DOKUZ EYLÜL UNIVERSITY
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER ENGINEERING

CME 2210
Object Oriented Analysis and Design

AIRLINE MANAGEMENT SYSTEM

by
Ecem Tunur 2019510076
Rana Gül 2019510040
Muzaffer Sevili 2019510069

CONTENT

CHAPTER ONE

INTRODUCTION

1.1 What the problem is

An airline needs a system where it can perform operations related to flights for both its employees and passengers. In accordance with this need, the Airplane Management system has been developed. This system can be accessed by branches and ticket transactions of customers can be performed. At the same time, the customer himself can log into the system and perform different operations. Flight attendants can check their flights. This program will allow access through the computer. This system can be entered in five different ways. System accessible by airline manager, flight workers, ticket office workers, maintenance workers and customers.

1.2 Goals for the Project

The purpose of this project is to enable the airline to perform the following operations with different actors.

Airline Manager:

Adding and removing branches

Adding and removing employees

Access to reports

Flight planning

Creating a campaign

Flight Workers:

Access to the timetable of flights

Access to the list of past flights

Passengers' information

Ticket Office Workers:

Ticket sales

Ticket control

Ticket cancellation

Ticket refund

Maintenance Worker:

List of required treatments

Reporting of maintenance performed

Customers:

Ticket inquiry

Buying tickets

Ticket cancellation request

Ticket refund request

1.3 Process Description

❖ The login screen appears to the user's connection.

❖ Login is performed by an employee or user.

❖ After the employee entry:

- According to the ID check, the Office worker or Airline Manager distinction is made.

❖ After entering the Airline Manager:

- It forms the top of the system hierarchically. All operations of adding, deleting are performed from this section.
- Add and delete operations: Employee, MemberShip, Plane, Flight

❖ After entering the office worker:

- It can perform ticket transactions after any customer request.
- The office employee has the right to perform operations such as ticket purchase, exchange, cancellation, inquiry and checkIn.

❖ After user login:

- He can request to become a member of the system. It can display the kampayas formed because of this.

❖ He has the authority to perform all ticket processing. Purchase, exchange, cancellation, inquiry and checkIn of tickets....

CHAPTER TWO

REQUIREMENTS

Airlines system A few factors that directs us to develop a new system are: Features that the airlines system needs and needs to be realized in this project:

- ❖ Faster System
- ❖ Accuracy
- ❖ Reliability
- ❖ Informative
- ❖ Reservations and cancellations from anywhere to any place

2.1 Possible Classes and Methods

2.1.1. Login Class : Allows the user to log in or register in the system. Checks the user from the database. If there are no users, the registration function works. Saves the last entry date to the database, if available. If the user is not registered in the system, “Do you want to register ?" if it is registered, the inscription "Login is successful" appears.

Attributes: username (String), password (String)

Methods: Getters & Setters, boolean isLogin(String username,String password), void signUp(), void logIn()

2.1.2 Ticket Class : Creates the ticket in the system.

Attributes: passengerName (String), ticketClass (String), passengerId (Int), ticketID (int), ticketType (char), airportInfo (String), destinationType (char), source (String), destination (String), price (double), paymentType (String),

Methods: Getters & Setters, void printTicket()

2.1.3. Flight Class : It contains flight details for use in other operations.

Attributes: flightID (int), planeId(int), workersList (String[]), passengersList (String[]), source (String), destination (String), flightTime (String), weatherForecast (String), flightGate (int), departureTime (String), arrivalTime (String)

Methods: Getters & Setters, String flightDetails()

2.1.4. Plane Class : It contains plane features for flights details.

Attributes: planeId (int), age (int), seats (Seat[]), type (String), isAvailable (boolean)

Methods: Getters & Setters, String planeDetails (), boolean isPlaneAvailable (), boolean Maintenance ()

2.1.5. Seat Class : It contains seats features for plane details.

Attributes: type (char), seatID (int), isAvailable (Boolean),

Methods: Getters & Setters

2.1.6. Passenger Class : It contains passenger features. All properties are displayed in display() methods.

Attributes: name (string), surname (string), age (int), gender(char), weatherForecast(String), passengerID (int), passengerPoint (int), member (String), tickets(Ticket[])

Methods: Getters & Setters, void display(), void getMembership(), void listTickets()

2.1.7. Employee Class : It contains employee features. All properties are displayed in display() methods. It extends four different class (airline manager, flight workers, ticket office workers, maintenance workers)

Attributes: employeeID(int), name (String), surname (String), age (int), gender (char), salary (int), bonus (int), field (String)

Methods: Getters & Setters, void display(), int calculateBonus (int ticketPrice),

2.1.8. Luggage Class : It contains luggage informations.

Attributes: ticketID (int), surname (String), weight (double), isTransit (Boolean)

Methods: Getters & Setters, void display(), double calculateExtraPrice(int weight)

2.1.9. TicketProcessing Class : It includes all ticket processing.

Attributes: ticket (Ticket)

Methods: Getters & Setters, void ticketExchange (Ticket previousTicket, Ticket newTicket), double ticketCancellation (Ticket ticket), void ticketInquiry(Ticket ticket), void buyingTicket(Passenger passenger, Flight flight, Seat seat), void checkIn(Ticket ticket)

2.1.10. AirlineManager Class : It includes all airline features.

Attributes: name (String), employeeList(Employee[]), memberShipList(Passenger[]), planeList(Plane[]), flightList(Flight[])

Methods: Getters & Setters,

- void addEmployee(String name, String surname, int age, char gender, int salary , int bonus , String field),
- void addMemberShip(),
- void addPlane(int age, Seats[] seats, String type, booelan isAvailable
- void addFlight(int planeId, FlightWorker workersList, Passenger[] passengersList, String source, String destination, Time flightTime, String weatherForecast, int flightGate,Time departureTime, Time arrivalTime,
- void deleteEmployee(int employeeId),
- void deleteMemberShip(int passengerId),
- void deletePlane(int planeId),
- void deleteFlight(int FlightId)

2.1.11 FlightWorkerClass :

Attributes: name (String), surname (String), age (int), gender (char), salary (int), bonus (int), field (String)

Methods: void printTimeTable(), void listPreviousFlights(), void printPassengersInfo()

