LABSHEET-3

Mapreduce WordCount Java Program:

```
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;

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

public class WordCount {

  public static class TokenizerMapper extends Mapper<LongWritable, Text, Text,
IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable 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 {
        if (args.length != 2) {
            System.err.println("Usage: WordCount <input path> <output path>");
            System.exit(-1);
        }

        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);
    }
}
```

Save the file as WordCOunt.java

<u>Execution Steps:</u>

```
Step 1:hadoop@nisha-VirtualBox:~$ hdfs dfs -mkdir /count
Step 2:hadoop@nisha-VirtualBox:~$ hdfs dfs -touchz /count/outputcount
Step 3:hadoop@nisha-VirtualBox:~$ hdfs dfs -appendToFile -  /count/outputcount
                one two three
                three two one
                four five six
                ctrl+d Ctrl+d
Step 4:mkdir -p /home/hadoop/outputmap
Step 5:hadoop@nisha-VirtualBox:~$ javac -classpath $(hadoop classpath) -d /home/hadoop/outputmap /home/hadoop/WordCount.java
Step 6:hadoop@nisha-VirtualBox:~$ jar -cvf wordcount.jar -C /home/hadoop/outputmap/ .
```

```
added manifest
adding: WordCount.class(in = 1656) (out= 917)(deflated 44%)
adding: WordCount$TokenizerMapper.class(in = 1846) (out= 768)(deflated 58%)
adding: WordCount$IntSumReducer.class(in = 1739) (out= 742)(deflated 57%)
```

Step 7:hadoop jar wordcount.jar WordCount /count /outputcount

Step 8:hadoop@nisha-VirtualBox:~$ hadoop fs -ls /outputcount

Step 9:hadoop fs -cat /outputcount/part*

```
five    1
four    1
one     2
six     1
three   2
two     2
```