



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	傅浩东
学号	1190202105
班号	1903002
电子邮件	1091288450@qq.com
手机号码	13881165621

目录

1 实验目标概述	1
2 实验环境配置	1
3 实验过程	1
3.1 待开发的三个应用场景	1
3.1.1 值班表管理 (DutyRoster)	2
3.1.2 操作系统进程调度管理 (ProcessSchedule)	2
3.1.3 大学课表管理 (CourseSchedule)	2
3.2 面向可复用性和可维护性的设计: IntervalSet<L>	3
3.2.1 IntervalSet<L>的共性操作	3
3.2.2 局部共性特征的设计方案	4
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)	6
3.3 面向可复用性和可维护性的设计: MultiIntervalSet<L>	8
3.3.1 MultiIntervalSet<L>的共性操作	8
3.3.2 局部共性特征的设计方案	9
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)	11
3.3.3.1 面向 Process 进程安排的应用设计	11
3.3.3.2 面向 Course 课表安排的应用设计	13
3.4 面向复用的设计: L	15
3.4.1 员工 (Employee)	15
3.4.2 进程 (Process)	16
3.4.3 课程 (Course)	17
3.5 可复用 API 设计	18
3.5.1 计算相似度	18
3.5.2 计算时间冲突比例	19
3.5.3 计算空闲时间比例	20
3.6 应用设计与开发	21
3.6.1 排班管理系统	21
3.6.2 操作系统的进程调度管理系统	26
3.6.3 课表管理系统	27

3.7 基于语法的数据读入	30
3.8 应对面临的新变化	34
3.8.1 变化 1	34
3.8.2 变化 2	35
3.9 Git 仓库结构	36
4 实验进度记录	37
5 实验过程中遇到的困难与解决途径	38
6 实验过程中收获的经验、教训、感想	38
6.1 实验过程中收获的经验教训	38
6.2 针对以下方面的感受	38

1 实验目标概述

本次实验覆盖课程第 2、3 章的内容，目标是编写具有可复用性和可维护性的软件，主要使用以下软件构造技术：

- 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）。

2 实验环境配置

Java 环境实验一以及配置好，各种工具如 Git 也在实验一安装好。本实验要求安装配置的 EcIEmma（用于统计 Junit 测试代码覆盖度的 plugin）在最新版的 Eclipse 中自带。

我的 GitHub Lab3 仓库的 URL 地址：

<https://github.com/ComputerScienceHIT/HIT-Lab3-1190202105>

3 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！）。

3.1 待开发的三个应用场景

简要介绍三个应用。

分析三个应用场景的异同，理解需求：它们在哪些方面有共性、哪些方面有

差异。

3.1.1 值班表管理 (DutyRoster)

一个单位有 n 个员工，在某个时间段内（例如寒假 1 月 10 日到 3 月 6 日期间），每天只能安排唯一一个员工在单位值班，且不能出现某天无人值班的情况；每个员工若被安排值班 m 天 ($m > 1$)，那么需要安排在连续的 m 天内。值班表内需要记录员工的名字、职位、手机号码，以便于外界联系值班员。

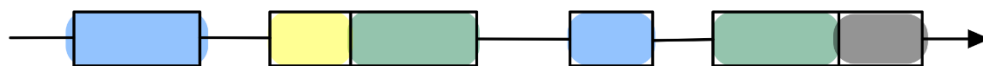
即每个员工值班在一个连续的时间段，不能让一个人出现在多个时间段，并且时间段之间不能有重合，最终的安排要使得在某个期间内每个时间段都有人值班。具体图解如下：



3.1.2 操作系统进程调度管理 (ProcessSchedule)

单核 CPU，操作系统创建多个进程被调度在 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。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 execution 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。

即在一个期间内，插入时间段，每个时间段的标志可以重复，但是不允许有时间段的重合部分，允许有空闲时间。图解如下：



3.1.3 大学课表管理 (CourseSchedule)

课程安排将以“周”为单位进行周期性的重复，直到学期结束，每周课表完全相同，运行同一个时间段有重复课程。

允许有空闲时间、允许不同标签的时间段重复，并且是周期性存储的，每个

周期内完全是相等的内容。如图一个周期内容：



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

为接口 IntervalSet 设计应提供的共性接口方法：创建一个空对象、在当前对象中插入新的时间段和标签、获得当前对象中的标签集合、从当前对象中移除某个标签所关联的时间段、返回某个标签对应的时间段的开始时间、返回某个标签对应的时间段的结束时间、返回全部的时间段安排、改变某个标签、判断是否是空表。这些都放入 IntervalSet<L>接口封装起来。

方法	含义
empty()	创建一个空对象
boolean insert(long start, long end, L label)	插入新的时间段和标签
Set<L> labels()	标签集合
boolean remove(L label)	移除某个标签所关联的时间段
long start (L label)	标签对应的时间段的开始时间
long end (L label)	标签对应的时间段的结束时间
list<Interval<L>> getIntervals()	全部的时间段及标签
boolean changeLabel(L old, L new)	改变某个标签为新标签
boolean isEmpty()	判断是否是空的安排表

CommonIntervalSet<L>是 IntervalSet<L>的具体实现类，实现上述方法。

首先有具体的规约、AF、RI 等：

```
// Abstraction function:
// AF(intervals) = the list of intervals in the interval set

// Representation invariant:
// each interval has a unique label
// the start time of the interval should be <= the end time

// Safety from rep exposure:
// Check the rep invariant is true. That means setting
// list intervals unchangeable / private / final
// and return value MUST be immutable
```

存储结构：采用简单的 List<Interval<L>>存储，每次在插入前要检查是否有重复的标签。

3.2.2 局部共性特征的设计方案

首先，对于方法 `insert` 检测不符合的要求的条件，例如开始时间大于结束时间、存储结构中存在时间段与待插入的标签是相等的，若有上述情况就直接退出返回 `false`；否则就往 `IntervalSet` 中插入该标签以及其对映的时间段。

```
@Override
public boolean insert(long start, long end, L label) throws Exception {
    if (!label().contains(label) && start <= end) {
        Interval<L> newInterval = new Interval<L>(start, end, label);
        intervals.add(newInterval);
        checkRep();
        return true;
    }
    checkRep();
    return false;
}
```

对于获取全部标签的方法，遍历全部时间段，将它们的标签全部插入 `Set` 最终返回。

```
@Override
public Set<L> label() {
    Set<L> labels = new HashSet<L>();
    for (Interval<L> I : intervals) {
        labels.add(I.label());
    }
    return labels;
}
```

根据标签删除时间段集合中的某个时间段，因为标签唯一存在，所以检测 `interval List` 中的全部时间段，若有标签相等则删除并根据删除结果返回 `true` 或者 `false`，具体来说就是调用方法 `removeIf`。

```
@Override
public boolean remove(L label) {
    return intervals.removeIf(interval->interval.label().equals(label));
}
```

根据标签获取某个时间段的开始时间，若是不存在与该标签对映时间段则返回时间为 -1（因为在本实验中，所有时间都是正数，-1 为非法时间）。对获取结束时间也是同理。

```
@Override
public long start(L label) {
    for (Interval<L> I : intervals) {
        if (I.label().equals(label))
            return I.start();
    }
    return -1;
}
```

```

@Override
public long end(L label) {
    for (Interval<L> I : intervals) {
        if (I.label().equals(label))
            return I.end();
    }
    return -1;
}

```

获取全部的时间段及其标签，创建新的关于 `Interval` 的列表，将全部时间段都加载进去，然后将该时间列表返回。

```

@Override
public List<Interval<L>> getIntervals() {
    List<Interval<L>> ret = new ArrayList<Interval<L>>();
    for (Interval<L> I : intervals)
        ret.add(I);
    return ret;
}

```

改变某个标签为新标签，首先检测是否存在旧标签，若不存在则返回 `false`；否则根据插入情况来返回（若是新标签已经存在还是会返回 `false`）。

```

@Override
public boolean changeLabel(L oldlabel, L newlabel) throws Exception {
    if (!this.label().contains(oldlabel))
        return false;
    if (oldlabel.equals(newlabel))
        return true;
    long start = this.start(oldlabel);
    long end = this.end(oldlabel);
    this.remove(oldlabel);
    this.insert(start, end, newlabel);
    return true;
}

```

以下方法简单就不再赘述。

```

@Override
public boolean isEmpty() {
    return intervals.isEmpty();
}

@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append(intervals.toString());
    return sb.toString();
}

```

最终得到测试结果及其覆盖度如下所示：

CommonIntervalSet.java	94.8 %	221	12	233
CommonIntervalSet<L>	94.8 %	221	12	233
checkRep()	78.9 %	30	8	38
changeLabel(L, L)	93.9 %	31	2	33
CommonIntervalSet()	100.0 %	8	0	8
end(L)	100.0 %	22	0	22
getIntervals()	100.0 %	22	0	22
insert(long, long, L)	100.0 %	29	0	29
isEmpty()	100.0 %	4	0	4
label()	100.0 %	23	0	23
remove(L)	100.0 %	6	0	6
start(L)	100.0 %	22	0	22
toString()	100.0 %	13	0	13

