CS/SE 4352.0U1
Team Send Help

Project Phase I Final

Fareen Chowdhury
Fxc150930

Ross McNew (Team Leader)
rlm160030

Midhat Wahab
maw160030

http://www.utdallas.edu/~rlm160030/search_engine.html
https://github.com/fareench/CS_4352_Project_1

i. Project Organization

Phase I

Meetings occurred after class each day, twice a week. About an hour spent discussing strategies and plans.

Dates:

5/28 (Fareen & Midhat)	5/30 (Fareen & Midhat)
6/4 (Fareen & Midhat)	6/6 (Fareen & Midhat)
6/11 (Fareen, Ross, Midhat)	6/13 (Fareen, Ross, Midhat)

1.Introduction

The program should allow the user to perform the circular and alphabetical shift function on input data. The program should allow the user to enter the input into a text box or text bar that is designed to accept data in order to perform the required functions on it. The KWIC (Key Word in Context) index system shall accept no more than 1 ordered line, where each line is an ordered set of no more than 30 words, and each word is an ordered set of characters. The number of characters shall not exceed over 300 characters per line. Each line shall be “circularly shifted” by repeatedly removing the first word and appending it at the end of the phrase. Once the circular shift function has been performed, the program must output the results on the screen step by step for the user to view. The data should also be carried over to the alphabetical function. Once the alphabetical function is performed, the KWIC index system shall output a listing of all lines in ascending alphabetical order. The program should then provide the option for the user to restart the program or terminate it.

2.2 Non-Functional Requirement

Constraints: The code should be easily understood by other developers in order to maintain reusability and enhanceability. The user interface of the program must be easily understandable and navigable, in order to be user-friendly. The user shall perform no more than two tasks after starting the program, i.e. typing in the input and clicking enter. The program must also be adaptable and portable in order to allow different devices to access the program's capabilities.

External Interface: The program must receive the user's input from the keyboard, whether it is a physical keyboard connected to a PC or the keyboard from the user's smartphone. The results must also be displayed on the screen of the device the user is using. The option to restart or terminate must also be displayed on the screen and allow for the user to navigate, whether by touchscreen or mouse, to choose which option they'd like.

Performance: The program should respond quickly and efficiently, providing accurate results within 3 seconds. The results from each function should also not take much memory space and be removed after the user decides to terminate or restart the application.

3.Architectural Specification

3.1 The 3C's

Components: The program uses several different components to perform the circular and alphabetical shift. The following are the components or objects that the program depends on:

Input: Receives data from the user through the keyboard/screen interface

Master Control: Will be used to initiate the program.

Output: Will be used to display results onto the screen interface

Line Storage: Will be used to store the data input in proper structure

Circular Shift: Will perform the circular shift function on direct input into an ordered list

Alphabetizer: Will alphabetize on direct input into a sorted list

Connections: There exists a connection between all components of the program.

Connection 1: Master Control will call upon Input's setup function to begin the program

Connection 2: Input will wait for data from the input medium

Connection 3: Master Control will call upon Line Storage setup function

Connection 4: Line Storage will receive data from Input and store each line and each word

Connection 5: Master Control will call upon Circular Shift's setup function

Connection 6: Circular Shift will receive its input from Line Storage, perform the circular shift function and store each character, word, and line.

Connection 7: Master Control will call upon Alphabetizer's setup function

Connection 8: Alphabetizer will call upon circular shift to receive the input required to perform an alphabetize function.

Connection 9: Master Control will call upon Output's setup and print function.

Connection 10: Output will call upon data from Circular Shift and from Alphabetizer

Connection 11: Output will send the data, organized, to the output medium.

Constraints:

Several constraints apply to the components of the program:

1) The program cannot accept more than 300 characters in one run. Since it is a low level program with a small size array and in order to allow the program to meet the performance criteria from non-functional requirements, it is necessary to limit the number of characters that will be read into the program.
2) The program cannot accept more than 30 words in one run. Since it's a small program the same restrictions that apply to the character limit applies to the word limit.
3) The program will accept 1 sentence at a time. The program is designed to apply the functions after 1 full sentence is entered. Once the results are displayed the program will prompt the user for a new sentence. Multiple sentences cannot be read into the program simultaneously.

