Format of the input file

```
#
CMB(Colombo,Sri lanka)
CCU(Kolkata,India)
LHR(London,United Kingdom)
HND(Tokyo,Japan)
AKL(Auckland,New Zealand)
JFK(New York,United States)
CDG(Paris,France)
.
.
.
.
.
.
.
.
.
#
**
QR,Qatar Airways
*
QR 0655,29-11-2016,CMB,CDG,20:55,06:30
ffff@ffff@fftf@ffff@
ffffff@ffffff@ffffff@ffffff@ ….
Ffffffff@ffffffff@ffffffff@ffffffff@ …….
*
QR 0041,30-11-2016,JFK,AKL,10:15,20:30
ffff@ffff@fftf@ffff@
ffffff@ffffff@ffffff@ffffff@ ….
ffffffff@ffffffff@ffffffff@ffffffff@ …….
**
EK,Emirates Airways
*
EK 1655,10-12-2016,JFK,CDG,06.00,12:30
ffff@ffff@fftf@ffff@
ffffff@ffffff@ffffff@ffffff@ ….
ffffffff@ffffffff@ffffffff@ffffffff@ …….
*
.
.
.
.
**
```

This symbol indicates the start of airport names

This is a single airport. This contains Airport ID, city and the country.
EX:
Airport ID = CMB
City = Colombo
Country = Sri lanka

This symbol indicates the end of airport names

This symbol indicates the start of an airline

This line contains the information of airline separated with commas
EX:
Airline ID = QR
Name = Qatar Airways

This symbol indicates the start of the information of a flight of the above airline

This line contains the basic information of the flight separated with commas
EX:
Flight Number = QR 0655
Departure Date = 29-11-2016
Departure Airport = CMB
Destination Airport = CDG
Departure Time = 20:55
Arrival time = 06:30
(Note: If the arrival time is before the departure time, it means, that the flight arrives on the next day.)

These 3 lines contains information of the seats in the above flight.
First line shows the seat arrangement of the First class seats of the flight. Each seat row ends with "@" symbol. "f" indicates that the seat is available, if the seat is booked values should be "t".

Second line shows the seat arrangement of the Business class and third line shows the arrangement of seats in the Economy class

This symbol indicates the start of information of the next flight of the above airline

This symbol indicates the end of the information of the first airline.

Guidelines for Documenting a Student Programming Assignment

Introduction

This guideline is aimed at a very typical programming assignment, where the classes of people interested are respectively the course lecturer, possible user of the program and the maintenance.

The assignment does not only include the program itself, but also the documentation of the program. The main purpose of program documentation is to describe the design, use and maintenance of your program, through the use of manuals, listings, diagrams etc. It is important that both the functionality and the implementation of a program are well illustrated for other people who might need this information – even a thoroughly commented source code will not be sufficient to describe larger, more complex software.

Each member in the group should document individual contribution clearly.

Documentation for any program falls into two categories: internal and external.

Internal documentation

The details of the program are explained by comments and placed within the code. Make use of Javadoc to generate useful documentation for Java code. Document while you're writing your program, not after you have it working correctly.

Internal documentation includes comments on classes, methods, and other places where appropriate.

Documentation about the "Classes":

When writing the code for a class in an object–oriented programming language, it should be preceded by a block comment minimally containing the following:

a. The class name, author name (name of member of the team), the names of any external packages upon which the class depends, the name of the package for the classes containing this class (if any), and the inheritance information (if any).
b. An explanation of the purpose of the class.

Documentation about "Methods"

This will include the method name, the purpose of the method, the method's pre– and post–conditions, the method's return value (if any), and a list of all parameters and their purposes.

Comments on other places

a. Each variable and constant (if meaningful identifiers are not used) must have a brief comment immediately after its declaration that explains its purpose.
b. Complex sections of the program that need some more explanations should have comments just before or embedded in those program sections.

External documentation

External documentation is typically written as a document separate from the program itself (use a word processor to prepare the document).

Good documentation should be complete, correct, concise, clear and structured. Don't write pages about your program (after all, a well laid-out, carefully constructed Java program should

be very readable). Most student assignments should only have a small number of pages of documentation.

General Format

This document is to be typed, 1.5 spaced, and with normal one-inch margins. The font should be Times New Roman size 12. The document must have correct spelling, be grammatically correct and use complete English sentences. These will be taken into consideration when your document is graded.

A document needs to be divided into sections and subsections, all with clear headings. If a document extends to several pages, it should have numbered pages and a contents list.

Documentation would consist of the following sections:

Title page:

This page includes the course code and name, title of the programming assignment, group name, name and index numbers of the students (members of the group), date submitted.

Introduction

This is a brief description of the purpose of the program and what it is designed to do. May include about the programming environment as well. State how you divided the problem among the team members and responsibilities of each member.

Interface (user manual)

Give the reader an overview of how the program works. It must be stated how the user will start running the program. It must also be stated how the user terminates the program. Explain the interactivity (command-line or GUI) of your system. Provide the inputs and the outputs of the program, the descriptions of menu options and different parts of the program, etc. State the error conditions, possible system or program messages, what to do about them. It is a good practice to include screen shots of your system. Divide this section into subsections for ease of reading.

Input and Output

The format of the input and output will be explained in this section. If the program also gets some input from input files and produces some output on output files, then the exact format of the files must also be given. It is advisable to include some example inputs and outputs, and their explanations.

Program Structure (maintenanance/technical manual)

An overall structure of the program should be given. Draw the UML class diagram to show the structure of the system. Give an overview of the algorithm used in the program. Include anything that is not immediately obvious. Any limitations and restrictions should be stated.

Improvements and Extensions

In this section, discuss the parts of the program that need improvement and some possible future extensions.

Problems encountered

State the problems/difficulties encounterd while doing the assignment. State the difficulties encountered due to the programming language, due to the nature of the problem, etc. Also, the effects of these difficulties (forcing to use different ways, abandoning some planned features of the program, etc.) should be explained.

Test Plan and Test Cases

Briefly describe how you planned to test each method and the overall program. You must show evidence of tests – this includes a statement of what you were trying to show, test data, expected results, and the observed results. A table of tests or other systematic approach can be a great help in presenting test results.

Conclusion

This section will contain your views, recommendations, etc. regarding the assignment. State the planned and actual schedule of progress. Comments on your own/team efforts, and on the assignment in general.