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）

易知，例如 IntervalSet 这种不能插入相同标签的类只适用于排班 APP，即在排班系统中使用该类 CommonIntervalSet。

实现方案：个人采用简单的方案是将各特殊操作分别放入底层的应用子类。将针对不同特征取值的具体操作分别放在三个应用的子类中加以实现，针对本实验来说代码量不大，可以轻松地完成，就算是重复的方法代码，也能轻松修改。

在排班系统中还定义了一系列的个性化方法和存储结构，如下所示。首先是规约等：定义了存储排班表的整体开始和结束时间，以及存储职员单独列表。

```
private final IntervalSet<Employee> dutySet = IntervalSet.empty();
private long startTime = -1;
private long endTime = -1;
private final List<Employee> employees = new ArrayList<Employee>();

// Abstraction function:
// AF(dutySet) = the interval set in the duty APP
// AF(startTime) = the start time of the duty period
// AF(endTime) = the end time of the duty period
// AF(employees) = the list of employees in the duty set

// Representation invariant:
// each interval has a unique label
// and each dont allow any blank period

// Safety from rep exposure:
// Check the rep invariant is true. That means setting
// dutySet/employees unchangeable / private / final
// but the start and end time need be changable
// and return value MUST be immutable
```

然后是一些自定义方法：

方法	含义
long GetStartTime()	获取时间表开始时间
long GetEndTime()	获取时间表结束时间
boolean addEmployee(String name, String post, String number)	加入新职员
List<Employee> employees()	获得职员列表
Employee getEmployee(String name)	根据职员名字得到该职员
boolean removeEmployeeName(String name)	根据名字将职员移除职员列表（注意此时要要求职员不能在安排表中）
boolean checkNoBlank()	检测不能有空闲时间
boolean checkNonOverlap()	检查不能有重复时间段
void DutyTable()	输出值班表
void EmptyDutyTable()	输出还没有安排的时间表

它们的规约、声明等如下所示：

```
/**
 * get the start data of this duty table
 * @return long the start data
 */
public long GetStartTime() { ...

/**
 * get the end data of this duty table
 * @return long the end data
 */
public long GetEndTime() { ...

/**
 * add an employee into the employee list
 * @param name String name of this employee
 * @param post String post of this employee
 * @param number long phone number of this employee
 * @return true (as specified by addEmployee),
 * if employee already is in the list return false
 */
public boolean addEmployee(String name, String post, String number) { ...

/**
 * get the list of employees
 * @return employee list
 */
public List<Employee> employees() { ...

/**
 * get the employee according to input name
 * @param name String type of an employee
 * @return the found employee, if there is not such employee, return null
 */
public Employee getEmployee(String name) { ...

/**
 * remove the employee according to input name
 * @param name String name of an employee
 * @return true if the employee was in the list and was removed, false otherwise
 */
public boolean removeEmployeeName(String name) { ...
```

```

/**
 * check if there is any time when no employee is on duty
 * @return true if no blank time between start and end time, false otherwise
 */
public boolean checkNoBlank() { ...

/**
 * check if there has an interval that has repeated part with given interval
 * @param start long start time of the interval
 * @param end long end time of the interval
 * @return true if no overlap, false otherwise
 */
private boolean checkNonOverlap (long start, long end) { ...

/**
 * insert an interval into the list of intervals
 * if start < startTime or end > endTime, remind the client out of bounds
 * if there is no overlap after inserting the interval, just insert
 * otherwise, remind of the client overlap, and don't insert
 * @param start long the start time of the interval
 * @param end long the end time of the interval
 * @param label Employee who is on duty
 * @throws Exception
 */
public void insert(long start, long end, Employee label) throws Exception { ...

/**
 * show the table when there has employee on duty
 */
public void DutyTable() { ...

/**
 * show the table when there are no employees on duty
 */
public void EmptyDutyTable() { ...

```

测试结果以及覆盖度：

▼ DutyIntervalSet.java	98.2 %	481	9	490
▼ DutyIntervalSet	98.2 %	481	9	490
● insert(long, long, Employee)	87.5 %	35	5	40
● checkNoBlank()	93.5 %	58	4	62
● DutyIntervalSet()	100.0 %	17	0	17
● addEmployee(String, String, St	100.0 %	31	0	31
■ checkNonOverlap(long, long)	100.0 %	40	0	40
● DutyTable()	100.0 %	50	0	50
● employees()	100.0 %	11	0	11
● EmptyDutyTable()	100.0 %	157	0	157
● getEmployee(String)	100.0 %	21	0	21
● GetEndTime()	100.0 %	3	0	3
● GetStartTime()	100.0 %	3	0	3
● removeEmployeeName(String)	100.0 %	39	0	39
● SetStartEndTime(long, long)	100.0 %	11	0	11

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

为接口 MultiIntervalSet 设计应提供的共性接口方法：创建一个空对象、创建一个

非空对象、在当前对象中插入新的时间段和标签、获得当前对象中的标签集合、从当前对象中移除某个标签所关联的全部时间段、从当前对象中获取与某个标签所关联的所有时间段、返回全部的时间段安排、判断是否是空表。这些都放入 `MultiIntervalSet<L>` 接口封装起来。

方法	含义
<code>empty()</code>	创建一个空对象
<code>Create(IntervalSet<L>)</code>	创建一个非空对象
<code>boolean insert(long start, long end, L label)</code>	插入新的时间段和标签
<code>Set<L> labels()</code>	标签集合
<code>boolean remove(L label)</code>	移除某个标签所关联的全部时间段
<code>IntervalSet<Integer></code>	获取某个标签所关联的所有时间段
<code>list<Interval<L>> allIntervals()</code>	获取全部的时间段及标签

`CommonMultiIntervalSet<L>` 是 `MultiIntervalSet<L>` 的具体实现类，实现上述方法。首先有具体的规约、AF、RI 等：

```
private final List<Interval<L>> multiIntervalSets = new ArrayList<Interval<L>>();

// Abstraction function:
// AF(multiIntervalSets) = the list of intervals in the MiltiInterval set

// Representation invariant:
// intervals' labels can be repeated
// the start time of the interval should be <= the end time

// Safety from rep exposure:
// Check the rep invariant is true. That means setting
// list intervals unchangeable / private / final
// and return value MUST be immutable
```

存储结构：采用简单的 `List<Interval<L>>` 存储，每次在插入前不用检查是否有重复的标签。

3.3.2 局部共性特征的设计方案

首先，对于方法 `insert` 检测不符合的要求的条件，例如开始时间大于结束时间，若有上述情况就直接退出返回 `false`；否则就往 `MultiIntervalSet` 中插入该标签以及其对映的时间段。（在这里不检测标签重复）

```
@Override
public boolean insert(long start, long end, L label) {
    if (start <= end) {
        Interval<L> e = new Interval<L>(start, end, label);
        multiIntervalSets.add(e);
        return true;
    }
    else return false;
}
```

对于获取全部标签的方法，遍历全部时间段，将它们的标签全部插入 `Set` 最终返回，在这里利用了 `Set` 的插入特性，即重复插入不会出现多个标签。

```

@Override
public Set<L> labels() {
    Set<L> ret = new HashSet<L>();
    for (Interval<L> s : multiIntervalSets) {
        ret.add(s.label());
    }
    return ret;
}

```

根据标签删除时间段集合中的某个时间段，因为标签唯一存在，所以检测 interval List 中的全部时间段，若有标签相等则删除并根据删除结果返回 true 或者 false，具体来说就是调用方法 removeIf。

```

@Override
public boolean remove(L label) {
    int initSize = multiIntervalSets.size();
    multiIntervalSets.removeIf(s->s.label().equals(label));
    if (multiIntervalSets.size() < initSize) return true;
    return false;
}

```

根据输入标签获取全部相关的数据段，并按开始时间进行排序，存储在一个 IntervalSet<L>中返回，并且标签 L 为排序序列，所以这里具体使用的是 Integer。具体来说，首先遍历时间段序列查询与标签相关的时间段，将其插入构建的 IntervalSet 中，在插入的过程中进行比较开始时间，若第一个插入为 0，之后的和 IntervalSet 中每一段时间的开始时间比较，谁的开始时间大谁的序列就加一，比较完成之后就可以得到相应的序列，则完成排序。

```

@Override
public IntervalSet<Integer> intervals(L label) throws Exception {
    IntervalSet<Integer> ret = new CommonIntervalSet<Integer>();
    for (Interval<L> s : multiIntervalSets) {
        if (s.label().equals(label)) {
            long start = s.start();
            long end = s.end();
            Integer IntegerLabel = 0;
            // count label which is Integer, Interval having bigger start's label ++
            for (Interval<Integer> r : ret.getIntervals()) {
                if (r.start() > start)
                    ret.changeLabel(r.label(), r.label()+1);
                else if (r.start() < start)
                    IntegerLabel ++;
            }
            ret.insert(start, end, IntegerLabel);
        }
    }
    return ret;
}

```

将全部的时间段及其序列添加到创建的 List 中，然后返回。

```

@Override
public List<Interval<L>> allIntervals() {
    return multiIntervalSets;
}