3.2 Parameters

Some of the parameters that will be used in the program are:

Number of parameters: 3

Char input : for the characters that will be read into the program. The program will then parse the characters into words using buffer and a loop.

String word: for the words that will be used for the function. The program will use parse information and read it in a separate parameter and use it to display the original sentence and will be used for the function.

Integer counter: will keep track of the number of words read in to make sure it does not exceed the word limit constraint the program holds.

3.3 Rationale

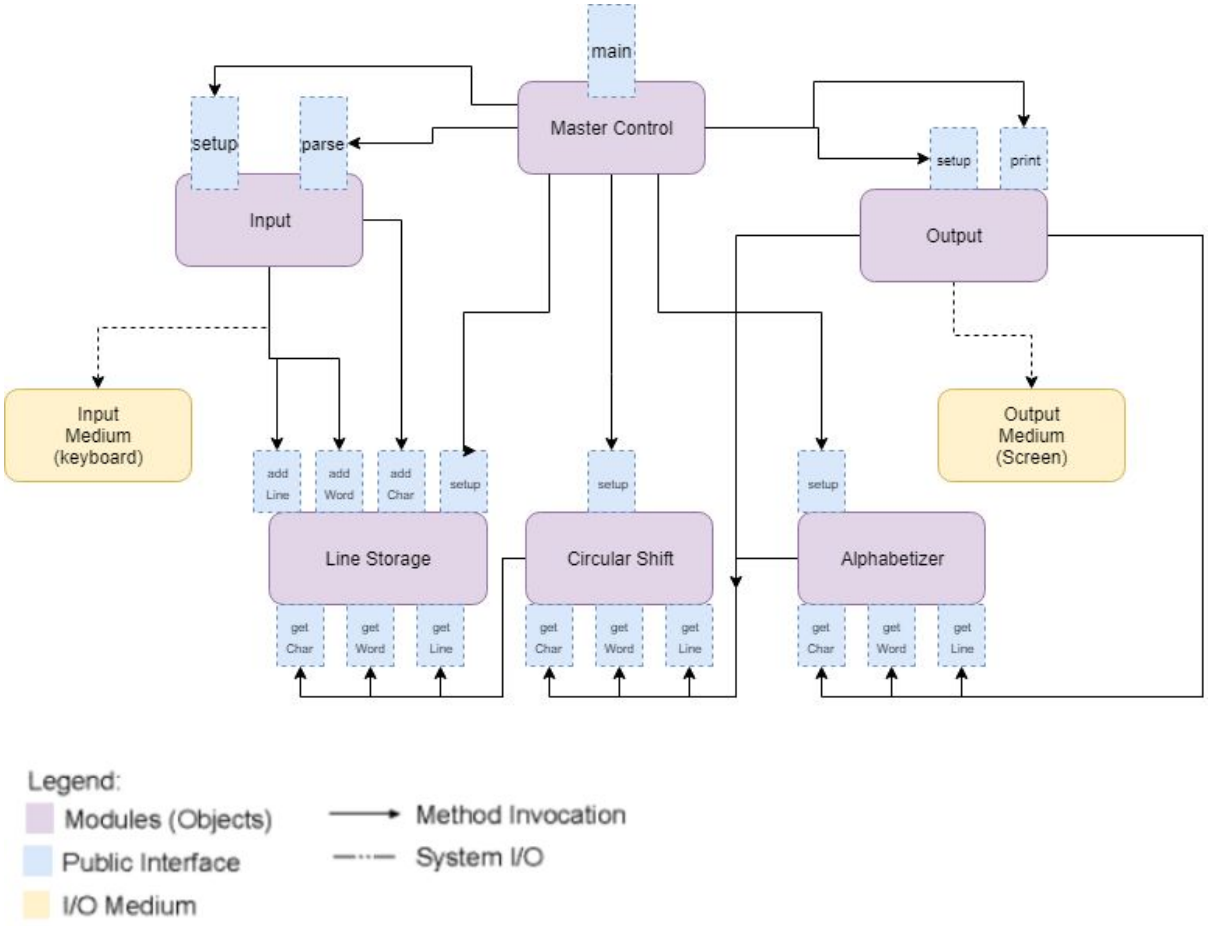
Advantage:

- The program will have fast performance because it is taking a limited number of characters and words.
- Since the program only takes 1 sentence at a time it is extremely straightforward and user friendly for the user. The users don't need additional instructions on how to enter multiple sentences since the program will be reading only one sentence at a time.
- The program requires the user to perform only 1 task which is to enter the sentence. This makes the program very adaptable.

