**面向对象程序设计**

**学期实训报告**

**院 系： 计算机科学与技术学院**

**专 业： 物联网专业1706班**

**学 号： 2017115050614**

**姓 名：**   **杨雨婷**

**2018　年　　7月　 25日**

**学期实训**

目录

[一、实训目的 3](#_Toc520275997)

[二、实训内容 3](#_Toc520275998)

[1. 分析、阅读鼎利集团提供的“机票在线销售系统”源代码 3](#_Toc520275999)

[1.1 com.booking.constants与com.booking.util包 3](#_Toc520276000)

[1.1.1 com.booking.constants包 3](#_Toc520276001)

[1.1.2 com.booking.util包 4](#_Toc520276002)

[1.2基于面向对象现封装的对属性和对方法分别封装的类 4](#_Toc520276003)

[1.2.1 Flight类（属性） 4](#_Toc520276004)

[1.2.2 Seat类（属性） 4](#_Toc520276005)

[1.2.3 Ticket类（属性） 4](#_Toc520276006)

[1.2.4 BookingService接口以及其实现类BookingServiceImpl（方法） 5](#_Toc520276007)

[1.3基本组件与基础Frame类，封装通用界面设置及拖拽事件 5](#_Toc520276008)

[1.4在线订票系统主控制器以及其控制的各界面 5](#_Toc520276009)

[1.4.1 主控制器 5](#_Toc520276010)

[1.4.2 MainView 6](#_Toc520276011)

[1.4.3 BookingView 6](#_Toc520276012)

[1.4.4 RefundOrAlterView 6](#_Toc520276013)

[1.4.5 CheckinView 6](#_Toc520276014)

[1.4.6 类图的简单框架 7](#_Toc520276015)

[2. 修改源系统中存在的缺陷（Bug）。 8](#_Toc520276016)

[2.1 购票时机票出发日期问题 8](#_Toc520276017)

[2.1.1 源代码的修改 8](#_Toc520276018)

[2.1.2 测试结果 8](#_Toc520276019)

[2.2 选完座位机票上却没有座位信息问题 11](#_Toc520276020)

[2.2.1 源代码的修改 11](#_Toc520276021)

[2.2.2 测试结果 12](#_Toc520276022)

[3. 在原系统的基础上扩充功能 12](#_Toc520276023)

[3.1 增加购票功能 12](#_Toc520276024)

[3.1.1 源代码 12](#_Toc520276025)

[3.1.2 测试结果 25](#_Toc520276026)

[4. 选作内容 28](#_Toc520276027)

[4.1增加查看所有航班功能 28](#_Toc520276028)

[4.1.1 源代码 28](#_Toc520276029)

[4.1.2 测试结果 34](#_Toc520276030)

[4.2 添设管理员权限 34](#_Toc520276031)

[4.2.1 源代码 34](#_Toc520276032)

[4.2.2 测试结果 37](#_Toc520276033)

[4.3 选完座位机票增加座位信息 39](#_Toc520276034)

[4.3.1 源代码的修改 39](#_Toc520276035)

[4.3.1 测试结果 40](#_Toc520276036)

[三、实训总结 40](#_Toc520276037)

# 一、实训目的

1.掌握面向对象的设计、架构与实现方法。

2.掌握图形用户界面的事件驱动机制。

3.掌握Java对象流的基本操作。

4. 掌握图形用户界面、面向对象的设计以及文件的综合应用。

5. 提高分析、阅读源代码的能力。

# 二、实训内容

**必做内容：**

1. **分析、阅读鼎利集团提供的“机票在线销售系统”源代码**

提示：写出系统的分层结构、每个类中各个方法的功能（要详细，不能照抄鼎利提供的文档）。

1. **修改源系统中存在的缺陷（Bug）。**

提示：先指出在哪个类、哪个方法中存在什么样的bug，这个bug有什么危害。

然后给出修改后的代码（只需给出一个方法的代码，并注明修改的位置。

**3. 在原系统的基础上扩充功能**

增加添加航班信息的窗口，将新的航班信息追加到flight.data文件的尾部。

要求：给出源代码和测试结果

**选作内容：**

根据自己的能力做适当的扩充。

要求: 1：说明扩充的功能

2：给出源代码和测试结果

# 分析、阅读鼎利集团提供的“机票在线销售系统”源代码

## com.booking.constants与com.booking.util包

### com.booking.constants包

用于存放各种标实常量便于查看和更改常量。

### com.booking.util包

存放程序要使用的各种外部工具，可借助外部工具使程序更快捷的编写出来。

其中包括日历类：CalendarPanel、购票信息数据操作工具类：DataOperateUtil、时间格式化工具类：DateFormatUtil、订单编码工具类：OrderUtil、 数据验证工具类：ValidateDataUtil、设置界面背景工具类：ViewBackgroundUtil

## 1.2基于面向对象现封装的对属性和对方法分别封装的类

### 1.2.1 Flight类（属性）

本类将预定机票的属性进行了封装，里面拥有预定机票的各种属性，分别包括航班号、始发地、目的地、起飞时间时间、到达时间、航空公司、机型以及票价。如以下代码对类的属性进行了是申明。

private String flightNo;

private String startAddress;

private String endAddress;

private String startTime;

private String arrivedTime;

private String airLineName;

private String planeType;

private String price;

由于属性都是private修饰的属性，所以要想对类的属性进行更改也必然少不了各自对应的get、set方法。

### Seat类（属性）

本类将座位的属性进行了封装，里面拥有座位的各种属性，分别包括状态、座位号。如以下代码对类的属性进行了是申明。

private String status;

private String seatNo;

由于属性都是private修饰的属性，所以要想对类的属性进行更改也必然少不了各自对应的get、set方法。

### Ticket类（属性）

本类将机票的属性进行了封装，里面拥有已购机票的各种属性，分别包括始发地、目的地、出行时间、付款方式、订单编号、身份证以及姓名。如以下代码对类的属性进行了是申明。

private String startAddress;

private String endAddress;

private String travelDate;

private int payWay;

private String orderNo;

private String identityNo;

private String userName;

