CSE1322L Assignment 1 - Fall 2024

Introduction:

You probably used emojis while using your cellphone or your computer. They are those tiny pictures that you can embed into your text to add context to your words.

While it may not be immediately apparent, from the computer's perspective, emojis work like any other character such as the ones on your keyboard (though emojis are usually a combination of two or more "regular" characters). As such, if you knew the codepoint associated with a particular emoji, you could type that emoji into the text you are writing. Fortunately, you don't have to remember any codepoints: your device keeps a complete list of all available emojis.

The Unicode Consortium maintains the Technical Standard for emojis which currently lists thousands of emojis, with new ones being added every other year. A problem with that is that screen space is limited, so you cannot display all of them at once; even if you could, searching for the specific one that you want to use every time would be a chore. Fortunately, our devices keep track of the emojis we've used the most recently, keeping a shortlist of said emojis at the top of the general list.

In this assignment, we will write a simple program which does something similar, albeit with a much smaller range of emojis.

Requirements

The features described below must be in your program.

- A total of two classes: the driver, and Symbol
- The Symbol class keeps track of the information of a particular symbol:
    - Must have 3 fields:
        - A public character field to keep track of what symbol it stores, called "symbol".
        - A public integer to keep track of how many times that symbol has been used, called "uses".
        - A public double to keep track of that symbol's frequency, called "frequency".
    - It must have 1 constructor:
        - An overloaded constructor which takes in one character as its parameter, assigning that character to the "symbol" field. It then sets the "uses" and "frequency" fields to 0.
- The driver must contain the following:
    - A static method called "updateFrequencies()"
        - Takes in a Symbol array as input and returns no output.
        - It first counts the total number of times all symbols have been used. It then updates their individual frequencies by dividing their individual use by the total number of uses across all symbols.

        - E.g.: If we have the 3 symbols below, and their respective uses:
            - + used 5 times.
            - - used 2 times.
            - * used 9 times.
        - Given that they add to a total use of 16, their frequencies will be:
            - + 0.3125 (5/16)
            - - 0.125 (2/16)
            - * 0.5625 (9/16)
    - A <u>static</u> method called “sortSymbols()”
        - Takes in a Symbol array as input and returns no output.
        - It sorts the symbols by descending frequency (i.e.: the symbol with the highest frequency must be in the 0-th index and the symbol with the lowest frequency must be in the last index).
            - In the previous example, the symbols would have been sorted as such:
                - * (0.5625)
                - + (0.3125)
                - – (0.125)
        - If two symbols have the same frequency, they must be sorted by ascending codepoint value (i.e.: a symbol with a lower codepoint value must come before a symbol with a higher codepoint value).
            - See the table below to determine the ordering of the symbols we’ll be using. The table is already sorted in ascending order by its codepoints.
        - **<u>You may not use any built-in sorting methods or libraries. You must write the sorting algorithm yourself.</u>**
            - You can find the Bubble Sort algorithm in the slides for 1321L, under Module 5 part 1.
    - The main method.
        - Creates an array of Symbols (called “symbols”) of size 9 and fills it up with 9 Symbol objects using their overloaded constructor, with each Symbol object using one of the characters in the order they appear below:

Symbol	Codepoint
∞	\u221E
☺	\u263A
♀	\u2640
♂	\u2642
♠	\u2660
♣	\u2663
♥	\u2665
♦	\u2666
♫	\u266B

            - You can simply type in the codepoint above between single quotes in your source code. Your computer will convert it to the appropriate character once you try to run your code.
                - char c1 = '\u221E';
            - Don't worry if your computer has trouble printing the characters correctly. As long as your code is using the correct codepoint, you will get full credit.
    - Write a loop which does the following:
        - Prints options 1 through 0.
            - Options 1 through 9 should all be symbols, sorted top to bottom according to sortSymbols() (i.e. sorted by frequency, followed by codepoint).
            - Option 0 should be "Exit".
        - Reads a number from the user.
        - If the user selects 0, it should terminate the program.
        - If the user selects a valid option:
            - Print the symbol associated with that option.
            - Increment that symbol's "use" by 1.
            - Call updateFrequencies(), passing "symbols" as its parameter.
            - Call sortSymbols(), passing "symbols" as its parameter.
        - If the user selects an invalid option, print an error message.

