
Software Design Description

for

Java Air

Prepared by Rui Zhang

Team: Avian Limited

Contents

1.	Introduction	1
1.1	Purpose	1
1.2	Scope	1
1.3	Definitions, acronyms, and abbreviations	1
2.	Reference	3
3.	Decomposition description	4
3.1	Module decomposition	4
3.1.1	CustomerDAO Entity	4
3.1.2	ReservationDAO Entity	4
3.1.3	FlightDAO Entity	4
3.1.4	EmployeeDAO Entity	4
3.1.5	MySQLReservationQueryBuilder Entity	4
3.1.6	MySQLCustomerQueryBuilder Entity	4
3.1.7	MySQLFlightQueryBuilder Entity	5
3.1.8	MySQLEmployQueryBuilder Entity	5
3.1.9	Aircraft Entity	5
3.1.10	FlightMaintenanceStatus Entity	5
3.1.11	PriceCalculator Entity	5
3.1.12	GenerateCustomerNum Entity	5
3.1.13	GenerateReservationNum Entity	5
3.1.14	GenerateFlightNum Entity	6
3.1.15	PanelMap Entity	6
3.1.16	DataAccess Entity	6
3.2	Concurrent process decomposition	6
3.2.1	CustomerDBController Entity	6
3.2.2	ReservationDBController Entity	6
3.2.3	FlightDBController Entity	6
3.2.4	EmployeeDBController Entity	6
3.2.5	CustomerLandingController Entity	7
3.2.6	CustomerPasswordResetController Entity	7
3.2.7	FlightStatusController Entity	7
3.2.8	FlightSearchController Entity	7

Software Design Description for Java Air

3.2.9	FlightSelectController Entity.....	7
3.1.10	PanelSwitch Entity	7
3.2.11	ReservPassengerContinueController Entity	8
3.2.12	ReservBillInfoContinueController Entity.....	8
3.2.13	ReservConfirmController Entity	8
3.3	Data decomposition	8
3.3.1	Customer Entity	8
3.3.2	Flight Entity.....	8
3.3.3	Reservation Entity	8
3.3.4	Employee Entity	8
3.3.5	Passenger Entity	8
4.	Dependency description	10
4.1	Intermodule dependencies	10
4.1.1	Overview	10
4.1.2	Model dependencies	12
4.1.3	View dependencies	13
4.1.4	Controller dependencies	14
4.1.5	Database dependencies	15
4.2	Interprocess dependencies	16
4.3	Data dependencies	16
5.	Interface description	17
5.1	Module interface.....	17
5.1.1	Home Panel	17
5.1.2	Flight Status Panel	18
5.1.3	Check in Panel	18
5.1.4	Flight Result Panel	19
5.1.5	Reservation Passenger Panel	20
5.1.6	Reservation Bill information Panel	20
5.1.7	Reservation Confirm Panel.....	21
5.1.8	Account Welcome Panel	21
5.1.9	Account Registration Panel	22
5.1.10	Account Password Reset Panel.....	22
5.1.11	Account Reservation Panel.....	23
5.1.12	Account Reward Panel	24
5.2	Process interface	24

6.	Detailed design	25
6.1	Module detailed design.....	25
6.1.1	CustomerDAO Detail	25
6.1.2	ReservationDAO Detail.....	26
6.1.3	FlightDAO Detail	27
6.1.4	EmployeeDAO Detail.....	28
6.1.5	MySQLCustomerQueryBuilder Detail.....	29
6.1.6	MySQLReservationQueryBuilder Detail	29
6.1.7	MySQLFlightQueryBuilder Detail.....	30
6.1.8	MySQLEmployeeQueryBuilder Detail	31
6.1.8	Aircraft Detail.....	31
6.1.9	FlightMaintenaceStatus Detail	32
6.1.10	PriceCalculator Detail.....	32
6.1.11	GenerateReservationNum Detail.....	34
6.1.12	GenerateFlightNum Detail	34
6.1.13	GenerateCustomerNum Detail	34
6.1.14	PanelMap Detail	34
6.1.15	DataAccess Detail	36
6.1.16	CustomerDBController Detail.....	37
6.1.17	ReservationDBController Detail	37
6.1.18	FlightDBController Detail.....	38
6.1.19	EmployeeDBController Detail	38
6.1.20	CustomerLandingController Detail	38
6.1.21	CustomerPasswordResetController Detail	39
6.1.22	FlightStatusController Detail.....	40
6.1.23	FlightSearchController Detail.....	40
6.1.24	FlightSelectController Detail.....	41
6.1.25	PanelSwitch Detail	42
6.1.26	ReservPassengerContinueController Detail	42
6.1.27	ReservBillInfoContinueController Detail.....	42
6.1.28	ReservConfirmController Detail	43
6.2	Data detailed design	43
6.2.1	Customer Detail.....	43
6.2.2	Flight Detail.....	45
6.2.3	Reservation Detail	48

Software Design Description for Java Air

6.2.4	Employee Detail	50
6.2.5	Passenger Detail	53

Figure

Figure 1 Module overview	11
Figure 2 Model modules overview	12
Figure 3 View modules overview	13
Figure 4 Controller modules overview	14
Figure 5 Database modules overview	15
Figure 6 Database SQL diagram	16
Figure 7 Home panel	17
Figure 8 Flight status panel	18
Figure 9 Check in panel	18
Figure 10 Flight result panel	19
Figure 11 Reservation passenger panel	20
Figure 12 Reservation bill information panel	20
Figure 13 Reservation confirm panel	21
Figure 14 Account welcome panel	21
Figure 15 Account registration panel	22
Figure 16 Account password reset panel	22
Figure 17 Account reservation panel	23
Figure 18 Account reward panel	24
Figure 19 CustomerDAO class	25
Figure 20 ReservationDAO class	26
Figure 21 FlightDAO class	27
Figure 22 EmployeeDAO class	28
Figure 23 MySQLCustomerQueryBuilder class	29
Figure 24 MySQLReservationQueryBuilder class	30
Figure 25 MySQLFlightQueryBuilder class	30
Figure 26 MySQLEmployeeQueryBuilder class	31
Figure 27 Aircraft class	32
Figure 28 FlightMaintenanceStatus class	32
Figure 29 PriceCalculator class	33
Figure 30 GenerateReservationNum class	34
Figure 31 GenerateReservationNum class	34
Figure 32 GenerateCustomerNum class	34
Figure 33 PanelMap class	35
Figure 34 DataAccess class	37
Figure 35 CustomerDBController class	37
Figure 36 ReservationDBController class	38
Figure 37 FlightDBController class	38
Figure 38 EmployeeDBController class	38
Figure 39 CustomerLandingController class	39
Figure 40 CustomerPasswordResetController class	39
Figure 41 FlightStatusController class	40
Figure 42 FlightSearchController class	40
Figure 43 FlightSelectController class	41
Figure 44 PanelSwitch class	42

Software Design Description for Java Air

Figure 45 ReservPassengerContinueController class	42
Figure 46 ReservBillInfoContinueController class	43
Figure 47 ReservConfirmController class	43
Figure 48 Customer class	44
Figure 49 Flight class	46
Figure 50 Reservation class	49
Figure 51 Employee class	51
Figure 52 Passenger class	53

Revision history

Name	Date	Changes Made	Version
Rui Zhang	2016-09-28	Initial draft. Based on the IEEE standard, the outline of document is implemented. The basic MVC design pattern is applied in the design.	1.0
Rui Zhang	2016-10-15	The chapter 3 Decomposition description, chapter 4 Dependency description and chapter 5 Interface description are completed.	2.0
Rui Zhang	2016-10-25	Based on the changes in SRS, the corresponding parts are revised.	3.0
Rui Zhang	2016-11-02	The detail design is finished. Based on the latest version of SRS, the document and UML diagram is revised.	4.0
Rui Zhang	2016-11-20	In the development, some detail design is modified. Update the interface design.	4.2

1. Introduction

1.1 Purpose

The Software Requirements Specification (SRS) is intended to describe the software architecture and design in detail, which is used to guide the team members' code development. This document also describe the implantation of front-end and back-end in detail.

1.2 Scope

This document will cover the software design for release version 0.0.1 of the Java Air software product, and the functional and non-functional requirements in the Software Requirements Specification (SRS) will be satisfied. The details of architecture, module description, process description, database design and algorithm design will be demonstrated.

1.3 Definitions, acronyms, and abbreviations

CI = Configuration Item

CMMI = Capability Maturity Model Integration

IEEE = Institute of Electrical and Electronics Engineers

QA = Quality Assurance

SEI = Software Engineering Institute

SCMP = Software Configuration Management Plan

SPMP = Software Project Management Plan (this document)

SRS = Software Requirements Specification

SDD = Software Design Document

SQAP = Software Quality Assurance Plan

SVVP = Software Verification and Validation Plan

STP = Software Test Plan

UD = User Documentation

WBS = Work Breakdown Structure

U/PD = User/Product Director

PM = Project Manager

Software Design Description for Java Air

RE = Requirement Engineer

SA = Software Architect

IE = Integration Engineer

TE = Testing Engineer

CD = Code Developer

2. Reference

Software Engineering Modern Approaches, 2nd ed. By Eric J. Braude, Michael E. Bernstein.

Software Configuration Management Plan (SCMP) for Java Air

Software Design Description (SDD) for Java Air

Software Project Management Plan (SPMP) for Java Air

Software Quality Assurance Plan (SQAP) for Java Air

Software User Documentation Plan (SUDP) for Java Air

Software Test Document (STD) for Java Air

Software Verification and Validation Plan (SVVP) for Java Air

3. Decomposition description

3.1 Module decomposition

3.1.1 CustomerDAO Entity

CustomerDAO module is providing customer database support. The functions in this module will implement DataAccess method. Based on user choices, the function will update the database or return the customer information as Customer object type. A detailed description will be given in section 6.

3.1.2 ReservationDAO Entity

ReservationDAO module is providing reservation database support. The functions in this module will implement DataAccess method. Based on user choices, the function will update the database or return the reservation information as Reservation object type. A detailed description will be given in section 6.

3.1.3 FlightDAO Entity

FlightDAO module is providing flight database support. The functions in this module will implement DataAccess method. Based on user choices, the function will update the database or return the flight information as Flight object type. A detailed description will be given in section 6.

3.1.4 EmployeeDAO Entity

EmployeeDAO module is providing employee database support. The functions in this module will implement DataAccess method. Based on user choices, the function will update the database or return the employee information as Employee object type. A detailed description will be given in section 6.

3.1.5 MySQLReservationQueryBuilder Entity

MySQLCustomerQueryBuilder is designed to obtain specific executable database query. The responsibility of this module is to support the CustomerDAO module. In every method, the information in the input Customer object will be used to build the specific query. A detailed description will be given in section 6.

3.1.6 MySQLCustomerQueryBuilder Entity

MySQLReservationQueryBuilder is designed to obtain specific executable database query. The responsibility of this module is to support the ReservationDAO module. In every method, the information in the input Reservation object will be used to build the specific query. A detailed description will be given in section 6.

3.1.7 MySQLFlightQueryBuilder Entity

MySQLFlightQueryBuilder is designed to obtain specific executable database query. The responsibility of this module is to support the FlightDAO module. In every method, the information in the input Flight object will be used to build the specific query. A detailed description will be given in section 6.

3.1.8 MySQLEmployeeQueryBuilder Entity

MySQLEmployeeQueryBuilder is designed to obtain specific executable database query. The responsibility of this module is to support the EmployeeDAO module. In every method, the information in the input Employee object will be used to build the specific query. A detailed description will be given in section 6.

