

Write a program to find occurrence of any specific word

Code:

```
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;

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

        @Override
        public void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {
            String line = value.toString();
            String find = "big";
            for (String word : line.split("W+")) {
                if (find.equals(word.toLowerCase())) {
                    context.write(new Text(find), new IntWritable(1));
                }
            }
        }

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

            @Override
            public void reduce(Text key, Iterable<IntWritable> values, Context
                context) throws IOException, InterruptedException {

                int wordCount = 0;
                for (IntWritable value: values) {
                    wordCount += value.get();
                }
            }
        }
    }
}
```

```

}
context.write(key, new IntWritable(wordCount));
}
}

public static void main(String[] args) throws Exception{
if(args.length != 2){
System.out.printf("&quot;Usage: WordCount &lt;input dir> &lt;output

dir>\n&quot;);

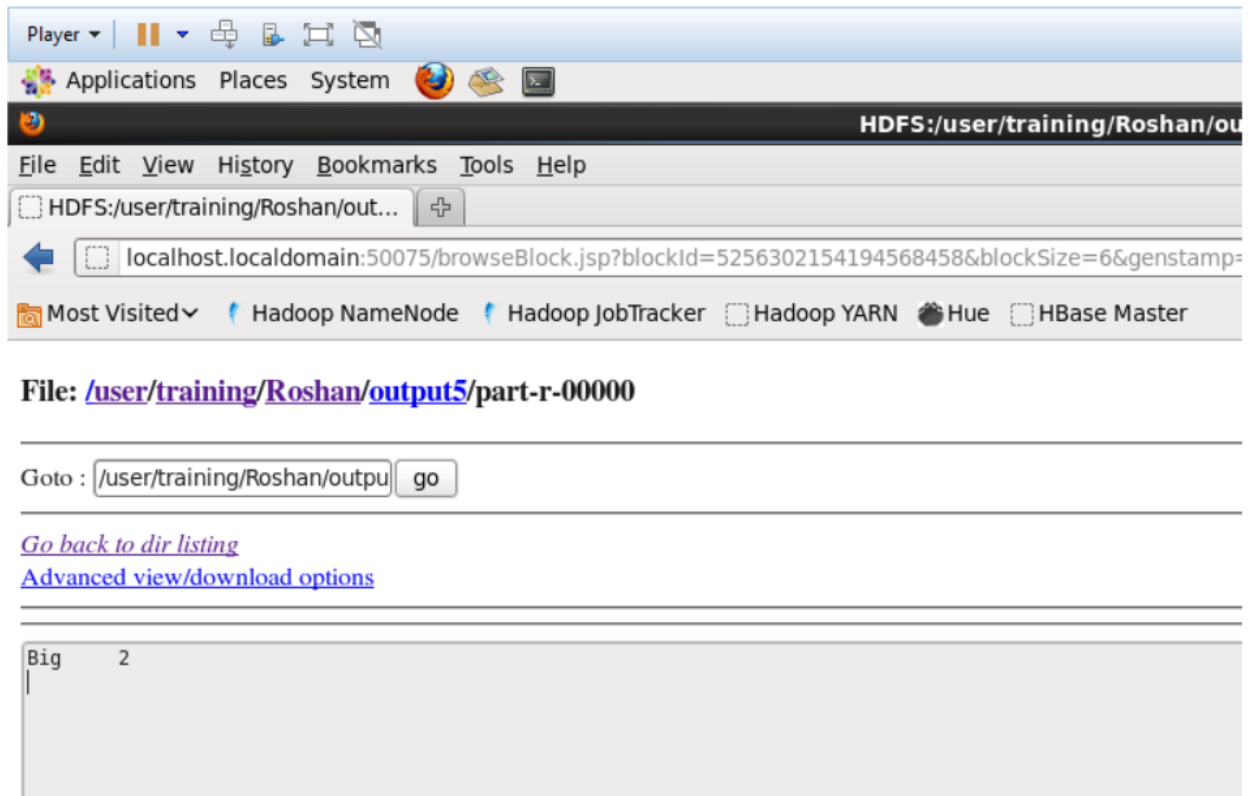
System.exit(-1);
}
Job job = new Job();
job.setJarByClass(Count.class);
job.setJobName("&quot;WordCount&quot;);
FileInputFormat.setInputPaths(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
job.setMapperClass(WordMapper.class);
job.setReducerClass(SumReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
boolean success = job.waitForCompletion(true);
System.exit(success ? 0 : 1);
}
}

