***Instructor:*** *William Phillips*

***Due Date:***

***Group Member:-****Dhara Patel 1820905*

*Anmol Pandya 1869107*

*Yi Xu 1589452*

|  |
| --- |
| **Final\_Exam\_Hadoop\_Project**  **World Count Hadoop Map Reduce Problem** |

**Aim:-**For the final project we would like to use an example from Chapter 2 from the textbook Big Data Analytics with R and Hadoop where the concept and job of MapReduce with Hadoop is explained using an example of word count. Hadoop MapReduce has mainly three objects: Mapper, Reducer and Driver and they can be developed with three Java classes where Map class denotes the Map phase, Reduce class denotes the Reduce phase, and Driver class denotes the class with the main() method to initialize the Hadoop MapReduce program. Since, we have considered a word count problem we will three Java classes to compile for the MapReduce Program; they are Map.java, Reduce.java, and WordCount.java which can be used for calculating the frequency of the word.

The goal of the example would be to determine how many times each word occurs in the code. The code would help the user identify the frequency of the unique words existing in the file.

|  |
| --- |
| First we set up all installation for this assignment and after that we implemented world count Map reduce program:  **Installing Hadoop on Linux, Ubuntu flavor (single node cluster)** |

1. install Ubuntu on your PC/Desktop.

Prerequisites: step 1 to 4

2.install latest oracle java 8 in Ubuntu

3. adding a dedicated Hadoop system user

[**dhara@dhara-VirtualBox**](mailto:dhara@dhara-VirtualBox)**~$ sudo adduser hadoop**

**Password:**

4.Configuring SSH

5. Disabling IPV6

**Install Hadoop:**

1. Download latest Hadoop sources from the Apache software foundation.

// Locate to Hadoop installation directory

$ cd /usr/local

// Extract the tar file of Hadoop distribution

$ sudo tar xzf hadoop-3.0.1.tar.gz

// To move Hadoop resources to hadoop folder

$ sudo mv hadoop-3.0.1 hadoop

// Make user-hduser from group-hadoop as owner of hadoop directory

$ sudo chown -R hadoop:hadoop hadoop

2. Add the $JAVA\_HOME and $HADOOP\_HOME variables to the.bashrc file of Hadoop system user and the updated .bashrc file looks as follows:

// Setting the environment variables for running Java and Hadoop commands

export HADOOP\_HOME=/usr/local/hadoop

export JAVA\_HOME=/usr/lib/jvm/java-8-oracle

// alias for Hadoop commands

unalias fs &> /dev/null

alias fs="hadoop fs"

unalias hls &> /dev/null

aliashls="fs -ls"

// Defining the function for compressing the MapReduce job output by lzop command

lzohead ()

{

hadoopfs -cat $1 | lzop -dc | head -1000 | less

}

// Adding Hadoop\_HoME variable to PATH

export PATH=$PATH:$HADOOP\_HOME/bin

3. Update the Hadoop configuration files with the conf/\*-site.xml format.

**• conf/core-site.xml:**

<property>

<name>hadoop.tmp.dir</name>

<value>/app/hadoop/tmp</value>

<description>A base for other temporary directories.</description>

</property> <property>

<name>fs.default.name</name>

<value>hdfs://localhost:54310</value>

<description>The name of the default filesystem. A URI whose scheme and authority determine the FileSystem implementation. The uri's scheme determines the config property (fs.SCHEME.impl) naming theFileSystem implementation class. The uri's authority is used to determine the host, port, etc. for a filesystem.

</description>

</property>

**• conf/mapred-site.xml:**

<property>

<name>mapred.job.tracker</name>

<value>localhost:54311</value>

<description>The host and port that the MapReduce job tracker runs at. If "local", then jobs are run in- process as a single map and reduce task.

</description>

</property>

**• conf/hdfs-site.xml:**

<property>

<name>dfs.replication</name>

<value>1</value>

<description>Default block replication. The actual number of replications can be specified when the file is created. The default is used if replication is not specified in create time.

</description>

</property>

After completing the editing of these configuration files ,we need to set up the distributed filesystem across the Hardoop cluster or node

**Formatting the HDFS filesystem via the NameNode**

**hadoop**[**@dhara-VirtualBox**](mailto:dhara@dhara-VirtualBox):~$ /usr/local/hadoop/bin/hadoop namenode -format

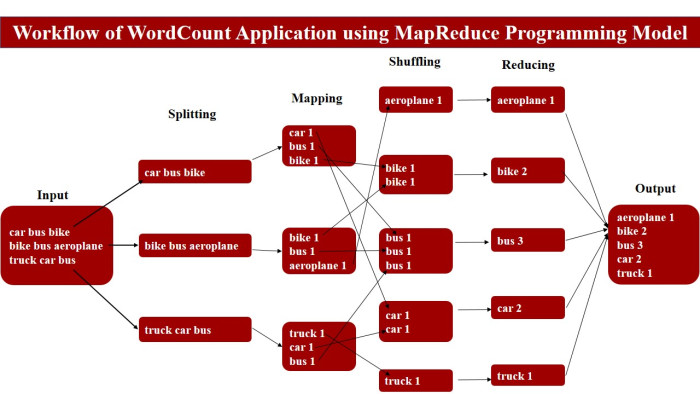
• Start your single node cluster by using the following command line:

**hadoop**[**@dhara-VirtualBox**](mailto:dhara@dhara-VirtualBox):~$ /usr/local/hadoop/bin/start-all.sh

**hadoop**[**@dhara-VirtualBox**](mailto:dhara@dhara-VirtualBox):~$ jps

if you want to stop signgle-node cluster then you can use

**hadoop**[**@dhara-VirtualBox**](mailto:dhara@dhara-VirtualBox):~$ /usr/local/hadoop/bin/stop-all.sh



|  |
| --- |
| **Writing a Hadoop MapReduce example** |

World count example for Hadoop Map reduce file.

* The goal of this example is to calculate how many times each world occurs in the documents.

These documents can be considered as input to MapReduce's file.

