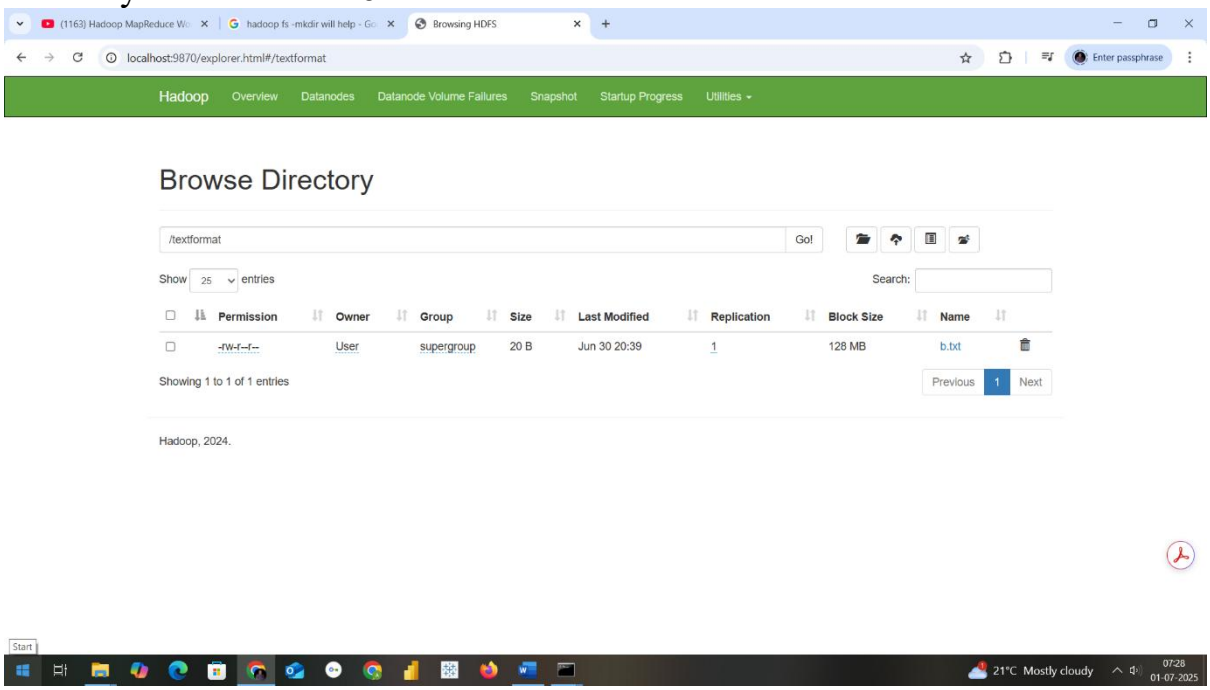
PROGRAM 8A : Word Count program using MapReduce WITH hadoop

Step1:goto Search button run cmd prompt as administrator
Step2: Initiate all required component of hadoop by command
start-all.cmd
Step3: Create a text file where we have folder running hadoop java program in my case I have created in Desktop
U can create anywhere as u wish
Step4: Create a Directory in HDFS console using command
hadoop fs -mkdir /textformat
Where textform is ur input directory
Step5: We have to add text into a directory<textformat> in my case b.txt is in desktop folder copy the path and type command
hadoop fs -put C:\Users\User\Desktop\b.txt /textformat
After completion of above command u be able to see b.txt in /textformat directory of hdfs console

