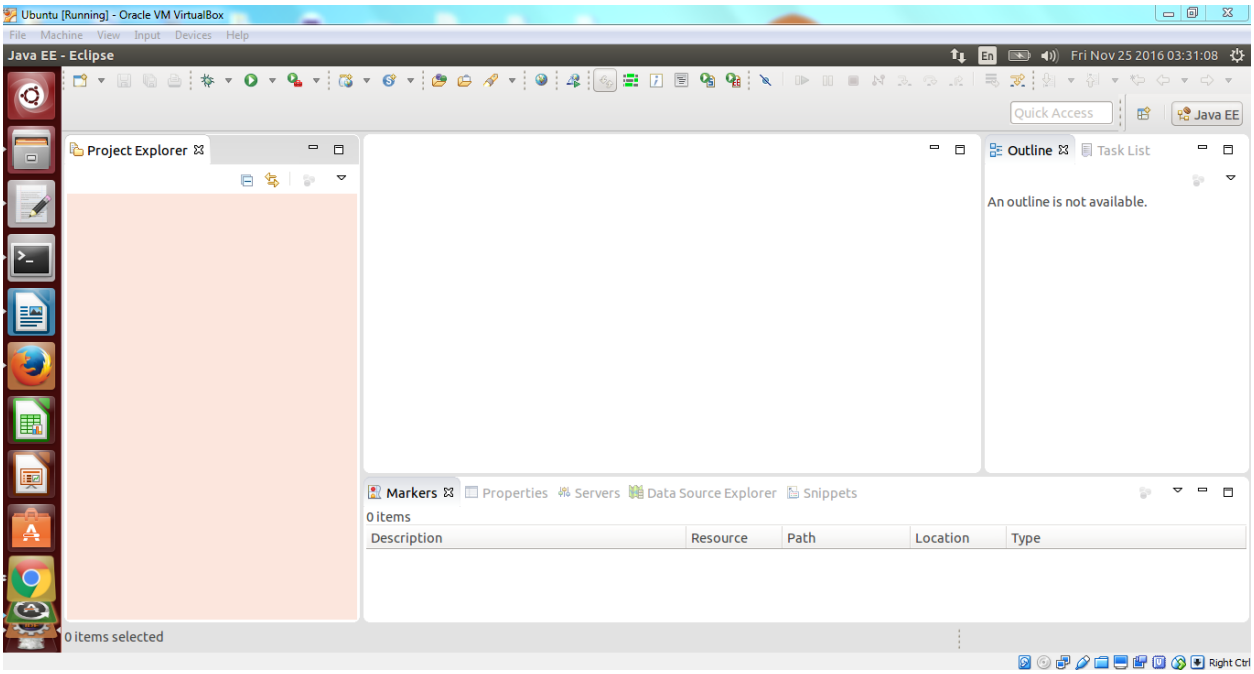
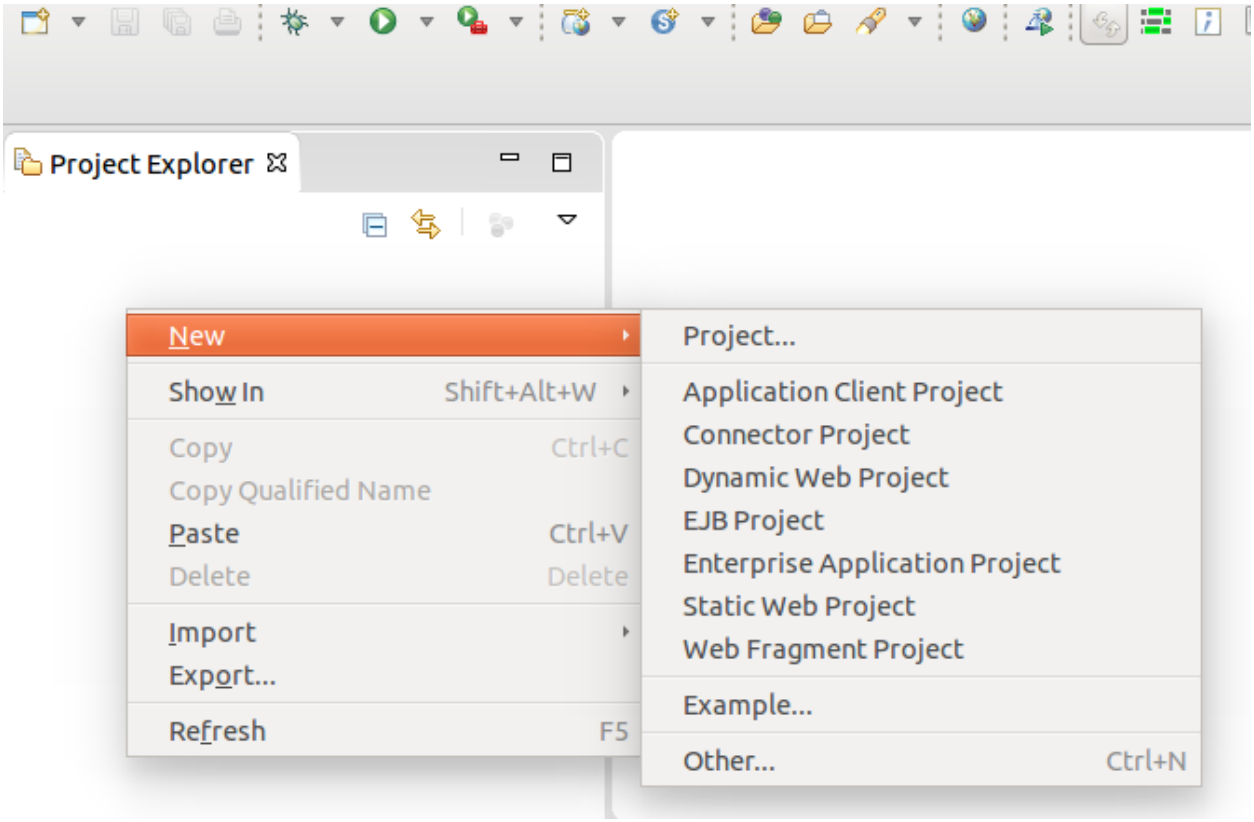
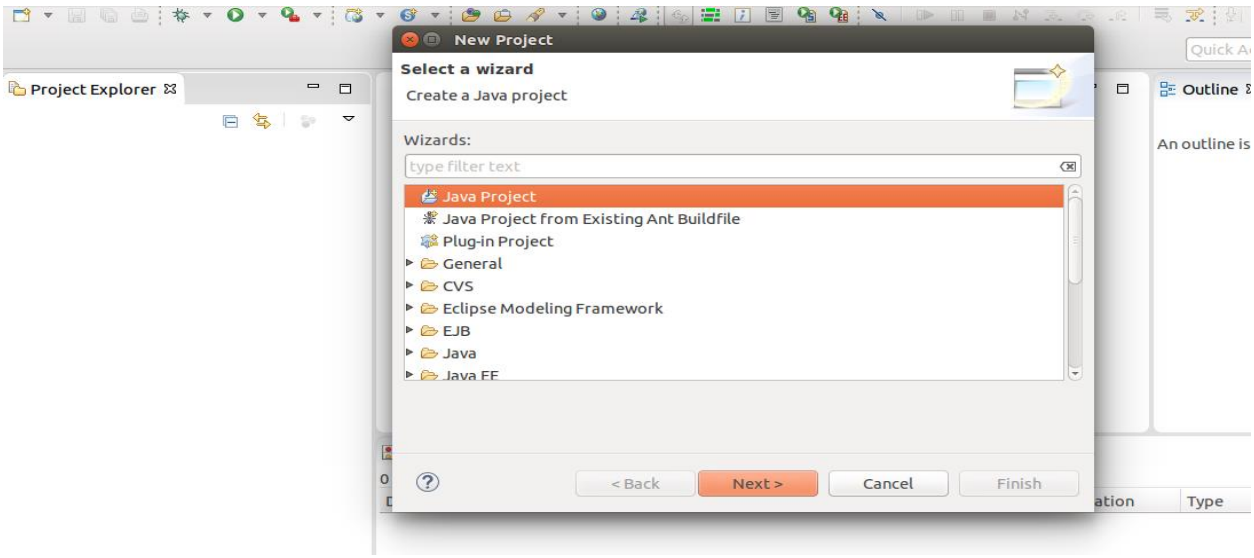
Start eclipse, once it is started.



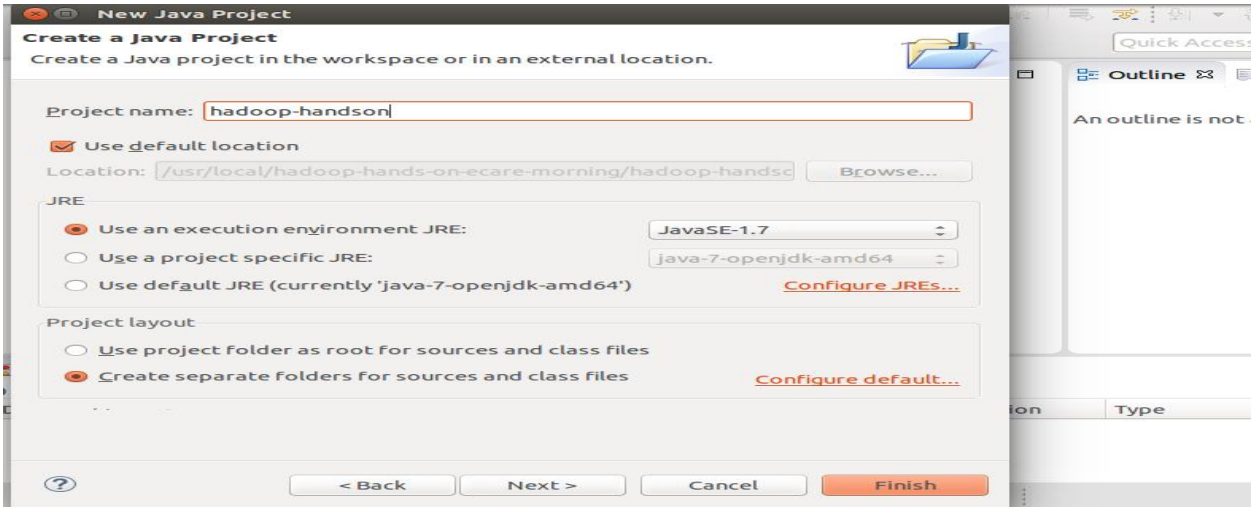
Create java project. Right click on the project explorer→ select New→ project →java project.

Follow below screens to create project

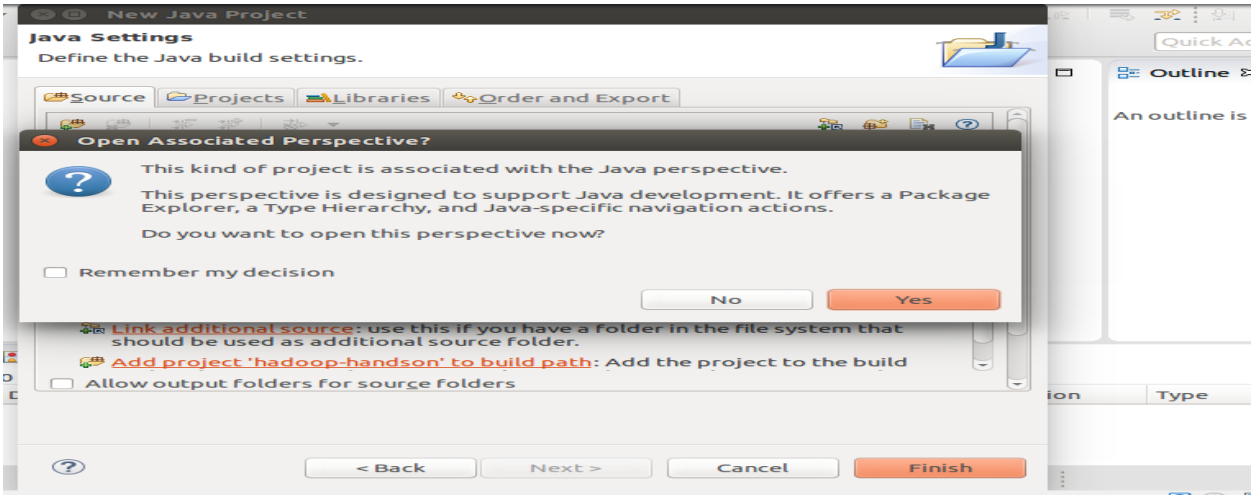




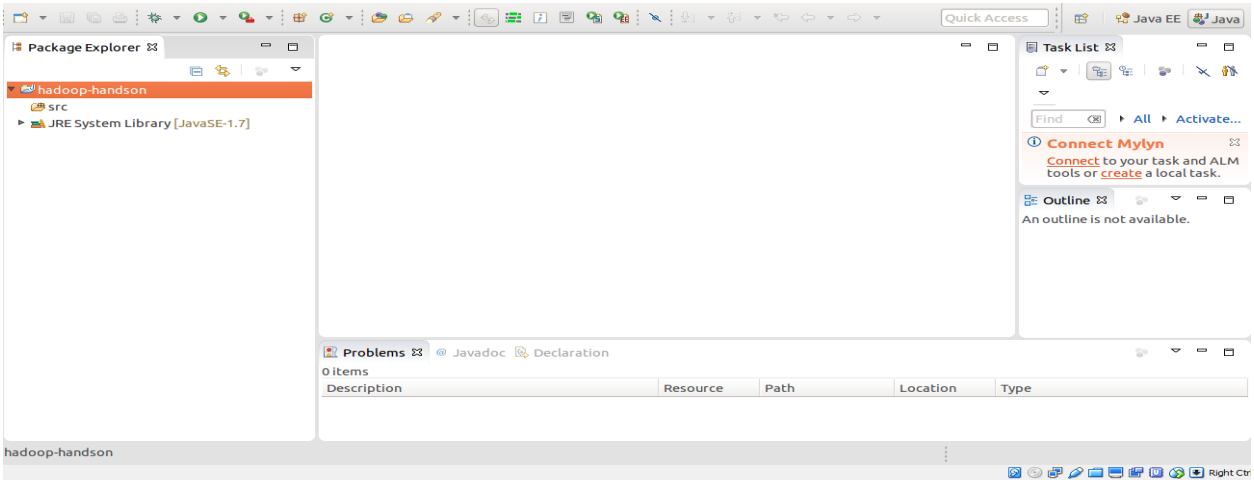
Give the project Name.. ex: mapreduce-hands-on and click on **finish** button



If It ask for prospective change—click on yes button.

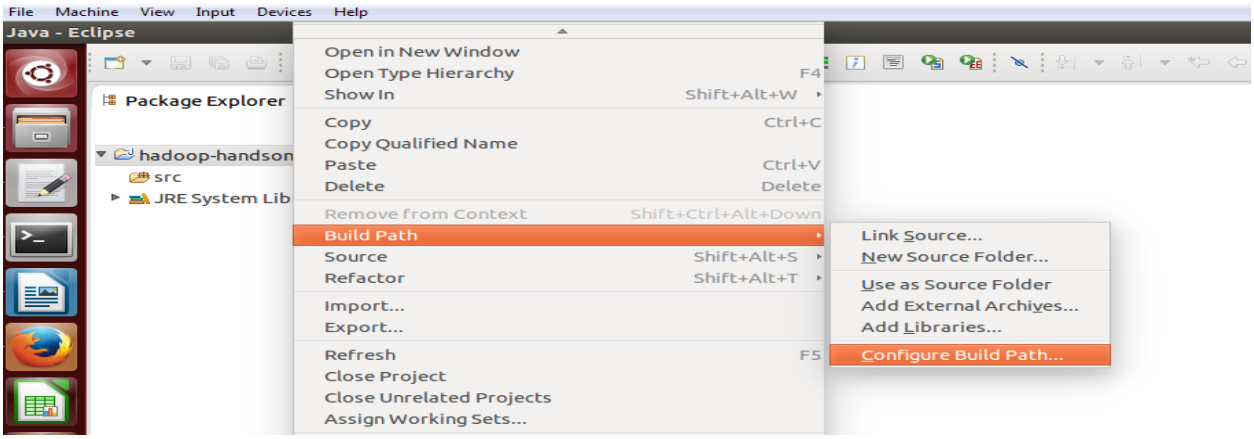


It will create new java-project on project Explorer

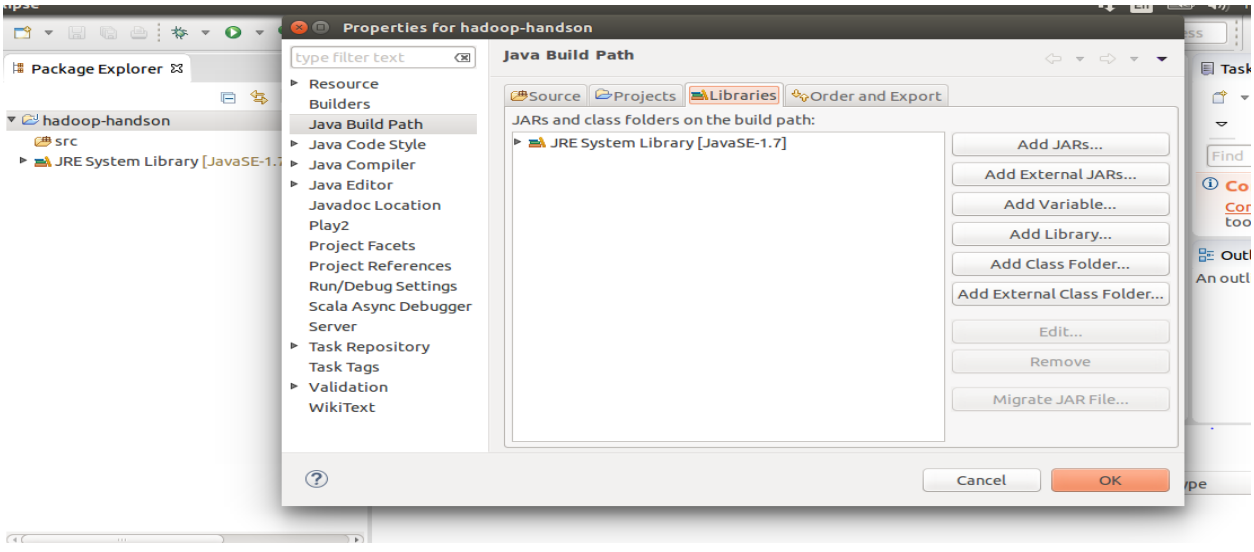


Set the Build Path:

