

# CSCI 250 Advanced Software Engineering

## Project -1 -KWIC(Modularization)

Submitted By- Raghav                -301390675  
                         Ritvik Gaur       -301426477

### Introduction

KWIC (Key Word In Context) is a text analysis technique used to index and search for specific words or phrases within a larger body of text. It involves creating an index of all the words or phrases in a document, and then listing them in alphabetical order, along with their context (i.e. the words immediately preceding and following them).

The purpose of KWIC indexing is to allow users to quickly and easily locate specific information within a large corpus of text. It is often used in information retrieval systems, such as search engines, where users can enter a keyword or phrase and receive a list of all the occurrences of that keyword or phrase within the indexed documents, along with their context.

In this Report we have applied the following 3 requests to Both Modularization 1 and Modularization 2 and then analyse how each request causes the source code to change.  
The Three Requests:

(a) Request 1: All the input remains string. Sort the result in the reverse alphabetic order.  
Also, allows input from the command line.

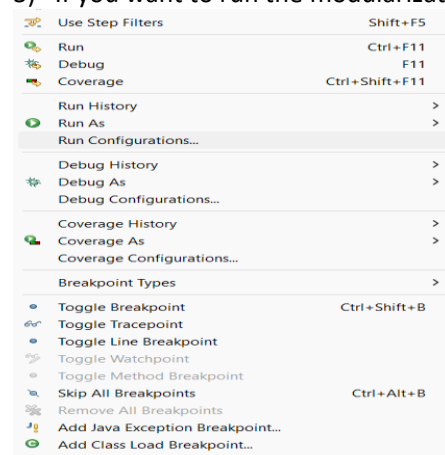
(b) Request 2: All the input remains string. Introduces noise elimination in KWIC (you determine which module in both Modularizations 1 and 2 and how) that eliminates “a”, “an”, “the”, “and”, “or” (both upper and lower cases).

(c) Request 3: All the input are integers instead of string.

