PRACTICAL NO :02

WordCount using MapReduce Programming

1.1 AIM

Write a simple program for Word Count using Mapreduce Programming.

1.2 OBJECTIVES

In MapReduce word count example, we find out the frequency of each word. Here, the role of Mapper is to map the keys to the existing values and the role of Reducer is to aggregate the keys of common values. So, everything is represented in the form of a Key-value pair.

1.3 THEORY

In Hadoop, Map Reduce is a computation that decomposes large manipulation jobs into individual tasks that can be executed in parallel across a cluster of servers. The results of tasks can be joined together to compute final results.

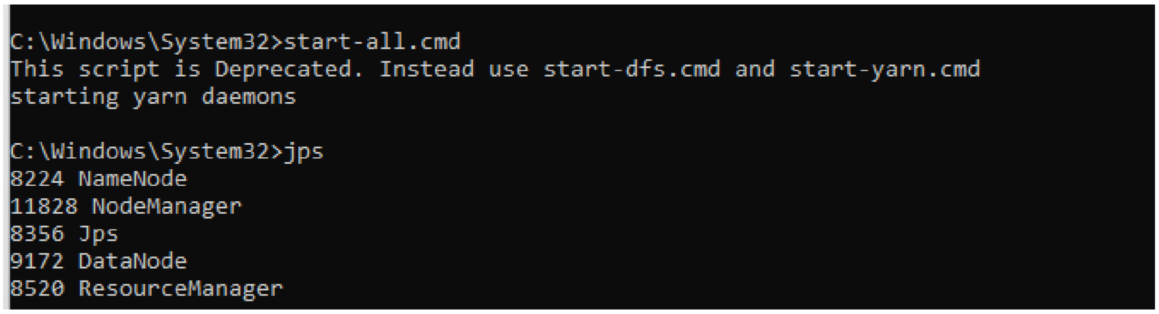
1.4 PREREQUISITE

Java version : 1.8.091 should be installed
Hadoop version : 3.3.0 should be installed
JAVA_HOME should be set
HADOOP_HOME should be set
[Note: All prerequisites of Hadoop are required]

1.5 STEPS

1. Run Hadoop start command to start hadoop 3.3.0

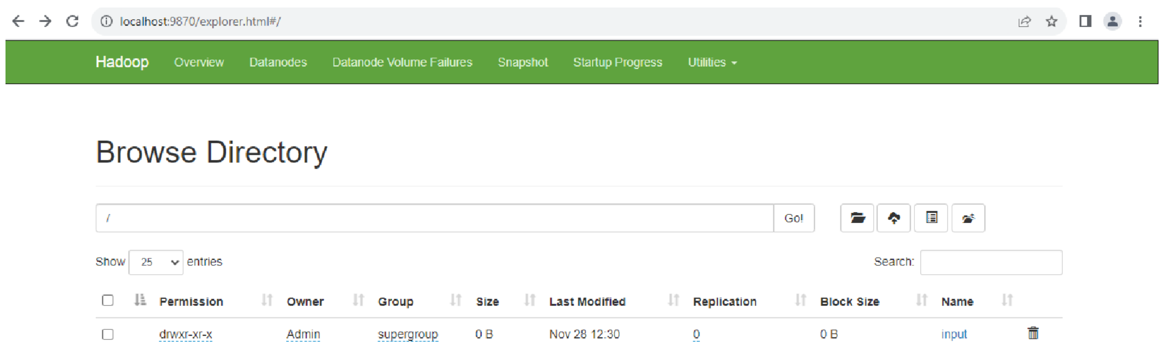
> start-all.cmd



```
C:\Windows\System32>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\Windows\System32>jps
8224 NameNode
11828 NodeManager
8356 Jps
9172 DataNode
8520 ResourceManager
```

2. Create a directory **"input"** in HDFS.

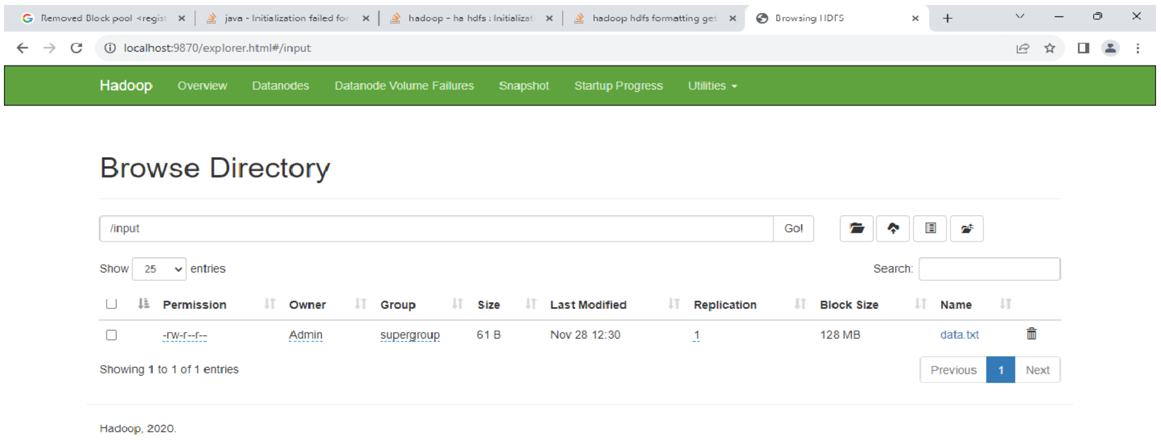
>hdfs dfs -mkdir /input


localhost:9870/explorer.html#/
Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities
Browse Directory
/ Go!
Show 25 entries
Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	Admin	supergroup	0 B	Nov 28 12:30	0	0 B	input

3. Upload or put local file **"data.txt"** to **"input"** directory in HDFS

>hdfs dfs -put "E:\hadoop-3.3.0\data.txt" /input


localhost:9870/explorer.html#/input
Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities
Browse Directory
/input Go!
Show 25 entries
Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	Admin	supergroup	61 B	Nov 28 12:30	1	128 MB	data.txt

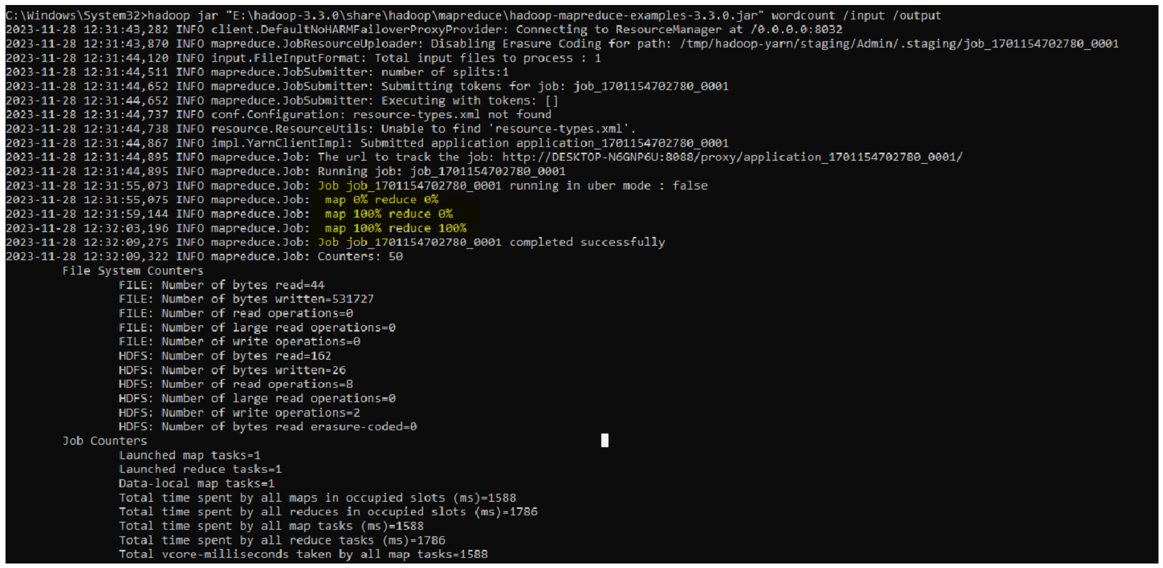
Showing 1 to 1 of 1 entries

Previous 1 Next

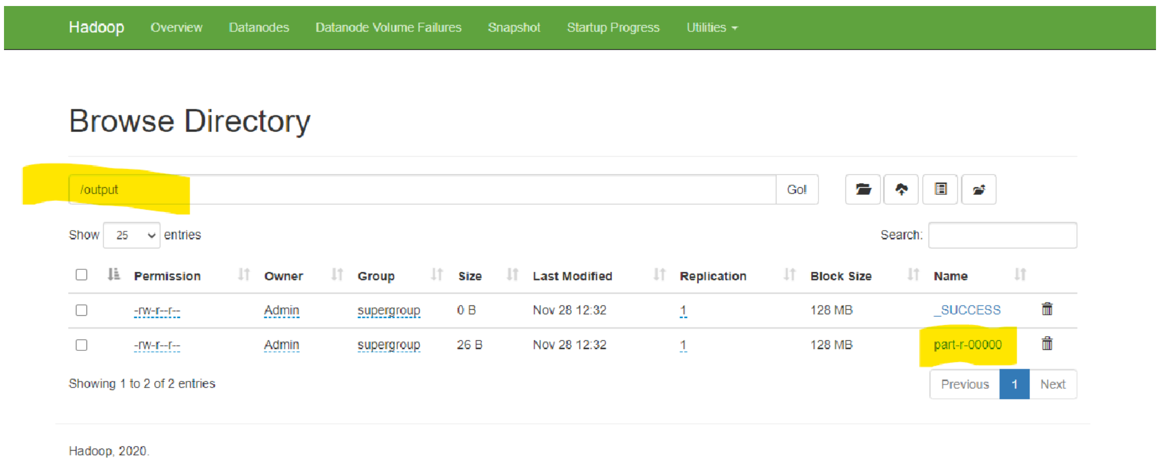
Hadoop, 2020.

