

## Practical - 5 Matrix Multiplication

MapReduce is a technique in which a huge program is subdivided into small tasks and run parallel to make computation faster, save time, and mostly used in distributed systems. It has 2 important parts:

- **Mapper:** It takes raw data input and organizes into key, value pairs. For example, In a dictionary, you search for the word “Data” and its associated meaning is “facts and statistics collected together for reference or analysis”. Here the Key is *Data* and the **Value** associated with is *facts and statistics collected together for reference or analysis*.
- **Reducer:** It is responsible for processing data in parallel and produce final output.

### Matrix Multiplication:

Matrix-vector and matrix-matrix calculations fit nicely into the MapReduce style of computing.

### Matrix Data Model for MapReduce

We represent matrix M as a relation M(I,J,V), with tuples  $(i, j, m_{ij})$  and matrix N as a relation N(J,K,W), with tuples  $(j, k, n_{jk})$ . Most matrices are sparse so large amount of cells have value zero. When we represent matrices in this form, we do not need to keep entries for the cells that have values of zero to save large amount of disk space. As input data files, we store matrix M and N on HDFS

M, i, j, m<sub>ij</sub>

M, 0, 0, 10.0  
M, 0, 2, 9.0  
M, 0, 3, 9.0  
M, 1, 0, 1.0

N, j, k, n<sub>jk</sub>

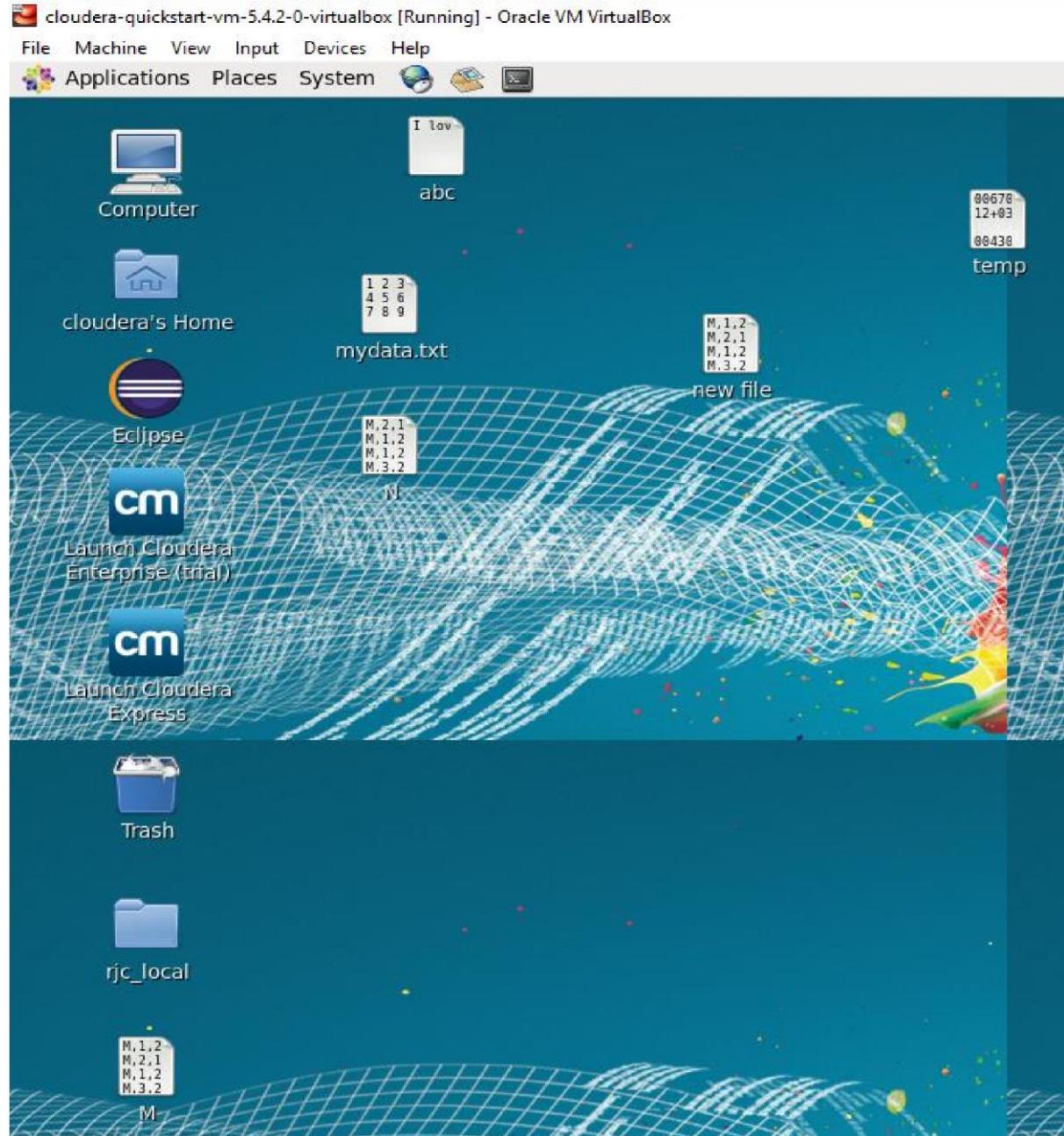
N, 0, 0, 1.0  
N, 0, 2, 3.0  
N, 0, 4, 2.0  
N, 1, 0, 2.0  
...

### Map Reduce:

We will write Map and Reduce functions to process input files. Map function will produce *key, value* pairs from the input data as it is described in Algorithm 1. Reduce function uses the output of the Map function and performs the calculations and produces *key, value* pairs as described in Algorithm 2. All outputs are written to HDFS.

