WC_Mapper.java

```
package com.wc;

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.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reporter;


public class WC_Mapper extends MapReduceBase implements
Mapper<LongWritable,Text,Text,IntWritable>

{

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

        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);

                }

        }

}
```

WC_Reducer.java

```
package com.wc;


import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;


public class WC_Reducer  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));

	}

}
```

WC_Runner.java

```
package com.wc;


import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;

public class WC_Runner
{
public static void main(String[] args) throws IOException
	{
		JobConf conf = new JobConf(WC_Runner.class);
		conf.setJobName("WordCount");
		conf.setOutputKeyClass(Text.class);
			conf.setOutputValueClass(IntWritable.class);
		conf.setMapperClass(WC_Mapper.class);
			conf.setCombinerClass(WC_Reducer.class);
		conf.setReducerClass(WC_Reducer.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);
        }
}
```

javac -classpath "$(hadoop classpath)" -d . WC_Mapper.java WC_Reducer.java WC_Runner.java

/* This command compiles the three Java files (`WC_Mapper.java`, `WC_Reducer.java`, and `WC_Runner.java`) using the Hadoop classpath for resolving dependencies and places the compiled `.class` files in the current directory. */

jar -cvf wordcount.jar com

/* By running this command, a JAR file named `wordcount.jar` will be created, containing all the contents of the `com` package directory and its subdirectories. This JAR file can then be used to distribute and execute the Hadoop MapReduce application. */

hadoop jar /home/hadoop/hadoop/wordcount.jar com.javatpoint.WC_Runner /test_wc/data1.txt /r_output

/* This command executes the Hadoop MapReduce job defined in the JAR file `/home/hadoop/hadoop/wordcount.jar`, using the `com.javatpoint.WC_Runner` class as the main entry point. It takes `/tes_wc/data1.txt` as input and writes the output to `/r_output`. */

hdfs dfs -cat /r_output/part-00000

/* when you run `hdfs dfs -cat /r_output/part-00000`, you'll see the contents of the `/r_output/part-00000` file displayed in the terminal. This file should contain the output of your MapReduce job, which, in the case of a word count program, would likely consist of word-count pairs. */