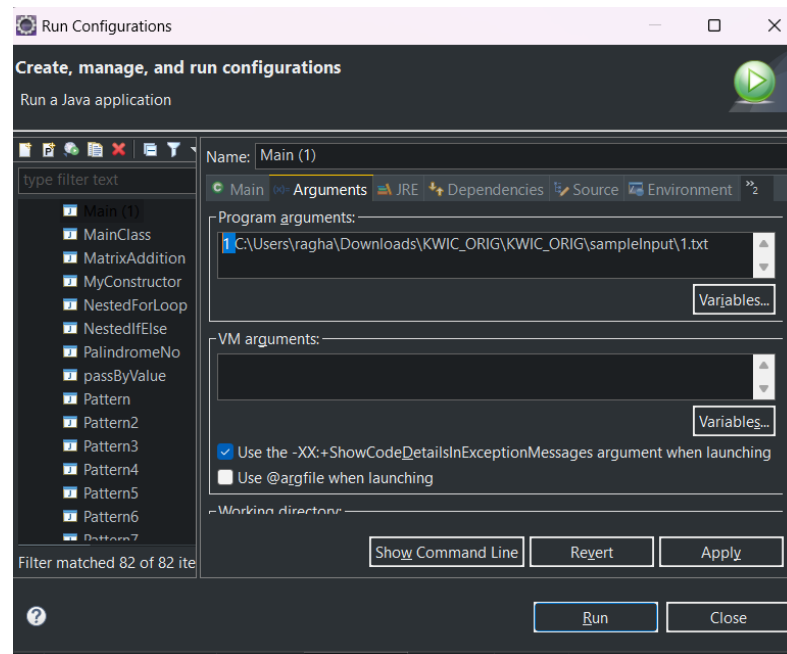
### Usage

#### Set up -

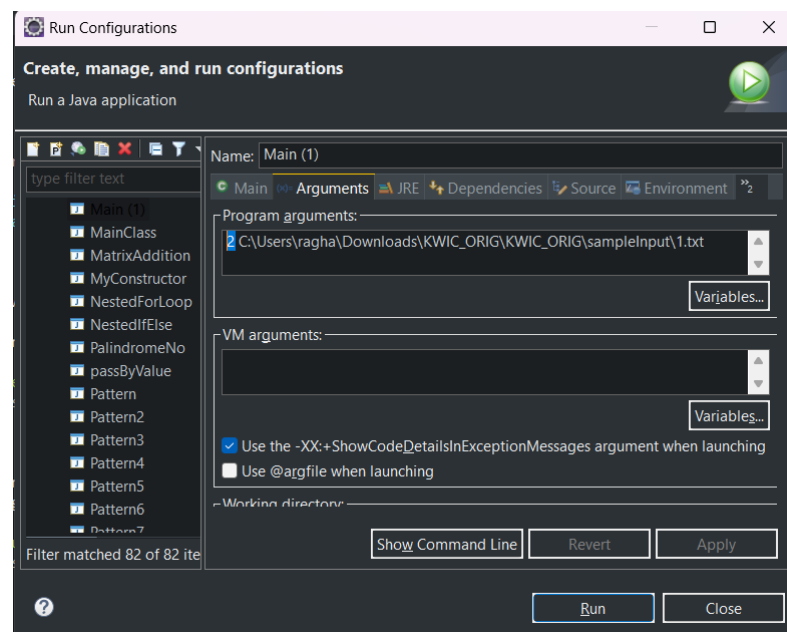
- 1) Extract the files from the zip folder
- 2) Run the file at \KWIC\_ORIG\src\com\company\main.java
- 3) If you want to run the modularization 1 go to run configurations



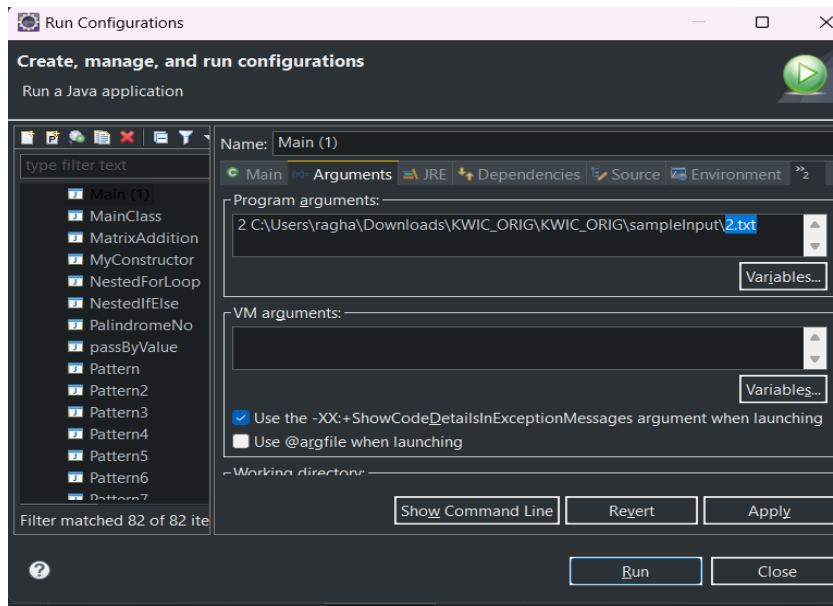
4) If you want to run Modularization 1 do this



