

**2020年春季学期  
计算机学院《软件构造》课程**

**Lab 3实验报告**

|  |  |
| --- | --- |
| 姓名 | 张瑞豪 |
| 学号 | 1180800811 |
| 班号 | 1803002 |
| 电子邮件 | m17779349381@163.com |
| 手机号码 | 17779349381 |

**目录**

[1 实验目标概述 1](#_Toc40510882)

[2 实验环境配置 1](#_Toc40510883)

[3 实验过程 1](#_Toc40510884)

[3.1 待开发的三个应用场景 2](#_Toc40510885)

[3.2 面向可复用性和可维护性的设计：PlanningEntry<R> 3](#_Toc40510886)

[3.2.1 PlanningEntry<R>的共性操作 3](#_Toc40510887)

[3.2.2 局部共性特征的设计方案 4](#_Toc40510888)

[3.2.3 面向各应用的PlanningEntry子类型设计（个性化特征的设计方案） 4](#_Toc40510889)

[3.3 面向复用的设计：R 7](#_Toc40510890)

[3.4 面向复用的设计：Location 10](#_Toc40510891)

[3.5 面向复用的设计：Timeslot 11](#_Toc40510892)

[3.6 面向复用的设计：EntryState及State设计模式 12](#_Toc40510893)

[3.7 面向应用的设计：Board 15](#_Toc40510894)

[3.8 Board的可视化：外部API的复用 19](#_Toc40510895)

[3.9 可复用API设计及Façade设计模式 21](#_Toc40510896)

[3.9.1 检测一组计划项之间是否存在位置独占冲突 21](#_Toc40510897)

[3.9.2 检测一组计划项之间是否存在资源独占冲突 21](#_Toc40510898)

[3.9.3 提取面向特定资源的前序计划项 22](#_Toc40510899)

[3.10 设计模式应用 23](#_Toc40510900)

[3.10.1 Factory Method 23](#_Toc40510901)

[3.10.2 Iterator 24](#_Toc40510902)

[3.10.3 Strategy 27](#_Toc40510903)

[3.11 应用设计与开发 29](#_Toc40510904)

[3.11.1 航班应用 29](#_Toc40510905)

[3.11.2 高铁应用 34](#_Toc40510906)

[3.11.3 课程应用 38](#_Toc40510907)

[3.12 基于语法的数据读入 43](#_Toc40510908)

[3.13 应对面临的新变化 49](#_Toc40510909)

[3.13.1 变化1 49](#_Toc40510910)

[3.13.2 变化2 51](#_Toc40510911)

[3.13.3 变化3 52](#_Toc40510912)

[3.14 Git仓库结构 53](#_Toc40510913)

[4 实验进度记录 54](#_Toc40510914)

[5 实验过程中遇到的困难与解决途径 55](#_Toc40510915)

[6 实验过程中收获的经验、教训、感想 55](#_Toc40510916)

[6.1 实验过程中收获的经验和教训 55](#_Toc40510917)

[6.2 针对以下方面的感受 55](#_Toc40510918)

# 实验目标概述

本次实验覆盖课程第 3、4、5 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

⚫ 常见的 OO 设计模式

⚫ 语法驱动的编程、正则表达式

⚫ 基于状态的编程

⚫ API 设计、API 复用

本次实验给定了五个具体应用（高铁车次管理、航班管理、操作系统进程管

理、大学课表管理、学习活动日程管理），学生不是直接针对五个应用分别编程实现，而是通过 ADT 和泛型等抽象技术，开发一套可复用的 ADT 及其实现，充分考虑这些应用之间的相似性和差异性，使 ADT 有更大程度的复用（可复用性）和更容易面向各种变化（可维护性）。

# 实验环境配置

这次实验需要的环境前两个实验已经配置完成。

在这里给出你的GitHub Lab3仓库的URL地址（Lab3-学号）。

https://github.com/ComputerScienceHIT/Lab3-1180800811.git

# 实验过程

请仔细对照实验手册，针对每一项任务，在下面各节中记录你的实验过程、阐述你的设计思路和问题求解思路，可辅之以示意图或关键源代码加以说明（但千万不要把你的源代码全部粘贴过来！）。

## 待开发的三个应用场景

列出你所选定的三个应用。

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。

**选用的三个应用：** 1、航班管理

2、高铁车次管理

3、大学课表管理

三个应用的共性：①均表示一个待执行的活动/任务，它需要配置特定类型/数的资源，并在特定的物理位置加以执行。

②活动的状态有交集：都有未分配资源、已

分配资源但未启动、已启动、已完成、已取消五个状态。

三个应用的差异：①位置差异：航班：两个位置，位置可共享

高铁车次:多个位置，位置可共享

大学课程：一个位置，位置不可共享

②资源差异：航班/课程：单个可区分的资源(飞机)

高铁:多个可区分的资源

③状态差异：高铁相较于航班/课程，多一个可阻塞的状态

④位置是否可更改差异：航班/高铁：可提前设定，设定后

不可更改

课程：可提前设定，设定后可更改

⑤时间差异: 高铁：多个起止时间对

课程/航班：1个时间起止对

## 面向可复用性和可维护性的设计：PlanningEntry<R>

该节是本实验的核心部分。

### PlanningEntry<R>的共性操作

1. 和状态有关的方法

⚫ 启动一个计划项

⚫ 结束一个计划项

⚫ 取消一个计划项

⚫ 获取当前的状态

⚫ 判断当前状态是否可接受

⚫ 设定计划项的状态

1. getter方法

⚫ 获取计划项使用的全部资源

⚫ 获取计划项的起止时间

⚫ 获取计划项的名字

⚫ 得到计划项所用的位置和时间对的对应关系(主要用于API设计，判断位置冲突)

1. setter方法

⚫ 设定计划项的名字

说明：①在PlanningEntry中提供一个统一的获取资源的方法getresource，这个方法可以用来APIs中判断资源冲突，返回值类型是一个List，因为考虑到可能使用多个资源，也可能使用单个资源，所以一起放在一个List列表存储所有的资源，

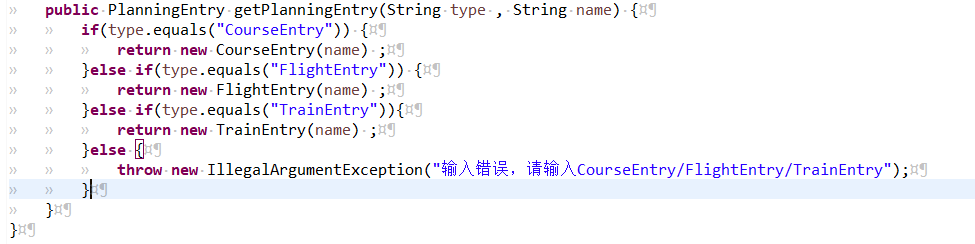
②得到计划项所用的位置和时间对的对应关系是为了**3.10** API的设计需要判断位置冲突。通过判断相同位置对应的时间是否重叠来判断是否有位置冲突。这个方法既可以得到所有的时间对，也可以得到所有的位置。

以上两个方法抽象出来放在公共接口PlanningEntry中，完全是为了设计API。经验证，这样设计之后，314change我不需要改动任何的APIs类的代码。可复用性很高。

### 局部共性特征的设计方案

创建一个新的子类型：利用工厂方法能够创建一个指定类型的子类







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

1.设计方案：方案5：CRP，通过接口组合

思想: ①把三个应用公有的特征抽象成一个公共的接口PlanningEntry<R>

1. 对于三个应用的不同维度的差异，为每个维度上的不同特征取值分别定义不同的接口，在接口中定义特殊的操作，分别实现针对该维度接口的不同实现类，实现其特殊操作逻辑。
2. 对于具体的应用，通过接口组合，将各种局部共性行为复合在一起形成满足每个应用要求的特殊接口（包含了该应用内的全部特殊功能），从而该应用子类可直接实现该组合接口。
3. 在应用子类内，不是直接实现每个特殊操作，而是通过 delegation 到外部每个维度上的各具体实现类的相应特殊操作逻辑

以TrainEntry为例:**通过接口的组合来形成满足应用的接口。**





2.个性化特征方案

三个子类的差异方法就是Location 、Timeslot 、Resource 属性的获取和修改。以及状态的转化。

TrainEntry:

构造器方法： 无参数、带参数

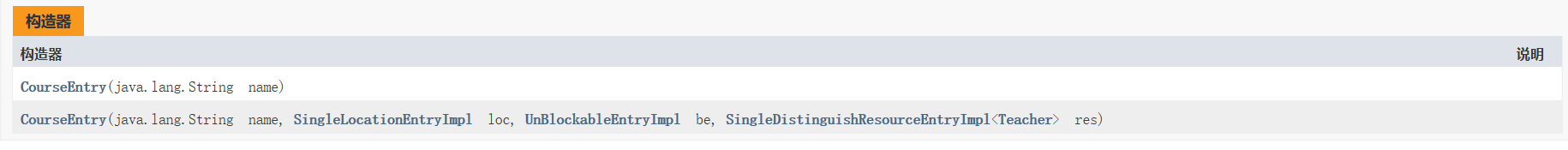


**方法**



CourseEntry:

构造器方法

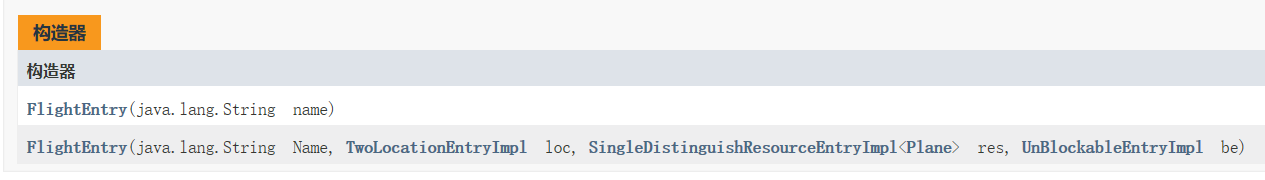


其他方法



FlightEntry

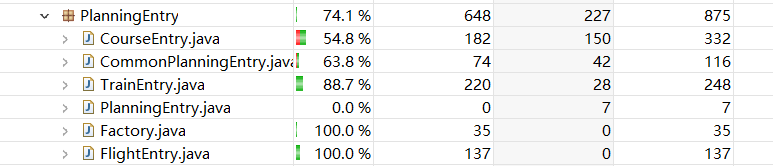
构造器方法



其他方法



测试：





## 面向复用的设计：R

1.Plane

属性：

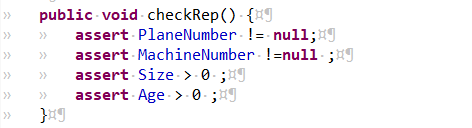
**private** String PlaneNumber ;//飞机编号

**private** String MachineNumber ;//机型号

**private** **int** Size ;//座位数

**private** **double** Age ;//机龄

checkRep:



方法：



2.Railway

属性:

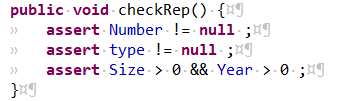
**private** String Number ;//编号

**private** Type type ; //车厢类型

**private** **int** Size ; //定员数

**private** **int** Year ;//出厂年份

checkRep：



方法：



3.Teahcer

属性：

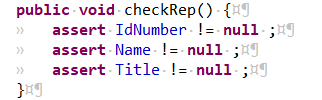
**private** String IdNumber ;//教师的身份证号

**private** String Name ;//教师的名字

**private** **boolean** Sex ; //教师的性别,true表示男，false表示女

**private** String Title ;//教师职称

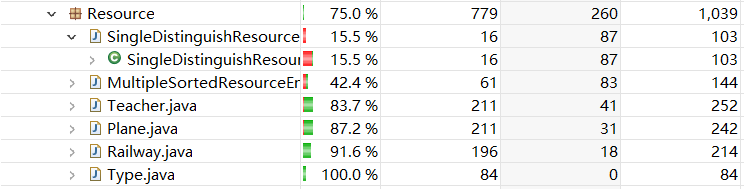
checkRep:

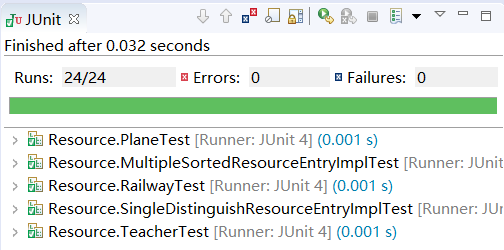


方法



测试：





## 面向复用的设计：Location

属性：

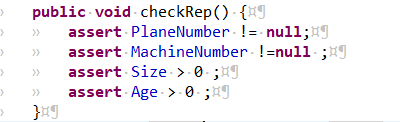
**private** **double** longitude ; //经度

**private** **double** latitude ;//纬度

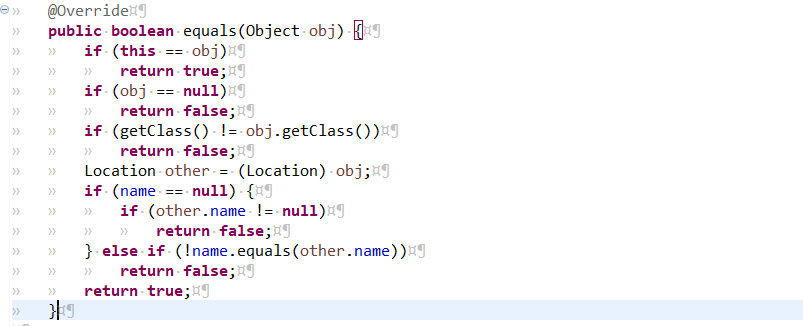
**private** String name ; //位置名称

**private** **boolean** share ;//位置是否可以共用，true表示可以共用，false表示不能共用

checkRep:



Equals方法：由于考虑到老师给的航班管理的文件对位置的读取没有经纬度的信息，而且老师说经纬度是为了以后复用可能被用来求解距离，所以我觉得在本次实验中，可以认为经纬度没有什么作用，只要位置的名字相等，就可以认为两个位置相同。

