Experiment 3: Map reduce program for word count

```
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);
  }
}
```

Experiment 4: Map Reduce Program

```
package my.mapred.pack;

import java.io.IOException;

import java.util.*;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapred.*;

public class TransactionCount {


//MAPPER CODE


public static class Map extends MapReduceBase implements
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, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {

      String myString = value.toString();

      String[] userCount = myString.split(",");

      output.collect(new Text(userCount[3]), one);


      }

}


//REDUCER CODE

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

```
public void reduce(Text key, Iterator<IntWritable> values,
OutputCollector<Text, IntWritable> output, Reporter reporter) throws
IOException { //{little: {1,1}}

      int finaluserCount = 0 ;

      Text mykey = key ;

      while(values.hasNext()) {

            IntWritable value = values.next();

            finaluserCount += value.get();

      }

      output.collect(mykey, new IntWritable(finaluserCount));

      }

}

//DRIVER CODE

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

      JobConf conf = new JobConf(TransactionCount.class);

      conf.setJobName("wordcount");

      conf.setOutputKeyClass(Text.class);

      conf.setOutputValueClass(IntWritable.class);

      conf.setMapperClass(Map.class);

      conf.setCombinerClass(Reduce.class);

      conf.setReducerClass(Reduce.class);

      conf.setInputFormat(TextInputFormat.class);

      conf.setOutputFormat(TextOutputFormat.class); // hadoop jar
jarname classpath inputfolder outputfolder

      FileInputFormat.setInputPaths(conf, new Path(args[0]));

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

      JobClient.runJob(conf);

}

}
```