由于属性都是private修饰的属性，所以要想对类的属性进行更改也必然少不了各自对应的get、set方法。

### BookingService接口以及其实现类BookingServiceImpl（方法）

接口也就是为了实现类的设计，并将在线服务的方法进行了封装。其中包括的方法有，创建购票信息、获取购票信息、根据始发地和目的地获取航班信息、根据订单获取订单信息、存储座位信息和订单信息、修改出行时间以及退票删除订单编码。这时也要注意各个方法各自对应的参数。

public Map<String,Object> createBookingInfo(Ticket ticket);

public Map<String,Object> readData(String orderNoOrIdentityNo);

public Map<String,Object> readFlightData(String startAddress, String endAddress);

public Map<String,Object> readSeatData(String order);

public Map<String,Object> createSeatInfo(Seat seat,List<String> orderList);

public Map<String,Object> updateTravelDateInfo(String orderNo,String travelDate);

public Map<String,Object> deleteOrderInfo(String orderNo);

BookingServiceImpl类则专用于实现对应的每个方法，以便后期使用。他的具体实现为：

以创建购票信息方法为例，他是首先获取当前时间的毫秒值作为账单号orderNo。再分别创建一个结果和一个数据集，当验证身份证号不合法就返回错误提示，若正确则进行下一步验证姓名是否合法，当姓名不合法就返回错误提示，都合法就可将订单编号写入对象ticket.在这都完成之后再进行读取磁盘中的文件，若文件为空则将机票集写入到结果集中，并将其加入到结果集中，如若文件不为空就将机票加入到文件链中，并将其加入到结果集中。最后返回机票结果集。其他方法以此类推。

### 1.3基本组件与基础Frame类，封装通用界面设置及拖拽事件

两个自定义按钮CustomButton与CustomButton2用于对图形界面进行设计时使用。

BaseFrame用于基础Frame类，封装通用界面设置及拖拽事件，相当于对界面的初始化，初始化包括界面边框大小，位置、布局的初始化和对拖拽事件的处理。

也就是说以后的每一个界面的父类都是继承BaseFrame类的，并且很有可能会用到自定义按钮。

## 1.4在线订票系统主控制器以及其控制的各界面

### 1.4.1 主控制器

主控制器首先在构造方法中将主界面（MainView）显示出来，并且对主界面的四个按钮进行监听，在加上子界面的另外“BacktoPaView”,”toMainView”,”homepage”几个按钮进行监听，虽然另有其他有几个按钮，但实际上并没有使用，所以可将其注释掉。所以在这个地方主控制器起到了主控制的作用，控制各界面的开关。

### 1.4.2 MainView

最开始执行打开的界面，共有四个按钮添加到界面上。

### 1.4.3 BookingView

在监听下按”bokingBtn”随即打开BokingView,随后将主控制器生成对象（监听）传参给BokingView，BokingView可见。

BokingView界面首先是有始发地，目的地，姓名，身份证号等机票信息标签及文本框的初始化和按钮的初始化与监听。

若点击选择航班，便打开FlightListView界面，并将对应始发地，目的地等其他参数传入（public FlightListView(JFrame frame, boolean modal, String title, JTextField flightNo, Ticket ticket)），其中最重要的功能是运用了BookingService类中的readFlightData(ticket.getStartAddress(),ticket.getEndAddress())方法根据始发地和目的地将航班信息读取到表格中。完成信息填写后用户点击确认按钮就会打开PayView界面进行选择支付。

### 1.4.4 RefundOrAlterView

在监听下按” alterBtn”随即打开RefundOrAlterView,随后将主控制器生成对象（监听）传参给RefundOrAlterView，RefundOrAlterView可见。

RefundOrAlterView界面首先是订单号或身份证号标签及文本框的初始化和按钮的初始化与监听。

在用户输入合法信息后点击确认按钮就会打开AlterOrderView界面，并将监听对象以及账单号或者身份证号参数传入（public AlterOrderView(ActionListener actionListener,String orderNoOrIdentityNo)）其中最重要的功能是运用BookingService类中的Map<String,Object>resultMap = bookingService.deleteOrderInfo(AlterOrderView.this.orderNo);方法根据始订单号或身份证号将机票退掉或者BookingService类中的Map<String,Object> resultMap = bookingService.updateTravelDateInfo(AlterOrderView.this.orderNo, inputContent);方法根据始订单号和输入日期将机票改签。

### 1.4.5 CheckinView

在监听下按” checkinBtn”随即打开CheckinView,随后将主控制器生成对象（监听）传参给CheckinView，CheckinView可见。

CheckinView界面首先是订单号或身份证号标签及文本框的初始化和按钮的初始化与监听。

在用户输入合法信息后点击确认按钮就会打开OrderView界面，并将监听对象以及账单号或者身份证号参数传入（public OrderView(ActionListener actionListener,String orderNoOrIdentityNo)）其中最重要的功能是运用BookingService类中的方法Map<String, Object> resultMap = bookingService.readData(orderNoOrIdentityNo); 将符合信息的航班读取到链表中。用户选取所想要值机的航班点击确认按钮后就会打开SeatView界面进行选座。

### 1.4.6 类图的简单框架

MainView (“bokingBtn”,” alterBtn”,” checkinBtn”)

PayView

(readFlightData(ticket.getStartAddress(),ticket.getEndAddres))

BookingView BookingService

Flight

FlightListView

bookingService

(readFlightData(ticket.getStartAddress(),ticket.getEndAddress())

BookingController

bookingService

(deleteOrderInfo(AlterOrderView.this.orderNo))

CheckinView OrderView

SeatView Seat bookingService

(readSeatData(orderList.get(0)))

RefundOrAlterView AlterOrderView bookingService

(updateTravelDateInfo(AlterOrderView.this.orderNo, inputContent) )

源代码中 主控制器控制控制四个界面MainView，BookingView，CheckinView和RefundOrAlterView。将主界面切换到BookingView，CheckinView和RefundOrAlterView界面。所有的View类都继承于BaseFrame类。

# 2. 修改源系统中存在的缺陷（Bug）。

## 2.1 购票时机票出发日期问题

### 2.1.1 源代码的修改

当选择出发日期的时候机票生成的是填入的出发日期，而不是航班原本的出发时相当于为顾客自动生成了一个发车时间，这并不合理。

增加代码

1. **定义时FlightListView类中**

private Ticket ticket;

private JButton choose;

private String startTime; //此处增加起飞时间字符串

1. **FlightListView类鼠标监听中**

startTime=orderList.get(4);

