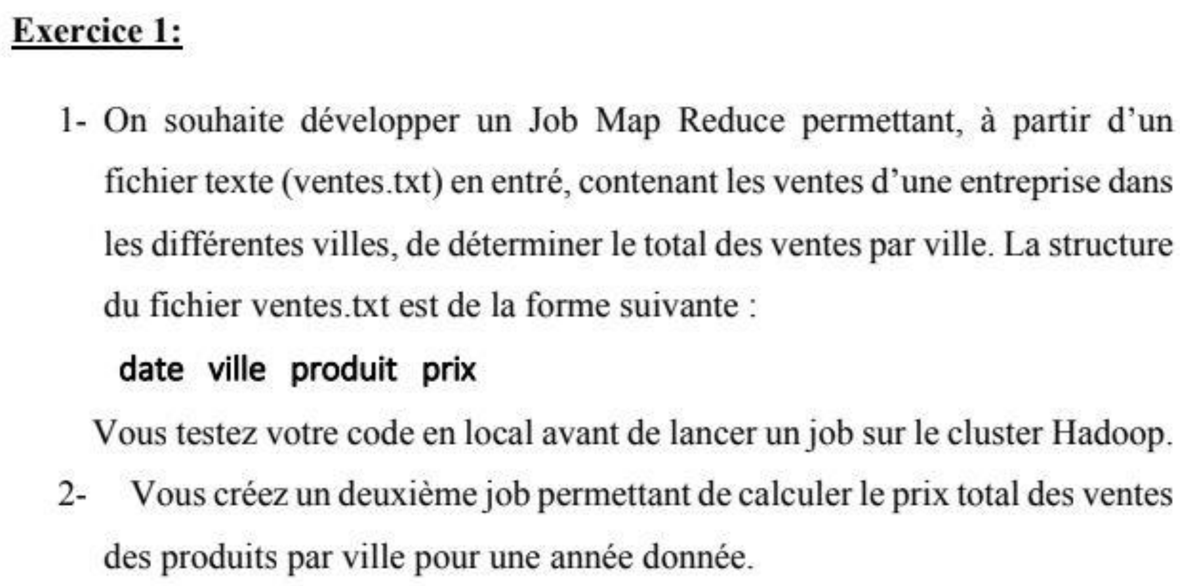
ENSET-M	II-BDCC 2 S4 2022
Cours Big Data: Fondements et Architectures de stockage	EL AAMIRI Essadeq

TPs- MapReduce

Exercice 1 : Total des ventes :



Exercice 1:

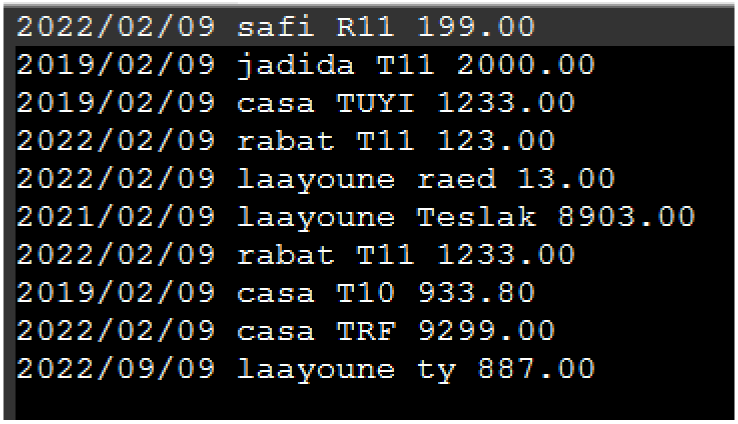
1- On souhaite développer un Job Map Reduce permettant, à partir d'un fichier texte (ventes.txt) en entré, contenant les ventes d'une entreprise dans les différentes villes, de déterminer le total des ventes par ville. La structure du fichier ventes.txt est de la forme suivante :

date ville produit prix

Vous testez votre code en local avant de lancer un job sur le cluster Hadoop.

2- Vous créez un deuxième job permettant de calculer le prix total des ventes des produits par ville pour une année donnée.

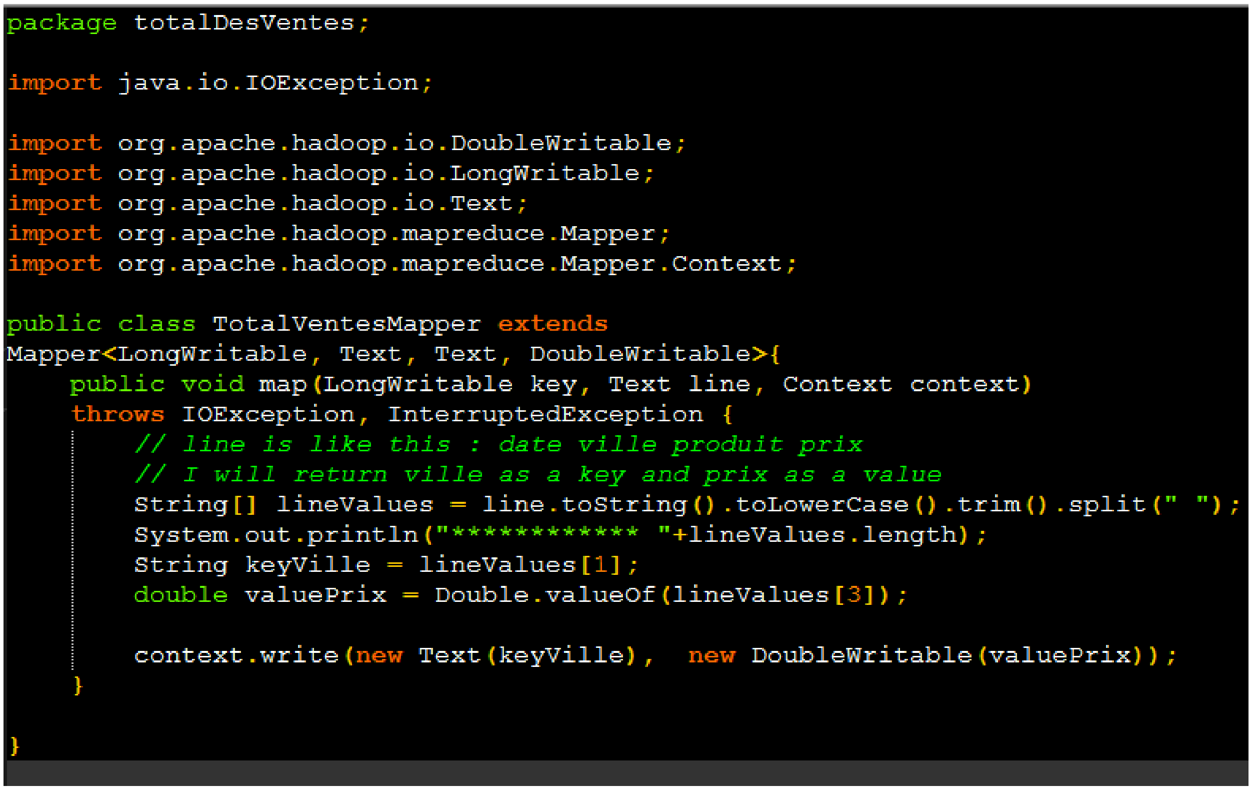
Les données :



```
2022/02/09 safi R11 199.00
2019/02/09 jadida T11 2000.00
2019/02/09 casa TUYI 1233.00
2022/02/09 rabat T11 123.00
2022/02/09 laayoune raed 13.00
2021/02/09 laayoune Teslak 8903.00
2022/02/09 rabat T11 1233.00
2019/02/09 casa T10 933.80
2022/02/09 casa TRF 9299.00
2022/09/09 laayoune ty 887.00
```

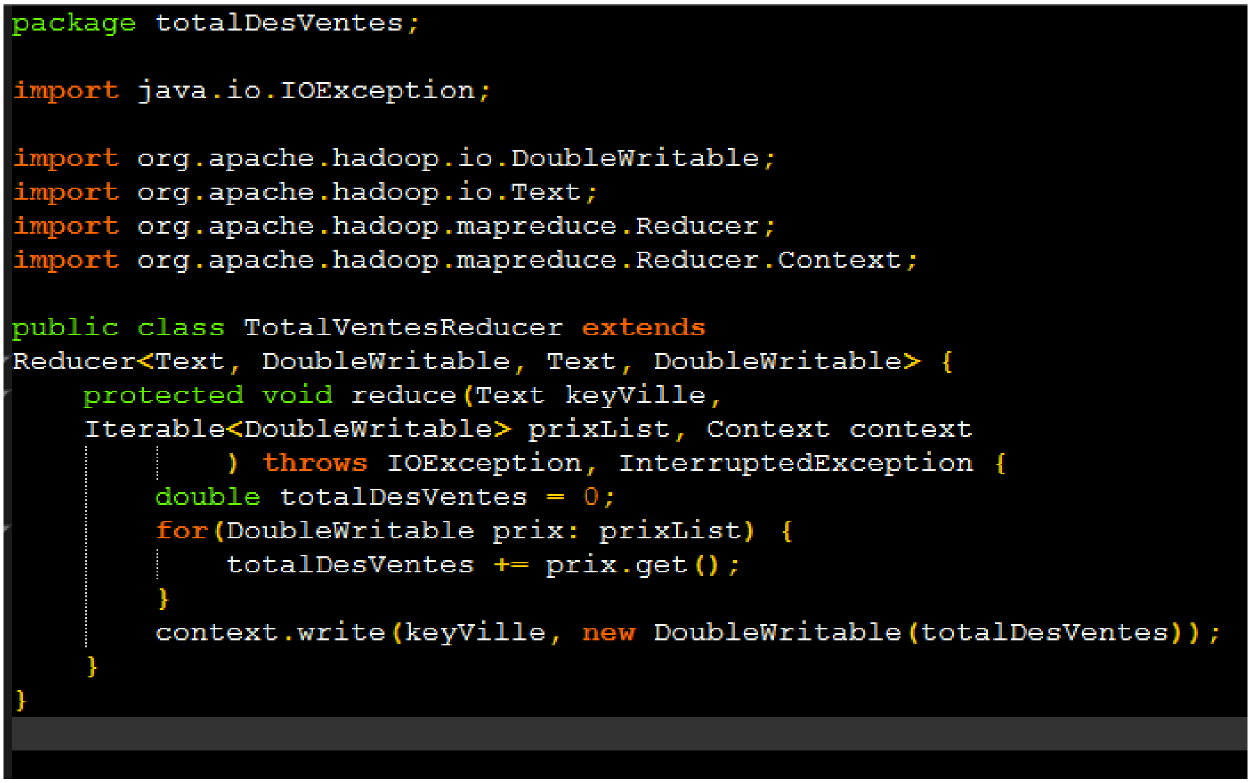
Question 1 :

Mapper :