```

最后是测试结果以及覆盖度：

▼ CommonMultiIntervalSet.java	100.0 %	158	0	158
▼ CommonMultiIntervalSet<L>	100.0 %	158	0	158
CommonMultiIntervalSet()	100.0 %	8	0	8
allIntervals()	100.0 %	3	0	3
insert(long, long, L)	100.0 %	20	0	20
intervals(L)	100.0 %	80	0	80
labels()	100.0 %	23	0	23
remove(L)	100.0 %	19	0	19

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）

3.3.3.1 面向 Process 进程安排的应用设计

实现方案：个人采用简单的方案是将各特殊操作分别放入底层的应用子类。将针对不同特征取值的具体操作分别放在三个应用的子类中加以实现，针对本实验来说代码量不大，可以轻松地完成，就算是重复的方法代码，也能轻松修改。

在进程系统中还定义了一系列的个性化方法和存储结构，如下所示。首先是规约等：定义了当前时间，以及存储每个进程已运行时间（若运行结束，移除出该 HashMap）。

```
private final MultiIntervalSet<Process> processSet = MultiIntervalSet.empty()
// process and run time
private final Map<Process,Long> processes = new HashMap<Process,Long>();
// update by the methods in this class
public long NowTime = 0;
```

它们的规约以及声明等：

```
/**
 * add a process to the process list
 * @param id int ID of the process
 * @param name string the name of the process
 * @param shortest long Shortest execution time of the process
 * @param longest long longest execution time of the process
 * @return true if the process is added, false otherwise(the id already exists)
 */
public boolean AddProcess(int id, String name, long shortest, long longest) { ...

/**
 * choose a process according to an ID
 * @param id int ID of a process
 * @return Process if the ID matches it, null no such process whose ID's this
 */
public Process ChooseProcess(int id) { ...

/**
 * get a process randomly
 * @param num int a random number to get a process
 * @return Process
 */
public Entry<Process, Long> GetProcess(int num) { ...

/**
 * check if there has an interval that has repeated part with given interval
 * @param start long start time of the interval
 * @param end long end time of the interval
 * @return true if no overlap, false otherwise
 */
private boolean checkNonOverlap (long start, long end) { ...

/**
 * insert a process who execute for period time
 * @param period long the execution time
 * @param label Process the process who executes
 * @throws Exception
 */
public void insert(long period, Process label) throws Exception { ...
```

```

/**
 * don't execute any process
 * @param period long the stop time of the process
 */
public void insertNull(long period) { ...

/**
 * get the run time of the process
 * @param id int ID of the process
 * @return long the run time of the process, -1 if no this process
 */
public long getRunTime(int id) { ...

/**
 * remove the EOR processes from the process Map
 * run time is between shortest and longest
 */
public void removeEORProcess() { ...

/**
 * get the processes that have NOT ended up
 * @return int the number of NOT ended processes
 */
public int UnfinishedProcessNumber() { ...

/**
 * get the processes' schedule
 * @return list
 */
public List<Interval<Process>> getSchedule() { ...

/**
 * get process whosw time from the Langedst execution time is the shortest
 * @return Process found
 */
public Entry<Process, Long> GetShortestProcess() { ...

```

测试结果以及覆盖度：

▼ [J] ProcessIntervalSet.java	<div><div></div></div> 98.6 %	346	5	351
▼ [G] ProcessIntervalSet	<div><div></div></div> 98.6 %	346	5	351
● insert(long, Process)	<div><div></div></div> 94.8 %	55	3	58
■ checkNonOverlap(long, long)	<div><div></div></div> 95.0 %	38	2	40
● AddProcess(int, String, long, lc	<div><div></div></div> 100.0 %	36	0	36
● ChooseProcess(int)	<div><div></div></div> 100.0 %	21	0	21
● GetProcess(int)	<div><div></div></div> 100.0 %	26	0	26
● getRunTime(int)	<div><div></div></div> 100.0 %	30	0	30
● getSchedule()	<div><div></div></div> 100.0 %	4	0	4
● GetShortestProcess()	<div><div></div></div> 100.0 %	49	0	49
● insertNull(long)	<div><div></div></div> 100.0 %	11	0	11
● removeEORProcess()	<div><div></div></div> 100.0 %	58	0	58
● UnfinishedProcessNumber()	<div><div></div></div> 100.0 %	4	0	4

3.3.3.2 面向 Course 课表安排的应用设计

在课表系统中还定义了一系列的个性化方法和存储结构，如下所示。首先是规约等：定义了学期开始时间、学期结束时间、学期周数，以及存储每门课程还

没有安排的时间（若安排结束 value 为 0，移除该 HashMap）。

```
// private final MultiIntervalSet
private final long SemesterStart;
private final int WeekNumber;
private final long SemesterEnd;
// Course, WeekHour(left hours)
private final Map<Course,Integer> courses = new HashMap<>();
private final MultiIntervalSet<Course> curriculum = MultiIntervalSet.empty();
```

自定义方法如下规约和声明（注意这里的两个比例计算方法与下述 APIs 设计是不同的，所以重新书写）：

```
/**
 * add a NOT setted course
 * @param id the id of the course
 * @param CourseName the name of the course
 * @param teacher the teacher's name of the course
 * @param place the place of the course
 * @param WeekHour how many hours per week of the course
 * @return
 */
public boolean AddCourse(int id, String CourseName, String teacher, String place, int WeekHour);

/**
 * get when this semester start
 * @return the start date
 */
public long GetSemesterStart() {
    return SemesterStart;
}

/**
 * the date input is in which week of the semester
 * @param date input date
 * @return the week number of the semester
 */
public int GetWeekNumber(String date) { ...

/**
 * get one day's course list according to the date input
 * @param date the date wantted to get
 * @return the list of courses of this day
 * null if the day is not in the semester
 */
public List<Interval<Course>> GetOneDayCourseList(String date) { ...

/**
 * choose a course according to its ID
 * @param id int the course ID
 * @return Course if it is found, null otherwise
 */
public Course GetCourse(int id) { ...
```

```

/**
 * set curriculum
 * @param start the start time of the course
 * @param end the end time of the course
 * @param id the course ID
 * @return true if inserted successfully, false illegal input
 */
public boolean InsertCurriculum(long start, long end, int id) { ...

/**
 * not setted courses
 * @return map, can be a new one
 */
public Map<Course,Integer> GetLeftNOTSetCourse() { ...

/**
 * get the week day of the date
 * @param date the date that is waiting to be check
 * @return 0-6 symbol From Monday to Sunday, -1 if the date is not in semester
 */
public int GetDayFromStart(String date) { ...

/**
 * get conflict ratio
 * @return conflict ratio above 0 below 1
 */
public double GetConflictRatio() { ...

/**
 * get free time ratio
 * @return free time ratio above 0 below 1, the Total time is certain
 */
public double GetFreeTimeRatio() { ...

```

测试结果及覆盖度：

CourseIntervalSet.java	93.4 %	478	34	512
CourseIntervalSet	93.4 %	478	34	512
GetConflictRatio()	79.8 %	83	21	104
GetOneDayCourseList(String)	94.4 %	84	5	89
InsertCurriculum(long, long, int)	95.5 %	84	4	88
AddCourse(int, String, String, String)	95.6 %	43	2	45
GetFreeTimeRatio()	96.9 %	62	2	64
CourseIntervalSet(long, int)	100.0 %	31	0	31
GetCourse(int)	100.0 %	21	0	21
GetDayFromStart(String)	100.0 %	26	0	26
GetLeftNOTSetCourse()	100.0 %	10	0	10
GetSemesterStart()	100.0 %	3	0	3
GetWeekNumber(String)	100.0 %	31	0	31

3.4 面向复用的设计：L

3.4.1 员工（Employee）

在排班系统中关心的属性应该为“员工“，包括员工的姓名、职务、手机号码，用一个 Employee 类来将这些都包括在里面。如下是这个类的规约、存储和方法及测试：

```

// fields, employee's name, post and phone number
private final String name;
private final String post;
private final String number;

// Abstraction function:
// AF(name) = name of the employee
// AF(post) = post of the employee
// AF(munber) = phone number of the employee

// Representation invariant:
// phone number consists of numbers

// Safety from rep exposure:
// Check the rep invariant is true
// All fields MUST be private (all the String type are immutable)
// So make defensive copies instead of just return mutable data

```

方法/Constructor	含义
Employee(String name, String post, String number)	构建一个职员，增加其信息
String GetName()	返回职员姓名
String GetPost()	返回职员职位
String GetNumber()	返回职员电话号码
boolean equals(Object o)	（重写）比较是否是同一人，以名字作判断条件
String toString()	（重写）将职员信息转为String 型返回