//在所选行里面筛选出出行时间

1. **FlightListView类中增加方法**

public String readStartTime() { //返回起飞时间方法

return startTime;

}

4.**BokingView类中设置机票时**

ticket.setTravelDate(dialog.readStartTime());//此处将所选航班的出发时间返回

5. **BokingView类中判断航班起飞时间与用户出发时间**

if(startdate.compareToIgnoreCase(dialog.readStartTime()) < 0){

JOptionPane.showMessageDialog(BookingView.this, "出行时间不能早于起飞时间！","提示信息", JOptionPane.ERROR\_MESSAGE);

return;

### 2.1.2 测试结果

选择出行日期不能早于航班起飞时间，也就是在选一个早于出行日期的航班。以免耽误行程。

}



图2.1键入时间



图2.1选择出行日期是7月31，20点

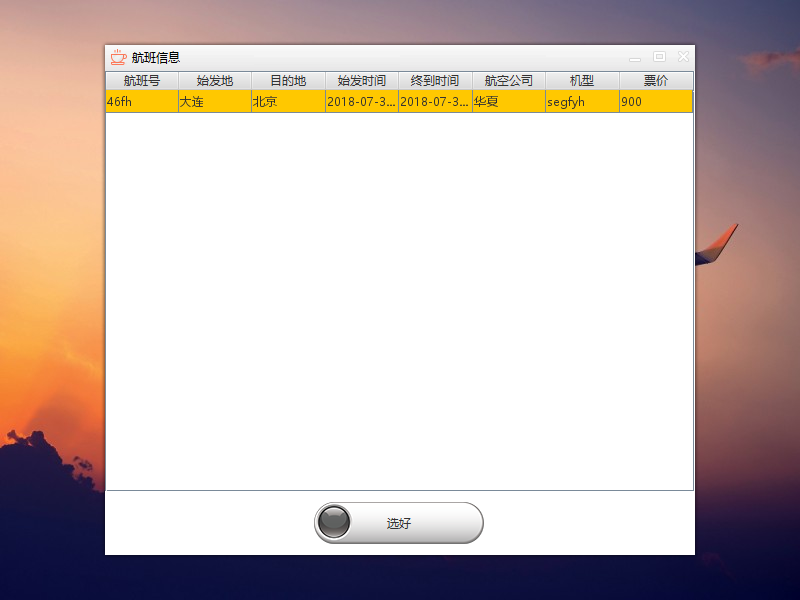


图2.2航班出发时间为另一时期



图2.3生成机票出发时间为航班出发时间

## 2.2 选完座位机票上却没有座位信息问题

### 2.2.1 源代码的修改

**1 Ticket类中增加属性**

public String getSeat() {

return seat;

}

public void setSeat(String seat) {

this.seat = seat;

}

/\*\*

\*座位

\*/

private String seat;

package com.booking.service.impl;

**2 BookingServiceImpl类中的createSeatInfo方法增加**

if (!"".equals(order)) {

// 读取本地磁盘数据

List<Ticket> ticketList = dataOperateUtil.readTicketData(Constants.FILE\_NAME);

// 判断数据是否为空

if (ticketList != null) {

// 循环获取购票信息

for (Ticket ticket : ticketList) {

// 获取订单编号

String order1 = ticket.getOrderNo();

// 判断获取的订单编号与参数是否相等，相等则修改该订单出行时间

if (order1.equals(order)) {

ticket.setSeat(seat.getSeatNo());

}

}

}

operateResult = dataOperateUtil.writeTicketData(ticketList);

}

//Map<String,Object> resultMap = new HashMap<>();

// 返回消息

resultMap.put(Constants.RESULT\_KEY\_CODE, operateResult);

// 操作成功返回卡对象

if (operateResult.equals(Constants.MSG\_SUCCESS)) {

resultMap.put(Constants.RESULT\_KEY\_DATA, Constants.MSG\_SUCCESS);

}

**3 BookingView中ticket写入时增加**

ticket.setSeat("未选");

**4 AlterOrderView类中**

columnNames.add("座位"); //增加代码

和

vector.add(t.getSeat());

### 2.2.2 测试结果



# 3. 在原系统的基础上扩充功能

## 3.1 增加购票功能

### 3.1.1 源代码

**package com.booking.controller;**

public class BookingController implements ActionListener {

/\*\*

\* @Fields mainView : 主界面

\*/

private MainView mainView;

/\*\*

\* @Fields bookingView : 购票界面

\*/

private BookingView bookingView;

/\*\*

\* @Fields refundOrAlterView : 退票/改签界面

\*/

private RefundOrAlterView refundOrAlterView;

/\*\*

\* @Fields checkinView : 值机界面

\*/

private CheckinView checkinView;

/\*\*

\* @Fields PayView : 支付界面

\*/

private PayView payView;

/\*\*

\* @Fields OrderView : 订单界面

\*/

private OrderView orderView;

private CheckAllFlightView dialog;

private AddFlightLsitView addFlightLsitView;

public BookingController() {

showMainView();

}

@Override

public void actionPerformed(ActionEvent e) {

// 获取点击事件发生的按钮上的ActionCommand字符串

String actionCommand = e.getActionCommand();

// 根据不同按钮点击，显示不同窗口

switch (actionCommand) {

case "addFlight":

// 显示添加机票票窗口

addFlightLsitView=new AddFlightLsitView();

addFlightLsitView.setVisible(true);

break;

case "findFlight":

// 显示查看所有机票窗口

dialog = new CheckAllFlightView("航班信息");

dialog.setSize(600,520);

dialog.setVisible(true);

dialog.setLocationRelativeTo(null);

break;

// 点击“购票”按钮

case "bookingBtn":

// 显示购票窗口

bookingView = new BookingView(BookingController.this);

bookingView.setVisible(true);

break;

// 点击“退票/改签”按钮

case "alterBtn":

// 显示值机窗口

refundOrAlterView = new RefundOrAlterView(this);

refundOrAlterView.setVisible(true);

break;

// 点击“值机”按钮

case "checkinBtn":

