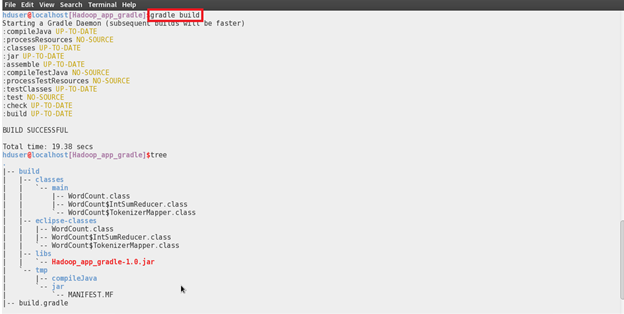
**ASSIGNMENT – 15.6**

**PROBLEM STATEMENT :**

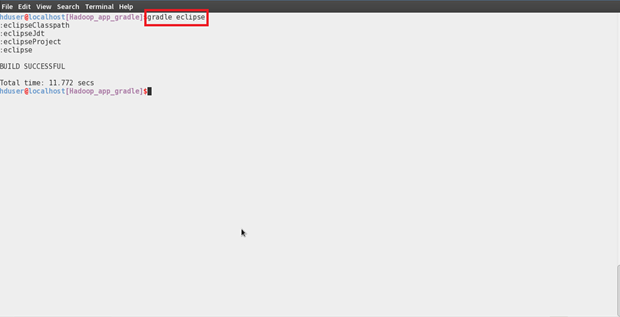
**Build and Deploy Hadoop word count program using Gradle**

creating a Folder named **Hadoop\_app\_gradle** and inside the **src/main/java** directory,writing the Wordcount program

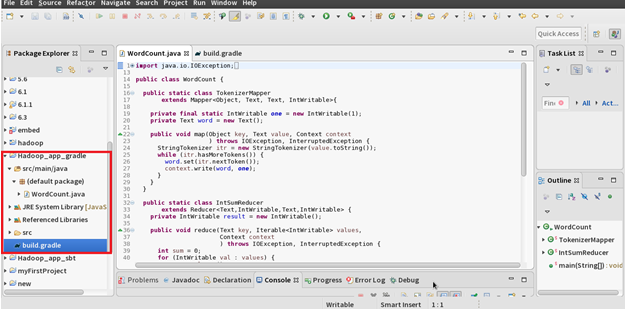
**Gradle Build Command:**



**Building the Project using the Gradle eclipse Command:**



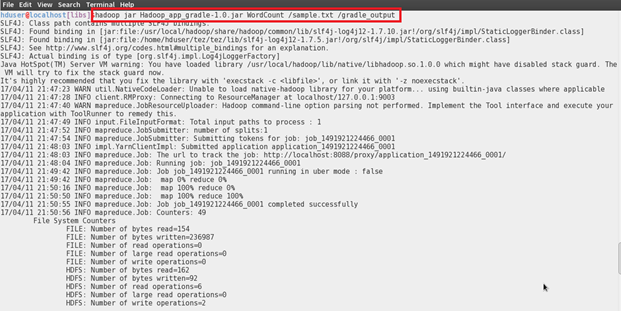
**Importing the project into the eclipse:**



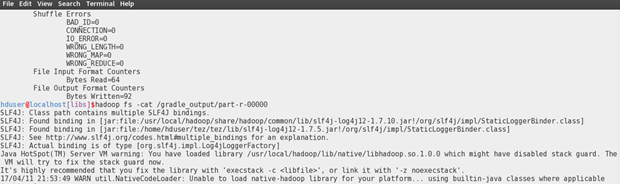
**Jar Created in the target folder:**

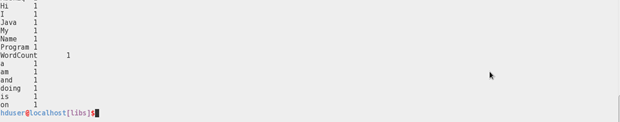


**Running the Jar File:**



**Output:**





**BUILD GRADE.TXT FILE :**

|  |
| --- |
| apply plugin: 'java' //necessary for any java project |
|  |  | |
|  | // Set up group and version info for the artifact | |
|  |  | |
|  | group = "com.acadgild.hadoop\_wc\_gradle" | |
|  |  | |
|  | version = "1.0" | |
|  |  | |
|  | repositories { | |
|  |  | |
|  | mavenCentral() //repositories for downloading your dependencies | |
|  |  | |
|  | } | |
|  |  | |
|  | //dependencies for hadoop application | |
|  |  | |
|  | dependencies { | |
|  |  | |
|  | compile group: 'org.apache.hadoop', name: 'hadoop-mapreduce-client-core', version: '2.7.1' | |
|  |  | |
|  | } | |
|  |  | |
|  | dependencies { | |
|  |  | |
|  | compile group: 'org.apache.hadoop', name: 'hadoop-common', version: '2.7.1' | |
|  |  | |
|  | } | |
|  |  | |
|  | jar { | |
|  |  | |
|  | from configurations.compile.collect { it.isDirectory() ? it : zipTree(it) } | |
|  |  | |
|  | } | |
|  |  | |
|  | ext.hadoopVersion = "2.7.1" | |
|  |  | |
|  | //eclipse plugin | |
|  |  | |
|  | apply plugin: "eclipse" | |
|  |  | |
|  | eclipse { | |
|  |  | |
|  | // Ensure Eclipse build output appears in build directory | |
|  |  | |
|  | classpath { | |
|  |  | |
|  | defaultOutputDir = file("${buildDir}/eclipse-classes") | |
|  |  | |
|  | } | |
|  |  | |
|  | }  **JAVA SCRIPT PROGRAM FOR WORDCOUNT:** | |
| importjava.io.IOException; | |
|  | |
|  | |
|  | |
|  | |
|  | | import java.util.StringTokenizer; |
|  | |  |
|  | | 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.Mapper; |
|  | | import org.apache.hadoop.mapreduce.Reducer; |
|  | | import org.apache.hadoop.mapreduce.lib.input.FileInputFormat; |
|  | | import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; |
|  | |  |
|  | | public class WordCount { |
|  | |  |
|  | | public static class TokenizerMapper |
|  | | extends Mapper<Object, Text, Text, IntWritable>{ |
|  | |  |
|  | | private final static IntWritable one = new IntWritable(1); |
|  | | private Text word = new Text(); |
|  | |  |
|  | | public void map(Object key, Text value, Context context |
|  | | ) throws IOException, InterruptedException { |
|  | | StringTokenizer itr = new StringTokenizer(value.toString()); |
|  | | while (itr.hasMoreTokens()) { |
|  | | word.set(itr.nextToken()); |
|  | | context.write(word, one); |
|  | | } |
|  | | } |
|  | | } |
|  | |  |
|  | | public static class IntSumReducer |
|  | | extends Reducer<Text,IntWritable,Text,IntWritable> { |
|  | | private IntWritable result = new IntWritable(); |
|  | |  |
|  | | public void reduce(Text key, Iterable<IntWritable> values, |
|  | | Context context |
|  | | ) throws IOException, InterruptedException { |
|  | | int sum = 0; |
|  | | for (IntWritable val : values) { |
|  | | sum += val.get(); |
|  | | } |
|  | | result.set(sum); |
|  | | context.write(key, result); |
|  | | } |
|  | | } |
|  | |  |
|  | | 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(TokenizerMapper.class); |
|  | | job.setCombinerClass(IntSumReducer.class); |
|  | | job.setReducerClass(IntSumReducer.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); |
|  | | } |
|  | | } |