Generic Explanation:

This assignment required multiple things such as file handling, string parsing etc. I am using file handling for *positive.txt and negative.txt.* Whereas the input file (*Comments.txt*) is being read by the mapper function itself. I am converting the whole line into a string and then parsing the string by storing the string into a different variable. For this I hardcoded the parser code as I saw the input file and before every comment, there is always a “T”.

Another implementation was that in order to see if the reducer has ended, I am using a built-in function Cleaner (). This function is basically called when the reducer has finished its tasks.

Furthermore, for this assignment I am using HDFS as for some reason when I installed HDFS, my Hadoop-standalone stopped working.

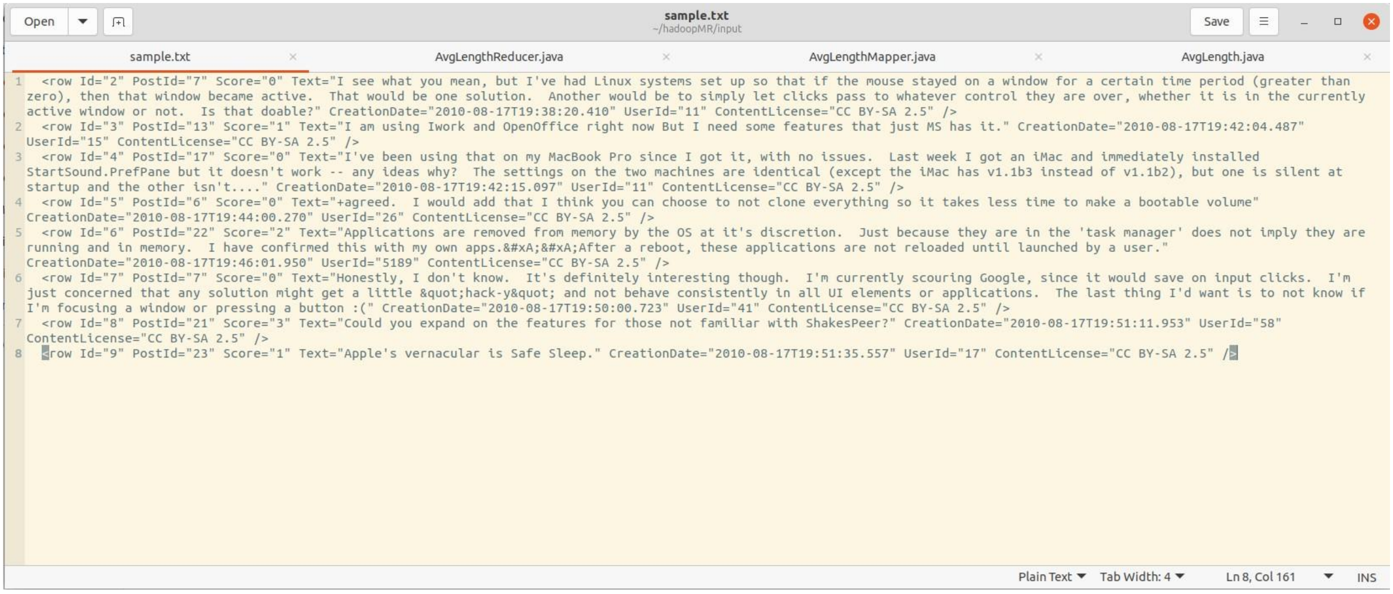
Rest of the implementation is requirement specific explained below:

Average Length of Comments:

For this task, I started finding the length of each comment using the built-in functions. Then I added them in a variable and was counting the iterations. Once the iterations ended, I calculated the average length of all comments using the formula below:

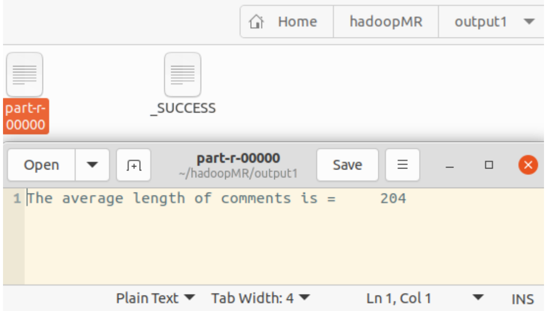
Average = Sum of lengths of comments / Total number of comments

