Link to the log file:
https://drive.google.com/file/d/109WGNaT_j4PHI66F05hiz6rdWPuzkNmt/view?usp=drivesdk

Code:
```
import java.io.IOException;
 import java.text.ParseException;
 import org.apache.hadoop.conf.Configuration;
 import org.apache.hadoop.fs.Path;
 import org.apache.hadoop.io.IntWritable;
 import org.apache.hadoop.io.LongWritable;
 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;
 import org.apache.hadoop.util.GenericOptionsParser;

 public class LogFile {
        public static void main(String [] args) throws Exception{
                Configuration c = new Configuration();
                String[] files = new GenericOptionsParser(c,args).getRemainingArgs();
                Path input=new Path(files[0]);
                Path output=new Path(files[1]);
                Job j=new Job(c,"Logfile");
                j.setJarByClass(LogFile.class);
                j.setMapperClass(MapForWordCount.class);
                j.setReducerClass(ReduceForWordCount.class);
                j.setOutputKeyClass(Text.class);
                j.setOutputValueClass(IntWritable.class);
                FileInputFormat.addInputPath(j, input);
                FileOutputFormat.setOutputPath(j, output);
                System.exit(j.waitForCompletion(true)?0:1);
        }

        public static class MapForWordCount extends Mapper<LongWritable, Text, Text,
IntWritable>{
                public void map(LongWritable key, Text value, Context con) throws IOException,
InterruptedException{
                        String text = value.toString();
                        String[] words=text.split(" ");
                        Text outputKey = new Text(words[0].toUpperCase().trim());
                        int datein = Integer.parseInt(words[1]);
                        int dateout = Integer.parseInt(words[2]);
                        long total = dateout - datein;
                        int total_in = (int) total;
                        IntWritable outputValue = new IntWritable(total_in);
                    con.write(outputKey, outputValue);
                }
        }

        public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text,
IntWritable>{
                int max=0;
                Text maxWord = new Text();
                int min=999;
                Text minWord = new Text();

                public void reduce(Text key, Iterable<IntWritable> values, Context con) throws
IOException, InterruptedException{
                        int sum = 0;

                        for(IntWritable value : values)
                                sum += value.get();

                        if(max<sum){
                                max = sum;
                                maxWord.set(key);
                        }
                        if(min>sum){
                                min = sum;
                                minWord.set(key);
                        }

                }

                @Override
                protected void cleanup(Context context) throws IOException, InterruptedException {
                      context.write(maxWord, new IntWritable(max));
                      context.write(minWord, new IntWritable(min));
                }
        }
}
```