SOFT1001 assignment

Semester 2, 2007

Due: Monday, 8th October 12noon (yes, a little extension)

Design draft must be shown to your tutor at your second lab during week 9, to get feedback before the break.

The flight centre problem

In this assignment you will create a program to allow users look up scheduled flights between two given cities and choose the one that suits best their needs in terms of time, fares, length of trip.
Note: You do NOT need to build a graphical user interface (since you have not learned this topic), so you will use the standard command interface.

Flights depart a city at a certain time, on certain days of the week, and arrive at a specific time at the destination city. For each flight, you will determine rules for calculating three or more different fares: the basic full fare and a range of available decreasing fares depending on various factors that you will determine (for example, a 2 week advance purchase fare is 50% off, a one week advance fare is 20%, some days of the week might be 10% more expensive and so on). In addition some of these fares will have a quota. You must explain your rationale in your code (as comments in the relevant methods).

Your program MUST offer the following services:

- let the user enter the departure city, the destination city, the desired date and time, and display all the flights and corresponding fares available. The flight information should include the flight number, the date and arrival times, the length of the trip. Only the applicable fares should be displayed (i.e. do not display a 2 week advance fare for a request for a flight in one week time)
- allow the user to order the results by:
    - price (best to more expensive)
    - time (increasing)
    - travel time (shortest to longest)
    - best compromise price/travel time (you will create your own way of calculating this compromise and explain it in the method comments)

Your program should also offer at least one additional feature, which you can pick from the following list or create your own, after discussion with your tutor

- Handle time differences: eg Perth is 3 hours behind Sydney, so you need to adjust the calculation of the travel times accordingly.
- Handle availability status: if there are 5 seats available for sale and the user buys 4, there should be only one left for future requests.
- Handle bills: as the user "books" flights, the costs are added up to the bill. To make it more interesting assume that users shop for more than one flight.
- Handle user accounts: rather than having one unique default user, the user can login or your program can ask before each request/flight booking the name of the user.

Your code MUST make use of at least one collection (ArrayList, LinkedList or primitive array).

DATA CREATION: The data (the information about the flights) needs to be entered somehow. Since you have not yet learned how to read and write to a file, you could create a special class to generate this data. For instance you could create manually a set of cities and then generating (using a loop) a

certain number of flights between them with a raw fare. Or you can create them all manually but it is certainly not efficient. That way you can run your program several times always using the same data.

You need to do systematic testing of the classes you create, and submit the test classes along with your code. For one (non-trivial) class you will need to provide a unit testing report (~1 page) summarising the results of your tests.

SUBMISSION **Late submissions will incur a penalty of 10% per day or part thereof.**
Submit before Monday 8 October, 12noon:

1. your program (all the java classes)
2. a one page describing your program and explaining the additional feature you implemented, the way your fares are calculated, the way you calculated the compromise price/travel and any other information need to understand your program.
3. your unit testing report

to your tutor in a single .zip archive named with your login name, e.g. if your login is "xyz123" the archive must be called "xyz123.zip". Your tutor may also require that you provide a hard copy of 2 and 3 as well as asking you to provide more details about your program and tests in the week the assignment is due.

MARKING
There are 4 sections in which your submission will be marked, with a possible adjustment of the mark:

1) **Class design (10 marks):** How good is your choice of classes, how well designed they are (instance variables, public methods with appropriate parameters and names. Good use of preconditions and postconditions. **Design draft (week 9, 1st or 2nd lab) worth 2 marks out of these 10.**
2) **Implementation (10 marks):** How your code works, what it does, and whether it does it correctly. Use of code cliches, helper methods/classes when necessary. Lack of bad practices (useless or repetitive code etc..)
3) **Standard of writing (10 marks)**: Clarity of the code, layout, naming choices, descritive commenting use of javadoc tags
4) **Testing (10 marks):** How well you can prove that your code works (or find errors if it doesn't work). Each of the following is worth 2 marks, however, the tester must have some usefulness to score marks in this section. For example, a tester that just creates an object and then just calls methods without checking any returned values or the state of the object doesn't test anything, so the testing class and testing report would score 0 marks in this section.
    - Tester class detects faults if they exist
    - Tests are thorough, covering various cases as required
    - Test is automated and can be repeated without possible error from human input
    - Presentation of testing code and testing report is clear
    - Testing proves that the non-trivial class is useful to other code and that it can be used as needed

Difficulty adjustment: In addition your tutor will make an overall assessment of the level of difficulty of your program. If the difficulty is of the expected level of a SOFT1001 student, your mark will not be affected. If it is too simple, your mark may be lowered by up to 20%. If the complexity is much greater than what is expected at your level, then your mark may be raised (within the limit of 40 marks) by up to 10%. If in doubt, discuss with your tutor ahead of time the complexity of the additional feature you would like to implement. It is expected that most students will NOT have their mark scaled.