```java
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import java.util.HashSet;
import java.util.Set;
import java.util.Map;


/*
 *       YOUSSEF NIDABRAHIM
 *  ZZ3 - F2
 *
 *       MAP/REDUCE PATTERN FOR COUNTING WORDS
 */


public class WordCount {

  public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWrita
ble>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    private Set<String> patternsToSkip = new HashSet<String>();


    public void map(Object key, Text value, Context context) throws IOException,
 InterruptedException {

      StringTokenizer itr = new StringTokenizer(value.toString());

      while (itr.hasMoreTokens()) {
                        word.set(itr.nextToken());
                        String txt = word.toString();
                        txt = txt.toLowerCase();
                        txt = txt.replaceAll("\p{Punct}", "");
                        //txt = txt.replaceAll("[^a-zA-Z\\p{L}]", " ");
                        word.set(txt);
                        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer extends Reducer<Text,IntWritable,Text,IntWri
table> {

    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values, Context context )
 throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
```

```java
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");

    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

```java
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import java.util.HashSet;
import java.util.Set;
import java.util.Map;

/*
 *      YOUSSEF NIDABRAHIM
 *  ZZ3 - F2
 *
 *      MAP/REDUCE PATTERN FOR COUNTING ANAGRAMMES
 */



public class Anagrammes {

  public static class AnagrammeMapper extends Mapper<Object, Text, Text, Text>{

    public void map(Object key, Text value, Context context) throws IOException,
 InterruptedException {

                char[] letters = value.toString().toLowerCase().toCharArray();
                Arrays.sort(letters);

                context.write(new Text(new String(letters)), value);
    }
  }

  public static class AnagrammeReducer extends Reducer<Text,Text,Text,Text> {

    public void reduce(Text key, Iterable<Text> values, Context context ) throws
 IOException, InterruptedException {

                Iterator<Text> i = values.iterator();
                String result = "";
                Boolean first = true;
                while(i.hasNext()){
                        if(first){
                                result = i.next().toString();
                                first = false;
                        }else
                                result = result+"|"+i.next().toString();
                }
                context.write(key, new Text(result));
    }
  }

  public static void main(String[] args) throws Exception {

    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "anagrammes counter");
```

```java
    job.setJarByClass(Anagrammes.class);
    job.setMapperClass(AnagrammeMapper.class);
    job.setReducerClass(AnagrammeReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

```java
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import java.util.HashSet;
import java.util.Set;
import java.util.Map;


/*
 *      YOUSSEF NIDABRAHIM
 *  ZZ3 - F2
 *
 *      MAP/REDUCE PATTERN FOR CALCULATING MAXIMUM AND AVERAGE WORDS PER SENTENC
E
 */


public class StylePhrase {

  public static class StylePhraseMapper extends Mapper<Object, Text, Text, Custom
MaxAverageTuple>{

        private CustomMaxAverageTuple tuple = new CustomMaxAverageTuple();
        private Text sentence = new Text("Sentence");

    public void map(Object key, Text value, Context context) throws IOException,
 InterruptedException {

          StringTokenizer itr = new StringTokenizer(value.toString());
      long wordCounter = 0;
      while (itr.hasMoreTokens()) {
                        wordCounter++;
      }
      tuple.setAverage(wordCounter);
      tuple.setMax(wordCounter);
      tuple.setCount(wordCounter);

          context.write(sentence, tuple);
    }
  }

  public static class StylePhraseReducer extends Reducer<Text,CustomMaxAverageTup
le,Text,CustomMaxAverageTuple> {

        private CustomMaxAverageTuple result = new CustomMaxAverageTuple();

    public void reduce(Text key, Iterable<CustomMaxAverageTuple> values, Context
 context ) throws IOException, InterruptedException {

          int max = 0;
          int moy = 0;
          int wordCounter = 0;
          int sentenceCounter = 0;
```

```java
      for (CustomMaxAverageTuple tuple : values) {
                  sentenceCounter++;
                  wordCounter = wordCounter + tuple.getCount();
                  result.setCount(wordCounter);
                  if(tuple.getMax() > result.getMax())
                        result.setMax(tuple.getMax());
      }
      result.setAverage(result.getCount()/sentenceCounter);

      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {

    Configuration conf = new Configuration();
    conf.set("textinputformat.record.delimiter", ".");
    Job job = Job.getInstance(conf, "Style sentences");

    job.setJarByClass(StylePhrase.class);
    job.setMapperClass(StylePhraseMapper.class);
    job.setReducerClass(StylePhraseReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(CustomMaxAverageTuple.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

```
import org.apache.hadoop.io.Writable;


/*
 *      YOUSSEF NIDABRAHIM
 *  ZZ3 - F2
 *
 *      WRITABLE OBJECT THAT STORES THREES VALUES
 */

public class CustomMaxAverageTuple implements Writable {

        private Double average = new Double(0);
        private Double max = new Double(0);
        private long count = 1;


        public Double getAverage() {
                return average;
        }

        public void setAverage(Double average) {
                this.average = average;
        }

        public Double getMax() {
                return max;
        }

        public void setMax(Double max) {
                this.max = max;
        }

        public long getCount() {
                return count;
        }

        public void setCount(long count) {
                this.count = count;
        }

        public String toString() {
                return average + "\t" + max + "\t" + count;
        }

}
```