Assignment 1: KWIC **CS3219 SEM1 2016/17**

Code Repository URL: https://github.com/CS3219-Assignments/Assignment-1-KWIC

Student Name	Tang Wei Ren	Razali
Matriculation Number	A0125531R	A0133267H

1. Introduction

KWIC (Key Word In Context) sorts and aligns words within a *title* to allow each word in the *title* to be alphabetically indexed. A new line is indexed for each *keyword* found in each sentence. Users are allowed to specify *noise words,* which will not become *keywords*. *Keywords* would retain their input format but with their first letter capitalised while *noise words* will have all of their letters lower-cased.

2. Requirements

Functional Requirements

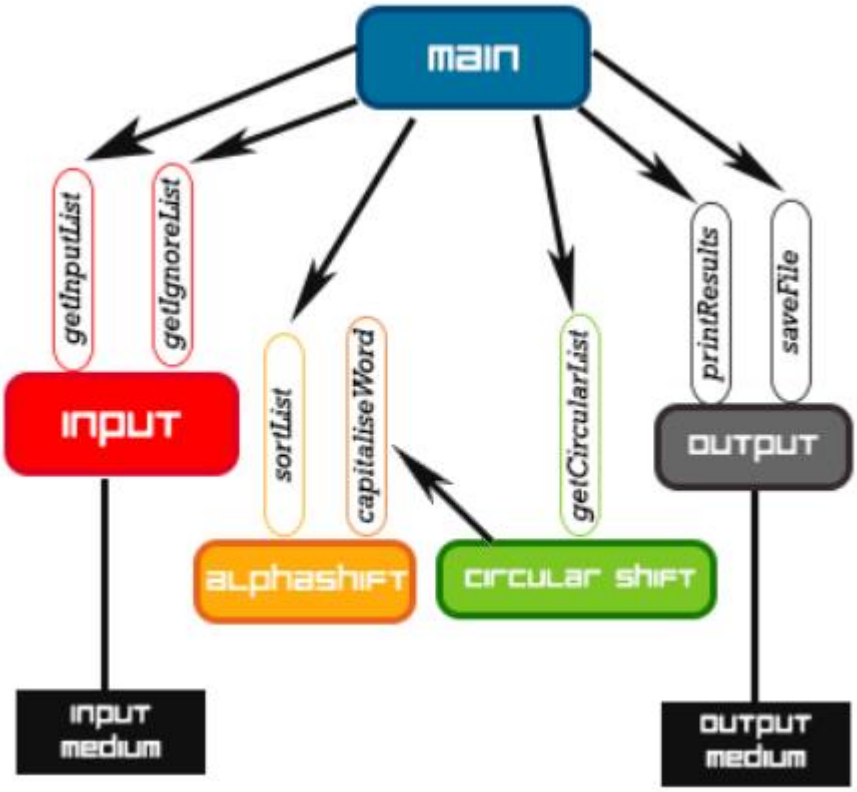
1. Have an interface which:
    - Allow users to specify the path of the text file which contains all the titles
    - Allow users to specify the path of the noise file (if any).
2. The system has to circularly shift each sentence in which:
    - Each keyword should have its first letter capitalised
    - Should not begin with a noise word
    - Noise words should be of lowercase
3. The system outputs all the circular shifted sentences which:
    - Has to be sorted lexicographically
    - Should not have duplicates
    - Output to Console and into a text file named output.txt
4. Upon every run, the output file *output.txt* should be overwritten to capture the current new output.