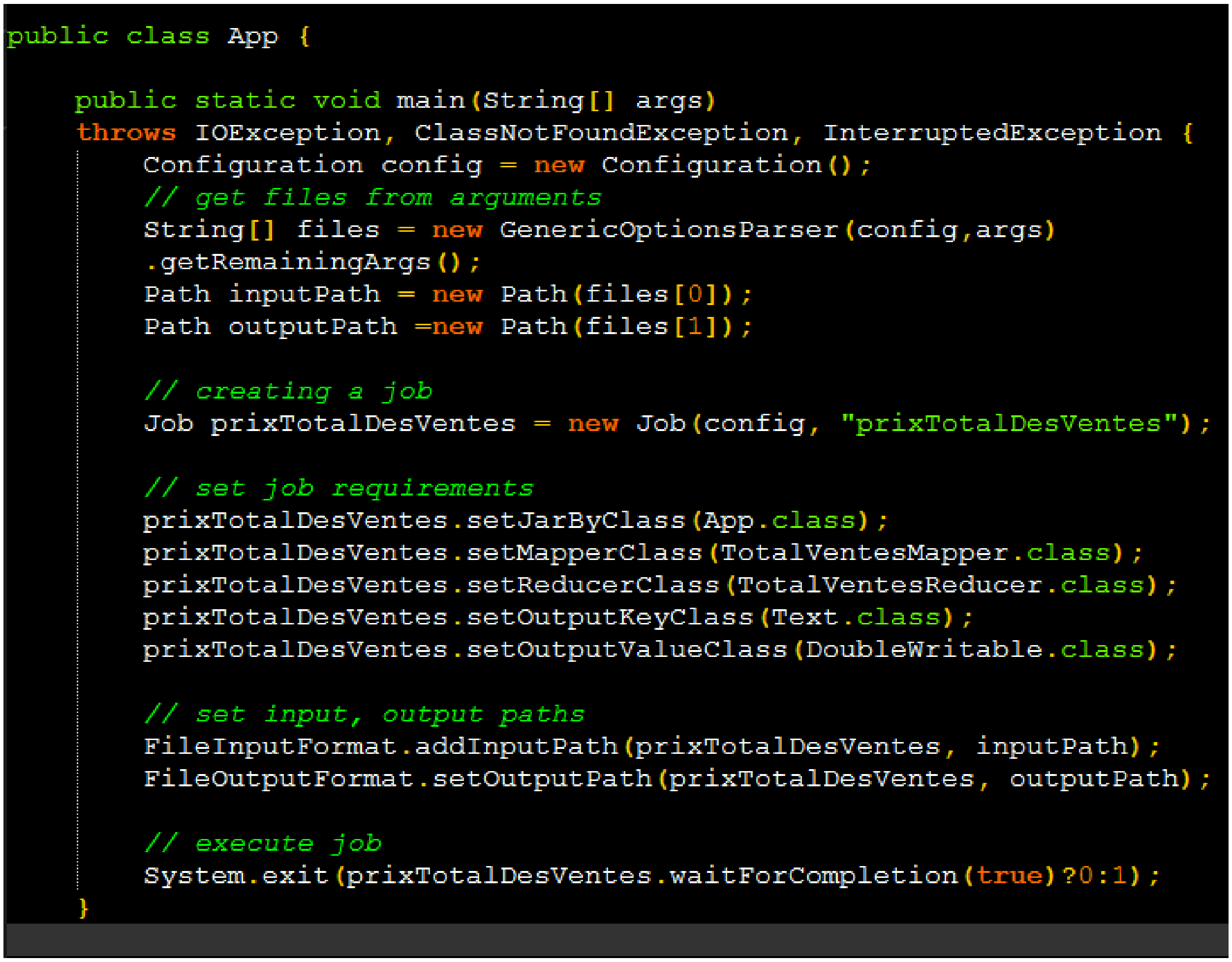
```java
package totalDesVentes;

import java.io.IOException;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Mapper.Context;

public class TotalVentesMapper extends
Mapper<LongWritable, Text, Text, DoubleWritable>{
    public void map(LongWritable key, Text line, Context context)
    throws IOException, InterruptedException {
        // line is like this : date ville produit prix
        // I will return ville as a key and prix as a value
        String[] lineValues = line.toString().toLowerCase().trim().split(" ");
        System.out.println("************ "+lineValues.length);
        String keyVille = lineValues[1];
        double valuePrix = Double.valueOf(lineValues[3]);

        context.write(new Text(keyVille),  new DoubleWritable(valuePrix));
    }

}
```

Reducer :



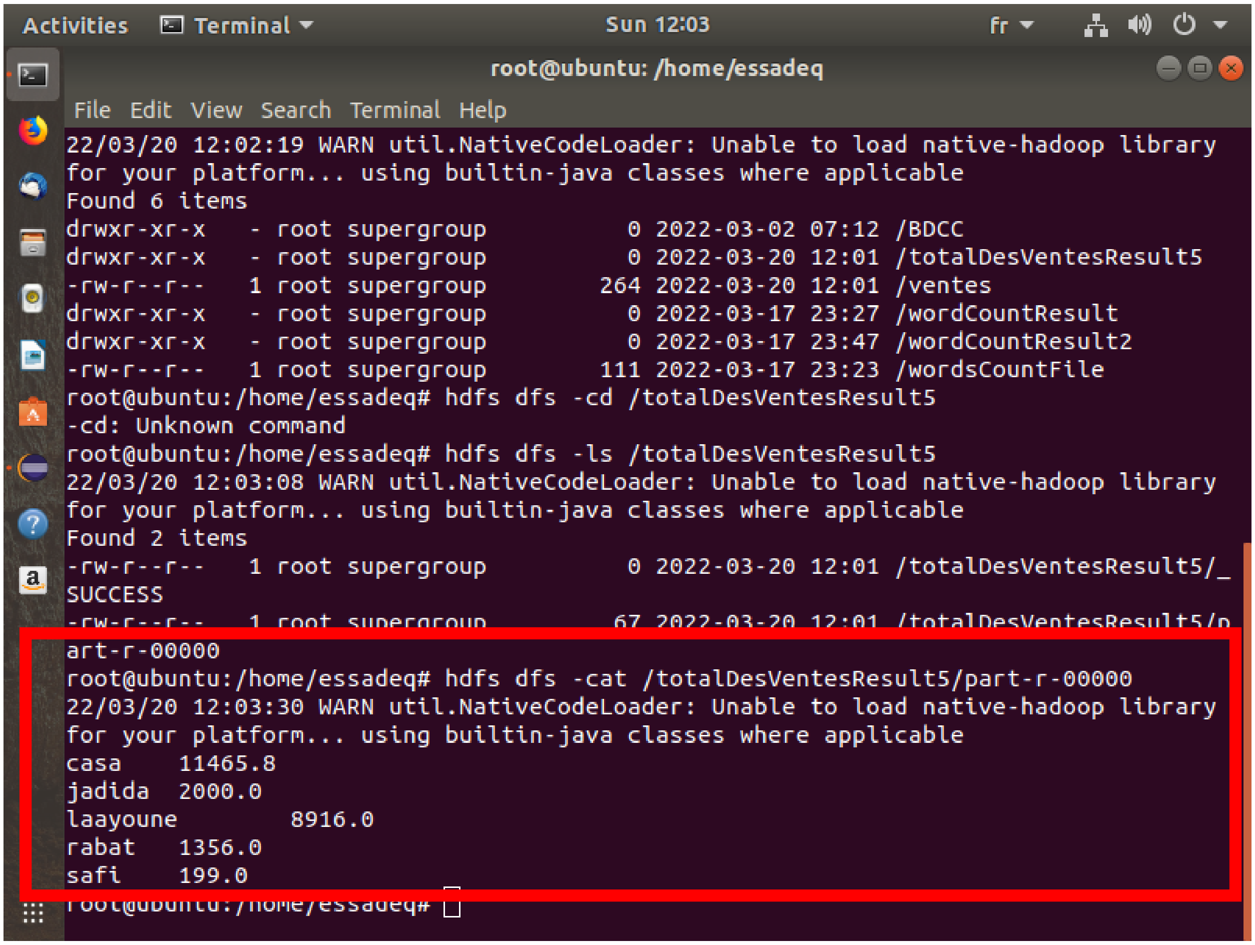
```java
package totalDesVentes;

import java.io.IOException;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Reducer.Context;

public class TotalVentesReducer extends
Reducer<Text, DoubleWritable, Text, DoubleWritable> {
    protected void reduce(Text keyVille,
    Iterable<DoubleWritable> prixList, Context context
            ) throws IOException, InterruptedException {
        double totalDesVentes = 0;
        for(DoubleWritable prix: prixList) {
            totalDesVentes += prix.get();
        }
        context.write(keyVille, new DoubleWritable(totalDesVentes));
    }
}
```

Application :



```java
public class App {

    public static void main(String[] args)
    throws IOException, ClassNotFoundException, InterruptedException {
        Configuration config = new Configuration();
        // get files from arguments
        String[] files = new GenericOptionsParser(config,args)
        .getRemainingArgs();
        Path inputPath = new Path(files[0]);
        Path outputPath =new Path(files[1]);

        // creating a job
        Job prixTotalDesVentes = new Job(config, "prixTotalDesVentes");

        // set job requirements
        prixTotalDesVentes.setJarByClass(App.class);
        prixTotalDesVentes.setMapperClass(TotalVentesMapper.class);
        prixTotalDesVentes.setReducerClass(TotalVentesReducer.class);
        prixTotalDesVentes.setOutputKeyClass(Text.class);
        prixTotalDesVentes.setOutputValueClass(DoubleWritable.class);

        // set input, output paths
        FileInputFormat.addInputPath(prixTotalDesVentes, inputPath);
        FileOutputFormat.setOutputPath(prixTotalDesVentes, outputPath);

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

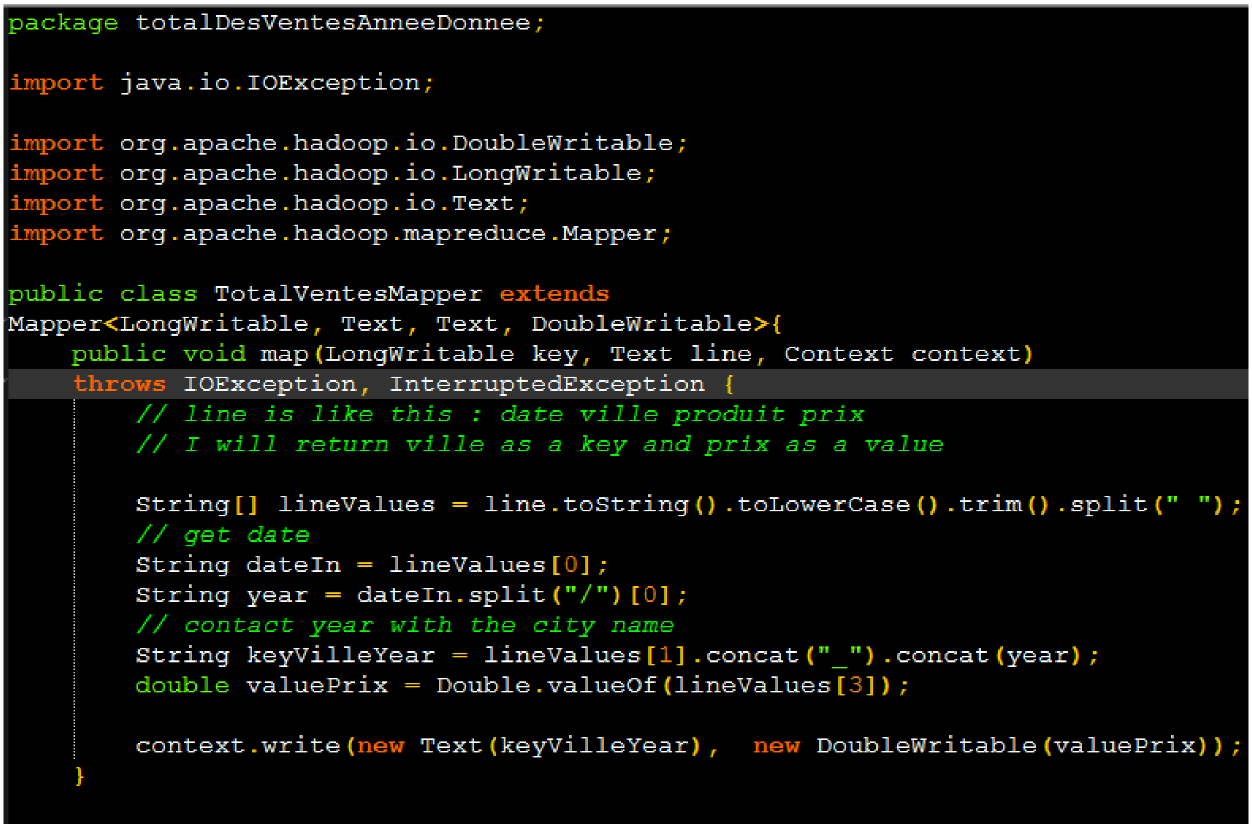
Resultat :


Activities
Terminal
Sun 12:03
fr
root@ubuntu: /home/essadeq
File Edit View Search Terminal Help

