

**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 | 许倩 |
| 学号 | 1183710104 |
| 班号 | 1837101 |
| 电子邮件 | 417716852@qq.com |
| 手机号码 | 15537237025 |

**目录**

[1 实验目标概述 1](#_Toc37611518)

[2 实验环境配置 1](#_Toc37611519)

[3 实验过程 1](#_Toc37611520)

[3.1 待开发的三个应用场景 1](#_Toc37611521)

[3.2 面向可复用性和可维护性的设计：PlanningEntry<R> 1](#_Toc37611522)

[3.2.1 PlanningEntry<R>的共性操作 1](#_Toc37611523)

[3.2.2 局部共性特征的设计方案 2](#_Toc37611524)

[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） 2](#_Toc37611525)

[3.3 面向复用的设计：R 2](#_Toc37611526)

[3.4 面向复用的设计：Location 2](#_Toc37611527)

[3.5 面向复用的设计：Timeslot 2](#_Toc37611528)

[3.6 面向复用的设计：EntryState及State设计模式 2](#_Toc37611529)

[3.7 面向应用的设计：Board 2](#_Toc37611530)

[3.8 Board的可视化：外部API的复用 2](#_Toc37611531)

[3.9 可复用API设计及Façade设计模式 2](#_Toc37611532)

[3.9.1 检测一组计划项之间是否存在位置独占冲突 2](#_Toc37611533)

[3.9.2 检测一组计划项之间是否存在资源独占冲突 2](#_Toc37611534)

[3.9.3 提取面向特定资源的前序计划项 2](#_Toc37611535)

[3.10 设计模式应用 2](#_Toc37611536)

[3.10.1 Factory Method 3](#_Toc37611537)

[3.10.2 Iterator 3](#_Toc37611538)

[3.10.3 Strategy 3](#_Toc37611539)

[3.11 应用设计与开发 3](#_Toc37611540)

[3.11.1 航班应用 3](#_Toc37611541)

[3.11.2 高铁应用 3](#_Toc37611542)

[3.11.3 进程应用 3](#_Toc37611543)

[3.11.4 课表应用 3](#_Toc37611544)

[3.11.5 学习活动应用 3](#_Toc37611545)

[3.12 基于语法的数据读入 3](#_Toc37611546)

[3.13 应对面临的新变化 3](#_Toc37611547)

[3.13.1 变化1 3](#_Toc37611548)

[3.13.2 变化2 4](#_Toc37611549)

[3.13.3 变化3 4](#_Toc37611550)

[3.14 Git仓库结构 4](#_Toc37611551)

[4 实验进度记录 4](#_Toc37611552)

[5 实验过程中遇到的困难与解决途径 4](#_Toc37611553)

[6 实验过程中收获的经验、教训、感想 5](#_Toc37611554)

[6.1 实验过程中收获的经验和教训 5](#_Toc37611555)

[6.2 针对以下方面的感受 5](#_Toc37611556)

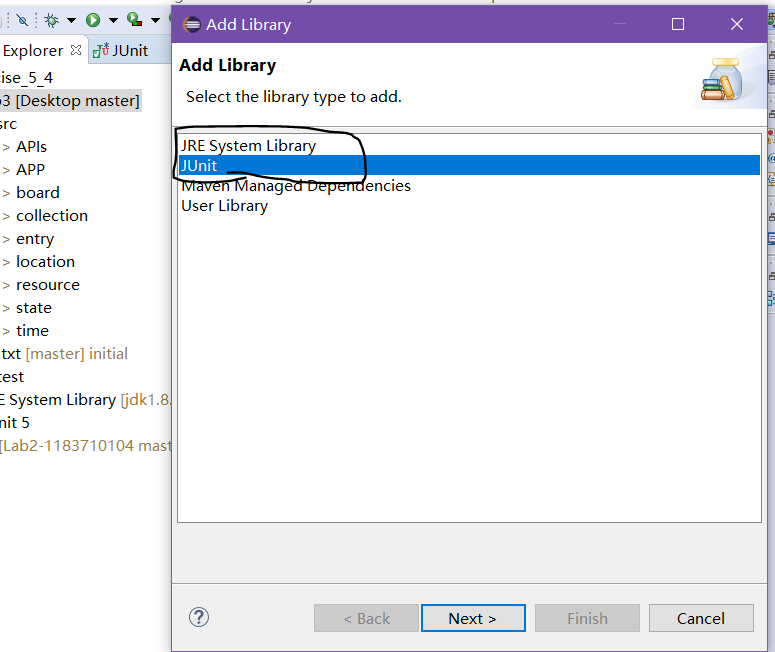
# 实验目标概述

目标：使用3，4，5章所学内容编写具有可复用性和可维护性的软件。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）

