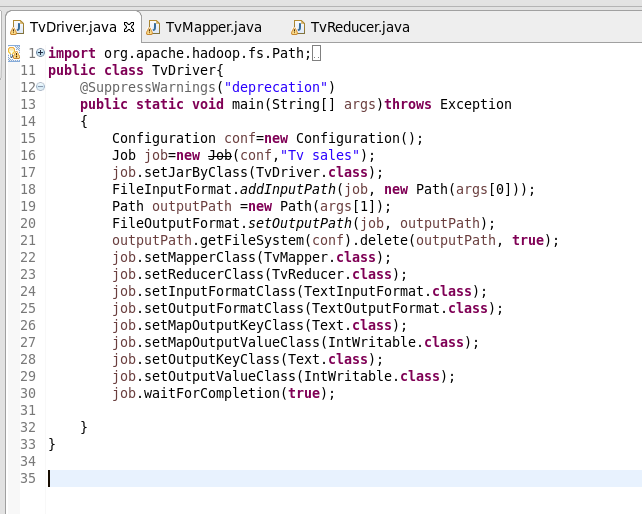
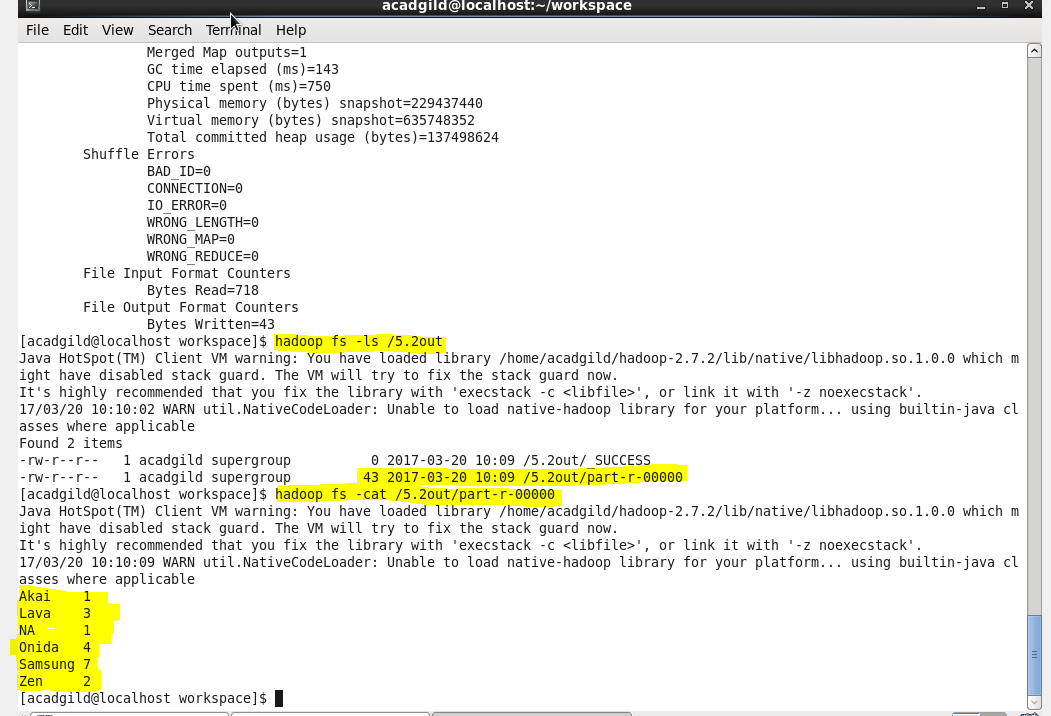
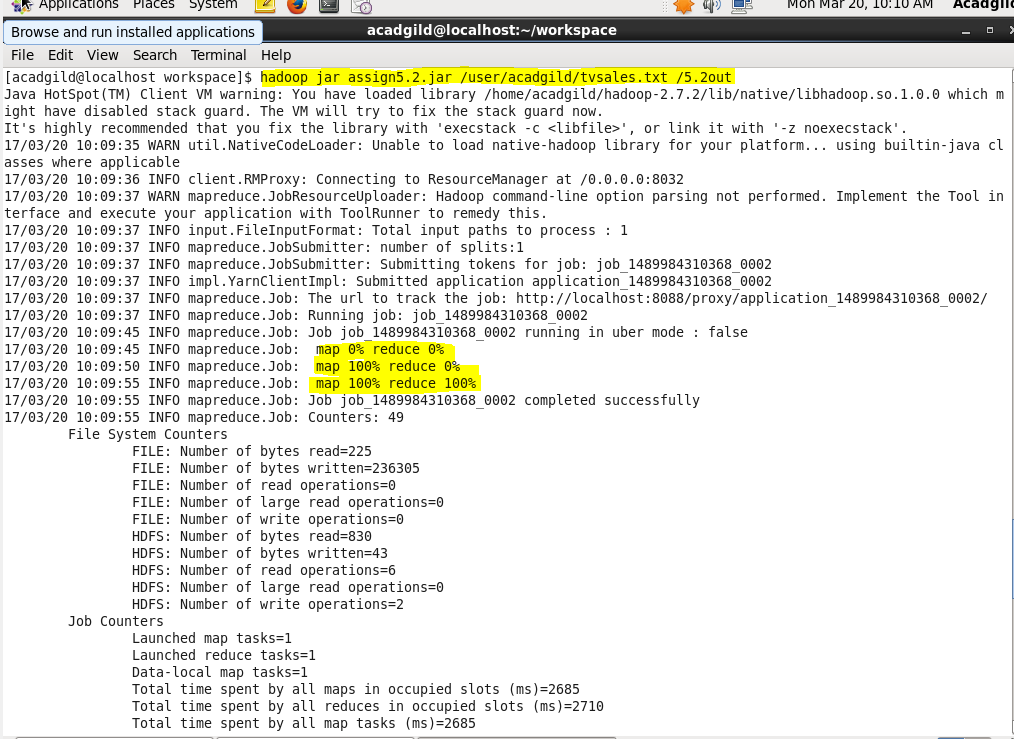
