Readme

Weizhou Ren
44939962

The data structure I chose is HashMap and my own linked list, because it is efficient. I implemented my own HashMap and collection called *NewMap* and *MySet*. Also, *NewEntry* is designed to implement the HashMap.
In my HashMap, there is an array called *bucketTable* which is used to store the entries. The key must be distinct in my HashMap, because it will be searched by its hash code.
How to determine where to put or find out the specific entry?
The position, which means the index that will store the entry is determined by using Math.abs(k.hashCode()) % capacity;
Such method is guarantee that the index number is in the range of capacity of the *bucketTable*. However, if this index has already stored an entry, the added entry should be linked follow the existing entry.
The idea of implementing my HashMap is that the expected search running time is O(1), because it is easy to get the hash code of the key and get it index of *bucketTable* by a short calculating. The shortage of my hash map is the capacity which should be defined when construct, which may have some wasted memory usage; Although the worst case is O(n), but if we have enough memory, it should be efficient to use this hash map because it would make the value stores evenly in the *bucketTable*.

As for *MySet*<T>, its essence is a linked list, but I defined my own method for implementing my data structure, such as, *getUnion*, *getIntersection*, etc. it can be used to do some Boolean search. The set can be iterated by for each loop, which is easy to get the element and the length of *MySet*<T> is dynamic.

Both *NewMap*<K, V> and *MySet*<T> have been used multiple times to implement my data structure.

```
private NewMap<String, MySet<Triple<Integer, Integer,
String>>> wordMap;
```

The main structure is *wordMap* which is used to store every word in document file and its positions. The positions should be stored as *MySet<Triple<Integer,Integer,String>>* which represent by row, column and title. Same words may occurrence multiple times. And in each set, the position is indicated by *Triple* which contains the row number, column number and its chapter name (from index file); Therefore, it is easy to search for a word and get its occurrences. So, the wordcount method is easy to get by getting the *size()* of is value;

```
private NewMap<Integer, String[]> rowMap;
```

rowMap stores every line number and the words in this line, line number is distinct, the words can be search by getting specific row number. It was used in *phraseOccurence* and wordsOnLine.

```
private NewMap<Integer, String> indexMap;
```

the *indexMap* is used to get chapter name. it is used to determine which chapter the words appear by counting its row when read the document file in my structure.

```
private MySet<String> stopSet;
```

stopSet is used to store all the stop words. If the specific word is contained in the set, it will be ignored and won't enter the further steps, except phraceOccurence;

```
private NewMap<Triple<Integer,Integer,String>,String>
simpleAndSection;
private NewMap<Triple<Integer,Integer,String>,String>
simpleOrSection;
```

Both *simpleAndSection* and *simpleOrSection* is helper map to implement *coumpoundAndOrSearch*, which stores the occurrences in required sections.