COMP 1406 Intro to Computer Science II Fall 2021

Final Project: OPP Implementation of Search Engine

Karan Patel

Department of Computer Science, Carleton University

Comp 1406

David Mckenney

December 10th, 2021

Note all assigned functionalities have been completed as requested.

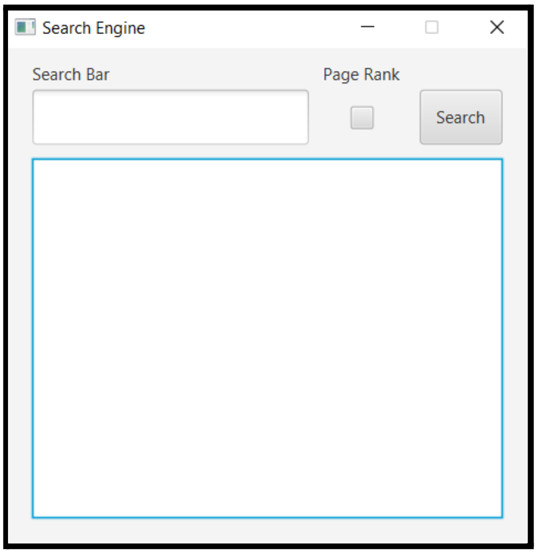
GUI Application:

The Search Engine Application(GUI) as seen below can be run from the SearchEngineApp.java file. This file is the controller class of the GUI. The application its self has 4 components:

1. A search bar: where the user can enter a query.
2. A check box: that indicates whether the user wants to have the page rank score contribute to the overall search results ranking.
3. A search button: which is required to be pressed by the user to indicate they want to search with the given information (update results).
4. A list view that lists the top 10 results from the given query and PageRank check box. The system can show X results however 10 was the given requirement.

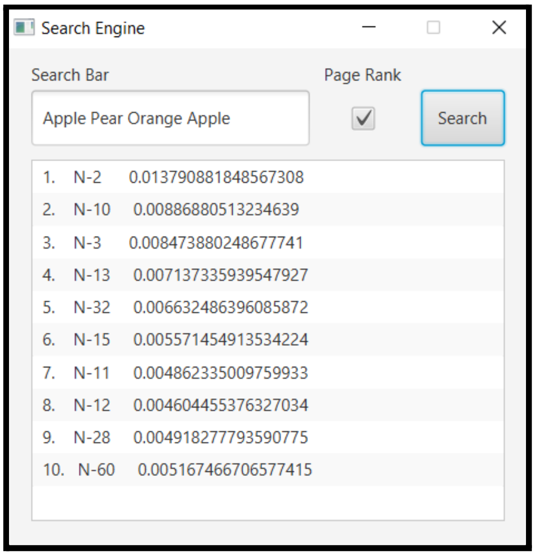
This particular model design is minimal and simple to use. I decided to use a checkbox as it is simple to use and uses less space whereas a radio button can confuse users and requires more space. I also decided to require the user to press the search button before updating the results as it is less distracting to the user and less error-prone compared to auto-update.

On Start-up



Page Rank □

Example Search



Search Bar
Apple Pear Orange Apple

Page Rank ☑

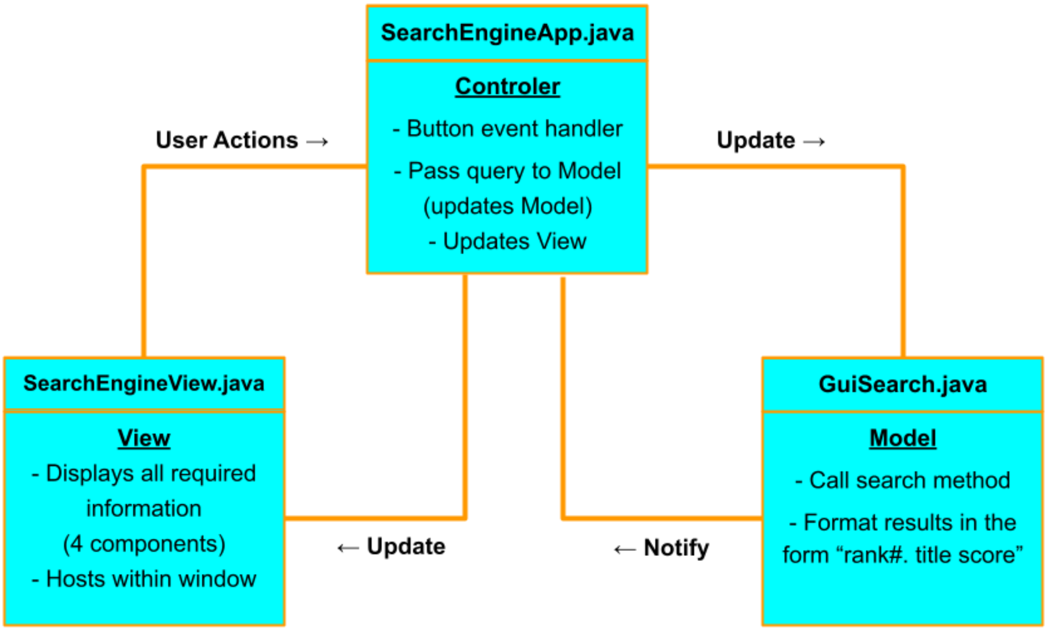
```
1.   N-2    0.013790881848567308
2.   N-10   0.00886880513234639
3.   N-3    0.008473880248677741
4.   N-13   0.007137335939547927
5.   N-32   0.006632486396085872
6.   N-15   0.005571454913534224
7.   N-11   0.004862335009759933
8.   N-12   0.004604455376327034
9.   N-28   0.004918277793590775
10.  N-60   0.005167466706577415
```

