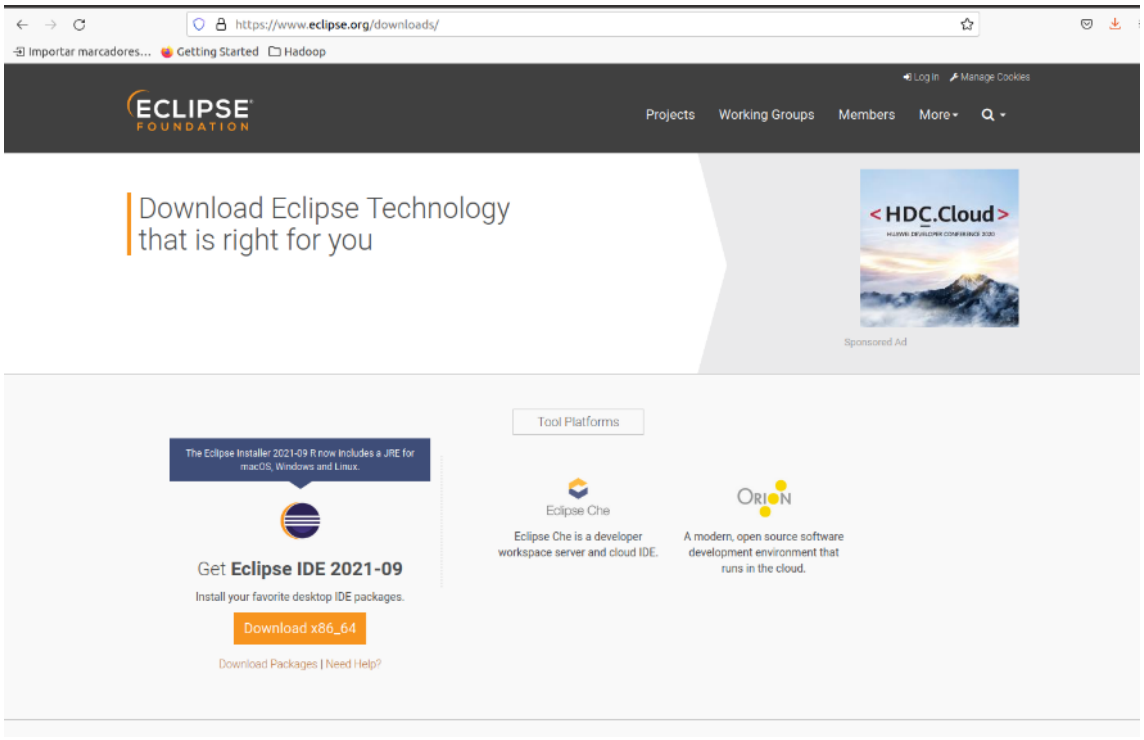
5. Desarrollo de aplicaciones mapreduce con eclipse y java

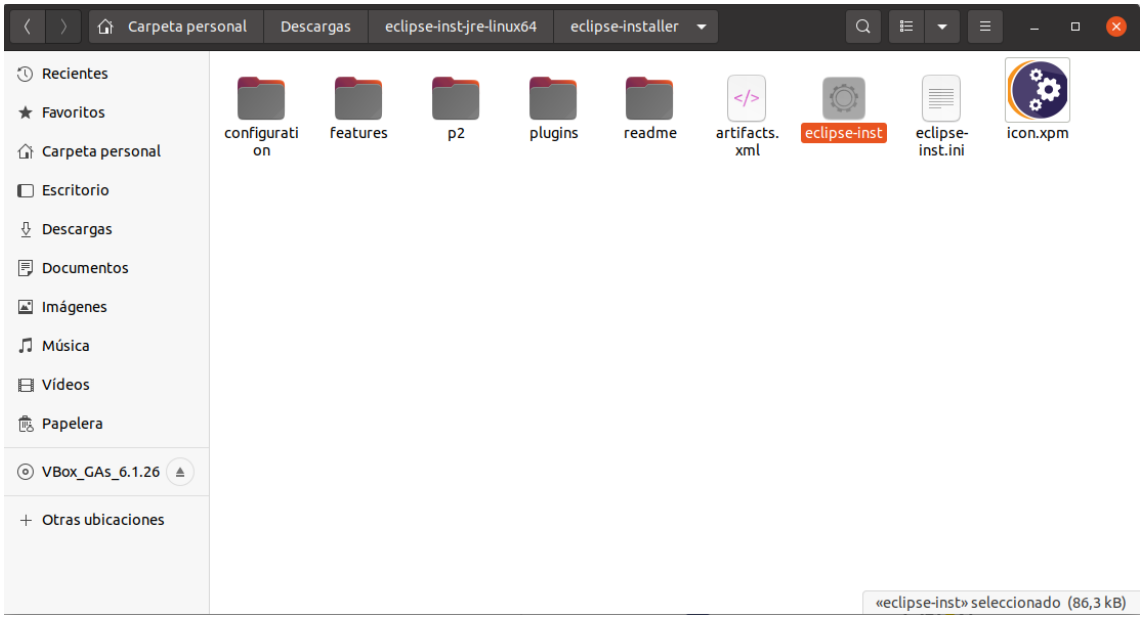
Se puede crear aplicaciones directamente con el editor y compilarlas con java, pero resulta más cómodo utilizar entornos de desarrollo como eclipse

5.1. Instalar Eclipse

Descargar eclipse desde https://www.eclipse.org/downloads/



Descargar y descomprimir el archivo con el instalador y ejecutar el programa “eclipse-inst”



Una vez instalado eclipse ya podemos iniciarlo y crear el primer proyecto

5.2. Configurar eclipse

Modificaremos el nivel de compilación y el JRE (Java Runtime Environment) con la versión de java instalada

5.2.1. Verificar versión de java

Desde el terminal comprobar la versión instalada

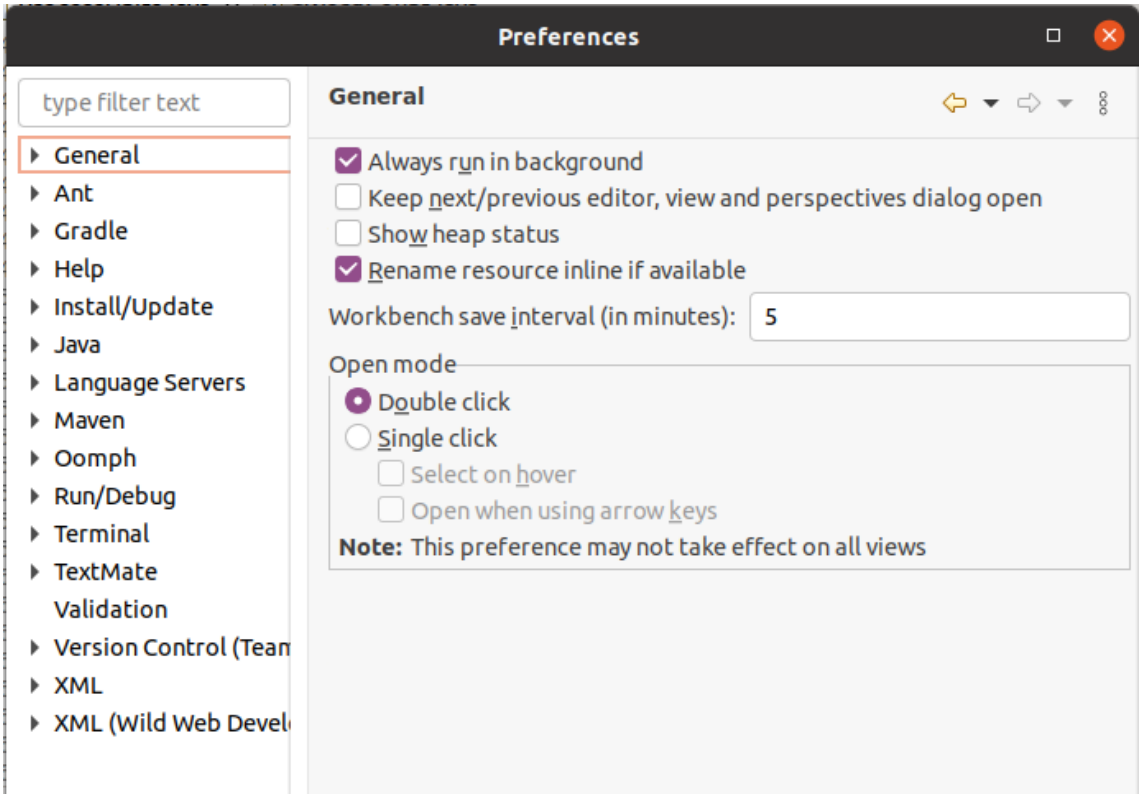
```
$ java -version
```

```
hadoop@hadoop2:~/hadoop/hadoop-3.3.1-aarch64/hadoop-3.3.1$ java -version
openjdk version "1.8.0_292"
OpenJDK Runtime Environment (build 1.8.0_292-8u292-b10-0ubuntu1~20.04-b10)
OpenJDK 64-Bit Server VM (build 25.292-b10, mixed mode)
```

