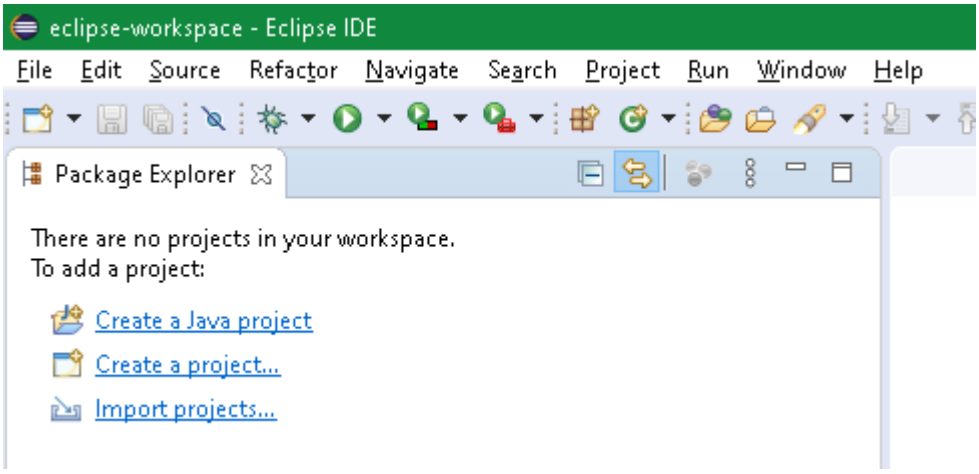
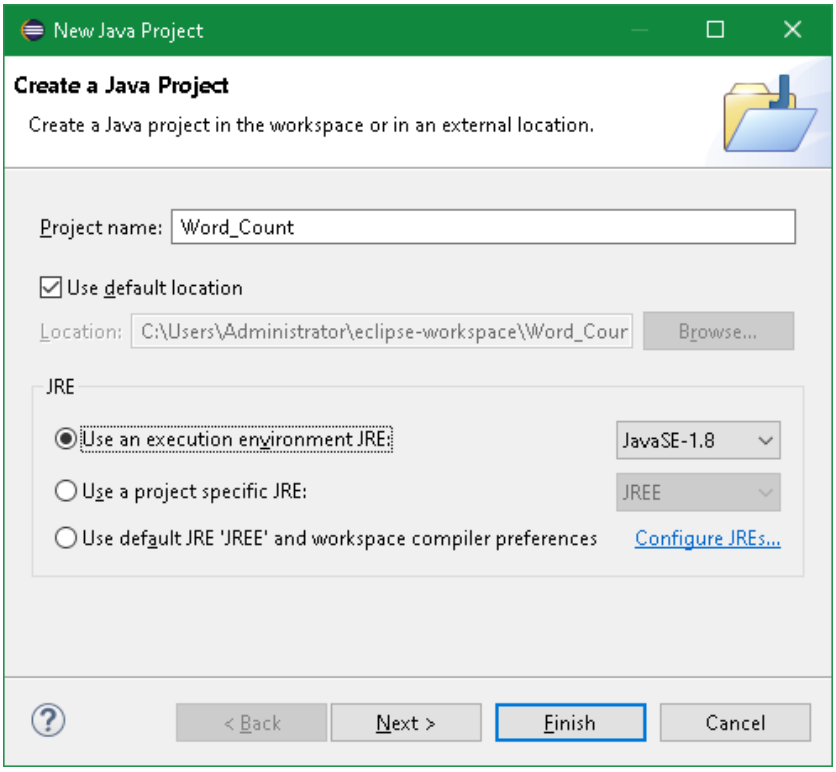
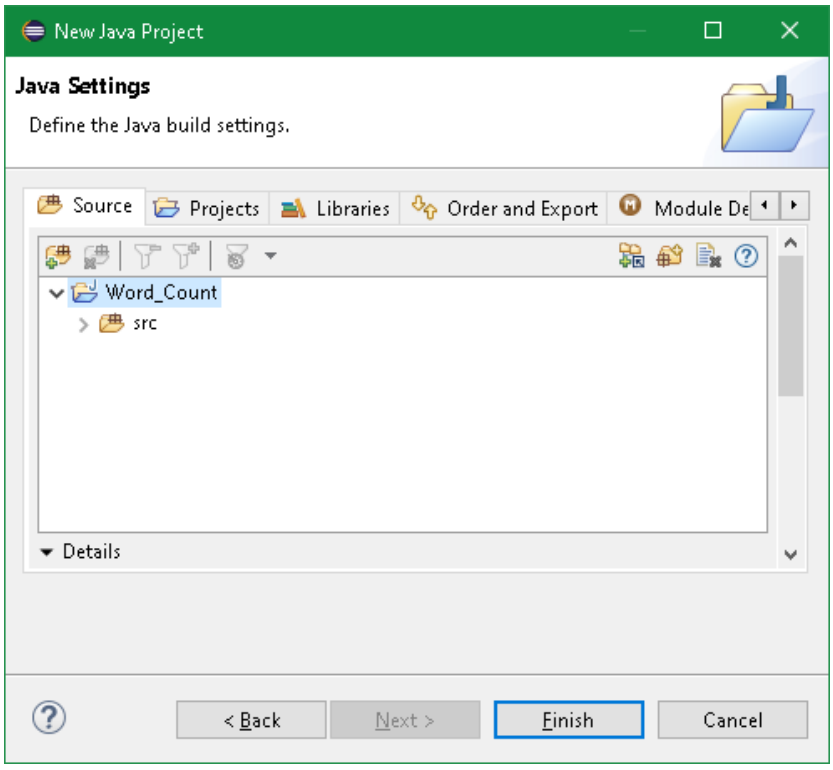
Running Hadoop MapReduce Application using Eclipse IDE

Open your Eclipse and create a new Java program.



