# Creating MapReduce Program to Calculating Pi

San Francisco Bay University
Dipali Gajera (19645)

# Contents

## INTRODUCTION

- ❖ To access massive data kept in the Hadoop File System, the MapReduce programming paradigm or pattern is part of the Hadoop framework (HDFS).
- ❖ As we all know, pi (π)—is the ratio of the circumference of any circle to the diameter of that circle. Regardless of the circle's size, this ratio will always equal pi. In decimal form, the value of pi is approximately 3.14.
- ❖ Here, this project is performed on Virtual Machine using MapReduce for the cs246 Ubantu vid.
- ❖ For the development of this project, Eclipse software has been used.

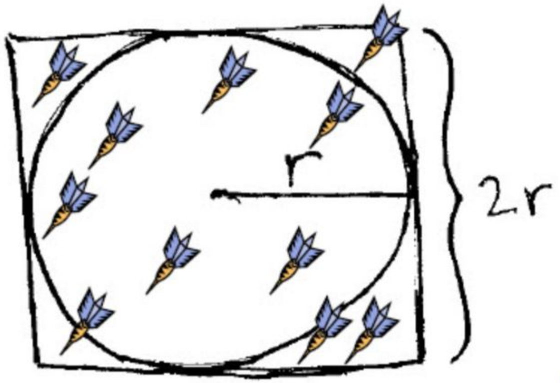
## DESIGN

- Here, we have an example of dart board where, we have to count the number and radius of dart.
- Therefore, we have a java program in which, will generate the random number and give two commands  radius and number  of (x,y) pairs gor calculation of pi.
- If, we want accurate estimate of pi , for that we need larger number of random sample.
- For the fastness of the program, we need N number of darts.

# MapReduce Pi

## Overview

- Throw *N* darts on the board. Each dart lands at a random position $(x,y)$ on the board.



- Note if each dart landed inside the circle or not
  - Check if $x^2+y^2<r$
- Take the total number of darts that landed in the circle as *S*

$$4\left(\frac{S}{N}\right) = \pi$$

## Explanation

- If radius is 5, then based on the input values in Map, we can calculate

```
pi = 4 * (S / N)
   = 4 * (Insde / (Inside + Outside))
   = 4 * (5 / (5 + 7))
   = 4 * (5 / 12)
   = 1.66
```

