MapReduce program using each of the classes that extend FileInputFormat<k,v> (CombineFileInputFormat, NLineInputFormat, SequenceFileInputFormat, TextInputFormat)

TextInputFormat:

Mapper function:

```
package sayali.lab1;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class WordMapper extends Mapper<LongWritable,Text,Text,IntWritable>
	{
	Text word= new Text();
	IntWritable one = new IntWritable(1);
	public void map(LongWritable key, Text value, Context context) throws IOException{
		String line = value.toString();
	    String [] tokens = line.split(" ");
	     for (String token:tokens)
	     { word.set(token);
	      try {
					context.write(word, one);
				} catch (InterruptedException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}}
	     }
	}
```

Reducer Function:

```
package sayali.lab1;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
```

```
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class WordReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
                        int sum = 0;
                        for (IntWritable v : values)
                        {
                                sum += v.get();
}    IntWritable count= new IntWritable(sum);
     context.write(key, count);
}
}
```

MAIN Class:

```
package sayali.lab1;
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
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.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
public class App
{
   public static void main( String[] args ) throws IOException

   {        Configuration conf =new Configuration();
             // Create a new Job
            Job job = Job.getInstance(conf, "Word count example");
            job.setJarByClass(App.class);
            // Specify various job-specific parameters
            job.setJobName("myjob");
            //set the mapper and reducer
            job.setMapperClass(WordMapper.class);
            job.setReducerClass(WordReducer.class);
```

```
// set the format of mapper and reducer
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
//set the key nd value format of the output
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
//set the output and input format
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
// Submit the job, then poll for progress until the job is complete
try {
                job.waitForCompletion(true);
        } catch (ClassNotFoundException e) {
                e.printStackTrace();
        } catch (InterruptedException e) {

                e.printStackTrace(); } } }
```

OUTPUT: ./hadoop jar /home/sayali/Desktop/wordCount.jar sayali.lab1.App /ebook/lab1Text.txt /wordcount

```
     4769
"Defects," 1
"Information    1
"Plain    2
"Project   5
"Right    1
#60463]   1
$5,000)   1
&    6
'AS-IS',   1
("the 1
($1   1
(801) 1
(_italics_).    1
(a)   1
(and  1
(any  1
(b)   1
(c)   1
(does 1
(if   1
(or   3
(trademark/copyright) 1
(www.gutenberg.org),  1
*    59
***   4
***** 2
-    3
1--Driven  1
1.   1
```

NLineInputFormat:

whether we can control the number of mappers for a job. We can - there are a few ways of controlling the number of mappers, as needed. Using NLineInputFormat is one way. With this functionality, you can specify exactly how many lines should go to a mapper. E.g. If your file has 500 lines, and you set number of lines per mapper to 10, you have 50 mappers.

Input data:

```
sayali@ubuntu:/usr/local/bin/hadoop-3.2.1/bin$ ./hadoop fs -head /employee/EmployeeData.csv
2019-10-21 13:48:14,914 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted
= false
EmpID,First Name,Last Name,Gender,Date of Birth,Year of Joining
677509,Lois,Walker,F,6/9/1959,2003
940761,Brenda,Robinson,F,6/10/1959,2008
428945,Joe,Robinson,M,6/11/1959,2016
408351,Diane,Evans,F,6/12/1959,1999
193819,Benjamin,Russell,M,6/13/1959,2013
499687,Patrick,Bailey,M,6/14/1959,2005
539712,Nancy,Baker,F,6/15/1959,2016
380086,Carol,Murphy,F,6/16/1959,1983
477616,Frances,Young,F,6/17/1959,1994
162402,Diana,Peterson,F,6/10/1959,2014
231469,Ralph,Flores,M,2/5/1975,2009
153989,Jack,Alexander,M,5/19/1995,2017
386158,Melissa,King,F,2/24/1972,2015
301576,Wayne,Watson,M,6/26/1996,2017
441771,Cheryl,Scott,F,2/23/1958,1990
528509,Paula,Diaz,F,8/22/1966,1994
912990,Joshua,Stewart,M,5/18/1970,2002
214352,Theresa,Lee,F,12/5/1992,2015
890290,Julia,Scott,F,7/15/1959,2005
622406,Thomas,Lewis,M,10/4/1967,1998
979607,Carol,Edwards,F,12/14/1994,2016
969580,Matthew,Turner,M,10/26/1993,2016
426038,Joan,Stewart,F,11/20/1972,2009
388642,Ruby,Rogers,F,5/1/1980,2013
560455,Carolyn,Hayes,F,5/2/1980,2001
477253,Anne,Russell,F,5/sayali@ubuntu:/usr/local/bin/hadoop-3.2.1/bin$
```

