**BIGDATA 3rd Experiment:**

**Priyanka Dattatreya Hegde**

**1NT20Is407**

**C3 batch**

**Code:**

**package priyanka407;**

**import java.io.IOException;**

**import java.util.\*;**

**import java.util.StringTokenizer;**

**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.mapred.JobClient;**

**import org.apache.hadoop.mapred.JobConf;**

**import org.apache.hadoop.mapred.MapReduceBase;**

**import org.apache.hadoop.mapred.Mapper;**

**import org.apache.hadoop.mapred.OutputCollector;**

**import org.apache.hadoop.mapred.Reporter;**

**import org.apache.hadoop.mapred.Reducer;**

**import org.apache.hadoop.mapred.TextOutputFormat;**

**import org.apache.hadoop.mapred.TextInputFormat;**

**import org.apache.hadoop.mapred.FileInputFormat;**

**import org.apache.hadoop.mapred.FileOutputFormat;**

**public class WordCount {**

**public static class Map extends MapReduceBase implements**

**Mapper<LongWritable, Text, Text, IntWritable> {**

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

**private Text word = new Text();**

**public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable>**

**output, Reporter reporter) throws IOException {**

**String line = value.toString();**

**StringTokenizer tokenizer = new StringTokenizer(line);**

**while (tokenizer.hasMoreTokens()) {**

**word.set(tokenizer.nextToken());**

**output.collect(word, one);**

**}**

**}**

**}**

**public static class Reduce extends MapReduceBase implements**

**Reducer<Text, IntWritable, Text, IntWritable> {**

**public void reduce(Text key, Iterator<IntWritable> values,**

**OutputCollector<Text, IntWritable> output, Reporter reporter)**

**throws IOException {**

**int sum = 0;**

**while (values.hasNext()) {**

**sum += values.next().get();**

**}**

**output.collect(key, new IntWritable(sum));**

**}**

**}**

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

**JobConf conf = new JobConf(WordCount.class);**

**conf.setJobName("Word Count");**

**conf.setOutputKeyClass(Text.class);**

**conf.setOutputValueClass(IntWritable.class);**

**conf.setMapperClass(Map.class);**

**conf.setCombinerClass(Reduce.class);**

**conf.setReducerClass(Reduce.class);**

**conf.setInputFormat(TextInputFormat.class);**

**conf.setOutputFormat(TextOutputFormat.class);**

**FileInputFormat.setInputPaths(conf, new Path(args[0]));**

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

**JobClient.runJob(conf);**

**}**

**}**

**Output: Jps and starting the hadoopText

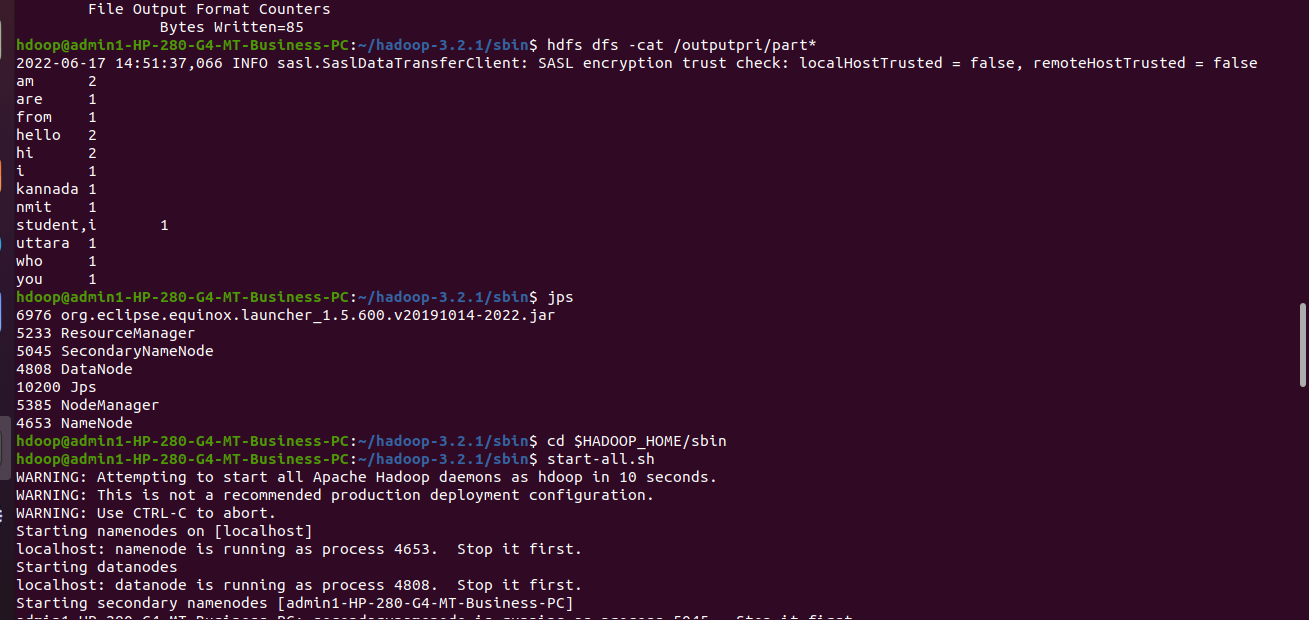
Description automatically generated**

**Creating input file inputpri and input text file pd.txt**

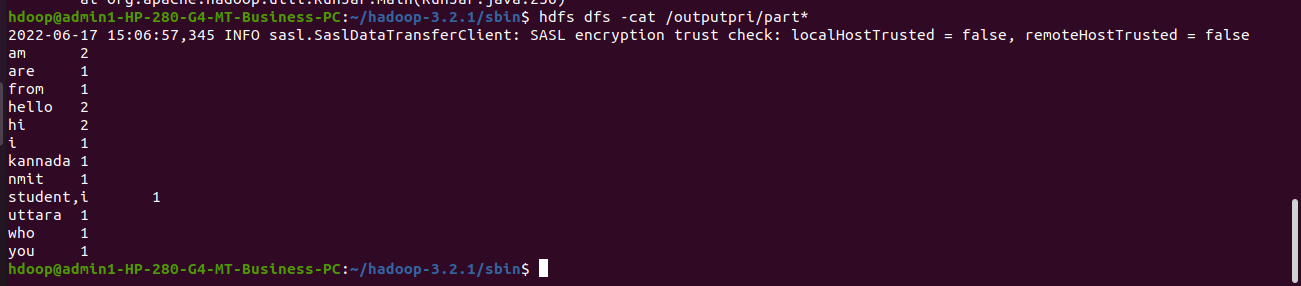
**Text

Description automatically generated**

**Creating output txt and inputing jar file:**

****

**Output:**

****