

**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 | 张朝 |
| 学号 | 1180300639 |
| 班号 | 1803006 |
| 电子邮件 | LancelWhite@163.com |
| 手机号码 | 16645515082 |

**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

根据实验手册简要撰写。

# 实验环境配置

在这里给出你的GitHub Lab3仓库的URL地址（Lab3-学号）。

Lab3-1180300629

# 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

航班管理

高铁车次管理

学习日程管理

## 面向可复用性和可维护性的设计：PlanningEntry<R>

### PlanningEntry<R>的共性操作

### 局部共性特征的设计方案

通过CommonPlanningEntry实现PlanningEntry接口：

**public** **class** CommonPlanningEntry<R> **implements** PlanningEntry<R>

{

EntryState state;

String strPlanningEntryType;

String planningEntryNumber;

Location location;

Timeslot timeslot;

List<R> resources = **new** ArrayList<R>();

@Override

**public** Boolean start()

{

**return** **this**.state.setNewState(strPlanningEntryType, "Running");

}

@Override

**public** Boolean block()

{

**return** **this**.state.setNewState(strPlanningEntryType, "Blocked");

}

@Override

**public** Boolean cancel()

{

**return** **this**.state.setNewState(strPlanningEntryType, "Cancelled");

}

@Override

**public** Boolean end()

{

**return** **this**.state.setNewState(strPlanningEntryType, "Ended");

}

@Override

**public** Location getLocation()

{

**return** **this**.location;

}

@Override

**public** Timeslot getTimeslot()

{

**return** **this**.timeslot;

}

@Override

**public** EntryState getState()

{

**return** **this**.state;

}

@Override

**public** String getPlanningEntryType()

{

**return** **this**.strPlanningEntryType;

}

@Override

**public** List<R> getResources()

{

**return** **this**.resources;

}

@Override

**public** String getPlanningEntryNumber()

{

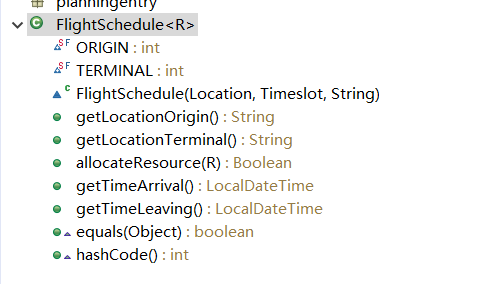
**return** **this**.planningEntryNumber;

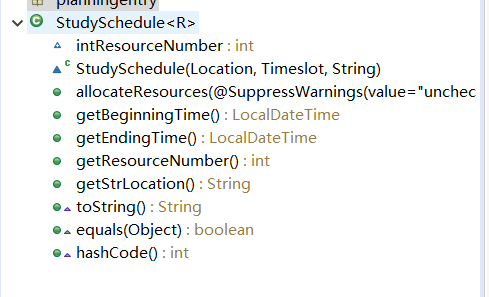
}

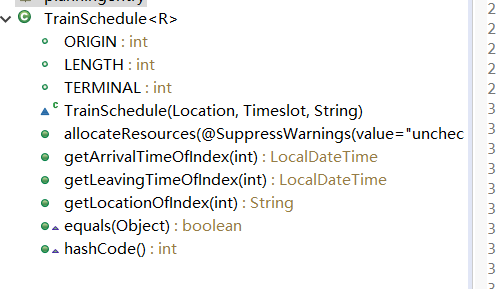
}

再用三个类分别继承CommonPlanningEntry。

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

构造三个子类的方法分别为：





## 面向复用的设计：R

将R设计为接口，并包含三种工厂方法，分别在子类中逐一实现不同功能。

## 面向复用的设计：Location

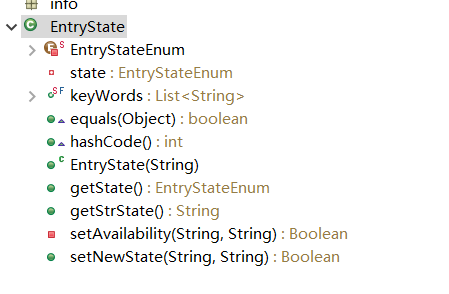
讲location设置为数组以方便存储多种信息，从而更好的复用。

## 面向复用的设计：Timeslot

Timeslot包含LocalDateTime的数组，与Location相对应。

IMG_256

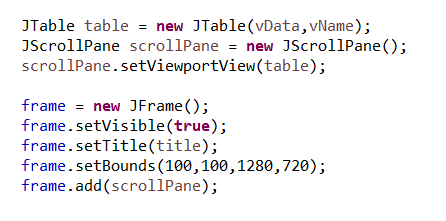
## 面向复用的设计：EntryState及State设计模式



IMG_256

## 面向应用的设计：Board

## Board的可视化：外部API的复用



## 可复用API设计及Façade设计模式

### 检测一组计划项之间是否存在位置独占冲突

通过遍历查看是否存在冲突

### 检测一组计划项之间是否存在资源独占冲突

与3.9.1相同

### 提取面向特定资源的前序计划项

先找同一资源的所有计划项，遍历更新前序值。

## 设计模式应用

### Factory Method

将工厂方法应用在PlanningEntry中使其更快捷方便。

### Iterator

### Strategy

## 应用设计与开发

### 航班应用

### 高铁应用

### 进程应用

### 课表应用

### 学习活动应用

## 基于语法的数据读入

## 应对面临的新变化

### 变化1

### 变化2

### 变化3

## 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 |
|  |  |
|  |  |
|  |  |

# 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？

复用可以提高编写效率，减少工作量

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？

增强可复用性和健壮性。

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？
2. 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？
3. 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？
4. 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？

工作量巨大。增加做实验的时间。

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
2. 关于本实验的工作量、难度、deadline。

工作量大，难度高

1. 到目前为止你对《软件构造》课程的评价。

学起来比较难，前置java没有学习的话又很大困难