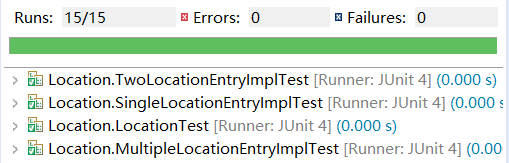


其他方法



测试：





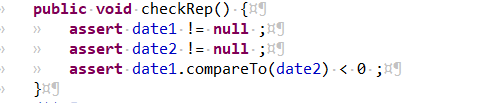
## 面向复用的设计：Timeslot

1.属性：

**private** Calendar date1 = Calendar.*getInstance*();//起始时间

**private** Calendar date2 = Calendar.*getInstance*();//终止时间

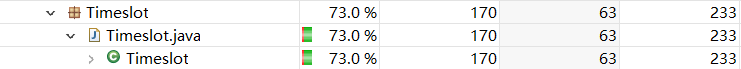
2.checkRep

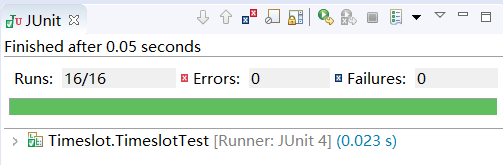


3.方法



测试：

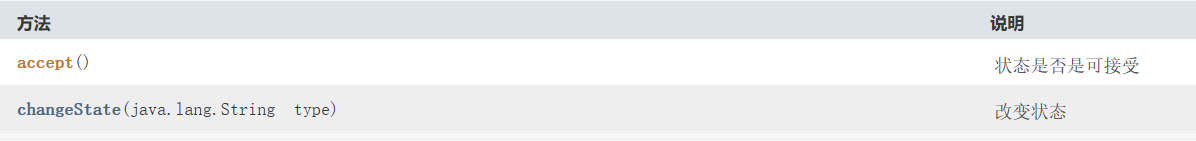




## 面向复用的设计：EntryState及State设计模式

1.State接口

State接口时状态类的公共接口，只有两个方法，改变状态和判断是否是可接受转态



2.具体的状态类

WAITTING.java//未分配资源、

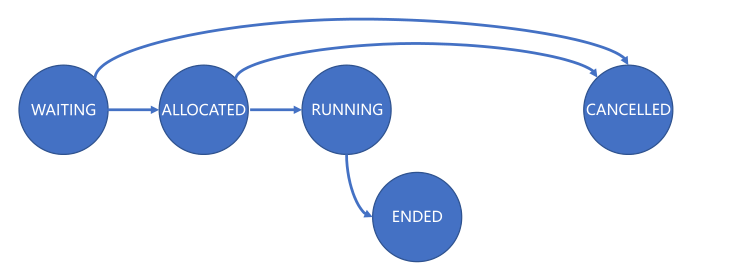
ALOCATED.java//已分配资源但未启动

RUNNING.java//已启动

CANCELLED.java//已取消

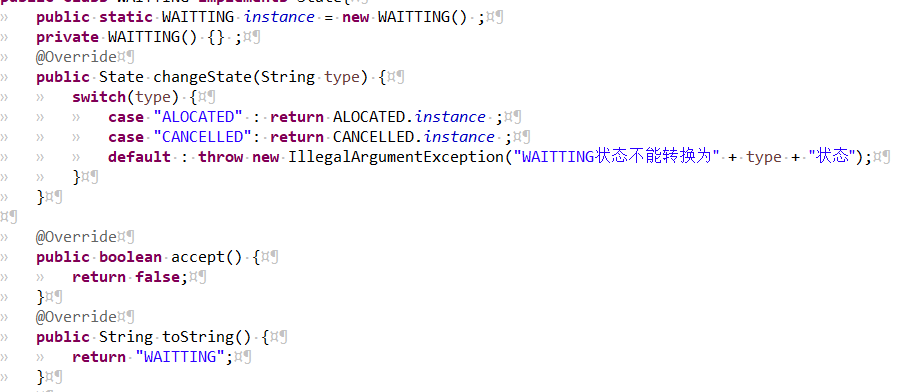
ENDED.java//已完成

BLOCKED.java//挂起

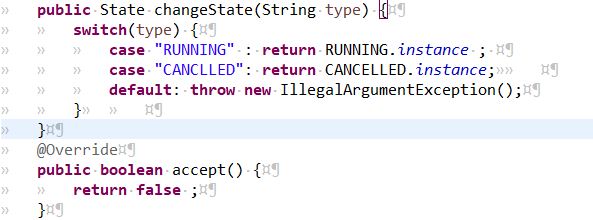


由状态关系转移图可知，对于每个状态类来说，只能转换为特定的状态类。

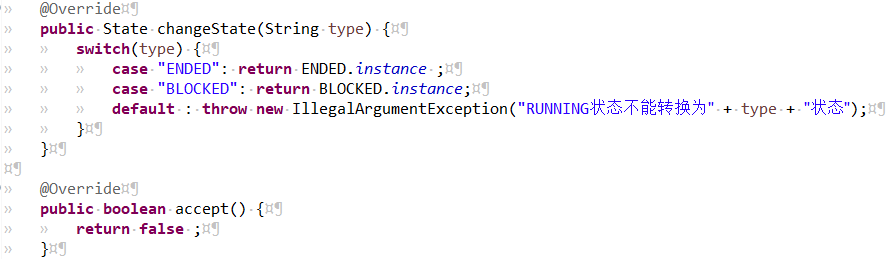
1. 以WAITTING状态为例，只能转换为ALOCATED和CANCALLED两个状态，并且状态不可接受



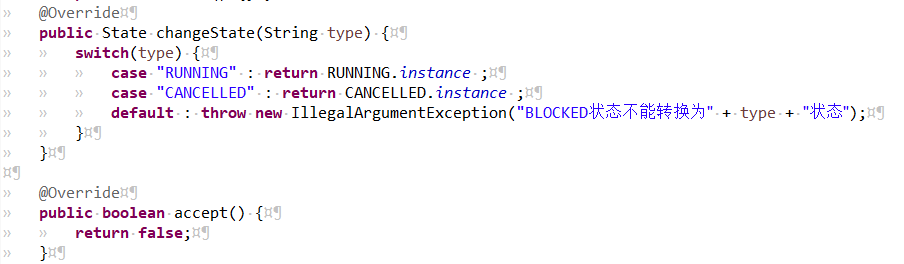
1. ALOCATED状态



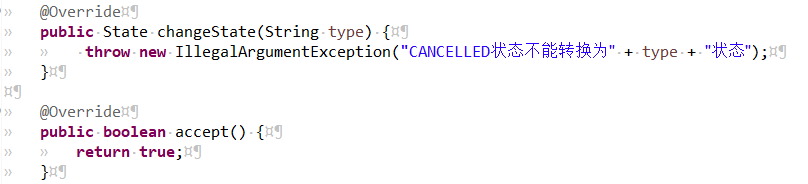
1. RUNNING状态



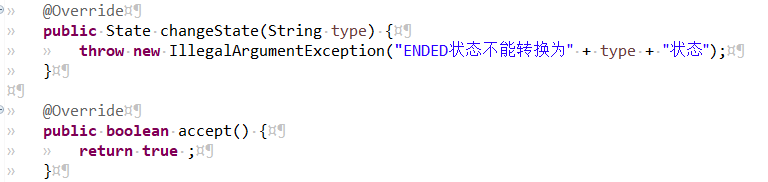
1. BLOCKED状态



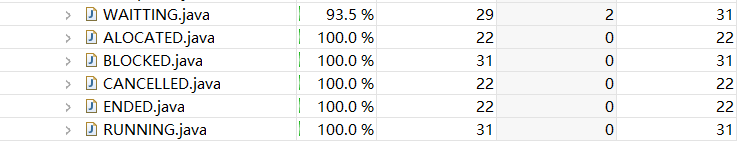
1. CANCELLED状态

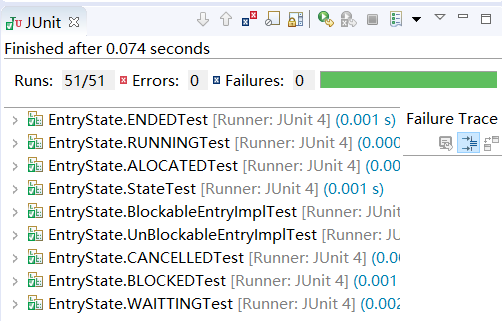


1. ENDED状态



测试：





## 面向应用的设计：Board

思路:对于每一个信息板，都有一个当前时间，当前位置与他相对应，并且包含了一组计划项。因此属性就是当前时间、当前位置、一组计划项

功能：①对于每个信息板，首先实现按照实验指导书所说显示出发生在计划板所在位置和当前时间相差一个小时内的计划项信息，②其次展示所有的发生在该计划板的计划项信息(没有相差一个小时的限制)。

实现方法:首先在所有的计划项中找出和信息板位置相关的计划项，将他们按照发生在当前位置的时间排好序。然后利用jtable组件可视化信息板即可。

注意：而且当前时间并不是系统显示的时间，每次当前时间都是需要由用户输入的。

1.CourseCalendarBoard

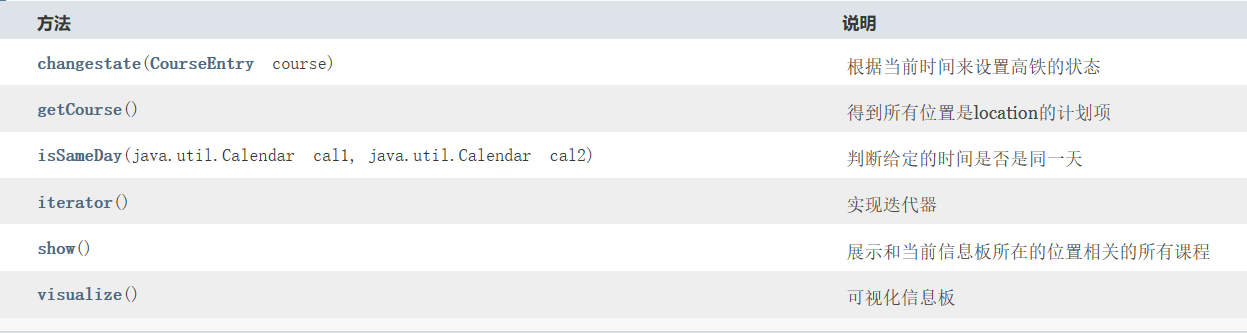
属性:

**private** Location location ;//信息板所在的的位置

**private** List<CourseEntry> course = **new** ArrayList<CourseEntry>() ;//一系列的计划项

**private** Calendar time = Calendar.*getInstance*() ; //当前时间

方法:



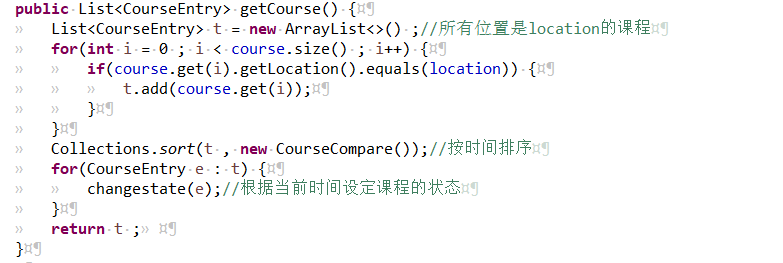
注释：visualize方法显示的是按照实验指导书所说显示出发生在计划板所在位置和当前时间同一天的课程信息。

show方法显示的是发生在当前位置的所有计划项的信息，没有时间要求。

changestate方法：由于每次显示信息板都要输入当前时间，那么每次输入当前时间的不同，相交于用户输入的当前时间而言，所有的课程的状态也是不同的，因此changestate方法就是根据用户输入的当前时间来改变计划项的状态。



getCourse()方法：得到的是所有发生在该位置的课程



isSameDay()方法：判断两个时间是否是同一天

2. FlightScheduleBoard

属性： **private** Calendar time = Calendar.*getInstance*() ;//屏幕版当前时间

**private** Location location ; //屏幕板所在的位置

**private** List<FlightEntry> flight = **new** ArrayList<FlightEntry>() ;//一系列的计划项

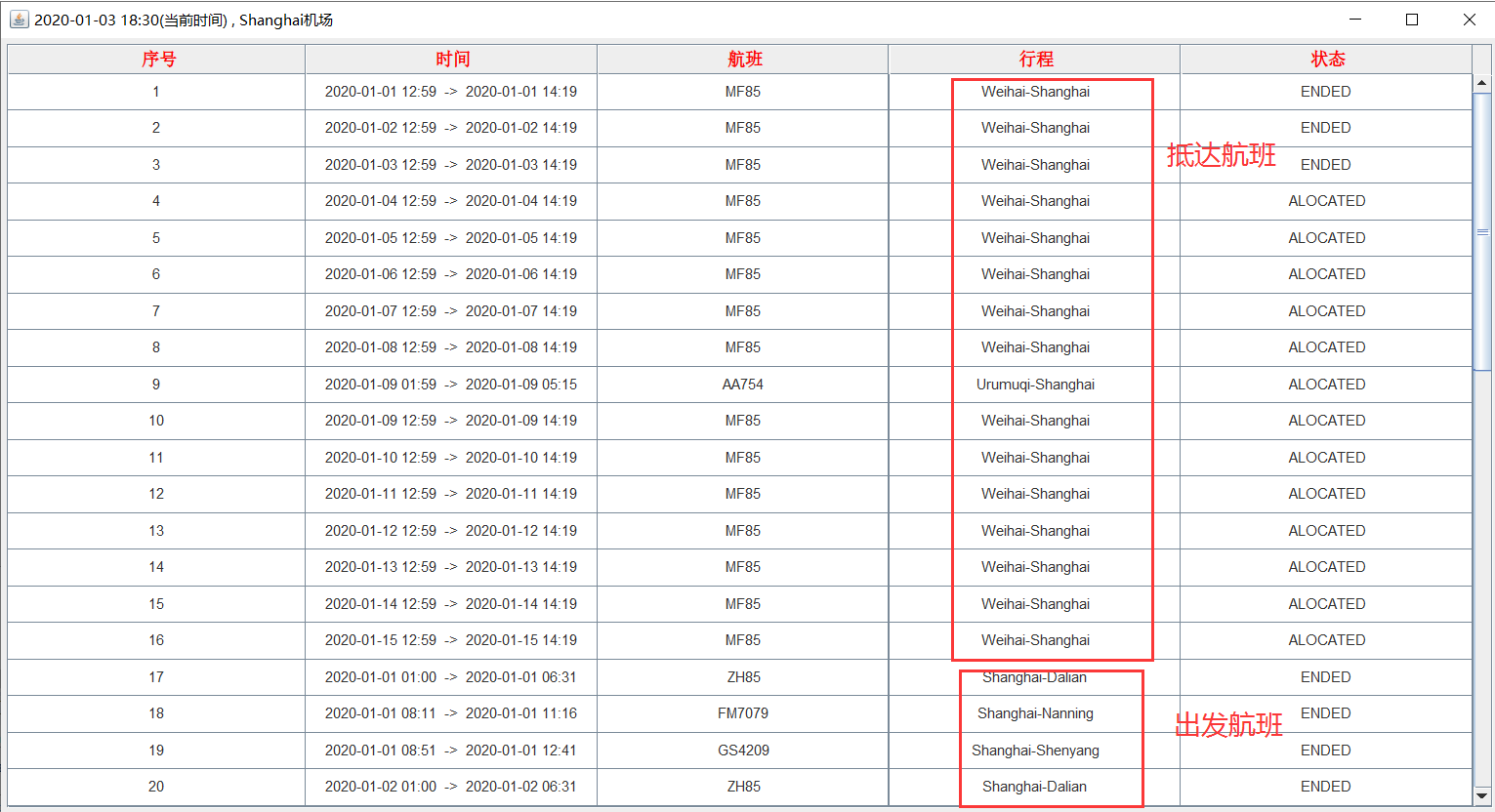
方法：



注释：visualize方法显示的是按照实验指导书所说显示出发生在计划板所在位置和当前时间相差一个小时的航班信息。

show方法显示的是发生在当前位置的所有计划项的信息，没有时间要求。

Show方法示例



Visualize方法示例



3. TrainScheduleBoard

属性:

**private** Calendar time = Calendar.*getInstance*() ;//屏幕版当前时间

**private** Location location ; //屏幕板所在的位置

**private** List<TrainEntry> train = **new** ArrayList<TrainEntry>() ;//一系列的计划项

方法：





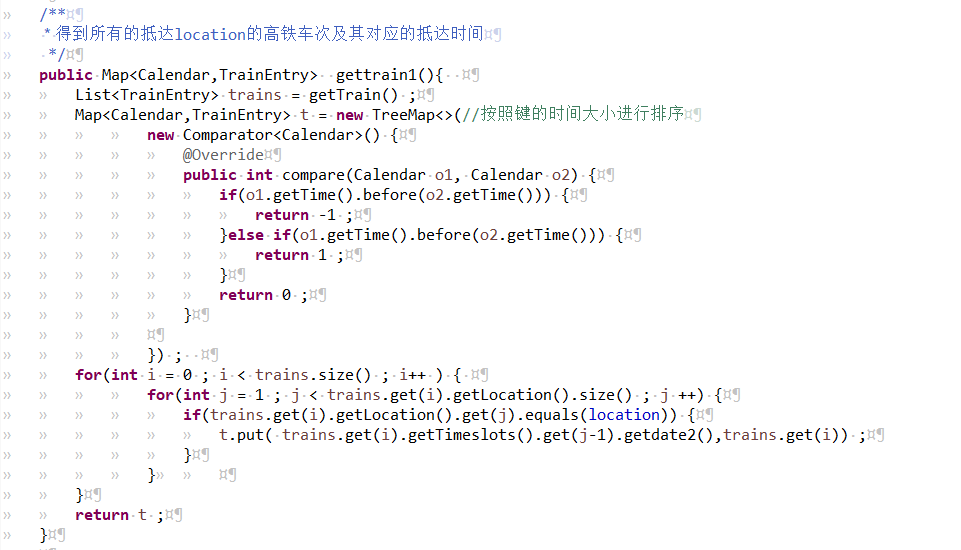
注释：visualize方法显示的是按照实验指导书所说显示出发生在计划板所在位置和当前时间相差一个小时的高铁车次信息。

show方法显示的是发生在当前位置的所有计划项的信息，没有时间要求。

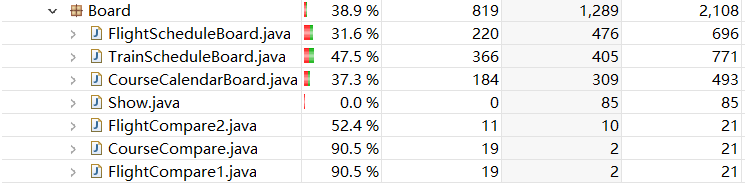
设计思路：对于每一个经停location的高铁车次(由getTrain方法得到)，都对应一个抵达时间/出发时间，所以我是实现了gettrain1方法，得到所有抵达location的高铁车次和对应的抵达时间，这种对应关系用Treemap<Calendar,TrainEntry>存储，然后对键按照时间进行排序。然后visualize方法就可以按照时间增序显示所有的高铁车次信息。同理对于出发车次也是如此。