## Steps for MatrixMultiplication :

- 1) Open virtual box and then start cloudera quickstart

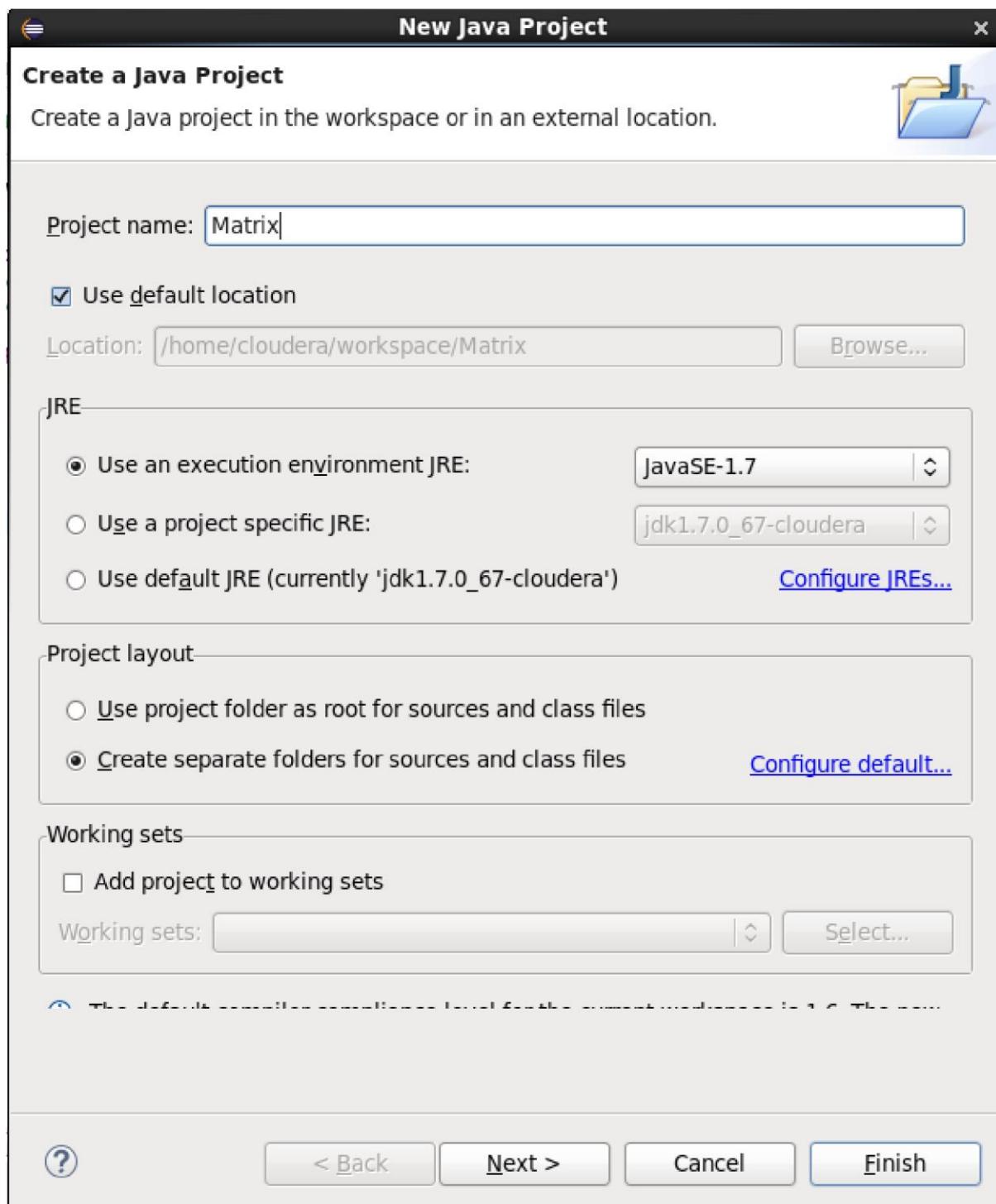


- 2) Open Eclipse present on the cloudera desktop

**Name: - Zen Dsouza**  
**Roll No: - 34**  
**Subject - BDT**

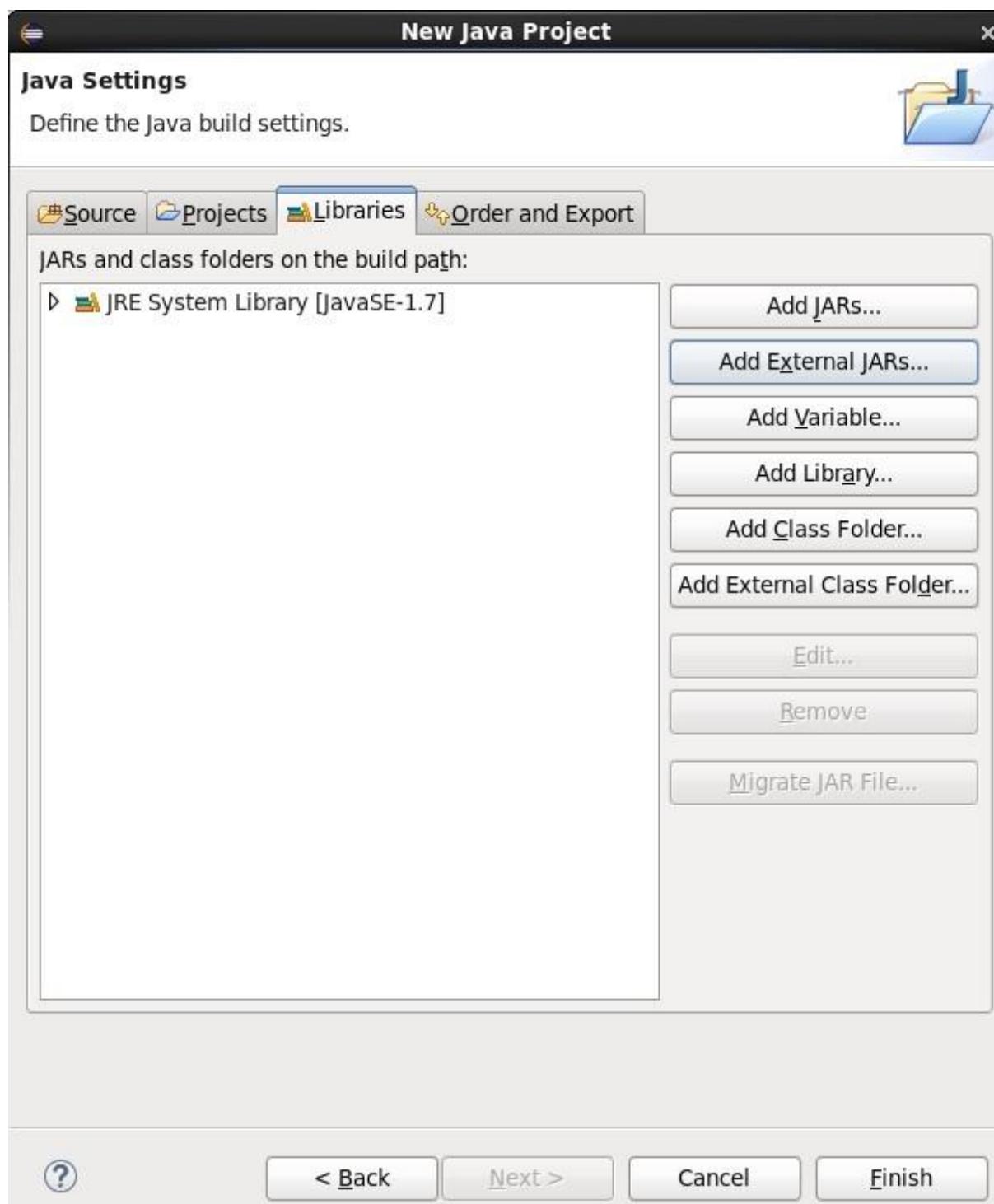


- 3) Create a new Java project clicking: File -> New -> Project -> Java Project -> Next ("Matrix" is the project name).

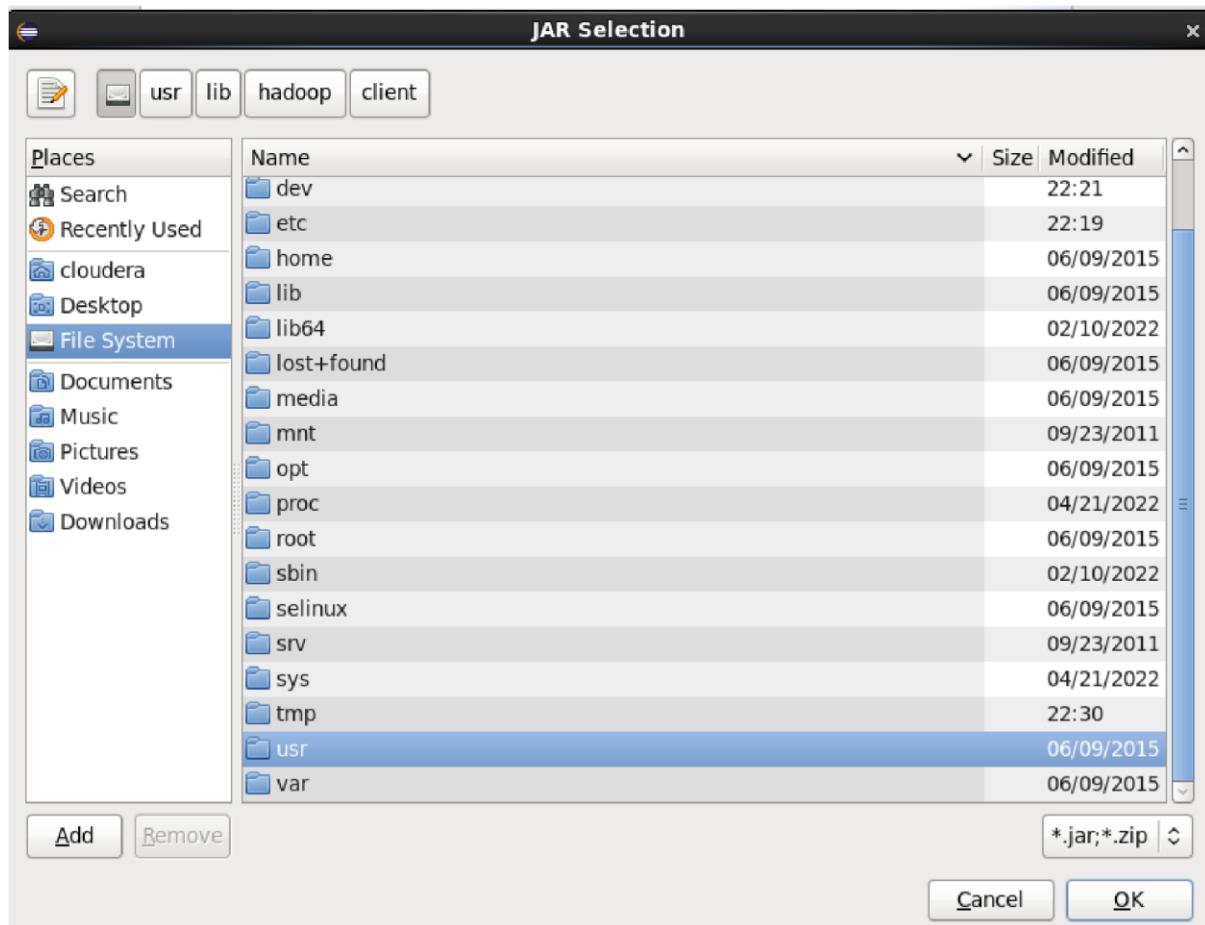


- 4) Adding the Hadoop libraries to the project Click on Libraries -> Add External JARs Click on File System -> usr -> lib -> Hadoop, Select all the libraries (JAR Files) -> click OK Click on Add External jars, -> client -> select all jar files -> ok -> Finish