Para esta versión la ubicación en el sistema es /usr/lib/jvm/java-1.8.0-openjdk-amd64. Mirar punto 1.3)

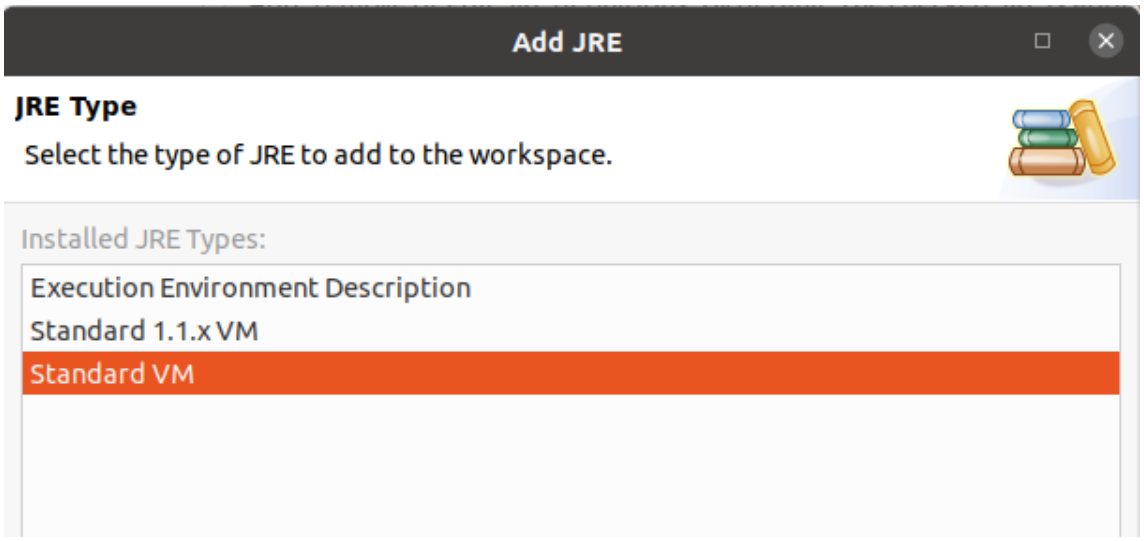
5.2.2. Modificar las propiedades

Acceder a las propiedades desde el menú Window/Preferences

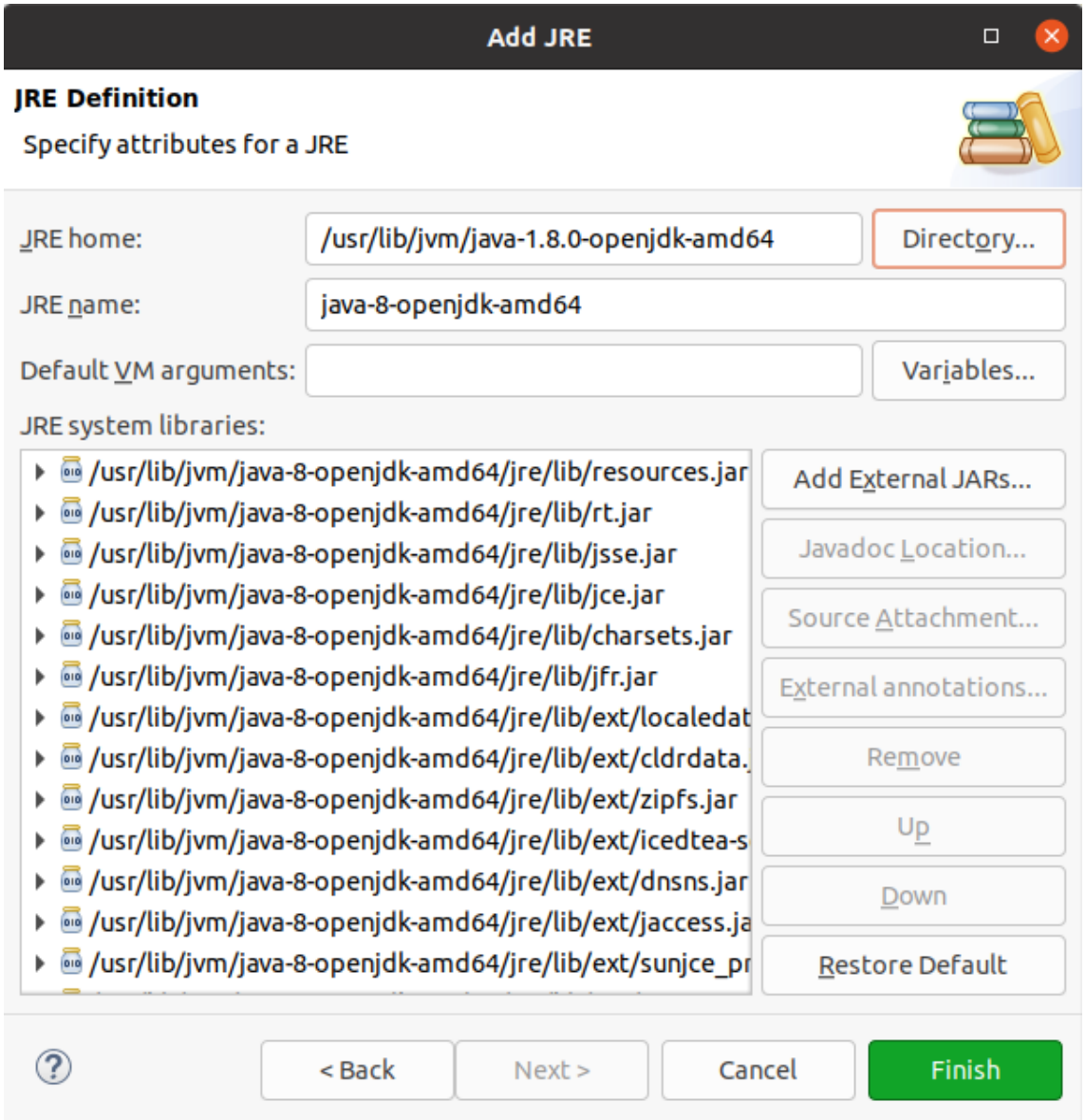


Java/Installed JREs y agregar (pulsar ADD) el JRE instalado en la carpeta es /usr/lib/jvm/java-1.8.0-openjdk-amd64

