Task-1 Write a Map Reduce program to filter out the invalid records. Map only job will fit for this context.

Solution-

MAPPER LOGIC

```
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.output.FileOutputFormat;

import java.io.IOException;
import java.util.logging.Logger;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.TextInputFormat;

public class TvSales {

public static class TvSalesMapper extends Mapper<LongWritable, Text, LongWritable, Text>{

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);
}
}

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;
}
```

MAIN CLASS

```
 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.output.FileOutputFormat;

 import java.io.IOException;
 import java.util.logging.Logger;

 import org.apache.hadoop.conf.Configuration;
 import org.apache.hadoop.fs.Path;
 import org.apache.hadoop.io.*;
 import org.apache.hadoop.mapred.TextInputFormat;

public static void main(String[] args) throws Exception{

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Tv Sales Invalid Records");
job.setJarByClass(TvSales.class);

job.setMapOutputKeyClass(LongWritable.class);
job.setMapOutputValueClass(Text.class);

job.setMapperClass(TvSalesMapper.class);

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

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

Task2 Write a Map Reduce program to calculate the total units sold for each Company.

Solution:

```
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.output.FileOutputFormat;

import java.io.IOException;
import java.util.logging.Logger;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.TextInputFormat;
public class TvSales {
public static class TvSalesMapper extends Mapper<LongWritable, Text, Text, IntWritable>{
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 );
}

}
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;
}
}
```

Reducer Class

```
 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.output.FileOutputFormat;

 import java.io.IOException;
 import java.util.logging.Logger;

 import org.apache.hadoop.conf.Configuration;
 import org.apache.hadoop.fs.Path;
 import org.apache.hadoop.io.*;
 import org.apache.hadoop.mapred.TextInputFormat;
public static class TvSalesReducer extends Reducer<Text, IntWritable, Text, IntWritable>{

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);
}
}
```

Main Class

```
 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.output.FileOutputFormat;

 import java.io.IOException;
 import java.util.logging.Logger;

 import org.apache.hadoop.conf.Configuration;
 import org.apache.hadoop.fs.Path;
 import org.apache.hadoop.io.*;
 import org.apache.hadoop.mapred.TextInputFormat;

public static void main(String[] args) throws Exception{

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Tv Sales Invalid Records");
job.setJarByClass(TvSales.class);

job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(IntWritable.class);

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

job.setMapperClass(TvSalesMapper.class);
job.setCombinerClass(TvSalesReducer.class);
job.setReducerClass(TvSalesReducer.class);

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

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

Task3 Write a Map Reduce program to calculate the total units sold in each state for Onida Company.

Solution:

Mapper logic

```
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.output.FileOutputFormat;

import java.io.IOException;
import java.util.logging.Logger;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.TextInputFormat;

public class TvSales {

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

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 );
}

}

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;
}
}
```

Reducer logic

```
 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.output.FileOutputFormat;

 import java.io.IOException;
 import java.util.logging.Logger;

 import org.apache.hadoop.conf.Configuration;
 import org.apache.hadoop.fs.Path;
 import org.apache.hadoop.io.*;
 import org.apache.hadoop.mapred.TextInputFormat;
public static class TvSalesReducer extends Reducer<Text, IntWritable, Text, IntWritable>{

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);
}
}
```

Main Class

```
 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.output.FileOutputFormat;

 import java.io.IOException;
 import java.util.logging.Logger;

 import org.apache.hadoop.conf.Configuration;
 import org.apache.hadoop.fs.Path;
 import org.apache.hadoop.io.*;
 import org.apache.hadoop.mapred.TextInputFormat;
public static void main(String[] args) throws Exception{

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Tv Sales Invalid REcords");
job.setJarByClass(TvSales.class);

job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(IntWritable.class);

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

job.setMapperClass(TvSalesMapper.class);
job.setCombinerClass(TvSalesReducer.class);
job.setReducerClass(TvSalesReducer.class);

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

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