MVC Model:

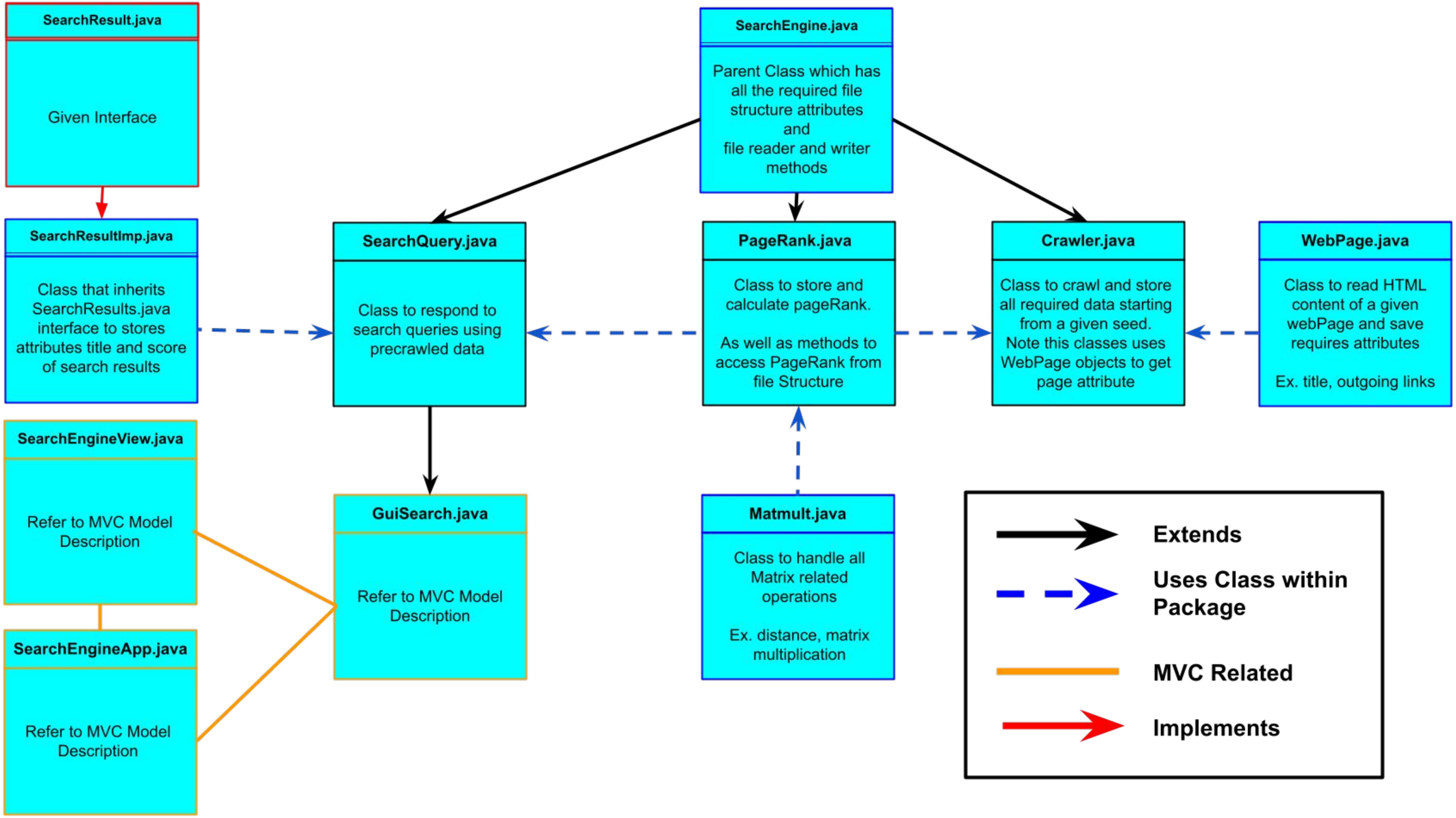
The GUI application uses the MVC model to ensure clean and efficient operations. My MVC model is comprised of 3 classes:

1. M(Model) - GuiSearch.java: This class is responsible for the logic of the application i.e. what are the results given the query. I decided to extend this class from SearchQuery.java class mainly because the output of the SearchEngineQuery.java does not match how I wanted the result information to be displayed within the view. In this class, I first take the query and get the new results by calling the search method in the SearchQuery.java class. I take these unformatted results and format them in the form "Rank#. Title Score" for example "1. N-0 0.98305930". This way all the formatting is done within the model and the view will not have to do any extra processing than required.
2. V(View) - SearchEngineView.java: This class displays all the necessary information the user will need to interact with, in this case, this would be the 4 components listed above hosted within a window.
3. C(Controller) - SearchEngineApp.java: This class accepts input from the user (through the view) and updates the model and view accordingly. This class will only perform this process when the event handler detects the user has pressed the search button.

As you can see below this is a basic diagram of how my MVC Model operates.



The Project Design



SearchResult.java

Given Interface

SearchResultImp.java

Class that inherits SearchResults.java interface to stores attributes title and score of search results

SearchEngineView.java

Refer to MVC Model Description

SearchEngineApp.java

Refer to MVC Model Description

SearchEngine.java

Parent Class which has all the required file structure attributes and file reader and writer methods

SearchQuery.java

Class to respond to search queries using precrawled data

GuiSearch.java

Refer to MVC Model Description

PageRank.java

Class to store and calculate pageRank.

As well as methods to access PageRank from file Structure

Matmult.java

Class to handle all Matrix related operations

Ex. distance, matrix multiplication

Crawler.java

Class to crawl and store all required data starting from a given seed. Note this classes uses WebPage objects to get page attribute

WebPage.java

Class to read HTML content of a given webPage and save requires attributes

Ex. title, outgoing links

Extends

Uses Class within Package

MVC Related

Implements

Crawler.java: This function is responsible for crawling the web starting from the given seed. This function acts more like a central command where it calls and coordinates all the operations smoothly. The Crawler uses Webpage.java, PageRank.java, parent methods(SearchEngine.java via inheritance), and its methods to perform this process. The reason I decided to have some processes and methods in different classes when even though they are only used in the crawling process is for modularity and extensibility purposes. PageRank.java, for example, is only used to calculate page rank and doesn't require any information from the crawler directly. This is why I believed it should not be limited to the crawler. As for the method within the crawler like "saveOutgoingAndIncomingLink()", they were highly optimized for the crawler which mean having them in different classes would not be beneficial later down the line. A quick overview of how the crawler operates is as follows: The crawler takes the given seed and constructs a WebPage object which then parses all the data. After that, the crawler will talk directly to the WebPage object to access all the information (title, URL, index…etc) and will store it within the file structure. The crawler will loop for all links until every reachable link is crawled. Note after one iteration of the loop the Webpage object is deleted and the new one is created meaning there will never be more than one active WebPage object(This is done to save memeroy).

Webpage.java: This class is a crucial part of the crawling process. The Crawler.java class will create objects of this class by passing the URL of the Webpage and the index. This class has 5 attributes:

1. URL(a string containing the webpage URL),
2. index(a string containing the webpage index),
3. title(a string containing the webpage title),
4. words(an array of strings containing all webpage words),
5. outgoingLinkList(an array of strings containing all webpages outgoing links)