5) If you want to run Modularization -2 Do this

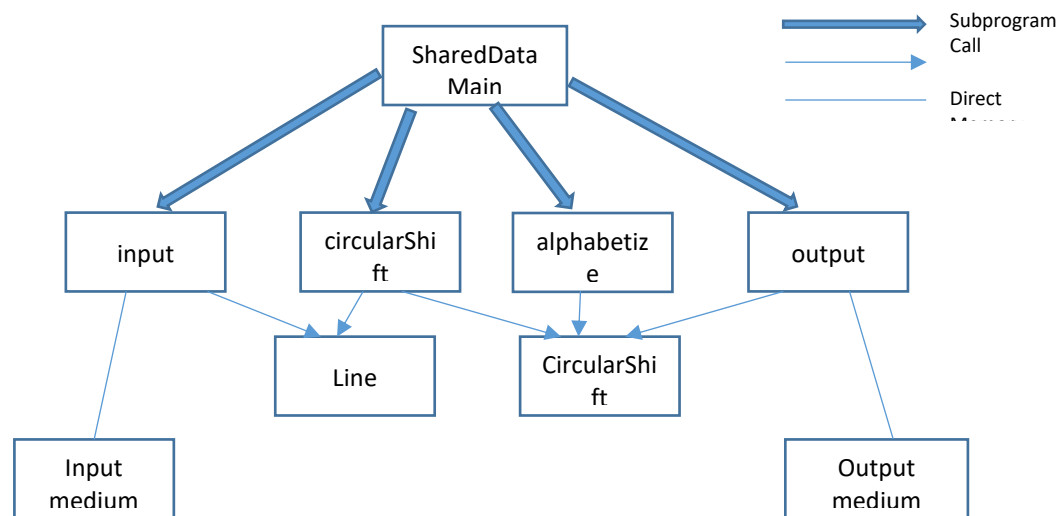


In this run configuration the text file is 1.txt which contains the input for request 1 and request 2, if you want to run request 3 we need to change the text file from 1 to 2 which can be done easily as shown below



## In Modularization 1-

UML Diagram for Modularization 1



### A) Request 1

Sort the result in reverse alphabetic order

To sort the KWIC output in reverse alphabetical order and allow input from the command line, the following changes need to be made to the original source code: alphabetize() method was predefined for Modularization 1, Here we did the following steps:

- 1 - Removed the original alphabetize() method functioning.
- 2 - Creating a new alphabetize() method that sorts the circular-shifts output in descending order.

Disadvantages of Modularization 1:

The main disadvantage of Modularization 1 is that the code becomes more complex and difficult to maintain as more features are added. Also, the code may become bloated with repetitive code for each new feature that is added.

## B) Request -2

To eliminate noise words from the KWIC output, the following changes need to be made to the original source code:

setWordsToIgnore method was predefined for Modularization 1, Here we did the following steps:

- 1 - Removed the source code for the manual input of the words to ignore
- 2 - Modified and added the new method and code for noise elimination in KWIC for "a", "an", "the", "and", "or" (both upper and lower cases).

Disadvantages of Modularization 1:

The main disadvantage of this approach is that it requires modifying an existing function, which can be error-prone and difficult to maintain. Also, if more noise words need to be added in the future, the function will need to be modified again.

## C) Request -3

To process integer inputs instead of string inputs, the following changes need to be made to the original source code:

1- Creating a new run method that only works when an input is a string to run the part three we need to uncomment the runInt method just right below the run method.

