```
bigdata@bigdata-VirtualBox:~/A1$ export
HADOOP_CLASSPATH=$($HADOOP_HOME/bin/hadoop classpath)
bigdata@bigdata-VirtualBox:~/A1$ javac -classpath ${HADOOP_CLASSPATH}
solution3.java
bigdata@bigdata-VirtualBox:~/A1$ jar cf solution3.jar solution3*.class
bigdata@bigdata-VirtualBox:~/A1$ $HADOOP_HOME/bin/hadoop jar
solution3.jar solution3 sales.txt /user/bigdata/output
21/10/13 10:18:02 INFO client.RMProxy: Connecting to ResourceManager at
bigdata-VirtualBox/10.0.2.15:8032
21/10/13 10:18:03 WARN mapreduce.JobResourceUploader: Hadoop command-line
option parsing not performed. Implement the Tool interface and execute
your application with ToolRunner to remedy this.
21/10/13 10:18:03 INFO input.FileInputFormat: Total input paths to
process : 1
21/10/13 10:18:03 INFO mapreduce.JobSubmitter: number of splits:1
21/10/13 10:18:03 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1634070937113_0022
21/10/13 10:18:03 INFO impl.YarnClientImpl: Submitted application
application_1634070937113_0022
21/10/13 10:18:03 INFO mapreduce.Job: The url to track the job:
http://bigdata-VirtualBox:8088/proxy/application_1634070937113_0022/
21/10/13 10:18:03 INFO mapreduce.Job: Running job: job_1634070937113_0022
21/10/13 10:18:10 INFO mapreduce.Job: Job job_1634070937113_0022 running
in uber mode : false
21/10/13 10:18:10 INFO mapreduce.Job:  map 0% reduce 0%
21/10/13 10:18:15 INFO mapreduce.Job:  map 100% reduce 0%
21/10/13 10:18:20 INFO mapreduce.Job:  map 100% reduce 100%
21/10/13 10:18:20 INFO mapreduce.Job: Job job_1634070937113_0022
completed successfully
21/10/13 10:18:20 INFO mapreduce.Job: Counters: 49
      File System Counters
            FILE: Number of bytes read=88
            FILE: Number of bytes written=237407
            FILE: Number of read operations=0
            FILE: Number of large read operations=0
            FILE: Number of write operations=0
            HDFS: Number of bytes read=164
            HDFS: Number of bytes written=95
            HDFS: Number of read operations=6
            HDFS: Number of large read operations=0
            HDFS: Number of write operations=2
      Job Counters
            Launched map tasks=1
            Launched reduce tasks=1
            Data-local map tasks=1
            Total time spent by all maps in occupied slots (ms)=2614
            Total time spent by all reduces in occupied slots (ms)=2953
            Total time spent by all map tasks (ms)=2614
            Total time spent by all reduce tasks (ms)=2953
            Total vcore-milliseconds taken by all map tasks=2614
            Total vcore-milliseconds taken by all reduce tasks=2953
            Total megabyte-milliseconds taken by all map tasks=2676736
            Total megabyte-milliseconds taken by all reduce tasks=3023872
      Map-Reduce Framework
            Map input records=7
            Map output records=7
            Map output bytes=68
            Map output materialized bytes=88
```

```
        Input split bytes=109
        Combine input records=0
        Combine output records=0
        Reduce input groups=3
        Reduce shuffle bytes=88
        Reduce input records=7
        Reduce output records=12
        Spilled Records=14
        Shuffled Maps =1
        Failed Shuffles=0
        Merged Map outputs=1
        GC time elapsed (ms)=124
        CPU time spent (ms)=1290
        Physical memory (bytes) snapshot=447455232
        Virtual memory (bytes) snapshot=3828072448
        Total committed heap usage (bytes)=319815680
    Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
    File Input Format Counters
```

```
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.FloatWritable;
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 solution3 {

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "solution3");
    job.setJarByClass(solution3.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setReducerClass(IntMinMaxReducer.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);
  }

 public static class TokenizerMapper
      extends Mapper<Object, Text, Text, IntWritable>{

   private final static IntWritable counter = new IntWritable(0);
   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());
       counter.set( Integer.parseInt(itr.nextToken()) );
       context.write(word, counter);
     }
   }
 }
```

```
 public static class IntMinMaxReducer
      extends Reducer<Text,IntWritable,Text,IntWritable> {
      private IntWritable result_min_max = new IntWritable();
      private IntWritable result_sum = new IntWritable();
      private IntWritable result_avg = new IntWritable();

   public void reduce(Text key, Iterable<IntWritable> values,
                      Context context
                      ) throws IOException, InterruptedException {
     int max = Integer.MIN_VALUE;
       int min = Integer.MAX_VALUE;
       int value;
       int total = 0;
       int counter = 0;

     for (IntWritable val : values) {
         value = val.get();
         total = total + value;

         counter++;

         if ( value > max )
           max = value;
         if ( value < min )
             min = value;
     }
     result_sum.set(total);
     context.write(key, result_sum);

     result_avg.set(total/counter);
     context.write(key, result_avg);

     result_min_max.set(max);
     context.write(key, result_min_max);
     result_min_max.set(min);
     context.write(key, result_min_max);
   }
 }
}
```