Name: - Zen Dsouza  
Roll No: - 34  
Subject - BDT



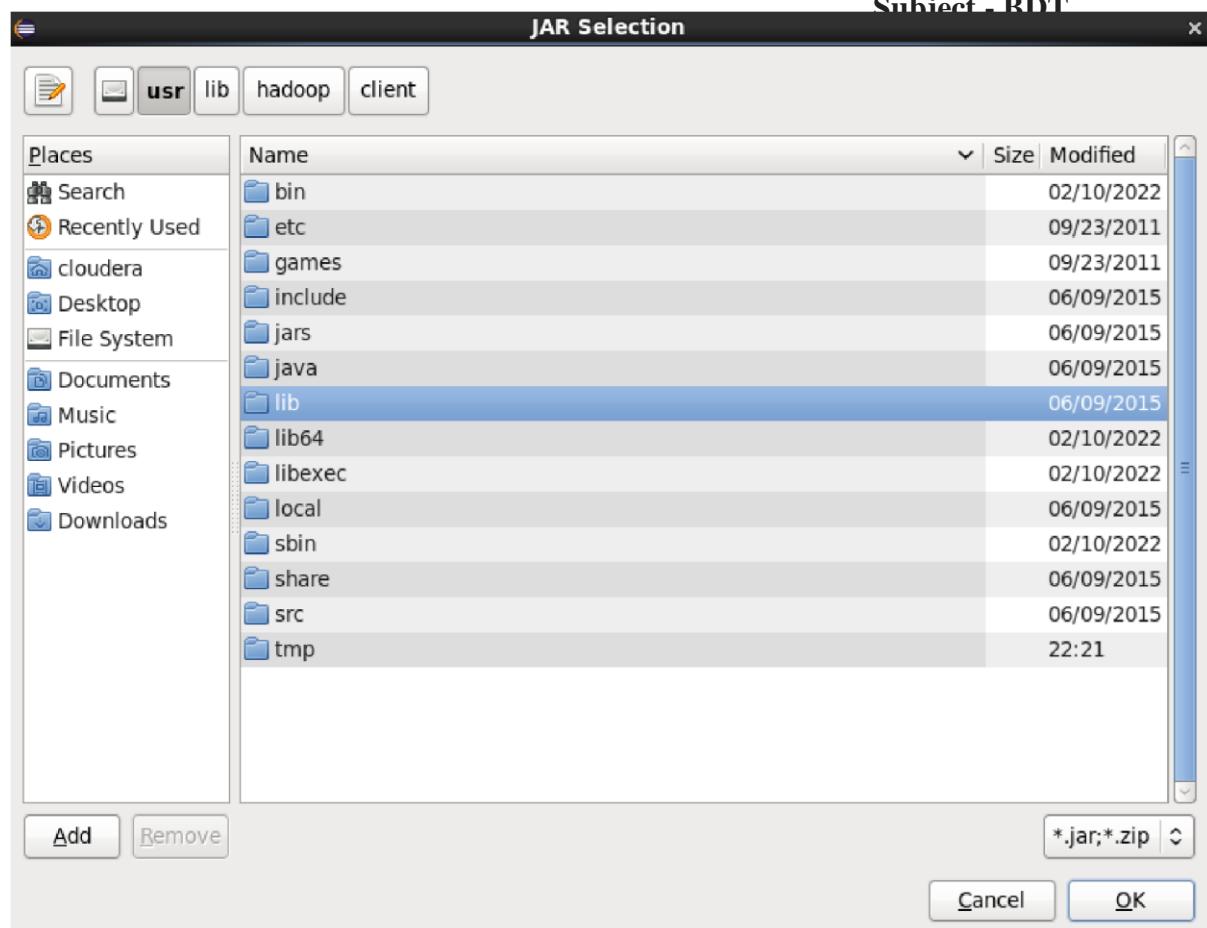
Name: - Zen Dsouza  
Roll No: - 34  
Subject - BDT



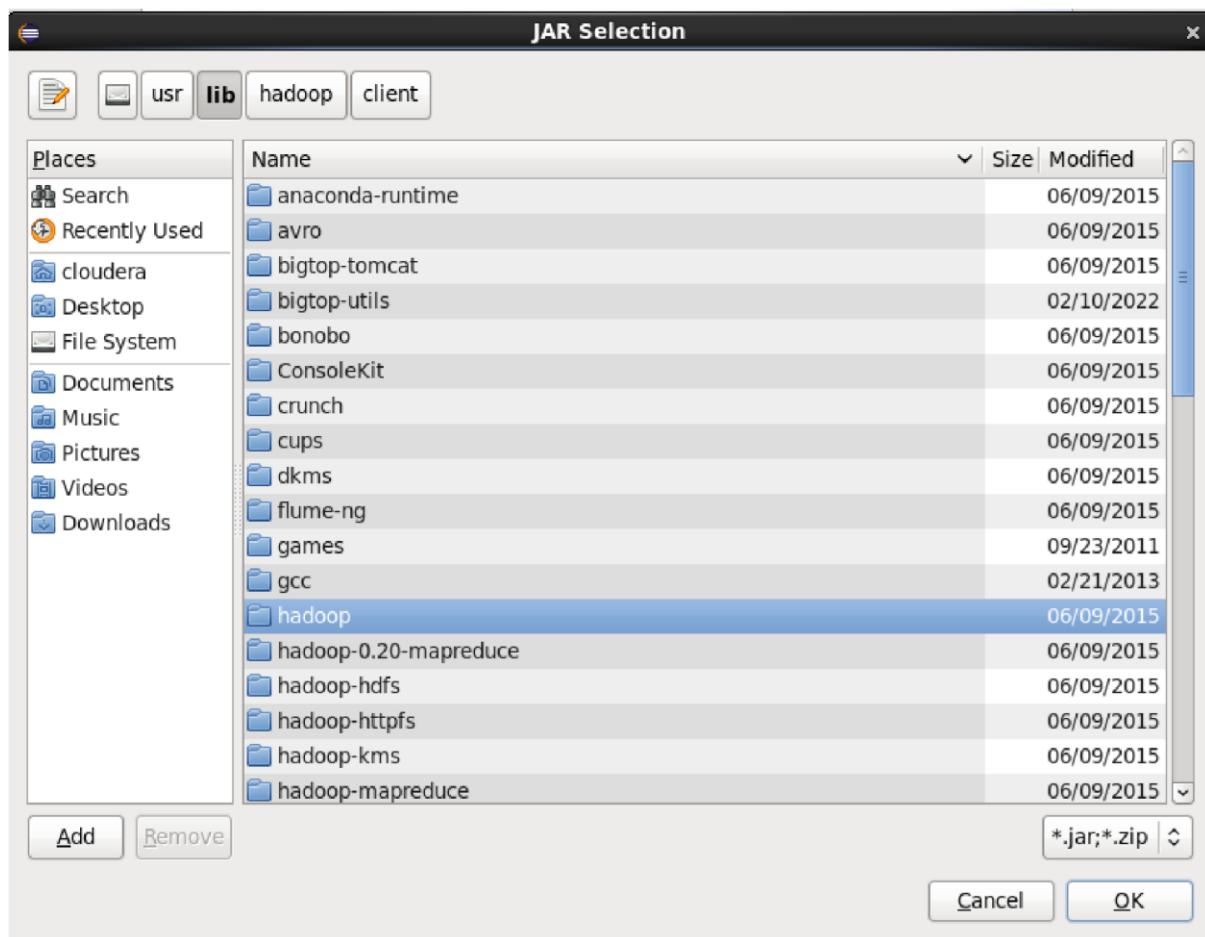
Name: - Zen Dsouza

Roll No: - 34

Subject - BDT



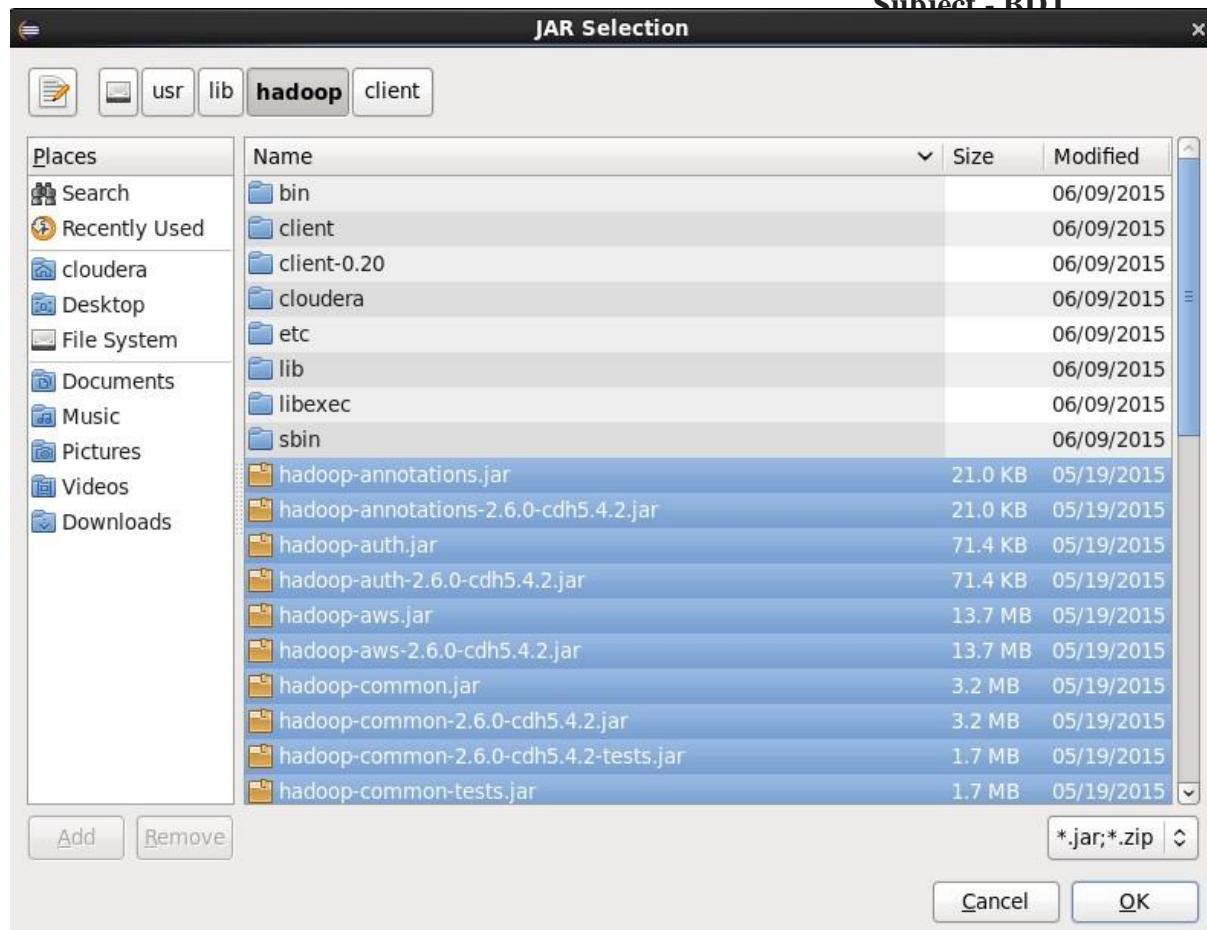
Name: - Zen Dsouza  
Roll No: - 34  
Subject - BDT