// 显示值机窗口

checkinView = new CheckinView(this);

checkinView.setVisible(true);

break;

// 购票窗口点击“返回”按钮

case "backToMainView":

JButton btn = (JButton) e.getSource();

JFrame f = (JFrame) btn.getParent().getParent().getParent().getParent();

f.setVisible(false);

f.dispose();

mainView.setVisible(true);

break;

// 支付窗口点击“返回”按钮

case "backToPayView":

JButton backPay = (JButton) e.getSource();

JFrame pay = (JFrame) backPay.getParent().getParent().getParent().getParent();

pay.setVisible(false);

pay.dispose();

bookingView = new BookingView(this);

bookingView.setVisible(true);

break;

// 值机窗口点击“返回”按钮

case "toMainView":

JButton backToMainbtn = (JButton) e.getSource();

JFrame main = (JFrame) backToMainbtn.getParent().getParent().getParent().getParent();

main.setVisible(false);

main.dispose();

mainView.setVisible(true);

break;

// 订单窗口点击“返回”按钮

case "homepage":

JButton ToMainbtn = (JButton) e.getSource();

JFrame toMain = (JFrame) ToMainbtn.getParent().getParent().getParent().getParent();

toMain.setVisible(false);//设置为主窗口不可见

toMain.dispose();

mainView.setVisible(true);

break;

}

}

/\*\*

\* @Title: showMainView

\* @Description: 显示主界面

\*/

private void showMainView() {

mainView = new MainView(this);

mainView.setVisible(true);

}

}

**package com.booking.service.impl;**

public class AddFlightServiceImpl {

public void createBookingInfo(Flight flight) {

DataOperateUtil dataOperateUtil = DataOperateUtil.getDataOperateUtil();

// 数据结果集

List<Flight> tic = new ArrayList<>();

// 调用写入数据方法

// 读取本地磁盘数据

List<Flight> ticketList = dataOperateUtil.readFlightData(Constants.FILE\_FLIGHTS\_NAME);

if (ticketList == null){

tic.add(flight); //

dataOperateUtil.writeFlightData(tic);

}else {

ticketList.add(flight); //

dataOperateUtil.writeFlightData(ticketList);

}

}

}

**package com.booking.util;**

增加方法：

public class CalendarPanel extends JPanel {

public void writeFlightData(List<Flight> flight) {

FileOutputStream fos = null;

ObjectOutputStream oos = null;

try {

fos = new FileOutputStream(Constants.FILE\_FLIGHTS\_NAME+".data");

oos = new ObjectOutputStream(fos);

oos.writeObject(flight);

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}finally {

if(oos !=null) {

try {

oos.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

}

**package com.booking.view;**

public class AddFlightLsitView extends BaseFrame { //班号、始发地、目的地、起飞时间时间、到达时间、航空公司、机型以及票价

/\*\*

\* @Fields serialVersionUID

\*/

private static final long serialVersionUID = 2870945371327830406L;

/\*\*

\* @Fields flightNo : 航班号

\*/

private JTextField flightNo;

/\*\*

\* @Fields startPlace : 始发地

\*/

private JComboBox<String> startComboBox;

/\*\*

\* @Fields endPlace : 目的地

\*/

private JComboBox<String> endComboBox;

/\*\*

\* @Fields date : 出行日期

\*/

private JTextField date;

/\*\*

\* @Fields idNumber : 到达时间

\*/

private JTextField enddate;

/\*\*

\* @Fields name :航空公司

\*/

private JTextField name;

/\*\*

\* @Fields name :机型

\*/

private JTextField Type;

/\*\*

\* @Fields name :票价

\*/

private JTextField Money;

/\*\*

\* @Fields confirmBtn : 确认按钮

\*/

private JButton confirmBtn;

/\*\*

\* @Fields backBtn : 返回按钮

\*/

private JButton backBtn;

/\*\*

\* @Fields dialog : 航班选择对话窗口

\*/

public JTextField getFlightNo() {

return flightNo;

}

public AddFlightLsitView() {

init();

}

private void init() {

// 设置背景图片

ViewBackgroundUtil.setBG(this, "img/bg2.jpg");

JLabel flightLabel = new JLabel("添加航班号：");

flightLabel.setBounds(180, 120, 90, 33);

// 文字右对齐

flightLabel.setHorizontalAlignment(SwingConstants.RIGHT);

// 航班号码文本框

flightNo = new JTextField();

flightNo.setBounds(380, 120, 200, 33);

flightNo.setToolTipText("填入大写字母");

// 定义下拉列表的条目

String[] startComboData = new String[]{"大连","北京","上海","广州","深圳","成都","昆明","哈尔滨","长春","沈阳","天津","西安","乌鲁木齐","郑州","武汉","无锡","南京","温州","重庆","三亚","厦门","长沙","青岛","杭州"};

String[] endComboData = new String[]{"大连","北京","上海","广州","深圳","乌鲁木齐","郑州","武汉","成都","三亚","厦门","长沙","昆明","哈尔滨","长春","沈阳","天津","西安","无锡","南京","温州","重庆","青岛","杭州"};

// 始发地标签

JLabel startLabel = new JLabel("始发地:");

startLabel.setBounds(180, 160, 90, 33);

// 文字右对齐

startLabel.setHorizontalAlignment(SwingConstants.RIGHT);

// 始发地下拉列表

startComboBox = new JComboBox<>(startComboData);

startComboBox.setBounds(380, 160, 200, 33);

// 设置默认选中的条目

startComboBox.setSelectedIndex(0);

// 目的地标签

JLabel endLabel = new JLabel("目的地:");

endLabel.setBounds(180, 200, 90, 33);

// 文字右对齐

endLabel.setHorizontalAlignment(SwingConstants.RIGHT);

// 目的地下拉列表

endComboBox = new JComboBox<>(endComboData);

endComboBox.setBounds(380, 200, 200, 33);

endComboBox.setToolTipText("请输入目的地");

JLabel dateLabel = new JLabel("出行日期:");

dateLabel.setBounds(180, 240, 90, 33);

dateLabel.setForeground(Color.ORANGE);

// 文字右对齐

dateLabel.setHorizontalAlignment(SwingConstants.RIGHT);

