Advanced Programming

Lab Sheet 13 – Handling CSV Data

Introduction

Comma Separated Values (CSV) files are a common, but simple, file format used to transfer simple tabular data between programs. Many pieces of software have the ability to import data from, or export data to, CSV files. If a CSV file is fairly small, it is even possible to view them in your preferred spreadsheet application, although there have been some famous cases of Microsoft Excel mangling CSV data with serious consequences.

A CSV file can be thought of as a spreadsheet or table, where each line of the file corresponds to one row of the table. The data in each cell of each row is separated by a single comma character. Thus, the data for a simple table might look like:

```
Size,Quantity
S,58
M,46
L,22
XL,11
XXL,3
```

Table 1: T-Shirt Stock Levels

Size	Quantity
S	58
M	46
L	22
XL	11
XXL	3

One common issue with data in the CSV format is obvious: what if we want to store data that itself contains a comma? The standard way of dealing with this is to enclose the field in quotes, with most software doing this automatically.

```
Numbers,Value
"Eins, Zwei",Polizei
"Drei, Vier",Grenadier
"Fünf, Sechs",Alte Hex
"Sieben, Acht",Gute Nacht
```

Table 2: Awesome 90's Dance Lyrics

Numbers	Value
Eins, Zwei	Polizei
Drei, Vier	Grenadier
Fünf, Sechs	Alte Hex
Sieben, Acht	Gute Nacht

This is only a partial solution, because the data might also contain quotes. The standard way of handling *this* scenario is to escape the inner quote with a backslash (i.e. \"). Fortunately, none of the data that you'll be working with in the lab task today had any commas or quotes, which keeps things simple.

In this lab session, you will be creating an energy price comparison program. Your program will work in a similar way to the popular uswitch website, where users enter the amount of electricity and gas that they use over the course of a month (or quarter, or year) and the site calculates an estimate of what they would pay on each tariff. Unfortunately, due to the current energy crisis and the prominence of Ofgem's price cap, there isn't a great deal of money to be saved by comparing providers. You don't have to use the real data and can edit it if you wish.

Gas and electricity bills are each made up of two key components: a *standing charge* and a *unit rate*. The standing charge is the cost (in pence) per day paid to the supplier just to have a working Gas/Electricity supply without using anything. The unit rate is the amount paid per kWh of electricity or gas. Customers who get both types of energy from the same supplier pay both standing charges, and the applicable unit rate for each type of energy used.

Task 1: Reading the CSV Data

I have gathered a small table of energy price tariffs from popular energy providers, saved it as a CSV file and made it available to you on moodle. You should download this data, and you may wish to take a quick look at it in both Excel and Notepad++ so you can get a feel for the structure.

Because reading data from a disk (even an SSD) is much slower than working from data in memory, you will design your program such that the data is read from the file just once on start-up, with the tariff information held in a suitable data structure. You will need to design a Tariff class to hold all of the information about a tariff, add properties for each of the table columns, add getters and setters, and declare an ArrayList to hold your Tariff objects.

To actually read the data into your ArrayList, you will need to use an I/O stream, similar to Task 2 in the previous lab session. You will need to read the file contents line by line, but you'll need to do some additional processing to separate the individual cell values. There are several ways to go about this, but I would recommend using the String class's .split() method.

The .split() method takes a *delimiter* as a parameter. A delimiter is a sequence of characters that specifies the boundary between things, and in the case of a CSV file the delimiter is a comma. The .split() method will return a String array, with all of the text up to the first comma in the string (excluding the comma itself) in the first array element, then everything up to the next comma in the second, and so on. In effect, this means that each cell in the row will become an element in the array. From there, you can create an instance of your Tariff object, set its properties, and store it in your ArrayList. You can check that the ArrayList is correct at the end of the file using the debugger.

Task 2: Estimating Bills

In order to make the program useful, it needs to ask the user for an estimate of their electricity and/or gas usage over the course of a year. Sensible figures to use for testing are 6600 kWh of Electricity and 16,500 kWh of Gas, but you will want to try different figures when checking everything works. Your program should allow people to estimate electricity alone, gas alone, or combined usage. Some don't have both types of energy.

You can calculate an estimate of what the user would spend annually on each of the tariffs by multiplying the applicable standing charges by 365 days, and by multiplying their gas and/or electricity usage by the applicable unit rates. These numbers can just then be added to provide the overall estimate. At this stage, your program can simply list the estimate for each tariff in turn.

To make the following task easier, I would recommend adding an .estimateCost() method to your Tariff class that will provide an estimate of cost based on two parameters: *electricityUsage* and *gasUsage*. You can choose a sentinel value (e.g. -1, or 0) to indicate that a given supply type is not required and to skip the standing charge. To test your program, try comparing the tariff costs for several different usage levels.

Task 3: Recommending the Cheapest Tariff

There are two ways to go about this task, and either is acceptable. You could choose to sort the ArrayList of tariffs according to their price estimate, in a manner similar to Task 2 in Lab 12. You can then just output the details of the (now) first tariff in the list. Alternatively, you could simply loop through the list of tariffs and keep a variable updated with the cheapest one seen so far, and use that to generate your output.

Extension Task: Billing Discounts

Some energy providers, on some tariffs, offer a discount to customers who elect not to receive a paper bill, who elect to pay by Direct Debit, or to customers over the age of 65. You could add some additional columns to the CSV data to indicate which tariffs offer which discount (which might be a fixed amount, or a percentage), and prompt the user to answer the appropriate questions before calculating the cost estimates.