



哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2021 年春季学期 计算学部《软件构造》课程

Lab 3 实验报告

姓名	黄子扬
学号	1190401018
班号	1936603
电子邮件	1953315558@qq.com
手机号码	15686168912

目录

2 实验环境配置

3 实验过程

3.1 待开发的三个应用场景

3.2 面向可复用性和可维护性的设计：IntervalSet<L>

3.2.1 IntervalSet<L>的共性操作

3.2.2 局部共性特征的设计方案

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）

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

3.3.1 MultiIntervalSet<L>的共性操作

3.3.2 局部共性特征的设计方案

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）

3.4 面向复用的设计：L

3.5 可复用 API 设计

3.5.1 计算相似度

3.5.2 计算时间冲突比例

3.5.3 计算空闲时间比例

3.6 应用设计与开发

3.6.1 排班管理系统

3.6.2 操作系统的进程调度管理系统

3.6.3 课表管理系统

3.7 基于语法的数据读入

3.8 应对面临的新变化

3.8.1 变化 1

3.8.2 变化 2

3.9 Git 仓库结构

4 实验进度记录

5 实验过程中遇到的困难与解决途径

6 实验过程中收获的经验、教训、感想

6.1 实验过程中收获的经验教训

6.2 针对以下方面的感受

1 实验目标概述

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））。

2 实验环境配置

创建 maven 项目，配置 maven 文件，在依赖中添加 junit。

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4   <groupId>HIT-Lab3-1190401018</groupId>
5   <artifactId>HIT-Lab3-1190401018</artifactId>
6   <version>0.0.1-SNAPSHOT</version>
7   <build>
8     <sourceDirectory>src</sourceDirectory>
9     <testSourceDirectory>test</testSourceDirectory>
10    <resources>
11      <resource>
12        <directory>lib</directory>
13        <excludes>
14          <exclude>**.java</exclude>
15        </excludes>
16      </resource>
17    </resources>
18    <plugins>
19      <plugin>
20        <artifactId>maven-compiler-plugin</artifactId>
21        <version>3.8.1</version>
22        <configuration>
23          <source>1.8</source>
24          <target>1.8</target>
25        </configuration>
26      </plugin>
27    </plugins>
28  </build>
29  <dependencies>
30    <!-- https://mvnrepository.com/artifact/junit/junit -->
31    <dependency>
32      <groupId>junit</groupId>
33      <artifactId>junit</artifactId>
34      <version>4.13.2</version>
35      <scope>test</scope>
36    </dependency>
37  </dependencies>
38  <properties>
39    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
40  </properties>
41</project>
```

创建 src、lib、doc、test 目录。

创建 github 仓库。

<https://github.com/ComputerScienceHIT/HIT-Lab3-1190401018.git>

```
git remote add origin https://github.com/ComputerScienceHIT/HIT-Lab3-1190401018.git 链接远程仓库。
```

```
git add * git commit -m "init the repo" git push origin master
```

将初始化好的工程 push 到 github 上。

在这里给出你的 GitHub Lab3 仓库的 URL 地址（HIT-Lab3-学号）。
<https://github.com/ComputerScienceHIT/HIT-Lab3-1190401018.git>

3 实验过程

3.1 待开发的三个应用场景

三个场景：

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。

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。

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”）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。

共同点：

1. 每个场景中都有一个时间轴，在这个时间轴上都有一段一段的时间间隔 (interval)，每个间隔作为一个元素。

2. 我们可以将其这三个场景抽象为一个时间段集合。值班表即是多个值班段的集合, 进程调度管理即使多个进程执行时间段的集合, 大学课表即使多个课程的集合。

3. 每个时间段都有对应的一个标签 (label) 作为标记, 来标识其信息。

4. 每个标签都是一个不可变的对象, 在这三个场景中分别是 Employee、Process、Course。

差异:

1. 有的场景中多个时间段可以被同一个标签标记, 有的场景每一个时间段都有一个的标签。如在排班表中, 每个值班周期都必须人不一样, 即标签不同, 而在课程表中, 一个课可能在一周的不同时间内上多次, 所以其有多个时间段可能对应的是同一个标签, 即同一门课。

2. 有的场景中的时间轴上不能有空白的时间段, 也就是说不能有一段时间段, 其没有标签对应。比如排班表中, 从开始值班到结束值班, 每一天都必须有人值班, 不能有间隙无人值班。

3. 有的场景中时间轴上的时间段是不能有重叠的, 即两个时间段之间不能有公共的部分, 如[2, 4]和[3, 5]就在[3, 4]部分重叠了。在值班表中, 是不能有重叠的, 即每一个值班期之间是没有重叠时间的。

4. 有的场景中时间轴上的时间段是重复出现的, 比如在课程表中, 每一门课程都是重复出现的, 在每周固定的时间上课。

开发需求:

1. 引入 IntervalSet, 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。

2. 引入 MultiIntervalSet, 这也是一个 mutable 的 ADT, 弥补了 IntervalSet 中标签不能重复的弱点, 利用 IntervalSet, 使得 MultiIntervalSet 具有使多个时间段对应同一标签的能力。


3.2 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。


3.2.1 IntervalSet<L>的共性操作


IntervalSet<L>


intervalset


 empty(): IntervalSet<L>


 insert(start: long, end: long, label: L): boolean

 labels(): Set<L>

 remove(label: L): boolean

 start(label: L): long

 end(label: L): long

 span(): long

针对需求，IntervalSet 的设计如上。

其中，接口中定义了一下几种操作。

1.insert: 插入一个时间段到集合中

```
/**
 * insert an interval in the set, start < end
 *
 * @param start the start time of the interval( >= 0 )
 * @param end   the end time of the interval( >= 0 )
 * @param label the label of this interval, can't be null
 * @return true if insertion succeed, otherwise false
 * @throws if start or the end or the label is illegal
 */
boolean insert(long start, long end, L label) throws Exception;
```

2.labels: 得到集合中所有时间段的标签

```
/**
 * get all the labels in the set
 *
 * @return a set that includes all the labels in the IntervalSet, if the
 *         IntervalSet is empty, then return an empty set
 */
Set<L> labels();
```

3.remove: 根据标签删除对应的时间段

```

/**
 * remove an interval in this IntervalSet
 *
 * @param label the label of one interval
 * @return true if the removal succeed, otherwise false
 */
boolean remove(L label);

```

4.start: 根据标签得到该时间段的开始时间

```

/**
 * get the start time of one interval
 *
 * @param label an interval label
 * @return the start time of the interval if the interval is in the set,
 *         otherwise -1
 */
long start(L label);

```

5.end: 根据标签得到该时间段的结束时间

```

/**
 * get the end time of one interval
 *
 * @param label an interval label
 * @return the end time of this interval if the interval is in the set,
 *         otherwise -1
 */
long end(L label);

```

6.span: 用时间段集合中的最大结束时间减去最小开始时间

```

/**
 * calculate the span of timeline
 * @return the max time - the min time
 */
long span();

```

7.empty: 静态工厂方法，返回一个 CommonIntervalSet 实例。

接口的设计遵循了 DIP 和 OCP，在接口中只定义了一组方法的 spec 来描述这个 ADT，接着我们实现了一个其子类，给出其中的一种实现。

实现子类为 CommonIntervalSet:

CommonIntervalSet<L>

intervalset

■ intervals: List<interval>

■ checkRep(): void

● insert(start: long, end: long, label: L): boolean

● labels(): Set<L>

● remove(label: L): boolean

● start(label: L): long

● end(label: L): long

● equals(obj: Object): boolean

● span(): long

Rep: 用一个 `interval` 对象的列表来存储整个时间段集合。由于每一个时间段都具有三个属性，即开始时间、结束时间、标签，为了方便统一管理，我们将其实现为一个内部类，并命名为 `interval`。

```
// rep
private List<interval> intervals = new ArrayList<>();
```

AF: 代表一个时间段集合，每一个在 `intervals` 中的对象都是一个时间段

```
// AF:
// represents a timeline, each entry in the intervals represent an interval
// from start to end, labeled with label
```

RI: 每一个时间段都有唯一的标签，且开始时间必须小于结束时间

```
// RI:
// two intervals can't have the same label
// for any interval in the set, start < end
```

Safety from rep-exposure: `rep` 是 `private` 的，而且所有的方法都无法从外界得到 `rep` 的引用。

```
// safety from rep exposure:
// intervals is private, and we don't offer alias of intervals to the
// client
```

checkRep: 遍历 `intervals`，根据 **RI** 检查即可。

insert: 首先检查前置条件是否满足，不满足抛出异常 `IntervalParamsWrongException`，这是一个自定义的异常，专门用于标识输入时参数有问题。无异常后直接插入即可。

labels: 遍历 `intervals`，得到每一个 `interval` 的 `label` 属性即可。

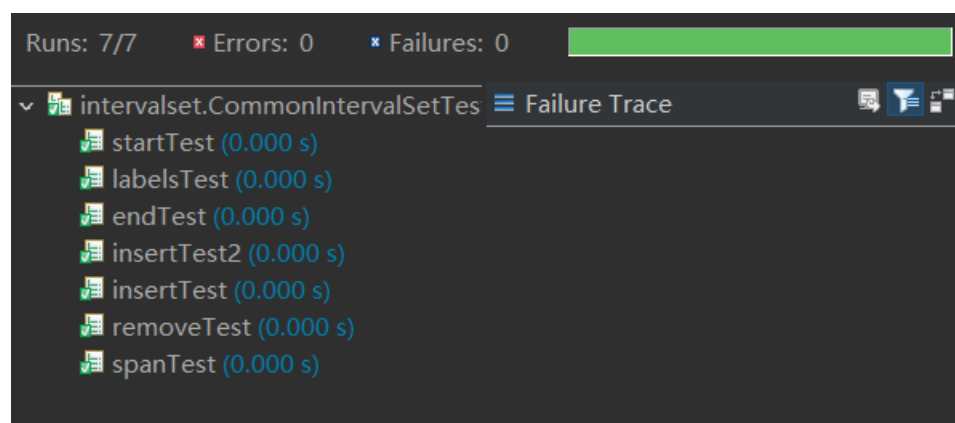
remove: 首先检查是否存在输入的 `label` 标记的 `interval`，如果没有返回 `false`，否则直接将对应的 `interval` 删除，返回 `true` 即可。

start/end: 遍历，如果存在标签对应的 `interval`，就返回其开始时间或是结束时间。

span: 遍历，得到最小的开始时间和最大的结束时间，最后相减。

Testing strategy:

```
// insert:  
// an interval not in the set  
// an interval already in the set  
// an interval already overlap with some intervals already in the set  
// start >= end  
// parameters are not illegal  
  
// labels:  
// empty IntervalSet  
// IntervalSet with some intervals already in  
  
// remove:  
// an interval not in the set  
// an interval already in the set  
  
// start:  
// an interval not in the set  
// an interval already in the set  
  
// end:  
// an interval not in the set  
// an interval already in the set  
  
// span:  
// no overlapped timeline  
// overlapped timeline
```



3.2.2 局部共性特征的设计方案

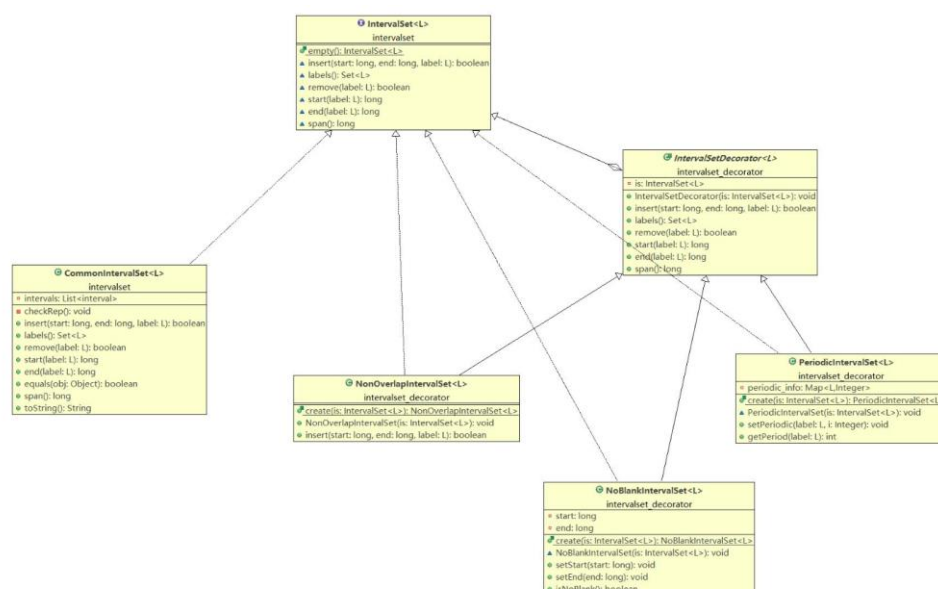
在 3.2.1 节的设计中，对时间段的集合的限制很少，只要标签不重复且时间段合理都可以插入。故其是可重叠、可有空白、无重复标记的时间段集合。接下来，我们要根据这三个不同维度的特征，来设计更加具体的类，使其具有一些更加特殊的功能。

设计方案：使用装饰器（decorator）模式，装饰器模式能够更加灵活的组合不同维度的特征。如果使用单纯的继承的话，就会使得继承树变得非常之复杂，且复用性不好，针对每一个最特殊的子类，需要实现的功能很复杂。装饰器的好处就是使得不同的特征分开实现，但是又使得需要不同功能的组合十分方便，简化了继承树的设计。

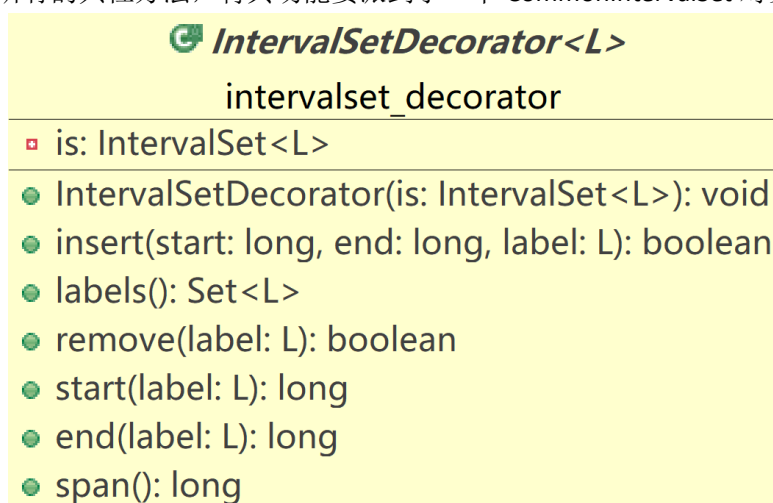
三个需要特殊实现的特征分别是：

- 1.不可重叠
- 2.不可有空白
- 3.时间段可以以特定的周期重复

继承树如下：



其中，IntervalSetDecorator 是一个抽象类，实现了 IntervalSet 接口，并通过委派的方式先实现了所有的共性方法，将其功能委派到了在一个 CommonIntervalSet 对象上。



