Given Task:

Description: Write a program that reads a file and finds matches against a predefined set of words. There can be up to 10K entries in the list of predefined words. The output of the program should look something like this:

Predefined word	**Match count**
FirstName	3500
LastName	2700
Zipcode	1601

Requirement details:

- Input file is a plain text (ascii) file, every record separated by a new line.
- For this exercise, assume English words only
- The file size can be up to 20 MB
- The predefined words are defined in a text file, every word separated by a newline. Use a sample file of your choice for the set of predefined keywords for the exercise.
- Assume that the predefined words file doesn't contain duplicates
- Maximum length of the word can be upto 256
- Matches should be case-insensitive
- The match should be word to word match, no substring matches.
- Consider a sample file with only the following two lines:

System Constraints:

1. Input file size upto 20 mb
2. Predefined set of words - upto 10k

Assumptions/Facts about output:

1. We only have single words in the file, Eg. University
2. Wouldn't is NOT same as Would (same for all words with ')
3. My code works even if Predefined cache is null or output file path exists already
4. The output consist of word in lowercase only, not how it was in input. Same as Predefined file.
5. We return the sorted order of words, in desc order
6. If a word is not present in Predefined, it will not be shown in final output
7. If there is a word in predefined, but absent in input, it will not be in final output

Calculations for PreDefined file:

1. Assume the text file has upto 10k records
2. We will be accessing this data frequently and comparing every input word. We can possibly convert it to dictionary/HashMap/HashTable, etc. But the most suitable data structure I felt was using a HashSet, since we are only storing words - HashMap also provides better lookup efficiency, when preDefined words are set as keys. But we have a trade off for space, as every key will have same value, (key: word, value: true).

Approach for PreDefined file:

1. We create a HashSet and populate it with our predefined data. We want to ensure that both Mapper and Reducer tasks have access to cache. If our cache is filled upto its capacity, we re-hash it depending on the load factor value provided.

Calculations for Input file:

1. Assume, each sentence is approx 100 chars long, there are almost 2,00,000 sentences in a 20 MB file.
2. Assume there are at least 10 words in each sentence, total number of words that have to be compared (number of input words) is approx 20,00,000

Approach for Input file processing:

1. Between Multithreading and Map-Reduce, latter is the clear choice of implementation. We have several advantages of using Map-Reduce, but the most important ones I considered for this task are - Scalability, parallel processing of the input task file and resiliency (Node restart/ failures can be handled better) instead of Multi threading.
2. Even before we start the input file processing, we have our cache ready.
3. The mapper task will split the file for parallel processing and each of the partitions will be treated separately.
4. We filter the words using cache in our Mapper task, so we pass less words to Reducer, to avoid network overhead issues due to multiple data.
5. Once we are in our reducer task, we just sum up the frequencies of each words, passed as output of previous mapper task.

Why did I use maxheap in Reducer?