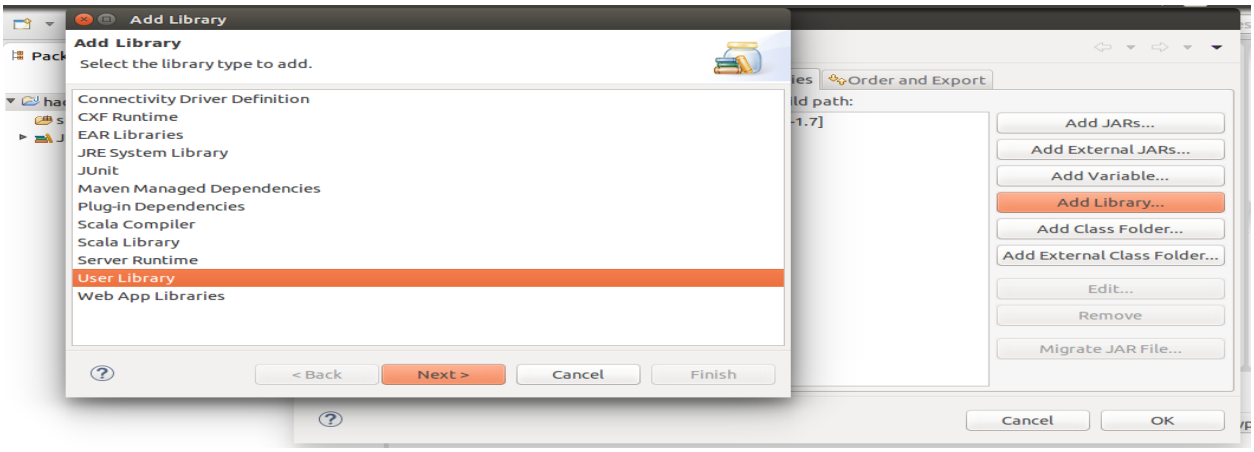
Right click on project→ select **Build path→configure Build path**.



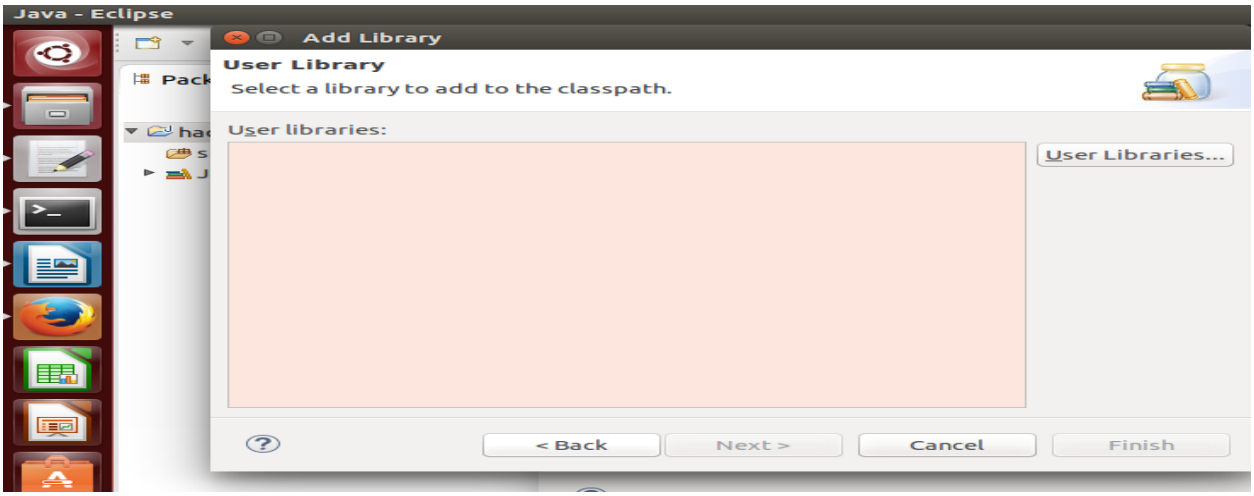
Click on "**Libraries**" from top pane and click on **Add library** from the right pane.



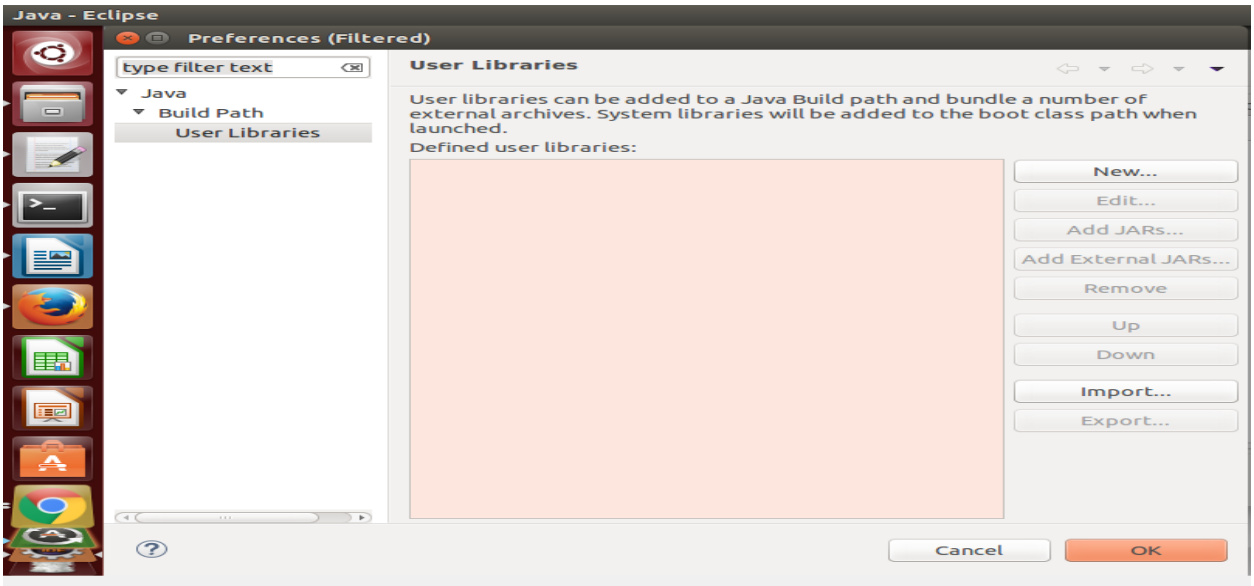
Select **user library** from left menu. See in below window and click next



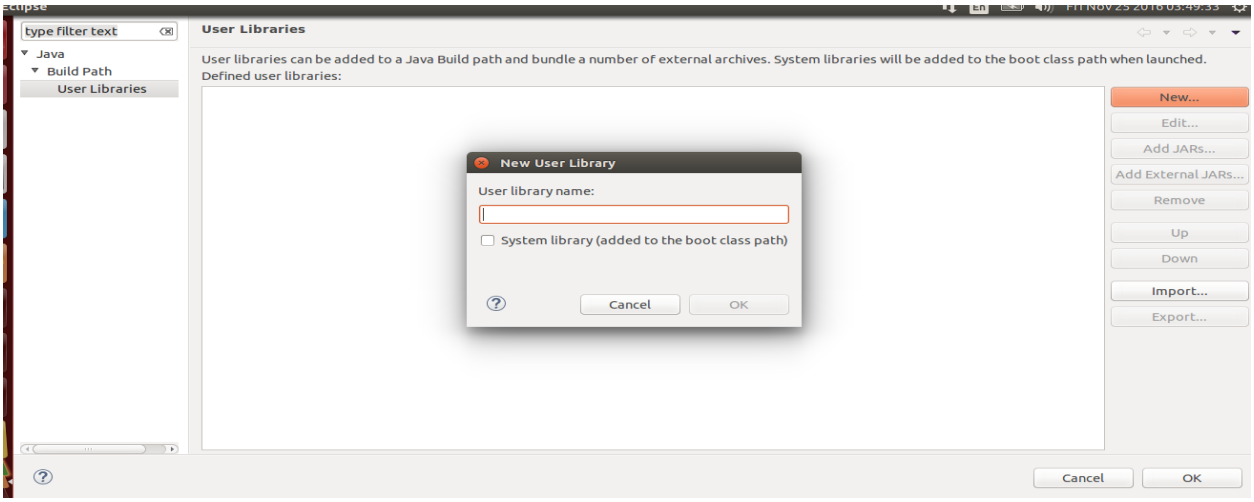
Click on **User-Libraries** from right pane

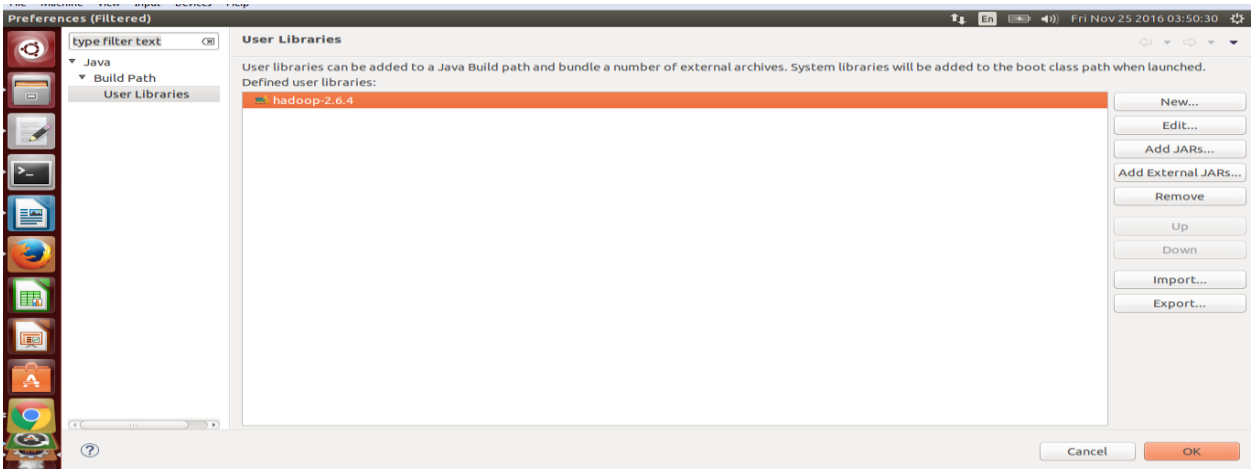


Click on new button from right pane for creating new userlibrary

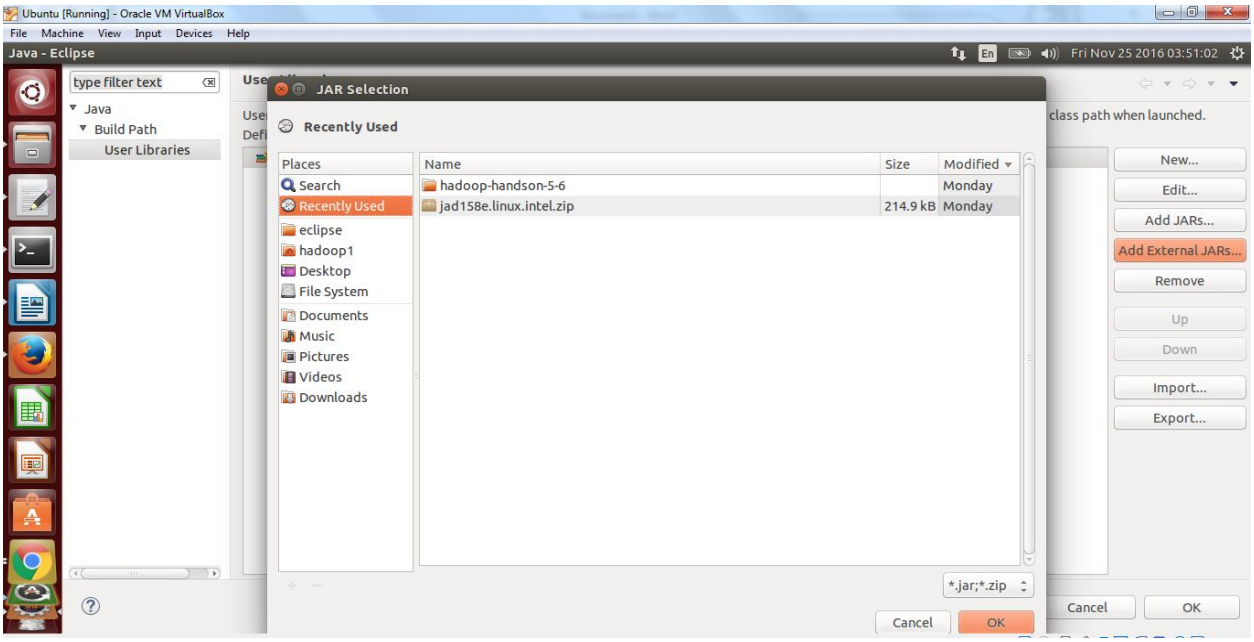


Give a name to user library and click on **OK**





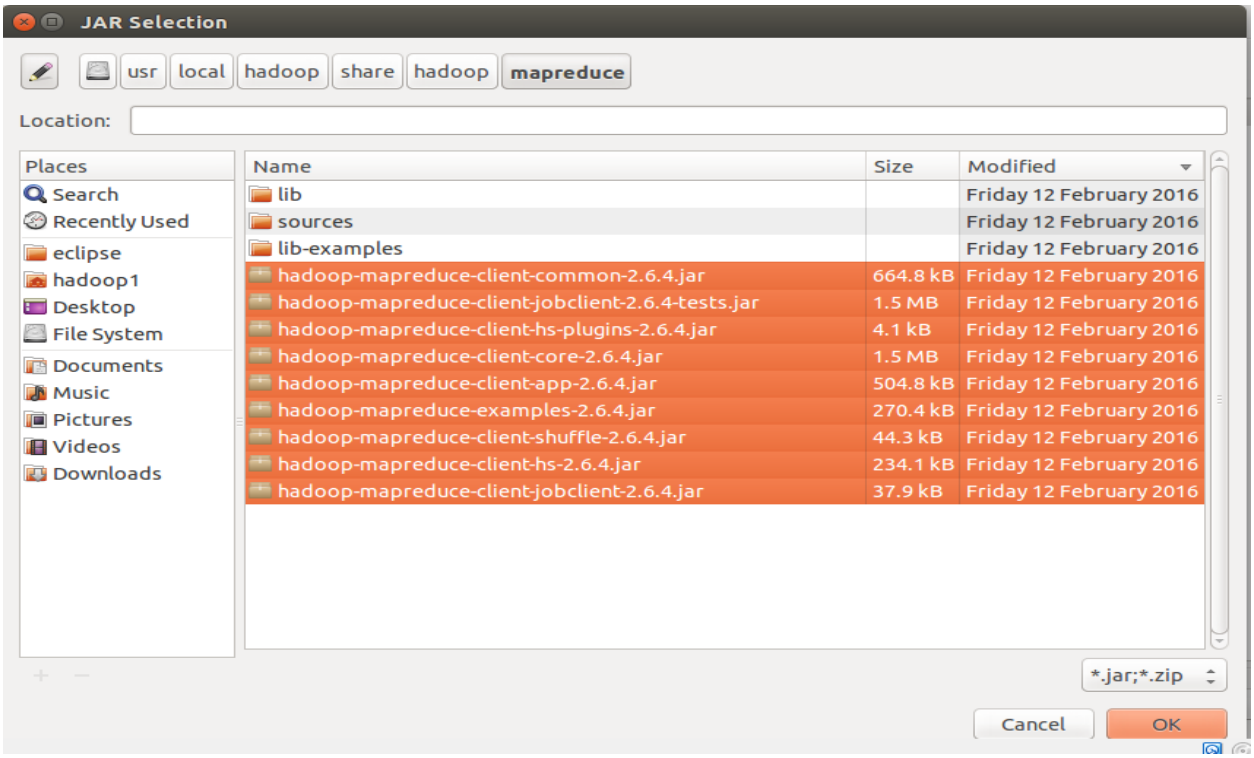
Click on Add external jars and it will display below window, please navigate go to Hadoop home dir from left pane.



