```
package it.polito.bigdata.hadoop.exercise1;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

/**
 * Driver class.
 */
public class DriverBigData extends Configured implements Tool {

  @Override
  public int run(String[] args) throws Exception {

    Path inputPath;
    Path outputDir;
      int exitCode;

      // Parse the parameters
    inputPath = new Path(args[0]);
    outputDir = new Path(args[1]);

    Configuration conf = this.getConf();

    // The fist job is a word count job
    // It counts the number of occurrences of each movie in watchedmovies.txt

    // Define a new job
    Job job = Job.getInstance(conf);

    // Assign a name to the job
    job.setJobName("Exam 2 - Exercise 1 - count");

    // Set path of the input file/folder (if it is a folder, the job reads all the
files in the specified folder) for this job
    FileInputFormat.addInputPath(job, inputPath);

    // Set path of the output folder for this job
    // A temporary folder
    FileOutputFormat.setOutputPath(job, outputDir);

    // Specify the class of the Driver for this job
    job.setJarByClass(DriverBigData.class);

    // Set input format
    job.setInputFormatClass(TextInputFormat.class);

    // Set job output format
    job.setOutputFormatClass(TextOutputFormat.class);

    // Set map class
```

```
    job.setMapperClass(MapperBigData.class);

    // Set map output key and value classes
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(NullWritable.class);

    // Map only job. Set number of reducers to 0
    job.setNumReduceTasks(0);

    // Execute the job and wait for completion
    if (job.waitForCompletion(true)==true)
            exitCode=0;
    else
      exitCode=1;

    return exitCode;
  }

  /** Main of the driver
   */

  public static void main(String args[]) throws Exception {
      // Exploit the ToolRunner class to "configure" and run the Hadoop application
    int res = ToolRunner.run(new Configuration(), new DriverBigData(), args);

    System.exit(res);
  }
}
```

```
package it.polito.bigdata.hadoop.exercise1;

import java.io.IOException;

import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Mapper;

/**
 * Mapper
 */

class MapperBigData extends Mapper<
                            LongWritable , // Input key type
                    Text,         // Input value type
                    Text,         // Output key type
                    NullWritable> {// Output value type

    protected void map(
            LongWritable  key,   // Input key type
            Text value,         // Input value type
            Context context) throws IOException, InterruptedException {

            String[] fields=value.toString().split(",");

            // fields[2] = PM10 value
            double PM10value=Double.parseDouble(fields[2]);

            // Select the line if the value is greater than 45 or less than 0.
            if (PM10value>45 || PM10value<0) {
                  // Emit a pair (record,null)
                  context.write(new Text(value), NullWritable.get());
            }
    }

}
```