Group B assignment 1

write a code in java for a simple word count application that counts the number of occurrences of each word in a given input set using the hadoop map reduce framework on local standalone setup

```
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);

  }

}
```

To run this code on a local standalone setup, you need to have Hadoop installed and properly configured on your system. Once you have Hadoop set up, you can compile the above code using **`javac`**, and then create a JAR file using **`jar`** command. After that, you can use the following command to execute the code:

```
hadoop jar WordCount.jar WordCount <input_path> <output_path>
```

Replace **`<input_path>`** with the path to your input file or directory, and **`<output_path>`** with the desired location for the output. The results will be stored in the specified output directory.

Please note that setting up Hadoop and running MapReduce jobs can be a complex process, and this example assumes you have a working Hadoop setup on your local machine. If you don't have Hadoop installed, you can try running the code in a cloud-based Hadoop environment like AWS EMR or Cloudera.