4. Run a **jar** file to count the number of words from **“/input”** directory and create output in **“/output”** directory.

>hadoop jar "E:\hadoop-3.3.0\share\hadoop\mapreduce\hadoop-mapreduce-examples-3.3.0.jar" wordcount /input /output



```
C:\Windows\System32>hadoop jar "E:\hadoop-3.3.0\share\hadoop\mapreduce\hadoop-mapreduce-examples-3.3.0.jar" wordcount /input /output
2023-11-28 12:31:43,282 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2023-11-28 12:31:43,870 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/Admin/.staging/job_1701154702780_0001
2023-11-28 12:31:44,120 INFO input.FileInputFormat: Total input files to process : 1
2023-11-28 12:31:44,511 INFO mapreduce.JobSubmitter: number of splits:1
2023-11-28 12:31:44,652 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1701154702780_0001
2023-11-28 12:31:44,652 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-11-28 12:31:44,737 INFO conf.Configuration: resource-types.xml not found
2023-11-28 12:31:44,738 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2023-11-28 12:31:44,867 INFO impl.YarnClientImpl: Submitted application application_1701154702780_0001
2023-11-28 12:31:44,895 INFO mapreduce.Job: The url to track the job: http://DESKTOP-N6GNP6U:8088/proxy/application_1701154702780_0001/
2023-11-28 12:31:44,895 INFO mapreduce.Job: Running job: job_1701154702780_0001
2023-11-28 12:31:55,073 INFO mapreduce.Job: Job job_1701154702780_0001 running in uber mode : false
2023-11-28 12:31:55,075 INFO mapreduce.Job:  map 0% reduce 0%
2023-11-28 12:31:59,144 INFO mapreduce.Job:  map 100% reduce 0%
2023-11-28 12:32:03,196 INFO mapreduce.Job:  map 100% reduce 100%
2023-11-28 12:32:09,275 INFO mapreduce.Job: Job job_1701154702780_0001 completed successfully
2023-11-28 12:32:09,322 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=44
                FILE: Number of bytes written=531727
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=162
                HDFS: Number of bytes written=26
                HDFS: Number of read operations=8
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=1588
                Total time spent by all reduces in occupied slots (ms)=1786
                Total time spent by all map tasks (ms)=1588
                Total time spent by all reduce tasks (ms)=1786
                Total vcore-milliseconds taken by all map tasks=1588
```



Browse Directory

/output

Show 25 entries Search:

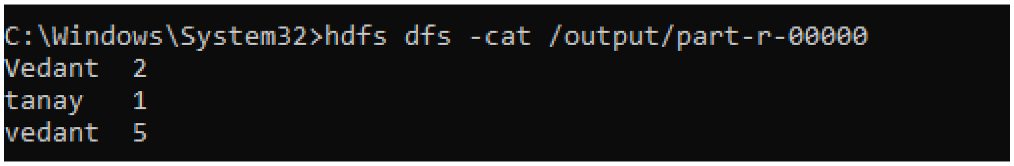
	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
☐	-rw-r--r--	Admin	supergroup	0 B	Nov 28 12:32	1	128 MB	_SUCCESS
☐	-rw-r--r--	Admin	supergroup	26 B	Nov 28 12:32	1	128 MB	part-r-00000

Showing 1 to 2 of 2 entries

Hadoop, 2020.

5. Find the output generated by Mapreduce wordcount in the **“part-r-00000”** file of the **“/output”** directory .

>hdfs dfs -cat /output/part-r-00000



```
C:\Windows\System32>hdfs dfs -cat /output/part-r-00000
Vedant  2
tanay   1
vedant  5
```

1.5 SUMMARY

With this practical, we are now able to:

1. Install hadoop on windows
2. Run mapreduce job using hadoop 3.3.0 to count number of words

1.6 REFERENCES

1. https://www.youtube.com/watch?v=nsi4nVS16lc [preferred]
2. https://www.youtube.com/watch?v=uH5y6nTo_04

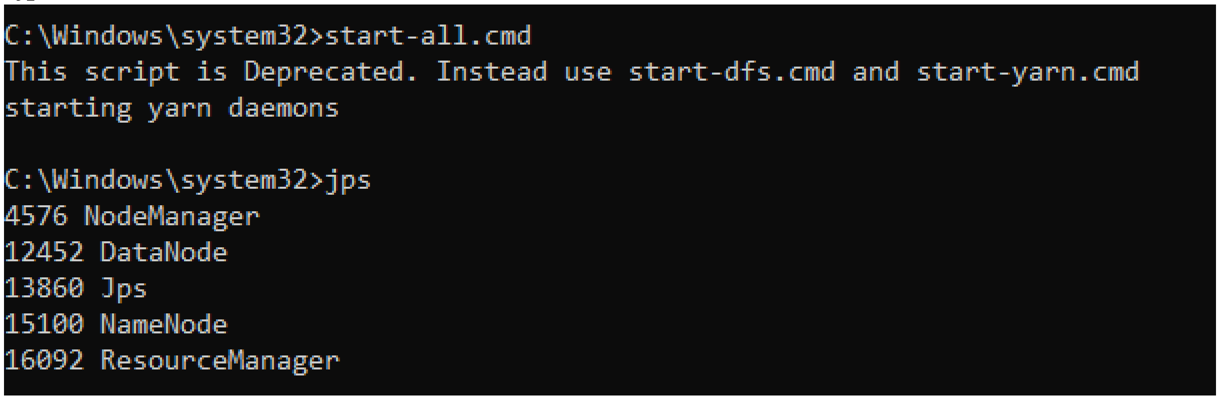
PART II

1.1 AIM

Write a simple program for Word Count using Mapreduce Programming.

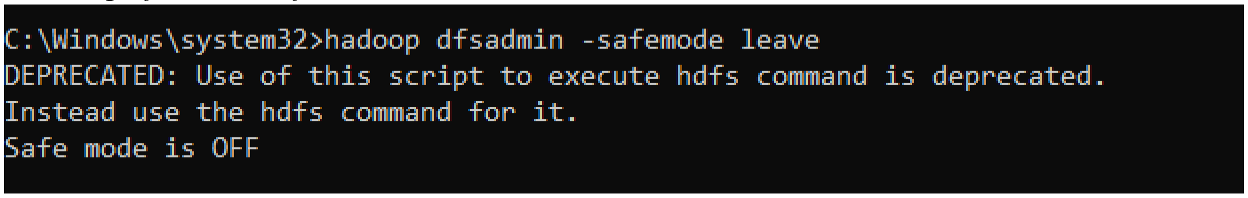
1.2 Steps

1. Start Hadoop 3.3.0
   >start-all.cmd
   >jps



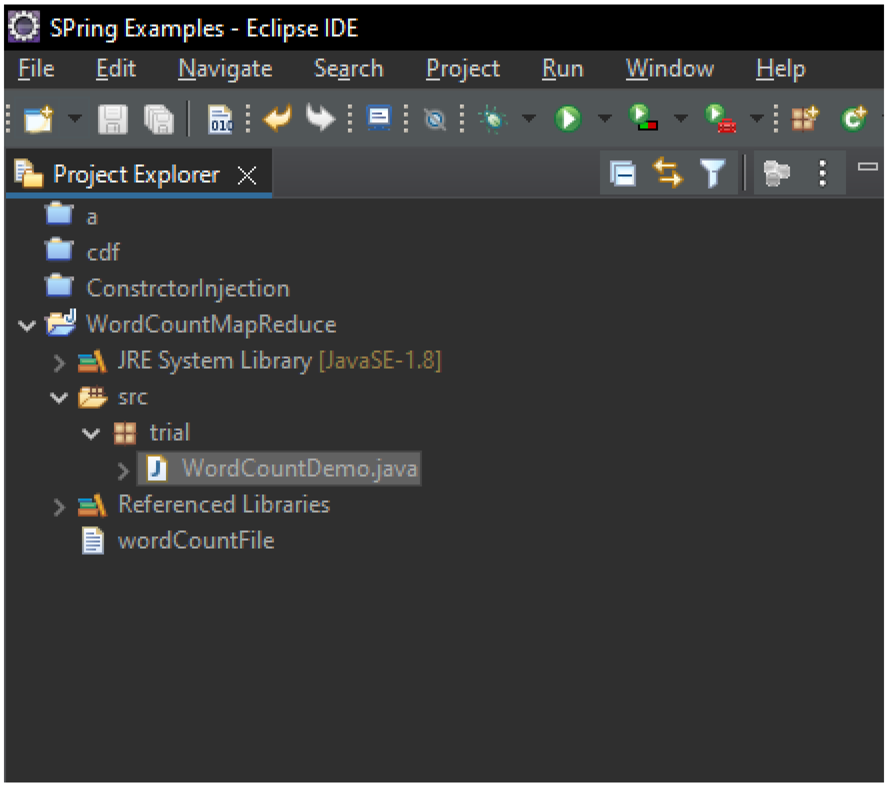
```
C:\Windows\system32>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\Windows\system32>jps
4576 NodeManager
12452 DataNode
13860 Jps
15100 NameNode
16092 ResourceManager
```