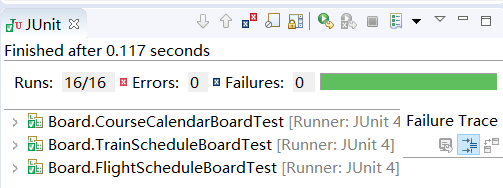
changestate方法：由于每次显示信息板都要输入当前时间，那么每次输入当前时间的不同，相交于用户输入的当前时间而言，所有的高铁车次的状态也是不同的，因此changestate方法就是根据用户输入的当前时间来改变计划项的状态。

gettrain1方法



测试：

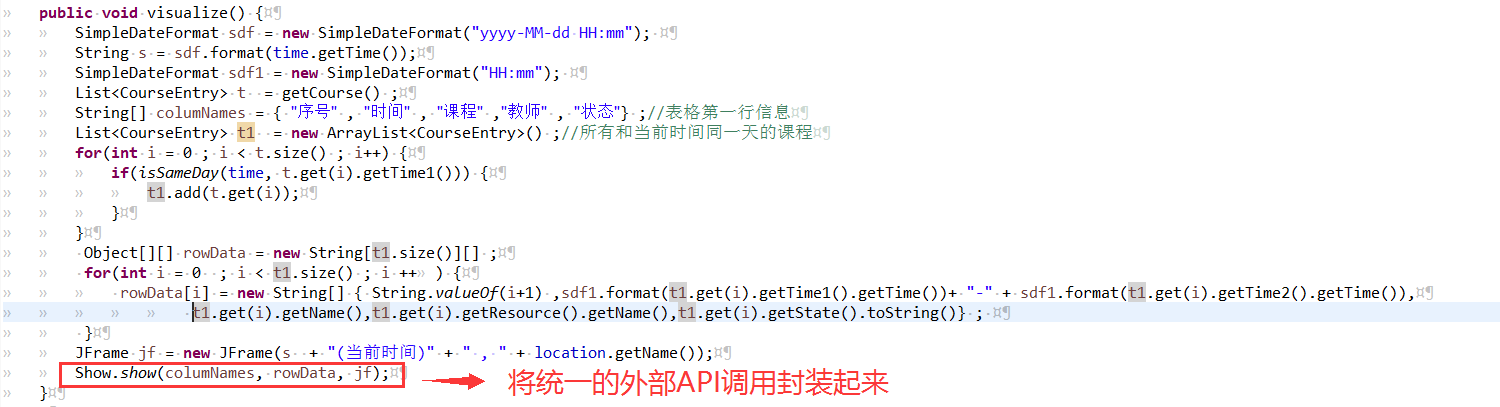




## Board的可视化：外部API的复用

只要把所有显示的东西放进一个二维数组，再利用jtable组件就能显示出当前的信息板。

Visualize方法思路：首先用getcourse得到所有按时间排好序的和该位置相关的课程信息，然后遍历，如果课程和当前时间是同一天，就把相应的信息放进二维数组内。最后借助于外部API就可以显示出来。



说明：Show类的show方法，参数是一个二维数组rowData,和一维数组(第一行信息)和一个JFrame容器。该方法用来可视化。



Show方法：和visualize方法类似，只是少了判断和当前时间是否是同一天的约束。



## 可复用API设计及Façade设计模式

Façade设计模式：客户端需要通过一个简化的接口来访问复杂系统内的功能，我们只需要提供一个统一的接口来取代一系列小接口调用，相当于对复杂系统做了一个封装，简化客户端使用。

在lab3可复用API设计中，我们需要把检查资源冲突、检查位置冲突、获取前序计划项的方法封装在一个APIs类中，通过提供这个统一的类来提供功能，简化客户端的使用。

### 检测一组计划项之间是否存在位置独占冲突

位置冲突：即不同的计划项在同一个时间点上占用了相同的不可共享位置(在这个意义上，这个方法不适用与高铁车次应用和航班应用)。

设计思路：对于使用不可共享的位置计划项而言，遍历每个计划项，如果任意两个计划项的运行时间存在重叠，则比较他们的位置，如果位置相等，则说明存在位置冲突。否则不存在位置冲突。



### 检测一组计划项之间是否存在资源独占冲突

资源冲突：即不同的计划项在同一个时间点上占用了相同的资源

设计思路1：对于使用不可共享的位置计划项而言，遍历每个计划项，如果任意两个计划项的运行时间存在重叠，则比较他们的位置，如果位置相等，则说明存在位置冲突。否则不存在位置冲突。

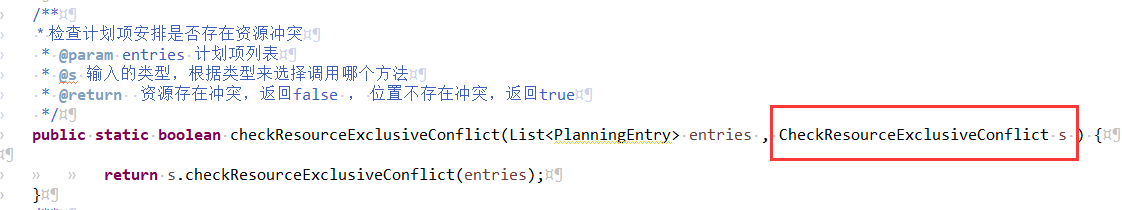


设计思路2：对于使用不可共享的位置计划项而言，遍历每个计划项，如果任意两个计划项存在使用相同的资源，则比较他们的运行时间，如果运行时间重叠，则说明存在位置冲突。否则不存在位置冲突。



采用策略模式，由用户来选择调用的方法.其中公共接口是CheckResourceExclusiveConflict.java,两种不同的实现分别是checkResourceExclusiveConflict1.java

和 checkResourceExclusiveConflict2.java。根据传入的参数来选择调用哪个方法来实现。



3.10节模式的引用中，策略模式会重新提及这个方法。

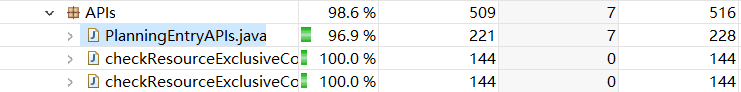
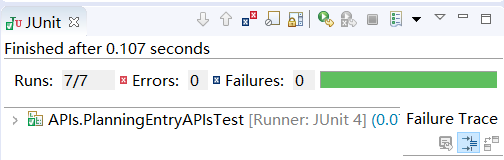
### 提取面向特定资源的前序计划项

特定资源的前序计划项：即在一组计划项中，针对某个资源 r 和使用 r 的某个计划项 e，从一组计划项中找出 e 的前序 f，f 也使用资源 r，f 的执行时间在 e 之前，且在 e 和 f 之间不存在使用资源 r 的其他计划项。

设计思路：首先初始化一个前序计划项x和对应的最晚结束时间latestTime,遍历所有的计划项，如果某个计划项和计划项e共用资源r，而且该计划项的结束时间晚于latestTime，则重新设定前序计划项e和最晚结束时间latestTime。循环结束，返回x即可。



测试：

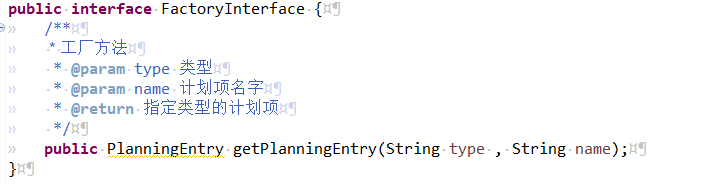
 

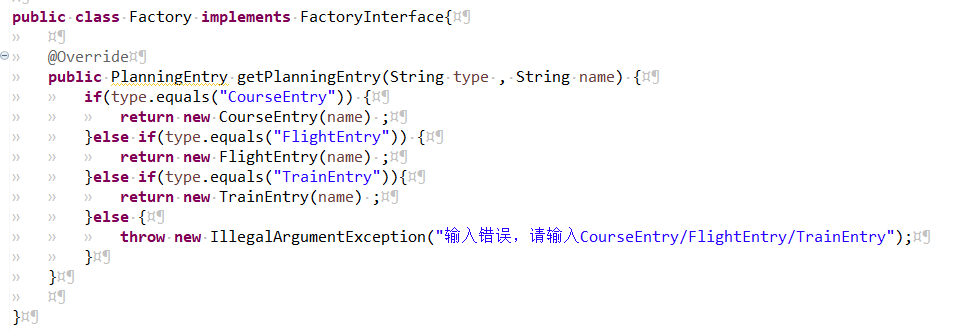
## 设计模式应用

请分小节介绍每种设计模式在你的ADT和应用设计中的具体应用。

### Factory Method

思路：设计一个工厂类统一的接口factoryInterface，工厂类Factory继承这个接口并实现里面的工厂方法。工厂类的工厂方法可以根据指定的类型来创建相应的计划项。





在PlanningEntry类，就可以调用工厂方法来创建一个新的计划项。



### Iterator

我的思路:让board类实现Iterable接口，并且实现Iterator方法即可。对于每一个信息板而言，都对应一个当前位置，信息板的迭代器只需要遍历和这个位置相关的计划项既可。首先，先得到所有和当前位置相关的计划项，然后按照时间进行排序。最后实现一个自己的迭代器，board类的迭代器返回自己实现的迭代器既可。

1.CourseCalendarBoard的迭代器方法

自己实现的迭代器：

属性： **private** List<CourseEntry> course1 = getCourse() ;//获取和当前计划板所 在的位置相关的计划项

**private** **int** curror = -1;//

**private** **int** size = course1.size() ;//计划项的数量

方法: hasNext :判断是否还有下个元素

next：下一个元素

remove方法：本次实验用不上这个方法，所以不需要实现这个方法

实现Comparator方法：





因此，在CourseCalendarBoard类中，只要返回我们实现的迭代器即可。



2. FlightScheduleBoard的迭代器方法

自己实现的迭代器：

属性： **private** List<FlightEntry> flight1 = getFlight2() ;//抵达位置是location的所有航班

**private** List<FlightEntry> flight2 = getFlight1() ;//出发位置是 location的所有航班

**private** **int** curror = -1;

**private** **int** size1 = flight1.size() ; //抵达位置是location的航班个数

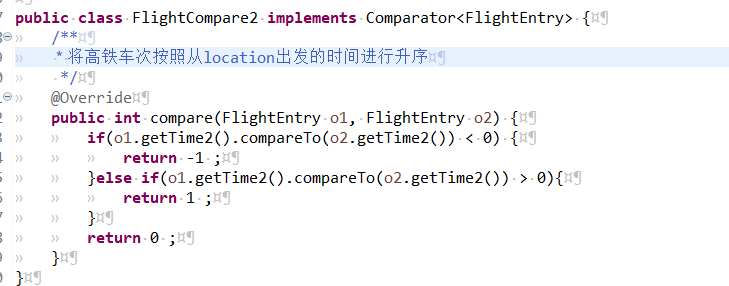
**private** **int** size2 = flight2.size();//出发位置是location的航班的个数

属性说明:由于机场对应抵达航班和出发航班，所以迭代器要遍历所有的抵达航班和出发航班。

方法：hasNext , next方法同上

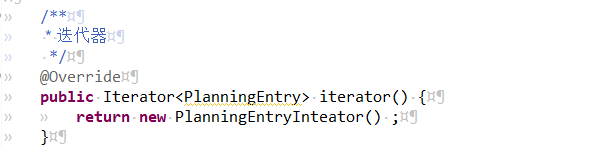
实现Comparator方法：







最后FlightScheduleBoard类中，只要返回我们实现的迭代器即可。



3.TrainScheduleBoard方法

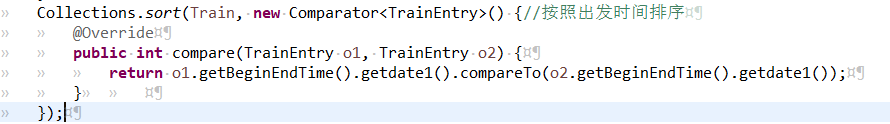
属性：**private** List<TrainEntry> t = getTrain() ;//所以经停location的高铁车次

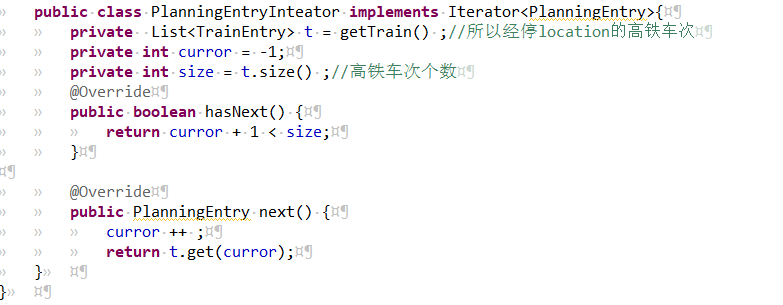
**private** **int** curror = -1;

**private** **int** size = t.size() ;//高铁车次个数

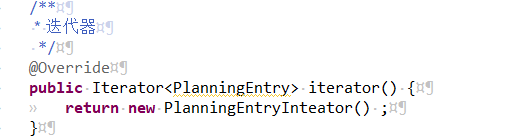
属性说明: 所有经停location的高铁车次已经按照时间排好序

实现Comparator方法：





最后TrainScheduleBoard类中，只要返回我们实现的迭代器即可。

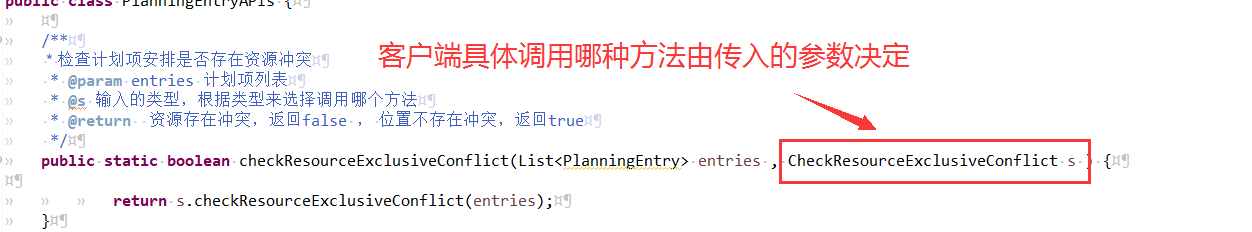


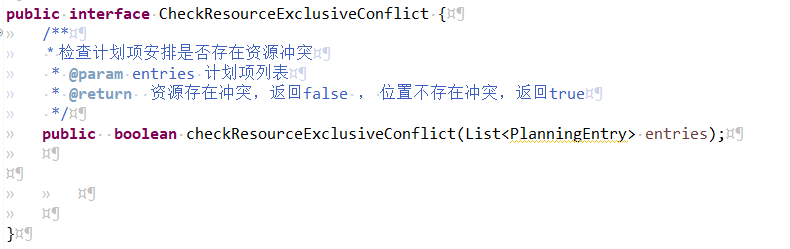
### Strategy

策略模式:同一个功能用不同的方法去实现，具体客户端选择哪种实现由用户决定。

思路：我把策略模式用在了3.9节中判断资源冲突的方法中。首先创建一个公共接口CheckResourceExclusiveConflict.java,两种不同的实现分别是checkResourceExclusiveConflict1.java