// 出行日期文本框

date = new JTextField();

date.setBounds(380, 240, 200, 33);

date.setToolTipText("请输入出行日期");

// 定义日历控件面板类

CalendarPanel acalendar = new CalendarPanel(date, "yyyy-MM-dd HH:mm:ss");

acalendar.initCalendarPanel();

acalendar.add(date);

JLabel enddateLabel = new JLabel("到达时间");

enddateLabel.setBounds(180, 280, 90, 33);

// 文字右对齐

enddateLabel.setHorizontalAlignment(SwingConstants.RIGHT);

// 姓名文本框

enddate = new JTextField();

enddate.setBounds(380, 280, 200, 33);

enddate.setToolTipText("请输入到达时间");

CalendarPanel abalendar = new CalendarPanel(enddate, "yyyy-MM-dd HH:mm:ss");

abalendar.initCalendarPanel();

abalendar.add(enddate);

JLabel nameLabel = new JLabel("航空公司：");

nameLabel.setBounds(180, 320, 90, 33);

// 文字右对齐

nameLabel.setHorizontalAlignment(SwingConstants.RIGHT);

// 航班号码文本框

name = new JTextField();

name.setBounds(380, 320, 200, 33);

name.setToolTipText("请输入航空公司");

//jixing

JLabel TypeLabel = new JLabel("机型：");

TypeLabel.setBounds(180, 360, 90, 33);

// 文字右对齐

TypeLabel.setHorizontalAlignment(SwingConstants.RIGHT);

// 航班号码文本框

Type = new JTextField();

Type.setBounds(380, 360, 200, 33);

Type.setToolTipText("请输入机型");

//票价

JLabel MoneyLabel = new JLabel("票价：");

MoneyLabel.setBounds(180, 400, 90, 33);

// 文字右对齐

MoneyLabel.setHorizontalAlignment(SwingConstants.RIGHT);

// 航班号码文本框

Money = new JTextField();

Money.setBounds(380, 400, 200, 33);

Money.setToolTipText("请输入票价");

// 实例化确认按钮

confirmBtn = new CustomButton(460, 480, CustomButton.RIGHT);

confirmBtn.setText("确认");

confirmBtn.addActionListener(new ActionListener() {

@Override//表示下面为方法重写

public void actionPerformed(ActionEvent e) {

String startAddress = startComboData[startComboBox.getSelectedIndex()];

String endAddress = endComboData[endComboBox.getSelectedIndex()];

String planeType = Type.getText();

String airLineName = name.getText();

String startTime = date.getText();

String arrivedTime = enddate.getText();

String price = Money.getText();

String aflightNo = flightNo.getText();

if (startAddress.equals(endAddress)) {

JOptionPane.showMessageDialog(AddFlightLsitView.this, "始发地和目的地不能相同！",

"提示信息", JOptionPane.ERROR\_MESSAGE);

return;

}

if ("".equals(planeType)) {

JOptionPane.showMessageDialog(AddFlightLsitView.this, "机型不能为空，请填写机型！",

"提示信息", JOptionPane.ERROR\_MESSAGE);

return;

}

if ("".equals(airLineName)) {

JOptionPane.showMessageDialog(AddFlightLsitView.this, "航空公司不能为空，请填写航空公司！",

"提示信息", JOptionPane.ERROR\_MESSAGE);

return;

}

if ("".equals(startTime)) {

JOptionPane.showMessageDialog(AddFlightLsitView.this, "出行时间不能为空，请填写出行时间！",

"提示信息", JOptionPane.ERROR\_MESSAGE);

return;

}

if ("".equals(arrivedTime)) {

JOptionPane.showMessageDialog(AddFlightLsitView.this, "到达时间不能为空，请填写到达时间！",

"提示信息", JOptionPane.ERROR\_MESSAGE);

return;

}

if ("".equals(price)) {

JOptionPane.showMessageDialog(AddFlightLsitView.this, "票价不能为空，请填写票价！",

"提示信息", JOptionPane.ERROR\_MESSAGE);

return;

}

if ("".equals(aflightNo)) {

JOptionPane.showMessageDialog(AddFlightLsitView.this, "航班号不能为空，请填写航班号！",

"提示信息", JOptionPane.ERROR\_MESSAGE);

return;

}

else {

// 获取当前时间

String nowDate = DateFormatUtil.getDateFormatUtil().formatStrByDate(new Date());

// 判断出行时间不能早于当前时间

if(startTime.compareToIgnoreCase(nowDate) < 0){

JOptionPane.showMessageDialog(AddFlightLsitView.this, "出行时间不能早于当前时间！",

"提示信息", JOptionPane.ERROR\_MESSAGE);

return;

}

if(arrivedTime.compareToIgnoreCase(startTime) < 0){

JOptionPane.showMessageDialog(AddFlightLsitView.this, "到达时间不能早于出行时间！",

"提示信息", JOptionPane.ERROR\_MESSAGE);

return;

}

}

Flight flight = new Flight();

flight.setStartAddress(startAddress);

flight.setEndAddress(endAddress);

flight.setPlaneType(planeType);

flight.setAirLineName(airLineName);

flight.setArrivedTime(arrivedTime);

flight.setStartTime(startTime);

flight.setPrice(price);

flight.setFlightNo(aflightNo);

if (JOptionPane.showConfirmDialog(null, "确定创建新的机票？", "提示", JOptionPane.YES\_NO\_OPTION) == JOptionPane.YES\_OPTION) {

AddFlightServiceImpl addFlightServiceImpl = new AddFlightServiceImpl();

addFlightServiceImpl.createBookingInfo(flight);

JOptionPane.showMessageDialog(null, "创建机票成功！",

"银联卡支付成功！", JOptionPane.ERROR\_MESSAGE);

AddFlightLsitView.this.setVisible(false);

}

}

});

// 实例化返回按钮

backBtn = new CustomButton(160, 480, CustomButton.LEFT);

backBtn.setText("退出");

backBtn.addActionListener(new ActionListener() {

@Override//表示下面为方法重写

public void actionPerformed(ActionEvent e) {

if (JOptionPane.showConfirmDialog(null, "确定退出系统？", "提示", JOptionPane.YES\_NO\_OPTION) == JOptionPane.YES\_OPTION) {

AddFlightLsitView.this.setVisible(false);