2.1.12 MaintenanceWorker Class :

Attributes: name (String), surname (String), age (int), gender (char), salary (int), bonus (int), field (String)

Methods: void listReqTreatments(), void reportMaintenance()

2.1.13 TicketOfficeWorker Class : Office employees have the right to access all ticket processing.

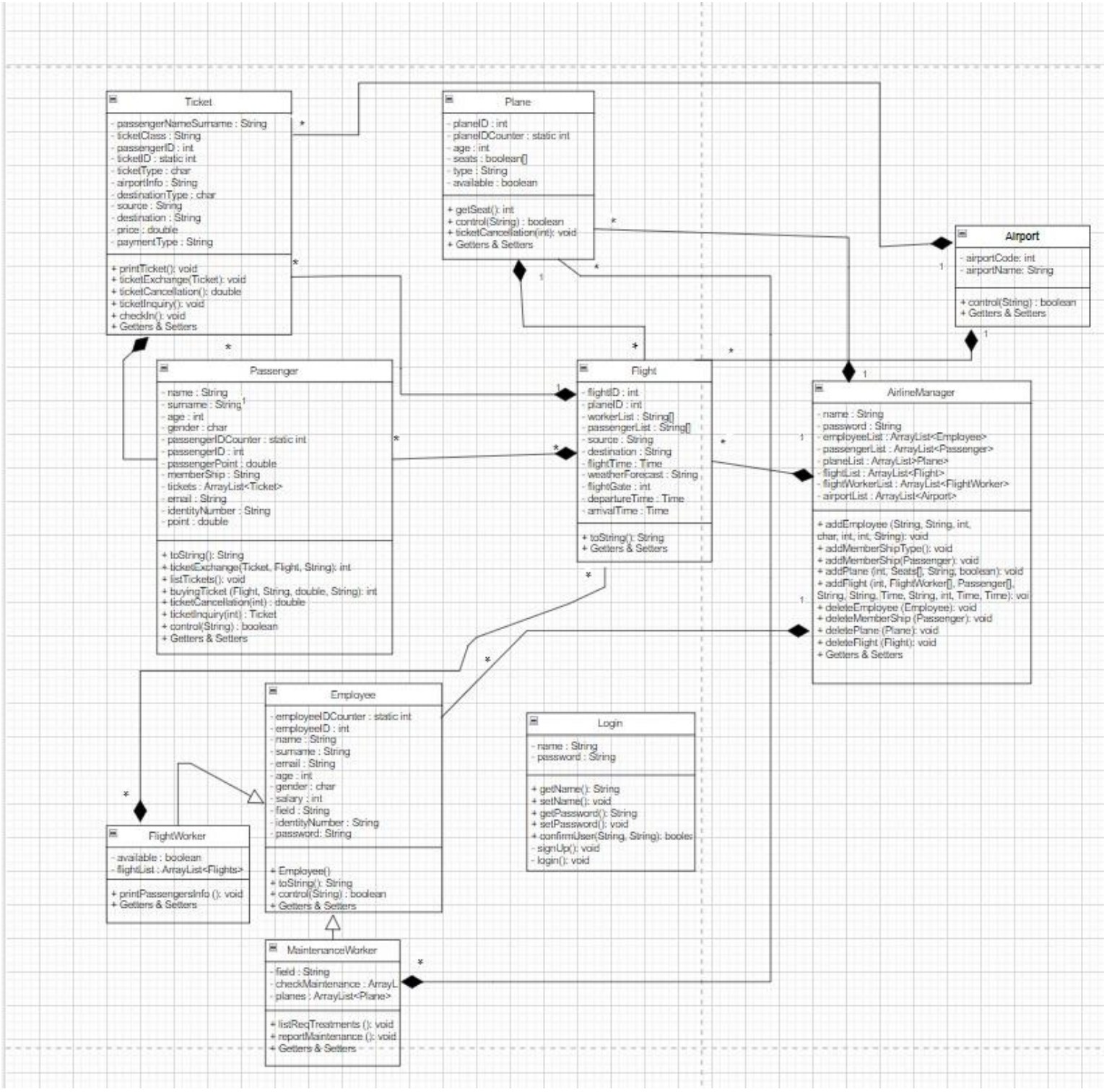
Attributes: name (String), surname (String), age (int), gender (char), salary (int), bonus (int), field (String)

Methods: boolean ticketControl(), void buyTicket(),void cancelTicket(), int ticketRefund()