和 checkResourceExclusiveConflict2.java。根据传入的参数来选择调用哪个方法来实现。









## 应用设计与开发

### 航班应用

1.FlightSchedule 类

设计思路：所有的方法均是针对APP来设计，方法的参数也是根据实际需求来设计。创建一个航班管理，可以读取文件创建，也可以自己手动输入信息进行创建。

航班管理：航班管理应用由一系列的航班组成，其中包含了一系列的可用飞机和可用机场，并且对应一个当前时间。

属性：

**private** List<Plane> resource = **new** ArrayList<Plane>() ;//可用的资源

**private** List<Location> location = **new** ArrayList<Location>() ;//可用的位置

**private** List<FlightEntry> flightEntry = **new** ArrayList<FlightEntry>() ;//一系列计划项

**private** Calendar time = Calendar.*getInstance*();//当前的时间

主要实现的功能：

⚫用户提供必要信息，管理（查看、增加、删除）可用的资源；

⚫ 用户提供必要信息，管理（查看、增加、删除）可用的位置；

⚫ 用户提供必要信息，增加一条新的计划项；

⚫ 用户提供必要信息，取消某个计划项；

⚫ 用户提供必要信息，为某个计划项分配资源；

⚫ 用户从文件读取数据创建一个航班应用

⚫ 用户提供必要信息，启动某个计划项；

⚫ 用户提供必要信息，以重启动某个已挂起的计划项；

⚫ 用户提供必要信息，结束某个计划项；

⚫ 用户选定一个计划项，查看它的当前状态；

⚫ 检测当前的计划项集合中可能存在的位置和资源独占冲突

⚫ 针对用户选定的某个资源，列出使用该资源的所有计划项（包含尚未开的，始执行的、执行中的、已经结束的）。用户选中其中某个计划项之后，可以找出它的前序计划项

⚫ 选定特定位置，可视化展示当前时刻该位置的信息板

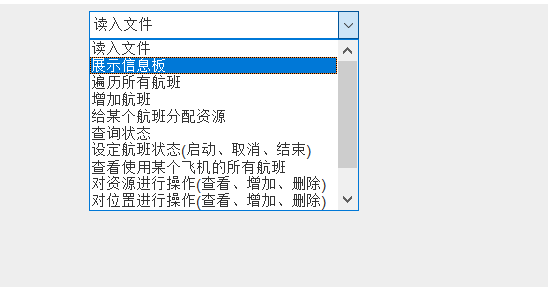
⚫ 展示和某个位置相关的所有计划项信息

方法：



2.TrainScheduleAPP

用GUI图形界面结合展示



我的思路：1.用户可以选择从文件中创建计划项集合，也可以手动输入创建计划项集合。

2.增加一个计划项的时候，只需要分配位置和时间即可，不需要分配资源。而且分配了的位置会自动添加为可用位置。

3.展示资源冲突和位置冲突的时候，提示信息显示在终端，并没有用GUI组件来提示信息，GUI只是提示了有无冲突。

4.对于和状态有关的功能，一般我都指明需要用户输入一个当前时间。这样能够更灵活的展示状态变化。

执行流程： 1.增加可用的资源

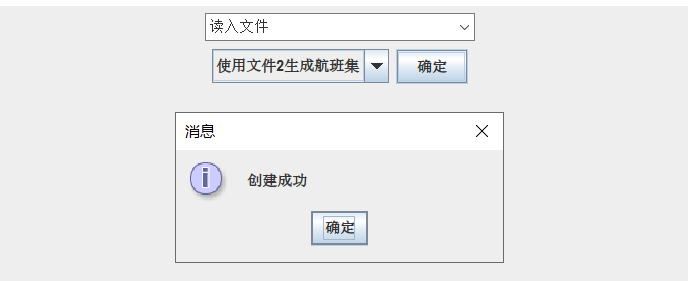
2.增加一个新的计划项（可以通过读取文件创建一系列计划项）

（也可以先2后1，但必须保证计划项分配资源之前得增加可用资源）

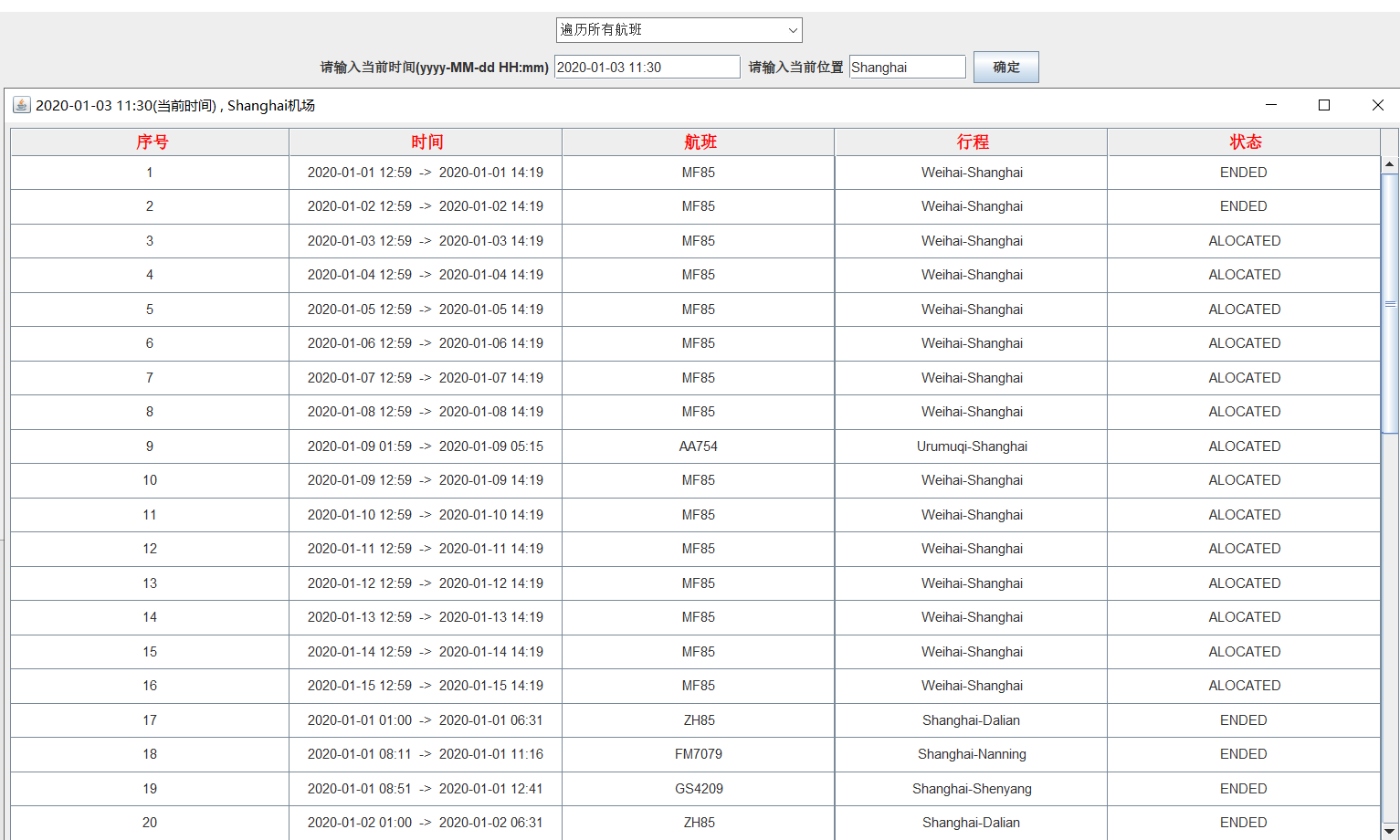
3.给新的计划项分配资源

其他的功能可以随便选择顺序。

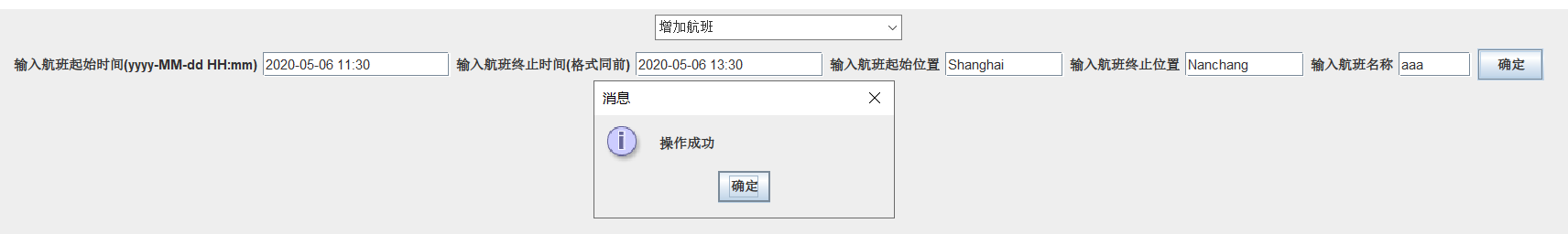
1.读取文件创建

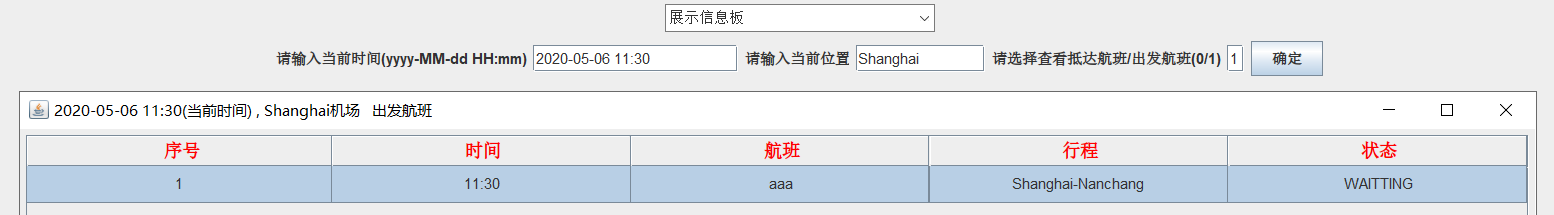




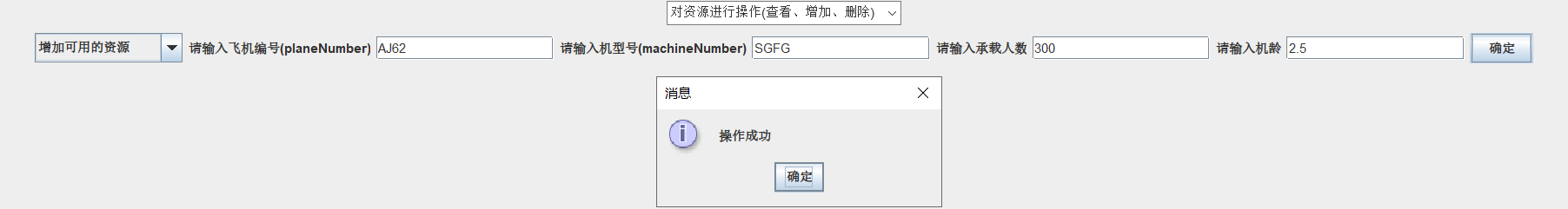


2.手动增加一个计划项

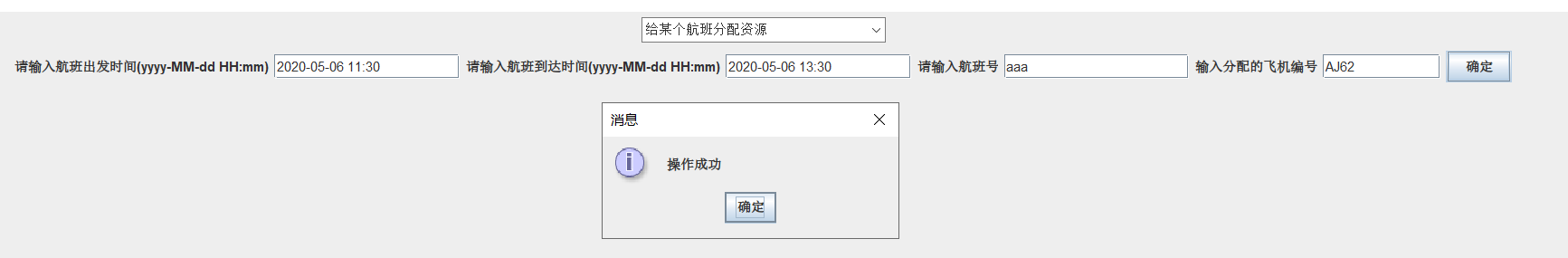




3.增加资源

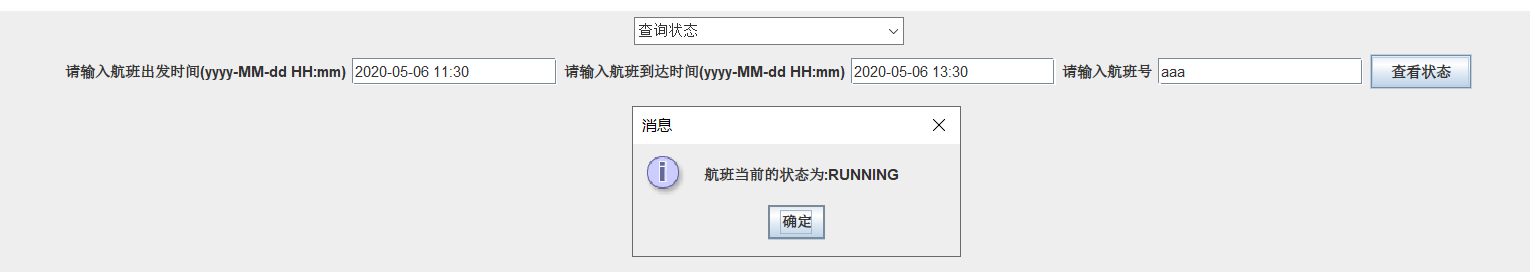


4.分配资源

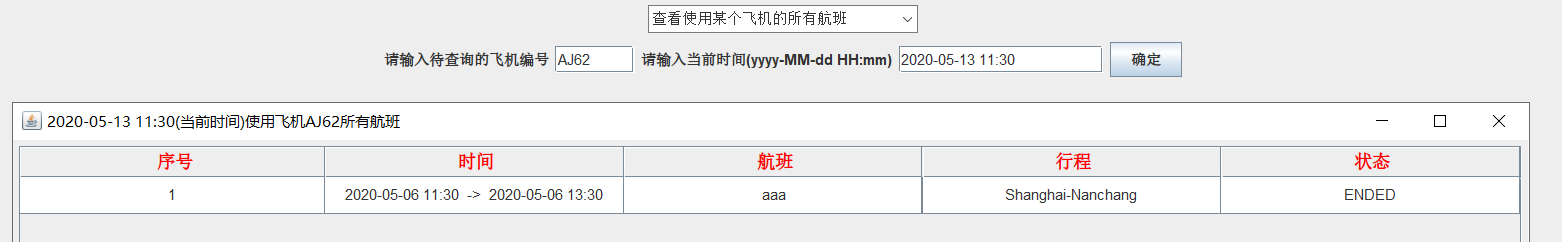




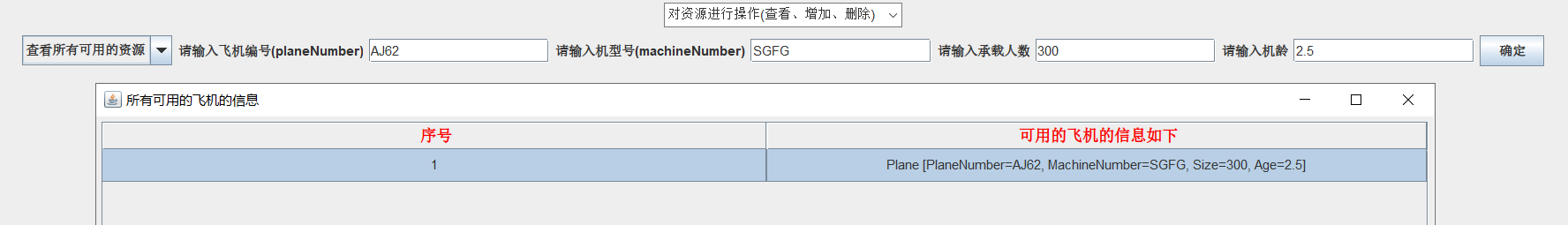
5.查询状态



6.查询使用某个飞机的航班



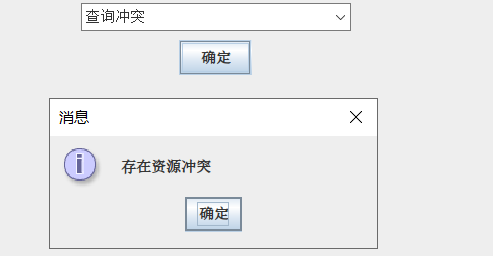
7.查询所有可用的资源



8.查询冲突



当读入文件2进行创建航班集合时：





冲突信息显示在终端上

其他的功能不再一一阐述

### 高铁应用

1.TrainSchedule 类

设计思路：所有的方法均是针对APP来设计，方法的参数也是根据实际需求来设计。和其他的两个类的功能及方法特别类似。

高铁管理：高铁管理应用由一系列的高铁车次组成，其中包含了一系列的可用车厢和可用站，并且对应一个当前时间。

属性：