3.1.9 Aircraft Entity

Aircraft is a module for holding information of different aircraft type. There are no setter method to reset aircraft information, because aircraft type information is fixed. Private attributes are used to store aircraft information. The getter method will be included in this module for obtaining aircraft information. A detailed description will be given in section 6.

3.1.10 FlightMaintenanceStatus Entity

This entity is used to hold flight maintenance status information, every kind of maintenance problem will delay the departure of airplane. A detailed description will be given in section 6.

3.1.11 PriceCalculator Entity

PriceCalculator is designed to support price calculation in a reservation. A PriceCalculator object has fare, tax fee, total fee, reward point usage, rate of reward to cash and rate of cash to reward attributes, and corresponding getter method is provided in this class. In addition, the reward point usage in a reservation could be set in this class. A detailed description will be given in section 6.

3.1.12 GenerateCustomerNum Entity

This entity is used to generate customer number in specific format. A detailed description will be given in section 6.

3.1.13 GenerateReservationNum Entity

This entity is used to generate reservation number in specific format. A detailed description will be given in section 6.

3.1.14 GenerateFlightNum Entity

This entity is used to generate flight number in specific format. A detailed description will be given in section 6.

3.1.15 PanelMap Entity

PanelMap module is the core of panel system. The JPanel hashmap structure in this module holds all panel reference. The getter method in this PanelMap module should return specific panel reference, and getting panel string name of panel methods are included in this module. A detailed description will be given in section 6.

3.1.16 DataAccess Entity

DataAccess module supports CustomerDAO module, FlightDAO module, ReservationDAO module and EmployeeDAO module. All basic database operation functions are included in this module. The database connection function is the most necessary function. This function is designed as a public static function, so database could be connected to immediately. A detailed description will be given in section 6.

3.2 Concurrent process decomposition

3.2.1 CustomerDBController Entity

The CustomerDBController is only responsible for getting a new CustomerDAO object and return to required module. A detailed description will be given in section 6.

3.2.2 ReservationDBController Entity

The ReservationDBController is only responsible for getting a new ReservationDAO object and return to required module. This controller is the bridge between DAO module and other controller module. A detailed description will be given in section 6.

3.2.3 FlightDBController Entity

The FlightDBController is only responsible for getting a new FlightDAO object and return to required module. This controller is the bridge between DAO module and other controller module. A detailed description will be given in section 6.

3.2.4 EmployeeDBController Entity

The EmployeeDBController is only responsible for getting a new EmployeeDAO object and return to required module. This controller is the bridge between DAO module and other controller module. A detailed description will be given in section 6.

3.2.5 CustomerLandingController Entity

This CustomerLandingController is used to check user input for login system, and support the customer login panel. If the customer exits, all customer information will be loaded and switch current panel to account panel, or error window will be thrown and keep the CustomerLanding panel visible for next login attempt. A detailed description will be given in section 6.

3.2.6 CustomerPasswordResetController Entity

This CustomerPasswordResetController is used to reset customer password. The customer information will be obtained from the input of password reset panel and corresponding database information will be got by implementing CustomerDAOController Module, then this controller will check the accuracy of customer information. The password cannot be reset except that the customer input information is correct. A detailed description will be given in section 6.

3.2.7 FlightStatusController Entity

This entity is used to control FlightStatus panel to display all active flight status and switch from home panel to flight status panel. The FlightDAOController will be called to obtain all flights information. A detailed description will be given in section 6.

3.2.8 FlightSearchController Entity

This entity is used to search user input flight and control FlightResult panel to show search results. The related flight information from Database will be obtained and display on the FlightResult panel. A detailed description will be given in section 6.

3.2.9 FlightSelectController Entity

This entity is used to select Flight and switch panel. The first step of this controller is to check whether this trip is one way. If the trip is one way, the select flight will be stored into current reservation as departure flight and switch from FlightResult panel to ReservationPassengerPanel. If the trip is round trip and the departure flight has not been selected, the select flight will be stored into current reservation as departure flight and control the FlightResult panel to show return flights. If the trip is round trip and the departure flight has been selected, the select flight will be stored into current reservation as return flight and switch from FlightResult panel to ReservationPassengerPanel. A detailed description will be given in section 6.

3.1.10 PanelSwitch Entity

This PanelSwitch controller is used to control current content window to display specific panel. The PanelMap module is applied in this module. The string name of specific panel is required as input for setting method. A detailed description will be given in section 6.

3.2.11 ReservPassengerContinueController Entity

This ReservPassengerContinueController is used to check all user input on ReservationPassenger panel, store available passenger information to current reservation and switch to ReservationBillinformation panel. If the input does not pass the check process, an error window will be thrown and keep current panel visible. A detailed description will be given in section 6.

3.2.12 ReservBillInfoContinueController Entity

This ReservBillInfoContinueController is used to check user input on ReserationBillinformation panel, store available bill information to current reservation and switch to ReservationConfirm panel. If the input does not pass the check process, an error window will be thrown and keep current panel visible. A detailed description will be given in section 6.

3.2.13 ReservConfirmController Entity

This ReservConfirmController is used to save current reservation into database, if user click the confirm button on reservation confirmation panel. The ReservationDAOController would be used in this module. A detailed description will be given in section 6.

3.3 Data decomposition

3.3.1 Customer Entity

This entity represents the customer account, which holds all related qualities about customer, such as name, gender, data of birth and so on. A detailed description will be given in section 6.

3.3.2 Flight Entity

This entity holds the all qualities of flights. A detailed description will be given in section 6.

3.3.3 Reservation Entity

The reservation qualities are held in this entity, like reservation number, passenger information, flight number and so on. A detailed description will be given in section 6.

3.3.4 Employee Entity

This entity represents the detailed account information of employee. A detailed description will be given in section 6.

3.3.5 Passenger Entity

This entity represents the detailed account information of employee. A detailed description will be given in section 6.

4. Dependency description

4.1 Intermodule dependencies

4.1.1 Overview

The design implements the Model-view-controller (MVC) software design pattern, because this is a user interfaces software. The following Figure show the overview of class diagram design. The model includes customer, flight, reservation, employee modules to store corresponding information. The view part is responsible for the GUI. The user input information will be obtained from view module, then controller will use this information to contact with database and control view to show specific information.

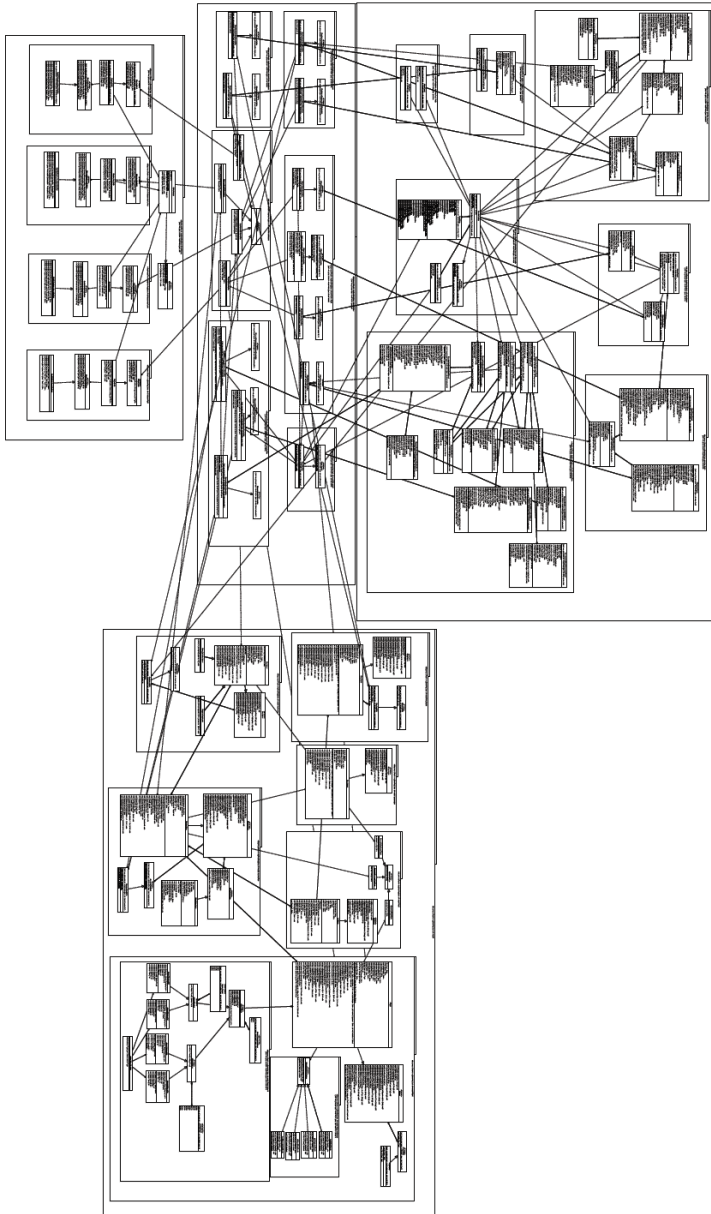


Figure 1 Module overview

4.1.2 Model dependencies

The model includes customer, flight, reservation and employee modules in general. The relationship is shown in the following figure.



Figure 2 Model modules overview

4.1.3 View dependencies

The principle of view design is that all panels are included in the JPanel hashmap structure. Every panel is stored in this structure. The relationship of view is displayed in the following figure.

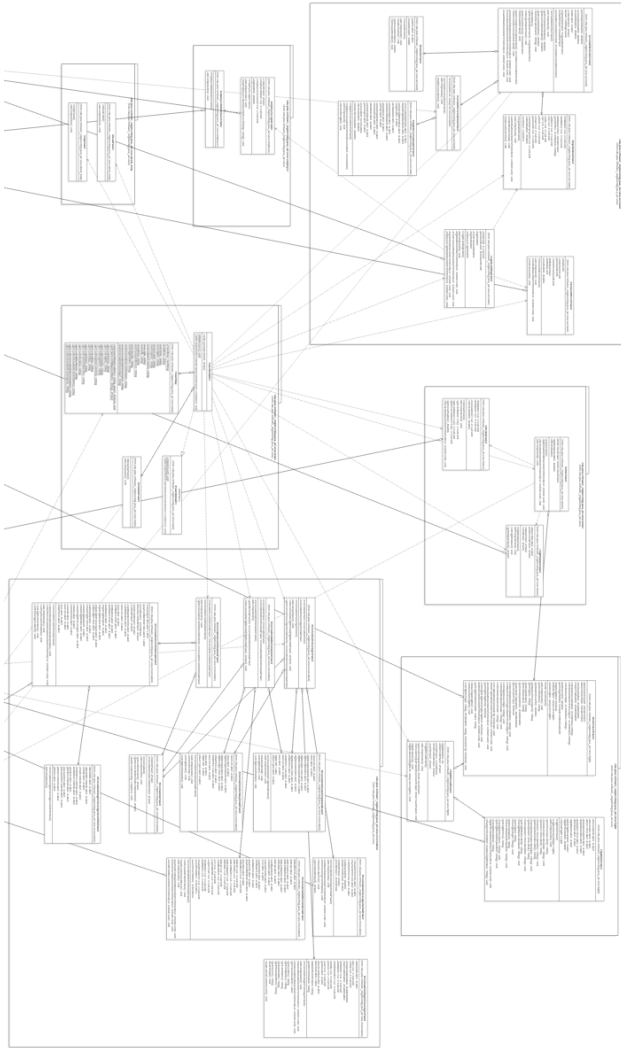


Figure 3 View modules overview

4.1.4 Controller dependencies

In controller part, PanelSwitch is used by other controller frequently. For database operations, the CustomerDAOController, FlightDAOController, ReservationDAOController and EmployeeDAOController is reusable. The relationship between controllers are shown in the following figure.