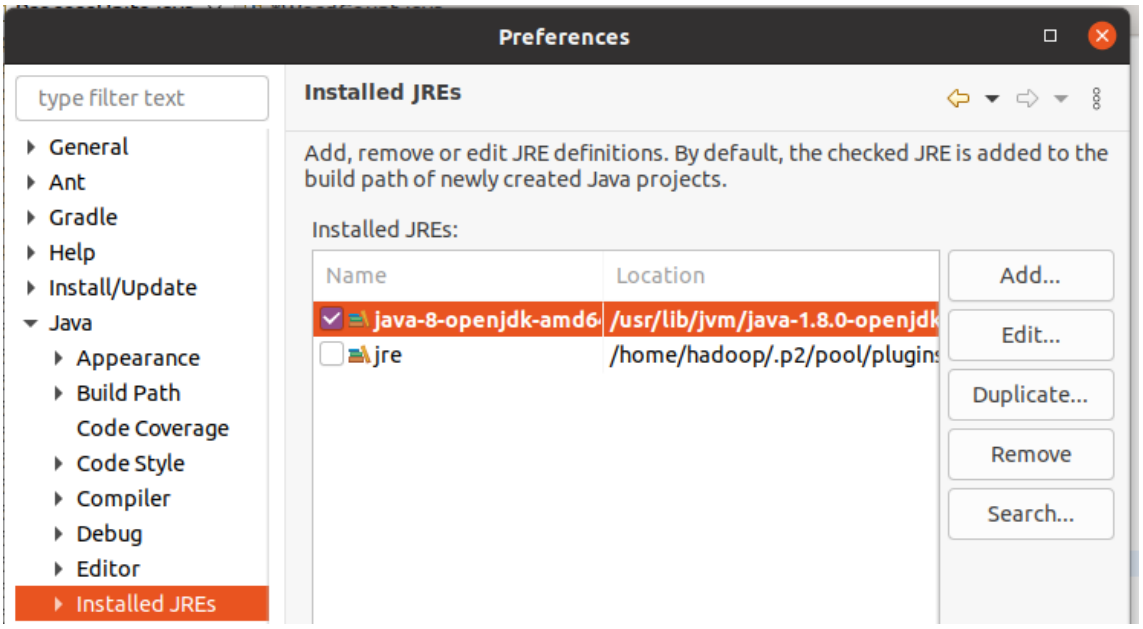
En la ventana JRE Type seleccionar Standard VM y pulsar Siguiente



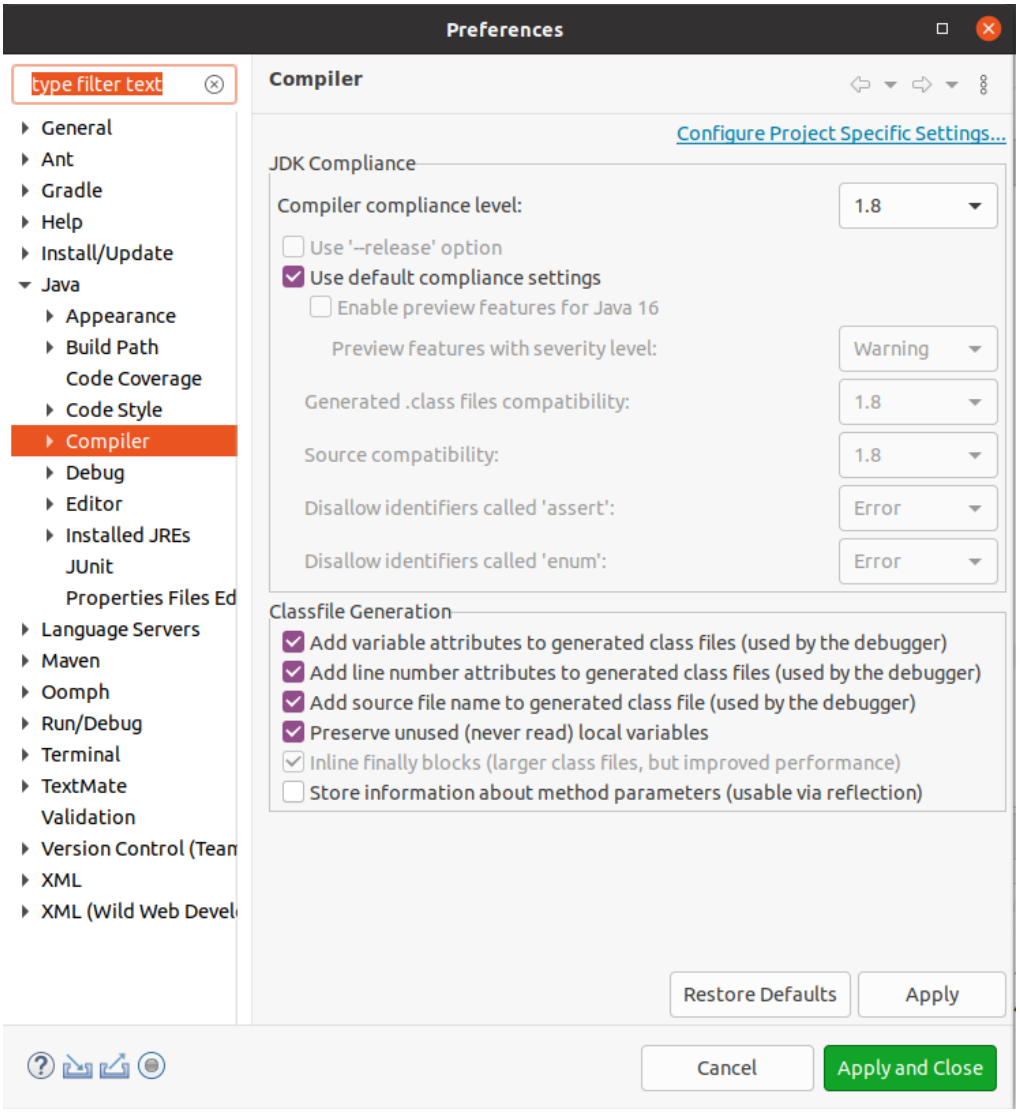
Indicar la ruta en JRE Home con el botón Directory y pulsar Finalizar



Marcarlo como JRE por defecto y pulsar Apply

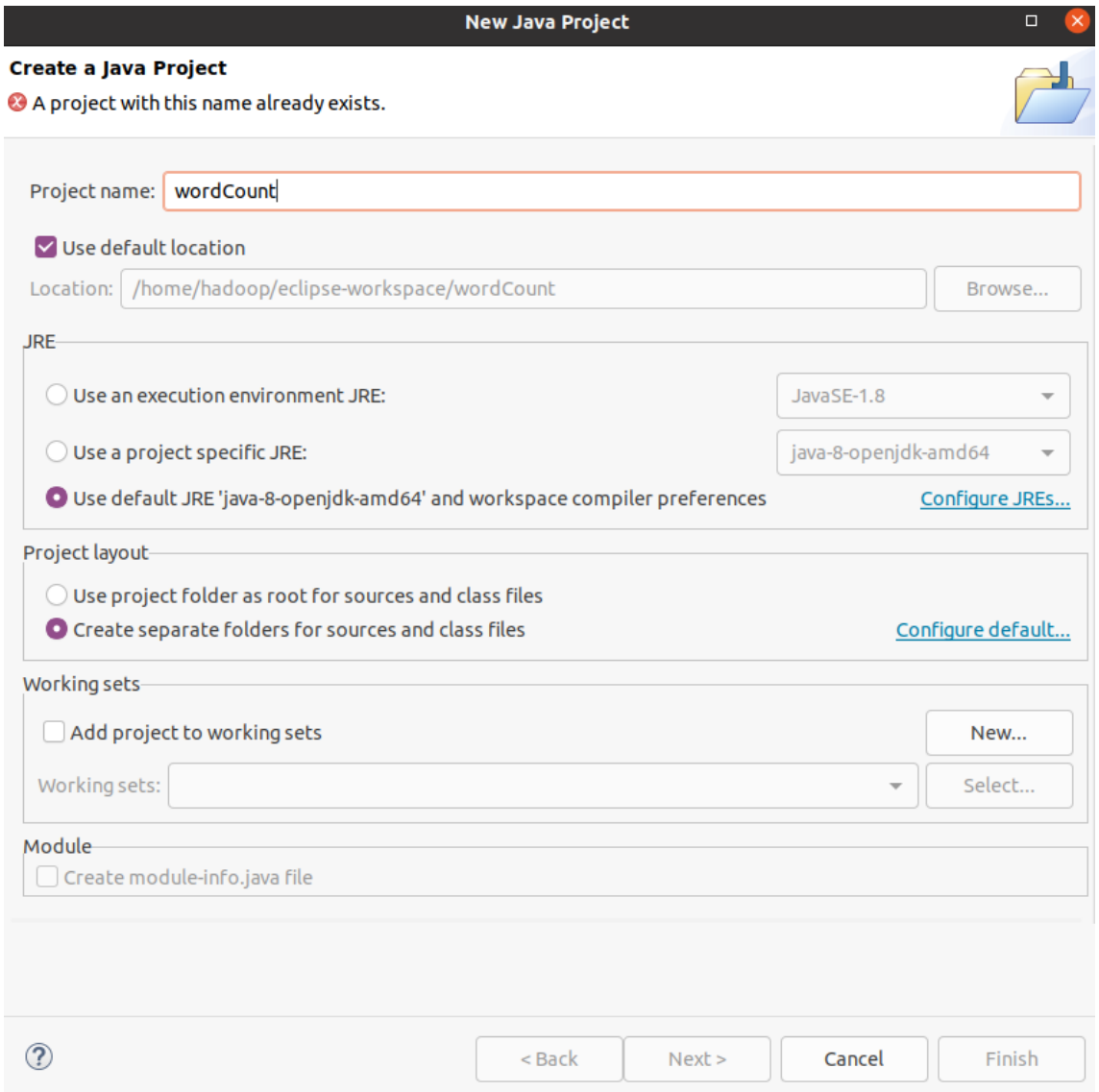


Desde Java/Compiler, modificar JDK Compilance level a 1.8 y pulsar Apply and Close