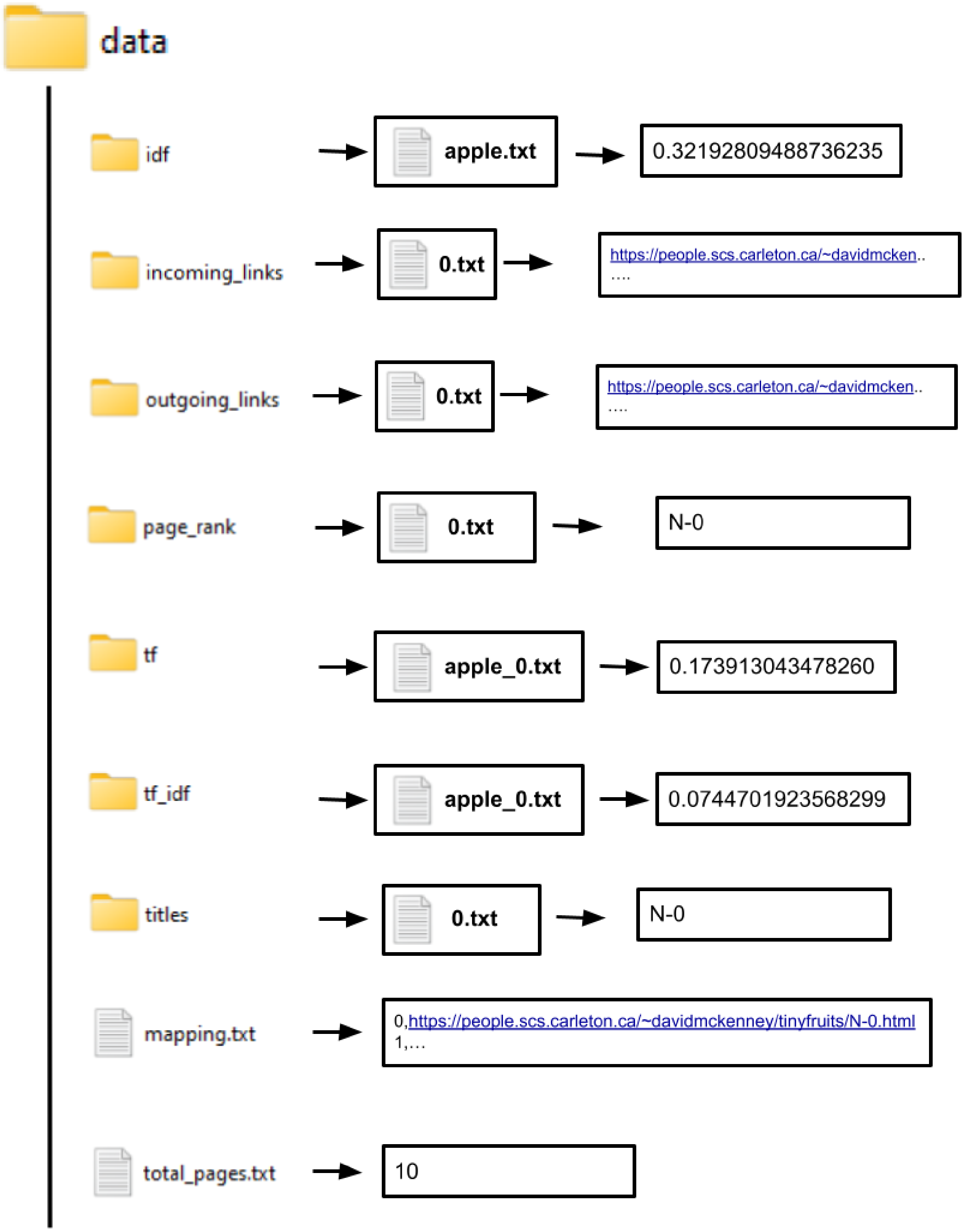
There is also a parse method within the class that uses the URL of the webpage object and retrieves all the attributes (i.e. title, words, outgoingLinkList) from the webpage by accessing its HTML content via WebRequester.java class. Once the HTML content is received the same processes as the 1405 projects are applied in terms of differentiating the attributes. I decided to create a webpage class as it makes sense to separate the gathering of data from storing and using the data. This is also useful if we ever wanted to access the content of a webpage outside the crawler.

File Structure:

The file structure of the crawler data is as follows

There are 3 types of file naming: Index.txt, Word.txt and Word_Index.txt

This is how I have decided to store all crawled data in a file structure.



PageRank.java: This class is solely to calculate PageRank. This is a separate class as it is not directly connected to the crawler. Once the "pageRankCalculator()" method is called the method accesses all the given information from the file structure, calculates page rank (with assists from Matmult.java methods), and stores it in the same file structure.

Matmult.java: This class only contains 2 static methods: the mult_matrix method which takes in 2 matrices and returns a multiplied matrix. And a euclidean_dist method that takes 2 vectors and returns the Euclidean distance of both vectors in the form of a double. This class is used only within the PageRank.java class but I decided to make it a class instead of having the 2 functions within the PageRank.java class for abstraction purposes. The mult_matrix and euclidean_dist methods are used by the PageRank method but are not directly related meaning they should be separate from the PageRank.java class.