Disadvantages:

- The program has several constraints regarding the input that user needs to be aware of before using the program. The constraints are that user must meet the character, word, and sentence limitations.
- The program does not end automatically, user is required to enter "end" or exit out of the program to end the program.

4. Diagram



5. Java Applet Specification

Using an Object Oriented model, the connections to be made are:
[MasterControl] calls -> [Input].setup()
[MasterControl] calls -> [LineStorage].setup()
[MasterControl] calls -> [CircularShifter].setup()
[MasterControl] calls -> [Alphabetizer].setup()
[MasterControl] calls -> [Output].setup()
[MasterControl] calls -> [Input].parse(Input Medium)
[MasterControl] calls -> [Output].print(Output Medium)

[Input] calls -> [LineStorage].addChar()
[Input] calls -> [LineStorage].addWord()
[Input] calls -> [LineStorage].addLine()

[CircularShifter] calls -> [LineStorage].getChar()
[CircularShifter] calls -> [LineStorage].getWord()
[CircularShifter] calls -> [LineStorage].getLine()

[Alphabetizer] calls -> [CircularShifter].getChar()
[Alphabetizer] calls -> [CircularShifter].getWord()
[Alphabetizer] calls -> [CircularShifter].getLine()

[Output] calls -> [CircularShifter].getChar()
[Output] calls -> [CircularShifter].getWord()
[Output] calls -> [CircularShifter].getLine()
[Output] calls -> [Alphabetizer].getChar()
[Output] calls -> [Alphabetizer].getWord()
[Output] calls -> [Alphabetizer].getLine()

Class Diagrams:

-Variables and Methods for each .js Object
Objects are listed in [] and each section before the following object represents the variables and methods for that object.
Variables are listed inside {}
Methods are listed directly on their own line each below variables

[MasterControl]

public main(): primary function called when program is run externally.

[Input]
public setup(): initialize object.

public parse(Input Medium): parse input given by user from (Input Medium).

[LineStorage]
{private lineArray : ArrayList}

public setup(): initialize object.

public setChar(c:char, position:int, word:int, line:int): set character in specified word in specified line in object.
public setWord(w:String, position:int, word:int, line:int): set word in specified line in object.
public setLine(l:String[], position:int, word:int, line:int): set line in object.

public addChar(c:char, word:int, line:int): append a char to a specified word in a specified line from the object.
public addWord(w:String, word:int, line:int): append a word to a specified line from the object.
public addLine(l:String[], word:int, line:int): append a line to the object.

public getChar(out c:char, position:int, word:int, line:int): retrieve a char in specified word in specified line from the object.
public getWord(out w:String, word:int, line:int): retrieve a word in specified line from the object.
public getLine(out l:String[], line:int): retrieve a line from the object.

public deleteChar(position:int, word:int, line:int): remove a char in specified word in specified line from the object.
public deleteWord(word:int, line:int): remove a word in specified line from the object.
public deleteLine(line:int): remove a line from the object.

[CircularShifter]
{private lineShifts : LineStorage}

public setup(): initialize object.

public getChar(out c:char, position:int, word:int, line:int): retrieve a char in specified word in specified line from the object.
public getWord(out w:String, word:int, line:int): retrieve a word in specified line from the object.
public getLine(out l:String[], line:int): retrieve a line from the object.

[Alphabetizer]

{private shift : CircularShifter}
{private sortIndex : int[]}

public setup(): initialize object.

public getLine(out l:String[], line:int): retrieve a line from the object.

[Output]
public setup(): initialize object.

public print(): print to (Output Medium).

-Comparison of OO model versus data-sharing model

-Comparison of default image OO model versus completely specified model

Our project, while still in the developmental phase, is the product of the team members' hard work and intense learning phase. With last minute member changes and some inexperience with Javascript, our team worked hard on delivering the final project while learning as much as possible. We are proud of our work and will continue to work and improve on both our skills and our product.