CHAPTER THREE

UML DIAGRAMS

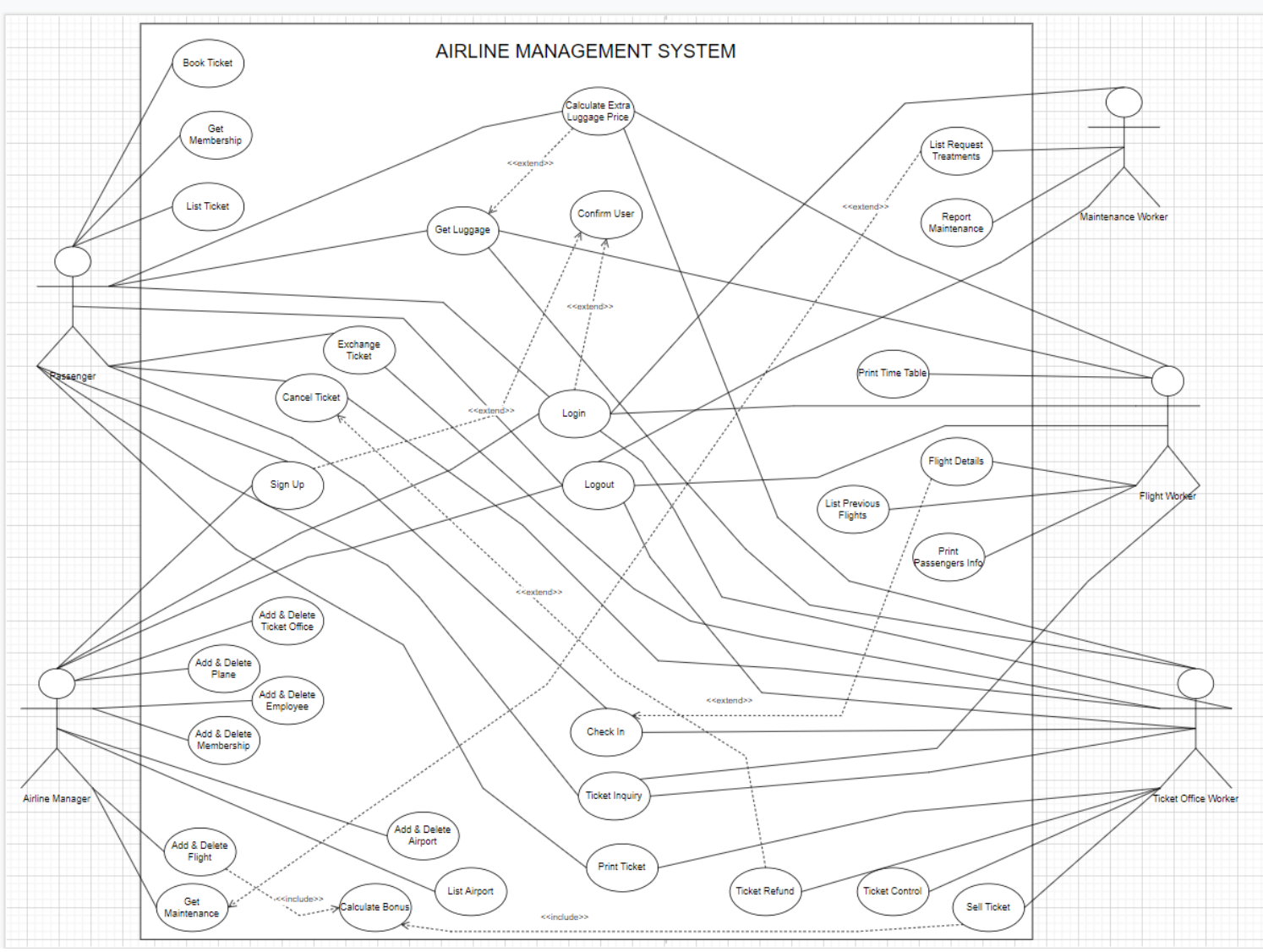
3.1 CLASS DIAGRAM



This diagram represents the classes in the program and the relationships between the classes, and the arrows represent inheritance, association, aggregation, composition relationships.

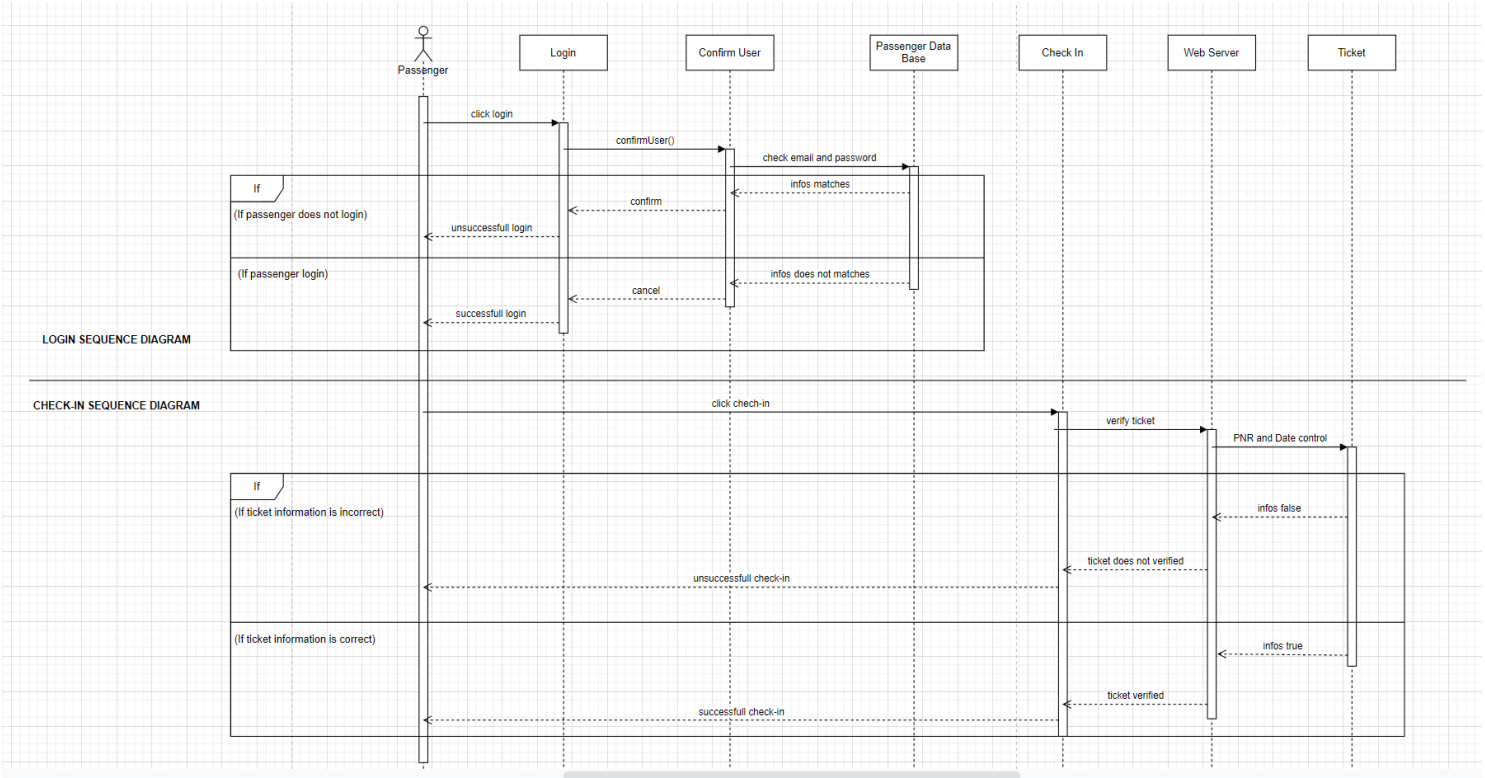
This diagram represents the correct class order and the arrows represent the inheritance, association, aggregation, composition relationships. For example, one passenger can have many luggages, but any lugggage can only have one passenger. Another example is that the FlightWorker, MaintenanceWorker, and TicketOfficeWorker classes inherit the Employee class.