```

```

[training@localhost ~]$ hadoop jar /home/training/Custs8.jar /user/training/custs /user/training/Roshan/cust-op8
22/08/08 11:08:08 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
22/08/08 11:08:08 INFO input.FileInputFormat: Total input paths to process : 1
22/08/08 11:08:08 WARN snappy.LoadSnappy: Snappy native library is available
22/08/08 11:08:08 INFO snappy.LoadSnappy: Snappy native library loaded
22/08/08 11:08:09 INFO mapred.JobClient: Running job: job_202208080931_0014
22/08/08 11:08:10 INFO mapred.JobClient: map 0% reduce 0%
22/08/08 11:08:13 INFO mapred.JobClient: map 100% reduce 0%
22/08/08 11:08:16 INFO mapred.JobClient: map 100% reduce 100%
22/08/08 11:08:16 INFO mapred.JobClient: Job complete: job_202208080931_0014
22/08/08 11:08:16 INFO mapred.JobClient: Counters: 32
22/08/08 11:08:16 INFO mapred.JobClient: File System Counters
22/08/08 11:08:16 INFO mapred.JobClient: FILE: Number of bytes read=2514
22/08/08 11:08:16 INFO mapred.JobClient: FILE: Number of bytes written=366752
22/08/08 11:08:16 INFO mapred.JobClient: FILE: Number of read operations=0
22/08/08 11:08:16 INFO mapred.JobClient: FILE: Number of large read operations=0
22/08/08 11:08:16 INFO mapred.JobClient: FILE: Number of write operations=0
22/08/08 11:08:16 INFO mapred.JobClient: HDFS: Number of bytes read=391459
22/08/08 11:08:16 INFO mapred.JobClient: HDFS: Number of bytes written=10
22/08/08 11:08:16 INFO mapred.JobClient: HDFS: Number of read operations=2
22/08/08 11:08:16 INFO mapred.JobClient: HDFS: Number of large read operations=0
22/08/08 11:08:16 INFO mapred.JobClient: HDFS: Number of write operations=1
22/08/08 11:08:16 INFO mapred.JobClient: Job Counters
22/08/08 11:08:16 INFO mapred.JobClient: Launched map tasks=1
22/08/08 11:08:16 INFO mapred.JobClient: Launched reduce tasks=1
22/08/08 11:08:16 INFO mapred.JobClient: Data-local map tasks=1
22/08/08 11:08:16 INFO mapred.JobClient: Total time spent by all maps in occupied slots (ms)=2974
22/08/08 11:08:16 INFO mapred.JobClient: Total time spent by all reduces in occupied slots (ms)=2273
22/08/08 11:08:16 INFO mapred.JobClient: Total time spent by all maps waiting after reserving slots (ms)=0

```



Count of specific profession

Code:

```
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;

public class Cust {
    public static class WordMapper extends Mapper<LongWritable,Text,Text,IntWritable>{
        @Override
        public void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException{
            String line = value.toString();
```

```

String find = "Pilot";
for (String word : line.split(",")) {
    if (find.equals(word)) {
        context.write(new Text(find), new IntWritable(1));
    }
}

}

}

}

public static class SumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
        IOException, InterruptedException {
        int wordCount = 0;
        for (IntWritable value: values) {
            wordCount += value.get();
        }
        context.write(key, new IntWritable(wordCount));
    }
}

public static void main(String[] args) throws Exception {
    if (args.length != 2) {
        System.out.printf("Usage: WordCount <input dir> <output dir>\n");
        System.exit(-1);
    }
    Job job = new Job();
    job.setJarByClass(Cust.class);
    job.setJobName("Custs");

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

    job.setMapperClass(WordMapper.class);
    job.setReducerClass(SumReducer.class);

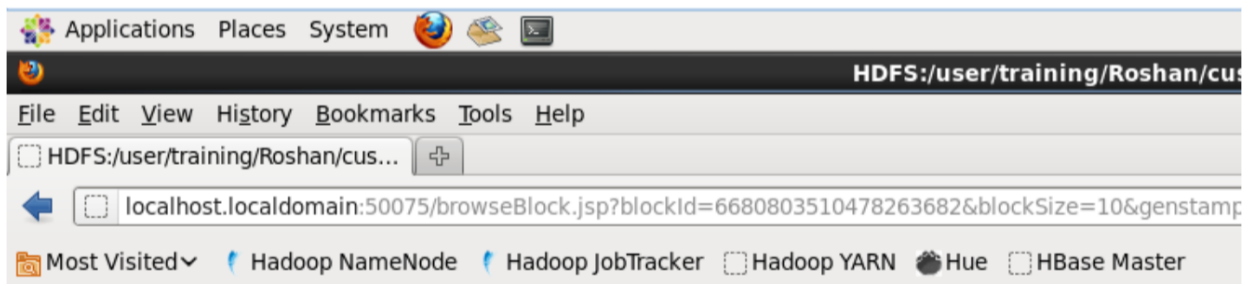
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
}