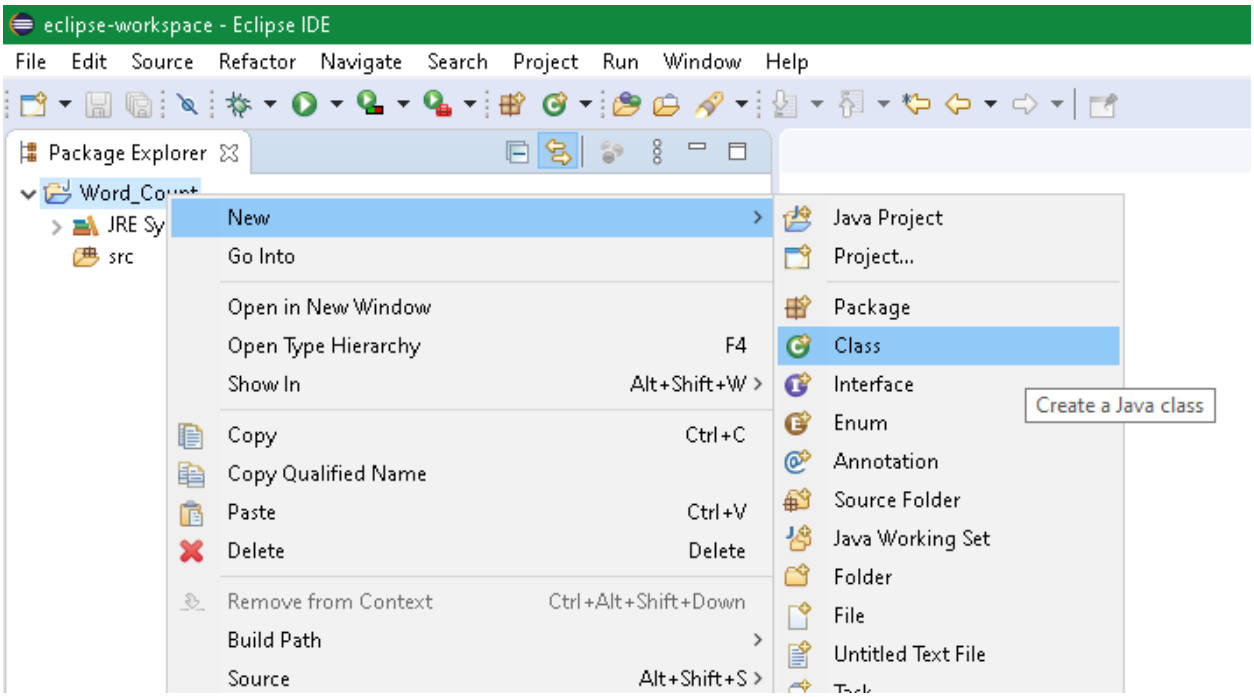
Give the project name Word_Count and press Next



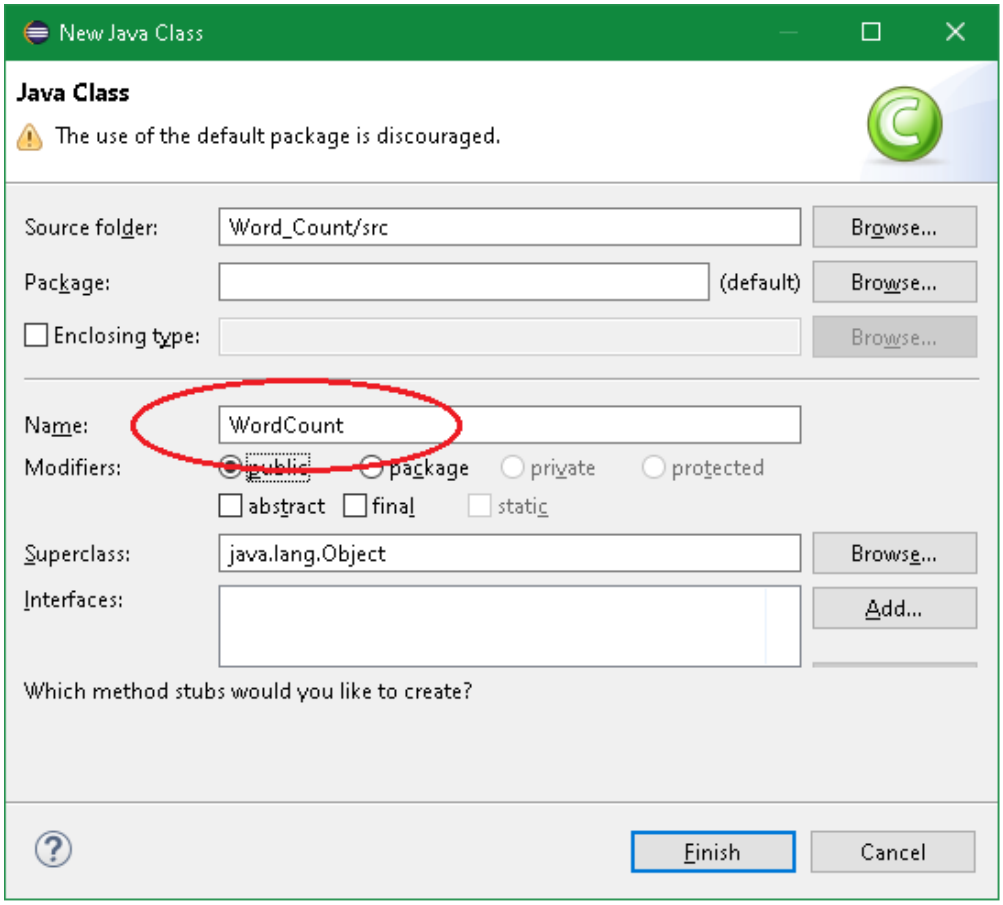


Click on Finish, to create project.

