```java
import java.IoException;
import java.utils.*;
import org.apache.hadoop.fs.*
import org.apache.hadoop mapreduce.*
import org.apache.hadoop.io.*;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.mapreduce.lib.input.
                              FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.
                              FileOutputFormat;


public class minimumno {

public static class MMapper extends Mapper
     <LongWritable, Text, Text, IntWritable> {

    public void map (LongWritable key, Text value,
                 Context context) throws Exception
    {
        String [] numbers = value.toString().split(" ");
        for (string num : numbers)
        {
            Context.Write (new Text ("Min"),
                    new IntWritable (Integer.parseInt(
                                    num));
        }
    }
}

public static class MReducer extends Reducer
    <Text, IntWritable, Text, IntWritable> {

    public void reduce (Text key, Iterable<IntWritable>
                                                value
                    Context context) throws exception
    {
```

```
Intiger minno = Intiger. MAX_VALUE;
for (IntWritable val : values)
{
    if (val.get() ≤ minno)
    {
        minno = val.get()
    }
}
context.write (key, new IntWritable (minno));

}
}

~~public class Main~~
public static void main (string [] args) throws Exception
{
    Configuration conf = new Configuration();
    Job job = Job.getInstance (conf, "Min");
    job.setJarByClass (Minimumno.class);
    job.setMapperClass (MMapper.class);
    job.setReducerClass (MReducer.class);
    job.setOutputKeyClass (Text.class);
    job.setOutputValueClass (IntWritable.class);

    FileInputFormat.addInputPath (job,
                              new Path(args[0]));
    FileOutputFormat.getOutputPath (job,
                              new Path (args[1]));

    System.exit (Job.waitForCompletion(true)?0:1);

}
}
```