```

```
boolean success = job.waitForCompletion(true);
System.exit(success ? 0 : 1);
```

```
}
}
```

```
[training@localhost ~]$ hadoop jar /home/training/Custs8.jar /user/training/custs /user/training/Roshan/cust-op8
22/08/08 11:08:08 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
22/08/08 11:08:08 INFO input.FileInputFormat: Total input paths to process : 1
22/08/08 11:08:08 WARN snappy.LoadSnappy: Snappy native library is available
22/08/08 11:08:08 INFO snappy.LoadSnappy: Snappy native library loaded
22/08/08 11:08:09 INFO mapred.JobClient: Running job: job_202208080931_0014
22/08/08 11:08:10 INFO mapred.JobClient: map 0% reduce 0%
22/08/08 11:08:13 INFO mapred.JobClient: map 100% reduce 0%
22/08/08 11:08:16 INFO mapred.JobClient: map 100% reduce 100%
22/08/08 11:08:16 INFO mapred.JobClient: Job complete: job_202208080931_0014
22/08/08 11:08:16 INFO mapred.JobClient: Counters: 32
22/08/08 11:08:16 INFO mapred.JobClient: File System Counters
22/08/08 11:08:16 INFO mapred.JobClient: FILE: Number of bytes read=2514
22/08/08 11:08:16 INFO mapred.JobClient: FILE: Number of bytes written=366752
22/08/08 11:08:16 INFO mapred.JobClient: FILE: Number of read operations=0
22/08/08 11:08:16 INFO mapred.JobClient: FILE: Number of large read operations=0
22/08/08 11:08:16 INFO mapred.JobClient: FILE: Number of write operations=0
22/08/08 11:08:16 INFO mapred.JobClient: HDFS: Number of bytes read=391459
22/08/08 11:08:16 INFO mapred.JobClient: HDFS: Number of bytes written=10
22/08/08 11:08:16 INFO mapred.JobClient: HDFS: Number of read operations=2
22/08/08 11:08:16 INFO mapred.JobClient: HDFS: Number of large read operations=0
22/08/08 11:08:16 INFO mapred.JobClient: HDFS: Number of write operations=1
22/08/08 11:08:16 INFO mapred.JobClient: Job Counters
22/08/08 11:08:16 INFO mapred.JobClient: Launched map tasks=1
22/08/08 11:08:16 INFO mapred.JobClient: Launched reduce tasks=1
22/08/08 11:08:16 INFO mapred.JobClient: Data-local map tasks=1
```



File: [/user/training/Roshan/cust-op8/part-r-00000](#)

Goto:

[Go back to dir listing](#)

[Advanced view/download options](#)

Pilot 209

Write a program to find average word length.