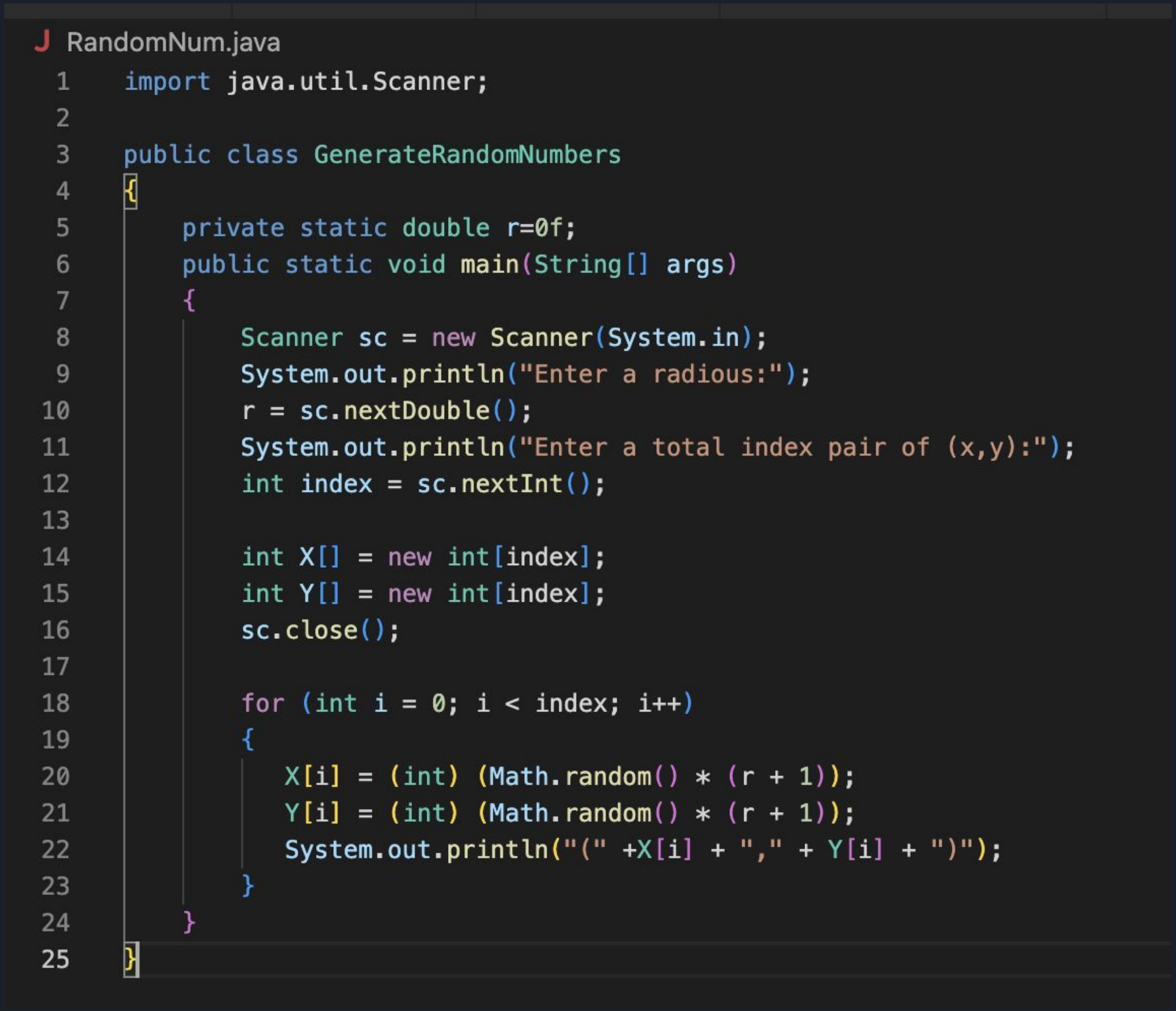
**Note:**

- To get more accurate pi value you can
  - increase the value of the radius, and
  - create much much more input values (e.g., a million values)

| Job: Pi | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Map Task | | | | | | | | Reduce Task | | |
| map() | | | | combine() | | | | reduce() | | |
| Input (Given) | | Output (Program) | | Input (Given) | | Output (Program) | | Input (Given) | | Output (Program) |
| Key | Value (radius=2) | Key | Value (radius=2) | Key | Values | Key | Value | Key | Values | |
| file1 | (0, 1) | Outside | 1 | Inside | [1] | Inside | 1 | Inside | [1, 3, 1] | Inside 5 |
| | (1, 3) | Inside | 1 | Outside | [1, 1] | Outside | 2 | Outside | [2, 1, 4] | Outside 7 |
| | (4, 3) | Outside | 1 | | | | | | | |
| file2 | (2, 3) | Inside | 1 | Inside | [1, 1, 1] | Inside | 3 | | | |
| | (1, 3) | Inside | 1 | Outside | [1] | Outside | 1 | | | |
| | (1, 4) | Outside | 1 | | | | | | | |
| | (3, 2) | Inside | 1 | | | | | | | |
| file3 | (3, 0) | Outside | 1 | Inside | [1] | Inside | 1 | | | |
| | (3, 3) | Inside | 1 | Outside | [1, 1, 1, 1] | Outside | 4 | | | |
| | (3, 4) | Outside | 1 | | | | | | | |
| | (0, 0) | Outside | 1 | | | | | | | |
| | (4, 4) | Outside | 1 | | | | | | | |

