School of Innovation, Design and Engineering
DVA422 – Software Engineering 3
20 February 2018

Naida Demirovic
ndc17001@student.mdh.se

Seminar 1: Software Architecture

Abstract

This report aims to show the reader basis definitions about KWIC system. After the reader has that knowledge one of the solutions for design and implementation is represented. It also shows how the main parts of the system (classes) are related to each other, which one of them calls and which one of them uses some other class. The aim of the task II of alternative A in the assignment description is to improve logical thinking and to make sure that the reader has understood the purpose of the KWIC system.

Table of Contents

1 Introduction

KWIC is an acronym for **Key Word in Context**, the most common format for concordance lines. The term KWIC was first coined by Hans Peter Luhn. [1] The system was based on a concept called *keyword in titles* which was first proposed for Manchester libraries in 1864 by Andrea Crestadoro.[2]

A **KWIC** index is formed by sorting and aligning the words within an article title to allow each word (except the stop words) in titles to be searchable alphabetically in the index. [3]

A KWIC index is a special case of a *permuted index*. [4] This term refers to the fact that it indexes all cyclic permutations of the headings.

The KWIC (Key Word in Context) index system accepts an ordered set of lines, each line is an ordered set of words, and each word is an ordered set of characters. Any line may be “circularly shifted” by repeatedly removing the first word and appending it at the end of the line. The KWIC index system outputs a listing of all circular shifts of all lines in alphabetical order. [5]

In this case, KWIC system has been made in a way that 6 classes were created: Input, Characters, CircularShifter, Alphabetizer, Output and Program. The purpose of each of them will be explained in the class diagram section and in the uses structure section it will be shown in what way they use and call each other.

2 Class diagram

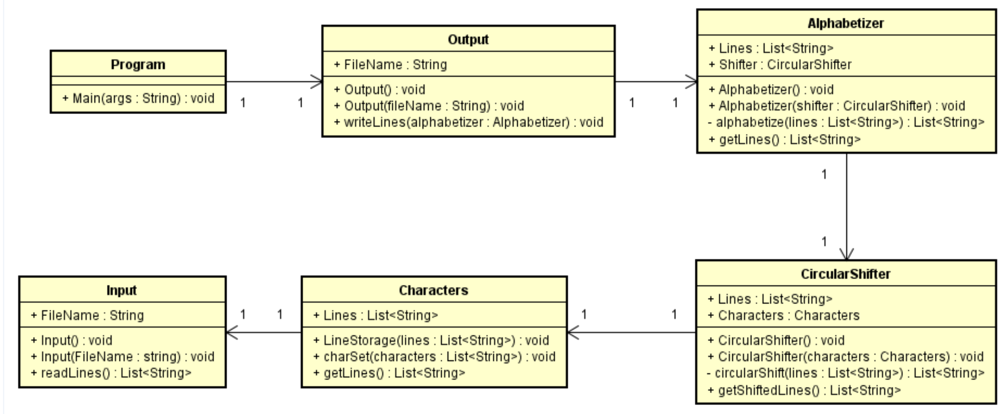
Program
+ Main(args : String) : void
Output
+ FileName : String
+ Output() : void
+ Output(fileName : String) : void
+ writeLines(alphabetizer : Alphabetizer) : void
Alphabetizer
+ Lines : List<String>
+ Shifter : CircularShifter
+ Alphabetizer() : void
+ Alphabetizer(shifter : CircularShifter) : void
- alphabetize(lines : List<String>) : List<String>
+ getLines() : List<String>
CircularShifter
+ Lines : List<String>
+ Characters : Characters
+ CircularShifter() : void
+ CircularShifter(characters : Characters) : void
- circularShift(lines : List<String>) : List<String>
+ getShiftedLines() : List<String>
Characters
+ Lines : List<String>
+ LineStorage(lines : List<String>) : void
+ charSet(characters : List<String>) : void
+ getLines() : List<String>
Input
+ FileName : String
+ Input() : void
+ Input(FileName : string) : void
+ readLines() : List<String>
1
1

Figure 1. Class diagram

In the **Input** class there is the attribute fileName that is String data type. It has default constructor, the constructor that has FileName as parameter and initializes that parameter with the fileName attribute. There is also method called readLines() that is List<String> data type and it reads lines from the text file and initializes them in a list of lines that will be used later in program.

