itinerary-prettifier

You'll create a command line tool which can prettify flight itineraries.

The situation

"Anywhere Holidays" is a brand new online travel agent, finding cheap holidays for their customers.

They are so new, that they have not yet built all of the systems they require to fully automate their processes. They have fully automated their end-to-end hotel booking system. But the flight booking system requires some intervention from back-office administrators.

When a customer buys a flight, a back-office administrator will use an online-portal to reserve the flight for the customer. That portal generates a text-based itinerary, which can be downloaded as a text file to the administrator's computer.

The main problem, is that the itinerary is formatted for administrators, and some of the information is not customer-friendly.

Of course the right thing to do is to fully automate this process, but this will take a significant effort. Administrators waste a lot time converting those itineraries to make them customer-friendly. And so it was decided to solve that very specific problem in the short term.

Functional requirements

Create a command line tool, which reads a text-based itinerary from a file (input), processes the text to make it customer-friendly, and writes the result to a new file (output).

Usage

The tool will be launched from the command line with three arguments:

1. Path to the input
2. Path to the output
3. Path to the airport lookup

$ java Prettifier.java ./input.txt ./output.txt ./airports_lookup.csv

Except for error handling, your program is not required to print any output.

You must implement a -h flag which displays the usage like so:

$ java Prettifier.java -h

itinerary usage:

$ java Prettifier.java ./input.txt ./output.txt ./airport-lookup.csv

Airport names

The tool will need to convert airport codes into airport names. The problem is, there's two types of code:

- IATA code is a three-letter code. It will be encoded with a single # followed by three letters. For example, #LAX represents "LAX" which is the IATA code for Los Angeles International Airport.
- ICAO code is a four-letter code. It will be encoded with double hash symbols (##) followed by four letters. For example, ##EGLL represents "EGLL" which is the ICAO code for London Heathrow Airport.

You'll be provided with an **airport lookup** CSV, which your program can use to extract airport names for a given airport code. Your tool can lookup the codes to reveal airport names at **runtime**. You must not hard-code this data into your program.

The order of columns will be: name, iso_country,municipality, icao_code, iata_code and coordinates. The first row will always be a header row. If any columns are missing, the data is considered to be malformed.

If any column contains blank data (empty cell), the data is considered to be malformed.

If an airport code is not found within the airport lookup, it should be left unchanged.

Dates and times

Dates and times are presented in ISO 8601 standard. While they are human readable, they are not customer friendly. The tool will need to change them into customer friendly dates and times.

Since we're dealing with flight times, they will **always** be expressed as date and time with a timezone offset. The input will describe whether to display a date or time as follows:

- **Dates:** D(2007-04-05T12:30−02:00). Dates must be displayed in the output as DD-Mmm-YYYY format. E.g. "05 Apr 2007"
- **12 Hour time:** T12(2007-04-05T12:30−02:00). These must be displayed as "12:30PM (-02:00)".
- **24 Hour time:** T24(2007-04-05T12:30−02:00). These must be displayed as "12:30 (-02:00)".

If the offset is "Z", it must be displayed as "(+00:00)". This is known as "Zulu time". You'll need to look up how it is formatted in ISO dates.

If the date is malformed, it should be left unchanged. For example:

- T13(2007-04-05T12:30−02:00) would be unchanged because "T13" does not describe 12 or 24 hour clock.
- T12(2007-04-05T12:30−2:00) would be left unchanged because the offset is malformed.

Trim vertical white space

Line-break characters (\v, \f, \r) must be converted to a new-line character: \n.

There should be no more than one consecutive blank lines in the output (no more than two consecutive new-line characters).

Some text
blank line
blank line
blank line
Some more text

Should be trimmed to:

Some text
blank line
Some more text

Error handling

If the incorrect number of arguments are provided, display the usage.

If the input does not exist, your program must display "Input not found".

If the airport lookup does not exist, your program must display "Airport lookup not found".

If the airport data in question is malformed within the airport lookup, your program must display "Airport lookup malformed". The data is considered malformed if any column in the lookup is missing or blank.

In any case of an error, the output file must not be created or overwritten.

Extra requirements

Use of formatting

Your program is not required to print any output (except for error handling).

You are welcome to additionally print the "output" to stdout. If so, use of colour and other formatting is encouraged.

It will be even more impressive if you highlight specific information like dates, times, offsets, airport names, cities etc.

City names

Implement city names with ICAO and/or IATA airport codes. There is a column for cities in the airport codes CSV.

When a city name is desired instead of an airport name, the encoding will be preceded by a * symbol, so that *#LHR is converted to "London" (For London Heathrow Airport).

Dynamic airport lookup column order

Add support for random column orders in the airport lookup table, to ensure that the program still works when some of the columns have been reordered.

Remember, the first row is a header row. So the column name can be used as a *key*.