```
22/03/20 12:02:19 WARN util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
Found 6 items
drwxr-xr-x   - root supergroup          0 2022-03-02 07:12 /BDCC
drwxr-xr-x   - root supergroup          0 2022-03-20 12:01 /totalDesVentesResult5
-rw-r--r--   1 root supergroup        264 2022-03-20 12:01 /ventes
drwxr-xr-x   - root supergroup          0 2022-03-17 23:27 /wordCountResult
drwxr-xr-x   - root supergroup          0 2022-03-17 23:47 /wordCountResult2
-rw-r--r--   1 root supergroup        111 2022-03-17 23:23 /wordsCountFile
root@ubuntu:/home/essadeq# hdfs dfs -cd /totalDesVentesResult5
-cd: Unknown command
root@ubuntu:/home/essadeq# hdfs dfs -ls /totalDesVentesResult5
22/03/20 12:03:08 WARN util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--   1 root supergroup          0 2022-03-20 12:01 /totalDesVentesResult5/_
SUCCESS
-rw-r--r--   1 root supergroup         67 2022-03-20 12:01 /totalDesVentesResult5/p
art-r-00000
root@ubuntu:/home/essadeq# hdfs dfs -cat /totalDesVentesResult5/part-r-00000
22/03/20 12:03:30 WARN util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
casa    11465.8
jadida  2000.0
laayoune        8916.0
rabat   1356.0
safi    199.0
root@ubuntu:/home/essadeq# 
```

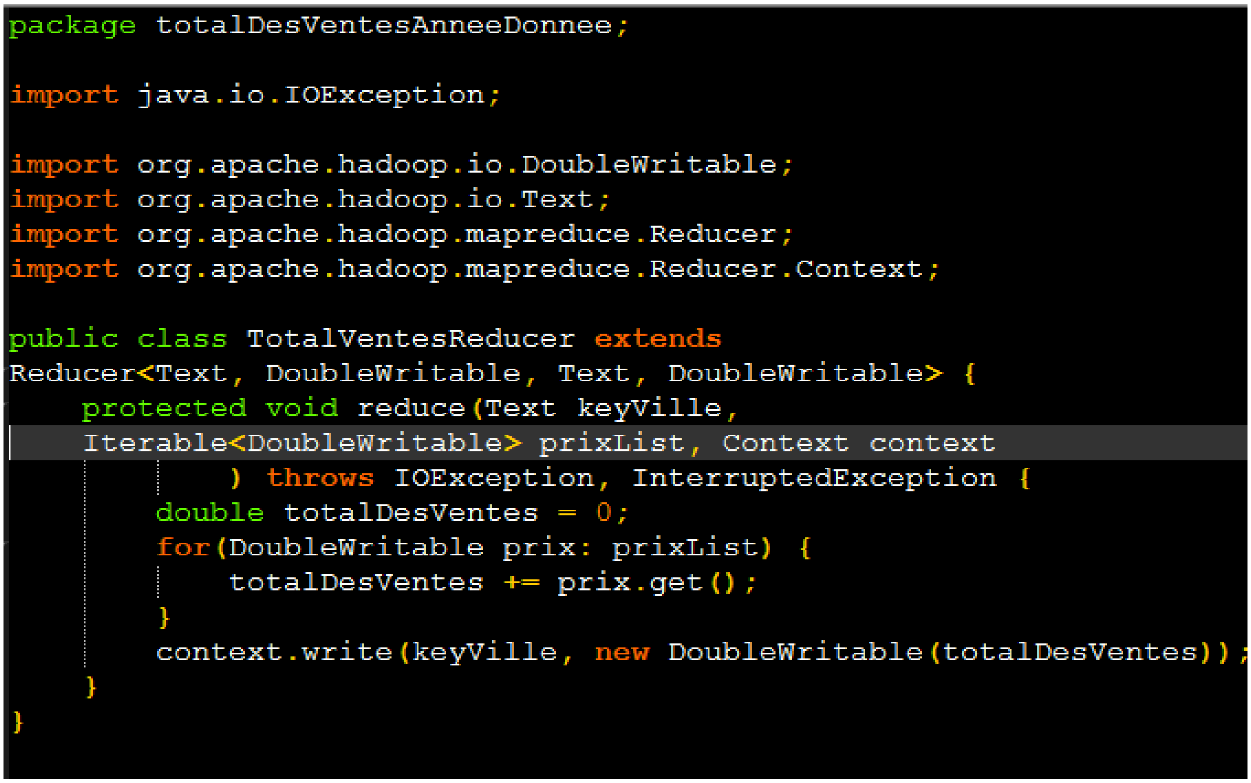
Question 2:

Mapper



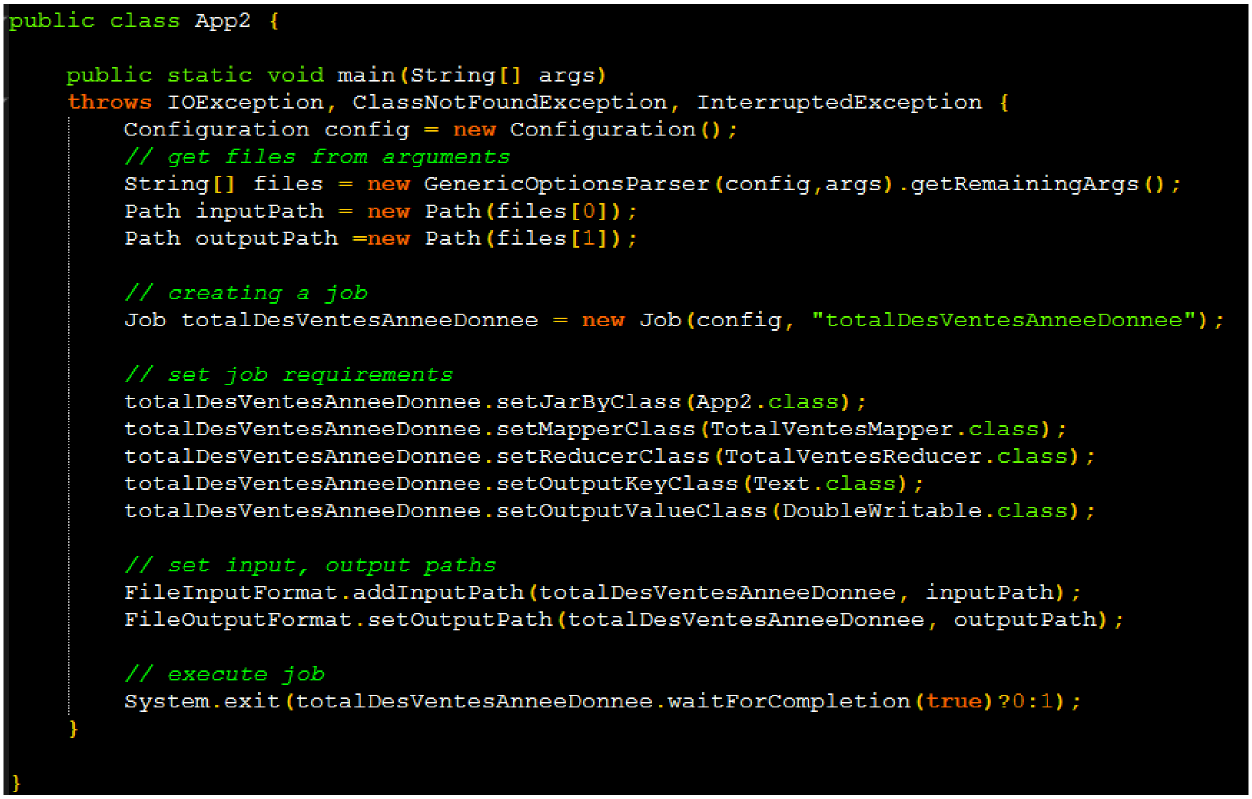
```
package totalDesVentesAnneeDonnee;

import java.io.IOException;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class TotalVentesMapper extends
Mapper<LongWritable, Text, Text, DoubleWritable>{
    public void map(LongWritable key, Text line, Context context)
    throws IOException, InterruptedException {
        // line is like this : date ville produit prix
        // I will return ville as a key and prix as a value

        String[] lineValues = line.toString().toLowerCase().trim().split(" ");
        // get date
        String dateIn = lineValues[0];
        String year = dateIn.split("/")[0];
        // contact year with the city name
        String keyVilleYear = lineValues[1].concat("_").concat(year);
        double valuePrix = Double.valueOf(lineValues[3]);

        context.write(new Text(keyVilleYear),  new DoubleWritable(valuePrix));
    }
```

Reducer



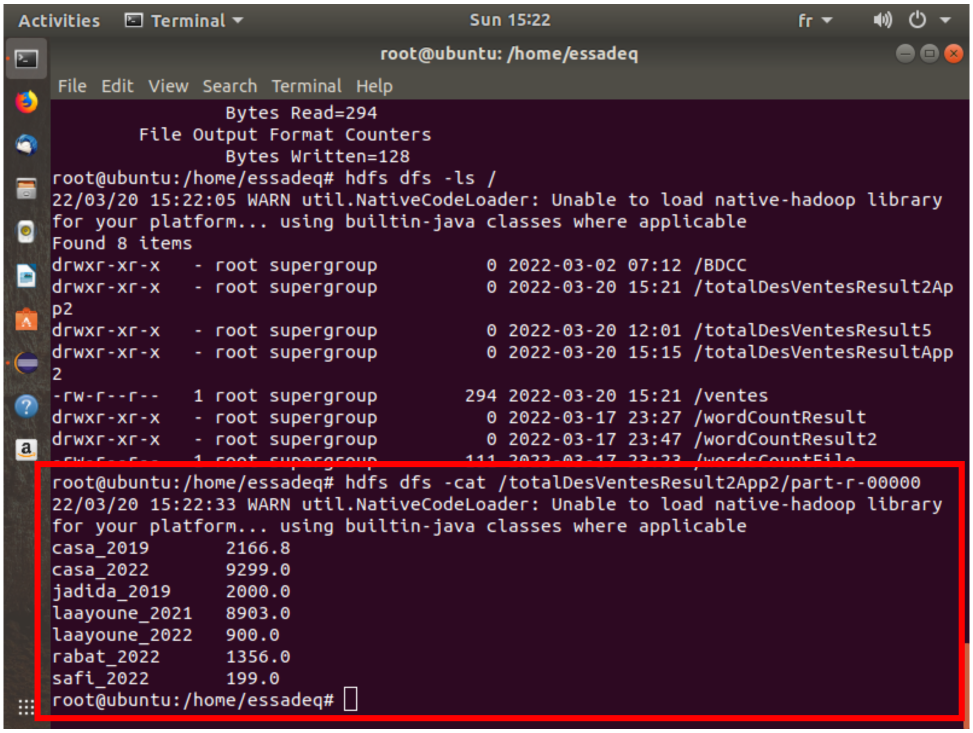
```java
package totalDesVentesAnneeDonnee;

import java.io.IOException;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Reducer.Context;

public class TotalVentesReducer extends
Reducer<Text, DoubleWritable, Text, DoubleWritable> {
    protected void reduce(Text keyVille,
    Iterable<DoubleWritable> prixList, Context context
            ) throws IOException, InterruptedException {
        double totalDesVentes = 0;
        for(DoubleWritable prix: prixList) {
            totalDesVentes += prix.get();
        }
        context.write(keyVille, new DoubleWritable(totalDesVentes));
    }
}
```

Application Job