```

private IntervalSet<L> is = IntervalSet.empty();

public IntervalSetDecorator(IntervalSet<L> is) {
    this.is = is;
}

@Override
public boolean insert(long start, long end, L label) throws Exception {
    return is.insert(start, end, label);
}

@Override
public Set<L> labels() {
    return is.labels();
}

@Override
public boolean remove(L label) {
    return is.remove(label);
}

@Override
public long start(L label) {









```

每个方法的实现都是通过委派到 `IntervalSet` 的实例对象实现的。

接着，我们从这个类延展出三个特性的子类。

这几个类在测试时，由于其共性方法通过委派实现的，而委派的对象的方法已充分测试，故在测试具体子类时只需针对其特性进行测试。

NoBlankIntervalSet:

 NoBlankIntervalSet<L> intervalset_decorator	
	start: long
	end: long
	create(is: IntervalSet<L>): NoBlankIntervalSet<L>
	NoBlankIntervalSet(is: IntervalSet<L>): void
	setStart(start: long): void
	setEnd(end: long): void
	isNoBlank(): boolean

在这个特征子类中，主要实现了一种特殊的个性方法，`isNoBlank` 来判断，当所有的 `intervals` 都插入后，是否还有空白。

Rep、AF、RI、safe from rep exposure:

```
// rep
private long start; // the start time of the timeline
private long end; // the end time of the timeline

// AF
// represents a IntervalSet with no blank intervals

// RI
// start < end
// when all the intervals have been added, the isNoBlank() should return true

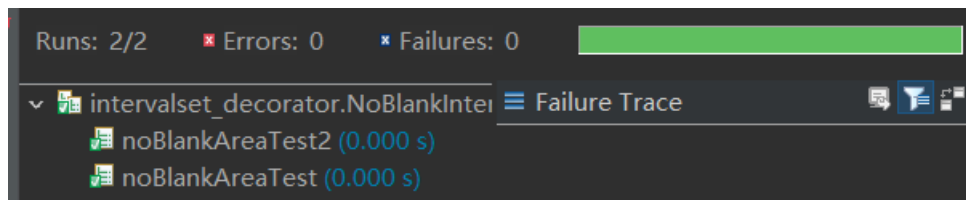
// safety from rep exposure:
// the rep is private and mutable
```

在这 ADT 中增加了 `start` 和 `end` 两个属性。因为我们如果不知道开始时间和结束时间是什么的话，是无法定义空白时间这个概念的，所以要增加定义这两个属性。

而 `isNoBlank` 就是根据这两个属性进行检测是否有空白时间段的。

Testing strategy:

```
// Testing Strategy:
// have no blank area
// have no blank area but have overlapped area
// have blank area
```



NonOverlapIntervalSet:

NonOverlapIntervalSet<L>

intervalset_decorator

`create(is: IntervalSet<L>): NonOverlapIntervalSet<L>`

`NonOverlapIntervalSet(is: IntervalSet<L>): void`

`insert(start: long, end: long, label: L): boolean`

这个类在实现时不需要增加 `rep`，其可能出现的标识泄露问题已经在其父类中所代理的对象中保证了。

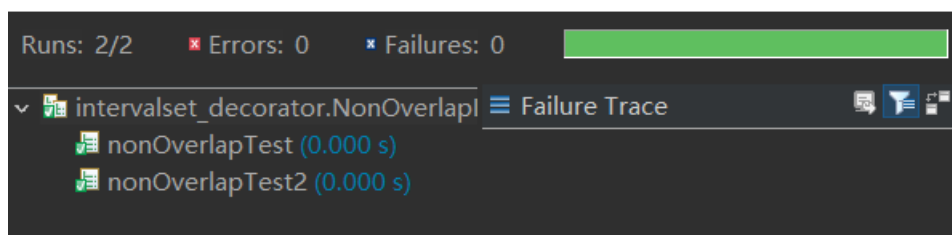
```
// AF:
// represents a intervalset with no intervals overlapped

//RI:
// each interval should not overlapped with other intervals

// safety from rep exposure:
// the rep of super class guarantees the client can't get the reference of the rep
```

Testing strategy:

```
// TestingStrategy:
// have overlapped area
// have no overlapped area but have blank area
// have no overlapped area and have no blank area
```



PeriodicIntervalSet:

PeriodicIntervalSet<L>

intervalset_decorator

periodic_info: Map<L,Integer>

create(is: IntervalSet<L>): PeriodicIntervalSet<L>

PeriodicIntervalSet(is: IntervalSet<L>): void

setPeriodic(label: L, i: Integer): boolean

getPeriod(label: L): int

Rep、AF、RI、 safe from rep exposure:

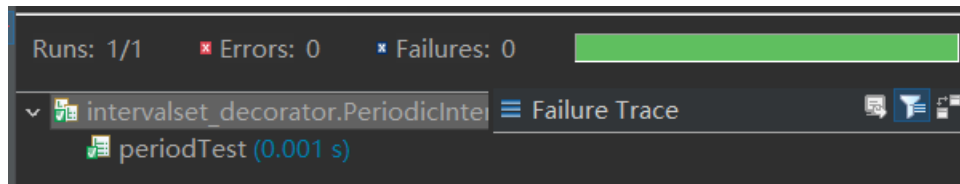
```
// AF:
// represents an intervalset which intervals can repeat in a period

// RI:
// the period of every interval >= 0
// the period of the repeated intervals >0

// safety from rep exposure:
// periodic_info is private
```

Testing strategy:

```
// Testing strategy:
// set a period for a non-existent interval
// set a period for a existent interval
// get a period of a non-existent interval
// set a period of a existent interval
```



针对以上三个子类，均实现了静态工厂方法，在创建对象时不用 `new` 关键字，而是使用 `xxx.create()` 方法来创建对象。

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）

由于我采用的是装饰器的方案，故不再需要除以上那三个具体子类的其他子类。不像接口组合方案，还需要针对组合后的不同方案来实现更加具体的子类。

装饰器模式完全的了分离各个维度的特征，等到创建具体对象时，直接使用组合不同特征即可。

排班表中需要创建一个时间段集合，是无重叠、无空白的，且需要调用 `NoBlankIntervalSet` 中的特殊方法（`isNoBlank()`），故在其 `spec` 中直接声明一个 `NoBlankIntervalSet`，然后通过嵌套的方式来创建即可。

```
private NoBlankIntervalSet<Employee> calendar = NoBlankIntervalSet
    .create(NonOverlapIntervalSet.create(IntervalSet.empty()));
```

这就是装饰器的好处，不需要通过接口组合再来实现一个具体类，而是在需要用到的时候就可以直接创建对象。

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

3.3.1 `MultiIntervalSet<L>` 的共性操作

1 MultiIntervalSet<L>

multiintervalset

- 🟢 empty(): MultiIntervalSet<L>
- 🟢 create_from_intervalset(is: IntervalSet<L>): MultiIntervalSet<L>
- 🟢 insert(start: long, end: long, label: L): boolean
- 🟢 labels(): Set<L>
- 🟢 remove(label: L): boolean
- 🟢 intervals(label: L): IntervalSet<Integer>
- 🟢 span(): long

1.insert:插入一个时间段到集合中去，可以一个标签对应多个时间段

```
/**
 * this method insert an interval into the timeline, start < end
 *
 * @param start the start time of the interval ( >= 0 )
 * @param end   the end time of the interval ( >= 0 )
 * @param label the label of a interval, can't be null
 * @return true if insertion succeed, otherwise false
 * @throws if the start or the end is illegal
 */
boolean insert(long start, long end, L label) throws Exception;
```

2.labels: 得到集合中所有时间段的标签

```
/**
 * get all the labels in the set
 *
 * @return the set of labels
 */
Set<L> labels();
```

3.remove: 根据标签删除掉集合中的某时间段

```
/**
 * remove an interval in the set
 *
 * @param label the label of the interval
 * @return true if the removal succeed, otherwise false
 */
boolean remove(L label);
```

4.intervals: 得到某个标签对应的所有子时间段的集合

```

/**
 * get the sub-intervals set of one interval labeled with label in ascending
 * order according to the start time of every sub-intervals
 *
 * @param label the label of the interval
 * @return an IntervalSet<Integer> includes all the sub-intervals of the input
 *         interval if the input interval doesn't exist in the timeline, return
 *         an empty IntervalSet
 * @throws Exception if the construction of intervals is illegal
 */
IntervalSet<Integer> intervals(L label);

```










5.span: 用所有时间段中最大的结束时间减去最小的开始时间

```

/**
 * calculate the span of timeline
 * @return the max time - the min time
 */
long span();

```

接口设计完成后，开始进行实现，实现子类为 CommonMultiIntervalSet:

 CommonMultiIntervalSet<L> multiintervalset
 mis: Map<L, IntervalSet<Integer>>
 CommonMultiIntervalSet(): void  CommonMultiIntervalSet(initial: IntervalSet<L>): void  insert(start: long, end: long, label: L): boolean  labels(): Set<L>  remove(label: L): boolean  intervals(label: L): IntervalSet<Integer>  span(): long

Rep、AF、RI、safety from rep exposure:

```

// rep
private final Map<L, IntervalSet<Integer>> mis = new HashMap<>();

// AF:
// represents MultiIntervalSet, which means more
// than one intervals can linked to one label

// RI:
// for any sub-interval in the set, start < end

// safety from rep exposure:
// the mis is private and final, and the client can't get the reference of the
// mis

```


具体到每个方法的实现，和 `IntervalSet` 中的类似，只不过是去遍历每个 `label` 所代表的 `IntervalSet` 而已。

Testing strategy:

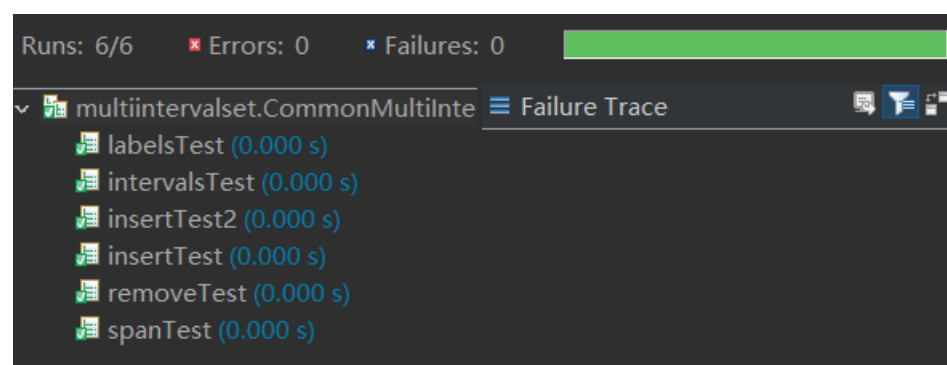
```
// Testing strategy:

// insert:
// an interval not in the set
// an interval already in the set
// an interval already overlap with some intervals already in the set
// start >= end
// parameters are illegal

// labels:
// empty IntervalSet
// IntervalSet with some intervals already in
// some labels marked more than one interval

// remove:
// an interval not in the set
// an interval already in the set
// an interval composed with more than one sub-intervals

// intervals:
// an interval not int the set
// an interval already in the set, but only has one sub-interval
// an interval in the set, ans has more than one sub-intervals in the set
```



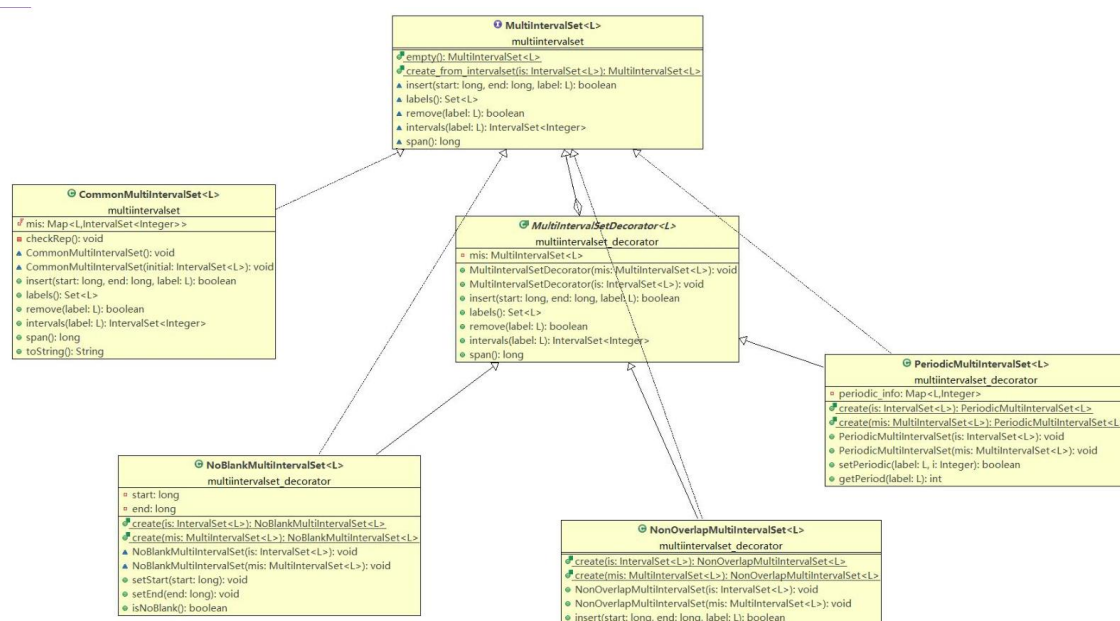
3.3.2 局部共性特征的设计方案

类似于 `IntervalSet` 的设计，采用装饰器方案，对三个维度不同特征进行子类设计，通过委派实现共性方法，在子类中可以实现其他个性方法。

三个特征分别是：没有空白、没有重叠、能够重复。

分别实现为 `NoBlankMultiIntervalSet`、`NonOverlapMultiIntervalSet`、`PeriodicMultiIntervalSet`。

继承树如下：



`NoBlankMultiIntervalSet`:

NoBlankMultiIntervalSet<L> multiintervalset_decorator
<div>start: long</div> <div>end: long</div>
<div>create(is: IntervalSet<L>): NoBlankMultiIntervalSet<L></div> <div>create(mis: MultiIntervalSet<L>): NoBlankMultiIntervalSet<L></div> <div>NoBlankMultiIntervalSet(is: IntervalSet<L>): void</div> <div>NoBlankMultiIntervalSet(mis: MultiIntervalSet<L>): void</div> <div>setStart(start: long): void</div> <div>setEnd(end: long): void</div> <div>isNoBlank(): boolean</div>

在这个特征子类中，主要实现了一种特殊的个性方法，`isNoBlank` 来判断，当所有的 `intervals` 都插入后，是否还有空白。

Rep、AF、RI、safety from rep exposure:

```
// rep
private long start;
private long end;

// AF:
// represents a MultiIntervalSet with no blank intervals

// RI:
// start < end
// when all the intervals have been added, the isNoBlank() should return true

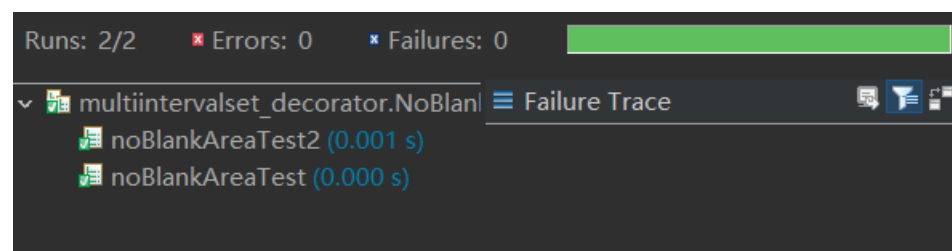
// safety from rep exposure:
// the rep is private and mutable
```

在这 ADT 中增加了 start 和 end 两个属性。因为我们如果不知道开始时间和结束时间是什么的话，是无法定义空白时间这个概念的，所以要增加定义这两个属性。

而 isNoBlank 就是根据这两个属性进行检测是否有空白时间段的。

Testing strategy:

```
// Testing Strategy:
// have no blank area
// have no blank area but have overlapped area
// have blank area
```



NonOverlapMultiIntervalSet:

NonOverlapMultiIntervalSet<L>	
multiintervalset_decorator	
	create(is: IntervalSet<L>): NonOverlapMultiIntervalSet<L>
	create(mis: MultiIntervalSet<L>): NonOverlapMultiIntervalSet<L>
	NonOverlapMultiIntervalSet(is: IntervalSet<L>): void
	NonOverlapMultiIntervalSet(mis: MultiIntervalSet<L>): void
	insert(start: long, end: long, label: L): boolean

Rep、AF、RI、safety from rep exposure:

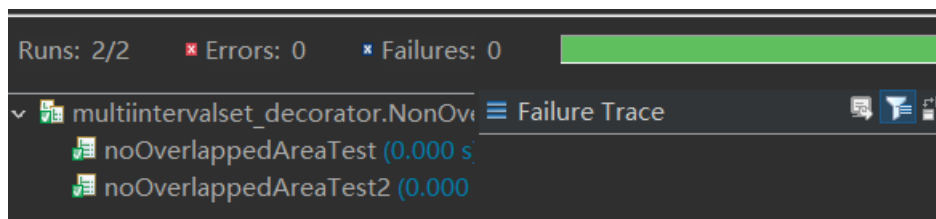
```
// AF:
// represents a intervalset with no intervals overlapped

// RI:
// each interval should not overlapped with other intervals









// the rep of super class guarantees the client can't get the reference of the rep
```

Testing strategy:

```
// TestingStrategy:
// have overlapped area
// have no overlapped area but have blank area
// have no overlapped area and have no blank area
```



PeriodicMultiIntervalSet:

 PeriodicMultiIntervalSet<L> multiintervalset_decorator
 periodic_info: Map<L,Integer>
 create(is: IntervalSet<L>): PeriodicMultiIntervalSet<L>
 create(mis: MultiIntervalSet<L>): PeriodicMultiIntervalSet<L>
 PeriodicMultiIntervalSet(is: IntervalSet<L>): void
 PeriodicMultiIntervalSet(mis: MultiIntervalSet<L>): void
 setPeriodic(label: L, i: Integer): boolean
 getPeriod(label: L): int

Rep、AF、RI、safety from rep exposure:

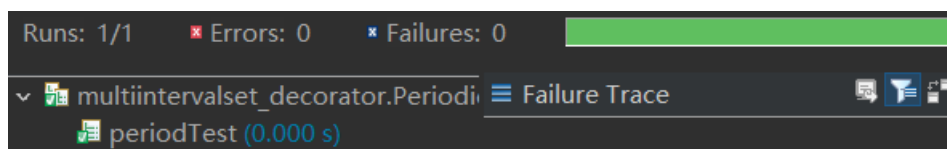
```
// AF:
// represents an intervalset which intervals can repeat in a period

// RI:
// the period of every interval >= 0
// the period of the repeated intervals >0

// safety from rep exposure:
// periodic_info is private
```

Testing strategy:

```
// Testing strategy:
// set a period for a non-existent interval
// set a period for a existent interval
// get a period of a non-existent interval
// set a period of a existent interval
// insert more than one intervals to by one label
```



以上三个具体的实现类都实现了静态工厂方法，故在使用时可以用工厂方法来替代 new 操作。如果要叠加多重特征的话，嵌套定义即可。

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）

和 IntervalSet 中不用实现更具体的子类理由相同，MultiIntervalSet 除了上面三个特殊的子类也不同实现更加具体的子类了。

进程调度表是一个可以用空白，但是不能重叠的时间段集合。故我们建立一个 NonOverlapMultiIntervalSet 作为其 rep 的一部分，用以管理进程。

```
// rep
private MultiIntervalSet<MyProcess> process_table = NonOverlapMultiIntervalSet.create(MultiIntervalSet.empty());
```

而在课程表的例子中，我们可以看到，这是一个可重叠、可有空白、可重复的时间集合，而且我们需要调用到 PeriodicMultiIntervalSet 中的个性方法，故我们直接建立一个 PeriodicMultiIntervalSet 作为其 rep。

```
private PeriodicMultiIntervalSet<Course> courseschedule = PeriodicMultiIntervalSet.create(MultiIntervalSet.empty());
```

3.4 面向复用的设计：L

注意到上面的所有的 ADT 的实现中，我们使用 label 的类型都是 L，即采用了泛型的技术，等到实际创建对象的时候，这个类型才会被确定。

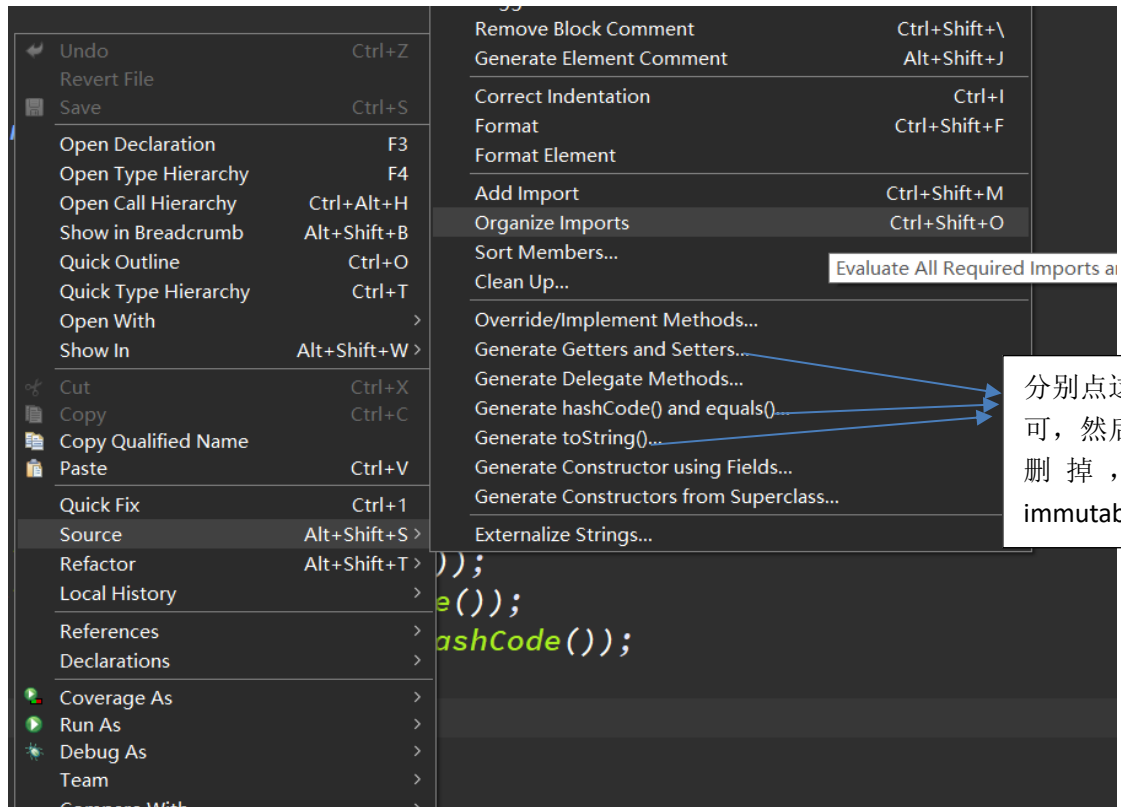
而根据本次实验所实现的三个应用，我们的 label 类型有三种，均为 immutable 的，一旦输入信息确定后，就不可变。分别是值班表中的 Employee，代表一个员工，进程表中的进程 MyProcess，代表一个进程，课程表中的 Course，代表一门课程。

为了保证 immutable 对象的等价性，我们需要实现 hashCode 和 equals 方法，为了方便输出信息，我们需要实现 toString 方法，为了能够得到成员变量的信息，我们需要实现 setter 方法。

这三个对象都只是存储信息的简单类，且是 immutable 的，我们只要定义好其私有成员变量，然后用 IDE 自动生成功能即可生成 hashCode、equals、toString、setter 方法。

Employee：代表一个员工，其信息有年龄、职务、电话。

```
private final String name;
private final String position;
private final String phone;
```



MyProcess: 代表一个进程，其信息有 pid，名字，最短执行时间，最长执行时间。

```
private final String pid;
private final String name;
private final long shortest_exe_time;
private final long longest_exe_time;
```

同样利用 IDE 自动生成 hashCode、equals、toString、getter 方法。

Course: 代表一门课，其信息有课程 id，名字，教师，教学地点，周学时

```
private final String course_id;
private final String name;
private final String teacher;
private final String teachingposition;
private final int hoursperweek;
```

同样利用 IDE 自动生成 hashCode、equals、toString、getter 方法。

3.5 可复用 API 设计

定义了一个叫做 `APIs` 的类, 在其中实现了五个 `static` 方法, 分别计算相似度、冲突比例、空白比例。

3.5.1 计算相似度

```
/**
 * calculate the similarity between two MultiIntervalSet s1 and s2
 * @param <L> the type of label
 * @param s1 the first MultiIntervalSet
 * @param s2 the second MultiIntervalSet
 * @return the similarity of two set
 */
public static <L> double Similarity(MultiIntervalSet<L> s1, MultiIntervalSet<L> s2) {
```

实现是通过遍历两者的时间段集合, 将其按照标签取出, 然后通过循环比对两个集合之间重叠的部分, 如果有则记录长度, 最后除以总长度。

```
@Test
public void similarityTest() throws Exception {
    MultiIntervalSet<String> s1 = MultiIntervalSet.empty();
    MultiIntervalSet<String> s2 = MultiIntervalSet.empty();
    s1.insert(0, 5, "A");
    s1.insert(20, 25, "A");
    s1.insert(10, 20, "B");
    s1.insert(25, 30, "B");

    s2.insert(20, 35, "A");
    s2.insert(10, 20, "B");
    s2.insert(0, 5, "C");
    assertEquals(0.42857142857142855, APIs.Similarity(s1, s2), 1e-5);
}
```

通过测试。

3.5.2 计算时间冲突比例

```
/**
 * calculate the conflict ratio of the IntervalSet
 * @param <L> the type of label
 * @param set the IntervalSet
 * @return the conflict ratio
 */
public static <L> double calcConflictRatio(IntervalSet<L> set) {
```

```

/**
 * calculate the conflict ratio of the MultiIntervalSet
 * @param <L> the type of label
 * @param set the MultiIntervalSet
 * @param start the start time of the MultiIntervalSet
 * @param end the end time of the MultiIntervalSet
 * @return the conflict ratio
 */
public static <L> double calcConflictRatio(MultiIntervalSet<L> set, long start, long end) {

```

对于如何确定时间轴长度来说，有两种实现角度，一种是通过遍历直接得到，用最大的结束时间减去最小的开始时间；另一种是我们人工确定，通过输入开始时间和结束时间。对于 NoBlankIntervalSet 和 NoBlankMultiIntervalSet 来说，其开始时间和结束时间必须是人为确定的，否则根本无法定义所谓空白时间，因为开头和结尾也是有空白时间的，这个开头和结尾必须是人为确定的。

对于课程表来说，其是有 start 和 end 属性的，其时间轴长度是 end-start。所以我们针对 IntervalSet 实现一种，而针对 MultiIntervalSet 实现另一种。

```

@Test
public void calcConflictRatioTest() throws Exception {
    IntervalSet<String> is = IntervalSet.empty();
    is.insert(0, 10, "t1");
    is.insert(5, 20, "t2");
    assertEquals(0.25, APIS.calcConflictRatio(is), 1e-5);
    is.insert(30, 50, "t3");
    is.insert(40, 60, "t4");
    assertEquals(0.3, APIS.calcConflictRatio(is), 1e-5);
}

@Test
public void calcConflictRatioTest2() throws Exception {
    MultiIntervalSet<String> mis = MultiIntervalSet.empty();
    mis.insert(0, 10, "t1");
    mis.insert(0, 10, "t2");
    mis.insert(10, 20, "t4");
    mis.insert(20, 30, "t3");
    assertEquals(1.0/3, APIS.calcConflictRatio(mis, 0, 30), 1e-5);
}

```

分别进行测试。

3.5.3 计算空闲时间比例

这个功能主要是针对课程表，同上面的理由，必须手动输入 start 和 end 参数来确定时间轴长度。