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

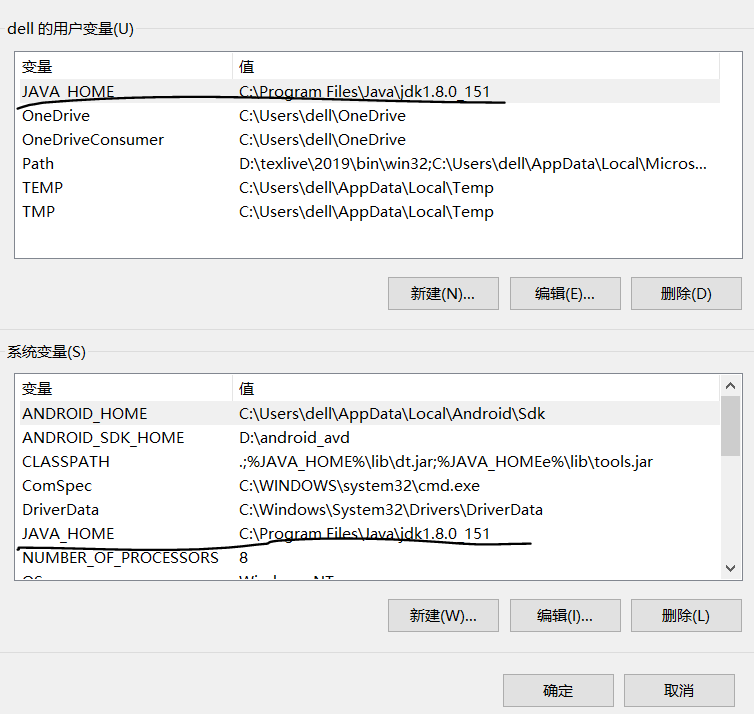
# 实验环境配置

Eclipse IDE配置：项目下添加前两个library



Git bash

系统环境变量配置：



在这里给出你的GitHub Lab3仓库的URL地址

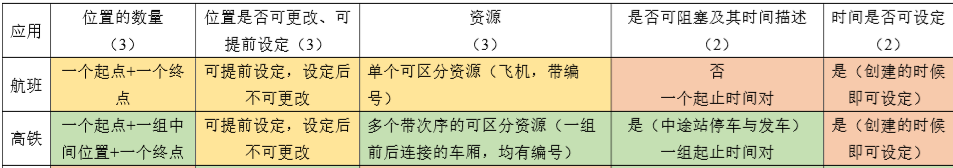
https://github.com/ComputerScienceHIT/Lab3-1183710104.git

# 实验过程

## 待开发的三个应用场景

1. 航班管理
2. 高铁车次管理
3. 大学课表管理

三者的相同和不同如下图：

## 面向可复用性和可维护性的设计：PlanningEntry<R>

### PlanningEntry<R>的共性操作

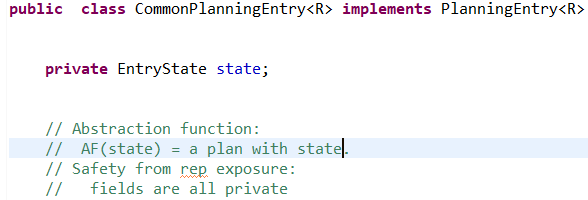
主要思想：采用decorator模式，共性操作在CommomPlanningEntry<R>类中。

PlanningEntry<R>接口定义以下方法：



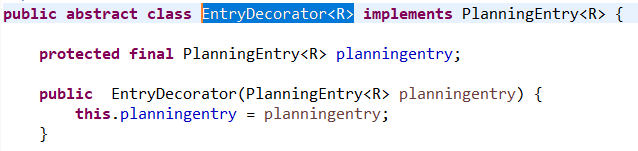
### 局部共性特征的设计方案

共性操作在CommomPlanningEntry<R>类中具体实现，



并增加装饰器类

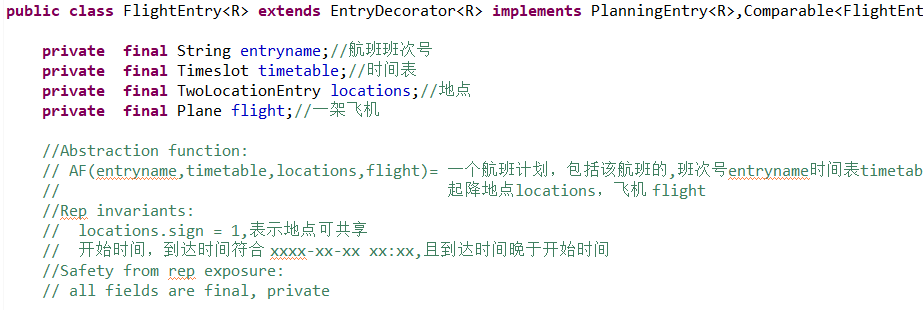
EntryDecorator<R>



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

#### FlightEntry<R>

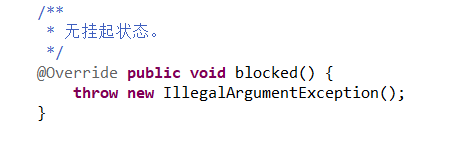
该类为immutable



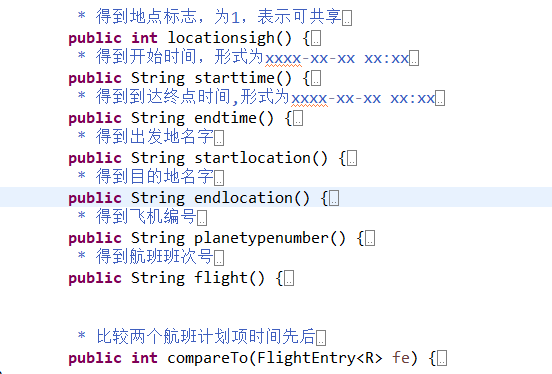
共性方法采用decorator模式实现：



其中，无挂起状态



除了共性方法，其他方法为：



#### TrainEntry<R>

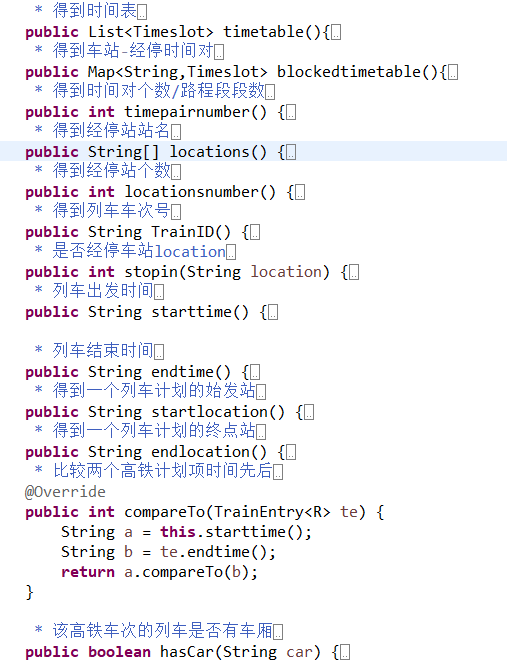
该类为immutable



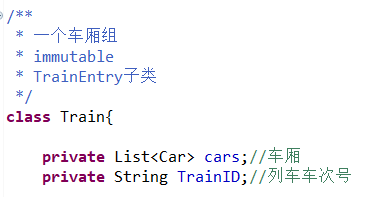
共性方法采用decorator模式实现



个性方法：

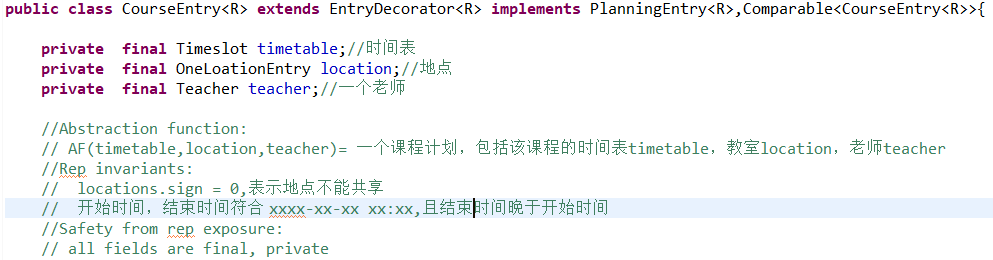


且包含子类Train



#### CourseEntry<R>

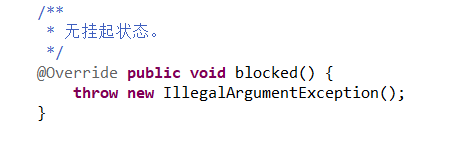
该类为immutable



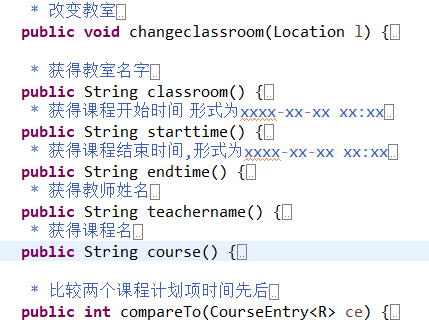
共性方法采用decorator模式实现：



其中，无挂起状态

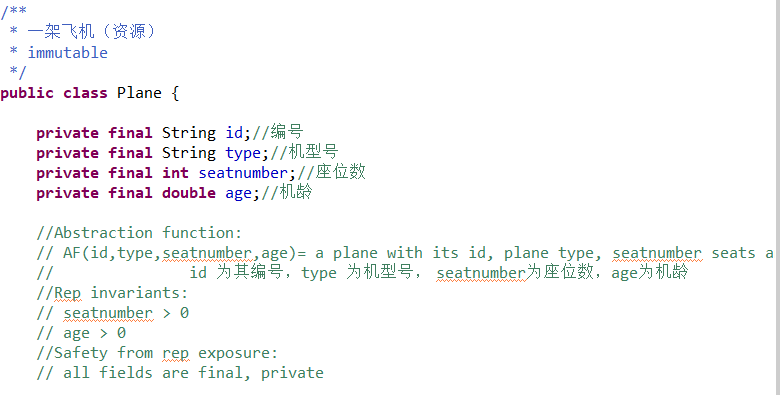


个性方法：

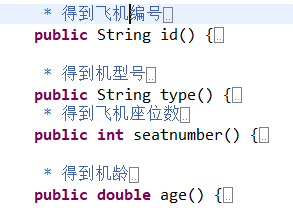


## 面向复用的设计：R

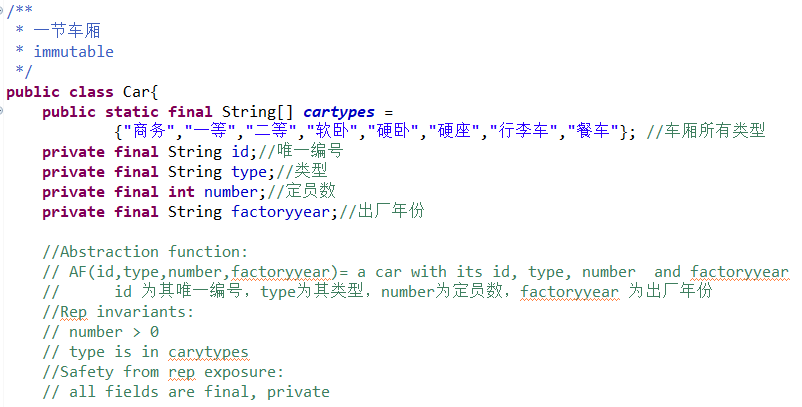
### Plane



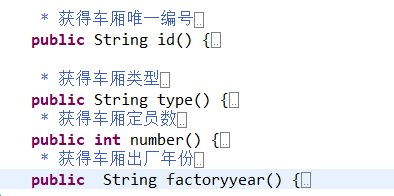