```
public class App2 {

    public static void main(String[] args)
    throws IOException, ClassNotFoundException, InterruptedException {
        Configuration config = new Configuration();
        // get files from arguments
        String[] files = new GenericOptionsParser(config,args).getRemainingArgs();
        Path inputPath = new Path(files[0]);
        Path outputPath =new Path(files[1]);

        // creating a job
        Job totalDesVentesAnneeDonnee = new Job(config, "totalDesVentesAnneeDonnee");

        // set job requirements
        totalDesVentesAnneeDonnee.setJarByClass(App2.class);
        totalDesVentesAnneeDonnee.setMapperClass(TotalVentesMapper.class);
        totalDesVentesAnneeDonnee.setReducerClass(TotalVentesReducer.class);
        totalDesVentesAnneeDonnee.setOutputKeyClass(Text.class);
        totalDesVentesAnneeDonnee.setOutputValueClass(DoubleWritable.class);

        // set input, output paths
        FileInputFormat.addInputPath(totalDesVentesAnneeDonnee, inputPath);
        FileOutputFormat.setOutputPath(totalDesVentesAnneeDonnee, outputPath);

        // execute job
        System.exit(totalDesVentesAnneeDonnee.waitForCompletion(true)?0:1);
    }

}
```

Résultat :

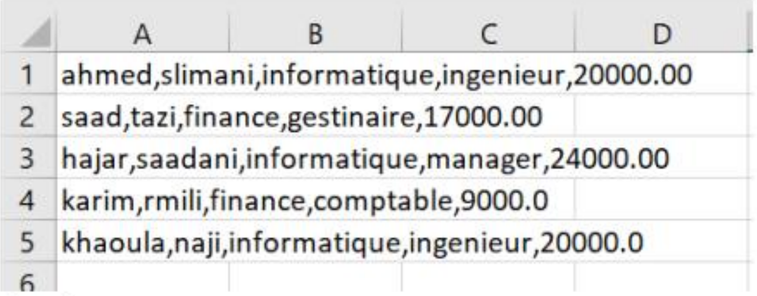

Activities
Terminal
Sun 15:22
fr
root@ubuntu: /home/essadeq
File Edit View Search Terminal Help

```
                Bytes Read=294
        File Output Format Counters
                Bytes Written=128
root@ubuntu:/home/essadeq# hdfs dfs -ls /
22/03/20 15:22:05 WARN util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
Found 8 items
drwxr-xr-x   - root supergroup          0 2022-03-02 07:12 /BDCC
drwxr-xr-x   - root supergroup          0 2022-03-20 15:21 /totalDesVentesResult2Ap
p2
drwxr-xr-x   - root supergroup          0 2022-03-20 12:01 /totalDesVentesResult5
drwxr-xr-x   - root supergroup          0 2022-03-20 15:15 /totalDesVentesResultApp
2
-rw-r--r--   1 root supergroup        294 2022-03-20 15:21 /ventes
drwxr-xr-x   - root supergroup          0 2022-03-17 23:27 /wordCountResult
drwxr-xr-x   - root supergroup          0 2022-03-17 23:47 /wordCountResult2
-rw-r--r--   1 root supergroup        111 2022-03-17 23:23 /wordsCountFile
root@ubuntu:/home/essadeq# hdfs dfs -cat /totalDesVentesResult2App2/part-r-00000
22/03/20 15:22:33 WARN util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
casa_2019       2166.8
casa_2022       9299.0
jadida_2019     2000.0
laayoune_2021   8903.0
laayoune_2022   900.0
rabat_2022      1356.0
safi_2022       199.0
root@ubuntu:/home/essadeq# 
```

Exercice 2 : Analyse de données Employés

Exercice 1

- **Analyse de données Employés :**
  - **Exemple de données d'entrée**



	A	B	C	D
1	ahmed,slimani,informatique,ingenieur,20000.00			
2	saad,tazi,finance,gestinaire,17000.00			
3	hajar,saadani,informatique,manager,24000.00			
4	karim,rmili,finance,comptable,9000.0			
5	khaoula,naji,informatique,ingenieur,20000.0			
6				

  - **Les noms de colonnes**

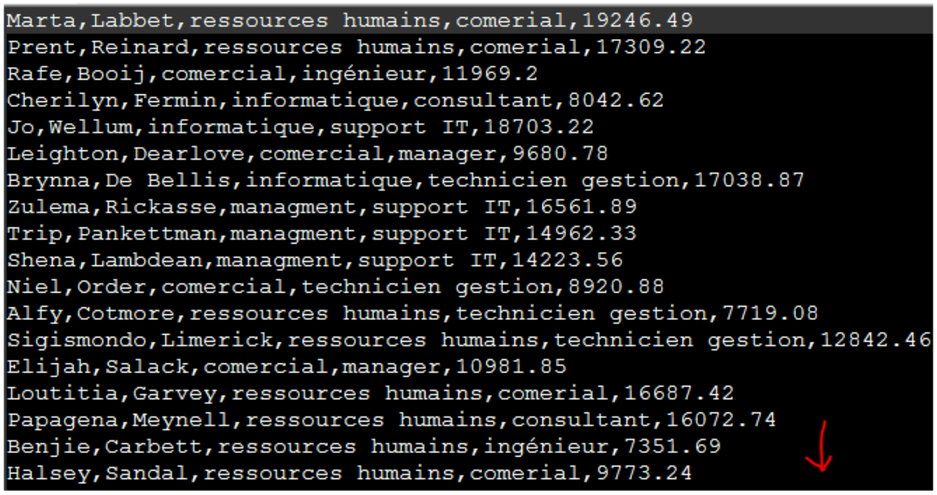
    firstName,lastName,department, jobTitle,salary

  - **Problème à résoudre**

    1. **Étant donné une liste d'employés avec leur département et leur salaire, trouvez le salaire maximum et minimum dans chaque département.**

    2. **Étant donné une liste d'employés avec leur département, trouvez le nombre d'employés dans chaque département.**

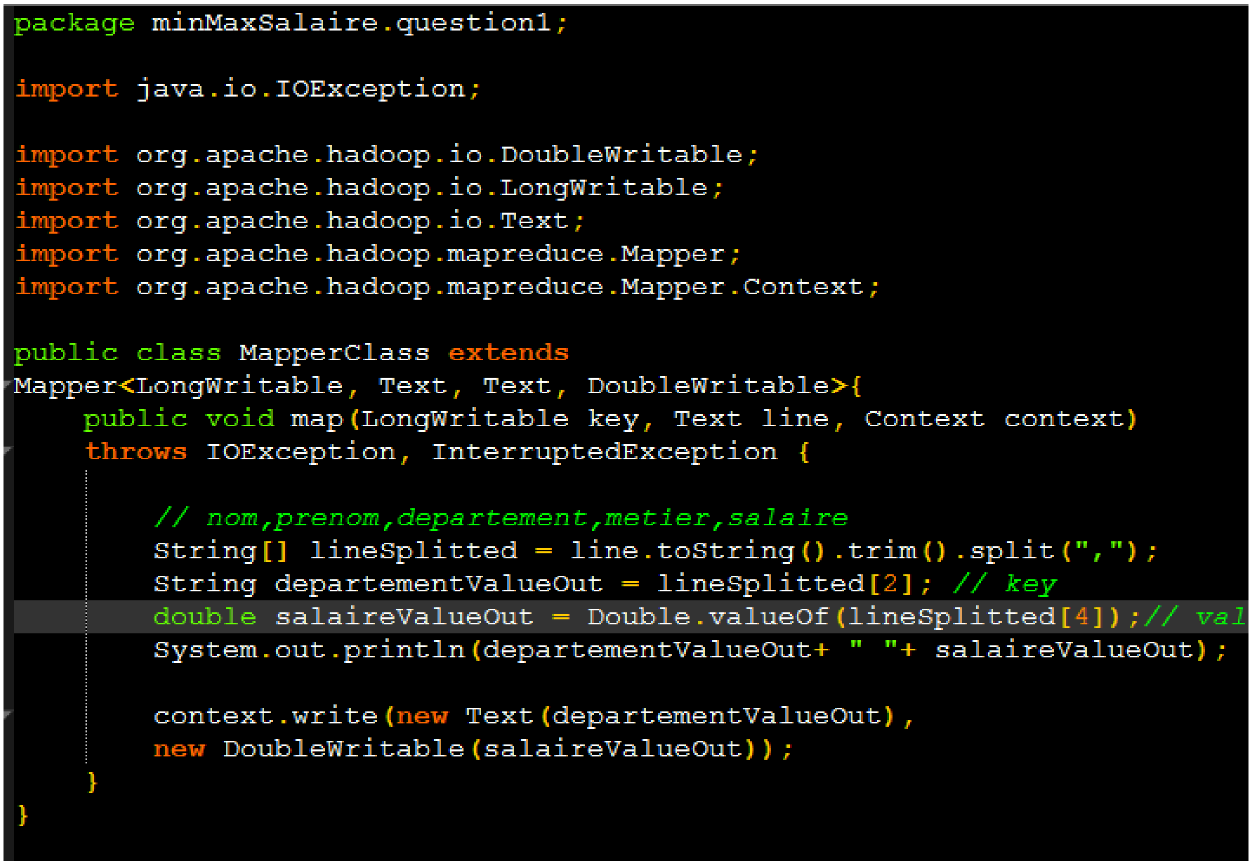
Données :



