# <u>ASSIGNMENT 4</u>

## Problem Statement:

We have a dataset of sales of different TV sets across different locations.

Records look like:

Samsung|Optima|14|Madhya Pradesh|132401|14200

The fields are arranged like:

Company Name|Product Name|Size in inches|State|Pin Code|Price

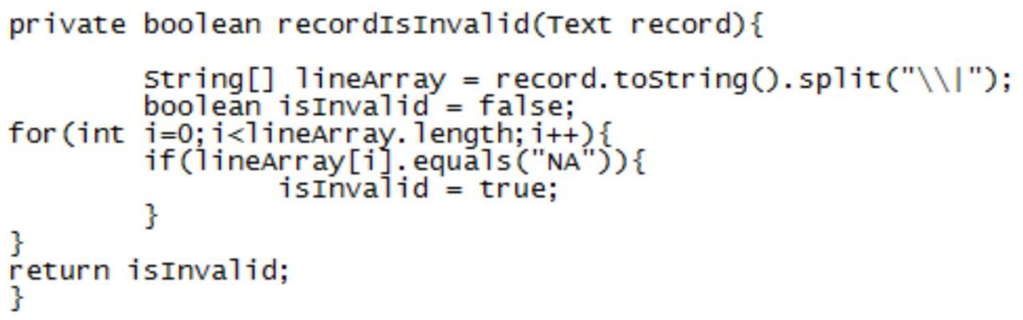
There are some invalid records which contain 'NA' in either Company Name or Product Name.

**TASK 1:**

**Write a Map Reduce program to filter out the invalid records. Map only job will fit for this context.**

**SOLUTION:**

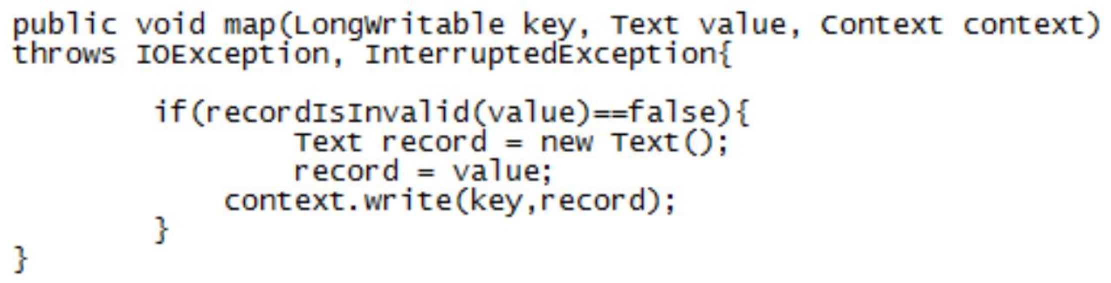
For this task I created a method that checks if a record is invalid. To achieve this, we split every record using "\" character as rule, into an array. Then we check every part of the array and we validate if that part contains "NA", then we return a Boolean variable:



```
private boolean recordIsInvalid(Text record){

        String[] lineArray = record.toString().split("\\|");
        boolean isInvalid = false;
for(int i=0;i<lineArray.length;i++){
        if(lineArray[i].equals("NA")){
                isInvalid = true;
        }
}
return isInvalid;
}
```

The mapper uses this method to filter out invalid records:



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

        if(recordIsInvalid(value)==false){
                Text record = new Text();
                record = value;
            context.write(key,record);
        }
}
```

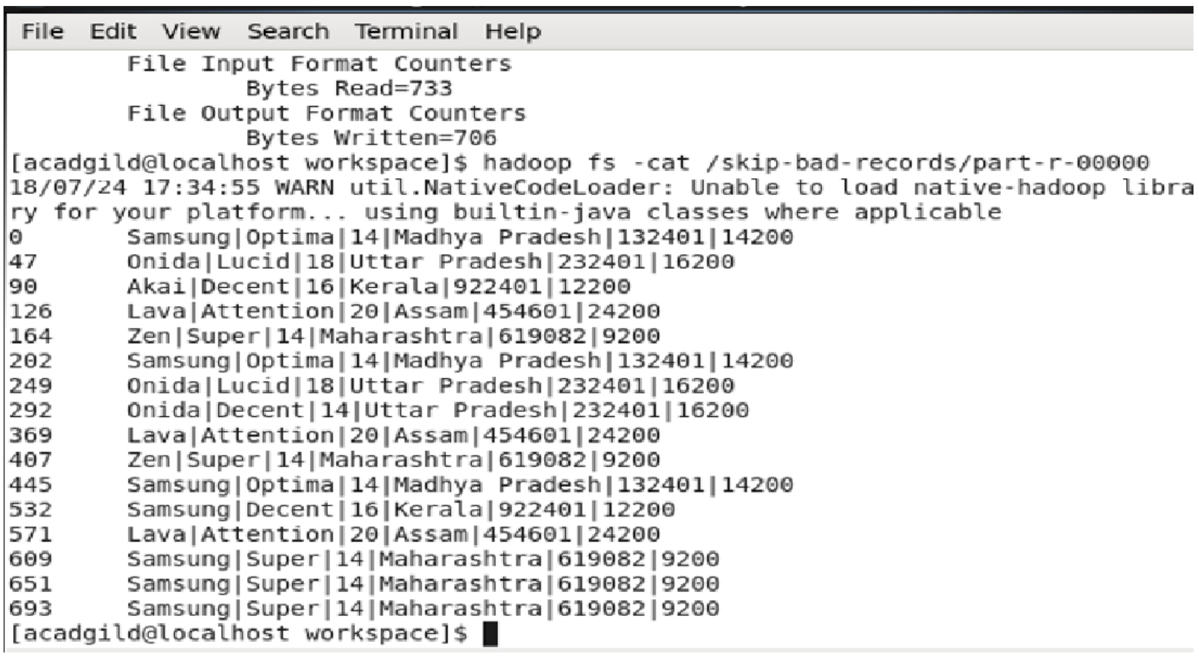
Then we need to export the .jar file from Eclipse and in the command shell we type:

**hadoop jar /home/acadgild/workspace/TvSales.jar television.txt /skip-bad-records**

- **jar** is the command to execute the mapping and reducing
- **/home/acadgild/workspace/TvSales.**jar location on file system of the jar exported before
- **television.txt** is the file location on HDFS containing the records
- **/skip-bad-records** is the destination directory on HDFS after reducing the records. Since the no reducer for this task, Hadoop uses identity Reducer .

To check the results obtained on HDFS we can type:

**hadoop fs –cat /filter-bad-records/part-r-00000**



```
        File Input Format Counters
                Bytes Read=733
        File Output Format Counters
                Bytes Written=706
[acadgild@localhost workspace]$ hadoop fs -cat /skip-bad-records/part-r-00000
18/07/24 17:34:55 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
0       Samsung|Optima|14|Madhya Pradesh|132401|14200
47      Onida|Lucid|18|Uttar Pradesh|232401|16200
90      Akai|Decent|16|Kerala|922401|12200
126     Lava|Attention|20|Assam|454601|24200
164     Zen|Super|14|Maharashtra|619082|9200
202     Samsung|Optima|14|Madhya Pradesh|132401|14200
249     Onida|Lucid|18|Uttar Pradesh|232401|16200
292     Onida|Decent|14|Uttar Pradesh|232401|16200
369     Lava|Attention|20|Assam|454601|24200
407     Zen|Super|14|Maharashtra|619082|9200
445     Samsung|Optima|14|Madhya Pradesh|132401|14200
532     Samsung|Decent|16|Kerala|922401|12200
571     Lava|Attention|20|Assam|454601|24200
609     Samsung|Super|14|Maharashtra|619082|9200
651     Samsung|Super|14|Maharashtra|619082|9200
693     Samsung|Super|14|Maharashtra|619082|9200
[acadgild@localhost workspace]$ 
```

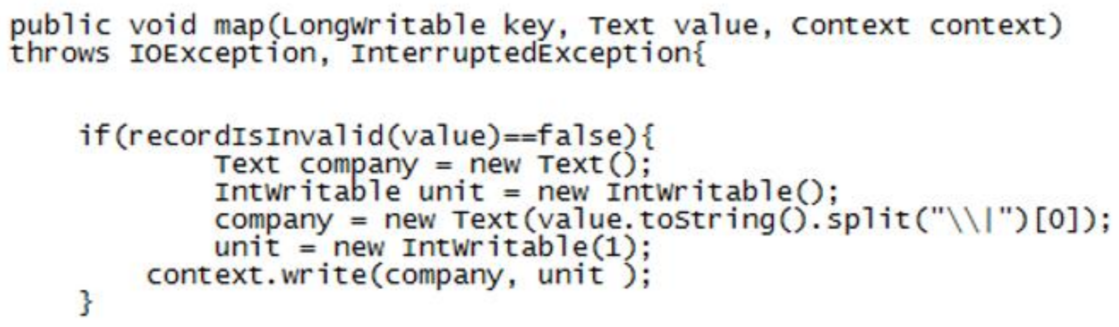
The complete code is in **filter-invalid-records-program.txt** and the filtered records in **filtered-records.txt** obtained from HDFS using **hadoop fs –copyToLocal** command.

**TASK 2:**

Write a Map Reduce program to calculate the total units sold for each Company.

**SOLUTION:**

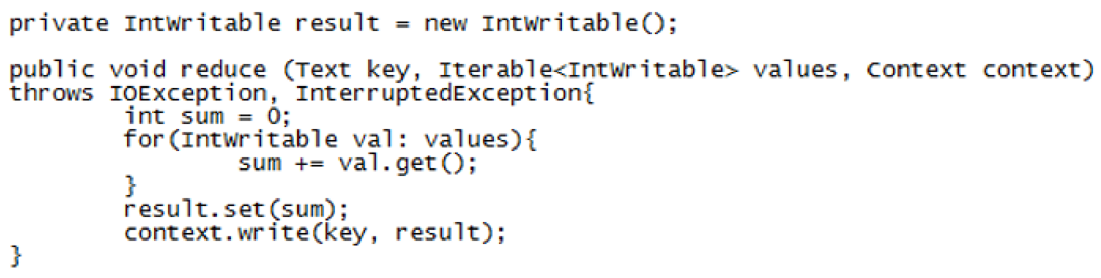
To calculate the total units sold for each, we need to do “the sort and shuffle” in the mapper class method. For this, we split the records, and we select the “key” as company and the value as “1” as integer, so for every record there’s a 1 value. By the way, we use the method to filter out bad records.



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


    if(recordIsInvalid(value)==false){
            Text company = new Text();
            IntWritable unit = new IntWritable();
            company = new Text(value.toString().split("\\|")[0]);
            unit = new IntWritable(1);
        context.write(company, unit );
    }
```