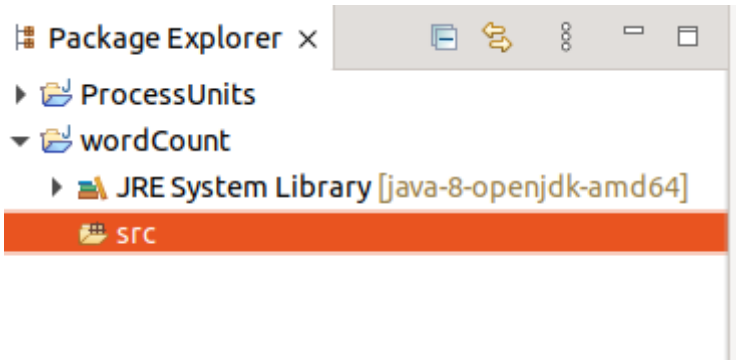


5.3. Crear el proyecto WordCount

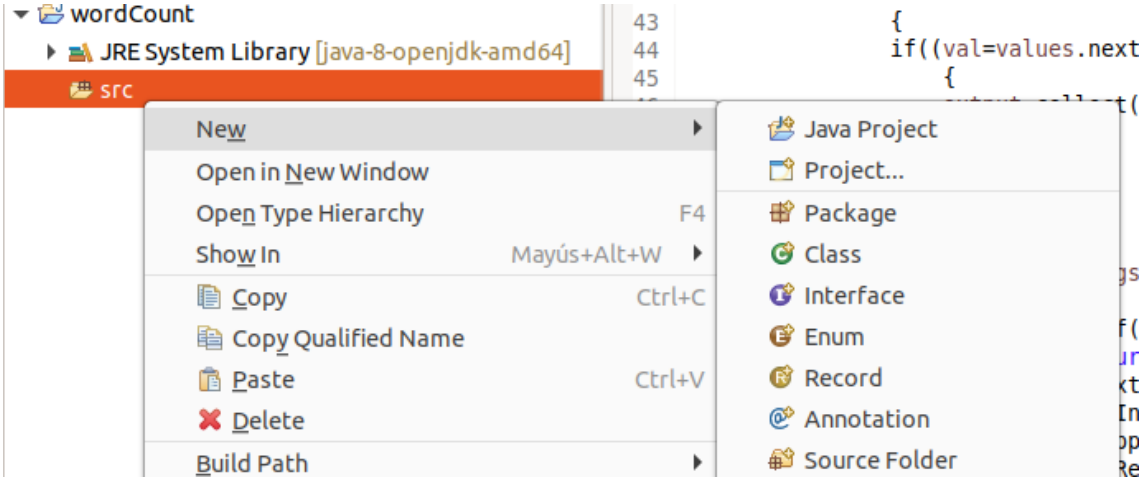
Desde el menú File/New/ Java Proyect indicarle el nombre wordCount, usar la ubicación por defecto y el JRE por defecto que acabamos de configurar



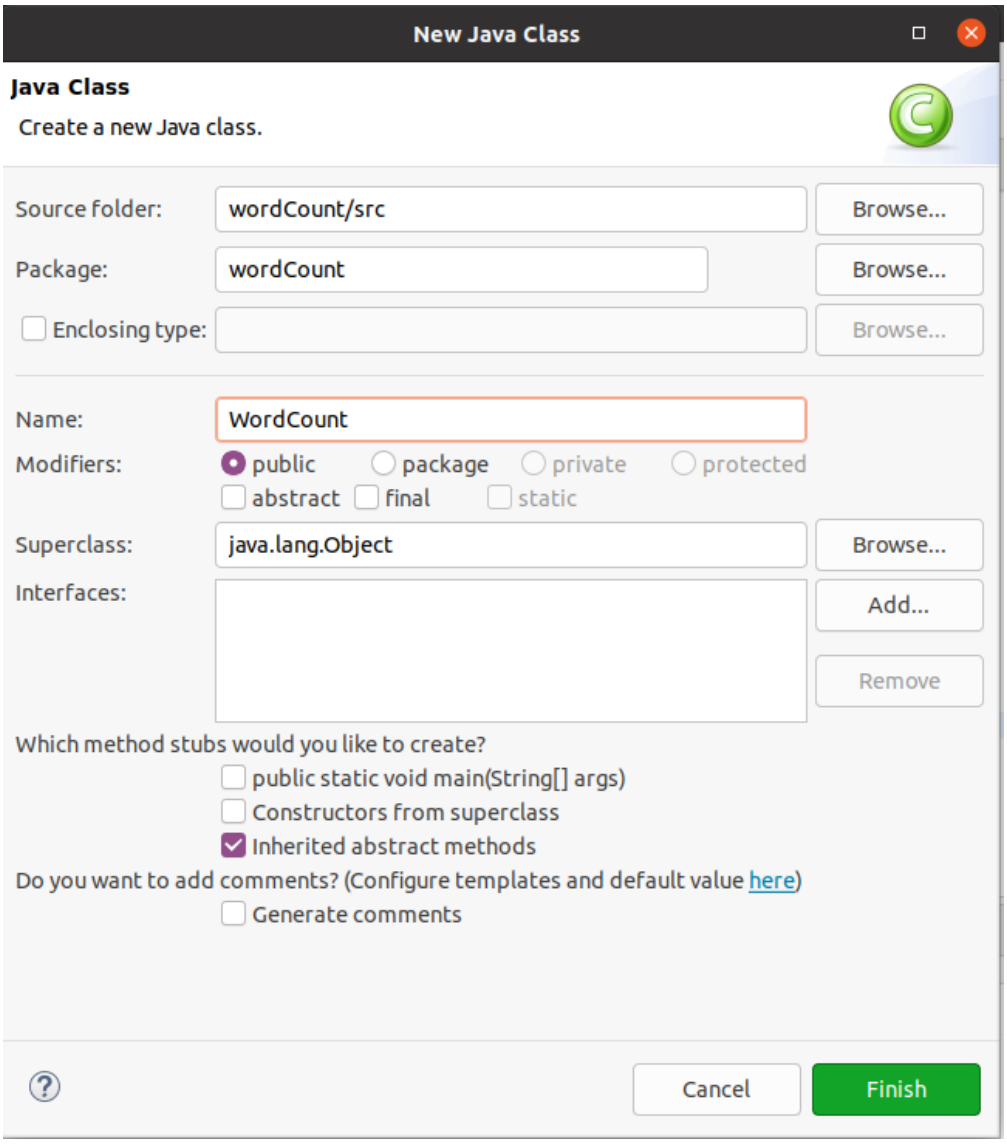
Muestra el proyecto en el árbol de proyectos