Non-Functional Requirements

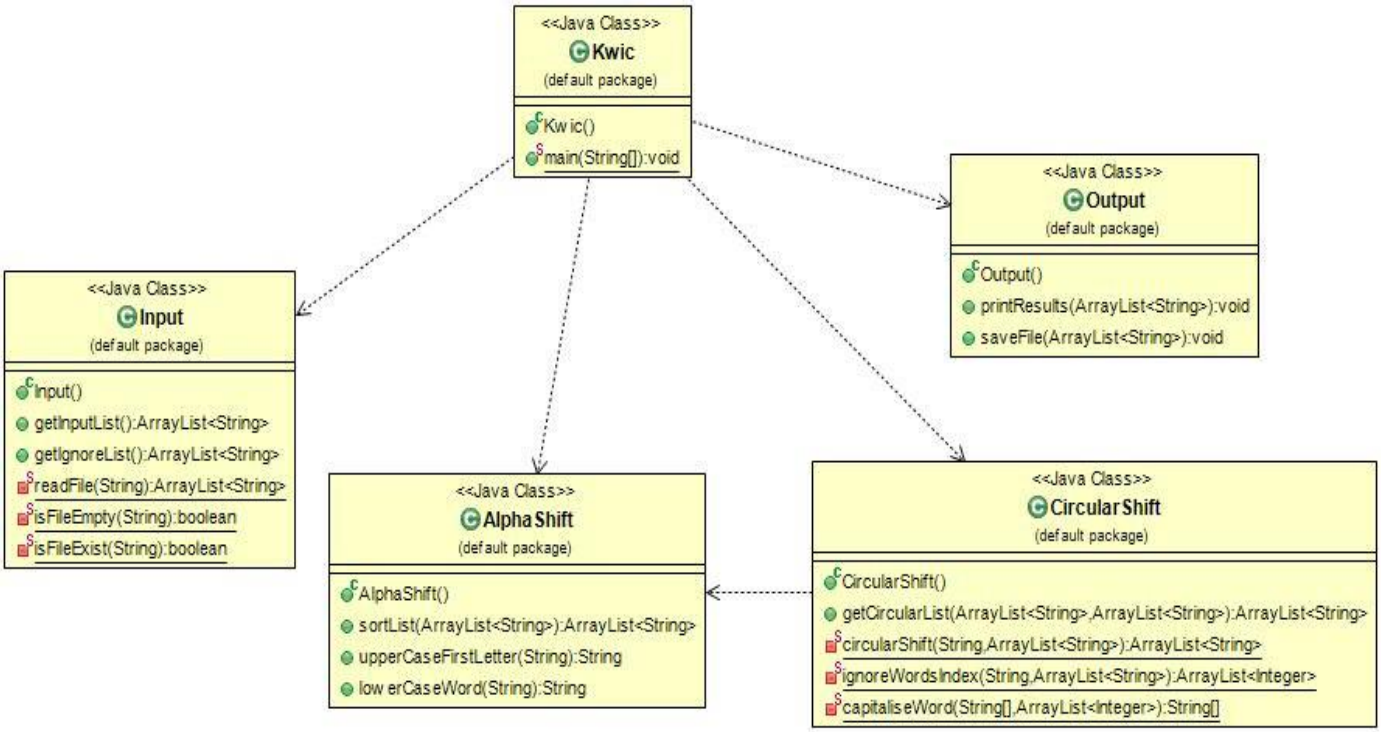
- The system must be able to process at least 1000 lines of titles
- Output must be generated within 1 minute