Considerations

- This assignment may seem intimidating, but that's just because of the number of things you have to do; the assignment itself isn't very hard, so don't be discouraged.
- Remember that you will get partial credit for partial work. Try to deliver as much of the assignment as you can.
- The fields in your Symbol class must all have the "public" keyword before their type keyword. In general, object fields will be private while object methods will be public, unless specifically stated to the contrary.
- Some of the methods in the driver must have the "static" keywork. We'll talk more about them at a future module.
- Recall an array of objects works just like any other array. The only difference is that it stores a complex data type.
- Your array of Symbols should have 9 Symbols, each with a different character in their "symbol" field, as per the table above.
- Remember that the symbol used the most must be at the top of your options menu, as well as at the start of your array.
- Comparing two characters using any comparison operator will tell you their ordering (e.g.: ∞ < ♠ == true, ♫ < ☺ == false, a < b == true, c < b == false)
- **Remember that you must write the sorting code yourself (i.e.: you cannot use built-in or imported methods)**.

    - We provide code for Bubble Sort in the slides for 1321L, Module 5 Part 1. It is written in Python, but you should have no trouble converting it to your specific language.
    - Any sorting algorithm will do, but you are strongly encouraged to use a common one, so grading can be done more easily.
- You may add any other helper methods you believe are necessary, but they won't count towards your grade.

Example: [User input in red]

[Symbol Recommender]
Here are all available symbols
1 - ∞
2 - ☺
3 - ♀
4 - ♂
5 - ♠
6 - ♣
7 - ♥
8 - ♦
9 - ♫
0 - Exit
Please select a symbol to print: 9

You selected the ♫ symbol.

Here are all available symbols
1 - ♫
2 - ∞
3 - ☺
4 - ♀
5 - ♂
6 - ♠
7 - ♣
8 - ♥
9 - ♦
0 - Exit
Please select a symbol to print: 6

You selected the ♠ symbol.

Here are all available symbols
1 - ♠
2 - ♫

3 - ∞
4 - ☺
5 - ♀
6 - ♂
7 - ♣
8 - ♥
9 - ♦
0 - Exit
Please select a symbol to print: 1

You selected the ♠ symbol.

Here are all available symbols
1 - ♠
2 - ♫
3 - ∞
4 - ☺
5 - ♀
6 - ♂
7 - ♣
8 - ♥
9 - ♦
0 - Exit
Please select a symbol to print: 4

You selected the ☺ symbol.

Here are all available symbols
1 - ♠
2 - ☺
3 - ♫
4 - ∞
5 - ♀
6 - ♂
7 - ♣
8 - ♥
9 - ♦
0 - Exit
Please select a symbol to print: -1

Invalid option!

Here are all available symbols
1 - ♠

2 - ☺
3 - ♫
4 - ∞
5 - ♀
6 - ♂
7 - ♣
8 - ♥
9 - ♦
0 - Exit
Please select a symbol to print: 10

Invalid option!

Here are all available symbols
1 - ♠
2 - ☺
3 - ♫
4 - ∞
5 - ♀
6 - ♂
7 - ♣
8 - ♥
9 - ♦
0 - Exit
Please select a symbol to print: 6

You selected the ♂ symbol.

Here are all available symbols
1 - ♠
2 - ☺
3 - ♂
4 - ♫
5 - ∞
6 - ♀
7 - ♣
8 - ♥
9 - ♦
0 - Exit
Please select a symbol to print: 0
Shutting off...

Submitting your answer:

Please follow the posted submission guidelines here:
https://ccse.kennesaw.edu/fye/submissionguidelines.php

Ensure you submit before the deadline listed on the lab schedule for CSE1322L here:
https://www.kennesaw.edu/ccse/first-year-experience/cse-1322-lab.php