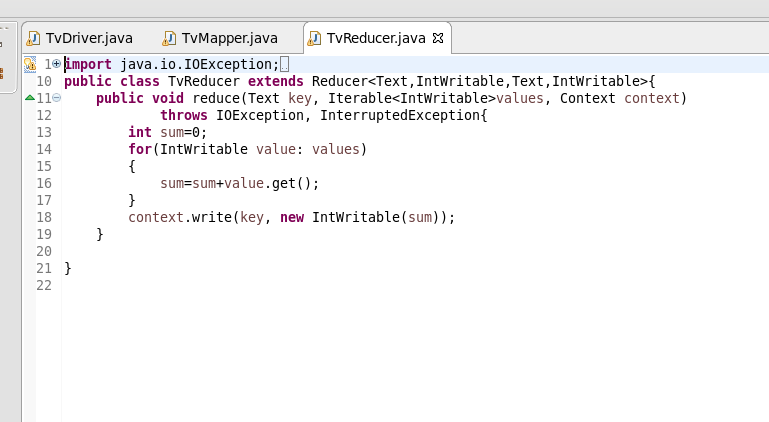
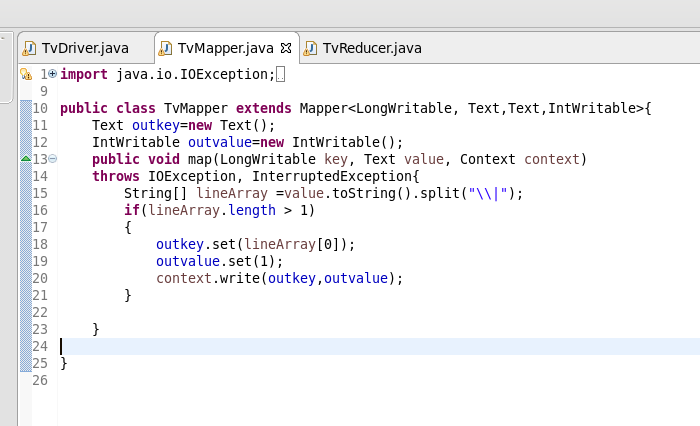
Write a MapReduce program to calculate the total units sold for each Company.