}

}

});

this.getContentPane().add(startLabel);

this.getContentPane().add(startComboBox);//1

this.getContentPane().add(endLabel);

this.getContentPane().add(endComboBox);//2

this.getContentPane().add(dateLabel);

this.getContentPane().add(date);

this.getContentPane().add(enddateLabel);//3

this.getContentPane().add(enddate);

this.getContentPane().add(startLabel);

this.getContentPane().add(startComboBox);//4

this.getContentPane().add(acalendar);

this.getContentPane().add(abalendar);

this.getContentPane().add(flightLabel);

this.getContentPane().add(flightNo);//5

this.getContentPane().add(nameLabel);

this.getContentPane().add(name);//6

this.getContentPane().add(TypeLabel);

this.getContentPane().add(Type);//7

this.getContentPane().add(MoneyLabel);

this.getContentPane().add(Money);//6

this.getContentPane().add(backBtn);

this.getContentPane().add(confirmBtn);

// 设置窗口无标题栏

this.setUndecorated(true);

this.getRootPane().setWindowDecorationStyle(JRootPane.NONE);

}

}

public class MainView extends BaseFrame {

/\*\*

\* @Fields serialVersionUID

\*/

private static final long serialVersionUID = 8142213655847297255L;

private JButton AddFlight;

/\*\*

\* @Fields bookingBtn : 增加机票

\*/

private JButton findFlight;

/\*\*

\* @Fields bookingBtn : 查看机票

\*/

private ActionListener actionListener;

/\*\*

\* @Fields bookingBtn : 购票按钮

\*/

private JButton bookingBtn;

/\*\*

\* @Fields checkinBtn : 退票/改签按钮

\*/

private JButton alterBtn;

/\*\*

\* @Fields checkinBtn : 值机按钮

\*/

private JButton checkinBtn;

/\*\*

\* @Fields exitBtn : 退出按钮

\*/

private JButton exitBtn;

public MainView(ActionListener actionListener) {

this.actionListener = actionListener;

init();

}

private void init() {

// 设置背景图片

ViewBackgroundUtil.setBG(this, "img/bg1.jpg");

// 实例化购票按钮

AddFlight = new CustomButton(120, 400, CustomButton.LEFT);

AddFlight.setText("添加机票");

AddFlight.addActionListener(actionListener);

AddFlight.setActionCommand("addFlight");

// 实例化购票按钮

findFlight = new CustomButton(520, 400, CustomButton.RIGHT);

findFlight.setText("查看机票");

findFlight.addActionListener(actionListener);

findFlight.setActionCommand("findFlight");

// 实例化购票按钮

bookingBtn = new CustomButton(120, 460, CustomButton.LEFT);

bookingBtn.setText("购票");

bookingBtn.addActionListener(actionListener);

bookingBtn.setActionCommand("bookingBtn");

// 实例退改签按钮

alterBtn = new CustomButton(120, 520, CustomButton.LEFT);

alterBtn.setText("退票/改签");

alterBtn.addActionListener(actionListener);

alterBtn.setActionCommand("alterBtn");

// 实例化值机按钮

checkinBtn = new CustomButton(520, 460, CustomButton.RIGHT);

checkinBtn.setText("值机");

checkinBtn.addActionListener(actionListener);

checkinBtn.setActionCommand("checkinBtn");

// 实例退出按钮

exitBtn = new CustomButton(520, 520, CustomButton.RIGHT);

exitBtn.setText("退出系统");

exitBtn.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

System.exit(0);

}

});

JLabel welcomelabel = new JLabel("欢迎使用在线购票系统!");

welcomelabel.setBounds(30, 560, 500, 30);

// 转换日期显示格式

SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

JLabel time = new JLabel(df.format(new Date(System.currentTimeMillis())));

time.setBounds(660, 10, 200, 30);

Timer timeAction = new Timer(1000, new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

long timemillis = System.currentTimeMillis();

time.setText(df.format(new Date(timemillis)));

}

});

timeAction.start();

this.getContentPane().add(AddFlight);

this.getContentPane().add(findFlight);

this.getContentPane().add(bookingBtn);

this.getContentPane().add(alterBtn);

this.getContentPane().add(checkinBtn);

this.getContentPane().add(exitBtn);

this.getContentPane().add(welcomelabel);

this.getContentPane().add(time);

// 设置窗口无标题栏

this.setUndecorated(true);

this.getRootPane().setWindowDecorationStyle(JRootPane.NONE);

this.setTitle("在线购票系统");

}

}

### 3.1.2 测试结果



3.1主界面



3.2添加机票界面



3.3添加机票界面



3.4询问



3.5退出询问

# 4. 选作内容

## 4.1增加查看所有航班功能

### 4.1.1 源代码

public class CheckAllFlightView extends JFrame {

private static final long serialVersionUID = 1L;

private JFrame frame = null;

private boolean modal = false;

private JTextField flightNo;

private JButton choose;

public CheckAllFlightView(String title) {

super(title);

this.init();

}

@SuppressWarnings({ "unchecked", "rawtypes" })

private void init() {

if(modal) {

frame.setEnabled(false);//仅把窗口设置为不可用

}

// 航班列表表格标题

Vector columnNames = new Vector();

columnNames.add("航班号");

columnNames.add("始发地");

columnNames.add("目的地");

columnNames.add("始发时间");

columnNames.add("终到时间");

columnNames.add("航空公司");

columnNames.add("机型");

columnNames.add("票价");

DefaultTableModel tableModel = new DefaultTableModel(getTableData(), columnNames);

// 创建指定列名和数据的表格

//单击JTable某行背景改变颜色

@SuppressWarnings("serial")

JTable table = new JTable(tableModel){

public Component prepareRenderer(TableCellRenderer renderer, int row, int column){

Component comp = super.prepareRenderer(renderer,row,column);

Point p = getMousePosition();

if(p!=null){

int rowUnderMouse = rowAtPoint(p);//

if(rowUnderMouse == row){//选中行，背景设置为橙色

comp.setBackground(Color.ORANGE);

}else{//没有选中的行，颜色还原 comp.setBackground(DefaultLookup.getColor(this,ui,"Table.alternateRowColor"));

}

}

return comp;