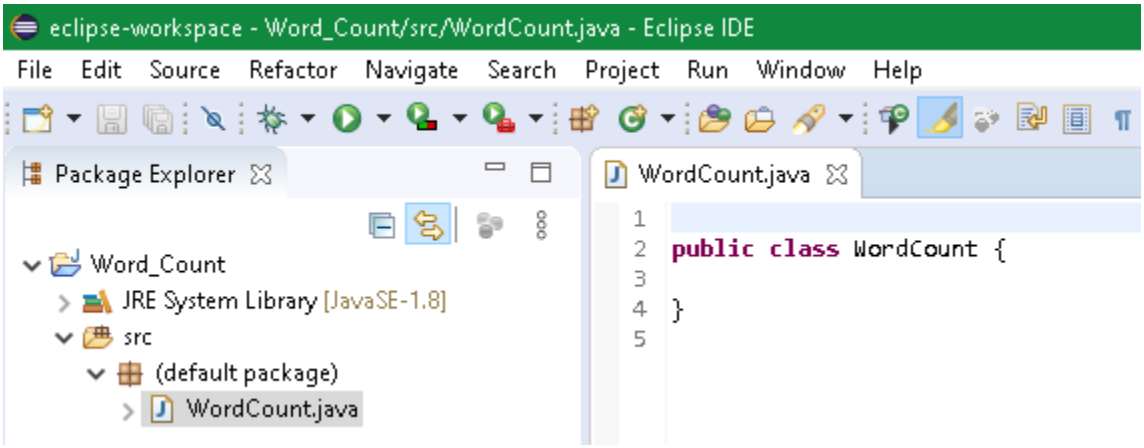
Create a new Java class by right click on project New → Class



Give the class name as **WordCount**. Inside the **src,** a file with name **WordCount.java** has been created.



Click on the file and write the MapReduce code for the word count program.



Write the MapReduce code for the word count program.

```
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()) {
        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);
  }
}
```

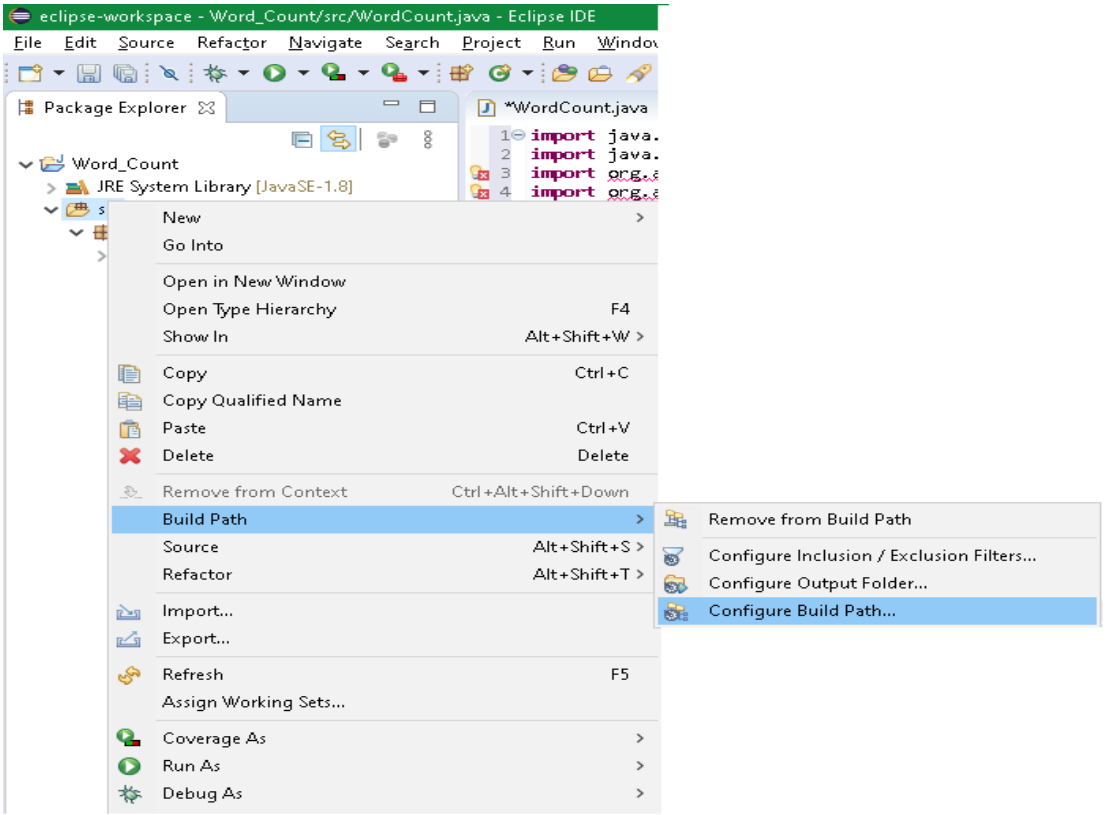
After copying the code save the file. Now you need to add a few dependency files for running this program in Windows.

