UNIVERSITY PARTNER

UNIVERSITY OF WOLVERHAMPTON

HERALD COLLEGE KATHMANDU

DISTRIBUTE AND CLOUD SYSTEMS PROGRAMMING (5CS022)

WEEK 9 WORKSHOP
THE APACHE SHARK FRAMEWORK

Student Id : 2065697
Student Name : Dhiraj Kumar Sah Kanu
Group : L5CG12

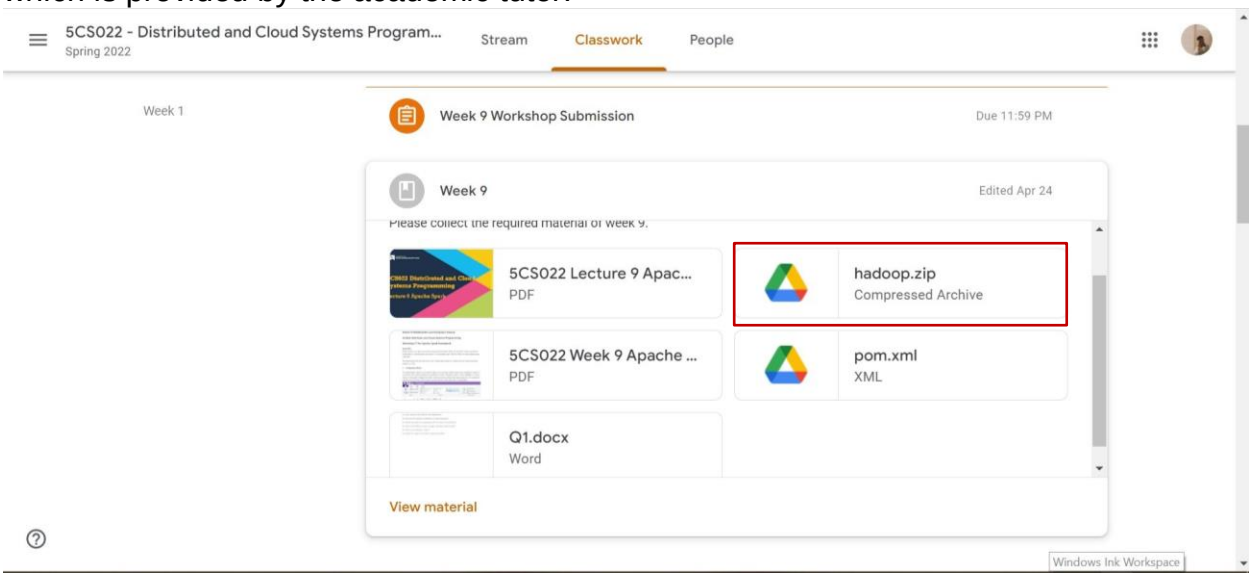
Submitted on : May 06, 2022

Overview

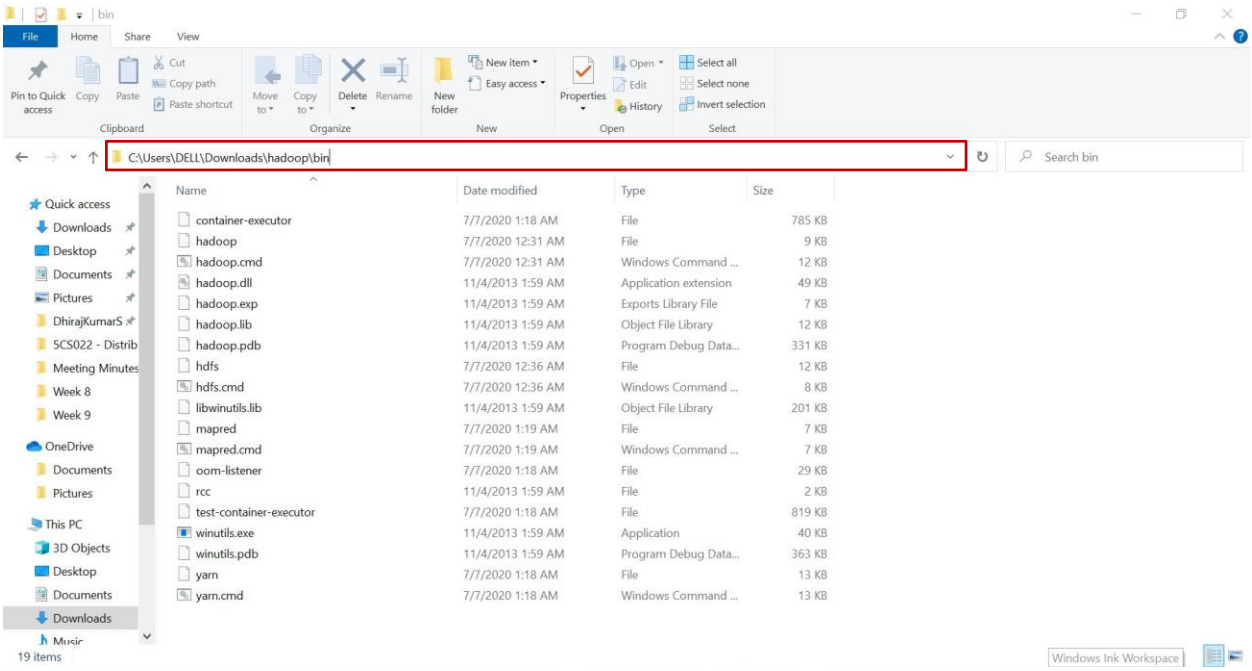
This assignment will help you how to create a small Spark project in Eclipse and run a run a word and letter counting application in Java. Here, you will set the environment variable to instruct Apache Spark where to find Hadoop. You will start by making a new project in Eclipse. You will edit and replace the code and will run the file. Then you will create the WordCount Program, run the program and see the resultant output. Then you will be creating a Spark program to count letters instead of words.

Step 1: Downloading Hadoop

To begin, download the file 'Hadoop.zip' from your Week 9 folder in Google Classroom which is provided by the academic tutor.

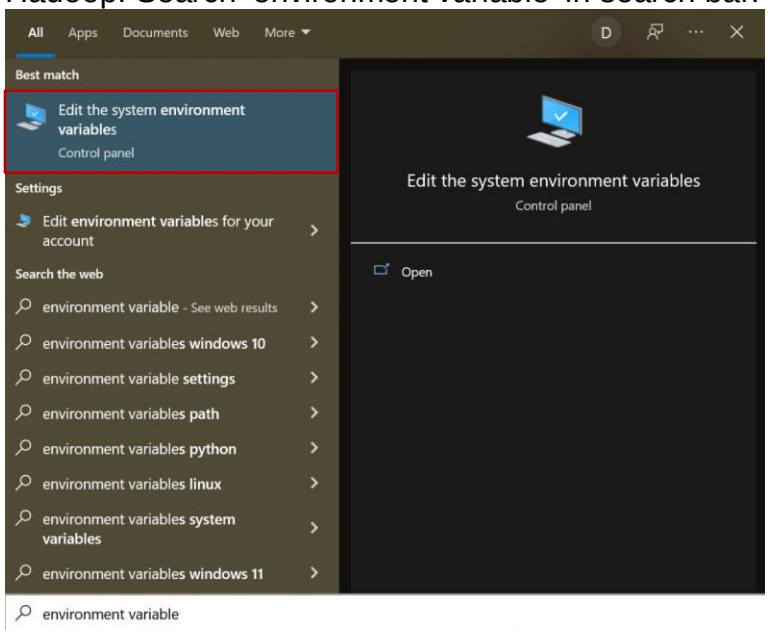


I have downloaded the file and have saved the hadoop file in Downloads, and copy the file loaction.

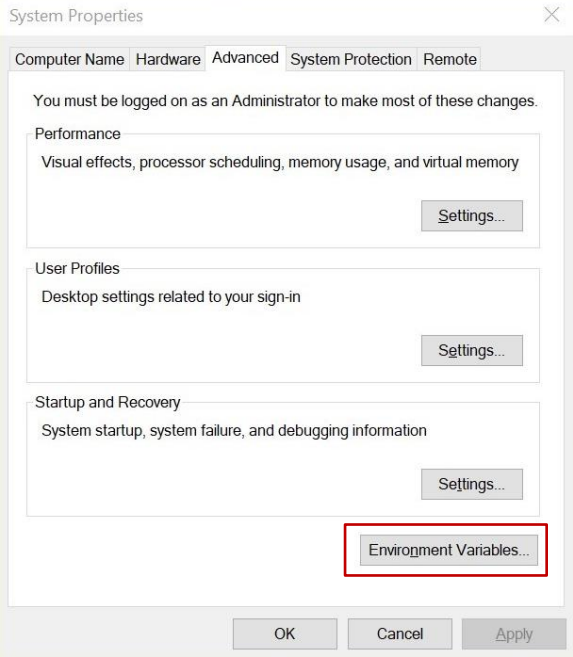


Step 2: Configuring Hadoop

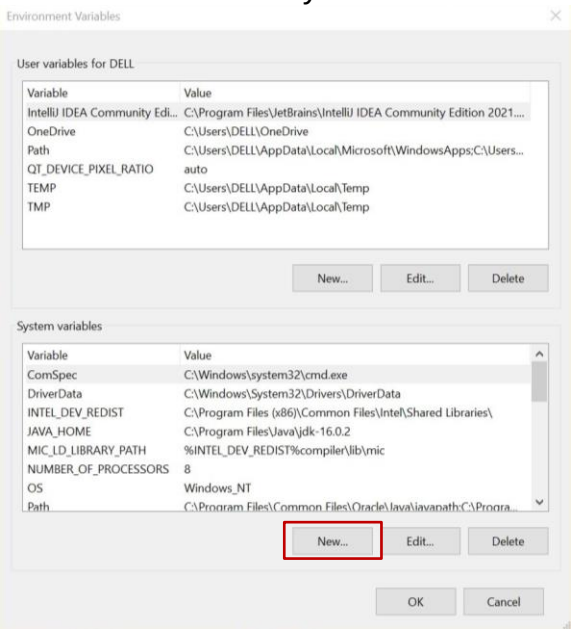
Now you need to set the environment variable to instruct Apache Spark where to find Hadoop. Search ‘environment variable’ in search bar. Click the first option.