Mapper:

```
package sayali.NlineInputFormat;
import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class MapperNLineInputFormat extends
        Mapper<LongWritable, Text, LongWritable, Text> {
```

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

                context.write(key, value); }
}
```

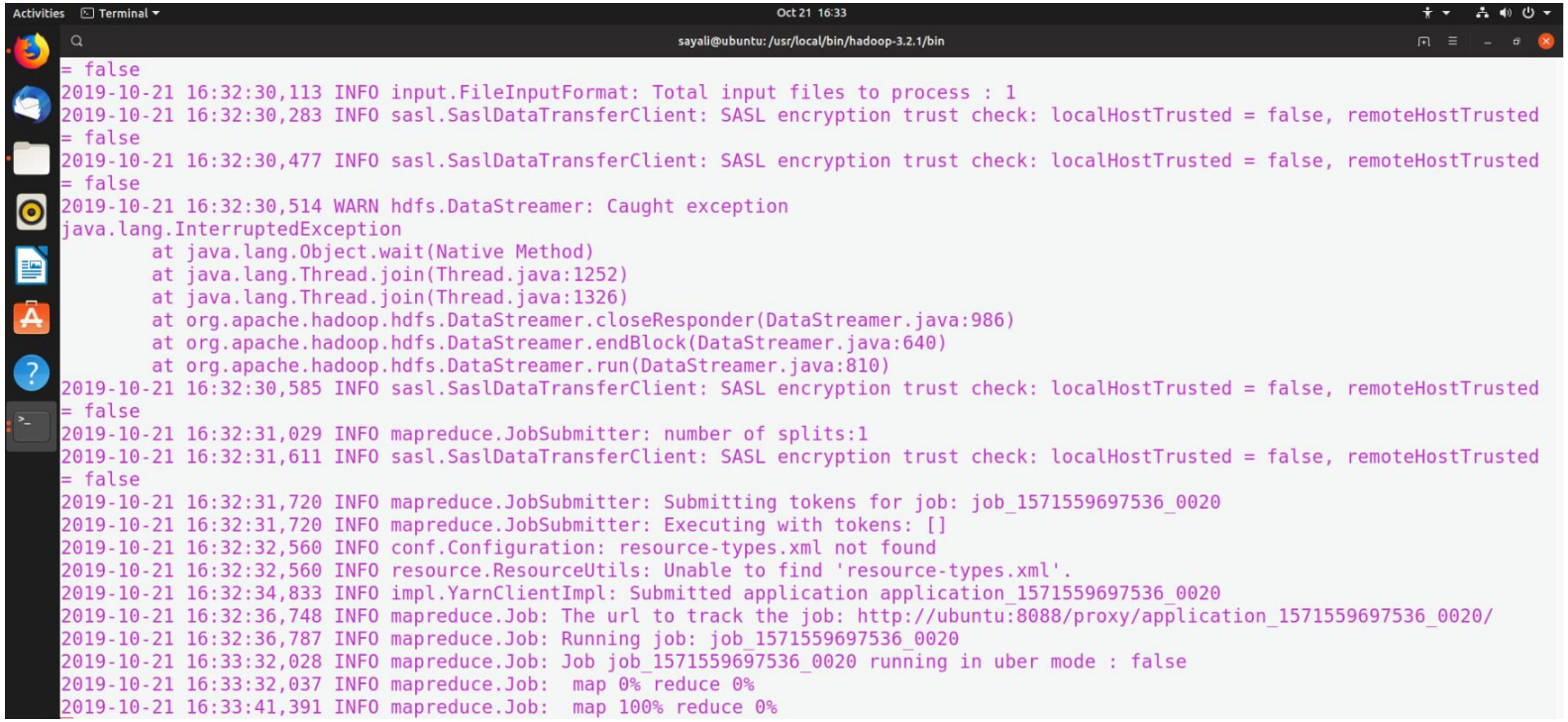
Driver:

```
package sayali.NlineInputFormat;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.NLineInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.LazyOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
public class DriverNLineInputFormat extends Configured implements Tool {
 public int run(String[] args) throws Exception {

                 // Create a new Job
                Job job = new Job(getConf());
                job.setJobName("NLineInputFormat example");
                job.setJarByClass(DriverNLineInputFormat.class);
                //set the mapper and reducer
                job.setInputFormatClass(NLineInputFormat.class);
                NLineInputFormat.addInputPath(job, new Path(args[0]));
                job.getConfiguration().setInt("mapreduce.input.lineinputformat.linespermap", 10000);

                LazyOutputFormat.setOutputFormatClass(job, TextOutputFormat.class);
                FileOutputFormat.setOutputPath(job, new Path(args[1]));

                job.setMapperClass(MapperNLineInputFormat.class);
                job.setNumReduceTasks(0);
                boolean success = job.waitForCompletion(true);
```

```
		return success ? 0 : 1;
	}

	public static void main(String[] args) throws Exception {
		int exitCode = ToolRunner.run(new Configuration(),
				new DriverNLineInputFormat(), args);
		System.exit(exitCode);
	}
}
```

Output:

./hadoop jar /home/sayali/Desktop/NLine.jar sayali.NlineInputFormat.DriverNLineInputFormat /employee/EmployeeData.csv /NLine



```
sayali@ubuntu:/usr/local/bin/hadoop-3.2.1/bin$ ./hadoop fs -head /NLine
head: `/NLine': Is a directory
sayali@ubuntu:/usr/local/bin/hadoop-3.2.1/bin$ ./hadoop fs -head /NLine/part-m-00000
2019-10-21 16:36:23,339 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted
= false
0       EmpID,First Name,Last Name,Gender,Date of Birth,Year of Joining
64      677509,Lois,Walker,F,6/9/1959,2003
99      940761,Brenda,Robinson,F,6/10/1959,2008
139     428945,Joe,Robinson,M,6/11/1959,2016
176     408351,Diane,Evans,F,6/12/1959,1999
212     193819,Benjamin,Russell,M,6/13/1959,2013
253     499687,Patrick,Bailey,M,6/14/1959,2005
292     539712,Nancy,Baker,F,6/15/1959,2016
328     380086,Carol,Murphy,F,6/16/1959,1983
365     477616,Frances,Young,F,6/17/1959,1994
403     162402,Diana,Peterson,F,6/10/1959,2014
442     231469,Ralph,Flores,M,2/5/1975,2009
478     153989,Jack,Alexander,M,5/19/1995,2017
517     386158,Melissa,King,F,2/24/1972,2015
554     301576,Wayne,Watson,M,6/26/1996,2017
591     441771,Cheryl,Scott,F,2/23/1958,1990
628     528509,Paula,Diaz,F,8/22/1966,1994
663     912990,Joshua,Stewart,M,5/18/1970,2002
702     214352,Theresa,Lee,F,12/5/1992,2015
738     890290,Julia,Scott,F,7/15/1959,2005
774     622406,Thomas,Lewis,M,10/4/1967,1998
811     979607,Carol,Edwards,F,12/14/1994,2016
850     969580,Matthew,Turner,M,10/26/1993,2016
890     426038,Joan,Stewart,F,11/20/1972,2009
928     sayali@ubuntu:/usr/local/bin/hadoop-3.2.1/bin$
```

CombineFileInputFormat:

Hadoop works best with large files but the reality is that we still have to deal with small files. When we want to process many small files in a mapreduce job, by default, each file is processed by a map task (So, 1000 small files = 1000 map tasks). Having too many tasks that finish in a matter of seconds is inefficient.

CombineFileInputFormat packs many files into a split, providing more data for a map task to process.

The sample program demonstrates that using CombineFileInput, we can process multiple small files (each file with size less than HDFS block size), in a single map task.

Input data:

For performing this opration I have divided the employee dataset used in above example into 5 diffferent parts.

Key goal of demonstration: Process 5 small files in one map task.