}

};

table.addMouseMotionListener(new MyTableMouseMotionListener());

// 表格排序

RowSorter sorter = new TableRowSorter(tableModel);

table.setRowSorter(sorter);

table.setRowHeight(23);

table.setAutoscrolls(true);

table.setFillsViewportHeight(true);

//设置填充高度

table.setEnabled(false);

// 设置表格只能单选行

table.getSelectionModel().setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

table.setColumnSelectionAllowed(false);

table.setRowSelectionAllowed(true);

List<String> orderList = new ArrayList<>();

table.addMouseListener(new MouseAdapter() {

private int rowUnderMouse = -1;

public void mouseClicked(MouseEvent e) {

// 清空选中行数据列表

orderList.clear();

// 获取所选中行号

int row =((JTable)e.getSource()).rowAtPoint(e.getPoint());

if(row == -1) {

return;

}

int numCols = table.getColumnCount();

// 判断获取选中行中所有数据

TableModel model = table.getModel();

// 循环获取每列数据

for (int i = 0; i < numCols; i++) {

orderList.add(model.getValueAt(row, i).toString());

CheckAllFlightView.this.flightNo.setText(model.getValueAt(row, 0).toString());

}

JTable table = (JTable)e.getSource();

Point p = table.getMousePosition();

if(p != null){

rowUnderMouse = table.rowAtPoint(p);

if(rowUnderMouse >= 0){

for(int i=0;i<table.getColumnCount();i++){

table.prepareRenderer(table.getCellRenderer(rowUnderMouse,i),rowUnderMouse,i);

if(rowUnderMouse != 0){

table.prepareRenderer(table.getCellRenderer(rowUnderMouse-1,i),rowUnderMouse-1,i);

}

if(rowUnderMouse != table.getRowCount()-1){

table.prepareRenderer(table.getCellRenderer(rowUnderMouse+1,i),rowUnderMouse+1,i);

}

}

table.repaint(table.getVisibleRect());

}

}

}

});

// 创建显示表格的滚动面板

JScrollPane scrollPane = new JScrollPane(table);

scrollPane.setBounds(0, 0, 600, 300);

// 实例化选好按钮

choose = new CustomButton(490, 380, CustomButton.LEFT);

choose.setText("退出");

// 将滚动面板添加到边界布局的中间

this.getContentPane().add(scrollPane, BorderLayout.NORTH);

this.getContentPane().add(choose);

choose.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

/\*// 判断是否选中行，没有选中则给予提示，否则打开选座页面

if (orderList.size() == 0) {

JOptionPane.showMessageDialog(CheckAllFlightView.this, "您未选择航班！","提示信息", JOptionPane.ERROR\_MESSAGE);

}else {

CheckAllFlightView.this.setVisible(false);

CheckAllFlightView.this.dispose();

frame.setEnabled(true);

frame.requestFocus();

// 设置父窗口航班号文本框value为选中航班号

CheckAllFlightView.this.flightNo.setText(orderList.get(0));

}\*/

CheckAllFlightView.this.setVisible(false);

CheckAllFlightView.this.dispose();

}

});

}

public void windowOpened(WindowEvent windowEvent) {

}

public void windowClosing(WindowEvent windowEvent) {

// 关闭对话框，激活父窗口

if(modal) {

frame.setEnabled(true);

this.flightNo.setText(CheckAllFlightView.this.flightNo.getText());

}

}

/\*\*

\* @Title: getTableData

\* @Description: 获取航班信息table数据

\* @return

\*/

@SuppressWarnings({ "rawtypes", "unchecked" })

private Vector getTableData() {

Vector data = new Vector();

// 根据始发地和目的地读取相应航班信息

DataOperateUtil dataOperateUtil = DataOperateUtil.getDataOperateUtil();

// 数据结果集

List<Flight>flights=dataOperateUtil.readFlightData(Constants.FILE\_FLIGHTS\_NAME);

// 如果航班信息为空，直接返回空Vector

if (flights == null || flights.isEmpty()) {

return data;

}

for (int i = 0; i < flights.size(); i++) {

Flight t = flights.get(i);

Vector vector = new Vector();

vector.add(t.getFlightNo());

vector.add(t.getStartAddress());

vector.add(t.getEndAddress());

vector.add(t.getStartTime());

vector.add(t.getArrivedTime());

vector.add(t.getAirLineName());

vector.add(t.getPlaneType());

vector.add(t.getPrice());

data.add(vector);

}

return data;

}

// 内部类座位鼠标行为监听器

class MyTableMouseMotionListener extends MouseAdapter{

private int rowUnderMouse = -1;

public void mouseClicked(MouseEvent e){

JTable table = (JTable)e.getSource();

Point p = table.getMousePosition();

if(p != null){

rowUnderMouse = table.rowAtPoint(p);

if(rowUnderMouse >= 0){

for(int i=0;i<table.getColumnCount();i++){

table.prepareRenderer(table.getCellRenderer(rowUnderMouse,i),rowUnderMouse,i);

if(rowUnderMouse != 0){

table.prepareRenderer(table.getCellRenderer(rowUnderMouse-1,i),rowUnderMouse-1,i);

}

if(rowUnderMouse != table.getRowCount()-1){

table.prepareRenderer(table.getCellRenderer(rowUnderMouse+1,i),rowUnderMouse+1,i);

}

}

table.repaint(table.getVisibleRect());

}

}

}

}

}

### 测试结果



图4.1 首页



图4.2 查看所有机票

## 添设管理员权限

### 源代码

**package com.booking.view;中添加ManegerView类**

public class ManegerView extends BaseFrame {

/\*\*

\* @Fields serialVersionUID

\*/

private static final long serialVersionUID = 2870945371327830406L;

private ActionListener actionListener;

/\*\*

\* @Fields searchCondition : 搜索条件

\*/

private JTextField searchCondition;

/\*\*

\* @Fields confirmBtn : 确认按钮

\*/

private JButton confirmBtn;

/\*\*

\* @Fields backBtn : 返回按钮

\*/

private JButton backBtn;

public ManegerView(ActionListener actionListener) {

this.actionListener = actionListener;

init();

}

private void init() {

// 设置背景图片

ViewBackgroundUtil.setBG(this, "img/bg2.jpg");