First, we need to add the jar files that are present in **hadoop-2.6.0/share/hadoop** directory.

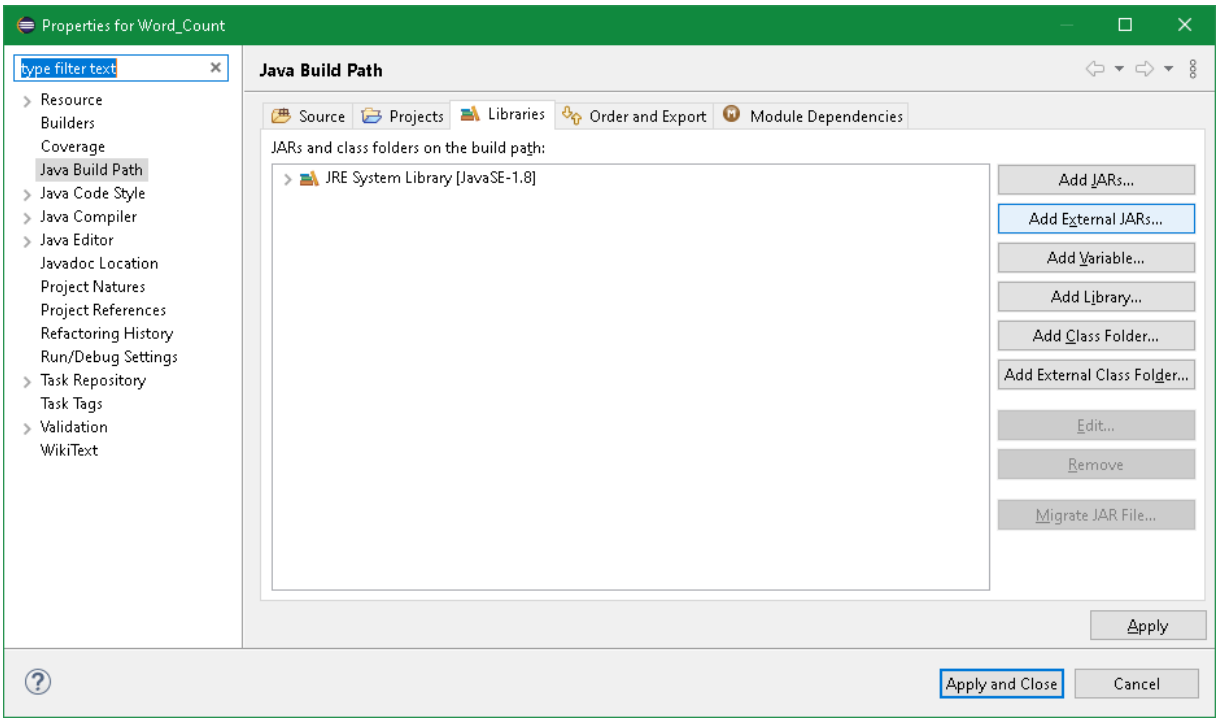
(Alternatively, you can download the .JAR files from the below link)

https://github.com/jijim/HADOOP-Windows10/blob/master/Eclipse_JAR_Imports_Final.rar

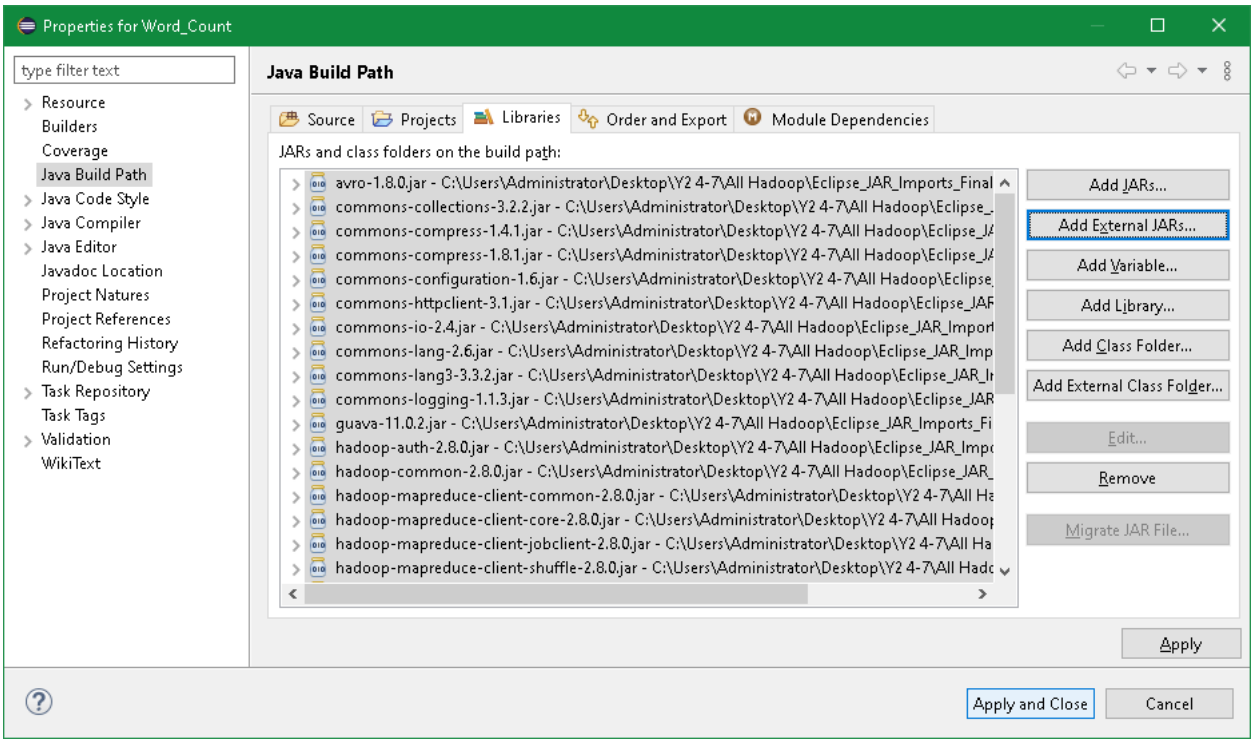
Righ click on src–>Build path–>Configure build path as shown in the below.



Click on Libraries and select **Add External JAR** Button as per the below screen shot



Add all the JAR files and click on Apply and Close button

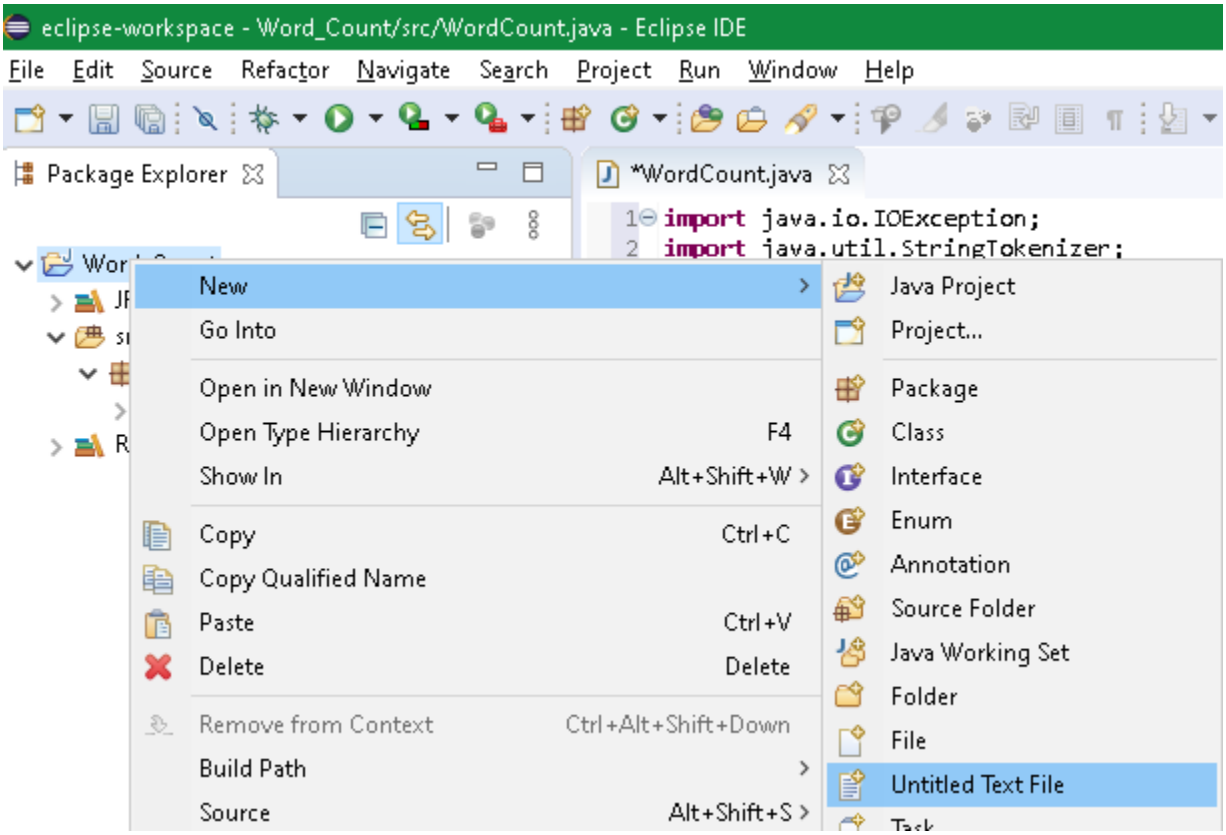


That’s it all the set up required for running your Hadoop application in Windows.