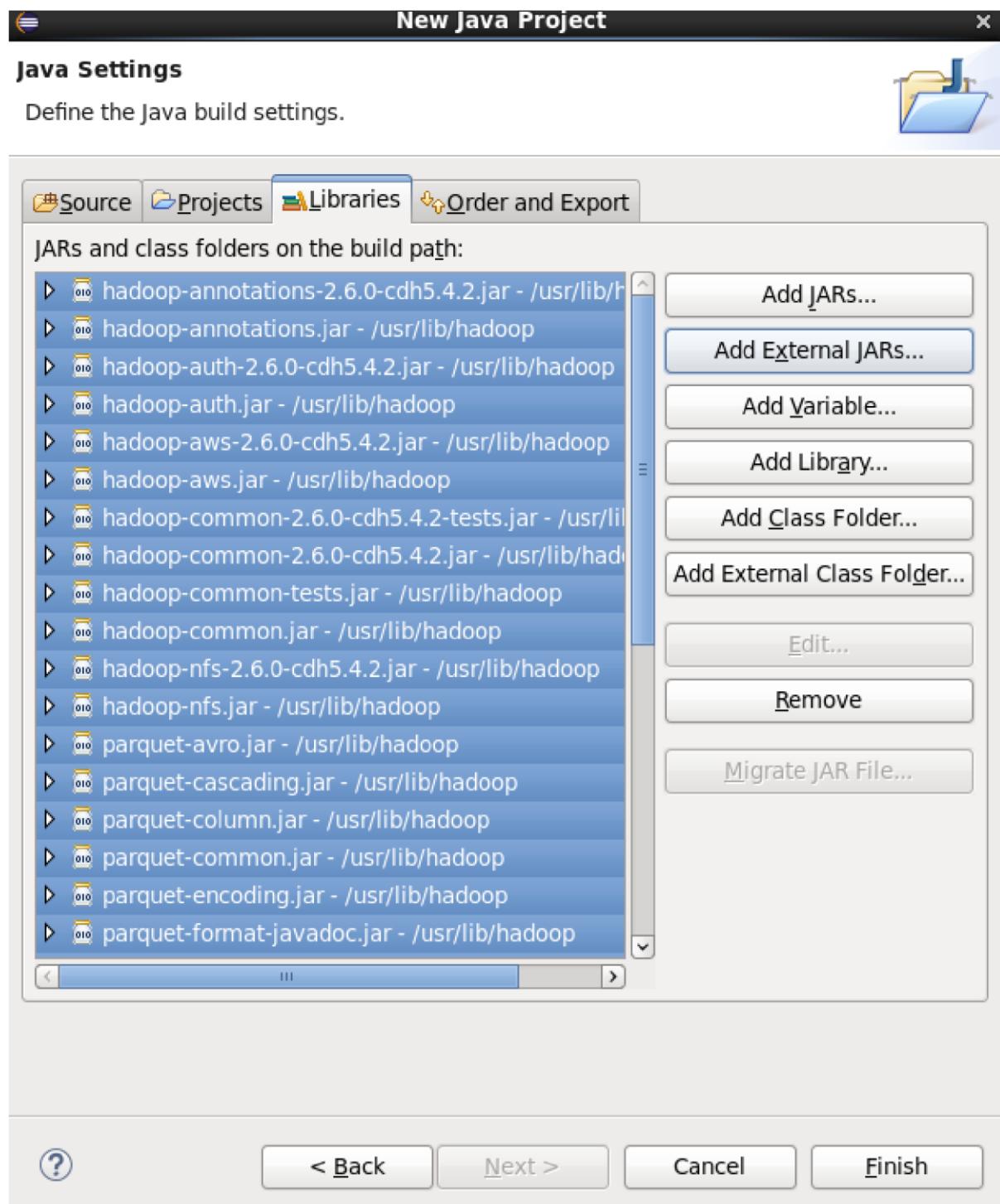
Name: - Zen Dsouza

Roll No: - 34

Subject - BDT



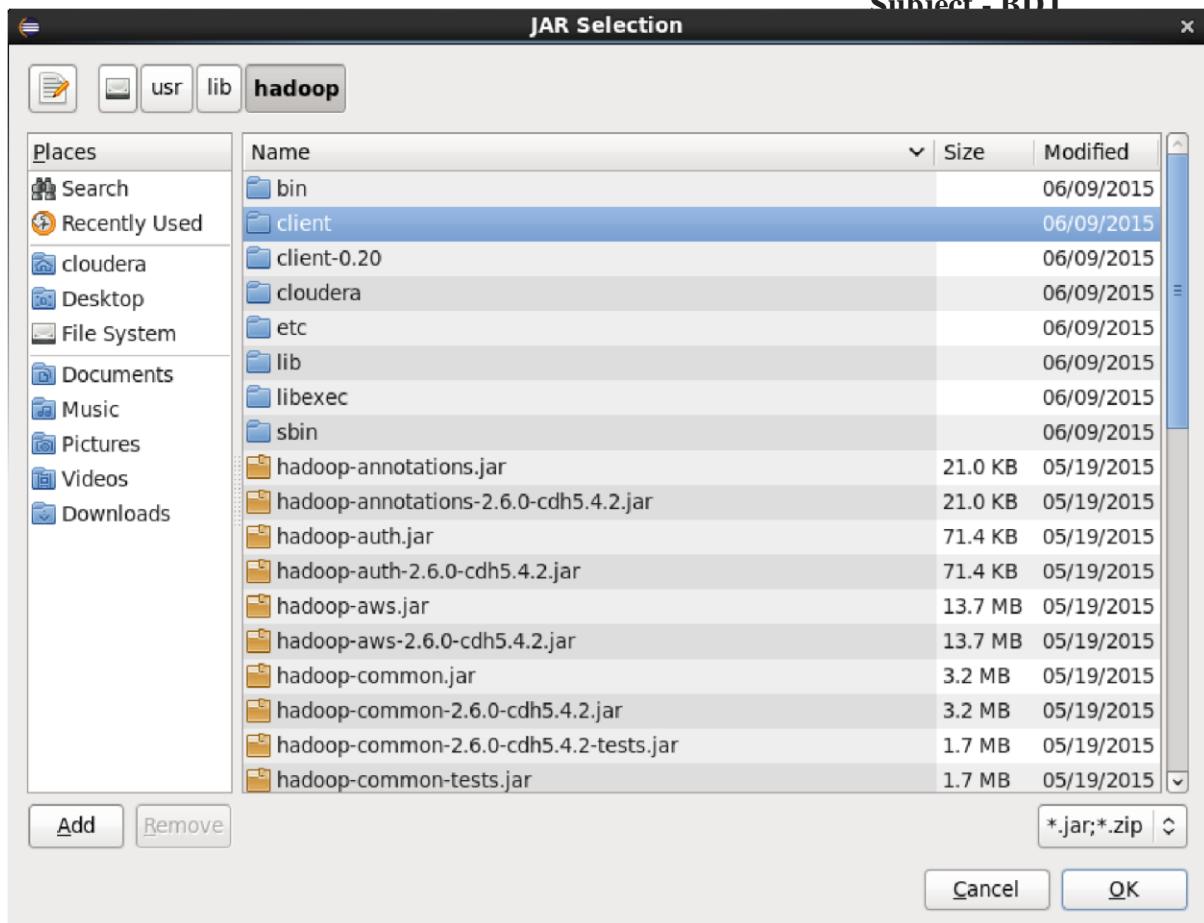
Name: - Zen Dsouza  
Roll No: - 34  
Subject - BDT