2. Leave safe mode using following command
   > hadoop dfsadmin -safemode leave

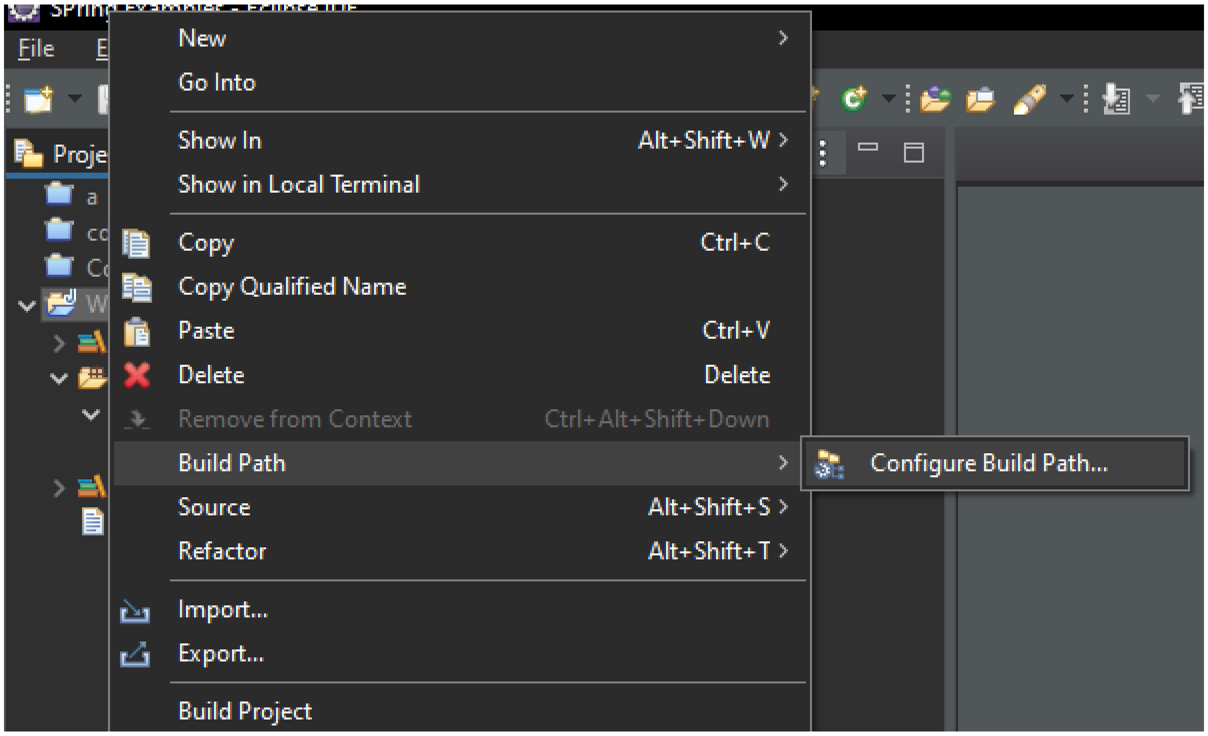


```
C:\Windows\system32>hadoop dfsadmin -safemode leave
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
Safe mode is OFF
```

3. Open eclipse
   a. Create a java project in it
   b. Create package “trial” in that project
   c. Create the class “WordCountDemo.java” in that “trial” package.



d. Add reference of JAR files in the project [Right click on project -> Build path -> Configure build path]

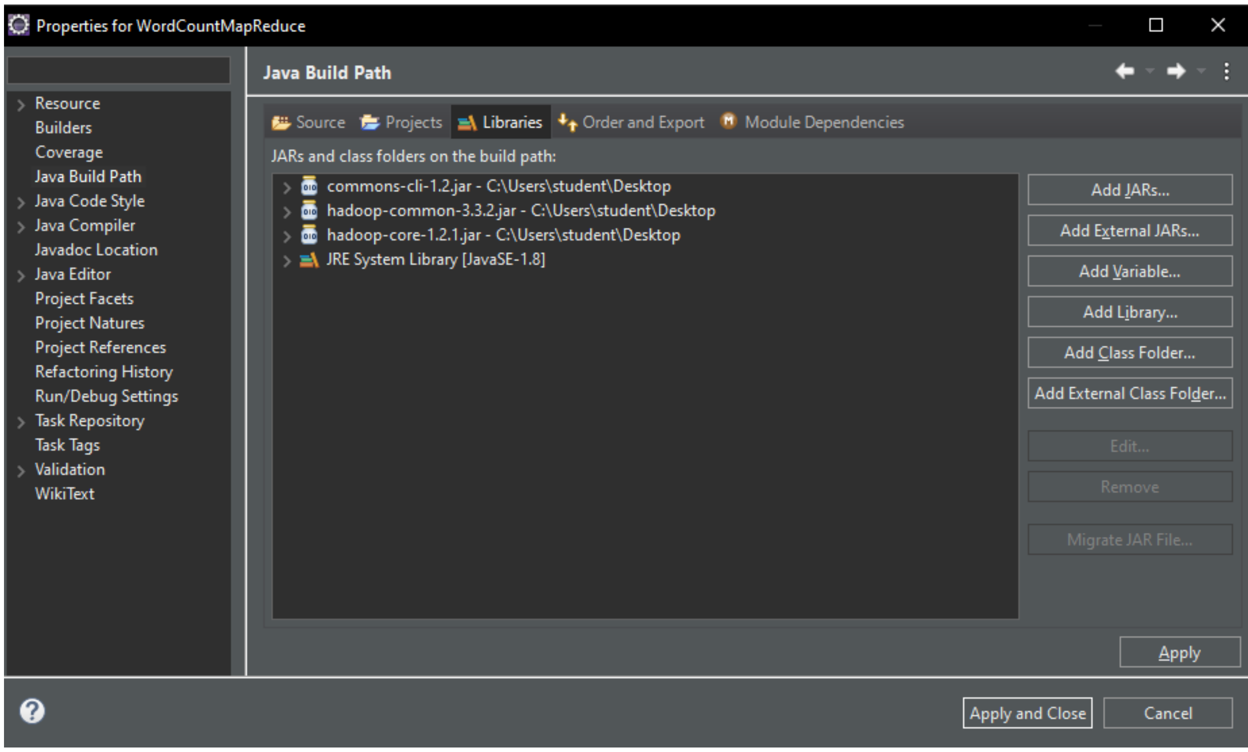


e. Add reference of 3 jars namely:

hadoop-core-1.2.1.jar

commons-cli-1.2.jar

hadoop-common-3.3.2.jar



JARs and class folders on the build path:

```
commons-cli-1.2.jar - C:\Users\student\Desktop
hadoop-common-3.3.2.jar - C:\Users\student\Desktop
hadoop-core-1.2.1.jar - C:\Users\student\Desktop
JRE System Library [JavaSE-1.8]
```