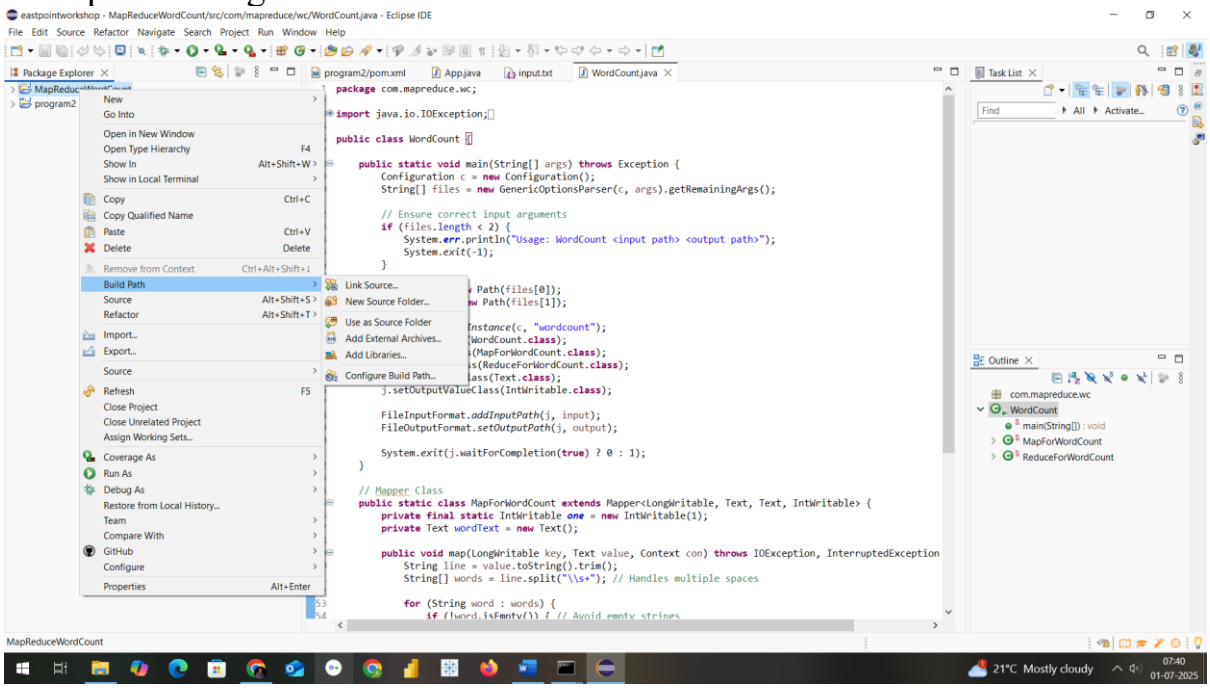


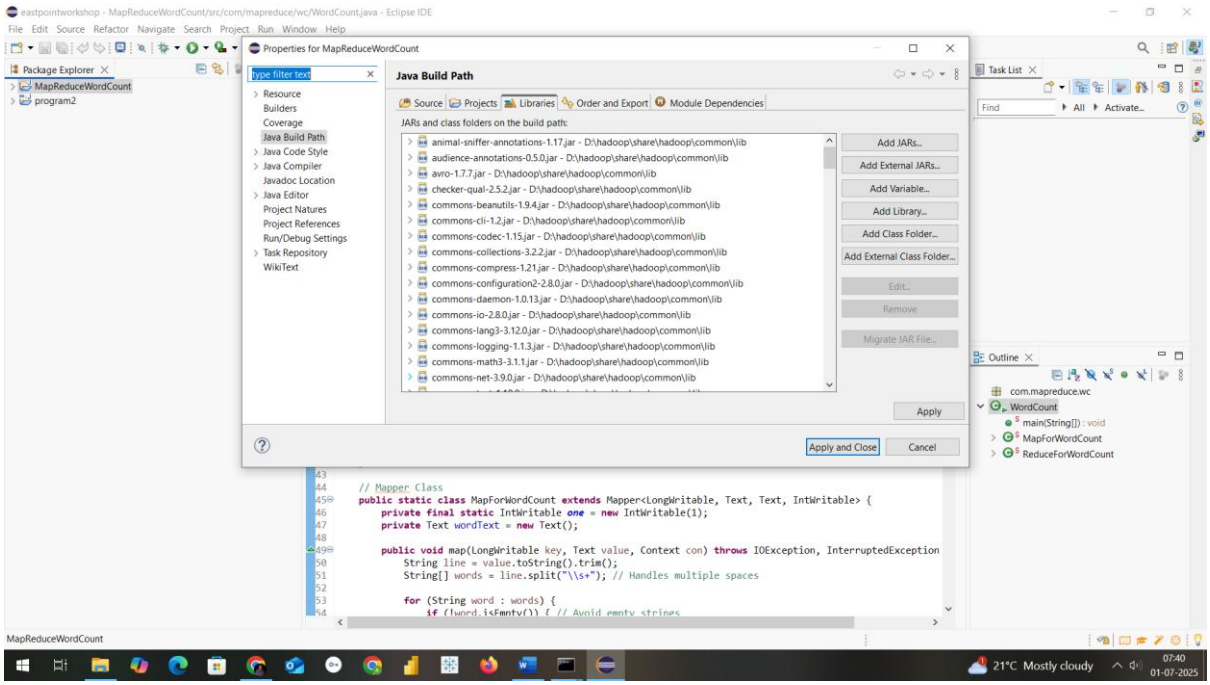
In cmd prmpt if u want to check type the command as
hadoop fs -ls /textformat

