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();

StringTokenizer tokenizer=new StringTokenizer(line);

while (tokenizer.hasMoreTokens()) {

String x,y;

x=tokenizer.nextToken();

if(tokenizer.hasMoreTokens()) {

y=tokenizer.nextToken();

}

else {

y="0";

}

int xvalue=(int)(Integer.parseInt(x));

int yvalue= (int) (Integer.parseInt(y));

double check=Math.sqrt(Math.pow((4-xvalue),2)+Math.pow((4-yvalue),2));

if(check<4){

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, "word count");

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";

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();

}

}