I ran it on a sample dataset first and then verified my answer with an online compiler and the answer was correct. Attached are the screenshots of the sample code, its output and the output of an online compiler:



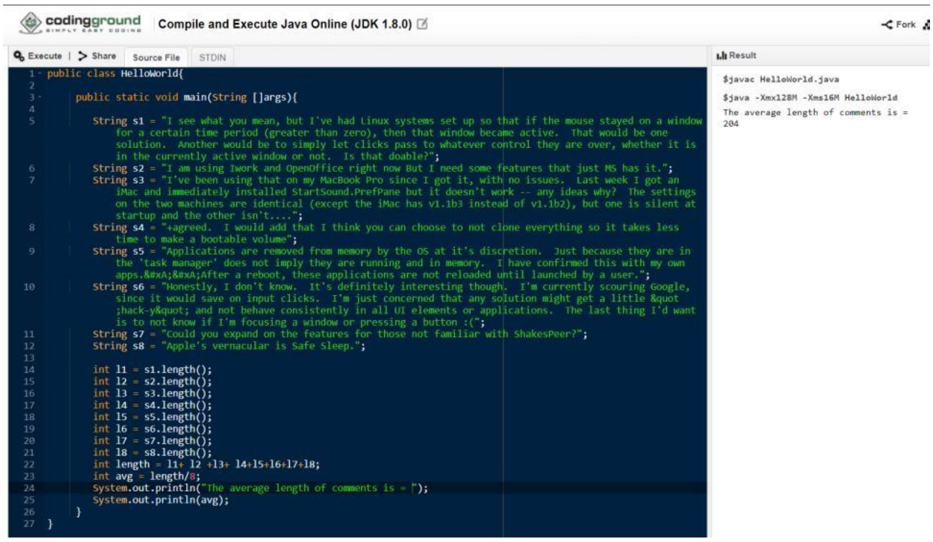
```
 <row Id="2" PostId="7" Score="0" Text="I see what you mean, but I've had Linux systems set up so that if the mouse stayed on a window for a certain time period (greater than zero), then that window became active.  That would be one solution.  Another would be to simply let clicks pass to whatever control they are over, whether it is in the currently active window or not.  Is that doable?" CreationDate="2010-08-17T19:38:20.410" UserId="11" ContentLicense="CC BY-SA 2.5" />
 <row Id="3" PostId="13" Score="1" Text="I am using Iwork and OpenOffice right now But I need some features that just MS has it." CreationDate="2010-08-17T19:42:04.487" UserId="15" ContentLicense="CC BY-SA 2.5" />
 <row Id="4" PostId="17" Score="0" Text="I've been using that on my MacBook Pro since I got it, with no issues.  Last week I got an iMac and immediately installed StartSound.PrefPane but it doesn't work -- any ideas why?  The settings on the two machines are identical (except the iMac has v1.1b3 instead of v1.1b2), but one is silent at startup and the other isn't...." CreationDate="2010-08-17T19:42:15.097" UserId="11" ContentLicense="CC BY-SA 2.5" />
 <row Id="5" PostId="6" Score="0" Text="+agreed.  I would add that I think you can choose to not clone everything so it takes less time to make a bootable volume" CreationDate="2010-08-17T19:44:00.270" UserId="26" ContentLicense="CC BY-SA 2.5" />
 <row Id="6" PostId="22" Score="2" Text="Applications are removed from memory by the OS at it's discretion.  Just because they are in the 'task manager' does not imply they are running and in memory.  I have confirmed this with my own apps.&#xA;&#xA;After a reboot, these applications are not reloaded until launched by a user." CreationDate="2010-08-17T19:46:01.950" UserId="5189" ContentLicense="CC BY-SA 2.5" />
 <row Id="7" PostId="7" Score="0" Text="Honestly, I don't know.  It's definitely interesting though.  I'm currently scouring Google, since it would save on input clicks.  I'm just concerned that any solution might get a little &quot;hack-y&quot; and not behave consistently in all UI elements or applications.  The last thing I'd want is to not know if I'm focusing a window or pressing a button :(" CreationDate="2010-08-17T19:50:00.723" UserId="41" ContentLicense="CC BY-SA 2.5" />
 <row Id="8" PostId="21" Score="3" Text="Could you expand on the features for those not familiar with ShakesPeer?" CreationDate="2010-08-17T19:51:11.953" UserId="58" ContentLicense="CC BY-SA 2.5" />
 <row Id="9" PostId="23" Score="1" Text="Apple's vernacular is Safe Sleep." CreationDate="2010-08-17T19:51:35.557" UserId="17" ContentLicense="CC BY-SA 2.5" />
```

