**CPS501 SECTION 02, 2021: FINAL INDIVIDUAL ASSIGNMENT**

Name: Sai Karun Sandugula

UD ID:101691752

**Program:**

Pizzas are usually cooked as a circle, and are often cut “triangularly” along lines moving out from the center, as shown on the left in Figure 1 below. However, some heretical people prefer that the pizza be cut into rectangular pieces.

Besides the obvious aesthetic issues involved, the pieces that intersect the boundary of the circle may not be complete rectangles. Sometimes these pieces are close to the largest size piece, but other times they are so small as to be useless. Given a pizza radius, lengths and widths of the rectangular cuts, and a “too small” percentage (for example, any slice that is less than 50% of the largest slice is too small), how many slices are cut that will be “too small”?

**Analysis:**

* Inorder to develop the pizza cutting logic, We are using variable r which defines the radius of the pizza, the width and height of the pizza cuts are defined by the variables dx and dy, x and y variables indicates the intersection point of one vertical and one horizontal cut, p variable defines the percentage defining a “too small” piece.
* The “count” variable is used to calculate the number of too small pizzas.
* The array list “areas” is used to find the areas that need to be checked.
* We are using “kx1,kx2,ky1,ky2,llx and lly” variables to find the enclosed rectangle for the pizza.
* The formula used to calculate the value kx1 variable is Math.ceil(1. \* (r - x) / dx), similarly the values are calculated for other variables.
* The number of boxes in the pizza which we consider as cuts is stored in the variables kx and ky.
* We are going to check all rectangles left in the pizza, hence multiple conditions need to be checked.
* In the first condition, we are going to check for the rectangles which don’t have any intersection.
* In the second condition, we are checking rectangle contained in the circle to check if all corners of the rectangle lie within the circle.
* In the third condition, we check circle which contains inside a rectangle.
* Then we find the upper-right, upper-left, lower-left, lower-right quarter intersection areas.
* intersectionAreaOfRectangle() method returns area of interaction of rectangle with upper right portion of circle.
* The formula that calculates the intersection areas is area = intersectionAreaOfRectangle(lx, ly, ux, uy) + intersectionAreaOfRectangle(-ux, ly, -lx, uy) + intersectionAreaOfRectangle(-ux, -uy, -lx, -ly) + intersectionAreaOfRectangle(lx, -uy, ux, -ly).
* trapezoidalArea() method returns the trapezoidal area using the coordinates.
* aSqMBSq() method finds the square of two variables a and band returns the sum value for checking the condition and calculating the area.
* All the areas are stored in the array list.
* We calculate the number of pieces of pizza which are less than p times the largest piece size and store in the count variable.
* The count value is calculated by the formula areas.stream().filter(a -> ((a / maximumarea) - 1.e-6 < p)).count().

**Issues during the implementation:**

* I have faced issues when trying to find the areas of the rectangles which have cut in irregular size.

**Solutions to overcome the Issues:**

* I have researched mathematical geometry concepts – area of finding the curved pieces and found that we can find the area using the points of intersection of the curve.

**Code:**

package com.mycompany.myjavaproject;

import java.util.ArrayList;

import java.util.Scanner;

/\*\*

\*

\* @author sai

\*/

public class PizzaCuttingCodePuzzle {

public static int r, dx, dy, x, y, count; //count variable to calculate number of too small pizzas

public static double p, maximumarea;

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter radius value(r) of pizza");

r = sc.nextInt();

System.out.println("Enter width(dx value) of pizza cuts");

dx = sc.nextInt();

System.out.println("Enter height(dy value) of pizza cuts");

dy = sc.nextInt();

System.out.println("Enter x value of intersection point");

x = sc.nextInt();

System.out.println("Enter y value of intersection point");

y = sc.nextInt();

System.out.println("Enter percentage value(p) defining a too small piece");

p = sc.nextDouble();

maximumarea = 0;

ArrayList<Double> areas = new ArrayList<>();

// finding the enclosing rectangle for the pizza :)

int kx1 = (int) Math.ceil(1. \* (r - x) / dx);

int ky1 = (int) Math.ceil(1. \* (r - y) / dy);

int kx2 = (int) Math.ceil(1. \* (r + x) / dx);

int ky2 = (int) Math.ceil(1. \* (r + y) / dy);

int llx = x - dx \* kx2;

int lly = y - dy \* ky2;