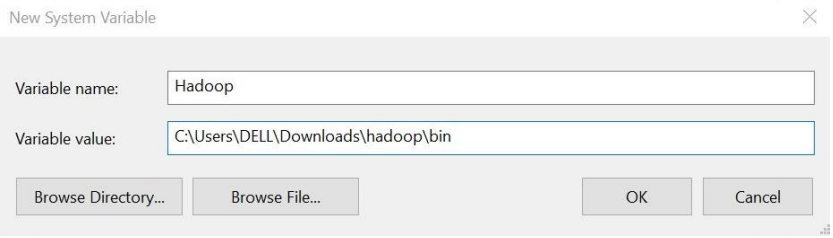
You will see the box of system properties, here click on ‘Environment Variables’.



Click on New under system variables section.

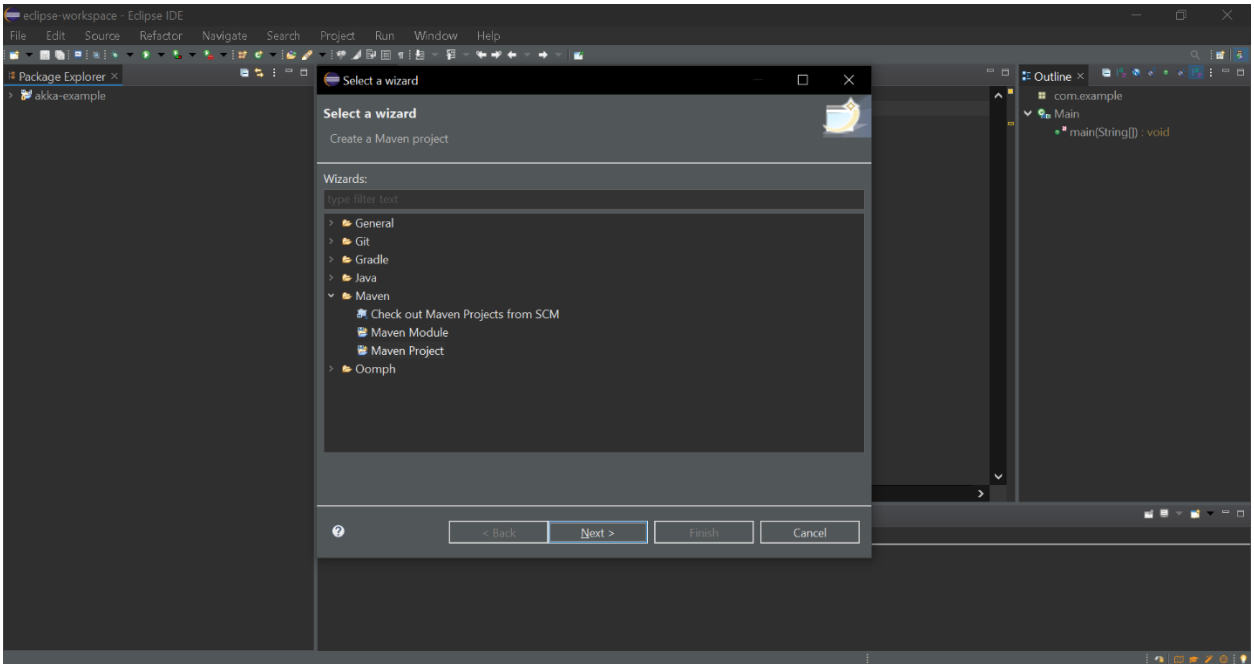


Next, you must configure the Environment Variable for this directory as follows, and paste the directory which you have copied previously.

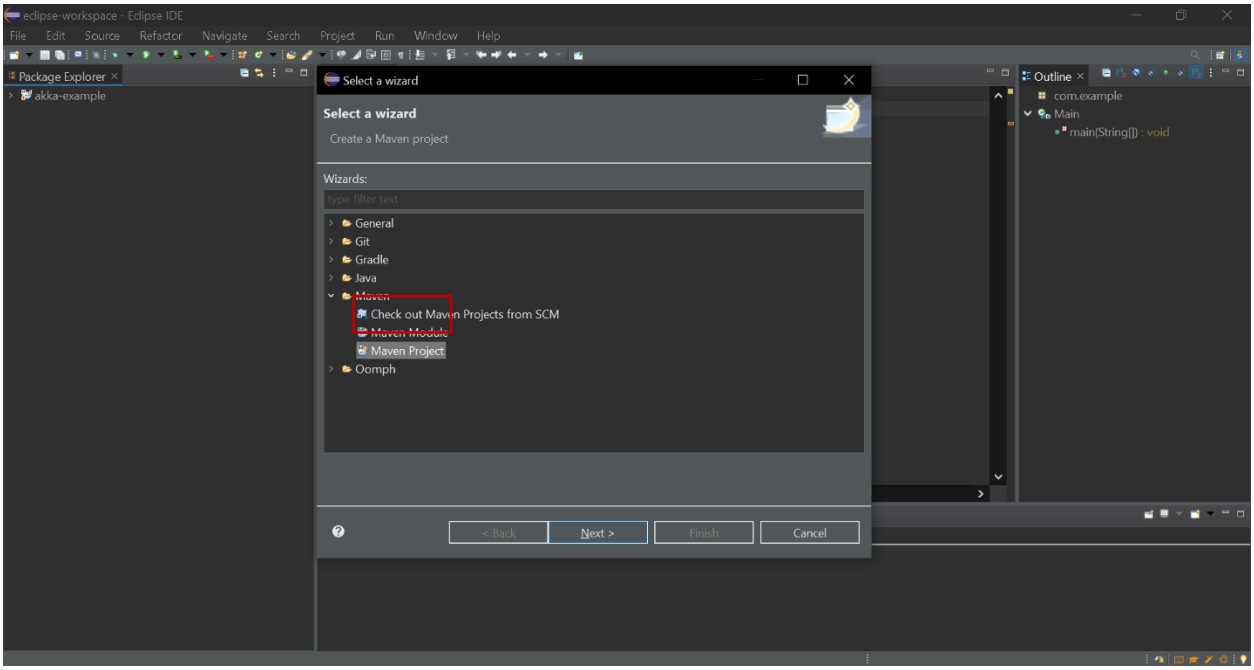


Step 3: Creating project in Eclipse

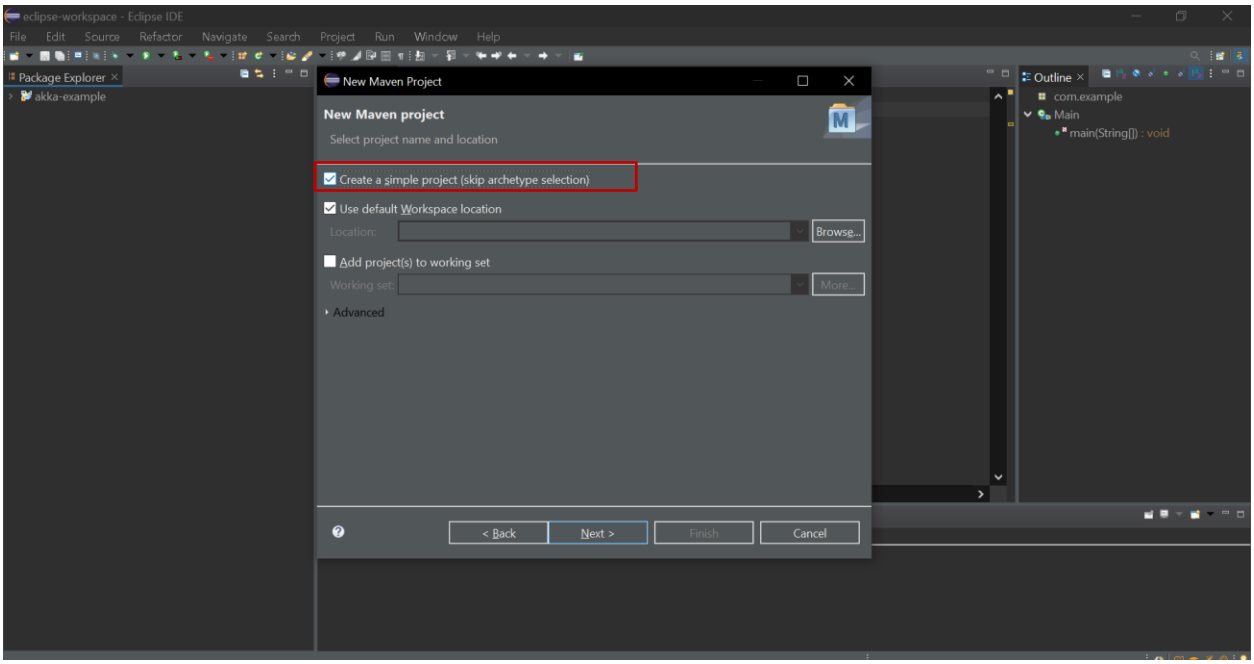
Here you have to start by making a new project in Eclipse.
Launch Eclipse and then select New Project.