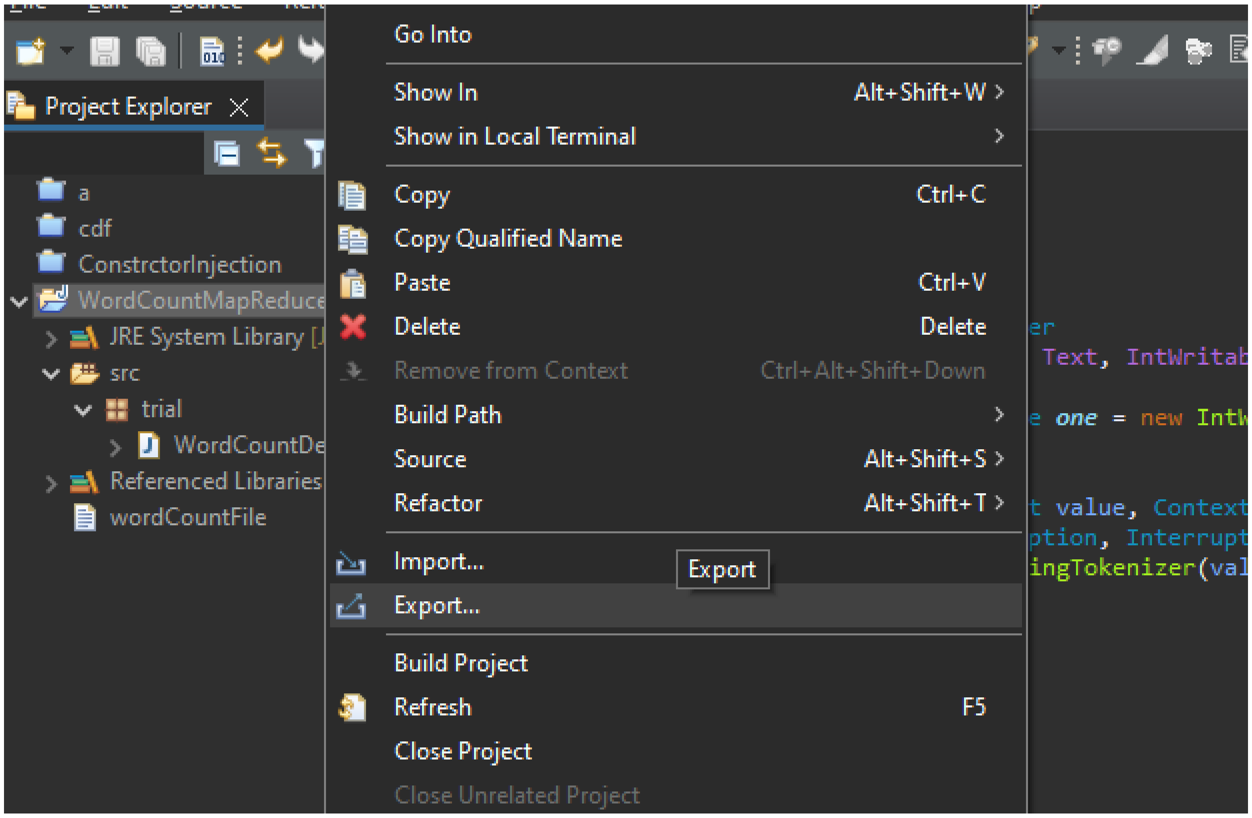
f. Copy following code into your class:

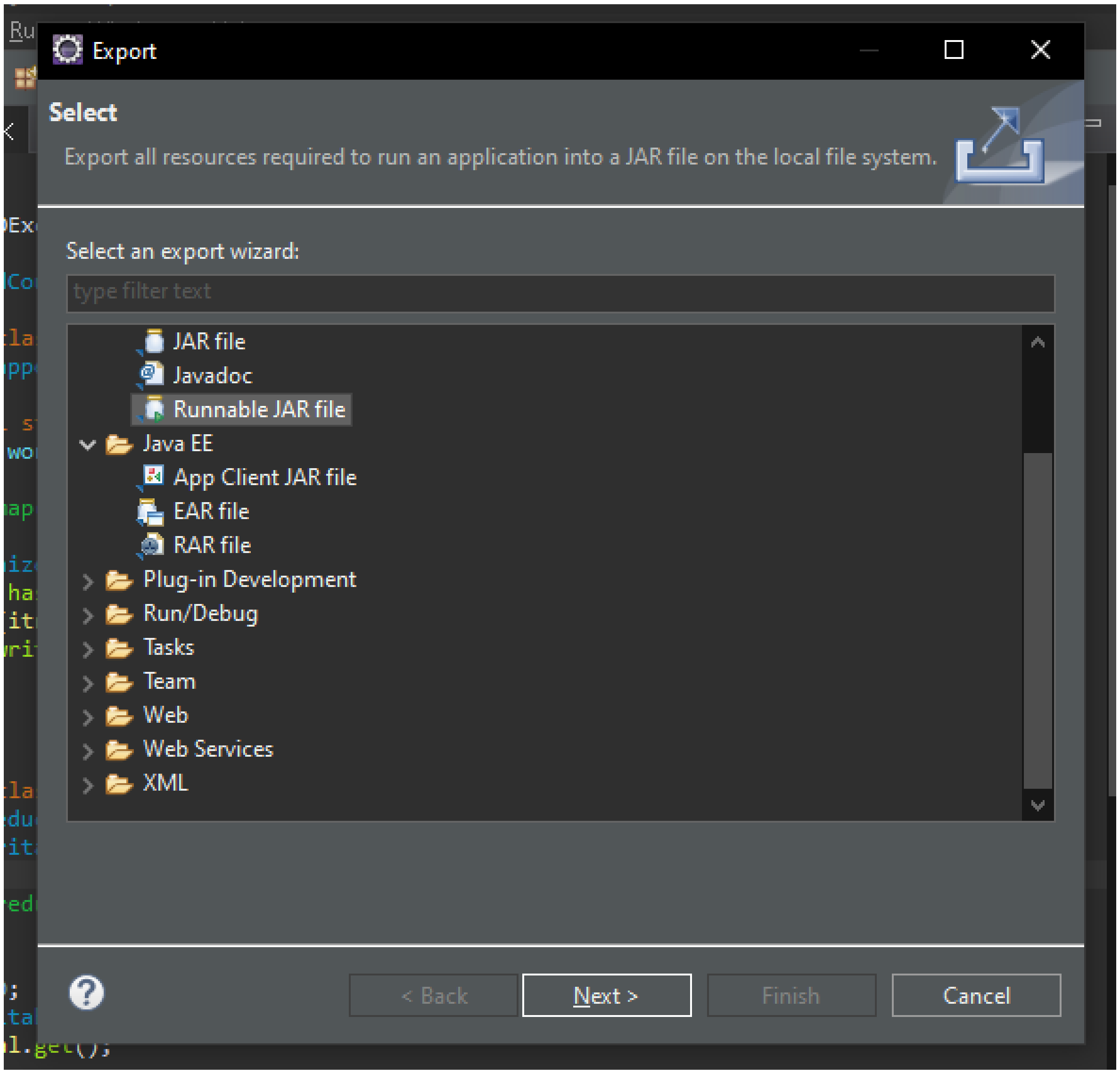
```
package trial;


import java.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 WordCountDemo {

  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
```

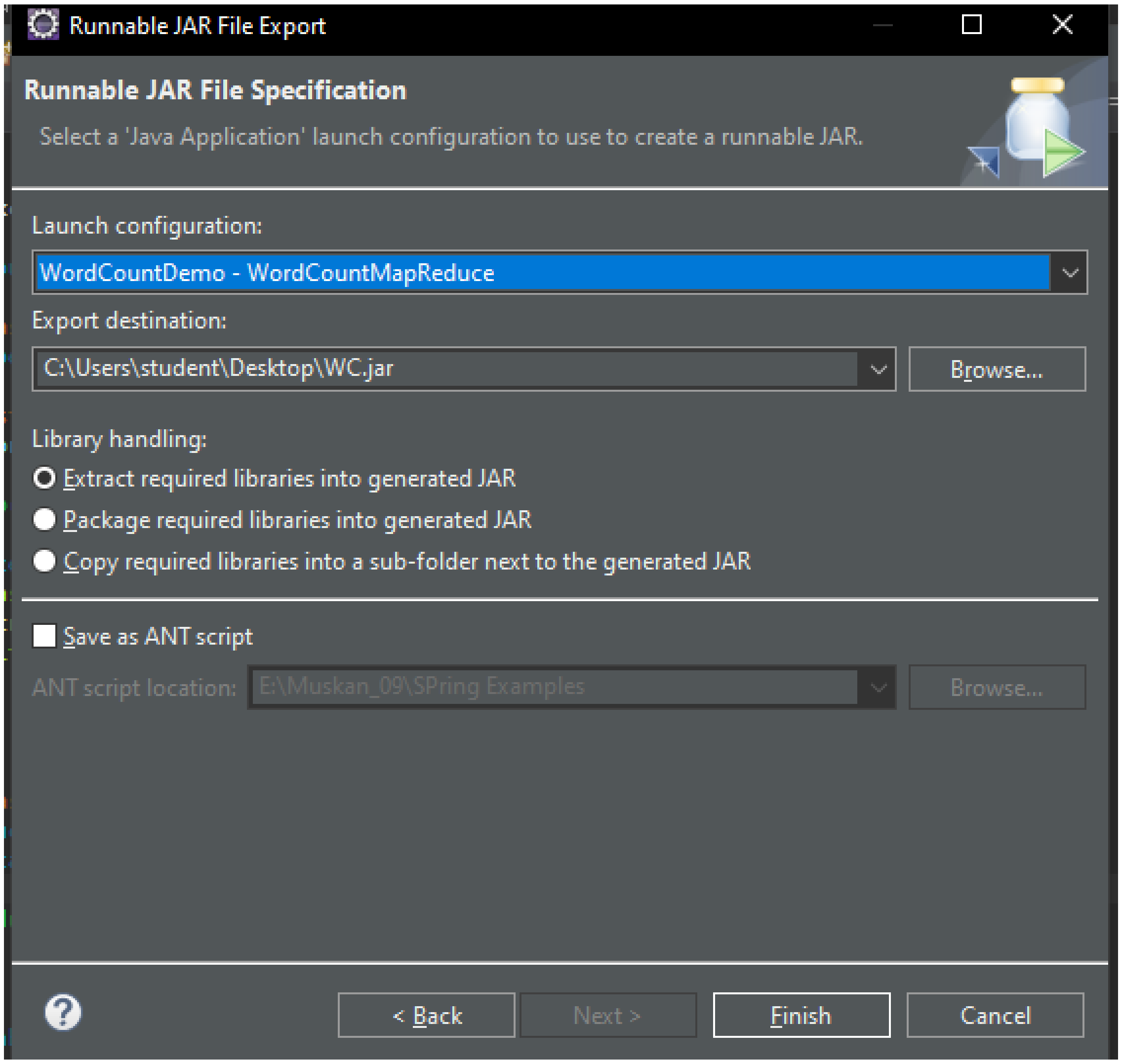
```
          ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCountDemo.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);
  }
}
```

g. Build jar using following

Right click on project and click on export:




Export
Select
Export all resources required to run an application into a JAR file on the local file system.
Select an export wizard:
type filter text
JAR file
Javadoc
Runnable JAR file
Java EE
App Client JAR file
EAR file
RAR file
Plug-in Development
Run/Debug
Tasks
Team
Web
Web Services
XML
< Back
Next >
Finish
Cancel


Runnable JAR File Export

Runnable JAR File Specification

Select a 'Java Application' launch configuration to use to create a runnable JAR.

