```
import java.io.BufferedReader;
import java.io.*;
import java.io.File;
import java.io.FileWriter;
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.Reducer.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;

public class Pi {

     public static class Map extends Mapper
       {
         private final static IntWritable one = new IntWritable(1);
         private static int IN = 0;
         private static int OUT = 0;
         private Text word = new Text();

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

```
      String line = value.toString();
      StringTokenizer tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()) {
          String x,y;
          x=tokenizer.nextToken();
          y=tokenizer.nextToken();
          int xvalue=(int)(Integer.parseInt(x));
          int yvalue=(int)(Integer.parseInt(y));
          double check=Math.sqrt(Math.pow((2-xvalue),2)+Math.pow((2-yvalue),2));
                              if(check<2)
                                      IN++;
                              OUT++;
                              double pi=4*(IN/(IN+OUT));
        word.set("pi value:  "+pi);
      }
    }
  }

  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);
    }
  }
```

Random Integer

```
import java.util.Scanner;

public class GenerateRandomNumbers {
	public static void main(String[] args) {
		System.out.println("How many random numbers to generate:");
```

```
        Scanner input =new Scanner(System.in);
        int RandomNumCount = input.nextInt();

        System.out.println("What's the radius number?");
        int radius = input.nextInt();
        int diameter = radius * 2;

        int xvalue = 0;
        int yvalue = 0;
        int inside = 0;
        int outside = 0;
        for(int i=0;i<RandomNumCount;i++){
          xvalue = (int)(Math.random()*diameter);
          yvalue = (int)(Math.random()*diameter);

          double check = Math.sqrt(Math.pow((radius-xvalue), 2) +
                    Math.pow((radius-yvalue), 2));

            if (check < radius) {
                inside++;
            } else {
                outside++;
            }
        }
        System.out.println("");
        System.out.println("inside value " + inside);
        System.out.println("outside value " + outside);

        double possibility = (double)inside / (double)(inside + outside);
        System.out.println("p:" + possibility);
        double piValue = 4 * possibility;
        System.out.println("Pi value is " + piValue);

    }
}
```