

**2024年春季学期  
计算学部《软件构造》课程**

**Lab 3实验报告**

|  |  |
| --- | --- |
| 姓名 | 余蒙廷 |
| 学号 | 2023120255 |
| 班号 | 23E3711 |
| 电子邮件 | 2023120255@stu.hit.edu.cn |
| 手机号码 | 18375836033 |

**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> 1](#_Toc164601174)

[3.2.1 IntervalSet<L>的共性操作 1](#_Toc164601175)

[3.2.2 局部共性特征的设计方案 2](#_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） 2](#_Toc164601177)

[3.3 面向可复用性和可维护性的设计：MultiIntervalSet<L> 2](#_Toc164601178)

[3.3.1 MultiIntervalSet<L>的共性操作 2](#_Toc164601179)

[3.3.2 局部共性特征的设计方案 2](#_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） 2](#_Toc164601181)

[3.4 面向复用的设计：L 2](#_Toc164601182)

[3.5 可复用API设计 2](#_Toc164601183)

[3.5.1 计算相似度 2](#_Toc164601184)

[3.5.2 计算时间冲突比例 2](#_Toc164601185)

[3.5.3 计算空闲时间比例 2](#_Toc164601186)

[3.6 应用设计与开发 2](#_Toc164601187)

[3.6.1 排班管理系统 2](#_Toc164601188)

[3.6.2 操作系统的进程调度管理系统/课表管理系统 2](#_Toc164601189)

[3.7 基于语法的数据读入 3](#_Toc164601190)

[4 实验进度记录 3](#_Toc164601191)

[5 实验过程中遇到的困难与解决途径 3](#_Toc164601192)

[6 实验过程中收获的经验、教训、感想 3](#_Toc164601193)

[6.1 实验过程中收获的经验和教训 3](#_Toc164601194)

[6.2 针对以下方面的感受 3](#_Toc164601195)