Launch configuration: WordCountDemo - WordCountMapReduce

Export destination: C:\Users\student\Desktop\WC.jar

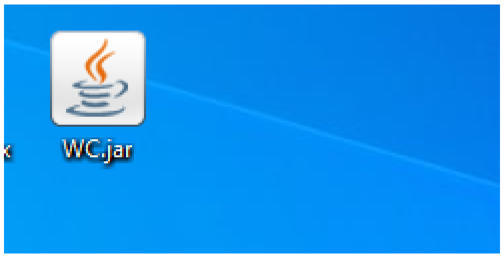
Library handling:

- [✓ Extract required libraries into generated JAR]
- Package required libraries into generated JAR
- Copy required libraries into a sub-folder next to the generated JAR

☐ Save as ANT script

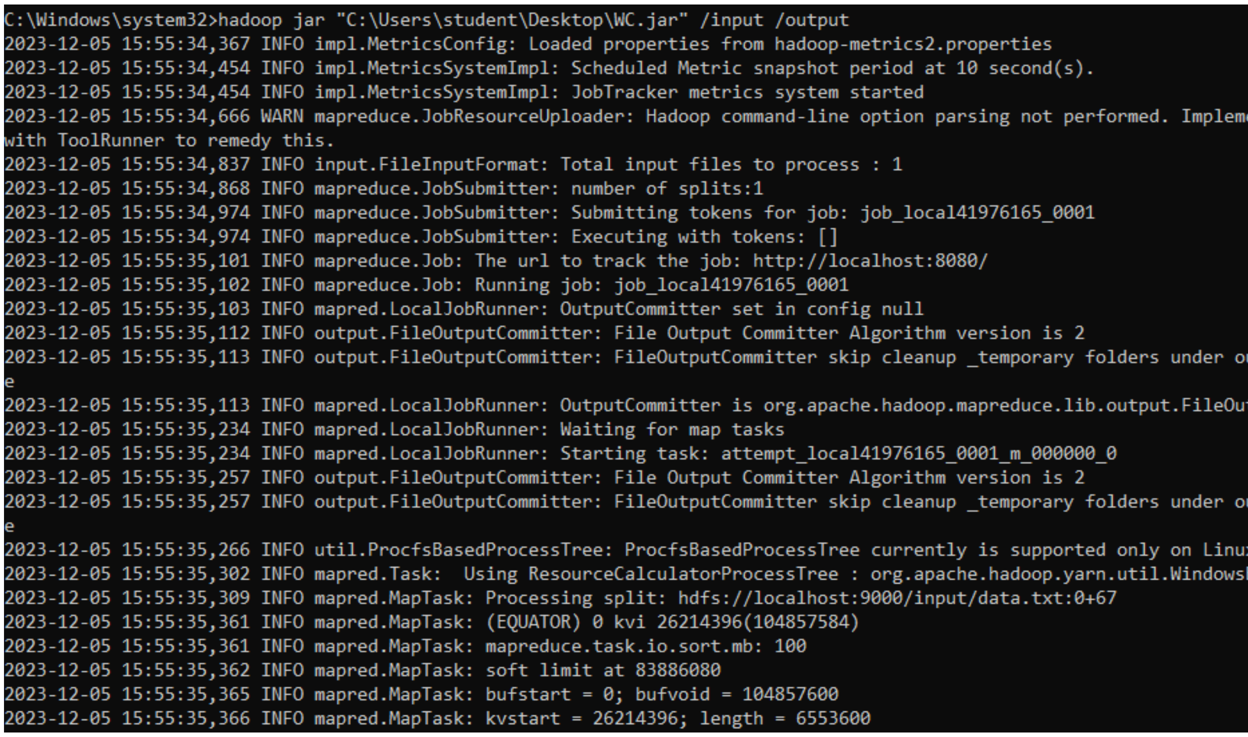
ANT script location: E:\Muskan_09\SPring Examples

h. Now we have WC.jar file

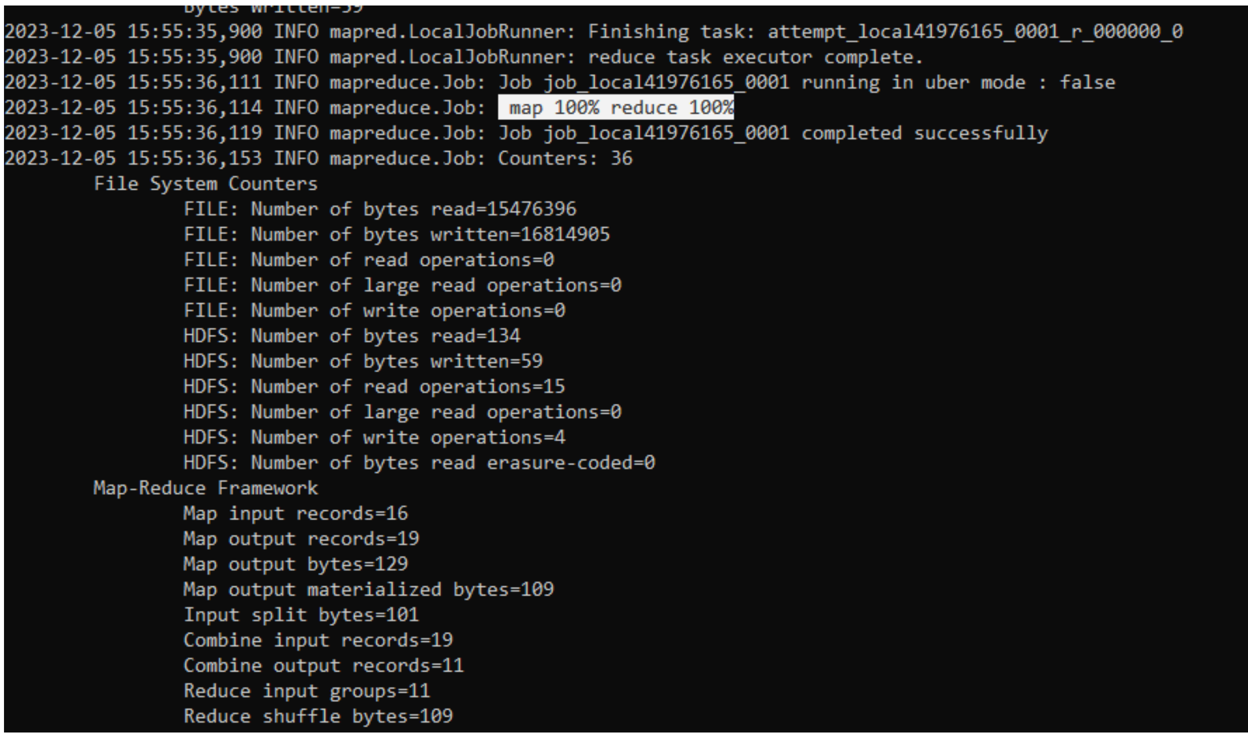


i. Now run command:

```
hdfs dfs -mkdir /input
hdfs dfs -put "E:/data.txt" /input
hadoop jar "C:\Users\student\Desktop\WC.jar" /input /output
```