**private** List<Teacher> resource = **new** ArrayList<Teacher>() ;//可用的资源

**private** List<Location> location = **new** ArrayList<Location>() ;//可用的位置

**private** List<CourseEntry> courseEntry = **new** ArrayList<CourseEntry>() ;//一系列计划项

**private** Calendar time = Calendar.*getInstance*() ;//当前时间

主要实现的功能：

⚫用户提供必要信息，管理（查看、增加、删除）可用的资源；

⚫ 用户提供必要信息，管理（查看、增加、删除）可用的位置；

⚫ 用户提供必要信息，增加一条新的计划项；

⚫ 用户提供必要信息，取消某个计划项；

⚫ 用户提供必要信息，为某个计划项分配资源；

⚫ 用户提供必要信息，阻塞某个计划项；

⚫ 用户提供必要信息，启动某个计划项；

⚫ 用户提供必要信息，以重启动某个已挂起的计划项；

⚫ 用户提供必要信息，结束某个计划项；

⚫ 用户选定一个计划项，查看它的当前状态；

⚫ 检测当前的计划项集合中可能存在的位置和资源独占冲突

⚫ 针对用户选定的某个资源，列出使用该资源的所有计划项（包含尚未开的，始执行的、执行中的、已经结束的）。用户选中其中某个计划项之后，可以找出它的前序计划项

⚫ 选定特定位置，可视化展示当前时刻该位置的信息板

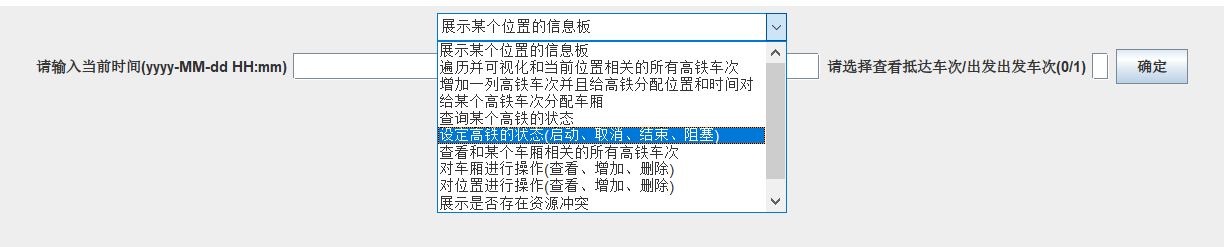
⚫ 展示和某个位置相关的所有计划项信息

方法：



2.TrainScheduleAPP

用GUI图形界面结合展示



我的思路：1.增加一个计划项的时候，需要分配若干个时间对和位置对，我的设计是每次只能增加一对时间和一对位置。如果有多个位置和时间，需要添加多次。

2.分配资源也是一样，每次只能分配一个车厢，如果需要分配多个车厢，则需要分配多次。

3. 展示资源冲突和位置冲突的时候，提示信息显示在终端，并没有用GUI提示信息，GUI只是提示了有无冲突。

4.对于和状态有关的功能，一般我都指明需要用户输入一个当前时间。这样能更灵活的展示状态变化。

提示：由于对GUI设计不是很熟练，界面做的比较难看。每次查看信息后如果点击关闭表格窗口，会导致GUI退出。所以建议老师每次如果输入命令的时候不要把表格的窗口关闭。

执行流程： 1.增加可用的资源

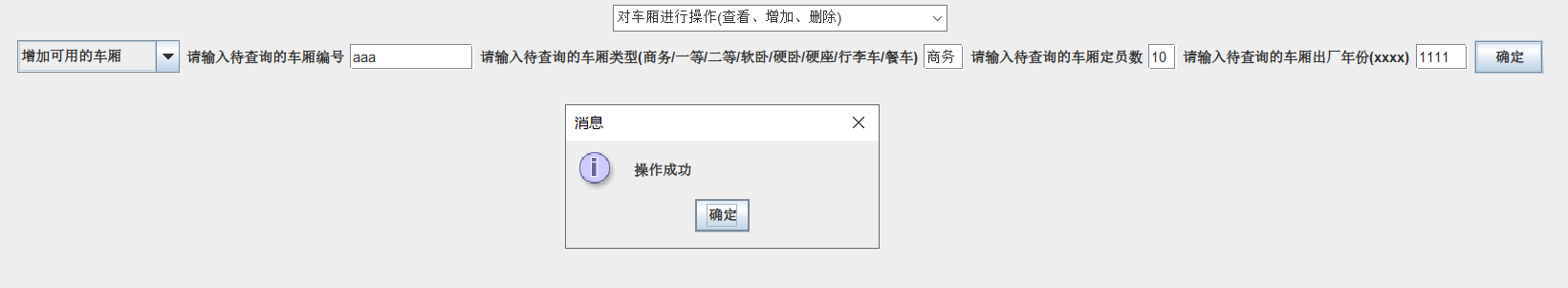
2.增加一个新的计划项

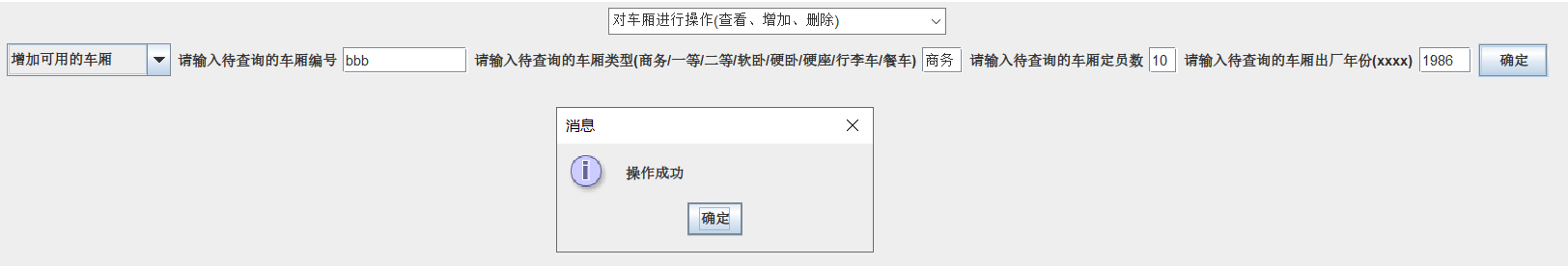
（也可以先2后1，但必须保证计划项分配资源之前得增加可用资源）

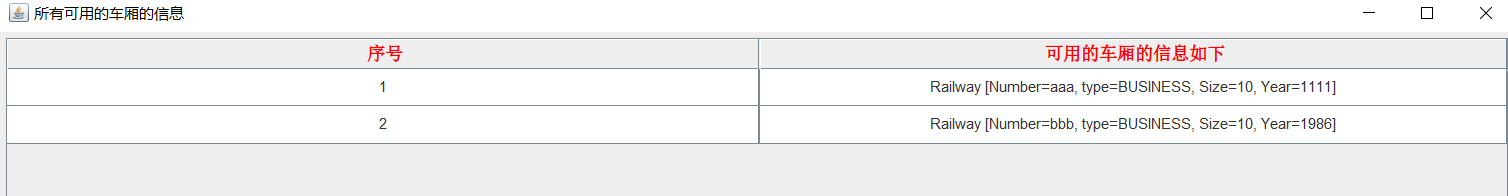
3.给新的计划项分配资源

其他的功能可以随便选择顺序。

1.增加可用的资源

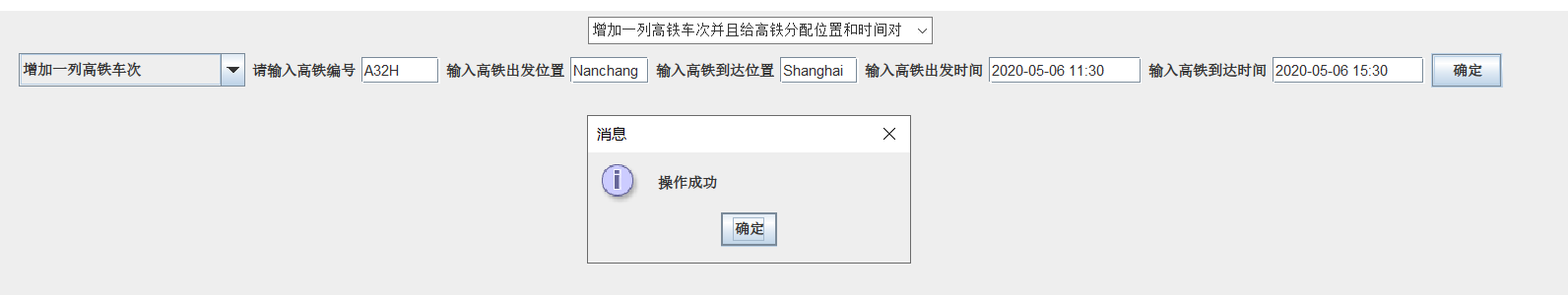






2.增加一个新的计划项

增加一个新的计划项的时候，需要分配若干个时间对和位置对，我的设计是每次只能增加一对时间和一对位置。如果有多个位置和时间，需要添加多次。



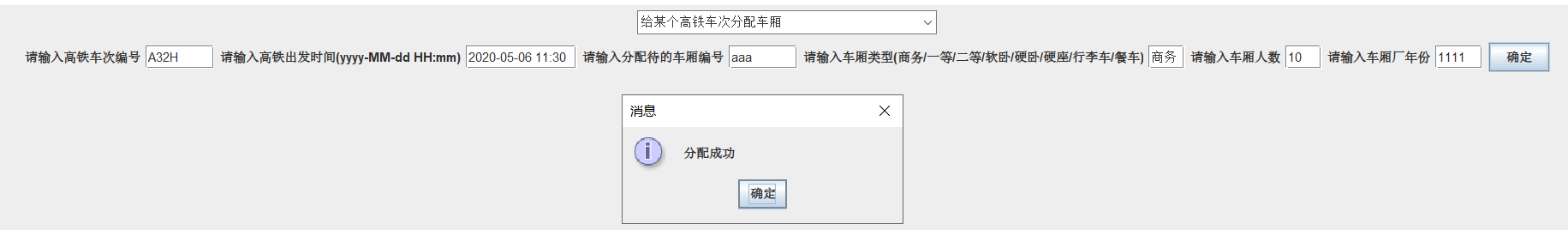
必须保证第一次输入的终止位置和第二次输入的起始位置是一样的

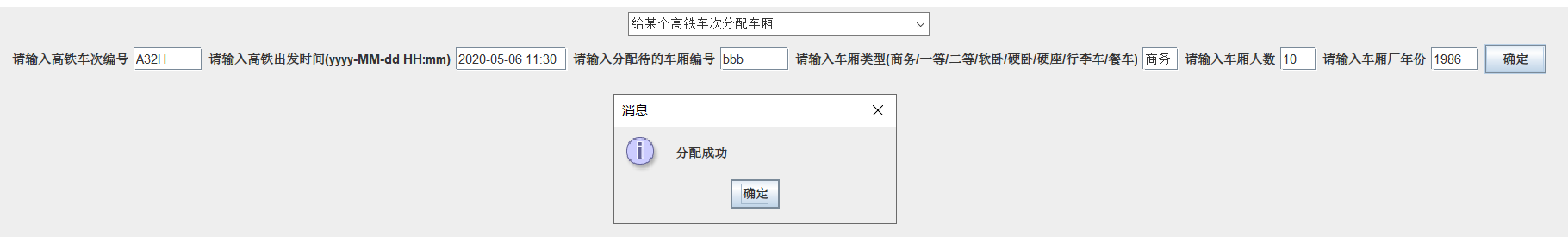




3.给计划项分配资源

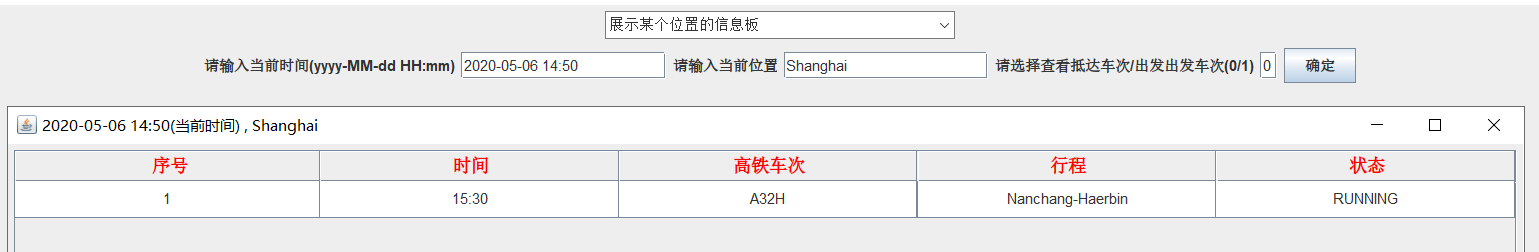
分配资源也是一样，每次只能分配一个车厢，如果需要分配多个车厢，则需要分配多次。每次分配资源的时候都会展示出当前的可用资源



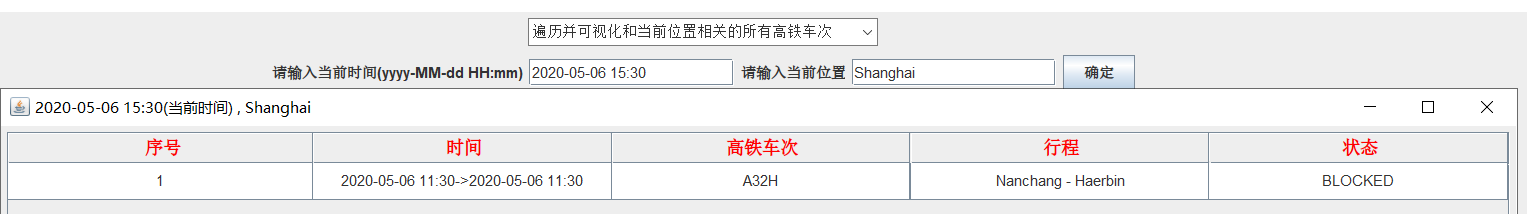


4.展示信息板

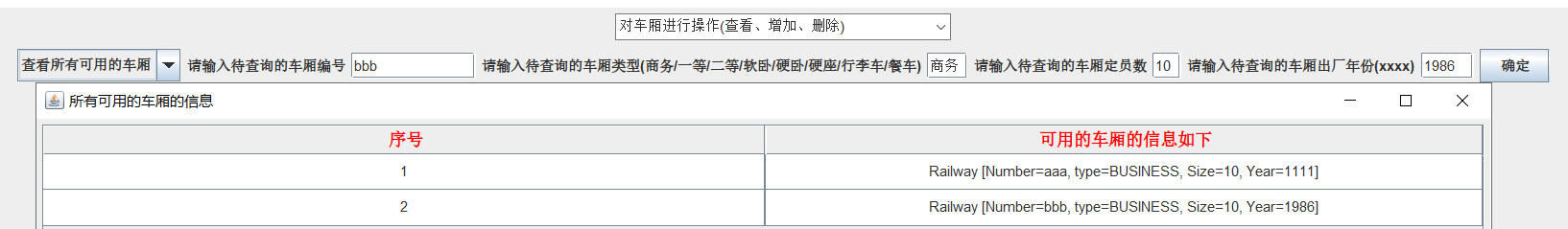




5.遍历和当前位置相关的所有计划项



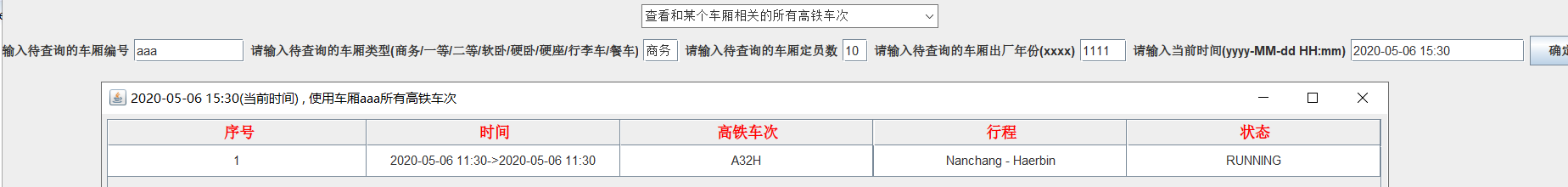
6.查询所有的车厢



7.查询是否存在资源冲突



8.查看使用某个车厢的全部计划项



还有很多功能不再一一阐述，

### 课程应用

1.CourseCalendar 类

设计思路: CourseCalendar类中所有的方法均是针对APP来设计的，参数是根据实际需要来设定的。比如增加一个计划项：需要提供计划项基本的信息:计划项的名字，计划项的位置，计划项的起止时间。启动一个计划项：也需要计划项的基本信息: 计划项的名字，计划项的位置，计划项的起止时间。具体的方法实现就是基本的调用前面的已经实现的方法。(详细见代码)