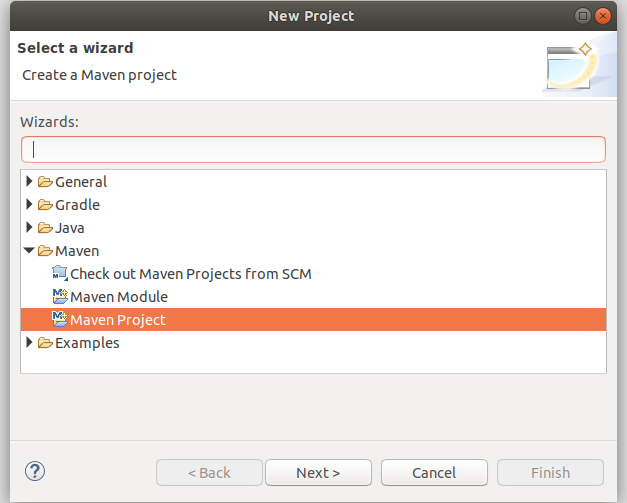
* Here we are using Hadoop with Installing Hadoop on Linux, Ubuntu flavor (single node cluster) .
* Hadoop MapReduce has three objects: **Mapper, Reducer, and Driver**. They can be developed with three Java classes; they are the **Map class, Reduce class,** and **Driver class,** where the Map class denotes the Map phase, the Reducer class denotes the Reduce phase, and the Driver class denotes the class with the main() method to initialize the Hadoop MapReduce program.

|  |
| --- |
| **Steps to run a MapReduce job** |

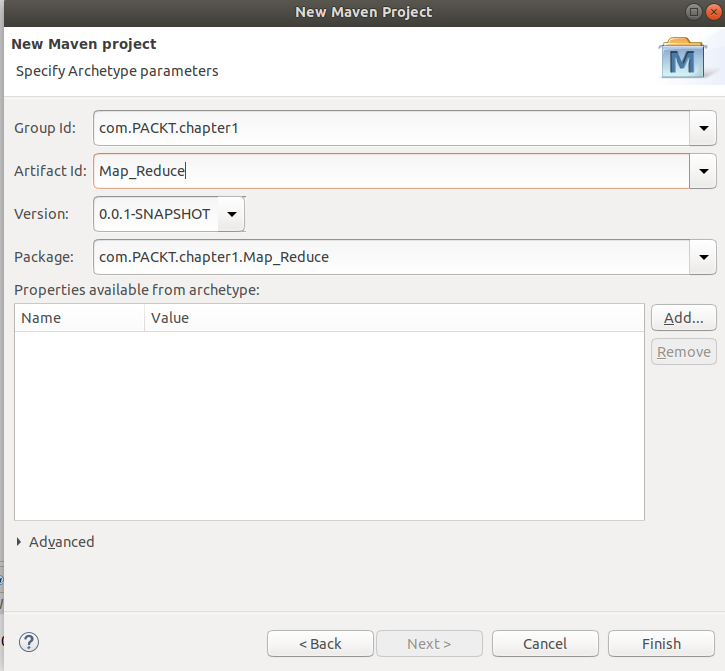
First We install Eclipse and configured with Maven repository and use MapReduce API.

**Create a new Maven Project in eclips by following the steps below**

**1. Go to File→New→Other→Maven Project**

****

**2.Next enter Group Id and Artifact Id for the project .And click Finish button.**

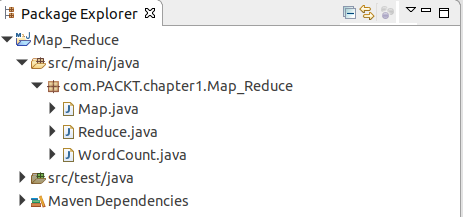


**3. we implement three java classes for world count for Hadoop Map reduce file.**

**(I)Map.java :** Map class for world count Mapper.

**(II) Reduce.java :** Reduce class for world count Reducer.

**(III) WorldCount.java :** This is the task of Driver in the Hadoop MapReduce Driver main file.

****

**Map.java** :

|  |
| --- |
| *// Defining package of the class*  package com.PACKT.chapter1.Map\_Reduce;  *// Importing java libraries*  import java.io.\*;  importjava.util.\*;  import org.apache.hadoop.io.\*;  import org.apache.hadoop.mapred.\*;  *//Defining the Map class*  public class Map extends MapReduceBase implements  Mapper<LongWritableText,  Text,  IntWritable>  {  *//Defining the map method – for processing the data with problem specific logic*  public void map(LongWritable key,  Text value,  OutputCollector<Text,  IntWritable> output,  Reporter reporter) throws IOException  {  *// For breaking the string to tokens and convert them to lowercase*  StringTokenizer st = new StringTokenizer(value.toString(). ToLowerCase());  *// For every string tokens*  while(st.hasMoreTokens())  {  *// Emitting the (key,value) pair with value 1.*  output.collect(new Text(st.nextToken()),  new IntWritable(1));  }  }  } |

**Reduce.java:**

|  |
| --- |
| *// Defining package of the class*  *//Defining package of the class*  package com.PACKT.chapter1.Map\_Reduce;  *// Importing java libraries*  import java.io.\*;  importjava.util.\*;  import org.apache.hadoop.io.\*;  importorg.apache.hadoop.mapred.\*;  *// Defining the Reduce class*  public class Reduce extends MapReduceBase implements  Reducer<Text,  IntWritable,  Text,  IntWritable>  {  *// Defining the reduce method for aggregating the generated output of Map phase*  public void reduce(Text key,  Iterator<IntWritable> values,  OutputCollector<Text,IntWritable>  output,  Reporter reporter) throws IOException  {  *// Setting initial counter value as 0*  int count = 0;  *//For every element with similar key attribute, increment its counter value by adding 1.*  while(values.hasNext())  {  count += values.next().get();  }  *// Emitting the (key,value) pair*  output.collect(key, new IntWritable(count));  }  } |

