**Assignment 1: KWIC-KWAC-KWOC**

|  |
| --- |
| Code Repository URL: https://github.com/HyungJon/KWIC |

|  |  |  |
| --- | --- | --- |
| Name | Kim Hyung Jon |  |
| Matric number | A0115864B |  |

1. **Introduction**

This software was developed with user-friendliness, speed and efficiency, and extendibility in mind. It was also important that the methods and the overall work scenario is easy to understand even without the comments.

**2.1 Design**

The program runs by executing the UI class, a common component for both approaches. Upon execution, UI displays a simple GUI containing 4 main parts: two text fields for specifying input and output file paths, and two buttons for selecting the method to use.

Both methods require that the input file contains the data in the following format: the first line contains all words to ignore, and each of the following lines contains one title.

e.g.

is the of and as a after

The Day after Tomorrow

Fast and Furious

Man of Steel

The UI was implemented by Hyung Jon.

**2.1 Method 1: Shared Data**

Method 1 by Jerrold

**2.2 Method 2: Abstract Data Types**

Method 2 implemented by Hyung Jon follows the Solution 2 in Reading 2: abstract data types with slight modifications for simplicity and speed. It consists of the following components:

**Controller**

Master control in charge of invoking other components in the correct order with correct parameters.

**InputReader**

Reads the input file and stores the data line by line, without knowledge regarding what each line is.

Offers the method getLine() for LineProcessor to retrieve a line at specified index.

**LineProcessor**

Takes all lines stored in InputReader and converts them to a form usable by CircularShifter. It constructs a list of words to ignore from the first line, and constructs a list of all titles.

Rather than keeping a 2D list to store each title as separate words, it contains the method getTitle() that converts the title in string form to an arraylist of words, to be used by CircularShifter.

**CircularShifter**

Constructs all circular shifts, not starting with words to ignore and with first word in upper case, from each title obtained from LineProcessor and stores them. Contains getCircularShifts() method that Alphabetizer can call in order to retrieve all constructed circular shifts.

**Alphabetizer**

Sorts and stores the circular shifts in the alphabetical order.

**OutputWriter**

Writes all sorted circular shifts to the output file specified by user and opens the output file.

**2.2.2 Benefits**

**Modifiability**

The software is highly modularized, each component in charge of a specific purpose and given only the knowledge and methods relevant to that purpose. Therefore, modifying one module does not require the knowledge regarding any other module.

**Extendibility**

Data is stored separately for each stage of the process, and each stage offers separate methods to access the data produced at that stage. This makes it easier to add additional components or functions that change the data.

A good example of this is the function to edit the list of words to ignore. In shared repository, this would require another function to update the input file, and the entire process will have to be re-executed. On the other hand, editing the data of words to ignore in the abstract data approach requires modification in LineProcessor only, and only the components starting from CircularShifter need to be executed to produce the new outcome.

**2.2.3 Drawbacks**

**Extendibility**

The modularization makes it difficult make major modifications involving more than one module, such as extending the software by adding another module to the process. This means to add a component A between CircularShifter and Alphabetizer, both components must be modified so that CircularShifter converts the data into a form usable by A and Alphabetizer accepts data processed by A.

**Data storage**

Unlike the core data approach where all components work on a shared repository, this approach requires each component to store the output data separately, thus requiring more data space.