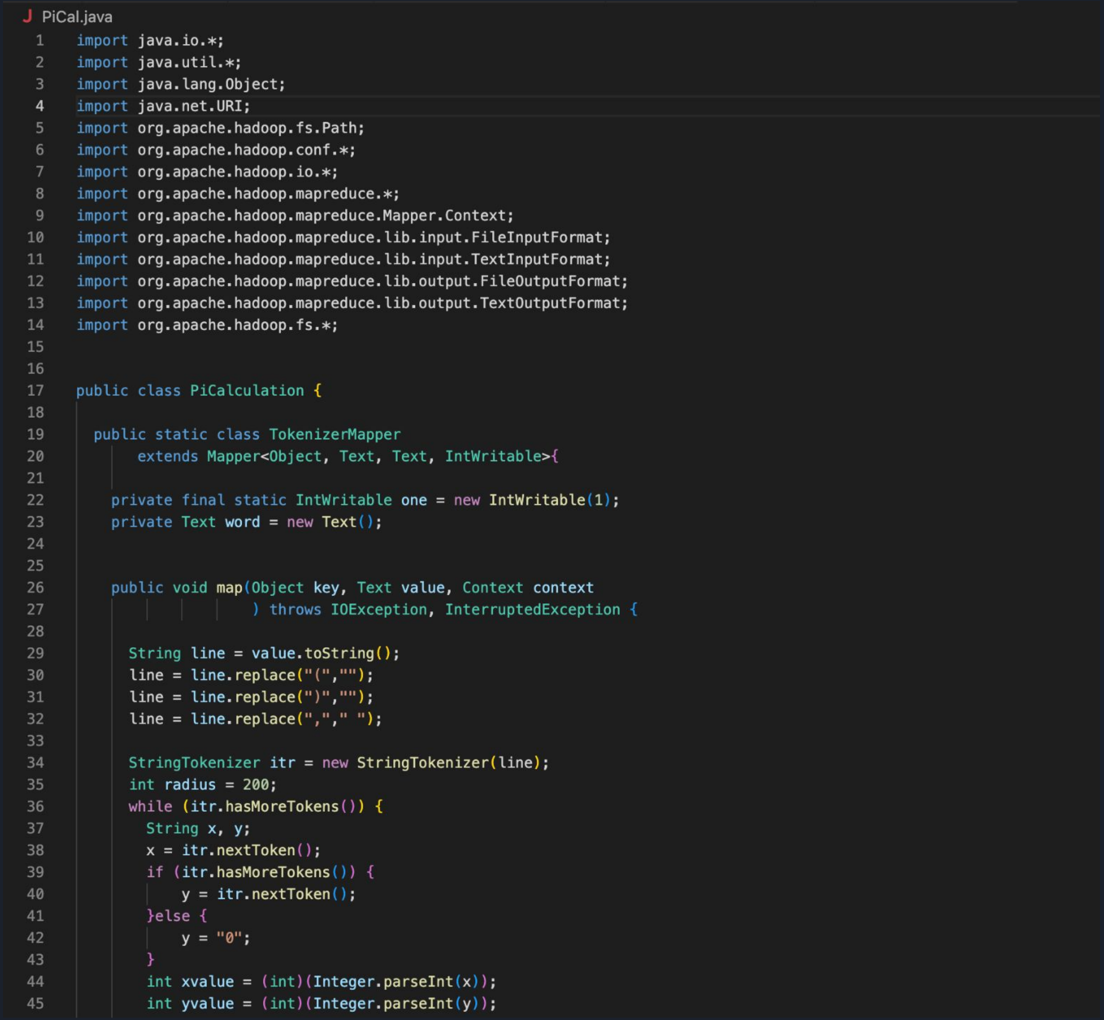
## IMPLEMENTATION

- ❖ This program is performed in Java programing language.
- ❖ The output of the MapReduce program included random values that needed to be recorded into the output file.
- ❖ The algorithm made sure that each number generated was assessed independently.
- ❖ Here, we have the sample code;



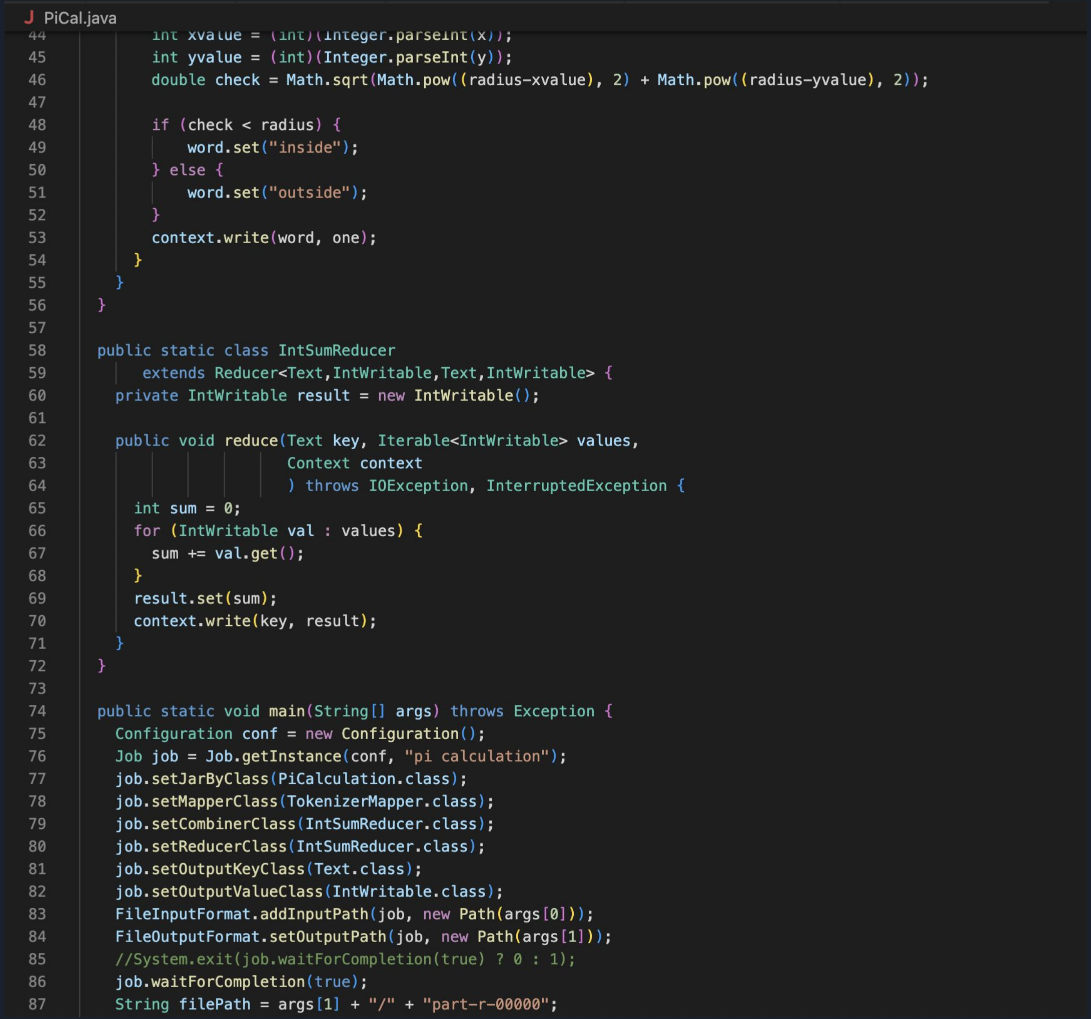
RandomNum.java

```java
import java.util.Scanner;

public class GenerateRandomNumbers
{
    private static double r=0f;
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a radious:");
        r = sc.nextDouble();
        System.out.println("Enter a total index pair of (x,y):");
        int index = sc.nextInt();

        int X[] = new int[index];
        int Y[] = new int[index];
        sc.close();

        for (int i = 0; i < index; i++)
        {
            X[i] = (int) (Math.random() * (r + 1));
            Y[i] = (int) (Math.random() * (r + 1));
            System.out.println("(" +X[i] + "," + Y[i] + ")");
        }
    }
}
```

