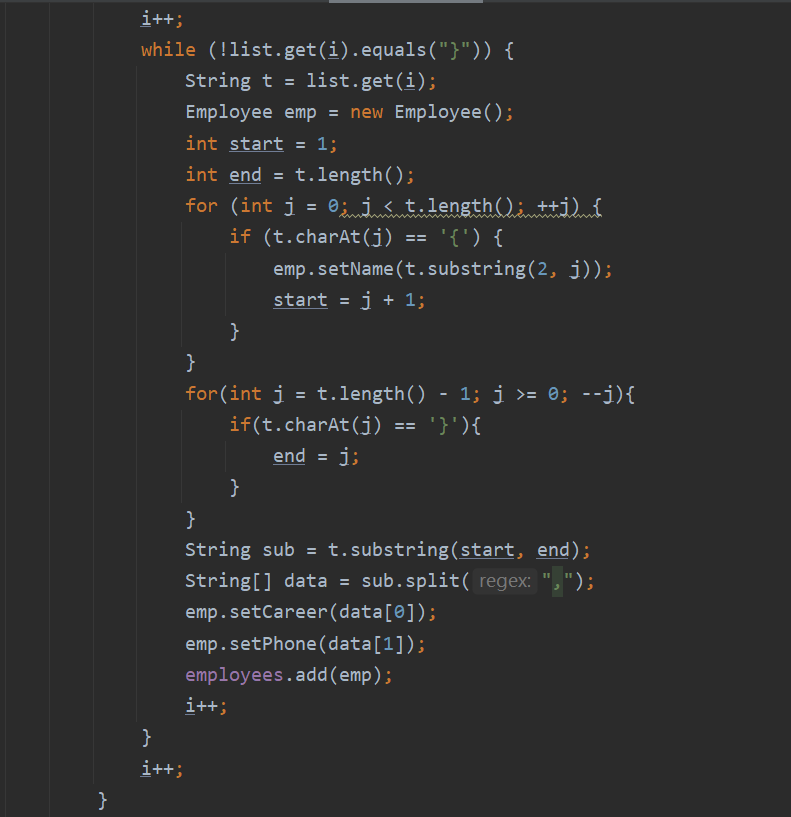
定义了FileUtils和ParserDateUtils来对不同的对象读取文件，具体代码如下：