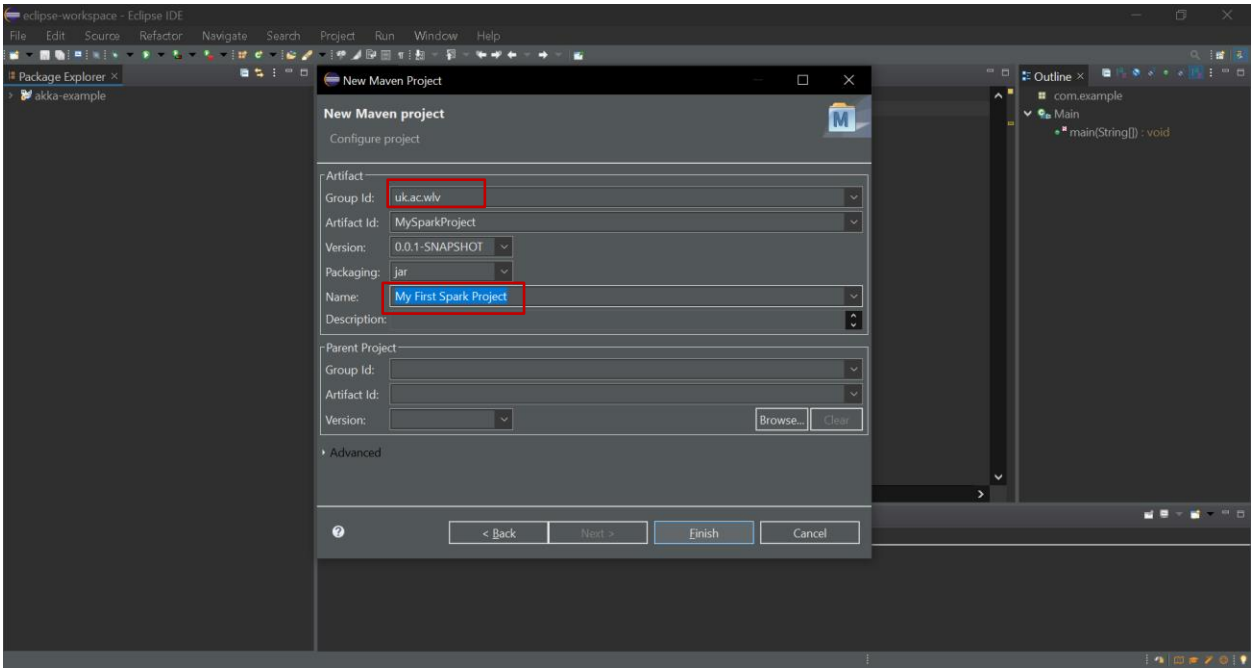
There should be a list of the many sorts of projects you may build; simply select Maven Project and click next.



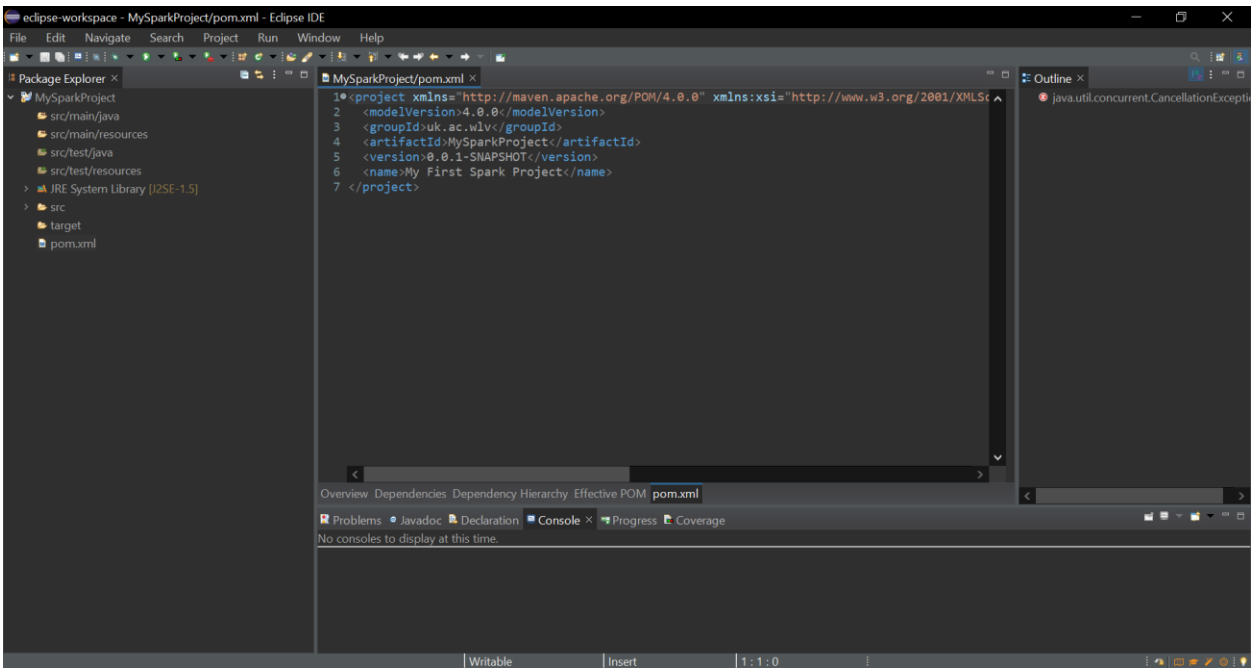
Then, after clicking next. Make sure you choose "simple project" and then click the next button.



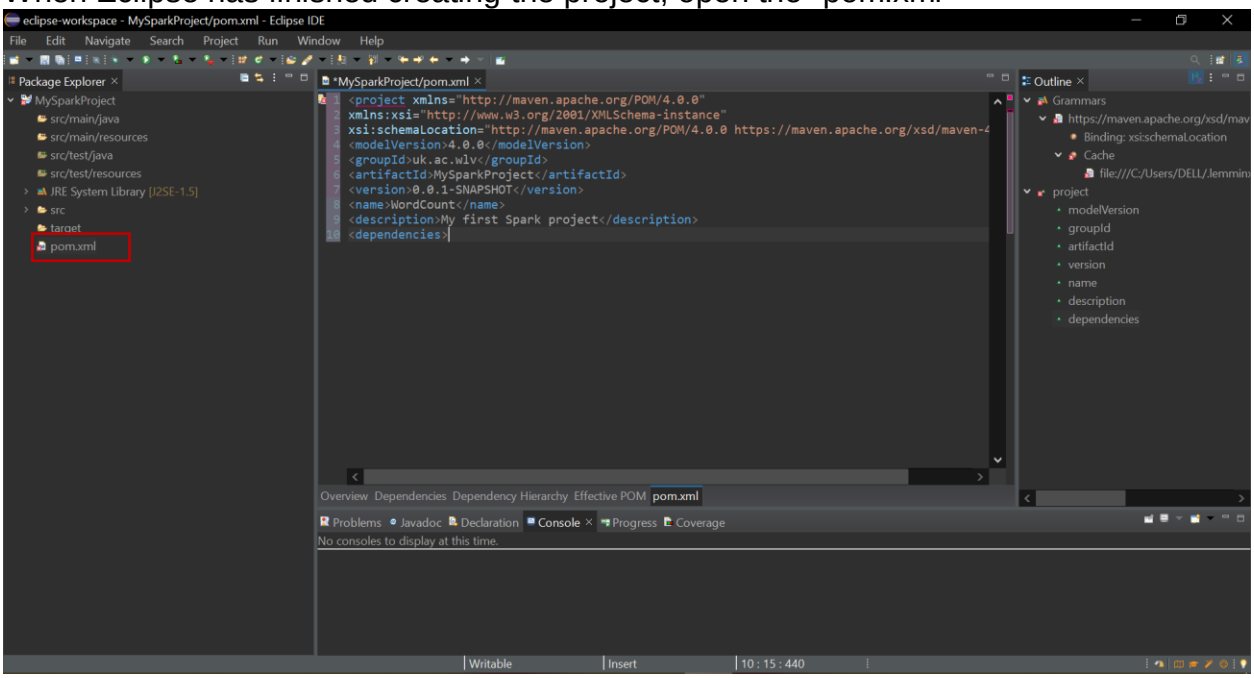
You need to configure. So fill the form as below, name your file as My First Spark Project and click finish.
Note: Your group ID must be named as uk.ac.wlv



After clicking on finish, you will see the text editor and package explorer in the left section.