3.2 USE CASE DIAGRAM



This diagram represents which actor can use which part of the program, and in which case it shows what action is taking place. For example, a passenger can buy a ticket, cancel it, or ask about his ticket. The status of canceling the ticket depends on the refund function with the extend tag.

3.3 SEQUENCE DIAGRAM



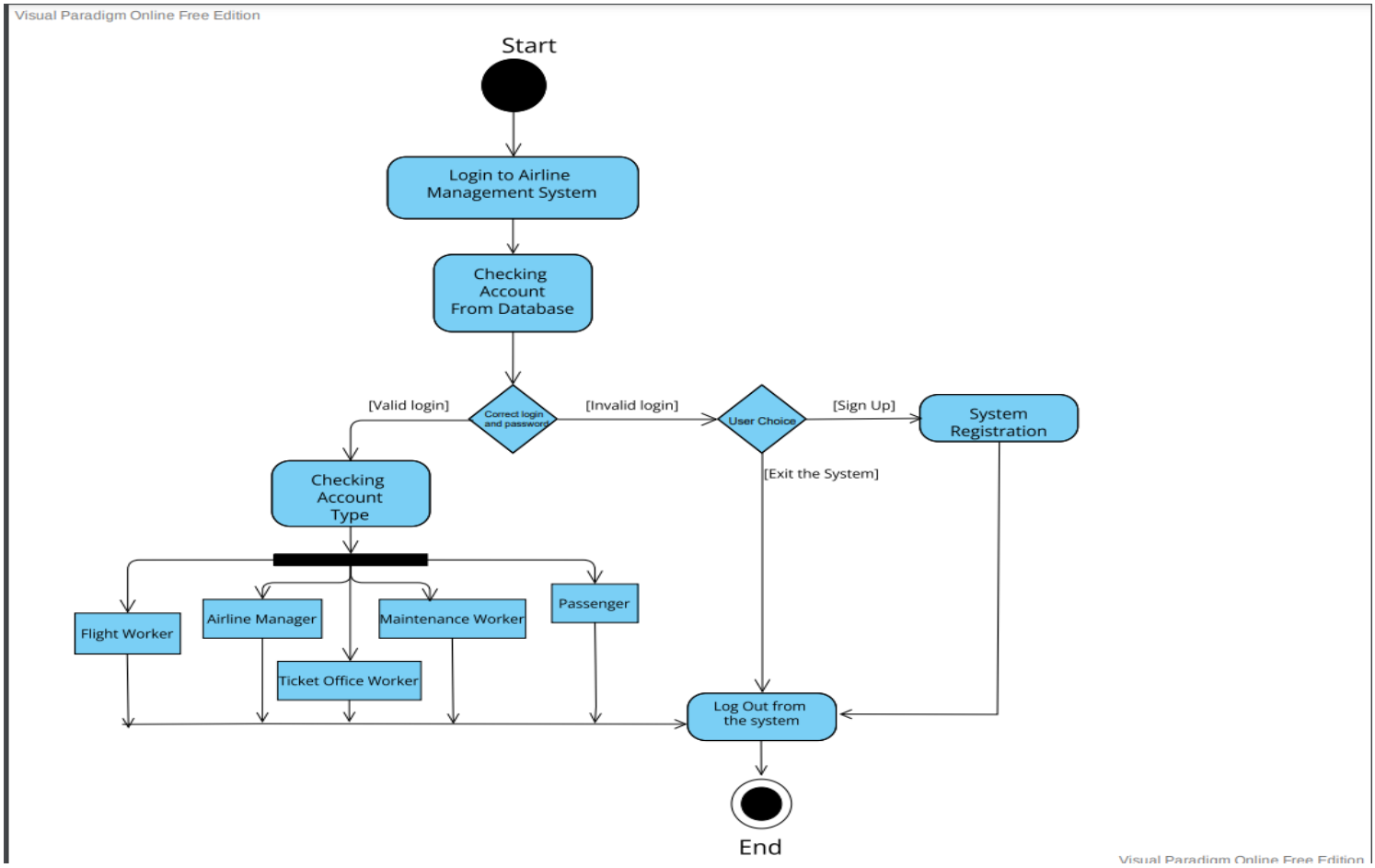
The sequence diagram represents the flow of a scenario in the program. In this diagram, we see the scenarios that may occur as a result of the user's login and the check-in request after login.

Login steps:
i. User presses login button
ii. User enters information
iii. It is checked whether the entered information is compatible with the information in the data base.

iv. If the password is correct, you will be logged in.
v. If the password is incorrect, the operation is unsuccessful

Afterwards, the **Check-in** scenario is seen.
i. User presses check-in button
ii. Ticket information is given
iii. It is checked whether the ticket information is correct and the date is suitable for check-in.
iv. If the ticket and date are correct, check-in is done.
v. If the information is incorrect, the operation is unsuccessful.

