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

**Code Repository URL:** [**https://github.com/CS3219-Assignments/Assignment-1-KWIC**](https://github.com/CS3219-Assignments/Assignment-1-KWIC)

|  |  |  |
| --- | --- | --- |
| **Student Name** | Tang Wei Ren | Razali |
| **Matriculation Number** | A0125531R | A0133267H |

**1. Introduction**

KWIC provides a search mechanism for information in a list of lines. Given a list of lines and a list of ``words to ignore'', KWIC system should generate a KWIC (Key Word In Context) index of the input lines. In a KWIC-index, a line is listed once for each keyword that occurs in the line. The keyword cannot be in "words to ignore". Also, KWIC-index is alphabetized by keyword. For each input line, it shall be “circularly shifted” exhaustively by removing the first word and appending it at the end of the line to create a set of circularly shifted lines. The first word (not including "words to ignore") will be the keyword. The system outputs a listing of the circularly shifted lines for all input lines in ascending alphabetical order. The keyword is often output with its starting letter in upper case and the rest in lower case.

**2. Requirements**

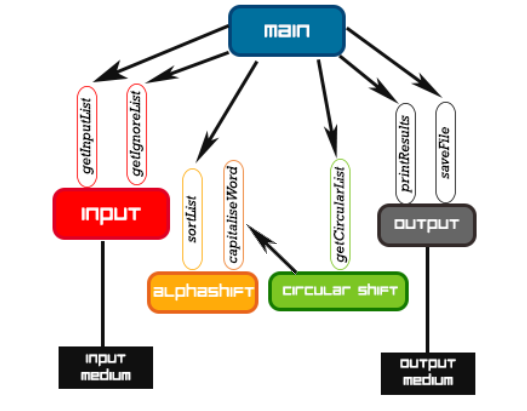
Functional Requirements

* Interface for user to enter the sentences
* Interface for user to enter the ignore words
* The system has to circularly shift each sentence
* The starting word of each sentence much not start with the words found in the list of ignore words
* All ignore words in the sentence must be of lower case
* The first letter of all keywords (non-ignored words) much be capitalised
* The system output all the circular shift sentences in alphabetical order

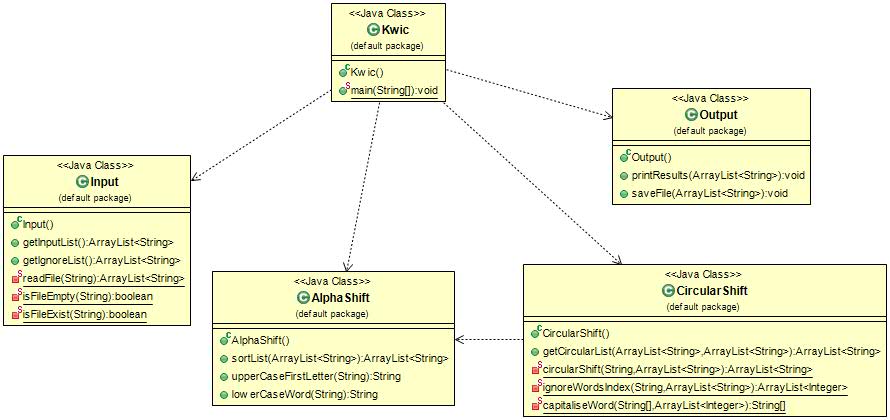
Non-Functional Requirements

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

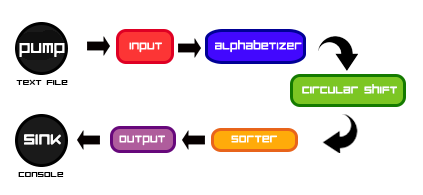
**3. Architectural Designs**

**Abstract Data Type** 

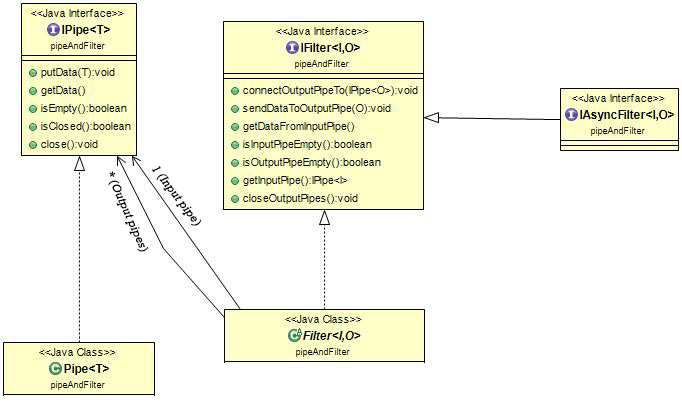
**Class diagram (Four Dependency Modules)**



