Assignment 1: KWIC CS3219 Sem 1 2016/2017

Code Repository: https://github.com/junhaoyap/CS3219-jh-leo-assg1/

Student Name	Yap Jun Hao	Leonardo Sjahputra
Matriculation No	A0113694A	A0114088H

1. Introduction

Our assignment is about KWIC - Key Word In Context indexing system. We are to implement a KWIC system with a list of lines as well as “words to ignore” as inputs. The resulting KWIC-index will list a line for each keyword that occurs in the line, alphabetized. These keywords will ignore the words in the “words to ignore” list.

2. Requirements

- The output has to include all keywords
- The output cannot have keywords that is in the “words to ignore” list
- The output should be a listing of the circularly shifted lines for all input lines in ascending alphabetical order
- The non-keywords (words in “words to ignore” list) should not be capitalized, the keywords should be output with its starting letter capitalized and the rest in lower case (case is only considered when shifting - check section 5 on some assumptions and technicalities)
- Easy to use interface
- Reasonable response time
- Extendable (new functions can be added easily)

3. Architectural Designs

Selected Design 1 - Main Program / Subroutine with Shared Data

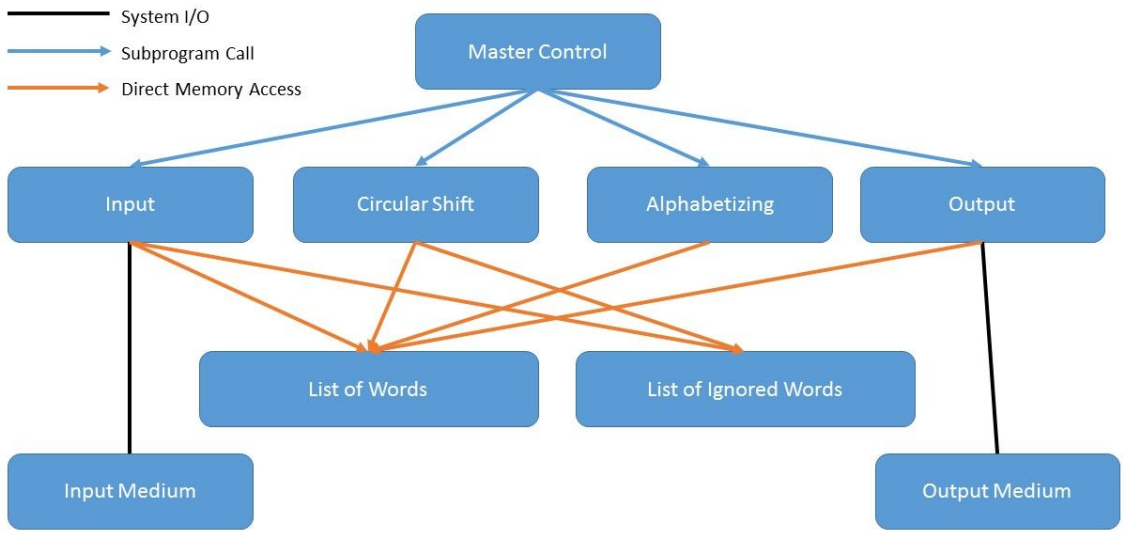

Image 1: Architecture design for selected design 1

Selected Design 2 - Abstract Data Types (Modified - without Characters module as Java String and Character is powerful enough by themselves to act as the module)

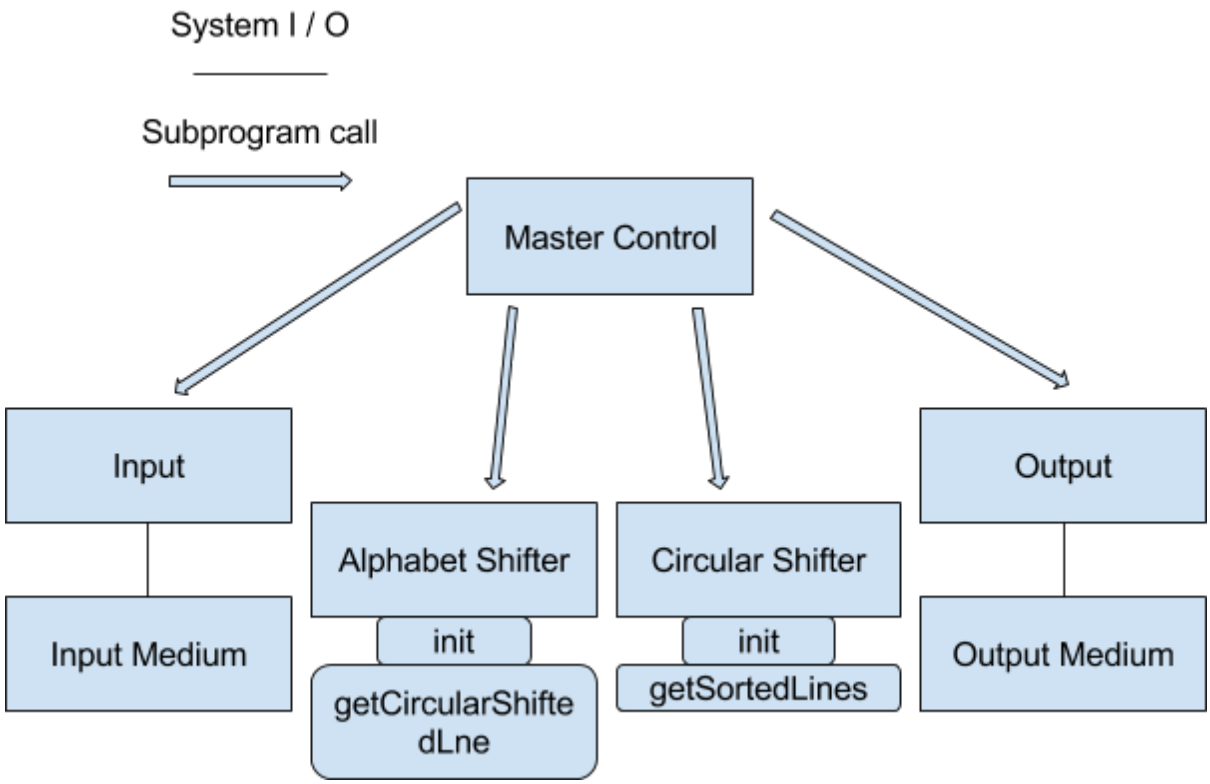

Image 2: Architecture design for selected design 2 (Modified)

4. Limitation & Benefits of Selected Designs

For design 1:

Benefit(s)

- Highly intuitive
- Efficient use and representative of data
- As it is modular, it is relatively easy to add new functions
- It is quick performance wise due to the shared data

Limitation(s)

- Does not support robust data format; a change in data format will affect a large part of the code
- The dependency on shared data does not fully support reusing
- Changes in overall processing algorithm cannot be easily accommodated

For design 2:

Benefit(s)

- Algorithmic changes can be made with changes in the individual modules without affecting the other modules
- Data representations can be made with changes in the individual modules without affecting the other modules
- Modules make less assumptions about each other, resulting in better reuse

Limitation(s)

- Not as extensible, programmer has to either modify the existing modules and make them more complicated / make more assumptions about other modules or add new modules which either complicates the whole program or result in lower performance

5. Some assumptions / technicalities of implemented program

- We do not consider duplicates, if there are duplicates we accept them and print them anyway at the end
- We do not consider symbols

- We ignore casing when checking for words to ignore
- We only consider the capitalization of words in the line when building them for output / when shifting