```

/**
 * calculate the free time ratio of the IntervalSet
 * @param <L> the type of the ratio
 * @param set the IntervalSet
 * @param start the start time of the IntervalSet
 * @param end the end time of the IntervalSet
 * @return the free time ratio
 */
public static <L> double calcFreeTimeRatio(IntervalSet<L> set, long start, long end) {

```



```

/**
 * calculate the free time ratio of the MultiIntervalSet
 * @param <L> the type of the ratio
 * @param set the MultiIntervalSet
 * @param start the start time of the MultiIntervalSet
 * @param end the end time of the MultiIntervalSet
 * @return the free time ratio
 */
public static <L> double calcFreeTimeRatio(MultiIntervalSet<L> set, long start, long end)

```

```

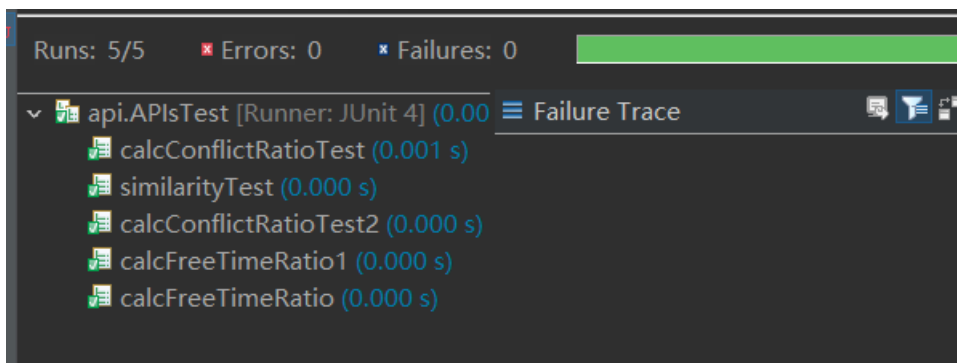
@Test
public void calcFreeTimeRatio() throws Exception {
    IntervalSet<String> is = IntervalSet.empty();
    is.insert(0, 50, "t1");
    assertEquals(0, APIS.calcFreeTimeRatio(is, 0, 50), 1e-5);
    is.insert(100, 200, "t2");
    assertEquals(0.25, APIS.calcFreeTimeRatio(is, 0, 200), 1e-5);
}

@Test
public void calcFreeTimeRatio1() throws Exception {
    MultiIntervalSet<String> mis = MultiIntervalSet.empty();
    mis.insert(0, 20, "t1");
    assertEquals(0, APIS.calcFreeTimeRatio(mis, 0, 20), 1e-5);
    mis.insert(30, 40, "t1");
    assertEquals(0.25, APIS.calcFreeTimeRatio(mis, 0, 40), 1e-5);
    mis.insert(40, 50, "t2");
    assertEquals(0.2, APIS.calcFreeTimeRatio(mis, 0, 50), 1e-5);
}

```

分别进行测试。

对于以上三种五个方法的测试用例通过。



3.6 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。

3.6.1 排班管理系统

我们首先实现一个排班管理的 ADT，在实现好的子类去构造一个命令行 APP。

```
> ConcreteDutyRoster.java
> DutyRoster.java
> DutyRosterApp.java
```

DutyRoster 是接口，ConcreteDutyRoster 是实现类，DutyRosterApp 是命令行程序。

DutyRoster	
duty	
	create(starttime: String, endtime: String): ConcreteDutyRoster
	create(): ConcreteDutyRoster
	addStaff(name: String, position: String, phone: String): boolean
	deleteStaff(name: String, position: String, phone: String): boolean
	getStaff(name: String): Employee
	setStartTime(starttime: String): void
	setEndTime(endtime: String): void
	isarranged(name: String, position: String, phone: String): boolean
	deleteArrangement(name: String, position: String, phone: String): boolean
	arrange_manually(name: String, position: String, phone: String, starttime: String, days: long): boolean
	arrange_automatically(): boolean
	display(): void
	isNoBlank(): boolean

约定：2021-02-10 ~ 2021-02-10 代表的是一整天，而 2021-01-21 ~ 2021-01-22 代表的是两天

- 1.addStaff: 添加一个员工，输入姓名，职务，电话。
- 2.deleteStaff: 删除一个员工，输入姓名，职务，电话
- 3.getStaff: 根据姓名得到一个员工的信息
- 4.setStartTime: 设置开始时间，输入格式 yyyy-MM-dd
- 5.setEndTime: 设置结束时间，输入格式 yyyy-MM-dd
- 6.isarranged: 判断某个人是否被排班了，输入姓名，职务，电话。
- 7.deleteArrangement: 删除某人的值班任务，输入姓名，职务，电话。
- 8.arrange_munually: 手动为某人安排值班任务，输入姓名，职务，电话，开始时间，持续时间。如果当前此人已被排班，会将其以前的排班信息删除再重新排班。
- 9.arrange_automatically: 自动排班
- 10.展示当前的排班情况。
- 11.isNoBlank: 检查是否有空白

并设置有静态工厂方法来创建对象。

其实现类 ConcrteDutyRoster:

ConcreteDutyRoster
duty
<ul style="list-style-type: none"> calendar: NoBlankIntervalSet<Employee> staffset: Set<Employee> start: long end: long
<ul style="list-style-type: none"> ConcreteDutyRoster(starttime: String, endtime: String) ConcreteDutyRoster() addStaff(name: String, position: String, phone: String): boolean deleteStaff(name: String, position: String, phone: String): boolean setStartTime(starttime: String): void setEndTime(endtime: String): void isarranged(name: String, position: String, phone: String): boolean arrange_manually(name: String, position: String, phone: String, starttime: String, days: long): boolean arrange_automatically(): boolean display(): void isNoBlank(): boolean toString(): String deleteArrangement(name: String, position: String, phone: String): boolean getStaff(name: String): Employee

```
// rep
private NoBlankIntervalSet<Employee> calendar = NoBlankIntervalSet
    .create(NonOverlapIntervalSet.create(IntervalSet.empty()));
private Set<Employee> staffset = new HashSet<>();
private long start;
private long end;

// AF
// represents a duty roster system

// RI
// dutytime can't overlap

// safety from rep exposure
// all the fields are private
```

由于排班表是一个不能有空白、不能重复的时间段集合，故在其 rep 中设置一个通过 NoBlankIntervalSet、NonOverlapIntervalSet 装饰的集合。

其中的方法均依赖于这个时间段集合实现。具体实现细节大同小异，多是以循环遍历比较为主，不再赘述。

在 DutyRosterTest 简单实现一个测试用例，打印输出结果来展示效果。

```

@Test
public void test3() throws Exception {
    DutyRoster dr = DutyRoster.create("2021-01-10", "2021-01-18");
    dr.addStaff("zhang-san", "sceretary", "10000001");
    dr.addStaff("zhang-si", "sceretary", "10000002");
    dr.addStaff("zhang-wu", "sceretary", "10000003");
    dr.addStaff("zhang-liu", "sceretary", "10000004");
    dr.addStaff("zhang-qi", "sceretary", "10000005");
    dr.addStaff("zhang-ba", "sceretary", "10000006");
    dr.arrange_manually("zhang-san", "sceretary", "10000001", "2021-01-10", 2);
    System.out.print(dr.toString());
    dr.arrange_manually("zhang-wu", "sceretary", "10000003", "2021-01-14", 1);
    System.out.print(dr.toString());
    dr.arrange_manually("zhang-ba", "sceretary", "10000006", "2021-01-16", 1);
    System.out.print(dr.toString());
    dr.display();
}

```

```

duty time: 2021-01-10 -- 2021-01-11      Employee [name=zhang-san, position=sceretary, phone=10000001]

No arranged interval:
2021-01-12 -- 2021-01-18

Blank ratio: 0.7777777777777778
duty time: 2021-01-10 -- 2021-01-11      Employee [name=zhang-san, position=sceretary, phone=10000001]
duty time: 2021-01-14 -- 2021-01-14      Employee [name=zhang-wu, position=sceretary, phone=10000003]

No arranged interval:
2021-01-12 -- 2021-01-13
2021-01-15 -- 2021-01-18

Blank ratio: 0.6666666666666666
duty time: 2021-01-10 -- 2021-01-11      Employee [name=zhang-san, position=sceretary, phone=10000001]
duty time: 2021-01-14 -- 2021-01-14      Employee [name=zhang-wu, position=sceretary, phone=10000003]
duty time: 2021-01-16 -- 2021-01-16      Employee [name=zhang-ba, position=sceretary, phone=10000006]

No arranged interval:
2021-01-12 -- 2021-01-13
2021-01-15 -- 2021-01-15
2021-01-17 -- 2021-01-18

Blank ratio: 0.5555555555555556
duty time: 2021-01-10 -- 2021-01-11      Employee [name=zhang-san, position=sceretary, phone=10000001]
duty time: 2021-01-14 -- 2021-01-14      Employee [name=zhang-wu, position=sceretary, phone=10000003]
duty time: 2021-01-16 -- 2021-01-16      Employee [name=zhang-ba, position=sceretary, phone=10000006]

No arranged interval:
2021-01-12 -- 2021-01-13
2021-01-15 -- 2021-01-15
2021-01-17 -- 2021-01-18

Blank ratio: 0.5555555555555556

```

输出结果和安排一致。

其他几个测试用例，则不输出，用 assertEquals 方法进行对比。

```

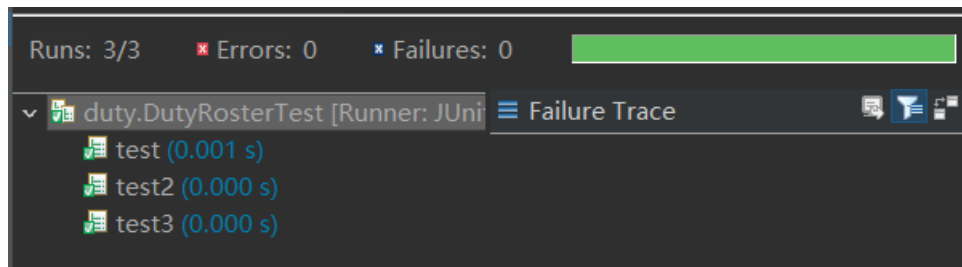
@Test
public void test() throws Exception {
    DutyRoster dr = DutyRoster.create("2021-01-10", "2021-01-18");
    dr.addStaff("zhang-san", "sceretary", "10000001");
    dr.addStaff("zhang-si", "sceretary", "10000002");
    dr.addStaff("zhang-wu", "sceretary", "10000003");
    dr.addStaff("zhang-liu", "sceretary", "10000004");
    dr.addStaff("zhang-qi", "sceretary", "10000005");
    dr.addStaff("zhang-ba", "sceretary", "10000006");
    dr.arrange_automatically();
    assertEquals(
        "duty time: 2021-01-10 -- 2021-01-10      Employee [name=zhang-si, position=sceretary, phone=10000002]\n"
        + "duty time: 2021-01-11 -- 2021-01-11      Employee [name=zhang-qi, position=sceretary, phone=10000005]\n"
        + "duty time: 2021-01-12 -- 2021-01-12      Employee [name=zhang-liu, position=sceretary, phone=10000004]\n"
        + "duty time: 2021-01-13 -- 2021-01-13      Employee [name=zhang-ba, position=sceretary, phone=10000006]\n"
        + "duty time: 2021-01-14 -- 2021-01-14      Employee [name=zhang-san, position=sceretary, phone=10000001]\n"
        + "duty time: 2021-01-15 -- 2021-01-18      Employee [name=zhang-wu, position=sceretary, phone=10000003]\n"
        + "\n" + "The duty time has been fully arranged!\n",
        dr.toString());
}

```

```

@Test
public void test2() throws Exception {
    DutyRoster dr = DutyRoster.create("2021-01-10", "2021-01-17");
    dr.addStaff("zhang-san", "sceretary", "10000001");
    dr.addStaff("zhang-si", "sceretary", "10000002");
    dr.addStaff("zhang-wu", "sceretary", "10000003");
    dr.addStaff("zhang-liu", "sceretary", "10000004");
    dr.addStaff("zhang-qi", "sceretary", "10000005");
    dr.addStaff("zhang-ba", "sceretary", "10000006");
    dr.arrange_manually("zhang-san", "sceretary", "10000001", "2021-01-11", 3);
    assertEquals(
        "duty time: 2021-01-11 -- 2021-01-13   Employee [name=zhang-san, position=sceretary, phone=10000001]\n"
        + "\n" + "No arranged interval: \n" + "2021-01-10 -- 2021-01-10\n"
        + "2021-01-14 -- 2021-01-17\n" + "\n" + "Blank ratio: 0.625\n",
        dr.toString());
}

```



简单的测试用例完成后，我们根据实现的类来构造一个命令行 app。

测试时注意：如果当前此人已被排班，会将其以前的排班信息删除再重新排班。

创建一个 DutyRoser 代表排班系统，接着通过不断地让用户输入指令来决定接下来要进行什么操作。

程序是在 DutyRosterApp 中。

```

Welcome to the DutyRosterApp!
-----
Menu:
1.Set the start time of duty interval
2.Set the end time of duty interval
3.Add a staff information
4.Delete a staff information
5.Arrange a staff to be on duty
6.Arrange the duty time table automatically
7.Check the duty time table arrangement situation
8.Delete the arrangemet of a staff
9.Arrange the duty time according to the input file
10.Exit

You must firstly set the start and end time! Or the system can't work properly!

```

在测试时一定要先设定好开始开始和结束时间。

输入 1 设置开始时间：

```

Please the start time, the format is yyyy-MM-dd:
2021-01-20

```

输入一定要符合格式 yyyy-MM-dd。

输入 2 设置结束时间：

```

Please the end time, the format is yyyy-MM-dd:
2021-01-30

```

输入一定要符合格式 yyyy-MM-dd。

接下来输入几个员工的信息：

```
3
Please input the staff's information according to the instructions!
Please input the name:
zhang-san
Please input the position:
secretary
Please input the phone number:
100001
If you want to continue to add the staff, please input 1!
1
Please input the staff's information according to the instructions!
Please input the name:
li-si
Please input the position:
guard
Please input the phone number:
100002
If you want to continue to add the staff, please input 1!
1
Please input the staff's information according to the instructions!
Please input the name:
wang-wu
Please input the position:
teacher
Please input the phone number:
1000003
If you want to continue to add the staff, please input 1!
0
```

根据命令行中的提示信息来不断地输入信息。

接着我们手动的来排班，每安排一次班就展示一次信息。