方法：



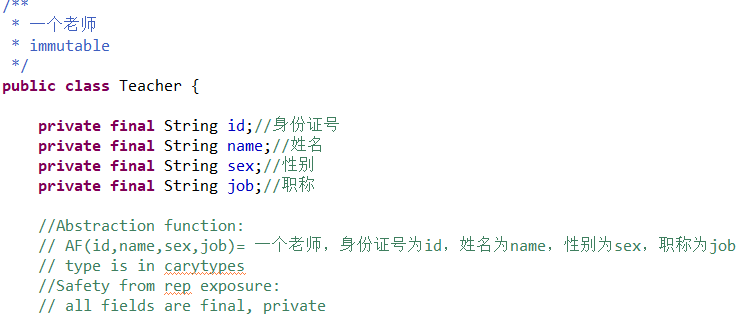
### Car



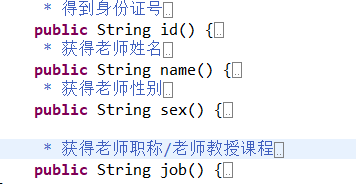
方法：



### Teacher

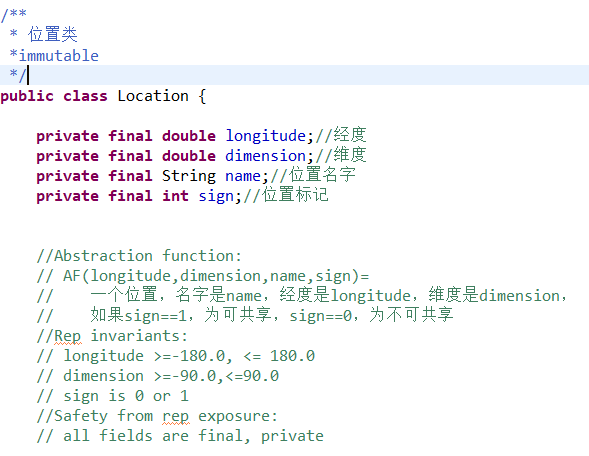


方法：

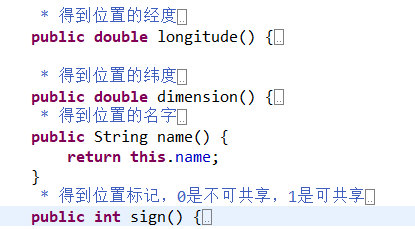


## 面向复用的设计：Location

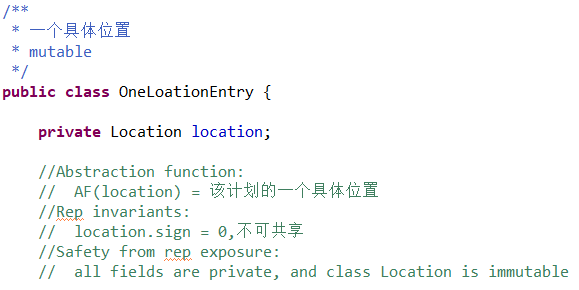
### Location



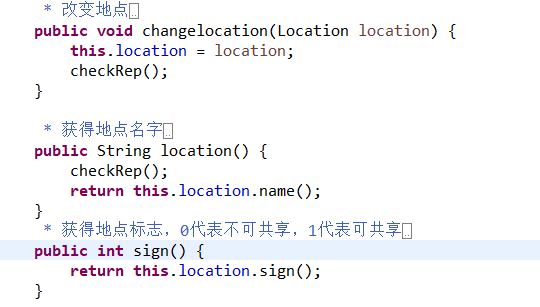
方法：



### OneLoationEntry



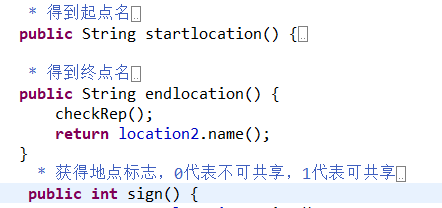
方法：



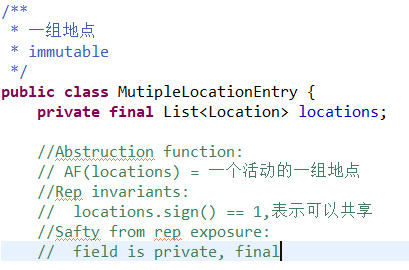
### TwoLocationEntry



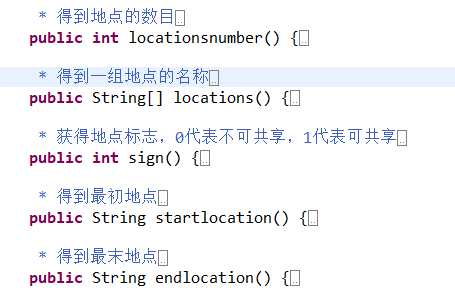
方法：



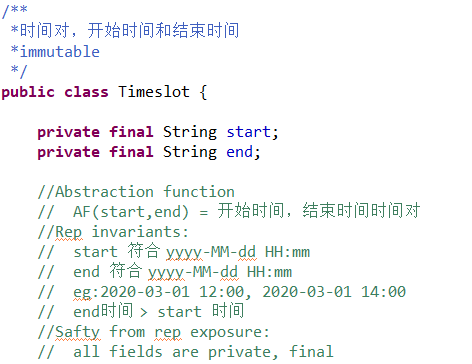
### MutipleLocationEntry



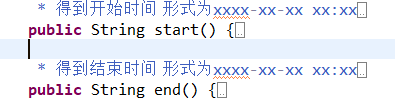
方法：



## 面向复用的设计：Timeslot

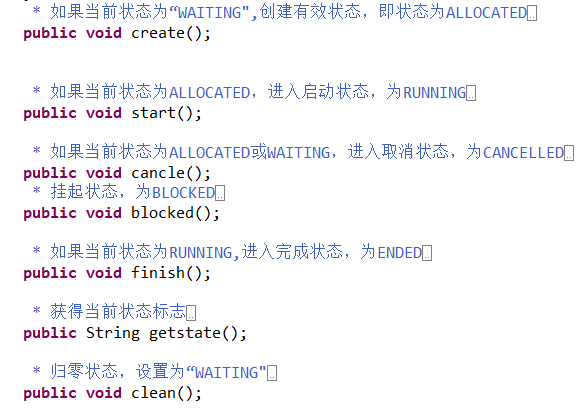


方法：

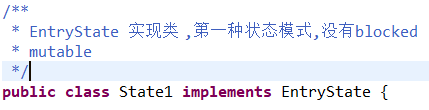


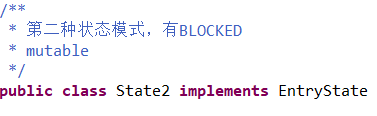
## 面向复用的设计：EntryState及State设计模式

EntryState：状态接口，定义方法



两个实现类：



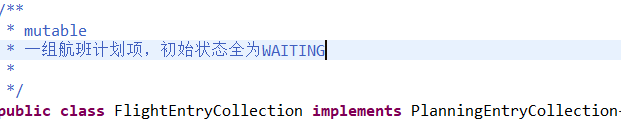


## 面向应用的设计（新增）：计划集合Collection

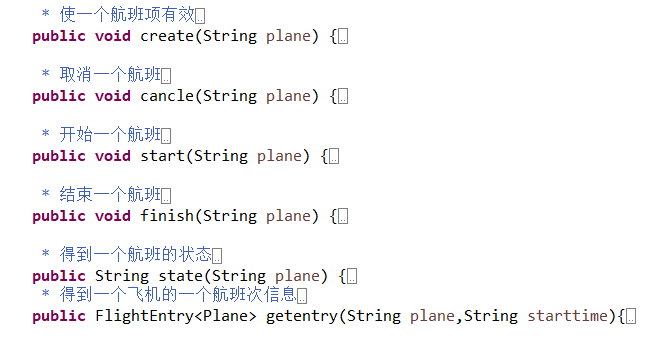