课表管理:课表应用由一系列的课程组成，其中包含了一系列的可用教师和可用教室，并且对应一个当前时间。

属性：

**private** List<Teacher> resource = **new** ArrayList<Teacher>() ;//可用的资源

**private** List<Location> location = **new** ArrayList<Location>() ;//可用的位置

**private** List<CourseEntry> courseEntry = **new** ArrayList<CourseEntry>() ;//一系列计划项

**private** Calendar time = Calendar.*getInstance*() ;//当前时间

主要实现的的功能：

⚫用户提供必要信息，管理（查看、增加、删除）可用的资源；

⚫ 用户提供必要信息，管理（查看、增加、删除）可用的位置；

⚫ 用户提供必要信息，增加一条新的计划项；

⚫ 用户提供必要信息，取消某个计划项；

⚫ 用户提供必要信息，为某个计划项分配资源；

⚫ 用户提供必要信息，启动某个计划项；

⚫ 用户提供必要信息，以变更某个已存在的计划项的位置；

⚫ 用户提供必要信息，以重启动某个已挂起的计划项；

⚫ 用户提供必要信息，结束某个计划项；

⚫ 用户选定一个计划项，查看它的当前状态；

⚫ 检测当前的计划项集合中可能存在的位置和资源独占冲突

⚫ 针对用户选定的某个资源，列出使用该资源的所有计划项（包含尚未开的，始执行的、执行中的、已经结束的）。用户选中其中某个计划项之后，可以找出它的前序计划项

⚫ 选定特定位置，可视化展示当前时刻该位置的信息板

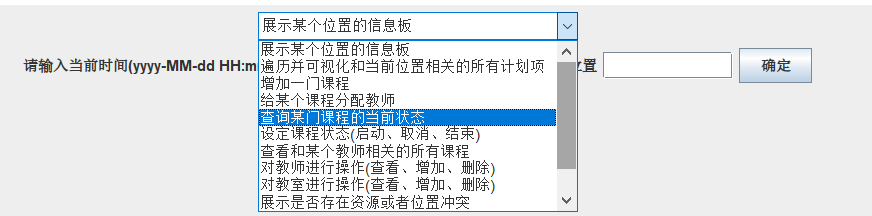
⚫ 展示和某个位置相关的所有计划项信息





2.CourseCalendarAPP

用GUI图形界面展示出来



我的思路：1.增加一个计划项的时候，只需要分配位置和时间即可，不需要分配资源。而且分配了的位置会自动添加为可用位置。

2.展示资源冲突和位置冲突的时候，提示信息显示在终端，并没有用GUI提示信息，GUI只是提示了有无冲突。

3.对于和状态有关的功能，一般我都指明需要用户输入一个当前时间。这样能够更灵活的展示状态变化。

运行流程：1.先增加可用的资源

2.增加一个新的计划项

（也可以先2后1，但必须保证计划项分配资源之前得增加可用资源）

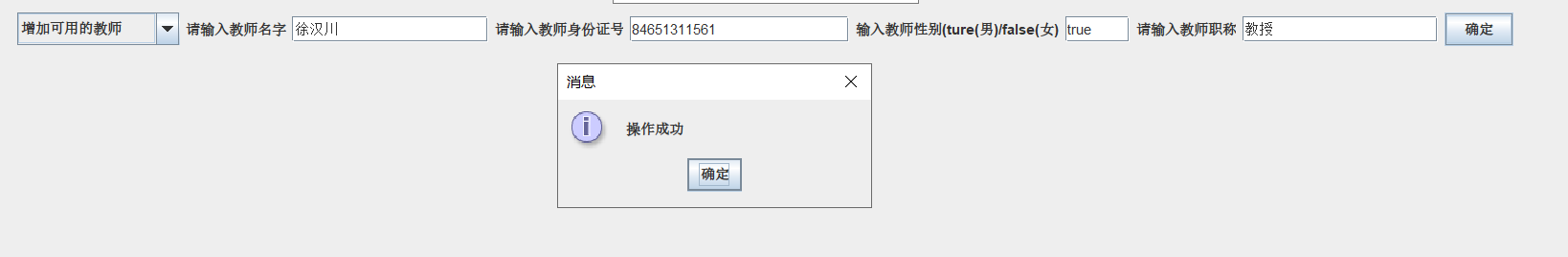
3.给新的计划项分配资源

其他的功能可以随便选择顺序。

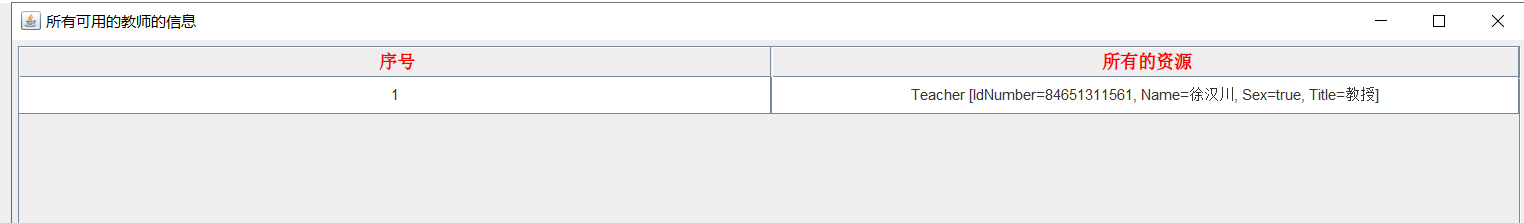
提示：由于对GUI设计不是很熟练，界面做的比较难看。每次查看信息后如果点击关闭表格窗口，会导致GUI退出。所以建议老师每次如果输入命令的时候不要把表格的窗口关闭。

演示：

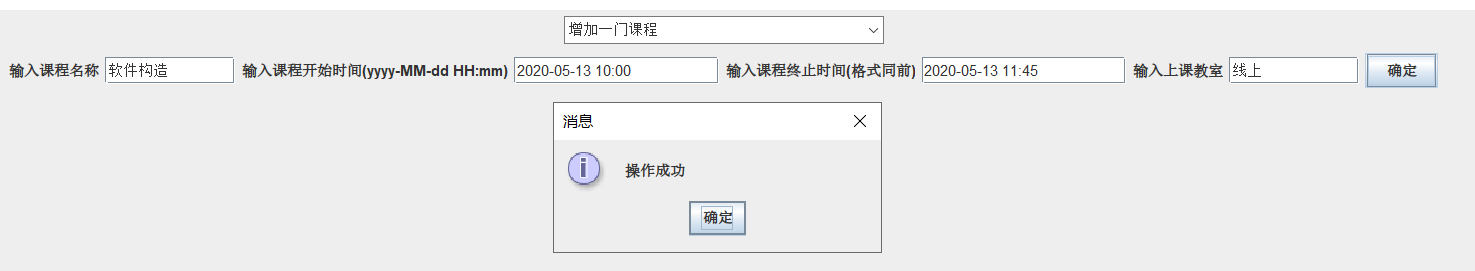
1.增加可用的资源



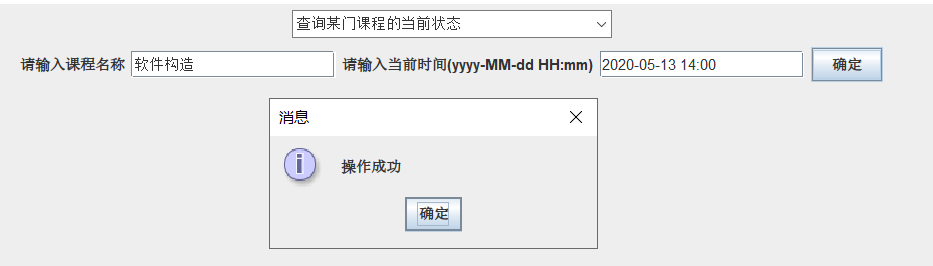
然后查看可用的教师之后：

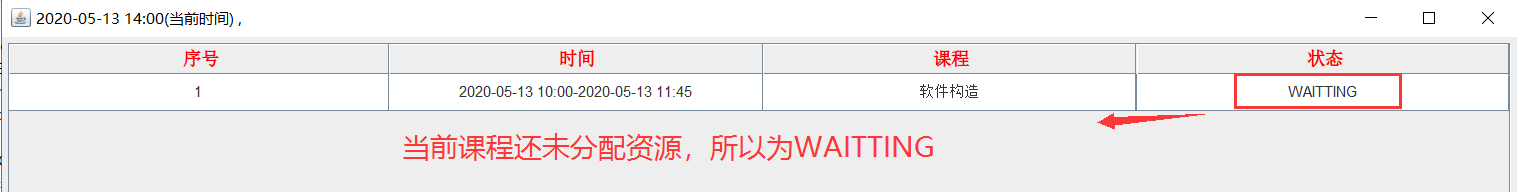


2.增加一个新的计划项



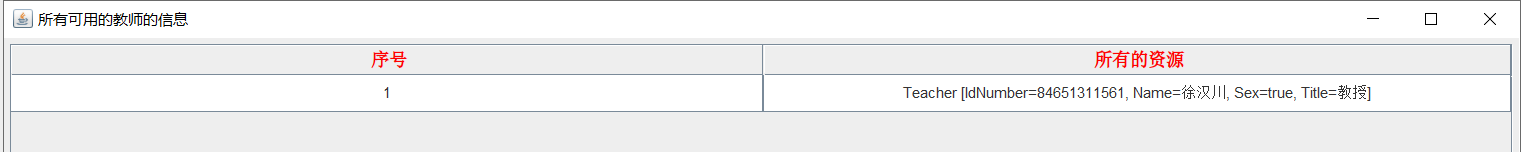
然后查看计划项

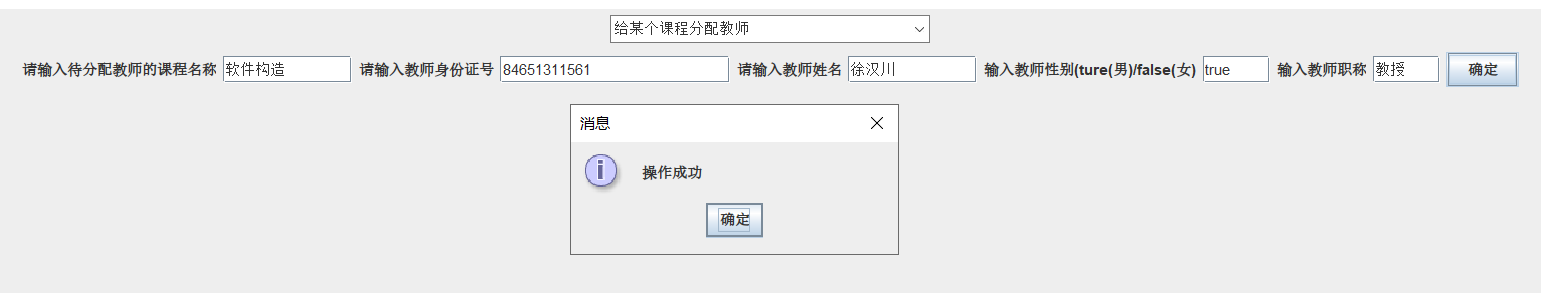




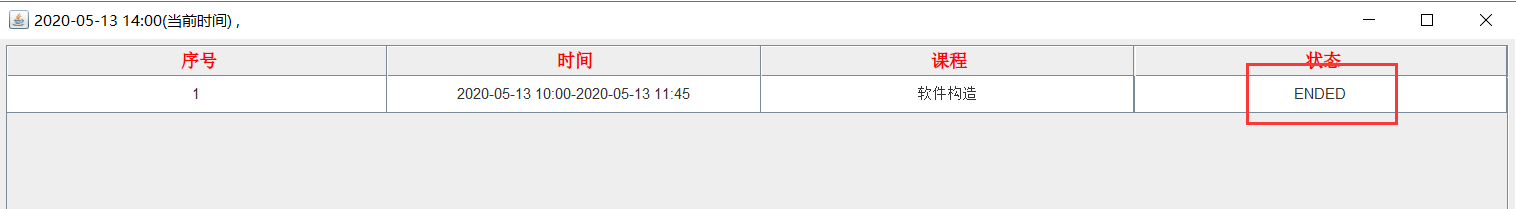
3.给计划项分配资源

每次给计划项分配资源的时候，都会显示所有的可用资源的信息





然后再次查询课程信息



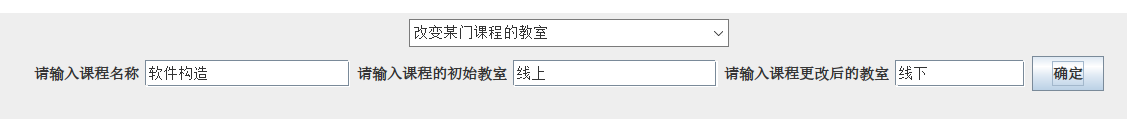
4.展示信息板



5.查询冲突

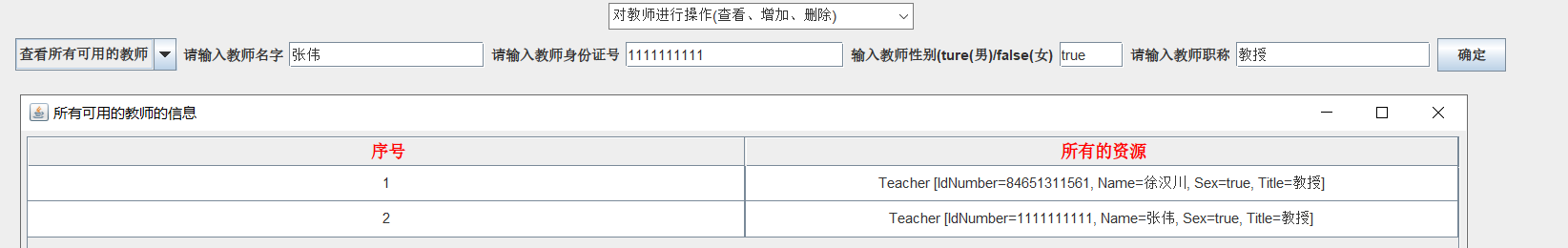


6.改变某门课程的教室



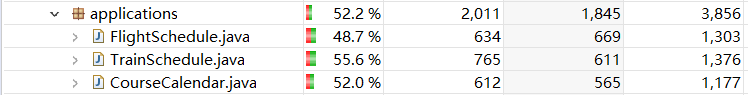


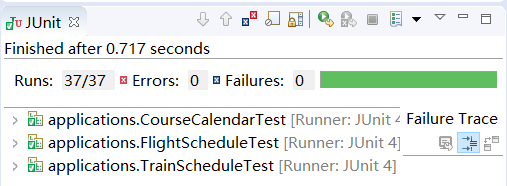
7.查看所有可用的教师



其他的功能就不一一阐述。

测试：

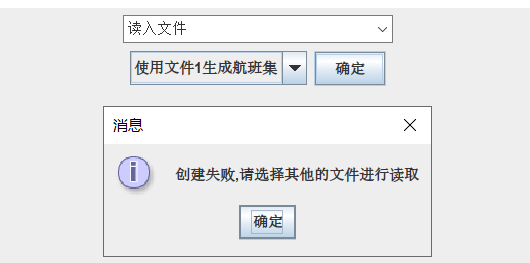


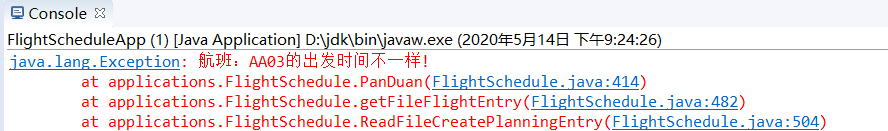


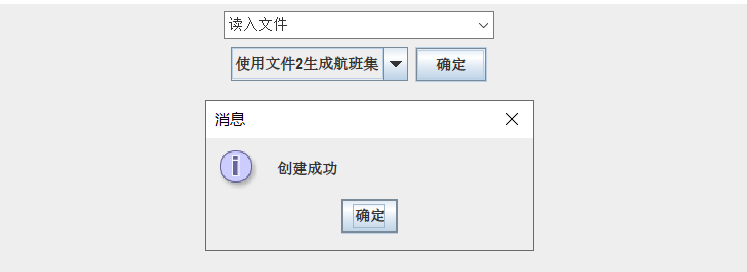
## 基于语法的数据读入

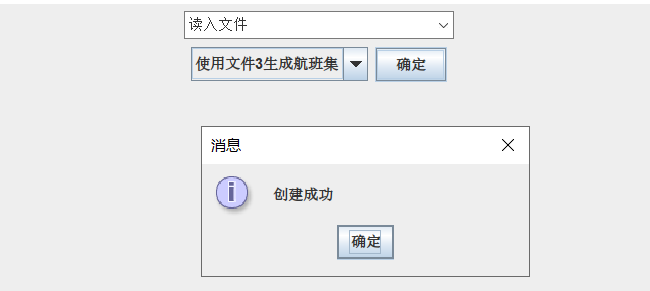
从文件中读取数据创建航班计划项分为两个过程，由于航班数据每13行代表一个计划项，因此每次读取13行就创建一个计划项。所以第一个过程是将13行数据读取到一个字符串中，第二个过程将这个字符串进行解析，提取创建航班所需要的数据创建一个新的航班。

