WORD COUNT

```
import java.io.IOException;
import java.util.regex.Pattern;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;

import org.apache.log4j.Logger;

public class WordCount extends Configured implements Tool {

  private static final Logger LOG = Logger.getLogger(WordCount.class);

  public static void main(String[] args) throws Exception {
    int res = ToolRunner.run(new WordCount(), args);
```

```
  System.exit(res);
}

public int run(String[] args) throws Exception {
  Job job = Job.getInstance(getConf(), "wordcount");
  job.setJarByClass(this.getClass());
  FileInputFormat.addInputPath(job, new Path("input.txt"));
  FileOutputFormat.setOutputPath(job, new Path("output"));
  job.setMapperClass(Map.class);
  job.setReducerClass(Reduce.class);
  job.setOutputKeyClass(Text.class);
  job.setOutputValueClass(IntWritable.class);
  return job.waitForCompletion(true) ? 0 : 1;
}

public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();
  private long numRecords = 0;
  private static final Pattern WORD_BOUNDARY = Pattern.compile("\\s*\\b\\s*");

  public void map(LongWritable offset, Text lineText, Context context)
      throws IOException, InterruptedException {
    String line = lineText.toString();
    Text currentWord = new Text();
    for (String word : WORD_BOUNDARY.split(line)) {
      if (word.isEmpty()) {
```

```
            continue;
        }
            currentWord = new Text(word);
            context.write(currentWord,one);
        }
    }
  }

  public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
    @Override
    public void reduce(Text word, Iterable<IntWritable> counts, Context context)
        throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable count : counts) {
        sum += count.get();
      }
      context.write(word, new IntWritable(sum));
    }
  }
}
```

CHARACTER COUNT

```
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.io.IOException;


public class CharacterCount {


    public static class CharCountMapper extends Mapper<Object, Text, Text,
IntWritable> {

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

        private Text character = new Text();
```

```
        public void map(Object key, Text value, Context context) throws
IOException, InterruptedException {
            String line = value.toString();
            for (char c : line.toCharArray()) {
                character.set(new String(new char[]{c}));
                context.write(character, one);
            }
        }
    }

    public static class CharCountReducer 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, "Character Count");
    job.setJarByClass(CharacterCount.class);
    job.setMapperClass(CharCountMapper.class);
    job.setCombinerClass(CharCountReducer.class);
    job.setReducerClass(CharCountReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path("input.txt"));
    FileOutputFormat.setOutputPath(job, new Path("output"));

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

WEATHER MAP REDUCE

```
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.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WeatherMapReduce {
public static class WeatherMapper
extends Mapper<LongWritable, Text, Text, DoubleWritable>{
private final static DoubleWritable temperature = new DoubleWritable();
private Text date = new Text();
public void map(LongWritable key, Text value, Context context
) throws IOException, InterruptedException {
String[] line = value.toString().split(",");
if (line.length == 3) {
date.set(line[0]);
temperature.set(Double.parseDouble(line[2]));
context.write(date, temperature);
}
}
}
```

```
public static class WeatherReducer
extends Reducer<Text,DoubleWritable,Text,DoubleWritable> {
private DoubleWritable result = new DoubleWritable();
public void reduce(Text key, Iterable<DoubleWritable> values,
Context context
) throws IOException, InterruptedException {
double sum = 0;
int count = 0;
for (DoubleWritable val : values) {
sum += val.get();
count++;
}
double avg = sum / count;
result.set(avg);
context.write(key, result);
}
}
public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
Job job = Job.getInstance(conf, "weather analysis");
job.setJarByClass(WeatherMapReduce.class);
job.setMapperClass(WeatherMapper.class);
job.setReducerClass(WeatherReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(DoubleWritable.class);
FileInputFormat.setInputPaths(job, new Path("input.txt"));
FileOutputFormat.setOutputPath(job, new Path("output"));
```

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

SORT STUDENT NAMES

```
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

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 SortStudNames {

public static class SortMapper extends Mapper <LongWritable, Text, Text, Text >

{

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

String[] token = value.toString().split(",");

context.write(new Text(token[1]), new Text(token[0]+ "-" +token[1]));
```

```
}
}
public static class SortReducer extends Reducer <Text, Text, NullWritable, Text>
{
public void reduce(Text key, Iterable <Text> values, Context context) throws IOException, InterruptedException {
for (Text details : values )
{
context.write(NullWritable.get(), details);
}
}
}
public static void main (String[] args) throws IOException, InterruptedException, ClassNotFoundException
{
Configuration conf = new Configuration();
Job job = new Job(conf);
job.setJarByClass(SortStudNames.class);
job.setMapperClass(SortMapper.class);
job.setReducerClass(SortReducer.class);
job.setOutputKeyClass(Text.class); job.setOutputValueClass(Text.class);
```

```
FileInputFormat.setInputPaths(job,new Path("input.csv"));

FileOutputFormat.setOutputPath(job,new Path("output"));

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

}

}
```