# Hadoop MapReduce Log Processor in Java

## LogProcessor.java

```
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.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 LogProcessor {

    public static class LogMapper extends Mapper<Object, Text, Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text logLevel = new Text();

          public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
            String[] parts = value.toString().split(" ");
            if (parts.length >= 3) {
                String level = parts[2].trim();
                logLevel.set(level);
                context.write(logLevel, one);
            }
        }
    }

    public static class LogReducer extends Reducer<Text, IntWritable, Text, IntWritable>
{
        private IntWritable count = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values, Context context)
                throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values)
                sum += val.get();
            count.set(sum);
            context.write(key, count);
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "log level count");
        job.setJarByClass(LogProcessor.class);
        job.setMapperClass(LogMapper.class);
        job.setCombinerClass(LogReducer.class);
        job.setReducerClass(LogReducer.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);
    }
}
```

### Input File (system.log)

```
2025-04-20 12:00:00 INFO Starting system
2025-04-20 12:00:01 WARN Low disk space
2025-04-20 12:00:02 ERROR Disk read failure
2025-04-20 12:00:03 INFO System check complete
2025-04-20 12:00:04 ERROR Disk write failure
2025-04-20 12:00:05 WARN Memory usage high
```

### Expected Output (part-r-00000)

```
ERROR   2
INFO    2
WARN    2
```