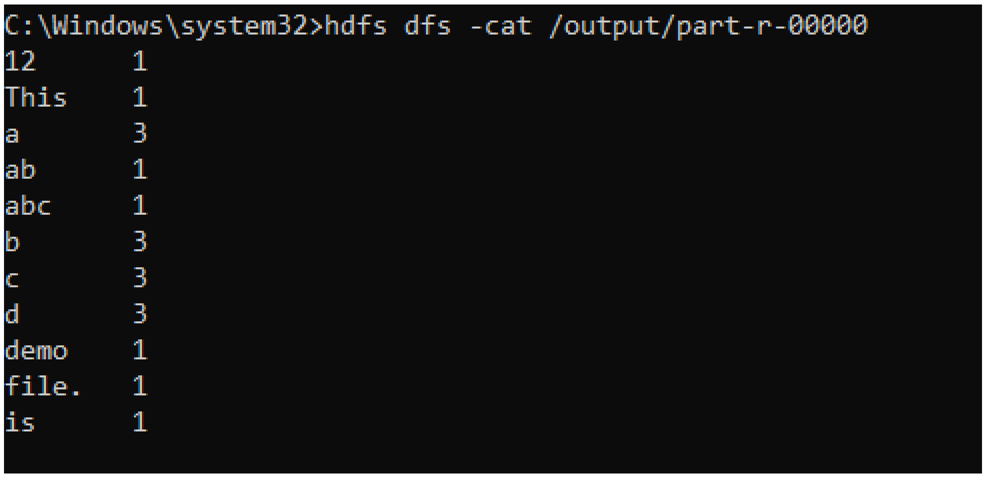
```
C:\Windows\system32>hadoop jar "C:\Users\student\Desktop\WC.jar" /input /output
2023-12-05 15:55:34,367 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2023-12-05 15:55:34,454 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2023-12-05 15:55:34,454 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2023-12-05 15:55:34,666 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Impleme
with ToolRunner to remedy this.
2023-12-05 15:55:34,837 INFO input.FileInputFormat: Total input files to process : 1
2023-12-05 15:55:34,868 INFO mapreduce.JobSubmitter: number of splits:1
2023-12-05 15:55:34,974 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local41976165_0001
2023-12-05 15:55:34,974 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-12-05 15:55:35,101 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2023-12-05 15:55:35,102 INFO mapreduce.Job: Running job: job_local41976165_0001
2023-12-05 15:55:35,103 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2023-12-05 15:55:35,112 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-12-05 15:55:35,113 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under ou
e
2023-12-05 15:55:35,113 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOu
2023-12-05 15:55:35,234 INFO mapred.LocalJobRunner: Waiting for map tasks
2023-12-05 15:55:35,234 INFO mapred.LocalJobRunner: Starting task: attempt_local41976165_0001_m_000000_0
2023-12-05 15:55:35,257 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-12-05 15:55:35,257 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under ou
e
2023-12-05 15:55:35,266 INFO util.ProcfsBasedProcessTree: ProcfsBasedProcessTree currently is supported only on Linu
2023-12-05 15:55:35,302 INFO mapred.Task:  Using ResourceCalculatorProcessTree : org.apache.hadoop.yarn.util.Windows
2023-12-05 15:55:35,309 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/input/data.txt:0+67
2023-12-05 15:55:35,361 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2023-12-05 15:55:35,361 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2023-12-05 15:55:35,362 INFO mapred.MapTask: soft limit at 83886080
2023-12-05 15:55:35,365 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2023-12-05 15:55:35,366 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
```



```
                Bytes Written=59
2023-12-05 15:55:35,900 INFO mapred.LocalJobRunner: Finishing task: attempt_local41976165_0001_r_000000_0
2023-12-05 15:55:35,900 INFO mapred.LocalJobRunner: reduce task executor complete.
2023-12-05 15:55:36,111 INFO mapreduce.Job: Job job_local41976165_0001 running in uber mode : false
2023-12-05 15:55:36,114 INFO mapreduce.Job:  map 100% reduce 100%
2023-12-05 15:55:36,119 INFO mapreduce.Job: Job job_local41976165_0001 completed successfully
2023-12-05 15:55:36,153 INFO mapreduce.Job: Counters: 36
        File System Counters
                FILE: Number of bytes read=15476396
                FILE: Number of bytes written=16814905
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=134
                HDFS: Number of bytes written=59
                HDFS: Number of read operations=15
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=4
                HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
                Map input records=16
                Map output records=19
                Map output bytes=129
                Map output materialized bytes=109
                Input split bytes=101
                Combine input records=19
                Combine output records=11
                Reduce input groups=11
                Reduce shuffle bytes=109
```

j. Now find output using

hdfs dfs -cat /output/part-r-00000



```
C:\Windows\system32>hdfs dfs -cat /output/part-r-00000
12      1
This    1
a       3
ab      1
abc     1
b       3
c       3
d       3
demo    1
file.   1
is      1
```

References:

1. https://hadoop.apache.org/docs/r3.3.0/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
2. https://stackoverflow.com/questions/70928711/map-only-job-is-not-running-stuck-at-running-job
3. https://stackoverflow.com/questions/45687607/waiting-for-am-container-to-be-allocated-launched-and-register-with-rm
4. https://stackoverflow.com/questions/51704288/deploying-application-with-spark-submit-application-is-added-to-the-scheduler-a

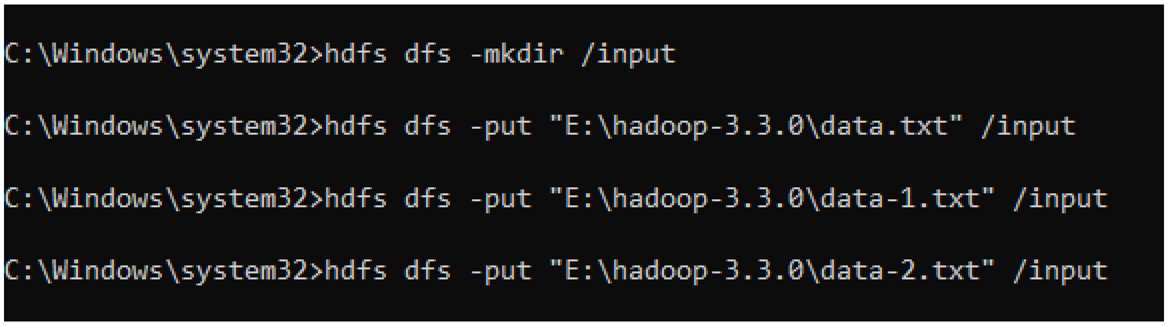
PART III

1.1 AIM

Write a simple program for Word Count using Mapreduce Programming for UNION.

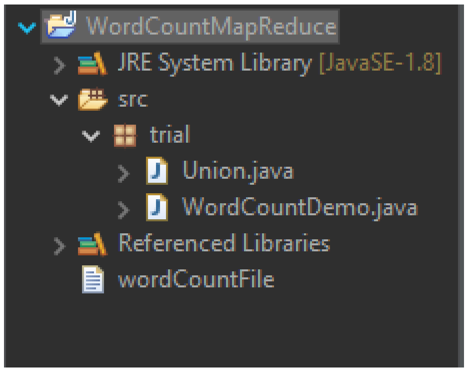
1.2 Steps

1. Create a directory "/input" in HDFS and upload 2-3 files with different data



```
C:\Windows\system32>hdfs dfs -mkdir /input

C:\Windows\system32>hdfs dfs -put "E:\hadoop-3.3.0\data.txt" /input

C:\Windows\system32>hdfs dfs -put "E:\hadoop-3.3.0\data-1.txt" /input

C:\Windows\system32>hdfs dfs -put "E:\hadoop-3.3.0\data-2.txt" /input
```

2. Create a class "Union.java" in same package



```
WordCountMapReduce
  JRE System Library [JavaSE-1.8]
  src
    trial
      Union.java
      WordCountDemo.java
  Referenced Libraries
  wordCountFile
```

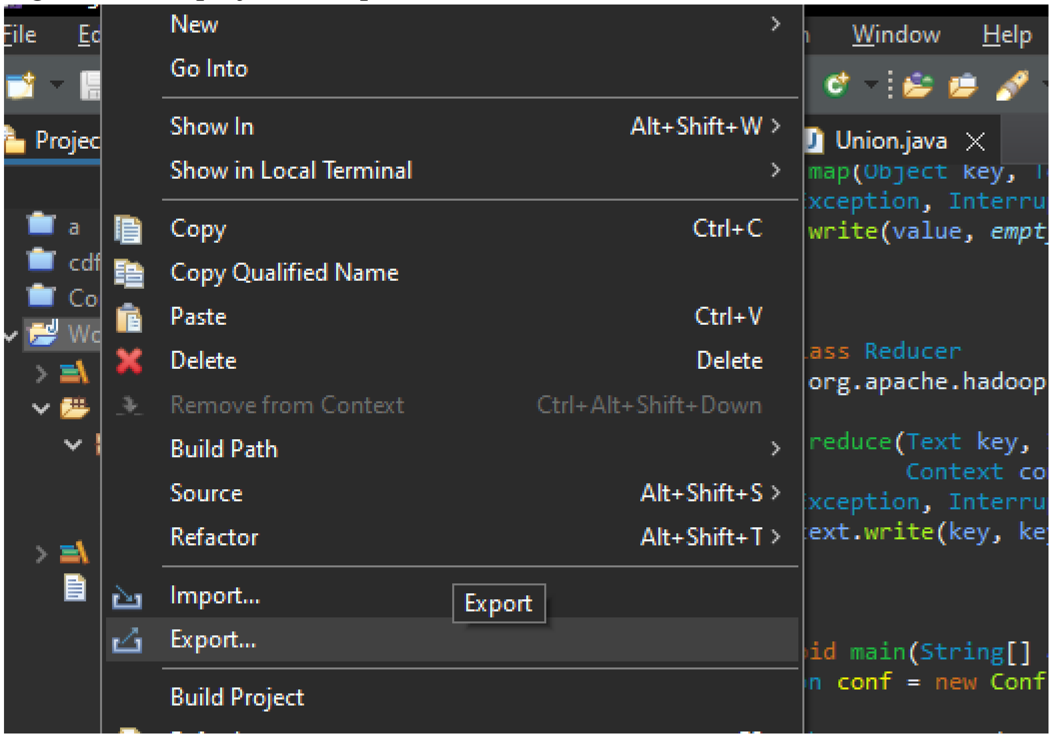
3. Write the following code into "Union.java" files.