```
5
Please input the staff's information and time interval according to the instructions!
Please input the name:
zhang-san
Please input the position:
secretary
Please input the phone number:
100001
Please input the start time, the format is yyyy-MM-dd:
2021-01-20
Please input the lasting time:
4
```

```

Menu:
1.Set the start time of duty interval
2.Set the end time of duty interval
3.Add a staff information
4.Delete a staff information
5.Arrange a staff to be on duty
6.Arrange the duty time table automatically
7.Check the duty time table arrangement situation
8.Delete the arrangement of a staff
9.Arrange the duty time according to the input file
10.Exit

7
duty time: 2021-01-20 -- 2021-01-23      Employee [name=zhang-san, position=secretary, phone=100001]

No arranged interval:
2021-01-24 -- 2021-01-30

Blank ratio: 0.6363636363636364

```

排班信息和空闲率均正确。继续安排。

```

5
Please input the staff's information and time interval according to the instructions!
Please input the name:
li-si
Please input the position:
guard
Please input the phone number:
100002
Please input the start time, the format is yyyy-MM-dd:
2021-01-24
Please input the lasting time:
4
..

```

展示:

```

duty time: 2021-01-20 -- 2021-01-23      Employee [name=zhang-san, position=secretary, phone=100001]
duty time: 2021-01-24 -- 2021-01-27      Employee [name=li-si, position=guard, phone=100002]

No arranged interval:
2021-01-28 -- 2021-01-30

Blank ratio: 0.2727272727272727

```

继续安排:

```

5
Please input the staff's information and time interval according to the instructions!
Please input the name:
wang-wu
Please input the position:
teacher
Please input the phone number:
1000003
Please input the start time, the format is yyyy-MM-dd:
2021-01-28
Please input the lasting time:
3

```

此时整个时间段都已经被安排满了。

展示。

```

7
duty time: 2021-01-20 -- 2021-01-23      Employee [name=zhang-san, position=secretary, phone=100001]
duty time: 2021-01-24 -- 2021-01-27      Employee [name=li-si, position=guard, phone=100002]
duty time: 2021-01-28 -- 2021-01-30      Employee [name=wang-wu, position=teacher, phone=1000003]

The duty time has been fully arranged!

```

接着我们再重新运行程序，尝试一下输入职工后，自动排班。

首先仍然时设置开始、结束时间。

```

Welcome to the DutyRosterApp!
-----
Menu:
1.Set the start time of duty interval
2.Set the end time of duty interval
3.Add a staff information
4.Delete a staff information
5.Arrange a staff to be on duty
6.Arrange the duty time table automatically
7.Check the duty time table arrangement situation
8.Delete the arrangemet of a staff
9.Arrange the duty time according to the input file
10.Exit

You must firstly set the start and end time! Or the system can't work properly!
1
Please the start time, the format is yyyy-MM-dd:
2021-01-20
Menu:
1.Set the start time of duty interval
2.Set the end time of duty interval
3.Add a staff information
4.Delete a staff information
5.Arrange a staff to be on duty
6.Arrange the duty time table automatically
7.Check the duty time table arrangement situation
8.Delete the arrangemet of a staff
9.Arrange the duty time according to the input file
10.Exit

2
Please the end time, the format is yyyy-MM-dd:
2021-01-30

```

接着输入员工：
输入四个员工。


```

3
Please input the staff's information according to the instructions!
Please input the name:
zhang-san
Please input the position:
guard
Please input the phone number:
100001
If you want to continue to add the staff, please input 1!
1
Please input the staff's information according to the instructions!
Please input the name:
li-si
Please input the position:
guard
Please input the phone number:
100002
If you want to continue to add the staff, please input 1!
1
Please input the staff's information according to the instructions!
Please input the name:
wang-wu
Please input the position:
teacher
Please input the phone number:
100003
If you want to continue to add the staff, please input 1!
1
Please input the staff's information according to the instructions!
Please input the name:
zhao-liu
Please input the position:
teacher
Please input the phone number:
100004
If you want to continue to add the staff, please input 1!
0

```

自动排班:

接着展示:

```

6
Menu:
1.Set the start time of duty interval
2.Set the end time of duty interval
3.Add a staff information
4.Delete a staff information
5.Arrange a staff to be on duty
6.Arrange the duty time table automatically
7.Check the duty time table arrangement situation
8.Delete the arrangement of a staff
9.Arrange the duty time according to the input file
10.Exit

7
duty time: 2021-01-20 -- 2021-01-20      Employee [name=zhao-liu, position=teacher, phone=100004]
duty time: 2021-01-21 -- 2021-01-21      Employee [name=zhang-san, position=guard, phone=100001]
duty time: 2021-01-22 -- 2021-01-22      Employee [name=li-si, position=guard, phone=100002]
duty time: 2021-01-23 -- 2021-01-30      Employee [name=wang-wu, position=teacher, phone=100003]

The duty time has been fully arranged!

```

可以发先全部排好了。

我们试着删除一个员工:

```

4
Please input the staff's information according to the instructions!
Please input the name:
zhao-liu
Please input the position:
teacher
Please input the phone number:
100004
This staff has been arranged to be on duty, you can't delete him/her from the system!
If you want to delete her/him, you should delete the arrangement of her/him!

```

发现提示我们这个人已经被排班，不能将其删除，我们要先将他的排班任务删除。输入 8，根据指示删除。

```

8
Please input the staff's information according to the instructions!
Please input the name:
zhao-liu
Please input the position:
teacher
Please input the phone number:
100004

```

再次展示，可以发现其排班信息已被删除。

```

7
duty time: 2021-01-21 -- 2021-01-21      Employee [name=zhang-san, position=guard, phone=100001]
duty time: 2021-01-22 -- 2021-01-22      Employee [name=li-si, position=guard, phone=100002]
duty time: 2021-01-23 -- 2021-01-30      Employee [name=wang-wu, position=teacher, phone=100003]

No arranged interval:
2021-01-20 -- 2021-01-20

Blank ratio: 0.09090909090909091

```

再次尝试删除他。

```

4
Please input the staff's information according to the instructions!
Please input the name:
zhao-liu
Please input the position:
teacher
Please input the phone number:
100004










```

删除成功。

3.6.2 操作系统的进程调度管理系统



















我们首先构造一个 ProcessSchedule，作为进程调度管理系统。由于这个时间段集合是一个标签对应多个时间段的，且是可以有空白，但不能有重叠的，所以我们创建一个 NonOverlapMultiIntervalSet 作为其 rep 存储。

首先构造接口，描述其行为：

 ProcessSchedule
process
 <u>create(): ProcessSchedule</u>  fork(pid: String, name: String, shortest_exe_time: long, longest_exe_time: long): boolean  schedule_stochastic(): boolean  schedule_shortestfirst(): boolean  isAllTerminated(): boolean  display(): void  noProcessRun(): void  schedule_automatically(): void

1. fork: 添加一个新的进程到进程管理系统中。
2. schedule_stochastic: 随机调度一个进程开始执行一段时间。
3. schedule_shortestfirst: 根据 shortest 算法调度一个进程执行一段时间。
4. iaAllTerminated: 判断当前进程表中是否所有的进程都已经终止。
5. display: 展示当前时刻之前所有的进程执行情况，和当前正在执行的进程。
6. noProcessRun: 将 cpu 悬空一段时间，即该段时间内没有进程执行，全部挂起。
7. schedule_automatically: 自动安排进程执行，并直接显示进程执行情况。

根据接口进行实现：

 ConcreteProcessSchedule
process
 process_table: MultiIntervalSet<MyProcess>  process_group: Map<String,MyProcess>  suspended_group: Set<MyProcess>  running_time: Map<MyProcess,Long>  running_instance: MyProcess  process_state: Map<MyProcess,ProcessState>  now: long
 fork(pid: String, name: String, shortest_exe_time: long, longest_exe_time: long): boolean  schedule_stochastic(): boolean  schedule_shortestfirst(): boolean  stop(): void  isTerminated(pid: String): boolean  isAllTerminated(): boolean  display(): void  toString(): String  noProcessRun(): void  schedule_automatically(): void

```

// rep
private MultiIntervalSet<MyProcess> process_table = NonOverlapMultiIntervalSet.create(MultiIntervalSet.empty());
private Map<String, MyProcess> process_group = new HashMap<>();
private Set<MyProcess> suspended_group = new HashSet<>();
private Map<MyProcess, Long> running_time = new HashMap<>();
private MyProcess running_instance = null;
private Map<MyProcess, ProcessState> process_state = new HashMap<>();
private long now = 0;

// AF
// represents a process schedule system

// RI
// each process can't overlap with each other

// safety from rep exposure
// all filed are private, and the client can't get the reference of process_table

```

设置了一个枚举类来代表进程的三个状态：运行时、挂起、终结

```

/**
 * represents the state of process, every process must be in one of these states
 */
private enum ProcessState {
    RUNNING, SUSPENDED, TERMINATED
}

```

进程调度的时候其执行时间是不确定的，所以用随机数产生一个不会超过剩余最大执行时间的数。

在测试类中创建几个进程，随便输入几个数据进行测试。

```

public class ProcessScheduleTest {
    @Test
    public void processTest() {
        ProcessSchedule ps = ProcessSchedule.create();
        ps.fork("0001", "p1", 10, 20);
        ps.fork("0002", "p2", 30, 50);
        ps.fork("0003", "p3", 15, 30);
        while(!ps.isAllTerminated()) {
            ps.schedule_stochastic();
        }
        ps.display();
    }

    @Test
    public void processTest2() {
        ProcessSchedule ps = ProcessSchedule.create();
        ps.fork("0001", "p1", 10, 20);
        ps.fork("0002", "p2", 30, 50);
        ps.fork("0003", "p3", 15, 30);
        while(!ps.isAllTerminated()) {
            ps.schedule_shortestfirst();
        }
        ps.display();
    }
}

```

直接查看输出结果:

```

The process running situation:
0 -- 12: Process [pid=0001, name=p1]
12 -- 37: Process [pid=0003, name=p3]
37 -- 82: Process [pid=0002, name=p2]

The process running situation:
0 -- 14: Process [pid=0001, name=p1]
14 -- 38: Process [pid=0003, name=p3]
38 -- 82: Process [pid=0002, name=p2]

```

再执行一次, 验证其随机性:

```

The process running situation:
0 -- 46: Process [pid=0002, name=p2]
46 -- 67: Process [pid=0003, name=p3]
67 -- 68: Process [pid=0001, name=p1]
68 -- 79: Process [pid=0001, name=p1]

The process running situation:
0 -- 18: Process [pid=0001, name=p1]
18 -- 26: Process [pid=0003, name=p3]
26 -- 40: Process [pid=0003, name=p3]
40 -- 88: Process [pid=0002, name=p2]

```

我们可以看到，所有进程调度执行时间都是在最大执行时间内的。

接着我们构造一个进程调度管理系统的命令行 app，由用户的输入来控制进程调用。首先在主方法中创建一个 ProcessSchedule，接着根据用户的输入来调用其方法。

```
Welcome to ProcessSechedule simulator!
-----
Please input a set of process,
the process format is pid name shortest_exe_time longest_exe_time(each term delimited by one blank space):
```

首先根据提示信息，按照给出的格式进行输入进程信息。

```
Welcome to ProcessSechedule simulator!
-----
Please input a set of process,
the process format is pid name shortest_exe_time longest_exe_time(each term delimited by one blank space):
001 p1 10 20
If you want to fork another process, input 1.
Otherwise to terminate the input.
1
Please input a set of process,
the process format is pid name shortest_exe_time longest_exe_time(each term delimited by one blank space):
002 p2 20 25
If you want to fork another process, input 1.
Otherwise to terminate the input.
1
Please input a set of process,
the process format is pid name shortest_exe_time longest_exe_time(each term delimited by one blank space):
003 p3 5 15
If you want to fork another process, input 1.
Otherwise to terminate the input.
1
Please input a set of process,
the process format is pid name shortest_exe_time longest_exe_time(each term delimited by one blank space):
004 p4 40 60
If you want to fork another process, input 1.
Otherwise to terminate the input.
0
```

接着弹出菜单，我们根据提示信息进行进程调度即可。

```
Attention: if all the processes have been terminated,
the app will exit automatically and display the schedule situation!
Menu:
1.schedule a process stochastically
2.schedule a process with shortestfrist algorithm
3.cpu works with no process running
4.display the current situation
5.schedule all the processes automatically
6.exit
```

先随机调度一个进程，然后检查运行情况：

```

Menu:
1.schedule a process stochastically
2.schedule a process with shortestfrist algorithm
3.cpu works with no process running
4.display the current situation
5.schedule all the processes automatically
6.exit
1
Menu:
1.schedule a process stochastically
2.schedule a process with shortestfrist algorithm
3.cpu works with no process running
4.display the current situation
5.schedule all the processes automatically
6.exit
4
Running process:
Process [pid=004, name=p4]

```

接着多随机调度几次，或者按 shortestfrist 调度几次，或者空挂 cpu 几次，然后查看结果：

```

The process running situation:
0 -- 25: Process [pid=004, name=p4]
25 -- 73: no process is running
73 -- 82: Process [pid=004, name=p4]
82 -- 92: no process is running
92 -- 95: Process [pid=003, name=p3]
95 -- 103: Process [pid=003, name=p3]
103 -- 124: Process [pid=004, name=p4]
124 -- 137: Process [pid=001, name=p1]

```

我们可以查看到以往的进程运行情况。

最后我们一直调度，直到所有的进程均执行完毕，此时会自动退出并显示所有的实行信息。

```

The process running situation:
0 -- 25: Process [pid=004, name=p4]
25 -- 73: no process is running
73 -- 82: Process [pid=004, name=p4]
82 -- 92: no process is running
92 -- 95: Process [pid=003, name=p3]
95 -- 103: Process [pid=003, name=p3]
103 -- 124: Process [pid=004, name=p4]
124 -- 137: Process [pid=001, name=p1]
137 -- 177: no process is running
177 -- 194: Process [pid=002, name=p2]
194 -- 199: Process [pid=002, name=p2]

```

我们重新执行，享受一下随机执行，系统会根据生成的随机数随机选择执行 1~3，即随即调度、shortestfrist 调度，cpu 空挂。

首先 fork 四个进程：