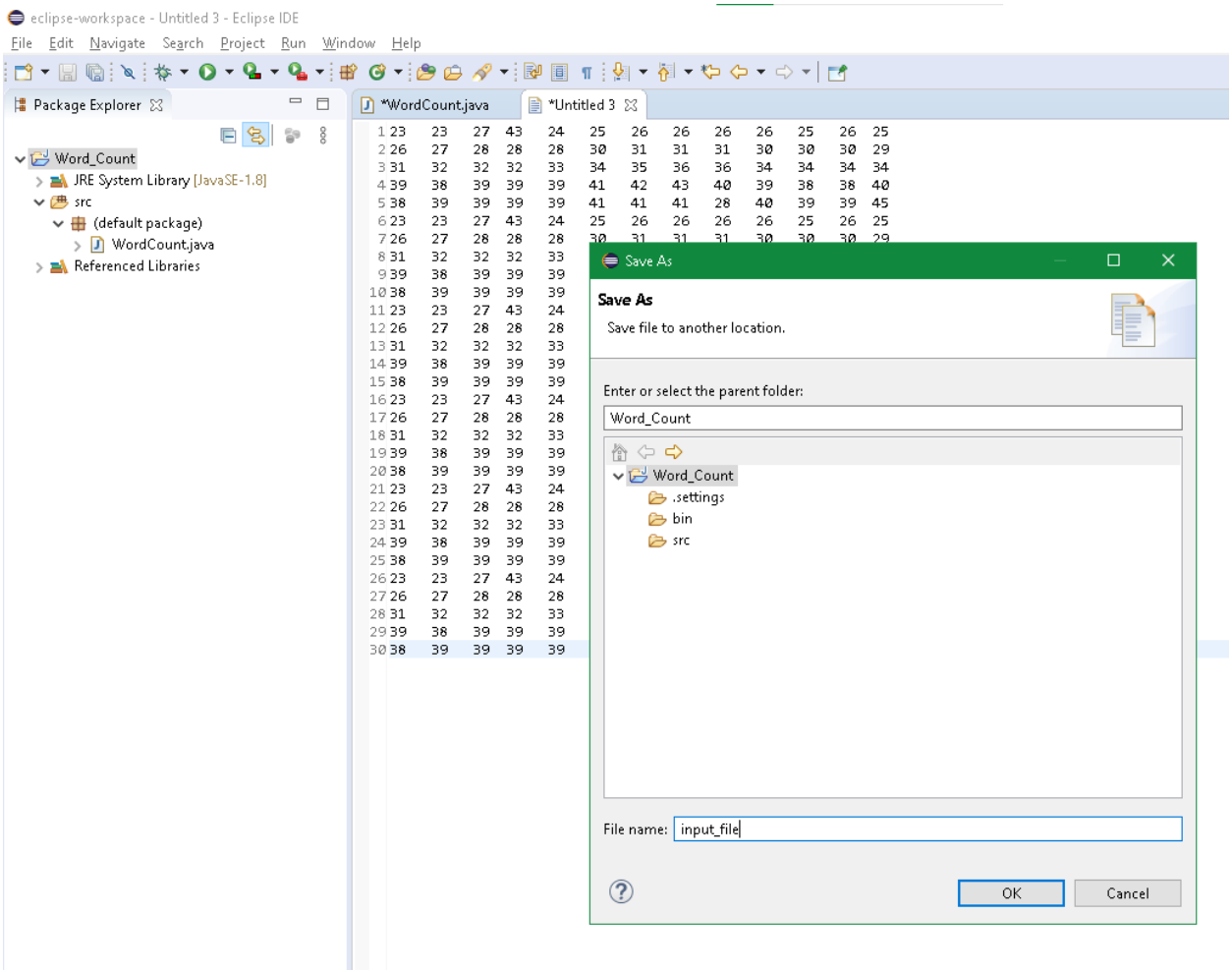
Make sure that your input file is ready.

Lets create an input Text file in the project directory as input-file

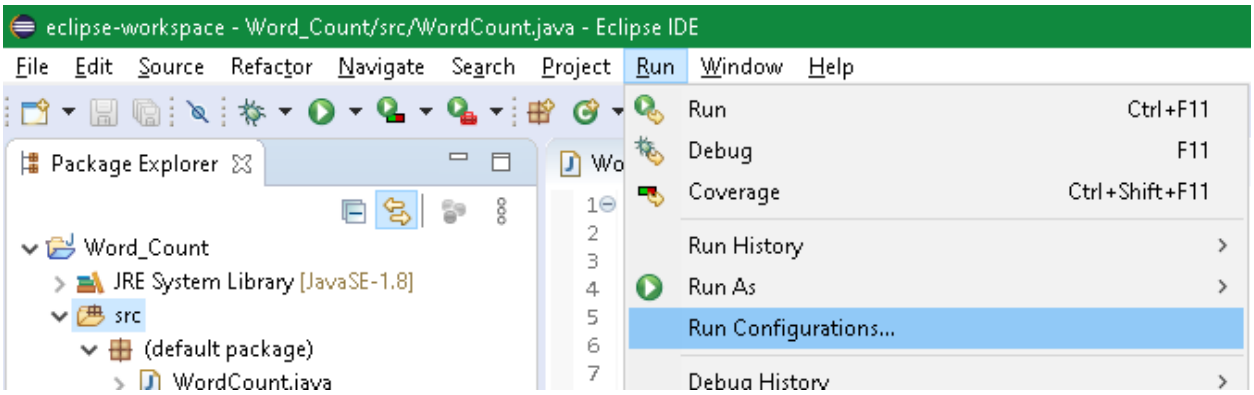
Right Button click on Project → Add Text File



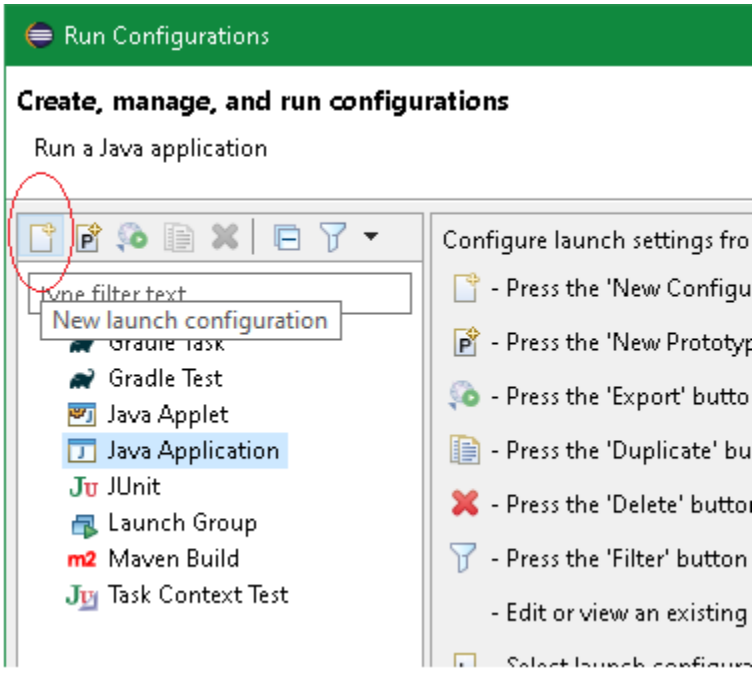
Enter the file content and save as input_file as per below screen shot.



