Assignment 1: KWIC-KWAC-KWOC

Code Repository URL: https://github.com/Yanqing93/KWIC

Name	Zhang Yanqing	Nicholas Chew
Matriculation Number	A0100985E	A0101885E

1. Introduction

The KWIC (Key Word in Context) index system accepts an ordered set of lines, each line is an ordered set of words, and each word is an ordered set of characters. Any line may be "circularly shifted" by repeatedly removing the first word and appending it at the end of the line. The KWIC index system outputs a listing of all circular shifts of all lines in alphabetical order.

2. Design

Two designs are implemented for this project: Main program with Shared Data, and Abstract Data Type.

Design 1: Main Program with Shared Data

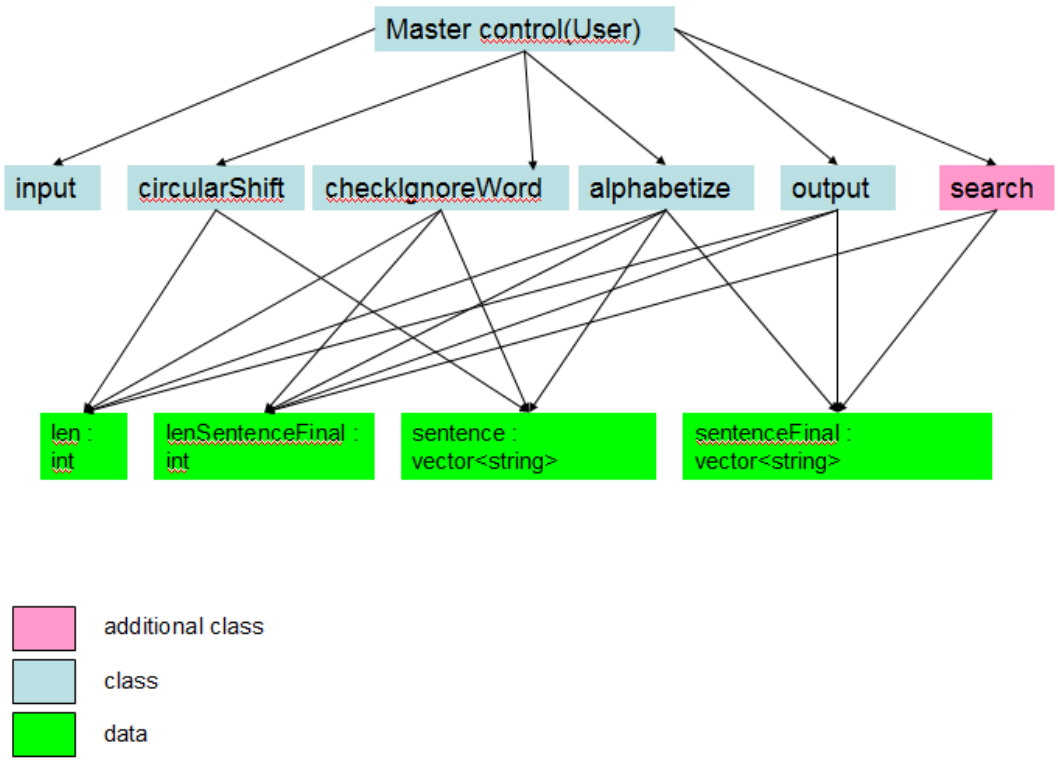


Fig 1: KWIC: shared data solution

There are seven main classes in this program:

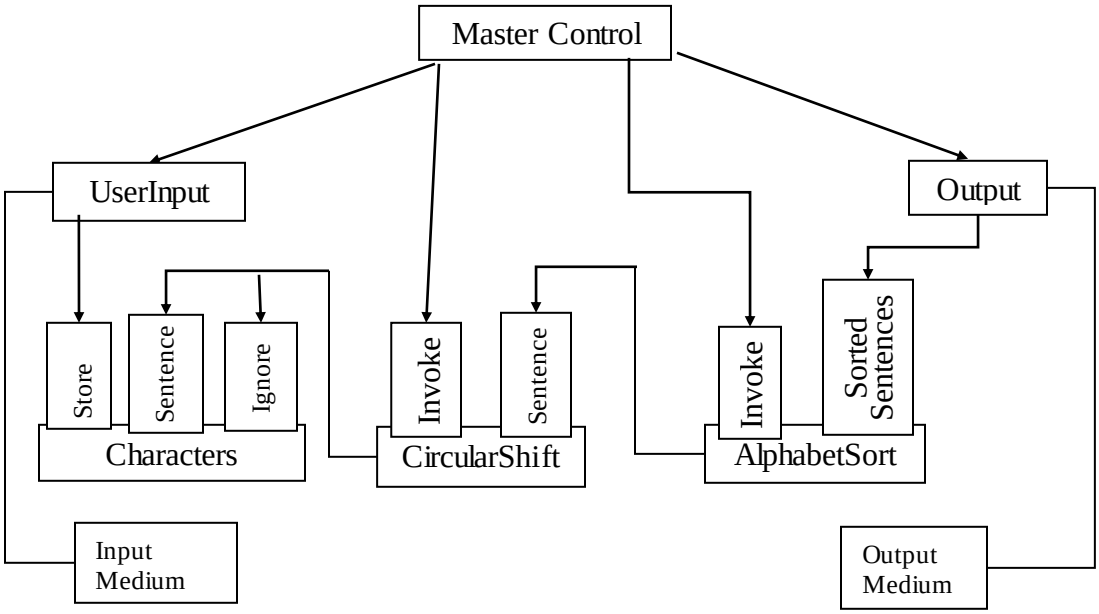
1. Input

input module reads the input from the user
2. circularShift
module exhaustively remove the first word and appending it at the end of the line to create a set of circularly shifted line
3. checkIgnoreWord
checkIgnoreWord will check the entire data for any “words to ignore” as the first word, if there is will be removed
4. alphabetize
Alphabetize module uses the selection sort to sort the data alphabetically
5. output
Output module prints all the data to the user
6. search
Search module will search the data for any sentence which contains the key provided by the user

Limitation & Benefit of Shared Data solution

The data can be processed efficiently as data is shared among all the computation. This solution is natural and intuitive too. However, a change in data type or container will require a modification in all of the classes. Furthermore, it is no easily reusable due to its tight coupling.

Design 2: Abstract Data Type (ADT)



There are 6 main classes in this implementation:

1. ADT_Main
   This serves as the main program for ADT implementation.
2. UserInput
   Reads inputs from user and store data into Characters.
3. Characters
   Stores data read from UserInput.
4. CircularShift
   Shifts sentences and put keywords at the first index, filtering ignore words at the same time.
5. AlphabetSort
   Sorts shifted sentences in ascending alphabetical order.
6. Output
   Outputs sorted results onto console.

Limitation & Benefit of ADT solution

Benefit:
This implementation supports reuse, without necessarily compromising performance, because modules make fewer assumptions about the other modules they interact with.
Algorithms and data representations can be changed in individual modules without affecting the other modules.

Limitation:
Interactions between components in ADT are fixed, so changing the overall processing algorithm or adding new functions may involve large number of changes to the existing system.

The solution is not particularly well-suited to enhancements as well. To add new functions to the system, the implementation must be modified in the existing modules, compromising integrity and simplicity in the process, or add new modules which will lead to reduction in performance.