**WordCount.java:**

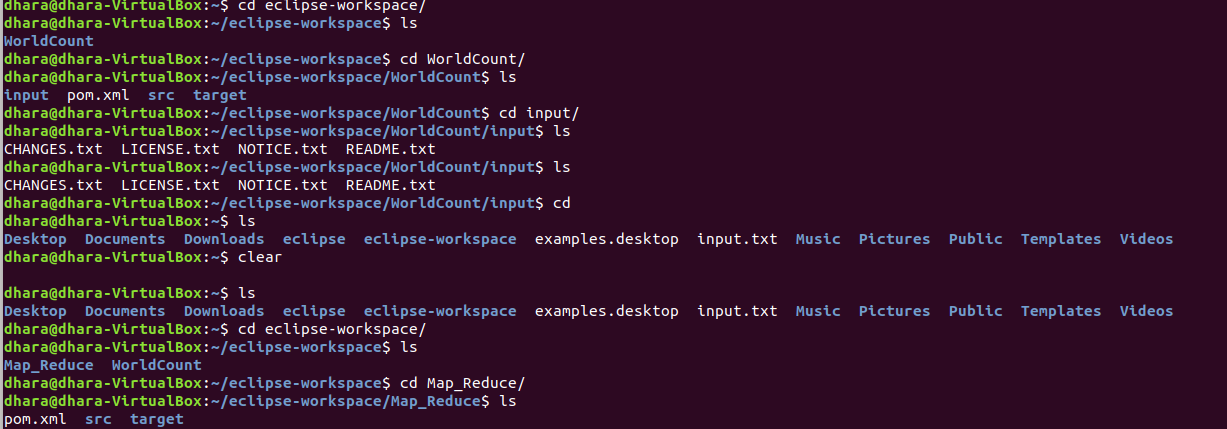
|  |
| --- |
| *//Defining package of the class*  package com.PACKT.chapter1.Map\_Reduce;  *// Importing java libraries*  import java.io.\*;  import org.apache.hadoop.fs.\*;  import org.apache.hadoop.io.\*;  import org.apache.hadoop.mapred.\*;  importorg.apache.hadoop.util.\*;  import org.apache.hadoop.conf.\*;  *//Defining wordcount class for job configuration information*  public class WordCount extends Configured implements Tool  {  public int run(String[] args) throws IOException  {  JobConf conf = new JobConf(WordCount.class);  conf.setJobName("wordcount");  conf.setOutputKeyClass(Text.class); *//For defining the output key format*  conf.setOutputValueClass(IntWritable.class); *//For defining the output value format*    conf.setMapperClass(Map.class); *// For defining the Mapper class implementation*    conf.setReducerClass(Reduce.class); // For defining the Reducer class implementation  conf.setInputFormat(TextInputFormat.class); *// For defining the type of input format*    conf.setOutputFormat(TextOutputFormat.class); *// For defining the type of output format*  *//For defining the command line argument sequence for input data set path*  FileInputFormat.setInputPaths(conf, new Path(args[0]));  *//For defining the command line argument sequence for output data set path*  FileOutputFormat.setOutputPath(conf, new Path(args[1]));  JobClient.runJob(conf);*// For submitting the configuration object*  return 0;  }  *// Defining the main() method to start the execution of the MapReduce program*  public static void main(String[] args) throws Exception  {  intexitCode = ToolRunner.run(new WordCount(), args);  System.exit(exitCode);  }  } |

**5.Setting up Maven pom.xml file**

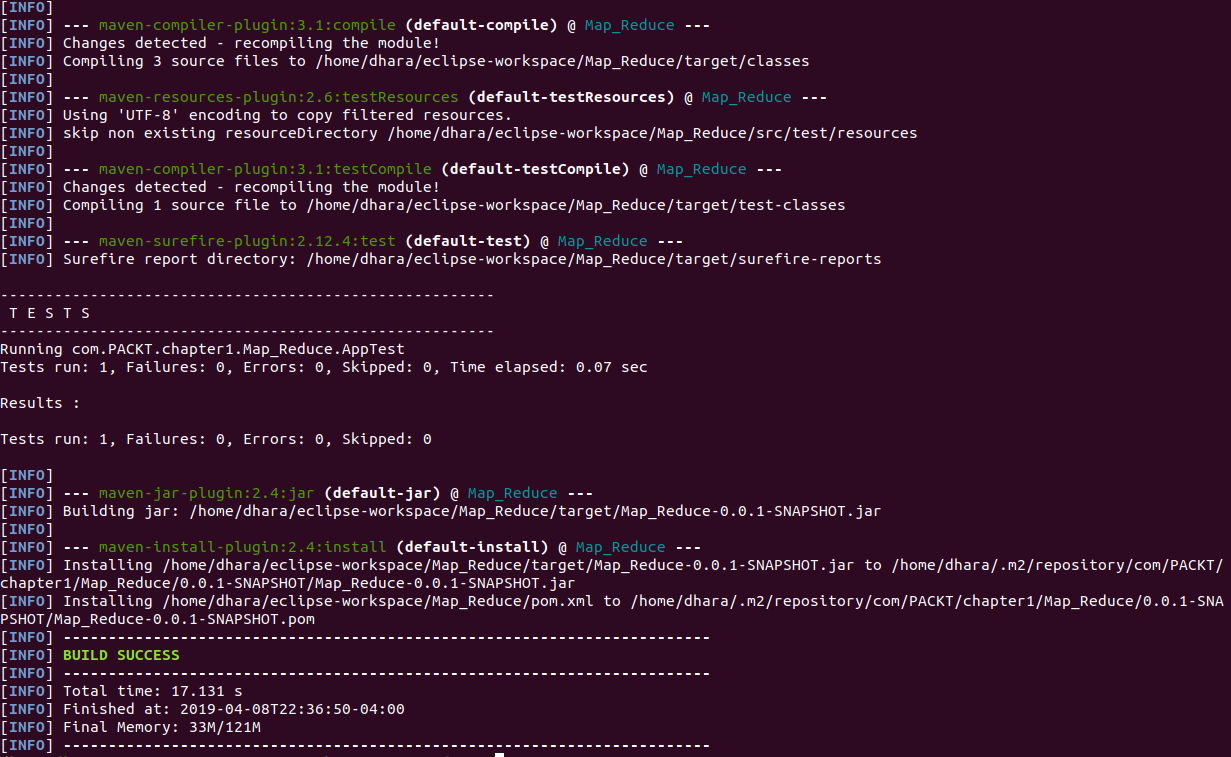
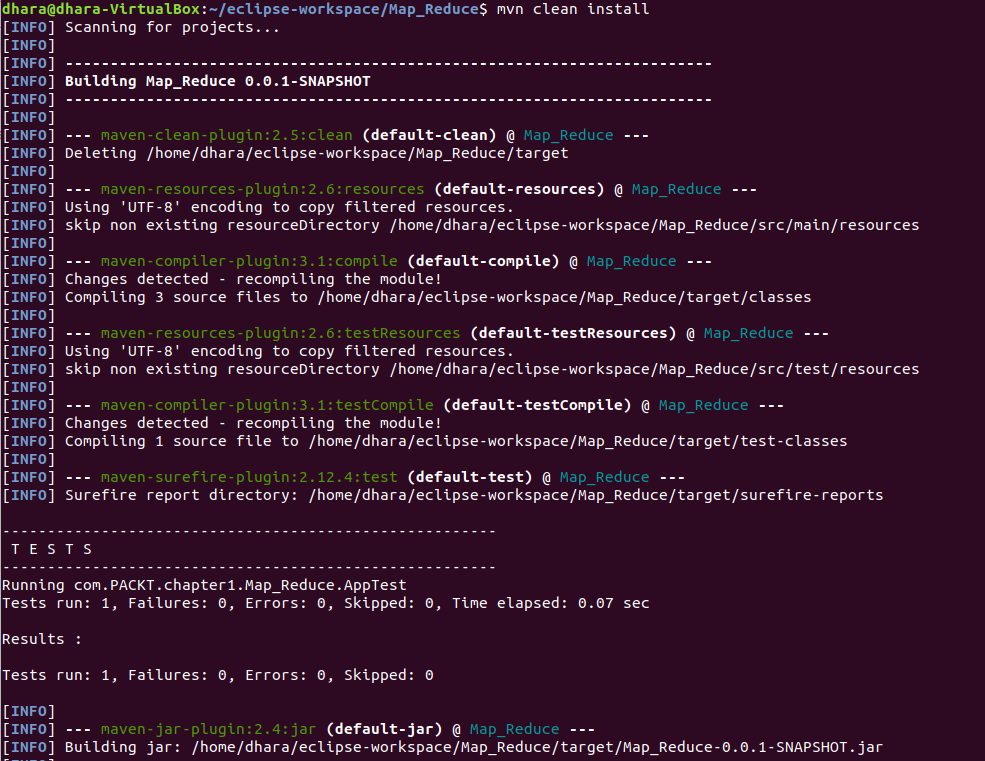
**Add maven repository Hadoop-client,hadoop-common,hadoop-maven-plugins.**