Now we need to reduce the data in the Reducer class method. The objective is to sum every value for each key, with this; we will get the total units sold by each company. To achieve this we write the following code:



```
private IntWritable result = new IntWritable();

public void reduce (Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException{
        int sum = 0;
        for(IntWritable val: values){
                sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
}
```

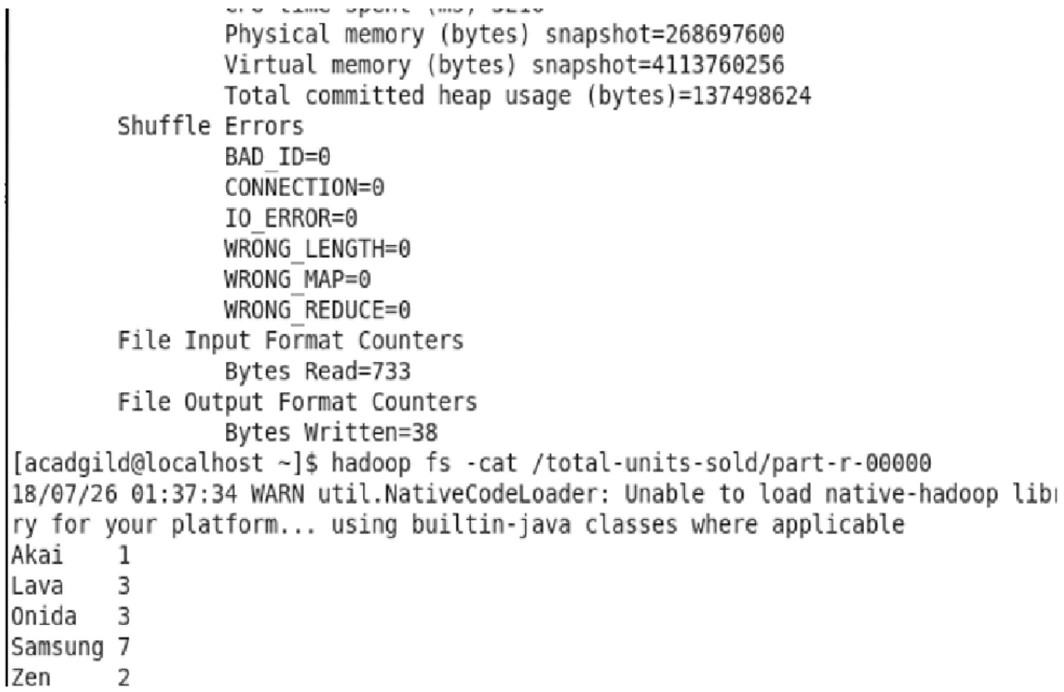
Then we need to export the .jar file from Eclipse and in the command shell we type:

**hadoop jar /home/acadgild/workspace/TvSales.jar television.txt /total-units-sold**

- **jar** is the command to execute the mapping and reducing
- **/home/acadgild/workspace/TvSales.jar** location on file system of the jar exported before.
- **television.txt** is the file location on HDFS containing the records
- **/total-units-sold** is the destination directory on HDFS after reducing the records.