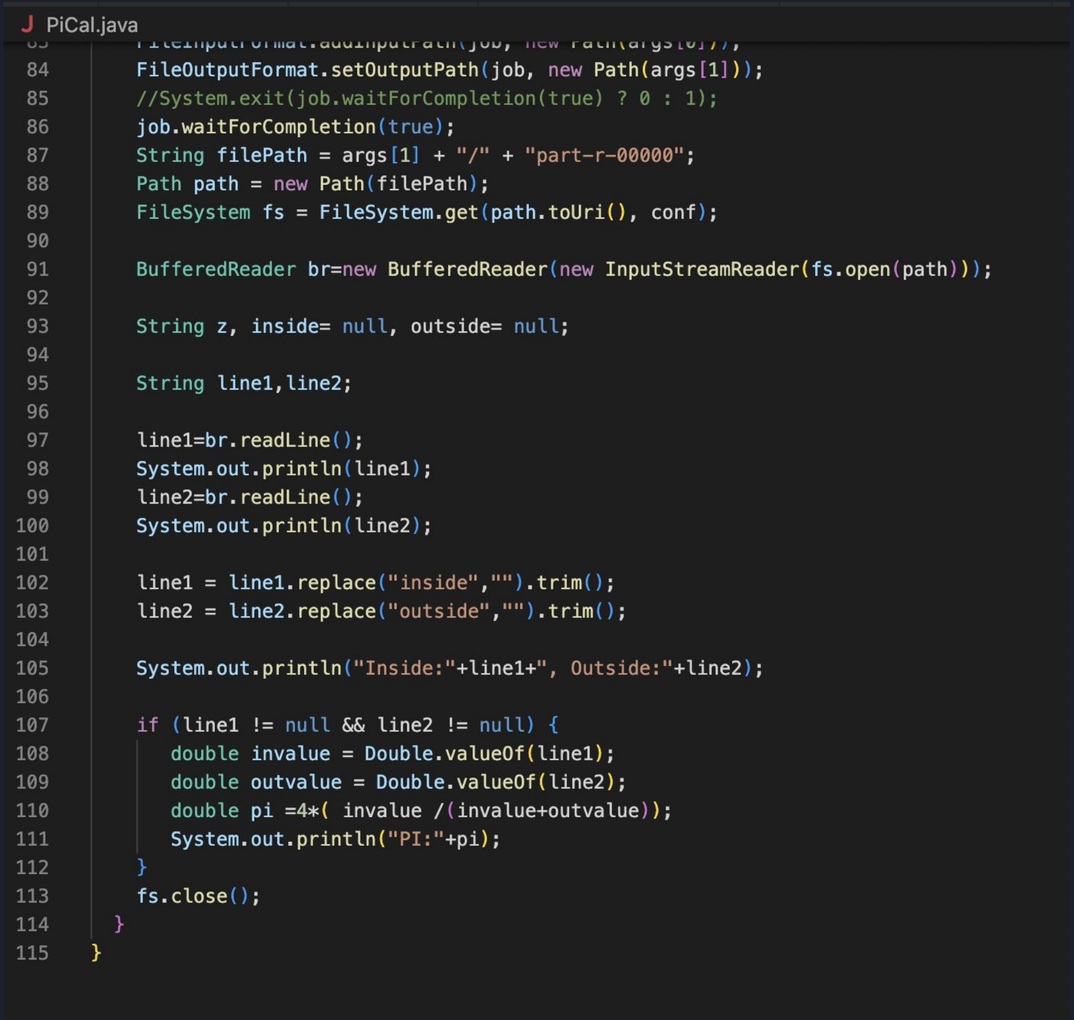


PiCal.java

```java
import java.io.*;
import java.util.*;
import java.lang.Object;
import java.net.URI;
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 org.apache.hadoop.fs.*;


public class PiCalculation {

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

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();


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

      String line = value.toString();
      line = line.replace("(","");
      line = line.replace(")","");
      line = line.replace(","," ");

      StringTokenizer itr = new StringTokenizer(line);
      int radius = 200;
      while (itr.hasMoreTokens()) {
        String x, y;
        x = itr.nextToken();
        if (itr.hasMoreTokens()) {
            y = itr.nextToken();
        }else {
            y = "0";
        }
        int xvalue = (int)(Integer.parseInt(x));
        int yvalue = (int)(Integer.parseInt(y));
```