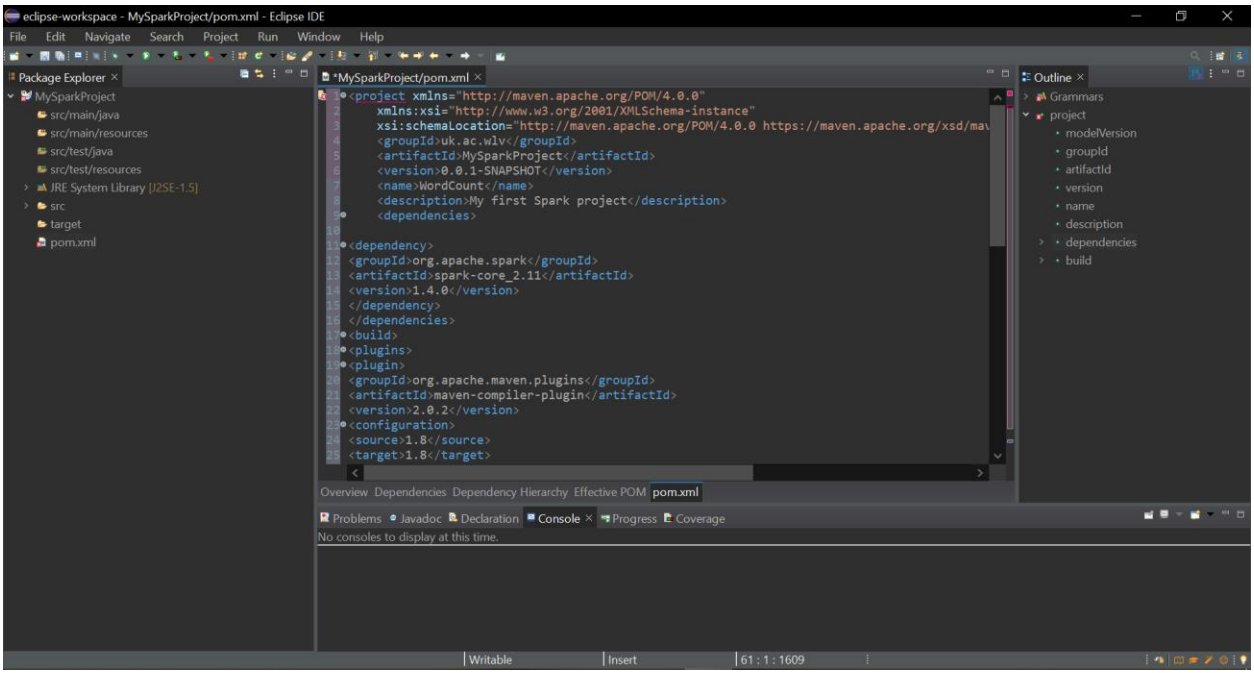
When Eclipse has finished creating the project, open the "pom.xml"



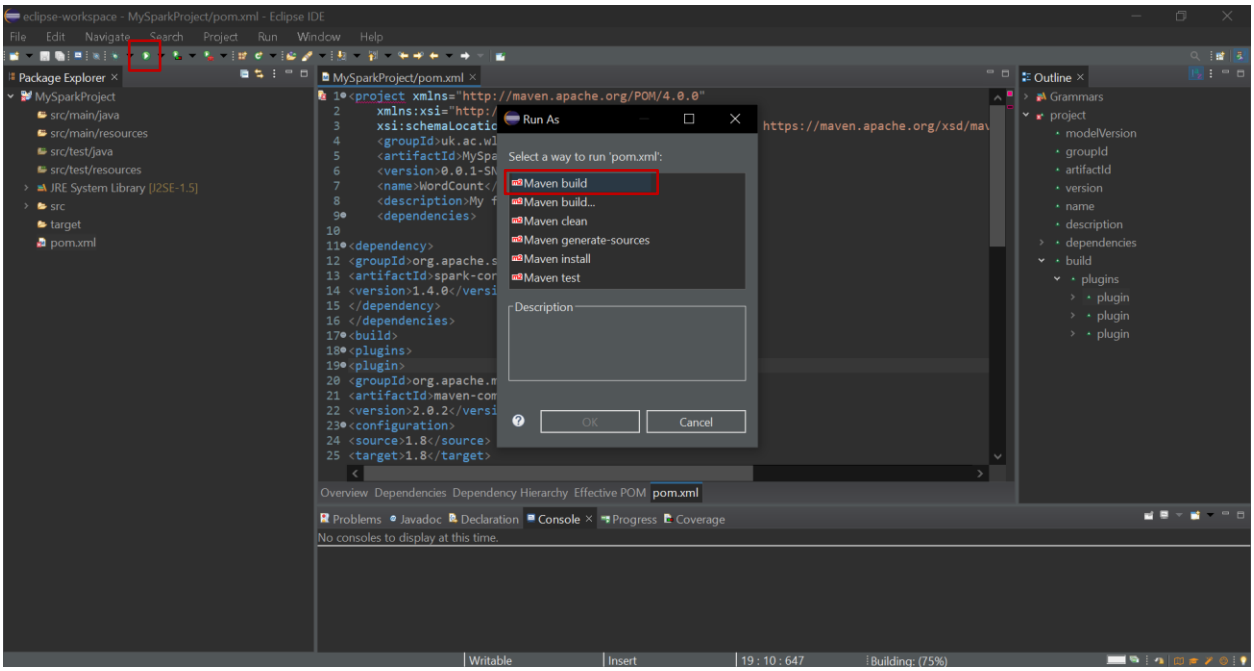
When you open the pom.xml file, you will see a sample java code in the text editor section. Simply replace all of the following code with the following.

```
<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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>uk.ac.wlv</groupId>
  <artifactId>MySparkProject</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>WordCount</name>
  <description>My first Spark project</description>

  <dependencies>
    <dependency>
        <groupId>org.apache.spark</groupId>
        <artifactId>spark-core_2.11</artifactId>
        <version>1.4.0</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>2.0.2</version>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-jar-plugin</artifactId>
            <configuration>
                <archive>
                    <manifest>
                        <addClasspath>true</addClasspath>
                        <classpathPrefix>lib/</classpathPrefix>
                        <mainClass>uk.ac.wlv.WordCount</mainClass>
                    </manifest>
                </archive>
            </configuration>
```

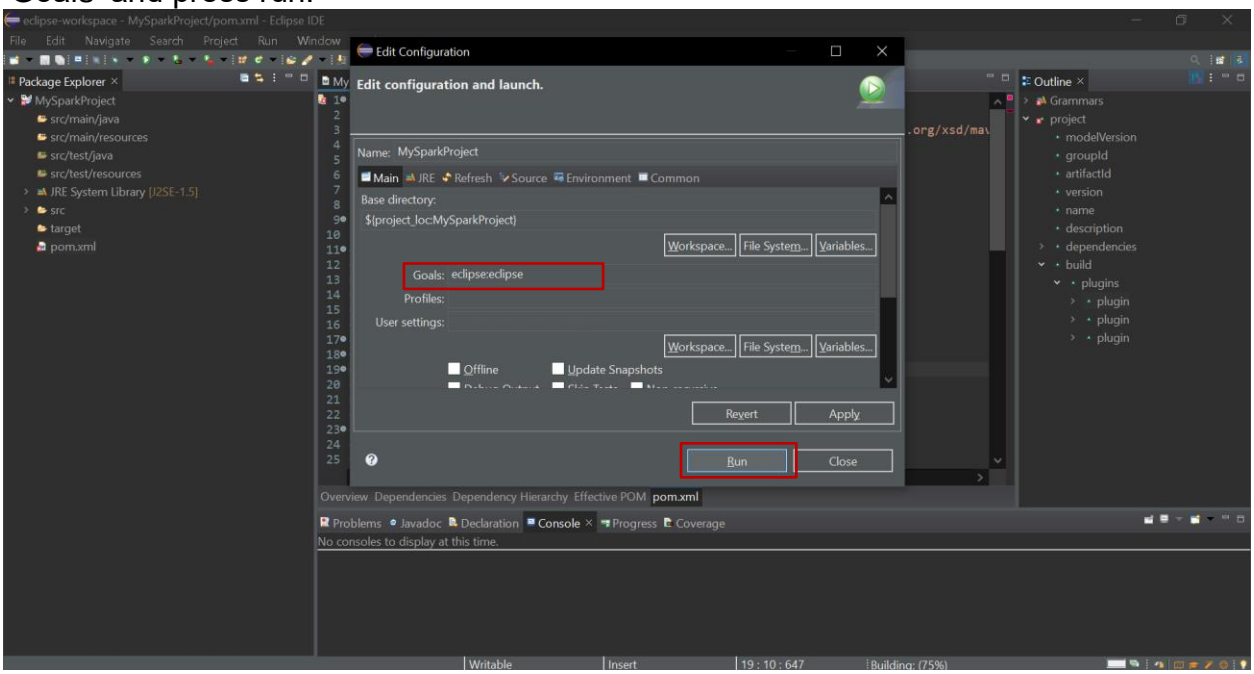
```
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-dependency-plugin</artifactId>
            <executions>
                <execution>
                    <id>copy</id>
                    <phase>install</phase>
                    <goals>
                        <goal>copy-dependencies</goal>
                    </goals>
                    <configuration>
                        <outputDirectory>${project.build.directory}/lib</outp
utDirectory>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    </plugins>
  </build>


</project>
```