```
Marta,Labbet,ressources humains,comerial,19246.49
Prent,Reinard,ressources humains,comerial,17309.22
Rafe,Booij,comercial,ingénieur,11969.2
Cherilyn,Fermin,informatique,consultant,8042.62
Jo,Wellum,informatique,support IT,18703.22
Leighton,Dearlove,comercial,manager,9680.78
Brynna,De Bellis,informatique,technicien gestion,17038.87
Zulema,Rickasse,managment,support IT,16561.89
Trip,Pankettman,managment,support IT,14962.33
Shena,Lambdean,managment,support IT,14223.56
Niel,Order,comercial,technicien gestion,8920.88
Alfy,Cotmore,ressources humains,technicien gestion,7719.08
Sigismondo,Limerick,ressources humains,technicien gestion,12842.46
Elijah,Salack,comercial,manager,10981.85
Loutitia,Garvey,ressources humains,comerial,16687.42
Papagena,Meynell,ressources humains,consultant,16072.74
Benjie,Carbett,ressources humains,ingénieur,7351.69
Halsey,Sandal,ressources humains,comerial,9773.24
```

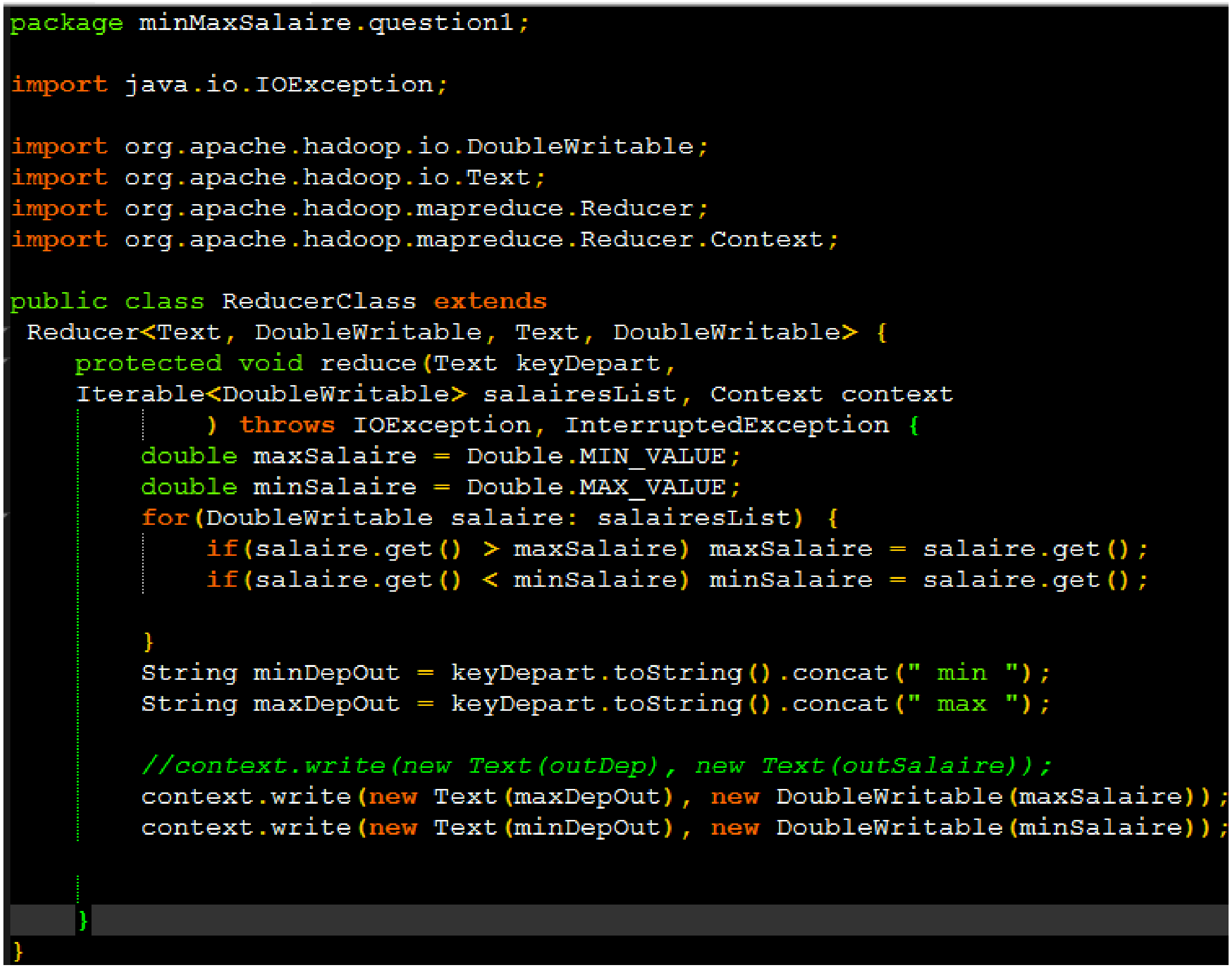
Question 1 :

Mapper



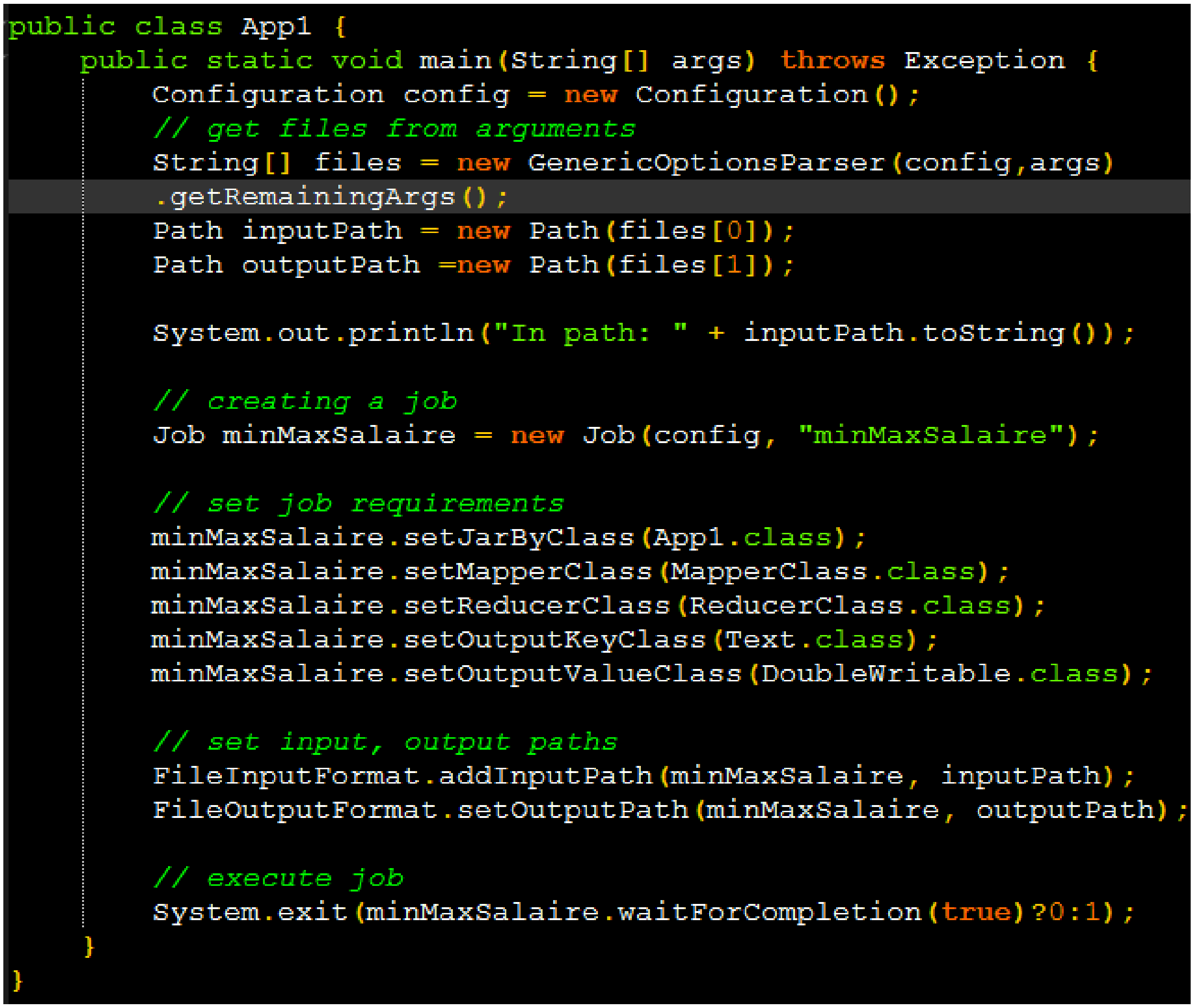
```java
package minMaxSalaire.question1;

import java.io.IOException;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Mapper.Context;

public class MapperClass extends
Mapper<LongWritable, Text, Text, DoubleWritable>{
    public void map(LongWritable key, Text line, Context context)
    throws IOException, InterruptedException {

        // nom,prenom,departement,metier,salaire
        String[] lineSplitted = line.toString().trim().split(",");
        String departementValueOut = lineSplitted[2]; // key
        double salaireValueOut = Double.valueOf(lineSplitted[4]);// val
        System.out.println(departementValueOut+ " "+ salaireValueOut);

        context.write(new Text(departementValueOut),
        new DoubleWritable(salaireValueOut));
    }
}
```

Reducer



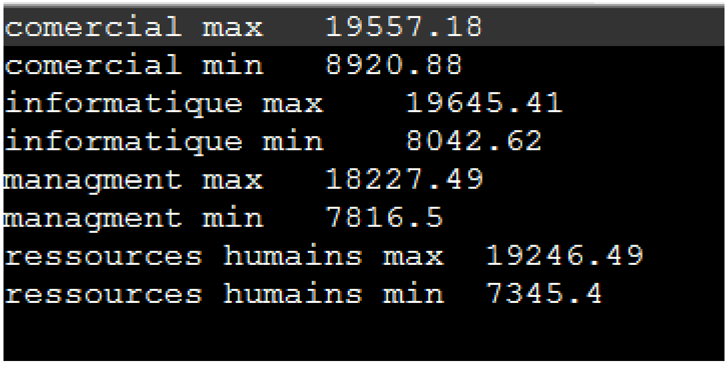
```
package minMaxSalaire.question1;

import java.io.IOException;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Reducer.Context;

public class ReducerClass extends
 Reducer<Text, DoubleWritable, Text, DoubleWritable> {
    protected void reduce(Text keyDepart,
    Iterable<DoubleWritable> salairesList, Context context
            ) throws IOException, InterruptedException {
        double maxSalaire = Double.MIN_VALUE;
        double minSalaire = Double.MAX_VALUE;
        for(DoubleWritable salaire: salairesList) {
            if(salaire.get() > maxSalaire) maxSalaire = salaire.get();
            if(salaire.get() < minSalaire) minSalaire = salaire.get();

        }
        String minDepOut = keyDepart.toString().concat(" min ");
        String maxDepOut = keyDepart.toString().concat(" max ");

        //context.write(new Text(outDep), new Text(outSalaire));
        context.write(new Text(maxDepOut), new DoubleWritable(maxSalaire));
        context.write(new Text(minDepOut), new DoubleWritable(minSalaire));


    }
}
```

Application Job



```
public class App1 {
    public static void main(String[] args) throws Exception {
        Configuration config = new Configuration();
        // get files from arguments
        String[] files = new GenericOptionsParser(config,args)
        .getRemainingArgs();
        Path inputPath = new Path(files[0]);
        Path outputPath =new Path(files[1]);

        System.out.println("In path: " + inputPath.toString());

        // creating a job
        Job minMaxSalaire = new Job(config, "minMaxSalaire");

        // set job requirements
        minMaxSalaire.setJarByClass(App1.class);
        minMaxSalaire.setMapperClass(MapperClass.class);
        minMaxSalaire.setReducerClass(ReducerClass.class);
        minMaxSalaire.setOutputKeyClass(Text.class);
        minMaxSalaire.setOutputValueClass(DoubleWritable.class);

        // set input, output paths
        FileInputFormat.addInputPath(minMaxSalaire, inputPath);
        FileOutputFormat.setOutputPath(minMaxSalaire, outputPath);