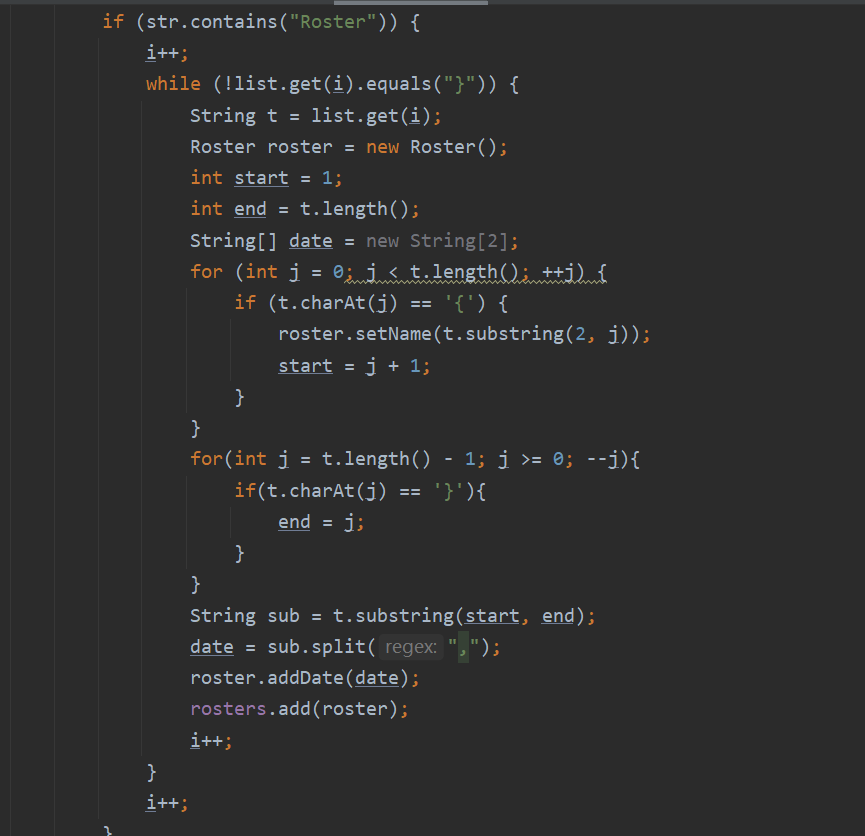
FileUtils

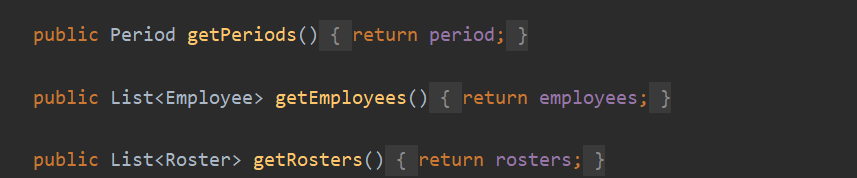




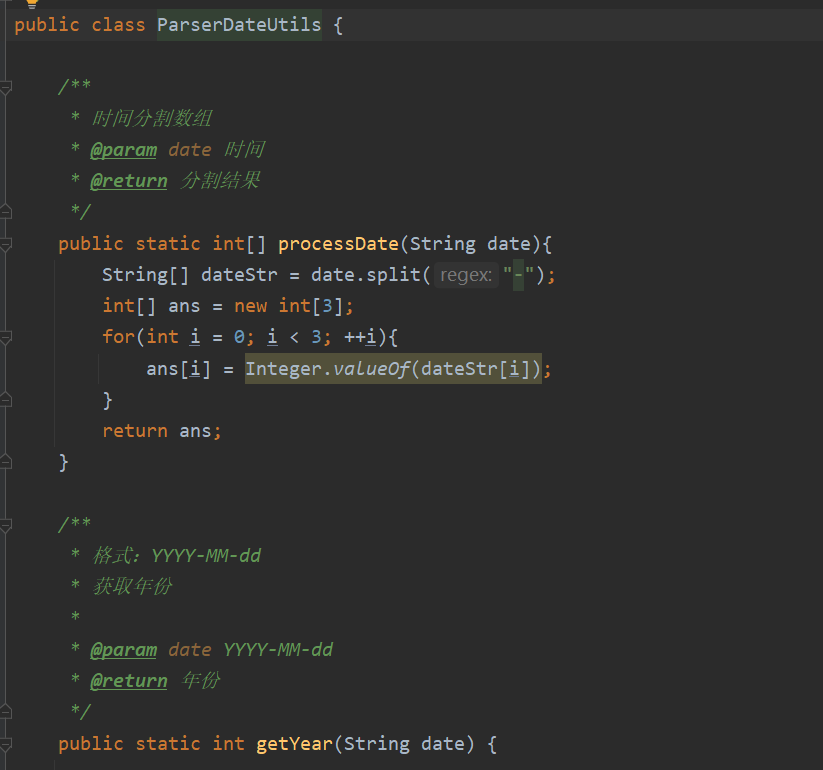


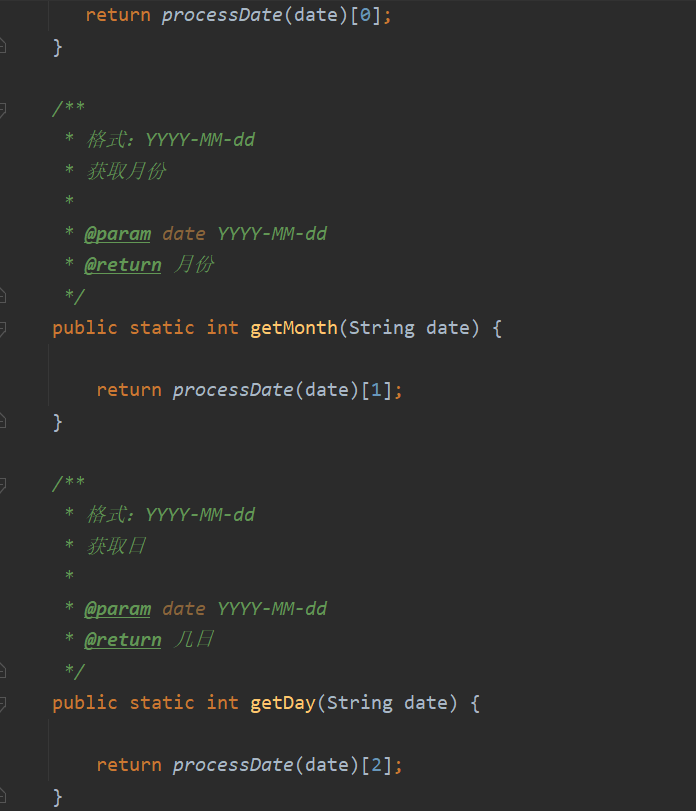


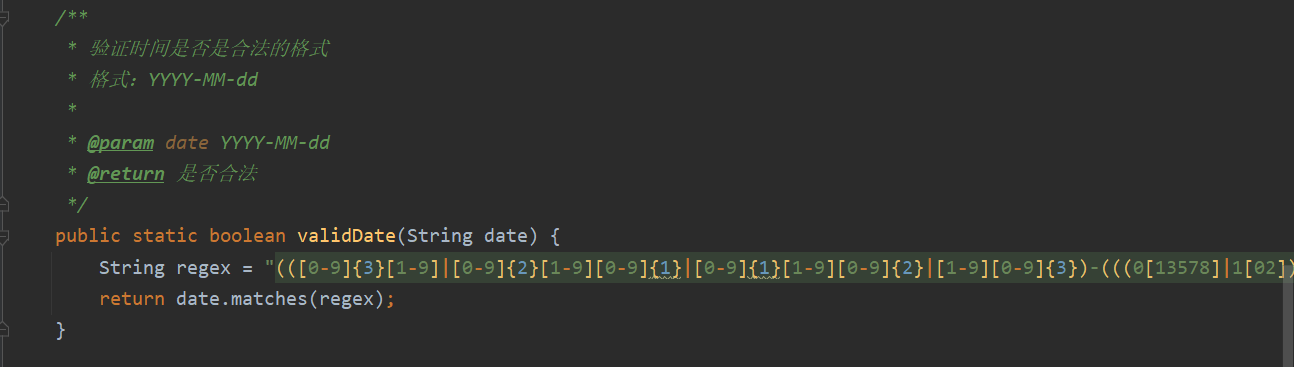




ParserDateUtils







## 应对面临的新变化

### 变化1

评估之前的设计是否可应对变化、代价如何

如何修改设计以应对变化

### 变化2

评估之前的设计是否可应对变化、代价如何

如何修改设计以应对变化

## 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 |
| 6.15 | 18：30-20：30 | 明白实验要干什么，弄懂整体思路，并编写第一个ADT | 未完成 |
| 6.22 | 18：30-20：30 | 完成3.2的全部内容 | 完成 |
| 6.29 | 18：30-20：30 | 提取共性并实现 | 未完成 |
| 6.30 | 14：00-17：30 | 选择方案并试用 | 完成 |
| 6.30 | 19：00-23：00 | 完成3.3全部内容 | 完成 |
| 7.1 | 14：00-17：30 | L的设计 | 完成 |
| 7.1 | 19：00-23：00 | 可复用API设计 | 未完成 |
| 7.2 | 14：00-17：30 | 可复用API设计 | 完成 |
| 7.2 | 19：00-23：00 | 三个管理系统的设计 | 未完成 |
| 7.3 | 14：00-17：30 | 三个管理系统的设计 | 完成 |
| 7.3 | 19：00-23：00 | 值班管理的拓展功能 | 完成 |
| 7.4 | 14：00-17：30 | 新的变化 | 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 |
