MapReduce WordCount Program using Hadoop in Java

Step 1:

Open Eclipse > File > New > Java Project (Name it MRProgramsDemo) > Finish.

Step 2:

Right Click > New > Package (Name it - PackageDemo) > Finish.

Step 3:

Right Click on Package > New > Class (Name it - WordCount).

Step 4:

Add Reference Libraries:
- /usr/lib/hadoop-0.20/hadoop-core.jar
- /usr/lib/hadoop-0.20/lib/Commons-cli-1.2.jar

Step 5:

Type the following code:

Explanation:

The above program consists of:
Driver class (main method)
Mapper class (MapForWordCount)
Reducer class (ReduceForWordCount)

Step 6:

Make a JAR file: Right Click on Project > Export > Java > JAR File > Finish

Step 7:

Move input text file to HDFS:
```
$ hadoop fs -put wordcountFile wordCountFile
```

Step 8:

Run the JAR file:
```
$ hadoop jar MRProgramsDemo.jar PackageDemo.WordCount wordCountFile MRDir1
```

Step 9:

View output:

MapReduce WordCount Program using Hadoop in Java

```
$ hadoop fs -ls MRDir1
$ hadoop fs -cat MRDir1/part-r-00000
```

Sample Output:

```
BUS     7
CAR     4
TRAIN   6
```

WordCount.java Source Code

```
package PackageDemo;

import java.io.IOException;
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 WordCount {
    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, "wordcount");
        j.setJarByClass(WordCount.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 line = value.toString();
            String[] words = line.split(",");
```

```
            for (String word : words) {
                Text outputKey = new Text(word.toUpperCase().trim());
                IntWritable outputValue = new IntWritable(1);
                con.write(outputKey, outputValue);
            }
        }
    }

      public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text,
IntWritable> {
         public void reduce(Text word, Iterable<IntWritable> values, Context con) throws
IOException, InterruptedException {
            int sum = 0;
            for (IntWritable value : values) {
                sum += value.get();
            }
            con.write(word, new IntWritable(sum));
        }
    }
}
```