Assignment 1: KWIC

Code Repository URL:

https://github.com/ruiwen905/CS3219A1.git

Name	**Chen Rui Wen**	**Seow Wei Jie**
Matriculation Number	**A0138431L**	**A0139515A**

1. Introduction

Our program is about the Key Word In Context (KWIC) indexing system. This program takes in a input text file which contains a list of lines and also a text file which contains the list of words to ignore. Each line will be circularly shifted and appended at the end of the line. First word (not including “words to ignore”) will be the keyword. The output will be a text file that contains all the circularly shifted lines arranged in alphabetical order.

We have two architecture designs - Abstract Data Type (done by Seow Wei Jie) and Pipe and Filter (done by Chen Rui Wen).

2. Requirements

Functional Requirements:

1) (Pipe and Filter) Developers should be able to add new functions by adding new filters without modifying other filters.
2) (ADT) Developers should be able to add new functions without modifying major modules of the programs.
3) The output must include all the keyword specified by the user.
4) The program will be able to produce the correct output even when the user modifies the input and words to ignore files.
5) The program is case-sensitive and will output the keyword in the user specified format.
6) User will be able to specify which input files they want to feed into the program and choose their desired output file.

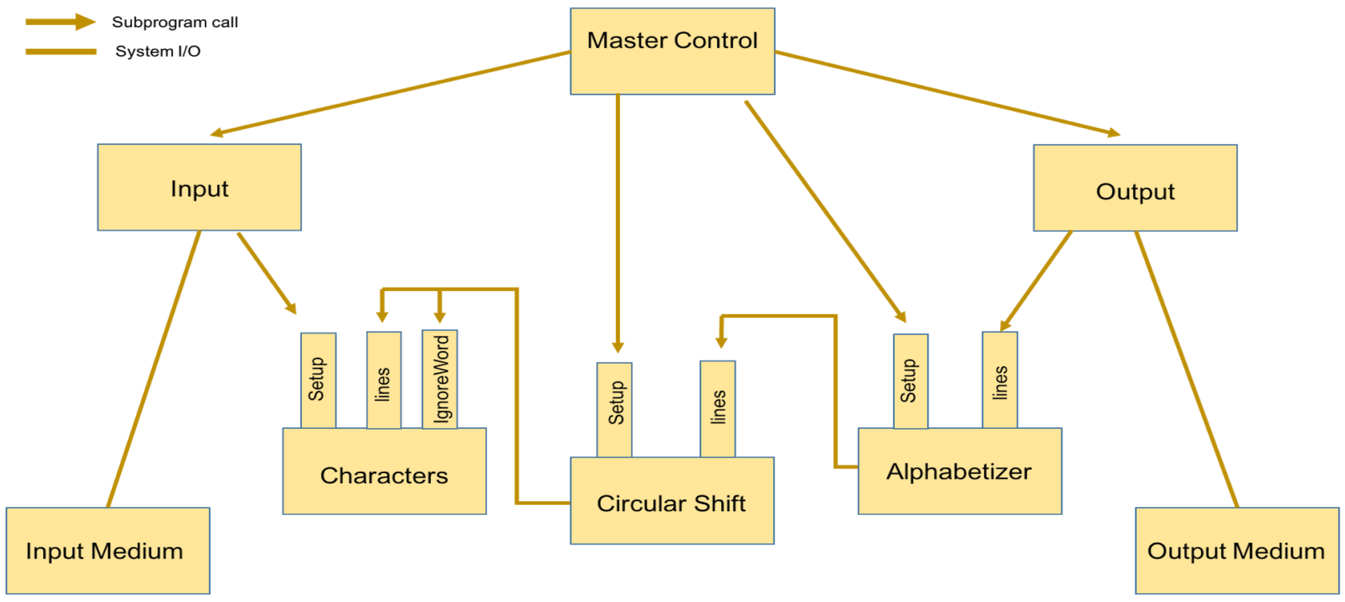
Non-functional Requirements:

1) The user should receive the output file within 1 second.
2) The program should be able to take in as least 1000 lines.
3) The user should not require any user manual to be able to use the program.
4) The program should work in any computer with java installed.
5) The program should be easily extendible so that algorithms like circular shift can be easily modified or other algorithms can be added without affecting other major components.
6) The program should not crash when the files input by the user are large.

3. Architectural Design

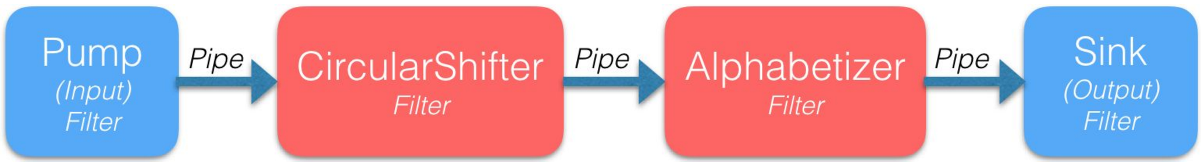
Design 1: Abstract Data Type (ADT)

ADT is a object-oriented design and data is not shared directly within modules. Also, each module has its own interface and it determines how other modules can access its functions.



1) Master Control - controls the flow of the program and handle exceptions
2) Input - take in input files and set up character module
3) Characters - storage of the lines
4) Circular Shift - perform circular shift on the lines and allow other modules to retrieve circularly shifted lines
5) Alphabetizer - sort the lines in alphabetical order
6) Output - output the ordered lines in a text file

Design 2: Pipe and Filter



The filters are independent of each other and they can only interact through pipes. Pipes act as data streams that controls the sequential data flow and transformation.

1) KWIC - Set up all filters and filter wrapper to start the program
2) Filter - Abstract class used by all the filters to read and write word
3) Filter Wrapper - Set up the pipes to be (number of filters - 1)
4) Pipes - Contains list of input words and list of words to ignore
5) Pump(Input) - take in input files and set up character module
6) Characters - storage of the lines
7) Circular Shifter - circular shift on the lines and allow other modules to retrieve circularly shifted lines
8) Alphabetizer - sort the lines in alphabetical order
9) Sink(Output) - output the ordered lines in a text file

4. Limitation & Benefits of Selected Designs

State limitation and benefits of the design you implemented.

Benefits of ADT:

1) Both algorithms and data representations can be changed in individual modules without affecting others.
2) Better support for reuse. E.g. input module can be reused for any procedure that requires file input.
3) Provides encapsulation and lesser assumptions are made about other modules.

Cons of ADT:

1) Design cannot be extended without modifying the current modules as new functions will have to be added to existing modules. This in turn can affects the modules by making it more complicated than before.
2) Since ADT is object-oriented, object must know the identity of the other objects for it to use their interface.
3) Consume more space as information are duplicated when objects are created.

Benefits of Pipe and Filter:

1) Pipes and filters allows objects of slow filters to run in parallel, enabling the system to spread the load and improve throughput.
2) Since the filters does not depend on each other, failure in one filter does not necessarily result in failure of the entire system. In addition, the speed of processing can be boost by distributing the filters across different servers.
3) Pipe and filters design is also very flexible and versatile as developers can add new functions by simply adding new filters without changing other parts of the program.

Cons of Pipe and Filter:

1) Batch-oriented processing
2) Performance may be compromised as the use of a common pipe may force a lowest common denominator on data transmission and format such as vague parses and latency.
3) Stateless data transformation, hence not suitable for interactive system that requires cache memory.

5. Any other information

Assumptions:

- Words to ignore are all in lower cases.
- There is only one blank space in between each word.
- Files are in .txt format

Testing methods:

- Manual testing by using text files that contains all the lines.