Code:

```
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;

public class Average {
    public static class WordMapper extends Mapper<LongWritable, Text,
    Text, FloatWritable>
    {
        @Override
        public void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {
            Text firstLetter = new Text();
            FloatWritable wordLength = new FloatWritable();
            String line = value.toString().toLowerCase();
            for(String word:line.split(" ")) {
                if (word.length() > 0) {
                    firstLetter.set(String.valueOf(word.charAt(0)));

                    wordLength.set(word.length());
                    context.write(firstLetter, wordLength);
                }
            }
        }

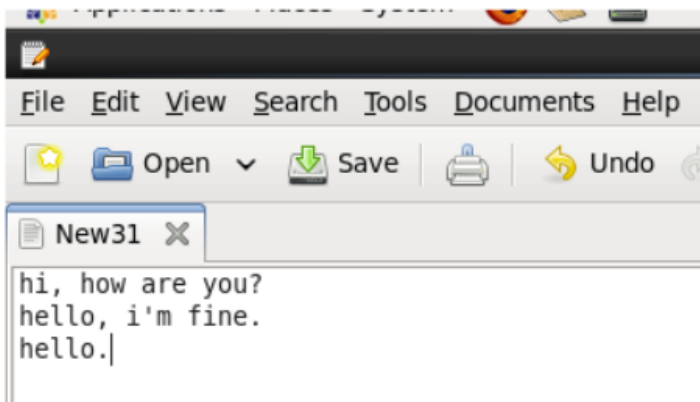
        public static class SumReducer extends
        Reducer<Text, FloatWritable, Text, FloatWritable> {
            public void reduce(Text key, Iterable<FloatWritable> values, Context
            context)
            throws IOException, InterruptedException {
                int sum =0;
                int count =0;
```

```

int Average =0;
for(FloatWritable val:values) {
    sum += val.get();
    count = count+1;
}
Average = sum/count;
context.write(key,new FloatWritable(Average));
}
}

public static void main(String[] args) throws Exception{
    if(args.length != 2){
        System.out.printf(
            &quot;Usage: WordCount &lt;input dir&gt; &lt;output dir&gt; \n&quot;);
        System.exit(-1);
    }
    Job job = new Job();
    job.setJarByClass(Average.class);
    job.setJobName(&quot;Word Count&quot;);
    FileInputFormat.setInputPaths(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    job.setMapperClass(WordMapper.class);
    job.setReducerClass(SumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(FloatWritable.class);
    boolean success = job.waitForCompletion(true);
    System.exit(success ? 0 :1);
}
}

```





Average1.jar



Average2.jar

```
[training@localhost ~]$ hadoop jar /home/training/Average2.jar /user/training/Ro
shan/Avg /user/training/Roshan/output-avg2
22/08/17 10:20:09 WARN mapred.JobClient: Use GenericOptionsParser for parsing th
e arguments. Applications should implement Tool for the same.
22/08/17 10:20:09 INFO input.FileInputFormat: Total input paths to process : 1
22/08/17 10:20:09 WARN snappy.LoadSnappy: Snappy native library is available
22/08/17 10:20:09 INFO snappy.LoadSnappy: Snappy native library loaded
22/08/17 10:20:09 INFO mapred.JobClient: Running job: job_202208170950_0010
22/08/17 10:20:10 INFO mapred.JobClient: map 0% reduce 0%
22/08/17 10:20:13 INFO mapred.JobClient: map 100% reduce 0%
22/08/17 10:20:15 INFO mapred.JobClient: Job complete: job_202208170950_0010
22/08/17 10:20:15 INFO mapred.JobClient: Counters: 32
22/08/17 10:20:15 INFO mapred.JobClient:   File System Counters
22/08/17 10:20:15 INFO mapred.JobClient:       FILE: Number of bytes read=70
22/08/17 10:20:15 INFO mapred.JobClient:       FILE: Number of bytes written=36190
6
22/08/17 10:20:15 INFO mapred.JobClient:       FILE: Number of read operations=0
22/08/17 10:20:15 INFO mapred.JobClient:       FILE: Number of large read operatio
ns=0
22/08/17 10:20:15 INFO mapred.JobClient:       FILE: Number of write operations=0
22/08/17 10:20:15 INFO mapred.JobClient:       HDFS: Number of bytes read=150
22/08/17 10:20:15 INFO mapred.JobClient:       HDFS: Number of bytes written=30
22/08/17 10:20:15 INFO mapred.JobClient:       HDFS: Number of read operations=2
22/08/17 10:20:15 INFO mapred.JobClient:       HDFS: Number of large read operatio
ns=0
22/08/17 10:20:15 INFO mapred.JobClient:       HDFS: Number of write operations=1
22/08/17 10:20:15 INFO mapred.JobClient: Job Counters
```

File
Edit
View
History
Bookmarks
Tools
Help

☐ HDFS:/user/training/Roshan/out...