提示：一旦我设计的正则表达式和文件中的数据匹配不到，立即抛出异常，提示错误信息。(经运行，文件1存在和实验指导书指明的语法错误)

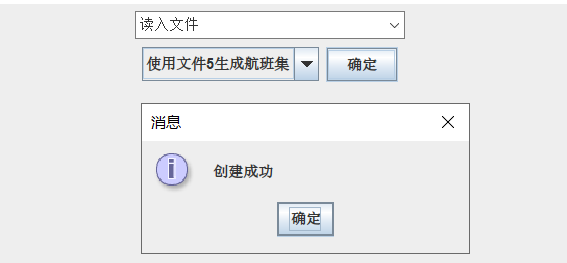












1.从文件中提取13行数据到一个字符串s中，然后调用getFileFlightEntry函数创建一个新的计划项。(跳过空行)



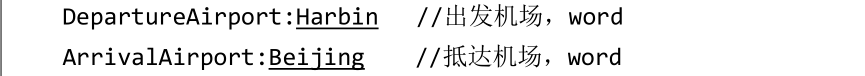
2.利用字符串来提取构建航班需要的信息，匹配的正则表达式为：

Pattern pattern1 =Pattern.*compile*("Flight:([\\d]{4}-[\\d]{2}-[\\d]{2}),([A-Z]{2}[\\d]{2,4})\n\\{\nDepartureAirport:(.\*?)\nArrivalAir port:(.\*?)\nDepatureTime:([0-9]{4}-[0-9]{2}-[0-9]{2}\\s[0-9]{2}:[0-9]{2})\nArrivalTime:([0-9]{4}-[0-9]{2}-[0-9]{2}\\s[0-9]{2}:[0-9]{2})\nPlane:([B || N][0-9]{4})\n\\{\nType:([0-9a-zA-Z]{4})\nSeats:([1-9][\\d]{1,2})\nAge:(.\*?)\n\\}\\n\\}\n");



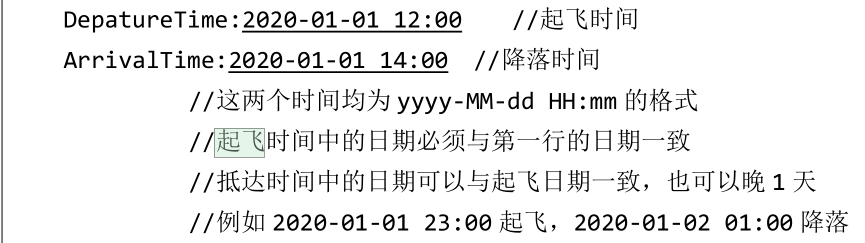
1.使用的匹配航班计划项的时间和航班号的正则表达式为：

"Flight:([\\d]{4}-[\\d]{2}-[\\d]{2}),([A-Z]{2}[\\d]{2,4})\n"



2.匹配机场位置：机场位置是一个字符串，没有指明元素类型，因此正则表达式为：

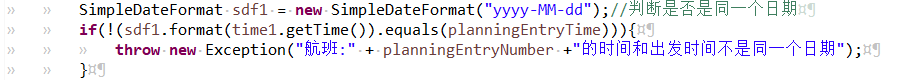
"DepartureAirport:(.\*?)\nArrivalAir port:(.\*?)\n"

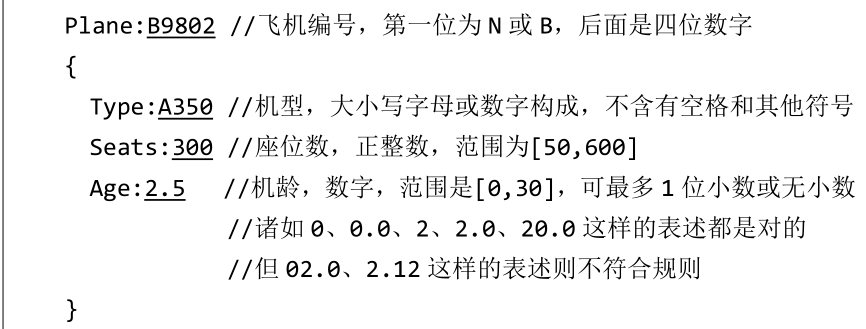


3.匹配起止时间格式的正则表达式为：

"DepatureTime:([0-9]{4}-[0-9]{2}-[0-9]{2}\\s[0-9]{2}:[0-9]{2})\nArrivalTime:([0-9]{4}-[0-9]{2}-[0-9]{2}\\s[0-9]{2}:[0-9]{2})\n"

注意：判断起飞时间的日期和第一行的日期是否一致：





4.匹配飞机编号的正则表达式为：

"Plane:([B || N][0-9]{4})\n"

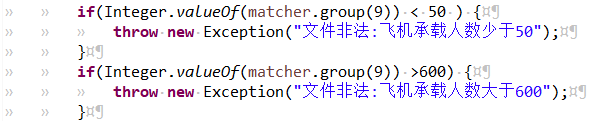
5.匹配机型的正则表达式为

"Type:([0-9a-zA-Z]{4})\n"

6.匹配座位数的正则表达式为：

Seats:([1-9][\\d]{1,2})\nAge:(.\*?)\n

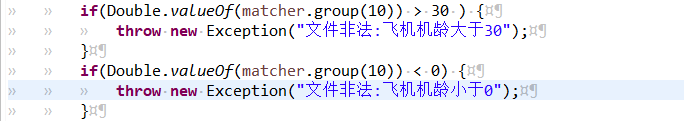
注意：判断座位数是否符合要求：



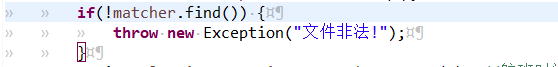
匹配机龄的正则表达式为

"Age:(.\*?)\n

注意：判断机龄是否符合要求：



利用正则表达式去和提取出来的字符串进行匹配，若匹配失败，直接抛出异常并提示信息

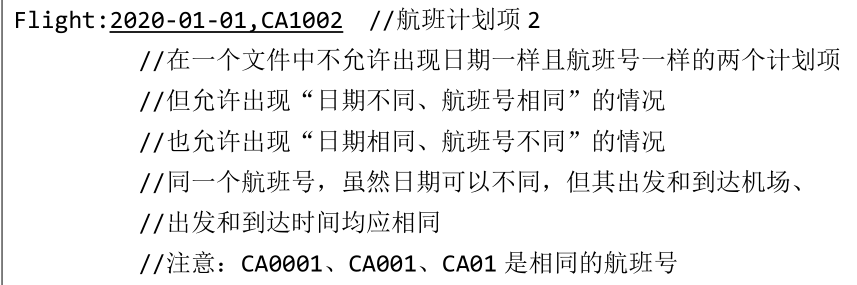


否则，则提取字符串中的信息来创建新的计划项。



一直循环这两个步骤就可以将文件中的所有信息提取出来

判断航班的时间是否符合指导书要求







## 应对面临的新变化

只考虑你选定的三个应用的变化即可。

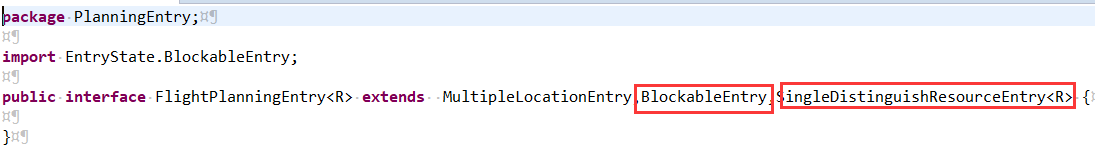
### 变化1



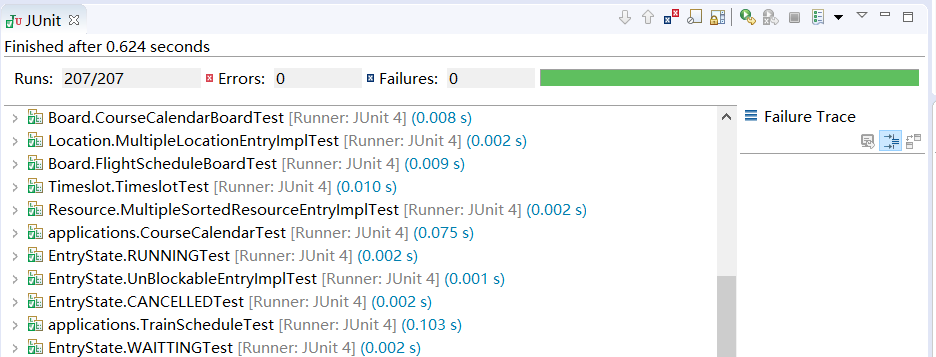
变化1:单一位置变成多个位置 ,因此接口组合中的单个位置的接口变为多个位置的接口

变化2：不可阻塞变为可阻塞，因此接口组合中的不可阻塞的接口变为可阻塞的接口。

改变:首先根据两个变化改变一下接口组合。



其次，重写一下新增的接口的方法。然后重写一下相关的类的测试方法。



航班change最主要需要改的地方是board，因为时间和位置都变成了多个，所以对位置的和时间的一一对应的信息的获取工作量更大。

主要change：1.重写获取和某个位置相关的出发航班和抵达航班的方法

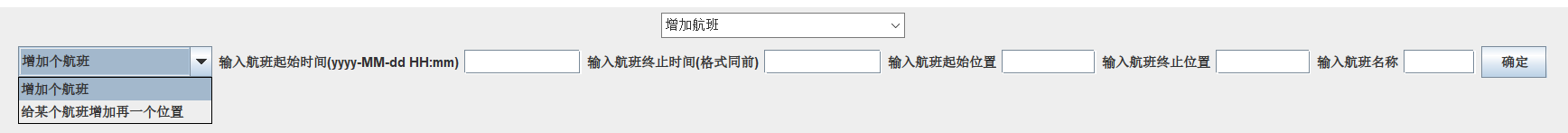
2.稍微改动visualize方法和show方法

3.对于改变状态的方法需要增加blocked状态的改变。(详细见代码)

另外一个change是:1.控制一个航班的最大的位置数目：因为最多只有一个经停站，所以对于一个航班来说，分配的位置最多数目是3.

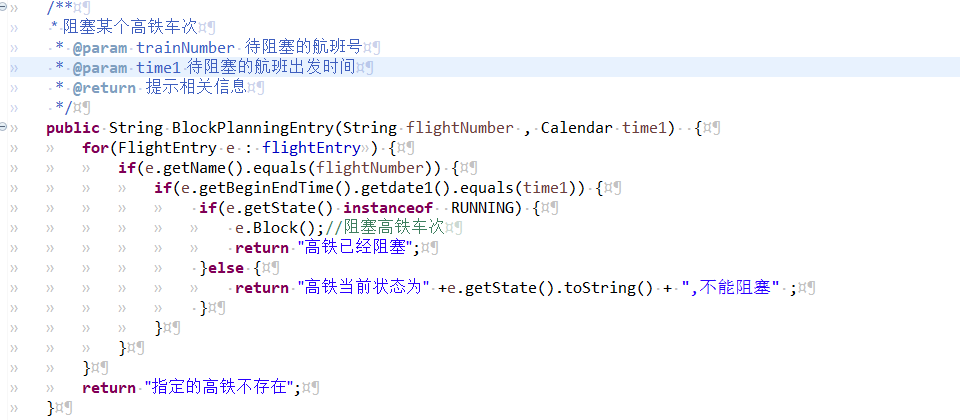


2.和TrainEntryApp一样，如果航班有经停站的话，需要一次一次输入，即每次输入只能输入两个位置和两个时间。



输入异常的条件有:时间格式错误，第一个时间第二个时间大，第一次输入的终止位置和第二次输入的起始位置不相同，为同一个航班输入的位置超过3(经停站超过1)

3.航班增加了阻塞的功能。





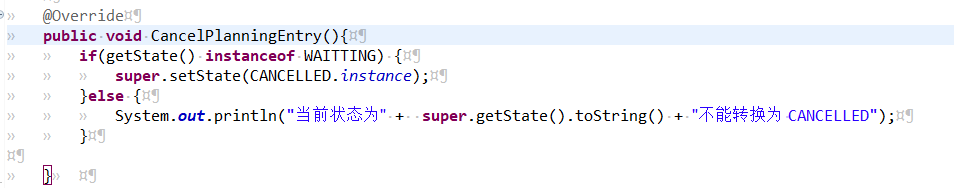
修改时间:3个小时

### 变化2



变化之前，高铁车次分配了资源仍可以取消，可以直接调用状态类的方法实现。

变化之后，高铁车次分配了资源不可以取消，则不能直接调用状态类的方法，为了保证代价更小，无需重新为ALOCATED状态类重新设计changestate方法，只需要trainEntry类只要重写取消计划项的方法就行。



测试: 

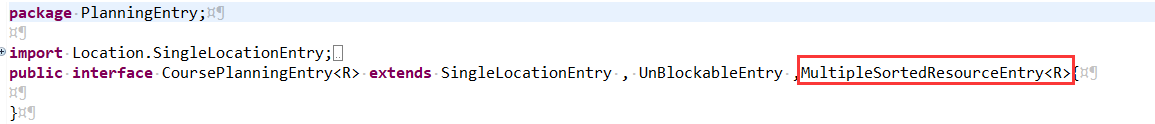


修改耗费的时间:10分钟

### 变化3

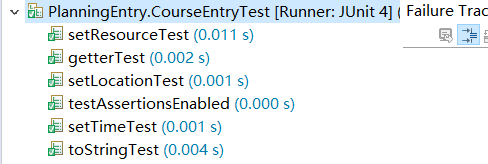


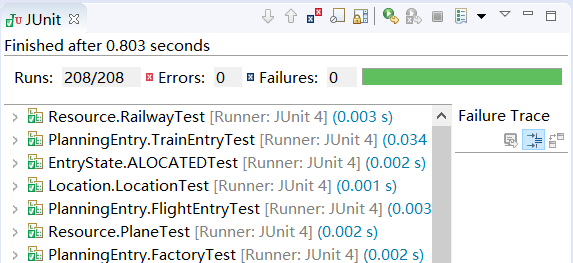
变化1：由于我采用的是方案5，接口组合，所以courseEntry的接口组合中单一资源的接口得改变为多资源的接口,同时，我默认为区分老师的优先级是从给某门课程分配教师的顺序来区分。