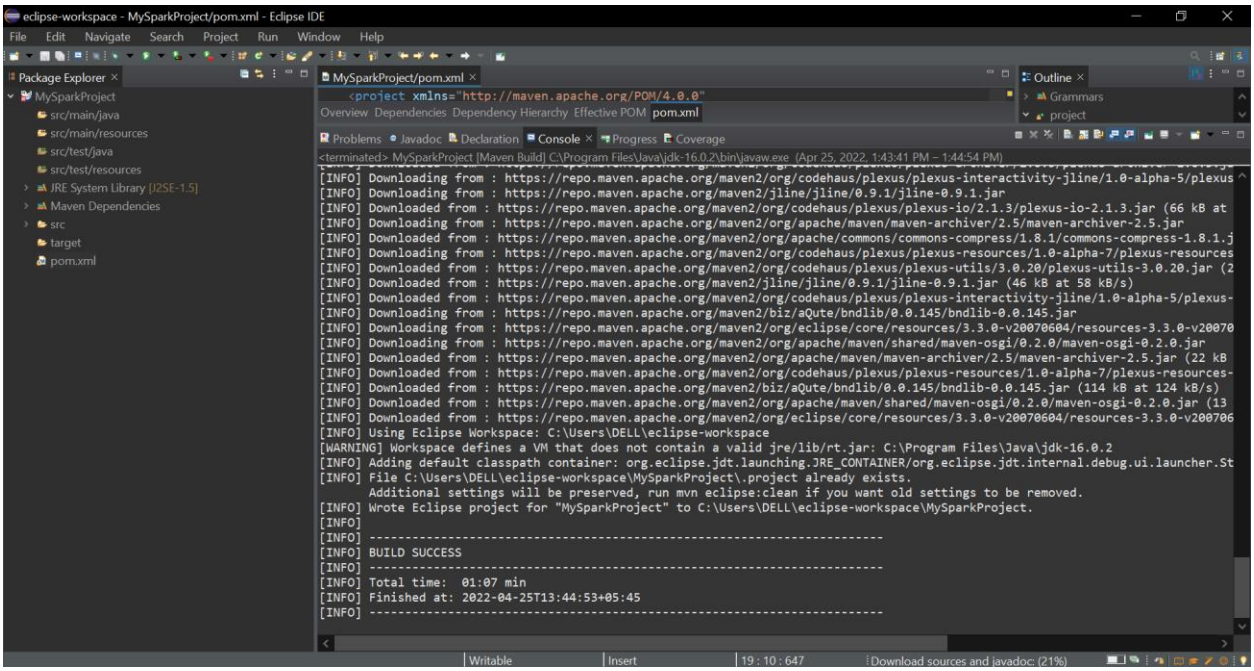
Now, you need to save the file. After saving, you must run this file by clicking the Run button in the upper left corner of the window. Hover your mouse over the button and choose 'Maven Build.'



After that in edit the configuration of the build, simply insert ‘eclipse.eclipse’ inside the ‘Goals’ and press run.

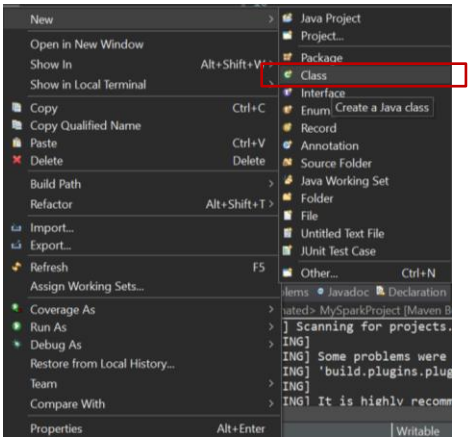


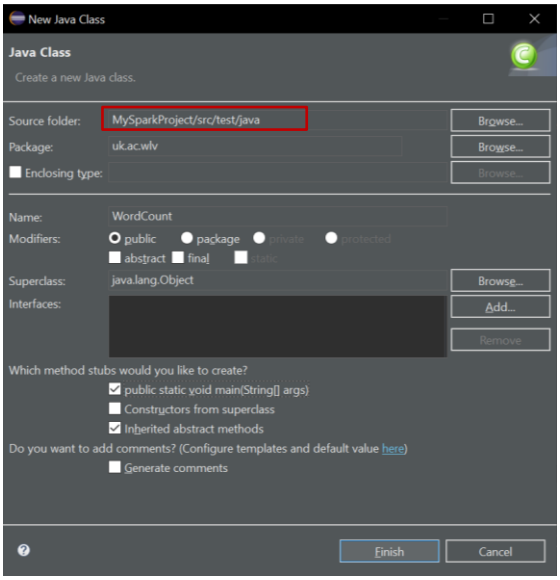
You will see this output in your console.


eclipse-workspace - MySparkProject/pom.xml - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Package Explorer
MySparkProject
src/main/java
src/main/resources
src/test/java
src/test/resources
JRE System Library [J2SE-1.5]
Maven Dependencies
src
target
pom.xml
MySparkProject/pom.xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
Overview Dependencies Dependency Hierarchy Effective POM pom.xml
Outline
Grammars
project
Problems Javadoc Declaration Console Progress Coverage
<terminated> MySparkProject [Maven Build] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (Apr 25, 2022, 1:43:41 PM – 1:44:54 PM)
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-interactivity-jline/1.0-alpha-5/plexus
[INFO] Downloading from : https://repo.maven.apache.org/maven2/jline/jline/0.9.1/jline-0.9.1.jar
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-io/2.1.3/plexus-io-2.1.3.jar (66 kB at
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/apache/maven/maven-archiver/2.5/maven-archiver-2.5.jar
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/apache/commons/commons-compress/1.8.1/commons-compress-1.8.1.j
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-resources/1.0-alpha-7/plexus-resources
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.0.20/plexus-utils-3.0.20.jar (2
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/jline/jline/0.9.1/jline-0.9.1.jar (46 kB at 58 kB/s)
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-interactivity-jline/1.0-alpha-5/plexus-
[INFO] Downloading from : https://repo.maven.apache.org/maven2/biz/aQute/bndlib/0.0.145/bndlib-0.0.145.jar
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/eclipse/core/resources/3.3.0-v20070604/resources-3.3.0-v20070
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-osgi/0.2.0/maven-osgi-0.2.0.jar
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/apache/maven/maven-archiver/2.5/maven-archiver-2.5.jar (22 kB
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-resources/1.0-alpha-7/plexus-resources-
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/biz/aQute/bndlib/0.0.145/bndlib-0.0.145.jar (114 kB at 124 kB/s)
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-osgi/0.2.0/maven-osgi-0.2.0.jar (13
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/eclipse/core/resources/3.3.0-v20070604/resources-3.3.0-v200706
[INFO] Using Eclipse Workspace: C:\Users\DELL\eclipse-workspace
[WARNING] Workspace defines a VM that does not contain a valid jre/lib/rt.jar: C:\Program Files\Java\jdk-16.0.2
[INFO] Adding default classpath container: org.eclipse.jdt.launching.JRE_CONTAINER/org.eclipse.jdt.internal.debug.ui.launcher.St
[INFO] File C:\Users\DELL\eclipse-workspace\MySparkProject\.project already exists.
Additional settings will be preserved, run mvn eclipse:clean if you want old settings to be removed.
[INFO] Wrote Eclipse project for "MySparkProject" to C:\Users\DELL\eclipse-workspace\MySparkProject.
[INFO]
[INFO] --
[INFO] BUILD SUCCESS
[INFO] --
[INFO] Total time: 01:07 min
[INFO] Finished at: 2022-04-25T13:44:53+05:45
[INFO] --
Writable
Insert
19 : 10 : 647
Download sources and javadoc: (21%)