# 实验目标概述

﻿本次实验覆盖课程第 4-12 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

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。

特别是要记录配置过程中遇到的问题和困难，以及如何解决的。

在这里给出你的GitHub Lab3仓库的URL地址（Lab3-学号）。

# 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

**1. 值班表管理系统**

- 功能：管理一个单位的值班安排。单位有多个员工，每天安排一个员工值班。记录包括员工姓名、职位、手机号码等信息。要求每个员工连续值班几天，不能有无人值班的情况。

日期 值班人名字 职位 手机号码

2024-01-10 孙XX 主任 13900000000

2024-01-11 孙XX 主任 13900000000

**2. 操作系统进程调度管理系统**

- 功能：管理单核CPU上多个进程的调度。每个时间点只有一个进程在运行，其他进程处于休眠状态。进程可以被挂起并在后续时刻恢复执行。调度无规律，可看作是随机调度。

时间段 进程ID 状态

0-5 P1 运行

5-10 P2 运行

**3. 大学课表管理系统**

- 功能：管理一个班级的课程安排。每周课程安排重复，包含课程ID、课程名称、教师名字、地点等信息。可以有空闲时间段。

- 示例：

星期 时间段 课程名称 教师名字 地点

周一 8:00-10:00 软件构造 张XX 正心楼209

周三 8:00-10:00 软件构造 张XX 正心楼209

**应用场景分析**

**共性**

1. 时间管理：三个应用都涉及时间段的管理。

2. 唯一性：在任何一个时间段内，只能有一个事件或任务在进行（值班、进程、课程）。

3. 标签：每个时间段都附带一个标签（员工、进程、课程），且标签不重复。

4. 可复用性：都可以通过抽象数据类型 (ADT) 和泛型技术来实现，提高代码的复用性和维护性。

**差异**

1. 空闲时间：

- 值班表管理系统不允许有空闲时间。

- 进程调度管理系统允许有空闲时间。

- 大学课表管理系统也允许有空闲时间。

2. 时间段重叠：

- 值班表管理系统和进程调度管理系统不允许时间段重叠。

- 大学课表管理系统允许时间段重叠。

3. 周期性：

- 值班表管理系统和进程调度管理系统没有周期性。

- 大学课表管理系统有周期性，课程安排按周重复。

**我对本次实验需求的理解**

共性需求

1. 时间段管理：需要一个可以管理不同时间段的统一数据结构。

2. 唯一性保证：需要保证同一时间段内的唯一性。

3. 标签管理：需要一个可以绑定不同标签的系统。

差异需求

1. 空闲时间的处理：需要支持是否允许空闲时间的选项。

2. 时间段重叠的处理：需要支持是否允许时间段重叠的选项。

3. 周期性时间段：需要支持是否有周期性时间段的选项。

所以可以得出设计一个通用的ADT的需求，并根据具体应用的需求进行扩展。这种设计可以提高软件的可复用性和可维护性。

## 面向可复用性和可维护性的设计：IntervalSet<L>

### IntervalSet<L>的共性操作



### 局部共性特征的设计方案

我采用的是方案3，方案3的特点是为每个维度上的不同特征取值分别定义不同的接口，并在具体实现类中实现这些接口的方法。

我为 IntervalSet<L> 和 MultiIntervalSet<L> 设计了接口，定义了共性操作。

通过 CommonIntervalSet<L> 类实现了 IntervalSet<L> 接口的方法。

**IntervalSet<L> 的局部共性特征**

在插入新时间段时检查是否与已有时间段冲突。

