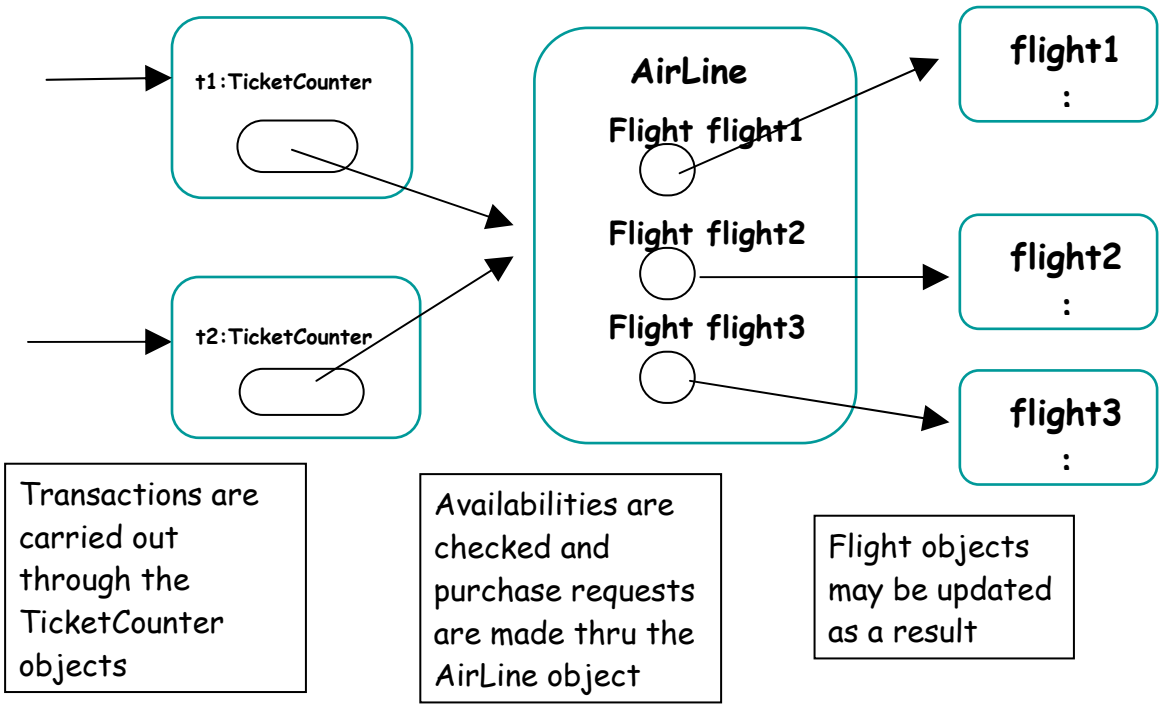
CS 21a Introduction to Computing **First Semester, 2007-2008**
Project 1: Airline Ticket Purchasing Simulation **Deadline: August 22**

Project Overview:

In an airline ticket purchasing system, tickets are purchased through ticket counters that all connect to a central airline database. In turn, the airline database can be viewed as an aggregate object that manages flight objects. For this project, we assume that the airline carries only three flights: AL101, from Manila to Cebu, AL555 from Manila to Davao, and AL503 from Manila to Hongkong. We will assume for simplicity that we are selling airline tickets for a single date. However, each flight has a different passenger capacity for first class and economy class seats, with different associated fares. Refer to the following diagram:



Create a simulation of a simple airline ticket purchasing system.

To create this simulation, you will need four (4) Java classes:

1. `TicketCounter.java`
2. `AirLine.java`
3. `Flight.java`
4. `AirLineSimulation.java` -- a driver for testing will be provided for you, but you also have to create your own according to the specs below

Purchasing of tickets is a transaction made through TicketCounter objects. Each TicketCounter should keep track of the total number of tickets that have been purchased through that counter, as well as its total sales.

Each TicketCounter object is associated with exactly one AirLine object. The AirLine manages three (3) Flight objects. Each Flight object should store the number of available and occupied first class seats, the number of available and occupied economy class seats, an economy-class seat fare and a first-class seat fare.

When purchasing a ticket through an instance of a TicketCounter object, the following needs to be specified:

1. The flight name, either "AL101", "AL555" or "AL503"
2. The type of flight, either "first class" or "economy"
3. The quantity of tickets to be purchased

The TicketCounter object in turn passes this information to the AirLine it is connected to, and checks if the purchase is possible. The purchase is possible if the specified flight (specified by the flight name) still has enough slots for that particular type of flight. If the purchase is possible, the AirLine will proceed with the purchase. This implies:

- The available seats on the specified flight will be reduced by the number of tickets purchased. The decision whether to reduce economy seats or first class seats depends on the specified flight type in the purchase.
- The sales of the TicketCounter that sold the tickets will increase. The increase depends on the type of tickets sold (either economy or first class) and the quantity.

Each instance of TicketCounter must be able to report its own statistics, which consist of the following:

1. Total number of tickets sold, broken down into economy and first class tickets.
2. Total sales

These statistics are specific for each instance of TicketCounter.