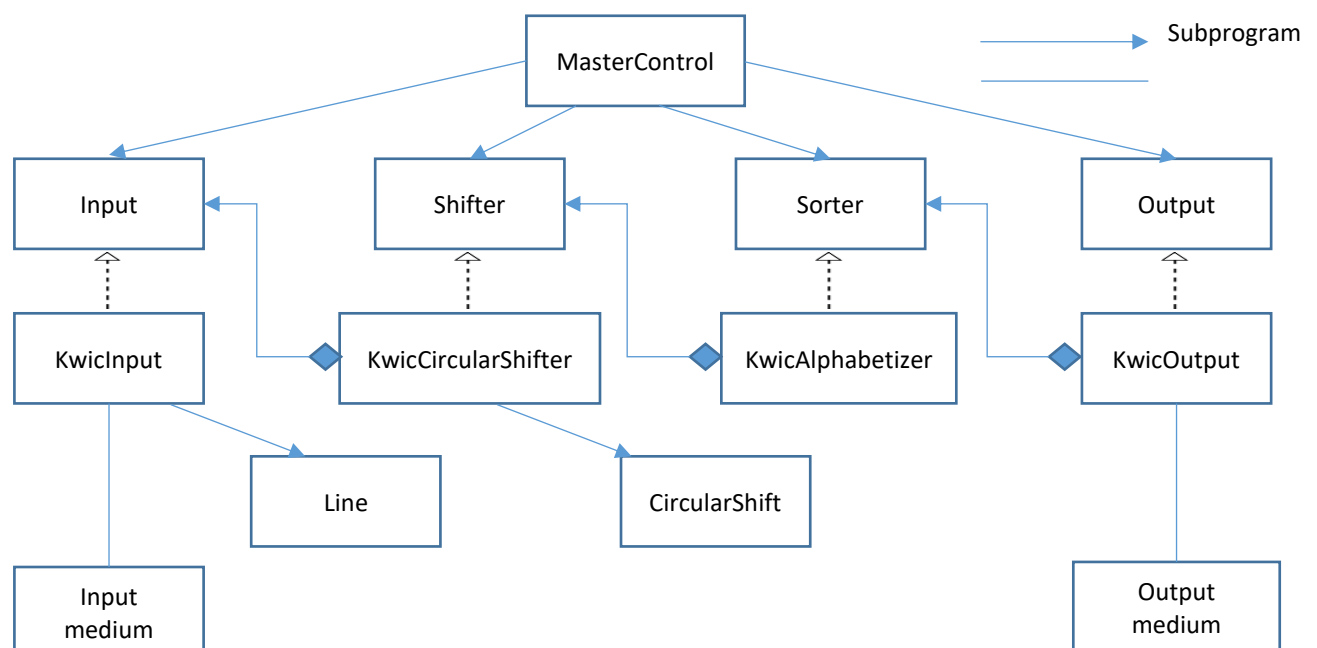
2 - As we can see that everything information is shared we can see that the run method bellow calls the input method which takes string a input so we also have to create a new method for that too and we need to call that method from SharedStorageMain that also needs to be created new.

3-Here as input method that was already defined takes string as input so we have to create a new input method that accepts only strings as input.

Disadvantages of Modularization 1:

The main disadvantage of this approach is that it requires modifying an existing function, which can be error-prone and difficult to maintain. Also, the "output" function will need to be modified to work with integer inputs instead of strings.

## UML diagram from Modularization 2



### Request -1

To sort the KWIC output in reverse alphabetical order and allow input from the command line, the following changes need to be made to the original source code:

New method `reversegenerateSortedList()` that perform list of all output valid circular shifts of all lines in descending order.

### Request -2

To eliminate noise words from the KWIC output, the following changes need to be made to the original source code:

Addition of new method `readWordsToIgnoreSpecific()`, `processArgumentsSpecific()` for sorting the noise elimination in KWIC that eliminates "a", "an", "the", "and", "or" (both upper and lower cases).

### Request-3

To process integer inputs instead of string inputs, the following changes need to be made to the original source code:

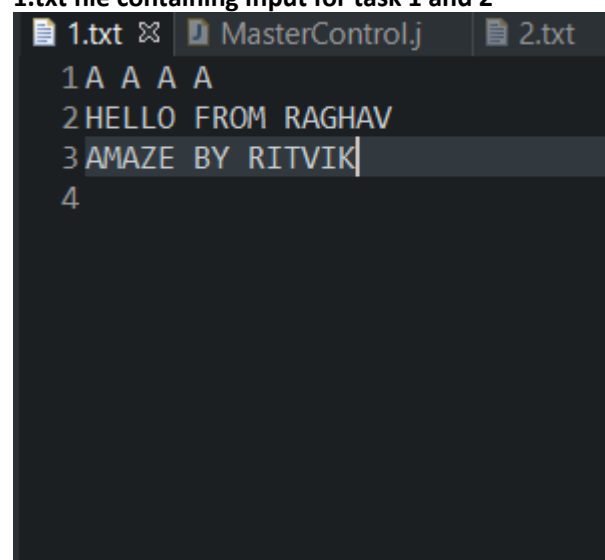
/Here, the regular expression `d+`, which matches one or more digits, is used to filter out any elements that do not match. The last element is then converted to an integer using the `mapToInt()` method, and the resulting `IntStream` is then utilized to generate an `int[]` array using the `toArray()` method.