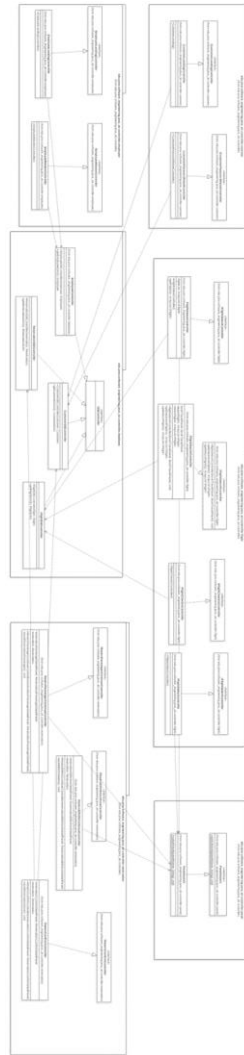


Figure 4 Controller modules overview

4.1.5 Database dependencies

The database module includes Customer database, Flight database, Reservation database and Employee database. The DataAccess module provide database operation methods. The relationship between different modules are shown in the following figure.

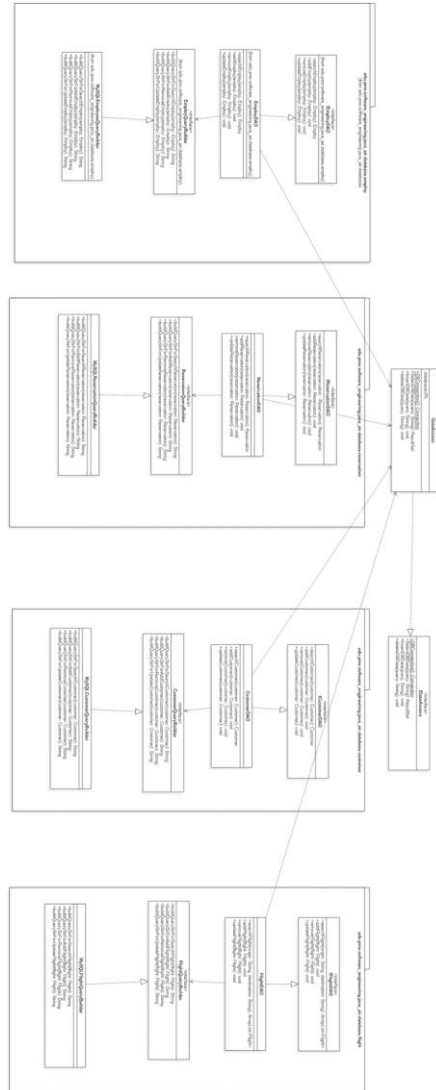
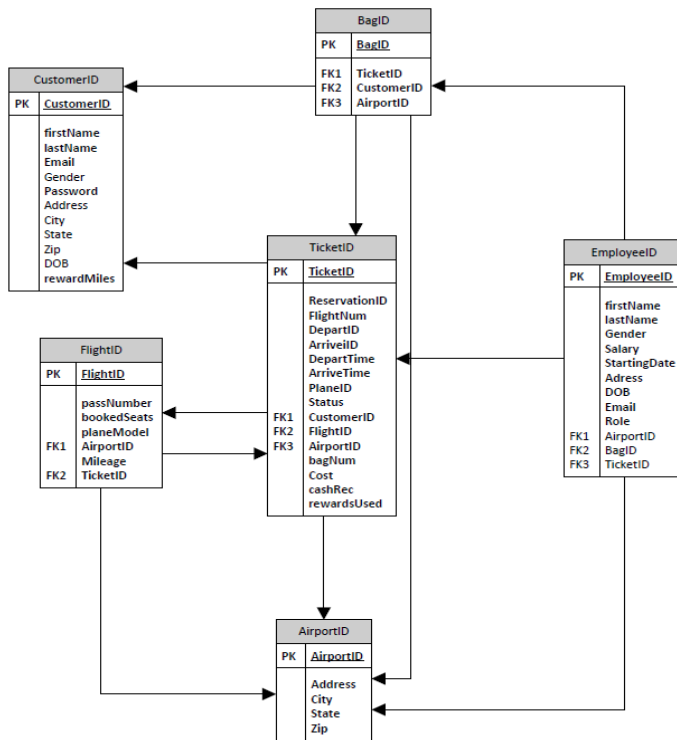


Figure 5 Database modules overview

4.2 Interprocess dependencies

4.3 Data dependencies



Java Air Data Base Diagram 1.0

Figure 6 Database SQL diagram

5. Interface description

5.1 Module interface

5.1.1 Home Panel

This is the initial display for a new instance of the Java Air software application. This form will show the title text/picture, the menu bar, as well as a set of welcome information and shortcuts. This includes flight-search inputs, link to flight-status, reservation-status and check-in forms. Optionally, if the customer user is logged in, it will display the account's rewards information, or else it will display a "signup for rewards" button that links to the user registration



Figure 7 Home panel

5.1.2 Flight Status Panel

This form is intended for guest customers to view reservation information and status (valid, cancelled, etc.). The form requires a combination of reservation ID and passenger name in order to display the search result.

Java Air Flight Status					
JAF0001 Chicago 12:20 Tue, Nov 22	New York 14:40 Tue, Nov 22	nonstop 2.4	On Time	JAF0002 New York 10:40 Tue, Nov 22	Chicago 13:44 Tue, Nov 22
JAF0003 Chicago 10:20 Tue, Nov 22	Los Angeles 14:23 Tue, Nov 22			JAF0005 New York 09:15 Tue, Nov 22	Los Angeles 12:55 Tue, Nov 22
JAF0006 Los Angeles 10:30 Tue, Nov 22	New York 13:00 Tue, Nov 22	nonstop 5.5	On Time	JAF0007 Chicago 16:45 Tue, Nov 22	New York 19:50 Tue, Nov 22
JAF0009 Chicago 07:00 Tue, Nov 22	St. Louis 08:35 Tue, Nov 22	nonstop 1.5	On Time	JAF0010 St. Louis 19:30 Tue, Nov 22	Chicago 20:40 Tue, Nov 22

Figure 8 Flight status panel

5.1.3 Check in Panel

This form is intended for guest customers to check into their flight(s). The form requires reservation ID, first name, and last name to be entered before passengers are checked-in.

Check In

Fill out the fields below to check in for your flight:

Reservation ID Number:

First: Last:

CHECK IN

Figure 9 Check in panel

5.1.4 Flight Result Panel

This form is where a customer can view a list of flights. There should be two sections of this form. The left-hand side contains the search-input functions, and the right-hand side contains the search result. The user must enter trip-type, origin, destination, departure date, and number of passengers into the search-input fields before clicking on the search button. Returning date shall be required if the customer chose “round-trip” as his trip-type. After the search button is clicked, a list of matching flights will be displayed on the right-hand side of the form. Then the user can select a departing flight and/or returning flight and click the “purchase” button.

Welcome to Smooth Flying!

JAVA AIR

Home Flights Account Help About Login

Book travel

From: Chicago To: New York

Departure: Tue 12/13/2016 Return: Thu 12/22/2016

Passenger: 1

☒ Roundtrip ☐ One-way

Find Flights

JAF0001
Chicago New York nonstop
12:20 -----> 14:40 2.4 \$177.6
Tue, Dec 13 Tue, Dec 13

JAF0007
Chicago New York nonstop
16:45 -----> 19:50 2.25 \$177.6
Tue, Dec 13 Tue, Dec 13

Figure 10 Flight result panel

5.1.5 Reservation Passenger Panel

Welcome to Smooth Flying!

JAVA AIR

Home Flights Account Help About Login

Traveler ----- Payment ----- Confirm

Traveler 1 - primary traveler

First name* Last name*

Date of birth* Gender*
MM* DD* YYYY* ☐ Male ☐ Female

Know Traveler Number/PASS ID, 0 if Guest

Travel contact information
Mobile number Email address

Continue

Flight

RoundTrip (1 traveler)

Tue, Dec 13
Chicago - New York 12:20 - 14:40

Thu, Dec 22
New York - Chicago 10:40 - 13:44

Fare \$177.6
Taxes and fees \$25.43
TOTAL \$203.03

Figure 11 Reservation passenger panel

5.1.6 Reservation Bill information Panel

Welcome to Smooth Flying!

JAVA AIR

Home Flights Account Help About Login

Traveler ----- Payment ----- Confirm

Payment

First name Last name

Card number

Expire date CVC

Billing Address

First name Last name

Street address

City State/Province

Country Postal Code

Continue

Flight

RoundTrip (1 traveler)

Tue, Dec 13
Chicago - New York 12:20 - 14:40

Thu, Dec 22
New York - Chicago 10:40 - 13:44

Fare \$177.6
Taxes and fees \$25.43
TOTAL \$203.03

Figure 12 Reservation bill information panel

5.1.7 Reservation Confirm Panel

This form is a confirmation form intended to be displayed after a successful reservation purchase. The confirmation form should display the flight details, passenger information, purchase dollar or point amount, remaining reward point and a unique reservation number.

Figure 13 Reservation confirm panel

5.1.8 Account Welcome Panel

This form should be displayed after a customer user has logged in. The welcome form should display a menu list, “option list”, that shows “personal information”, “reservations”, and “rewards”. The welcome form should be simple and shows a welcome message for the user by displaying “Hello <name>”. Optionally, the welcome page can also display upcoming reservation(s) and rewards summary information.

Figure 14 Account welcome panel

Commented [M1]: Add reward point remaining after transaction

Commented [D2R1]: Revised.

5.1.9 Account Registration Panel

This form shall be where the customer can sign up for a new account. To sign up for a new account, the user must enter first name, last name, address, gender, birthday, phone number, e-mail address, password, and a personal pin number. After entering the required information, the user will click on the “submit” button on the form to send the information to the database.

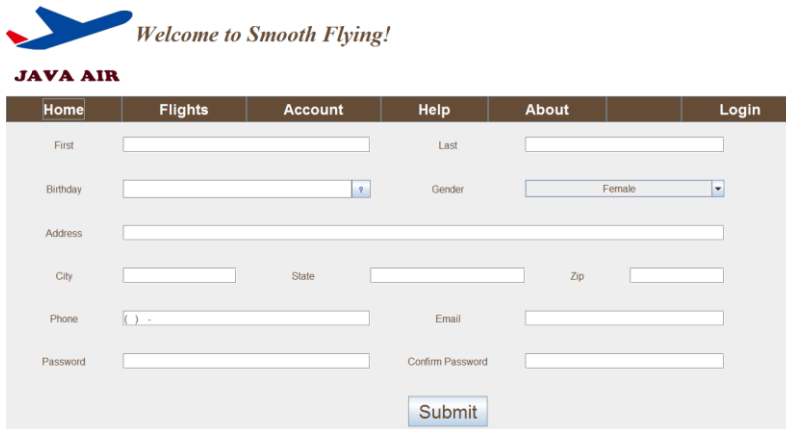


Figure 15 Account registration panel

5.1.10 Account Password Reset Panel

This form is for customer users who forgot their password. Users must enter his e-mail address, date of birth, customer ID in order to type in a new password and to update it.

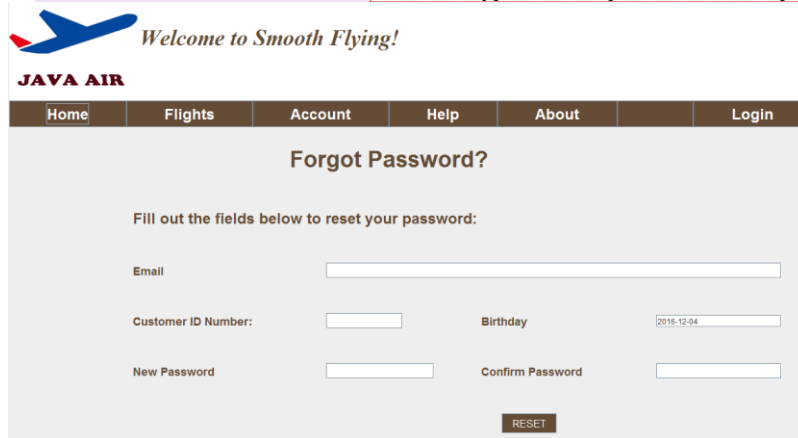


Figure 16 Account password reset panel

Commented [M3]: Are we keeping pin numbers?

Commented [D4R3]: We won't have PIN. Revised.

5.1.11 Account Reservation Panel

This form should be displayed after a customer user has clicked on the “reservations” button on the option list. The option list should still be visible after form-load. The customer user’s reservations shall be displayed on the form in a list, a click on specific reservation on the list will unfolds information including reservation number, flight number(s), origin/destination, departure/arrival date and time, passenger detail, and an option for check-in.



PERSONAL INFORMATION	RESERVATIONS	REWARDS
Reservation # JAR01		
Flight Number: JAF0001	Chicago	New York
	2016-11-09 12:20	2016-11-09 14:40
		Duration: 2.4
Reservation # JAR01		
Flight Number: JAF0002	New York	Chicago
	2016-11-12 10:40	2016-11-12 13:44
		Duration: 2.5
Reservation # JAR02		
Flight Number: JAF0008	Chicago	Los Angeles
	2016-11-15 15:20	2016-11-15 18:04
		Duration: 4.5
Reservation # JAR04		
Flight Number: JAF0005	New York	Los Angeles

Figure 17 Account reservation panel

5.1.12 Account Reward Panel

This form should be displayed after a customer user has clicked on the “rewards” button on the option list. The option list should still be visible after form-load. The customer user’s current reward-point count and rewards-level shall be displayed on this form. Optionally, the rewards-level information is presented as a graph or image, and the customer user has the ability to view their miles earned breakdown report.

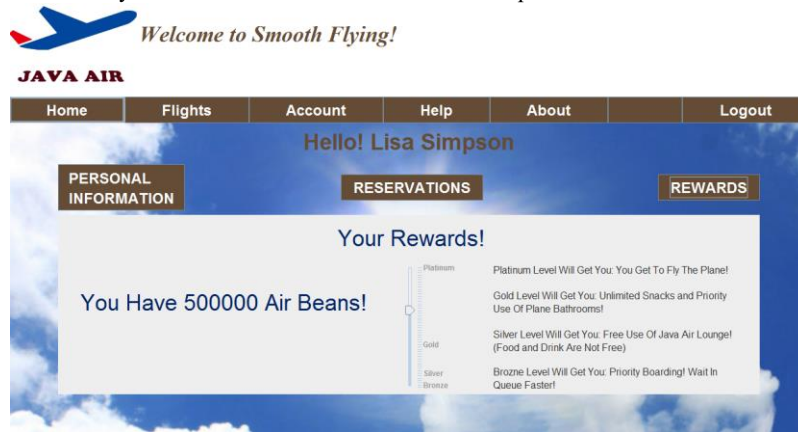


Figure 18 Account reward panel

5.2 Process interface

6. Detailed design

6.1 Module detailed design

6.1.1 CustomerDAO Detail

CustomerDAO module is providing customer database support. The functions in this module will implement DataAccess method. Based on user choices, the function will update the database or return the customer information as Customer object type. The class overview is shown in following figure.

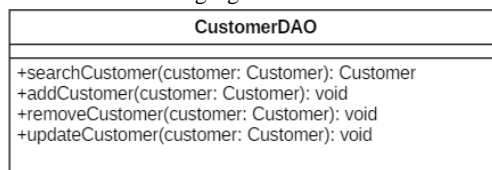


Figure 19 CustomerDAO class

There functions in this module are shown below:

- searchCustomer(in customer:Customer): This function will receive Customer object type input with simple information, like customerID or customerName. The buildQueryStrForSearchCustomer (in customer:Customer) fuction in MySQLCustomerQueryBuilder module will be called and return a corresponding string. The return String will be applied as input for selectDBDate(in query:String) in DataAccess module. The SelectDBDate function will return a ResultSet, this ResultSet will be iterated to obtain every attribute of the specific customer. The attributes are stored in the input Customer object for return.
- addCustomer(in customer:Customer): This function will receive completed Customer object, then call buildQueryStrForAddCustomer(in customer:Customer) function in MySQLCustomerQueryBuilder module to get corresponding string for database adding operation. This string will be applied as the input of insertDBData(in query:String) function in DataAccess module. The task of storage customer information into database will be finished by insertDBData function.
- removeCustomer(in customer:Customer): This function will receive completed Customer object, then call buildQueryStrForRemoveCustomer(in customer:Customer) function in MySQLCustomerQueryBuilder module to get corresponding string for database removing operation. This string will be applied as the input of deleteDBData(in query:String) function in DataAccess module. The task of removing specific customer in database will be finished by deleteDBData function.
- updateCustomer(in customer:Customer): This function will receive completed Customer object, then call buildQueryStrForUpdateCustomer(in customer:Customer) function in MySQLCustomerQueryBuilder module to get corresponding string for database adding operation. This string will be applied as the input of insertDBData(in query:String) function in DataAccess module. The

task of updating customer information into database will be finished by insertDBData function. In this function, updating customer information is completed through overwriting customer information in database, which means that this updateCustoemer function is indeed the same as addCustomer function.

6.1.2 ReservationDAO Detail

ReservationDAO module is providing reservation database support. The functions in this module will implement DataAccess method. Based on user choices, the function will update the database or return the reservation information as Reservation object type.

The class overview is shown in following figure.

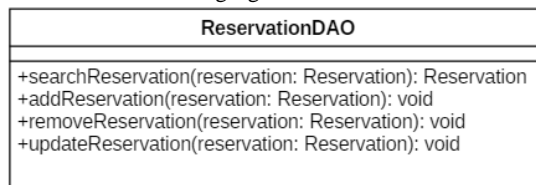


Figure 20 ReservationDAO class

There functions in this module are shown below:

- **searchReservation (in reservation: Reservation):** This function will receive Reservationobject type input with simple information, like reservation number. The buildQueryStrForSearchReservation (in reservation: Reservation) function in MySQLReservationQueryBuilder module will be called and return a corresponding string. The return String will be applied as input for selectDBDate(in query:String) in DataAccess module. The SelectDBDate function will return a ResultSet, this ResultSet will be iterated to obtain every attribute of the specific customer. The attributes are stored in the input Reservation object for return.
- **addReservation (in reservation: Reservation):** This function will receive completed Reservation object, then call buildQueryStrForAdd Reservation (in reservation: Reservation) function in MySQL ReservationQueryBuilder module to get corresponding string for database adding operation. This string will be applied as the input of insertDBData(in query:String) function in DataAccess module. The task of storage customer information into database will be finished by inserDBData function.
- **removeReservation (in reservation: Reservation):** This function will receive completed Reservation object, then call buildQueryStrForRemoveReservation (in reservation: Reservation) function in MySQLReservationQueryBuilder module to get corresponding string for database removing operation. This string will be applied as the input of deleteDBData (in query:String) function in DataAccess module. The task of removing specific reservation in database will be finished by deleteDBData function.
- **updateReservation (in reservation: Reservation):** This function will receive completed Reservation object, then call buildQueryStrForUpdateReservation(in reservation: Reservation) function in MySQLCustomerQueryBuilder module to

get corresponding string for database adding operation. This string will be applied as the input of insertDBData(in query:String) function in DataAccess module. The task of updating customer information into database will be finished by insertDBData function. In this function, updating reservation information is completed through overwriting Reservation information in database, which means that this updateReservation function is indeed the same as addReservation function.

6.1.3 FlightDAO Detail

FlightDAO module is providing flight database support. The functions in this module will implement DataAccess method. Based on user choices, the function will update the database or return the flight information as Flight object type.

The class overview is shown in following figure.

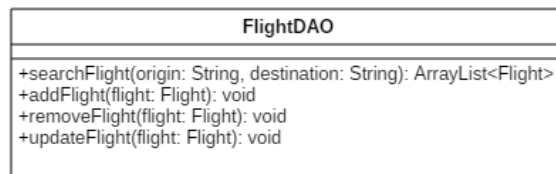


Figure 21 FlightDAO class

There functions in this module are shown below:

- **searchFlight (in flight: Flight):** This function will receive Reservation object type input with simple information, like reservation number. The `buildQueryStrForSearchReservation (in flight: Flight)` function in `MySQLReservationQueryBuilder` module will be called and return a corresponding string. The return String will be applied as input for `selectDBDate(in query:String)` in `DataAccess` module. The `SelectDBDate` function will return a `ResultSet`, this `ResultSet` will be iterated to obtain every specific flight. The attributes are stored in a `Flight` object, all `Flights` will be stored in a `Flight` arraylist as return.
- **addFlight (in flight: Flight):** This function will receive completed `Flight` object, then call `buildQueryStrForAddFlight(in flight: Flight)` function in `MySQLFlightQueryBuilder` module to get corresponding string for database adding operation. This string will be applied as the input of `insertDBData(in query:String)` function in `DataAccess` module. The task of storage flight information into database will be finished by `insertDBData` function.
- **removeFlight (in in flight: Flight):** This function will receive completed `Flight` object, then call `buildQueryStrForRemoveFlight (in in flight: Flight)` function in `MySQLFlightQueryBuilder` module to get corresponding string for database removing operation. This string will be applied as the input of `deleteDBData (in query:String)` function in `DataAccess` module. The task of removing specific reservation in database will be finished by `deleteDBData` function.
- **updateFlight (in flight: Flight):** This function will receive completed `Flight` object, then call `buildQueryStrForUpdateFlight(in flight: Flight)` function in

MySQLFlightQueryBuilder module to get corresponding string for database adding operation. This string will be applied as the input of insertDBData(in query:String) function in DataAccess module. The task of updating customer information into database will be finished by insertDBData function. In this function, updating flight information is completed through overwriting Flight information in database, which means that this updateFlight function is indeed the same as addFlight function.

•

6.1.4 EmployeeDAO Detail

EmployeeDAO module is providing employee database support. The functions in this module will implement DataAccess method. Based on user choices, the function will update the database or return the employee information as Employee object type.

The class overview is shown in following figure.

