COSC 210 – Object Oriented and GUI Programming

Assignment 3

Due: September 29th at 2:00 AM

30 points

The objectives of this exercise are to:

1. Gain further experience with using the Eclipse IDE.
2. Gain an understanding of arrays and use of control statements.
3. Develop and test a small Java program applying the concepts of objective 2.

**AFTER YOU HAVE COMPLETED,** put all your files into a zip file named [your name]Assignment3.zip. Include all .class and .java files. Upload your zip file to Moodle by the due date. Printout and turn-in all source files and output from your program.

COSC 210 – COSC 210 – Object Oriented and GUI Programming

Assignment 3 Problem Statement

ValAir wants a new reservation application to print the itinerary and boarding passes for flights of a trip booked by a passenger. A booking for a passenger consists of one or more reservations where each reservation identifies a date, a flight, and a seat on the flight. The itinerary and boarding passes are to be printed as follows:

V a l A i r

Itinerary

Passanger: Bill Jones

Fyler No: BJ103

From: Latrobe

To: Orlando

Stops: 2

Total Travel Time: 4 Hours 5 Minutes

Layover Time: 1 Hours 15 Minutes

Miles Earned: 1513

Flights

Date Flight Depart Arrive Miles

01/14/2012 25 Latrobe 06:50 AM Pittsburgh 07:15 AM 59

01/14/2012 54 Pittsburgh 08:00 AM Charlotte 09:00 AM 438

01/14/2012 96 Charlotte 09:40 AM Orlando 10:55 AM 513

ValAir Boarding Pass

Flight 25 Latrobe to Pittsburgh 01/14/2012

Passenger Seat Boarding Depart

Bill Jones 1A 06:20 AM 06:50 AM

ValAir Boarding Pass

Flight 54 Pittsburgh to Charlotte 01/14/2012

Passenger Seat Boarding Depart

Bill Jones 5F 07:30 AM 08:00 AM

ValAir Boarding Pass

Flight 96 Charlotte to Orlando 01/14/2012

Passenger Seat Boarding Depart

Bill Jones 10C 09:10 AM 09:40 AM

**The computations are as follows**:

From will be the departure city from of the flight of the first reservation of the list of reservations.

To will be the arrival city from the flight of the last reservation of the list of reservations.

Stops will be the number of reservations less one. If there is only one reservation, then print Non-stop in place of a number.

Total travel time will be the difference between the arrival time of flight of the last reservation and the departure time of the flight of the first reservation. Print in terms of hours and minutes. If hours is zero (i.e. total minutes was less than 6), then only print in terms of minutes.

Layover time will be sum of the differences between the arrival time of a flight of a reservation the departure time of a flight of then immediate next reservation. Print in terms of hours and minutes. If hours is zero (i.e. total minutes was less than 6), then only print in terms of minutes.

Miles earned is the sum of the miles for each fight of the reservations taking into account that the minimum miles for a given flight will be 500.

Boarding time will be 30 minutes before the departure time.

**Additional Specifications:**

You will need 4 classes to represent the business objects for this problem, one for the passenger, one for the booking (which includes code to print both the itinerary and boarding passes), for flights and one for reservations. Note that for a booking, there can be many reservations. For passenger, reservation and flights provide getter methods. Provide a setter method only for a seat in a reservation. For a booking, provide getter methods for passenger and for number of reservations.

Use Java classes LocalDate, LocalTime, LocalDateTime to represent dates, times, or date/time timestamps for dates and times as needed. LocalDate.of(year, month, day) will create a LocalDate. LocalTime.of(hour, min) will create a LocalTime using 24 hour clock. LocalDateTime.of(year,month,day,hour,min) will create a LocalDateTime. A class/method ChronoUnit.MINUTES.betwee(t1, t2) can be used to determine then number of minutes between two LocalDates or to LocalDateTimes. For example:

LocalTime t1 = LocalTime.*of*(1,30);

LocalTime t2 = LocalTime.*of*(13,20);

int minsBetween = ChronoUnit.MINUTES.between(t, t2);

All of these classes have a format method which returns the date, time, date/time as a string according to a DateTimeFormatter. Its ofPattern method is used to create a DateTimeFormatter according to a format string. A format string of “hh:mm a” can be used to format a time as a 12 hour clock with AM/PM. For example:

LocalTime t1 = LocalTime.*of*(1,30);

System.out.println(t.format(DateTimeFormatter.ofPattern("hh:mm a")));

Will print 01:30 AM.

A format string of “MM/dd/yyyy” can be used to format a date. For example:

LocalDate d1 = LocalDate.*of*(2017, 9, 10);

System.***out***.println(d1.format(DateTimeFormatter.*ofPattern*("MM/dd/yyyy")));

Note no flights are to be longer than 20 hours.

You do not need to write a program that accepts user input. Like assignment 2, just write a test program (this will be a fifth class). Is should instantiate a sufficient set of objects to print itineraries for three bookings. The test program can hard code the data which is passed to the constructors. (do **not** hard code other values in the business classes except for 500, then minimum miles earned for a flight), Make sure you develop test cases for a non-stop booking , for total travel time greater than an hour, for total travel time less than an hour, for layover time less than an hour, and layover time greater than an hour.

Have the test data be meaningful. That is the arrival time of one fight is less that the departure time of the next used in a booking (some exceptions for the bonus).

Add reservations to a booking one at a time using an add method (**do** **not** pass an array as an argument to the constructor, **do not** have getter or setter methods for an array instance variable).

Place the code to print the itinerary and code to print the boarding passes in separate methods.

Suggest the computation to compute the boarding time be placed in a getter method of the reservation.

Use good programming practices (javadoc, encapsulation, naming conventions, formatting).

**Bonus (5 points):**

Handle a booking that spans more than a day (LocalDateTime will need to be used). Here if the arrival time is less than the departure time, then the flight will start one day and end on the next.