Homework 1: KWIC-KWAC-KWOC

Code Repository URL: https://github.com/zhixing/KWIC.git

Name	Qing Cheng	Yang Zhixing
Matric Number	A0091747W	A0091726B

1. Introduction

The homework is an implementation of KWIC in Java. It runs in command line. First, the user is asked to key in a line of ignored words separated by space. Then, the user is asked to key in some lines of sentences. When an empty line is entered, the system begins to process and output the KWIC in alphabetical order.

2. Design

We used Pipes and Filters design pattern. There are four filters: Input, Circular Shift, Alphabetizer and Output. See figure below.

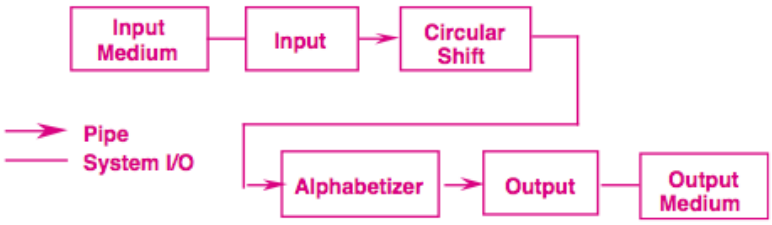

Figure: The design of our implementation

The jobs of the four filters are:

Filter	Function
Input	Get input: ignored words and sentences
Circular Shift	Shift the sentences according to the KWIC rule
Alphabetizer	Sort the shifted sentences alphabetically
Output	Output the sorted sentences

The main functions are as follows:

```
public static void main (String[] arg){

        ArrayList<String> ignoredList = getInputIgnoredList();

        ArrayList<String> inputStringList = getInputLines();
```

```
        ArrayList<String> circularShiftedList = circularShift(inputStringList,
ignoredList);

        ArrayList<String> alphabetizedList = alphabetizedList(circularShiftedList);

        outputResult(alphabetizedList);

    }
```

3. Limitations & Benefits of Selected Design

Benefits:

1. Workflow is clear: It keeps the natural flow of processing. So it's easy to track the flow, debug and maintain.
2. Control is separated: Each filter takes in values, process and return a set of values. Thus, each filter does its work separately based on input. They don't depend on each other.
3. Data is independent: Each filter works on their own input only. They have no shared data. So in their internal implementation, they are free to choose their own way to deal with data.
4. Ease of modification: It's easy to change the workflow. Just insert/remove new filters in the correct place.

Limitations:

1. Since there is no shared storage, so it's hard to extend the system to support interactive behaviors. For example, in order to delete a line, we need some shared storage to support this function, but it goes against the design and interrupts the workflow.
2. It wastes space, because each filter needs to return all its data to a new variable for the next filter.