Step6: Now we need a jar file to execute mapreduce program fr wordcount
So open Eclipse
Create a java project with name MapReduceWordCount
Select execution environment as JavaSE-1.8
And click on next&finish

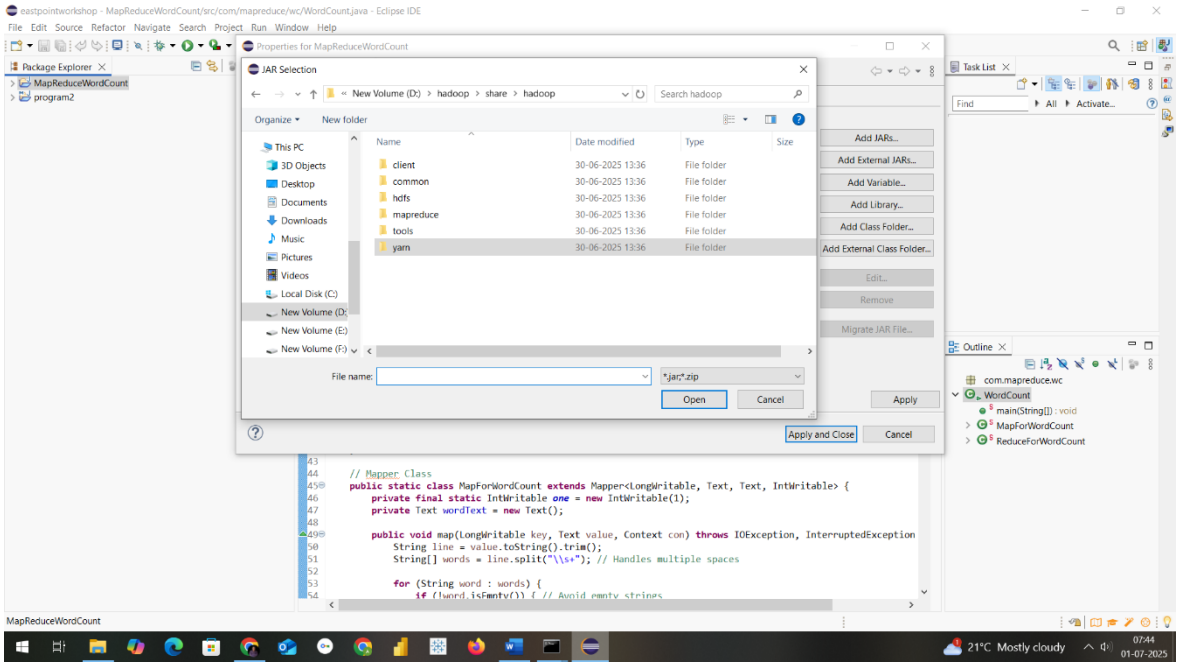
Step7: Right Click on Projectname---→ Click on New---→create a package

with name com.mapreduce.wc and click on Finish
Step8: Add required libraries to support hadoop by navigating as in below
Right Click on Project-------→Right Click on BuildPath-------→ Configure Buildpath Then goto Libraries as in second screen below





Step9: Add necessary External jar file from D:\hadoop\share\hadoop
Like clients,common,hdfs,mapreduce & yarn to support packages for hadoop