**The advantage of Modularization 2** is that it allows for easy extension and modification of the system. If we need to add new functionality, we can simply add a new module without modifying the existing code. Additionally, it follows the Single Responsibility Principle (SRP) by separating input processing and output formatting into two separate modules, making the code more maintainable and easier to understand.

**The tradeoff of Modularization 2** is that it requires more effort to initially set up the system. It requires the creation of two separate modules and additional code to communicate between the two modules. However, the benefits of modularity and maintainability outweigh the additional setup time in the long run.

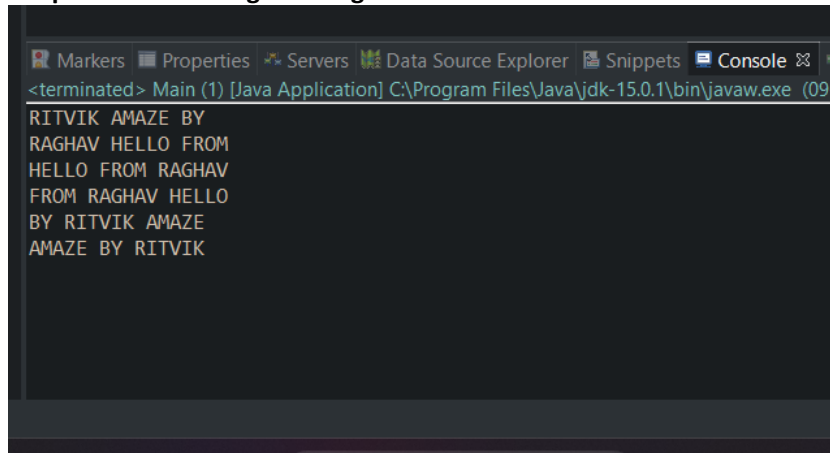
### Output ScreenShots-

1.txt file containing input for task 1 and 2



```
1.txt x MasterControl.j 2.txt
1 A A A A
2 HELLO FROM RAGHAV
3 AMAZE BY RITVIK
4
```

### Output After Running the Program.



A screenshot of the Eclipse IDE's Console window. The title bar shows 'Markers', 'Properties', 'Servers', 'Data Source Explorer', 'Snippets', and 'Console'. The console text shows the program has terminated and displays the following output:

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (09-  
RITVIK AMAZE BY  
RAGHAV HELLO FROM  
HELLO FROM RAGHAV  
FROM RAGHAV HELLO  
BY RITVIK AMAZE  
AMAZE BY RITVIK
```

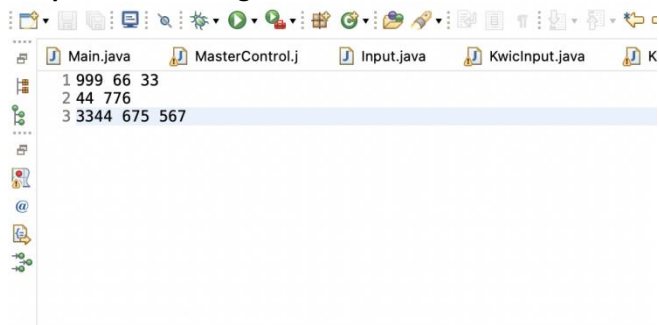
### Txt.2 File Containg input for task3



A screenshot of the Eclipse IDE's Console window. The title bar shows 'Console'. The console text shows the program has terminated and displays the following input data for task 3:

```
<terminated> Main (3) [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org  
999 66 33  
776 44  
675 567 3344  
66 33 999  
567 3344 675  
44 776  
3344 675 567  
33 999 66
```

### Output after running the Text File 3



A screenshot of the Eclipse IDE's Editor window. The title bar shows 'Main.java', 'MasterControl.j', 'Input.java', 'KwicInput.java', and 'K'. The editor content shows the following output for Text File 3:

```
1 999 66 33  
2 44 776  
3 3344 675 567
```