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;

**public class WordCount** {

 public static class TokenizerMapper

 extends Mapper<Object, Text, Text, IntWritable>{

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

 private Text word = new Text();

 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());

 context.write(word, one);

 }

 }

 }

 **public static class IntSumReducer**

 extends Reducer<Text,IntWritable,Text,IntWritable> {

 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();

 }

 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.setCombinerClass(IntSumReducer.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);

 }

}