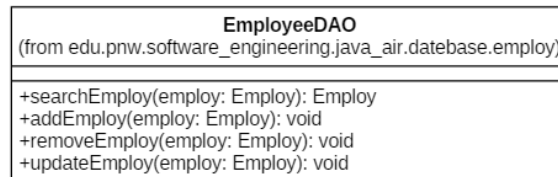


Figure 22 EmployeeDAO class

There functions in this module are shown below:

- searchEmployee (in employee: Employee): This function will receive Employee object type input with simple information, like Employee number. The buildQueryStrForSearchEmployee (in employee: Employee) fuction in MySQLEmployeeQueryBuilder module will be called and return a corresponding string. The return String will be applied as input for selectDBDate(in query:String) in DataAccess module. The SelectDBDate function will return a ResultSet, this ResultSet will be iterated to obtain every attribute of the specific customer. The attributes are stored in the input Employee object for return.
- addEmployee (in employee: Employee): This function will receive completed Employee object, then call buildQueryStrForAdd Employee (in employee: Employee) function in MySQL EmployeeQueryBuilder module to get corresponding string for database adding operation. This string will be applied as the input of insertDBData(in query:String) function in DataAccess module. The task of storage customer information into database will be finished by inserDBData function.
- removeEmployee (in employee: Employee): This function will receive completed Employee object, then call buildQueryStrForRemoveEmployee (in employee: Employee) function in MySQLEmployeeQueryBuilder module to get corresponding string for database removing operation. This string will be applied as the input of deleteDBData (in query:String) function in DataAccess module.

The task of removing specific employee in database will be finished by deleteDBData function.

- updateEmployee (in employee: Employee): This function will receive completed Employee object, then call buildQueryStrForUpdateEmployee(in employee: Employee) function in MySQLCustomerQueryBuilder module to get corresponding string for database adding operation. This string will be applied as the input of insertDBData(in query:String) function in DataAccess module. The task of updating customer information into database will be finished by insertDBData function. In this function, updating employee information is completed through overwriting employee information in database, which means that this updateEmployee function is indeed the same as addEmployee function.

6.1.5 MySQLCustomerQueryBuilder Detail

MySQLCustomerQueryBuilder is designed to obtain specific executable database query. The responsibility of this module is to support the CustomerDAO module. In every method, the information in the input Customer object will be used to build the specific query. The class overview is shown in following figure.

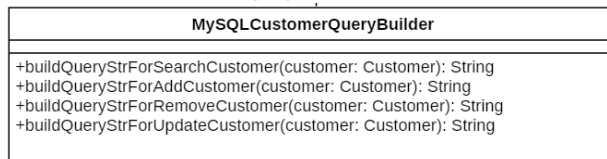


Figure 23 MySQLCustomerQueryBuilder class

There functions in this module are shown below:

- buildQueryStrForSearchCustomer(in customer:Customer): This function will return the query for selecting the input specific customer from database customer table.
- buildQueryStrForAddCustomer(in customer:Customer): This function will return query for inserting the input specific customer into database customer table.
- buildQueryStrForRemoveCustomer(in customer:Customer): This function will return query for removing specific customer in database customer table.
- buildQueryStrForUpdateCustomer(in customer:Customer): This function will return query for inserting specific customer into database customer table. In this project, the update query is exactly the same as the add query, which is kind of repeat. However, this method is still necessary in DAO structure for future extension and modification.

6.1.6 MySQLReservationQueryBuilder Detail

MySQLReservationQueryBuilder is designed to obtain specific executable database query. The responsibility of this module is to support the ReservationDAO module. In every method, the information in the input Reservation object will be used to build the specific query.

The class overview is shown in following figure.

MySQLReservationQueryBuilder
+buildQueryStrForSearchReservation(reservation: Reservation): String +buildQueryStrForAddReservation(reservation: Reservation): String +buildQueryStrForRemoveReservation(reservation: Reservation): String +buildQueryStrForUpdateReservation(reservation: Reservation): String

Figure 24 MySQLReservationQueryBuilder class

There functions in this module are shown below:

- buildQueryStrForSearchReservation(in reservation:Reservation): This function will return query for selecting the input specific reservation from database reservation table.
- buildQueryStrForAddReservation(in reservation:Reservation): This function will return query for inserting the input specific reservation into database reservation table.
- buildQueryStrForRemoveReservation(in reservation:Reservation): This function will return query for removing specific reservation in database reservation table.
- buildQueryStrForUpdateReservation(in reservation:Reservation): This function will return query for inserting specific reservation into database reservation table. In this project, the update query is exactly the same as the add query, which is kind of repeat. However, this method is still necessary in DAO structure for future extension and modification.

6.1.7 MySQLFlightQueryBuilder Detail

MySQLFlightQueryBuilder is designed to obtain specific executable database query. The responsibility of this module is to support the FlightDAO module. In every method, the information in the input Flight object will be used to build the specific query. The class overview is shown in following figure.

MySQLFlightQueryBuilder
+buildQueryStrForSearchFlight(flight: Flight): String +buildQueryStrForAddFlight(flight: Flight): String +buildQueryStrForRemoveFlight(flight: Flight): String +buildQueryStrForUpdateFlight(flight: Flight): String

Figure 25 MySQLFlightQueryBuilder class

There functions in this module are shown below:

- buildQueryStrForSearchFlight(in flight:Flight): This function will return query for selecting the input specific flight from database flight table.
- buildQueryStrForAddFlight(in flight:Flight): This function will return query for inserting the input specific flight into database flight table.
- buildQueryStrForRemoveFlight(in flight:Flight): This function will return query for removing specific flight in database flight table.

- **buildQueryStrForUpdateFlight(in flight:Flight):** This function will return query for inserting specific flight into database flight table. In this project, the update query is exactly the same as the add query, which is kind of repeat. However, this method is still necessary in DAO structure for future extension and modification.

6.1.8 MySQLEmployeeQueryBuilder Detail

MySQLEmployeeQueryBuilder is designed to obtain specific executable database query. The responsibility of this module is to support the EmployeeDAO module. In every method, the information in the input Employee object will be used to build the specific query.

The class overview is shown in following figure.

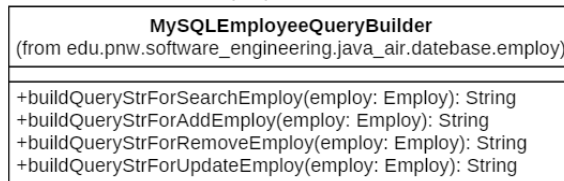


Figure 26 MySQLEmployeeQueryBuilder class

There functions in this module are shown below:

- **buildQueryStrForSearchEmployee(in employee:Employee):** This function will return query for selecting the input specific employee from database employee table.
- **buildQueryStrForAddEmployee(in employee:Employee):** This function will return query for inserting the input specific employee into database employee table.
- **buildQueryStrForRemoveEmployee(in employee:Employee):** This function will return query for removing specific employee in database employee table.
- **buildQueryStrForUpdateEmployee(in employee:Employee):** This function will return query for inserting specific employee into database employee table. In this project, the update query is exactly the same as the add query, which is kind of repeat. However, this method is still necessary in DAO structure for future extension and modification.

6.1.8 Aircraft Detail

Aircraft is a module for holding information of different aircraft type. There are no setter method to reset aircraft information, because aircraft type information is fixed. Private attributes are used to store aircraft information. The getter method will be included in this module for obtaining aircraft information.

The class overview is shown in following figure.

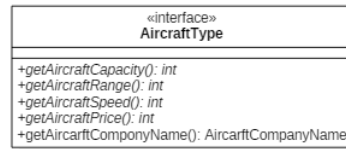


Figure 27 Aircraft class

The attributes in this module are shown below:

- aircraftType: This attribute holds the aircraft type in string type.
- aircraftCapacity: This attribute holds the aircraft capacity in integer type, the accurate value is depend on aircraft type.
- aircraftRange: This attribute holds the aircraft range in integer type, the accurate value is depend on aircraft type.
- aircraftSpeed: This attribute holds the aircraft speed in integer type, the accurate value is depend on aircraft type.

There functions in this module are shown below:

- getAircraftType(): This function will return aircraft name in string type.
- getAircraftCapacity(): This function will return aircraft capacity in string type.
- getAircraftRange(): This function will return aircraft range in string type.
- getAircraftSpeed(): This function will return aircraft speed in string type.

6.1.9 FlightMaintenaceStatus Detail

FlightMaintenaceStatus is a module for holding information of flight maintenance status. This module will support the accurate depart time in Flight module.

The class overview is shown in following figure.

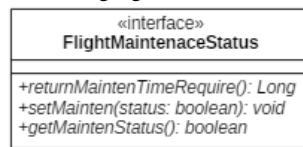


Figure 28 FlightMaintenaceStatus class

The attributes in this module are shown below:

- maintenaceTimeRequire: This attribute holds the maintenance require time.
- maintenaceStatus: boolean: This attribute holds maintenance status.

There functions in this module are shown below:

- returnMaintenTimeRequire(): This attribute holds the aircraft name.
- setMainten(status:boolean): This attribute set maintenance status.
- getMaintenStatus(): boolean

6.1.10 PriceCalculator Detail

PriceCalculator is designed to support price calculation in a reservation. A PriceCalcualtor object has fare, tax fee, total fee, reward point usage, rate of reward to cash and rate of cash to

reward attributes, and corresponding getter method is provided in this class. In addition, the reward point usage in a reservation could be set in this class.

The class overview is shown in following figure.

PriceCalculator
-fare: double -taxFee: double -totalFee: double -rewardPointUse: int -rewardRate2Cash: int -cashRate2Reward: int -decimalFormat: DecimalFormat
+getCashRate(): int +getFare(): String +getFareValue(): double +getRate(): int +getRewardPointsUsed(): String +getRewardRedeem(): String +getRewardRedeemValue(): int +getTaxFee(): String +getTotalFee(): String +getTotalFeeValue(): double +setRewardPointUse(rewardPointUse: int): void

Figure 29 PriceCalculator class

The attributes in this module are shown below:

- fare: This attribute holds the fare number in double type.
- taxFee: This attribute holds tax fee in double type.
- totalFee: This attribute holds total fee in double type.
- rewardPointUse: This attribute holds reward point usage in integer type.
- rewardRate2Cash: This attribute holds rate of reward to cash in integer type.
- rewardRate2Reward: This attribute holds rate of cash to reward in integer type.
- decimalFormat: This attribute holds decimal format of every fee in DecimalFormat type.

There functions in this module are shown below:

- getCashRate(): This function will return rate of reward to cash and rate in integer type.
- getFare(): This function will return fare in string type.
- getFareValue(): This function will return fare in double type.
- getRewardPointsUsed(): This function will return reward point usage in string type.
- getRewardRedeem(): This function will return reward point earned in string type.
- getRewardRedeemValue(): This function will return reward point earned in integer type.
- getTaxFee(): This function will return tax fee in string type.
- getTotalFee(): This function will return the total fee in string type.
- getTotalFeeValue(): This function will return the total fee in double type.
- setRewardPointUse(rewardPointUse: int): This function is used to set how much the reward point is used.

6.1.11 GenerateReservationNum Detail

GenerateReservationNum is a module for generating the reservation number in specific format. This module will support the Reservation module.

The class overview is shown in following figure.

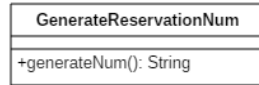


Figure 30 GenerateReservationNum class

There functions in this module are shown below:

- `generateNum()`: This function will return reservation number in specific format.

6.1.12 GenerateFlightNum Detail

GenerateFlightNum is a module for generating the flight number in specific format. This module will support the Fligth module.

The class overview is shown in following figure.

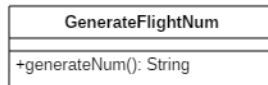


Figure 31 GenerateReservationNum class

There functions in this module are shown below:

- `generateNum()`: This function will return flight number in specific format.