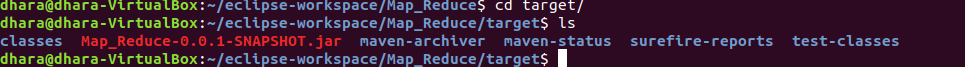
|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.PACKT.chapter1</groupId>  <artifactId>Map\_Reduce</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>  <name>Map\_Reduce</name>  <url>http://maven.apache.org</url>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  </properties>  <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>3.8.1</version>  <scope>test</scope>  </dependency>  <!-- https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-client -->  <dependency>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-client</artifactId>  <version>3.2.0</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-common -->  <dependency>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-common</artifactId>  <version>3.2.0</version>  </dependency>  <dependency>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-core</artifactId>  <version>1.2.1</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-maven-plugins -->  <dependency>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-maven-plugins</artifactId>  <version>2.6.0</version>  </dependency>    </dependencies>  </project> |

**6. Open terminal and go to your project directory eclipse-workspace**

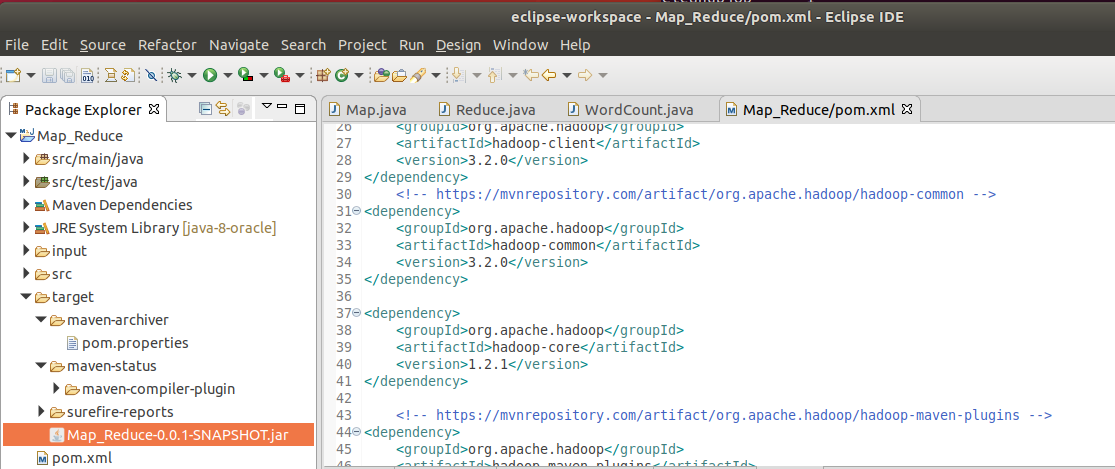


**7. mvn clean install to build jar file with all packaging done by Maven.After build you will get success message.**

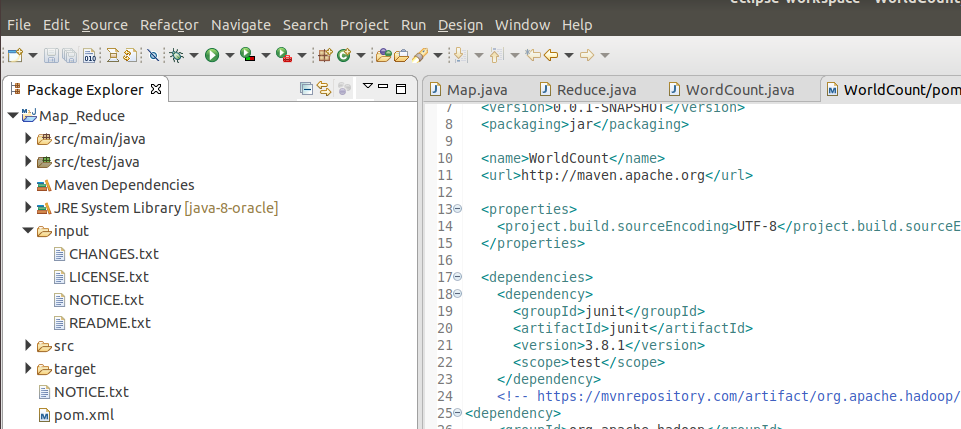