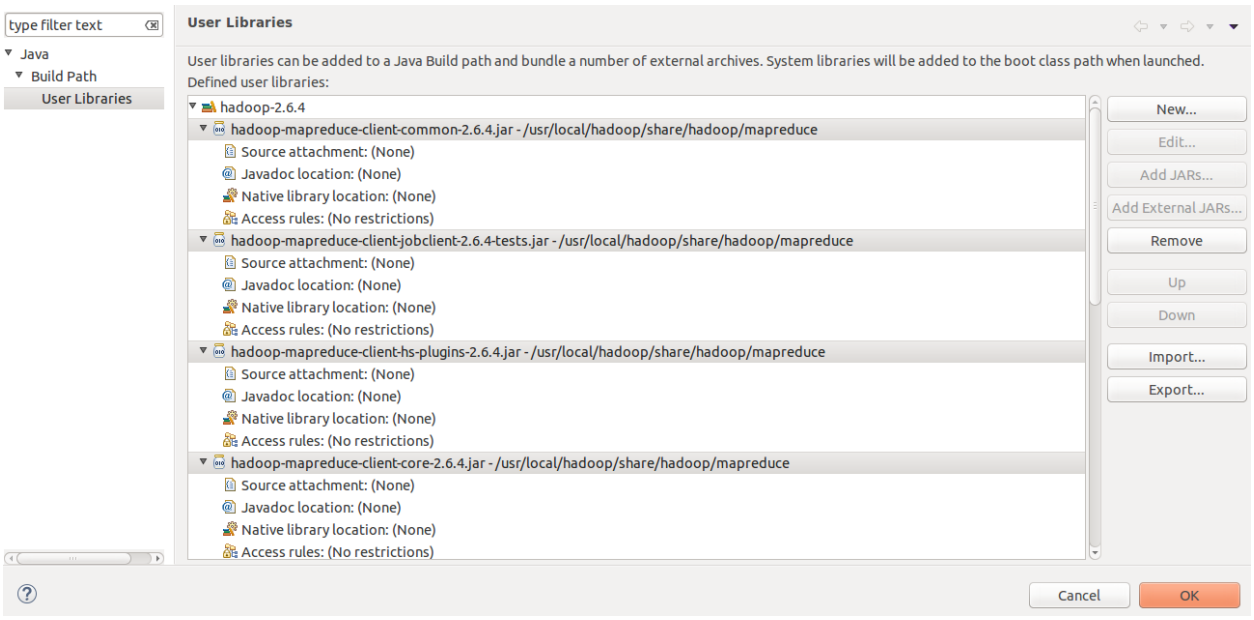
Go to Hadoop home dir (/usr/local/Hadoop-env/Hadoop-2.6.4/share/Hadoop/mapreduce) select all the jar..

Note: to run mapreduce, we do not need all the jar files, but it's good to add all the jar file for practice purpose.. But in real-time we will select only required jar file.

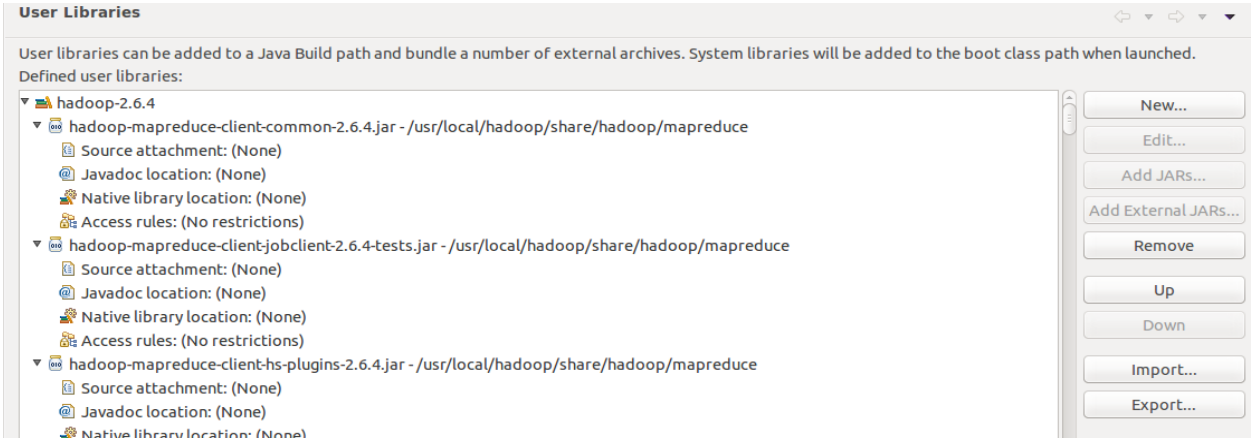
Adding MapReduce jar (/usr/local/Hadoop-env/Hadoop-2.6.4/share/Hadoop/mapreduce)



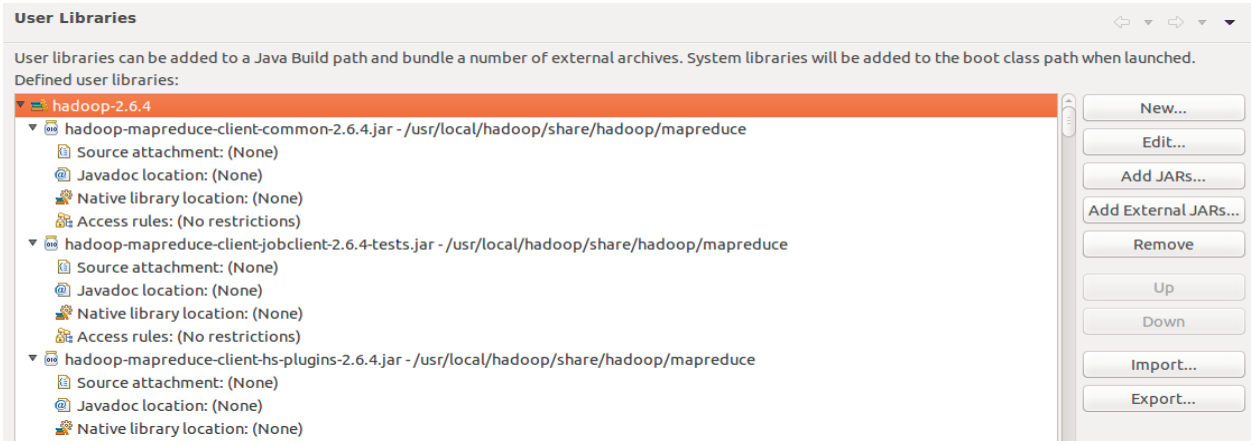
Click ok..



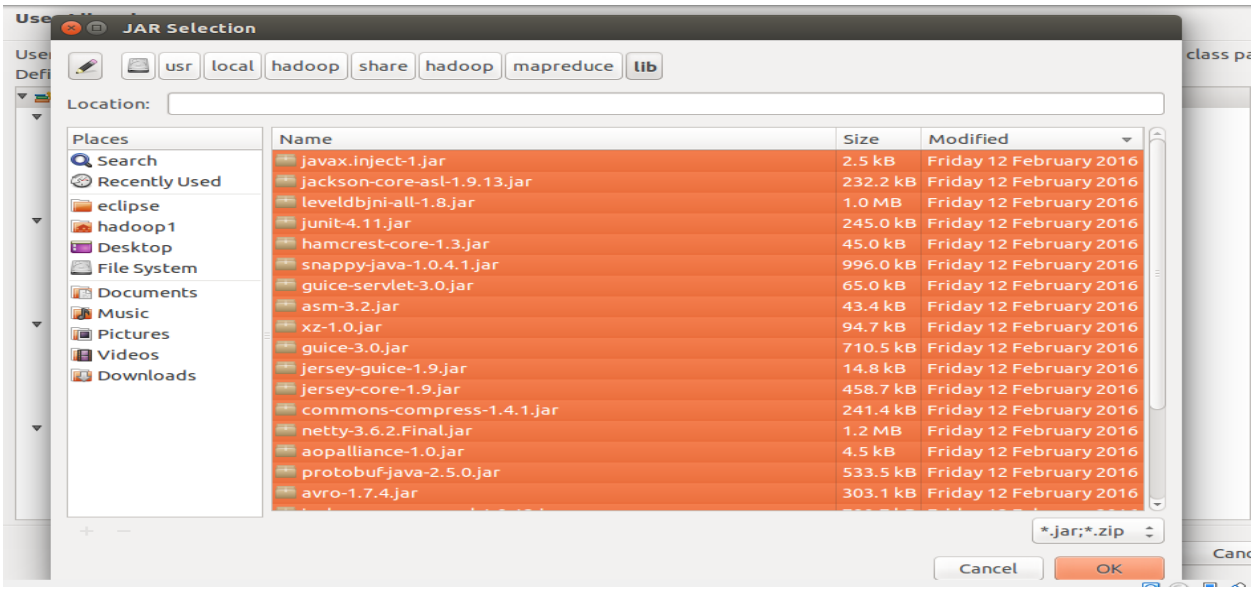
If the add external is disable, it means we did not selected the user library folder so, select created user library.



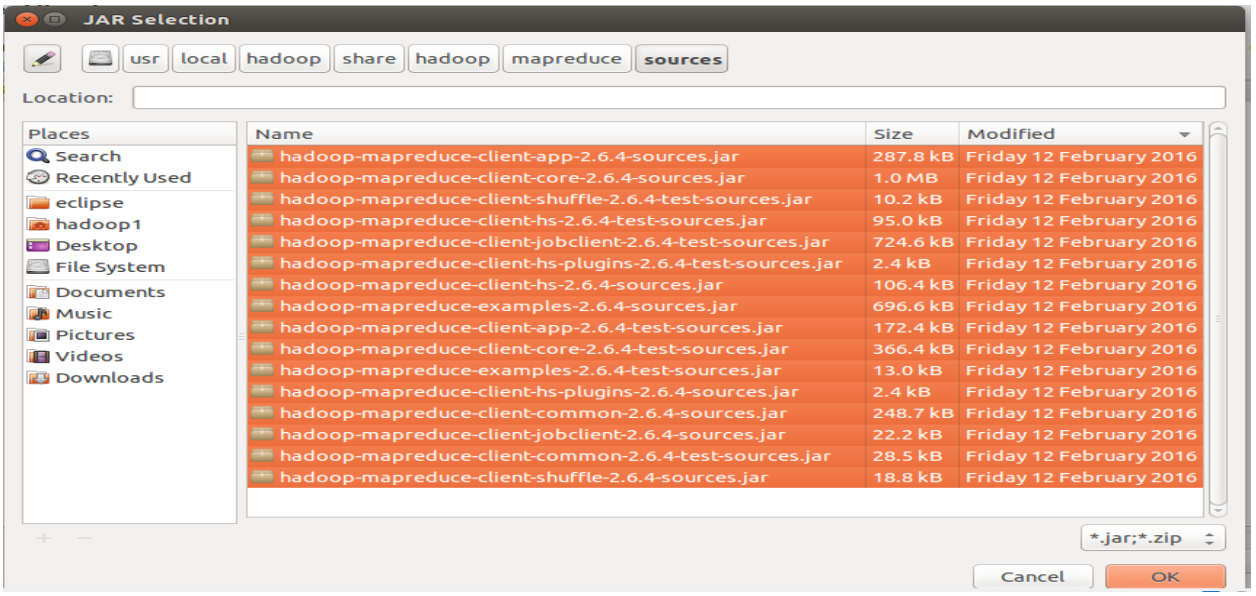
When you select the user library folder, the add external will enabled:



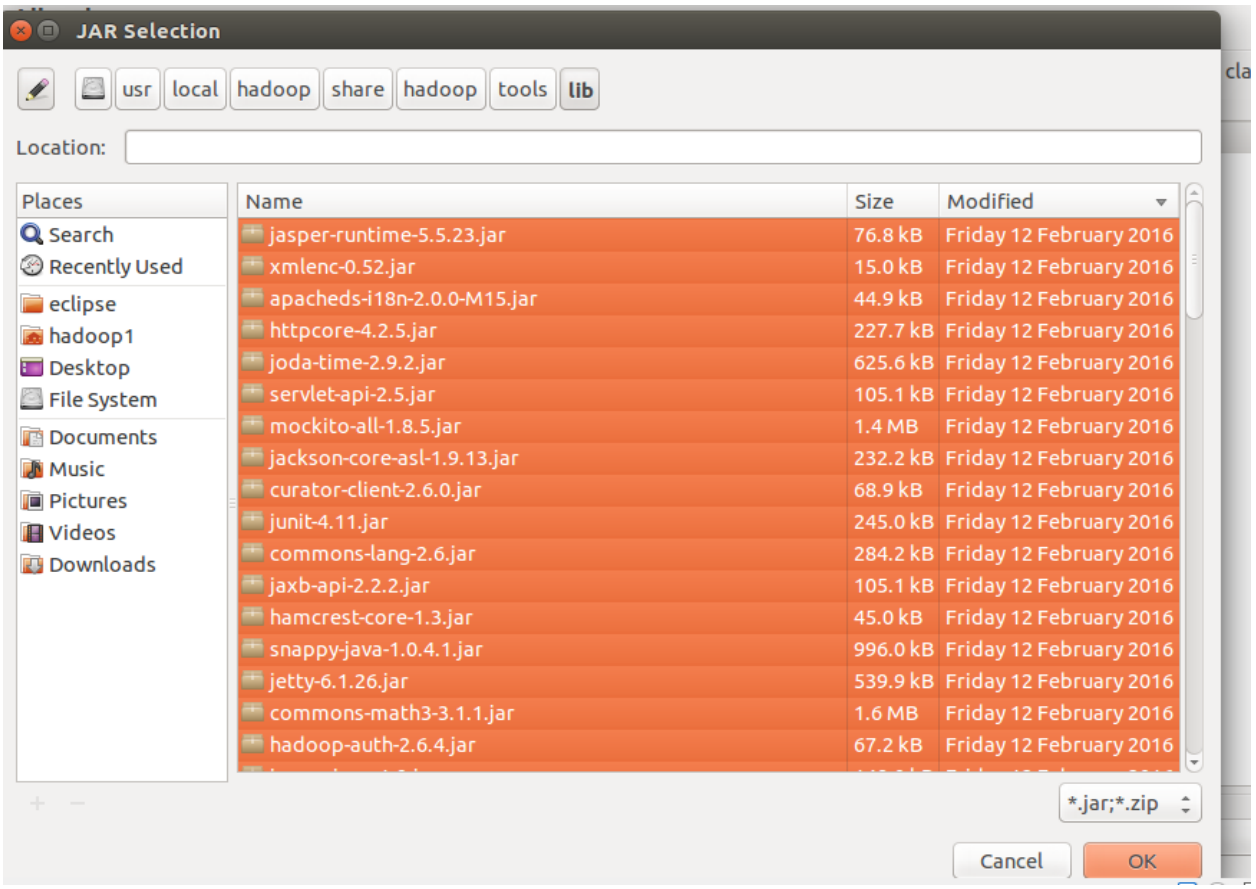
Again need to select few more jar file. So select user library name (should select) and click on add external jars (/usr/local/Hadoop-env/Hadoop-2.6.4/share/Hadoop/mapreduce/lib) Click OK



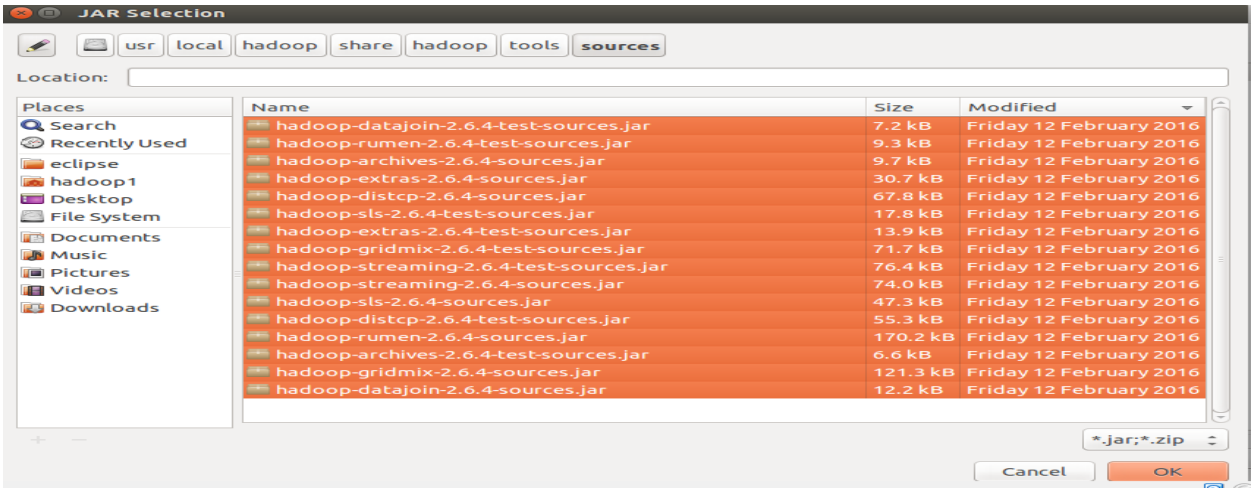
Repeat same step for add all the jars from Hadoop(/usr/local/Hadoop-env/Hadoop-2.6.4/share/Hadoop/mapreduce/source) and click OK