处理标签的添加、删除和查找。

**MultiIntervalSet<L> 的局部共性特征**

允许同一个标签对象绑定多个时间段。

返回某个标签所关联的所有时间段，并按开始时间排序。



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

#### 值班表管理系统 (Duty Roster Management System)

**个性化特征：**

- 不允许有空白时间段。

- 时间段不能重叠。

- 每个员工连续值班。

**DutyRoster 类**



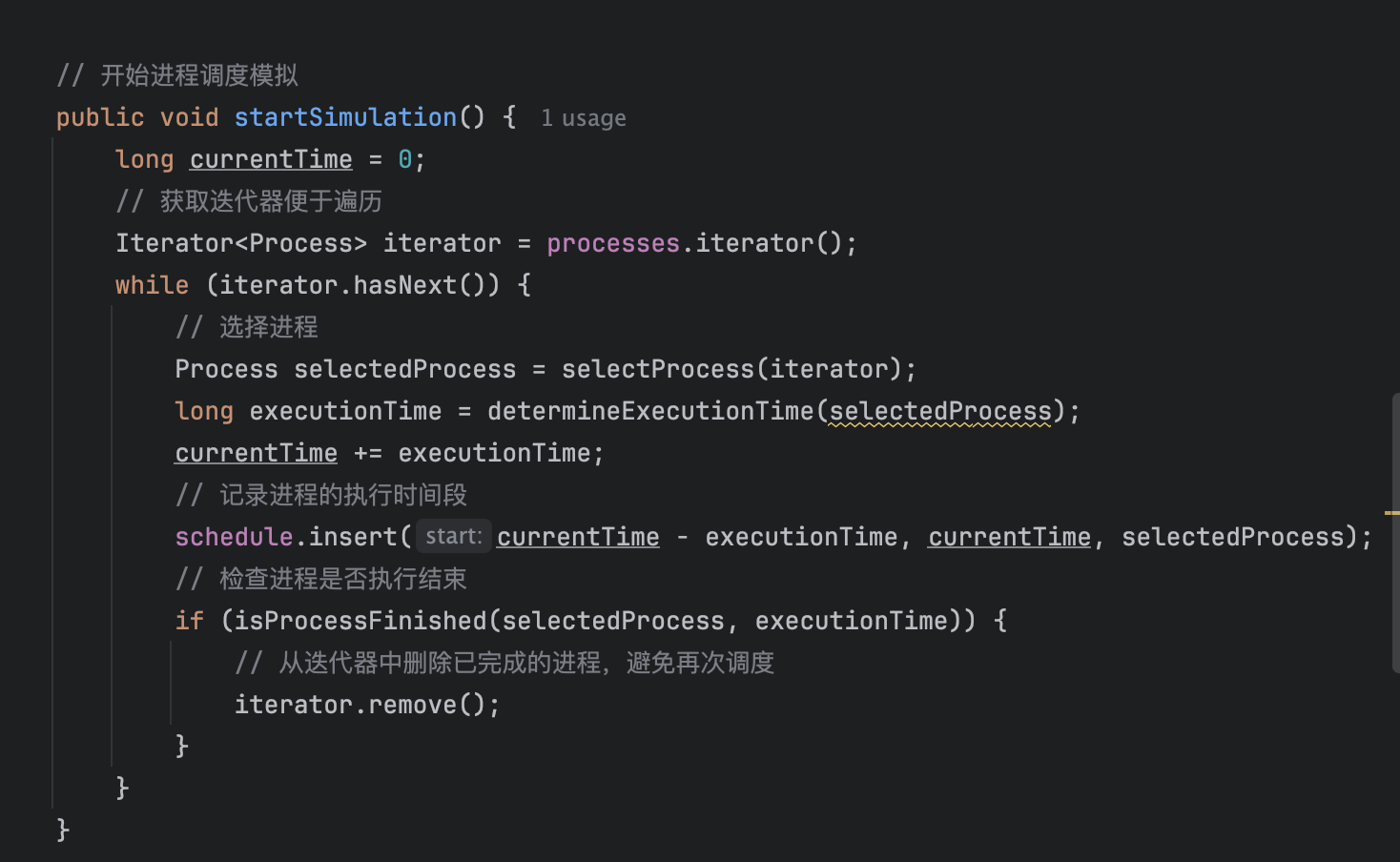
#### 操作系统进程调度管理系统

个性化特征：

- 允许有空闲时间段。

- 时间段不能重叠。

- 进程可以被挂起和恢复执行。



## 面向可复用性和可维护性的设计：MultiIntervalSet<L>

### MultiIntervalSet<L>的共性操作

|  |  |
| --- | --- |
| **方法声明** | **功能解释** |
| **static <L> MultiIntervalSet<L> from(IntervalSet<L> initial)** | 创建一个非空对象，使用**initial**中包含的数据创建非空对象。 |
| **void insert(long start, long end, L label)** | 在当前对象中插入新的时间段和标签。如果同一个标签对象L绑定多个时间段，则增加新的时间段。 |
| **boolean removeAll(L label)** | 从当前对象中移除某个标签所关联的所有时间段。 |
| **Set<Interval> 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。返回结果按开始时间从小到大排序。 |

### 局部共性特征的设计方案



**1. 插入新的时间段和标签**

方法声明: void insert(long start, long end, L label) throws IntervalException

功能解释: 在当前对象中插入一个新的时间段和标签。如果插入的时间段与已有时间段发生冲突，则抛出IntervalException。

**2. 获得当前对象中的标签集合**

方法声明: Set<L> labels()

功能解释: 返回当前对象中所有标签的集合。

**3. 从当前对象中移除某个标签所关联的时间段**

方法声明: boolean remove(L label)

功能解释: 从当前对象中移除某个标签及其关联的时间段。若移除成功，返回true，否则返回false。

**4. 返回某个标签对应的时间段的开始时间**

方法声明: long start(L label)

功能解释: 返回某个标签对应的时间段的开始时间。如果标签不存在，可以抛出一个自定义的异常

**5. 返回某个标签对应的时间段的结束时间**

方法声明: long end(L label)

功能解释: 返回某个标签对应的时间段的结束时间。如果标签不存在，可以抛出一个自定义的异常

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

1. 值班表管理系统

DutyRoster.java

|  |  |
| --- | --- |
| **方法签名** | **功能解释** |
| **public void addDuty(long start, long end, Employee employee) throws IntervalException** | 添加值班记录。检查新记录是否与已有记录时间段重叠，如果重叠则抛出异常。 |

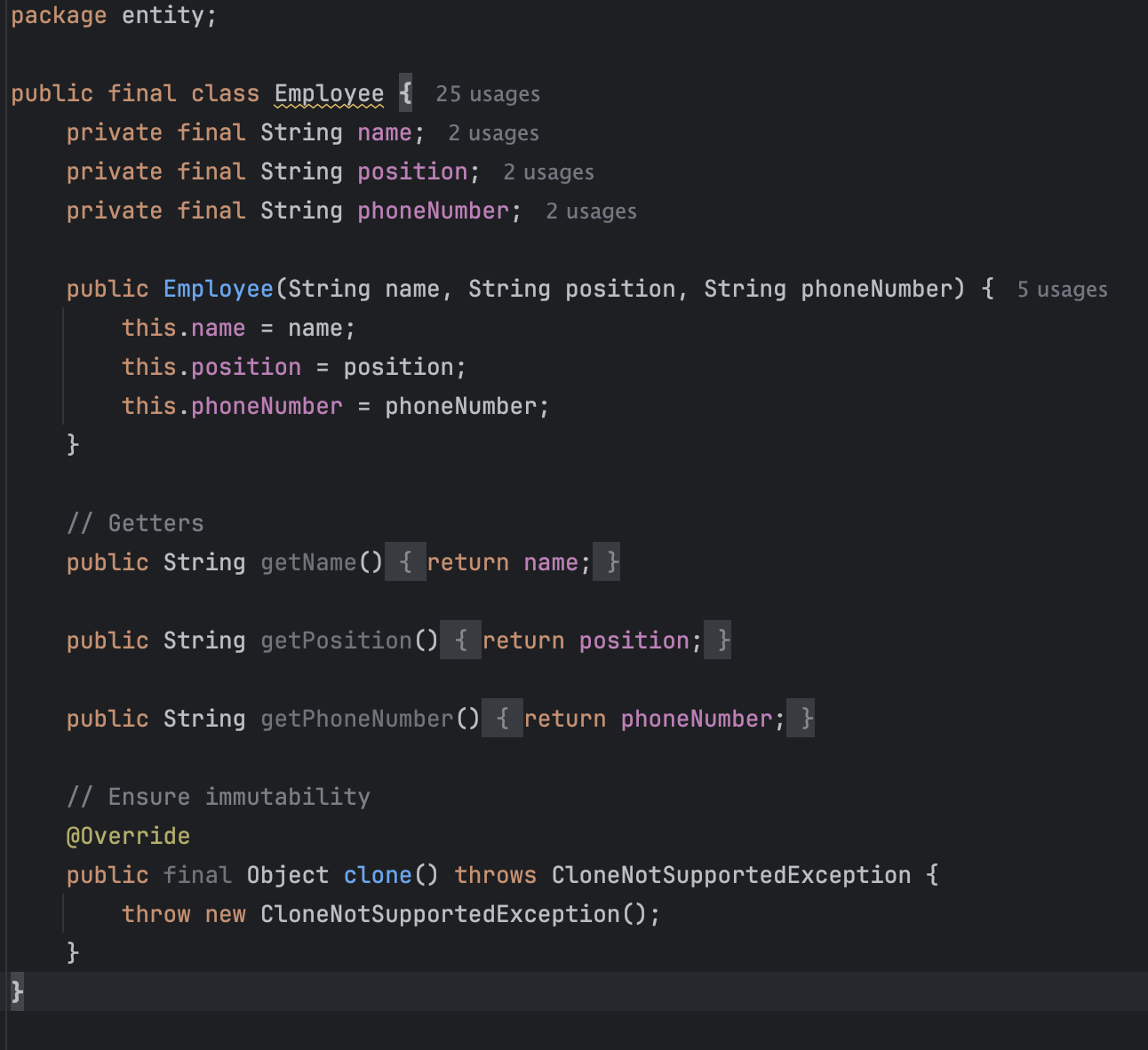
2. 操作系统进程调度管理系统

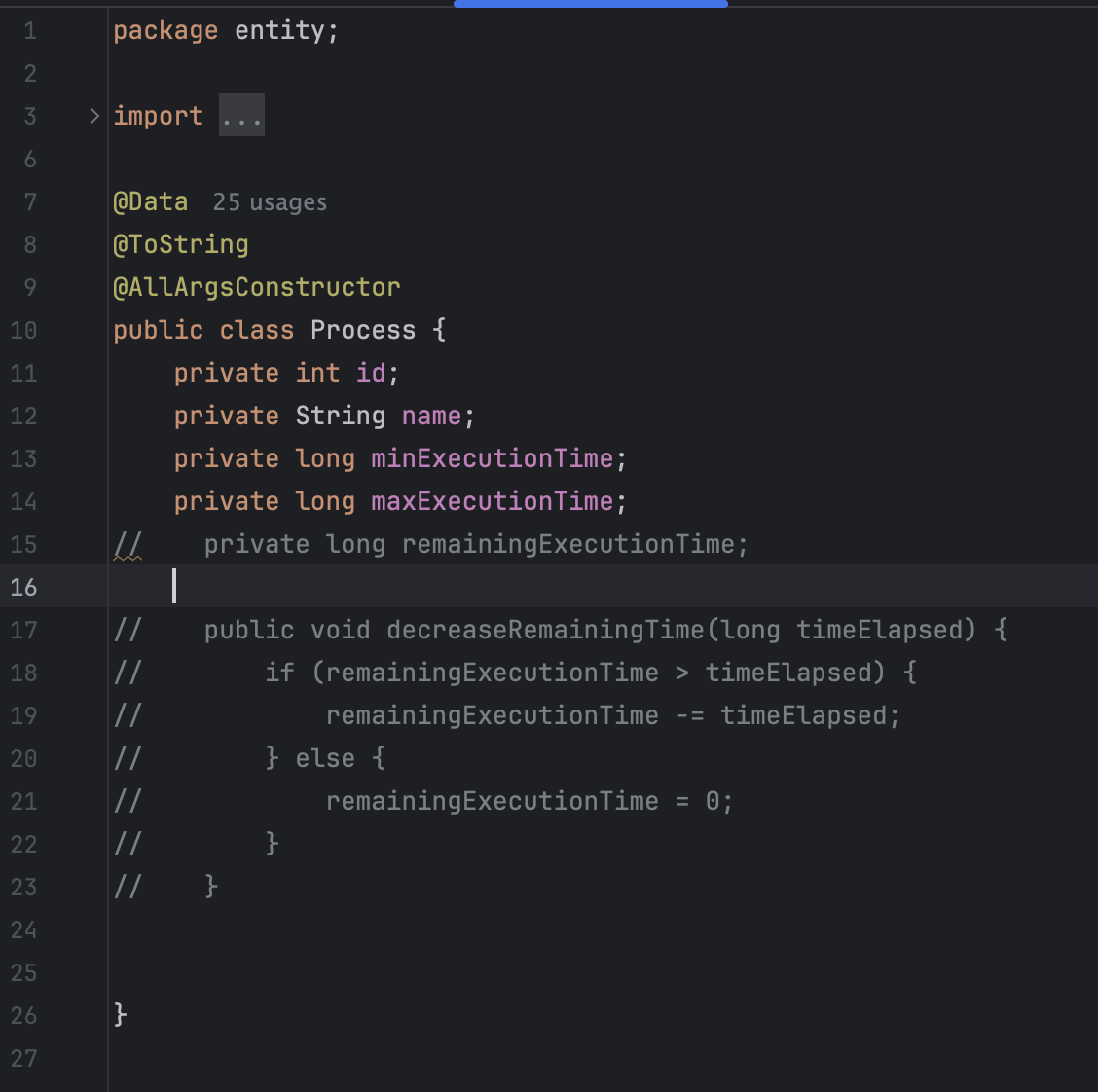
源文件名称: ProcessScheduleManager.java

|  |  |
| --- | --- |
| **方法签名** | **功能解释** |
| **public void addProcess(Process process)** | 添加一个新的进程到进程调度管理系统中。 |
| **public void scheduleProcess(long start, long end, Process process) throws IntervalException** | 为进程安排调度时间段。检查新调度时间段是否与已有进程调度时间段重叠，如果重叠则抛出异常。 |
| **public void startSimulation()** | 启动进程调度模拟。通过随机选择进程或其他策略进行进程调度。 |