5.3.1. Agregar el archivo de clase para wordCount. Con el botón derecho, seleccionar New/Class

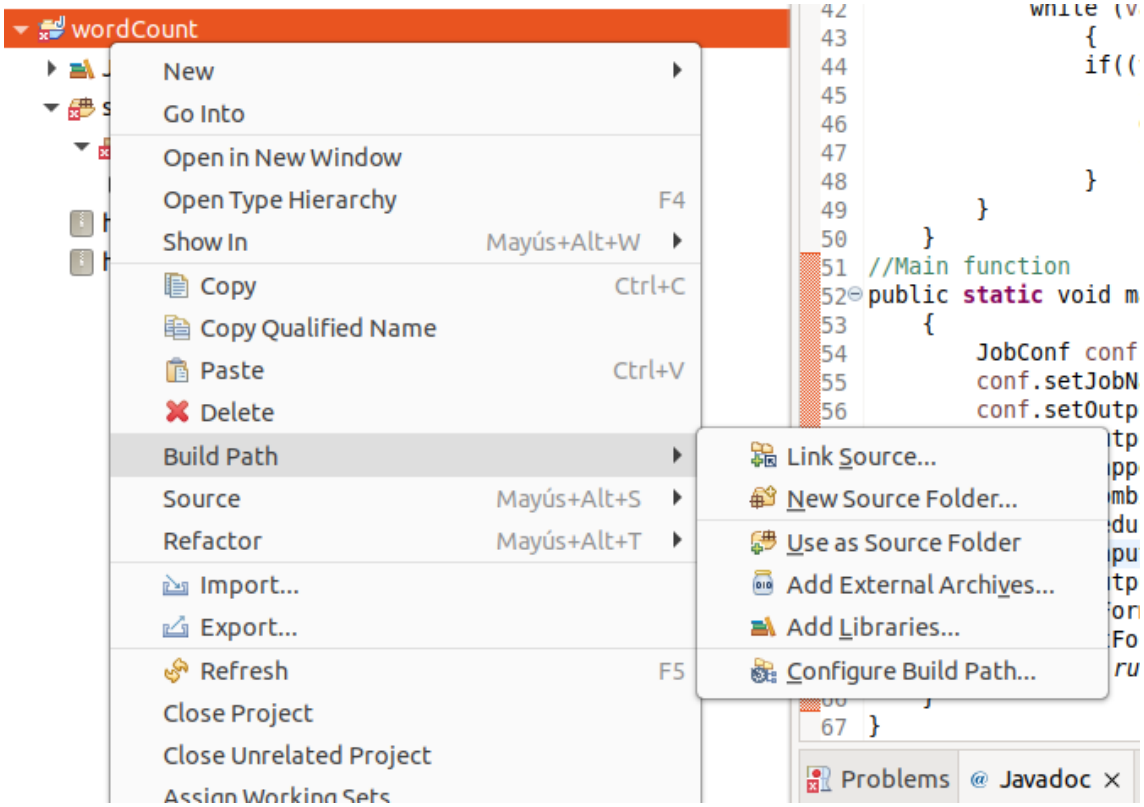


Indicar el nombre de la clase y pulsar Finish



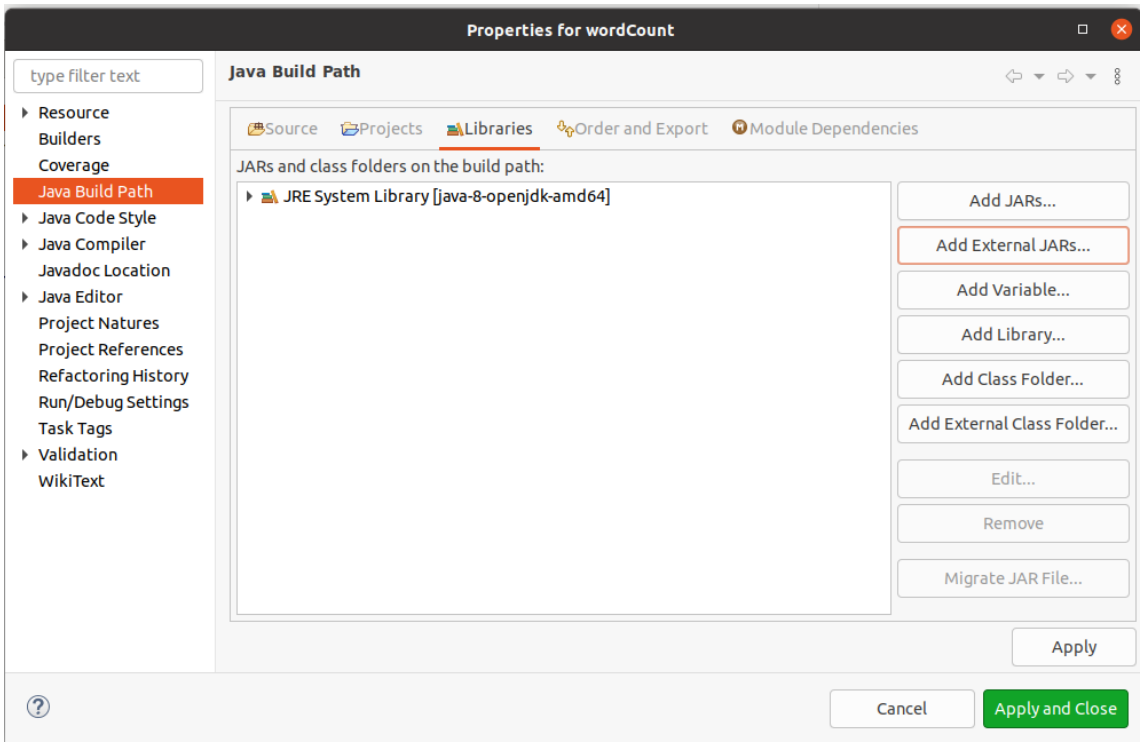
5.3.2. Configurar la ruta de compilación del proyecto

Desde el menú contextual del proyecto seleccionar Build Path/Configure Build Path

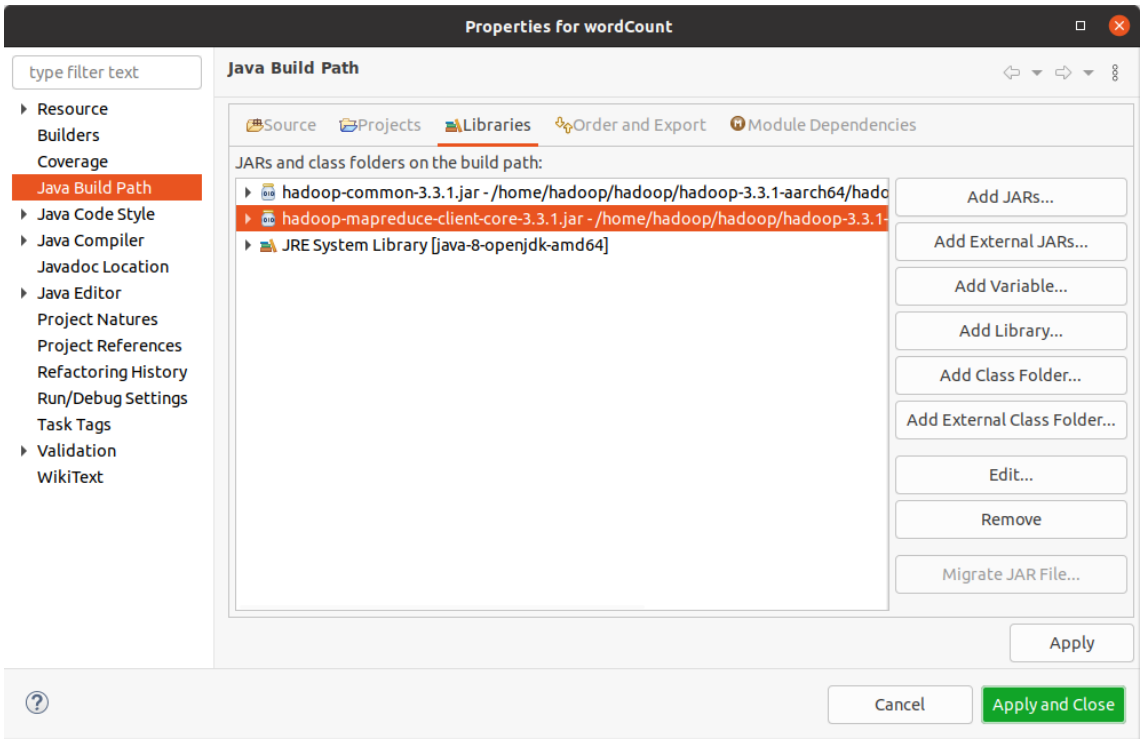


Son los archivos hadoop-common-3.3.1.jar ($HADOOP_HOME/share/Hadoop/common) y Hadoop-mapreduce-client-core-3.3.1.jar ($HADOOP_HOME/share/Hadoop/mapreduce)