Your program should display status messages that reflect the success or failure of an operation. Success messages include but are not limited to:

- *"A ticket counter has been created"*
- *"You've successfully purchased tickets for first class flight."*

Error messages include but not limited to the following:

- "*No such flight*." - Tickets may only be purchased for a flight that exists. That is, you cannot buy seats of flight "AL666", for example.
- "*No more first-class tickets available*." - Tickets may only be purchased for a seat that exists. That is, you cannot buy a first-class seat if all first-class seats have been sold.

II. Specifications per Object

A. TicketCounter.java

TicketCounter.java should at least have the following attributes (the names of the attributes are left to the programmer and it is possible to have more attributes):

1. A "ticket counter number" that identifies this instance of TicketCounter (it is like the name of a TicketCounter). This attribute is of type **int**.
2. An airline object that this TicketCounter is associated to.
3. The sales for the ticket counter, an accumulated total of economy and first class tickets sold. This attribute is of type **double**.
4. The total number of economy tickets sold through the counter.
5. The total number of first-class tickets sold through the counter.

TicketCounter.java should have the following methods (named exactly as it is written here, since it will be called by the driver).

1. TicketCounter(int ticketCounterNumber, AirLine airline)
   A constructor that takes in an integer as the number of that TicketCounter, and the AirLine object is connected to.
2. void purchaseTicket(String flightName, String flightType, int qty)
   The method called in the driver to purchase a ticket. Information needed are the name of the Flight (this refers to an attribute of the Flight object), the type of flight ("economy" or "first class") and the number of tickets to be bought. This method, in turn, calls a the puchaseTicket() method of the AirLine object and passes the values of the parameters to that method
3. void printStats()
   This is a method that displays the ticket counter number, the total sales, the number of economy tickets sold and number of first class tickets sold for this instance of TicketCounter.

B. AirLine.java

AirLine.java should have three flight objects as attributes, referring to the flight objects it manages. *AirLine.java* should have the following methods (note that method 4 should be named as written, methods 2 and 3 may be named differently but need to be coordinated with the TicketCounter class):

1. AirLine()
   The constructor. Whenever this constructor is called, the three Flight objects should be instantiated. This means the constructor for Flight objects is called three times in the AirLine constructor's body

2. void purchaseTicket(String flightName, String flightType, int qty)
   This method is called from within a TicketCounter object. It should check the existence of the particular flight name and type of flight specified in the parameters. Purchasing of tickets should not proceed if any of these parameters have erroneous values, or if the quantity being bought exceeds the number slots/seats available.

3. double getTicketPrice(String flightName, String flightType)
   This method returns the fixed price of an economy ticket or a first class ticket of a given flight.

4. void printStats()
   This method should be named exactly as it is written, since this will be called by the driver. It prints the statistics for the different flights, particularly, the number of seats purchased and available for economy and first-class.

C. Flight.java

The details for *Flight.java* are all up to the programmer, although you will likely need the following attributes: flight name (e.g., "AL101"), fares for first-class and economy class tickets, number of first-class and economy class seats, number of purchased economy class seats. For its methods, we suggest the following:

1. A constructor with at least a flight name as its parameter. Flight names are assigned to instances of Flight objects upon instantiation of a Flight object. You may include the seat capacities and fares in the constructor or set them separately through other methods.
2. Different methods that return flight name, economy seats left, first class seats left, price for an economy-class seat, and price for a first class seat.
3. A purchase method that the Airline object will call. You may decide to have two purchase methods, one each for economy and first-class purchases. Alternatively, a single purchase method that has a string parameter indicating "economy" or "first-class" will do. A quantity parameter, representing the number of tickets, will also be needed for the method(s).

Important: An airline object will have three flight objects with the following details (this fixed).

Flight	Economy Fare	First-Class Fare	Economy Seats	First-Class Seats
AL101	5000.00	10000.00	100	20
AL555	6000.00	12000.00	20	4
AL503	8888.00	22500.00	200	30

A driver and its corresponding output will be posted on the website, so that you could test your program. You will also be asked to create your own driver. Submission will be through moodle, unless otherwise specified by your instructor. Submit a zip file containing all source code by August 22, midnight.

Include a driver of your own in your final submission. Your driver should at least do the following:

1. Create an instance of AirLine (automatically creating 3 flights as explained above)
2. Create 3 to 5 instances of TicketCounter
3. Make several economy ticket purchases from each TicketCounter
4. Make several first class ticket purchases from each TicketCounter
5. Attempt to purchase 1000 economy and 1000 first class tickets (which should display a proper error message)
6. Attempt to purchase tickets for a flight with a flight name that does not exist (which should display a proper error message)
7. Attempt to purchase tickets for a flight with a flight type that does not exist (not "economy" and not "first class"…something like "second class", and this should also result in a display of an error message).
8. Display ticket counter stats for each ticket counter
9. Display airline stats

Visit the course website regularly for any corrections or updates regarding this project.