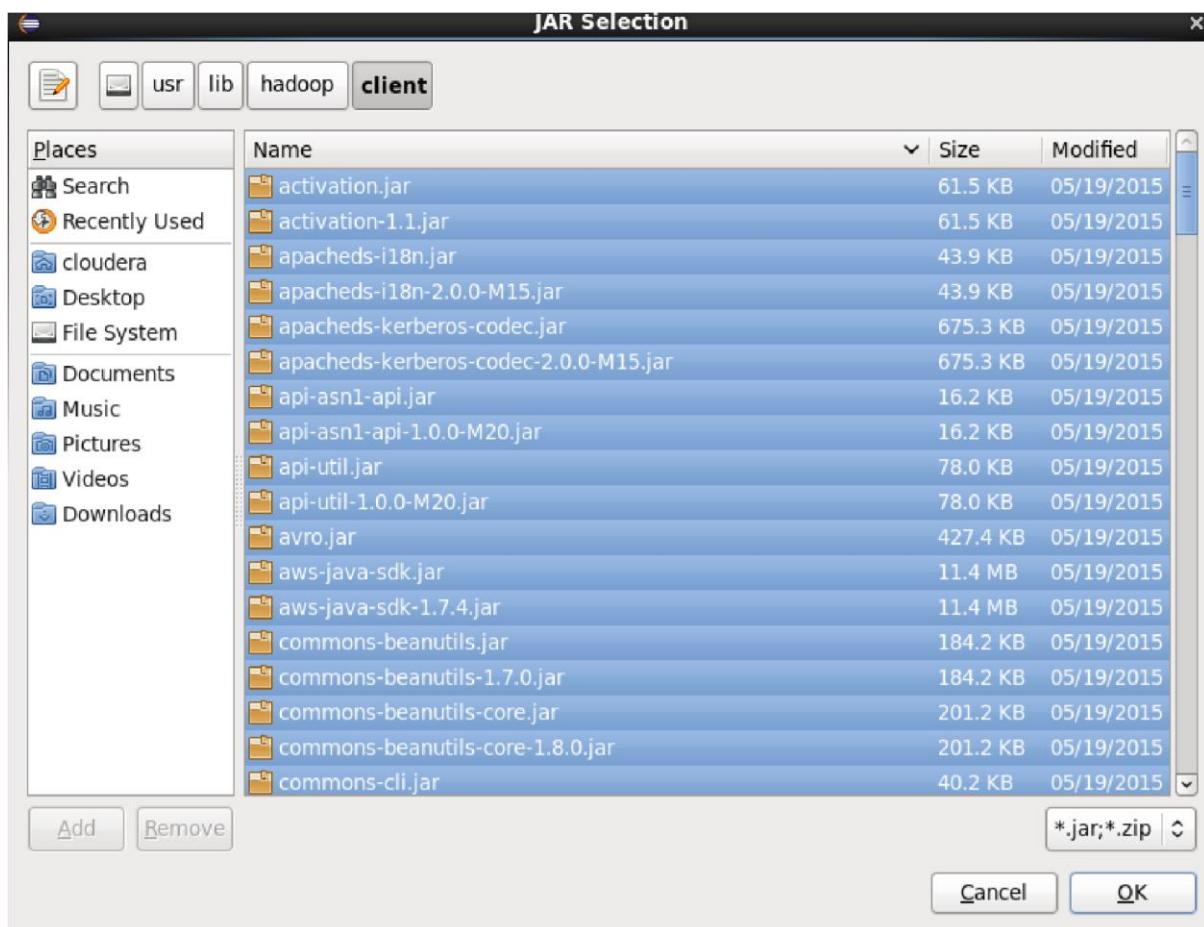
Name: - Zen Dsouza

Roll No: - 34

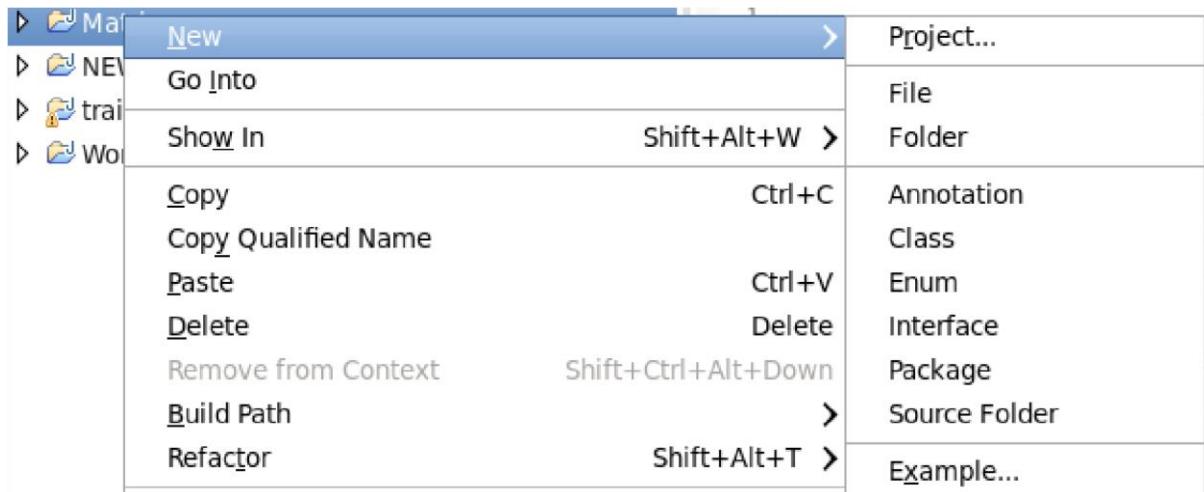
Subject - BDT

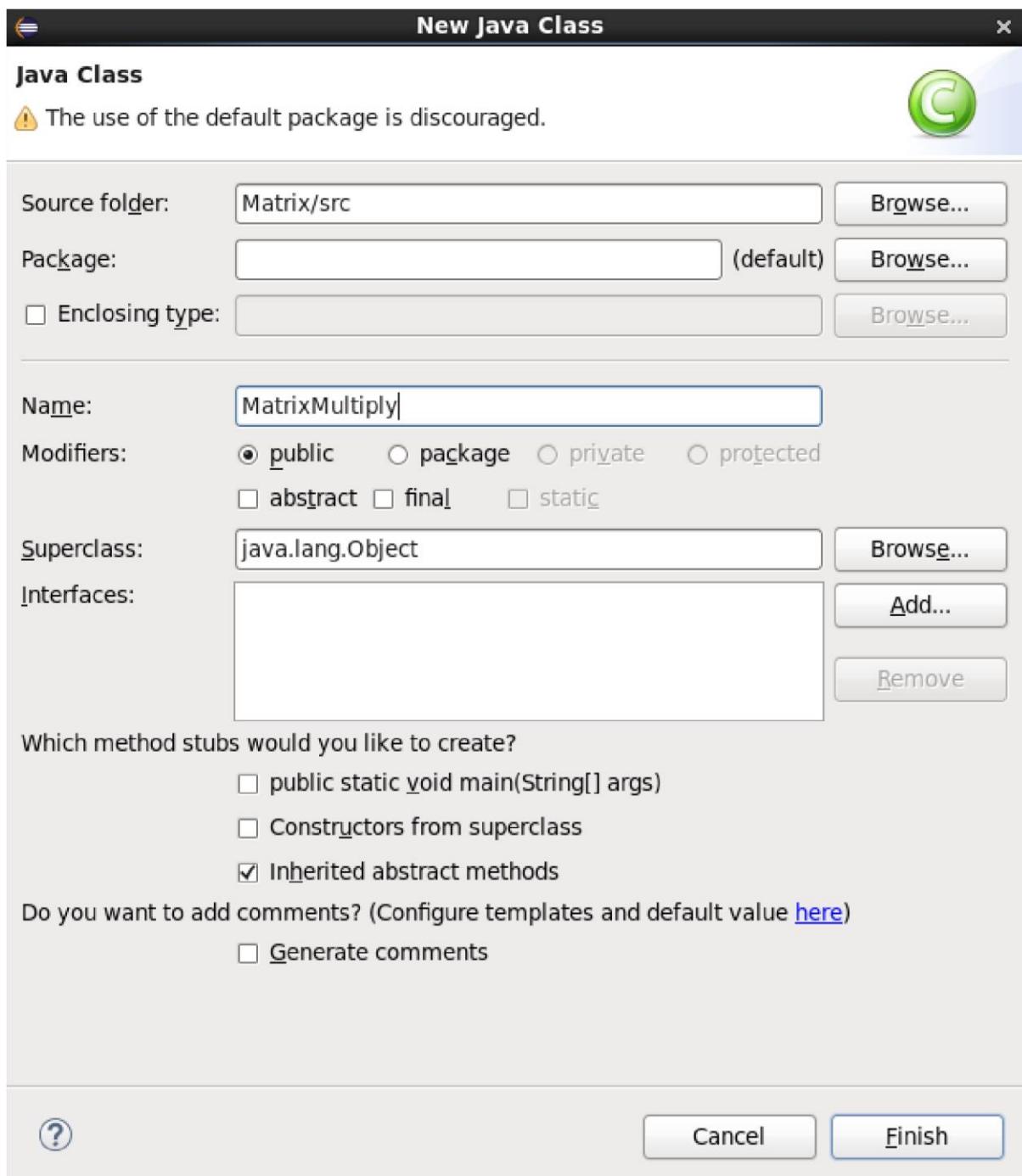


Name: - Zen Dsouza  
Roll No: - 34  
Subject - BDT



- 5) Right Click on the name of Project “Matrix” -> New -> class Don’t write anything for package Write Name Textbox write “Matrix” -> Finish Then MatrixMulltiple.java window will pop up





### Source Code:

```
import java.io.IOException;
import java.util.HashMap;

import org.apache.hadoop.conf.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
```

```
import org.apache.hadoop.mapreduce.Mapper.Context;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.conf.*;
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 java.io.IOException;
import java.util.HashMap;

public class MatrixMultiply {

    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: MatrixMultiply <in_dir> <out_dir>");
            System.exit(2);
        }
        Configuration conf = new Configuration();
        // M is an m-by-n matrix; N is an n-by-p matrix.
        conf.set("m", "1000");
        conf.set("n", "100");
        conf.set("p", "1000");
        @SuppressWarnings("deprecation")
        Job job = new Job(conf, "MatrixMultiply");
        job.setJarByClass(MatrixMultiply.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.waitForCompletion(true);
    }
    public class Map
        extends org.apache.hadoop.mapreduce.Mapper<LongWritable, Text,
Text, Text> {
        @Override
        public void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {
            Configuration conf = context.getConfiguration();
            int m = Integer.parseInt(conf.get("m"));
            int p = Integer.parseInt(conf.get("p"));
            String line = value.toString();
```

```

// (M, i, j, Mij);
String[] indicesAndValue = line.split(",");
Text outputKey = new Text();
Text outputValue = new Text();
if (indicesAndValue[0].equals("M")) {
    for (int k = 0; k < p; k++) {
        outputKey.set(indicesAndValue[1] + "," + k);
        // outputKey.set(i,k);
        outputValue.set(indicesAndValue[0] + "," + indicesAndValue[2]
            + "," + indicesAndValue[3]);
        // outputValue.set(M,j,Mij);
        context.write(outputKey, outputValue);
    }
} else {
    // (N, j, k, Njk);
    for (int i = 0; i < m; i++) {
        outputKey.set(i + "," + indicesAndValue[2]);
        outputValue.set("N," + indicesAndValue[1] + ","
            + indicesAndValue[3]);
        context.write(outputKey, outputValue);
    }
}
}

public class Reduce
extends org.apache.hadoop.mapreduce.Reducer<Text, Text, Text, Text> {
@Override
public void reduce(Text key, Iterable<Text> values, Context context)
    throws IOException, InterruptedException {
    String[] value;
    //key=(i,k),
    //Values = [(M/N,j,V/W),..]
    HashMap<Integer, Float> hashA = new HashMap<Integer, Float>();
    HashMap<Integer, Float> hashB = new HashMap<Integer, Float>();
    for (Text val : values) {
        value = val.toString().split(",");
        if (value[0].equals("M")) {
            hashA.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));
        } else {
            hashB.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));
        }
    }
    int n = Integer.parseInt(context.getConfiguration().get("n"));
    float result = 0.0f;
    float m_ij;
    float n_jk;
    for (int j = 0; j < n; j++) {
        m_ij = hashA.containsKey(j) ? hashA.get(j) : 0.0f;
        n_jk = hashB.containsKey(j) ? hashB.get(j) : 0.0f;
        result += m_ij * n_jk;
    }
    if (result != 0.0f) {
        context.write(null,