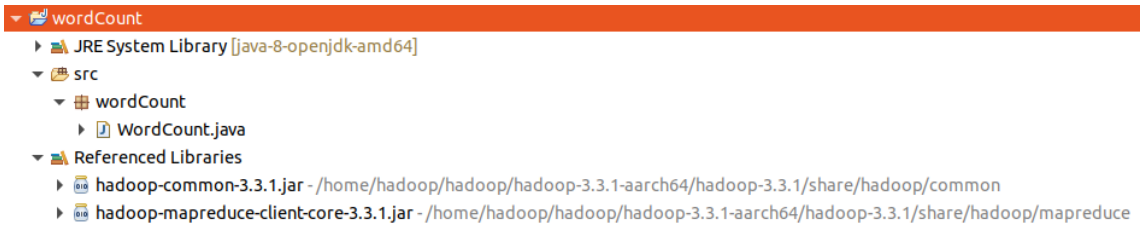
Añadir los archivos anteriores con ADD External JARs



Una vez agregados los archivos, pulsar Apply and Close



La estructura del proyecto tiene que quedar



5.3.3. Agregar el código de ejemplo de wordCount

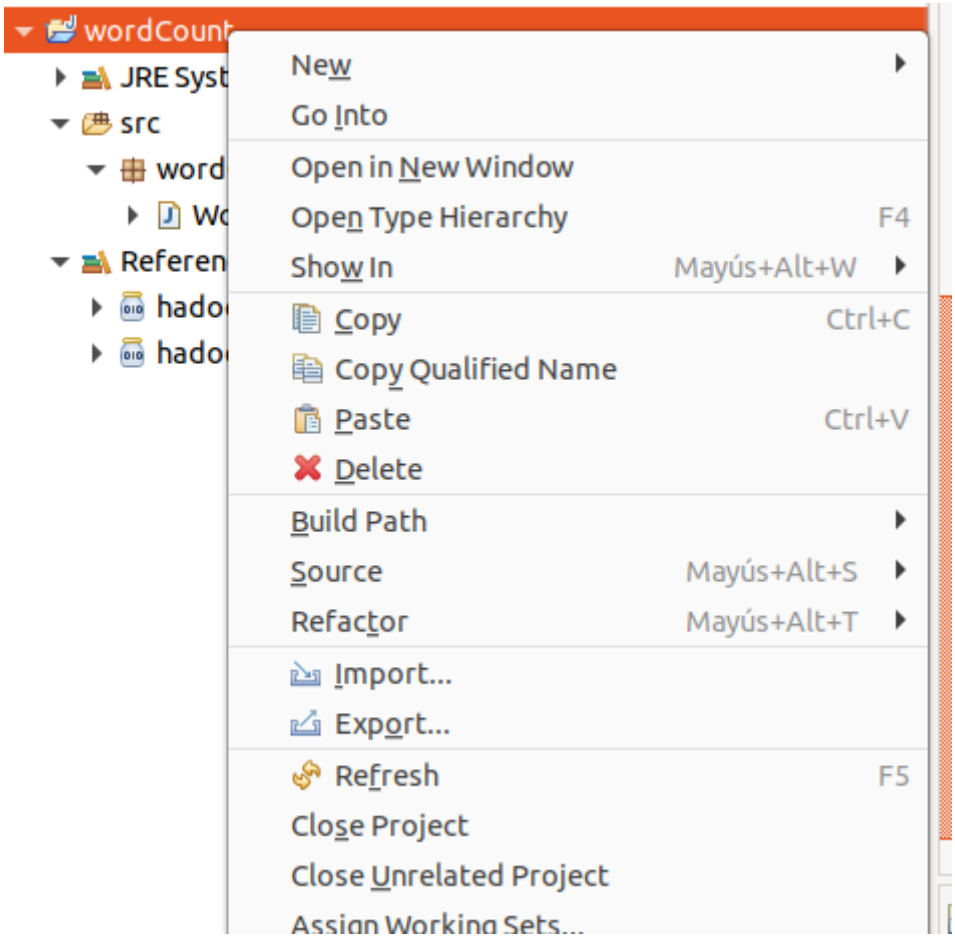
```
import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.conf.Configuration;
```

```
import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;

import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.fs.Path;


public class WordCount{

       public static class Map extends
Mapper<LongWritable,Text,Text,IntWritable> {

            public void map(LongWritable key, Text value, Context context)
throws IOException,InterruptedException {

                  String line = value.toString();

                  StringTokenizer tokenizer = new StringTokenizer(line);

                  while (tokenizer.hasMoreTokens()) {

                        value.set(tokenizer.nextToken());

                        context.write(value, new IntWritable(1));

                   }

            }

       }

       public static class Reduce extends
Reducer<Text,IntWritable,Text,IntWritable> {

            public void reduce(Text key, Iterable<IntWritable>
values,Context context)

                   throws IOException,InterruptedException {

                        int sum=0;

                        for(IntWritable x: values)

                        {

                              sum+=x.get();

                        }

                        context.write(key, new IntWritable(sum));

            }

       }
```

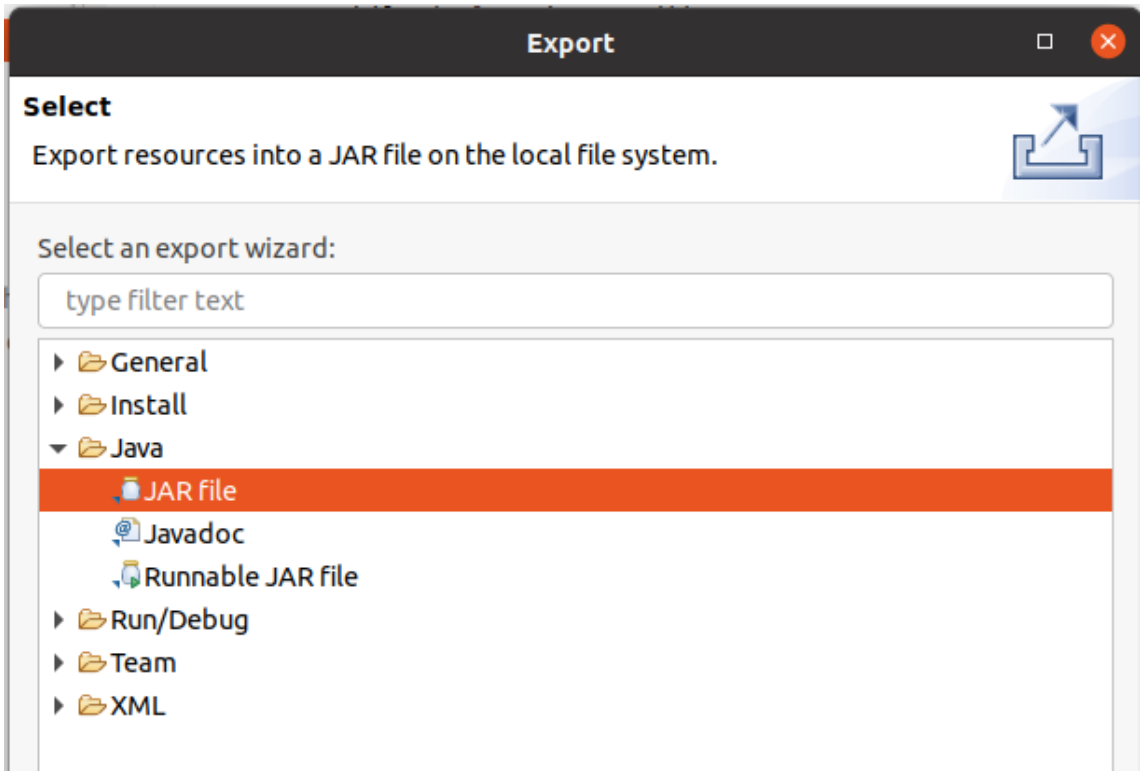
```
        @SuppressWarnings({ "deprecation" })
        public static void main(String[] args) throws Exception {
         Configuration conf= new Configuration();
         try (Job job = new Job(conf,"My Word Count Program")) {
                            job.setJarByClass(WordCount.class);
                            job.setMapperClass(Map.class);
                            job.setReducerClass(Reduce.class);
                            job.setOutputKeyClass(Text.class);
                            job.setOutputValueClass(IntWritable.class);
                            job.setInputFormatClass(TextInputFormat.class);
                            job.setOutputFormatClass(TextOutputFormat.class);
                            Path outputPath = new Path(args[1]);
                            //Configuring the input/output path from the
filesystem into the job
                            FileInputFormat.addInputPath(job, new
Path(args[0]));
                            FileOutputFormat.setOutputPath(job, new
Path(args[1]));
                            //deleting the output path automatically from hdfs
so that we don't have to delete it explicitly
                            outputPath.getFileSystem(conf).delete(outputPath);
                            //exiting the job only if the flag value becomes
false
                            System.exit(job.waitForCompletion(true) ? 0 : 1);
                }
    }
}
```