Step 4: Creating Word Count program

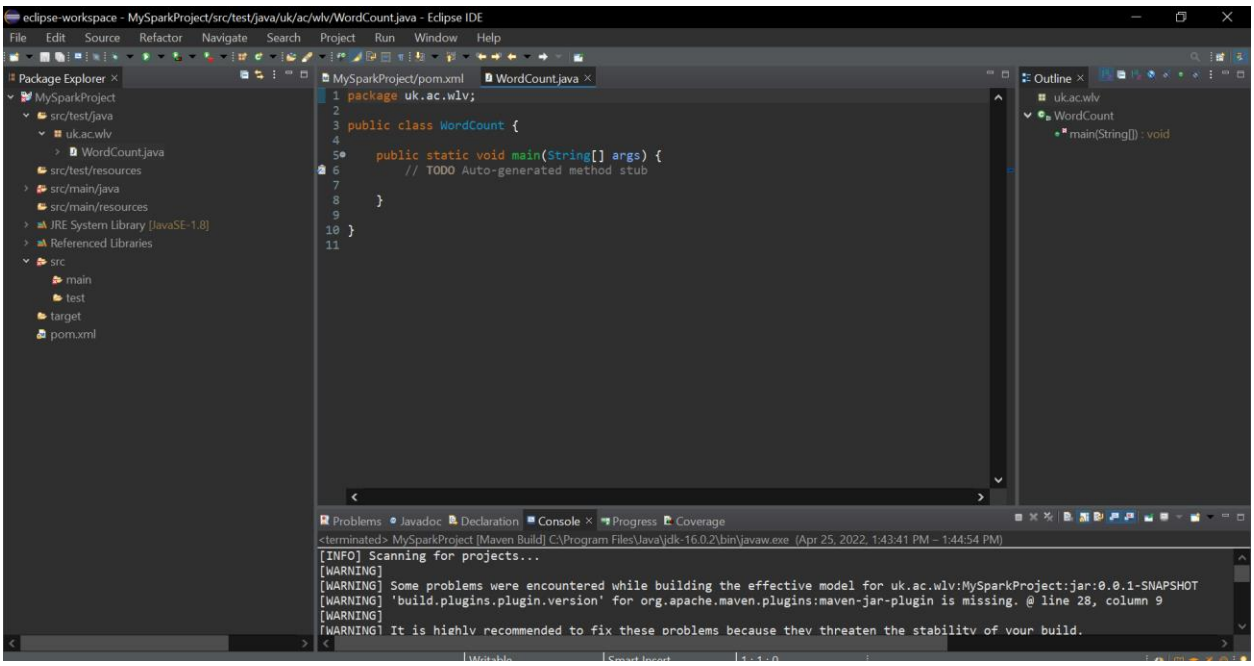
Now you need to create the WordCount Program. Create a new java class, by right-clicking on src/main/java folder.





Click on finish now.

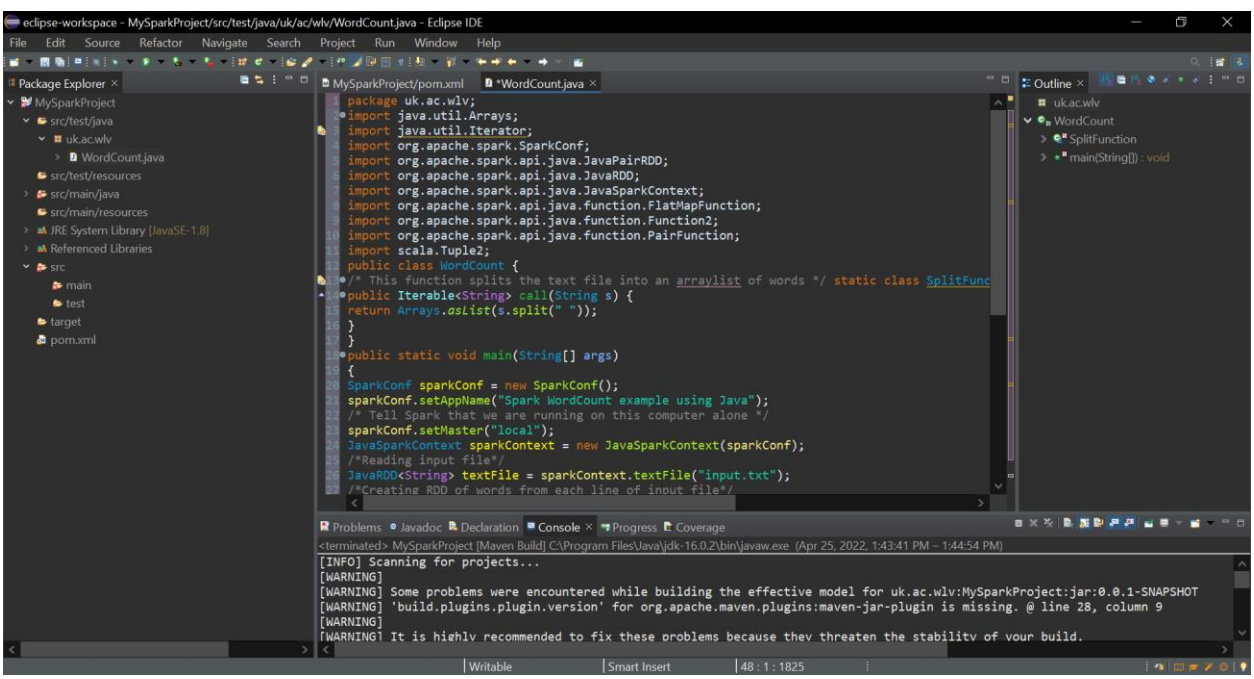
This generates a new Java class called 'WordCount.java.'



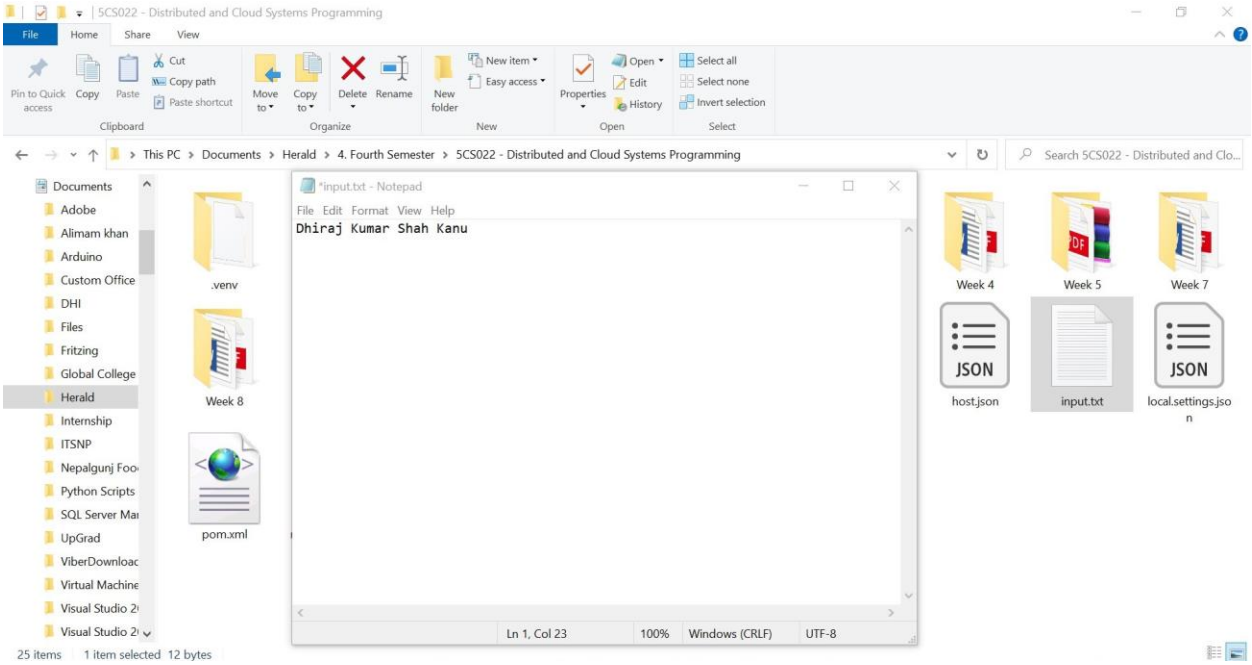
Now change the codes with the ones shown below.

```
package uk.ac.wlv;
import java.util.Arrays;
import java.util.Iterator;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.FlatMapFunction;
import org.apache.spark.api.java.function.Function2;
import org.apache.spark.api.java.function.PairFunction;
import scala.Tuple2;
public class WordCount {
  /* This function splits the text file into an arraylist of words */
  static class SplitFunction implements FlatMapFunction < String, String > {
    public Iterable < String > call(String s) {
      return Arrays.asList(s.split(" "));
    }
  }
  public static void main(String[] args) {
    SparkConf sparkConf = new SparkConf();
    sparkConf.setAppName("Spark WordCount example using Java");
    /* Tell Spark that we are running on this computer alone */
    sparkConf.setMaster("local");
    JavaSparkContext sparkContext = new JavaSparkContext(sparkConf);
```