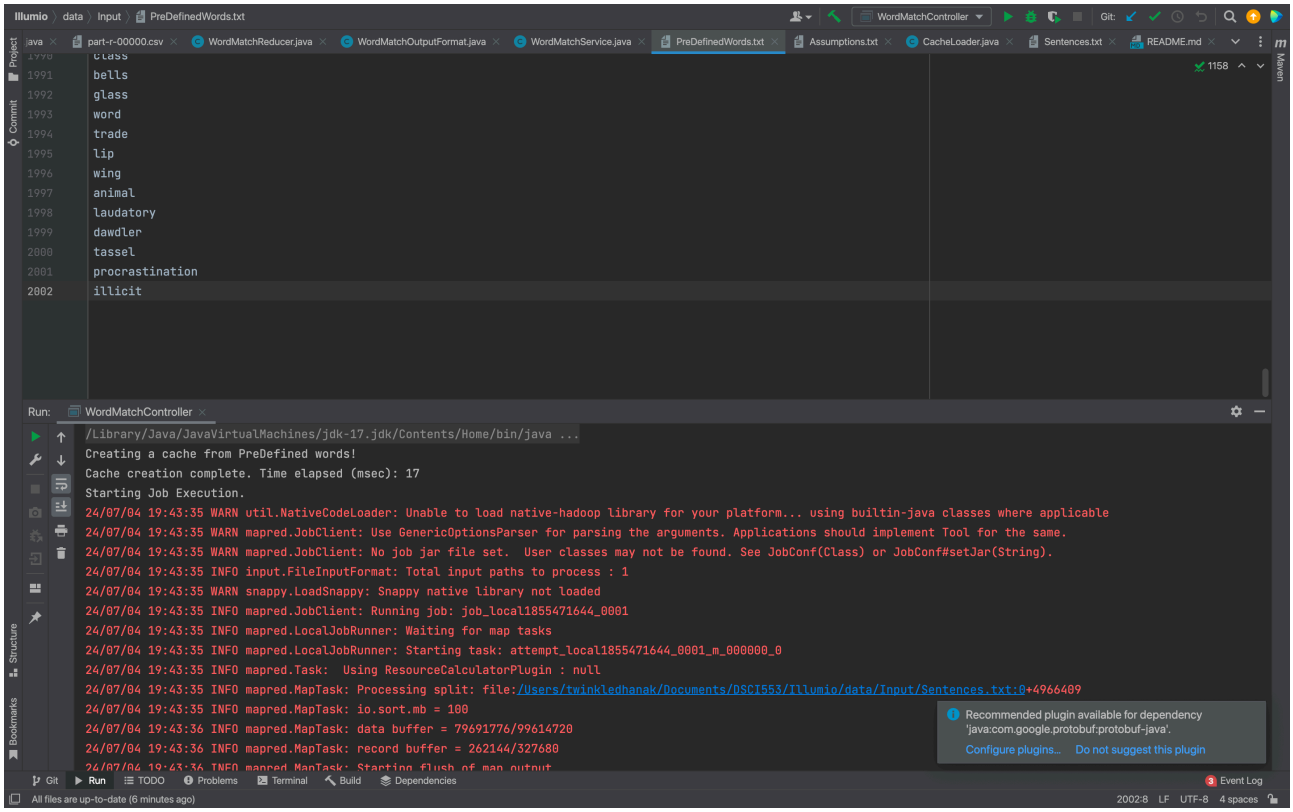
1. As per the given requirement, the data appears to be sorted in descending order of frequency.
2. If we maintain a Hashmap of all pairs and then try to sort it at the time of writing the output, we might have to sort a map of 10k record which is not good approach.
3. We can use TreeMap to maintain a sorted list of frequency of words, but, I found Maxheap a better solution for the same.
4. Using maxheap, in my Reducer, I am maintaining the list of output words from reducer in a sorted but desc order (property of max heap).

Final Output modification:

1. My initial thought was to write the output in a file directly after the Reduce() task is complete.
2. It is not a good idea for two reasons - there are multiple I/O and file handling operations in reducer which impacts performance and it is not a good idea to write in a file, when hdfs already provided a output, in best and efficient way possible.
3. I had to add a custom format class to add the column headers in the output file.
4. If we wish to have output as .csv or .txt, we can change the extension value in file Constants.java

```
public static final String OUTPUT_FILE_EXTENSION = ".csv";
```

The Output looks like:
(Scenario 2 in Readme file)



Illumio 〉 data 〉 Input 〉 PreDefinedWords.txt

```
bells
glass
word
trade
lip
wing
animal
laudatory
dawdler
tassel
procrastination
illicit
```

