//package org.apache.hadoop;

import java.io.IOException;

import java.util.\*;

import org.apache.hadoop.conf.\*;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

// Out of (LongWritable, Text, Text, IntWritable) in below line, first 2 are used for map input (K1,V1) and last 2 are for map output(K2,V2).

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

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

private Text word = new Text();

// Below line will have K1,V1 as input and the Context as collector. We can have one extra parameter i.e. Reporter which is to display the

// Program progress on the console. See this in Advanced Map reduce.

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

// Changing vale in above line into String (in below line)

String line = value.toString();

// Splitting the line into tokens using space as delimeter

StringTokenizer tokenizer = new StringTokenizer(line);

while (tokenizer.hasMoreTokens()) {

word.set(tokenizer.nextToken());

// Write the output as welcome,1

context.write(word, one);

}

}

}

// AND Then above output will be used by shuffle and sort and below NEW Output will be generated and sent to reducer

// Count,[1,1,1,1]

// So REDUCERS INPUT WILL BE

// count,[1,1,1,1]

//first 2 datatypes of below line come from map output which is like - k2,v2 i.e. count,1 and last 2 are k3,v3

public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {

// Below line will have K2,V2 as input. V2 is a list so we are using Iterable.

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

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

context.write(key, new IntWritable(sum));

}

}

// output will be Count,6

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

// Below we are providing our job configurations.

Configuration conf = new Configuration();

// Passing the name of our job (wordcount) to Configuration.

//conf.setJobName("mywc");

@SuppressWarnings("deprecation")

Job job = new Job(conf, "wordcount");

//if we dont use below line we will get error like -

//Error: java.lang.RuntimeException: java.lang.ClassNotFoundException: Class in.edureka.mapreduce.WordCounts$Map not found

job.setJarByClass(WordCount.class);

// Passig our mapper and reducer class names. class name should be exactly same.

job.setMapperClass(Map.class);

job.setReducerClass(Reduce.class);

// We used hadoop wrapper data types, so need to provide that configurations i.e. Text.class and IntWritable.class .

// Telling key is String|Text and Value is INT. Both mappers and reducers ouput is Text,IntWritable say welcome,1

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

// Passing the output InpuFormat. By default it is TextInputFormat- so we can skip this step.

// job.setInputFormatClass(TextInputFormat.class);

// job.setOutputFormatClass(TextOutputFormat.class);

//passing the input/output file/dir names as arguments. first argument input path and second argument output directory.

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

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

//Executing our job.

job.waitForCompletion(true);

}

}