import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.conf.\*;  
import org.apache.hadoop.mapreduce.Job;  
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;  
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;  
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.io.Text;  
public class TvDriver{  
    @SuppressWarnings("deprecation")  
    public static void main(String[] args)throws Exception  
    {  
        Configuration conf=new Configuration();  
        Job job=new Job(conf,"Tv sales");  
        job.setJarByClass(TvDriver.class);  
        FileInputFormat.addInputPath(job, new Path(args[0]));  
        Path outputPath =new Path(args[1]);  
        FileOutputFormat.setOutputPath(job, outputPath);  
        outputPath.getFileSystem(conf).delete(outputPath, true);  
        job.setMapperClass(TvMapper.class);  
        job.setReducerClass(TvReducer.class);  
        job.setInputFormatClass(TextInputFormat.class);  
        job.setOutputFormatClass(TextOutputFormat.class);  
        job.setMapOutputKeyClass(Text.class);  
        job.setMapOutputValueClass(IntWritable.class);  
        job.setOutputKeyClass(Text.class);  
        job.setOutputValueClass(IntWritable.class);  
        job.waitForCompletion(true);  
  
    }  
}  
  
import java.io.IOException;  
import java.util.StringTokenizer;  
  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.\*;  
  
public class TvMapper extends Mapper<LongWritable, Text,Text,IntWritable>{  
    Text outkey=new Text();  
    IntWritable outvalue=new IntWritable();  
    public void map(LongWritable key, Text value, Context context)  
    throws IOException, InterruptedException{  
        String[] lineArray =value.toString().split("\\|");  
        if(lineArray.length > 1)  
        {  
            outkey.set(lineArray[0]);  
            outvalue.set(1);  
            context.write(outkey,outvalue);  
        }  
  
    }  
  
}  
import java.io.IOException;  
import java.util.StringTokenizer;  
  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.\*;  
import org.apache.hadoop.mapreduce.Mapper.Context;  
public class TvReducer extends Reducer<Text,IntWritable,Text,IntWritable>{  
    public void reduce(Text key, Iterable<IntWritable>values, Context context)  
            throws IOException, InterruptedException{  
        int sum=0;  
        for(IntWritable value: values)  
        {  
            sum=sum+value.get();  
        }  
        context.write(key, new IntWritable(sum));  
    }  
  
}



Write a MapReduce program to calculate the total units sold in each state for Onida company.

import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.conf.\*;  
import org.apache.hadoop.mapreduce.Job;  
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;  
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;  
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.io.Text;  
public class TvDriver{  
    @SuppressWarnings("deprecation")  
    public static void main(String[] args)throws Exception  
    {  
        Configuration conf=new Configuration();  
        Job job=new Job(conf,"Tv sales");  
        job.setJarByClass(TvDriver.class);  
        FileInputFormat.addInputPath(job, new Path(args[0]));  
        Path outputPath =new Path(args[1]);  
        FileOutputFormat.setOutputPath(job, outputPath);  
        outputPath.getFileSystem(conf).delete(outputPath, true);  
        job.setMapperClass(TvMapper.class);  
        job.setReducerClass(TvReducer.class);  
        job.setInputFormatClass(TextInputFormat.class);  
        job.setOutputFormatClass(TextOutputFormat.class);  
        job.setMapOutputKeyClass(Text.class);  
        job.setMapOutputValueClass(IntWritable.class);  
        job.setOutputKeyClass(Text.class);  
        job.setOutputValueClass(IntWritable.class);  
        job.waitForCompletion(true);  
  
    }  
}  
    import java.io.IOException;  
  
  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.\*;  
  
public class TvMapper extends Mapper<LongWritable, Text,Text,IntWritable>{  
    Text outkey=new Text();  
    IntWritable outvalue=new IntWritable();  
    public void map(LongWritable key, Text value, Context context)  
    throws IOException, InterruptedException{  
        String[] lineArray = value.toString().split("\\|");  
        if((lineArray.length > 1)&&(lineArray[0].equalsIgnoreCase("Onida")))  
        {  
            outkey.set(lineArray[3]);  
            outvalue.set(1);  
            context.write(outkey,outvalue);  
        }  
  
    }  
  
}  
import java.io.IOException;  
import java.util.StringTokenizer;  
  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.\*;  
import org.apache.hadoop.mapreduce.Mapper.Context;  
public class TvReducer extends Reducer<Text,IntWritable,Text,IntWritable>{  
    public void reduce(Text key, Iterable<IntWritable>values, Context context)  
            throws IOException, InterruptedException{  
        int sum=0;  
        for(IntWritable value: values)  
        {  
            sum=sum+value.get();  
        }  
        context.write(key, new IntWritable(sum));  
    }  
  
}