同时得重写多资源接口中的方法、以及测试类、所有用到courseEntry中和资源操作有关的代码。

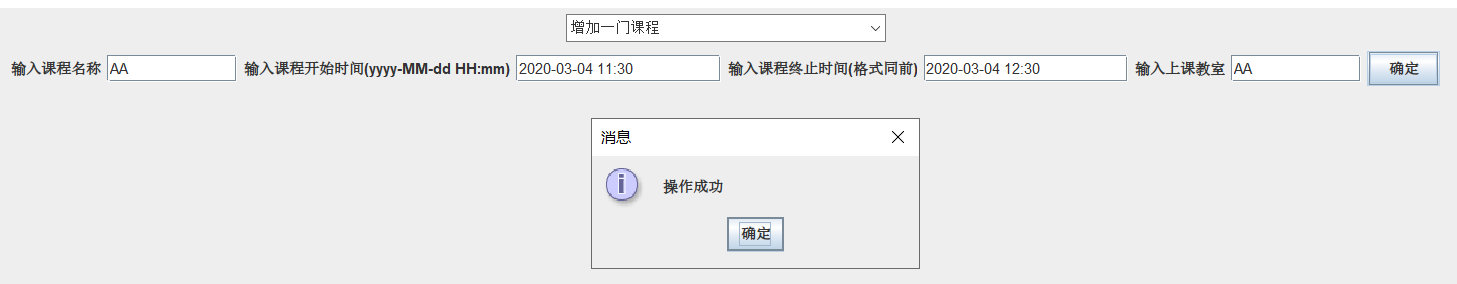
重写CourseEntryTest方法

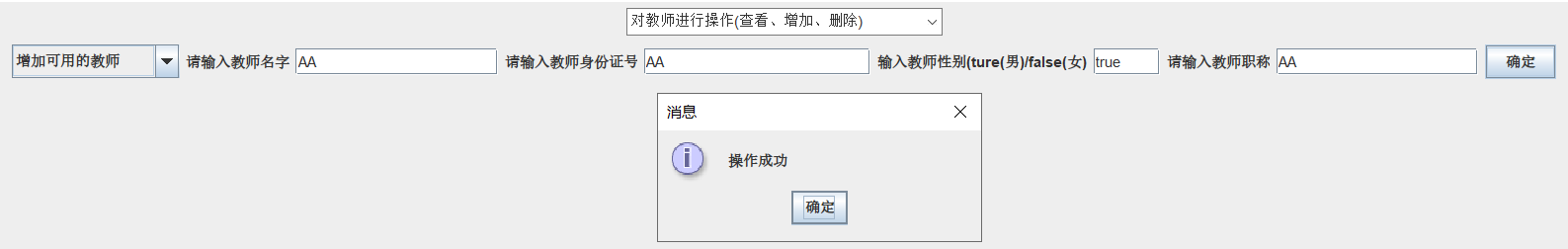


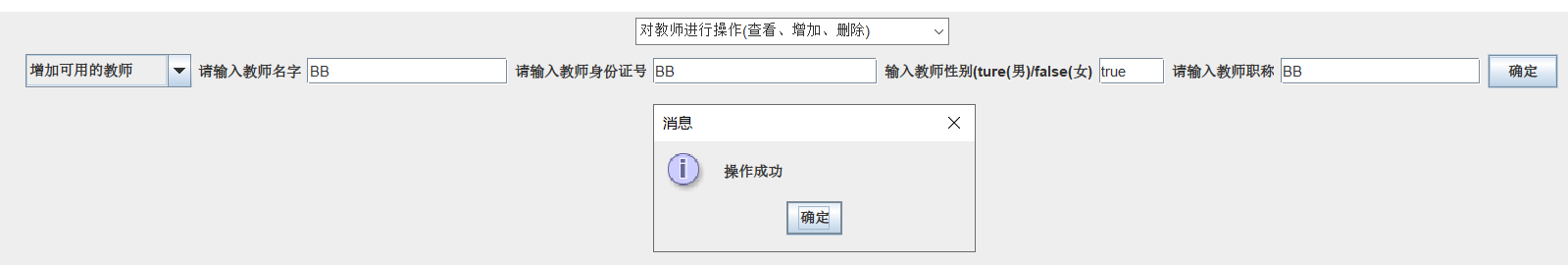


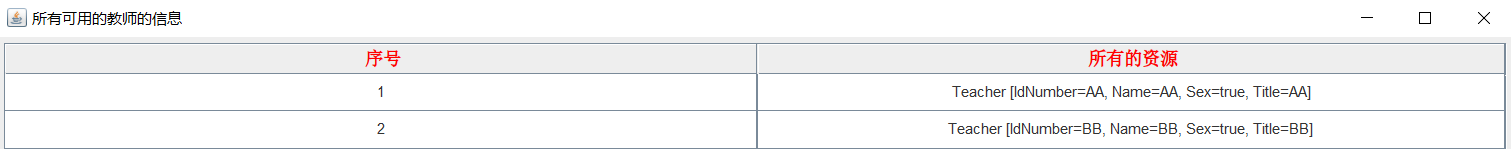
APP不需要修改代码，只是给某门课程分配教师的功能改变了，原来只能分配一个教师，现在可以分配多个教师。

如图所示

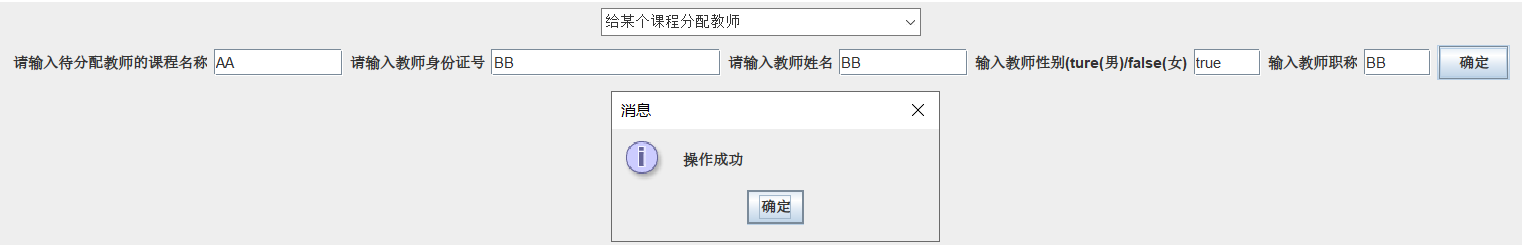


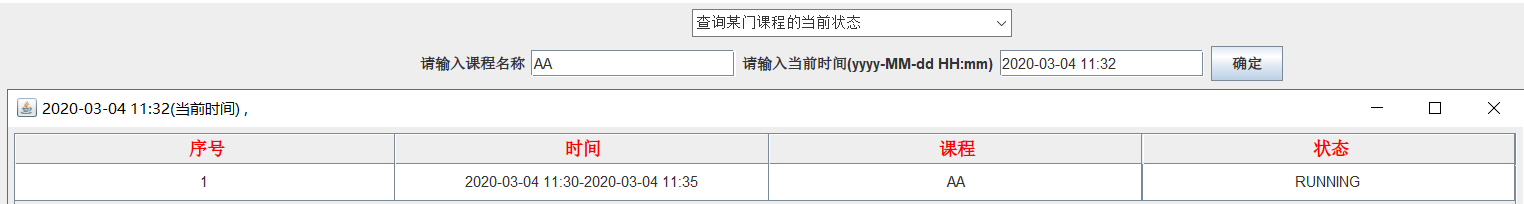


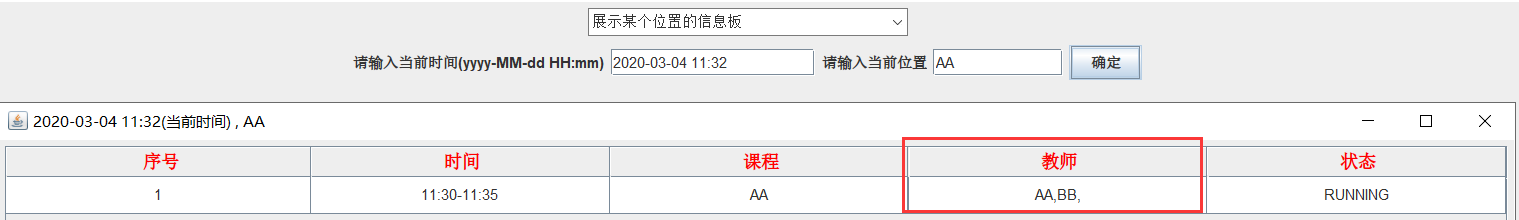






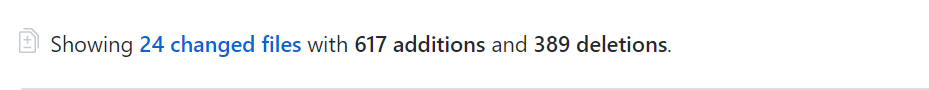






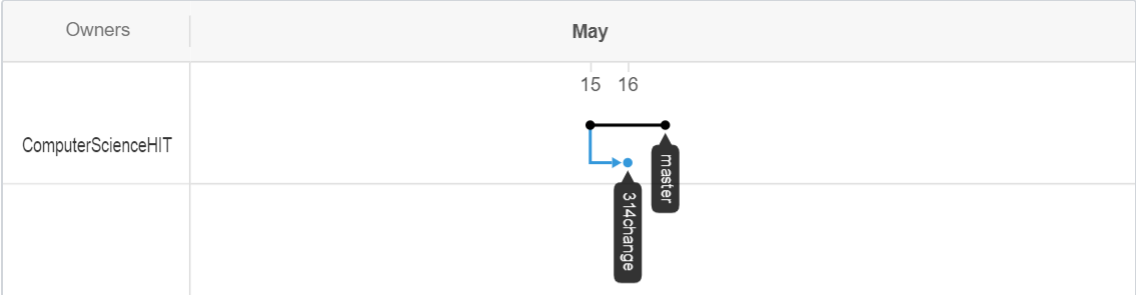
修改耗费的时间:不到1个小时

所有的change：



## 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314change分支和master分支所指向的位置。



# 实验进度记录

请使用表格方式记录你的进度情况，以超过半小时的连续编程时间为一行。

每次结束编程时，请向该表格中增加一行。不要事后胡乱填写。

不要嫌烦，该表格可帮助你汇总你在每个任务上付出的时间和精力，发现自己不擅长的任务，后续有意识的弥补。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 2020-04-20 | 晚上 | 理解题意 | 完成 |
| 2020-04-21 | 下午 | 完成所有的类的组织 | 完成 |
| 2020-04-23 | 晚上 | 设计基本的Timeslot类 | 完成 |
| 2020-04-25 | 晚上 | 设计基本的Location类和Resource类 | 完成 |
| 2020-04-26 | 晚上 | 完成基本类的测试 | 完成 |
| 2020-04-28 | 下午 | 完成planningEntry类的设计 | 提前完成 |
| 2020-05-03 | 晚上 | 完成TrainEntry等子类的设计 | 未完成 |
| 2020-05-04 | 上午 | 完成具体的子类的设计 | 完成 |
| 2020-05-05 | 全天 | 完成state设计模式 | 完成 |
| 2020-05-06 | 全天 | 完成board类设计 | 未完成 |
| 2020-05-07 | 全天 | 完成board类设计 | 完成 |
| 2020-05-08 | 全天 | 完成API及其迭代器 | 完成 |
| 2020-05-09 | 全天 | 完成应用类的设计并且自学GUI | 未完成 |
| 2020-05-10 | 全天 | 自学GUI并设计第一个APP | 完成 |
| 2020-05-10 | 晚上 | 设计第二个APP | 完成 |
| 2020-05-11 | 下午+晚上 | 设计第三个APP | 完成 |
| 2020-05-12 | 全天 | 完善注释、测试类、规约的编写 | 完成 |
| 2020-05-13 | 全天 | 完成实验报告 | 完成 |
| 2020-05-15 | 晚上 | 完成314change | 完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 对于Calendar类不了解 | 网上搜，学习 |
| 题意不理解，如何将位置和时间联系起来 | 自己思考并且和同学讨论 |
| 可阻塞的block方法的参数不理解，卡了很久 | 问同学，问老师 |
| Jtable不会 | 在网上自学 |
| GUI不会 | 在网上自学 |
| API设计如何通过PlanningEntry接口获取位置资源和时间相关信息 | 在接口中增加公共方法，返回资源，位置等 |
| Trainboard中如何获取和某个位置相关的时间及其计划项 | Map(Calendar,TrainEntry)来解决 |
| 迭代器不会 | 网上自学 |

# 实验过程中收获的经验、教训、感想

## 实验过程中收获的经验和教训

## 针对以下方面的感受

1. 重新思考Lab2中的问题：面向ADT的编程和直接面向应用场景编程，你体会到二者有何差异？本实验设计的ADT在五个不同的应用场景下使用，你是否体会到复用的好处？

面向应用的编程不仅要考虑面向ADT编程的问题，比如规约、不变量。更多的考虑是程序的复用性和健壮性。复用性很重要，314change中，如果复用性很好，修改的代码会很少很少。

1. 重新思考Lab2中的问题：为ADT撰写复杂的specification, invariants, RI, AF，时刻注意ADT是否有rep exposure，这些工作的意义是什么？你是否愿意在以后的编程中坚持这么做？

一是为意义就是保持不变量，防止泄露，这是保证后置条件正确的基本要求，同时防止外部因素改变内部的不变量。 规约是为了用户能够清楚理解我的程序的作用，这个自然很重要，以后的编程也必须得写。

1. 之前你将别人提供的API用于自己的程序开发中，本次实验你尝试着开发给别人使用的API，是否能够体会到其中的难处和乐趣？

难处:自己从0开发任务量真的特别特别大，需要考虑的问题很多很多，乐趣就是完成之后有一种自豪感。

1. 在编程中使用设计模式，增加了很多类，但在复用和可维护性方面带来了收益。你如何看待设计模式？

设计模式提高了程序的复用性和可维护性，追求高内聚、低耦合，健壮性更强。虽然需要写很多类，但是减少了很多重复的工作，类与类之间的影响也变得更小，便于发现和定位错误，提高正确性。

1. 你之前在使用其他软件时，应该体会过输入各种命令向系统发出指令。本次实验你开发了一个解析器，使用语法和正则表达式去解析输入文件并据此构造对象。你对语法驱动编程有何感受？

正则表达式太强大了，解析文件并且提取信息，更加的便捷。

1. Lab1和Lab2的大部分工作都不是从0开始，而是基于他人给出的设计方案和初始代码。本次实验是你完全从0开始进行ADT的设计并用OOP实现，经过五周之后，你感觉“设计ADT”的难度主要体现在哪些地方？你是如何克服的？

设计ADT的难度在于构建好类之间的框架，结构和厘清类与类之间关系。同时构思好每个功能应该怎么去求解。同时需要掌握扎实的Java编程知识，比如这次GUI就花费了我很多时间求学习，只学了一点皮毛。同时要选择合理的设计模式，这样不仅可以降低工作量和难度，也便于程序的复用。

1. “抽象”是计算机科学的核心概念之一，也是ADT和OOP的精髓所在。本实验的五个应用既不能完全抽象为同一个ADT，也不是完全个性化，如何利用“接口、抽象类、类”三层体系以及接口的组合、类的继承、设计模式等技术完成最大程度的抽象和复用，你有什么经验教训？

抽象出公共的方法放在顶层的接口中很重要，他很大程度的决定了程序的复用性，接口的组合很神奇，体现了高内聚、低耦合的思想，类之间的影响很小，类的继承也可以看作是复用已有的类来进行编程，缩短了开放时间。设计模式的运用最为强大。不管是状态模式、迭代器模式，感觉将更多的模式运用到程序找那个，程序的复用性会大大提高，314change给了我很深的感受。

1. 关于本实验的工作量、难度、deadline。

工作量很大，难度中等，deadline适合。(不得不说重复性的东西太多了，比如我的GUI和applications，很多方法都很类似)

1. 到目前为止你对《软件构造》课程的评价。

软件构造太强了，感受到设计模式的魅力。