CS 5433: Bigdata Management
Programming Assignment 1
PART 2 – MapReduce Program for Row Count

CWID: A20343337

RowCount.java

```java
import java.io.IOException; //imported java input-output package
//imported the apache.hadoop package
import org.apache.hadoop.io.*;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.fs.*;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
//Imported the necessary libraries for executing MapReduce row count
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
public class RowCount
{
    public static class Map extends Mapper < LongWritable, Text, Text,
IntWritable > //Extending the mapper class
    {
        private final static IntWritable one = new IntWritable (1); // setting
the static variable one which contains rowcount value “1”
        private Text sentence = new Text ("Total Number of Rows in the downloaded
Flume Data:"); //Intializing the text to be displayed in the output which acts as
key
        public void map (LongWritable key, Text value, Context c)
        {
            try
            {
                c.write(sentence,one); //Mapper output as key-value pairs in the
form of key, 1
            }
            catch(Exception e) //handles the exceptions
            {
                e.printStackTrace(); //prints the exception message
            }
        }


    }
    public static class Reduce extends Reducer < Text, IntWritable, Text,
IntWritable > //Extending the Reducer class
    {
        public void reduce (Text key, Iterable < IntWritable > values, Context c)
        {
            try
            {
                int sum = 0;
                for (IntWritable val : values) //looping through the values
                {
                    sum += val.get(); //Counts the number of rows
                }
                c.write(key, new IntWritable (sum)); //Produces the output as
Key, row count
            }
            catch(Exception e) //handles the exceptions
            {
                e.printStackTrace(); //prints the exception message
            }
        }
    }
    public static void main (String[]args) throws Exception
    {
        Configuration conf1 = new Configuration();//creating configuration object
        Job conf = Job.getInstance(conf1, "Row Count"); //creating job for conf1
instance
        conf.setJarByClass(RowCount.class); //setting main "RowCount" class
        conf.setOutputKeyClass (Text.class); //setting outputkey class
        conf.setOutputValueClass (IntWritable.class); //setting outputvalue class
        conf.setMapperClass (Map.class); //setting mapper class
        conf.setCombinerClass (Reduce.class); //setting combiner class
        conf.setReducerClass (Reduce.class); //setting reducer class
        conf.setOutputKeyClass(Text.class); //setting output key type
        conf.setOutputValueClass(IntWritable.class); //setting output key value
        FileInputFormat.addInputPath(conf, new Path(args[0])); //Passing argument
[0] as input path
        Path outputfolder = new Path(args[1]); //Passing argument [1] as output
folder
        FileOutputFormat.setOutputPath(conf, outputfolder); // output path
        FileSystem fs = FileSystem.get(conf1); //creating file system object for
configuration instance
        fs.delete(outputfolder, true); //deletes the output folder if already
exists
        System.exit(conf.waitForCompletion(true) ? 0 : 1);
    }
}
```