SearchEngine.java: This class is the parent of Crawler.java, PageRank.java, and SearchQuery.java. This class holds all common attributes and methods used by the classes. Most of these common attributes have to do with file structure, reading and writing, which are required by all child classes. This class has 11 attributes and 4 methods and some testing related methods which are only meant for testing purposes.

1. All the attributes are related to file structure, for example, DATA_DIR is the directory path where all the other files will go.
2. "readDataValue()" takes in a file path and returns its content. This is particularly for files with 1 line of content.
3. "readLinks()" takes in a file path of either an outgoing or incoming links file and returns a string list of links.
4. "writeDataInFile()" takes in a file path and string content and adds the given content into the file.
5. "getMappingIndex()" takes in a URL and returns its mapping index from the mappingfile.txt.
6. "getData()" is a testing-related method that is only used by the tester. In the way my project is structured I did not require this method, however, it could be used in the future because it is very modular. This method is for returning the data value of either TF, IDF and TF_IDF by reading the corresponding file given the requirements. Here I used the method overloading as a way to accept a URL and not accept a URL because TF doesn't require a URL while the rest do.

One key thing I would like to point out is that this class was not originally the parent class to Crawler.java, SearchData.java and PageRank.java, it was originally in a file-helper class. This was done because at the time I found that not all these methods were used in each of its child classes, some of these methods were only used in the crawler(i.e. “writeDataInFile()”) and some were only used in the search (“readDataValue()”). However, after further examination, I found the file-helper class was used in the 3 child classes anyways(via static method) so it made sense to have them as a parent class. Even though not all the methods are used yet, for extensibility purposes it makes sense to do this. For example, if we ever needed to store all search queries (i.e. something like search history in chrome) we would need access to the “writeDataInFile()” method.

SearchResultImp.java: This class inherits the SearchResult class, it has the attributes title and score which can be accessed via its getter methods. Objects of the SearchResultImp.java class will be used polymorphically in the SearchQuery.java class.

SearchQuery.java: This class is responsible to respond to search queries. Similar to the crawler function, this function acts more like a central command where it calls and coordinates all the operations smoothly to perform search queries. This class is a child class of SearchEngine.java, it uses the methods and attributes of its parent to conduct search queries. Note since the requirement was to have this class return a list of SearchResult objects this class is defined to return a list of SearchResult. However, if you look at the code it returns a list of SearchResultImp.java objects, This works because of polymorphism since SearchResultImp.java class implements SearchResult.java. I did not need to cast the list of SearchResultImp.java objects in the return statement (like so “return (searchResults) …”) because java automatically changes as it is required SearchResult objects as a return type.

MVC Classes: GuiSearch.java, SearchEngineApp.java, SearchEngineView.java: Have already been discussed above in the MVC Model description.

Overall Benefits of OOP

One thing I would like to mention about OOP is how beneficial it is to the development of projects. After creating most of the search engine logic in the 1405 python project, recreating the same functionality using OOP was very easy and straightforward. The organizational ability and grouping of processes/attributes in a structure which is easy to understand ultimately helped me understand the reusability, scalability, and efficiency that OOP provides.

One way I completely changed the modularity aspects of my code is by having all my file reader, writer, and file structures saved in the parent class (SearchEngine.java). For example, within the SearchEngine.java class, I have an attribute called FILE_EXTENSION which is the file type of all the data files. As of right now, it is simply ".txt" but if we ever wanted to change this to ".dat" so the information can be encrypted we would just need to change this variable and the reader and writer methods in the parent class so that they write encrypted lines. Previously in the 1405 project, we would have to change every filename, reader, and writer function individually multiple times, now depending on the purpose, we can completely change the file type in a matter of seconds.

Another example of how I completely changed the design of the project is specifically within the Crawler.java and WebPage.java classes. In the previous project, all the gathering and storing of data was done in the crawler class. Now, these two processes are completely different, once a web page object is constructed and the pares method is called, all the data is gathered such as titles, words, outgoing links, etc and stored as attributes in the object. The crawler creates these WebPage objects and uses the attributes of the object to calculate and store the required information. In the 1405 project, this was a very confusing process as the HTML contents of the given URL were passed down throughout the whole crawler and every function did a different task. This leads it to be hard to differentiate the purpose of each function. However, in this project, it is simply the WebPage class's job once created and parsed to have all of the required information ready. The crawler's job is to calculate and store all the required information. This is very important because in the future we might want to have more complicated WebPages that might have HTML Tags, images, etc, which will require us to change the process of analyzing HTML content. This can easily be done in the webpage class without any interference with the crawler class. This is a key benefit of OOP.

These are just some examples of the benefits of OOP, however, there are tons more within this project.