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

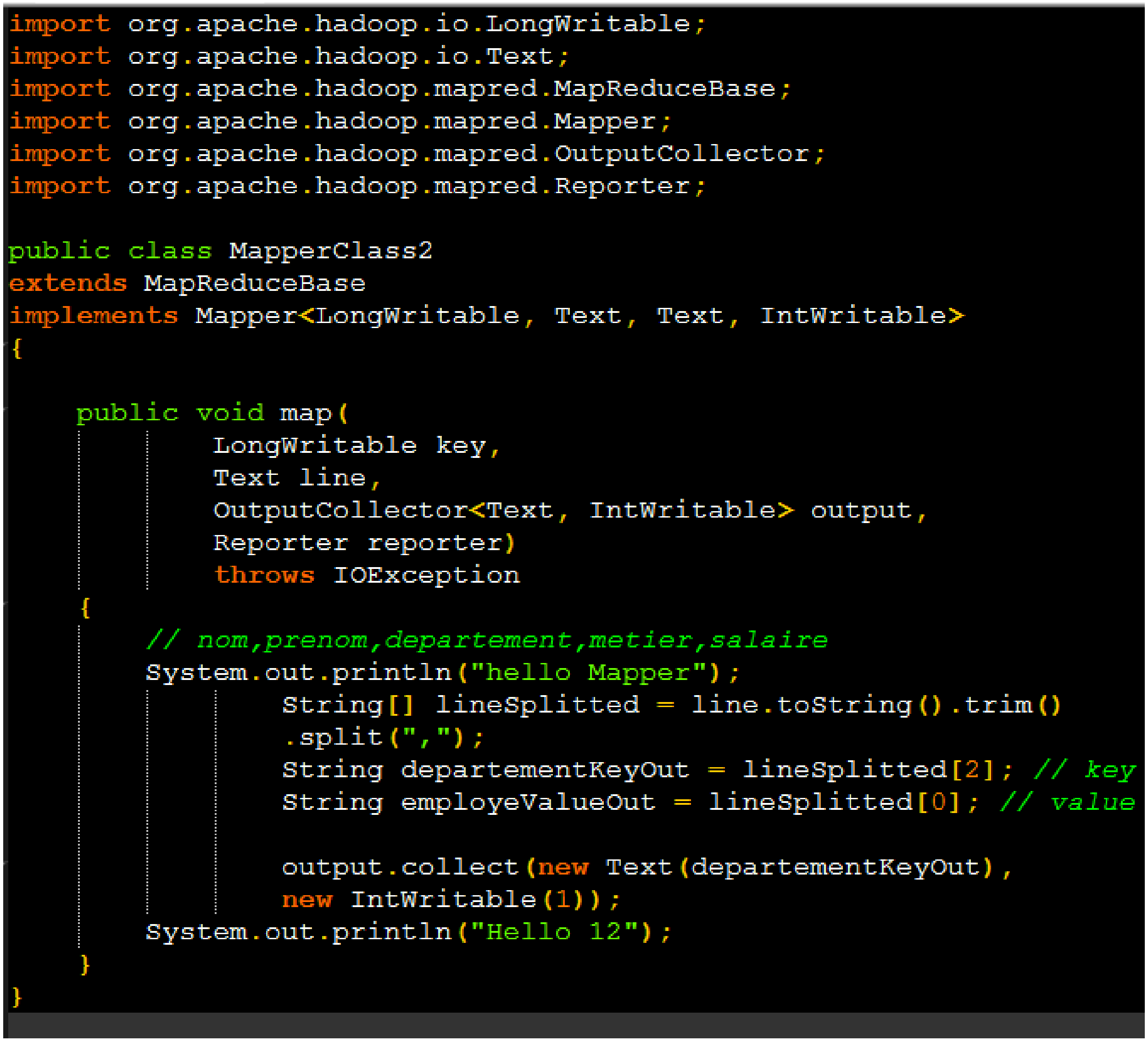
Résultat :



```
comercial max   19557.18
comercial min   8920.88
informatique max    19645.41
informatique min    8042.62
managment max   18227.49
managment min   7816.5
ressources humains max  19246.49
ressources humains min  7345.4
```

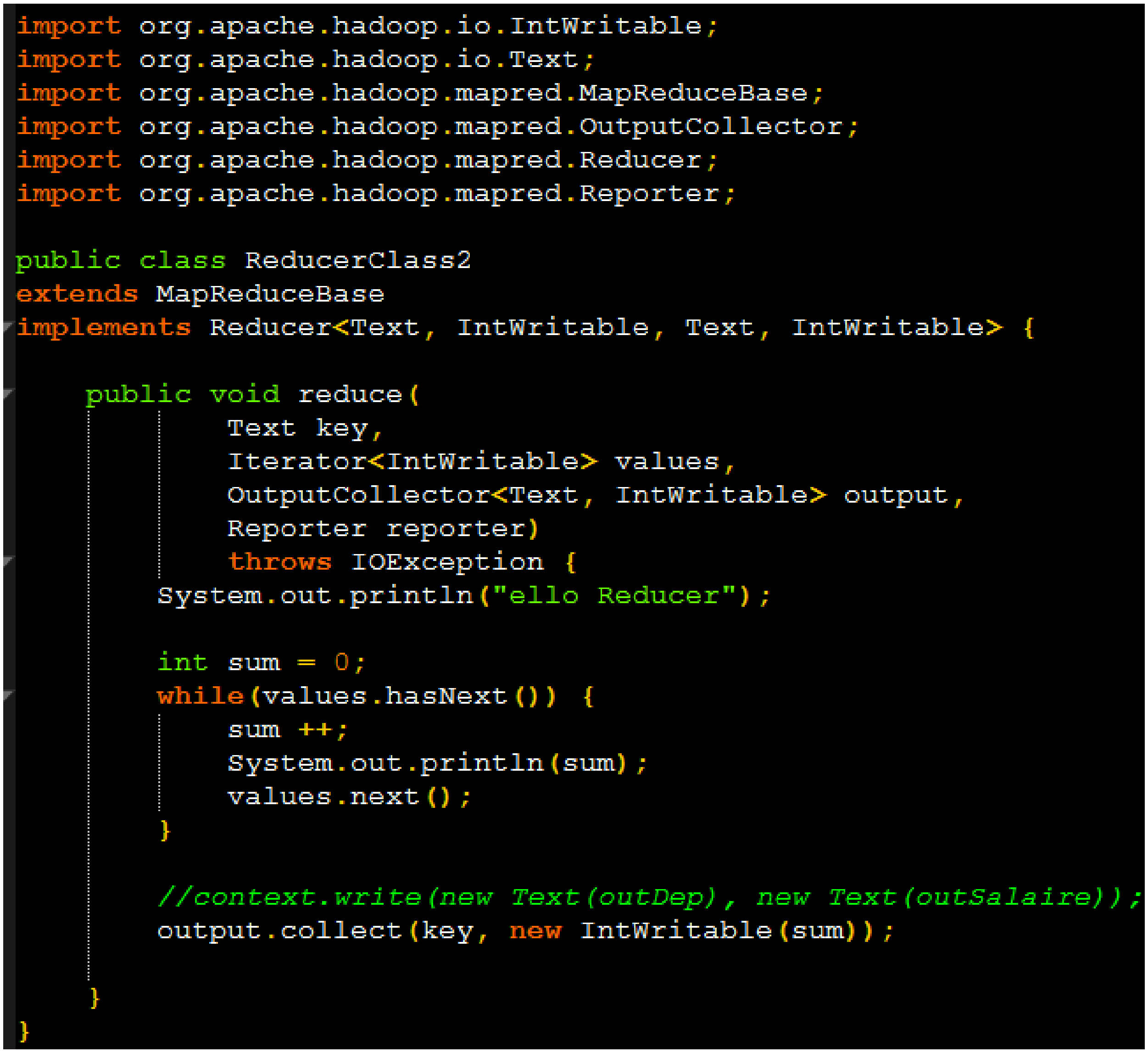
Question 2 :

Mapper



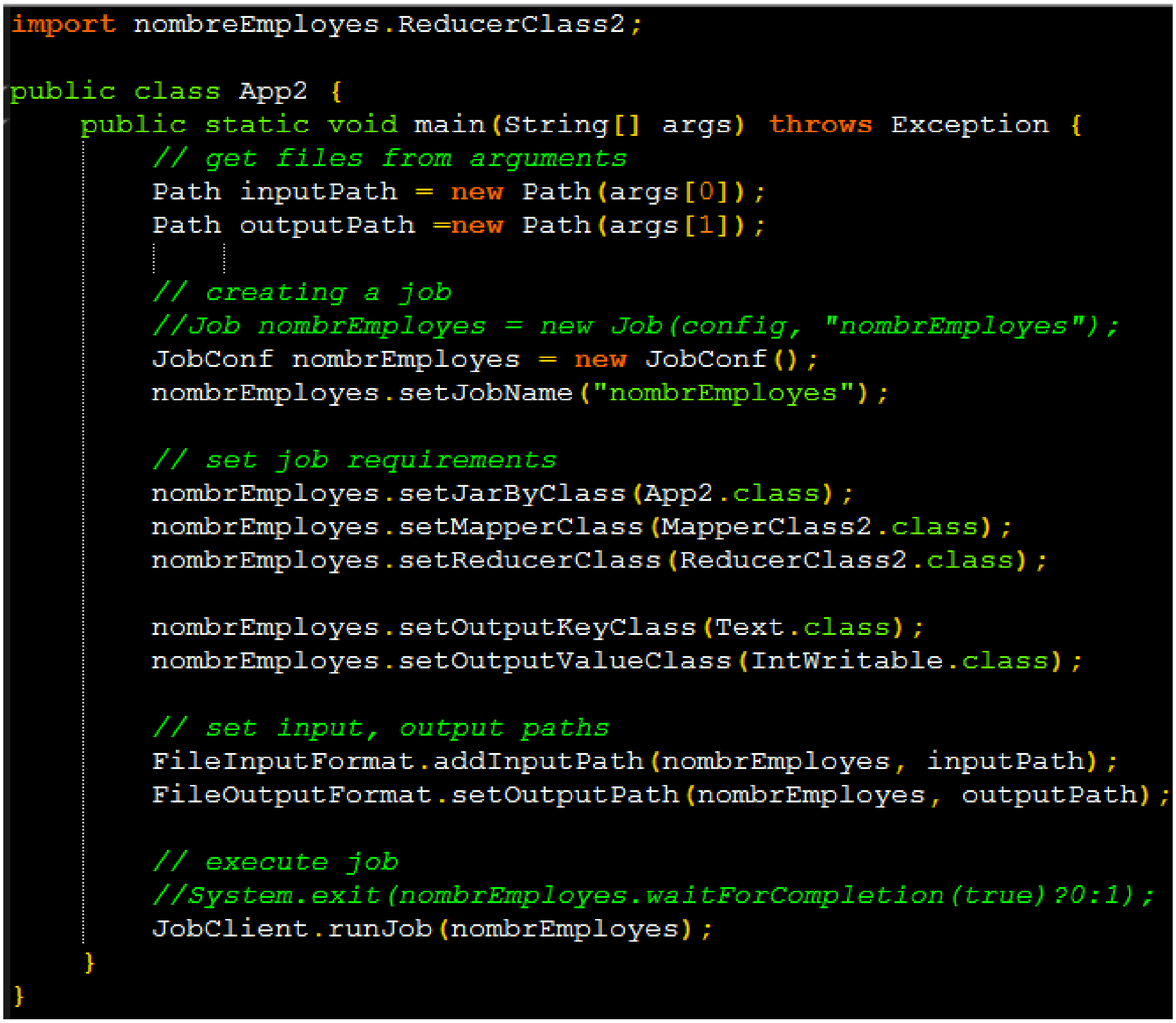
```
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class MapperClass2
extends MapReduceBase
implements Mapper<LongWritable, Text, Text, IntWritable>
{

    public void map(
            LongWritable key,
            Text line,
            OutputCollector<Text, IntWritable> output,
            Reporter reporter)
            throws IOException
    {
        // nom,prenom,departement,metier,salaire
        System.out.println("hello Mapper");
                String[] lineSplitted = line.toString().trim()
                .split(",");
                String departementKeyOut = lineSplitted[2]; // key
                String employeValueOut = lineSplitted[0]; // value

                output.collect(new Text(departementKeyOut),
                new IntWritable(1));
        System.out.println("Hello 12");
    }
}
```

Reducer



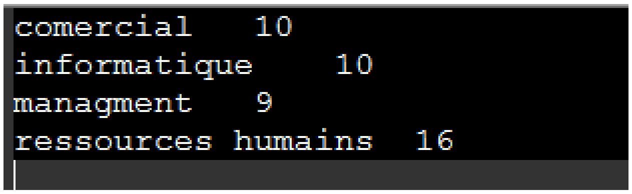
```
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class ReducerClass2
extends MapReduceBase
implements Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(
            Text key,
            Iterator<IntWritable> values,
            OutputCollector<Text, IntWritable> output,
            Reporter reporter)
            throws IOException {
        System.out.println("ello Reducer");

        int sum = 0;
        while(values.hasNext()) {
            sum ++;
            System.out.println(sum);
            values.next();
        }

        //context.write(new Text(outDep), new Text(outSalaire));
        output.collect(key, new IntWritable(sum));

    }
}
```

