//Example Driver class for Word count program

import java.io.\*;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

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 WebLogReader

{

  public static class WebLogWritable implements WritableComparable<WebLogWritable>

  {

   private Text siteURL, reqDate, timestamp, ipaddress;

   private IntWritable reqNo;

   //Default Constructor

   public WebLogWritable()

   {

    this.siteURL = new Text();

    this.reqDate = new Text();

    this.timestamp = new Text();

    this.ipaddress = new Text();

    this.reqNo = new IntWritable();

   }

   //Custom Constructor

   public WebLogWritable(IntWritable reqno, Text url, Text rdate, Text rtime, Text rip)

   {

    this.siteURL = url;

    this.reqDate = rdate;

    this.timestamp = rtime;

    this.ipaddress = rip;

    this.reqNo = reqno;

   }

   //Setter method to set the values of WebLogWritable object

   public void set(IntWritable reqno, Text url, Text rdate, Text rtime, Text rip)

   {

    this.siteURL = url;

    this.reqDate = rdate;

    this.timestamp = rtime;

    this.ipaddress = rip;

    this.reqNo = reqno;

   }

   //to get IP address from WebLog Record

   public Text getIp()

   {

    return ipaddress;

   }

   @Override

   //overriding default readFields method.

   //It de-serializes the byte stream data

   public void readFields(DataInput in) throws IOException

   {

    ipaddress.readFields(in);

    timestamp.readFields(in);

    reqDate.readFields(in);

    reqNo.readFields(in);

    siteURL.readFields(in);

   }

   @Override

   //It serializes object data into byte stream data

   public void write(DataOutput out) throws IOException

   {

    ipaddress.write(out);

    timestamp.write(out);

    reqDate.write(out);

    reqNo.write(out);

    siteURL.write(out);

   }

   @Override

   public int compareTo(WebLogWritable o)

   {

     if (ipaddress.compareTo(o.ipaddress)==0)

     {

       return (timestamp.compareTo(o.timestamp));

     }

     else return (ipaddress.compareTo(o.ipaddress));

   }

   @Override

   public boolean equals(Object o)

   {Rules for creating custom Hadoop Writable

     if (o instanceof WebLogWritable)

     {

       WebLogWritable other = (WebLogWritable) o;

       return ipaddress.equals(other.ipaddress) && timestamp.equals(other.timestamp);

     }

     return false;

   }

   @Override

   public int hashCode()

   {

     return ipaddress.hashCode();

   }

}

public static class WebLogMapper extends Mapper <LongWritable, Text, WebLogWritable, IntWritable>

{

  private static final IntWritable one = new IntWritable(1);

  private WebLogWritable wLog = new WebLogWritable();

  private IntWritable reqno = new IntWritable();

  private Text url = new Text();

  private Text rdate = new Text();

  private Text rtime = new Text();

  private Text rip = new Text();

  public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException

  {

    String[] words = value.toString().split("\t") ;

    System.out.printf("Words[0] - %s, Words[1] - %s, Words[2] - %s, length - %d", words[0], words[1], words[2], words.length);

    reqno.set(Integer.parseInt(words[0]));

    url.set(words[1]);

    rdate.set(words[2]);

    rtime.set(words[3]);

    rip.set(words[4]);

    wLog.set(reqno, url, rdate, rtime, rip);

    context.write(wLog, one);

  }

}

public static class WebLogReducer extends Reducer <WebLogWritable, IntWritable, Text, IntWritable>

{

  private IntWritable result = new IntWritable();

  private Text ip = new Text();

  public void reduce(WebLogWritable key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException

  {

    int sum = 0;

    ip = key.getIp();

    for (IntWritable val : values)

    {

      sum++ ;

    }

    result.set(sum);

    context.write(ip, result);

  }

}

public static void main(String[] args) throws Exception

{

      Configuration conf = new Configuration();

      Job job = new Job();

      job.setJobName("WebLog Reader");

      job.setJarByClass(WebLogReader.class);

      job.setMapperClass(WebLogMapper.class);

      job.setReducerClass(WebLogReducer.class);

      job.setOutputKeyClass(Text.class);

      job.setOutputValueClass(IntWritable.class);

      job.setMapOutputKeyClass(WebLogWritable.class);

      job.setMapOutputValueClass(IntWritable.class);

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

      FileOutputFormat.setOutputPath(job, new Path(args[1]));

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

}

}