King Abdulaziz University
Faculty of Computing and Information Technology
Summer (2020)

Course Code: CPCS 203 Course Name: Programming II

Assignment # 2 (Flight Booking System)

Assigned Date	**Monday, 22/06/2020**
Delivery Date and Time	**Saturday, 04/07/2020 at 11:59 PM**

WARNING:

- This program must ONLY be submitted on the Blackboard!
- This assignment is worth 10% of the overall module marks (100%).
- NO assignment will be accepted after **Saturday, 04/07/2020 at 11:59 pm** for any reasons.
- For discussion schedule, check the teacher name, date and time on the blackboard. **Further information is provided in the course syllabus.**

Objectives

- **Implementing classes and objects.**
- **Performing procedure on objects of different classes.**
- **Learn how to use and implement class, object concepts and association between classes.**
- **Learn to use File I/O (Reading/Writing from/to files).**

Delivery:

- Submit your assignment on the Blackboard ONLY.
- **Make sure to add your name / IDs / Section / course name / Assignment number, as comment at the beginning of your program**.

The program in this assignment is a **Flight Booking System** that can be used by any company to book seats for its customers. The sales staff in a company will use the system to:

(1) Add flight details in the system,
(2) Add passenger details in the system,
(3) Make a new reservation for a customer based on the request.

To use the flight booking system, sale staff has to set **initial parameters** one time. These parameters are the **total number of flights** the system has to manage, the **total number of passengers** the system has to create records for, and the **number of booking requests** the system has to make [see sample input file].

The Initial Procedure of the Program

To set the **initial parameters**, your program will **first** read the following parameters from [input.txt]:

1. **Total Number of Flight**
2. **Total Number of Passengers**
3. **Number of Booking Requests**

These parameters will be in the first line of the file [input.txt]. To read from the file, the system should first check if the file [input.txt] exist or not and will display an error message if the file does not exist.

- The first number (4) in the first line determines the **Total Number of Flights**
  [this means the system will have the maximum of FOUR Flights]
- The second number (10) in the first line determines the **Number of Passengers**
  [this means the system will have the maximum of TEN Passengers]
- The third number (4) in the first line determines the **Number of Booking Requests**
  [this means the system will create a maximum of FOUR Reservations]

The system (your program) then will read the following command to perform one of the programs methods:

(1) Add_Flight

Using this command, the system will add the **flight details** based on the **total number of flights** available in the system [see sample output file].

Example

Add_Flight SA220 Jeddah Makkah 416 8 450

In this line **Add_Flight** is the command and **SA220 Jeddah Makkah 416 8 450** are the details of the flight where **SA220** is the flight code, **Jeddah** is cityFrom, **Makkah** is cityTo, **416** is the total number of seats, **8** the remaining number of seats, and **450** is the price [see Figure1].

Note: that the system shall not allowed more than **Total Number of Flights** to be registered**.** If this happen, a message that says "Flight **Flight_code** is not Added, You exceed the maximum number of Flights" will be displayed.

(2) Add_Passenger

Using this command, the system will add the passenger details taking into account the **total number of passengers** allowed in the system [see sample output file].

Example

Add_Passenger Saud_Ahmad 25 M 053547099

In this line **Add_Passenger** is the command and **Saud_Ahmad 25 M 053547099** are the details of the passenger where **Saud_Ahmad** is the name, **25** is the age, **M** is the gender, and **053547099** is the phone number.

Note: that the system shall not allowed more than **Total Number of Passenger** to be registered**.** If this happen, a message that says "Passenger **Passenger_name** Was not Added, You exceed the maximum number of Passengers" will be displayed.

(3) Make_Booking

Through this command, the system will create a new reservation for a customer based on the **number of booking requests** allowed in the system [see sample output file].

Example

Make_Booking SA220 2 Faisal_Ahmad Sami_Mohammed

[**Make_Booking** flight_code number_of_passengers passenger1_name passenger2_name ….]

In the above command the system will book a flight for 2 passengers in the flight SA220.

Notes that when the system **makes a new booking**:

- The system will first check the **flight code** that the customer wants to book. If the flight code is not in the system, the system should display an error message that says, **"the flight code is not in the system"**.
- The system will check the **passengers' name** who want to travel in current booking [see sample output file]. If one of the names is not in the system, the system will display an error message that says, "**passenger_name is not a registered passenger**".

- If the seats requested exceed the number of available seats in this flight, the system will display an error message that says, "**seats requested exceed the number of available seats**".

4. The system will not allow more than **Number of Booking Requests** to be registered**.** If this happen, an error message that says, "**you exceeded the number of booking requests**" will be displayed.

Finally, the method will generate and return a unique passenger name record (PNR) for every booking, PNR must start from JED100 onwards, i.e., First booking JED100, next booking JED101, Next JED102, etc. [remember PNR is not fixed value]

(4) **Search_Print**

This command will **search and print a booking** information given a PNR code [see sample output file]. This option will show complete details of the booking, which includes the flight details, the passenger details, and the flight cost [see sample output file]. Note that this command will return an error massage if the PNR is not found. For example, "**PNR JED103 is not a valid PNR code**"

Example

Search_Print JED100

This command will show complete details of the booking JED100

(5) Flight_Status

This command will print / display complete flight status and details which includes available seats, destinations (from/ to), etc. Note that this command will return an error massage if the flight number is not found. For example, "**Flight number SA710 is not found in the system**"
[see sample output file].

Example

Flight_Status SV768

This command will display the complete information of the flight SV768.

Classes Description:

You have to create the following base classes in this program.