```

Welcome to ProcessSechdule simulator!
-----
Please input a set of process,
the process format is pid name shortest_exe_time longest_exe_time(each term delimited by one blank space):
001 p1 5 10
If you want to fork another process, input 1.
Otherwise to terminate the input.
1
Please input a set of process,
the process format is pid name shortest_exe_time longest_exe_time(each term delimited by one blank space):
002 p2 10 30
If you want to fork another process, input 1.
Otherwise to terminate the input.
1
Please input a set of process,
the process format is pid name shortest_exe_time longest_exe_time(each term delimited by one blank space):
003 p3 25 50
If you want to fork another process, input 1.
Otherwise to terminate the input.
1
Please input a set of process,
the process format is pid name shortest_exe_time longest_exe_time(each term delimited by one blank space):
004 p4 10 40
If you want to fork another process, input 1.
Otherwise to terminate the input.
0

```

接着输入 5，自动调度执行完所有的程序。

```

Attention: if all the processes have been terminated,
the app will exit automatically and display the schedule situation!
Menu:
1.schedule a process stochastically
2.schedule a process with shortestfrist algorithm
3.cpu works with no process running
4.display the current situation
5.schedule all the processes automatically
6.exit
5
The process running situation:
0 -- 2: Process [pid=001, name=p1]
2 -- 6373: no process is running
6373 -- 6375: Process [pid=001, name=p1]
6375 -- 6380: Process [pid=001, name=p1]
6380 -- 6402: no process is running
6402 -- 6425: Process [pid=002, name=p2]
6425 -- 6447: no process is running
6447 -- 6460: Process [pid=004, name=p4]
6460 -- 6482: no process is running
6482 -- 6505: Process [pid=003, name=p3]
6505 -- 6549: no process is running
6549 -- 6557: Process [pid=003, name=p3]

```

我们可以发先从 2 到 6373 一直在悬空，这是正常现象，如下图：


```

@Override
public void schedule_automatically() {
    int r = new Random(System.currentTimeMillis()).nextInt(3);
    while (true) {
        switch (r) {
            case 0:
                schedule_stochastic();
                break;
            case 1:
                schedule_shortestfirst();
            case 2:
                noProcessRun();
            default:
                break;
        }
        r = new Random(System.currentTimeMillis()).nextInt(3);
        if (isAllTerminated())
            break;
    }
}
}

```

实现自动调度的时候,因为很可能一直生成随机数 1,而导致整个 cpu 一直悬停,所以会出现较长的悬空期。

3.6.3 课表管理系统

我们首先创建一个 CourseSchedule 接口来描述整个课程管理系统的行为。

CourseSchedule
course
<ul style="list-style-type: none"> create(): CourseSchedule setStartTime(start: String): boolean setTotalWeeks(totalweeks: int): void addCourse(course_id: String, name: String, teacher: String, teachingposition: String, hoursperweek: int): boolean arrangeCourse(course_id: String, day: int, s_time: int): boolean checkSchedule(): void checkScheduleofOneDay(day: String): void

1. setStartTime: 设置开始时间
2. setTotalWeeks: 设置总周数
3. addCourse: 增加一门课程
4. arrangeCourse: 安排一门课程
5. checkSchedule: 查看整个课程安排
6. checkScheduleofOneDay: 查看某一天的课程

接着我们实现一个该接口的实现类: CommonCourseSchedule

ConcreteCourseSchedule
course
<ul style="list-style-type: none"> courseset: Map<String, Course> arranged_time: Map<String, Integer> courseschedule: PeriodicMultiIntervalSet<Course> starttime: long totalweeks: int
<ul style="list-style-type: none"> setStartTime(start: String): boolean setTotalWeeks(totalweeks: int): void addCourse(course_id: String, name: String, teacher: String, teachingposition: String, hoursperweek: int): boolean arrangeCourse(course_id: String, day: int, s_time: int): boolean checkSchedule(): void checkScheduleofOneDay(day: String): void toString(): String
<pre>// rep private Map<String, Course> courseset = new HashMap<>(); private Map<String, Integer> arranged_time = new HashMap<>(); private PeriodicMultiIntervalSet<Course> courseschedule = PeriodicMultiIntervalSet.create(MultiIntervalSet.empty()) private long starttime = 0; private int totalweeks = 0; // AF // represents a course schedule system // RI // every course can repeat week by week // safety from rep exposure // all the fields are private</pre>

我们简单输入几条信息，接着在测试类中打印其排课信息进行查看：

<pre>@Test public void coursescheduleTest() { CourseSchedule cs = CourseSchedule.create(); cs.setStartTime("2021-07-05"); cs.setTotalWeeks(18); cs.addCourse("cs001", "data structure", "wang", "201", 6); cs.addCourse("cs002", "c", "li", "302", 8); cs.addCourse("cs003", "algorithm", "zhao", "401", 4); cs.arrangeCourse("cs001", 3, 8); cs.arrangeCourse("cs003", 2, 10); cs.arrangeCourse("cs002", 7, 13); cs.checkSchedule(); }</pre>
<pre>Courses hasn't been arranged: NULL Schedule: Course [course_id=cs001, name=data structure, teacher=wang, teachingposition=201, hoursperweek=6] Wednesday 8~10 Course [course_id=cs003, name=algorithm, teacher=zhao, teachingposition=401, hoursperweek=4] Tuesday 10~12 Course [course_id=cs002, name=c, teacher=li, teachingposition=302, hoursperweek=8] Sunday 13~15 Blank Ratio: 0.9142857142857143 Conflict Ratio: 0.0</pre>

接着我们实现一个课程管理系统的命令行程序：根据用户输入的指令来进行课程安排。

先创建一个 CourseSchedule 对象，然后根据指令调用其各种方法。

```

Welcome to the course schedule system!
-----
Menu:
1.add a course to the system
2.set the start time of the semester
3.set the total weeks of this semester
4.arrange a course
5.check the course schedule
6.chech one day of the schedule
7.exit

You must firstly set the start and total weeks! Or the system can't work properly!

```

我们首先设定开始时间和总周数。

```

You must firstly set the start and total weeks! Or the system can't work properly!
2
Please input the starttime of the semester.
The format is yyyy-MM-dd,and the start day of the semeste should be on Monday!
Tips: 2021-07-05 is a legal example~
2021-07-05|

```

根据提示，由于学期的都是整周数的，所以开始日期必须是一个周一。
设置总周数：

```

3
Please the total weeks of the semester.
18

```

添加一些课程：

根据给出的格式输入。

```

Please input the course information according to the instructions:
the format is course_id name teacher teachingposition hoursperweek(each term is delimited by a single blank space)
Attention: course_id is unique for every course,
and the hoursperweek should be an even number.
Tips: cs001 DataStructure ZhangSan ZhengXin201 8 is a legal example~
cs001 datastructure zhang-san zheng-xin201 8
Menu:
1.add a course to the system
2.set the start time of the semester
3.set the total weeks of this semester
4.arrange a course
5.check the course schedule
6.check one day of the schedule
7.exit
1
Please input the course information according to the instructions:
the format is course_id name teacher teachingposition hoursperweek(each term is delimited by a single blank space)
Attention: course_id is unique for every course,
and the hoursperweek should be an even number.
Tips: cs001 DataStructure ZhangSan ZhengXin201 8 is a legal example~
cs002 programming li-si zheng-xin301 4
Menu:
1.add a course to the system
2.set the start time of the semester
3.set the total weeks of this semester
4.arrange a course
5.check the course schedule
6.chech one day of the schedule
7.exit
1
Please input the course information according to the instructions:
the format is course_id name teacher teachingposition hoursperweek(each term is delimited by a single blank space)
Attention: course_id is unique for every course,
and the hoursperweek should be an even number.
Tips: cs001 DataStructure ZhangSan ZhengXin201 8 is a legal example~
cs003 settheory wang-wu zheng-xin401 6

```

我们向系统中添加了三门课。

接着我们进行排课，并 check：

```
Please input the course arrangement information according to the instructions:
the format is course_id weekday start_time(each term is delimited by a single blank space)
the weekday is 1~7, which represents Monday~Sunday
the course could only be scheduled on 8~10/ 10~12/ 13~15/ 15~17/ 19~21, so the start time you input must be 8/10/13/15/19
Tips: cs001 4 8 is a legal example which means cs001 was scheduled on Thursday during 8~10
cs001 1 8
Menu:
1.add a course to the system
2.set the start time of the semester
3.set the total weeks of this semester
4.arrange a course
5.check the course schedule
6.check one day of the schedule
7.exit
5
Courses hasn't been arranged:
Course [course_id=cs002, name=programming, teacher=li-si, teachingposition=zheng-xin301, hoursperweek=4]
Course [course_id=cs003, name=settheory, teacher=wang-wu, teachingposition=zheng-xin401, hoursperweek=6]
Schedule:
Course [course_id=cs001, name=datastructure, teacher=zhang-san, teachingposition=zheng-xin201, hoursperweek=8]
Monday 8~10

Blank Ratio:
0.9714285714285714
Conflict Ratio:
0.0
```

接着我们多安排几次，并将两门课安排到同一时间，产生重叠的情况。

```
Courses hasn't been arranged:
NULL
Schedule:
Course [course_id=cs001, name=datastructure, teacher=zhang-san, teachingposition=zheng-xin201, hoursperweek=8]
Monday 8~10
Course [course_id=cs002, name=programming, teacher=li-si, teachingposition=zheng-xin301, hoursperweek=4]
Tuesday 10~12
Course [course_id=cs003, name=settheory, teacher=wang-wu, teachingposition=zheng-xin401, hoursperweek=6]
Monday 8~10
Wednesday 13~15

Blank Ratio:
0.9142857142857143
Conflict Ratio:
0.02857142857142857
```

其中空白率和冲突率的分母是一周的总时间，也就是 7 天*5 节课。

我们试着查看本学期内任意一天的课表

```
Please input the date, the format is yyyy-MM-dd
2021-08-18
2021-08-18
Wednesday
13~15 Course [course_id=cs003, name=settheory, teacher=wang-wu, teachingposition=zheng-xin401, hoursperweek=6]
```

可以发现，能够查找到。

我们输入一个不在学期内的时间

```
Please input the date, the format is yyyy-MM-dd
2020-01-01
The day you input should in the the semester!
```

可以发现会提示输入错误信息。

3.7 基于语法的数据读入

我们首先建立一个 `DutyRosterReader` 类，在其中写一个静态方法，其返回值是一个已经排好的排班系统。而参数一个读入的文件的路径信息。

我们首先将文件读入，然后用正则表达式去匹配其中的信息，分三块来匹配，找到创建系统所需要的信息。

首先从 Period 找到起止时间：

```
Pattern.compile("Period\\{\\{\\d{4}-\\d{2}-\\d{2}\\},(\\d{4}-\\d{2}-\\d{2})\\}\\}");
```

接着去找到每一个员工的信息，将其加入系统：

```
Pattern.compile("[a-zA-Z]+)\\{([a-zA-Z]+|s[a-zA-Z]+),(\\d{3}-\\d{4}-\\d{4})\\}\\}");
```

最后去找排班信息，为员工排班。

```
Pattern.compile("[a-zA-Z]+)\\{\\{\\d{4}-\\d{2}-\\d{2}\\},(\\d{4}-\\d{2}-\\d{2})\\}\\}");
```

我们首先创建一个叫做 DutyRosterRosterAppTest 的类来进行简单的测试，根据老师给出的 8 个文件我们依次查看输出是什么。

test1.txt:

```
8 DutyRosterReader.readfromfile("src/dutyreaderfile/test1.txt");
```

```
<terminated> DutyRosterReaderAppTest (1) [Java Application] D:\java\jdk\bin\javaw.exe (2021年7月3日 上午1:12:56 - 上午1:12:56)
```

duty time: 2021-01-10 -- 2021-01-11	Employee [name=ZhangSan, position=Manger, phone=139-0451-0000]
duty time: 2021-01-12 -- 2021-01-20	Employee [name=LiSi, position=Secretary, phone=151-0101-0000]
duty time: 2021-01-21 -- 2021-01-21	Employee [name=WangWu, position=Associate Dean, phone=177-2021-0301]
duty time: 2021-01-22 -- 2021-01-22	Employee [name=ZhaoLiua, position=Professor, phone=138-1920-3912]
duty time: 2021-01-23 -- 2021-01-29	Employee [name=ZhaoLiub, position=Lecturer, phone=138-1921-3912]
duty time: 2021-01-30 -- 2021-01-31	Employee [name=ZhaoLiuc, position=Professor, phone=138-1922-3912]
duty time: 2021-02-01 -- 2021-02-08	Employee [name=ZhaoLiud, position=Lecturer, phone=198-1920-3912]
duty time: 2021-02-09 -- 2021-02-15	Employee [name=ZhaoLieue, position=Professor, phone=178-1920-3912]
duty time: 2021-02-16 -- 2021-02-24	Employee [name=ZhaoLiuf, position=Lecturer, phone=138-1929-3912]
duty time: 2021-02-25 -- 2021-02-28	Employee [name=ZhaoLiug, position=Professor, phone=138-1920-0000]
duty time: 2021-03-01 -- 2021-03-01	Employee [name=ZhaoLiuh, position=AssociateProfessor, phone=138-1929-0000]
duty time: 2021-03-02 -- 2021-03-04	Employee [name=ZhaoLiui, position=Professor, phone=138-1920-0200]
duty time: 2021-03-05 -- 2021-03-05	Employee [name=ZhaoLiu, position=AssociateProfessor, phone=138-1920-0044]
duty time: 2021-03-06 -- 2021-03-06	Employee [name=ZhaoLiuk, position=Professor, phone=188-1920-0000]

The duty time has been fully arranged!

输出和所给的信息完全一致。

test2.txt:

```
8 // DutyRosterReader.readfromfile("src/dutyreaderfile/test1.txt");
9 DutyRosterReader.readfromfile("./src/dutyreaderfile/test2.txt");
```

```
<terminated> DutyRosterReaderAppTest (1) [Java Application] D:\java\jdk\bin\javaw.exe (2021年7月3日 上午1:14:41 - 上午1:14:43)
```

ZhaoLiua
This staff hasn't been added, can't be scheduled!

根据提示信息，我们查看原文件，

```

1 Employee{
2   ZhangSan{Manger,139-0451-0000}
3   LiSi{Secretary,151-0101-0000}
4   WangWu{Associate Dean,177-2021-0301}
5   ZhaoLiu1{Professor,138-1920-3912}
6   ZhaoLiu2{Lecturer,138-1921-3912}
7   ZhaoLiuc{Professor,138-1922-3912}
8   ZhaoLiud{Lecturer,198-1920-3912}
9   ZhaoLiuE{Professor,178-1920-3912}
10  ZhaoLiuf{Lecturer,138-1929-3912}
11  ZhaoLiug{Professor,138-1920-0000}
12  ZhaoLiuH{AssociateProfessor,138-1929-0000}
13  ZhaoLiuI{Professor,138-1920-0200}
14  ZhaoLiuJ{AssociateProfessor,138-1920-0044}
15  ZhaoLiuk{Professor,188-1920-0000}
16 }

```

发现确实在 Employee 中没有输入 ZhaoLiua 的信息

test3.txt:

```

10 //      DutyRosterReader.readfromfile("../src/dutyreaderfile/test3.txt");
11 //      DutyRosterReader.readfromfile("../src/dutyreaderfile/test3.txt");

```

Console x

<terminated> DutyRosterReaderAppTest (1) [Java Application] D:\java\jdk\bin\javaw.exe (2021年7月3日 上午1:16:28 - 上午1:16:30)

ZhaoLiue
This staff hasn't been added, can't be scheduled!