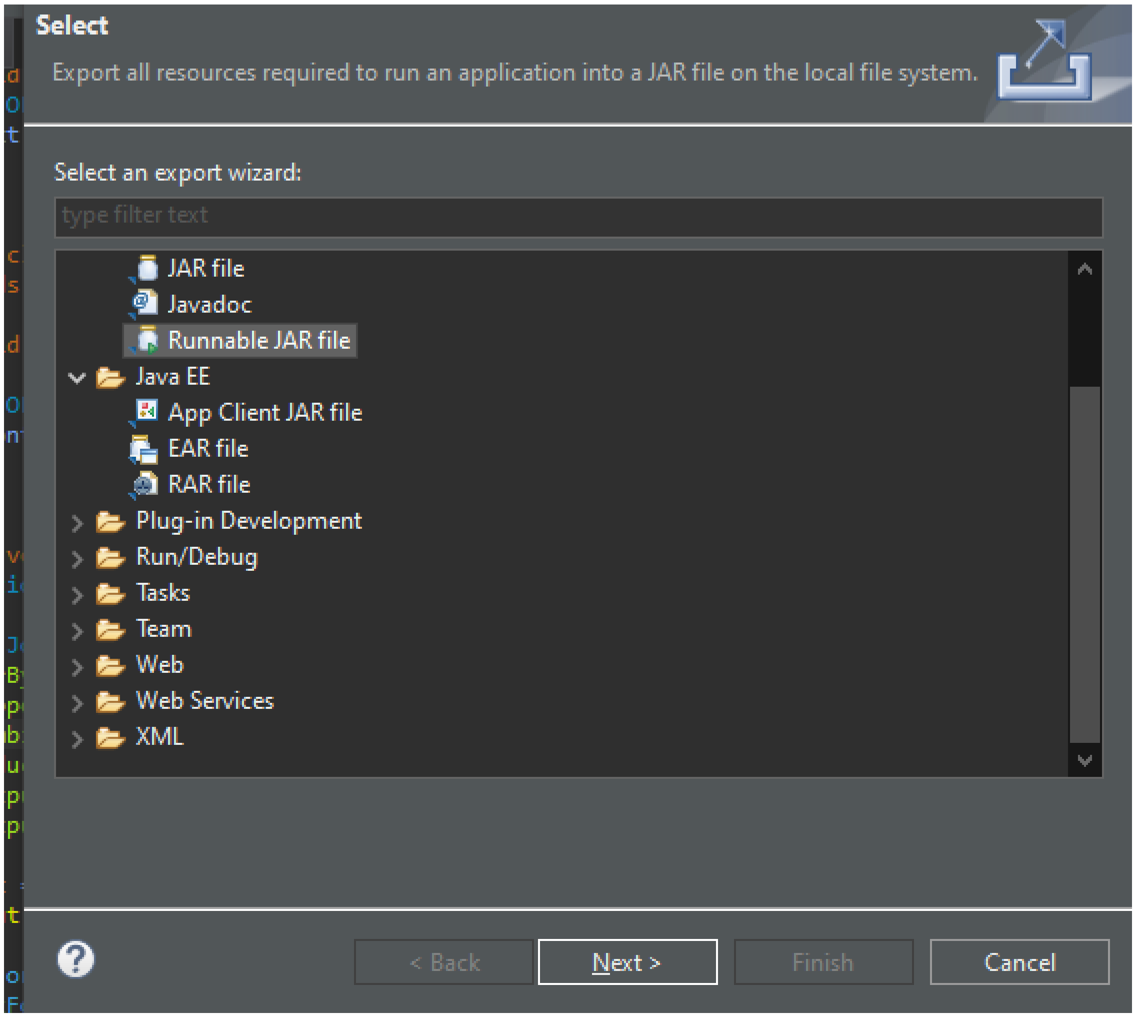
```
package trial;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import java.io.IOException;
public class Union {
	private static Text emptyWord = new Text("");
	public static class Mapper
			extends org.apache.hadoop.mapreduce.Mapper<Object, Text, Text, Text> {
		public void map(Object key, Text value, Context context
                ) throws IOException, InterruptedException {
                        context.write(value, emptyWord);
                }
        }
        public static class Reducer
                        extends org.apache.hadoop.mapreduce.Reducer<Text, Text, Text, Text> {
                public void reduce(Text key, Iterable<Text> _values,
                                                        Context context
                ) throws IOException, InterruptedException {
                                context.write(key, key);
                }
        }
        public static void main(String[] args) throws Exception {
                Configuration conf = new Configuration();
                Job job = Job.getInstance(conf, "Word sum");
                job.setJarByClass(Union.class);
                job.setMapperClass(Mapper.class);
                job.setCombinerClass(Reducer.class);
                job.setReducerClass(Reducer.class);
                job.setOutputKeyClass(Text.class);
                job.setOutputValueClass(Text.class);
                Path input = new Path( args[0]);
                Path output = new Path(args[1]);
                FileInputFormat.addInputPath(job, input);
                FileOutputFormat.setOutputPath(job, output);
                System.exit(job.waitForCompletion(true) ? 0 : 1);
        }
}
```

```
package trial;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
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;

import java.io.IOException;

public class Union {

        private static Text emptyWord = new Text("");
        public static class Mapper
                        extends org.apache.hadoop.mapreduce.Mapper<Object, Text, Text,
Text> {

                public void map(Object key, Text value, Context context
                ) throws IOException, InterruptedException {
                        context.write(value, emptyWord);
                }
        }

        public static class Reducer
                        extends  org.apache.hadoop.mapreduce.Reducer<Text,  Text,  Text,
Text> {

                public void reduce(Text key, Iterable<Text> _values,
                                                Context context
                ) throws IOException, InterruptedException {
                                context.write(key, key);
                }
        }

        public static void main(String[] args) throws Exception {
                Configuration conf = new Configuration();

                Job job = Job.getInstance(conf, "Word sum");
                job.setJarByClass(Union.class);
                job.setMapperClass(Mapper.class);
                job.setCombinerClass(Reducer.class);
                job.setReducerClass(Reducer.class);
                job.setOutputKeyClass(Text.class);
                job.setOutputValueClass(Text.class);
```

```
                    Path input = new Path( args[0]);
                    Path output = new Path(args[1]);

                    FileInputFormat.addInputPath(job, input);
                    FileOutputFormat.setOutputPath(job, output);
                    System.exit(job.waitForCompletion(true) ? 0 : 1);
            }
    }
```