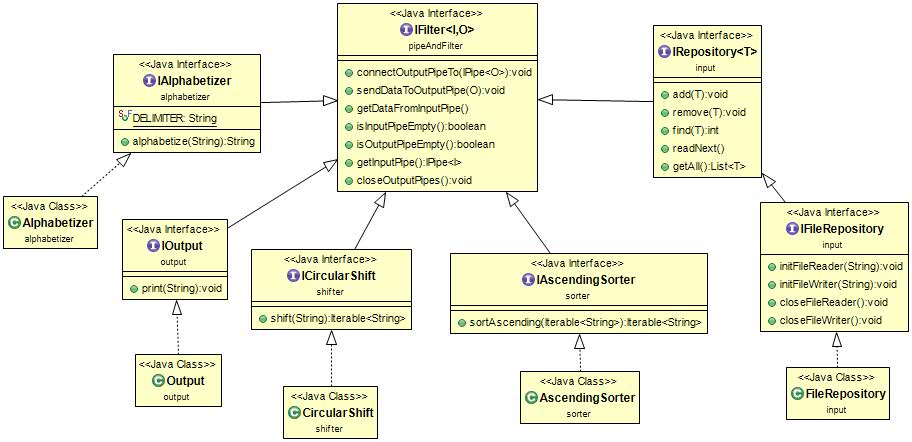
**Pipes and Filters**



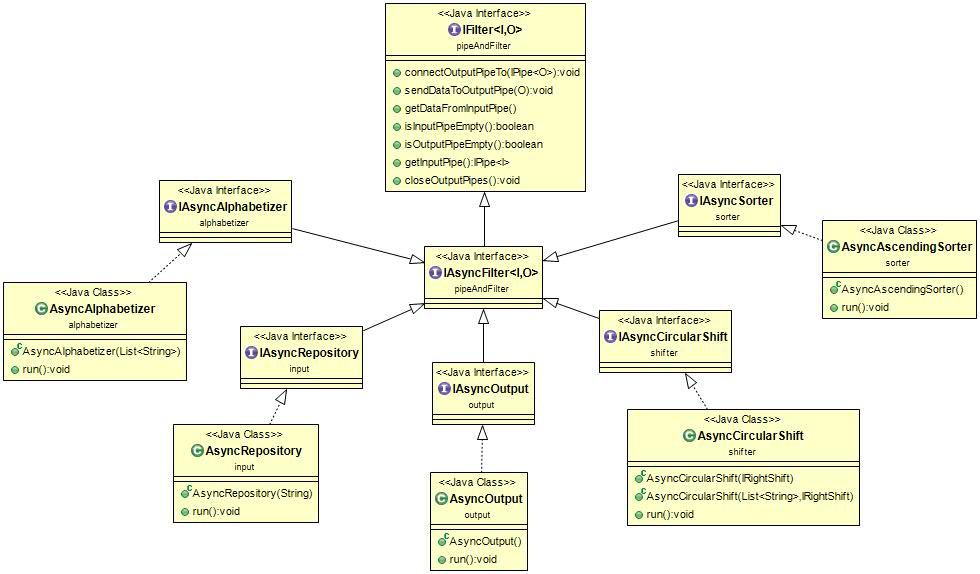
**Class Diagram 1 (Pipe-Filter-Asynchronous Filter)**



**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
* Reuse is better supported as modules make fewer assumptions about the others with which they interact – looser coupling

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:

* Maintains an intuitive flow of processing
* Supports reuse as each filter can function in isolation (provided upstream filters produce data in the form it expects)
* New functions can be added easily into the system by inserting filters at the appropriate point of the processing sequence
* Ease of modification as filters are logically independent of other filters

Limitations:

* Unable to support an interactive system as it is virtually impossible to modify the design
* Inefficient in terms of its use of space as 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 sentences in a text file (e.g. input.txt).
2. Separate each new sentence with a newline in the text file
3. Save all the ignore words in a text file (e.g. ignore.txt)
4. Separate each ignore word with a newline in the text file
5. Run the “Kwic.java” in the “ADT” folder
6. Enter the name of the input file (file which contains all the sentences)
7. Enter the name of the “ignore words” file
8. If an empty “ignore 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 show in the command line and save in a “output.txt” file

**6. Work Allocation**

Abstract Data Types (ADT): Tang Wei Ren

Pipes and Filters: Razali