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

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.ArrayWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Reducer.Context;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Matrix_Multiplication {
public static int Matrix1noofrows=0;
public static int Matrix1noofcolumns=0;
public static int Matrix2noofrows=0;
public static int Matrix2noofcolumns=0;
public static class Map_Function
extends Mapper < LongWritable, Text, Text, IntWritable >
{

private final static IntWritable one = new IntWritable( 1);
private Text word = new Text();

public ArrayList<String> input1 = new ArrayList<String>();
public ArrayList<String> input2 = new ArrayList<String>();
@Override
public void map( LongWritable key, Text value, Context context
) throws IOException,
InterruptedException
{
   //Map Function Code
   //Checking for Empty line
   if(String.valueOf(value).contains("\\n\n"))
      input1.add(" ");

   //Adding items based on empty line
```

```
    if(String.valueOf(input1).contains(", ]")==false)
       input1.add(String.valueOf(value));
    else
       input2.add(String.valueOf(value));

    //Checking size and matrix j columns and j rows condition
    String Z1[] = String.valueOf(input1).split(",");
    int Z1length = Z1.length;
    String Z2[] = Z1[0].split("\\s");
    int columncountofMatriz1 = Z2.length;

    String Z3[] = String.valueOf(input2).split(",");
    int rowcountofMatrix2 = Z3.length;

    //Executing map code once all the data is read
    if(columncountofMatriz1==rowcountofMatrix2)
    {
       System.out.println("****************************************Inside
Loop*************************************");
       System.out.println("input1: "+String.valueOf(input1));
       System.out.println("input2: "+String.valueOf(input2));

       String inputmatrix1 = String.valueOf(input1).replace("[", "");
       inputmatrix1 = inputmatrix1.replace("]", "");
       String inputmatrix2 = String.valueOf(input2).replace("[", "");
       inputmatrix2 = inputmatrix2.replace("]", "");
       System.out.println("inputmatrix1: "+inputmatrix1);
       System.out.println("inputmatrix2: "+inputmatrix2);

       //Calculate number of rows and columns of Matrix 1
       String matrix1[] = inputmatrix1.split(",");
       Matrix1noofrows = (matrix1.length-1); //because there is an extra comma for the empty line
       String temp1[] = matrix1[0].split("\\s");
       Matrix1noofcolumns = temp1.length;
       System.out.println("Matrix1noofrows: "+Matrix1noofrows);
       System.out.println("Matrix1noofcolumns: "+Matrix1noofcolumns);

       //Calculate number of rows and columns of Matrix 2
       String matrix2[] = inputmatrix2.split(",");
       Matrix2noofrows = matrix2.length;
       String temp2[] = matrix2[0].split("\\s");
       Matrix2noofcolumns = temp2.length;
       System.out.println("Matrix2noofrows: "+Matrix2noofrows);
       System.out.println("Matrix2noofcolumns: "+Matrix2noofcolumns);

       ArrayList<Integer> MapperOutput = new ArrayList<Integer>();
       //multiply and store
       int z=1;
```

```
      for(int k=0;k<Matrix1noofrows;k++)
      {
         String Mi[] = matrix1[k].trim().split("\\s");
         for(int n=0;n<Matrix2noofcolumns;n++)
         {
            for(int m=0;m<Matrix2noofrows;m++)
            {
               String temp[] = matrix2[m].trim().split("\\s");
               int Mj = Integer.parseInt(temp[n]);
               int X = Integer.parseInt(Mi[m])*Mj;
               MapperOutput.add(X);
               //send key value pairs to reducer
               one.set(X);
               word.set(String.valueOf(z));
               System.out.println("one: "+one);
               System.out.println("word: "+word);
               context.write( word,one);
            }
            z++;
         }
      }
      System.out.println("MapperOutput: "+MapperOutput);
      String MapOutput = String.valueOf(MapperOutput).replace("[", "");
      MapOutput = MapOutput.replace("]", "");
   }
}
}
public static class Reduce_Function
extends
Reducer < Text, IntWritable, Text, IntWritable > {

public ArrayList<String> OutputMatrix = new ArrayList<String>();
private IntWritable result = new IntWritable();
@Override
public void reduce( Text key, Iterable < IntWritable > values, Context context
) throws IOException,
InterruptedException
{
   int sum = 0;
   for (IntWritable val : values)
   {
      sum += val.get();
   }
   System.out.println("Sum: "+sum);
   result.set( sum);
   System.out.println("key: "+key);
   System.out.println("result: "+result);
```

```
   OutputMatrix.add(" "+key+":"+String.valueOf(result));
   String Output = String.valueOf(OutputMatrix);
   Output = Output.replace("[", "");
   Output = Output.replace("]", "");
   String OutputTemp[] = Output.split(",");
   int Olength = OutputTemp.length;
   System.out.println("Olength:********* "+Olength);
   System.out.println("Output**********: "+Output);
   int w = 1;
   int length = Matrix1noofrows*Matrix2noofcolumns;
   //Display output as a matrix  - keys are shuffled and sorted so included a for loop for it (g loop)
   if(String.valueOf(Olength).equalsIgnoreCase(String.valueOf(length)))
   {
      for(int r=0;r<Matrix1noofrows;r++)
      {
         String O="";
         for(int c=0;c<Matrix2noofcolumns;c++)
         {
            String s="";
            for(int g=0;g<Olength;g++)
            {
               System.out.println("OutputTemp: "+OutputTemp[g]);
               System.out.println("+w+:"+" "+w+":");
               if(OutputTemp[g].contains(" "+w+":"))
               {
                  String t[] = OutputTemp[g].split(":");
                  s=t[1];
                  System.out.println("s: "+s);
                  break;
               }
            }
            O = O + s +" ";
            w++;
         }
         O.trim();
         System.out.println("******** O :"+O);
         context.write(new Text(O),null);
      }
   }
}
}

public static void main( String[] args) throws Exception {
Configuration conf = new Configuration();
Job job = Job.getInstance( conf, "Matrices");

job.setJarByClass(Matrix_Multiplication.class);
```

```
FileInputFormat.addInputPath( job, new Path("input"));
FileOutputFormat.setOutputPath( job, new Path("output"));
job.setMapperClass( Map_Function.class);
//job.setNumReduceTasks(0);
//job.setSortComparatorClass(Keysorter.class);
job.setReducerClass( Reduce_Function.class);

job.setOutputKeyClass( Text.class);
job.setOutputValueClass( IntWritable.class);

System.exit( job.waitForCompletion( true) ? 0 : 1);
}
}
```