LogProcessor.java

```
import java.io.IOException;


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 LogLevelMapper 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 line = value.toString();


            // Split line by whitespace

            String[] parts = line.split("\\s+");

            if (parts.length >= 3) {

                String level = parts[2]; // INFO, WARN, ERROR

                logLevel.set(level);

                context.write(logLevel, one);

            }
        }
    }

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

        private IntWritable result = 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();
            }

            result.set(sum);
            context.write(key, result);
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Log Level Counter");

        job.setJarByClass(LogProcessor.class);
        job.setMapperClass(LogLevelMapper.class);
        job.setCombinerClass(CountReducer.class);
        job.setReducerClass(CountReducer.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);
    }
}
```

LogsFile.txt

```
2024-03-12 10:15:30 INFO  Starting system
2024-03-12 10:15:35 WARN  Disk usage high
2024-03-12 10:16:00 ERROR Failed to load config file
2024-03-12 10:16:05 INFO  Retrying connection
2024-03-12 10:16:10 ERROR Connection timed out
2024-03-12 10:17:00 INFO  System started successfully
```

Output:

```
ERROR 2
INFO  3
WARN  1
```