## 面向复用的设计：L

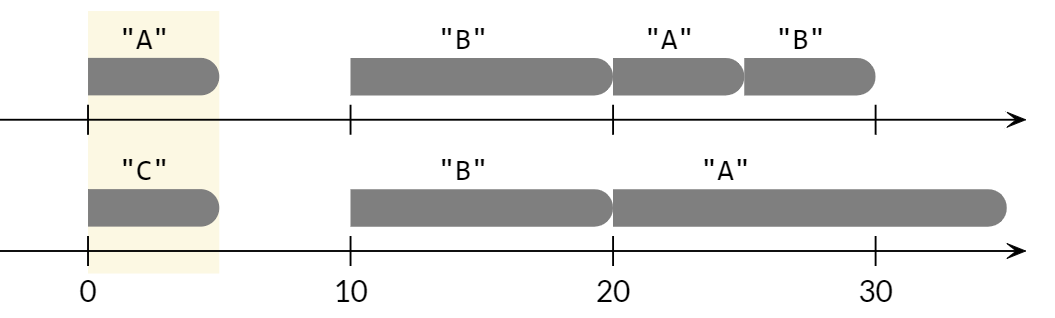
## 可复用API设计





### 计算相似度

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。



针对上图中的两个对象：

{ A = [[0,5),[20,25)], B = [[10,20),[25,30)] }

{ A = [[20,35)], B = [[10,20)], C = [[0,5)] }

它们的相似度计算如下：

( 5×0 + 5×0 + 10×1 + 5×1 + 5×0 + 5×0 ) ÷ ( 35 - 0 ) = 15 / 35 ≈ 0.42857

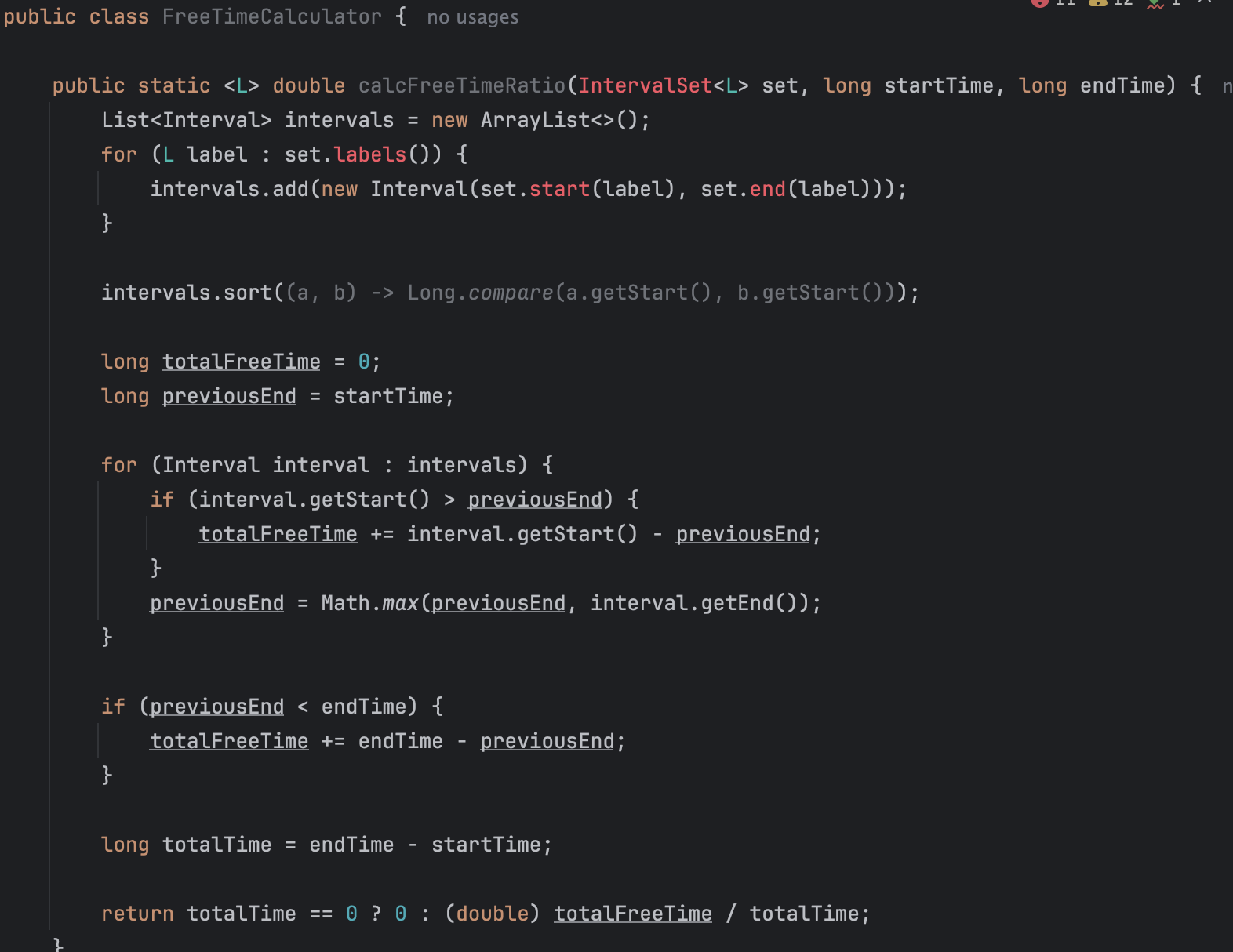
### 计算时间冲突比例

计算给定MultiIntervalSet对象中的时间冲突比例。遍历所有标签和它们的时间段，计算重叠部分，并返回冲突时间段长度与总时间段长度的比例。



### 计算空闲时间比例

计算给定IntervalSet对象中的空闲时间比例。遍历所有标签和它们的时间段，计算空闲部分，并返回空闲时间段长度与总时间段长度的比例。

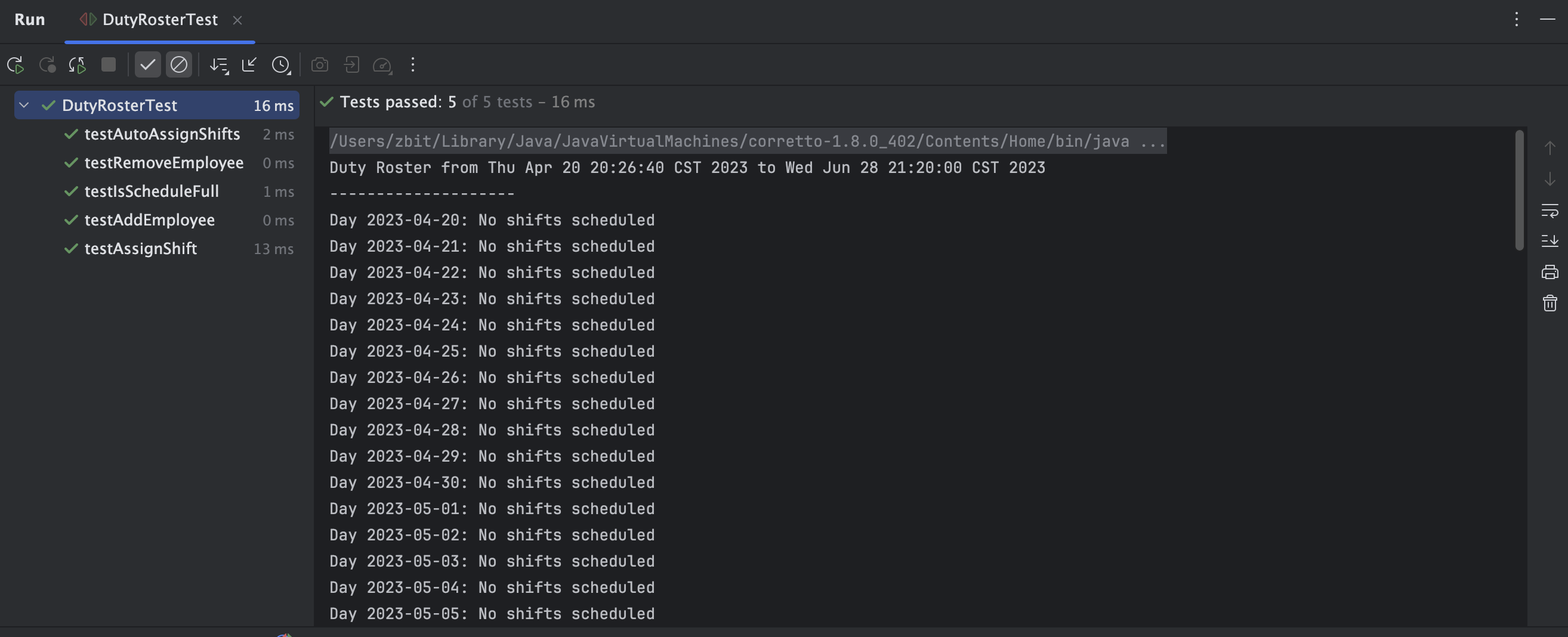


## 应用设计与开发

利用上述设计和实现的ADT，实现手册里要求的各项功能。

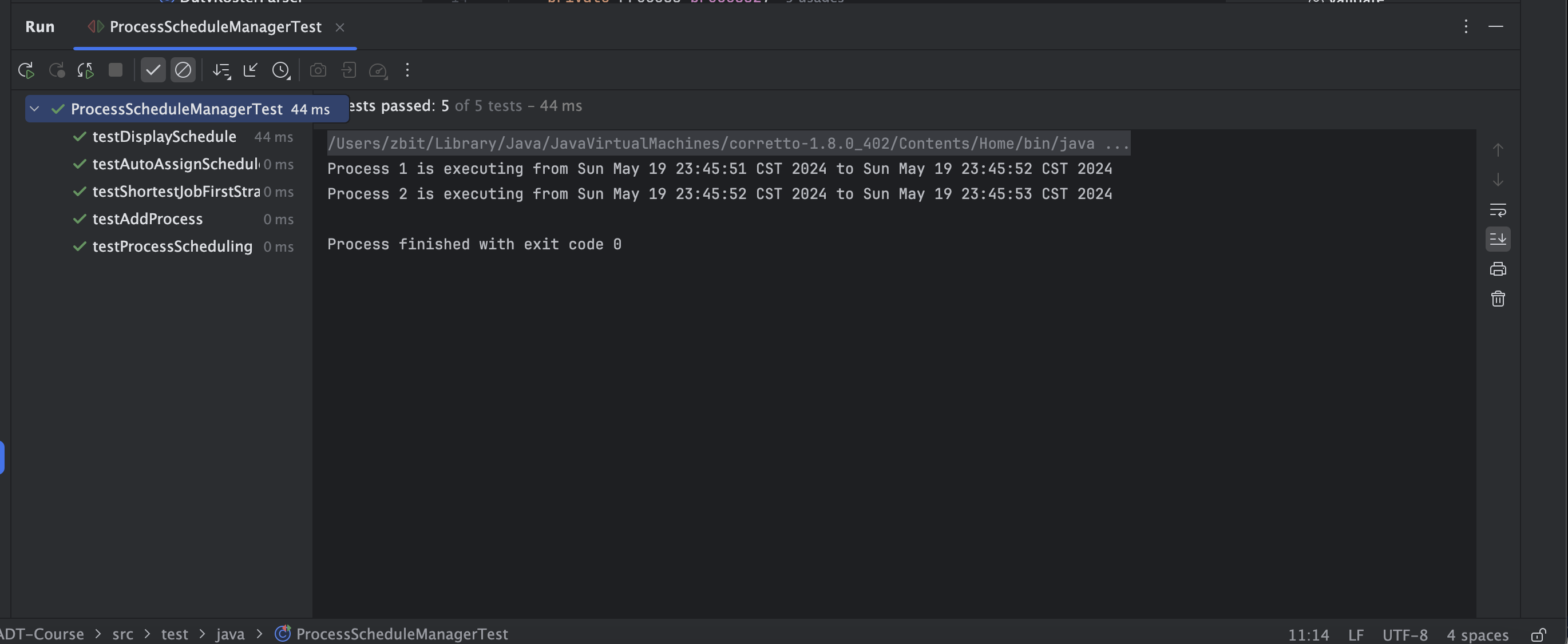
### 排班管理系统

|  |  |  |
| --- | --- | --- |
| **功能需求** | **方法签名** | **功能解释** |
| **设定排班开始日期和结束日期** | 构造函数 | **public DutyRoster(List<Employee> employees)** <br> 初始化一个空的排班表，并初始化员工列表。 |
| **增加员工** | **void addEmployee(Employee employee)** | 增加一个新的员工到员工列表中。 |
| **手工增加排班记录** | **void addDuty(long start, long end, Employee employee) throws IntervalException** | 检查新添加的时间段是否与已有时间段重叠，如果没有重叠，则将新的时间段插入排班表中。 |
| **检查排班是否已满** | **boolean isFull(long startTime, long endTime)** | 检查从开始时间到结束时间的所有时间段是否都被安排了员工。 |
| **自动生成排班表** | **void autoSchedule(long start, long end, List<Employee> employees)** | 自动为每个时间段分配员工，确保所有时间段都被安排。 <br> 每次值班一天，轮流分配员工。 |