6.1.13 GenerateCustomerNum Detail

GenerateCustomerNum is a module for generating the customer ID in specific format. This module will support the Customer module.

The class overview is shown in following figure.

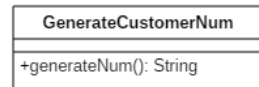


Figure 32 GenerateCustomerNum class

There functions in this module are shown below:

- `generateNum()`: This function will return customer number in specific format.

6.1.14 PanelMap Detail

PanelMap module is the core of panel system. The JPanel hashmap structure in this module holds all panel reference. The getter method in this PanelMap module should return specific panel reference, and getting panel string name of panel methods are included in this module. The class overview is shown in following figure.

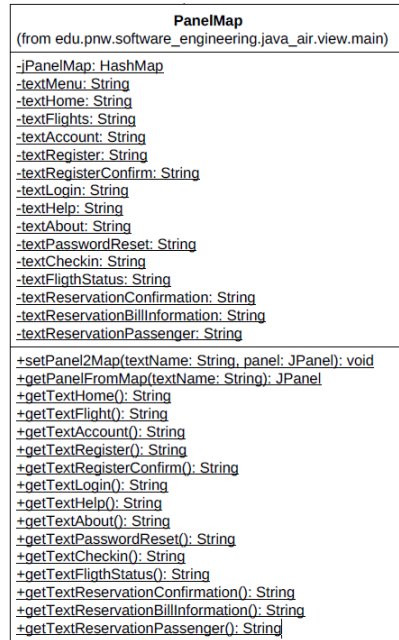


Figure 33 PanelMap class

The attributes in this module are shown below:

- jPanelMap: This private static attribute holds HashMap of all JPanel in the GUI module.
- textMenu: This private static attribute holds menu panel string name.
- textHome: This private static attribute holds home panel string name.
- textFlights: This private static attribute holds flight result panel string name.
- textAccount: This private static attribute holds account panel string name.
- textRegister: This private static attribute holds registration panel string name.
- textRegisterConfirm: This private static attribute holds registration confirm panel string name.
- textLogin: This private static attribute holds login panel string name.
- textHelp: This private static attribute holds help panel string name.
- textAbout: This private static attribute holds about panel string name.
- textPasswordReset: This private static attribute holds password reset panel string name.
- textCheckin: This private static attribute holds check in panel string name.
- textFlightStatus: This private static attribute holds flight status panel string name.
- textReservationConfirmation: This private static attribute holds reservation confirmation panel string name.

- textReservationBillInformation: This private static attribute holds reservation bill information panel string name.
- textReservationPassenger: This private static attribute holds reservation passenger panel string name.

There functions in this module are shown below:

- setPanel2Map (textName: String, panel: JPanel): The public static function will set input panel into panel hashmap structure. The new panel name and JPanel reference are required.
- getPanelFromMap(textName: String): This public static function could return a specific Panel reference from panel hashmap structure. The panel name is required.
- getTextHome(): This public static function returns the string name of home panel.
- getTextFlight(): This public static function returns the string name of flight results panel.
- getTextAccount(): This public static function returns the string name of account panel.
- getTextRegister(): This public static function returns the string name of registration panel.
- getTextRegisterConfirm(): This public static function returns the string name of registration confirm panel.
- getTextLogin(): This public static function returns the string name of login panel.
- getTextHelp(): This public static function returns the string name of help panel.
- getTextAbout(): This public static function returns the string name of about panel.
- getTextPasswordReset(): This public static function returns the string name of password reset panel.
- getTextCheckin(): This public static function returns the string name of check in panel.
- getTextFlightStatus(): This public static function returns the string name of flight status panel.
- getTextReservationConfirmation(): This public static function returns the string name of reservation confirmation panel.
- getTextReservationBillInformation(): This public static function returns the string name of reservation bill information panel.
- getTextReservationPassenger(): This public static function returns the string name of reservation passenger panel.

6.1.15 DataAccess Detail

DataAccess module supports CustomerDAO module, FlightDAO module, ReservationDAO module and EmployeeDAO module. All basic database operation functions are included in this module. The database connection function is the most necessary function. This function is designed as a public static function, so database could be connected to immediately. The class overview is shown in following figure.

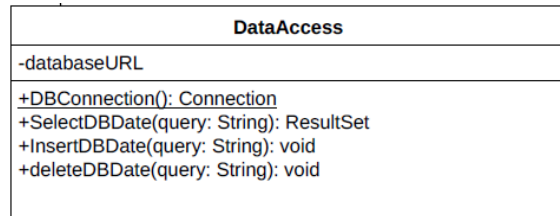


Figure 34 DataAccess class

The attributes in this module are shown below:

- databaseURL: This private attribute is used to store database URL for DBConnection method.

There functions in this module are shown below:

- DBConnection(): This public static method return the Connection object to the only database in this design.
- SelectDBDate(query: String): This public method uses the input query to obtain related data from database, and store all data into a ResultSet as a return.
- InsertDBDate(query: String): This public method inserts specific information into database by implementing input query.
- deleteDBDate(query: String): This public method deletes specific information from database by implementing input query.

6.1.16 CustomerDBController Detail

The CustomerDBController is only responsible for getting a new CustomerDAO object and return to required module. This controller is the bridge between DAO module and other controller module.

The class overview is shown in following figure.

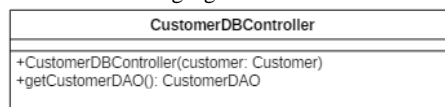


Figure 35 CustomerDBController class

There functions in this module are shown below:

- getCustomerDAO(): This method will return a new CustomerDAO object reference.

6.1.17 ReservationDBController Detail

The ReservationDBController is only responsible for getting a new ReservationDAO object and return to required module. This controller is the bridge between DAO module and other controller module.

The class overview is shown in following figure.



Figure 36 ReservationDBController class

There functions in this module are shown below:

- `getReservationDAO()`: This method will return a new `ReservationDAO` object reference.

6.1.18 FlightDBController Detail

The `FlightDBController` is only responsible for getting a new `FlightDAO` object and return to required module. This controller is the bridge between DAO module and other controller module.

The class overview is shown in following figure.

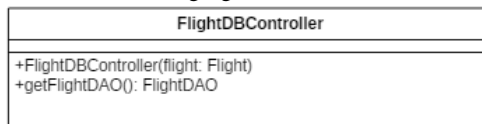


Figure 37 FlightDBController class

There functions in this module are shown below:

- `getFlightDAO()`: This method will return a new `FlightDAO` object reference.

6.1.19 EmployeeDBController Detail

The `EmployeeDBController` is only responsible for getting a new `EmployeeDAO` object and return to required module. This controller is the bridge between DAO module and other controller module.

The class overview is shown in following figure.

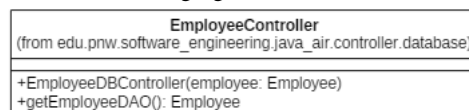


Figure 38 EmployeeDBController class

There functions in this module are shown below:

- `getEmployeeDAO()`: This method will return a new `EmployeeDAO` object reference.

6.1.20 CustomerLandingController Detail

This `CustomerLandingController` is used to check user input for login system, and support the customer login panel. If the customer exits, all customer information will be loaded and

switch current panel to account panel, or error window will be thrown and keep the customer login panel visible for next login attempt.

The class overview is shown in following figure.



Figure 39 CustomerLandingController class

There functions in this module are shown below:

- CustomerLandingController (): This constructor will get customer login panel reference from PanelMap module at first. Secondly, to call the getter method in the customer login panel to obtain user input of customer ID and password, then save these basic information into a new Customer object. The getCustomerDAO method in the CustomerDAOController will be called to get a new CustomerDAO object. The Customer object is used as input of searchCustomer method to validate whether there is corresponding customer. If the check is passed, the current window will point the currentCustomer reference in CustomerDTO module to this customer object.

6.1.21 CustomerPasswordResetController Detail

This CustomerPasswordResetController is used to reset customer password. The customer information will be obtained from the input of password reset panel and corresponding database information will be got by implementing CustomerDAOController Module, then this controller will check the accuracy of customer information. The password cannot be reset except that the customer input information is correct.

The class overview is shown in following figure.

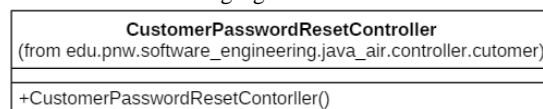


Figure 40 CustomerPasswordResetController class

There functions in this module are shown below:

- CustomerPasswordResetController (): This constructor will get password reset panel reference from PanelMap module at first. Secondly, to call the getter method in the customer login panel to obtain user input, then save these basic information into a new Customer object. The getCustomerDAO method in the CustomerDAOController will be called to get a new CustomerDAO object. The Customer object is used as input of updateCustomer method to update this customer information in the database customer table.

6.1.22 FlightStatusController Detail

This FlightStatusController is used to control FlightStatus panel to display all active flight status and switch from home panel to flight status panel. The FlightDAOController will be called to obtain all flights information.

The class overview is shown in following figure.

FlightStatusController (from edu.pnw.software_engineering.java_air.controller.flight)
-flightList: ArrayList<IFlight>
+FlightStatusController() +getFlights(): ArrayList<IFlight>

Figure 41 FlightStatusController class

The attributes in this module are shown below:

- flightList: This private attribute store the reference of Flight object ArrayList.

There functions in this module are shown below:

- FlightStatusController (): This method will call FlightDAOController to obtain a new FlightDAO object. Secondly, the selectFlight method will be applied to get all flight information in a Flight type ArrayList. With this list, the flight status panel is set to display all flight information. In detail, the iterator loop will be applied, and every Flight has a corresponding Flight information panel.
- getFlights(); This method will return the reference of Flight object ArrayList.

6.1.23 FlightSearchController Detail

This FlightSearchController is used to search user input flight and control FlightResult panel to show search results. The related flight information from Database will be obtained and display on the FlightResult panel.

The class overview is shown in following figure.

FlightSearchController (from edu.pnw.software_engineering.java_air.controller.flight)
-bookTravelPanel: BookTravelPanel -departFlights: ArrayList<IFlight> -returnFlights: ArrayList<IFlight>
+flightSearchController(bookTravelPanel: BookTravelPanel): void +getDepartureFlights(): ArrayList<IFlight> +getReturnFlights(): ArrayList<IFlight>

Figure 42 FlightSearchController class

The attributes in this module are shown below:

- bookTravelPanel: This private attribute store the reference of BookTravelPanel.
- departFlights: This private attribute store the reference of departure flights.
- returnFlights: This private attribute store the reference of return flights, in round trip reservation, this attribute is null.

There functions in this module are shown below:

- **FlightSearchController ()**: This method require a BookTravelPanel reference input. This reference is stored into local variable bookTravelPanel. The getter methods in the BookTravelPanel module are used to get flight information, including departure city, arrive city, departure date, arrive date and travel type. Secondly, the selectFlight method will be applied to get all flight information in a Flight type ArrayList. This list is stored into local variable departFlights. If the travel type is round trip, the departure city and arrive city will be exchanged to search return flights and store into returnFlights variable.
- **getDepartureFlights()**: This method will return departureFlights reference. The flight result panel will call this method to get search results.
- **getReturnFlights()**: This method will return returnFlights reference. The flight result panel will call this method to get search results, if the travel is round trip.