Fig 2.1: Contents of Sample.txt



```
The average length of comments is =     204
```

Fig 2.2: Output of Sample.txt

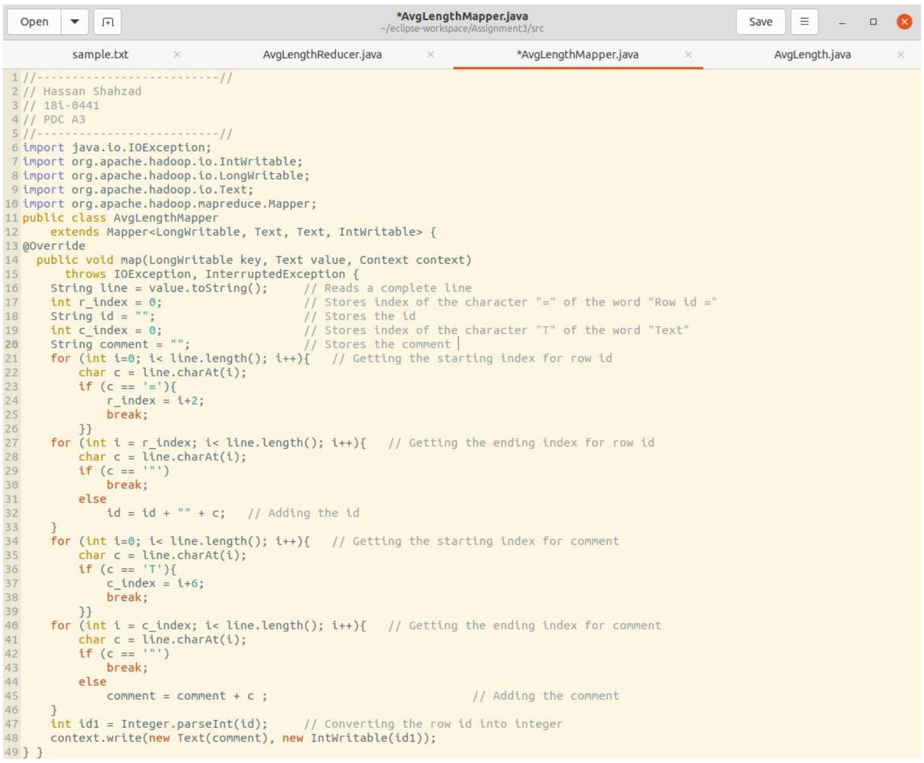


```java
public class HelloWorld{

    public static void main(String []args){

        String s1 = "I see what you mean, but I've had Linux systems set up so that if the mouse stayed on a window for a certain time period (greater than zero), then that window became active.  That would be one solution.  Another would be to simply let clicks pass to whatever control they are over, whether it is in the currently active window or not.  Is that doable?";
        String s2 = "I am using Iwork and OpenOffice right now But I need some features that just MS has it.";
        String s3 = "I've been using that on my MacBook Pro since I got it, with no issues.  Last week I got an iMac and immediately installed StartSound.PrefPane but it doesn't work -- any ideas why?  The settings on the two machines are identical (except the iMac has v1.1b3 instead of v1.1b2), but one is silent at startup and the other isn't....";
        String s4 = "+agreed.  I would add that I think you can choose to not clone everything so it takes less time to make a bootable volume";
        String s5 = "Applications are removed from memory by the OS at it's discretion.  Just because they are in the 'task manager' does not imply they are running and in memory.  I have confirmed this with my own apps.&#xA;&#xA;After a reboot, these applications are not reloaded until launched by a user.";
        String s6 = "Honestly, I don't know.  It's definitely interesting though.  I'm currently scouring Google, since it would save on input clicks.  I'm just concerned that any solution might get a little &quot;hack-y&quot; and not behave consistently in all UI elements or applications.  The last thing I'd want is to not know if I'm focusing a window or pressing a button :(";
        String s7 = "Could you expand on the features for those not familiar with ShakesPeer?";
        String s8 = "Apple's vernacular is Safe Sleep.";

        int l1 = s1.length();
        int l2 = s2.length();
        int l3 = s3.length();
        int l4 = s4.length();
        int l5 = s5.length();
        int l6 = s6.length();
        int l7 = s7.length();
        int l8 = s8.length();
        int length = l1+ l2 +l3+ l4+l5+l6+l7+l8;
        int avg = length/8;
        System.out.println("The average length of comments is = ");
        System.out.println(avg);
    }
}
```

```
$javac HelloWorld.java
$java -Xmx128M -Xms16M HelloWorld
The average length of comments is =
204
```

