Assignment 4

Task 1:

```
package television;


import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
importorg.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
importorg.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


public class FilterInvalidRecords {


  public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();
    Job job = new Job(conf, "FilterInvalidRecords");
    job.setJarByClass(FilterInvalidRecords.class);
    job.setMapperClass(FilterInvalidRecordsMapper.class);
    job.setOutputKeyClass(NullWritable.class);
    job.setOutputValueClass(Text.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
```

```
   public static class FilterInvalidRecordsMapper extends

         Mapper<Object, Text, NullWritable, Text> {

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

      String[] parts = value.toString().split("[|]");

      Text company = new Text(parts[0]);

      Text product = new Text(parts[1]);

      if (parts.length == 6 && !(company.toString().matches("NA") ||
product.toString().matches("NA"))) {

      context.write(NullWritable.get(), value);

      }

   }}

}
```

Task 2:

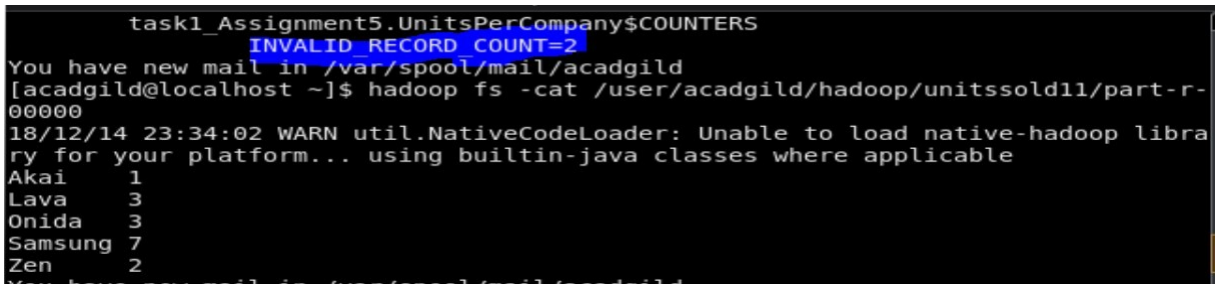
```
package television;

import java.io.IOException;
import java.util.HashSet;
import java.util.Set;

import org.apache.hadoop.conf.Configuration;
importorg.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
importorg.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.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class UnitsPerCompany

   { private enum COUNTERS {
         INVALID_RECORD_COUNT
   }

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

      Job job = new Job(conf, "Units sold per company");
      job.setJarByClass(UnitsPerCompany.class);
```

```
        job.setMapperClass(UnitsPerCompanyMapper.class);
        job.setReducerClass(UnitsPerCompanyReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
        org.apache.hadoop.mapreduce.Counters counters = job.getCounters();
        System.out.println("No. of Invalid Records :"
                + counters.findCounter(COUNTERS.INVALID_RECORD_COUNT)
                        .getValue());
    }

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

        public void
                reduce( Text
                company,
                Iterable<IntWritable> counts,
                Reducer<Text,IntWritable, Text, IntWritable>.Context context)
                throws IOException, InterruptedException {

            if(!(company.toString()).matches("NA"))
            { int i=0;
            for (IntWritable count : counts)
                { i+=count.get();
            }
            IntWritable size = new IntWritable(i);
            context.write(company, size);
        }
        }
    }

    public static class UnitsPerCompanyMapper extends
            Mapper<Object, Text,Text, IntWritable> {

        public void map(Object key, Text value,
                Mapper<Object, Text, Text,IntWritable>.Context context)
                throws IOException, InterruptedException {

            String[] parts = value.toString().split("[|]");
            Text company = new Text(parts[0]);
            Text product = new Text(parts[1]);
            IntWritable one = new IntWritable(1);
            if (parts.length == 6 && !(company.toString().matches("NA") ||
product.toString().matches("NA"))) {
                context.write(company,one);
            } else {
                // add counter for invalid records
                context.getCounter(COUNTERS.INVALID_RECORD_COUNT).increment(1L);
            }
```

```
    }
  }


}
```

Output:

The output shows the units sold per company in different states, which sums to 16 as 2 records are invalid.



Task3:

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

Code written to execute the task:

```
package television;


import java.io.IOException;


import org.apache.hadoop.conf.Configuration;

importorg.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

importorg.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.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

```
public class Onidaperstate {


   private enum COUNTERS
      { INVALID_RECORD_COUN
      T
   }

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

      Job job = new Job(conf, "Units sold per company");
      job.setJarByClass(Onidaperstate.class);
      job.setMapperClass(OnidaperstateMapper.class);
      job.setReducerClass(OnidaperstateReducer.class);
      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(IntWritable.class);
      FileInputFormat.addInputPath(job, new Path(args[0]));
      FileOutputFormat.setOutputPath(job, new Path(args[1]));
      System.exit(job.waitForCompletion(true) ? 0 : 1);
      org.apache.hadoop.mapreduce.Counters counters = job.getCounters();
      System.out.println("No. of Invalid Records :"
            + counters.findCounter(COUNTERS.INVALID_RECORD_COUNT)
                  .getValue());
   }

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

      public void reduce(
            Text state,
```

```
            Iterable<IntWritable> units,
            Reducer<Text,IntWritable, Text, IntWritable>.Context context)
            throws IOException, InterruptedException {


        int i=0;
        for (IntWritable unit : units) {
            i+=unit.get();
        }
        IntWritable size = new IntWritable(i);
        context.write(state, size);

      }


}
   public static class OnidaperstateMapper extends
   Mapper<Object, Text,Text, IntWritable> {

public void map(Object key, Text value,
      Mapper<Object, Text, Text,IntWritable>.Context context)
      throws IOException, InterruptedException {

   String[] parts = value.toString().split("[|]");
   Text company = new Text(parts[0]);
   Text product = new Text(parts[1]);
   Text state = new Text(parts[3]);
   IntWritable one = new IntWritable(1);
   if (parts.length == 6
      && !(company.toString().matches("NA") || product.toString().matches("NA"))
      && company.toString().matches("Onida")) {
```

```
        context.write(state,one);
    } else {
        // add counter for invalid records
        context.getCounter(COUNTERS.INVALID_RECORD_COUNT).increment(1L);
    }


}
}


}
```

Output: