Assignment 2

Due date:	Week 11 Friday (28 Sep 2018) 11:45 PM AEST
Weighting:	30%

Assignment task

Write a java console application that allows the user to read, validate, store, display, sort and search data such as flight departure city (String), flight number (integer), flight distance (integer), flight price (double) and discounted flight price (double) for N flights. N should be declared as a constant and it should be equal to the largest digit of your student id number (e.g. if your ID number is S7840213 then N should be equal to 8 and you can declare it as final int N=8;). The flight departure city, flight number, flight distance, flight price and discounted flight price must be stored in an Array/ArrayList (index 0 for flight 1 and index N-1 for flight N). The minimum and maximum flight numbers which can be stored are 1111 and 9999. The minimum and maximum flight prices for each flight which can be stored are $9 and $900. The flight number and price must be entered from the keyboard and a validation for minimum and maximum values must be done.

Your application should display and execute a menu with the following options. A switch statement must be used to execute the following menu options.

1. Read, validate and store data for N flights
2. Calculate and store discounted price for all flights
3. Display all flights
4. Search a flight by number
5. Display all flights with the lowest flight price
6. Sort and display sorted flights
7. Exit from the application

1. Read, validate and store data for N flights

This option reads flight departure city, flight number, flight distance and flight price for N flights from the keyboard and stores them in Array/ArrayList. If the flight number is less than 1111 and greater than 9999 then an appropriate message should be displayed and the user should be asked to enter a new flight number. Similarly if the flight price is less than $9 and greater than $900 then an appropriate message should be displayed and the user should be asked to enter a new price.

2. Calculate and store discounted price for all flights

This option asks the user to enter the discount percentage between 0 and 100 (e.g. 10) and calculates the discounted flight price based on the following formula (discounted flight price = flight price – (flight price * discount percentage)/100) for each flight and then stores discounted price in Array/ArrayList.

3. Display all flights

This option displays the data for all flights (flight departure city, flight number, flight distance, flight price and discounted flight price).

4. Search a flight by number

This option asks user to enter the flight number and searches for it. If flight number is found then it displays an appropriate message with flight details. If flight number is not found then it displays an appropriate message “flight is not found”. **A built-in search algorithm for searching can be used in this assignment**.

5. Display all flights with the lowest flight price
This option finds and displays the flight stored in Array/ArrayList which has the lowest flight price. If there is more than one flight with the lowest flight price then it displays all of them.

6. Sort and display sorted flights
This option sorts (by departure city) all flights stored in Array/ArrayList in descending order and displays all sorted flights (flight departure city, flight number, flight distance, flight price and discounted flight price). You can use any sorting algorithm. **A built-in sort algorithm for sorting is not allowed in this assignment**.

7. Exit from the application
The application should display an appropriate message with student id and then exit from the application.

The application should work in a loop to enable the user to Read, validate and store data for N flights, Calculate and store discounted price for all flights, Display all flights, Search a flight by number, Display all flights with the lowest flight price, Sort and display sorted flights, and Exit from the application.

Program design

You may use any design that meets the specification. However, a good design will adhere to the following guidelines:

- be logically correct
- be easy to read and maintain
- be well-designed
- use a UML activity diagram
- use appropriate classes, methods and fields

Your design MUST use the classes and methods as listed below.

```
public class Flight
{
   //fields
   //get and set methods
}
public class FlightTest
{
     Method to Read, validate and store data for N flights
     Method to Calculate and store discounted price for all flights
     Method to Display all flights
     Method to Search a flight by number
     Method to Display all flights with the lowest flight price
     Method to Sort and display sorted flights
     Method to Exit from the application
     public static void main(String[] args)
     {
     }
}
```

You may add/use other methods, parameters, fields/variables, constants, etc. which you need to complete the application.

Testing

Testing is important. You should:

- List the different types of test cases.
- Display the results for each test case.

What to submit

You should submit online the following files.

- FlightShop.java (this file contains java code for class Flight)
- FlightTest.java (this file contains java code for class FlightTest).
- Report.docx (this file contains a brief report that includes student name, student ID, unit name, unit code, UML activity diagram for menu option 2 (Calculate and store discounted price for all flights) and test results (screenshots/test cases with results to show that your application is correctly working)).

Warning: You must submit your own work and correct files.

Assessment 2 marking criteria

	Total Marks – 30	**Marks Allocated**
1	**Design, logic and testing – 14**	
	Code readability – appropriate use of comments, indentation and naming conventions	3
	Declaring and using fields/variables and constants	2
	Creating/declaring and using objects, methods and classes	3
	Overall design and logic (structure, efficiency)	2
	Validity of testing as evidenced by submitted test results	2
	UML activity diagram for menu option 2	2
2	**Compilation and execution - 14**	
	(0 - if application doesn't run)	
	Welcome and exit messages	1
	Input data validation and message	1
	Read and store all flights	1
	Calculate and store discounted price for all flights	2
	Display all flights	1
	Search for a flight	2
	Display all flights with the lowest flight price	2
	Sort and display sorted flights	2
	Overall program execution (user friendly, input/output and display)	2
3	**Report – 2**	
	Presentation (fonts, spaces, information, language)	2
4	**Penalty**	
	Penalty for submission of incorrect files including names and formats is 2 marks	
	Penalty for late submission is 5% mark / day or part of a day	