// 始发地标签

JLabel noteLabel = new JLabel("请输入管理员密码");

noteLabel.setBounds(295, 60, 240, 33);

// 搜索条件文本框

searchCondition = new JTextField();

searchCondition.setBounds(295, 100, 200, 33);

// 实例化确认按钮

confirmBtn = new CustomButton(480, 460, CustomButton.RIGHT);

confirmBtn.setText("确认");

confirmBtn.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

// 获取输入的订单号或身份证号

String inputItem = searchCondition.getText();

if ("".equals(inputItem)) {

JOptionPane.showMessageDialog(ManegerView.this, "请输入管理员密码！",

"提示信息", JOptionPane.ERROR\_MESSAGE);

} else {

if (!"123456".equals(inputItem)) {

JOptionPane.showMessageDialog(ManegerView.this, "管理员密码输入错误，请重新输入！",

"提示信息", JOptionPane.ERROR\_MESSAGE);

return;

}

// 传递订单号或身份证号到订单窗口

ManegerView.this.setVisible(false);

AddFlightLsitView addFlightLsitView=new AddFlightLsitView();

addFlightLsitView.setVisible(true);

}

}

});

// 实例化返回按钮

backBtn = new CustomButton(180, 460, CustomButton.LEFT);

backBtn.setText("返回");

backBtn.addActionListener(actionListener);

backBtn.setActionCommand("toMainView");

this.getContentPane().add(noteLabel);

this.getContentPane().add(searchCondition);

this.getContentPane().add(backBtn);

this.getContentPane().add(confirmBtn);

this.setLocationRelativeTo(null);

// 设置窗口无标题栏

this.setUndecorated(true);

this.getRootPane().setWindowDecorationStyle(JRootPane.NONE);

}

}

**package com.booking.controller;中BookingController类进行修改**

监听按钮时

case "addFlight":

// 显示添加机票票窗口

manegerView=new ManegerView(this);

manegerView.setVisible(true);

break;

### 测试结果



图4.3点击创建新的机票按钮后键入错误密码

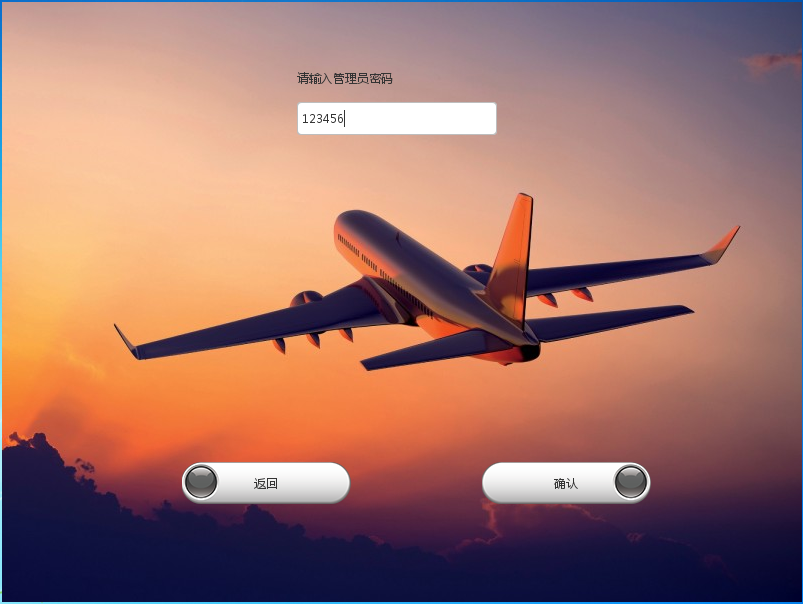


图4.4输入正确密码



图4.5进入添加界面

## 4.3 选完座位机票增加座位信息

### 4.3.1 源代码的修改

**1 Ticket类中增加属性**

public String getSeat() {

return seat;

}

public void setSeat(String seat) {

this.seat = seat;

}

/\*\*

\*座位

\*/

private String seat;

package com.booking.service.impl;

**2 BookingServiceImpl类中的createSeatInfo方法增加**

if (!"".equals(order)) {

// 读取本地磁盘数据

List<Ticket> ticketList = dataOperateUtil.readTicketData(Constants.FILE\_NAME);

// 判断数据是否为空

if (ticketList != null) {

// 循环获取购票信息

for (Ticket ticket : ticketList) {

// 获取订单编号

String order1 = ticket.getOrderNo();

// 判断获取的订单编号与参数是否相等，相等则修改该订单出行时间

if (order1.equals(order)) {

ticket.setSeat(seat.getSeatNo());

}

}

}

operateResult = dataOperateUtil.writeTicketData(ticketList);

}

//Map<String,Object> resultMap = new HashMap<>();

// 返回消息

resultMap.put(Constants.RESULT\_KEY\_CODE, operateResult);

// 操作成功返回卡对象

if (operateResult.equals(Constants.MSG\_SUCCESS)) {

resultMap.put(Constants.RESULT\_KEY\_DATA, Constants.MSG\_SUCCESS);

}

**3 BookingView中ticket写入时增加**

ticket.setSeat("未选");

**4 AlterOrderView类中**

columnNames.add("座位"); //增加代码

和

vector.add(t.getSeat());

### 4.3.1 测试结果



图4.6订单后面追加了座位信息。

# 三、实训总结

对于代码量较大的系统，首先应该弄懂类与类之间的继承与调用关系，每个类的作用，为什么要创立这个类，甚至哪些类应该打包放在同一个包里，这样可以使读代码的逻辑更清晰。

在此前提下再看每个类的属性和方法，会发现更加确定了面向对象封装的好处，这也是他与面向对象之间的巨大不同之处。也明白了在这个飞速发展的信息化时代，为什么面向对象会这么火，Java为什么这么受游戏开发者的欢迎。

总的来说，这次实训还是比较成功的，使个人动手能力得到了很好的锻炼。在实训过程中，老师交给了很多技巧。我们也学到了，做好一个项目固然重要，同时一个代码的优化也是必不可少的，无论从开发者的角度，还是使用者的角度来看都很重要。但仅凭这十多天的学习还远远不够，所以在今后的学习中要更加努力不断提升自己的能力，不断完善自己，做一个出色的工程师。