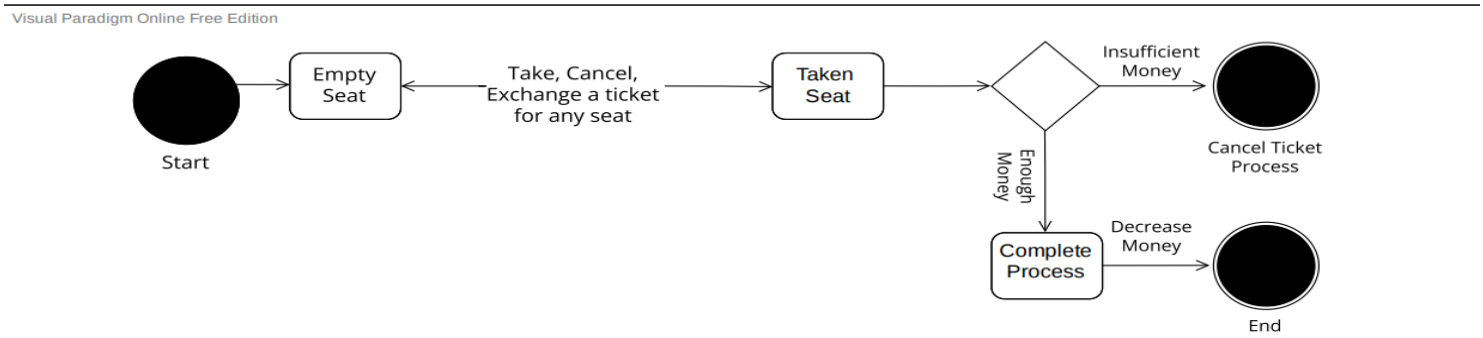
3.4 ACTIVITY DIAGRAM

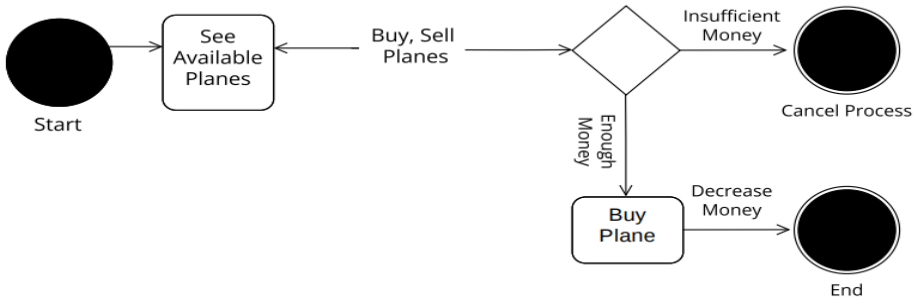


This activity diagram represents the login to the Airline Management System. The user enters his information into the system. User information is checked. If the user information is correct, the user login is successful and the user type is activated according to the informations. While the login is successful, the user performs some actions or can log out. If

the informations is incorrect, the user is given a choice. It is allowed to create a new record if the user wants, to log out if they don't want to.

3.5 STATE DIAGRAM





This state diagram stands for the steps of a class. For example, in the second example if the airline manager checks planes, and decides to buy new planes, then he/she first must check the money. If the money is enough to hire these planes then new planes will be added to the system and decrease the money, else system gives an output like “Our company does not have enough money for buying new planes!” and the money stays the same.

CHAPTER FOUR

IMPLEMENTATION

4.1.1 AirlineManager

Constructor contains ArrayLists used in the system.

The addEmployee method takes the name, surname, email, age, gender, salary, field and ID number and is used to add a new employee to the system by checking the ID number. According to the "field" parameter, the field in which the employee works is determined.

The deleteEmployee method takes the ID number and is used to delete an employee from the system by checking the ID number.

The addPassenger method takes the name, surname, email, age, gender, salary, field and ID number and is used to add a new passenger (customer) to the system by checking the ID number.

The addPlane method takes the age, number of seats and type of the aircraft as parameters and allows adding a new aircraft to the system.

The deletePlane method takes the id of the plane as a parameter and it is used to delete that plane from the system, if any.

The addFlight method takes the id of the plane, the number of seats, where and where it is traveling from, the departure and arrival time and the gate number of the plane, and adds the flight if it is not available in the system.

The deleteFlight method takes the flight ID as a parameter, and if there is this flight in the system, it cancels the tickets of the passengers and serves to delete the flight.

The addAirport method takes the airport name as a parameter and is used to add the airport if it is not present in the system.

The deleteAirport method takes the airport ID as a parameter and serves to delete the airport if it exists in the system.

The controlLoginPassenger method takes the passenger's name and password as a parameter and checks the accuracy of the information.

The controlWorker method takes the manager's name as a parameter and checks the accuracy of the information.

The getEmployee method takes the employee's name and password as a parameter and checks the accuracy of the information.

4.1.2 Airport

Constructor contains airport name and id.

The control() method controls the airport name.

4.1.3 Employee

Constructor contains informations of employee.

The toString() overrided method prints the information of employees.

4.1.4 Flight

Constructor contains informations of flights.

The toString() overrided method prints the information of flights.

4.1.5 FlightWorker

Constructor contains informations of flight workers.

The printPassengersInfo method takes flightID as a parameter and lists the passengers.

4.1.6 MaintenanceWorker

Constructor contains informations of maintenance workers.

The listReqTreatmens method add some problems to checkMaintenance ArrayList.

The reportMaintenance method returns to the checkMaintenance ArrayList of the specified aircraft and checks whether its maintenance is over.

4.1.7 Passenger

Constructor contains informations of passengers.

The toString() method prints the information of passengers.

The listTickets() method prints the ticket's informations.

The buyingticket() method takes flight, ticketClass, price and date parameters and create the new ticket. Passenger earns %5 point of the ticket price. If passenger have premium membership get %20 discount.

The ticketExchange() method takes ticket, flight and ticketClass parameters and changes the tickets.

The ticketCancellation() method takes the ticketID parameters, set empty the seat of selected ticketID, remove the ticket, gets the passenger's point back.

The Control() method takes the ID number parameter and controls the identity number of passengers.

4.1.8 Plane

Constructor contains informations of planes.

The Control() method takes the ID number parameter and controls the identity number of planes.

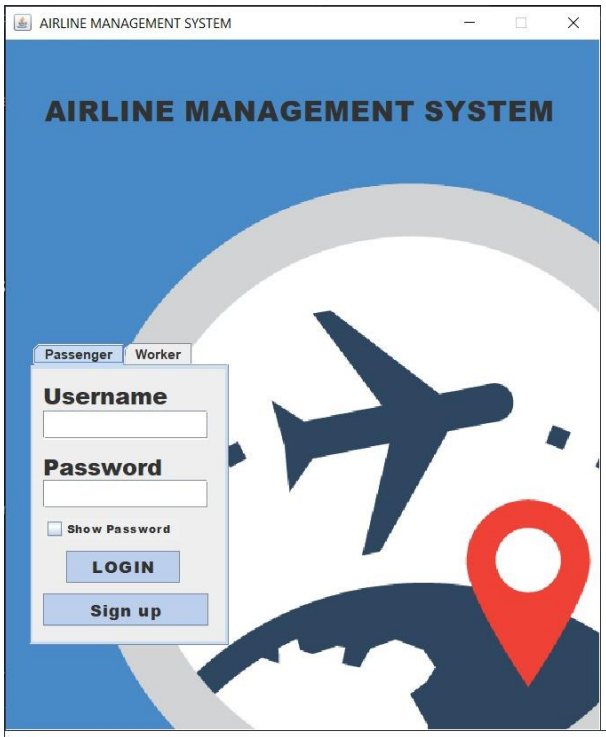
The GetSeat() method serves to change the status of the seat to occupied by giving an empty seat to the passenger.

