Homework 1: KWIC-KWAC-KWOC

Code Repository URL: https://github.com/culaucon/kwic-kwac-kwoc

Name	Nguyen Qui Hieu	Cao Luu Quang
Matric Number	A0083648U	A0084801J

1. Introduction

The objective of the project is to implement a simple Key Word In Context (referred to as KWIC) system, to provide a search mechanism for a long list of lines. The program takes in 2 lists: a list of titles, which are lines of text, and a list of “words to ignore”. From the input, each title will be “circularly shifted” exhaustively to create a list of all possible lines. Then, the lines with its starting word belonging in the list of “words to ignore” will be removed from the list, and the rest will be outputted to the users. For a line to be “circularly shifted”, the last word in the line will be cut out and be joined to the beginning of the current line. For example: for “A whole new world”, possible lines can be created are “whole new world a”, “new world a whole” etc.
For the implementation of this project, we will be using Java as the language of choice.

2. Design

For this project, we decide to implement a design similar to the “Modularization 1” in the “On the criteria to be used in decomposing systems into modules”. That is, we divide the workload into smaller portions, and implement them in separate modules, applying the Separation of Concern Principle.
We identify the main tasks are:

- Read the input from text file: the list of titles and the list of “word-to-ignore”.
- Generate the appropriate results from the data.
- Rearrange the data generated
- Display the processed data.

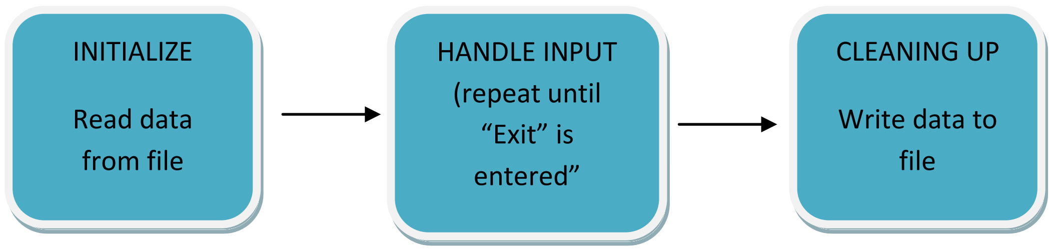
For our program, we offer a simple command-line interface with simple commands to edit and display the two lists, and to display the result list to the console. The program will continue to take in commands until the “exit” command is entered.
For handling of data:

- A class “File Operation” is created, as to handle all the file input and output.
- An interface of DataList is created, to serve as a generic type, as we will use two Lists of Strings in this program, both with implement the DataList data type.
- Two classes implementing the DataList data type: IgnoredList and MovieTitles. Both make use of the operations in “FileOperation” to get input from file and output to file.

- A “Shift” class. This will generate the “circularly shifted” strings from the input string. The class implements Comparable, so as the list of words generated can be sorted alphabetically.

For the main program, a “Control” class is created. Every module is set as Private, and the inner working of the program is hidden from other, following Open-Closed Principle.

The workflow of the program:



3. Limitations and Benefits of the Selected Design

The selected designs have its own merits, which are the reason why we decide to use it. However, it is not without limitations, as there are rooms to improve on it.

- Benefits:
    - The design is really intuitive. The program follows a set of operations that do not operate concurrently, it is intuitive to modularize each smaller task and design the program with pipe-and-filter approach.
    - The design is pretty secure. Most of the inner workings are set as private, so it is hard for other people to attack the program.
- Limitations:
    - The design lacks a GUI that would make the program become easier to use.
    - There are still loopholes in the program. For example, the two classes IgnoredList and MovieTitles take in data from a file whose filename is set as a constant in the main program. This can be modified to make the program to take in data from filename passed in as arguments.