// number of boxes in the pizza which we consider as cuts

int kx = kx1 + kx2;

int ky = ky1 + ky2;

count = 0;

int xi, yi;

// while there are possible intersecting rectangles left on the pizza, continue to find the area

for (yi = 0; yi < ky; yi++) {

for (xi = 0; xi < kx; xi++) {

int lx = llx + xi \* dx, ly = lly + yi \* dy, ux = llx + (xi + 1) \* dx, uy = lly + (yi + 1) \* dy;

// Case 1: no intersection:

if (aSqMBSq(lx, ly) >= r \* r && lx >= 0 && ly >= 0) { // quadrant 1

} else if (aSqMBSq(ux, ly) >= r \* r && ux <= 0 && ly >= 0) { // quadrant 2

} else if (aSqMBSq(ux, uy) >= r \* r && ux <= 0 && uy <= 0) { // quadrant 3

} else if (aSqMBSq(lx, uy) >= r \* r && lx >= 0 && uy <= 0) { // quadrant 4

} else // Case 2: rectangle contained in the circle, checking if all corners of the rectangle lie within the circle:

if (aSqMBSq(ux, uy) <= r \* r && aSqMBSq(ux, ly) <= r \* r && aSqMBSq(lx, ly) <= r \* r && aSqMBSq(lx, uy) <= r \* r) {

maximumarea = dx \* dy;

} else // Case 3: circle contained in the rectangle

if (-r >= lx && r <= ux && -r >= ly && r <= uy) {

System.out.println(0);

System.exit(0);

} else {

//finding upper-right, upper-left, lower-left, lower-right quarter intersection areas

double area = intersectionAreaOfRectangle(lx, ly, ux, uy) + intersectionAreaOfRectangle(-ux, ly, -lx, uy) + intersectionAreaOfRectangle(-ux, -uy, -lx, -ly) + intersectionAreaOfRectangle(lx, -uy, ux, -ly);

maximumarea = Math.max(area, maximumarea);

areas.add(area);

}

}

}

//formula that calculates the number of pieces of pizza which are less than p times the largest piece size

count = (int) areas.stream().filter(a -> ((a / maximumarea) - 1.e-6 < p)).count();

System.out.println("The number of pieces of pizza that are less than p times the size of the largest piece is : " + count);

}

//this method returns the trapezoidal area using the coordinates

public static double trapezoidalArea(double x1, double x2, double ymi) {

double y1 = Math.sqrt(r \* r - x1 \* x1) - ymi;

double y2 = Math.sqrt(r \* r - x2 \* x2) - ymi;

return (y1 + y2) / 2. \* (x2 - x1);

}

//returns area of interaction of rectangle with upper right portion of circle

public static double intersectionAreaOfRectangle(int a, int b, int c, int d) {

if (d < 0 || b > r || c < 0 || a > r) {

return 0;

}

double xmi = Math.max(0, a);

double xma = Math.min(r, c);

double ymi = Math.max(0, b);

double yma = d;

double xmd = xmi;

if (d < r) {

xmd = Math.sqrt(r \* r - yma \* yma);

if (xmd <= xmi) {

xmd = xmi;

} else if (xmd >= xma) {

xmd = xma;

}

}

double tmp = Math.sqrt(r \* r - ymi \* ymi);

if (tmp <= xma && tmp >= xmi) {

xma = tmp;

}

double area = (xmd - xmi) \* (yma - ymi);

double h = xma - xmd;

int k = 1;

double a1 = 0;

double a2 = trapezoidalArea(xmd, xma, ymi);

while (Math.abs(a2 - a1) > 1.e-5) {

k = 2 \* k;

h = h / 2.;

a1 = a2;

a2 = 0;

for (int i = 1; i <= k; i++) {

a2 += trapezoidalArea(xmd + (i - 1) \* h, xmd + i \* h, ymi);

}

}

return area + a2;