6.1.24 FlightSelectController Detail

This entity is used to select Flight and switch panel. The first step of this controller is to check whether this trip is one way. If the trip is one way, the select flight will be stored into current reservation as departure flight and switch from FlightResult panel to ReservationPassengerPanel. If the trip is round trip and the departure flight has not been selected, the select flight will be stored into current reservation as departure flight and control the FlightResult panel to show return flights. If the trip is round trip and the departure flight has been selected, the select flight will be stored into current reservation as return flight and switch from FlightResult panel to ReservationPassengerPanel.

The class overview is shown in following figure.

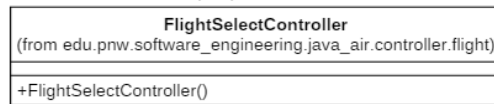


Figure 43 FlightSelectController class

There functions in this module are shown below:

- **FlightSelectController ()**: The first step of this constructor is to check whether this trip is one way. If the trip is one way, the select flight will be stored into current reservation as departure flight and switch from FlightResult panel to ReservationPassengerPanel. If the trip is round trip and the departure flight has not been selected, the select flight will be stored into current reservation as departure flight and control the FlightResult panel to show return flights. If the trip is round trip and the departure flight has been selected, the select flight will be stored into current reservation as return flight and switch from FlightResult panel to ReservationPassengerPanel..

6.1.25 PanelSwitch Detail

This PanelSwitch controller is used to control current content window to display specific panel. The PanelMap module is applied in this module. The string name of specific panel is required as input for setting method.

The class overview is shown in following figure.

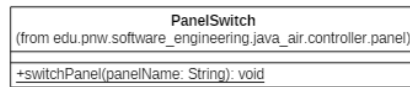


Figure 44 PanelSwitch class

There functions in this module are shown below:

- **PanelSwitch ()**: This public static method require a panel string name. With this input, panel reference will be obtained from PanelMap, then the current window is controlled to display this specific panel.

6.1.26 ReservPassengerContinueController Detail

This ReservPassengerContinueController is used to check all user input on ReservationPassenger panel, store available passenger information to current reservation and switch to ReservationBillinformation panel. If the input does not pass the check process, an error window will be thrown and keep current panel visible.

The class overview is shown in following figure.

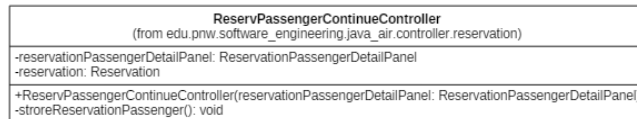


Figure 45 ReservPassengerContinueController class

There functions in this module are shown below:

- **ReservPassengerContinueController ()**: This constructor will check all user input on ReservationPassenger panel, store available passenger information to current reservation and switch to ReservationBillinformation panel. If the input does not pass the check process, an error window will be thrown and keep current panel visible.

6.1.27 ReservBillInfoContinueController Detail

This ReservBillInfoContinueController is used to check user input on ResercationBillinformation panel, store available bill information to current reservation and switch to ReservationConfirm panel. If the input does not pass the check process, an error window will be thrown and keep current panel visible.

The class overview is shown in following figure.



Figure 46 ReservBillInfoContinueController class

There functions in this module are shown below:

- **ReservBillInfoContinueController ()**: This constructor will check user input on ResercationBillinformation panel, store available bill information to current reservation and switch to ReservationConfirm panel. If the input does not pass the check process, an error window will be thrown and keep current panel visible.

6.1.28 ReservConfirmController Detail

This ReservConfirmController is used to save current reservation into database, if user click the confirm button on reservation confirmation panel. The ReservationDAOController would be used in this module.

The class overview is shown in following figure.

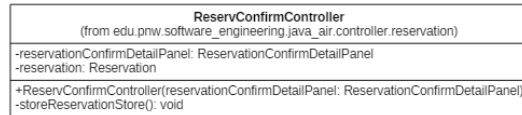


Figure 47 ReservConfirmController class

There functions in this module are shown below:

- **ReservConfirmController ()**: This constructor will save current reservation into database, if user click the confirm button on reservation confirmation panel. The ReservationDAOController would be used in this module.

6.2 Data detailed design

6.2.1 Customer Detail

Customer is a module for holding the data of all customer and transferring data to database module. The purpose of this module to be generated is to store the related customer information from database and allocate all customer interface applicants.

The class overview is shown in following figure.

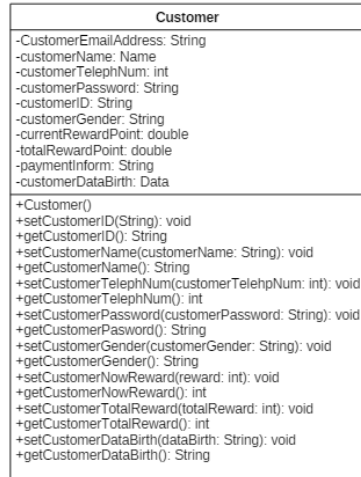


Figure 48 Customer class

The attributes in this module are shown below:

- **customerEmailAddress:** This attribute holds the customer email address in string type. For unregister customer, this attribute will be set as null.
- **customerFirstName:** This attribute holds the customer first name in string type. The name information for unregister customer will be obtained from reservation step.
- **customerMidName:** This attribute holds the customer mid name in string type. The name information for unregister customer will be obtained from reservation step.
- **customerLastName:** This attribute holds the customer last name in string type. The name information for unregister customer will be obtained from reservation step.
- **customerTelephNum:** This attribute holds the customer telephone number in string type. The telephone number information for unregister customer will be obtained from reservation step.
- **customerPassword:** This attribute holds the customer password in string type. The password for unregister customer will be a default string.
- **customerID:** This attribute holds the customer ID number in string type, which is specific for every customer.
- **customerDataBirth:** This attribute holds the customer date birth in date type. The name information for unregister customer will be obtained from Flight step.
- **customerGender:** This attribute holds the customer gender in string type. The gender information for unregister customer will be obtained from Flight step.
- **currentRewardPoint:** This attribute holds the customer current reward point in integer type.
- **totalRewardPoint:** This attribute holds the customer total reward point on history in integer type.

- paymentInform: This attribute holds the customer payment information, including the payment method, card number, expire data, card holder name and security code in string type.

There functions in this module are shown below:

- customerAccount(): This is constructor for this module.
- setCustomerID(in CustomerID:String): The customer ID will be set in CustomerAccount Object by this function.
- getCustomerID(): This function will return the customer account ID in string type.
- setCustomerName(in CustomerName:Name): The customer name will be set in CustomerAccount Object by this function.
- getCustomerName(): This function will return the customer name in string type.
- setCustomerTelephNum(in CustomerTelehpNum:int): The customer telephone number will be set in CustomerAccount Object by this function.
- getCustomerTelephNum(): This function will return the customer telephone number in string type.
- setCustomerPassword(in CustomerPassword:String): The customer password will be set in CustomerAccount Object by this function.
- getCustomerPassword(): This function will return the customer password in string type.
- setCustomerGender(in CustomerGender:String): The customer gender will be set in CustomerAccount Object by this function.
- getCustomerGender(): This function will return the customer gender in string type.
- setCustomerNowReward(in reward:int): The customer current rewards will be added or be decreased in CustomerAccount Object by this function.
- getCustomerNowReward(): This function will return the customer current reward in string type.
- setCustomerTotalReward(in totalReward:int): The customer total rewards will be added in CustomerAccount Object by this function.
- getCustomerTotalReward: This function will return the customer total reward in string type.
- setCustomerDateBirth(in dataBirth:Date): The customer data birth will be set in CustomerAccount Object by this function.
- getCustomerDateBirth(): This function will return the customer data birth reward in string type.

6.2.2 Flight Detail

Flight is a module for holding the flight information from database and transferring data to Reservation module. The purpose of this module to support the search flight interface applicants.

The class overview is shown in following figure.

Flight
-flightNum: String -originAirport: String -destinationAirport: String -transferAirport: String -scheduledDepartureDate: Date -scheduledArrivalDate: Date -scheduledDepartureTime: String -scheduledArrivalTime: String -ActualDepartureDate: Date -ActualArrivalDate: Date -ActualDepartureTime: String -ActualArrivalTime: String -prepareTime: Long -aircraftType: AircraftType -tripMileage: int -maintenanceTimeRequire: String -dateFormat: SimpleDateFormat -departurePrintDate: String -flightMaintenanceStatusList: ArrayList<FlightMaintenanceStatus>
+Flight(flightNum: String, originAirport: String, destinationAirport: String, scheduledDeparture: String, scheduledArrival: String) +setFlightNum(flightNum: String): void +getFlightNum(): String +setOriginAirport(originAirport: String): void +getOriginAirport(): String +setTransferAirport(transferAirport: String): void +getTransferAirport(): String +setDestinationAirport(destinationAirport: String): void +getDestinationAirport(): String +setScheduledDepartureDate(scheduledDepartureData: Date): void +getScheduledDepartureDate(): String +setScheduledArrivalDate(scheduledArrivalData: Date): void +getScheduledArrivalDate(): String +setScheduledDepartureTime(scheduledDepartureTime: String): void +getScheduledDepartureTime(): String +setScheduledArrivalTime(scheduledArrivalTime: String): void +getScheduledArrivalTime(): String +setActualDepartureDate(actualDepartureDate: Date): void +getActualDepartureDate(): String +setActualArrivalDate(actualArrivalDate: Date): void +getActualArrivalDate(): String +setActualDepartureTime(actualDepartureTime: String): void +getActualDepartureTime(): String +setActualArrivalTime(actualArrivalTime: String): void +getActualArrivalTime(): String +setAircraftType(aircraftType: String): void +getAircraftType(): String +setTripMileage(tripMileage: String): void +getAircraftType(): String +setMaintenanceTimeRequire(maintenanceTimeRequire: Long): void +getMaintenanceTimeRequire(): String +setMaintenList(flightMaintenanceStatusList: ArrayList<FlightMaintenanceStatus>): void +getMaintenList(): ArrayList<FlightMaintenanceStatus>

Figure 49 Flight class

The attributes in this module are shown below:

- flightNum: This private attribute holds the origin airport name in string type.
- originAirport: This private attribute holds the origin airport name in string type.
- destinationAirport: This private attribute holds the destination airport name in string type.
- transferAirport: This private attribute holds the transfer airport name in string type.
- scheduledDepartureDate: This private attribute holds the scheduled departure date in Date type.
- scheduledArrivalDate: This private attribute holds the scheduled arrival date in Date type.
- scheduledDepartureTime: This private attribute holds the scheduled departure time.
- scheduledArrivalTime: This private attribute holds the scheduled arrival time.

- actualDepartureData: This private attribute holds the actual departure date in Date type.
- actualArrivalData: This private attribute holds the actual arrival date in Date type.
- actualDepartureTime: This private attribute holds the actual departure time.
- actualArrivalTime: This private attribute holds the actual arrival time.
- prepareTime: This private attribute holds the required preparation time for departure.
- aircraftType: This private attribute holds the aircraft type of Flight.
- tripMileage: This private attribute holds the trip mileage in integer type.
- dateFormat: This private attribute holds the date format in SimpleDateFormat type.
- departurePrintDate: This private attribute holds the departure date in specific string type.

