## Design Decision

The design decision for this assignment is to use 2 jobs, each containing a set of Mapper and Reducer to retrieve the expected output. The description of each class in the source code are as followed:

### Job 1:

BDMapper – The objective of this class is to be able to read the text inside the given data file. It first scan each line of the data file and look if the line starts with "REVISION". If the file starts with revision, it will then store the article_id and the timestamp of the particular article. The mapper will then check and store the article_id and its modification counter (1) if the article is created in the given time range stated by the user input.

BDReducer (Combiner) – The combiner will consolidate the results obtained from the mapper so that each article_id is tagged with the total number of modification.

BDReducer (Reducer) – The reducer will obtain the data from combining and arranged it into ascending article_id. After the data is successfully reduced, it is being outputted in HDFS. However, the given data does not satisfy the requirement of the Assessed Exercise, thus creating another job.

### Job 2:

BDMapperTwo – The 2nd Mapper is created to read in the output file created from the 1st Reducer and create a KeyValuePair object and stores it as key, the value is not of an importance, thus storing a dummy value so that the mapper would be able to sort KeyValuePair object.

KeyValuePair – It is an object class to store key and value. There is an inbuilt sorting function, but it is not as efficient as raw comparator.

KVPComparator – By introducing raw comparator, the time taken to sort the KeyValuePair significantly decreased and it shows that raw comparator is more efficient than the one build in KeyValuePair. The comparator will sort KeyValuePair by its value in descending order, then sort article_id in ascending order

BDReducerTwo – This class is used to tabulate top K result obtained from the sorted mapper class. BDReducerTwo is explicitly set to 1 so as to prevent sorted result from getting messy in multiple output file, thus fulfilling the assignment specification.

## Performance evaluation

| Top K = 50 | Run 1 | Run 2 | Run 3 | Average | Standard Deviation |
|---|---|---|---|---|---|
| Query Process Time (ms) | 8350774 | 4285554 | 4574615 | 5736981 | 2268220.547 |
| No of bytes read from HDFS | 31290383266 | 31290383266 | 31290383266 | 31290383266 | 0 |
| No of bytes transferred over the network | 193279912 | 193279912 | 193279912 | 193279912 | 0 |

| Top K = 500 | Run 1 | Run 2 | Run 3 | Average | Standard Deviation |
|---|---|---|---|---|---|
| Query Process Time (ms) | 10645928 | 12222496 | 12422345 | 11763589.67 | 973067.6 |
| No of bytes read from HDFS | 31290383266 | 31290383266 | 31290383266 | 31290383266 | 0 |
| No of bytes transferred over the network | 193279912 | 193279912 | 193279912 | 193279912 | 0 |

Scalability: The program not only able to run against enwiki-20080103-small.txt (100MB), it is also able to tabulate results of the other given data i.e. enwiki-20080103-perftest.txt (300GB). The program can explicitly set a mapper and reducer from its default number to 1.