BLG252E OBJECT ORIENTED PROGRAMMING ASSIGNMENT 1

DUE DATE : 25.03.2015, Wednesday, 23.00

Submit via Ninova, submission via e-mail will not be graded.

Guidelines:

Academic dishonesty including but not limited to cheating, plagiarism, collaboration is unacceptable and subject to disciplinary actions. Any student found guilty will have -100 penalty points.
More clear: This is not a pair-design assignment or a research problem, do not use codes from Google and **Write your own code!!**

- Make sure you write your name and number at the top of all the files of your project.
- Your program **should be compiled and run on Linux** environment using **g++**. (test your program using ssh) Include all necessary header files to your code. Do not use precompiled header files and Windows specific header files and functions.
- Use comments wherever necessary in your code to explain what you did.
- Be careful with the methods/attributes which are supposed to be constant, static, private.
- Submit a working copy at least half an hour before the due time to avoid last minute rush accidents.
- If you have any question about the homework, contact with research assistant Neziha Akalın via email (akalinn@itu.edu.tr).

PROBLEM DESCRIPTION AND CONSTRAINTS

You are required to complete a C++ program that implements a basic Airport System. The program allows a user to create airports, airlines, and flights. Each airline is associated with a set of flights. A flight has an originating airport (origin) and destination airport (destination). The originating and destination airports cannot be the same. Each flight is associated with a flight section (e.g., first class and business class sections). Each flight section consists of seats organized in rows and columns. The system consists of a SystemManager that provides a single point of access to the functions.

An example to understand the system can be given as follows:
Ataturk Airport is an airport in Istanbul and Esenboga Airport is an airport in Ankara. There are flights in each airport. There are different airline alternatives to choose in each airport (for example Turkish Airlines can be chosen to fly from Istanbul to Ankara. The flight has a date such as 11.03.2015 and the flight has a flight number such as 1858K. Turkish Airlines has lots of other flights. Flight section is indicated in the ticket such as 2A, in the number 2A, 2 shows the row and A shows the column of flight

section and each flight section also associated with a seat class such as business class) there can be more than one flight sections such as one for business, one for economy etc.

In this assignment, you will be required to implement the following functionality:

1. **Create an airport:** An airport must have a name consisting of exactly three alphabetic characters. No two airport can have the same name.

2. **Create an airline:** An airline has a name and no two airlines can have the same name.

3. **Create a flight:** A flight has an identifier that is a string of alphanumeric (combination of alphabetic and numeric characters).

4. **Create a section for a flight:** The number of rows and columns must be provided when creating a section. Columns here indicates which column will be included, for example

   createSection("JET","123", 2, 2, 1) // create a section from airline JET, flight id 123, this section includes 2nd row , 2nd column and this section is business class

5. **Find available flights:** Finds all flights from an originating airport to a destination airport with seats that are not booked on a given date.

6. **Book a seat:** Books an available seat from a given originating airport to destination airport on a particular date, on a given flight.

7. **Print system details:** Displays attribute values for all objects (e.g., airports, airlines, flights) in system.

Required Classes

Your program must include the following classes:

class SystemManager: This class provides the interface to the system (this class can reach all the classes). The SystemManager is linked to all the airport and airline objects in the system. When it is firstly created, the SystemManager has no airport or airline objects linked to it. To create airports and airlines, the createAirport() and createAirline() operations defined in this class must be invoked. The class also contains operations for creating sections or flights (e.g., business class and economy class sections), finding available flights between two airports, and booking a seat on a flight. A printout of information on all the airports, airlines, flights, flight sections and seats is obtained by invoking displaySystemDetails(). For more details about methods please read the descriptions below.

Methods of SystemManager:

1. **createAirport(string n):** Creates an airport object and links it to the SystemManager. The airport will have a name (n); n must have exactly three characters. No two airports can have the same name. You can also use dynamic char array instead of string.

2. **createAirline(string n):** Creates an airline object with name n and links it to the SystemManager. No two airlines can have the same name. You can also use dynamic char array instead of string.

