**Software Design Description**

**for**

**Java Air**

**Prepared by Rui Zhang**

**Team: Avian Limited**

# Introduction

## 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.

## 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.

## 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

RE = Requirement Engineer

SA = Software Architect

IE = Integration Engineer

TE = Testing Engineer

CD = Code Developer

# Reference

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

Software Configuration Management Plan (SCMP) for Java Air version 1.0

Software Design Description (SDD) for Java Air version 1.0

Software Project Management Plan (SPMP) for Java Air version 1.0

Software Quality Assurance Plan (SQAP) for Java Air version 1.0

Software User Documentation Plan (SUDP) for Java Air version 1.0

Software Test Document (STD) for Java Air version 1.0

Software Verification and Validation Plan (SVVP) for Java Air version 1.0

# Decomposition description

## Module decomposition

### CustomerDAO Entity

This entity is used for database operations, like searching, adding, removing or updating customer account. A detailed description will be given in section 6.

### ReservationDAO Entity

This entity is used for database operations, like searching, adding, removing or updating reservation information. A detailed description will be given in section 6.

### FlightDAO Entity

This entity is used for database operations, like searching, adding, removing or updating flight information. A detailed description will be given in section 6.

### MySQLReservationQueryBuilder Entity

This entity is used to generate the specific query for reservation database operation. A detailed description will be given in section 6.

### MySQLCustomerQueryBuilder Entity

This entity is used to generate the specific query for customer account database operation. A detailed description will be given in section 6.

### MySQLFlightQueryBuilder Entity

This entity is used to generate the specific query for flight database operation. A detailed description will be given in section 6.

### Aircraft Entity

This entity is used to hold aircraft information, there will be three different kinds of aircraft in Java Air Company. A detailed description will be given in section 6.

### FlightMaintenaceStatus 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.

### Name Entity

This entity is used to hold name information, and support passenger entity, employee entity, customer entity. A detailed description will be given in section 6.

### CalculatePrice Entity

This entity is used to calculate flight price. A detailed description will be given in section 6.

### GenerateCustomerNum Entity

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

### GenerateReservationNum

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

### GenerateFlightNum

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

## 3.2 Concurrent process decomposition

### 3.2.1 CustomerDBController Entity

This entity is used to create a reservation, based on the user input of passenger information and flight information. This entity can be considered one of the basic entity in this system.

### 3.2.2 ReservationDBController Entity

This entity is intended to store reservation information into database file.

### 3.2.3 FlightDBController Entity

This entity is used to store Customer information into database file.

## 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

## 4.2 Interprocess dependencies

## 4.3 Data dependencies

# 5. Interface description

## 5.1 Module interface

### 5.1.1 Java Air Home Interface

### 5.1.2 Customer Account Registration

### 5.1.3 Customer Account Management: Welcome

### 5.1.4 Customer Account Management: Update Personal Information

### 5.1.5 Customer Account Management: View Existing Reservations

### 5.1.6 Customer Account Management: View Rewards Program Information

### 5.1.7 Customer Account Management: Reset Password

### 5.1.8 Flight Reservation: Search Flights

### 5.1.9 Flight Reservation: Purchase Flights

## 5.2 Process interface

### 5.2.1 Customer account landing interface

# 6. Detailed design

## 6.1 Module detailed design

### 6.1.1 CustomerDAO Detail

CustomerDAO is a module for providing database operations. The purpose of this module to support the customer module.

There functions in this module are shown below:

* searchCustomer(in customer:Customer): This function will receive Customer type input with simple information, like customerID or customerName, then connect database to obtain completed customer information and set in Customer Object by calling buildQueryStrForSearchCustomer function in MySQLCustomerQueryBuilder module
* addCustomer(in customer:Customer): This function will receive completed Customer object, then connect database to set in this customer information and set in Customer Object by calling buildQueryStrForAddCustomer function in MySQLCustomerQueryBuilder module
* removeCustomer(in customer:Customer): This function will receive completed Customer object, then connect database to remove this customer information and set in Customer Object by calling buildQueryStrForRemoveCustomer function in MySQLCustomerQueryBuilder module
* updateCustomer(in customer:Customer): This function will receive completed Customer object, then connect database to update this customer information and set in Customer Object by calling buildQueryStrForUpdateCustomer function in MySQLCustomerQueryBuilder module
* connectionDB(): This function will connect to specific customer database table.