3.4.2 进程（Process）

在进程安排系统中关心的属性应该为“进程“，包括进程 ID、进程名称、最短执行时间、最长执行时间，用一个 Process 类来将这些都包括在里面。如下是这个类的规约、存储和方法及测试：

```

// fields, process' ID, name, Shortest and Longest execution time
private final int ID;
private final String name;
private final long shortestTime;
private final long longestTime;

// Abstraction function:
// AF(ID) = Id of the process
// AF(name) = name of the process
// AF(shortestTime) = the Shortest execution time of the process
// AF(longestTime) = the Longest execution time of the process

// Representation invariant:
// ID must > 0
// the Longest execution time MUST be longer than the shortest

// Safety from rep exposure:
// Check the rep invariant is true
// All fields MUST be private (all the String type are immutable)
// So make defensive copies instead of just return mutable data

```

方法/Constructor	含义
Process(int ID, String name, long shortest, long longest)	构建一个进程，增加其信息
Int getID()	返回进程唯一 ID
String GetName()	返回进程姓名
long GetShortestTime()	返回进程最短运行时间
long GetLongestTime()	返回进程最长运行时间
boolean equals(Object o)	（重写）比较是否是同一进程，以 ID 作判断条件
String toString()	（重写）将进程信息转为 String 型返回

3.4.3 课程（Course）

在课程安排系统中关心的属性应该为“课程“，包括课程 ID、课程名称、教师名字、上课地点，用一个 Course 类来将这些都包括在里面。如下是这个类的规约、存储和方法及测试：

```

// fields, process' ID, name, Shortest and Longest execution time
private final int ID;
private final String name;
private final String teacherName;
private final String place;

// Abstraction function:
// AF(ID) = the ID of the course
// AF(name) = the name of the course
// AF(teacherName) = the name of the course's teacher
// AF(place) = the place of the course

// Representation invariant:
// ID must > 0, and unique
// the name of different courses can be repeated

// Safety from rep exposure:
// Check the rep invariant is true
// All fields MUST be private (all the String type are immutable)
// So make defensive copies instead of just return mutable data

```

方法/Constructor	含义
Process(int ID, String name, String teacher, String place)	构建一门课程，增加其信息
Int getID()	返回课程唯一 ID
String GetName()	返回课程姓名
String GetTeacherName()	返回课程教师名字
long GetPlace()	返回课程上课地点
boolean equals(Object o)	（重写）比较是否是同一课程，以 ID 作判断条件
String toString()	（重写）将课程信息转为 String 型返回

3.5 可复用 API 设计

3.5.1 计算相似度

要计算相似度，首先定义实现了两个方法，主方法为 `double Similarity(MultiIntervalSet<L> s1, MultiIntervalSet<L> s2)`，辅助方法有 `long getRepeatedCount(Interval<L> i1, Interval<L> i2)`。前者是计算相似度的主要过程，后者则是就算某两个时间段的重复区间长度。

首先来看 `getRepeatedCount` 方法，两个时间段若是标签不相等则认为它们没有重复区间，否则比较它们的开始时间和结束时间，若是某一方的开始时间大于另一方的结束时间，则没有重复段，否则用公式 $(\text{Max}(\text{end1}, \text{end2}) - \text{Min}(\text{start1}, \text{start2})) - \text{abs}(\text{start1} - \text{start2}) - \text{abs}(\text{end1} - \text{end2})$ 来计算重复时间段。

接着来分析主方法 `Similarity`，对于两个 `MultiIntervalSet<L>` 类，可以获得它们的全部时间段，接着，分别获得它们整体的起始时间和结束时间，用每一个 `s1` 中的 `Interval` 去与每一个 `s2` 中的 `Interval` 作比较获得重复时间段。累加除以整体的时间段就可以获得相似度。

```
public static <L> double Similarity(MultiIntervalSet<L> s1, MultiIntervalSet<L>
    List<Interval<L>> intervals1 = s1.allIntervals();
    List<Interval<L>> intervals2 = s2.allIntervals();
    Collections.sort(intervals1);
    Collections.sort(intervals2);
    long start1 = intervals1.get(0).start();
    long start2 = intervals2.get(0).start();
    long startTime = start1 <= start2 ? start1 : start2;
    long end1 = intervals1.get(0).end();
    for (Interval<L> I1 : intervals1) {
        if (I1.end() > end1) end1 = I1.end();
    }
    long end2 = intervals2.get(0).end();
    for (Interval<L> I2 : intervals2) {
        if (I2.end() > end2) end2 = I2.end();
    }
    long endTime = end1 >= end2 ? end1 : end2;
    double ret = 0.0;
    long sum = 0;
    for (Interval<L> I1 : intervals1) {
        for (Interval<L> I2 : intervals2) {
            sum += getRepeatCount(I1, I2);
        }
    }
    ret = (double)sum / (double)(endTime - startTime);
    return ret;
}
```

3.5.2 计算时间冲突比例

计算时间冲突比例，这里定义了两个方法：`double calcConflictRatio(IntervalSet<L> set)`和 `double calcConflictRatio(MultiIntervalSet<L> set)`，但是它们的思想与实现几乎都是一样的，所以这里就只解释一个方法即可。

对于 `double calcConflictRatio(MultiIntervalSet<L> set)` 首先获取 `MultiIntervalSet<L>` 的整体开始时间与结束时间，按照时间段开始时间从小到大排序之后用前面的时间段与后面的时间段进行重复计算，累加之后除以整体的结束时间减去开始时间的差，就可以得到时间冲突比例。

```

public static <L>double calcConflictRatio(MultiIntervalSet<L> set) {
    List<Interval<L>> list = set.allIntervals();
    Collections.sort(list);
    long conf = 0;
    int size = list.size();
    if (size == 0) return 0;
    for (int i = 0; i < size; i++) {
        int temp = 1;
        while(i + temp < size && list.get(i+temp).start() < list.get(i).end()) {
            if (list.get(i).end() > list.get(i+temp).end())
                conf += (list.get(i+temp).end() - list.get(i+temp).start());
            else conf += (list.get(i).end() - list.get(i+temp).start());
            temp++;
        }
    }
    long start = list.get(0).start();
    long end = Long.MIN_VALUE;
    for (int i = 0; i < size; i++) {
        long temp = list.get(i).end();
        end = end > temp ? end : temp;
    }
    return (double)conf / (double)(end - start);
}

```

3.5.3 计算空闲时间比例

与冲突时间比例一样，这里定义了两种计算空闲时间比例的方法：`double calcFreeTimeRatio(IntervalSet<L> set)` `double calcFreeTimeRatio(MultiIntervalSet<L> set)`。

对于 `double calcFreeTimeRatio(MultiIntervalSet<L> set)`解析：只需要在开始时间从小到大排序之后计算连续时间段之间的空闲部分，累加除以整体的结束时间减去开始时间的差，就可以得到时间冲突比例。

```

public static <L>double calcFreeTimeRatio(MultiIntervalSet<L> set) {
    List<Interval<L>> list = set.allIntervals();
    Collections.sort(list);
    long free = 0;
    int size = list.size();
    if (size == 0) return 1;
    for (int i = 0; i < size; i++) {
        if (i + 1 < size && list.get(i+1).start() > list.get(i).end()) {
            free += list.get(i+1).start() - list.get(i).end();
        }
    }
    long start = list.get(0).start();
    long end = 0;
    for (int i = 0; i < size; i++) {
        long temp = list.get(i).end();
        end = end > temp ? end : temp;
    }
    return (double)free / (double)(end - start);
}

```

测试结果及覆盖度如下所示：

API	97.0 %	622	19	641
APIs.java	97.0 %	622	19	641
APIs<L>	97.0 %	622	19	641
calcFreeTimeRatio(IntervalSet<L>)	95.7 %	89	4	93
calcFreeTimeRatio(MultiIntervalSet<L>)	95.7 %	89	4	93
Similarity(MultiIntervalSet<L>, MultiIntervalSet<L>)	97.7 %	127	3	130
calcConflictRatio(IntervalSet<L>, IntervalSet<L>)	98.5 %	128	2	130
calcConflictRatio(MultiIntervalSet<L>, MultiIntervalSet<L>)	98.5 %	128	2	130
getRepeatCount(Interval<L>, IntervalSet<L>)	98.4 %	61	1	62

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

按照步骤，每个功能设计一个方法，其实就是每个步骤调用的 `DutyIntervalSet` 方法的组合。注意这里没有进行健壮性测试，如果实现是手动那么需要先进行一二步即选择 0 和 1，否则就是从文件中全自动读入。

1. 设定排班开始日期、结束日期 (yyyy-MM-dd)。
2. 增加一组员工 (名字、职务、手机号码)，可删除未被编排进排班表的员工，从排班信息删掉之后才能删除该员工。员工信息一旦设定则无法修改。
3. 可手工选择某个员工、某个时间段，向排班表增加一条排班记录，该步骤可重复执行多次。随时展示给用户排班表中哪些时间段未安排、未安排的时间段占总时间段的比例。
4. 可自动编排的方法，随机生成排班表。
5. 可视化展示任意时刻的排班表。