3. Architectural Designs

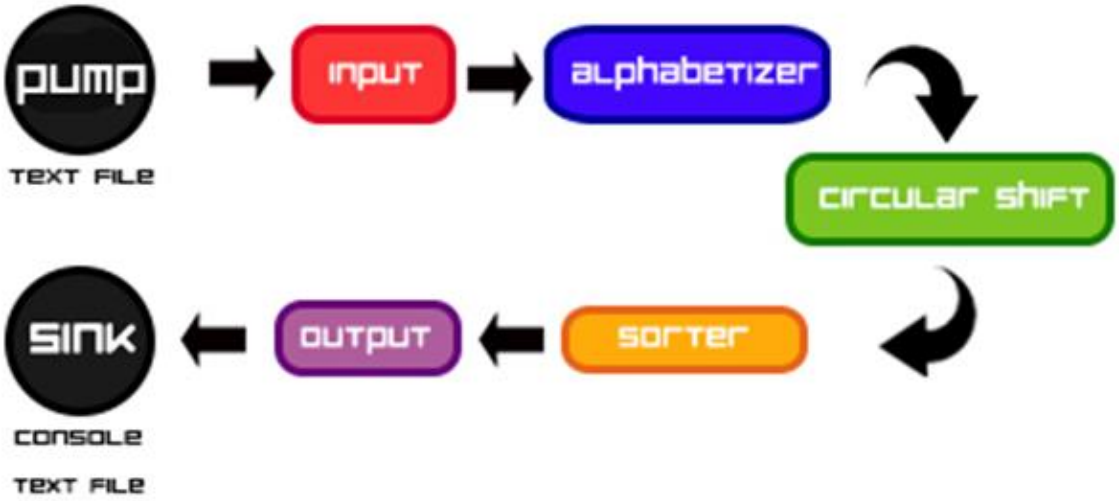
Abstract Data Type



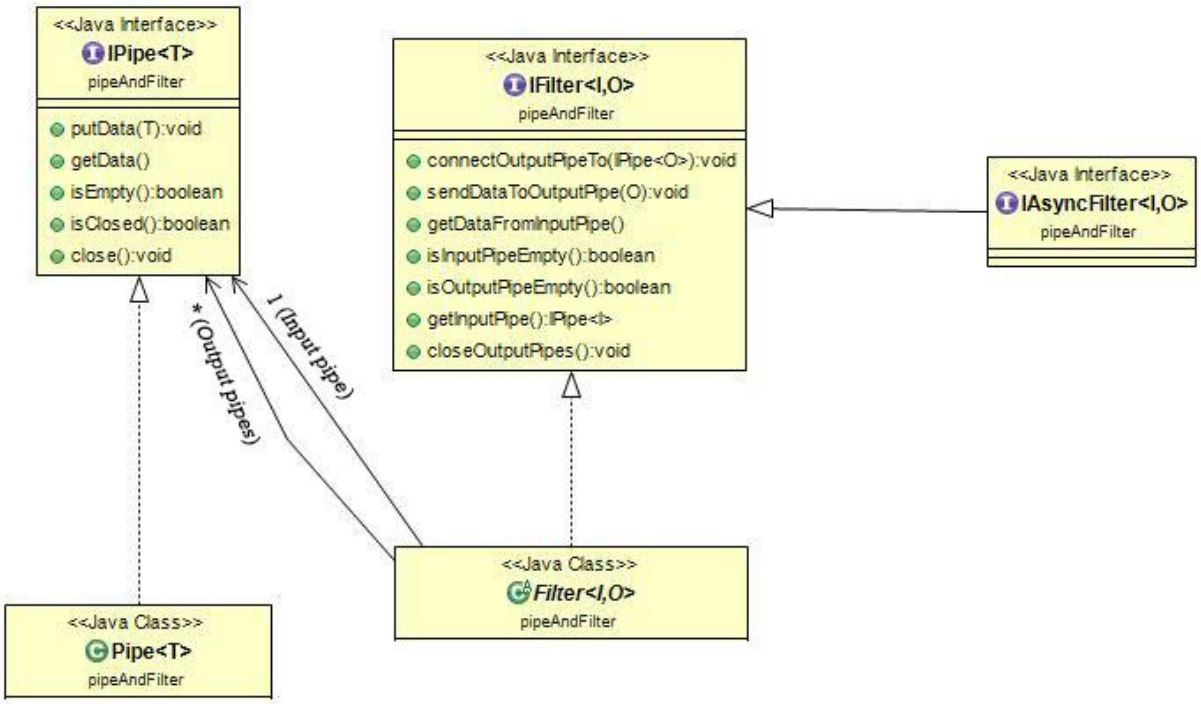
Class diagram (Four Dependency Modules)