4. Create a jar file using following steps:
   Right Click on project -> export ->

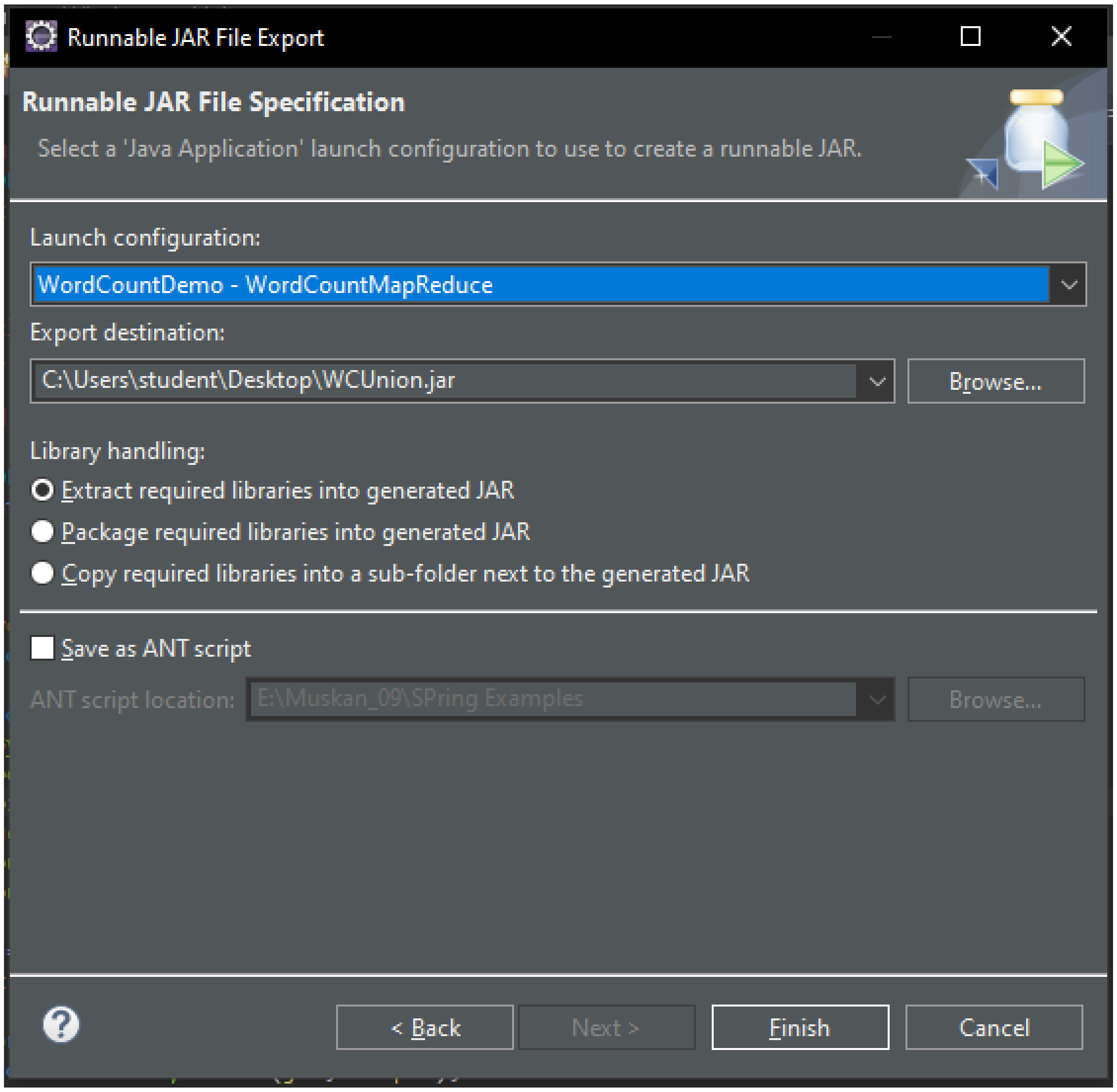




Select

Export all resources required to run an application into a JAR file on the local file system.

Select an export wizard:


Runnable JAR File Export

Runnable JAR File Specification

Select a 'Java Application' launch configuration to use to create a runnable JAR.

Launch configuration: WordCountDemo - WordCountMapReduce

Export destination: C:\Users\student\Desktop\WCUnion.jar

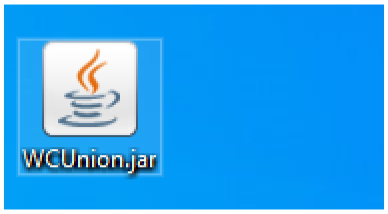
Library handling:

- [✓ Extract required libraries into generated JAR]
- Package required libraries into generated JAR
- Copy required libraries into a sub-folder next to the generated JAR

Save as ANT script

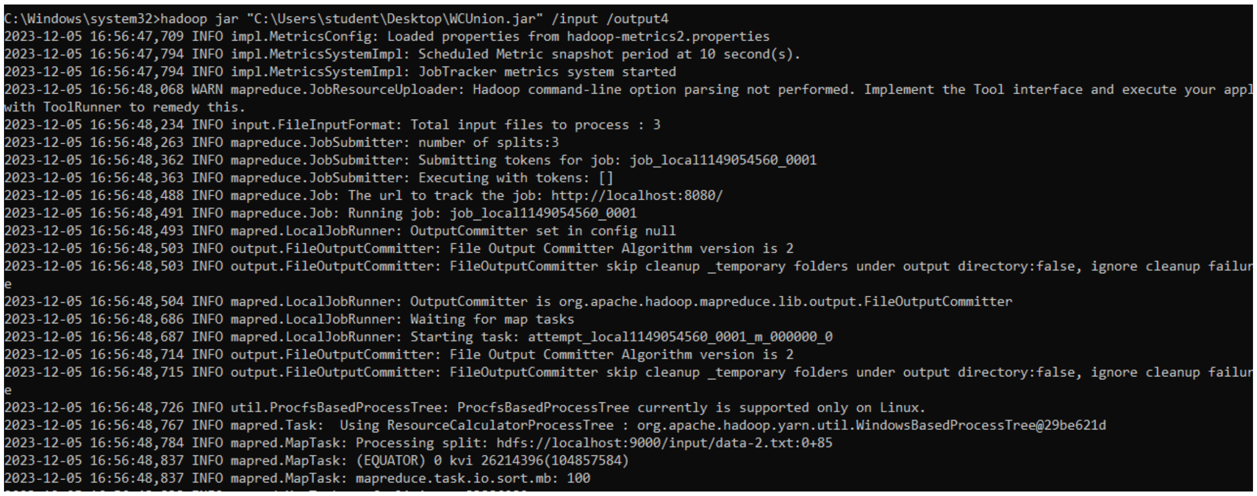
ANT script location: E:\Muskan_09\SPring Examples

Now we have a separate jar file



5. Run hadoop jar command as follows:

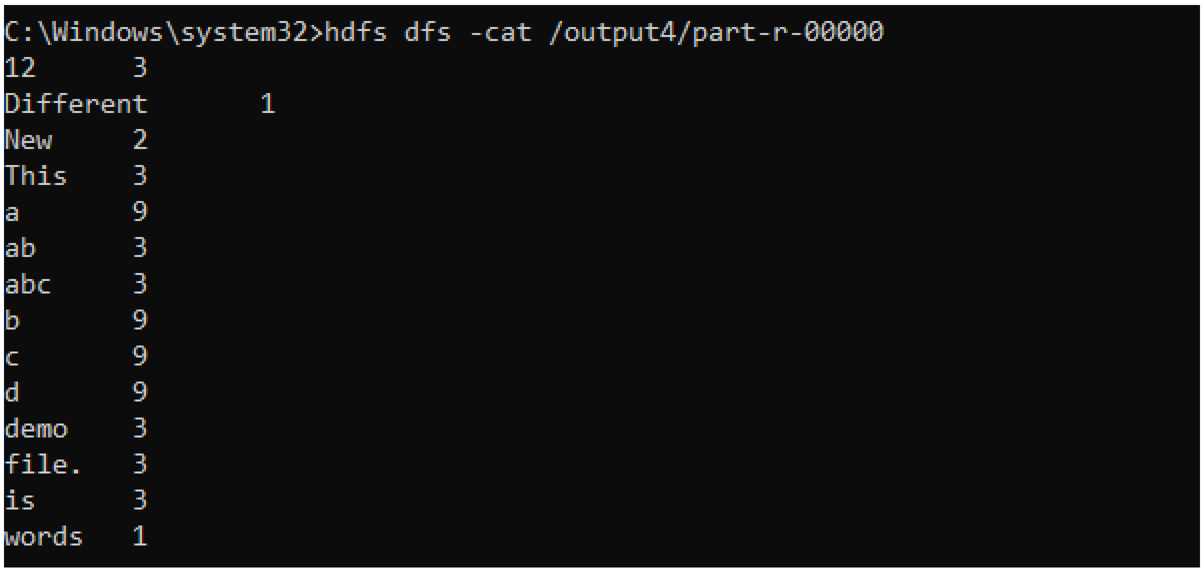
   Hadoop jar "C:\Users\student\Desktop\WCUnion.jar" /input /output4



```
C:\Windows\system32>hadoop jar "C:\Users\student\Desktop\WCUnion.jar" /input /output4
2023-12-05 16:56:47,709 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2023-12-05 16:56:47,794 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2023-12-05 16:56:47,794 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2023-12-05 16:56:48,068 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your appl
with ToolRunner to remedy this.
2023-12-05 16:56:48,234 INFO input.FileInputFormat: Total input files to process : 3
2023-12-05 16:56:48,263 INFO mapreduce.JobSubmitter: number of splits:3
2023-12-05 16:56:48,362 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1149054560_0001
2023-12-05 16:56:48,363 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-12-05 16:56:48,488 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2023-12-05 16:56:48,491 INFO mapreduce.Job: Running job: job_local1149054560_0001
2023-12-05 16:56:48,493 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2023-12-05 16:56:48,503 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-12-05 16:56:48,503 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failur
e
2023-12-05 16:56:48,504 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2023-12-05 16:56:48,686 INFO mapred.LocalJobRunner: Waiting for map tasks
2023-12-05 16:56:48,687 INFO mapred.LocalJobRunner: Starting task: attempt_local1149054560_0001_m_000000_0
2023-12-05 16:56:48,714 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-12-05 16:56:48,715 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failur
e
2023-12-05 16:56:48,726 INFO util.ProcfsBasedProcessTree: ProcfsBasedProcessTree currently is supported only on Linux.
2023-12-05 16:56:48,767 INFO mapred.Task:  Using ResourceCalculatorProcessTree : org.apache.hadoop.yarn.util.WindowsBasedProcessTree@29be621d
2023-12-05 16:56:48,784 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/input/data-2.txt:0+85
2023-12-05 16:56:48,837 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2023-12-05 16:56:48,837 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
```

6. Find output using cat command of hdfs as follows:

   hdfs dfs -cat /output4/part-r-00000



```
C:\Windows\system32>hdfs dfs -cat /output4/part-r-00000
12      3
Different       1
New     2
This    3
a       9
ab      3
abc     3
b       9
c       9
d       9
demo    3
file.   3
is      3
words   1
```

NOTE: change yarn to local if needed in mapred-site.xml