```
hadoop@401-9:~/hadoop$ export HADOOP_CLASSPATH=$(hadoop classpath)
hadoop@401-9:~/hadoop$ echo $HADOOP_CLASSPATH
hadoop@401-9:~$ javac -classpath $HADOOP_CLASSPATH -d 'hadoop'
'hadoop/WordCount.java' 'hadoop/WordCountMapper.java' 'hadoop/WordCountReducer.java'
hadoop@401-9:~/hadoop$ jar cvf WordCount.jar WordCount.class WordCountMapper.class
WordCountReducer.class
```

// create

- WordCountMapper.java
- WordCountReducer.java
- WordCount.java

// create input file

Step 1: Compiling the Java Code: The first step involves compiling the Java code using the javac command. This command compiles the Java source code files (WordCountMapper.java, WordCountReducer.java, and WordCount.java) into bytecode, producing .class files. The -cp option specifies the classpath, which includes the necessary Hadoop libraries required for compilation.

```
javac -cp "/home/sunil/hadoop-3.4.0/share/hadoop/common/*:/home/sunil/hadoop-
3.4.0/share/hadoop/hdfs/*:/home/sunil/hadoop-
3.4.0/share/hadoop/mapreduce/*:/home/sunil/hadoop-3.4.0/share/hadoop/yarn/*"
WordCountMapper.java WordCountReducer.java WordCount.java
```

// create input file

```
nano input.txt

hadoop fs -mkdir /input

hadoop fs -put input.txt /input

hadoop fs -cat /input/input.txt
```

Step 2:

```
jar cvf WordCount.jar WordCount.class WordCountMapper.class WordCountReducer.class
```

Step 3: Running the Hadoop Job: The second step involves executing the Hadoop job using the hadoop command. You create a JAR file (WordCount.jar) containing the compiled classes, and then you submit this JAR file to the Hadoop framework along with input and output paths. The Hadoop framework takes care of distributing the job across the cluster, executing the mapper and reducer tasks, and producing the final output in the specified output directory.

hadoop jar WordCount.jar WordCount input_path output_path

// hadoop jar WordCount.jar WordCount /input/input.txt /output

hadoop fs -cat /output/part-r-00000

```
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.mapreduce.Mapper;

public class WordCountMapper extends 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, Context context) throws IOException,
InterruptedException {
      String line = value.toString();
      StringTokenizer tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()) {
         word.set(tokenizer.nextToken());
         context.write(word, one);
      }
   }
}
```

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class WordCountReducer 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 += value.get();
      }
      context.write(key, new IntWritable(sum));
   }
}



import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

   public static void main(String[] args) throws Exception {
      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf, "word count");
      job.setJarByClass(WordCount.class);
      job.setMapperClass(WordCountMapper.class);
      job.setCombinerClass(WordCountReducer.class);
      job.setReducerClass(WordCountReducer.class);
      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(IntWritable.class);
      FileInputFormat.addInputPath(job, new Path(args[0]));
```

```
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```