Run: WordMatchController

```
24/07/04 19:43:36 INFO mapred.JobClient:     Reduce input groups=64
24/07/04 19:43:36 INFO mapred.JobClient:     Combine output records=64
24/07/04 19:43:36 INFO mapred.JobClient:     Reduce output records=64
24/07/04 19:43:36 INFO mapred.JobClient:     Map output records=89520
24/07/04 19:43:36 INFO mapred.JobClient:     Combine input records=89520
24/07/04 19:43:36 INFO mapred.JobClient:     Total committed heap usage (bytes)=589299712
24/07/04 19:43:36 INFO mapred.JobClient:   File Input Format Counters
24/07/04 19:43:36 INFO mapred.JobClient:     Bytes Read=4966409
24/07/04 19:43:36 INFO mapred.JobClient:   FileSystemCounters
24/07/04 19:43:36 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=105231
24/07/04 19:43:36 INFO mapred.JobClient:     FILE_BYTES_READ=9934083
24/07/04 19:43:36 INFO mapred.JobClient:   File Output Format Counters
24/07/04 19:43:36 INFO mapred.JobClient:     Bytes Written=779
Job execution complete. Time elapsed (msec): 1418

Process finished with exit code 0
```

Recommended plugin available for dependency 'java:com.google.protobuf:protobuf-java'.
Configure plugins... Do not suggest this plugin

All files are up-to-date (6 minutes ago)

Illumio 〉 data 〉 Input 〉 Sentences.txt

The file size (4.97 MB) exceeds the configured limit (2.56 MB). Code insight features are not available.

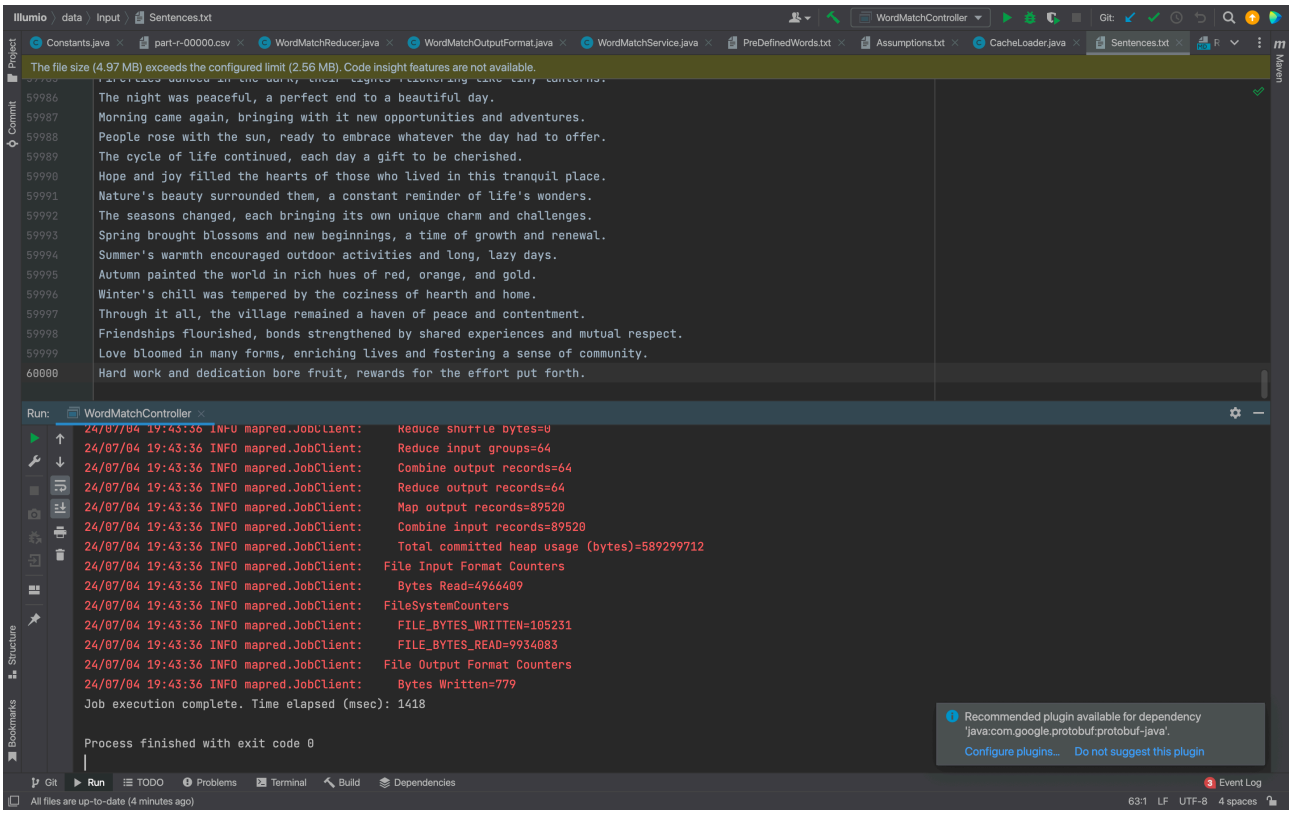
```
The night was peaceful, a perfect end to a beautiful day.
Morning came again, bringing with it new opportunities and adventures.
People rose with the sun, ready to embrace whatever the day had to offer.
The cycle of life continued, each day a gift to be cherished.
Hope and joy filled the hearts of those who lived in this tranquil place.
Nature's beauty surrounded them, a constant reminder of life's wonders.
The seasons changed, each bringing its own unique charm and challenges.
Spring brought blossoms and new beginnings, a time of growth and renewal.
Summer's warmth encouraged outdoor activities and long, lazy days.
Autumn painted the world in rich hues of red, orange, and gold.
Winter's chill was tempered by the coziness of hearth and home.
Through it all, the village remained a haven of peace and contentment.
Friendships flourished, bonds strengthened by shared experiences and mutual respect.
Love bloomed in many forms, enriching lives and fostering a sense of community.
Hard work and dedication bore fruit, rewards for the effort put forth.
```