### PlanningEntryCollection



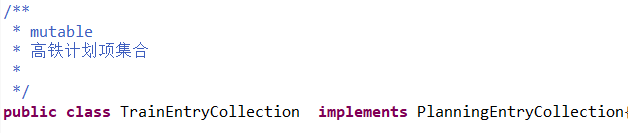
### FlightEntryCollection



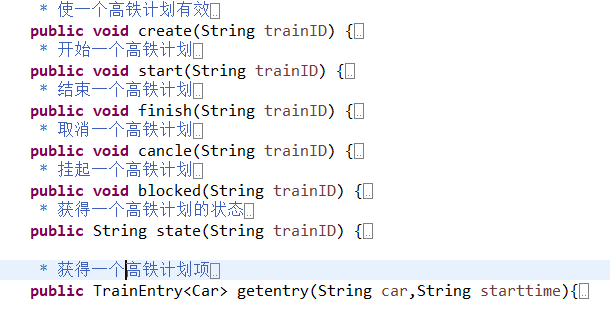
新方法：



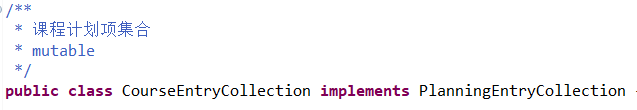
### TrainEntryCollection



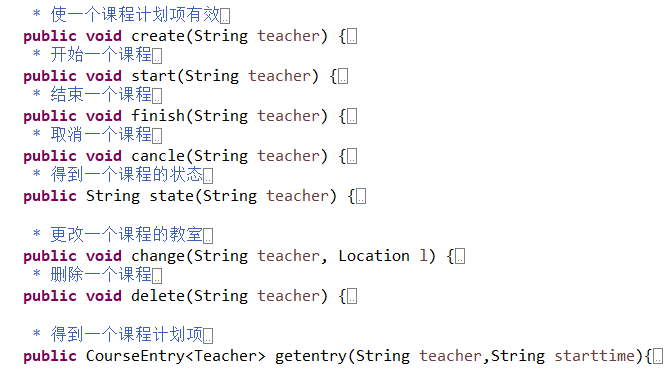
新方法：



### CourseEntryCollection



新方法：



## 面向应用的设计：Board

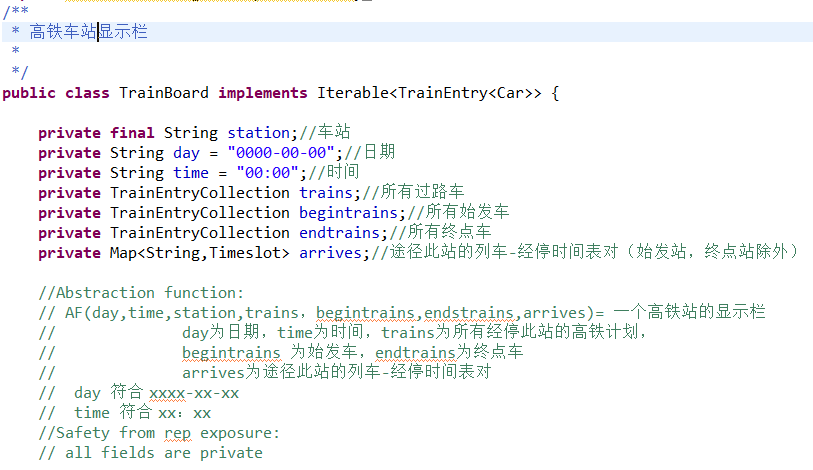
### FlightBoard



方法：



### TrainBoard



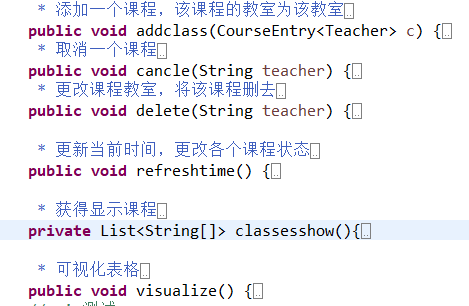
方法:



### CourseBoard



方法：



## Board的可视化：外部API的复用

**外部类：**

**import javax.swing.JFrame;**

**import javax.swing.JTable;**

**import javax.swing.JScrollPane;**

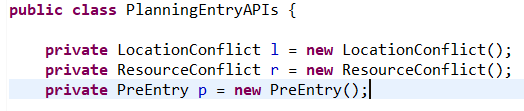
**import java.text.SimpleDateFormat;**

以CourseBoard为例：



## 可复用API设计及Façade设计模式

PlanningEntryAPIs将LocationConflict，ResourceConflict，PreEntry三个API作为自己的属性，实现Façade设计模式。





通过调用PlanningEntryAPIs的方法，实现三个API功能

使用方法：

PlanningEntryAPIs api = new PlanningEntryAPIs();

