**package** package4;

**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.Text;

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

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

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

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

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

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

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

**import** org.apache.hadoop.util.GenericOptionsParser;

**public** **class** WordCount

{

**public** **static** **class** MapForWordCount **extends** Mapper<LongWritable, Text, Text, IntWritable>

{

**public** **void** map(LongWritable key, Text value, Context con) **throws** IOException, InterruptedException

{

String line = value.toString();

StringTokenizer token = **new** StringTokenizer(line);

**while**(token.hasMoreTokens())

{

String status = **new** String();

String word = token.nextToken();

Text outputKey = **new** Text(word);

IntWritable outputValue = **new** IntWritable(1);

con.write(outputKey, outputValue);

}

} // end of map()

} //end of Mapper Class

/\*

sample output of the mapper phase :

<I , <1,1,1,1,1>>

<hadoop, <1,1,1>>

\*/

**public** **static** **class** ReduceForWordCount **extends** Reducer<Text, IntWritable, Text, IntWritable>

{

**public** **void** reduce(Text word, Iterable<IntWritable> values, Context con) **throws** IOException, InterruptedException

{

**int** sum = 0;

**for**(IntWritable value : values)

{

sum += value.get();

}

con.write(word, **new** IntWritable(sum));

} // end of reduce()

} // end of Reducer class

/\*

sample output of the reducer

hadoop 3

I 5

\*/

// job definition

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

{

Configuration c = **new** Configuration();

String[] files = **new** GenericOptionsParser(c, args).getRemainingArgs();

Path input = **new** Path(files[0]);

Path output = **new** Path(files[1]);

Job j = **new** Job(c, "wordcount");

j.setJarByClass(WordCount.**class**);

j.setMapperClass(MapForWordCount.**class**);

j.setReducerClass(ReduceForWordCount.**class**);

j.setOutputKeyClass(Text.**class**);

j.setOutputValueClass(IntWritable.**class**);

FileInputFormat.*addInputPath*(j, input);

FileOutputFormat.*setOutputPath*(j, output);

System.*exit*(j.waitForCompletion(**true**) ? 0:1);

} // end of main()

} //end of main class

Ip

Word.txt

Stored in Parth prroject