3. **createFlight(string aname, string orig, string dest, int year, int month, int day, string id):** Creates a flight for an airline named aname, from originating airport (orig) to a destination airport (dest) on a particular date. The flight has an identifier (id). You can also use dynamic char arrays instead of strings.

4. **createSection(string air, string flID, int rows, int cols, int s):** Creates a section of class s, for a flight with identifier flID, associated with an airline air. The section will contain the input number of rows and columns. You can also use dynamic char arrays instead of strings.

5. **findAvailableFlights(string orig, string dest):** Finds all flights from airport orig to dest with seats that are not booked. You can also use dynamic char arrays instead of strings.

6. **bookSeat(string air, string fl, Seat s, int row, char col):** Books seat in given row and column in section s, on flight fl of airline air. You can also use dynamic char arrays instead of strings.

7. **displaySystemDetails():** Displays attribute values for all objects (e.g., airports, airplanes) in system.

class Airport: Objects of this class represent airports. Airport class must have name exactly 3 characters length. If you need any other methods or attributes, you can add to the class.

class Airline: This class maintains information about airlines. An airline can have 0 or more flights associated with it. When an airline is created it is not associated with any flights. All flights for a given airline must have unique flight ids. If you need any other methods or attributes, you can add to the class.

class Flight: This class maintains information about flights. A flight can be associated with 0 or more flight sections. There can be only one flight section of a particular seat class in a flight, e.g., same rows and columns can not belong to two types. The seat types will indicated with an integer, such as business=1, first=2 and economy=3. If you need any other methods or attributes, you can add to the class.

class FlightSection: This class maintains information about flight sections. A flight section has a seat class (business=1, first=2 or economy=3) and must have at least 1 seat. hasAvailableSeats() returns true if the section has some seats that are not booked, and bookSeat() books an available seat. A flight section can contain at most 100 rows of seats and at most 10 columns of seats. If you need any other methods or attributes, you can add to the class.

class Seat: This class maintains information about seats. A seat has an identifier (row number and a column character from A to J) and a status which shows whether the seat is booked or not. If you need any other methods or attributes, you can add to the class.

Test Program: Use the test program below to test your codes.

```cpp
#include<iostream>
#include<string>
#include "SystemManager.h"

int main() {

		SystemManager res;

		res.createAirport("DEN");
		res.createAirport("dfw");
		res.createAirport("LON");
		res.createAirport("DEN");//invalid, same name created before
		res.createAirport("denw");//invalid more than 3 chars

		res.createAirline("delta");
		res.createAirline("AMER");
		res.createAirline("FRONT");

		res.createAirline("front"); //invalid same name created before

		res.createFlight("DELTA", "DEN", "LON", 2013, 10, 10, "123");
		res.createFlight("DELTA", "DEN", "DEH", 2013, 8, 8, "567abc");
		res.createFlight("DEL", "DEN", "LON", 2013, 9, 8, "567"); //invalid airline
		res.createFlight("DELTA", "LON33", "DEN33", 2013, 5, 7, "123");//invalid airports
		res.createFlight("AMER", "DEN", "LON", 2010, 40, 100, "123abc");//invalid date


		res.createSection("DELTA","123", 2, 2, 3);
		res.createSection("DELTA","123", 2, 3, 1);
		res.createSection("DELTA","123", 2, 3, 1);//Invalid
		res.createSection("SWSERTT","123", 5, 5, 3);//Invalid airline


		res.bookSeat("DELTA", "123", 1, 1, 'A');
		res.bookSeat("DELTA", "123", 3, 1, 'A');
		res.bookSeat("DELTA", "123", 3, 1, 'B');
		res.bookSeat("DELTA888", "123", 2, 1, 'A'); //Invalid airline
		res.bookSeat("DELTA", "123haha7", 2, 1, 'A'); //Invalid flightId
		res.bookSeat("DELTA", "123", 3, 1, 'A'); //already booked

		res.displaySystemDetails();

		res.findAvailableFlights("DEN", "LON");

		return 0;
	}
```