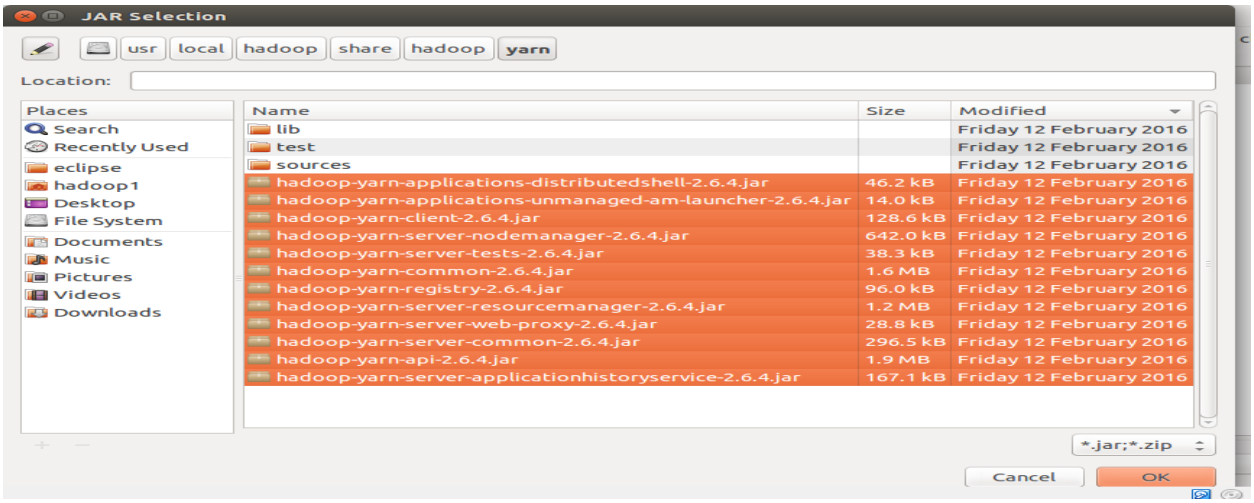
Repeat the same steps for other dirs(/usr/local/Hadoop-env/Hadoop-2.6.4/share/Hadoop/tools/lib) click OK



Every time you need to select the user library and add external jars file(/usr/local/Hadoop-env/Hadoop-2.6.4/share/Hadoop/sources) Click OK

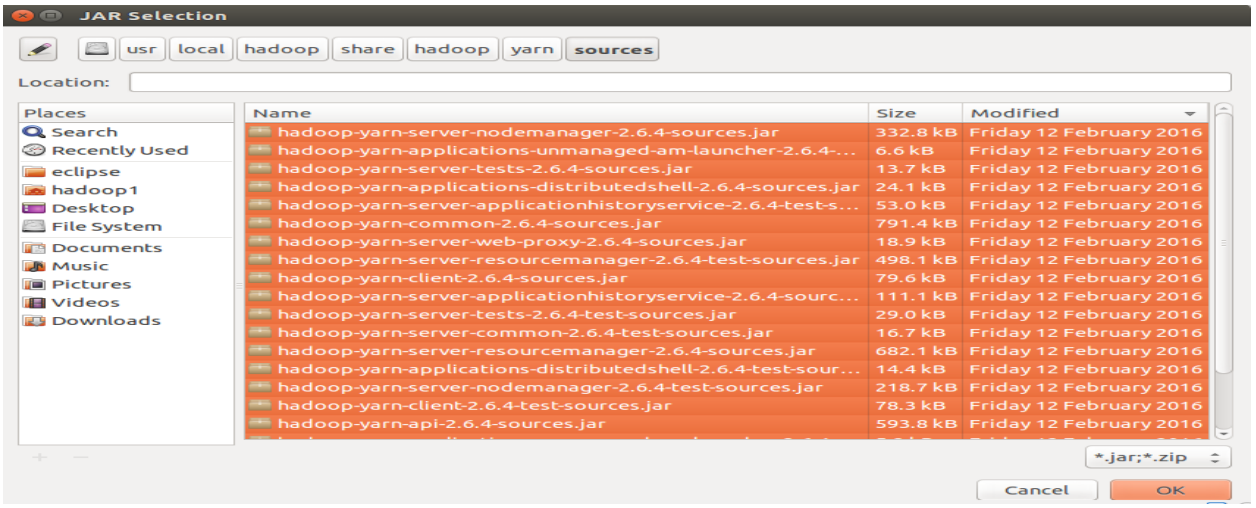


Selecting YARN related jars(/usr/local/Hadoop-env/Hadoop-2.6.4/share/Hadoop/yarn) click OK

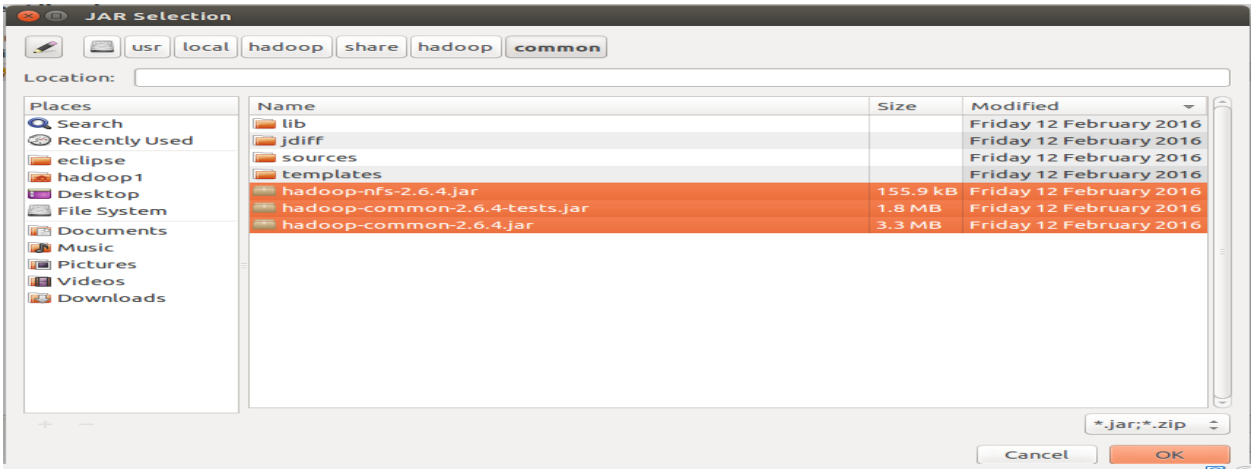


Select jar from Yarn lib folder (/usr/local/Hadoop-env/Hadoop-2.6.4/share/Hadoop/yarn/lib). Click OK

Yarn source folder: (/usr/local/Hadoop-env/Hadoop-2.6.4/share/Hadoop/yarn/sources).click OK

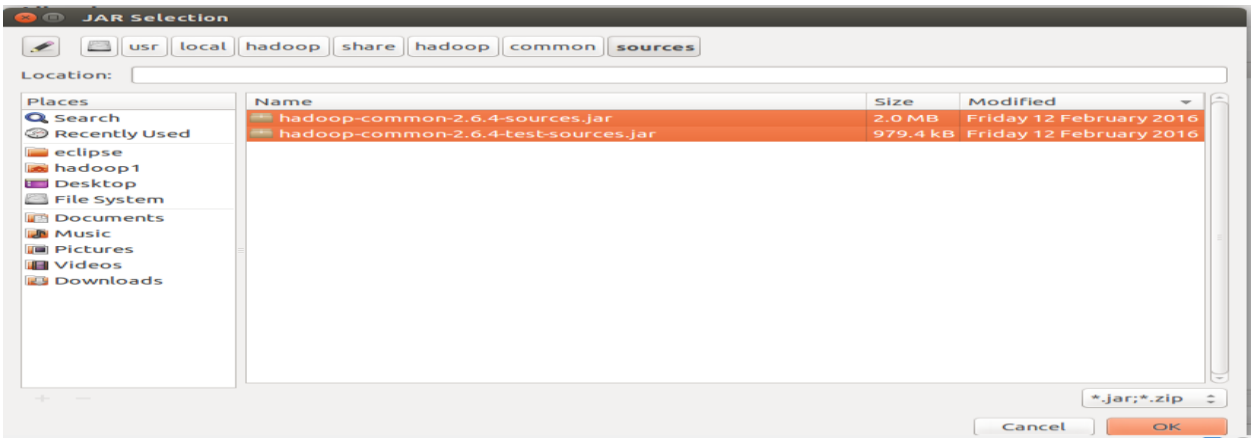


Hadoop-Common folder (/usr/local/Hadoop-env/Hadoop-2.6.4/share/Hadoop/Hadoop-common). Click OK

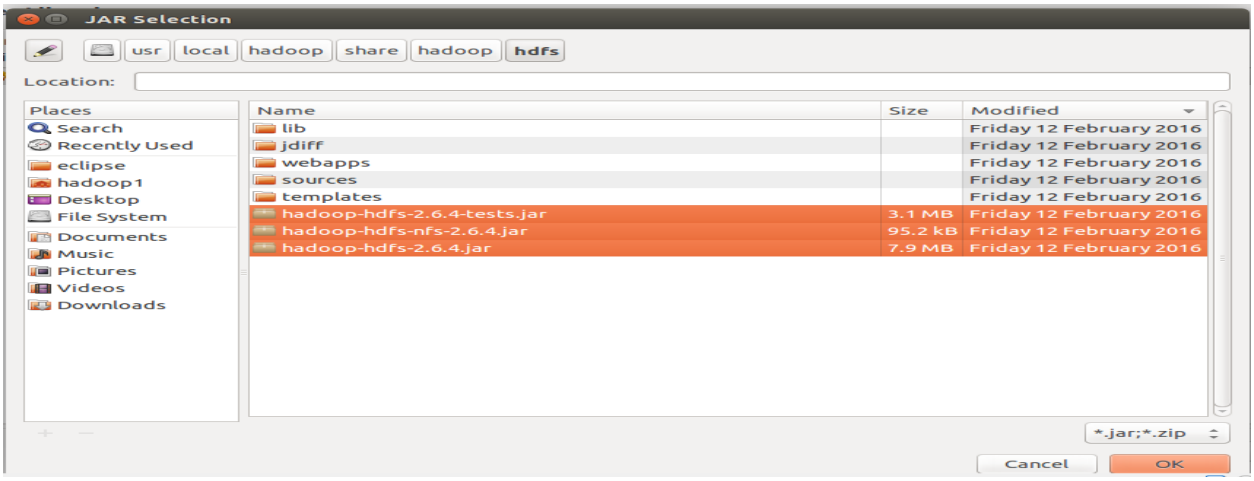


Hadoop-common-lib: (/usr/local/Hadoop-env/Hadoop-2.6.4/share/Hadoop/Hadoop-common/lib).click OK

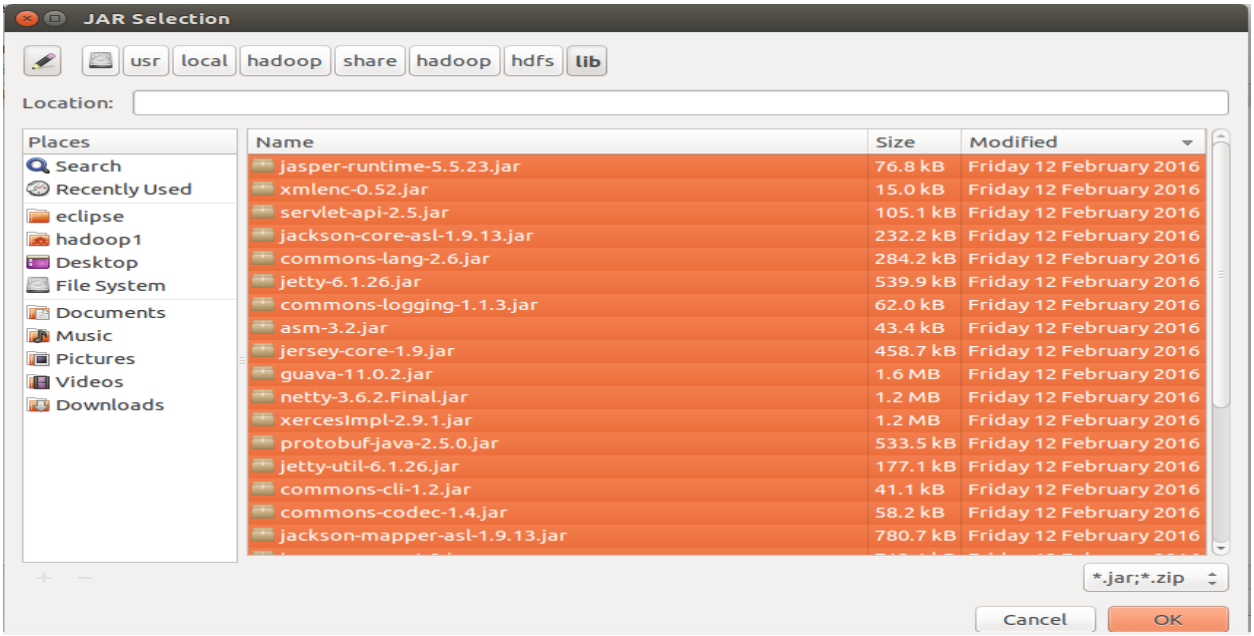
Common-source: (/usr/local/Hadoop-env/Hadoop-2.6.4/share/Hadoop/Hadoop-common/sources). Click OK