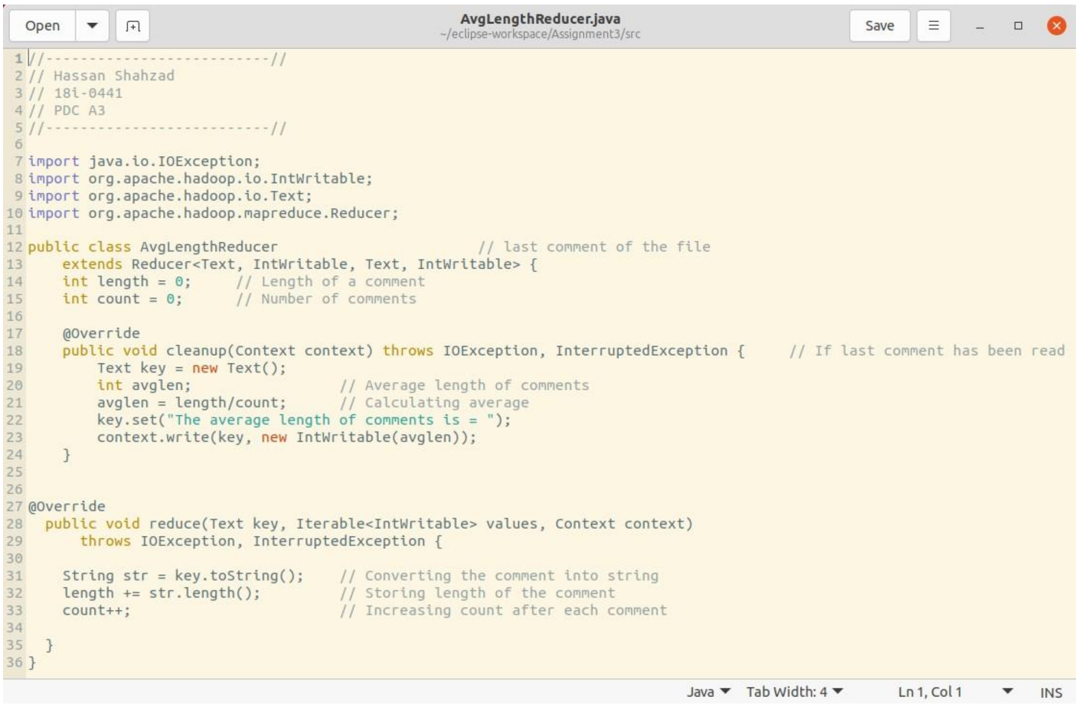
Fig 2.3: Output of Sample.txt on an online compiler

After verifying my output, I ran the same code on the whole Comments.txt file. For that the code used and the outputs are given below:



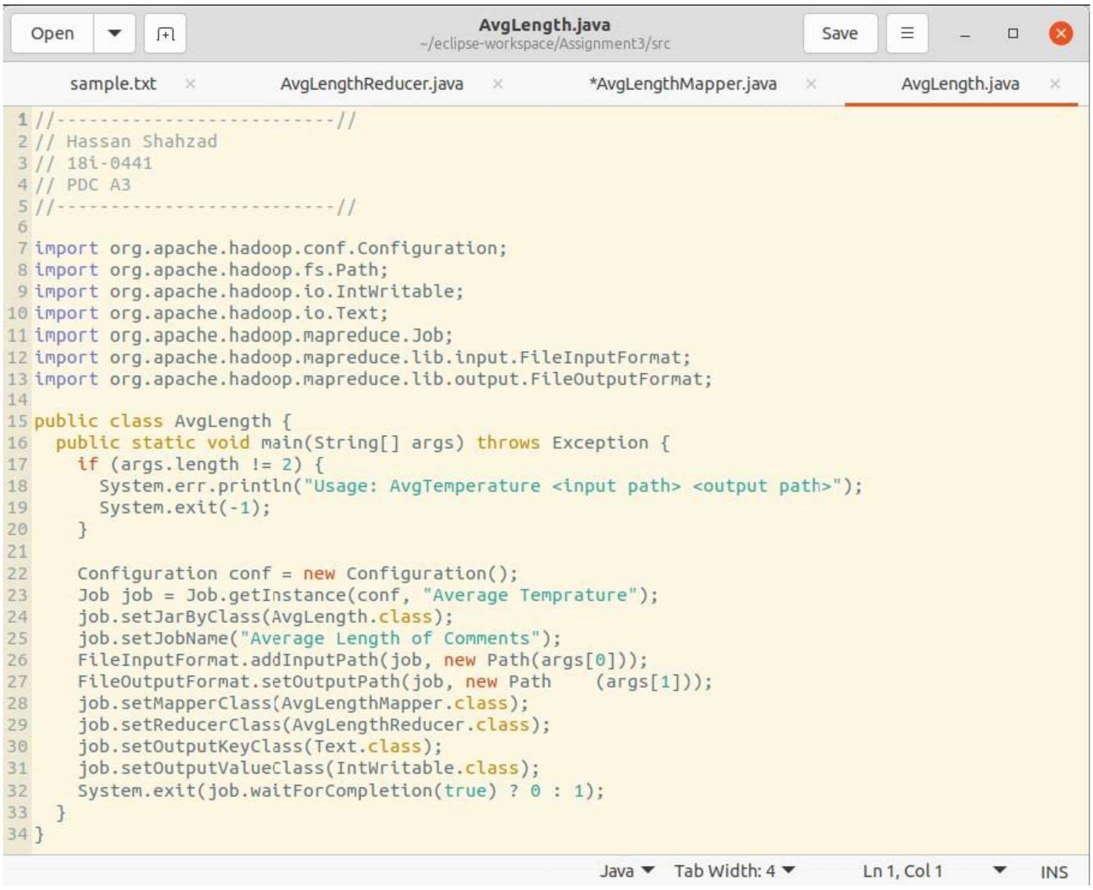
```java
//-------------------------//
// Hassan Shahzad
// 18i-0441
// PDC A3
//-------------------------//
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class AvgLengthMapper
    extends Mapper<LongWritable, Text, Text, IntWritable> {
@Override
  public void map(LongWritable key, Text value, Context context)
      throws IOException, InterruptedException {
    String line = value.toString();      // Reads a complete line
    int r_index = 0;                     // Stores index of the character "=" of the word "Row id ="
    String id = "";                      // Stores the id
    int c_index = 0;                     // Stores index of the character "T" of the word "Text"
    String comment = "";                 // Stores the comment
    for (int i=0; i< line.length(); i++){   // Getting the starting index for row id
        char c = line.charAt(i);
        if (c == '='){
            r_index = i+2;
            break;
        }}
    for (int i = r_index; i< line.length(); i++){   // Getting the ending index for row id
        char c = line.charAt(i);
        if (c == '"')
            break;
        else
            id = id + "" + c;   // Adding the id
    }
    for (int i=0; i< line.length(); i++){   // Getting the starting index for comment
        char c = line.charAt(i);
        if (c == 'T'){
            c_index = i+6;
            break;
        }}
    for (int i = c_index; i< line.length(); i++){   // Getting the ending index for comment
        char c = line.charAt(i);
        if (c == '"')
            break;
        else
            comment = comment + c ;                              // Adding the comment
    }
    int id1 = Integer.parseInt(id);     // Converting the row id into integer
    context.write(new Text(comment), new IntWritable(id1));
} }
```