There functions in this module are shown below:

- Flight(): This is constructor for this module.
- setFlightNum(in flightNum:String): The flight number will be set in Flight object by this function.
- getFlightNum (): This function will return the flight number in string type.
- setOriginAirport(in originAirport:String): The origin airport name will be set in Flight object by this function.
- getOriginAirport (): This function will return the origin airport name in string type.
- setTransferAirport(in transferAirport:String): The transfer airport name will be set in Flight object by this function.
- getTransferAirport (): This function will return the transfer airport name in string type.
- setDestinationAirport(in destinationAirport:String): The destination airport name will be set in Flight object by this function.
- getDestinationAirport (): This function will return the destination airport name in string type.
- setScheduledDepartureDate(in scheduledDepartureData:Date): The scheduled departure data will be set in Flight object by this function.
- getScheduledDepartureDate(): This function will return the scheduled departure data in string type.
- setScheduledArrivalDate(in scheduledArrivalDate:String): The scheduled arrival data will be set in Flight object by this function.
- getScheduledArrivalDate(): This function will return the scheduled arrival data in string type.
- setScheduledDepartureTime(in scheduledDepartureTime:String): The scheduled departure data will be set in Flight object by this function.
- getScheduledDepartureTime (): This function will return the scheduled departure data in string type
- setScheduledArrivalTime (in scheduledArrivalTime:String): The scheduled arrival time will be set in Flight object by this function.

- `getScheduledArrivalTime ()`: This function will return the scheduled arrival time in string type.
- `setActualDepartureDate(in actualDepartureDate:String)`: The actual departure data will be set in Flight object by this function.
- `getActualDepartureDate()`: This function will return the actual departure data in string type.
- `setActualArrivalData(in actualArrivalData:String)`: The actual arrival data will be set in Flight object by this function.
- `getActualArrivalData()`: This function will return the actual arrival data in string type.
- `setActualDepartureTime(in actualDepartureTime:String)`: The actual departure data will be set in Flight object by this function.
- `getActualDepartureTime ()`: This function will return the actual departure data in string type
- `setActualArrivalTime (in actualArrivalTime:String)`: The actual arrival time will be set in Flight object by this function.
- `getActualArrivalTime ()`: This function will return the actual arrival time in string type.
- `setPrepareTime (in prepareTime:String)`: The preparation time for departure will be increased or decreased in Flight object by this function.
- `getPrepareTime()`: This function will return the preparation time for departure in string type.
- `setAircraftType (in aircraftType:String)`: The aircraft type will be set in Flight object by this function.
- `getAircraftType ()`: This function will return the aircraft type in string type.
- `setTripMileage (in tripMileage:String)`: The trip mileage will be set in Flight object by this function.
- `getAircraftType ()`: This function will return the trip mileage in string type.

6.2.3 Reservation Detail

Reservation is a module for holding the Reservation information from database and transferring data to user interface module. The purpose of this module to support the Reservation applicants.

The class overview is shown in following figure.



Figure 50 Reservation class

The attributes in this module are shown below:

- customerAccount: This attribute holds the Reservation account reference.
- priceCalculator: This attribute holds the PriceCalculator object reference.
- paymentInform: This attribute holds the payment information.
- billAddress: This attribute holds the bill address information.
- travelType: This attribute holds the travel type.
- ReservationNum: This attribute holds the Reservation number.
- passengerList: This attribute holds the passenger object list.
- primaryPassenger: This attribute holds the primary passenger reference.
- departureReservation: This attribute holds the departure Reservation reference.
- returnReservation: This attribute holds the return Reservation reference.
- actualArrivalTime: This attribute holds the actual arrival time.

There functions in this module are shown below:

- Reservation(): This is constructor for this module.
- setReservationNum(in ReservationNum:String): The reservatioin number will be set in Reservation object by this function.
- getReservationNum (): This function will return the reservation number in string type.
- setPassenger(in passengerList:Passenger): The passengers will be set in Reservation object by this function.
- getPassenger(): This function will return passenger information in this reservation.

- `setBillAddress(in billAddress:Address)`: The bill address will be set in Reservation object by this function.
- `getBillAddress()`: This function will return bill address in this reservation.
- `setPrice(in price:double)`: The price will be set in Reservation object by this function.
- `getPrice()`: This function will return price in this reservation.
- `setPayInfor(in paymentInfor:String)`: The payment information will be set in Reservation object by this function.
- `getPaymentInfor()`: This function will return payment information in this reservation.
- `setPrimaryPassenger(in passenger:Passenger)`: The primary passenger will be set in Reservation object by this function.
- `getPrimaryPassenger()`: This function will primary passenger in this reservation.
- `setDepartureFlight(in flight: Flight)`: The departure flight will be set in Reservation object by this function.
- `getDepartureFlight ()`: This function will departure flight in this reservation.
- `setReturnFlight (in flight: Flight)`: The return flight will be set in Reservation object by this function.
- `getReturnFlight ()`: This function will return flight in this reservation.
- `setTravelType(in travelType:String)`: The travel type will be set in Reservation object by this function.
- `getTravelType()`: This function will travel type in this reservation.
- `setCustomerAccount(in customer:Customer)`: The customer account will be set in Reservation object by this function.
- `getCustomerAccount()`: This function will set customer account in this reservation.
- `setReservationNum(in ReservationNum:String)`: The reservation number will be set in Reservation object by this function.
- `getReservationNum()`: This function will reservation number in this reservation.
- `setPriceCalculator(priceCalculator: PriceCalculator)`: This function will set priceCalculator in this reservation.
- `getPriceCalculator()`: This function will return the priceCalculator reference.

6.2.4 Employee Detail

Employee is a module for holding the employee information from database and transferring data to user interface module. The purpose of this module to support the employee interface applicants.

The class overview is shown in following figure.

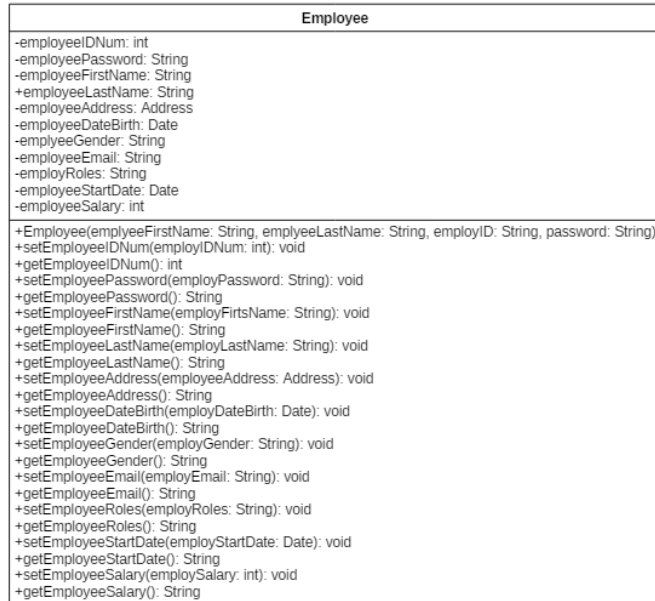


Figure 51 Employee class

The attributes in this module are shown below:

- employeeIDNum: This attribute holds the employee ID number.
- employeePassword: This attribute holds the employee password.
- employeeName: This attribute holds the employee name.
- employeeAddress: This attribute holds the employee address.
- employeeDateBirth: This attribute holds the employee date birth.
- employeeGender: This attribute holds the employee gender.
- employeeEmail: This attribute holds the employee email.
- employRoles: This attribute holds the employee role.
- employeeStartDate: This attribute holds the employee start date.
- employeeSalary: This attribute holds the employee salary.

There functions in this module are shown below:

- Employee(): This is the constructor.
- setEmployeeIDNum(in employIDNum:int): The employee ID number will be set in Employee object by this function.
- getEmployeeIDNum(): This function will return the employ ID number.
- setEmployeePassword(in employPassword:String): The employee password will be set in Employee object by this function.
- getEmployeePassword(): This function will return employee password.
- setEmployeeName(in employName:Name): The employee name will be set in Employee object by this function.

- `getEmployeeName()`: This function will return employee name.
- `setEmployeeAddress(in employeeAddress:Address)`: The employee address will be set in Employee object by this function.
- `getEmployeeAddress()`: This function will return employee address.
- `setEmployeeDateBirth(in employDateBirth:Date)`: The employee date birth will be set in Employee object by this function.
- `getEmployeeDateBirth()`: This function will return employee date birth.
- `setEmployeeGender(in employGender:String)`: The employee gender will be set in Employee object by this function.
- `getEmployeeGender()`: This function will return employee string.
- `setEmployeeEmail(in employEmail:String)`: The employee email will be set in Employee object by this function.
- `getEmployeeEmail()`: This function will return employee email.
- `setEmployeeRoles(in employRoles:String)`: The employee roles will be set in Employee object by this function.
- `getEmployeeRoles()`: This function will return employee role.
- `setEmployeeStartDate(in employStartDate:Date)`: The employee start date will be set in Employee object by this function.
- `getEmployeeStartDate()`: This function will return employee start date.
- `setEmployeeSalary(in employSalary:int)`: The employee salary will be set in Employee object by this function.
- `getEmployeeSalary()`: This function will return employee salary.

6.2.5 Passenger Detail

Passenger is a module for holding the passenger information from database and transferring data to user interface module. The purpose of this module to support the Flight module. The class overview is shown in following figure.

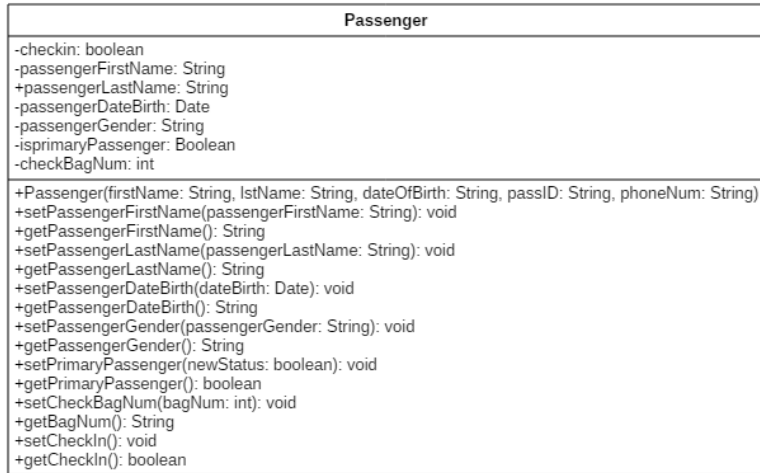


Figure 52 Passenger class

The attributes in this module are shown below:

- checkin: This attribute holds check in Boolean status.
- passengerName: This attribute holds passenger name.
- passengerDateBirth: This attribute holds passenger date birth.
- passengerGender: This attribute holds passenger passenger gender.
- primaryPassenger: This attribute holds passenger primary passenger.
- checkBagNum: This attribute holds passenger check bag number.

There functions in this module are shown below:

- Passenger(): This is the constructor.
- setPassengerName(in passengerName:Name): The passenger name will be set in Employee object by this function.
- getPassengerName():This function will return the passenger name.
- setPassengerDateBirth(dateBirth:Date): The passenger date birth will be set in Employee object by this function.
- getPassengerDateBirth():This function will return the passenger date birth.
- setPassengerGender(in passengerGender:String): The passenger gender will be set in Employee object by this function.
- getPassengerGender():This function will return the passenger gender.
- setPrimaryPassenger():The primary passenger will be set in Employee object by this function.
- getPrimaryPassenger():This function will return the primary passenger.

- `setCheckBagNum(in bagNum:int)`: The check bag number will be set in Employee object by this function.
- `getBagNum()`: This function will return the passenger bag number.
- `setCheckIn()`: The check in status will be set in Employee object by this function.
- `getCheckIn()`: This function will return the passenger check in status.