**Assignment 1: KWIC-KWAC-KWOC**

Code Repository URL: <https://github.com/anand-sundaram/KWIC>

|  |  |  |
| --- | --- | --- |
| Name | Rupali Roy Choudhury | Anand Sundaram |
| Matriculation Number | A0102796B | A0118897J |

**1. Introduction**

KWIC is a system that presents a search mechanism for information in a long list of lines. For our assignments, we are required to implement functional as well as non-functional requirements. We have successfully implemented the basic functionalities of KWIC using mainly two architectures – 1) Shared Repository (done by Anand) 2) Data Abstractions and Object-Oriented Organisation (done by Rupali).

NOTE: Please use JavaSE-1.8 for JRE System Library

**2. Design**

Main Program/Subroutine with Shared Data

In the Shared Data solution, the problem is decomposed in 4 components according to the basic functions performed – Input, Shift, Alphabetize and Output. The components are implemented as functions which are called sequentially by a main program. The main aspect of the Shared Data design is that all data communication between the different components occurs through a shared storage. Thus each component has an unrestricted read/write access to the shared data. I implemented the each component as a static function within a class. The shared data was simulated using static variables that were available globally within a class. Since every function has access to the static variables, there is no need for parameter passing.

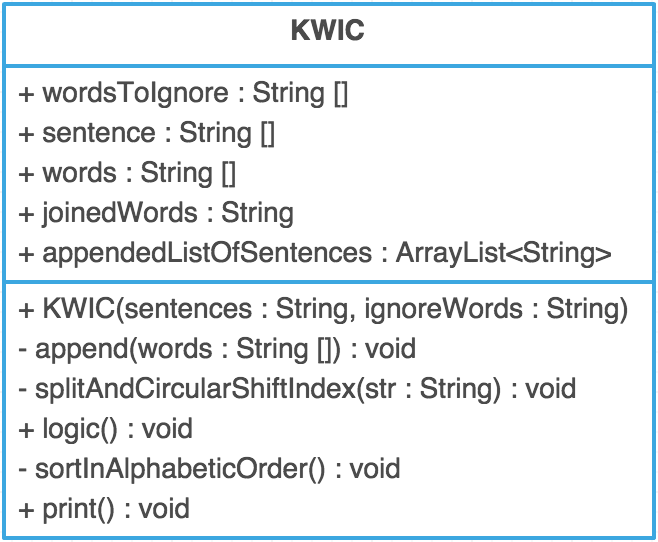
Class Variables:

* ArrayList<String> wordsToIgnore – stores the words to be ignored
* ArrayList<String> titles – stores all the titles in an array of sentences
* ArrayList<ArrayList<String>> titlesByWords – stores all the titles in an array, with each element being an array containing the words in the title
* ArrayList<ArrayList<Integer>> index – stores the (title index, word index) pairs for all keywords
* ArrayList<ArrayList<Integer>> alphabetizedIndex - stores the (title index, word index) pairs for all keywords, in alphabetized order

Class Methods:

* splitter(String sentences, String ignoreWords) – takes in the string for titles and words to be ignored from the common input
* circularShifter() – generates the list of pairs (title index, word index) for all keywords, after doing the shifting and ignoring the given words
* alphabetizer() – orders the list of pairs (title index, word index) in alphabetical order
* output() – prints the keywords in the right order and the specified format

Data Abstractions and Object-Oriented Organisation



KWIC class consists of mainly 6 functions, as can be seen from the class diagram. The user initially provides the sentence, words to ignore and the type of architecture. An object of KWIC is created. This object first calls the logic method, which in turn calls other methods as explained below. The purpose of the following methods is described below:

* KWIC(String sentences, String ignoreWords) – This constructor takes in the string values, which are provided by the user and separates them based on “,” (comma) or “ ,”(space + comma). These separated strings are stored inside an array.
* public void logic() – Takes in the array of string and sends one index at a time to the function splitAndCircularShiftIndex.
* private void splitAndCircularShiftIndex(String str) - It takes the individual index of the sentence array and further splits it to words and stores in an array called words. This array is then useful for performing the circular shift. After the circular shift is done once, it calls the append function.
* private void append(String words[]) – The aim of this function is to join the word array and append if the first word is not in the list of words to be ignored. The capitalization of the first word is performed here. After that, sortInAlphabeticOrder function is invoked
* private void sortInAlphabeticOrder() – The appended sentences are then ordered in alphabetical order.
* public void print() – displays the final outcome as mentioned under the basic requirements.

**3. Limitation & Benefits of Selected Design**

Benefits of Main Program/Subroutine with Shared Data

1. Intuitive: The design is intuitive, and hence easy to implement. Since each component handles one basic function, there is high cohesion. Further, each component does not depend on another component to run, hence there is low coupling.
2. Efficient: It could be more space efficient since the shared repository means that there is little data redundancy.
3. Lack of Data Conflicts: There is no scope for data conflicts, which are situations where multiple components try to modify the same data, since there is a main controller program that ensures sequential execution.

Limitations of Main Program/Subroutine with Shared Data

1. Not Reusable: Most of the code cannot be reused as they have been written to work specifically with the mechanism and data storage format used.
2. Single Point of Failure: Since the main program calls all the functions and controls the execution, it could potentially become a single point of failure.

Benefits of Data Abstractions and Object-Oriented Organisation

* Software maintenance - Programming using this architecture is not disposable. Maintaining and improving on an object-oriented program is much easier than a non-object oriented program.
* Design benefits - Large programs are very difficult to write. Object Oriented Programs force designers to go through an extensive planning phase, which makes for better designs with fewer flaws.
* Encapsulation - Once an Object is created, knowledge of its implementation is not necessary for its use. Objects have the ability to hide certain parts of themselves from programmers. This prevents programmers from tampering with values they shouldn’t. Additionally, the object controls how one interacts with it, preventing other kinds of errors.

Limitations of Data Abstractions and Object-Oriented Organisation

* Since the functions are invoked by another functions, a small change in one of the functions can mean that these changes will have to be taken care by other functions too.