}

//finding the square of two variables a and b

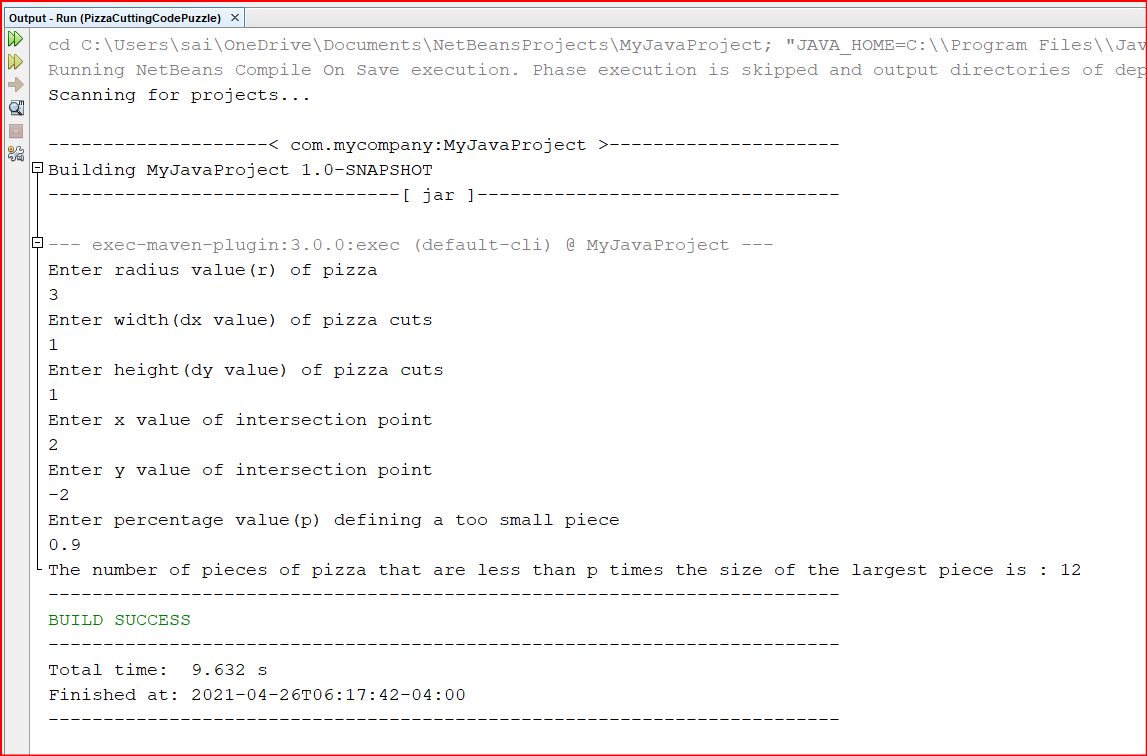
public static int aSqMBSq(int a, int b) {

return (int) (Math.pow(a, 2) + Math.pow(b, 2));