☐ localhost.localdomain:50075/browseBlock.jsp?blockId=-6230

Most Visited
 Hadoop NameNode
 Hadoop JobTracker
☐ H...

Goto :

[Go back to dir listing](#)

[Advanced view/download options](#)

a	3.0
f	5.0
h	4.0
i	3.0
y	4.0

Encrypt the email ids

use "Only mapper" design pattern

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class Encry {
    public static class MapForEncryption extends Mapper<LongWritable, Text, Text, Text> {
        {
            public void map(LongWritable key, Text Value, Context con) throws
                IOException, InterruptedException

            {
                String line=Value.toString();

                int encry; //created a object which will store the encrypted character
                String mask = """; //mask object initialized with null value which will store the actual
                output text
                for(int i=0;i<line.length();i++) //for loop which will iterate through line
                {
                    char c = line.charAt(i); //store each character at index i
                    encry = 4 + (int)c; //increment the alphabet by 4
                    c = (char)encry; //type cast encry into char
                    mask += c; //store the final text in mask
                }
                //String line1 = line.replace(""||",",");
                Text outputKey = new Text(mask);
                Text outputValue = new Text(""");
                con.write(outputKey,outputValue);
            }
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration c = new Configuration();

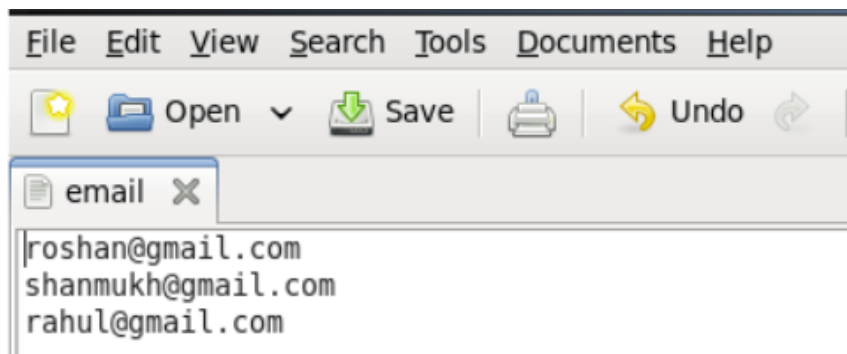
        String[] files = new GenericOptionsParser(c,args).getRemainingArgs();
        Path input = new Path(files[0]);
        Path output = new Path(files[1]);
```

```

Job j = new Job(c,&quot;FormatEmpFile&quot;);
j.setJarByClass(Encry.class);
j.setJobName(&quot;&quot;);

j.setMapperClass(MapForEncryption.class);
j.setNumReduceTasks(0);
j.setMapOutputKeyClass(Text.class);
j.setMapOutputValueClass(Text.class);
FileInputFormat.addInputPath(j,input);
FileOutputFormat.setOutputPath(j,output);
System.exit(j.waitForCompletion(true)? 0:1);
}}

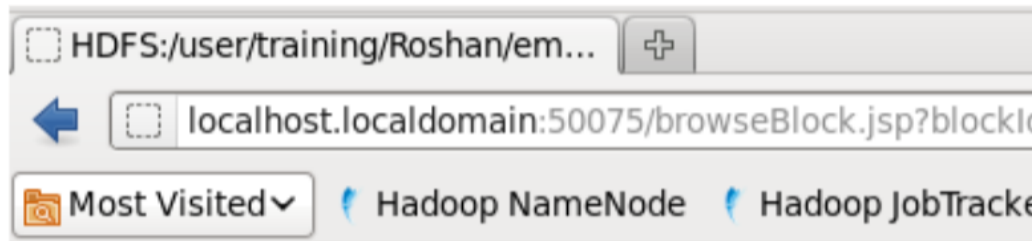
```



```

[training@localhost ~]$ hadoop jar /home/training/encry.jar /user/training/Roshan/email /user/training/Roshan/email-op9
22/08/23 11:51:49 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement
22/08/23 11:51:49 INFO input.FileInputFormat: Total input paths to process : 1
22/08/23 11:51:49 WARN snappy.LoadSnappy: Snappy native library is available
22/08/23 11:51:49 INFO snappy.LoadSnappy: Snappy native library loaded
22/08/23 11:51:50 INFO mapred.JobClient: Running job: job_202208230930_0016
22/08/23 11:51:51 INFO mapred.JobClient: map 0% reduce 0%
22/08/23 11:51:55 INFO mapred.JobClient: map 100% reduce 0%
22/08/23 11:51:56 INFO mapred.JobClient: Job complete: job_202208230930_0016
22/08/23 11:51:56 INFO mapred.JobClient: Counters: 24
22/08/23 11:51:56 INFO mapred.JobClient:   File System Counters
22/08/23 11:51:56 INFO mapred.JobClient:     FILE: Number of bytes read=0
22/08/23 11:51:56 INFO mapred.JobClient:     FILE: Number of bytes written=180563
22/08/23 11:51:56 INFO mapred.JobClient:     FILE: Number of read operations=0
22/08/23 11:51:56 INFO mapred.JobClient:     FILE: Number of large read operations=0
22/08/23 11:51:56 INFO mapred.JobClient:     FILE: Number of write operations=0
22/08/23 11:51:56 INFO mapred.JobClient:     HDFS: Number of bytes read=163
22/08/23 11:51:56 INFO mapred.JobClient:     HDFS: Number of bytes written=55
22/08/23 11:51:56 INFO mapred.JobClient:     HDFS: Number of read operations=2
22/08/23 11:51:56 INFO mapred.JobClient:     HDFS: Number of large read operations=0
22/08/23 11:51:56 INFO mapred.JobClient:     HDFS: Number of write operations=1

```



File: [/user/training/Roshan/email-op9/part-m-00000](#)

Goto :

[Go back to dir listing](#)

[Advanced view/download options](#)

```
vswlerDkqemp2gsq  
wlerqyolDkqemp2gsq  
velypDkqemp2gsq
```

Multiple file input demo assignment

Create 2 or 3 input files on your own , in which the data is present in different formats. Write a program to process these files using different map classes and perform any one aggregate function like sum, max, min etc. on it.

```
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.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class MultipleInputFiles {
    public static class map1 extends
    Mapper<LongWritable,Text,Text,IntWritable>
    {
        public void map(LongWritable key, Text value, Context con)

        throws IOException, InterruptedException

        {
            String line = value.toString();
            String[] line1 = line.split(",");
            String name = line1[0];
            Text outputKey = new Text(name);
            int cost = Integer.parseInt(line1[2]);
            IntWritable outputValue = new IntWritable(cost);
            con.write(outputKey, outputValue);
        }
    }
    public static class Map2 extends Mapper<LongWritable, Text, Text,
    IntWritable>
    {
```

```
public void map(LongWritable key, Text value, Context con)
```

```
throws IOException, InterruptedException
```

```
{
String line = value.toString();
String[] line1 = line.split(",");
String name = line1[0];
Text outputKey = new Text(name);
int cost = Integer.parseInt(line1[2]);
IntWritable outputValue = new IntWritable(cost);
con.write(outputKey, outputValue);
}
}
public static class Map3 extends Mapper<LongWritable, Text, Text,
IntWritable>
```

```
{
public void map(LongWritable key, Text value, Context con)
```

```
throws IOException, InterruptedException
```

```
{
String line = value.toString();
String[] line1 = line.split(",");
String name = line1[1];
Text outputKey = new Text(name);
int cost = Integer.parseInt(line1[3]);
IntWritable outputValue = new IntWritable(cost);
con.write(outputKey, outputValue);
}
}
public static class Red extends Reducer<Text, IntWritable,
Text,IntWritable>
{
public void reduce(Text name, Iterable<IntWritable> total_cost,
```

```
Context con) throws IOException, InterruptedException
```

```
{
```

```

//to get the sum of cost
// int sum = 0;
// for(IntWritable value : total_cost){
// sum += value.get();