```

9   ZhaoLiuE{Professor,178-192-03912}
10  ZhaoLiuf{Lecturer,138-1929-3912}
11  ZhaoLiug{Professor,138-1920-0000}

```

可以发现 ZhaoLiue 的输入信息格式有问题,故无法匹配成功,即无法录入,所以在为其排班时,找不到其信息,故提示错误,并退出。

test4.txt:

```

10 //      DutyRosterReader.readfromfile("../src/dutyreaderfile/test4.txt");
11 //      DutyRosterReader.readfromfile("../src/dutyreaderfile/test4.txt");
12 //      DutyRosterReader.readfromfile("../src/dutyreaderfile/test5.txt");
13 //      DutyRosterReader.readfromfile("../src/dutyreaderfile/test6.txt");

```

Console x

<terminated> DutyRosterReaderAppTest (1) [Java Application] D:\java\jdk\bin\javaw.exe (2021年7月3日 上午1:16:47 - 上午1:16:49)

duty time	Employee
2021-01-10 -- 2021-01-11	Employee [name=ZhangSan, position=Manger, phone=139-0451-0000]
2021-01-12 -- 2021-01-20	Employee [name=LiSi, position=Secretary, phone=151-0101-0000]
2021-01-21 -- 2021-01-21	Employee [name=WangWu, position=Associate Dean, phone=177-2021-0301]
2021-01-22 -- 2021-01-22	Employee [name=ZhaoLiua, position=Lecturor, phone=138-1920-3912]
2021-01-23 -- 2021-01-29	Employee [name=ZhaoLiub, position=Lecturer, phone=138-1921-3912]
2021-01-30 -- 2021-01-31	Employee [name=ZhaoLiuc, position=Professor, phone=138-1922-3912]
2021-02-01 -- 2021-02-08	Employee [name=ZhaoLiud, position=Lecturer, phone=198-1920-3912]
2021-02-09 -- 2021-02-15	Employee [name=ZhaoLiuE, position=Professor, phone=178-1920-3912]
2021-02-16 -- 2021-02-24	Employee [name=ZhaoLiuf, position=Lecturer, phone=138-1929-3912]
2021-02-25 -- 2021-02-28	Employee [name=ZhaoLiug, position=Professor, phone=138-1920-0000]
2021-03-01 -- 2021-03-01	Employee [name=ZhaoLiuH, position=Associate Professor, phone=138-1929-0000]
2021-03-02 -- 2021-03-04	Employee [name=ZhaoLiuI, position=Professor, phone=138-1920-0200]
2021-03-05 -- 2021-03-05	Employee [name=ZhaoLiuJ, position=AssociateProfessor, phone=138-1920-0044]
2021-03-06 -- 2021-03-06	Employee [name=ZhaoLiuk, position=Professor, phone=188-1920-0000]

The duty time has been fully arranged!

正常构建。

test5.txt:

```
11 //      DutyRosterReader.readfromfile("../src/dutyreaderfile/test4.txt");
12      DutyRosterReader.readfromfile("../src/dutyreaderfile/test5.txt");
13 //      DutyRosterReader.readfromfile("../src/dutyreaderfile/test6.txt");

< Console x
<terminated> DutyRosterReaderAppTest (1) [Java Application] D:\java\jdk\bin\javaw.exe (2021年7月3日 上午1:17:13 - 上午1:17:14)
Intervals can't overlapped!
Intervals can't overlapped!
duty time: 2021-01-10 -- 2021-01-11      Employee [name=ZhangSan, position=Manger, phone=139-0451-0000]
duty time: 2021-01-12 -- 2021-01-21      Employee [name=WangWu, position=Associate Dean, phone=177-2021-0301]
duty time: 2021-01-21 -- 2021-01-21      Employee [name=Zhaoliua, position=Professor, phone=138-1920-3912]
duty time: 2021-01-22 -- 2021-01-22      Employee [name=Zhaoliub, position=Lecturer, phone=138-1921-3912]
duty time: 2021-01-23 -- 2021-01-29      Employee [name=Zhaoliuc, position=Professor, phone=138-1922-3912]
duty time: 2021-02-01 -- 2021-02-08      Employee [name=Zhaoliud, position=Lecturer, phone=198-1920-3912]
duty time: 2021-02-09 -- 2021-02-15      Employee [name=Zhaoliue, position=Professor, phone=178-1920-3912]
duty time: 2021-02-16 -- 2021-02-27      Employee [name=Zhaoliuf, position=Lecturer, phone=138-1929-3912]
duty time: 2021-03-01 -- 2021-03-01      Employee [name=Zhaoliuh, position=Associate Professor, phone=138-1929-0000]
duty time: 2021-03-02 -- 2021-03-04      Employee [name=Zhaoliui, position=Professor, phone=138-1920-0200]
duty time: 2021-03-05 -- 2021-03-05      Employee [name=Zhaoliuj, position=AssociateProfessor, phone=138-1920-0044]
duty time: 2021-03-06 -- 2021-03-06      Employee [name=Zhaoliuk, position=Professor, phone=188-1920-0000]

No arranged interval:
2021-01-12 -- 2021-01-20
2021-02-28 -- 2021-02-28

Blank ratio: 0.17857142857142858
```

有重叠的排班，但是排班表要求的是不能有重叠部分。

test6.txt:

```
13      DutyRosterReader.readfromfile("../src/dutyreaderfile/test6.txt");

< Console x
<terminated> DutyRosterReaderAppTest (1) [Java Application] D:\java\jdk\bin\javaw.exe (2021年7月3日 上午1:18:07 - 上午1:18:09)
duty time: 2021-01-10 -- 2021-01-11      Employee [name=ZhangSan, position=Manger, phone=139-0451-0000]
duty time: 2021-01-12 -- 2021-01-20      Employee [name=LiSi, position=Secretary, phone=151-0101-0000]
duty time: 2021-01-21 -- 2021-01-21      Employee [name=WangWu, position=Associate Dean, phone=177-2021-0301]
duty time: 2021-01-22 -- 2021-01-22      Employee [name=Zhaoliua, position=Professor, phone=138-1920-3912]
duty time: 2021-01-23 -- 2021-01-29      Employee [name=Zhaoliub, position=Lecturer, phone=138-1921-3912]
duty time: 2021-01-30 -- 2021-01-31      Employee [name=Zhaoliuc, position=Professor, phone=138-1922-3912]
duty time: 2021-02-09 -- 2021-02-15      Employee [name=Zhaoliue, position=Professor, phone=178-1920-3912]
duty time: 2021-02-16 -- 2021-02-24      Employee [name=Zhaoliuf, position=Lecturer, phone=138-1929-3912]
duty time: 2021-03-01 -- 2021-03-01      Employee [name=Zhaoliuh, position=Associate Professor, phone=138-1929-0000]
duty time: 2021-03-02 -- 2021-03-04      Employee [name=Zhaoliui, position=Professor, phone=138-1920-0200]
duty time: 2021-03-05 -- 2021-03-05      Employee [name=Zhaoliuj, position=AssociateProfessor, phone=138-1920-0044]
duty time: 2021-03-06 -- 2021-03-06      Employee [name=Zhaoliuk, position=Professor, phone=188-1920-0000]

No arranged interval:
2021-02-01 -- 2021-02-08
2021-02-25 -- 2021-02-28

Blank ratio: 0.21428571428571427
```

没有安排满。

test7.txt:

```
14      DutyRosterReader.readfromfile("../src/dutyreaderfile/test7.txt");
15 //      DutyRosterReader.readfromfile("../src/dutyreaderfile/test8.txt");

< Console x
<terminated> DutyRosterReaderAppTest (1) [Java Application] D:\java\jdk\bin\javaw.exe (2021年7月3日 上午1:18:40 - 上午1:18:42)
Zhaoliue
This staff hasn't been added, can't be scheduled!
```

可以在下图中看到，Employee 中确实没有 Zhaoliue 的信息，即没有被加入系统。


```

1 }
2 Employee{
3   ZhangSan{Manger,139-0451-0000}
4   LiSi{Secretary,151-0101-0000}
5   WangWu{Associate Dean,177-2021-0301}
6   ZhaoLiua{Professor,138-1920-3912}
7   ZhaoLiub{Lecturer,138-1921-3912}
8   ZhaoLiuc{Professor,138-1922-3912}
9   ZhaoLiud{Lecturer,198-1920-3912}
10  ZhaoLiuf{Lecturer,138-1929-3912}
11  ZhaoLiug{Professor,138-1920-0000}
12  ZhaoLiuh{Associate Professor,138-1929-0000}
13  ZhaoLiuj{AssociateProfessor,138-1920-0044}
14  ZhaoLiuk{Professor,188-1920-0000}
15 }

```

test8.txt:

```

14 //      DutyRosterReader.readFromFile("../src/dutyreaderfile/test7.txt");
15      DutyRosterReader.readFromFile("../src/dutyreaderfile/test8.txt");|

```

Console X

<terminated> DutyRosterReaderAppTest (1) [Java Application] D:\java\jdk\bin\javaw.exe (2021年7月3日 上午1:18:57 - 上午1:18:59)

ZhaoLiue
This staff hasn't been added, can't be scheduled!

```

1 Employee{
2   ZhangSan{Manger,139-0451-0000}
3   LiSi{Secretary,151-0101-0000}
4   WangWu{Associate Dean,177-2021-0301}
5   ZhaoLiua{Professor,138-1920-3912}
6   ZhaoLiub{Lecturer,138-1921-3912}
7   ZhaoLiuc{Professor,138-1922-3912}
8   ZhaoLiud{Lecturer,198-1920-3912}
9   ZhaoLiub{Professor,178-1920-3912}
10  ZhaoLiuf{Lecturer,138-1929-3912}
11  ZhaoLiug{Professor,138-1920-0000}
12  ZhaoLiuh{Associate Professor,138-1929-0000}
13  ZhaoLiui{Professor,138-1920-0200}
14  ZhaoLiuj{AssociateProfessor,138-1920-0044}
15  ZhaoLiud{Professor,188-1920-0000}
16 }

```

同样也是 ZhaoLiue 没有被加入系统。

测试完毕，我们将其加入我们的主程序。即 DutyRosterApp。
我们输入 9，在输入想要读入文件的路径。


```

You must firstly see the start and end time of the system can work properly.
9
Input the filepath, which should be src/dutyreaderfile/test1.txt ~ src/dutyreaderfile/test8.txt
src/dutyreaderfile/test1.txt
duty time: 2021-01-10 -- 2021-01-11 Employee [name=ZhangSan, position=Manger, phone=139-0451-0000]
duty time: 2021-01-12 -- 2021-01-20 Employee [name=LiSi, position=Secretary, phone=151-0101-0000]
duty time: 2021-01-21 -- 2021-01-21 Employee [name=WangWu, position=Associate Dean, phone=177-2021-0301]
duty time: 2021-01-22 -- 2021-01-22 Employee [name=ZhaoLiua, position=Professor, phone=138-1920-3912]
duty time: 2021-01-23 -- 2021-01-29 Employee [name=ZhaoLiub, position=Lecturer, phone=138-1921-3912]
duty time: 2021-01-30 -- 2021-01-31 Employee [name=ZhaoLiuc, position=Professor, phone=138-1922-3912]
duty time: 2021-02-01 -- 2021-02-08 Employee [name=ZhaoLiud, position=Lecturer, phone=198-1920-3912]
duty time: 2021-02-09 -- 2021-02-15 Employee [name=ZhaoLieve, position=Professor, phone=178-1920-3912]
duty time: 2021-02-16 -- 2021-02-24 Employee [name=ZhaoLiuf, position=Lecturer, phone=138-1929-3912]
duty time: 2021-02-25 -- 2021-02-28 Employee [name=ZhaoLiug, position=Professor, phone=138-1920-0000]
duty time: 2021-03-01 -- 2021-03-01 Employee [name=ZhaoLiuh, position=AssociateProfessor, phone=138-1929-0000]
duty time: 2021-03-02 -- 2021-03-04 Employee [name=ZhaoLiui, position=Professor, phone=138-1920-0200]
duty time: 2021-03-05 -- 2021-03-05 Employee [name=ZhaoLiuji, position=AssociateProfessor, phone=138-1920-0044]
duty time: 2021-03-06 -- 2021-03-06 Employee [name=ZhaoLiuk, position=Professor, phone=188-1920-0000]

The duty time has been fully arranged!

```

可以发现读入成功。

我们尝试对读入的信息进行操作：

```

8.Delete the arragemet of a staff
9.Arrange the duty time according to the input file
10.Exit

8
Please input the staff's information according to the instructions!
Please input the name:
ZhaoLiuk
Please input the position:
Professor
Please input the phone number:
188-1920-0000

```

删除掉 ZhaoLiuk 的排班信息。

在查看删除后的排班表：

