Escenario Aeropuerto

El diagrama de clases de la Figura 1 modela la información de la operación de un pequeño aeropuerto. El aeropuerto es utilizado por tres líneas aéreas con un máximo de cinco vuelos diarios cada una.

Se requiere diseñar e implementar un sistema computacional que permita administrar la operación de los vuelos diarios que se registran en el aeropuerto. El sistema deberá manejar información del aeropuerto, líneas aéreas, vuelos y pasajeros.

También, se espera que el sistema programe vuelos asignando horarios, destinos, tripulaciones, asientos a los pasajeros y calcule y registre el consumo de combustible para cada viaje.

Cada vuelo tiene asientos dispuestos en filas. Las líneas aéreas sólo manejan aviones de 30 filas con 4 asientos por fila. Los clientes pueden reservar asientos, y se les asigna un número de asiento y fila. Pueden reservar varios asientos contiguos.

El sistema deberá generar reportes de todos los vuelos diarios, así como calcular los ingresos totales por línea aérea como los ingresos totales del aeropuerto.

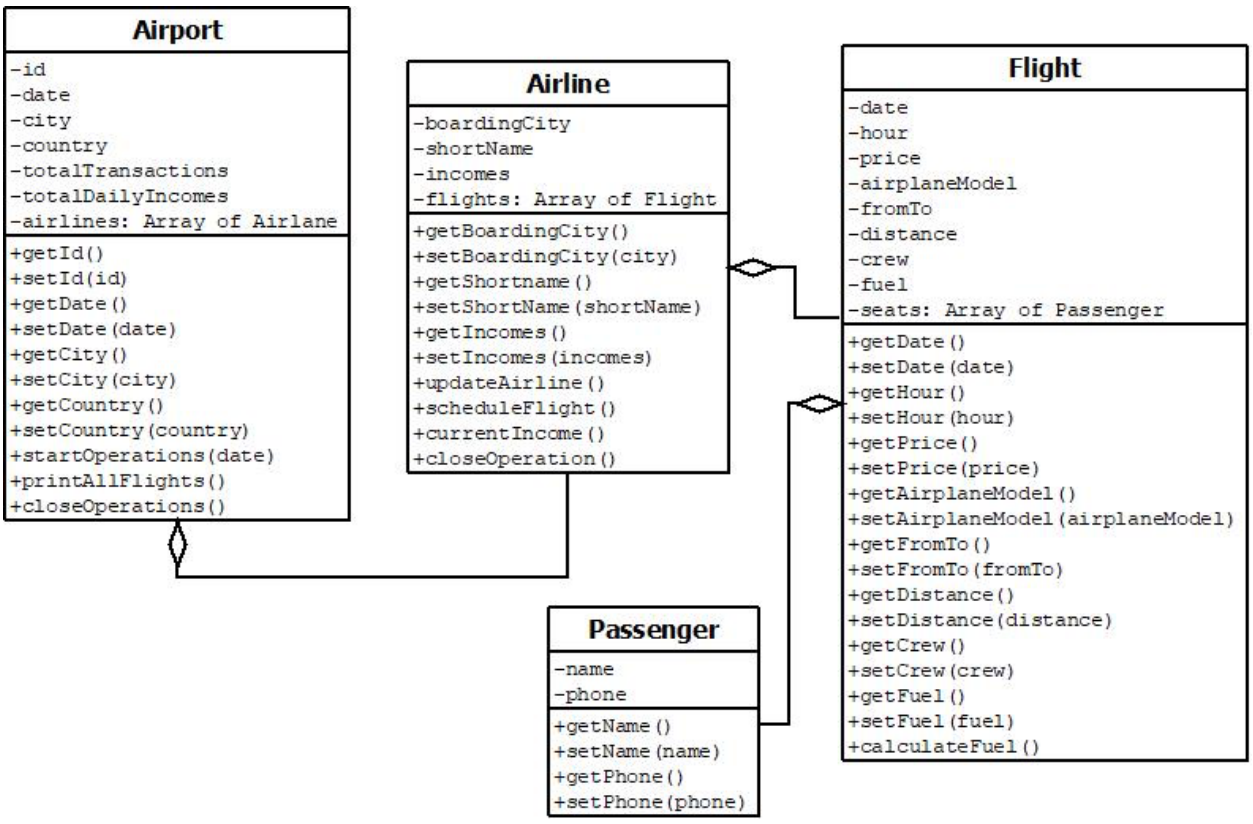

Figura 1. Diagrama de clases del escenario Aeropuerto

Instrucciones:

Implementa el código en C++ que solucione el escenario planteado.

Requerimientos funcionales

Clase *Airport*

Utiliza la constante NUM-AIRLINES = 3 para definir el tamaño del arreglo *airlines*.