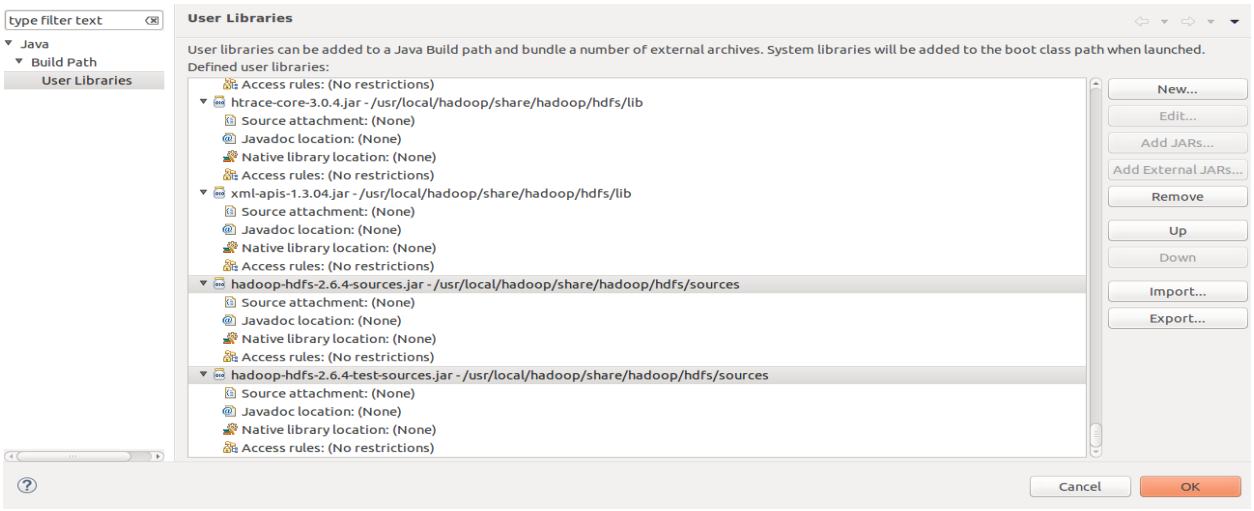


HDFS folder libs(/usr/local/Hadoop-env/Hadoop-2.6.4/share/Hadoop/hdfs). Click OK

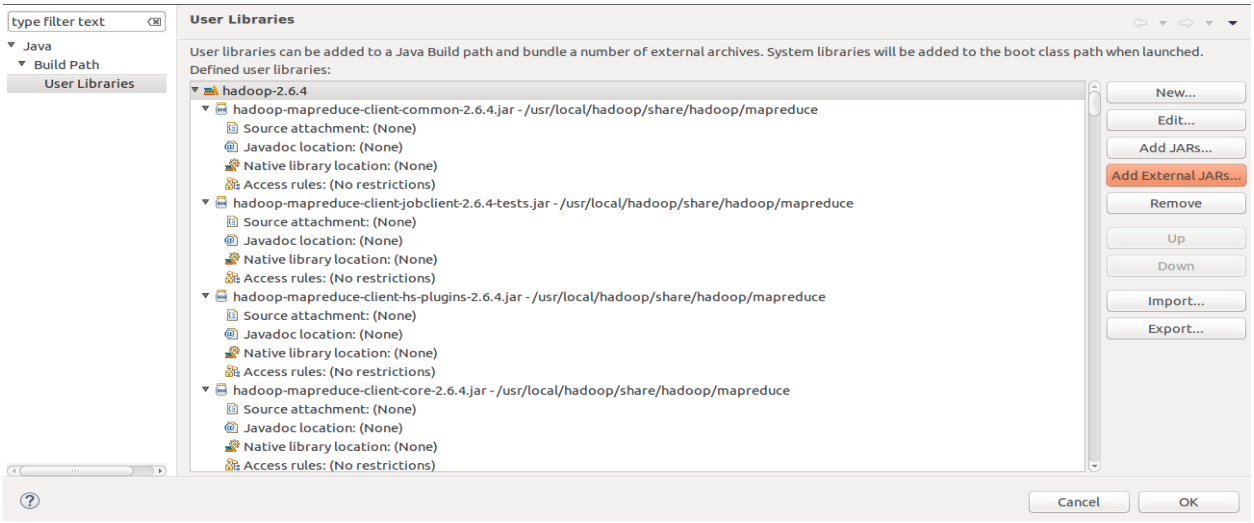


HDFS/lib: (/usr/local/Hadoop-env/Hadoop-2.6.4/share/Hadoop/HDFS/lib). Click OK

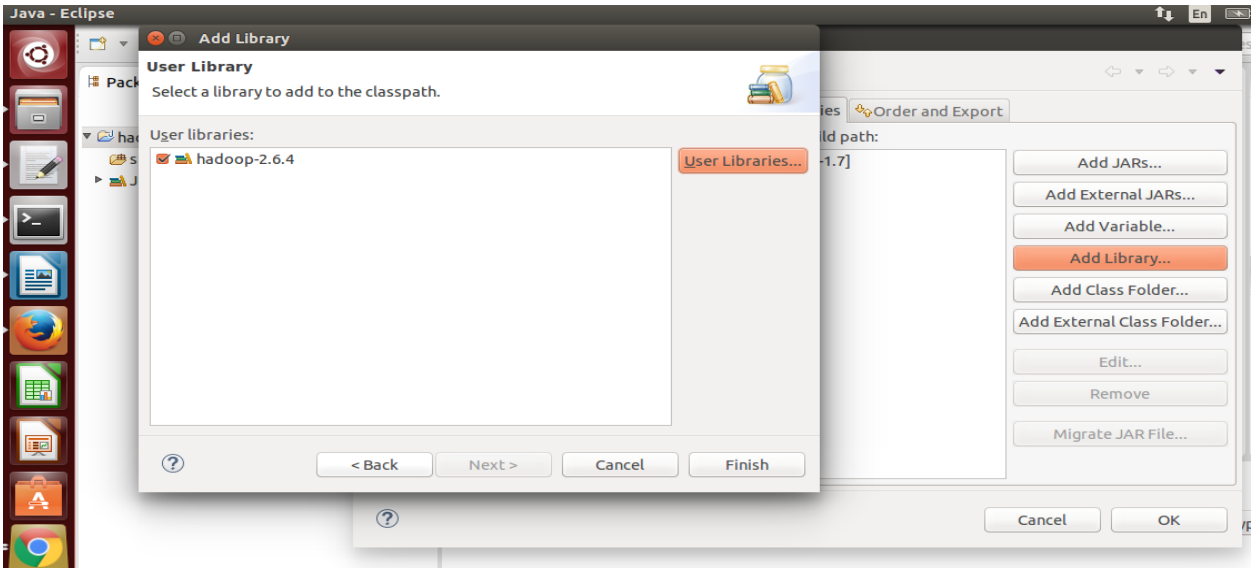




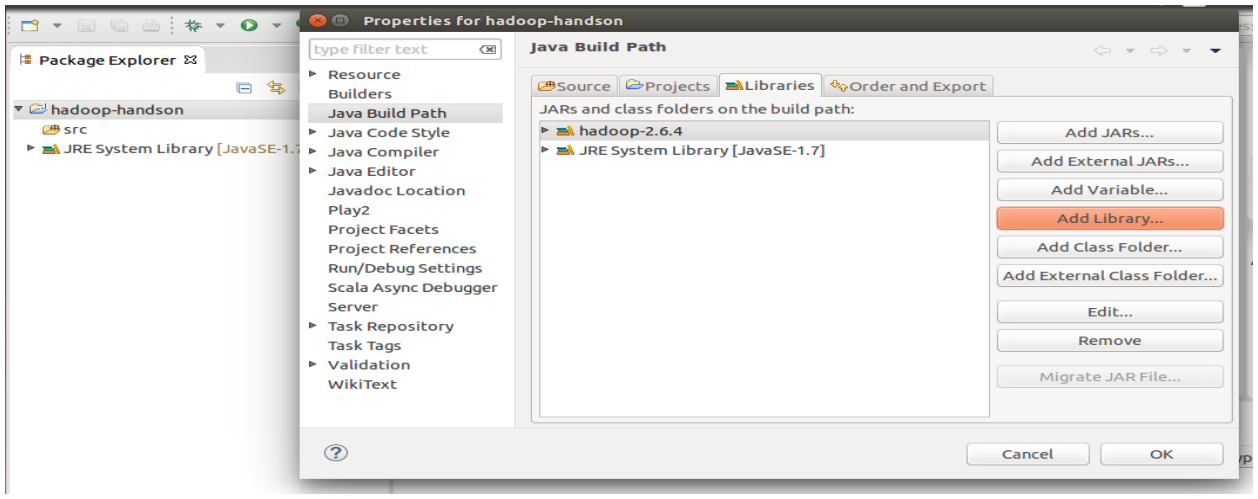
After adding all the libraries or jar files, click on OK



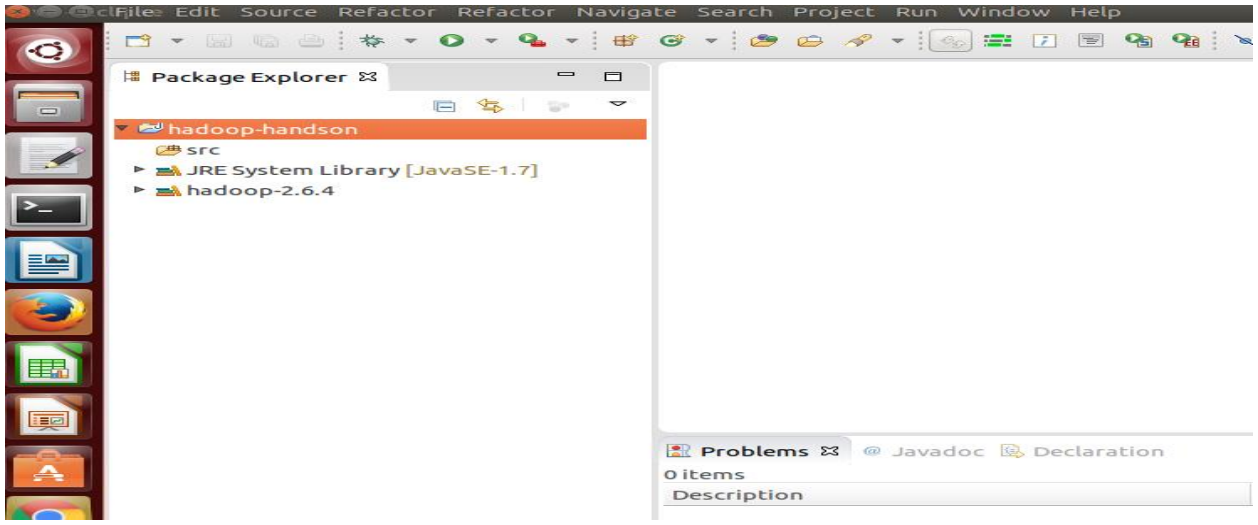
It will create library folder and click finish



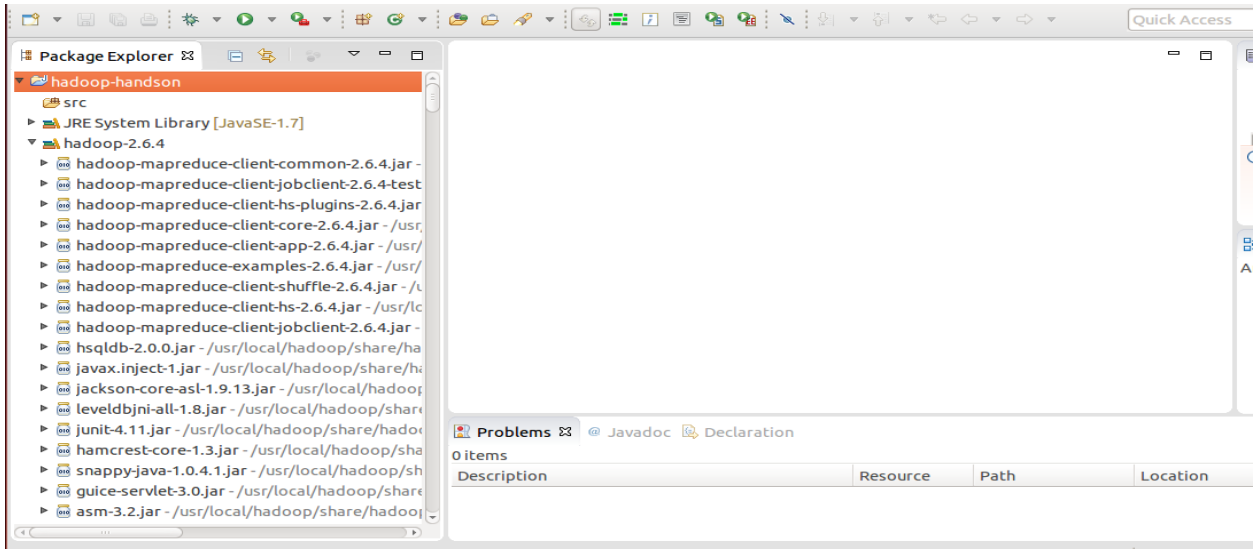
When clicked finish, it Is added to Build path libraries and click OK



After clicking OK, it is added to your project in eclipse

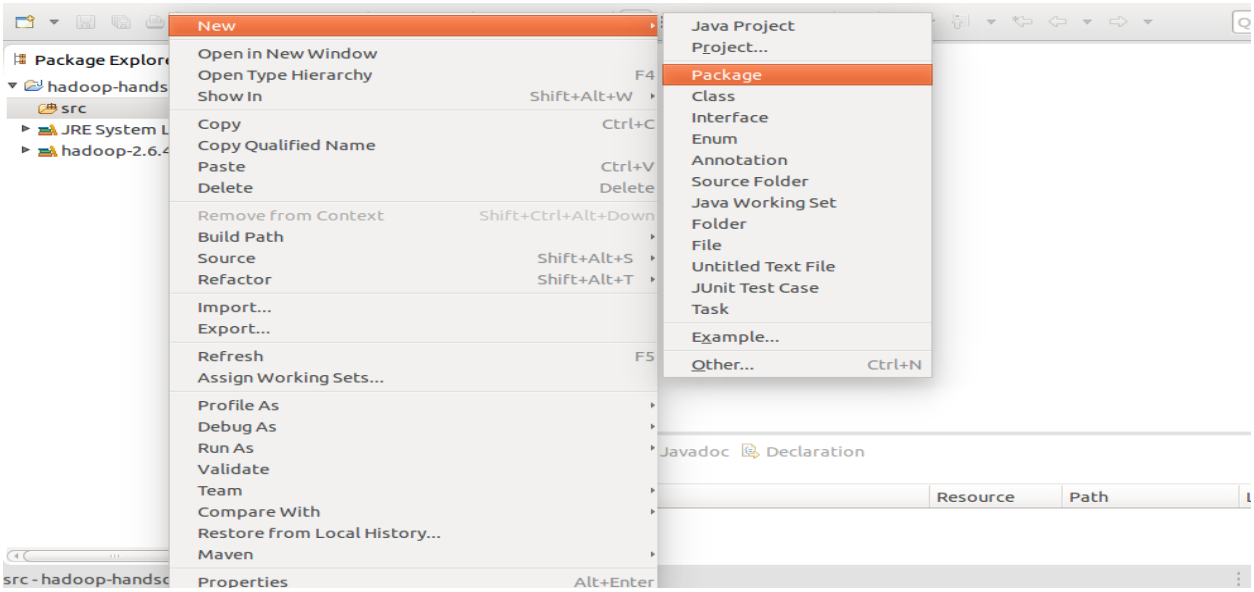


When we expand, the added user library, we can see all the JAR files

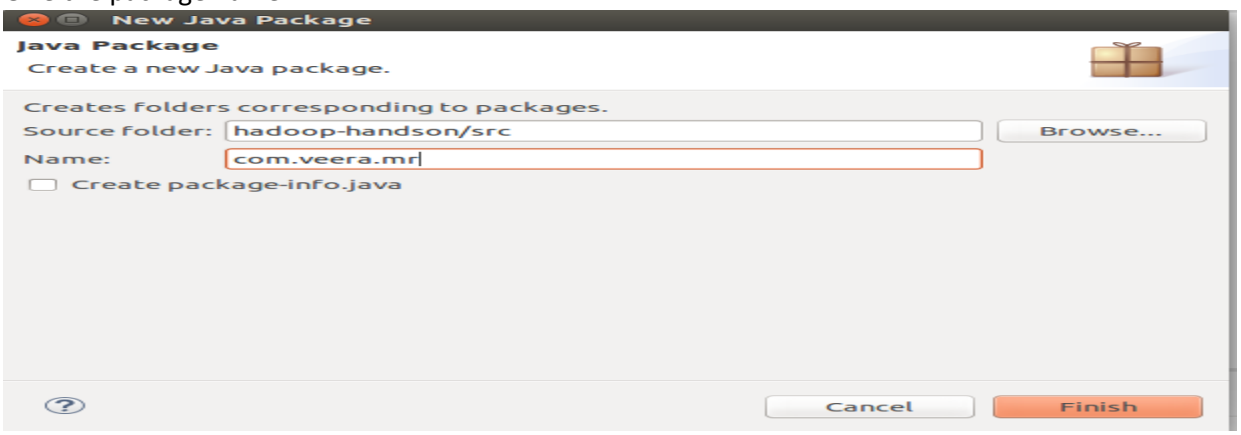


Now your project is ready to code for mapreduce programs.

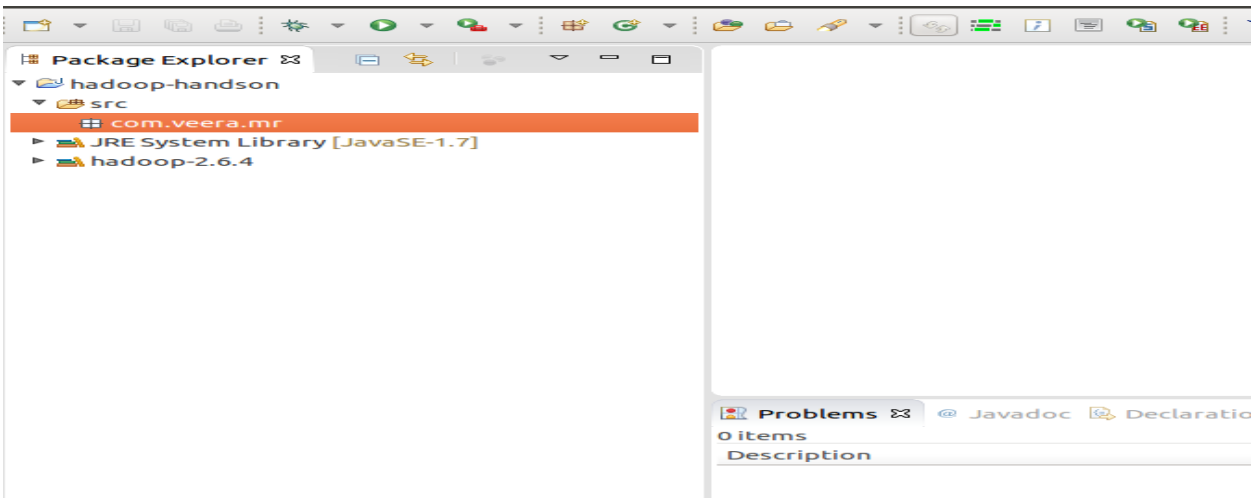
Example: Create package…



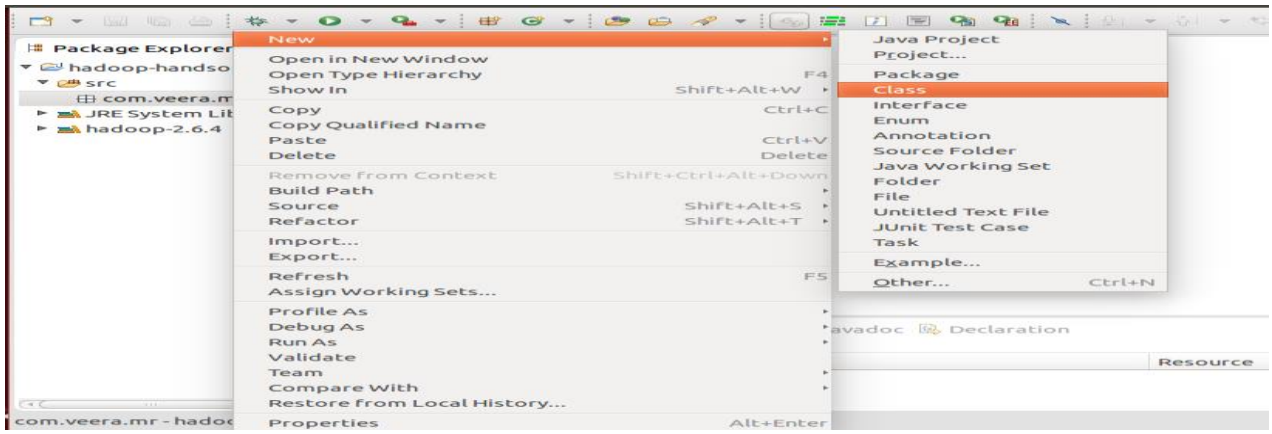
Give the package name:



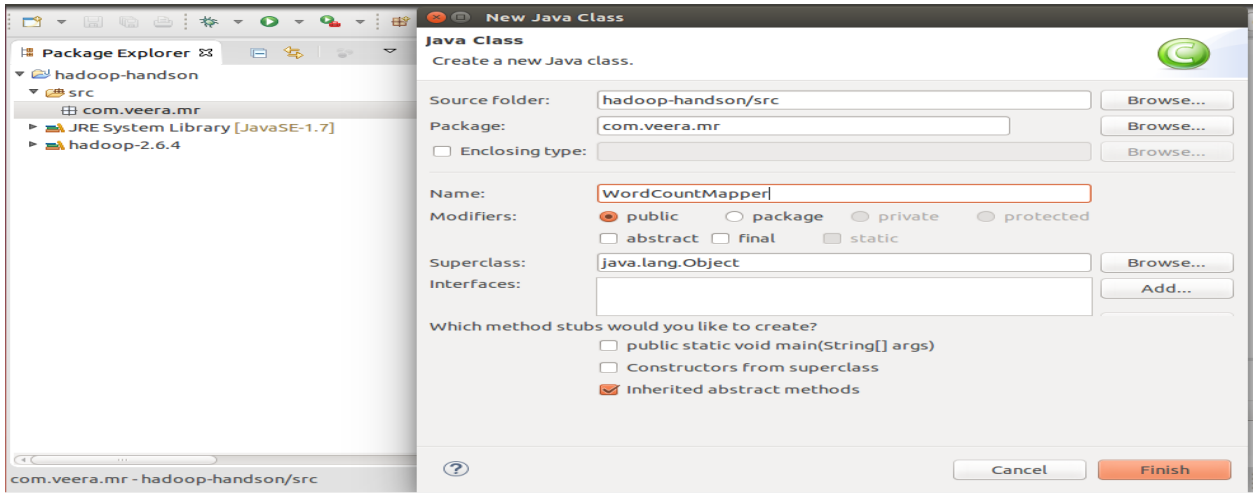
Package is created:



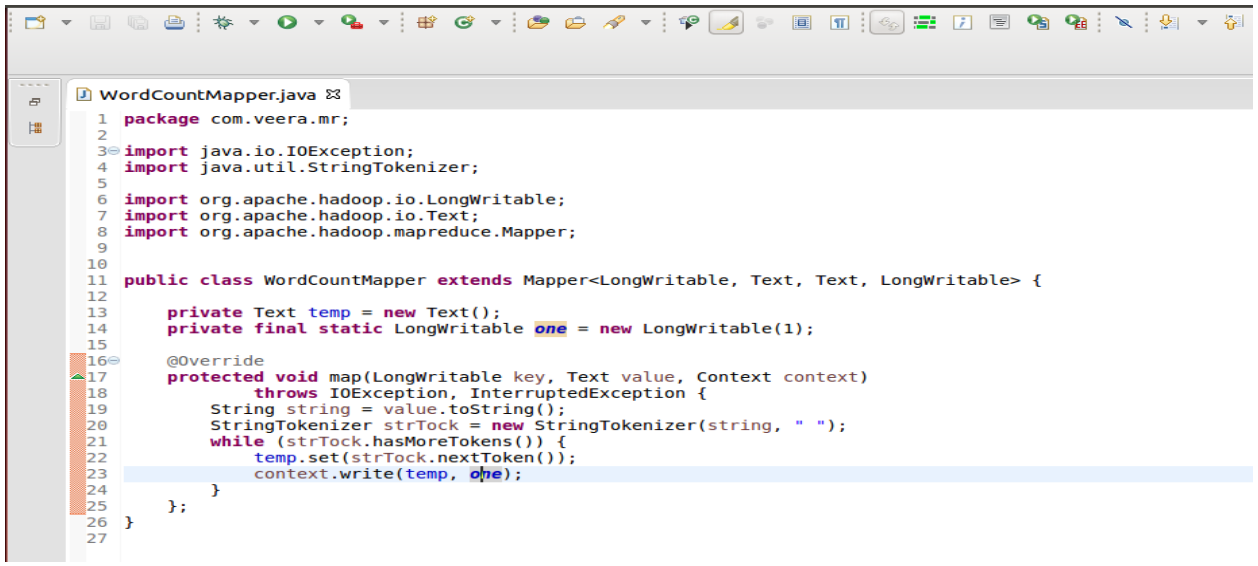
After creating the package, create class for Mapper



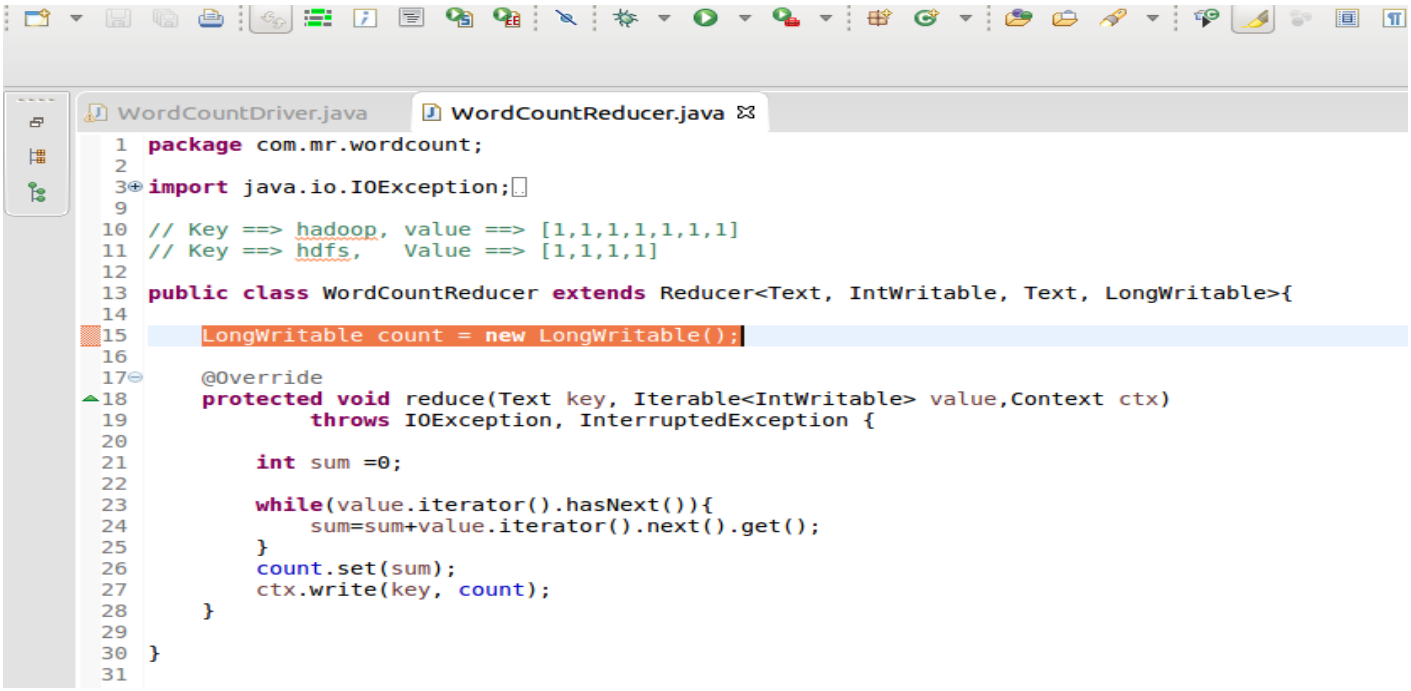
Give the class name and click finish



WordCountMapper.java

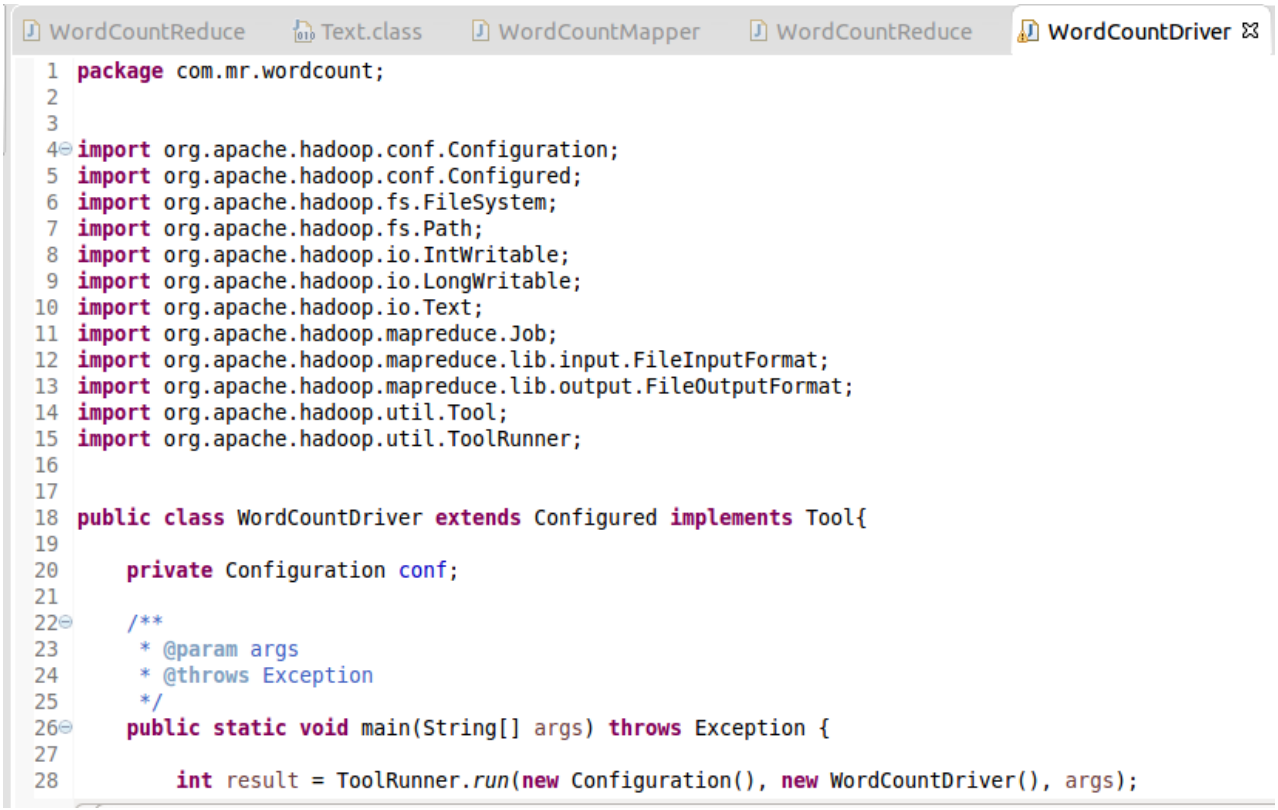


WordCountReducer.java



```java
package com.mr.wordcount;

import java.io.IOException;

// Key ==> hadoop, value ==> [1,1,1,1,1,1,1]
// Key ==> hdfs,   Value ==> [1,1,1,1]

public class WordCountReducer extends Reducer<Text, IntWritable, Text, LongWritable>{

    LongWritable count = new LongWritable();

    @Override
    protected void reduce(Text key, Iterable<IntWritable> value,Context ctx)
            throws IOException, InterruptedException {

        int sum =0;

        while(value.iterator().hasNext()){
            sum=sum+value.iterator().next().get();
        }
        count.set(sum);
        ctx.write(key, count);
    }

}
```

WordCountDriver.java

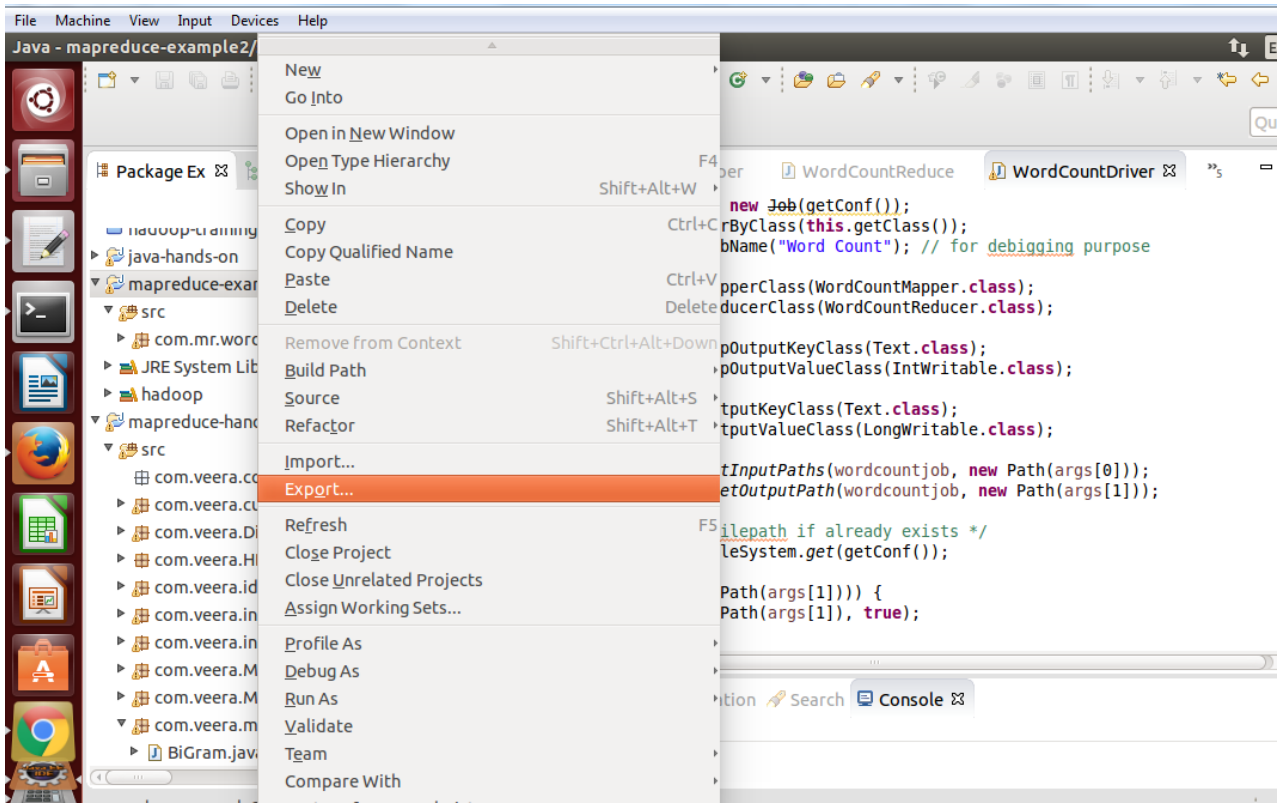


```java
package com.mr.wordcount;


import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.FileSystem;
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.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;


public class WordCountDriver extends Configured implements Tool{

    private Configuration conf;

    /**
     * @param args
     * @throws Exception
     */
    public static void main(String[] args) throws Exception {

        int result = ToolRunner.run(new Configuration(), new WordCountDriver(), args);
```

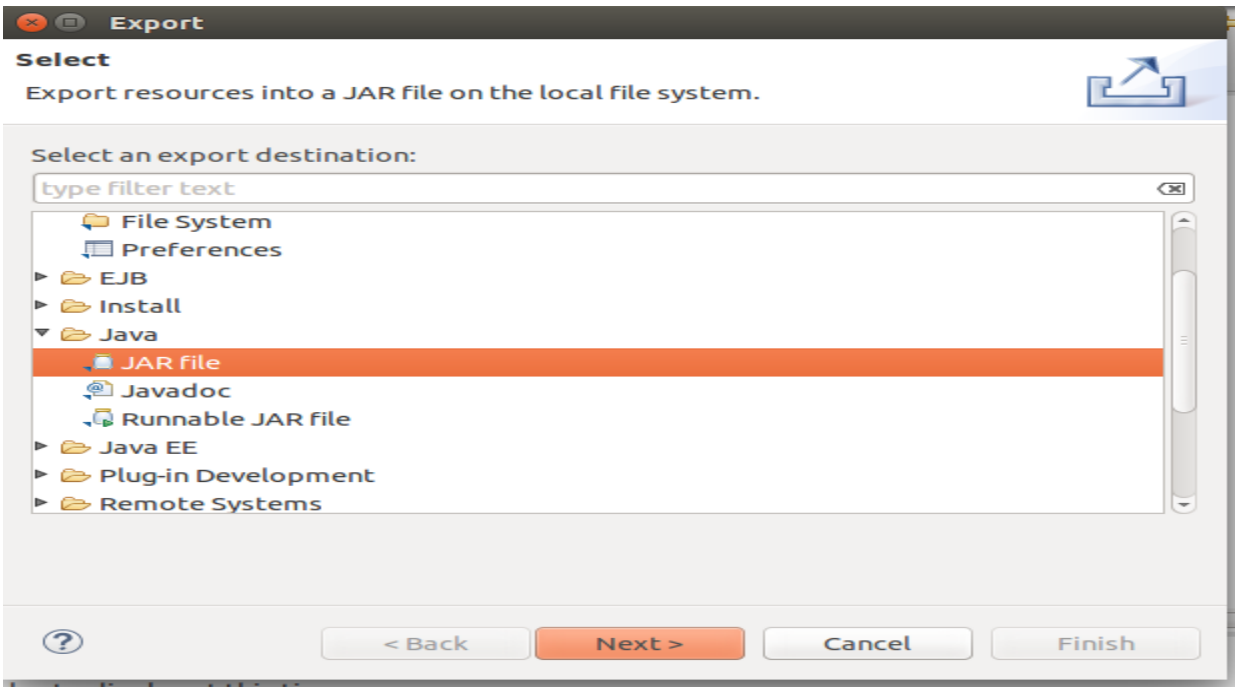
```
        System.out.println("Result of wordcount job " + result);

        if (result == 0) {
            System.out.println("Result of wordcount job Success");
        }
        else {
            System.out.println("Result of wordcount job Failure");
        }
    }

    @Override
    public void setConf(Configuration conf) {
        this.conf = conf;
    }

    @Override
    public Configuration getConf() {
        return conf;
    }

    @Override
    public int run(String[] args) throws Exception {

        Job wordcountjob = new Job(getConf());
        wordcountjob.setJarByClass(this.getClass());
        wordcountjob.setJobName("Word Count"); // for debigging purpose


        wordcountjob.setMapperClass(WordCountMapper.class);
        wordcountjob.setReducerClass(WordCountReducer.class);

        wordcountjob.setMapOutputKeyClass(Text.class);
        wordcountjob.setMapOutputValueClass(IntWritable.class);

        wordcountjob.setOutputKeyClass(Text.class);
        wordcountjob.setOutputValueClass(LongWritable.class);

        FileInputFormat.setInputPaths(wordcountjob, new Path(args[0]));
        FileOutputFormat.setOutputPath(wordcountjob, new Path(args[1]));

        /* Delete output filepath if already exists */
        FileSystem fs = FileSystem.get(getConf());

        if (fs.exists(new Path(args[1]))) {
            fs.delete(new Path(args[1]), true);
        }

        return wordcountjob.waitForCompletion(true) == true ? 0 : 1;
    }

}
```