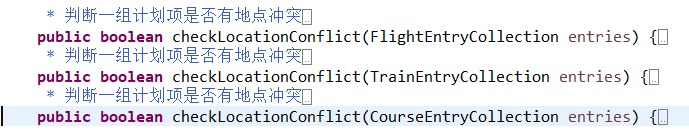
api.checkLocationConflict(p)；

api.checkResourceExclusiveConflict(p)；

api.findPreEntryPerResource(e ,fp);

### 检测一组计划项之间是否存在位置独占冲突

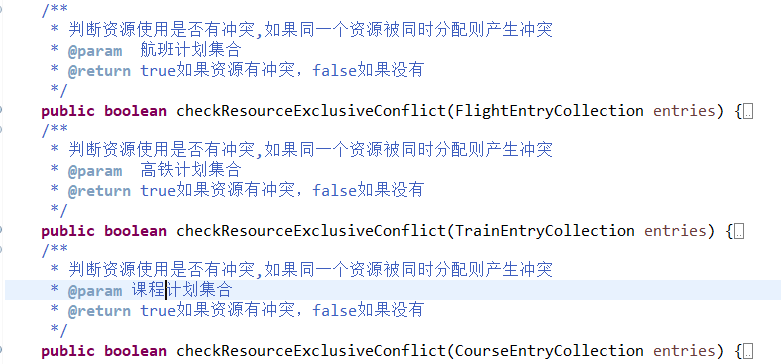
采用重载：



其中：航班计划和高铁计划直接返回true

### 检测一组计划项之间是否存在资源独占冲突

采用重载



### 提取面向特定资源的前序计划项

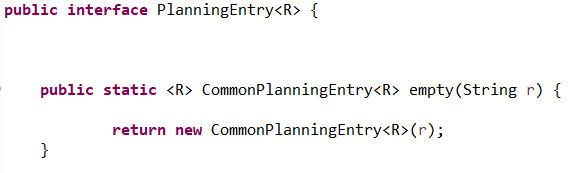
采用重载：



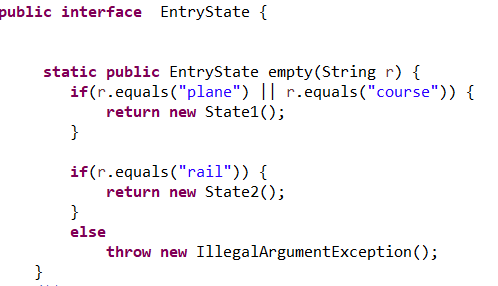
## 设计模式应用

### Factory Method

在PlanningEntry<R>中，静态工厂方法构造实例



在EntryState中，静态工厂方法构造两种状态实例



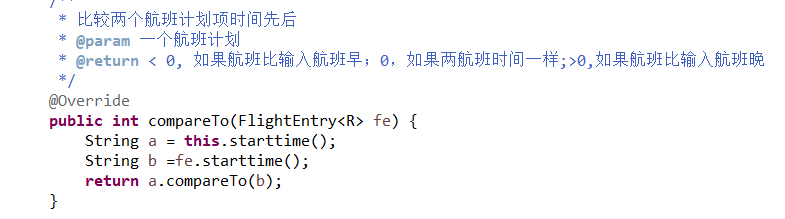
### Iterator

1、FlightBoard 中：

FlightBoard implements Iterable<FlightEntry<Plane>>

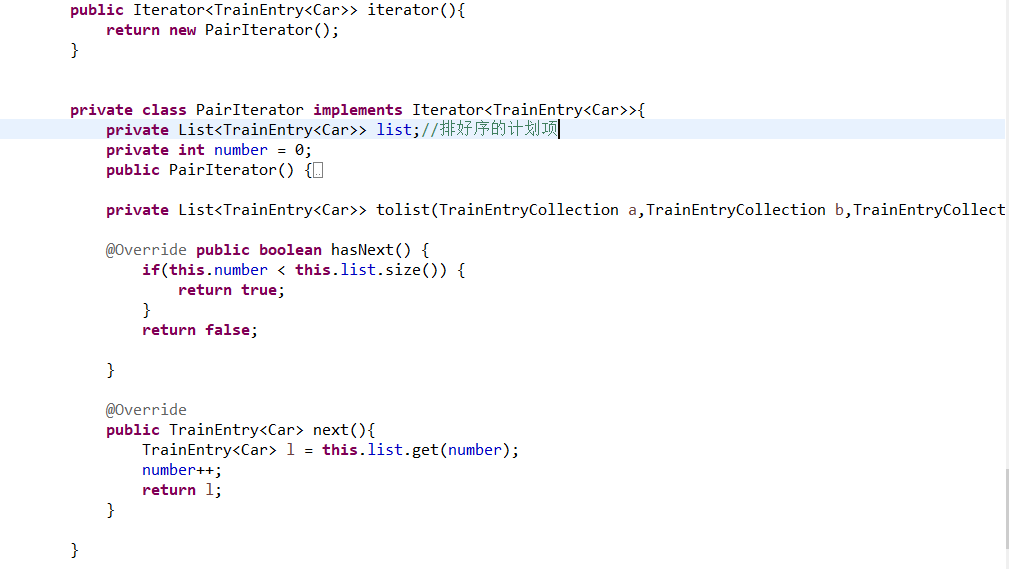


并在FlightEntry中重写compareTo（）



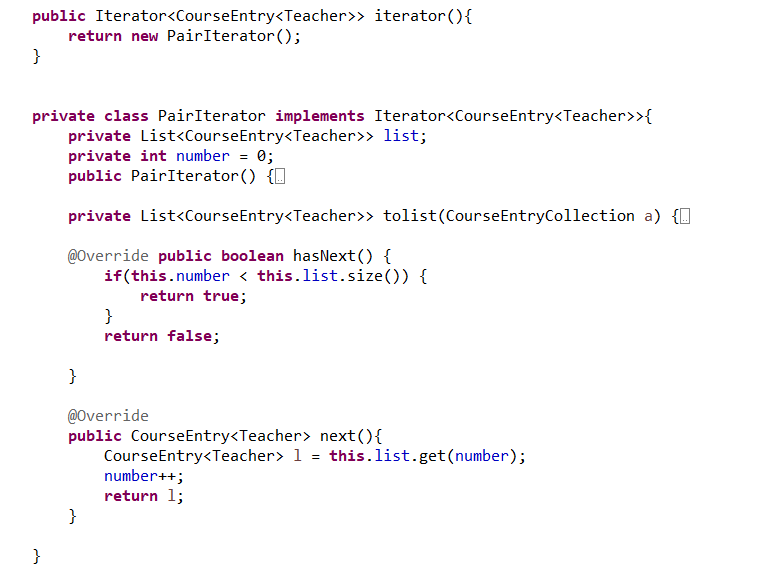
2、TrainBoard





3、CourseBoard

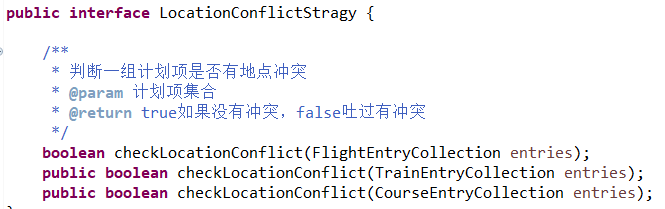




