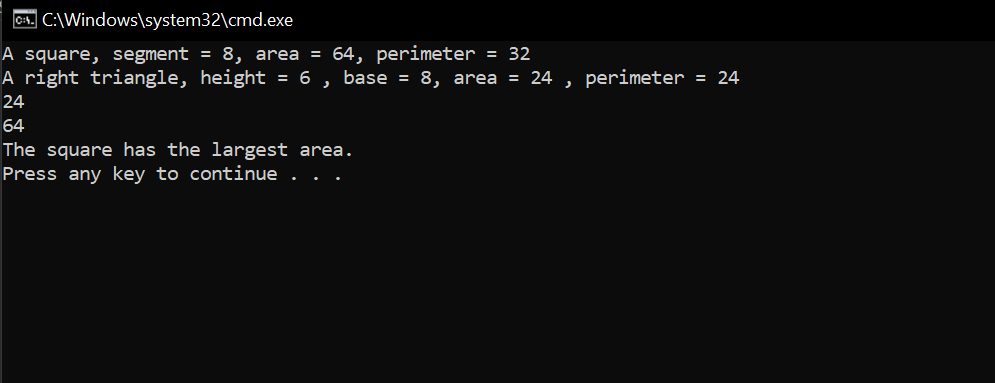
William Tam

5-12-20

ECE 2310

Final



class Program

{

static void Main(string[] args)

{

Square s1 = new Square(8);

Console.WriteLine($"A square, segment = {s1.DisplayLength()}, area = {s1.CalArea()}, perimeter = {s1.CalPerimeter()}");

RightTriangle t1 = new RightTriangle(6, 8);

Console.WriteLine($"A right triangle, height = {t1.DisplayHeight()} , base = {t1.DisplayBase()}, area = {t1.CalArea()} , perimeter = {t1.CalPerimeter()}");

GraphObject[] list = { new RightTriangle(6, 8), new Square(8) };

for( int i = 0; i < 2; i++)

{

Console.WriteLine($"{list[i].CalArea()}");

}

}

}

abstract class GraphObject

{

protected double area;

protected double perimeter;

public GraphObject()

{

area = 0;

perimeter = 0;

}

public GraphObject(double a, double p)

{

area = a;

perimeter = p;

}

public double Area

{

get { return area; }

set { area = value; }

}

public double Perimeter

{

get { return perimeter; }

set { perimeter = value; }

}

public abstract double CalArea();

public abstract double CalPerimeter();

}

class RightTriangle : GraphObject

{

private int height;

private int Base;

public RightTriangle() : base()

{

height = 6;

Base = 8;

}

public RightTriangle(int h, int b)

{

height = h;

Base = b;

}

public int Height

{

get { return height; }

set { height = value; }

}

public int GetBase

{

get { return Base; }

set { Base = value; }

}

public int DisplayHeight()

{

return height;

}

public int DisplayBase()

{

return Base;

}

public override double CalArea()

{

double a = (height \* Base) / 2;

return a;

}

public override double CalPerimeter()

{

double hypotenuse = Math.Sqrt(height \* height + Base \* Base);

double perimeter = hypotenuse + Base + height;

return perimeter;

}

}

class Square : GraphObject

{

private int length;

public Square() : base()

{

length = 8;

}

public Square(int l)

{

length = l;

}

public int Length

{

get { return length; }

set { length = value; }

}

public int DisplayLength()

{

return length;

}

public override double CalArea()

{

double a = length \* length;

return a;

}

public override double CalPerimeter()

{

double p = 4 \* length;

return p;

}

}

static class Utilities

{

public static void FindLargest(GraphObject[] g,int choose, ref int indexLargest, ref double largest)

{

double largestObject = 0;

int largestIndex = 0;

if(choose == 1)

{

largestObject = g[0].CalArea();

largestIndex = 0;

int length = g.Length;

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

{

if(g[i].CalArea() > 1)

{

largestObject = g[i].CalArea();

largestIndex = i;

}

}

}

else

{

largest = g[0].CalPerimeter();

largestIndex = 0;

int length = g.Length;

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

{

if (g[i].CalArea() > 1)

{

largestObject = g[i].CalPerimeter();

largestIndex = 1;

}

}

}

indexLargest = largestIndex;

largest = largestObject;

}

}