Steps to Run in MR-mode:

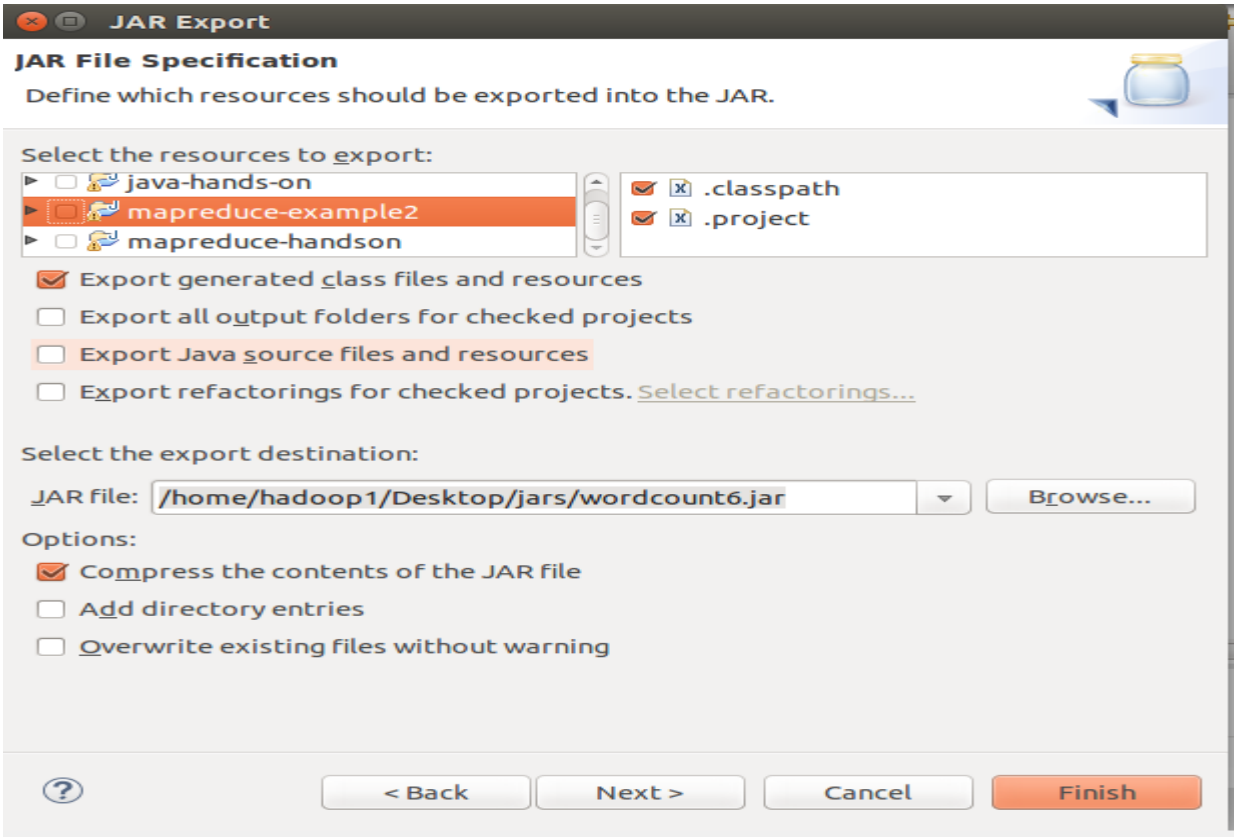
1) Create JAR.
   Right click on project (or package or select the files) → select **Export**



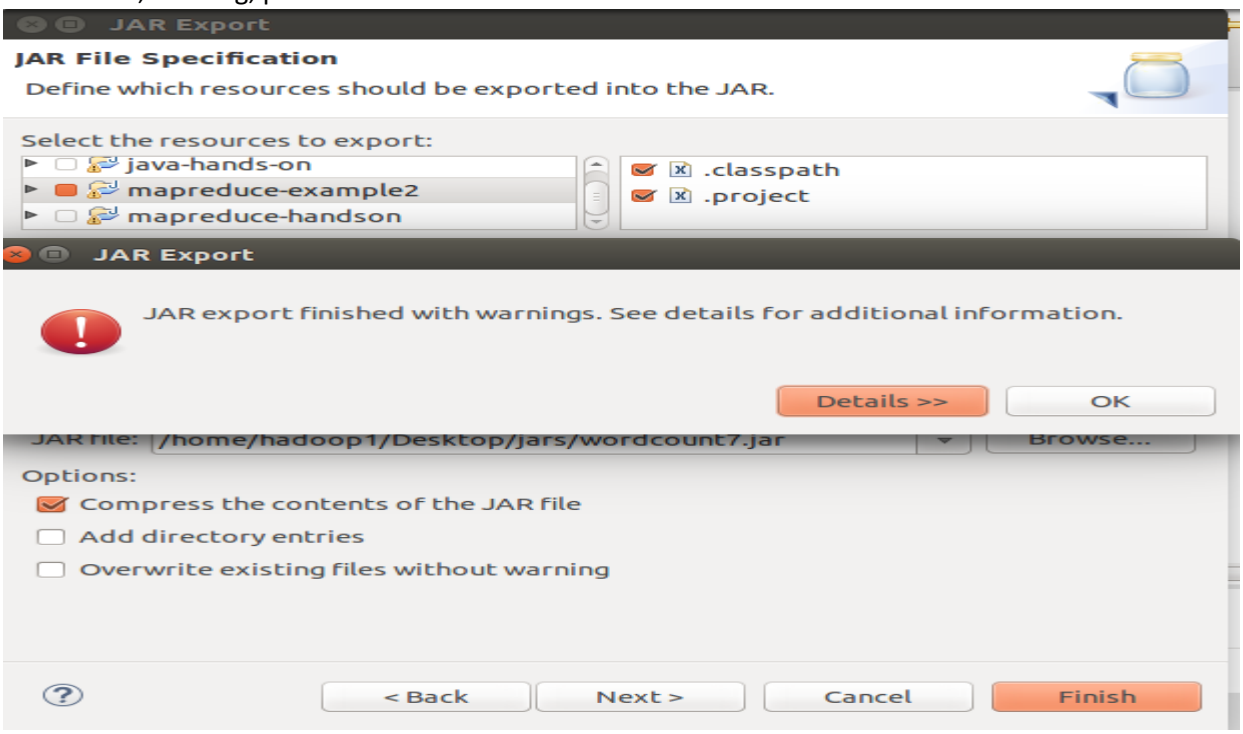
Select JAR from java and click on **Next** button



Select path to save JAR path and give name(/home/hadoop1/Desktop/jars/wordcount6.jar) and click on **Finish**: follow below screen



It will show, warning, please click on OK



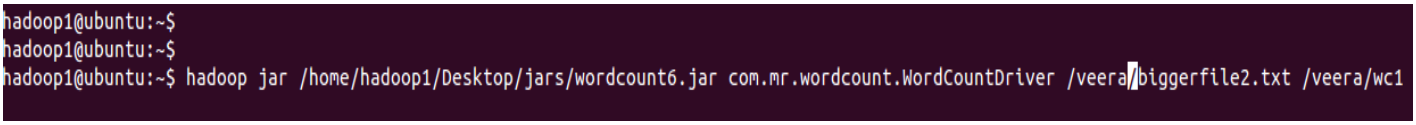
2) Start all the Hadoop services.
3) Execute below command to launch MR job.

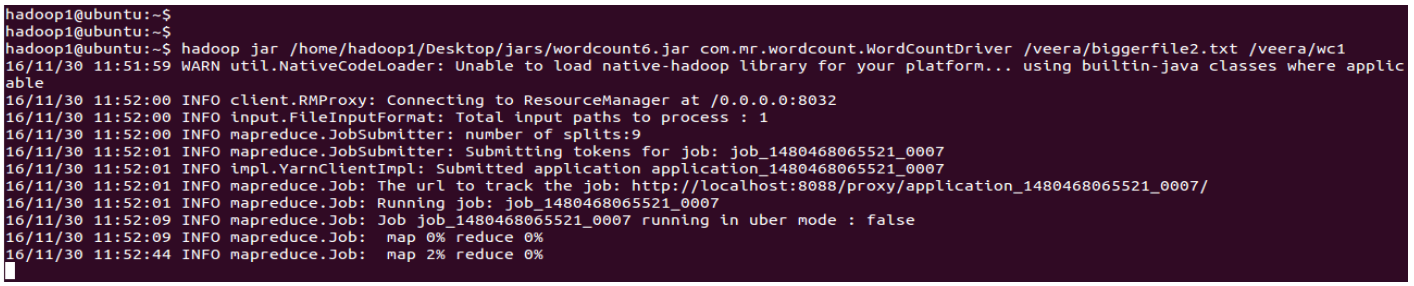
syntax: hadoop jar <path of the saved jar)> <fully qualified driver class name> <hdfs input path> <hdfs output path> …

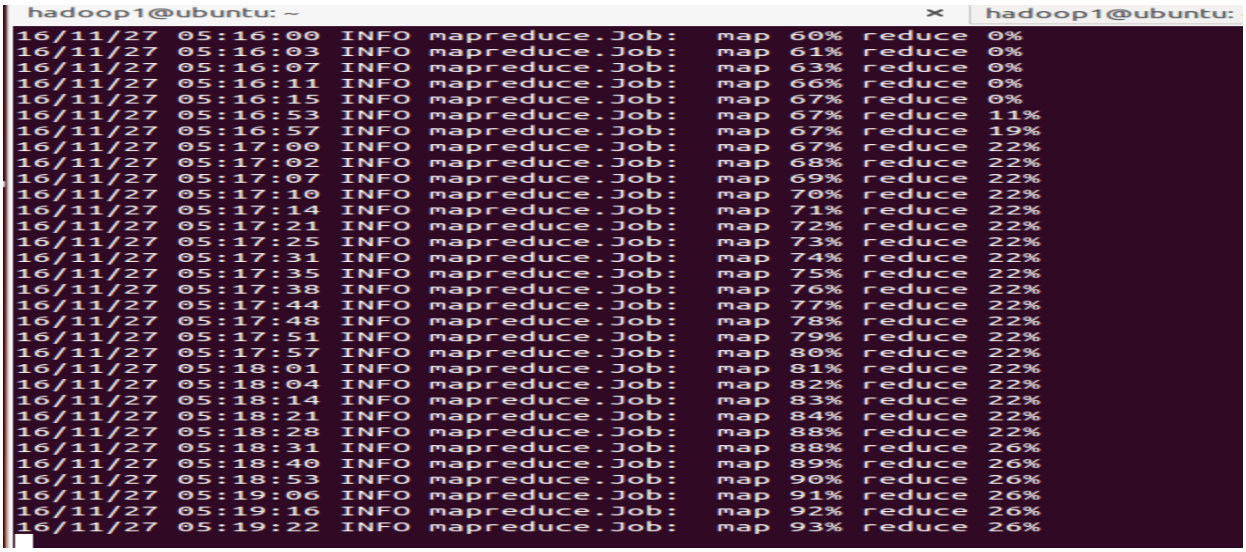
<hdfs input path> : Input file location in HDFS

<hdfs output path> : non existing(MR frame work will create dir in spacified path) output dir location in HDFS

ex: hadoop jar /home/hadoop1/Desktop/jars/wordcount6.jar com.mr.wordcount.wordCountDriver /veera/biggerFile.txt /veera/wc1







Job finished: see the counter:

```
hadoop1@ubuntu: ~                                    hadoop1@ubuntu: ~
16/11/27 05:20:24 INFO mapreduce.Job: Job job_1480223274108_0001 completed successfully
16/11/27 05:20:24 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=1008070035
                FILE: Number of bytes written=1529373125
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=286296619
                HDFS: Number of bytes written=43776
                HDFS: Number of read operations=30
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Killed map tasks=3
                Launched map tasks=11
                Launched reduce tasks=1
                Data-local map tasks=11
                Total time spent by all maps in occupied slots (ms)=2500254
                Total time spent by all reduces in occupied slots (ms)=245654
                Total time spent by all map tasks (ms)=2500254
                Total time spent by all reduce tasks (ms)=245654
                Total vcore-milliseconds taken by all map tasks=2500254
                Total vcore-milliseconds taken by all reduce tasks=245654
                Total megabyte-milliseconds taken by all map tasks=2560260096
                Total megabyte-milliseconds taken by all reduce tasks=251549696
        Map-Reduce Framework
                Map input records=4557361
                Map output records=40520729
                Map output bytes=439189453
                Map output materialized bytes=520230965
                Input split bytes=972
                Combine input records=0
```

```
hadoop1@ubuntu: ~                                    hadoop1@ubuntu: ~
                Map input records=4557361
                Map output records=40520729
                Map output bytes=439189453
                Map output materialized bytes=520230965
                Input split bytes=972
                Combine input records=0
                Combine output records=0
                Reduce input groups=3378
                Reduce shuffle bytes=520230965
                Reduce input records=40520729
                Reduce output records=3378
                Spilled Records=119039344
                Shuffled Maps =9
                Failed Shuffles=0
                Merged Map outputs=9
                GC time elapsed (ms)=7654
                CPU time spent (ms)=296400
                Physical memory (bytes) snapshot=2328567808
                Virtual memory (bytes) snapshot=8018878464
                Total committed heap usage (bytes)=1739718656
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=286295647
        File Output Format Counters
                Bytes Written=43776
Result of wordcount job 0
Result of wordcount job Success
```

4) Go to HDFS output dir path, it will contain, 2 files **_SUCCESS** and **part-r-00000**. The **part-r-00000** is final output file.