```

```
        new Text(key.toString() + "," + Float.toString(result)));
    }
}
}

StubDriver.java WordCount.java MaxTemp.java MatrixMul.java
```

```
1 import org.apache.hadoop.conf.*;
2 import org.apache.hadoop.fs.Path;
3 import org.apache.hadoop.io.*;
4 import org.apache.hadoop.mapreduce.*;
5 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
6 import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
7 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
8 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
9 import org.apache.hadoop.conf.*;
10 import org.apache.hadoop.io.LongWritable;
11 import org.apache.hadoop.io.Text;
12 import org.apache.hadoop.mapreduce.Mapper;
13 import org.apache.hadoop.io.Text;
14 import org.apache.hadoop.mapreduce.Reducer;
15
16 import java.io.IOException;
17 import java.util.HashMap;
18
19
20 import java.io.IOException;
21
22
23 public class MatrixMul {
24
25     public static void main(String[] args) throws Exception {
26         if (args.length != 2) {
27             System.err.println("Usage: MatrixMul <in_dir> <out_dir>");
28             System.exit(2);
29         }
30         Configuration conf = new Configuration();
31         // M is an m-by-n matrix; N is an n-by-p matrix.
32         conf.set("m", "1000");
33         conf.set("n", "100");
34         conf.set("p", "1000");
35         @SuppressWarnings("deprecation")
36         Job job = new Job(conf, "MatrixMul");
37         job.setJarByClass(MatrixMul.class);
38         job.setOutputKeyClass(Text.class);
39         job.setOutputValueClass(Text.class);
40
41         job.setMapperClass(Map.class);
42         job.setReducerClass(Reduce.class);
43
44         job.setInputFormatClass(TextInputFormat.class);
45         job.setOutputFormatClass(TextOutputFormat.class);
46     }
}
```

Name: - Zen Dsouza  
Roll No: - 34  
Subject - BDT

```
87④     public class Reduce
88      extends org.apache.hadoop.mapreduce.Reducer<Text, Text, Text, Text> {
89④       @Override
90       public void reduce(Text key, Iterable<Text> values, Context context)
91           throws IOException, InterruptedException {
92           String[] value;
93           //Key=(i,k),
94           //Values = [(M/N,j,V/W),...]
95           HashMap<Integer, Float> hashA = new HashMap<Integer, Float>();
96           HashMap<Integer, Float> hashB = new HashMap<Integer, Float>();
97           for (Text val : values) {
98               value = val.toString().split(",");
99               if (value[0].equals("M")) {
100                   hashA.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));
101               } else {
102                   hashB.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));
103               }
104           }
105           int n = Integer.parseInt(context.getConfiguration().get("n"));
106           float result = 0.0f;
107           float m_ij;
108           float n_jk;
109           for (int j = 0; j < n; j++) {
110               m_ij = hashA.containsKey(j) ? hashA.get(j) : 0.0f;
111               n_jk = hashB.containsKey(j) ? hashB.get(j) : 0.0f;
112               result += m_ij * n_jk;
113           }
114           if (result != 0.0f) {
115               context.write(null,
116                           new Text(key.toString() + "," + Float.toString(result)));
117           }
118       }
119   }
120 }
```

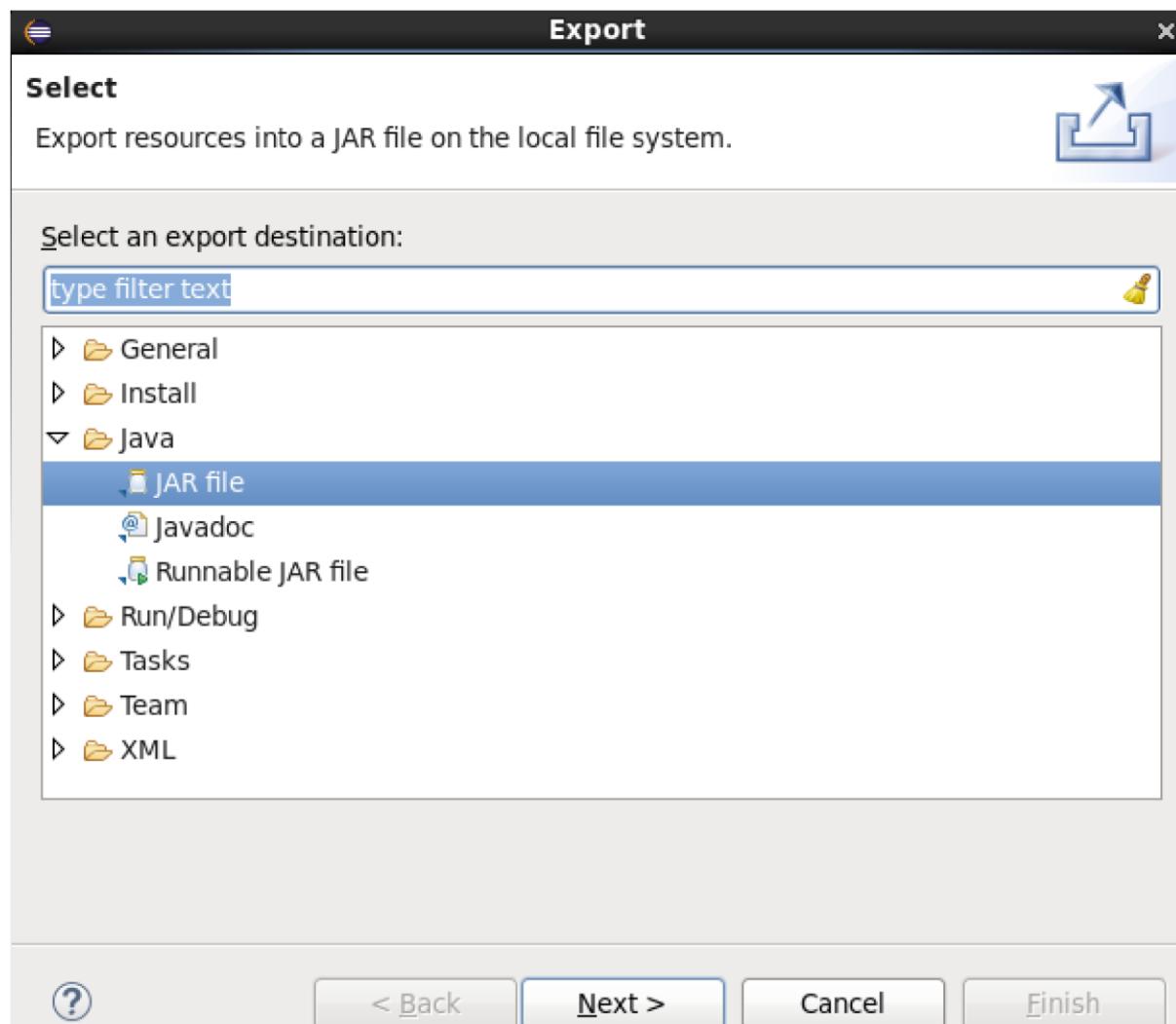
```

StubDriver.java WordCount.java MaxTemp.java MatrixMul.java
45         job.setOutputFormatClass(TextOutputFormat.class);
46
47     FileInputFormat.addInputPath(job, new Path(args[0]));
48     FileOutputFormat.setOutputPath(job, new Path(args[1]));
49
50     job.waitForCompletion(true);
51 }
52
53 public class Map
54     extends org.apache.hadoop.mapreduce.Mapper<LongWritable, Text, Text, Text> {
55
56     @Override
57     public void map(LongWritable key, Text value, Context context)
58         throws IOException, InterruptedException {
59         Configuration conf = context.getConfiguration();
60         int m = Integer.parseInt(conf.get("m"));
61         int p = Integer.parseInt(conf.get("p"));
62         String line = value.toString();
63         // (M, i, j, Mij);
64         String[] indicesAndValue = line.split(",");
65         Text outputKey = new Text();
66         Text outputValue = new Text();
67         if (indicesAndValue[0].equals("M")) {
68             for (int k = 0; k < p; k++) {
69                 outputKey.set(indicesAndValue[1] + "," + k);
70                 // outputKey.set(i,k);
71                 outputValue.set(indicesAndValue[0] + "," + indicesAndValue[2]
72                             + "," + indicesAndValue[3]);
73                 // outputValue.set(M,j,Mij);
74                 context.write(outputKey, outputValue);
75             }
76         } else {
77             // (N, j, k, Njk);
78             for (int i = 0; i < m; i++) {
79                 outputKey.set(i + "," + indicesAndValue[2]);
80                 outputValue.set("N," + indicesAndValue[1] + ","
81                             + indicesAndValue[3]);
82                 context.write(outputKey, outputValue);
83             }
84         }
85     }
86
87     public class Reduce
88         extends org.apache.hadoop.mapreduce.Reducer<Text, Text, Text, Text> {
89
90         @Override
91         public void reduce(Text key, Iterable<Text> values, Context context)
92             throws IOException, InterruptedException {
93             Text outputKey = new Text();
94             Text outputValue = new Text();
95             for (Text val : values) {
96                 outputKey.set(key + "," + val);
97                 context.write(outputKey, outputValue);
98             }
99         }
100    }
101}

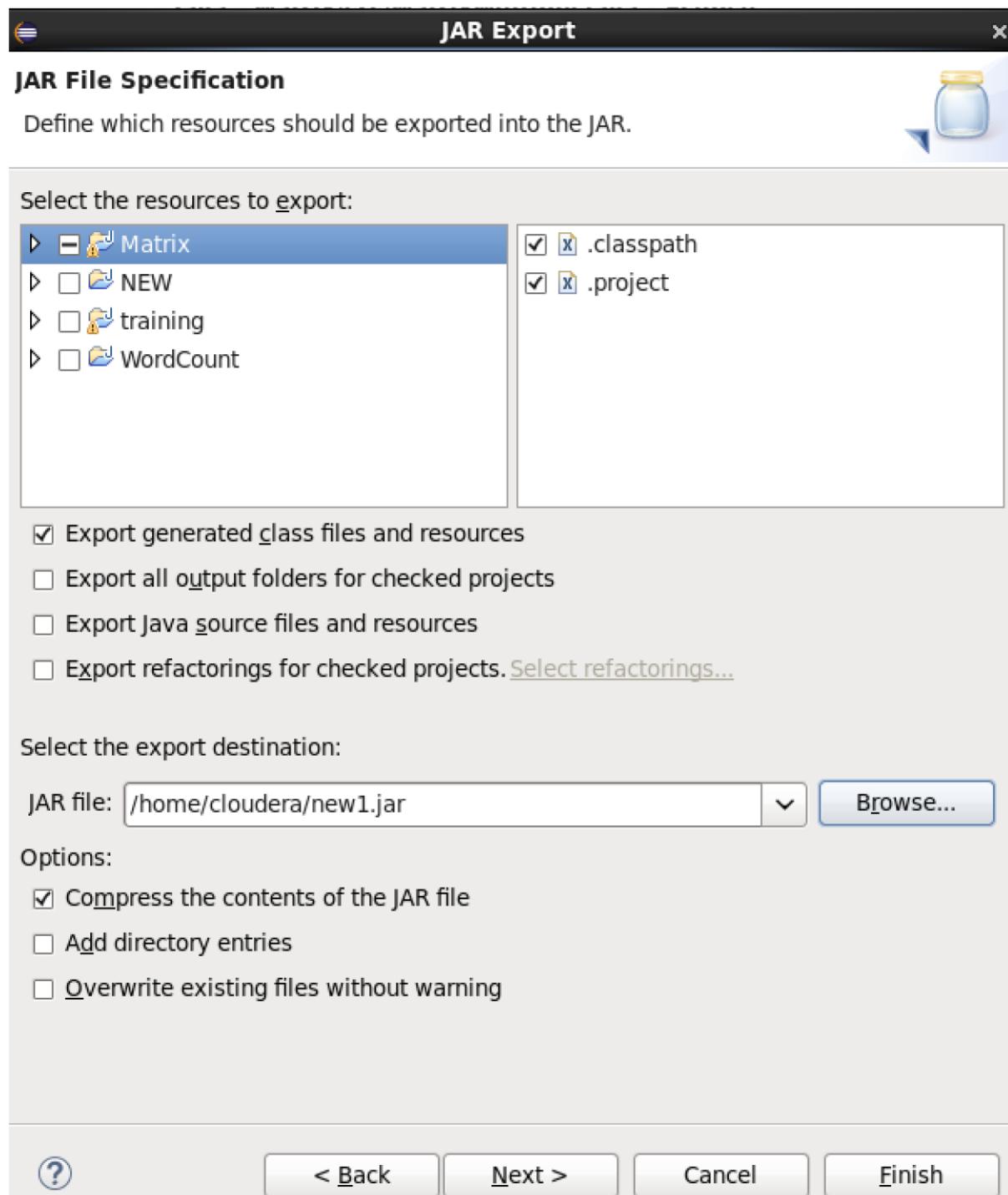
```