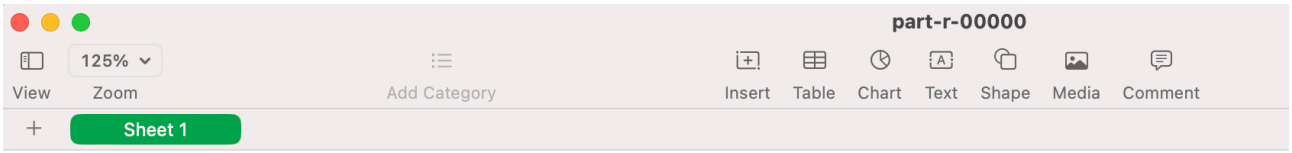
Run: WordMatchController

```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java ...
Creating a cache from PreDefined words!
Cache creation complete. Time elapsed (msec): 17
Starting Job Execution.
24/07/04 19:43:35 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
24/07/04 19:43:35 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
24/07/04 19:43:35 WARN mapred.JobClient: No job jar file set.  User classes may not be found. See JobConf(Class) or JobConf#setJar(String).
24/07/04 19:43:35 INFO input.FileInputFormat: Total input paths to process : 1
24/07/04 19:43:35 WARN snappy.LoadSnappy: Snappy native library not loaded
24/07/04 19:43:35 INFO mapred.JobClient: Running job: job_local1855471644_0001
24/07/04 19:43:35 INFO mapred.LocalJobRunner: Waiting for map tasks
24/07/04 19:43:35 INFO mapred.LocalJobRunner: Starting task: attempt_local1855471644_0001_m_000000_0
24/07/04 19:43:35 INFO mapred.Task:  Using ResourceCalculatorPlugin : null
24/07/04 19:43:35 INFO mapred.MapTask: Processing split: file:/Users/twinkledhanak/Documents/DSCI553/Illumio/data/Input/Sentences.txt:0+4966409
24/07/04 19:43:35 INFO mapred.MapTask: io.sort.mb = 100
24/07/04 19:43:36 INFO mapred.MapTask: data buffer = 79691776/99614720
24/07/04 19:43:36 INFO mapred.MapTask: record buffer = 262144/327680
24/07/04 19:43:36 INFO mapred.MapTask: Starting flush of map output
```

Recommended plugin available for dependency 'java:com.google.protobuf:protobuf-java'.
Configure plugins... Do not suggest this plugin

All files are up-to-date (4 minutes ago)





part-r-00000

Predefined word	Match count
love	6750
hope	6270
day	5910
new	5370
future	4320
nature	3870
resilience	3840
moment	3840
strength	3360
peace	2910