In the **Characters** class there is the attribute called Lines that is List<String> data type. It has LineStrogare constructor with lines parameter which is initialized with previously made Lines attribute with List<String> data type. charSet method is initializing Lines attribute with the characters parameter from the method. Also there is method getLines which returns those Lines that will be used somewhere in the program.

In the **CircularShifter** class there are attributes Lines which is List<String> data type and characters parameter which is Characters data type. In that class there is default constructor. Also there is a constructor circularShifter that has characters as its parameter which is Characters data type. circularShift method is a private method which is List<String> data type. As a parameter it has List<String> lines parameter. In the end, there is getShiftedLines method which returns a list of shifted lines.

In the **Alphabetizer** class there are two attributes: List<String> lines and CircularShiter Shifter. There are default constructor and Alphabetizer constructor that has CircularShifter shifter as its parameter. In a private method alphabetize there is List<String> lines parameter. That lines parameter calls Sort method and then alphabetize method returns sorted list of lines. There is getLines method which returns a list of sorted shifted lines.

In the **Output** class there is FileName attribute which is String data type. There are default constructor and Output constructor with the String fileName parameter that is initialized with FileName parameter. Then we have writeLines method with alphabetizer parameter.

In the **Program** class there is the main function that has args String array as its parameter. The objects for each class are created in the main function.

3 Uses structure

The uses structure supplies the authoritative picture of how the software interacts. The concept behind the uses structure is the uses relation. Procedure A is said to use procedure B if a correctly functioning procedure B must be present in order for procedure A to meet its requirements. In practice this relation is similar to but not quite the same as the calls relation. Procedure A usually calls procedure B because it uses it. [6]

- **Input**
- **Characters**
- **CircularShifter<-Characters**
- **Alphabetizer<-CircularShifer<-Characters**
- **Output<-Alphabetizer<-CircularShifer<-Characters**
- **Program<-Output<-Alphabetizer<-CircularShifer<-Characters<-Input**

Input and Characters classes don't use any other class.

CircularShifter uses Characters's object and its method getLines().

Alphabetizer uses CircularShifter's object, so it means that it also depends on Characters's class. That object calls method getShiftedLines().

Output uses Alphabetizer object and calls method getLines(). It means that it also uses CircularShifter and Characters classes even if it doesn't call them.

Program class creates objects of every mentioned class.

4 Task II

Description: the system that only reads in one line and outputs all circular shifts of that line in any order. The task is to think about which classes would have to be included in the subset. Which of these classes would have to be fully implemented and which (if any) could be only partially implemented?

In that case Alphabetizer class is not needed since it doesn't matter in which way they will be ordered, so shifted lines don't need to be sorted. Also in CircularShifter circularShift method should be changed in a way there is no need to go through each line since there is just one. The method can be started with initializing list of words. In the Output class for the writeLines method CircularShifter object would be used as its parameter instead of Alphabetizer object. Also it is not needed to go through each line in writeLines method, just get access to the text file and fill it with shifted words.

5 Conclusion

In this solution for the KWIC system lines of the output text file are alphabetically ordered and words are changed through circular shift methods. When the user sees the output text file information about input text file content is hidden. In the main program function we can realize that all classes are connected and that the program wouldn’t run without one of them. With the description of each class and the description of the main program we can realize that even if the class doesn’t call other class it uses it. It could be a big problem in some big projects with a lot lines of code. The problem would be to find where the error is. The best solution is to divide classes to subclasses.

6 References

[1] Manning, C. D., Schütze, H.: "Foundations of Statistical Natural Language Processing", p.35. The MIT Press, 1999.

[2] Muhammad Riaz: “Advanced Indexing and Abstracting Practices”, 1989.

[3] Northwestern University in United States, http://www.janda.org/workshop/content%20analysis/kwic.htm , 2003

[4] John W. Shipman, http://infohost.nmt.edu/tcc/help/lang/python/examples/kwic/web/theory.html , 2011

[5] David Parnas: "On the Criteria to Be Used in Decomposing Systems into Modules," In *Communications of the ACM*, December 1972.

[6] Len Bass, Paul Clements, Rick Kazman: “Software Architecture in Practice”, Second Edition. Boston: Addison-Wesley, 2003