| **插入新的时间段和标签** | **void insert(long start, long end, L label) throws IntervalException** | 在当前对象中插入一个新的时间段和标签。允许同一个标签对象绑定多个时间段，实现了时间段的灵活管理。 |
| **获得当前对象中的标签集合** | **Set<L> labels()** | 返回当前对象中所有标签的集合。通过这个方法，可以获取到所有的标签 |
| **从当前对象中移除某个标签所关联的时间段** | **boolean remove(L label)** | 从当前对象中移除某个标签及其关联的时间段。若移除成功，返回**true**，否则返回**false**。 |
| **返回某个标签对应的时间段的开始时间** | **long start(L label)** | 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** | **long end(L label)** | 返回某个标签对应的时间段的结束时间。 |



### 操作系统的进程调度管理系统/课表管理系统

|  |  |  |
| --- | --- | --- |
| **功能需求** | **方法签名** | **功能解释** |
| **增加进程** | **public void addProcess(Process process)** | 增加一个新的进程到进程列表中。 |
| **手工增加进程调度记录** | **public void scheduleProcess(long start, long end, Process process) throws IntervalException** | 检查新添加的时间段是否与已有时间段重叠，如果没有重叠，则将新的时间段插入进程调度表中。 |
| **随机选择进程** | **private Process selectProcess(Iterator<Process> iterator)** | 随机选择一个尚未执行结束的进程进行调度。 |
| **确定进程执行时间** | **private long determineExecutionTime(Process process)** | 随机确定进程的执行时间。 |
| **检查进程是否执行结束** | **private boolean isProcessFinished(Process process, long executionTime)** | 检查进程是否执行完毕，如果执行时间达到最大执行时间，则认为进程已完成。 |