//}
//to get the max cost

int max = 0;
for (IntWritable value : total_cost) {
if (value.get() > max) {
max = value.get();
}
}
con.write(name,new IntWritable(max));
}
}

public static void main(String[] args) throws Exception{
Configuration c = new Configuration();
GenericOptionsParser parser = new GenericOptionsParser(c,args);
String[] files = parser.getRemainingArgs();
Path p1 = new Path(files[0]);
Path p2 = new Path(files[1]);
Path p3 = new Path(files[2]);
Path p4 = new Path(files[3]);
Job j = new Job(c,"multiple");
j.setJarByClass(MultipleInputFiles.class);
j.setMapperClass(map1.class);
j.setMapperClass(Map2.class);
j.setReducerClass(Red.class);
j.setOutputKeyClass(Text.class);
j.setOutputValueClass(IntWritable.class);
MultipleInputs.addInputPath(j,p1,TextInputFormat.class,map1.class);
MultipleInputs.addInputPath(j, p2,

TextInputFormat.class,Map2.class);

MultipleInputs.addInputPath(j, p3,

```

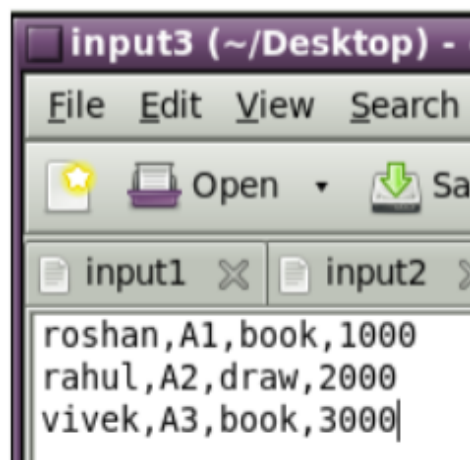
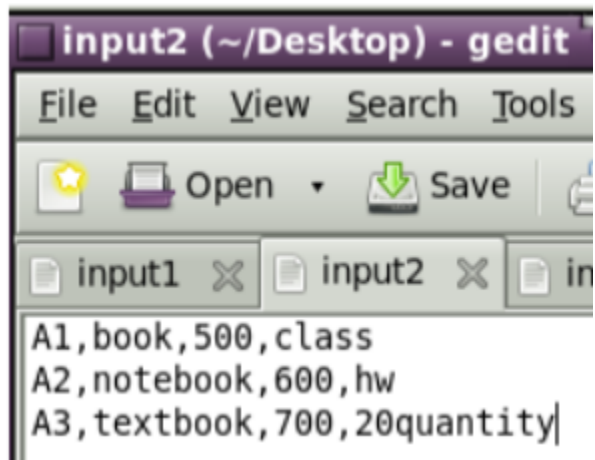
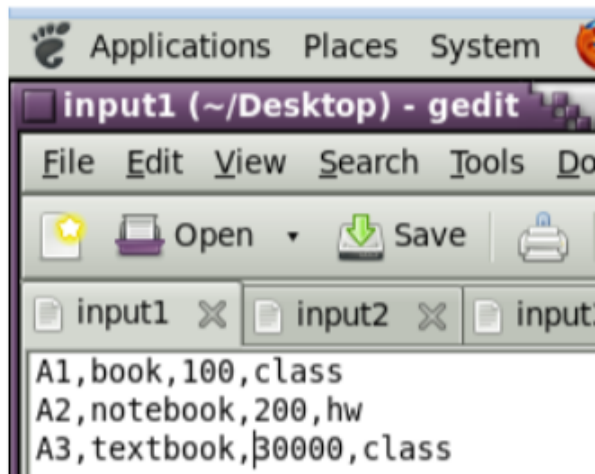
```
TextInputFormat.class,Map3.class);
```

```
FileOutputFormat.setOutputPath(j,p4);
```

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