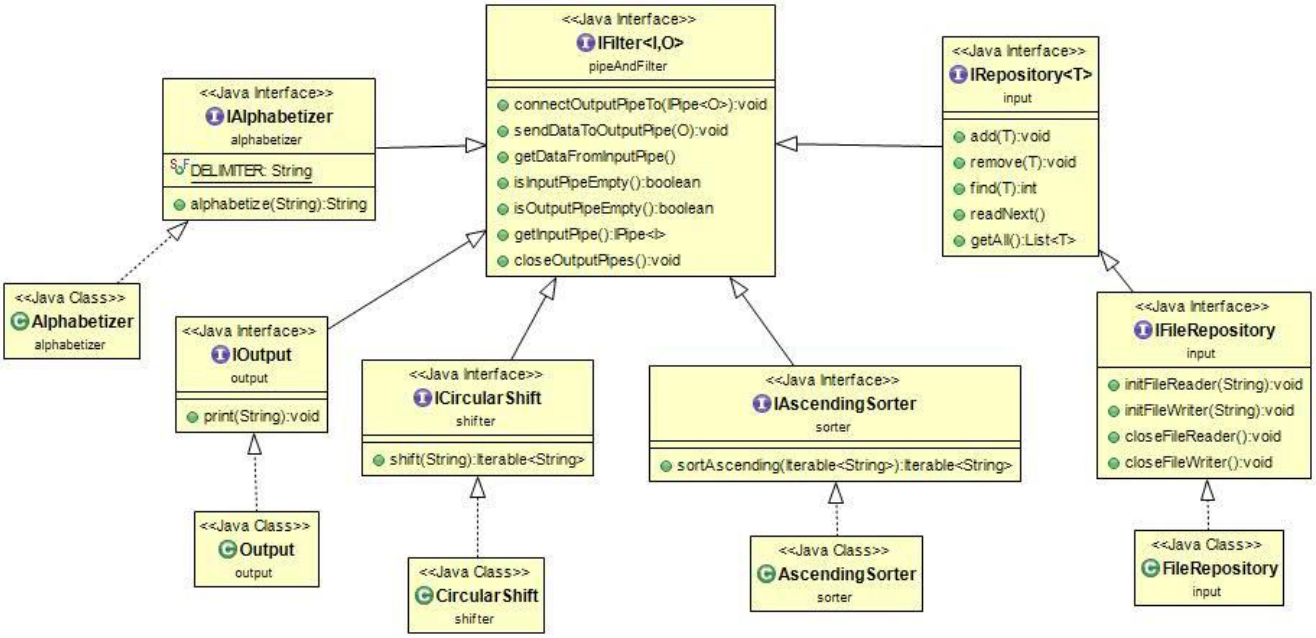
Pipes and Filters



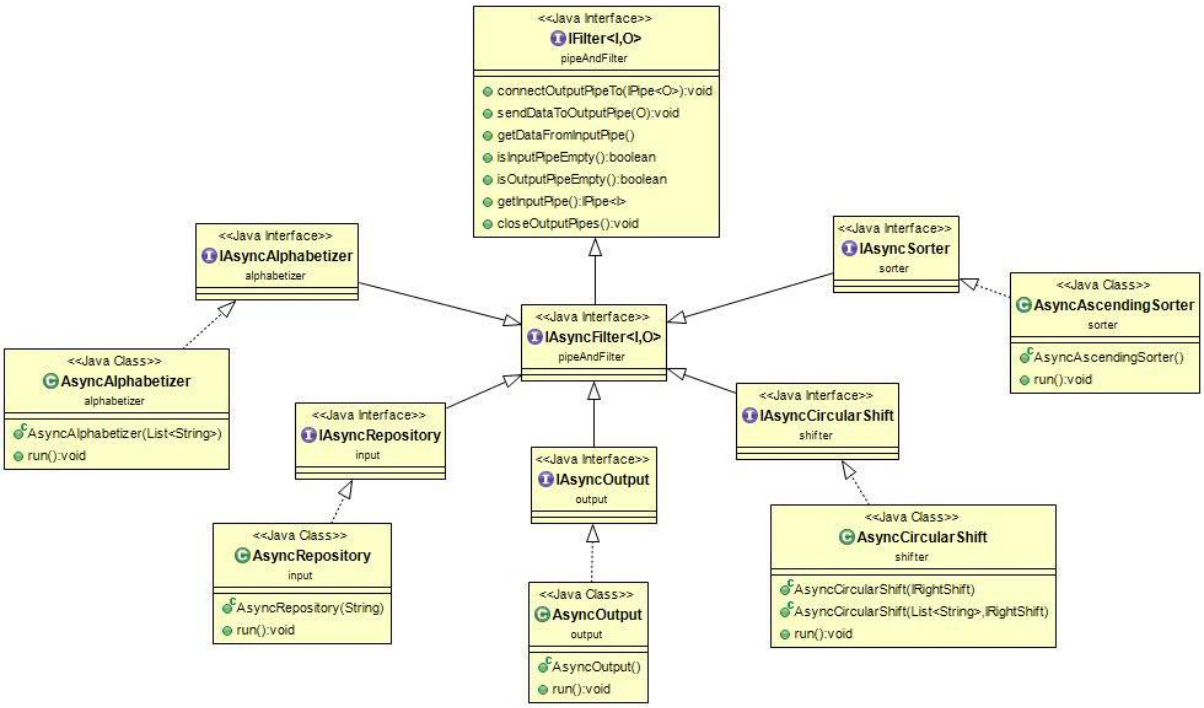
Class Diagram 1 (Pipe-Filter-Asynchronous Filter)



Class Diagram 2 (Five Filters)



Class Diagram 3 (Five Asynchronous Filters)



4. Limitation & Benefits of Selected Designs

Abstract Data Type

Benefits:

- Algorithms and data representations can be modified inside individual modules without affecting others
- Looser coupling as reuse is better supported as modules make fewer assumptions about the others with which they interact

Limitations:

- Not well-suited for enhancement
- Additional of new functionalities compromise the simplicity and integrity of the system as existing modules need to be modify
- Addition of new modules may lead to performance penalties

Pipes and Filters

Benefits:

- **Decoupled**: Ease of modification as each filter is a standalone component
- **Scalability**: More filters could easily be added to extend the pipeline when need arises
- **Parallel Processing:** As it's a pipeline, all filters are simultaneously processing data at different stages and are less likely to be in a blocked state
- **Improved Performance**: Delay in a filter could be reduced by adding more of such filters in that segment. *(Data would be load balanced between these filters in such a scenario)*
- **Broadcast:** Data from a single filter could be broadcasted to multiple different filters using multiple output pipes

Limitations:

- Unable to support an interactive system as it is virtually impossible to modify the design
- **Inefficient Memory Usage:** Each filter must copy all of the data to its output ports

5. User Guide

Abstract Data Type (ADT) & Pipes and Filters

1. Save all the titles in a text file (e.g. input.txt).
2. Separate each new titles with a newline in the text file
3. Save all the noise words in a text file (e.g. ignore.txt)
4. Separate each noise word with a newline in the text file
5. Run the “Kwic.java” in the “ADT” or “Pipes and Filters” folder
6. Enter the name of the input file (file which contains all the titles)
7. Enter the name of the “noise words” file
8. If an empty “noise words” file is entered, a message will be prompted to the user to determine if the user wants to enter a new file or proceed with the empty file.
9. The final output will be shown in the command line and saved in a “output.txt” file

6. Work Allocation

Abstract Data Types (ADT): Tang Wei Ren

Pipes and Filters: Razali