试运行结果如下：


```

-----
--      0.Set start and end date      --
--      1.Add employees                --
--      2.Set duty                     --
--      3.Set duty randomly            --
--      4.Show duty table              --
--      5.Delete Employees             --
--      6.Set from .txt                --
--      others.quit                    --
-----
YOUR CHOICE:
1
Employ Name:
a
Employ Duty:
duty1
Employ Phone Number:
123434
Continue Add Employ? Y(y)/N(others)
y
Employ Name:
b
Employ Duty:
duty2
Employ Phone Number:
8765432
Continue Add Employ? Y(y)/N(others)
y
Employ Name:
c
Employ Duty:
duty3
Employ Phone Number:
456787865
Continue Add Employ? Y(y)/N(others)
y
Employ Name:
d
Employ Duty:
duty4
Employ Phone Number:
7890345

-----
--      0.Set start and end date      --
--      1.Add employees                --
--      2.Set duty                     --
--      3.Set duty randomly            --
--      4.Show duty table              --
--      5.Delete Employees             --
--      6.Set from .txt                --
--      others.quit                    --
-----
YOUR CHOICE:
0
Input start data (yyyy-mm-dd):
2021-02-01
Input end data (yyyy-mm-dd):
2021-03-12

```

```
-----
--      0.Set start and end date      --
--      1.Add employees                --
--      2.Set duty                     --
--      3.Set duty randomly            --
--      4.Show duty table              --
--      5.Delete Employees             --
--      6.Set from .txt                --
--      others.quit                    --
-----
YOUR CHOICE:
3
-----
--      0.Set start and end date      --
--      1.Add employees                --
--      2.Set duty                     --
--      3.Set duty randomly            --
--      4.Show duty table              --
--      5.Delete Employees             --
--      6.Set from .txt                --
--      others.quit                    --
-----
YOUR CHOICE:
4
Duty Table:
2021-02-01 2021-02-24 a duty1 123434
2021-02-24 2021-02-28 b duty2 8765432
2021-02-28 2021-03-04 c duty3 456787865
2021-03-04 2021-03-12 d duty4 7890345
```

如上所示，是自动排班功能，只需要设置起始时间和结束时间以及设置员工，再选择自动排班就可以得到如图所示结果，该结果是符合要求地一个排班。下面有一个手动排班过程及其结果，还有一些异常处理：

```
-----
--      0.Set start and end date      --
--      1.Add employees               --
--      2.Set duty                    --
--      3.Set duty randomly           --
--      4.Show duty table             --
--      5.Delete Employees            --
--      6.Set from .txt               --
--      others.quit                   --
-----
YOUR CHOICE:
2
Empty Duty Table:
2021-03-20 2021-04-05
Choose an employee:
a
Date Form (yyyy-mm-dd):
2021-03-20
2021-03-29
Continue Set Duty? Y(y)/N(others)
y
Empty Duty Table:
2021-03-29 2021-04-05
Choose an employee:
b
Date Form (yyyy-mm-dd):
2021-03-29
2021-04-03
Continue Set Duty? Y(y)/N(others)
y
Empty Duty Table:
2021-04-03 2021-04-05
Choose an employee:
c
Date Form (yyyy-mm-dd):
2021-04-03
2021-04-05
Continue Set Duty? Y(y)/N(others)
n
```

```

-----
--      0.Set start and end date      --
--      1.Add employees                --
--      2.Set duty                     --
--      3.Set duty randomly            --
--      4.Show duty table              --
--      5.Delete Employees             --
--      6.Set from .txt                --
--      others.quit                    --
-----

```

YOUR CHOICE:

2

Empty Duty Table:

FULL DUTY TABLE!!

Choose an employee:

a

Date Form (yyyy-mm-dd):

2021-03-30

2021-04-02

aOverlap!

Continue Set Duty? Y(y)/N(others)

n

```

-----
--      0.Set start and end date      --
--      1.Add employees                --
--      2.Set duty                     --
--      3.Set duty randomly            --
--      4.Show duty table              --
--      5.Delete Employees             --
--      6.Set from .txt                --
--      others.quit                    --
-----

```

YOUR CHOICE:

4

Duty Table:

2021-03-20 2021-03-29 a duty1 123456

2021-03-29 2021-04-03 b duty2 567890

2021-04-03 2021-04-05 c duty3 456789

```

-----
--      0.Set start and end date      --
--      1.Add employees                --
--      2.Set duty                     --
--      3.Set duty randomly            --
--      4.Show duty table              --
--      5.Delete Employees             --
--      6.Set from .txt                --
--      others.quit                    --
-----

```

YOUR CHOICE:

4

Duty Table:

2021-03-20 2021-03-29 a duty1 123456

2021-03-29 2021-04-03 b duty2 567890

2021-04-03 2021-04-05 c duty3 456789

3.6.2 操作系统的进程调度管理系统

1. 增加一组进程（ID、名称、最短执行时间、最长执行时间）。
2. “随机选择进程”的模拟策略：从 0 开始启动模拟调度，随机地选择某个（小于等于一个）尚未执行结束进程执行，并在该进程最大时间之前的任意时刻停止执行，如果本次及其之前的累积执行时间已落到最短执行时间，最长执行时间的区间内，则该进程被设定为“执行结束”。重复直到所有进程结束。
3. “最短进程优先”的模拟策略：每次选择进程的时候，优先选择距离其最大执行时间差距最小的进程。
4. 可视化展示当前时刻之前的进程调度结果。

试运行结果如下，有两种自动实现方法（完全随机和最短优先）：

```
Add a group of processes.
ID:
1
Name:
a
Shortest execution time:
12
Longest execution time:
34
Add process continuously? Y(y)/N(others)
y
ID:
2
Name:
b
Shortest execution time:
23
Longest execution time:
34
Add process continuously? Y(y)/N(others)
y
ID:
3
Name:
c
Shortest execution time:
21
Longest execution time:
45
Add process continuously? Y(y)/N(others)
n
-----
--          0.Schedule Processes Randomly          --
--    1.Schedule Processes Shortest Process Next    --
-----
YOUR CHOICE:
0
Processes Schedule Results
Num    start    end    Name          ID    Shortest    Longest
0      1289     1316    b              2      23         34
1      1316     1348    a              1      12         34
2      2201     2239    c              3      21         45
```

```

Add a group of processes.
ID:
1
Name:
a
Shortest execution time:
12
Longest execution time:
36
Add process continuously? Y(y)/N(others)
y
ID:
2
Name:
b
Shortest execution time:
24
Longest execution time:
30
Add process continuously? Y(y)/N(others)
y
ID:
3
Name:
c
Shortest execution time:
45
Longest execution time:
56
Add process continuously? Y(y)/N(others)
n
-----
--          0.Schedule Processes Randomly          --
--    1.Schedule Processes Shortest Process Next    --
-----
YOUR CHOICE:
1
Processes Schedule Results
Num    start    end    Name          ID    Shortest    Longest
0      878      907    b              2      24          30
1      907      933    a              1      12          36
2      933      971    c              3      45          56
3     6048     6063    c              3      45          56

```

3.6.3 课表管理系统

1. 设定学期开始日期（年月日）和总周数。
2. 增加一组课程（课程 ID、课程名称、教师名字、地点、周学时数（偶数））。
3. 手工选择某个课程、上课时间（8-10 时、10-12 时、13-15 时、15-17 时、19-21 时），为其安排一次课，每次课的时间长度为 2 小时；可重复安排，直到达到周学时数目时该课程不能再安排。
4. 随时可查看哪些课程没安排、当前每周的空闲时间比例、重复时间比例。
5. 可查看本学期内任意一天的课表结果。

试运行结果如下：

```
--- Input Start date (yyyy-MM-dd) and number of weeks of this semester ---
2021-02-22
18
--- Add a group of courses ---
Left NOT setted Courses:
Conflict Ratio: 0.0
Free Time Ratio: 1.0
Course ID:
1
Course Name:
software
Teacher's name:
liuming
Course place:
zhengxin
Hours per week:
4
Add courses continuously? Y(y)/N(others)
y
Left NOT setted Courses:
[ID:1, Name:software, Teacher:liuming, Place:zhengxin] 4
Conflict Ratio: 0.0
Free Time Ratio: 1.0
Course ID:
2
Course Name:
csapp
Teacher's name:
zhengguibin
Course place:
zhizhi
Hours per week:
4
Add courses continuously? Y(y)/N(others)
y
Left NOT setted Courses:
[ID:2, Name:csapp, Teacher:zhengguibin, Place:zhizhi] 4
[ID:1, Name:software, Teacher:liuming, Place:zhengxin] 4
Conflict Ratio: 0.0
Free Time Ratio: 1.0
Course ID:
3
Course Name:
english
Teacher's name:
lisa
Course place:
zhengxin
Hours per week:
2
Add courses continuously? Y(y)/N(others)
n
```

```

2
Choose a period: 0[8-10h], 1[10-12], 2[13-15], 3[15-17], 4[19-21]
1
Left NOT setted Courses:
[ID:2, Name:csapp, Teacher:zhengguibin, Place:zhizhi] 4
[ID:3, Name:english, Teacher:lisa, Place:zhengxin] 2
[ID:1, Name:software, Teacher:liuming, Place:zhengxin] 2
Conflict Ratio: 0.0
Free Time Ratio: 0.9714285714285714
Choose Course according to ID:
1
Choose weekday: 1-7
4
Choose a period: 0[8-10h], 1[10-12], 2[13-15], 3[15-17], 4[19-21]
1
Left NOT setted Courses:
[ID:2, Name:csapp, Teacher:zhengguibin, Place:zhizhi] 4
[ID:3, Name:english, Teacher:lisa, Place:zhengxin] 2
Conflict Ratio: 0.0
Free Time Ratio: 0.9428571428571428
Choose Course according to ID:
3
Choose weekday: 1-7
1
Choose a period: 0[8-10h], 1[10-12], 2[13-15], 3[15-17], 4[19-21]
1
Left NOT setted Courses:
[ID:2, Name:csapp, Teacher:zhengguibin, Place:zhizhi] 4
Conflict Ratio: 0.0
Free Time Ratio: 0.9142857142857143
Choose Course according to ID:
2
Choose weekday: 1-7
1
Choose a period: 0[8-10h], 1[10-12], 2[13-15], 3[15-17], 4[19-21]
0
Left NOT setted Courses:
[ID:2, Name:csapp, Teacher:zhengguibin, Place:zhizhi] 2
Conflict Ratio: 0.0
Free Time Ratio: 0.8857142857142857
Choose Course according to ID:
2
Choose weekday: 1-7
3
Choose a period: 0[8-10h], 1[10-12], 2[13-15], 3[15-17], 4[19-21]
0

Watch the Curriculum of one day(yyyy-mm-dd):
Which day you want to watch
2021-02-22
Start          End          Course
2021-02-22 08    2021-02-22 10    [ID:2, Name:csapp, Teacher:zhengguibin, Place:zhizhi]
Start          End          Course
2021-02-22 10    2021-02-22 12    [ID:3, Name:english, Teacher:lisa, Place:zhengxin]
Watch another day? Y(y)/N(others):
y
Which day you want to watch
2022-01-14
Out of Semester!

