KING SAUD UNIVERSITY COLLEGE OF COMPUTER AND INFORMATION SCIENCES COMPUTER SCIENCE DEPARTMENT		
CSC 111: Java Programming 1	**The Project Airline**	**1st Semester 1439-1440**
Due Date: Tuesday 4/12/2018 -11:59 p.m.		

Program Tasks Overview

You need to write a program for Airline to manage domestic flights information. The program should enable the flights data manager to complete the following tasks:

- Add a new flight to the flights list.
- Find the information about a flight given its number.
- Print all the flights.
- Print all the flights having the same given date.
- Update the departure and arrival time of specific flight.
- Print the total number of flights.

Assumptions:

- All flights will depart from Riyadh (King Khalid Airport).
- When adding a new flight, the *FlightNumber* must be generated automatically using a specific formula before adding the flights to the system.
- *Date*, *DepartureTime* must be checked and validated before adding the flight.
- There is a counter for the number of flights, the *TotalFlights* will be incremented whenever a flight is added successfully.

For your project, you must implement and use two classes:

I. Class Flight:

A Flight class is identified by:

- flightNumber: the flight number.
- destination: arriving city. There is a list of 7 cities to choose the destination, which are {dammam, jeddah, yanbu, hail, abha, tabuk, taif}
- gate: the gate number for boarding the plane.
- date: the flight date in the format (dd/mm).
- departureTime: the departure time in the format (hh:mm) and 24-hours system.
- arrivalTime: the arrival time to the destination in the format (hh:mm) and 24-hours system.
- **totalFlights**: the total number of flights in the system

The methods to be included in the Flight class are:

Flight(): A default constructor that initializes all instance variables of the class:

- Integer types are initialized to zero
- Strings are initialized to null ("")

Flight (int destination, String date, int gate, String departure): Another constructor that initializes new flight with the initial values from the user for the following attributes : destination , gate, date and departure Time and call **calculateArrivalTime**() method to set the arrivalTime .

Setter Methods (one for each): That sets the values for (destination, date, gate, departure).

Getter Methods (one for each): That returns the values of: (destination, date, gate, departure, arrival).

generateFlightNumber(): Generates and stores a flight number in *FlightNumber*; The *FlightNumber* is formed by the first three characters in the destination (in uppercase) followed by "00" and concatenated with the *TotalFlights* number.

Example: If the destination is Dammam and the current *TotalFlights* is 3 → *FlightNumber* = DAM003

calculateArrivalTime(): Calculates the *ArrivalTime* depending on the *Destination* and the duration of the flight. The *ArrivalTime* should be in (HH:MM) format and 24-hours system. It should be checked if the arrival time will be in the next day, add ("+1") beside the *ArrivalTime.*

Example: If the destination is Dammam and the *departureTime* is 02:15 → *ArrivalTime*=03:20
If the destination is Jeddah and the *departureTime is 23:30* → *ArrivalTime*=00:15+1

The duration of the time between Riyadh and each city:

Dammam	65 minutes
Jeddah,Yanbu and Abha	105 minutes
Hail	75 minutes
Tabuk	80 minutes
Taif	95 minutes

printFlightInfo() : This method should print the information of a flight with 15 columns width and should be left justified.

Flight Number: ***flight number***	Gate: ***(gate)***
Destination: ***city***	Date: ***date***
Departure Time: ***departure***	Arrival Time: ***arrival***

2. Class Airline:

This class represents the Airline application. Class Airline contains a list of 100 flights, FlightsList[].

This is the class you use to test your program. The methods' prototypes and behaviors are defined as follows (all public):

o **main:** The main method that includes a menu for the user asking him what he would like to do, as follows:

(1) Add a Flight
(2) Find a flight
(3) List all flights
(4) List flights for a given date
(5) Update Departure & Arrival Time
(6) Total number of flights
(7) Exit

o **Add flight:** Creates a new flight and add it in the first available space of the array FlightsList[] , this method returns true if the add operation was completed successfully, and false otherwise. The flight is successfully added if it's date, departure and arrival time has the correct format, if the flight is not already in FlightsList []., and if number of flights doesn't exceed 100

- Method name: addFlight
- Method type: Boolean
- Formal parameters: destination, date, gate, departure

o **Find flight:** Returns the index of a given flight number if the flight is in the flights list, or -1 if the flight is not already in the flights list

o

- Method name: findFlight
- Method type: int
- Formal parameters: FlightNumber

- **List all flights :** Prints all the flights' information in the flights list. Nothing will be printed if there is no flight.
    - Method name: printAll
    - Method type: void
    - Formal parameters: non

- **List all flights for a given date:** Prints all the flights' information in the flights list that have the same date. Nothing will be printed if there is no flights in the same date.

    - Method name: printAll
    - Method type: void
    - Formal parameters: date

- **Update departure & arrival time:** Updates the departure time of specific flights by using FlightNumber, then prints that flight information after updated.
    - Method name: updateTime
    - Method type: void
    - Formal parameters: FlightNumber, departure

- **Total number of flights:** Returns the total number of flights in the flights list

    - Method name: getNumberOfFlights
    - Method type: int
    - Formal parameters: none

Requirements:

- You must define the classes Flight and Airline.
- Same as all programming assignments:
    - Use meaningful variable name and good indentation.
    - You must avoid code duplication, by calling appropriate methods (rather than cutting and pasting code).
    - Don't change the names of any class, attribute or method.
    - Submit your project even if you are not able to complete all the functions, however, for the methods of Flight class that you couldn't implement provide a shell (empty body, just a header) for the method. This will allow us to compile your program and test the components you have implemented.

The project has two phases :

Phase 1: (Upload UML on LMS before **Friday 23/11/2018 -11:59 p.m.)**

Deign the UML for the classes (**flight and Airline**). Please make it nice and neat.

Note: you can use *www.gliffy.com*to draw the UML

ON Saturday 24/11/2018 the correct UML will be available on LMS and you will use it to implement your classes (you will implement the correct one).

You will need to finish mapping the UML to the correct code (class, attributes and method headers).

Phase 2:

1) Complete the implementation of all methods bodies.
2) Test your code using the following test cases.

Flight 1	Flight 2	Flight 3
Destination: Dammam Date: 25/11 Gate:12 Departure Time: 03:15	Destination: Abha Date: 04/12 Gate:3 Departure Time: 23:00	Destination: Dammam Date: 15/12 Gate:1 Departure Time: 16:45

Submission:(Upload your project on LMS before **Tuesday 4/12/2018 -11:59 p.m.)**

- Your project should be submitted on LMS (with your name, section number, serial number and date) that includes your code and results. A print out for your results (with your name and section number) and test cases should be submitted as well.
  Note: since you are a group, the group leader will upload your work.
- Discussion will be on **Wednesday 5/12/2018.**
- On the discussion day you need to bring your laptop with the code ready for running.
- You need to be ready 10 minutes early before your time slot (time and date will be announced)
- Please be ready to explain your code and run your code using the test data that will be provided by the TA during your discussion time.