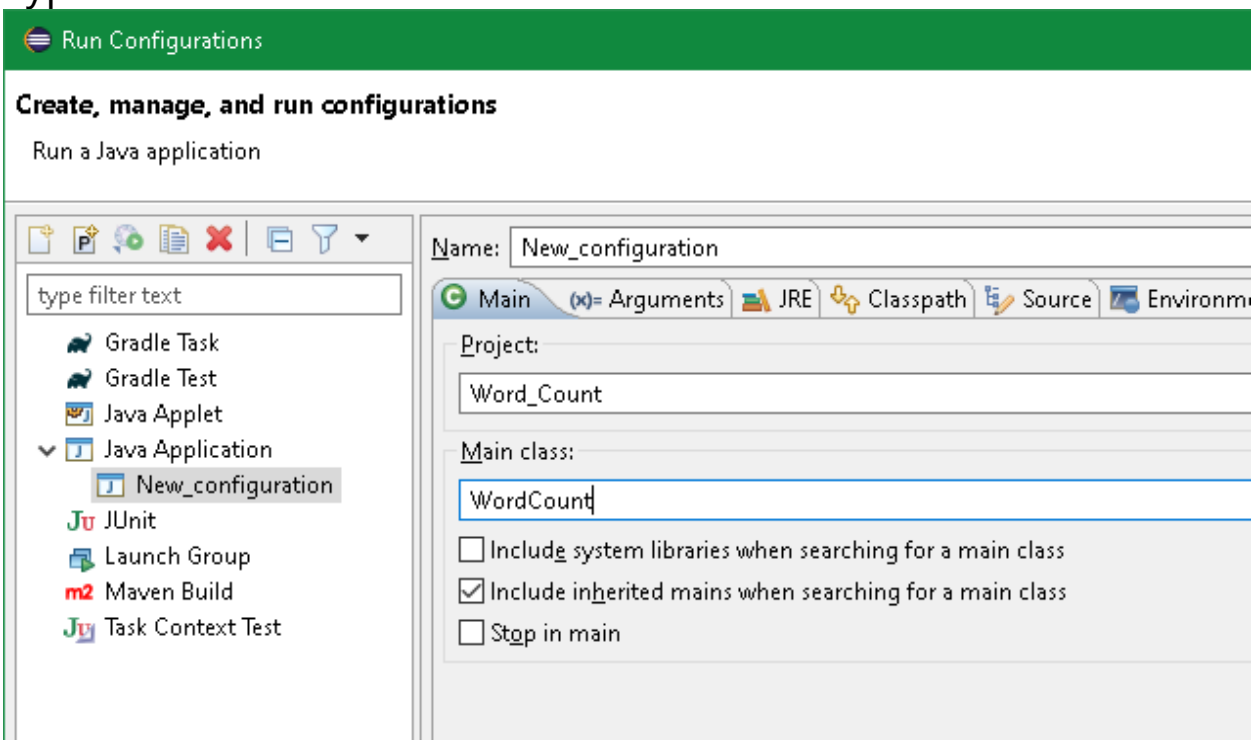
For giving the input and output file paths, **click on Run menu–>Run configurations** as shown in the below screen shot.



Click on Java Application and select New Launch Configurations from toolbar.