**The jar file is Map\_Reduce-0.0.1-SNAPSHOT.jar and it is created in target folder.**

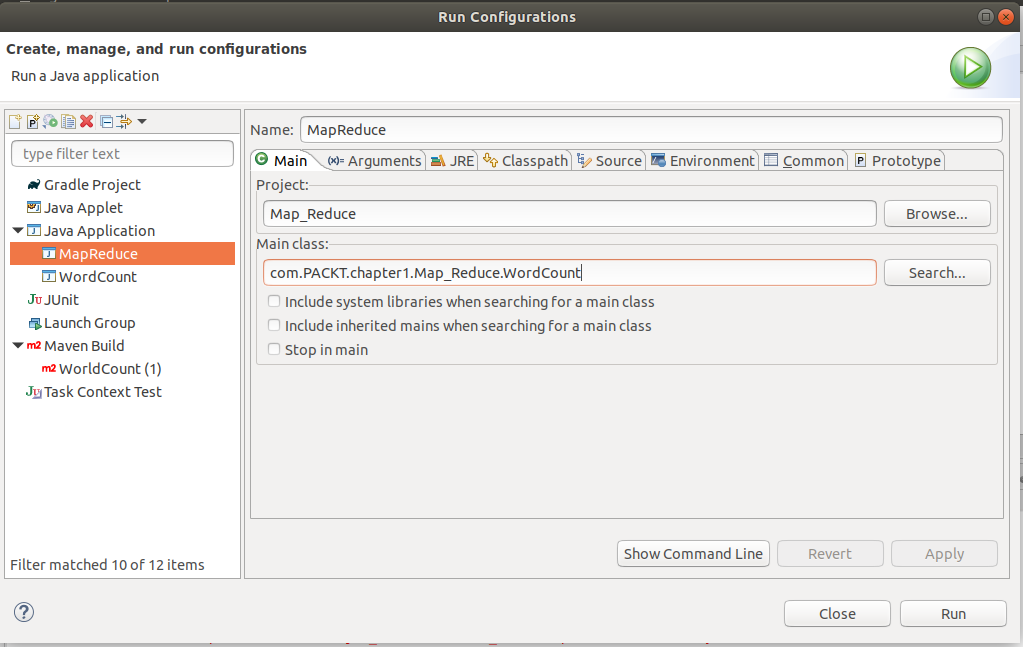


**8. Create one folder ‘input’ and put CHANGES.txt, LICENSE.txt, NOTICE.txt, README.txt in project**

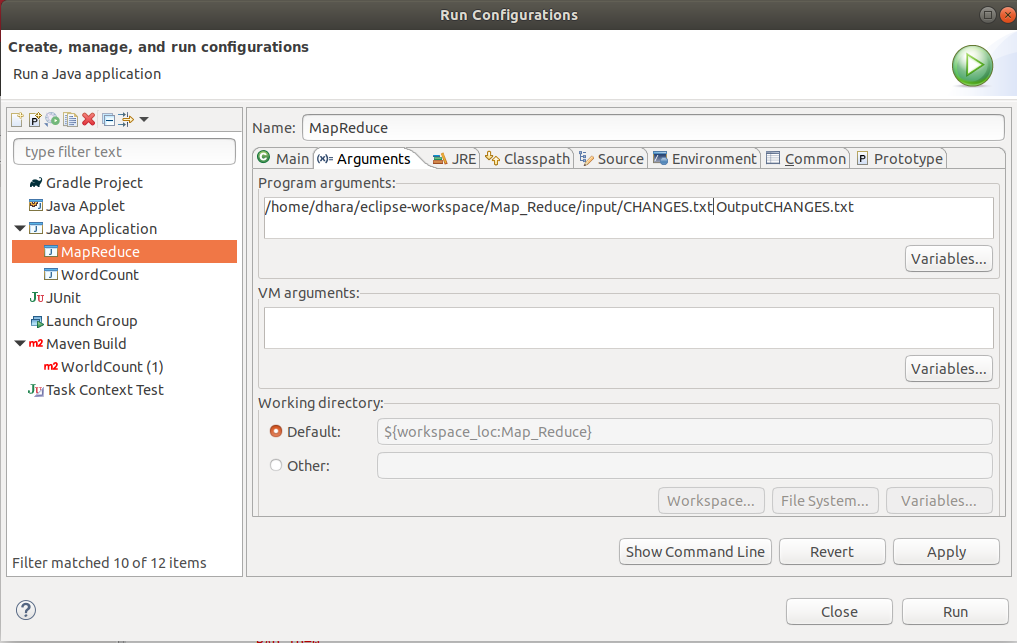


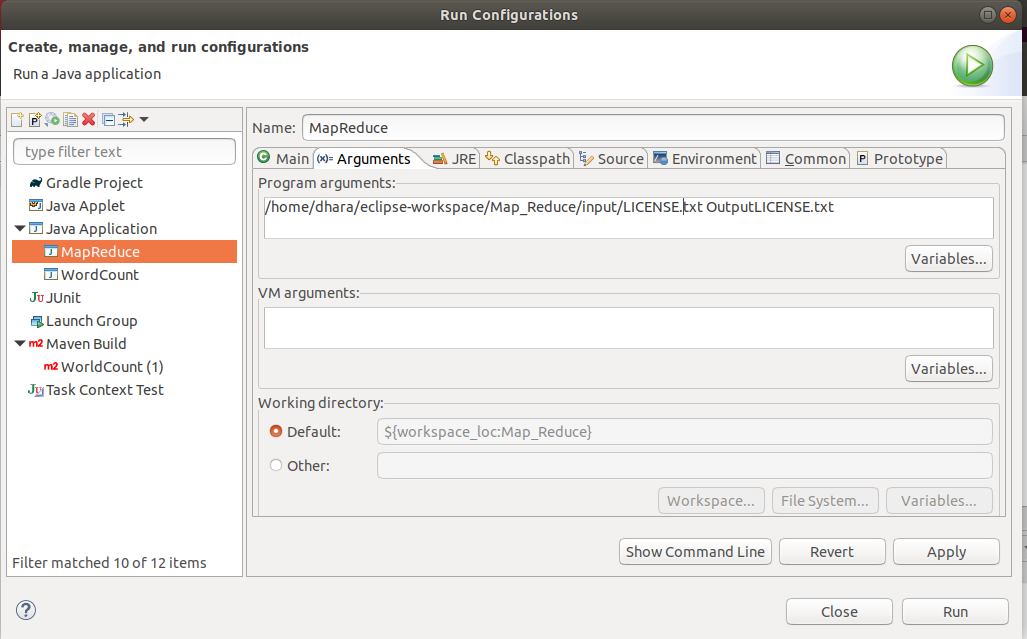
**9. Supply input and output**

**Go to Project folder and right click Map\_Reduce→Run As→Run configuration**

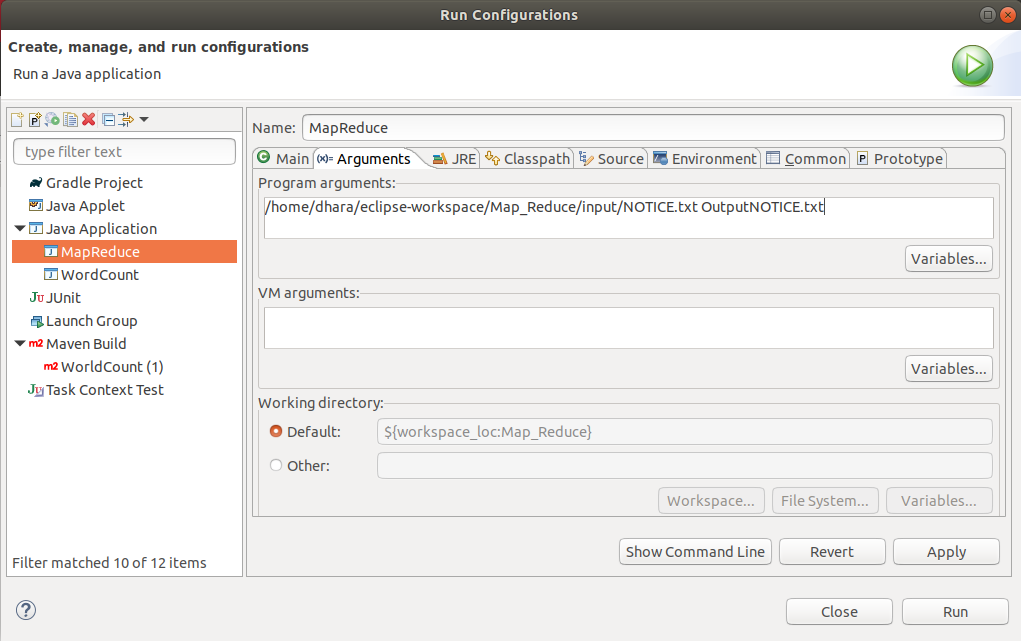


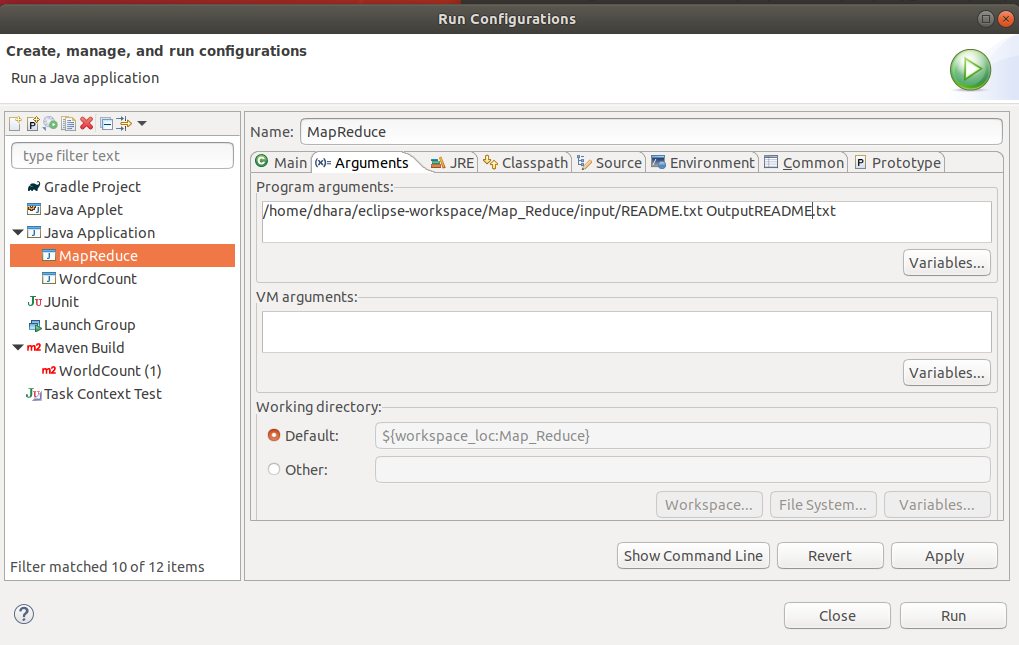
**Input file and output directory for CHANGES.txt**



**Input file and output directoryfor LICENSE.txt**

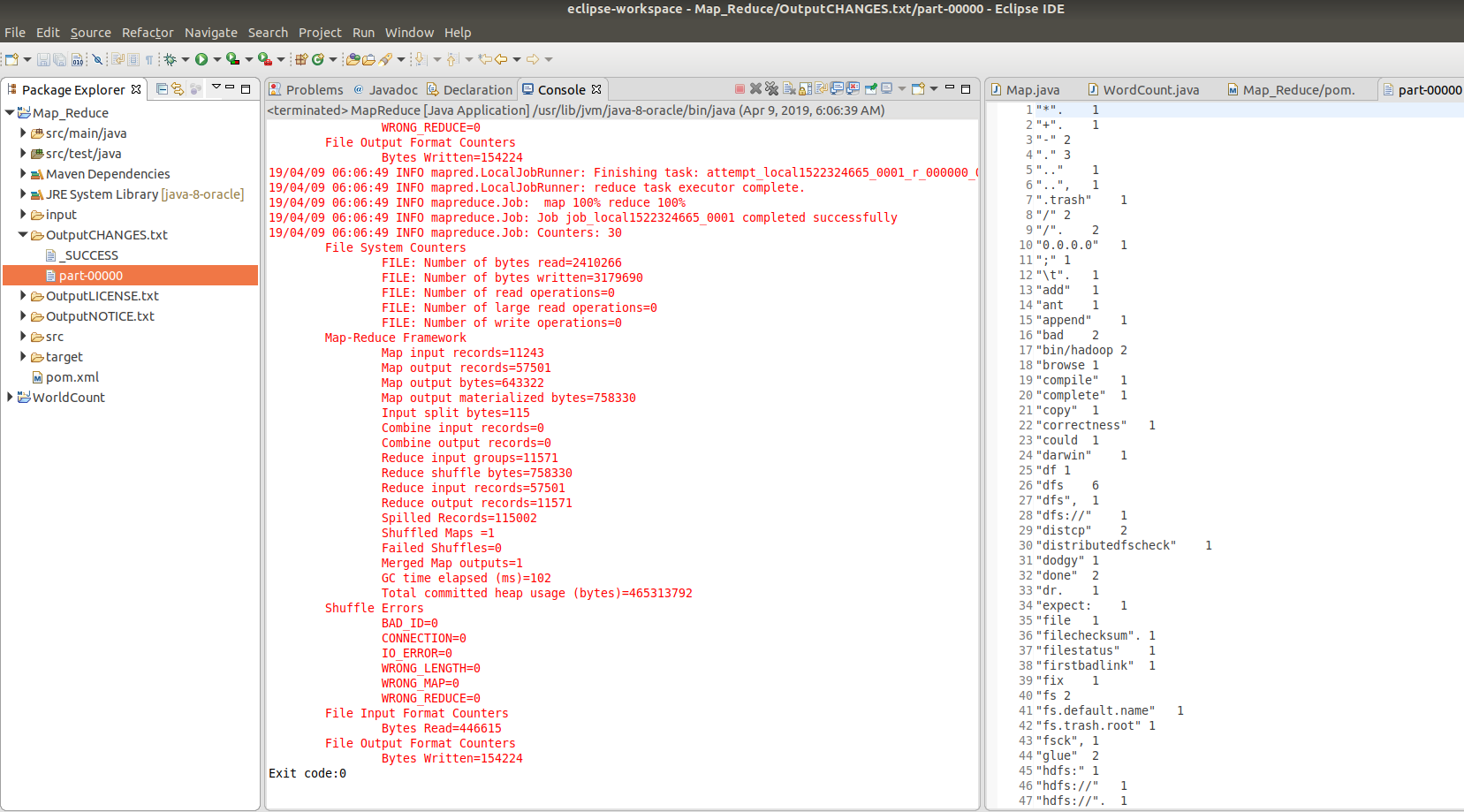
**Input file and output directory for NOTICE.txt**



