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

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
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;

public class Assignment1 {

public static class WordCountMapper
extends Mapper < LongWritable, Text, Text, IntWritable >
{

private final static IntWritable one = new IntWritable( 1);
private Text word = new Text();

@Override
public void map( LongWritable key, Text value, Context context
) throws IOException,
      InterruptedException
      {
                  //Splitting the content/value based on spaces(could be
one or more spaces)
                  String Temp[] = value.toString().split("\\s+");
                  String Finalwordsequence ="";
                  for(int i=4;i<Temp.length;i++)
                  {
                              //omitting null values/ empty strings
                              if(Temp[i-
4].equalsIgnoreCase("null")==false&&Temp[i-
4].equalsIgnoreCase(null)==false&&Temp[i-4].equalsIgnoreCase("
")==false&&Temp[i-4].equalsIgnoreCase("")==false&&Temp[i-
4].equalsIgnoreCase("\"")==false
                                                            &&Temp[i-
3].equalsIgnoreCase("null")==false&&Temp[i-
3].equalsIgnoreCase(null)==false&&Temp[i-3].equalsIgnoreCase("
")==false&&Temp[i-3].equalsIgnoreCase("")==false&&Temp[i-
3].equalsIgnoreCase("\"")==false
                                                            &&Temp[i-
2].equalsIgnoreCase("null")==false&&Temp[i-
2].equalsIgnoreCase(null)==false&&Temp[i-2].equalsIgnoreCase("
")==false&&Temp[i-2].equalsIgnoreCase("")==false&&Temp[i-
2].equalsIgnoreCase("\"")==false
                                                            &&Temp[i-
1].equalsIgnoreCase("null")==false&&Temp[i-
1].equalsIgnoreCase(null)==false&&Temp[i-1].equalsIgnoreCase("
```

```
")==false&&Temp[i-1].equalsIgnoreCase("")==false&&Temp[i-
1].equalsIgnoreCase("\"")==false

      &&Temp[i].equalsIgnoreCase("null")==false&&Temp[i].equalsIgnoreCase(
null)==false&&Temp[i].equalsIgnoreCase("
")==false&&Temp[i].equalsIgnoreCase("")==false&&Temp[i].equalsIgnoreCase("
\"")==false)
                                        Finalwordsequence = Temp[i-4]+"
"+Temp[i-3]+" "+Temp[i-2]+" "+Temp[i-1]+" "+Temp[i];
                                System.out.println("words:
"+Finalwordsequence);
                                //Final check to filter out only 5 word
sequences using length
                                String s[] = Finalwordsequence.split(" ");
                                int length = s.length;
                                if(length==5)
                                {
                                word.set(Finalwordsequence);
                                System.out.println("words:
"+Finalwordsequence);
                                context.write( word, one);
                                }
                        }
      }
}
public static class WordCountReducer
extends
Reducer < Text, IntWritable, Text, IntWritable > {
private IntWritable result = new IntWritable();
@Override
public void reduce( Text key, Iterable < IntWritable > values, Context
context
) throws IOException,
InterruptedException {
int sum = 0;
for (IntWritable val : values) {
sum += val.get();
}
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(Assignment1.class);

FileInputFormat.addInputPath( job, new Path("input"));
FileOutputFormat.setOutputPath( job, new Path("output"));
job.setMapperClass( WordCountMapper.class);
job.setCombinerClass( WordCountReducer.class);
job.setReducerClass( WordCountReducer.class);
```

```
job.setOutputKeyClass( Text.class);
job.setOutputValueClass( IntWritable.class);

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