Métodos:

- Métodos ***getters*** y ***setters*** para todos los atributos de la clase.
- **Constructor**: solicita al usuario la fecha de operación en formato "*28/12/2020*" y construye todos los atributos con dimensión (arreglos) asociados a las clases *Airport, Airline* y *Flight*. Llama al método *startOperation*.
- ***startOperations***: recibe la fecha del método constructor y asigna ese valor al atributo *date*. Además, solicita al usuario los siguientes valores:

    - Identificador del aeropuerto (ej. AIMX).
    - Ciudad de operación (ej. CdMex).
    - País (ej. México).

    Para cada registro del arreglo *airlines*, inicializa el valor del atributo *boardingCity* con el valor de la ciudad de operación del aeropuerto y coloca la fecha (*date*) en todos los registros del arreglo *flights* de la clase *AirLine*.

- ***printAllFlights***: imprime la información de todos los vuelos programados para ese día. El encabezado del reporte inicia con "*Los vuelos programados para 99/99/9999 son*: ", donde la fecha corresponde al valor del atributo *date* de la clase. El reporte no debe mostrar registros vacíos. La información de los vuelos debe mostrarse agrupada por línea aérea y para cada vuelo, deben mostrarse los asientos asignados a pasajeros.

- ***closeOperations***: debe calcular, almacenar y mostrar:

    - El número total de vuelos de todas las líneas aéreas que se programaron ese día (almacenar en *totalTransactions*)
    - El total de ingresos generados por todos los vuelos programados (almacenar en *totalDailyIncome*).

Clase *Airline*

Utiliza la constante NUM-FLIGHTS = 5 para definir el tamaño del arreglo *fligths*.

Métodos:

- Implementa los métodos ***getters*** y ***setters*** para todos los atributos.

- ***updateAirline***: llenará los datos *shortName* (ej. AM, JAA) y *name* (ej. Aeroméxico, Delta) de la línea aérea solicitando los datos al usuario. (Información sobre claves de líneas aéreas que puedes usar de referencia: https://es.wikipedia.org/wiki/Anexo:C%C3%B3digos_OACI_de_aerol%C3%ADneas).

- ***scheduleFlight***: programará un vuelo. Utiliza métodos de las clases implicadas (*Flight* y *Passenger*). Se puede programar un vuelo en una línea aérea siempre y cuando tenga aeronaves disponibles (espacio en su arreglo); de lo contrario, se marcará el mensaje de aviso "*Vuelos de la línea aérea X saturados, imposible programar*". Los datos a solicitar al usuario son:

    - Identificador de la línea aérea que realiza el vuelo (*shortName*).
    - Modelo del avión (*airplaneModel*).
    - Ciudad destino (la integración del valor de ciudad de embarque más el valor de ciudad destino genera el valor del atributo *fromTo* y debe quedar completo en este punto).
    - Distancia del vuelo (*distance*).
    - Clave de la tripulación (*crew*).
    - Horario del vuelo "12:00" (*hour*).
    - Precio del boleto (*price*).
    - Mientras el pasajero decida continuar, y no se llene el vuelo que se programa, se asignarán y mostrarán asientos del vuelo a pasajeros (arreglo bidimensional *seats*). Recuerda que los aviones tienen 4 columnas y 30 filas. El pasajero elige su asiento y un mismo pasajero puede seleccionar el número de asientos que requiera.

- ***currentIncome***: calcula, almacena (en *incomes*) y muestra el monto de ingresos de la línea aérea en función de los asientos vendidos y del precio del boleto de cada uno de los vuelos programados. Es importante que cada vez que se programen vuelos de una línea aérea, se actualice el atributo *incomes*.

- ***closeOperation***: acumula y muestra en pantalla el ingreso total del día ($).

Clase *Flight*

Utiliza las constantes COLUMNAS = 4 Y FILAS=30 para definir las dimensiones del arreglo de *seats*.

Métodos

- Implementa los métodos ***getters*** y ***setters*** para todos los atributos de la clase.

- ***calculateFuel***: calcula y almacena en el atributo *fuel* la cantidad de combustible que requerirá el vuelo según el destino. Este método se debe utilizar al momento de estar programando un vuelo. (*Tip. Un avión 747 se dice consume alrededor de 1,200 litros por cada 100 kilómetros. Para calcular el combustible de otro tipo de avión, puedes utilizar una proporción de estas cantidades*). No se pide datos al usuario y el dato calculado debe mostrar inmediatamente abajo de solicitar el valor del atributo *destino*, "*El combustible requerido para viajar a x es de z galones*", donde x es el valor del *destination* y z el valor calculado para *fuel*.

Clase *Passenger*

Métodos

- Implementa los métodos ***getters*** y ***setters*** para todos los atributos de la clase.

Material editado, diseñado, publicado y distribuido por el Instituto Tecnológico y de Estudios Superiores de Monterrey.

Ave. Eugenio Garza Sada 2501 Sur Col. Tecnológico C.P. 64849 | Monterrey, Nuevo León | México.