- 6) Right Click on the project name Matrix -> Export -> Java -> JAR File -> Next -> for select the export destination for JAR file: browse -> Name : Matrix1.jar -> save in folder -> cloudera -> Finish -> OK

Name: - Zen Dsouza  
Roll No: - 34  
Subject - BDT



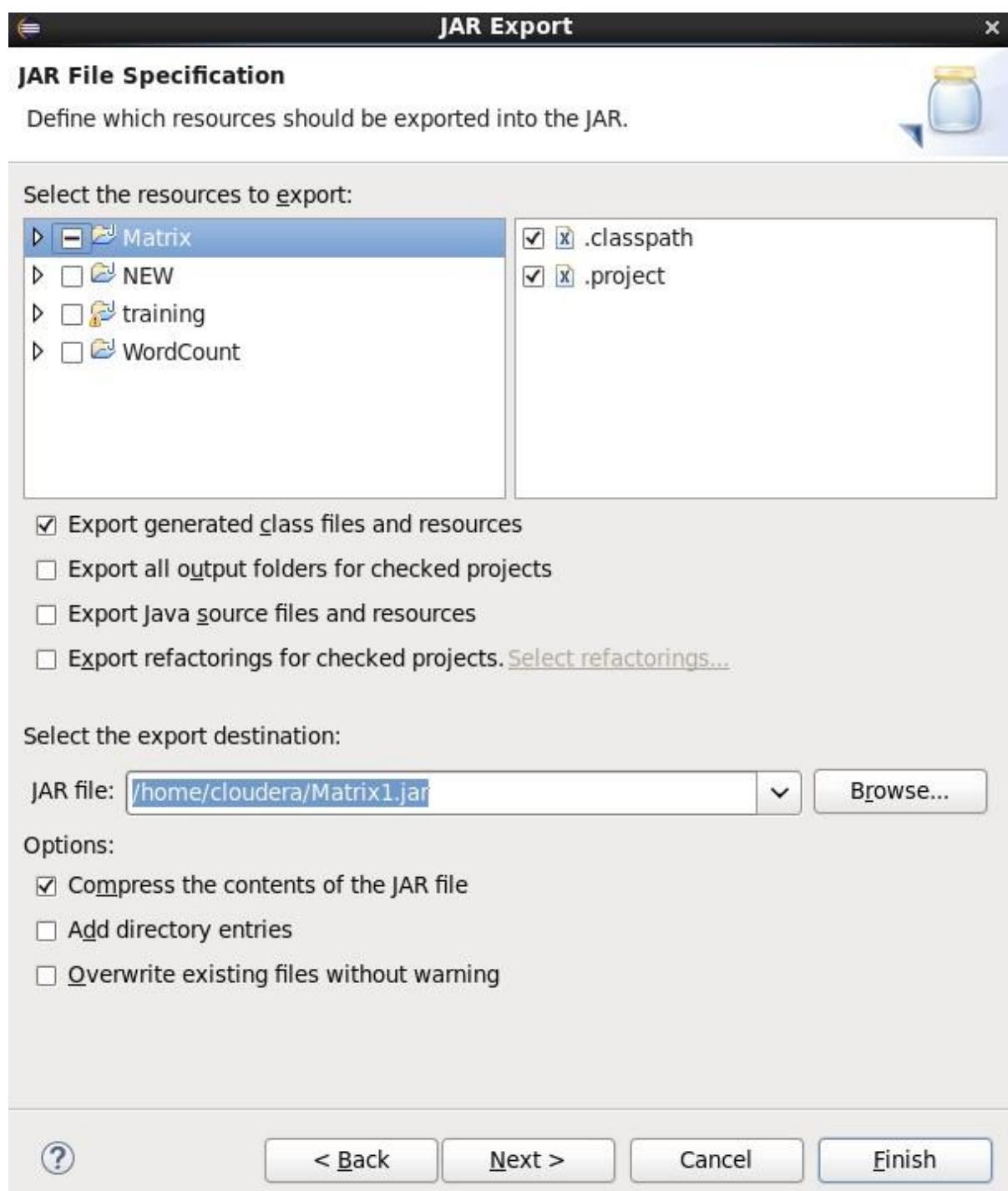
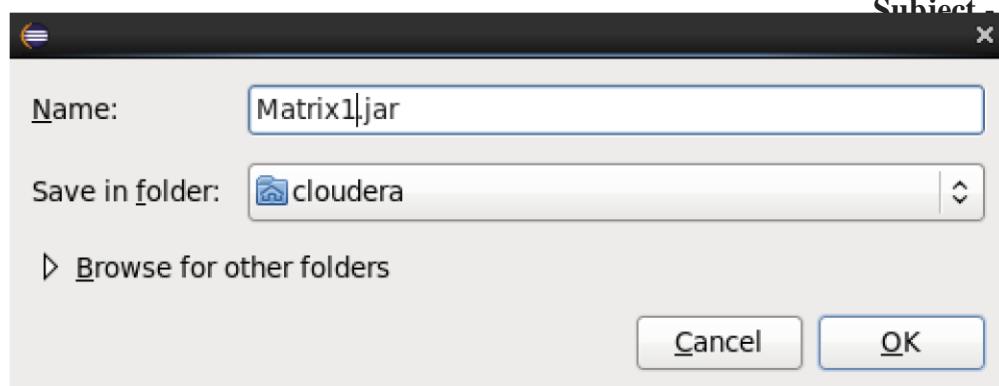
Name: - Zen Dsouza  
Roll No: - 34  
Subject - BDT



