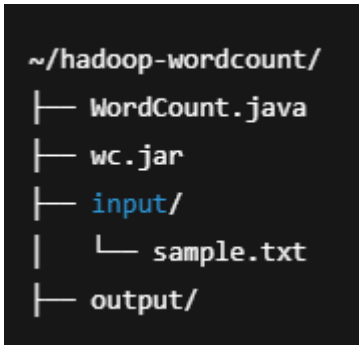
Experiment 11

Write a code in JAVA for a simple WordCount application that counts the number of occurrences of each word in a given input set using the Hadoop MapReduce framework on local-standalone set-up

Step 1: Directory Structure:



Step 2: Wordcount.java code

```
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 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()) {
            String clean = itr.nextToken().replaceAll("[^a-zA-Z]", "").toLowerCase();
            if (!clean.isEmpty()) {
                word.set(clean);
                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);
    }
}
```

Step 3: Create an input file:

```
mkdir input
echo "Hadoop is an open source framework. Hadoop is used for Big Data processing." > input/sample.txt
```

Step 4: Compile Java File:

export HADOOP_HOME=~/hadoop-3.3.6

export PATH=$HADOOP_HOME/bin:$PATH

javac -classpath `${HADOOP_HOME}/share/hadoop/common/hadoop-common-*.jar`:`${HADOOP_HOME}/share/hadoop/mapreduce/hadoop-mapreduce-client-core-*.jar` -d . WordCount.java

jar cf wc.jar WordCount*.class

Step 5: Run job:

hadoop jar wc.jar WordCount input output

Step 6: View output:

cat output/part-r-00000

```
an      1
big     1
data    1
for     1
framework       1
hadoop  2
is      2
open    1
processing      1
source  1
used    1
```

```
$ mkdir input
$ echo "Hadoop is an open source framework. Hadoop is used for Big Data processing." > input/samp

$ javac -classpath `~/hadoop-3.3.6/share/hadoop/common/hadoop-common-3.3.6.jar`:`~/hadoop-3.3.6/s
$ jar cf wc.jar WordCount*.class

$ hadoop jar wc.jar WordCount input output
...

$ cat output/part-r-00000
an      1
big     1
data    1
for     1
framework       1
hadoop  2
is      2
open    1
processing      1
source  1
used    1
```

Experiment 12:

Locate dataset (e.g., sample_weather.txt) for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed

Step 1: Sample Dataset

Date	TempC	DewPointC	WindKPH
2025-04-20	22	14	18
2025-04-19	25	16	20
2025-04-18	21	12	15
2025-04-17	24	15	22

Step 2: Place it inside

```
mkdir weather_input
cp sample_weather.txt weather_input/
```

Step 3: Java Hadoop Program

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WeatherAverage {
   public static class WeatherMapper extends Mapper<LongWritable, Text, Text, Text> {
      public void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {
         String line = value.toString();
         if (line.startsWith("Date")) return; // Skip header
         String[] parts = line.split("\\s+");
```

```
        if (parts.length == 4) {
            String temp = parts[1];
            String dew = parts[2];
            String wind = parts[3];
            context.write(new Text("weather"), new Text(temp + "," + dew + "," + wind));
        }
    }
}

public static class AverageReducer extends Reducer<Text, Text, Text, Text> {
    public void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException {
        int count = 0;
        double sumTemp = 0, sumDew = 0, sumWind = 0;

        for (Text val : values) {
            String[] fields = val.toString().split(",");
            if (fields.length == 3) {
                sumTemp += Double.parseDouble(fields[0]);
                sumDew += Double.parseDouble(fields[1]);
                sumWind += Double.parseDouble(fields[2]);
                count++;
            }
        }

        if (count > 0) {
            double avgTemp = sumTemp / count;
            double avgDew = sumDew / count;
            double avgWind = sumWind / count;
```

```
    context.write(new Text("Average Temperature (C), Dew Point (C), Wind Speed (KPH):"),
                  new Text(String.format("%.2f, %.2f, %.2f", avgTemp, avgDew, avgWind)));
            }
        }
    }
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Weather Averages");
        job.setJarByClass(WeatherAverage.class);
        job.setMapperClass(WeatherMapper.class);
        job.setReducerClass(AverageReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Step 4: Compile and Run

javac -classpath ~/hadoop-3.3.6/share/hadoop/common/hadoop-common-3.3.6.jar:~/hadoop-3.3.6/share/hadoop/mapreduce/hadoop-mapreduce-client-core-3.3.6.jar -d . WeatherAverage.java

jar cf weather_avg.jar WeatherAverage*.class

hadoop jar weather_avg.jar WeatherAverage weather_input weather_output

Step 5: Output

cat weather_output/part-r-00000

```
Average Temperature (C), Dew Point (C), Wind Speed (KPH):    23.00, 14.25, 18.75
```

Experiment 13:

Write a simple program in SCALA using Apache Spark framework

Before running this Scala program, ensure the following are installed:

- Java 8 or later (sudo apt install openjdk-11-jdk)
- Scala (sudo apt install scala)
- Apache Spark (download and extract from https://spark.apache.org/downloads.html)

Step 1: Create a Sample Input File

```
mkdir wordcount_input
echo "Apache Spark is fast. Spark is fun to use." > wordcount_input/input.txt
```

Step 2: Scala Program

```
import org.apache.spark.sql.SparkSession
object SimpleWordCount {
  def main(args: Array[String]): Unit = {
    // Create Spark session
    val spark = SparkSession.builder
      .appName("Simple Word Count")
      .master("local[*]") // run locally using all cores
      .getOrCreate()
    // Read input file
    val input = spark.sparkContext.textFile("wordcount_input/input.txt")

    // Perform Word Count
    val counts = input
      .flatMap(line => line.split(" "))
      .map(word => (word.toLowerCase().replaceAll("""[\p{Punct}]""", ""), 1))
      .reduceByKey(_ + _)
```

```
    // Save result
    counts.saveAsTextFile("wordcount_output")

    // Stop Spark
    spark.stop()
  }
}
```

Step 3: Compile and Run the Program

```
scalac -classpath $SPARK_HOME/jars/* SimpleWordCount.scala
```

```
$SPARK_HOME/bin/spark-submit \
  --class SimpleWordCount \
  --master local[*] \
  SimpleWordCount.jar
```

```
$SPARK_HOME/bin/spark-submit --class SimpleWordCount --master local[*] SimpleWordCount.scala
```

Step 4: Output

```
cat wordcount_output/part-*
```

```
(spark,2)
(is,2)
(fast,1)
(apache,1)
(fun,1)
(to,1)
(use,1)
```