5.4. Exportar WordCount

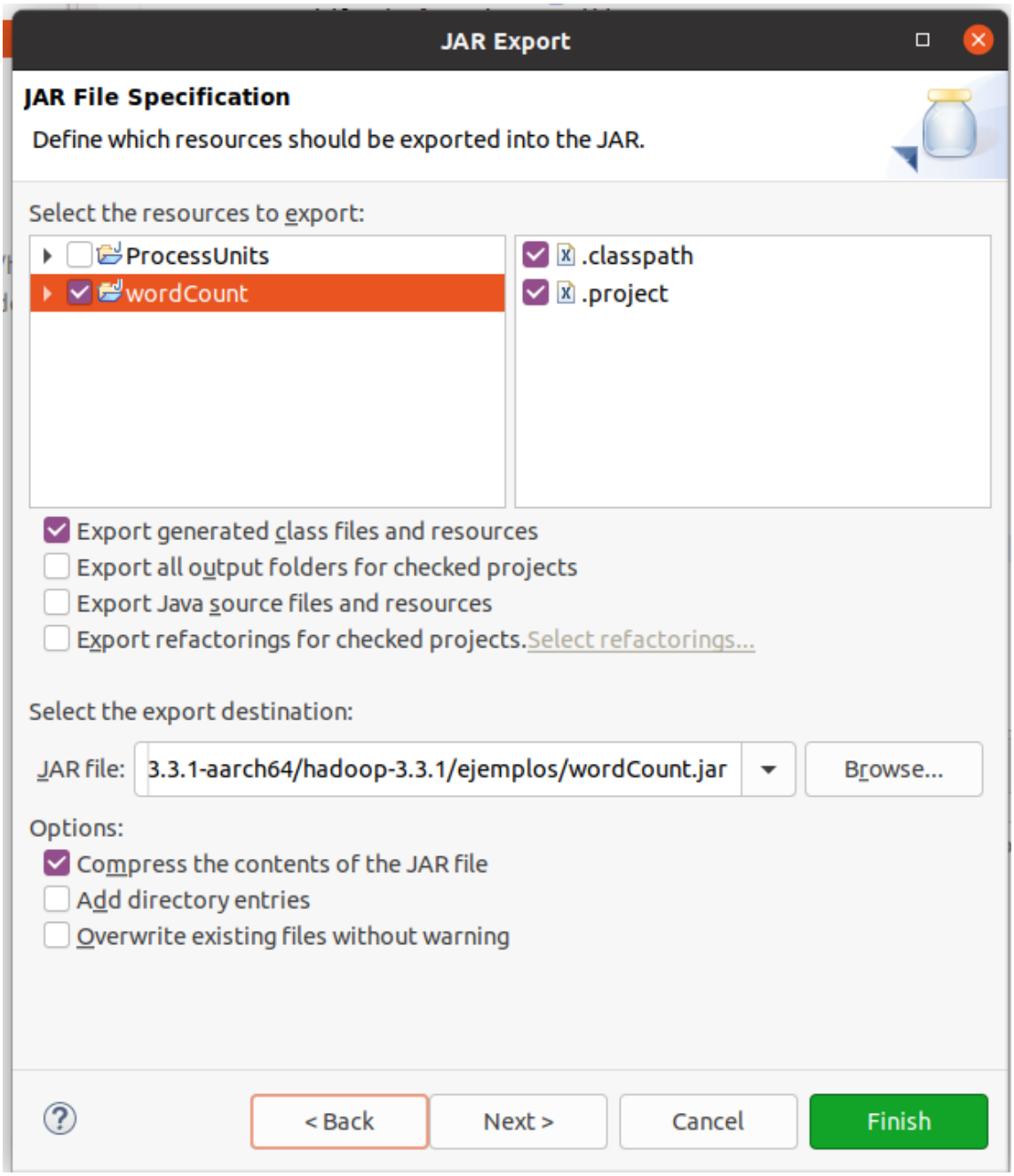
Una vez construido el proyecto, crear el JAR. Desde el menú contextual del proyecto seleccionar Export



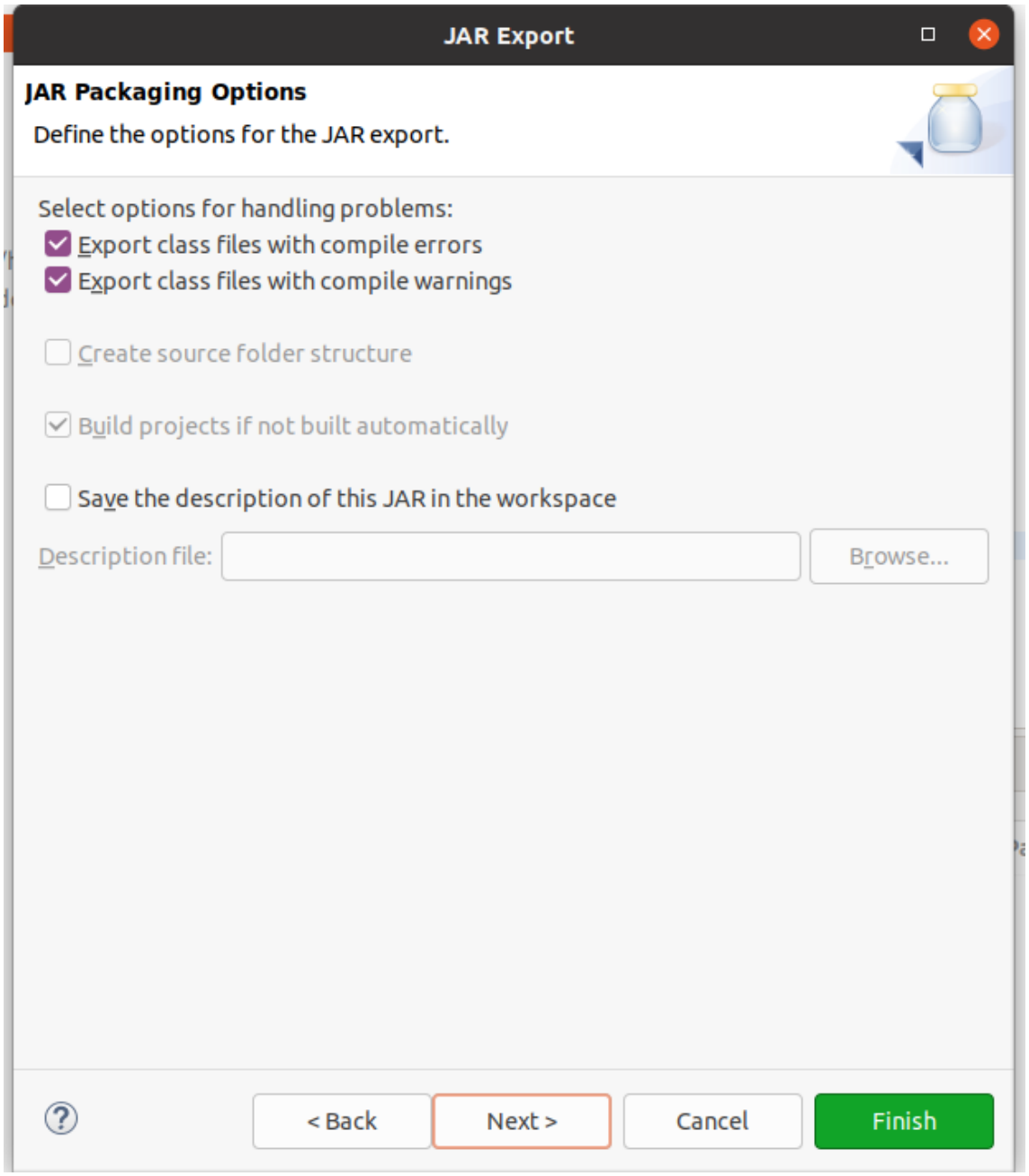
Seleccionar el tipo de archivo Java/Jar file y pulsar Next



