Assignment 1: KWIC-KWAC-KWOC

Code Repository URL: https://github.com/zephinzer/cs3213-assignment-1.git

Name	Joseph Goh Seow Meng	Jessica Ho
Matriculation Number	A0087072A	A0085082E

1. Introduction

KWIC-KWAC-KWOC is an application which retrieves key words from sentences. The application takes in two sets of data, an 'ignored words' list as well as a 'titles' list. The 'titles' list is

2. Design

2.1 Notation

Classes are notated in **bold.**

Interfaces are notated in ***bold and italics.***

2.2 Overview

KWIC-KWAC-KWOC was designed using the Model-View-Controller design pattern where Model represents our data, View represents our user interface and Controller represents the processes and mechanisms in place to retrieve the data and send it to the View.

The Model component comprises of the classes **Data** and **FileStorage** which are implementations of the interfaces ***IData*** and ***IStorage*** respectively. **Data** handles information loaded into the application's memory space while **FileStorage** handles information retrieval from a file-based data source.

The View component is implemented with **ConsolePrint** only.

The Processor component comprises of **Processor** only.

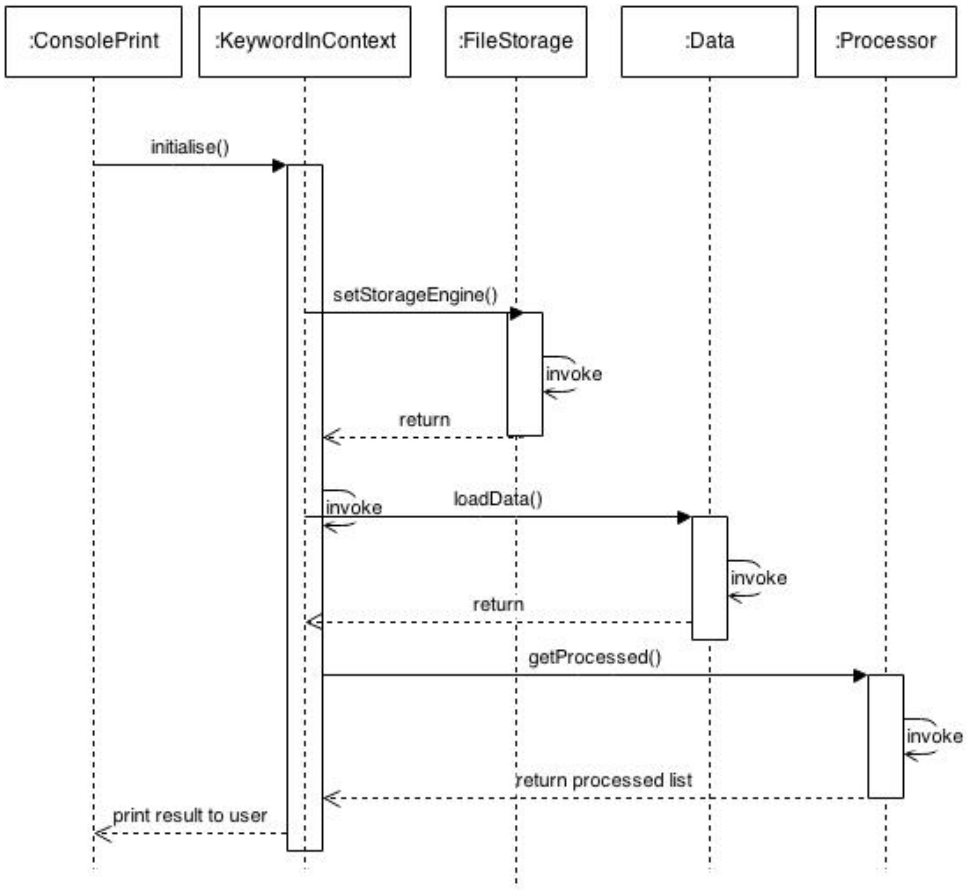

Fig 1 Basic overview of the system

2.3 Assignment of Work

Joseph: ***IData***, ***IStorage***, **Data**, **FileStorage**

Jessica: **ConsolePrint**, **Processor**, **KeyWordInContext**

3. Limitation & Benefits of Selected Design

3.1 Benefits

3.1.1 Code organization

Following the MVC design pattern allowed us to organize code more effectively, allowing our team as well as future developers who may come onto the project to understand the architecture more efficiently.

3.1.2 Extensibility

The MVC design pattern also allows us to build on code which has already been written. Should the need for a different data source be required, the ***IStorage*** interface contains the

API specifications needed for the new component to be written and integrated into our system.

3.1.3 Scalability

As the software progresses in terms of user base and data size, the MVC design pattern allows for individual component redesign without affecting the other components. Should our software prove to be successful, it is scalable depending on which components require upgrading.

3.1.4 Code re-use

Using the MVC design pattern with interfaces allows us to re-use code that has already been written in super-classes of our current code.

3.2 Limitations

The MVC pattern works for this software because there are clear layers which can be abstracted away, such as data access mechanisms, processing of data and presentation of data. However, following the MVC pattern to stringently may result in inefficient use of resources for a small scale software such as KWIC-KWAC-KWOC. For instance, the processing required for accessing data from two files, processing them and displaying the output can be more easily implemented in a pipes-and-filter manner where our 'ignored word's and 'title's are the data source and the processed words being our data sink.

That being said, the limitations of the MVC pattern lies in what are the limitations imposed by our requirements. Given a period of one day to complete this software, a pipes-and-filter approach might have suited the need because the code would be faster to write. Another factor which would affect our decision to use the MVC pattern is whether our software would be eventually integrated into a larger software as a core component. Should that be the case, using the MVC pattern results in increased workload for integration because the larger software would have it's own implementation of the MVC pattern and each component, the Model, View and Controller would have to be integrated separately as opposed to a pipes-and-filter pattern which could have consisted of one function.