Step10: Create aclass within package com.mapreduce.wc with name WordCount and paste this code

```
package com.mapreduce.wc;

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
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;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {

    public static void main(String[] args) throws Exception {
        Configuration c = new Configuration();
        String[] files = new GenericOptionsParser(c, args).getRemainingArgs();

        // Ensure correct input arguments
        if (files.length < 2) {
            System.err.println("Usage: WordCount <input path> <output path>");
```

```
            System.exit(-1);
        }

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

        Job j = Job.getInstance(c, "wordcount");
        j.setJarByClass(WordCount.class);
        j.setMapperClass(MapForWordCount.class);
        j.setReducerClass(ReduceForWordCount.class);
        j.setOutputKeyClass(Text.class);
        j.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(j, input);
        FileOutputFormat.setOutputPath(j, output);

        System.exit(j.waitForCompletion(true) ? 0 : 1);
    }

    // Mapper Class
    public static class MapForWordCount extends Mapper<LongWritable, Text,
Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text wordText = new Text();

        public void map(LongWritable key, Text value, Context con) throws
IOException, InterruptedException {
            String line = value.toString().trim();
            String[] words = line.split("\\s+"); // Handles multiple spaces

            for (String word : words) {
                if (!word.isEmpty()) { // Avoid empty strings
                    wordText.set(word.trim().toUpperCase());
                    con.write(wordText, one);
                }
            }
        }
    }

    // Reducer Class
    public static class ReduceForWordCount extends Reducer<Text, IntWritable,
Text, IntWritable> {
        public void reduce(Text word, Iterable<IntWritable> values, Context con)
```

```
throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable value : values) {
           sum += value.get();
        }
        con.write(word, new IntWritable(sum));
     }
  }
}
```

PROGRAM 8B:Word Program Count Procedure using Apache Spark

Step 1: Prerequisites

Ensure you have the following installed:

- Java JDK (8 or 11)
- Eclipse IDE (preferably Eclipse IDE for Java Developers)
- Apache Maven

Step 2: Create Maven Project in Eclipse

1. Open Eclipse → File → New → Maven Project → Next.
2. Select maven-archetype-quickstart, click Next.
3. Provide the following:
    - Group Id: com.example.spark
    - Artifact Id: spark-wordcount
    - Click Finish.

Step 3: Add Spark Dependencies in pom.xml

Add this to your pom.xml:

```
<dependencies>
<dependency>
      <groupId>org.apache.spark</groupId>
      <artifactId>spark-core_2.12</artifactId>
      <version>3.3.2</version>
  </dependency>

  <!-- Apache Spark SQL (Optional) -->
  <dependency>
      <groupId>org.apache.spark</groupId>
      <artifactId>spark-sql_2.12</artifactId>
```

```
      <version>3.3.2</version>
    </dependency>

    <!-- Logging (to avoid SLF4J warnings) -->
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-simple</artifactId>
      <version>1.7.30</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <!-- Plugin to build a fat JAR -->
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-shade-plugin</artifactId>
        <version>3.2.4</version>
        <executions>
          <execution>
            <phase>package</phase>
            <goals><goal>shade</goal></goals>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
```

Step 4: Create WordCount Program
Create a class WordCount.java under:
src/main/java/com/example/spark/WordCount.java

```
package com.example.spark.spark_wordcount;
import org.apache.spark.api.java.*;
import org.apache.spark.SparkConf;
import scala.Tuple2;
import java.util.Arrays;


public class App {
    public static void main(String[] args) {
        // Set Spark Configuration
        SparkConf conf = new
SparkConf().setAppName("WordCount").setMaster("local[*]");
```

```
        JavaSparkContext sc = new JavaSparkContext(conf);

        // Load input file
        JavaRDD<String> input = sc.textFile("D:/hadoopprograms/input.txt");  // Replace with your file path

        // FlatMap each line to words
        JavaRDD<String> words = input.flatMap(line -> Arrays.asList(line.split(" ")).iterator());

        // Map each word to a pair (word, 1)
        JavaPairRDD<String, Integer> wordPairs = words.mapToPair(word -> new Tuple2<>(word, 1));

        // Reduce by key (word)
        JavaPairRDD<String, Integer> wordCounts = wordPairs.reduceByKey(Integer::sum);

        // Print output
        wordCounts.foreach(result -> System.out.println(result._1() + ": " + result._2()));

        sc.close();
    }
}
```

Step 5: Input File
Create a input.txt file in folder where u r saving Hadoop programs in mine its in D:\hadoopprograms

Example content:
hello world
hello spark
spark is fast
Step 6: Run the Program

1. Right-click WordCount.java → Run As → Java Application.
2. Console should print:

hello: 2
world: 1
spark: 2
is: 1

fast: 1

RESULT