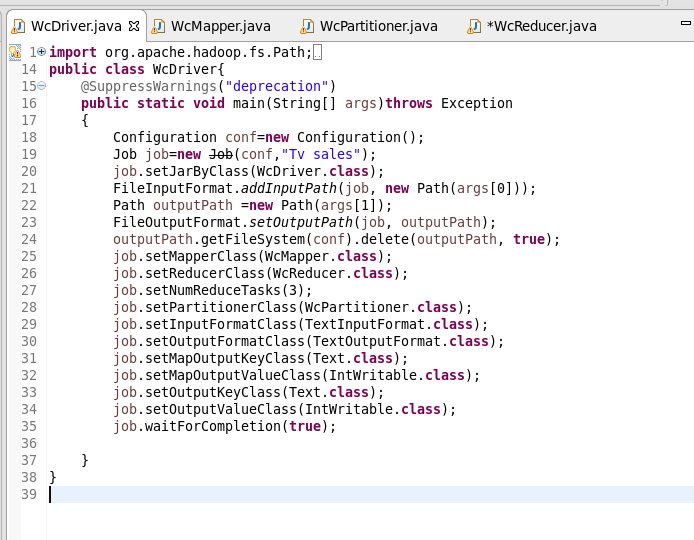
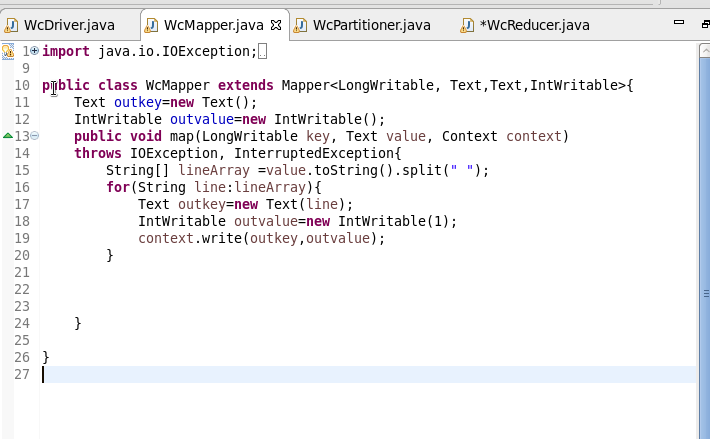
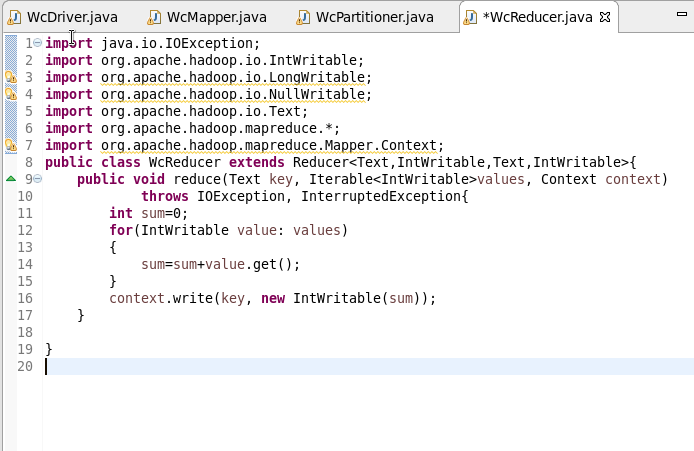
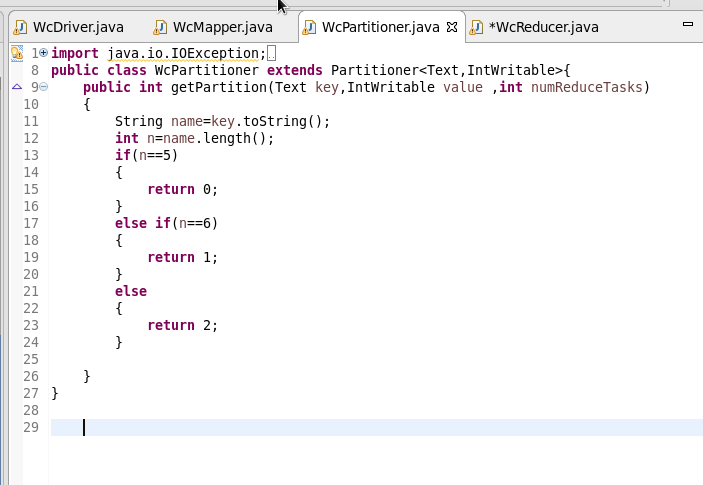
**Assignment 7.4**

**Question:** Write a word count program using partitioner and implement the following logic in the partitioner words with length 5 should go into reducer 1 and words with length 6 should go into reduer 2 and the rest of the words should go into reducer 3.

***Codes***import org.apache.hadoop.fs.Path;  
import java.io.IOException;  
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 WcDriver{  
    @SuppressWarnings("deprecation")  
    public static void main(String[] args)throws Exception  
    {  
        Configuration conf=new Configuration();  
        Job job=new Job(conf,"Tv sales");  
        job.setJarByClass(WcDriver.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(WcMapper.class);  
        job.setReducerClass(WcReducer.class);  
        job.setNumReduceTasks(3);  
        job.setPartitionerClass(WcPartitioner.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 WcMapper 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(" ");  
        for(String line:lineArray){  
            Text outkey=new Text(line);  
            IntWritable outvalue=new IntWritable(1);  
            context.write(outkey,outvalue);  
        }  
    }  
}

import java.io.IOException;  
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 WcPartitioner extends Partitioner<Text,IntWritable>{  
    public int getPartition(Text key,IntWritable value ,int numReduceTasks)  
    {  
        String name=key.toString();  
        int n=name.length();  
        if(n==5)  
        {  
            return 0;  
        }  
        else if(n==6)  
        {  
            return 1;   
        }  
        else  
        {  
            return 2;  
        }  
    }  
}

import java.io.IOException;  
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 WcReducer 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));  
    }}  
  


**INPUT:**