| **开始进程调度模拟** | **public void startSimulation()** | 启动进程调度模拟，按设定的策略（随机或最短进程优先）选择进程并进行调度。 |
| **检查进程是否已调度** | **public boolean isProcessScheduled(Process process)** | 检查进程是否已经被调度。 |



## 基于语法的数据读入

创建3个函数分别读取employee、period、roster。用ArrayList来存储每行读取到的字符串，然后创建文件流开始按行读取字符串，按照text中数据的特征进行分割。将“{”符号之前的字符串存在第一个数组中表示姓名，将“{”到“，”之间的字符串存在第二个数组中表示职业，将“,”到“}”的字符串存在第三个数组中表示电话号，这样得到了employee类的3个属性。同理对period和Roster进行解析。

# 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 |
| 5.16 | 8：00–23：00 | 看题，自顶而下设计代码 | 完成 |
| 5.17 | 8：00-23：00 | 向可复用性和可维护性的设计 | 完成 |
| 5.18 | 8：00-23：00 | 值班表管理系统 | 完成 |
| 5.19 | 8：00-23：00 | 值班表管理系统 | 完成 |
| 5.19 | 8：00-23：00 | 操作系统进程调度管理系统 | 完成 |
| 5.19 | 8：00-23：00 | 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 |
|  |  |
|  |  |
|  |  |

# 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

## 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。

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？

要是时间在充裕一点就好了

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？

一种隐秘的编程方式。提高私密性。

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？

开始的抽象化和调用关系。以及团队合作。

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？

最开始设计好思路和框架，后期一改则动全身。

1. 关于本实验的工作量、难度、deadline。

工作量太大，难度太高，deadline太紧凑

1. 到目前为止你对《软件构造》课程的评价。

一门很好的课，可惜压缩了学时，可惜跟计算机系统一个学期，很忙。