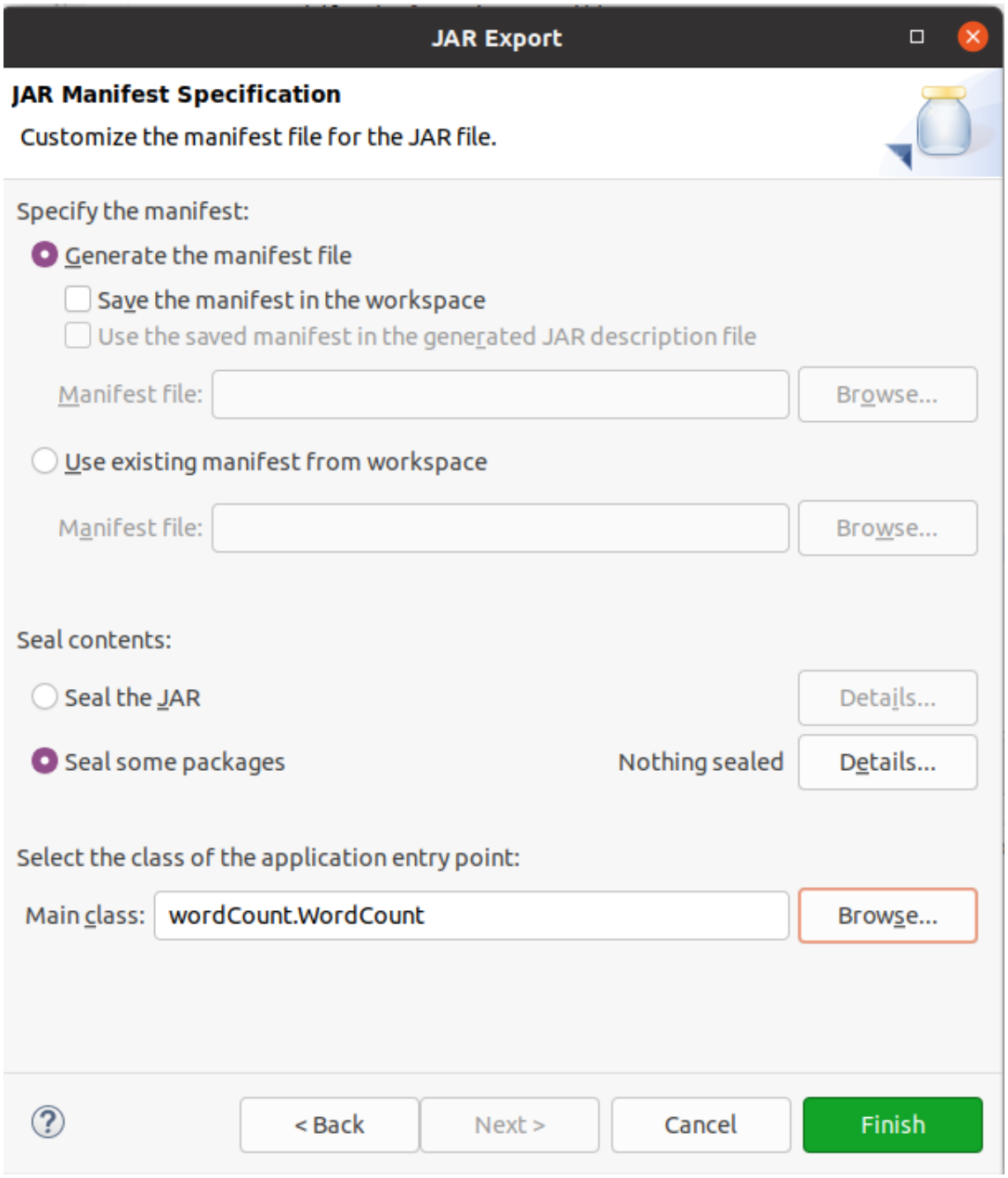
Indicar la ruta del archivo JAR, en el ejemplo, $HADOOP_HOME/ejemplos/wordCount.jar y pulsar Next



Dejar las opciones por defecto para el paquete Jar y pulsar Next



Indicar la clase principal de la aplicación, en este caso WordCount



En la carpeta indicada está el paquete JAR

5.5. Ejecutar WordCount

Ejecutamos la aplicación WordCount de la misma manera que ejecutamos en los ejemplos anteriores

Crear una carpeta y copiar los archivos base para el proceso. En el caso de que el namedata no tenga la carpeta user/<usuario> crearla previamente (punto 3.6)

```
$ hdfs dfs -mkdir wc_input

$ hdfs dfs -put $HADOOP_HOME/*.txt wc_input
```

Ejecutar el proceso wordCount de ejemplo

```
$ yarn jar $HADOOP_HOME/ejemplos/wordCount.jar wc_input wc_output
```

Verificar el resultado

```
  $ hdfs dfs -cat wc_output/*
```

5.6. Ejemplo Lectura_Medias

Repetir los pasos con el código siguiente, para obtener los datos máximos por año. Utilizar como datos de ejemplo LecturasEjemplo.txt

```
import java.util.*;

import java.io.IOException;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapred.*;




public class ProcessUnits

{

        //Mapper class

        public static class E_EMapper extends MapReduceBase implements
Mapper<LongWritable ,/*Input key Type */

        Text, /*Input value Type*/

        Text, /*Output key Type*/

        IntWritable> /*Output value Type*/

        { //Map function

                public void map(LongWritable key, Text value,

                OutputCollector<Text, IntWritable> output,

                Reporter reporter) throws IOException

                {

                        String line = value.toString();
```

```
                        String lasttoken = null;
                        StringTokenizer s = new StringTokenizer(line,"\t");
                        String year = s.nextToken();
                        while(s.hasMoreTokens()){lasttoken=s.nextToken();}
                        int avgprice = Integer.parseInt(lasttoken);
                        output.collect(new Text(year), new
IntWritable(avgprice));
                }
        }

//Reducer class
        public static class E_EReduce extends MapReduceBase implements
Reducer< Text, IntWritable, Text, IntWritable >
        { //Reduce function
                public void reduce(
                Text key,
                Iterator <IntWritable> values,
                OutputCollector<Text, IntWritable> output,
                Reporter reporter) throws IOException
                {
                        int maxavg=30;
                        int val=Integer.MIN_VALUE;
                        while (values.hasNext())
                                {
                                if((val=values.next().get())>maxavg)
                                        {
                                        output.collect(key, new
IntWritable(val));
                                        }
                                }
                }
        }
//Main function
```

```
public static void main(String args[])throws Exception
        {
                JobConf conf = new JobConf(ProcessUnits.class);
                conf.setJobName("max_lecturas");
                conf.setOutputKeyClass(Text.class);
                conf.setOutputValueClass(IntWritable.class);
                conf.setMapperClass(E_EMapper.class);
                conf.setCombinerClass(E_EReduce.class);
                conf.setReducerClass(E_EReduce.class);
                conf.setInputFormat(TextInputFormat.class);
                conf.setOutputFormat(TextOutputFormat.class);
                FileInputFormat.setInputPaths(conf, new Path(args[0]));
                FileOutputFormat.setOutputPath(conf, new Path(args[1]));
                JobClient.runJob(conf);
        }
}
```