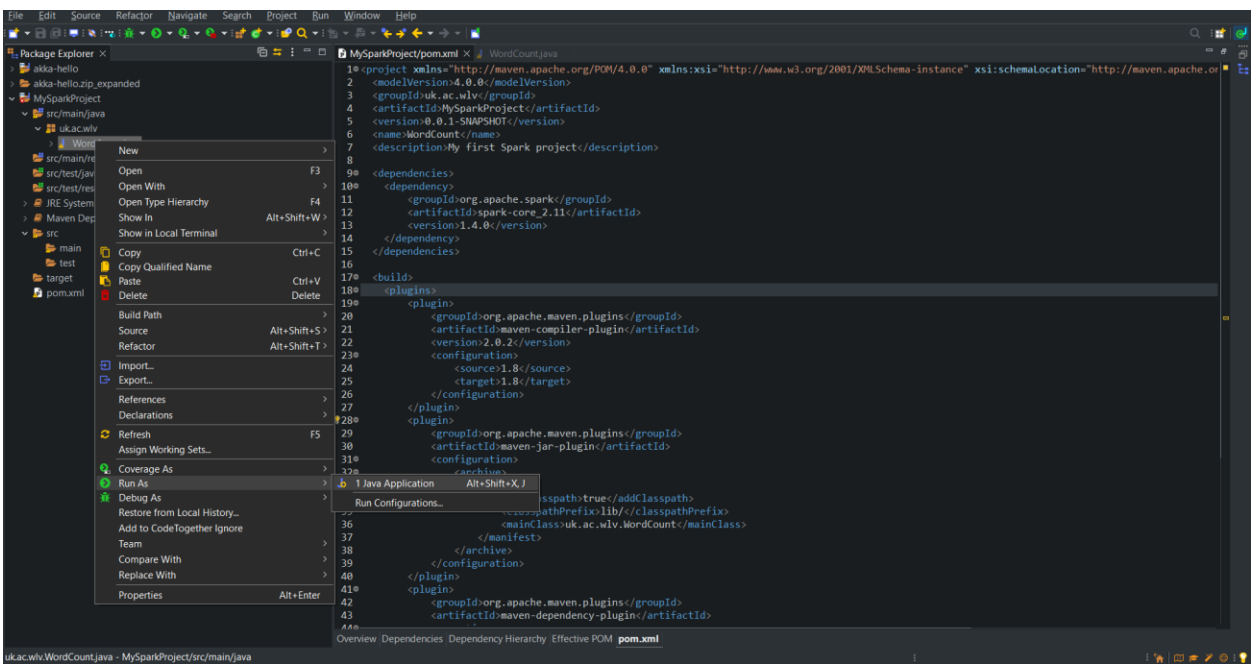
```
    /*Reading input file*/

    JavaRDD < String > textFile = sparkContext.textFile("input.txt");
    /*Creating RDD of words from each line of input file*/
    JavaRDD < String > words = textFile.flatMap(new SplitFunction());
    /*Generate Pair of Word with count */
    JavaPairRDD < String, Integer > pairs = words.mapToPair(new PairFunction
< String, String, Integer > () {
      public Tuple2 < String, Integer > call(String s) {
        return new Tuple2 < String, Integer > (s, 1);
      }
    });
    /* Aggregate Pairs of Same Words with count */
    JavaPairRDD < String, Integer > counts = pairs.reduceByKey(
      new Function2 < Integer, Integer, Integer > () {
        public Integer call(Integer a, Integer b) {
          return a + b;
        }
      });
    /*Saving the result file */
    counts.saveAsTextFile("output");
    sparkContext.stop();
    sparkContext.close();
  }
}
```



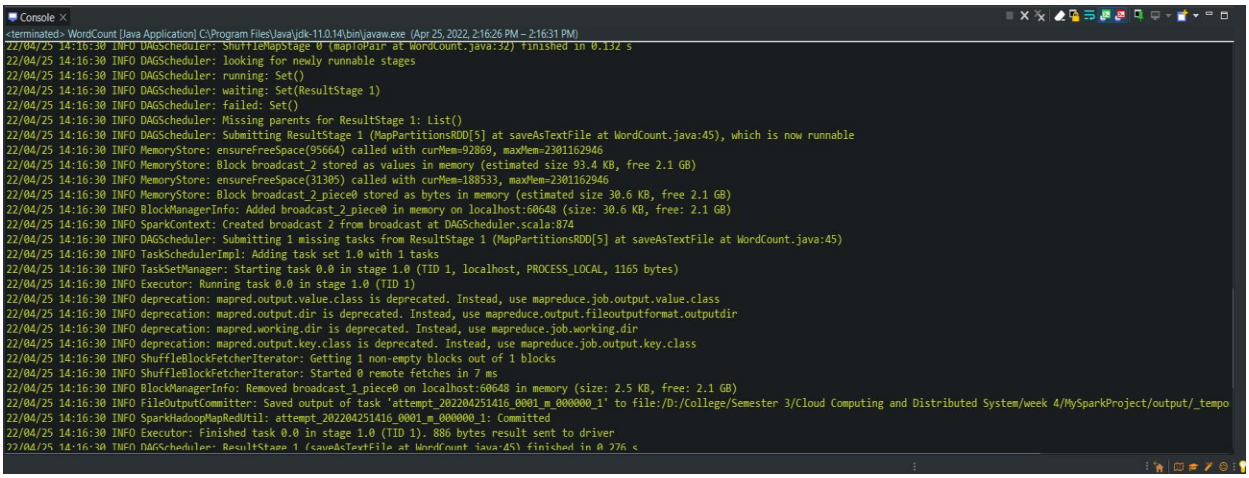
Create a new .txt file and save it in the same directory as the pom.xml file,



Step 5: Running the WordCount Spark Program



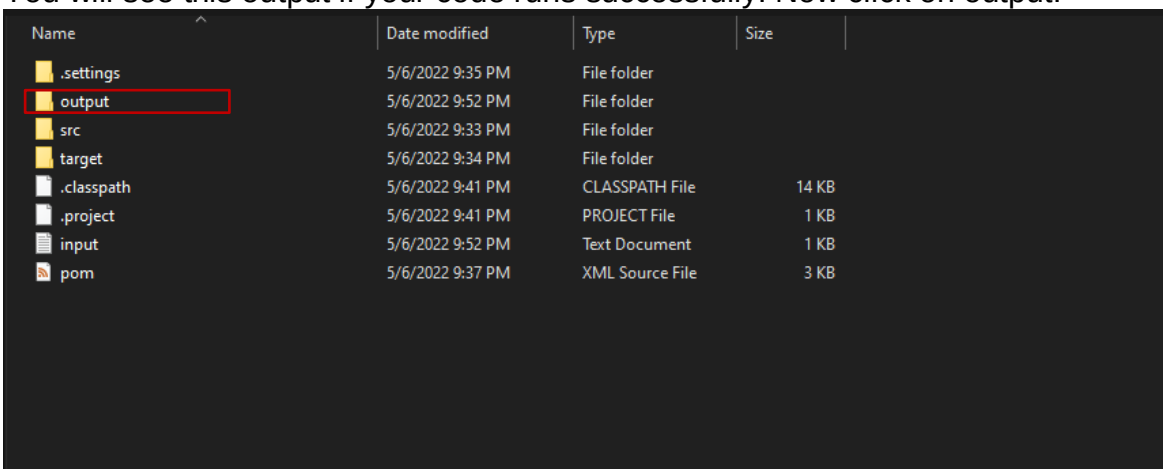
Make sure that the file WordCount.java is opened as the current file in the Eclipse editor and click the run button on the Eclipse menu toolbar.



If it runs successfully, you should see the Spark logging output in Eclipse

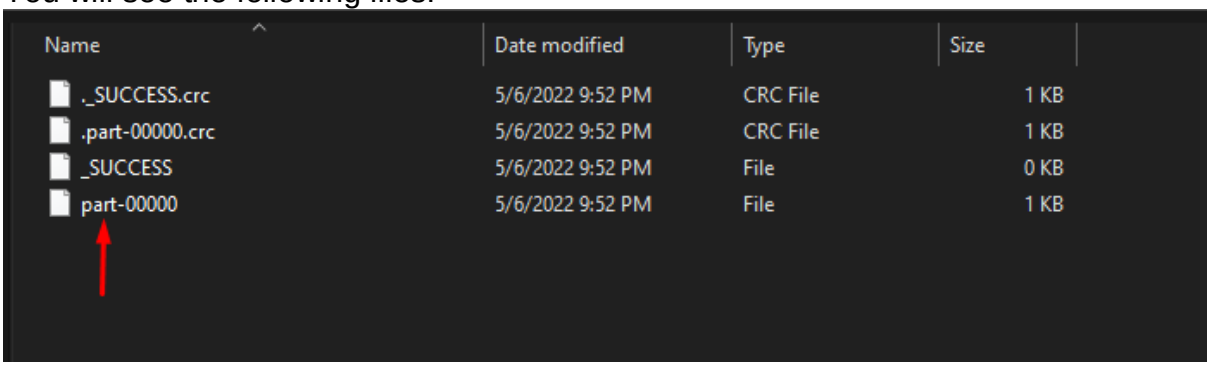
Step 6: View the output results

You will see this output if your code runs successfully. Now click on output.



