```java
package csv;
import java.io.IOException;
//Hadoop-specific classes for configuration, file system paths, and data types
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;  // Hadoop wrapper for double
import org.apache.hadoop.io.IntWritable;     // Hadoop wrapper for integer
import org.apache.hadoop.io.LongWritable;    // Hadoop wrapper for long
import org.apache.hadoop.io.Text;            // Hadoop wrapper for String
//Classes for creating a MapReduce job
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
//Classes to read input files and write output files
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
//Main class that contains Mapper, Reducer, and Driver
public class csv {
// ================================
//         MAPPER CLASS
// ================================
// Input: Line number (LongWritable), line content (Text)
// Output: Key = "Average", Value = Marks (IntWritable)
public static class AverageMapper extends Mapper<LongWritable, Text, Text,
IntWritable> {

    // The map method processes one line of input at a time
    public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {

        // Skip the header line (first line of CSV that contains column names)
        if (key.get() == 0 && value.toString().contains("Name")) {
            return;
        }
        // Split the CSV line by comma
        String[] fields = value.toString().split(",");
        try {
            // Extract the marks from index 2 (i.e., 3rd column)
            int marks = Integer.parseInt(fields[2]);
            // Write key-value pair: ("Average", marks)
            context.write(new Text("Average"), new IntWritable(marks));
        } catch (Exception e) {
            // If marks cannot be parsed (bad data), skip the line
        }
    }
}
// ================================
//         REDUCER CLASS
// ================================
// Input: Key = "Average", Values = List of marks
// Output: Key = "Average", Value = average marks (DoubleWritable)
public static class AverageReducer extends Reducer<Text, IntWritable, Text,
DoubleWritable> {
    // Reduce method receives all marks and calculates the average
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {
        int sum = 0;      // Sum of all marks
        int count = 0;    // Total number of students/records
        // Loop over all values (marks) and compute sum and count
        for (IntWritable val : values) {
            sum += val.get();     // Add mark to sum
            count++;              // Increment record count
        }
        // Calculate average as double
        double average = (double) sum / count;
        // Write the final output: ("Average", calculated average)
        context.write(key, new DoubleWritable(average));
    }
}
// ================================
//         DRIVER / MAIN CLASS
// ================================
public static void main(String[] args) throws Exception {
    // Set up Hadoop configuration
    Configuration conf = new Configuration();
    // Create a new Hadoop MapReduce job
    Job job = Job.getInstance(conf, "Average Marks Calculation");
    // Set the main class that contains Mapper and Reducer
    job.setJarByClass(csv.class);
    // Register the Mapper and Reducer classes
    job.setMapperClass(AverageMapper.class);
    job.setReducerClass(AverageReducer.class);
    // Set output types from the Mapper/Reducer
    job.setOutputKeyClass(Text.class);          // Output key is "Average"
    job.setOutputValueClass(IntWritable.class); // Intermediate values are
IntWritable
    // Set input path (args[1]) and output path (args[2]) from command line
    FileInputFormat.addInputPath(job, new Path(args[0]));   // CSV input file
in HDFS
    FileOutputFormat.setOutputPath(job, new Path(args[1])); // Output directory
in HDFS
    // Submit job and wait until it completes
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```