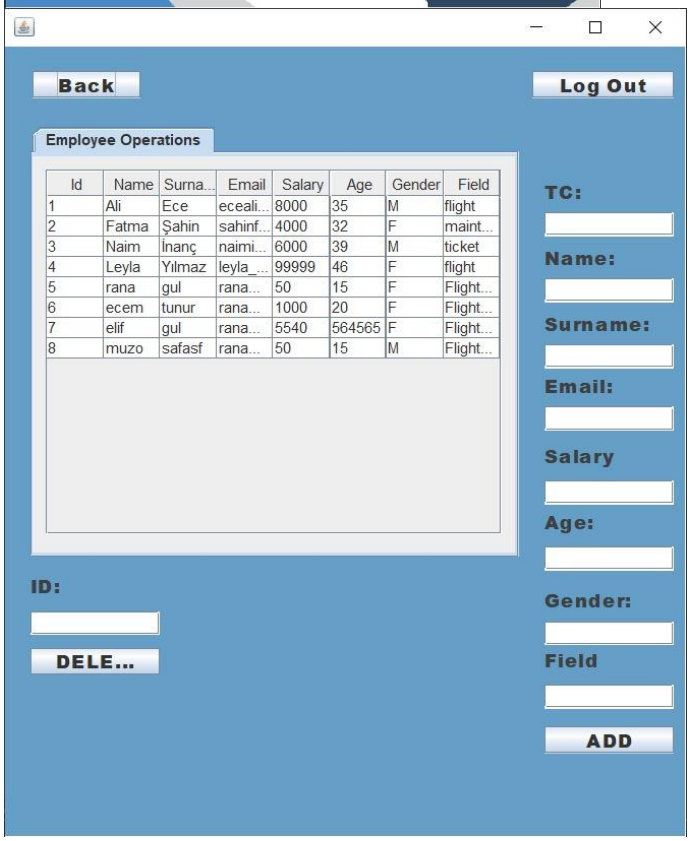
The ticketCancellation() method takes the seatID parameter and serves to change the state of the seat to empty.

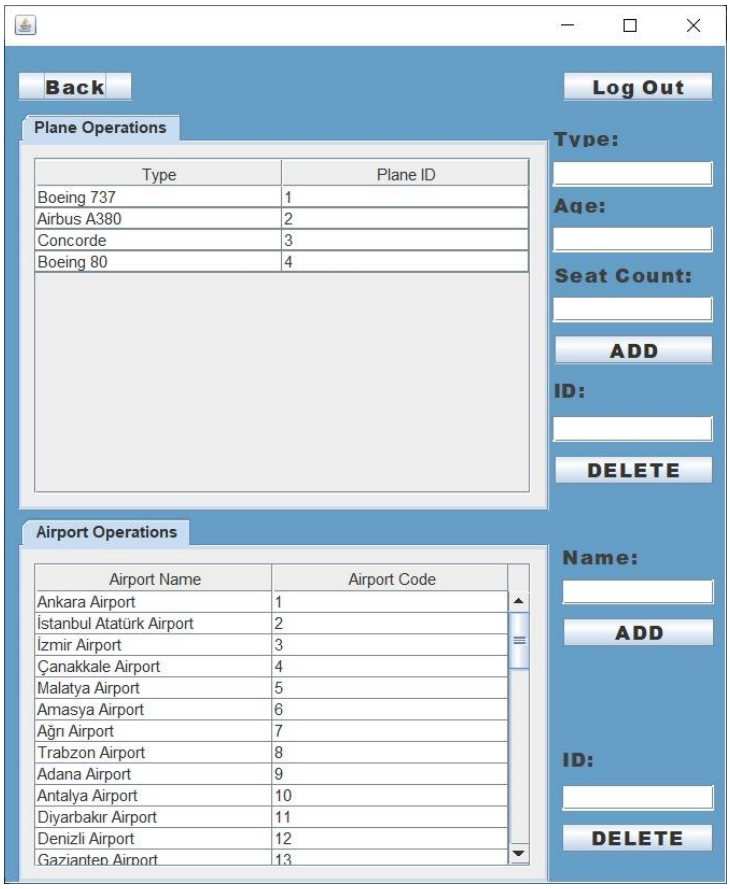
4.1.9 Ticket

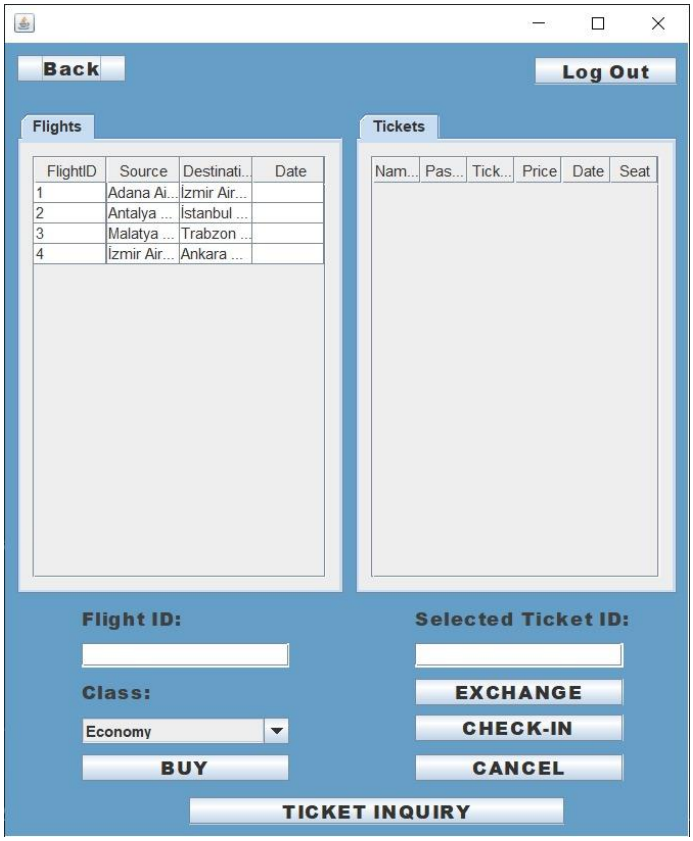
Constructor contains informations of tickets.