Application Job



```java
import nombreEmployes.ReducerClass2;

public class App2 {
    public static void main(String[] args) throws Exception {
        // get files from arguments
        Path inputPath = new Path(args[0]);
        Path outputPath =new Path(args[1]);

        // creating a job
        //Job nombrEmployes = new Job(config, "nombrEmployes");
        JobConf nombrEmployes = new JobConf();
        nombrEmployes.setJobName("nombrEmployes");

        // set job requirements
        nombrEmployes.setJarByClass(App2.class);
        nombrEmployes.setMapperClass(MapperClass2.class);
        nombrEmployes.setReducerClass(ReducerClass2.class);

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

        // set input, output paths
        FileInputFormat.addInputPath(nombrEmployes, inputPath);
        FileOutputFormat.setOutputPath(nombrEmployes, outputPath);

        // execute job
        //System.exit(nombrEmployes.waitForCompletion(true)?0:1);
        JobClient.runJob(nombrEmployes);
    }
}
```

Résultat



```
comercial    10
informatique     10
managment    9
ressources humains   16
```

ENSET-M	II-BDCC 2 S4 2022
Cours Big Data: Fondements et Architectures de stockage	EL AAMIRI Essadeq

Exercice 3 : Analyse de données météo

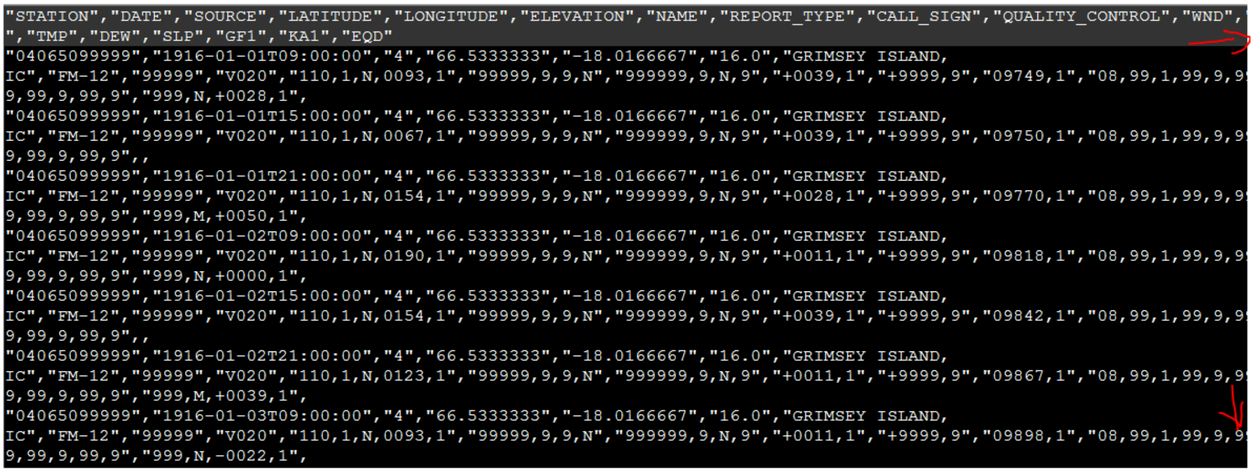
- **Analyse de données météo :**

  - **URI: https://www.ncei.noaa.gov/data/global-hourly/archive/csv/**
  - **C'est un bon exemple d'analyse avec MapReduce, car les capteurs collectent toutes les heures des données météorologiques du monde entier.**
  - **Les données sont semi-structurées.**
  - **Les données sont importées des National Centers from Environmental Information. Il s'agit d'un format ASCII orienté ligne. Chaque ligne est un enregistrement contenant de nombreuses informations.**
  - **Problème à résoudre**

    1. **Extraire les valeurs de température et calculer la température minimale et maximale pour chaque année.**

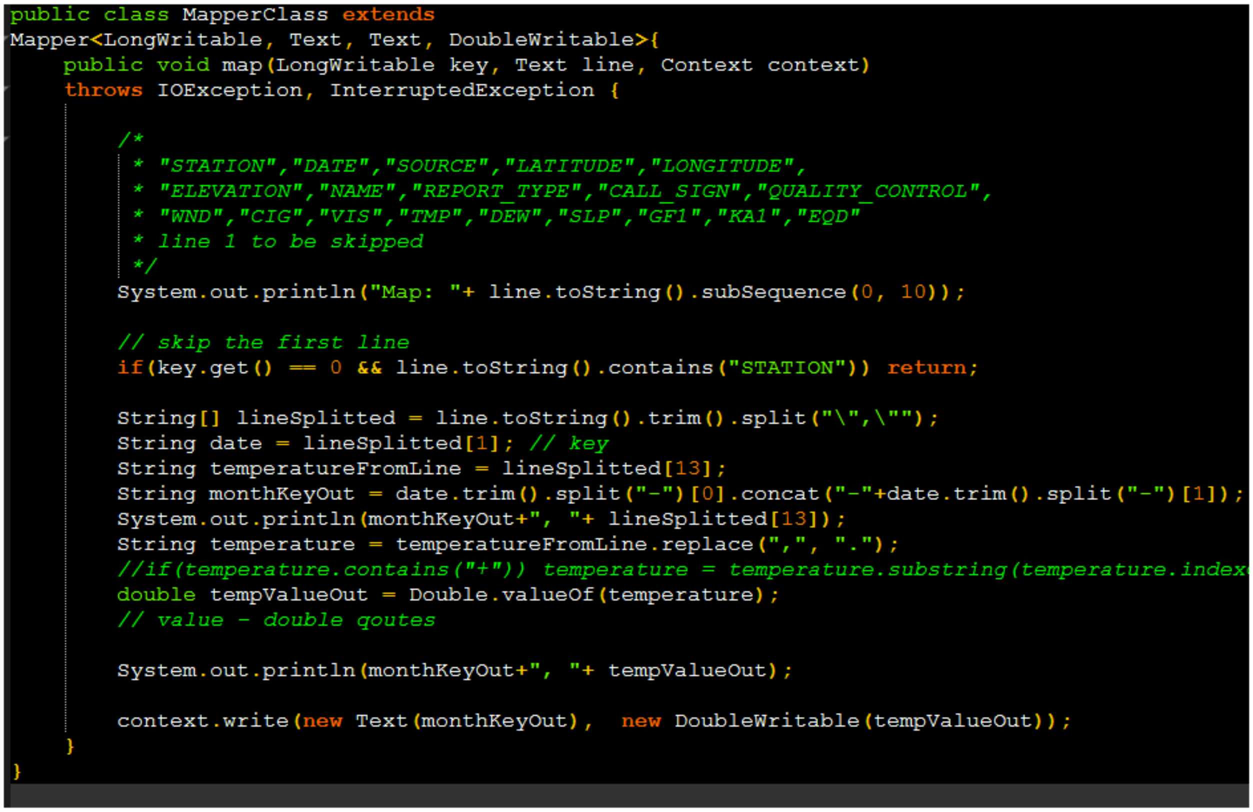
Question 1 : calculons la température minimale et maximal par mois pour l'année 1916.

Données :



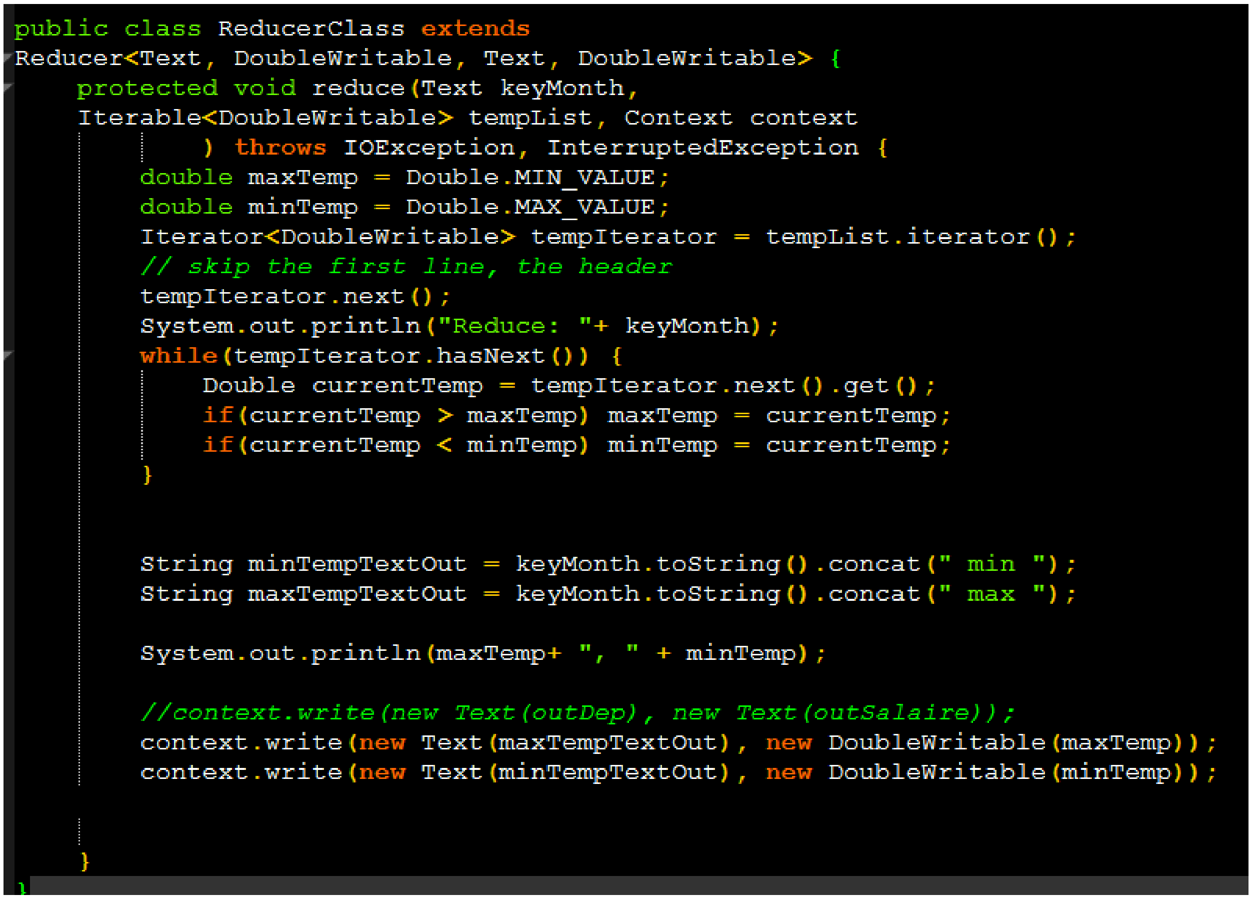
```
"STATION","DATE","SOURCE","LATITUDE","LONGITUDE","ELEVATION","NAME","REPORT_TYPE","CALL_SIGN","QUALITY_CONTROL","WND",
","TMP","DEW","SLP","GF1","KA1","EQD"
"04065099999","1916-01-01T09:00:00","4","66.5333333","-18.0166667","16.0","GRIMSEY ISLAND,
IC","FM-12","99999","V020","110,1,N,0093,1","99999,9,9,N","999999,9,N,9","+0039,1","+9999,9","09749,1","08,99,1,99,9,9
9,99,9,99,9","999,N,+0028,1",
"04065099999","1916-01-01T15:00:00","4","66.5333333","-18.0166667","16.0","GRIMSEY ISLAND,
IC","FM-12","99999","V020","110,1,N,0067,1","99999,9,9,N","999999,9,N,9","+0039,1","+9999,9","09750,1","08,99,1,99,9,9
9,99,9,99,9",,
"04065099999","1916-01-01T21:00:00","4","66.5333333","-18.0166667","16.0","GRIMSEY ISLAND,
IC","FM-12","99999","V020","110,1,N,0154,1","99999,9,9,N","999999,9,N,9","+0028,1","+9999,9","09770,1","08,99,1,99,9,9
9,99,9,99,9","999,M,+0050,1",
"04065099999","1916-01-02T09:00:00","4","66.5333333","-18.0166667","16.0","GRIMSEY ISLAND,
IC","FM-12","99999","V020","110,1,N,0190,1","99999,9,9,N","999999,9,N,9","+0011,1","+9999,9","09818,1","08,99,1,99,9,9
9,99,9,99,9","999,N,+0000,1",
"04065099999","1916-01-02T15:00:00","4","66.5333333","-18.0166667","16.0","GRIMSEY ISLAND,
IC","FM-12","99999","V020","110,1,N,0154,1","99999,9,9,N","999999,9,N,9","+0039,1","+9999,9","09842,1","08,99,1,99,9,9
9,99,9,99,9",,
"04065099999","1916-01-02T21:00:00","4","66.5333333","-18.0166667","16.0","GRIMSEY ISLAND,
IC","FM-12","99999","V020","110,1,N,0123,1","99999,9,9,N","999999,9,N,9","+0011,1","+9999,9","09867,1","08,99,1,99,9,9
9,99,9,99,9","999,M,+0039,1",
"04065099999","1916-01-03T09:00:00","4","66.5333333","-18.0166667","16.0","GRIMSEY ISLAND,
IC","FM-12","99999","V020","110,1,N,0093,1","99999,9,9,N","999999,9,N,9","+0011,1","+9999,9","09898,1","08,99,1,99,9,9
9,99,9,99,9","999,N,-0022,1",
```