### 6.1.2 ReservationDAO Detail

ReservationDAO is a module for providing database operations. The purpose of this module to support the reservation module.

There functions in this module are shown below:

* searchReservation (in reservation: Reservation): This function will receive Reservation type input with simple information, like reservation number, then connect database to obtain completed reservation information and set in Reservation Object by calling buildQueryStrForSearchReservation function in MySQLReservationQueryBuilder module
* addCustomer(in reservation: Reservation): This function will receive completed Reservation object, then connect database to set in this reservation information and set in Reservation Object by calling buildQueryStrForAddReservation function in MySQLReservationQueryBuilder module
* removeReservation (in reservation: Reservation): This function will receive completed Reservation object, then connect database to remove this reservation information and set in Reservation Object by calling buildQueryStrForRemoveReservation function in MySQLReservationQueryBuilder module
* updateReservation (in reservation: Reservation): This function will receive completed Reservation object, then connect database to update this reservation information and set in Customer Object by calling buildQueryStrForUpdateReservation function in MySQLReservationQueryBuilder module
* connectionDB(): This function will connect to specific reservation database table.

### 6.1.3 FlightDAO Detail

FlightDAO is a module for providing database operations. The purpose of this module to support the flight module.

There functions in this module are shown below:

* searchFlight (in flight: Flight): This function will receive Flight type input with simple information, like flight number, then connect database to obtain completed Flight information and set in Flight Object by calling buildQueryStrForSearchFlight function in MySQLFlightQueryBuilder module
* addCustomer(in flight: Flight): This function will receive completed Flight object, then connect database to set in this flight information and set in Flight Object by calling buildQueryStrForAddFlight function in MySQLFlightQueryBuilder module
* removeFlight (in flight: Flight): This function will receive completed Flight object, then connect database to remove this Flight information and set in Flight Object by calling buildQueryStrForRemoveFlight function in MySQLFlightQueryBuilder module
* updateFlight (in flight: Flight): This function will receive completed Flight object, then connect database to update this flight information and set in Customer Object by calling buildQueryStrForUpdateFlight function in MySQLFlightQueryBuilder module
* connectionDB(): This function will connect to specific Flight database table.

### 6.1.4 MySQLCustomerQueryBuilder Detail

MySQLCustomerQueryBuilder is a module for generate query of customer database operations. The purpose of this module to support the CustomerDAO module.

There functions in this module are shown below:

* buildQueryStrForSearchCustomer(in customer:Customer): This function will return query for searching specific customer.
* buildQueryStrForAddCustomer(in customer:Customer): This function will return query for adding specific customer.
* buildQueryStrForRemoveCustomer(in customer:Customer): This function will return query for removing specific customer.
* buildQueryStrForUpdateCustomer(in customer:Customer): This function will return query for updating specific customer.

### 6.1.5 MySQLReservationQueryBuilder Detail

MySQLReservationQueryBuilder is a module for generate query of Reservation database operations. The purpose of this module to support the ReservationDAO module.

There functions in this module are shown below:

* buildQueryStrForSearchReservation(in reservation:Reservation): This function will return query for searching specific Reservation.
* buildQueryStrForAddReservation(in reservation:Reservation): This function will return query for adding specific Reservation.
* buildQueryStrForRemoveReservation(in reservation:Reservation): This function will return query for removing specific reservation.
* buildQueryStrForUpdateReservation(in reservation:Reservation): This function will return query for updating specific reservation.

### 6.1.6 MySQLFlightQueryBuilder Detail

MySQLFlightQueryBuilder is a module for generate query of Flight database operations. The purpose of this module to support the FlightDAO module.

There functions in this module are shown below:

* buildQueryStrForSearchFlight(in flight:Flight): This function will return query for searching specific flight.
* buildQueryStrForAddFlight(in flight:Flight): This function will return query for adding specific flight.
* buildQueryStrForRemoveFlight(in flight:Flight): This function will return query for removing specific flight.
* buildQueryStrForUpdateFlight(in flight:Flight): This function will return query for updating specific flight.

### 6.1.7 Aircraft Detail

Aircraft is a module for holding information of different aircraft type.

The attributes in this module are shown below:

* aircraftName: This attribute holds the aircraft name.
* aircraftCapacity: This attribute holds the aircraft capacity.
* aircraftRange: This attribute holds the aircraft range.
* aircraftSpeed: This attribute holds the aircraft speed.
* aircraftPrice: This attribute holds the aircraft price.