```


3.7 基于语法的数据读入

修改“排班管理”应用以扩展该功能。

1. 首先，每个文件中前面的空格不能识别，所以先将其删除；
2. 定义多个正则表达式，用来逐行识别内容，若是与 `Employee` 匹配上则开始添加职员，此时需要注意职员名字只能由字母组成，电话号码改成 3-4-4、2-5-4 等多个形式都成立；若是与 `Period` 匹配上则设置起始结束时间；在这两个之后与 `Roster` 匹配上则开始设置值班表，每个职员都要到上述的职员名单中找，若是找不到则直接跳过。

```
-----
--      0.Set start and end date      --
--      1.Add employees                --
--      2.Set duty                     --
--      3.Set duty randomly            --
--      4.Show duty table              --
--      5.Delete Employees             --
--      6.Set from .txt                --
--      others.quit                    --
-----
YOUR CHOICE:
6
Choose a file(1-8, otherwise quit):
1
Duty Table:
2021-01-10 2021-01-11 ZhangSan Manger 139-0451-0000
2021-01-12 2021-01-20 LiSi Secretary 151-0101-0000
2021-01-21 2021-01-21 WangWu Associate Dean 177-2021-0301
2021-01-22 2021-01-22 ZhaoLiua Professor 138-1920-3912
2021-01-23 2021-01-29 ZhaoLiub Lecturer 138-1921-3912
2021-01-30 2021-01-31 ZhaoLiuc Professor 138-1922-3912
2021-02-01 2021-02-08 ZhaoLiud Lecturer 198-1920-3912
2021-02-09 2021-02-15 ZhaoLiue Professor 178-1920-3912
2021-02-16 2021-02-24 ZhaoLiuf Lecturer 138-1929-3912
2021-02-25 2021-02-28 ZhaoLiug Professor 138-1920-0000
2021-03-01 2021-03-01 ZhaoLiuh AssciateProfessor 138-1929-0000
2021-03-02 2021-03-04 ZhaoLiui Professor 138-1920-0200
2021-03-05 2021-03-05 ZhaoLiuj AssciateProfessor 138-1920-0044
2021-03-06 2021-03-06 ZhaoLiuk Professor 188-1920-0000
Free Ratio: 0.0
```

```

-----
--      0.Set start and end date      --
--      1.Add employees                --
--      2.Set duty                     --
--      3.Set duty randomly            --
--      4.Show duty table              --
--      5.Delete Employees             --
--      6.Set from .txt                --
--      others.quit                    --
-----
YOUR CHOICE:

6
Choose a file(1-8, otherwise quit):
2
Employee ZhaoLiua does not exist!
Employee ZhaoLiub does not exist!
Duty Table:
2021-01-10 2021-01-11 ZhangSan Manger 139-0451-0000
2021-01-12 2021-01-20 LiSi Secretary 151-0101-0000
2021-01-21 2021-01-21 WangWu Associate Dean 177-2021-0301
2021-01-30 2021-01-31 ZhaoLiuc Professor 138-1922-3912
2021-02-01 2021-02-08 ZhaoLiud Lecturer 198-1920-3912
2021-02-09 2021-02-15 ZhaoLieu Professor 178-1920-3912
2021-02-16 2021-02-24 ZhaoLiuf Lecturer 138-1929-3912
2021-02-25 2021-02-28 ZhaoLiug Professor 138-1920-0000
2021-03-01 2021-03-01 ZhaoLiuH AssciateProfessor 138-1929-0000
2021-03-02 2021-03-04 ZhaoLiuI Professor 138-1920-0200
2021-03-05 2021-03-05 ZhaoLiuJ AssciateProfessor 138-1920-0044
2021-03-06 2021-03-06 ZhaoLiuk Professor 188-1920-0000
Free Ratio: 0.14285714285714285
-

-----
--      0.Set start and end date      --
--      1.Add employees                --
--      2.Set duty                     --
--      3.Set duty randomly            --
--      4.Show duty table              --
--      5.Delete Employees             --
--      6.Set from .txt                --
--      others.quit                    --
-----
YOUR CHOICE:

6
Choose a file(1-8, otherwise quit):
3
Duty Table:
2021-01-10 2021-01-11 ZhangSan Manger 139-0451-0000
2021-01-12 2021-01-20 LiSi Secretary 151-0101-0000
2021-01-21 2021-01-21 WangWu Associate Dean 177-2021-0301
2021-01-22 2021-01-22 ZhaoLiua Professor 138-1920-3912
2021-01-23 2021-01-29 ZhaoLiub Lecturer 138-1921-3912
2021-01-30 2021-01-31 ZhaoLiuc Professor 138-1922-3912
2021-02-01 2021-02-08 ZhaoLiud Lecturer 198-1920-3912
2021-02-09 2021-02-15 ZhaoLieu Professor 178-192-03912
2021-02-16 2021-02-24 ZhaoLiuf Lecturer 138-1929-3912
2021-02-25 2021-02-28 ZhaoLiug Professor 13-81920-0000
2021-03-01 2021-03-01 ZhaoLiuH AssciateProfessor 138-1929-0000
2021-03-02 2021-03-04 ZhaoLiuI Professor 138-1920-0200
2021-03-05 2021-03-05 ZhaoLiuJ AssciateProfessor 138-1920-0044
2021-03-06 2021-03-06 ZhaoLiuk Professor 188-1920-0000
Free Ratio: 0.0

```

```

-----
--      0.Set start and end date      --
--      1.Add employees                --
--      2.Set duty                     --
--      3.Set duty randomly            --
--      4.Show duty table              --
--      5.Delete Employees             --
--      6.Set from .txt                --
--      others.quit                    --
-----
YOUR CHOICE:
6
Choose a file(1-8, otherwise quit):
4
Duty Table:
2021-01-10 2021-01-11 ZhangSan Manger 139-0451-0000
2021-01-12 2021-01-20 LiSi Secretary 151-0101-0000
2021-01-21 2021-01-21 WangWu Associate Dean 177-2021-0301
2021-01-22 2021-01-22 ZhaoLiua Professor 138-1920-3912
2021-01-23 2021-01-29 ZhaoLiub Lecturer 138-1921-3912
2021-01-30 2021-01-31 ZhaoLiuc Professor 138-1922-3912
2021-02-01 2021-02-08 ZhaoLiud Lecturer 198-1920-3912
2021-02-09 2021-02-15 ZhaoLieu Professor 178-1920-3912
2021-02-16 2021-02-24 ZhaoLiuf Lecturer 138-1929-3912
2021-02-25 2021-02-28 ZhaoLiug Professor 138-1920-0000
2021-03-01 2021-03-01 ZhaoLih Associate Professor 138-1929-0000
2021-03-02 2021-03-04 ZhaoLii Professor 138-1920-0200
2021-03-05 2021-03-05 ZhaoLij AssociateProfessor 138-1920-0044
2021-03-06 2021-03-06 ZhaoLiuk Professor 188-1920-0000
Free Ratio: 0.0

-----
--      0.Set start and end date      --
--      1.Add employees                --
--      2.Set duty                     --
--      3.Set duty randomly            --
--      4.Show duty table              --
--      5.Delete Employees             --
--      6.Set from .txt                --
--      others.quit                    --
-----
YOUR CHOICE:
6
Choose a file(1-8, otherwise quit):
5
LiSiOverlap!
ZhaoLiugOverlap!
Duty Table:
2021-01-10 2021-01-11 ZhangSan Manger 139-0451-0000
2021-01-21 2021-01-21 WangWu Associate Dean 177-2021-0301
2021-01-22 2021-01-22 ZhaoLiua Professor 138-1920-3912
2021-01-23 2021-01-29 ZhaoLiub Lecturer 138-1921-3912
2021-01-30 2021-01-31 ZhaoLiuc Professor 138-1922-3912
2021-02-01 2021-02-08 ZhaoLiud Lecturer 198-1920-3912
2021-02-09 2021-02-15 ZhaoLieu Professor 178-1920-3912
2021-02-16 2021-02-27 ZhaoLiuf Lecturer 138-1929-3912
2021-03-01 2021-03-01 ZhaoLih Associate Professor 138-1929-0000
2021-03-02 2021-03-04 ZhaoLii Professor 138-1920-0200
2021-03-05 2021-03-05 ZhaoLij AssociateProfessor 138-1920-0044
2021-03-06 2021-03-06 ZhaoLiuk Professor 188-1920-0000
Free Ratio: 0.17857142857142858

