```
import java.io.IOException;
import java.util.*;
import java.lang.Object;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
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.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import Pi.Map;
import Pi.Reduce;

public class CountObjectSize {

                public static final String OBJECT = "Object";
                public static final String BACKGROUND = "Background";
                private static String WORD = "Word";

                  protected void map(LongWritable key, Text value, Context context)
                     throws java.io.IOException, InterruptedException {
                   int tokens = value;
                   for(String token : tokens){
                           if value.getBytes()>15;
                           Counter counter= context.getCounter(WORD, token);
```

```
        counter.increment(1);
        System.out.println(counter);
}

public static class Reduce extends Reducer
        {
          public void reduce(Text key, Iterable<IntWritable> values,
             Context context) throws IOException, InterruptedException
          {
            int sum = 0;
            for (IntWritable val : values) {
              sum += val.get();
            }
            context.write(key, new IntWritable(sum));
          }
        }

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

          Job job = new Job(conf, "Calculate Pi");
          job.setJarByClass(Pi.class);
          job.setOutputKeyClass(Text.class);
          job.setOutputValueClass(IntWritable.class);

          job.setMapperClass(Map.class);
          job.setReducerClass(Reduce.class);
          job.setCombinerClass(Reduce.class);
```

```
                job.setInputFormatClass(TextInputFormat.class);
                job.setOutputFormatClass(TextOutputFormat.class);

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

                job.waitForCompletion(true);
            }


    }
}
```