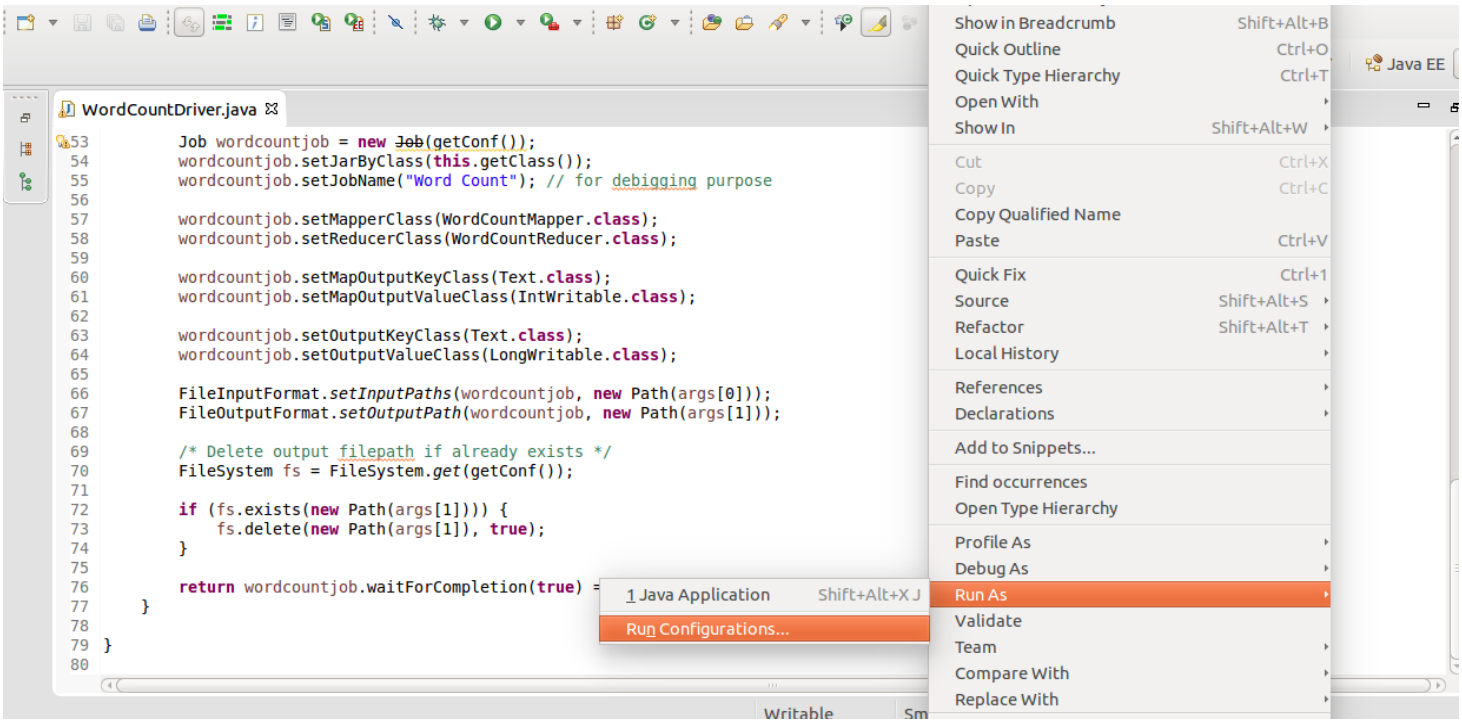
```
hadoop1@ubuntu:~$
hadoop1@ubuntu:~$
hadoop1@ubuntu:~$ hadoop fs -ls /veera/wc1
16/11/30 12:33:15 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your plat
able
Found 2 items
-rw-r--r--   1 hadoop1 supergroup          0 2016-11-30 11:56 /veera/wc1/_SUCCESS
-rw-r--r--   1 hadoop1 supergroup      35751 2016-11-30 11:56 /veera/wc1/part-r-00000
hadoop1@ubuntu:~$
```

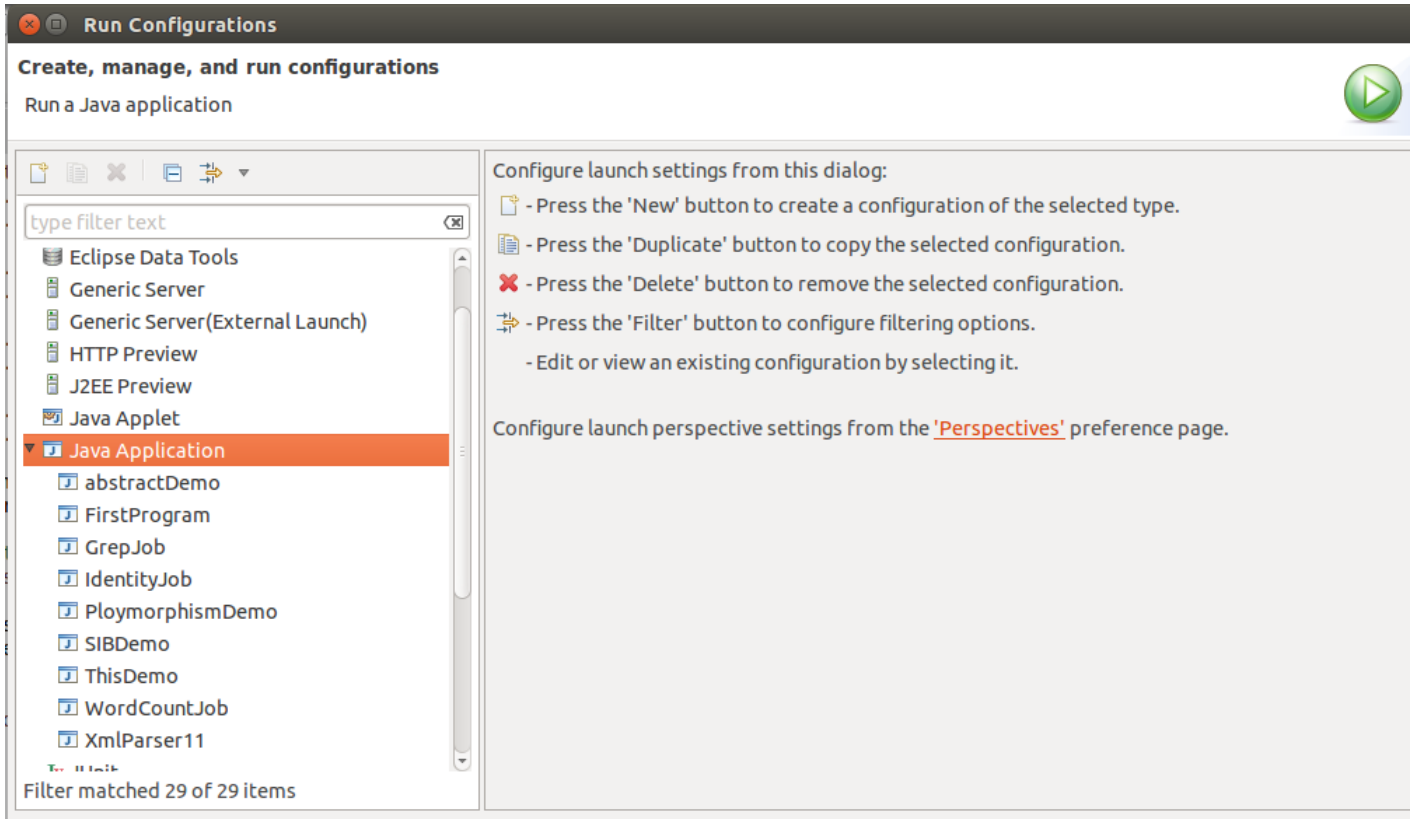
5) View the out of mapreduce job.

```
hadoop1@ubuntu:~$
hadoop1@ubuntu:~$ hadoop fs -cat /veera/wc1/part-r-00000 | tail -20
16/11/30 12:50:38 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform
able
zero    3
zip     1
{-m|-x  2
{-n     4
|       20177
Čeština 6720
Монгол  6720
Русский 6720
Српски  6720
Українська      6720
עברית   6720
العربية 6720
فارسی   6720
தமிழ்   6720
മലയാളം 6720
-       107520
_       40320
中文    6720
日本語  6720
한국어  6720
hadoop1@ubuntu:~$
```

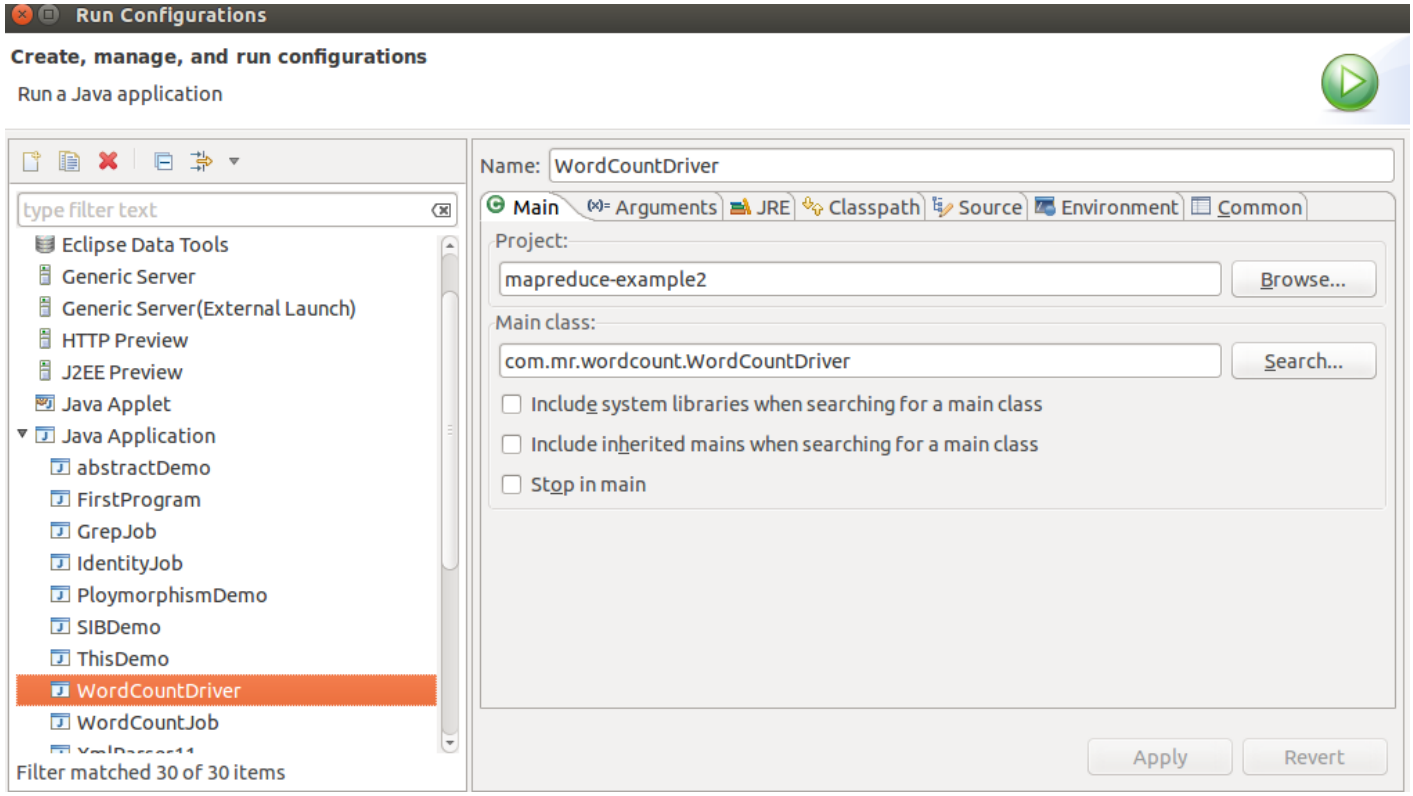
Steps to Run in Local Runner-mode:

1) Right click on Driver class ex: WordCountDriver.java
2) Select **Run As → Run configurations** and **Double click on Java Application**

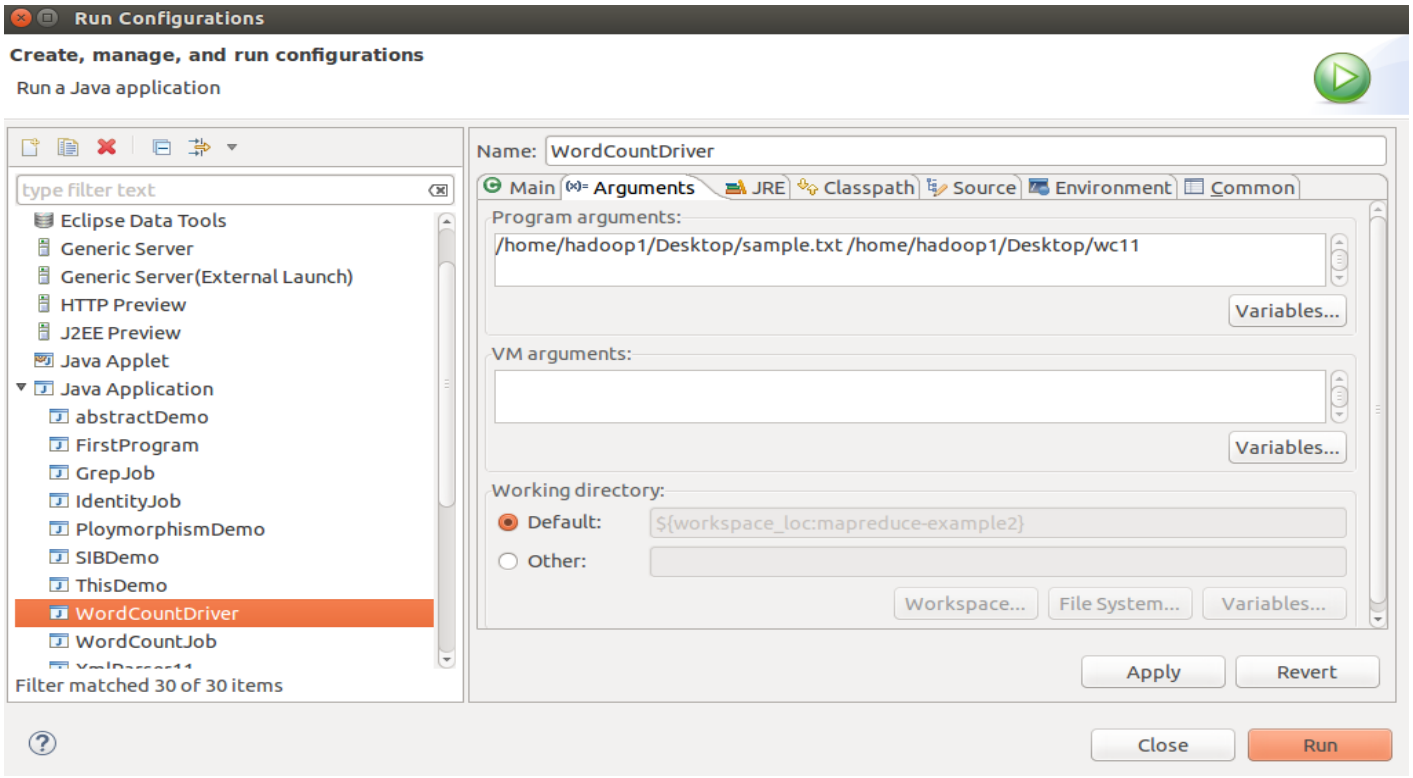




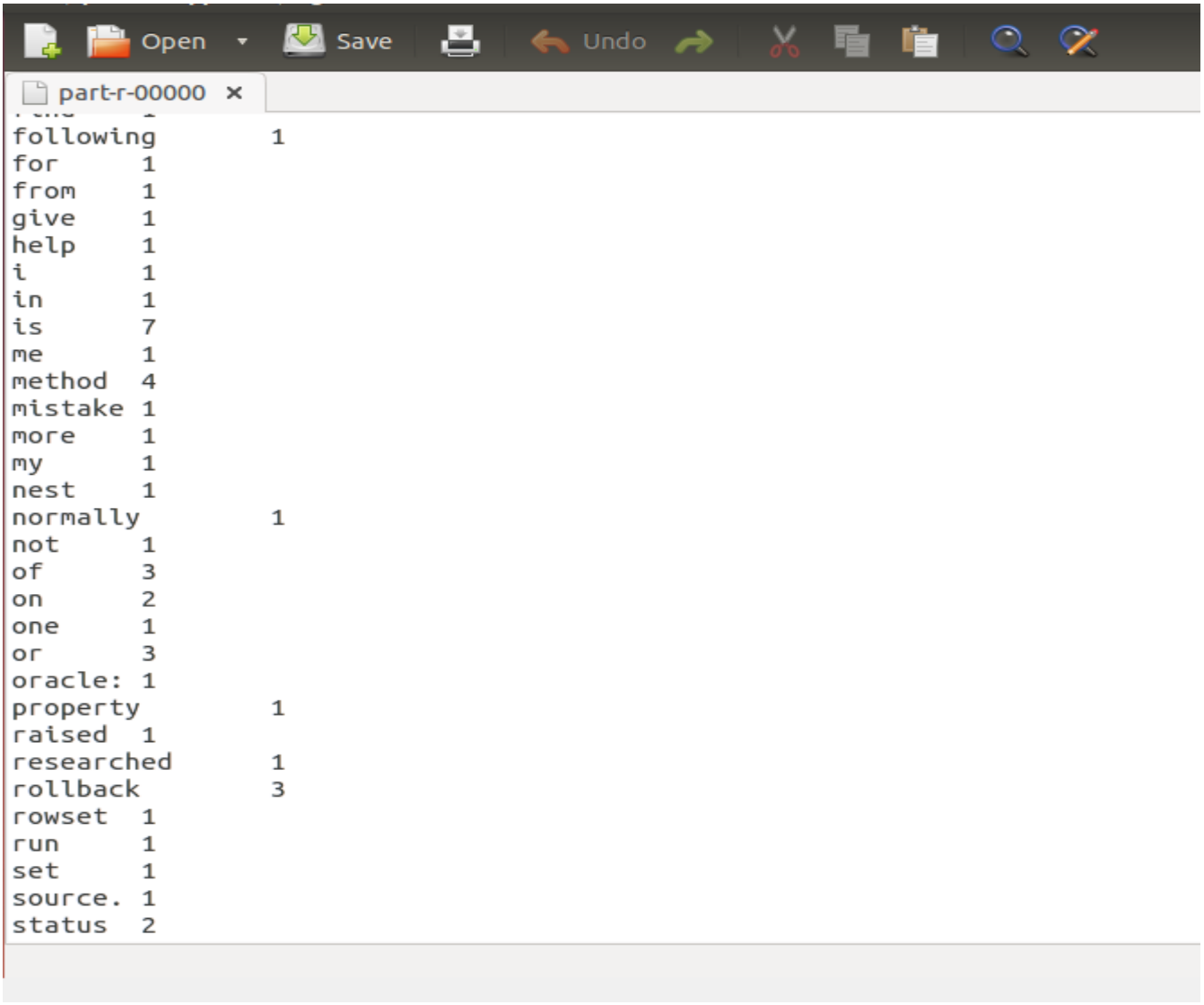
3) It will list out the respective driver class name (which you selected driver class)



4) Click on **Arguments** tab from right pane and give the **local input path** (not HDFS) , **local out path dir** (the dir should not exists) and click on **RUN**



5) Go to output dir in local file system (/home/hadoop1/Desktop/wc1), it will have **_SUCCESS** file and **part_00000** files. The **part_00000** is final output (reducer output)


Open
Save
Undo
part-r-00000
following 1
for 1
from 1
give 1
help 1
i 1
in 1
is 7
me 1
method 4
mistake 1
more 1
my 1
nest 1
normally 1
not 1
of 3
on 2
one 1
or 3
oracle: 1
property 1
raised 1
researched 1
rollback 3
rowset 1
run 1
set 1
source. 1
status 2