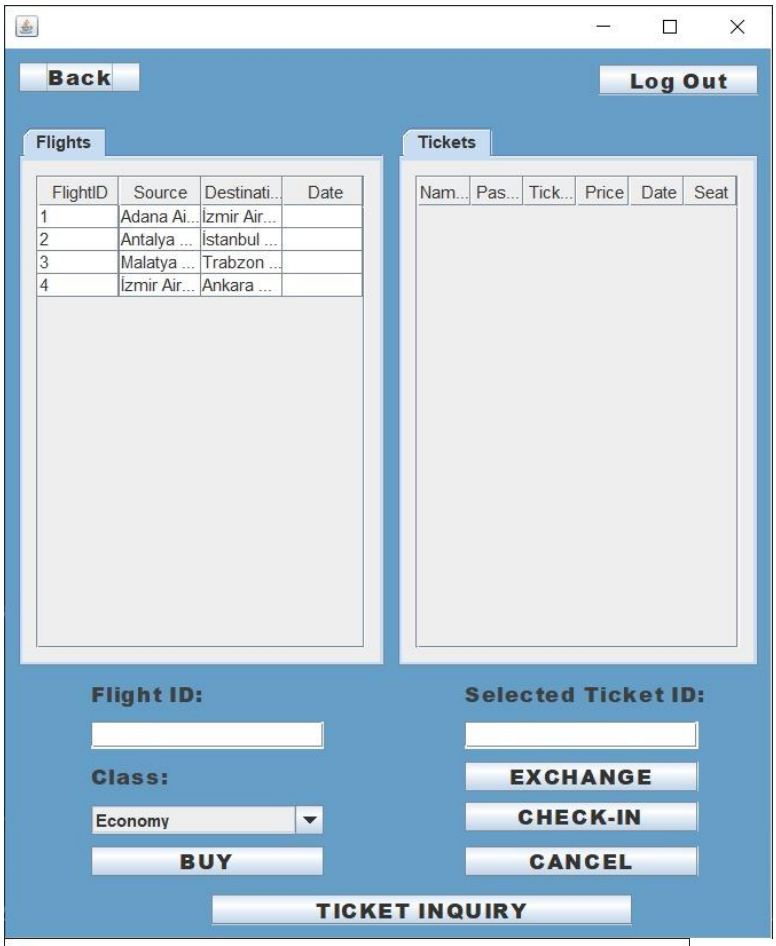
The toString() overrided method prints the information of tickets.

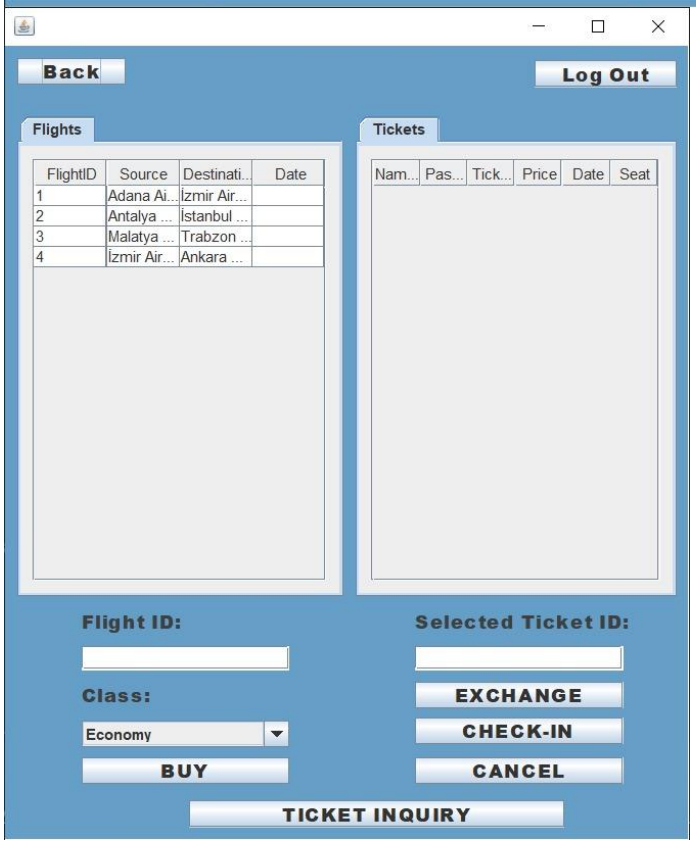

AIRLINE MANAGEMENT SYSTEM
AIRLINE MANAGEMENT SYSTEM
Passenger
Worker
Username
Password
Show Password
LOGIN
Sign up

