School of Innovation, Design and Engineering
DVA422 – Software Engineering 3
24 February 2019

Matko Butković
Mba18001@student.mdh.se

Seminar 1: Software Architecture

Abstract

Structural decomposition is a scope of software engineering is a process in which complex problem or system is divided into understandable parts that are easy to develop or analyse [1]. In this paper we are going to present implementation of a Key Word in Context (KWIC) system and perform the module decomposition. In first part we are going to mention the KWIC background and definition. Further, implementation will be described and structure of a system will be presented. In third part we are going to talk about module structure and give the decomposition view of the system. Finally, we address modifiability issues and discuss advantages and disadvantages.

Table of Contents

1 Introduction

In this paper we are going to describe implementation, architecture and a decomposition structure of a Key Word in Context (KWIC) system. To begin with, KWIC system is a type of index system which performs a circular shift on words in each line and alphabetically sorts the lines, before the lines are stored in textual document. KWIC index system is used in libraries, archives, journalism, etc. KWIC system presented in this report is implemented in Java code according to requirements specification provided by the lecturer. In the KWIC system we recognize five important steps: the input media reader, the format for storing the lines in the input, algorithm for circular shift, alphabetical sort by lines and the output media writer. These five steps will be implemented separately and called through the main class.

2 Decomposition Structure

In this part we are going to describe module structure and decomposition view, implementation and the structure of the KWIC system which was implemented in first assignment. Decomposition structure is useful for resource allocation and project structuring and planning; information hiding, encapsulation; configuration control [1].

2.1 Implementation description

Implementation was realized in Eclipse development environment according to the requirements specification mentioned in the introduction part. Each step of a KWIC system is coded as a separate module: input, split, circular shift, alphabetical sort and output module. Modules are assigned specific computational responsibilities [1]. Each module is coded in separate class with the task to accept variable, perform action and return the changed variable. One class named “Main” acts as a master and it calls methods from all other classes to execute. In figure 1 we present the module overview of our implemented KWIC system. Main module represents Main class in which we are calling methods from the following classes (main program calls each of the subroutines in turn):

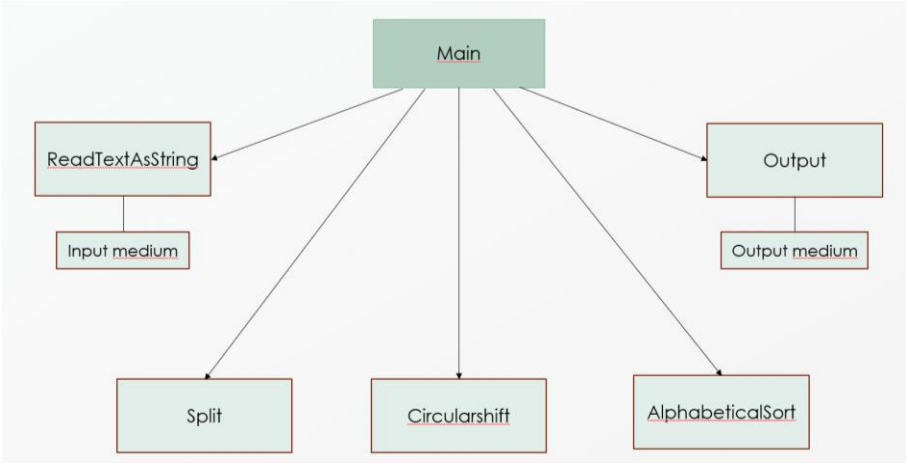

Figure 1: Overview of modules used in implementation of a KWIC system

Main or master control has direct memory acess to other modules. Input and output modules has system I/O connections. Main module has the task of sequencing tasks from other modules. Each module described is represented by one method of the class.

Class ReadTextAsString represent input module, in this class we read the textual file in which we have stored the sentences in few lines. Class implements a method readFileAsString which reads the .txt file and returns values in a string variable. Path to the textual file is added through argument into class.

Class Split represents split module and its purpose is to accept string and divide it by a space into the array of strings.