To check the results obtained on HDFS we can type :

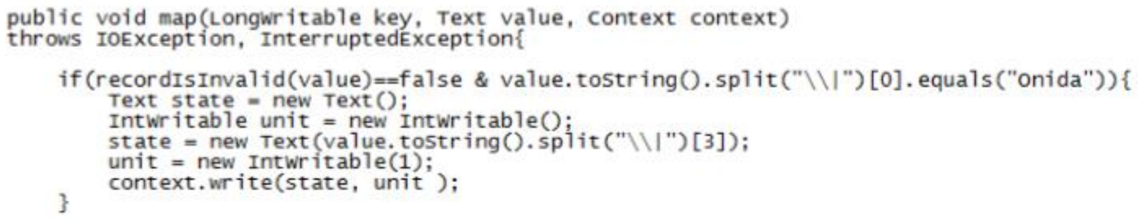
**hadoop fs –cat /total-units-sold/part-r-00000**



```
                Physical memory (bytes) snapshot=268697600
                Virtual memory (bytes) snapshot=4113760256
                Total committed heap usage (bytes)=137498624
        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=733
        File Output Format Counters
                Bytes Written=38
[acadgild@localhost ~]$ hadoop fs -cat /total-units-sold/part-r-00000
18/07/26 01:37:34 WARN util.NativeCodeLoader: Unable to load native-hadoop lib
ry for your platform... using builtin-java classes where applicable
Akai    1
Lava    3
Onida   3
Samsung 7
Zen     2
```

The complete code is in **total-units-sold-program.txt** and the results are in **total-units-sold.txt** obtained from HDFS using **hadoop fs – copyToLocal** command.

In this case, we will use the same reduce method than before, but we need to change the mapper method.



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

    if(recordIsInvalid(value)==false & value.toString().split("\\|")[0].equals("Onida")){
        Text state = new Text();
        IntWritable unit = new IntWritable();
        state = new Text(value.toString().split("\\|")[3]);
        unit = new IntWritable(1);
        context.write(state, unit );
    }
```

So besides filtering out invalid records, we filter records by company. In this case, we will only accept records with “Onida” as a company. As as key, we will assing the state, so we can sum every record by state, and the value the same as before.

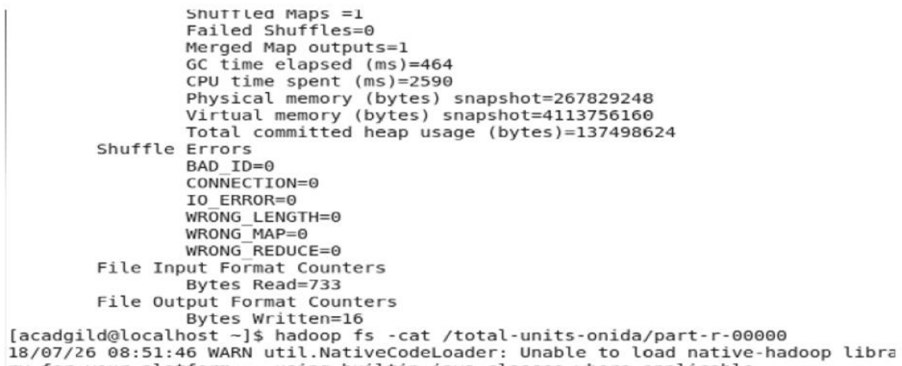
Then we need to export the .jar file from Eclipse and in the command shell we type:

**hadoop jar /home/acadgild/workspace/TvSales.jar television.txt /total-units-onida**

- **jar** is the command to execute the mapping and reducing
- **/home/acadgild/workspace/TvSales.jar** location on file system of the jar exported before
- **television.txt** is the file location on HDFS containing the records
- **/total-units-onida** is the destination directory on HDFS after reducing the records.

To check the results obtained on HDFS we can type

**hadoop fs –cat /total-units-onida/part-r-00000**



```
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=464
                CPU time spent (ms)=2590
                Physical memory (bytes) snapshot=267829248
                Virtual memory (bytes) snapshot=4113756160
                Total committed heap usage (bytes)=137498624
        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=733
        File Output Format Counters
                Bytes Written=16
[acadgild@localhost ~]$ hadoop fs -cat /total-units-onida/part-r-00000
18/07/26 08:51:46 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Uttar Pradesh   3
[acadgild@localhost ~]$ █
```

We only get one state, because the other records are invalid.

The complete code is in **total-units-onida-program.txt** and the results are in **total-units-onida.txt** obtained from HDFS using **hadoop fs –copyToLocal** command