```java
package applications;

import java.io.IOException;
import java.util.Iterator;

import mybeans.Reading;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;

public class SensorMapReduce
{

    public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, Text>
    {
        //private final static IntWritable one = new IntWritable(1);
        private Text outKey = new Text();
        private Text outValue = new Text();

        public void map(LongWritable key, Text value, OutputCollector<Text, Text> output, Reporter reporter) throws IOException
        {
            String line = value.toString();
            Reading rec = Reading.parse(line);    // create one object from one csv line. confidential.
            String unitName = rec.getUnitName();
            if (unitName != null)
            {
                unitName = unitName.trim();

                // my way of filtering out the garbage...
                if (true /* removed for confidentiality */ )
                {
                    outKey.set(unitName);
                    outValue.set("1"                        // count
                        + "," + rec.getTankTempCelcius()    // min
                        + "," + rec.getTankTempCelcius()    // max
                        + "," + rec.getTankTempCelcius()    // sum
                        );
                    output.collect(outKey, outValue);
                }
            }
        }
    }
```

```
55
56      public static class Reduce extends MapReduceBase implements Reducer<Text, Text, Text,
    Text>
57      {
58          public void reduce(Text key, Iterator<Text> values, OutputCollector<Text, Text>
    output, Reporter reporter) throws IOException
59          {
60              // reducer accepts a count, min, max, and sum.
61              // when no combiner is used, the count will always be 1.
62              // reducer outputs a count, min, max, sum, and average.
63              // when reducer is used as a combiner, the input average will be ignored.
64
65              Text value = new Text();
66
67              int countOut = 0;
68              double minOut = Double.MAX_VALUE;
69              double maxOut = Double.MIN_VALUE;
70              double sumOut = 0;
71
72              while (values.hasNext())
73              {
74                  String[] vals = values.next().toString().split(",");
75
76                  int countIn = Integer.parseInt(vals[0]);
77                  double minIn = Double.parseDouble(vals[1]);
78                  double maxIn = Double.parseDouble(vals[2]);
79                  double sumIn = Double.parseDouble(vals[3]);
80
81                  countOut += countIn;
82                  minOut = (minIn < minOut? minIn: minOut);
83                  maxOut = (maxIn > maxOut? maxIn: maxOut);
84                  sumOut += sumIn;
85              }
86
87              double meanOut = (countOut == 0? 0: sumOut/countOut);
88
89              value.set("" + countOut + "," + minOut + "," + maxOut + "," + sumOut + "," +
    meanOut);
90              output.collect(key, value);
91          }
92      }
93
94      public static void main(String[] args) throws Exception
95      {
96          // usage: SensorMapReduce inputpath outputpath {Y/N}
97          // where {Y/N} is to use combiner or not.
98
99          JobConf conf = new JobConf(SensorMapReduce.class);
100         conf.setJobName("sensor_map_reduce");
101
102         conf.setOutputKeyClass(Text.class);
103         conf.setOutputValueClass(Text.class);
104
105         conf.setMapperClass(Map.class);
106
107         if ("Y".equals(args[2]))
108         {
```

```
109             conf.setCombinerClass(Reduce.class);
110         }
111 
112         conf.setReducerClass(Reduce.class);
113 
114         conf.setInputFormat(TextInputFormat.class);
115         conf.setOutputFormat(TextOutputFormat.class);
116 
117         FileInputFormat.setInputPaths(conf, new Path(args[0]));
118         FileOutputFormat.setOutputPath(conf, new Path(args[1]));
119 
120         JobClient.runJob(conf);
121     }
122 }
123 
```