### Strategy

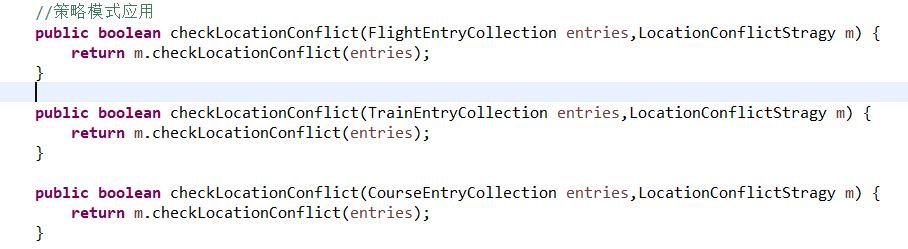
对API 判断地点冲突采用另一种算法实现为LocationConflict1

构建一个接口：



将LocationConflict， LocationConflict1作为接口的两个实现类

在PlanningEntryAPIs增加新方法：

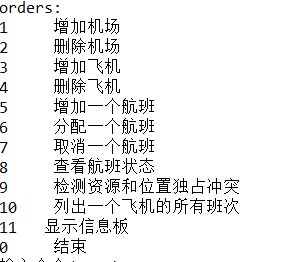


## 应用设计与开发

采用命令行输入

### 航班应用

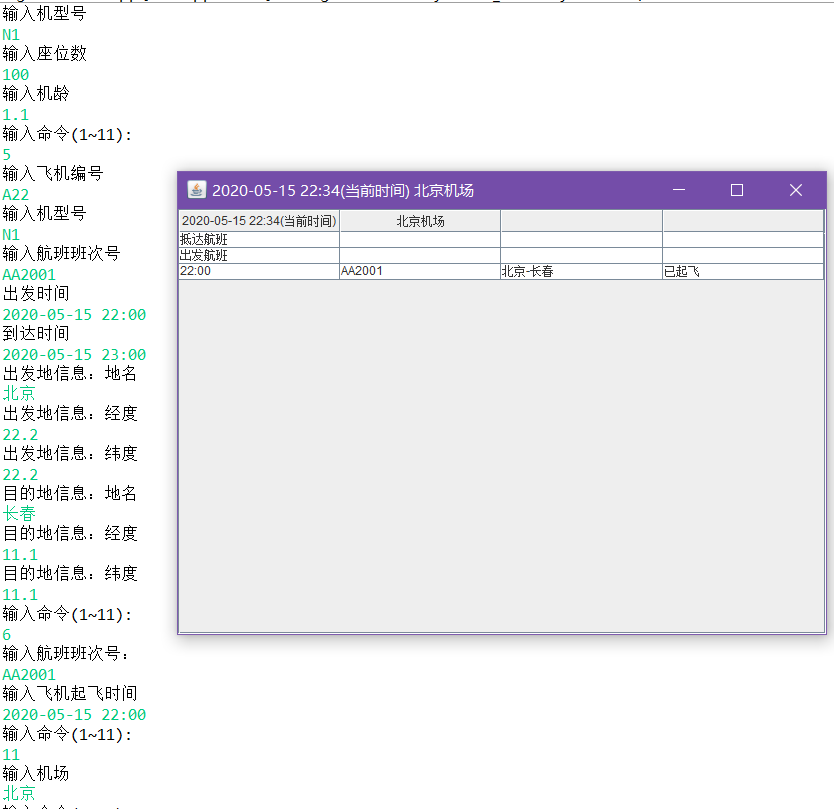
客户端采用命令行输入，且为交互应答模式



方法：

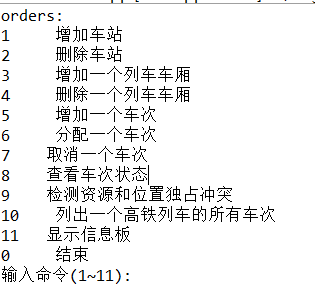


结果示例：



### 高铁应用

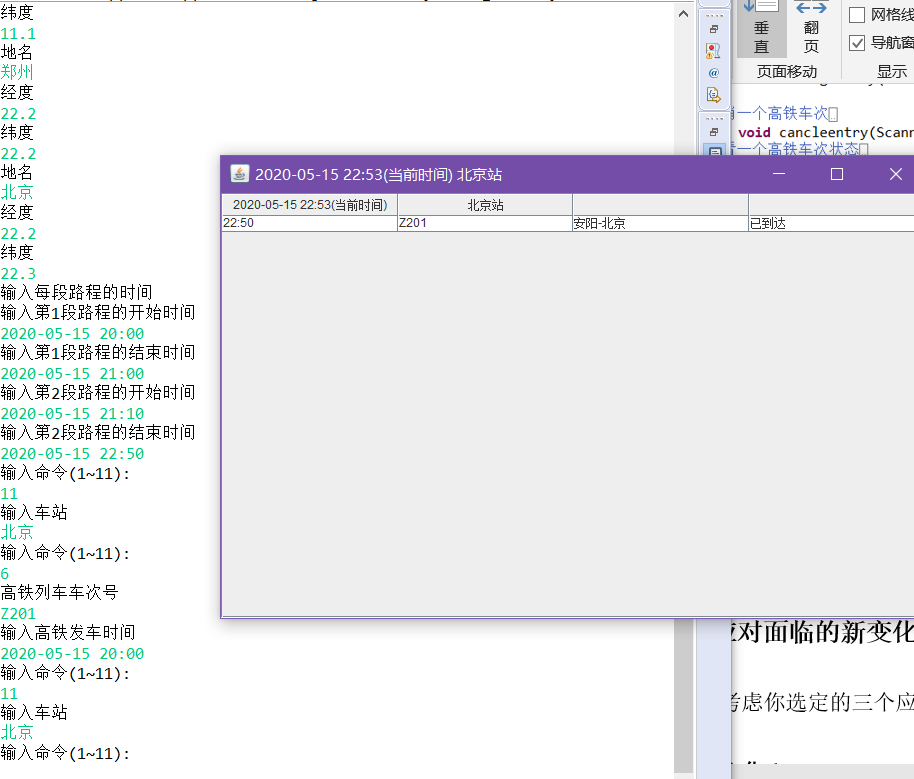
客户端采用命令行输入，且为交互应答模式



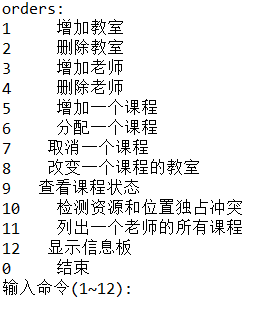
所有方法：



一个使用示例：



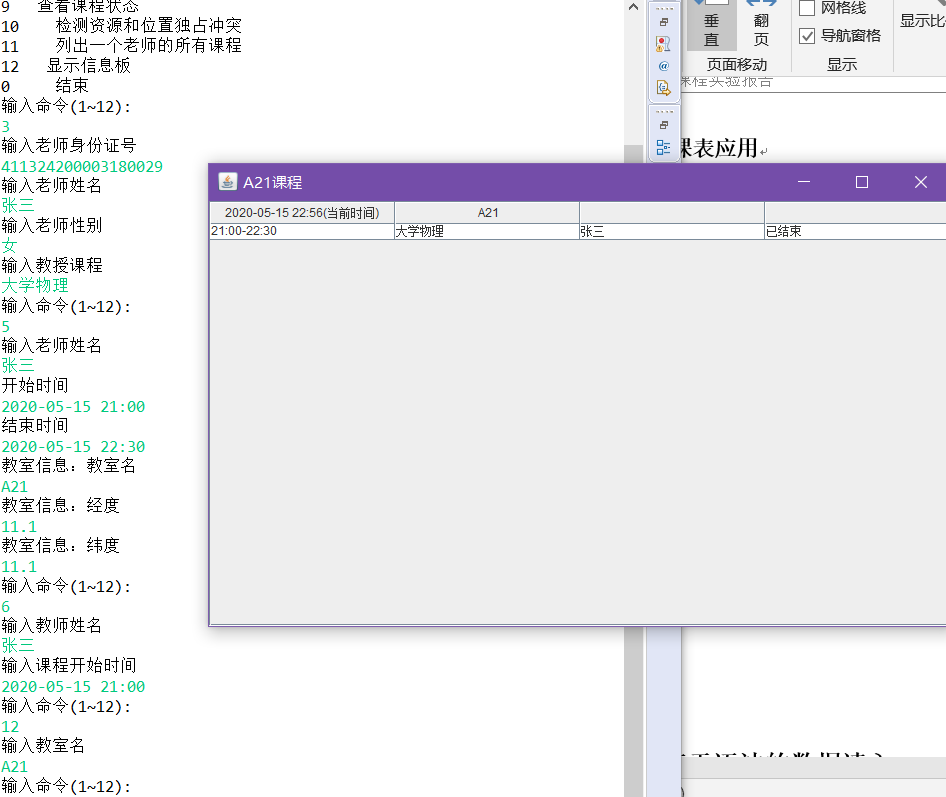
### 课表应用



方法：



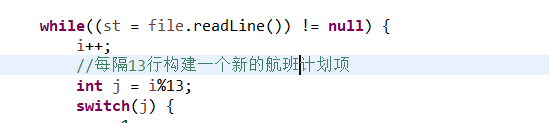
示例：



## 基于语法的数据读入

新建**public** **class** FlightScheduleInputApp **extends** FlightScheduleApp





## 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

评估之前的设计是否可应对变化、代价如何

如何修改设计以应对变化

### 变化2

评估之前的设计是否可应对变化、代价如何

如何修改设计以应对变化

### 变化3

评估之前的设计是否可应对变化、代价如何

如何修改设计以应对变化

## Git仓库结构

请在完成全部实验要求之后，利用Git log指令或Git图形化客户端或GitHub上项目仓库的Insight页面，给出你的仓库到目前为止的Object Graph，尤其是区分清楚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 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| Jswing窗口点击“x”关闭时无法回到命令行输入 | 设置窗口关闭模式  jf.setDefaultCloseOperation(JFrame.***DISPOSE\_ON\_CLOSE***); |
|  |  |
|  |  |

# 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？
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？
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？
4. 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？
5. 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？
6. 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？
7. “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？
8. 关于本实验的工作量、难度、deadline。
9. 到目前为止你对《软件构造》课程的评价。