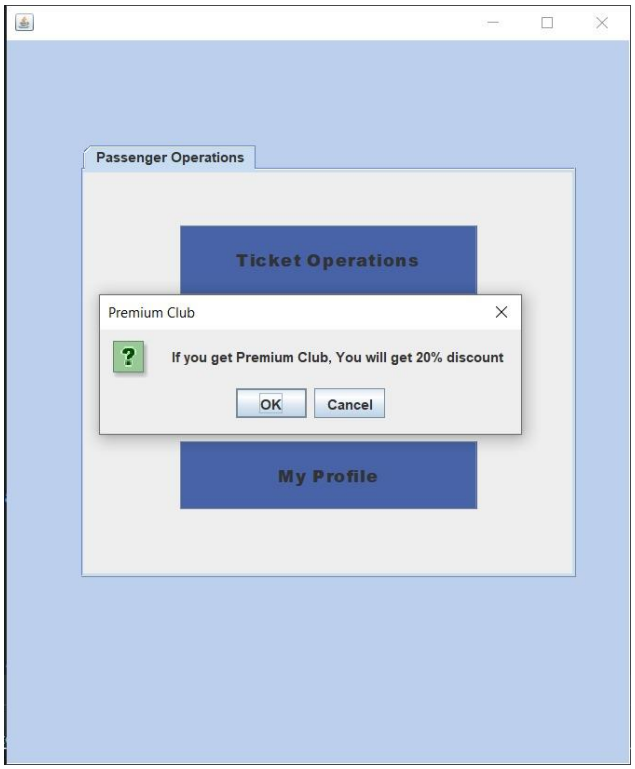

Back
Log Out
Employee Operations
Id Name Surna... Email Salary Age Gender Field
1 Ali Ece eceali... 8000 35 M flight
2 Fatma Şahin sahinf... 4000 32 F maint...
3 Naim İnanç naimi... 6000 39 M ticket
4 Leyla Yılmaz leyla_... 99999 46 F flight
5 rana gul rana... 50 15 F Flight...
6 ecem tunur rana... 1000 20 F Flight...
7 elif gul rana... 5540 564565 F Flight...
8 muzo safasf rana... 50 15 M Flight...
TC:
Name:
Surname:
Email:
Salary
Age:
Gender:
Field
ID:
DELE...
ADD

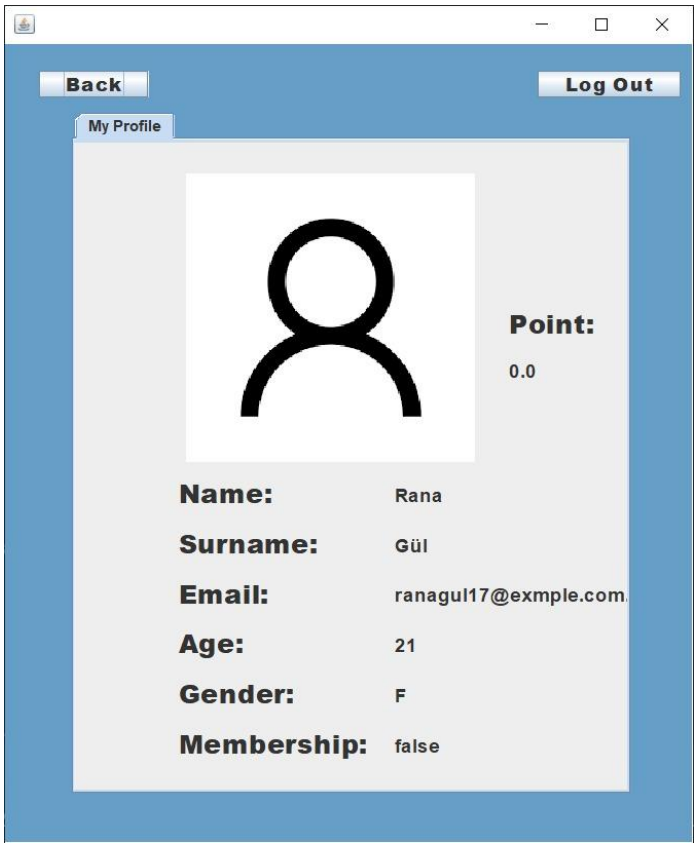

Back
Log Out
Plane Operations
Type
Plane ID
Boeing 737
1
Airbus A380
2
Concorde
3
Boeing 80
4
Type:
Age:
Seat Count:
ADD
ID:
DELETE
Airport Operations
Airport Name
Airport Code
Ankara Airport
1
İstanbul Atatürk Airport
2
İzmir Airport
3
Çanakkale Airport
4
Malatya Airport
5
Amasya Airport
6
Ağrı Airport
7
Trabzon Airport
8
Adana Airport
9
Antalya Airport
10
Diyarbakır Airport
11
Denizli Airport
12
Gaziantep Airport
13
Name:
ADD
ID:
DELETE


Back
Log Out
Flights
FlightID
Source
Destinati...
Date
1
Adana Ai...
İzmir Air...
2
Antalya ...
İstanbul ...
3
Malatya ...
Trabzon ...
4
İzmir Air...
Ankara ...
Tickets
Nam...
Pas...
Tick...
Price
Date
Seat
Flight ID:
Selected Ticket ID:
Class:
Economy
BUY
EXCHANGE
CHECK-IN
CANCEL
TICKET INQUIRY


Back
Log Out
Flights
Tickets
FlightID
Source
Destinati...
Date
1
Adana Ai...
İzmir Air...
2
Antalya ...
İstanbul ...
3
Malatya ...
Trabzon ...
4
İzmir Air...
Ankara ...
Nam...
Pas...
Tick...
Price
Date
Seat
Flight ID:
Selected Ticket ID:
Class:
Economy
EXCHANGE
CHECK-IN
BUY
CANCEL
TICKET INQUIRY


Back
Log Out
Flights
Tickets
FlightID
Source
Destinati...
Date
1
Adana Ai...
İzmir Air...
2
Antalya ...
İstanbul ...
3
Malatya ...
Trabzon ...
4
İzmir Air...
Ankara ...
Nam...
Pas...
Tick...
Price
Date
Seat
Flight ID:
Selected Ticket ID:
Class:
Economy
EXCHANGE
CHECK-IN
BUY
CANCEL
TICKET INQUIRY


Passenger Operations
Ticket Operations
Premium Club
If you get Premium Club, You will get 20% discount
OK
Cancel
My Profile


Back
Log Out
My Profile
Point:
0.0
Name:
Rana
Surname:
Gül
Email:
ranagul17@exmple.com
Age:
21
Gender:
F
Membership:
false

CHAPTER FIVE

CONCLUSION AND FUTURE WORKS

In this project we will try to create a new airline management system. This system aims at a faster and easily manageable program. That's why the data is kept in ArrayList structure instead of array structure. In addition, this system allows passengers (customers), employees and managers to access and work in the system in a flexible and simple way.

This system is a system produced to perform some simple operations. The project will be developed by adding many features such as interacting with the bank APIs and controlling the payment, keeping the profit and loss status of the company, offering different payment alternatives, keeping the data in the database, enabling more employee types, having more membership types, adding branches that can sell tickets.