| 初始ADT的建立思路 | 通过自己的思考和参考老师的讲义，以及询问同学的思路，取长补短克服。 |
| 如何抽象共性方法提高复用性 | 仔细阅读老师的讲义和网上查找相关资料 |
| 正则表达式的应用 | 上网查找相关说明和资料，并通过自己的尝试一步一步建立起来。 |

# 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

这次实验将课程上讲到的各种设计方法应用于实际编程中，极大的提高了程序的可复用性，对自身的代码能力有了很大的提高。

但是关于泛型的使用还是不太会，需要继续联系。

## 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编程复用性高，面向应用编程复用性低。好处很多，省略很多代码的重写。

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？

提高代码的可读性，抽象出类的具体。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？

深有体会啊！难处大于乐趣

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？

是一种提高代码复用性的好方法。设计起来难，但功能强大。

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？

是一种隐秘的编程方式，能够提高隐私性。

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？

一开始一定要设计好思路和框架，基础很重要，后期重新修改太麻烦。

1. 关于本实验的工作量、难度、deadline。

工作量太大，测试很繁琐，量太多，占用太多时间。难度大。截止日期和计算机系统时间冲突较大，希望可以给更多的时间。

1. 到目前为止你对《软件构造》课程的评价。

课程难度较大，实验有许多不会的地方。但是提高了我的编程能力。是一门能学到真本事的学科。