```

```

-----
--      0.Set start and end date      --
--      1.Add employees                --
--      2.Set duty                     --
--      3.Set duty randomly            --
--      4.Show duty table              --
--      5.Delete Employees             --
--      6.Set from .txt                --
--      others.quit                    --
-----
YOUR CHOICE:
6
Choose a file(1-8, otherwise quit):
6
Duty Table:
2021-01-10 2021-01-11 ZhangSan Manger 139-0451-0000
2021-01-12 2021-01-20 LiSi Secretary 151-0101-0000
2021-01-21 2021-01-21 WangWu Associate Dean 177-2021-0301
2021-01-22 2021-01-22 ZhaoLiua Professor 138-1920-3912
2021-01-23 2021-01-29 ZhaoLiub Lecturer 138-1921-3912
2021-01-30 2021-01-31 ZhaoLiuc Professor 138-1922-3912
2021-02-09 2021-02-15 ZhaoLiue Professor 178-1920-3912
2021-02-16 2021-02-24 ZhaoLiuf Lecturer 138-1929-3912
2021-03-01 2021-03-01 ZhaoLiuh Assciate Professor 138-1929-0000
2021-03-02 2021-03-04 ZhaoLiui Professor 138-1920-0200
2021-03-05 2021-03-05 ZhaoLiu j AssciateProfessor 138-1920-0044
2021-03-06 2021-03-06 ZhaoLiuk Professor 188-1920-0000
Free Ratio: 0.21428571428571427
-

-----
--      0.Set start and end date      --
--      1.Add employees                --
--      2.Set duty                     --
--      3.Set duty randomly            --
--      4.Show duty table              --
--      5.Delete Employees             --
--      6.Set from .txt                --
--      others.quit                    --
-----
YOUR CHOICE:
6
Choose a file(1-8, otherwise quit):
7
Employee ZhaoLiue does not exist!
Employee ZhaoLiui does not exist!
Duty Table:
2021-01-10 2021-01-11 ZhangSan Manger 139-0451-0000
2021-01-12 2021-01-20 LiSi Secretary 151-0101-0000
2021-01-21 2021-01-21 WangWu Associate Dean 177-2021-0301
2021-01-22 2021-01-22 ZhaoLiua Professor 138-1920-3912
2021-01-23 2021-01-29 ZhaoLiub Lecturer 138-1921-3912
2021-01-30 2021-01-31 ZhaoLiuc Professor 138-1922-3912
2021-02-01 2021-02-08 ZhaoLiud Lecturer 198-1920-3912
2021-02-16 2021-02-24 ZhaoLiuf Lecturer 138-1929-3912
2021-02-25 2021-02-28 ZhaoLiug Professor 138-1920-0000
2021-03-01 2021-03-01 ZhaoLiuh Assciate Professor 138-1929-0000
2021-03-05 2021-03-05 ZhaoLiu j AssciateProfessor 138-1920-0044
2021-03-06 2021-03-06 ZhaoLiuk Professor 188-1920-0000
Free Ratio: 0.17857142857142858
-

```

```

-----
--      0.Set start and end date      --
--      1.Add employees               --
--      2.Set duty                    --
--      3.Set duty randomly           --
--      4.Show duty table             --
--      5.Delete Employees            --
--      6.Set from .txt               --
--      others.quit                   --
-----
YOUR CHOICE:
6
Choose a file(1-8, otherwise quit):
8
Employee ZhaoLiue does not exist!
Employee ZhaoLiuk does not exist!
Duty Table:
2021-01-10 2021-01-11 ZhangSan Manger 139-0451-0000
2021-01-12 2021-01-20 LiSi Secretary 151-0101-0000
2021-01-21 2021-01-21 WangWu Associate Dean 177-2021-0301
2021-01-22 2021-01-22 ZhaoLiua Professor 138-1920-3912
2021-01-23 2021-01-29 ZhaoLiub Lecturer 138-1921-3912
2021-01-30 2021-01-31 ZhaoLiuc Professor 138-1922-3912
2021-02-01 2021-02-08 ZhaoLiud Lecturer 198-1920-3912
2021-02-16 2021-02-24 ZhaoLiuf Lecturer 138-1929-3912
2021-02-25 2021-02-28 ZhaoLiug Professor 138-1920-0000
2021-03-01 2021-03-01 ZhaoLiuh Assciate Professor 138-1929-0000
2021-03-02 2021-03-04 ZhaoLiui Professor 138-1920-0200
2021-03-05 2021-03-05 ZhaoLiu j AssciateProfessor 138-1920-0044
Free Ratio: 0.125

```

3.8 应对面临的新变化

3.8.1 变化 1

排班应用：可以出现一个员工被安排多段值班的情况，例如张三的值班日期为(2021-01-01, 2021-01-10), (2021-02-01, 2021-02-06)。但是此时任然不允许有任何重复段的出现，也不允许出现空白。

如何修改设计以应对变化：首先是，将 `IntervalSet` 改为 `MultiIntervalSet`，这样就可以应对插入重复的 `label` 情况。接下来就是将功能相同但是命名不同方法改正确，一个是 `label` 改为 `labels`，另一个是 `getIntervals` 改为 `allIntervals`。

评估之前的设计是否可应对变化、代价如何：修改的代码量就只有几行，很容易应对该改变，如果在设计 `IntervalSet` 和 `MultiIntervalSet` 的时候将相同功能的方法命名为一样，那么就几乎只用修改一行代码。

如下是在插入相同段的显示：

```

-----
--      0.Set start and end date      --
--      1.Add employees                --
--      2.Set duty                     --
--      3.Set duty randomly            --
--      4.Show duty table              --
--      5.Delete Employees             --
--      6.Set from .txt                --
--      others.quit                    --
-----
YOUR CHOICE:
2
Empty Duty Table:
2021-02-14 2021-03-22
Free Ratio: 1.0
Choose an employee:
a
Date Form (yyyy-mm-dd):
2021-02-20
2021-02-25
Continue Set Duty? Y(y)/N(others)
y
Empty Duty Table:
2021-02-14 2021-02-19
2021-02-26 2021-03-22
Free Ratio: 0.8378378378378378
Choose an employee:
a
Date Form (yyyy-mm-dd):
2021-03-10
2021-03-22
Continue Set Duty? Y(y)/N(others)
y
Empty Duty Table:
2021-02-14 2021-02-19
2021-02-26 2021-03-09
Free Ratio: 0.4864864864864865

```

3.8.2 变化 2

课表应用：不管学生选课状况如何，不能够出现两门课排在同一时间的情况（即“无重叠”）。那么在原过程中需要调用 `checkNonOverlap` 来检查重复段，若有重复则做出相应应答，可以报错，并且不再插入该段课表。