```

7
duty time: 2021-01-10 -- 2021-01-11 Employee [name=ZhangSan, position=Manger, phone=139-0451-0000]
duty time: 2021-01-12 -- 2021-01-20 Employee [name=LiSi, position=Secretary, phone=151-0101-0000]
duty time: 2021-01-21 -- 2021-01-21 Employee [name=WangWu, position=Associate Dean, phone=177-2021-0301]
duty time: 2021-01-22 -- 2021-01-22 Employee [name=ZhaoLiua, position=Professor, phone=138-1920-3912]
duty time: 2021-01-23 -- 2021-01-29 Employee [name=ZhaoLiub, position=Lecturer, phone=138-1921-3912]
duty time: 2021-01-30 -- 2021-01-31 Employee [name=ZhaoLiuc, position=Professor, phone=138-1922-3912]
duty time: 2021-02-01 -- 2021-02-08 Employee [name=ZhaoLiud, position=Lecturer, phone=198-1920-3912]
duty time: 2021-02-09 -- 2021-02-15 Employee [name=ZhaoLieve, position=Professor, phone=178-1920-3912]
duty time: 2021-02-16 -- 2021-02-24 Employee [name=ZhaoLiuf, position=Lecturer, phone=138-1929-3912]
duty time: 2021-02-25 -- 2021-02-28 Employee [name=ZhaoLiug, position=Professor, phone=138-1920-0000]
duty time: 2021-03-01 -- 2021-03-01 Employee [name=ZhaoLiuh, position=AssociateProfessor, phone=138-1929-0000]
duty time: 2021-03-02 -- 2021-03-04 Employee [name=ZhaoLiui, position=Professor, phone=138-1920-0200]
duty time: 2021-03-05 -- 2021-03-05 Employee [name=ZhaoLiuji, position=AssociateProfessor, phone=138-1920-0044]

No arranged interval:
2021-03-06 -- 2021-03-06

```

可以发现排班信息已经被删除了。

3.8 应对面临的新变化

3.8.1 变化 1

评估：根据需求，一个职工可以被安排多段值班，也就是说一个标签可以对

应多个时间段了，所以我们需要修改时间段集合的类型，改用NoBlankMultiIntervalSet、NonOverlapMultiIntervalSet装饰的集合。

而算法中的实现也会随着数据结构的变化，主要体现在循环遍历时，针对每个职工的子时间段集合来遍历。但是总体的思路是不变的，只是实现细节的变化。

如何修改设计以应对变化：

更改存储时间段的集合种类。

```
private NoBlankMultiIntervalSet<Employee> calendar = NoBlankMultiIntervalSet
    .create(NonOverlapMultiIntervalSet.create(MultiIntervalSet.empty()));
```

然后在所有的实现方法中用到循环的地方略作修改即可。

修改的代码量和时间：不到 10 行，20 分钟改完。

下面进行测试：

我们首先设置起止时间：

```
You must firstly set the start and end time! Or the system can't work properly!
1
Please the start time, the format is yyyy-MM-dd:
2021-01-10
Menu:
1.Set the start time of duty interval
2.Set the end time of duty interval
3.Add a staff information
4.Delete a staff information
5.Arrange a staff to be on duty
6.Arrange the duty time table automatically
7.Check the duty time table arrangement situation
8.Delete the arrangemet of a staff
9.Arrange the duty time according to the input file
10.Exit

2
Please the end time, the format is yyyy-MM-dd:
2021-01-30
Menu:
1.Set the start time of duty interval
2.Set the end time of duty interval
3.Add a staff information
4.Delete a staff information
5.Arrange a staff to be on duty
6.Arrange the duty time table automatically
7.Check the duty time table arrangement situation
8.Delete the arrangemet of a staff
9.Arrange the duty time according to the input file
10.Exit
```

然后添加员工：

```

3
Please input the staff's information according to the instructions!
Please input the name:
zhang-san
Please input the position:
teacher
Please input the phone number:
110
If you want to continue to add the staff, please input 1!
1
Please input the staff's information according to the instructions!
Please input the name:
li-si
Please input the position:
teacher
Please input the phone number:
119
If you want to continue to add the staff, please input 1!
0
..

```

为 zhang-san 排多次班:

```

5
Please input the staff's information and time interval according to the instructions!
Please input the name:
zhang-san
Please input the position:
teacher
Please input the phone number:
110
Please input the start time, the format is yyyy-MM-dd:
2021-01-15
Please input the lasting time:
5
Menu:
1.Set the start time of duty interval
2.Set the end time of duty interval
3.Add a staff information
4.Delete a staff information
5.Arrange a staff to be on duty
6.Arrange the duty time table automatically
7.Check the duty time table arrangement situation
8.Delete the arrangemet of a staff
9.Arrange the duty time according to the input file
10.Exit

5
Please input the staff's information and time interval according to the instructions!
Please input the name:
zhang-san
Please input the position:
teacher
Please input the phone number:
110
Please input the start time, the format is yyyy-MM-dd:
2021-01-22
Please input the lasting time:
2

```

查看结果:

```

Menu:
1.Set the start time of duty interval
2.Set the end time of duty interval
3.Add a staff information
4.Delete a staff information
5.Arrange a staff to be on duty
6.Arrange the duty time table automatically
7.Check the duty time table arrangement situation
8.Delete the arrangemet of a staff
9.Arrange the duty time according to the input file
10.Exit

7
duty time: 2021-01-15 -- 2021-01-19      Employee [name=zhang-san, position=teacher, phone=110]
duty time: 2021-01-22 -- 2021-01-23      Employee [name=zhang-san, position=teacher, phone=110]

No arranged interval:
2021-01-10 -- 2021-01-14
2021-01-20 -- 2021-01-21
2021-01-24 -- 2021-01-30

Blank ratio: 0.6666666666666666

```

我们尝试插入重复的时间段，可以发现插入失败：

```

5
Please input the staff's information and time interval according to the instructions!
Please input the name:
li-si
Please input the position:
teacher
Please input the phone number:
119
Please input the start time, the format is yyyy-MM-dd:
2021-01-15
Please input the lasting time:
3
Intervals can't overlapped!
Menu:
1.Set the start time of duty interval
2.Set the end time of duty interval
3.Add a staff information
4.Delete a staff information
5.Arrange a staff to be on duty
6.Arrange the duty time table automatically
7.Check the duty time table arrangement situation
8.Delete the arrangemet of a staff
9.Arrange the duty time according to the input file
10.Exit

7
duty time: 2021-01-15 -- 2021-01-19      Employee [name=zhang-san, position=teacher, phone=110]
duty time: 2021-01-22 -- 2021-01-23      Employee [name=zhang-san, position=teacher, phone=110]

No arranged interval:
2021-01-10 -- 2021-01-14
2021-01-20 -- 2021-01-21
2021-01-24 -- 2021-01-30

Blank ratio: 0.6666666666666666

```

3.8.2 变化 2

评估：可以发现需求变成两门课不能同时在同一时间段内出现，即时间段集合不能有重叠出现，所以我们在设计 CourseSchedule 的 rep 时需要用到 NonOverlapMultiIntervalSet、PeriodicMultiIntervalSet 修饰的集合。

修改：
这是原来的 rep:

```
private PeriodicMultiIntervalSet<Course> courseschedule = PeriodicMultiIntervalSet.create(MultiIntervalSet.empty());
```

修改后的 rep:

```
private PeriodicMultiIntervalSet<Course> courseschedule = PeriodicMultiIntervalSet.  
    .create(NonOverlapMultiIntervalSet.create(MultiIntervalSet.empty()));
```

只需要对原来的集合加上一层装饰即可。

```
try {  
    courseschedule.insert(s, s + 1, courseset.get(course_id));  
} catch (IntervalConflictException e2) {  
    System.out.println("Courses can't overlap!");  
}
```

在安排课程的代码中，捕获重叠的异常，打印提示信息。

修改代码量和时间：5 行、10 分钟

现在我们来进行测试。

首先设置起始时间和周数。

```
You must firstly set the start and total weeks! Or the system can't work properly!  
2  
Please input the starttime of the semester.  
The format is yyyy-MM-dd,and the start day of the semeste should be on Monday!  
Tips: 2021-07-05 is a legal example~  
2021-07-05  
Menu:  
1.add a course to the system  
2.set the start time of the semester  
3.set the total weeks of this semester  
4.arrange a course  
5.check the course schedule  
6.check one day of the schedule  
7.exit  
3  
Please the total weeks of the semester.  
18
```

然后添加两门课程:

```
1  
Please input the course information according to the instructions:  
the format is course_id name teacher teachingposition hoursperweek(each term is delimited by a single blank space)  
Attention: course_id is unique for every course,  
and the hoursperweek should be an even number.  
Tips: cs001 DataStructure ZhangSan ZhengXin201 8 is a legal example~  
cs001 datastructure zhang-san zheng-xin201 8  
Menu:  
1.add a course to the system  
2.set the start time of the semester  
3.set the total weeks of this semester  
4.arrange a course  
5.check the course schedule  
6.check one day of the schedule  
7.exit  
1  
Please input the course information according to the instructions:  
the format is course_id name teacher teachingposition hoursperweek(each term is delimited by a single blank space)  
Attention: course_id is unique for every course,  
and the hoursperweek should be an even number.  
Tips: cs001 DataStructure ZhangSan ZhengXin201 8 is a legal example~  
cs002 programming li-si zheng-xin 4
```

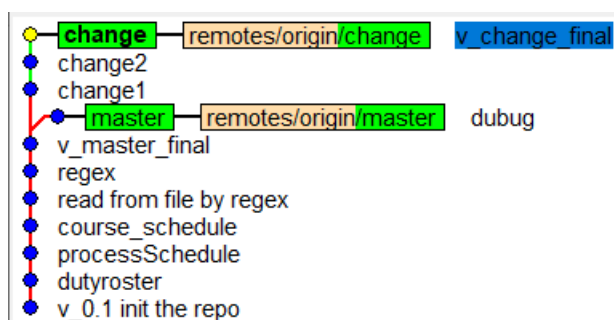
接着将两门课安排在同一时间：

```
Please input the course arrangement information according to the instructions:
the format is course_id weekday start_time(each term is delimited by a single blank space)
the weekday is 1~7, which represents Monday~Sunday
the course could only be scheduled on 8~10/ 10~12/ 13~15/ 15~17/ 19~21, so the start time you input must be 8/10/13/15/19
Tips: cs001 4 8 is a legal example which means cs001 was scheduled on Thursday during 8~10
cs001 1 8
Courses can't overlap!
Arrangement error!
```

可以发现会提示不能重复。

3.9 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。



4 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	8:00-10:00	完成 IntervalSet 的设计	按时完成
2021-06-20	20: 00-24: 00	完成 IntervalSet 三个维度的设计	按时完成
2021-06-24	20: 00-24: 00	完成 MultiIntervalSet 及其三个维度的设计	按时完成

2021-06-28	20: 00-22: 00	完成正则表达式的学习	按时完成
2021-06-030	20: 00-24: 00	完成 DutyRosterApp	按时完成
2021-07-01	20: 00-24: 00	完 成 ProcessSchedule 和 CourseSchedule	按时完成
2021-07-02	20: 00-24: 00	完成文件读入排班表	按时完成
2021-07-03	8: 00-11: 00	完成 change, 撰写报告	按时完成

5 实验过程中遇到的困难与解决途径

遇到的难点	解决途径
设计模式的理解和选用	综合比较接口组合模式和装饰器模式,根据后面需要对功能进行修改的特点,采用了装饰器模式,更有利于后续的修改。
学习装饰器模式	装饰器模式虽然以前利用过 Java 提供的一些 api 里有见过,但是没有自己写过,通过理论学习,设计不同维度的特征,构造继承树。
正则表达式	学习 java 正则表达式的用法, Pattern 和 Matcher 类等。
App 的设计	命令行 app 太不好用了,充分考虑输入的情况
代码量太大了	努力

6 实验过程中收获的经验、教训、感想

6.1 实验过程中收获的经验教训

通过本次的实验,终于彻底理解了可维护性和复用性的重要,如果没有一开始对三个场景进行抽象,提炼出共性,进行顶层接口的设计,那完成这三个任务将会无比的复杂,且有

大量重复的代码。

我在以后的工作中要注意，提前积累代码，不大量重复造轮子，能用以前写好的尽量用，也要学会利用他人的工作成果，尽量去复用前人的成果。

6.2 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？

差异：面向 ADT 编程给我们提供了复用代码的机会，可以在多个场景中利用同一套设计，而面向场景编程是效率十分低下的，只会减低人们的工作效率，一旦需求有所改变，修改起来十分不方便，且没有任何的可移植性和可复用性。换另一个场景，之前写的成果全部不能使用了。

复用的好处：在这样代码量十分庞大的工程中，复用带给我们的好处是显而易见的，每个场景中都能够利用之前已经写好的 ADT，而不需要从头再来，大大提高了效率。

- (2) 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？

提高工作的效率，`spec` 是用于连接用户和实现者的，为我们的开发提供了基础，我们要在 `spec` 的要求下进行开发；`invariants` 是程序中的不变量，保持不变量是程序不出错的基本要求，我们可以通过检查不变量来发现 bug；`RI` 即是表示不变量，我们要求整个程序在运行过程中保持 `RI`，`RI` 即是一组条件，我们要在每次对 `rep` 进行修改后，即调用 `mutator` 方法后检查是否仍然满足 `RI`，如果违反，要 `fail fast`，尽快调整；`rep exposure`，防止表示泄露，不应该暴露给 `client` 的 `rep`，如果暴露了，就有可能造成不应该的修改。

愿意坚持这么做，看似麻烦，但实际上给开发提供了以套有用且高效的流程。

- (3) 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？

开发很难，需要进行大量的抽象，细节很多；使用这些 `api` 可以减少开发

量。

- (4) 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？

正则表达式非常的实用，可以自动去匹配我们想要的子表达式，在学习形式语言与自动机时，只是从理论上学习了正则表达式，但是将其应用在实际场景中，才发现了其理论意义背后的实践意义。

语法驱动的编程可以提高提炼有用信息的效率。

- (5) Lab1 和 Lab2 的工作都不是从 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，经过三周之后，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？

难度：从现实场景中进行抽象，我们要从一组要求中识别什么是名词、什么是动词，分别抽象成属性和方法；如果是多场景的开发，还要抽象不同场景下的共性和个性。

通过多看别人的代码，尤其是习题课老师提供的代码，学习别人是如何抽象的。

- (6) “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。本实验的三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、委派、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？

开发一定要先抽象，否则开发的程序就是面向场景的，一旦要修改或者复用，就很难受，只有将共性先抽象出来，在后面的开发中才能提高效率。

- (7) 关于本实验的工作量、难度、deadline。

工作量很大，难度也很大，deadline 正常。

- (8) 下周就要进行考试了，你对《软件构造》课程总体评价如何？

自己认真写实验会给人带来极大的提升。极大的提高了我们独立完成工程的能力，非常锻炼人。