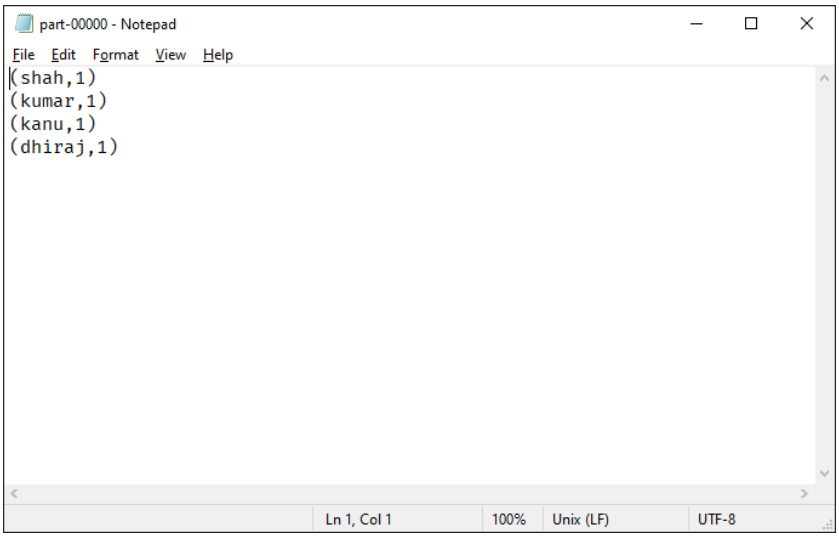
Name	Date modified	Type	Size
.settings	5/6/2022 9:35 PM	File folder	
output	5/6/2022 9:52 PM	File folder	
src	5/6/2022 9:33 PM	File folder	
target	5/6/2022 9:34 PM	File folder	
.classpath	5/6/2022 9:41 PM	CLASSPATH File	14 KB
.project	5/6/2022 9:41 PM	PROJECT File	1 KB
input	5/6/2022 9:52 PM	Text Document	1 KB
pom	5/6/2022 9:37 PM	XML Source File	3 KB

You will see the following files.



Name	Date modified	Type	Size
._SUCCESS.crc	5/6/2022 9:52 PM	CRC File	1 KB
.part-00000.crc	5/6/2022 9:52 PM	CRC File	1 KB
_SUCCESS	5/6/2022 9:52 PM	File	0 KB
part-00000	5/6/2022 9:52 PM	File	1 KB

The results that we are looking for will be in the file "part-0000". Open that file in Notepad, and you should see a list of words and their counts.



Task
Create a Spark program to count letters instead of words.

Here, in line number 17, remove the space between inverted commas.

week 4 - MySparkProject/src/main/java/uk/ac/wlv/WordCount.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

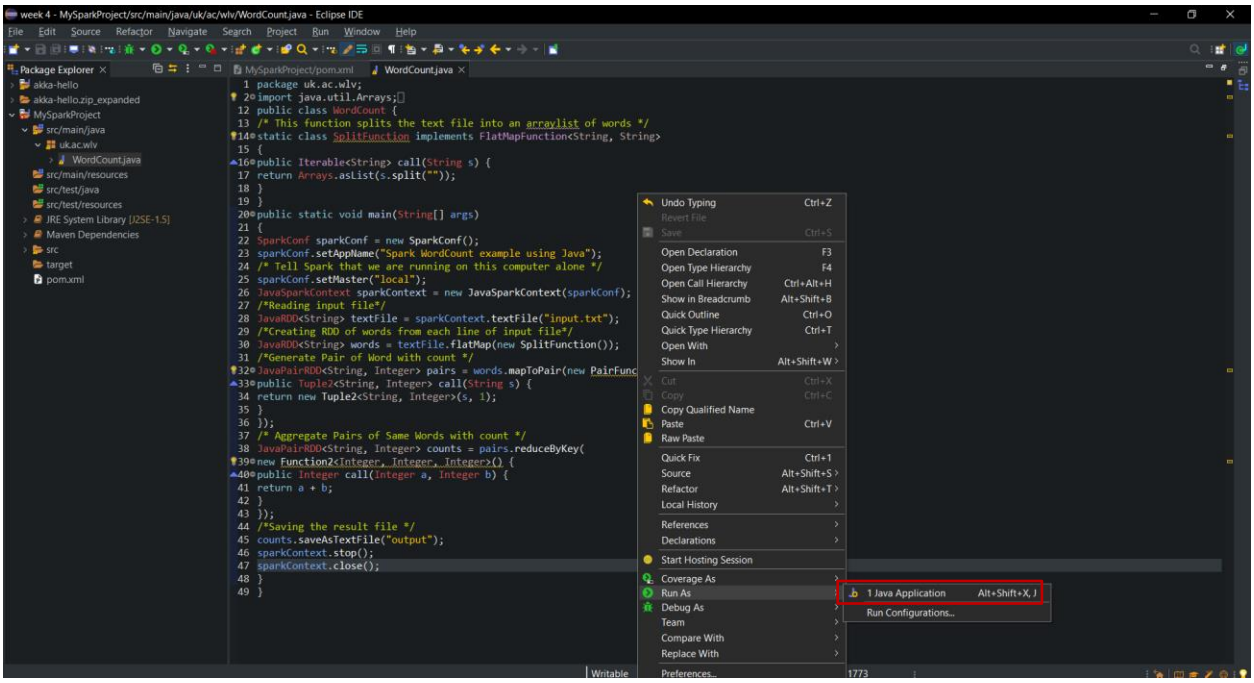
Package Explorer

- akka-hello
- akka-hello.zip_expanded
- MySparkProject
  - src/main/java
    - uk.ac.wlv
      - WordCount.java
  - src/main/resources
  - src/test/java
  - src/test/resources
  - JRE System Library [J2SE-1.5]
  - Maven Dependencies
  - src
  - target
  - pom.xml

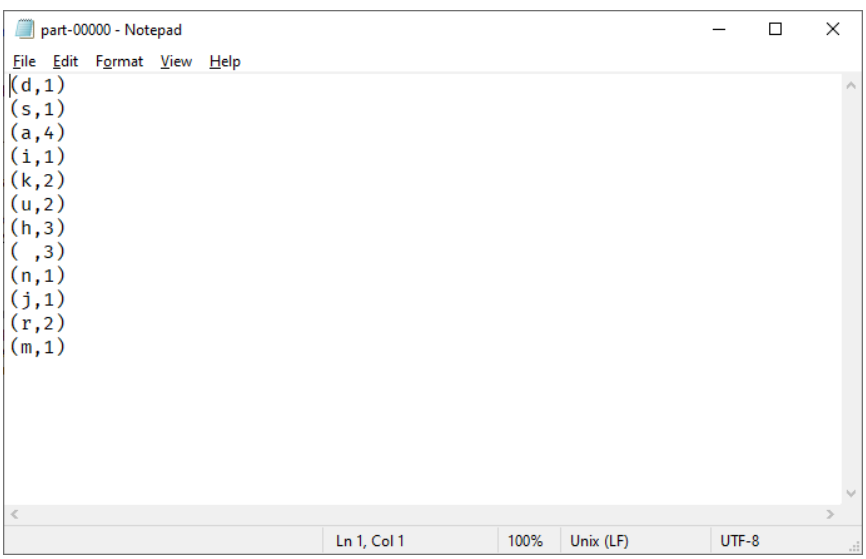
MySparkProject/pom.xml | WordCount.java

```
1  package uk.ac.wlv;
2  import java.util.Arrays;
12 public class WordCount {
13 /* This function splits the text file into an arraylist of words */
14 static class SplitFunction implements FlatMapFunction<String, Stri
15 {
16 public Iterable<String> call(String s) {
17 return Arrays.asList(s.split(""));
18 }
19 }
20 public static void main(String[] args)
21 {
22 SparkConf sparkConf = new SparkConf();
23 sparkConf.setAppName("Spark WordCount example using Java");
24 /* Tell Spark that we are running on this computer alone */
25 sparkConf.setMaster("local");
26 JavaSparkContext sparkContext = new JavaSparkContext(sparkConf);
27 /*Reading input file*/
28 JavaRDD<String> textFile = sparkContext.textFile("input.txt");
29 /*Creating RDD of words from each line of input file*/
30 JavaRDD<String> words = textFile.flatMap(new SplitFunction());
31 /*Generate Pair of Word with count */
32 JavaPairRDD<String, Integer> pairs = words.mapToPair(new PairFunct
33 public Tuple2<String, Integer> call(String s) {
34 return new Tuple2<String, Integer>(s, 1);
35 }
36 });
37 /* Aggregate Pairs of Same Words with count */
38 JavaPairRDD<String, Integer> counts = pairs.reduceByKey(
39 new Function2<Integer, Integer, Integer>() {
40 public Integer call(Integer a, Integer b) {
```

Right click on the file and run as a Java program



You will see the final output like this, which will be in the file "part-0000". Open that file in Notepad, and you should see a list of words and their counts.



Summary

So, first we downloaded the Hadoop file from the google classroom. Then we edited the environment variables, started a new spark project, modify the codes, and develop a functional program in this manner. The program we just created counts the total amount of words entered into the 'input.txt' file. The output is then shown as seen above.