评估之前的设计是否可应对变化、代价如何：代价一般高，只插入了二十行左右的代码，并且修改时间不长。

如何修改设计以应对变化：加入 `checkNonOverlap` 方法，并检测重复段。

如下例所示：在插入重复段的时候会提示错误并且不会将重复段插入。

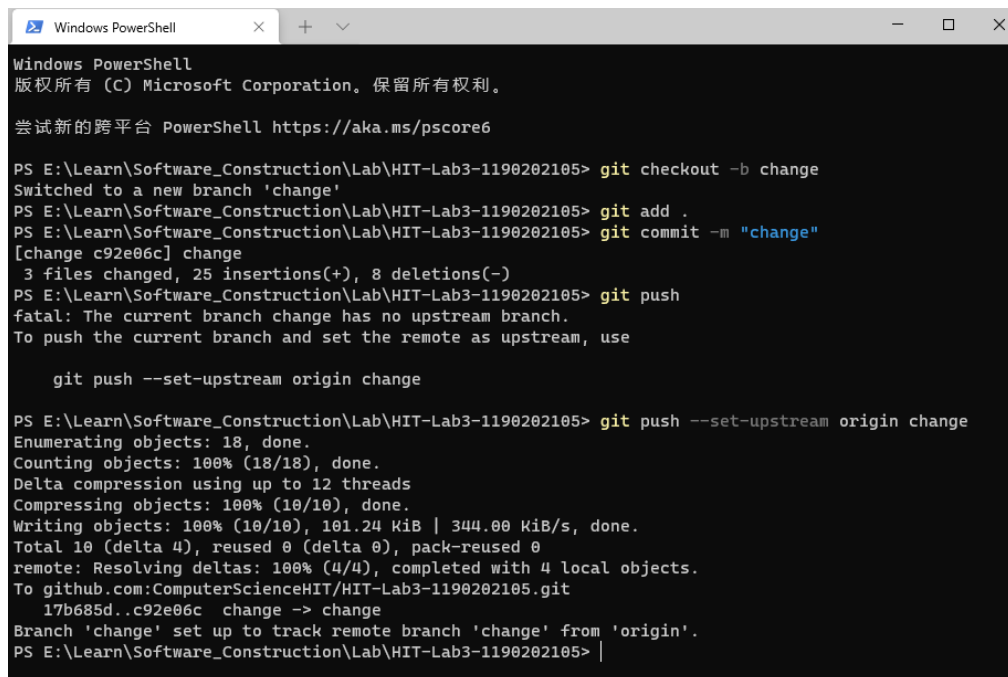
```

Add courses to the curriculum.
Left NOT setted Courses:
[ID:2, Name:b, Teacher:b, Place:b] 4
[ID:1, Name:a, Teacher:a, Place:a] 2
Conflict Ratio: 0.0
Free Time Ratio: 1.0
Choose Course according to ID:
1
Choose weekday: 1-7
1
Choose a period: 0[8-10h], 1[10-12], 2[13-15], 3[15-17], 4[19-21]
1
Left NOT setted Courses:
[ID:2, Name:b, Teacher:b, Place:b] 4
Conflict Ratio: 0.0
Free Time Ratio: 0.9714285714285714
Choose Course according to ID:
2
Choose weekday: 1-7
1
Choose a period: 0[8-10h], 1[10-12], 2[13-15], 3[15-17], 4[19-21]
1
Overlap!
Left NOT setted Courses:
[ID:2, Name:b, Teacher:b, Place:b] 4
Conflict Ratio: 0.0
Free Time Ratio: 0.9714285714285714

```

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。



```

Windows PowerShell
版权所有 (c) Microsoft Corporation. 保留所有权利。

尝试新的跨平台 PowerShell https://aka.ms/pscore6

PS E:\Learn\Software_Construction\Lab\HIT-Lab3-1190202105> git checkout -b change
Switched to a new branch 'change'
PS E:\Learn\Software_Construction\Lab\HIT-Lab3-1190202105> git add .
PS E:\Learn\Software_Construction\Lab\HIT-Lab3-1190202105> git commit -m "change"
[change c92e06c] change
 3 files changed, 25 insertions(+), 8 deletions(-)
PS E:\Learn\Software_Construction\Lab\HIT-Lab3-1190202105> git push
fatal: The current branch change has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin change

PS E:\Learn\Software_Construction\Lab\HIT-Lab3-1190202105> git push --set-upstream origin change
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 12 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 101.24 KiB | 344.00 KiB/s, done.
Total 10 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To github.com:ComputerScienceHIT/HIT-Lab3-1190202105.git
 17b685d..c92e06c change -> change
Branch 'change' set up to track remote branch 'change' from 'origin'.
PS E:\Learn\Software_Construction\Lab\HIT-Lab3-1190202105>

```

```
PS E:\Learn\Software_Construction\Lab\HIT-Lab3-1190202105> git log
commit c92e06c93d9958990198a9ea8865f1cecb2fb831 (HEAD -> change, origin/change)
Author: ifu <1091288450@qq.com>
Date: Sat Jul 3 21:09:50 2021 +0800

    change

commit b56001af9eb021d9d9acbb3d945105ef18eb0a5c (origin/master, origin/HEAD, master)
Author: ifu <1091288450@qq.com>
Date: Sat Jul 3 20:39:58 2021 +0800

    NOT change

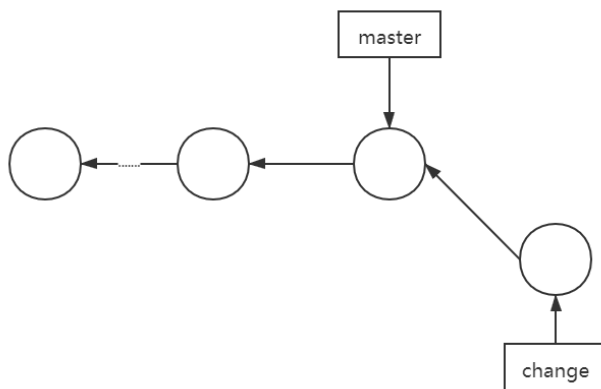
commit 0546a4c8d6ca90faeada9141c4877f2e9d9b64eb
Author: ifu <1091288450@qq.com>
Date: Sat Jul 3 19:50:31 2021 +0800

    NOT change

commit d60189cedbfc2abd9888d876efd69bc4e88eec36
Author: ifu <1091288450@qq.com>
Date: Sat Jul 3 18:54:13 2021 +0800

    NOT change

commit 46d72a72a6344dea10f85239f1ef62815a95d1c6
Author: ifu <1091288450@qq.com>
Date: Sat Jul 3 10:32:00 2021 +0800
```



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
2021.06.17	14:00-16:00	阅读实验要求，完成环境搭建等	完成
2021.06.24	14:00-16:00	根据实验要求，提取各类要点，编写 IntervalSet 的接口和实现代码	未完成
2021.06.27	19:00-23:00	提取 IntervalSet 共性方法、编写测试以及实现	未完成

2021.06.28	全天	IntervalSet 方法及实现, 提取共性方法 MutilIntervalSet, 设计测试和编写方法	未完成
2021.06.29	全天	完成以上没有完成的内容, 设计关于排班系统的 APP, 面向复用的设计 Employee	未完成, 在实际中对方案五的应用出现疑惑, 改变方案
2021.06.30	全天	面向复用的设计 Process 设计关于 Process 进程安排的 APP	完成
2021.07.01	全天	面向复用的设计 Course, 设计关于课表系统的 APP	未完成
2021.07.02	全天	设计关于课表系统的 APP, 基于语法的数据读入	未完成
2021.07.03	全天	课程系统设计, 改变排班和课表应用并提交到 change 分支	完成
2021.07.04	8:00-12:00	撰写剩余报告, 修改部分实现内容, 完善细节等	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
对于接口、继承和委托的使用不熟悉。	在考虑之后改变方法。
对于三个 APP 设计的方法不知到从何下手	面向具体应用编程, 要使用某个功能的时候再根据这来设计具体方法。
不清楚正则表达式的使用	询问同学, 根据 PPT 中有限的内容从网络上查找到相关方法和使用格式等。
各类之间调用关系混乱	仔细分辨内容

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) 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?

在面向 ADT 编程的时候需要考虑代码的可复用性, 需要使整个编程更具有普适性; 面向应用场景可以直接根据要求来编写程序, 更易于操作, 但是程序复用性很低, 难以在其他场景进行复用, 或者是需要修改的内容

增多。在不同的场景下的确能带来简化代码的功能，一定程度上体会了面向 ADY 编程带来的好处，但是由于三个应用场景中 Course 的应用还存在更为简单的操作，例如数组等，所以在应用过程中还增加了编程的复杂度。

- (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？

根据规约、不变量、*RI*、*AF* 来编写程序，能有效避免在代码中插入错误，再检查错误的过程，节约时间；同时能够尽早找到程序中的错误，避免错误累积到后续难以修改。在以后的编程中可以尽量这样做，但是对于简单逻辑的程序其实意义不是很大。

- (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？

还好，在本题中所开发的 API 很难被复用，因为按题意开发的 API 在实际应用中很难处理自定义的边界。例如，开发的 *calcFreeTimeRatio* 和 *calcConflictRatio* 在实际情况 Course 的应用中，是不应该自己去计算整体时间，而是应该定义一个全部的具体时间，例如 $7*5*2=70$ 个小时。

- (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？

语法驱动编程一定程度上固定了输入格式，例如本次实验中根据语法和正则表达式解析输入文件，能根据高效提取得到我们所期待的内容，还较为清晰和易解。同时这还是一种隐秘的编程方式，能够提高隐私性。

- (5) 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？

设计问题主要两点，一是提取共同点，要将不同开发内容的共同点提取出来；二就是方法设计和调用很茫然，在实现过程中，很多方法的实现都是根据 APP 设计所要使用的内容来添加的，以及调用关系会随着代码量的增加而变得非常混乱。

- (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？

在本实验中，利用接口、抽象类、类的组合继承等，有效地减少了代码量，避免了重复编程，但是在复用时，需要理清各个关系。

(7) 关于本实验的工作量、难度、deadline。

本实验工作量巨大，相对于实验二来说难度提高了很多，deadline 与期末考试时间冲突，导致实际上能花在实验中的时间很短。

(8) 到目前为止你对《软件构造》课程的评价。

能提高能力学到一些编程思想等，但是时间长度和内容多少匹配不上，太赶了。