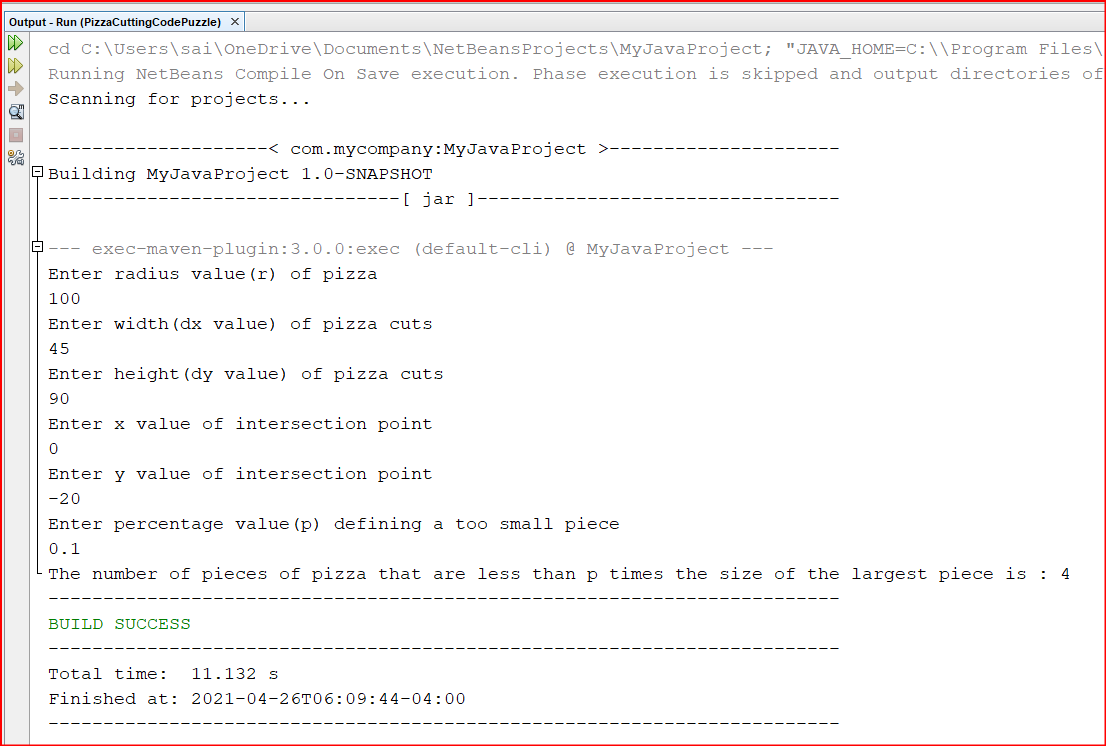
}

}

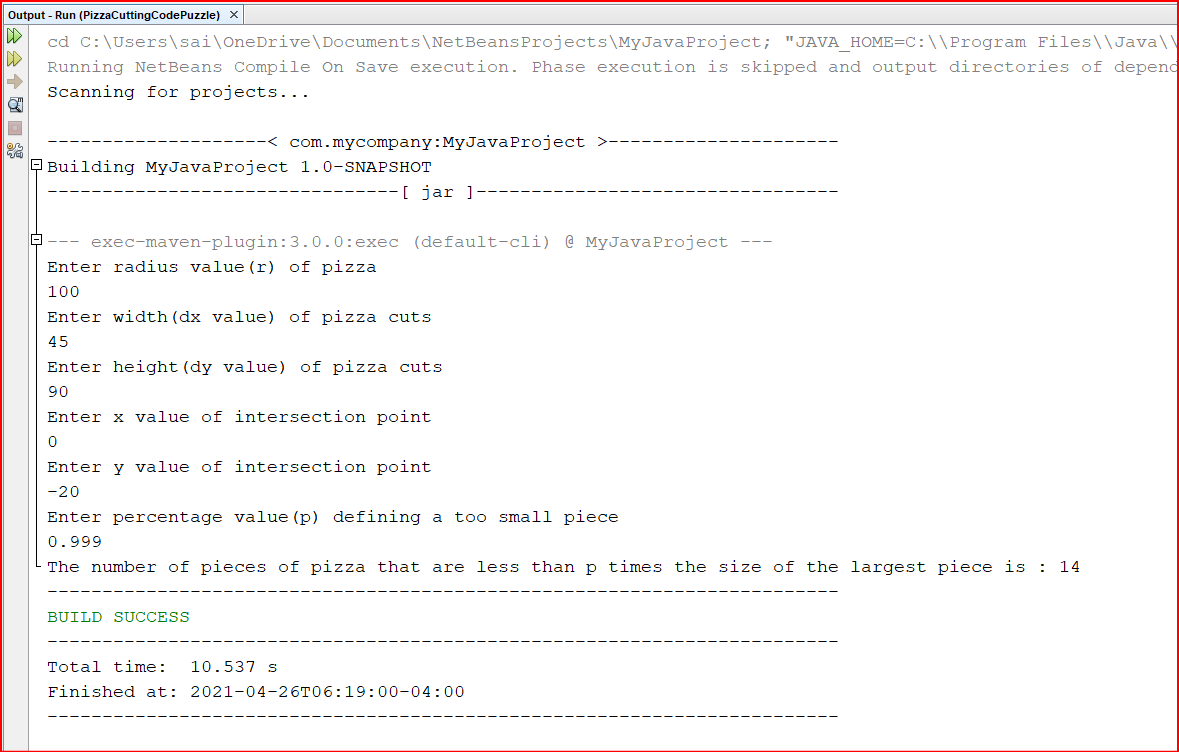
**Sample Output 1:**



**Sample Output 2:**

****

**Sample Output 3:**

****