Mapper



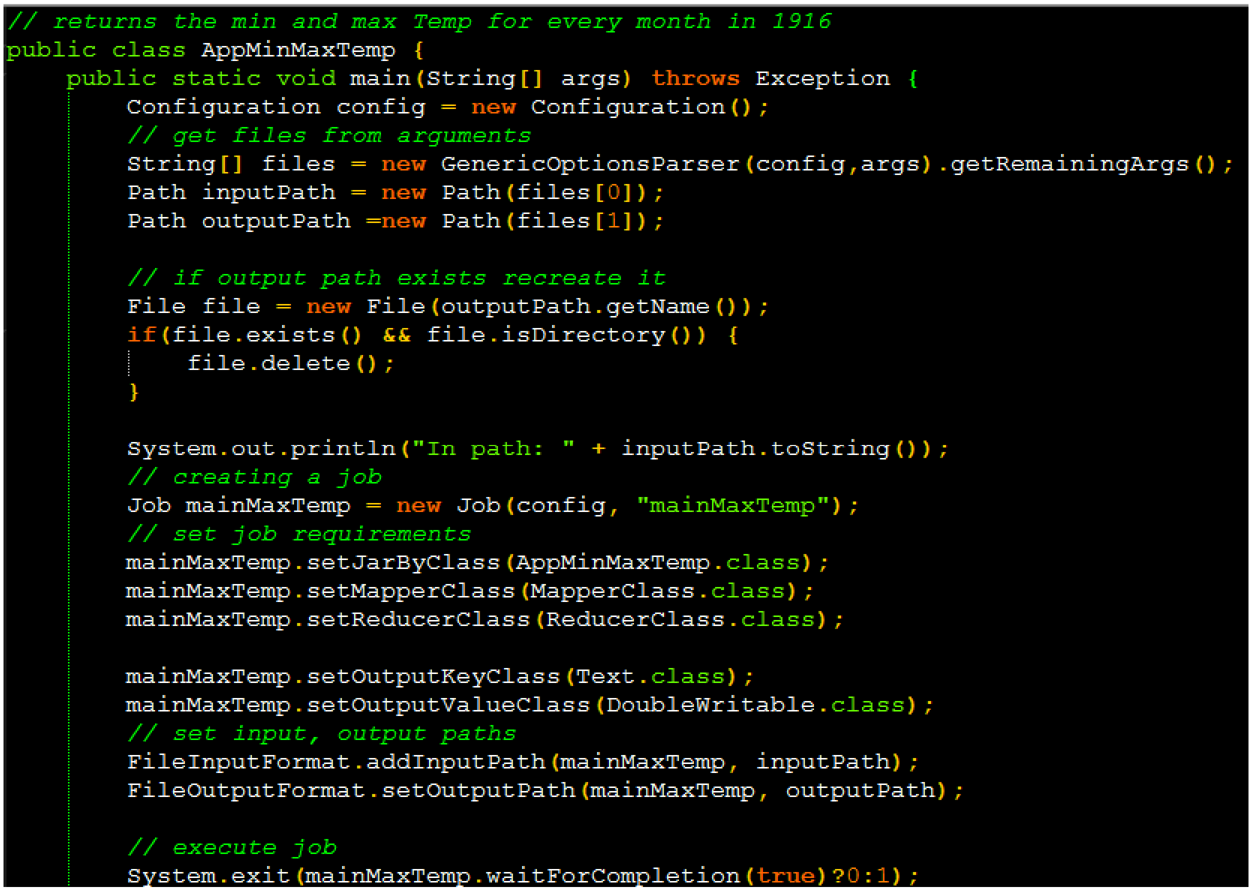
```
public class MapperClass extends
Mapper<LongWritable, Text, Text, DoubleWritable>{
    public void map(LongWritable key, Text line, Context context)
    throws IOException, InterruptedException {

        /*
         * "STATION","DATE","SOURCE","LATITUDE","LONGITUDE",
         * "ELEVATION","NAME","REPORT_TYPE","CALL_SIGN","QUALITY_CONTROL",
         * "WND","CIG","VIS","TMP","DEW","SLP","GF1","KA1","EQD"
         * line 1 to be skipped
         */
        System.out.println("Map: "+ line.toString().subSequence(0, 10));

        // skip the first line
        if(key.get() == 0 && line.toString().contains("STATION")) return;

        String[] lineSplitted = line.toString().trim().split("\",\"");
        String date = lineSplitted[1]; // key
        String temperatureFromLine = lineSplitted[13];
        String monthKeyOut = date.trim().split("-")[0].concat("-"+date.trim().split("-")[1]);
        System.out.println(monthKeyOut+", "+ lineSplitted[13]);
        String temperature = temperatureFromLine.replace(",", ".");
        //if(temperature.contains("+")) temperature = temperature.substring(temperature.index
        double tempValueOut = Double.valueOf(temperature);
        // value - double qoutes

        System.out.println(monthKeyOut+", "+ tempValueOut);

        context.write(new Text(monthKeyOut),  new DoubleWritable(tempValueOut));
    }
}
```

Reducer



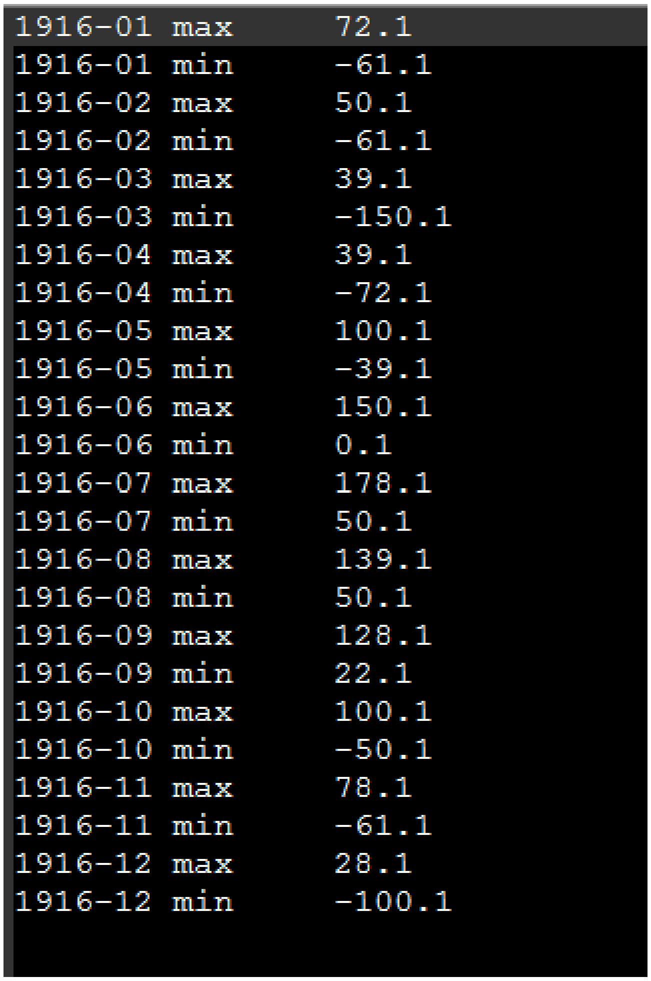
```
public class ReducerClass extends
Reducer<Text, DoubleWritable, Text, DoubleWritable> {
    protected void reduce(Text keyMonth,
    Iterable<DoubleWritable> tempList, Context context
            ) throws IOException, InterruptedException {
        double maxTemp = Double.MIN_VALUE;
        double minTemp = Double.MAX_VALUE;
        Iterator<DoubleWritable> tempIterator = tempList.iterator();
        // skip the first line, the header
        tempIterator.next();
        System.out.println("Reduce: "+ keyMonth);
        while(tempIterator.hasNext()) {
            Double currentTemp = tempIterator.next().get();
            if(currentTemp > maxTemp) maxTemp = currentTemp;
            if(currentTemp < minTemp) minTemp = currentTemp;
        }


        String minTempTextOut = keyMonth.toString().concat(" min ");
        String maxTempTextOut = keyMonth.toString().concat(" max ");

        System.out.println(maxTemp+ ", " + minTemp);

        //context.write(new Text(outDep), new Text(outSalaire));
        context.write(new Text(maxTempTextOut), new DoubleWritable(maxTemp));
        context.write(new Text(minTempTextOut), new DoubleWritable(minTemp));


    }
}
```

Application Job



```
// returns the min and max Temp for every month in 1916
public class AppMinMaxTemp {
    public static void main(String[] args) throws Exception {
        Configuration config = new Configuration();
        // get files from arguments
        String[] files = new GenericOptionsParser(config,args).getRemainingArgs();
        Path inputPath = new Path(files[0]);
        Path outputPath =new Path(files[1]);

        // if output path exists recreate it
        File file = new File(outputPath.getName());
        if(file.exists() && file.isDirectory()) {
            file.delete();
        }

        System.out.println("In path: " + inputPath.toString());
        // creating a job
        Job mainMaxTemp = new Job(config, "mainMaxTemp");
        // set job requirements
        mainMaxTemp.setJarByClass(AppMinMaxTemp.class);
        mainMaxTemp.setMapperClass(MapperClass.class);
        mainMaxTemp.setReducerClass(ReducerClass.class);

        mainMaxTemp.setOutputKeyClass(Text.class);
        mainMaxTemp.setOutputValueClass(DoubleWritable.class);
        // set input, output paths
        FileInputFormat.addInputPath(mainMaxTemp, inputPath);
        FileOutputFormat.setOutputPath(mainMaxTemp, outputPath);

        // execute job
        System.exit(mainMaxTemp.waitForCompletion(true)?0:1);
```

Résultat



```
1916-01 max     72.1
1916-01 min     -61.1
1916-02 max     50.1
1916-02 min     -61.1
1916-03 max     39.1
1916-03 min     -150.1
1916-04 max     39.1
1916-04 min     -72.1
1916-05 max     100.1
1916-05 min     -39.1
1916-06 max     150.1
1916-06 min     0.1
1916-07 max     178.1
1916-07 min     50.1
1916-08 max     139.1
1916-08 min     50.1
1916-09 max     128.1
1916-09 min     22.1
1916-10 max     100.1
1916-10 min     -50.1
1916-11 max     78.1
1916-11 min     -61.1
1916-12 max     28.1
1916-12 min     -100.1
```