Class Circularshift performs a circular shift by lines and returns the array list. Class is realized with 3 separate for loops: loop for going through the lines, loop for going through the words in lines and loop for taking the first word and move it to the end.

Class AlfabeticalSort represents the alphabetical sort module with purpose is to accept array list of strings an perform the sort.

Class Output accepts array list of strings with shifted and sorted lines and argument of the path to the textual file. We are using java class BufferedWriter to achieve writing to the document. Outputs alphabetically sorted circular shifts of the lines.

In table 1 we provide overview of all input and output variables to each module. All classes are implemented as independent separated modules with their own computational task.

Table 1. List of Input/Outputs variables for classes

Class name:	Function:	Input:	Output:
ReadTextAsString	Reads the .txt file	Path Argument to .txt file	String
Split	Splits words by space	String	Array of Strings
Circularshift	Perform circularshift	String	Array List
AlphabeticalSort	Sorts the lines	Array List	Array List
Output	Write to .txt document	Array List; Path Argument to .txt file	.txt file

2.2 Class diagram and module decomposition

In the figure 2 we present the class diagram of our implementation. Each class is associated to the main class. Main class has main executable method that is private (it is not used by other classes and it uses functionality from other classes). Other classes (ReadTextAsString, Split, CircularShift, AlfabeticalSort, Output) provide operational functionality for the master class. In operational module we recognize two submodules. Input/Output data handler submodules: ReadTextAsString and Output. Operational submodule: AlphabeticalSort, CircularShift and Split.

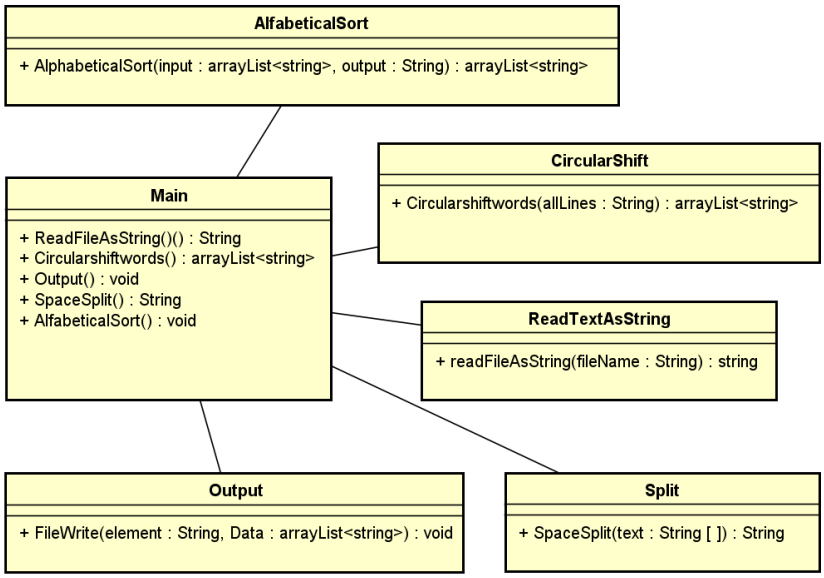

Figure 2: Class diagram

3 Conclusion

In this assignment we have made a decomposition structure of a implemented KWIC system. We presented our solution and type of architecture used. Decomposition structure provide layered insight into architecture design and flow between modules. Advantages of this approach are following: developers can divide their time and work on separate functionalities and afterwards just merge the code, code is highly modifiable, easy to understand. Code can be simply modified: in split function we can split words by some other character, different type of shift can be applied or different type of sorting. Furthermore, this approach is useful also for testing and verification, each module can be tested as separate unit of the system. One of the disadvantages of this architecture is that is not reusable. In addition, change in data storage format will influence on all of the modules.

4 References

[1] L. Bass, P. Clements, R. Kazman, ”Software Architecture in Practice”, 3rd edition, Addison-Wesley, 2012

[2] D.L. Parnas, “On the Criteria To Be Used in Decomposing Systems into Modules”, Comm. ACM, 1972, https://www.win.tue.nl/~wstomv/edu/2ip30/references/criteria_for_modularization.pdf