Name: - Zen Dsouza

Roll No: - 34

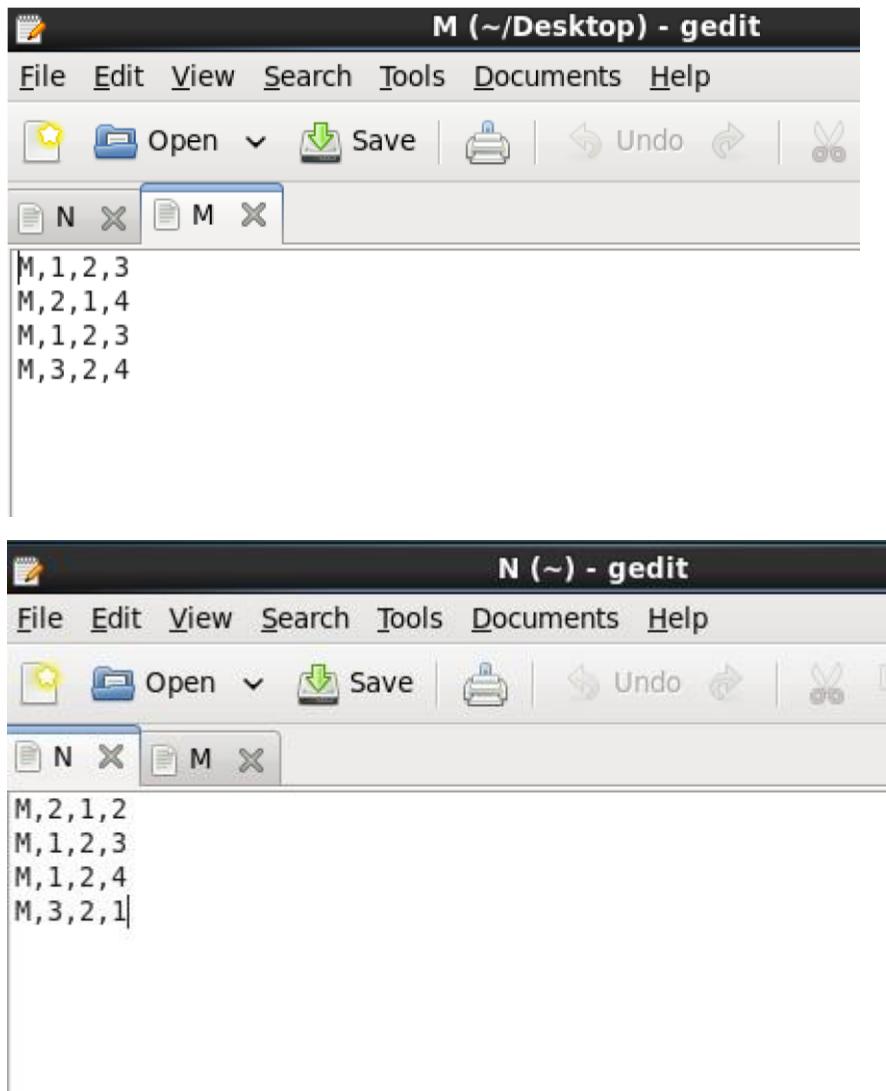
Subject - BDT



Name: - Zen Dsouza  
Roll No: - 34  
Subject - BDT

7) We need to create an input file in local file system

Creating an input file named as “M” and “N”



8) Open Terminal

Now we have to move this input file to **hdfs**. For this we create a directory on **hdfs** using command **hdfs dfs -mkdir /testip**.

9) Here listing all the directory present in **hdfs** using **hdfs dfs -ls /** command

Then we can verify whether this directory is created or not using ls command **hdfs dfs -ls /**

```
[cloudera@quickstart ~]$ hdfs dfs -mkdir /testip
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 12 items
drwxr-xr-x  - hbase    supergroup          0 2022-04-20 22:20 /hbase
drwxr-xr-x  - cloudera supergroup          0 2022-02-17 20:07 /inputdir
drwxr-xr-x  - cloudera supergroup          0 2022-02-17 20:24 /outputdir
drwxr-xr-x  - cloudera supergroup          0 2022-02-21 19:10 /rjc2122
drwxr-xr-x  - cloudera supergroup          0 2022-02-14 20:28 /rjcnew
drwxr-xr-x  - cloudera supergroup          0 2022-02-14 20:14 /rjcnew2
-rw-r--r--  4 cloudera supergroup         43 2022-02-14 19:03 /rjsuraj
drwxr-xr-x  - solr     solr                0 2015-06-09 03:38 /solr
drwxr-xr-x  - cloudera supergroup          0 2022-04-20 23:22 /testip
drwxrwxrwx  - hdfs    supergroup          0 2022-03-29 20:52 /tmp
drwxr-xr-x  - hdfs    supergroup          0 2022-03-14 21:22 /user
drwxr-xr-x  - hdfs    supergroup          0 2015-06-09 03:36 /var
```

- 10) Move the input file to this directory created in **hdfs** by using either put command or **copyFromLocal** command.

```
[cloudera@quickstart ~]$ hdfs dfs -put /home/cloudera/M /testip
[cloudera@quickstart ~]$ hdfs dfs -put /home/cloudera/N /testip
```

- 11) Now checking whether the “M” and “N” present in **/testip** directory of **hdfs** or not using **hdfs dfs -ls /testip** command.

```
[cloudera@quickstart ~]$ hdfs dfs -ls /testip
Found 2 items
-rw-r--r--  1 cloudera supergroup      32 2022-04-20 23:22 /testip/M
-rw-r--r--  1 cloudera supergroup      32 2022-04-20 23:23 /testip/N
[cloudera@quickstart ~]$
```

- 12) As we can see “M” and “N” file is present in **/testip** directory of **hdfs**. Now we will see the content of this file using **hdfs dfs -cat /testip/M** command

```
[cloudera@quickstart ~]$ hdfs dfs -cat /testip/M
M,1,2,3
M,2,1,4
M,1,2,3
M,3,2,4
[cloudera@quickstart ~]$ hdfs dfs -cat /testip/N
M,2,1,2
M,1,2,3
M,1,2,4
M,3,2,1
```

- 13) Running Mapreduce Program on Hadoop, syntax is **hadoop jar jarFileName.jar ClassName/InputFileAddress /outputdir**.

i.e. **hadoop jar /home/cloudera/Matrix1.jar MatrixMultiply /testip/\* /testip\_out**

**Name: - Zen Dsouza**  
**Roll No: - 34**  
**Subject - BDT**

```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/MatrixMultiply.jar MatrixMultiply /tempip /tempop
22/02/24 19:56:29 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
22/02/24 19:56:30 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application using ToolRunner
22/02/24 19:56:30 INFO input.FileInputFormat: Total input paths to process : 2
22/02/24 19:56:30 INFO mapreduce.JobSubmitter: number of splits:2
22/02/24 19:56:30 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1645544927537_0006
22/02/24 19:56:30 INFO impl.YarnClientImpl: Submitted application application_1645544927537_0006
22/02/24 19:56:30 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1645544927537_0006/
22/02/24 19:56:37 INFO mapreduce.Job: Job job_1645544927537_0006 running in uber mode : false
22/02/24 19:56:37 INFO mapreduce.Job: map 0% reduce 0%
22/02/24 19:56:47 INFO mapreduce.Job: map 50% reduce 0%
22/02/24 19:56:48 INFO mapreduce.Job: map 100% reduce 0%
22/02/24 19:56:54 INFO mapreduce.Job: map 100% reduce 100%
22/02/24 19:56:54 INFO mapreduce.Job: Job job_1645544927537_0006 completed successfully
22/02/24 19:56:54 INFO mapreduce.Job: Counters: 49
  File System Counters
    FILE: Number of bytes read=111126
    FILE: Number of bytes written=554795
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=274
    HDFS: Number of bytes written=36
    HDFS: Number of read operations=9
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=2
  Job Counters
    Launched map tasks=2
    Launched reduce tasks=1
    Data-local map tasks=2
    Total time spent by all maps in occupied slots (ms)=14429
    Total time spent by all reduces in occupied slots (ms)=4356
    Total time spent by all map tasks (ms)=14429
    Total time spent by all reduce tasks (ms)=4356
    Total vcore-seconds taken by all map tasks=14429
    Total vcore-seconds taken by all reduce tasks=4356
    Total megabyte-seconds taken by all map tasks=14775296
    Total megabyte-seconds taken by all reduce tasks=4466544
  Map-Reduce Framework
    Map input records=8
    Map output records=8000
    Map output bytes=95120
    Map output materialized bytes=111132
    Input split bytes=210
    Combine input records=0
    Combine output records=0
    Reduce input groups=3996
    Reduce shuffle bytes=111132
```

## Map Reduce Framework

Name: - Zen Dsouza  
Roll No: - 34  
Subject - BDT

```
Launched map tasks=2
Launched reduce tasks=1
Data-local map tasks=2
Total time spent by all maps in occupied slots (ms)=14429
Total time spent by all reduces in occupied slots (ms)=4356
Total time spent by all map tasks (ms)=14429
Total time spent by all reduce tasks (ms)=4356
Total vcore-seconds taken by all map tasks=14429
Total vcore-seconds taken by all reduce tasks=4356
Total megabyte-seconds taken by all map tasks=14775296
Total megabyte-seconds taken by all reduce tasks=4460544
Map-Reduce Framework
  Map input records=8
  Map output records=8000
  Map output bytes=95120
  Map output materialized bytes=111132
  Input split bytes=210
  Combine input records=0
  Combine output records=0
  Reduce input groups=3996
  Reduce shuffle bytes=111132
  Reduce input records=8000
  Reduce output records=4
  Spilled Records=16000
  Shuffled Maps =2
  Failed Shuffles=0
  Merged Map outputs=2
  GC time elapsed (ms)=170
  CPU time spent (ms)=1970
  Physical memory (bytes) snapshot=561012736
  Virtual memory (bytes) snapshot=4507758592
  Total committed heap usage (bytes)=391979008
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=64
File Output Format Counters
  Bytes Written=36
```

14) Now we can check what we have inside this **tempop2** directory using command as **hdfs dfs -ls /tempop2**.

```
[cloudera@quickstart ~]$ hdfs dfs -cat /tempop2/part-r-00000
1,1,36.0
1,2,41.0
2,1,64.0
2,2,73.0
```