Fig 2.4: Mapper Class



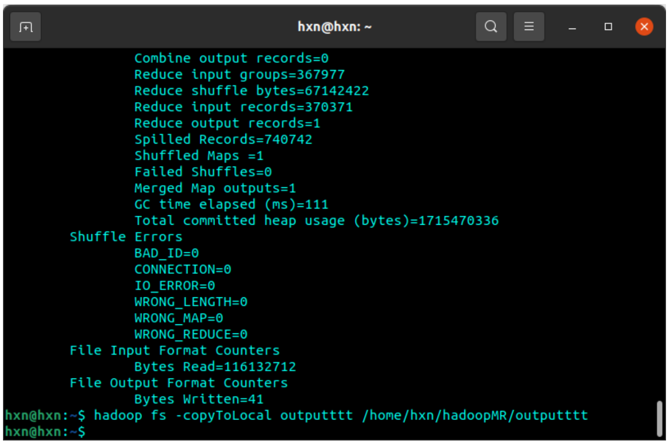
```java
//-------------------------//
// Hassan Shahzad
// 18i-0441
// PDC A3
//-------------------------//

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class AvgLengthReducer                       // last comment of the file
    extends Reducer<Text, IntWritable, Text, IntWritable> {
    int length = 0;      // Length of a comment
    int count = 0;       // Number of comments

    @Override
    public void cleanup(Context context) throws IOException, InterruptedException {     // If last comment has been read
        Text key = new Text();
        int avglen;                 // Average length of comments
        avglen = length/count;      // Calculating average
        key.set("The average length of comments is = ");
        context.write(key, new IntWritable(avglen));
    }


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

    String str = key.toString();    // Converting the comment into string
    length += str.length();         // Storing length of the comment
    count++;                        // Increasing count after each comment

  }
}
```

Fig 2.5: Reducer Class



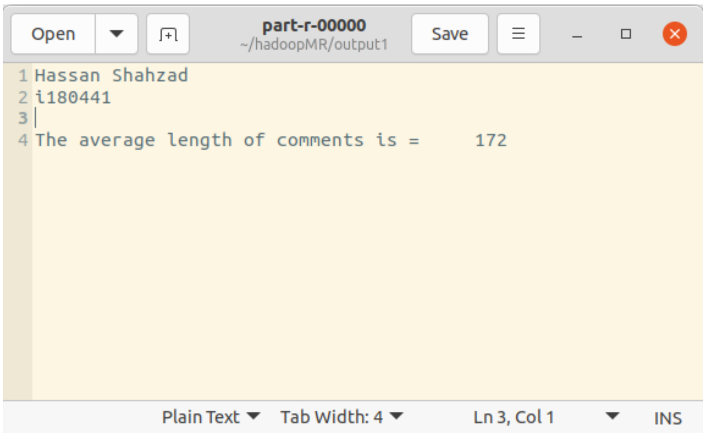
```java
//-------------------------//
// Hassan Shahzad
// 18i-0441
// PDC A3
//-------------------------//

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

public class AvgLength {
  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.println("Usage: AvgTemperature <input path> <output path>");
      System.exit(-1);
    }

    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "Average Temprature");
    job.setJarByClass(AvgLength.class);
    job.setJobName("Average Length of Comments");
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path    (args[1]));
    job.setMapperClass(AvgLengthMapper.class);
    job.setReducerClass(AvgLengthReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Fig 2.6: Main Java File



```
                Combine output records=0
                Reduce input groups=367977
                Reduce shuffle bytes=67142422
                Reduce input records=370371
                Reduce output records=1
                Spilled Records=740742
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=111
                Total committed heap usage (bytes)=1715470336
        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=116132712
        File Output Format Counters
                Bytes Written=41
hxn@hxn:~$ hadoop fs -copyToLocal outputttt /home/hxn/hadoopMR/outputttt
hxn@hxn:~$
```

Fig 2.7: Execution of Code via HDFS



```
Hassan Shahzad
i180441

The average length of comments is =     172
```

Fig 2.8: Output of File