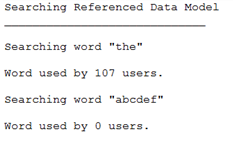
**Project Results**

The project results are displayed in the VM terminal itself as I print out all the output in Java code. The purpose of this documents is to interpret the results of the performance of the data models.

1. Searching a high frequency word:



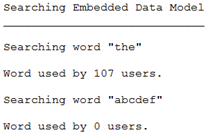
The program searches for a high frequency word “The” which is expected to be most common among tweets and the search result returns the word being used by 107 users. As it is expected the Embedded data model will also return the same results but let us take a look at the time.





So as we see here, there is a major difference in the time, as explained in the code documentation this a result of the referenced data model searching 2 collections to get the results, while on the other hand in the embedded data models aggregated output collection only once search is sufficient.

1. Searching a low frequency word:



Low frequency word abcdef is too rare to be used in tweets so as expected the users using word in their tweet is 0. Let us compare the timings.





In this scenario the referenced data model wins. In the code documentation as it mentions in the referenced data model the tweets collection is searched and when the word is not found, the list of output which should contain the UserIDs is empty so a simple if condition check exits the method of Embeddded data model. While on the other hand in Embedded data model the program searches in all tweets sub-document in the same collection ”reduced”, it even searches the empty tweets sub-document which is not mapped and hence the time difference. But it is noticed that the timings for embedded improves after consecutive runs.

Sample run after reducing data files:

