Assignment 1: KWIC-index system

Code Repository URL: https://github.com/Pingxia/CS3213_KWIC

Name	Bai Xuefeng	Zheng Pingxia
Matriculation Number	A0105655H	A0105662L
Implemented	Logic, Storage	UI

1. Introduction

KWIC index system is a search mechanism for information in a long list of lines. The input includes a list of 'words to ignore' and a list of 'titles to process'. For each input, it will be repeatedly removing the first word and appending it at the end of the line. The output will be a list of circularly shifted lines in ascending alphabetical order.

2. User Guide

Below is the list of supported functions:

1. Add input titles and words to ignore using keyboard.
2. Add inputs by browsing the file lists.
3. Export output list to any chosen file destination.

3. Design and Architecture

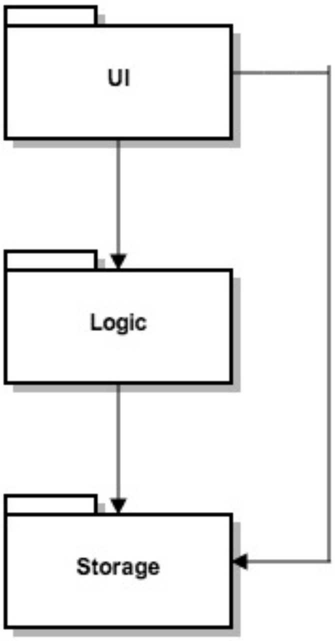

Figure 3.1

UI handles user input and displays output.

Logic executes the actual commands.

Storage stores all the data (lists etc)

UI is in charge of taking in user inputs and displaying outputs to screen. UI will set the storage component and then call logic component to calculate the actual circular shifts and sort the list.

4. Implementation

UI component includes a MainUI class, an InputPanel class and an OutputPanel class. MainUI calls Manager class in logic and pass all the input lists to Data class in storage.

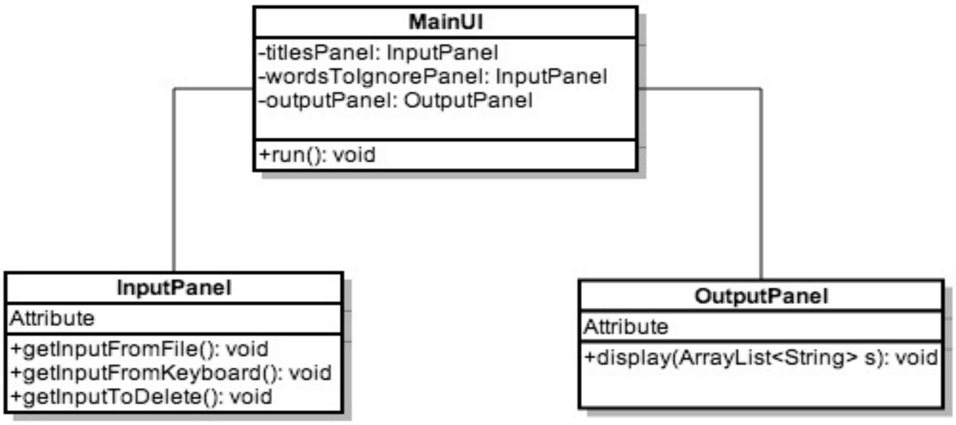
MainUI
-titlesPanel: InputPanel
-wordsToIgnorePanel: InputPanel
-outputPanel: OutputPanel
+run(): void
InputPanel
Attribute
+getInputFromFile(): void
+getInputFromKeyboard(): void
+getInputToDelete(): void
OutputPanel
Attribute
+display(ArrayList<String> s): void

Figure 4.1

Logic component includes a Manager class, a CircularShifter class and a Sorter class. Manager is acting as the façade between UI and Logic components, and calls CircularShifter and Sorter to do their jobs.

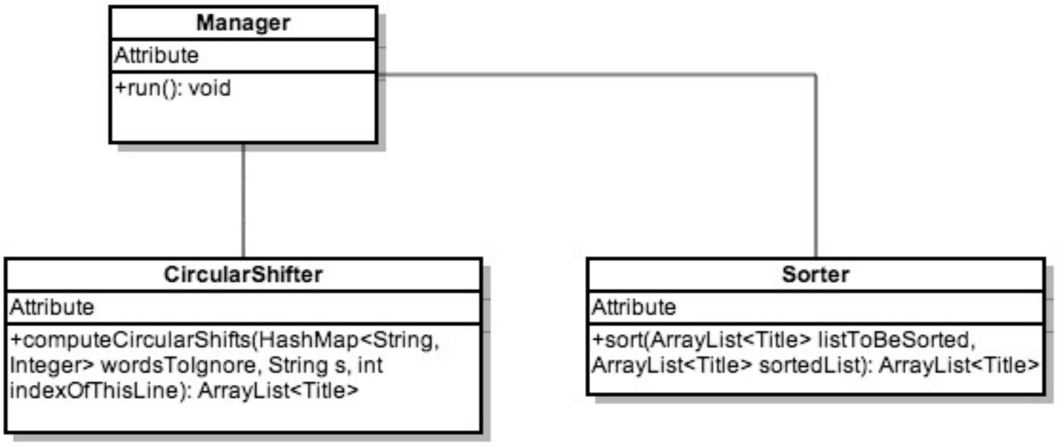
Manager
Attribute
+run(): void
CircularShifter
Attribute
+computeCircularShifts(HashMap<String, Integer> wordsToIgnore, String s, int indexOfThisLine): ArrayList<Title>
Sorter
Attribute
+sort(ArrayList<Title> listToBeSorted, ArrayList<Title> sortedList): ArrayList<Title>

Figure 4.2

Storage component includes a Data class, which stores all the data and provides a list of API for other classes to use.

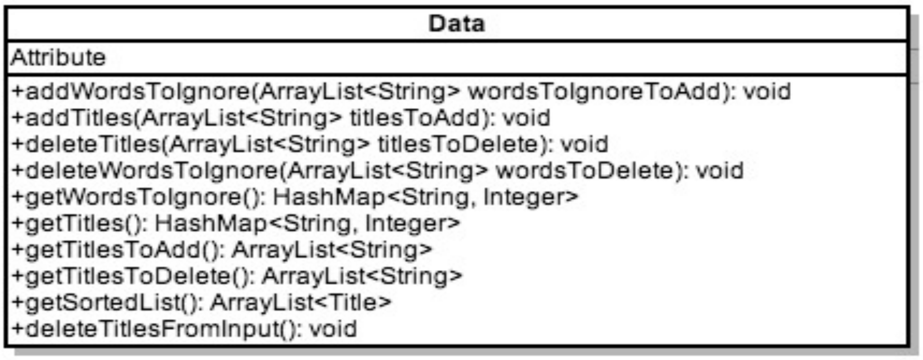
Data
Attribute
+addWordsToIgnore(ArrayList<String> wordsToIgnoreToAdd): void
+addTitles(ArrayList<String> titlesToAdd): void
+deleteTitles(ArrayList<String> titlesToDelete): void
+deleteWordsToIgnore(ArrayList<String> wordsToDelete): void
+getWordsToIgnore(): HashMap<String, Integer>
+getTitles(): HashMap<String, Integer>
+getTitlesToAdd(): ArrayList<String>
+getTitlesToDelete(): ArrayList<String>
+getSortedList(): ArrayList<Title>
+deleteTitlesFromInput(): void

Figure 4.3