- **Flight** class
- **Passenger** class
- **Reservation** class

1. The Data fields in the **Flight class** are as follows.

flightCode	SV768	**cityFrom**	Jeddah	**cityTo**	Dammam
totalSeats	5	**remmaningSeats**	1	**price**	450.0

2. The Data fields in the **Passenger class** are as follows.

Name	Ahmed Zahrani	**age**	24	**gender**	M
Phone	05354703383				

3. The Data fields in the **Reservation class** are as follows.

PNRNumber	JED100	**ReservationDate**	2015-02-26
Flight	Object from flight	**Passenger []**	Array of Passengers as there might be more than one passengers travelling in same booking.

You program also need to include a class called P2_134343.java class to test the Booking System. This is your main class in the application. Note that [134343] **will be different from student to student as it is StudentID.**

See the UML diagram in figure 1 to know complete data members and methods required in each class.

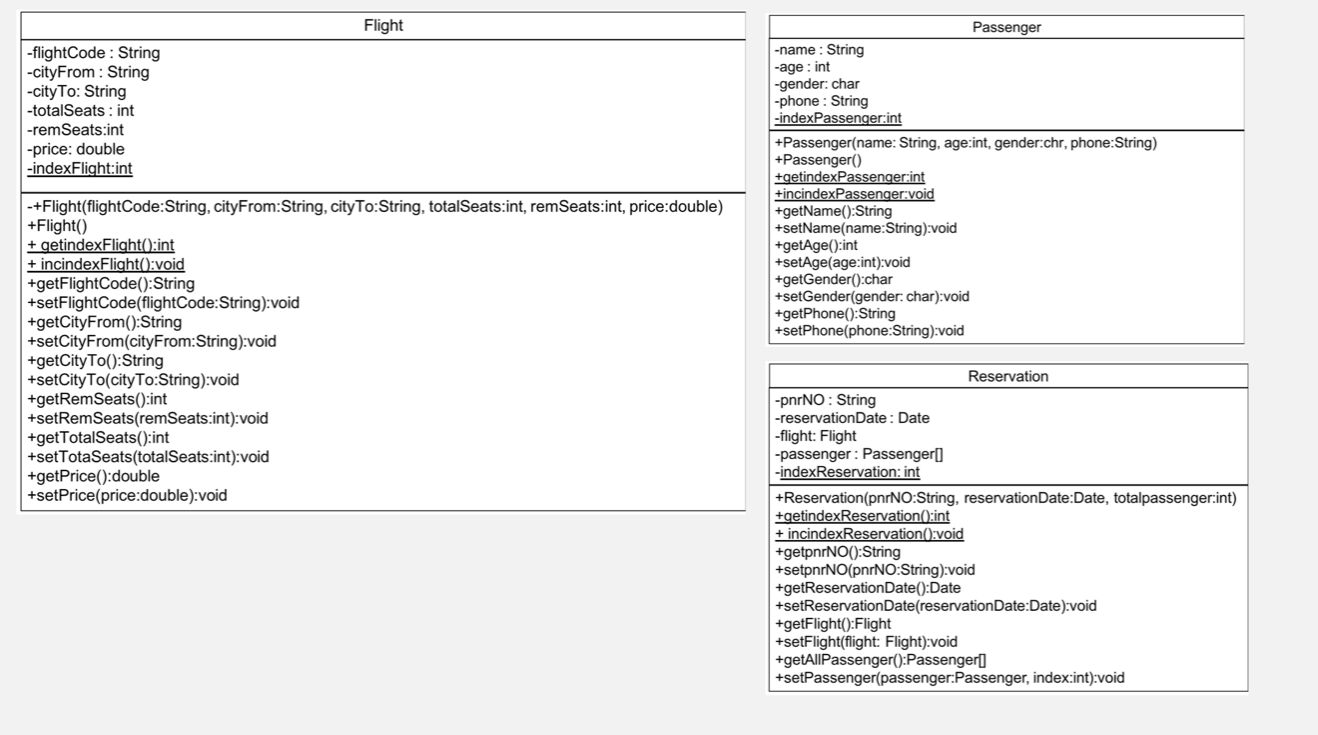

Figure 1: UML Diagram Flight Booking System

Zoom to see the diagram

Arrays and their description:

You will have to create the following arrays:

- ✓ **Flight [] flight**, which is an array of **Flight** objects.
- ✓ **Passenger [] passenger**, which is an array of **Passenger** objects.
- ✓ **Reservation [] reservation**, which is an array of **Reservation** objects.

Methods and their description:

a) inputFlight

This method will be used to read the flight details from the input file and add the flight details to the system. System only allows limited flight entry [see sample output file].

b) inputPassenger

This method will be used to read the passenger details from the file and enter passenger details to the system. System only allows limited passenger entry [see sample output file].

c) BookingFlight

This method will make booking based on the details from the file. System only allows limited booking entry [see sample output file].

d) **FlightStatus**
This method will search and print the details of a flight given its flight code. The method will also list all the passengers in these flights [see sample output file].

e) **Bookingdetails**
This method will search and display the details of a single booking given its PNR code.

The functionality of the classes is shown in UML diagram, see Figure 1, mainly there are mutators(setter) and accessors (getter) in every class see Figure 1. You are allowed to add functionality if required!

Important Notes:

- Your Code, output, results etc. must be in a readable form.
- Organize your code in separate methods.
- **Repeat the program until Quit command is read by your program.**
- Use comments in your code.
- Use meaningful variables.
- Use dash lines separator between each method.

Delayed submission will be marked according to the announced policy on BB

Deliverables:

You have to submit ONE File which contains all Java Classes. The file name must be like

P2_134343.java , Where **134343** is your ID

NOTE: your names, IDs, and section number must be included as comments in all files!

Output Format

Your program must generate output in a similar format to the given sample output file.

Good Luck and Start Early!