PiCal.java

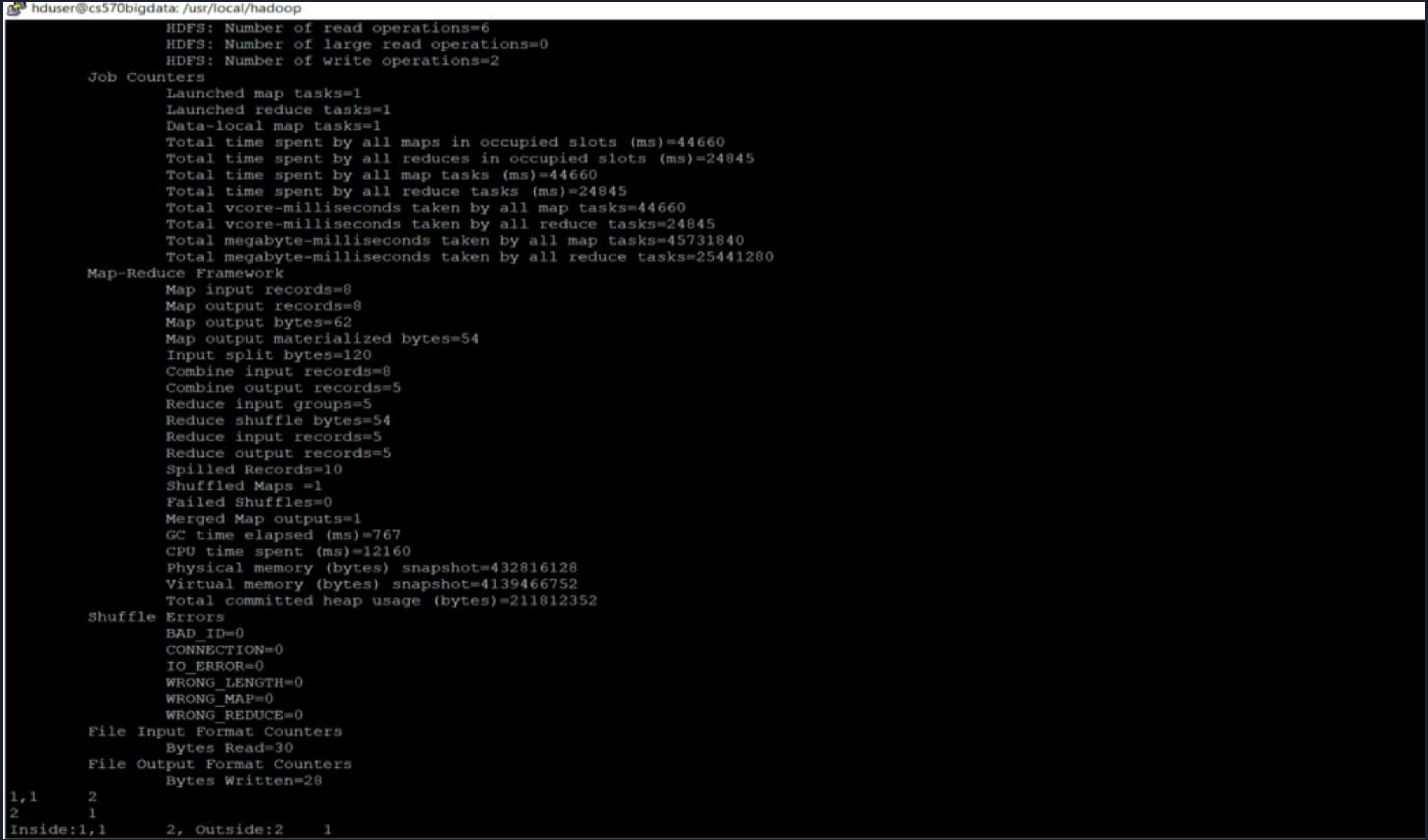
```java
        int xvalue = (int)(Integer.parseInt(x));
        int yvalue = (int)(Integer.parseInt(y));
        double check = Math.sqrt(Math.pow((radius-xvalue), 2) + Math.pow((radius-yvalue), 2));

        if (check < radius) {
            word.set("inside");
        } else {
            word.set("outside");
        }
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

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

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "pi calculation");
    job.setJarByClass(PiCalculation.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.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);
    job.waitForCompletion(true);
    String filePath = args[1] + "/" + "part-r-00000";
```