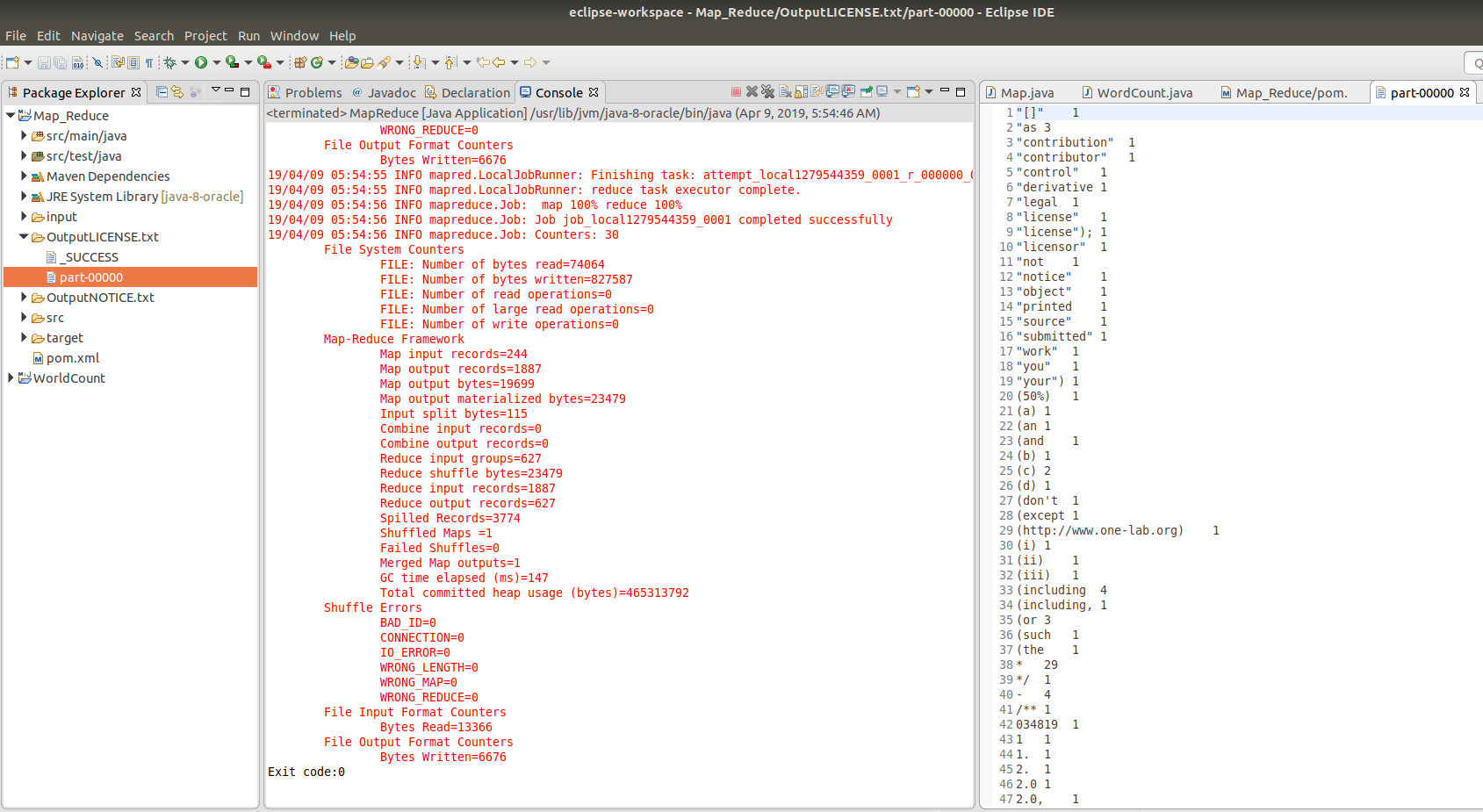
**Input file and output directory for README.txt**