```
sayali@ubuntu:/usr/local/bin/hadoop-3.2.1/bin$ ./hadoop fs -ls /employee_parts
Found 5 items
-rw-r--r--   1 sayali supergroup        365 2019-10-21 17:37 /employee_parts/EmployeeData_1.csv
-rw-r--r--   1 sayali supergroup        400 2019-10-21 17:37 /employee_parts/EmployeeData_2.csv
-rw-r--r--   1 sayali supergroup        401 2019-10-21 17:37 /employee_parts/EmployeeData_3.csv
-rw-r--r--   1 sayali supergroup        327 2019-10-21 17:37 /employee_parts/EmployeeData_4.csv
-rw-r--r--   1 sayali supergroup        327 2019-10-21 17:37 /employee_parts/EmployeeData_5.csv
sayali@ubuntu:/usr/local/bin/hadoop-3.2.1/bin$ ./hadoop fs -head /employee_parts/EmployeeData_1.csv
2019-10-21 17:38:40,678 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted
= false
EmpID,First Name,Last Name,Gender,Date of Birth,Year of Joining
677509,Lois,Walker,F,6/9/1959,2003
940761,Brenda,Robinson,F,6/10/1959,2008
428945,Joe,Robinson,M,6/11/1959,2016
408351,Diane,Evans,F,6/12/1959,1999
193819,Benjamin,Russell,M,6/13/1959,2013
499687,Patrick,Bailey,M,6/14/1959,2005
539712,Nancy,Baker,F,6/15/1959,2016
380086,Carol,Murphy,F,6/16/1959,1983
sayali@ubuntu:/usr/local/bin/hadoop-3.2.1/bin$
```

Mapper:

```
package sayali.CombinedFile;
import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
```

```
public class MapperCombineFileInputFormat extends MapReduceBase implements
                Mapper<LongWritable, Text, Text, Text> {

        Text txtKey = new Text("");
        Text txtValue = new Text("");

        @Override
        public void map(LongWritable key, Text value,
        OutputCollector<Text, Text> output, Reporter reporter)
                throws IOException {

                if (value.toString().length() > 0) {
                        String[] arrEmpAttributes = value.toString().split("\\t");
                        txtKey.set(arrEmpAttributes[0].toString());
                        txtValue.set(arrEmpAttributes[2].toString() + "\t" + arrEmpAttributes[3].toString());
                        output.collect(txtKey, txtValue);
                } }  }
```

Driver:

```
package sayali.CombinedFile;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.RunningJob;
import org.apache.hadoop.mapred.TextOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class DriverCombineFileInputFormat {
 public static void main(String[] args) throws Exception {
                JobConf conf = new JobConf("DriverCombineFileInputFormat");
                conf.set("mapred.max.split.size", "134217728");//128 MB
                conf.setJarByClass(DriverCombineFileInputFormat.class);
                String[] jobArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
                conf.setMapperClass(MapperCombineFileInputFormat.class);
                conf.setInputFormat(ExtendedCombineFileInputFormat.class);
                ExtendedCombineFileInputFormat.addInputPath(conf, new Path(jobArgs[0]));
```

```
		conf.setNumReduceTasks(0);

		conf.setOutputFormat(TextOutputFormat.class);
		TextOutputFormat.setOutputPath(conf, new Path(jobArgs[1]));
		conf.setOutputKeyClass(Text.class);
		conf.setOutputValueClass(Text.class);

		RunningJob job = JobClient.runJob(conf);
		while (!job.isComplete()) {
			Thread.sleep(1000);
		}

		System.exit(job.isSuccessful() ? 0 : 2);
	}
}
```