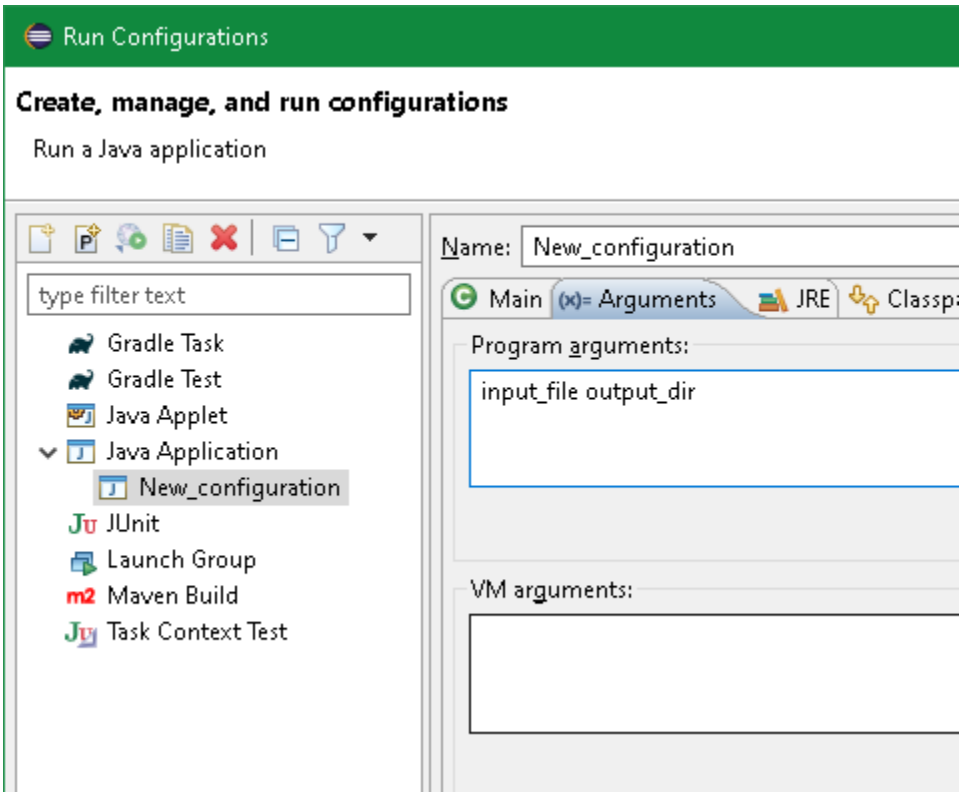


Type main class name WordCount

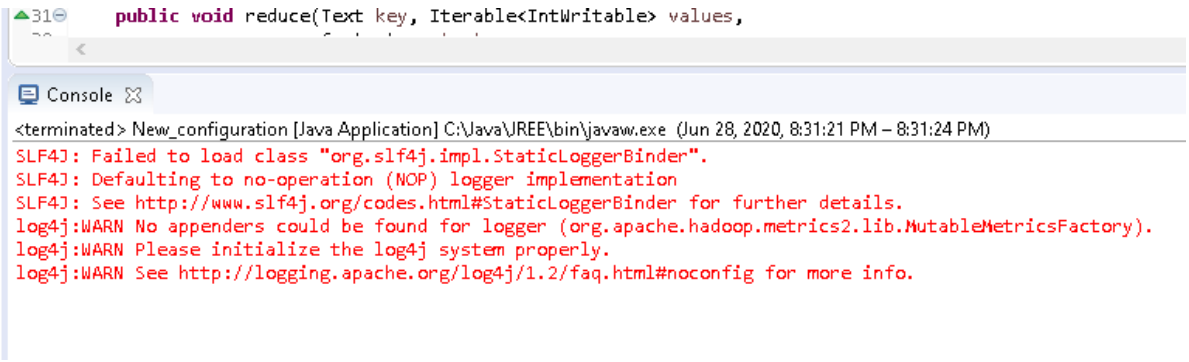


Click on Arguments Tab and write the name of input file and output directory and click on Run button

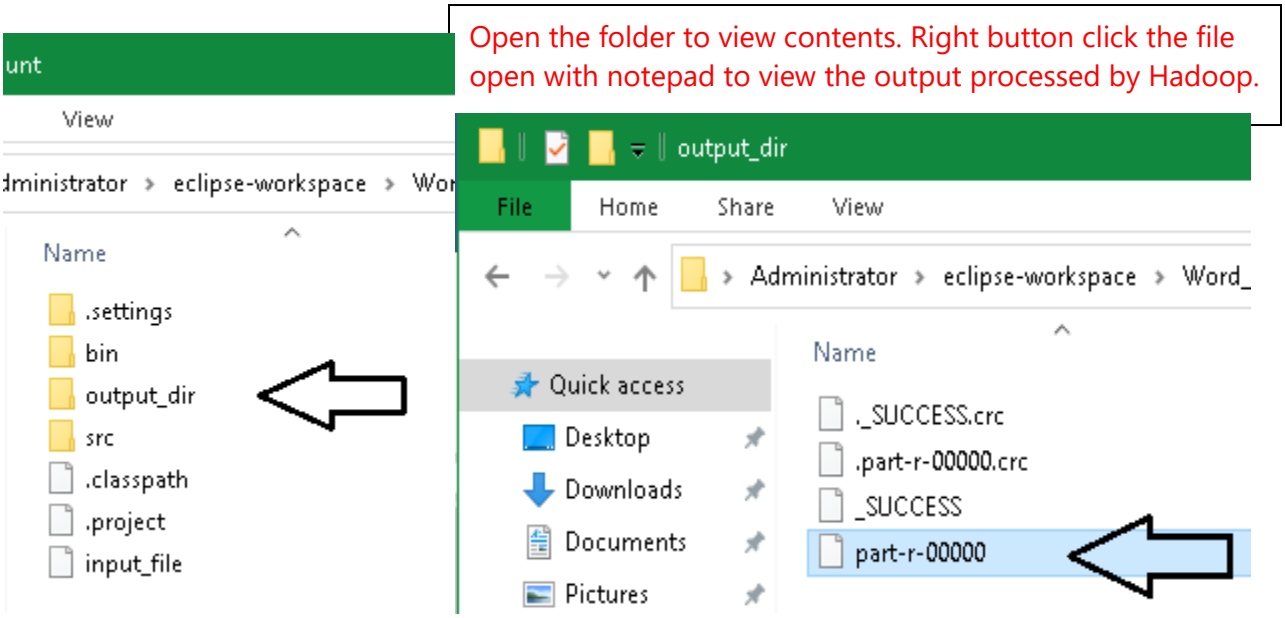
input_file output_dir



After the Run, see the warnings in the console window. If it appears same like below, just ignore. (If there is error, please fix it.)



After the project is successfully compiled, please verify the output folder created. (output_dir).



You can check the output file in the project directory and you can see the output in the **part-r-00000** file as shown in the above screen shot.

We have successfully run Hadoop application in Windows.