PiCal.java

```java
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    //System.exit(job.waitForCompletion(true) ? 0 : 1);
    job.waitForCompletion(true);
    String filePath = args[1] + "/" + "part-r-00000";
    Path path = new Path(filePath);
    FileSystem fs = FileSystem.get(path.toUri(), conf);

    BufferedReader br=new BufferedReader(new InputStreamReader(fs.open(path)));

    String z, inside= null, outside= null;

    String line1,line2;

    line1=br.readLine();
    System.out.println(line1);
    line2=br.readLine();
    System.out.println(line2);

    line1 = line1.replace("inside","").trim();
    line2 = line2.replace("outside","").trim();

    System.out.println("Inside:"+line1+", Outside:"+line2);

    if (line1 != null && line2 != null) {
       double invalue = Double.valueOf(line1);
       double outvalue = Double.valueOf(line2);
       double pi =4*( invalue /(invalue+outvalue));
       System.out.println("PI:"+pi);
    }
    fs.close();
  }
}
```

## TEST

- There are many functions has been used like, map() and reduce().
- In order to obtain the x, y N pairs for the pi estimation, it was necessary to parse the input file using map() function.
- The circle's inside and outside values were then added using the reduce () method.


hduser@cs570bigdata: /usr/local/hadoop

```
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=44660
                Total time spent by all reduces in occupied slots (ms)=24845
                Total time spent by all map tasks (ms)=44660
                Total time spent by all reduce tasks (ms)=24845
                Total vcore-milliseconds taken by all map tasks=44660
                Total vcore-milliseconds taken by all reduce tasks=24845
                Total megabyte-milliseconds taken by all map tasks=45731840
                Total megabyte-milliseconds taken by all reduce tasks=25441280
        Map-Reduce Framework
                Map input records=8
                Map output records=8
                Map output bytes=62
                Map output materialized bytes=54
                Input split bytes=120
                Combine input records=8
                Combine output records=5
                Reduce input groups=5
                Reduce shuffle bytes=54
                Reduce input records=5
                Reduce output records=5
                Spilled Records=10
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=767
                CPU time spent (ms)=12160
                Physical memory (bytes) snapshot=432816128
                Virtual memory (bytes) snapshot=4139466752
                Total committed heap usage (bytes)=211812352
        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=30
        File Output Format Counters
                Bytes Written=28
1,1     2
2       1
Inside:1,1      2, Outside:2    1
```

## CONCLUSION

- ❖ The above-illustrated MapReduce pi can be used in a variety of situations, such as astrophysics research for database design or the analysis of pageview numbers.
- ❖ Two key tasks are performed by MapReduce: the reducer, also known as the organizer, organizes and reduces the results from each node into a coherent response to a query, while the mapper filters and distributes work to various nodes within the cluster or map.

## REFERENCE

- https://hc.labnet.sfbu.edu/~henry/npu/classes/mapreduce/pi/slide/exercise_pi.html
- https://subscription.packtpub.com/book/hardware-and-creative/9781783286195/6/ch06lvl1sec34/a-hadoop-application-to-calculate-pi
- https://www.coursera.org/lecture/cloud-applications-part2/2-1-5-mapreduce-example-pi-estimation-image-smoothing-8mfUV