**10. Map Reduce job Execution with Final output of all input .txt file**

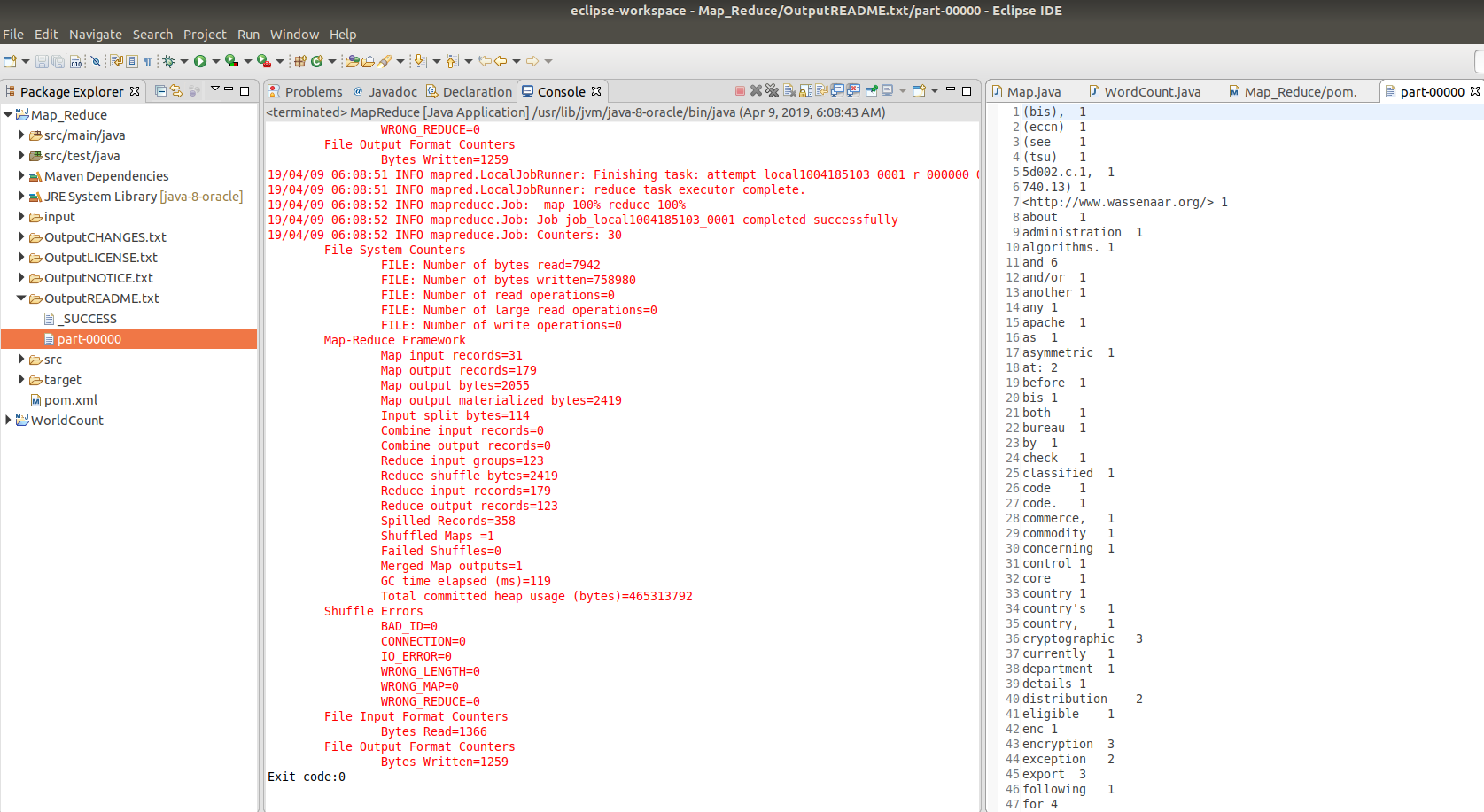
**Input file :- CHANGES.txt**

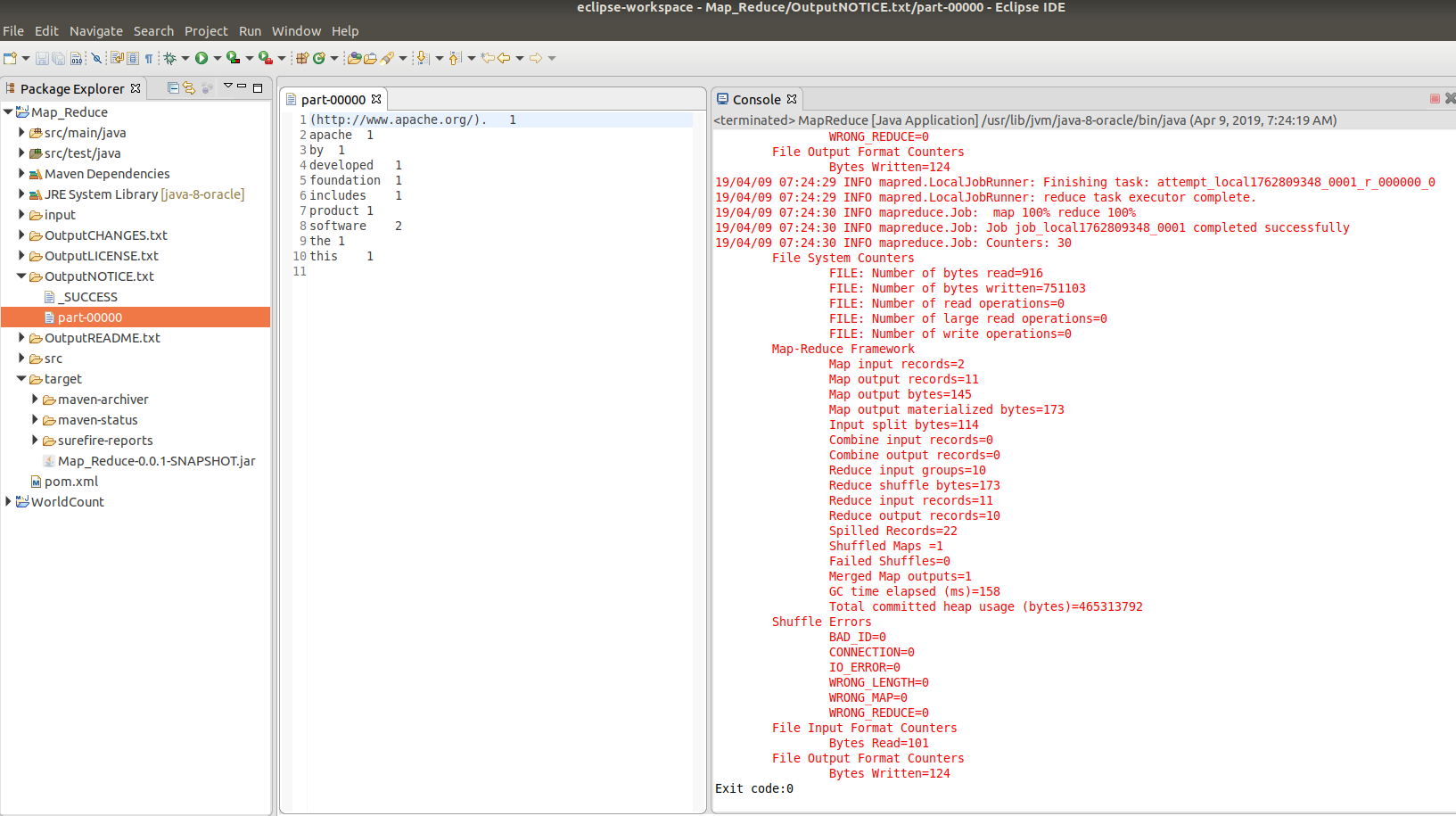


**Input file :- LICENSE.txt**



**I****nput file :- README.txt**



**Input file :-NOTICE.txt**

**-------------------------------------------------\*\*\*\*\*END\*\*\*\*\*---------------------------------------------------**