There functions in this module are shown below:

* getAircraftName(): This function will return aircraft name.
* getAircraftCapacity():This function will return aircraft capacity.
* getAircraftRange():This function will return aircraft range.
* getAircraftSpeed():This function will return aircraft speed.
* getAircraftPrice():This function will return aircraft price.

### 6.1.8 FlightMaintenaceStatus Detail

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

The attributes in this module are shown below:

* maintenceTimeRequire: This attribute holds the maintenance require time.
* maintenceStatus: 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

### 6.1.9 Name

Name is a module for holding name information. This module will support the Customer module, Passenger module and Employee module.

The attributes in this module are shown below:

* firstName: String: This attribute holds the first name.
* midName: String: This attribute holds the mid name.
* lastName: String: This attribute holds the last name.

There functions in this module are shown below:

* sperateName(in name:String): This function will separate the whole name string into first name, mid name and last name parts.
* mergeName(): This function will merge first name, mid name and last name parts into whole name string.
* setFirstName(in firstName:String): This function will set the first name.
* getFirstName(): This function will return the first name.
* setMidName(in firstName:String): This function will set the mid name.
* getMidName():This function will return the mid name.
* setLastName(in firstName:String): This function will set the last name.
* getLastName():This function will return the last name.

### 6.1.10 CalculatePrice

Calculate is a module for calculating price based on specific reservation. This module will support the Reservation module.

The attributes in this module are shown below:

* price: This attribute holds the price number.

There functions in this module are shown below:

* returnPrice: This function will return the price number.
* calculatePrice(in reseravation: Reservation): This function will calculate the reservation tickets price based on the input of reservation.

### 6.1.11 GenerateReservationNum

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

There functions in this module are shown below:

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

### 6.1.12 GenerateFlightNum

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

There functions in this module are shown below:

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

### 6.1.13 GenerateCustomerNum

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

There functions in this module are shown below:

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

## 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 attributes in this module are shown below:

* customerEmailAddress: This attribute holds the customer email address. For unregister customer, this attribute will be set as null.
* customerFirstName: This attribute holds the customer first name. The name information for unregister customer will be obtained from reservation step.
* customerMidName: This attribute holds the customer mid name. The name information for unregister customer will be obtained from reservation step.
* customerLastName: This attribute holds the customer last name. The name information for unregister customer will be obtained from reservation step.
* customerTelephNum: This attribute holds the customer telephone number. The telephone number information for unregister customer will be obtained from reservation step.
* customerPassword: This attribute holds the customer password. The password for unregister customer will be a default string.
* customerID: This attribute holds the customer ID number, which is specific for every customer.
* customerDataBirth: This attribute holds the customer data birth in string type, and the day, month and year are separated by comma. The name information for unregister customer will be obtained from Flight step.
* customerGender: This attribute holds the customer gender. The gender information for unregister customer will be obtained from Flight step.
* currentRewardPoint: This attribute holds the customer current reward point.
* totalRewardPoint: This attribute holds the customer total reward point on history, and this function could be used to identify whether a customer is new one.
* 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 attributes in this module are shown below:

* flightNum: This attribute holds the origin airport name.
* originAirport: This attribute holds the origin airport name.
* destinationAirport: This attribute holds the destination airport name.
* transferAirport: This attribute holds the transfer airport name.
* scheduledDepartureData: This attribute holds the scheduled departure data.
* scheduledArrivalData: This attribute holds the scheduled arrival data.
* scheduledDepartureTime: This attribute holds the scheduled departure time.
* scheduledArrivalTime: This attribute holds the scheduled arrival time.
* actualDepartureData: This attribute holds the actual departure data.
* actualArrivalData: This attribute holds the actual arrival data.
* actualDepartureTime: This attribute holds the actual departure time.
* actualArrivalTime: This attribute holds the actual arrival time.
* prepareTime: This attribute holds the required preparation time for departure.
* aircraftType: This attribute holds the aircraft type of Flight.
* tripMileage: This attribute holds the trip mileage.

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 attributes in this module are shown below:

* customerAccount: This attribute holds the Reservation account reference.
* price: This attribute holds the price number.
* 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 customer account in this reservation.
* setReservationNum(in ReservationNum:String): The reseravtion number will be set in Reservation object by this function.
* getReservationNum():This function will reservation number in this reservation.

### 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 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.
* emplyeeGender: 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 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.