```
}
```

```
}
```

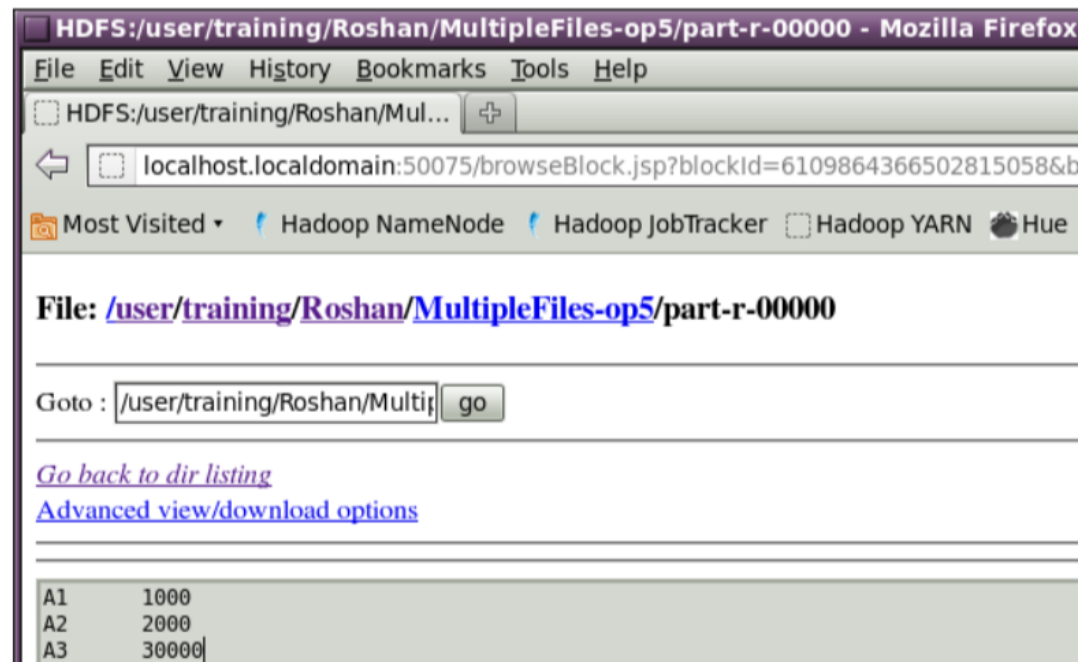


```

[training@localhost ~]$ hadoop jar /home/training/MultipleInput.jar /user/traini
ng/Roshan/Multiple-ipp1 /user/training/Roshan/Multiple-ip2 /user/training/Roshan
/Multiple-ip3 /user/training/Roshan/MultipleFiles-op5
22/08/29 10:47:49 WARN mapred.JobClient: Use GenericOptionsParser for parsing th
e arguments. Applications should implement Tool for the same.
22/08/29 10:47:49 INFO input.FileInputFormat: Total input paths to process : 1
22/08/29 10:47:49 WARN snappy.LoadSnappy: Snappy native library is available
22/08/29 10:47:49 INFO snappy.LoadSnappy: Snappy native library loaded
22/08/29 10:47:49 INFO input.FileInputFormat: Total input paths to process : 1
22/08/29 10:47:49 INFO input.FileInputFormat: Total input paths to process : 1
22/08/29 10:47:49 INFO mapred.JobClient: Running job: job_202208290924_0005
22/08/29 10:47:50 INFO mapred.JobClient: map 0% reduce 0%
22/08/29 10:47:54 INFO mapred.JobClient: map 66% reduce 0%
22/08/29 10:47:57 INFO mapred.JobClient: map 100% reduce 0%
22/08/29 10:47:58 INFO mapred.JobClient: map 100% reduce 100%
22/08/29 10:47:59 INFO mapred.JobClient: Job complete: job_202208290924_0005
22/08/29 10:47:59 INFO mapred.JobClient: Counters: 32
22/08/29 10:47:59 INFO mapred.JobClient: File System Counters
22/08/29 10:48:00 INFO mapred.JobClient: FILE: Number of bytes read=87
22/08/29 10:48:00 INFO mapred.JobClient: FILE: Number of bytes written=73445
4
22/08/29 10:48:00 INFO mapred.JobClient: FILE: Number of read operations=0
22/08/29 10:48:00 INFO mapred.JobClient: FILE: Number of large read operatio
ns=0

```

For max cost:



HDFS:/user/training/Roshan/MultipleFiles-op5/part-r-00000 - Mozilla Firefox

File Edit View History Bookmarks Tools Help

HDFS:/user/training/Roshan/Mul... +

localhost.localdomain:50075/browseBlock.jsp?blockId=6109864366502815058&b

Most Visited Hadoop NameNode Hadoop JobTracker Hadoop YARN Hue

File: /user/training/Roshan/MultipleFiles-op5/part-r-00000

Goto : /user/training/Roshan/Multi go

[Go back to dir listing](#)

[Advanced view/download options](#)

A1	1000
A2	2000
A3	30000

