Use virtual methods and polymorphism

```
/*
C#: The Complete Reference
by Herbert Schildt

Publisher: Osborne/McGraw-Hill (March 8, 2002)
ISBN: 0072134852
*/

// Use virtual methods and polymorphism.

using System;

class TwoDShape {
double pri_width; // private
double pri_height; // private
string pri_name; // private

// A default constructor.
public TwoDShape() {
width = height = 0.0;
name = “null”;
}

// Parameterized constructor.
public TwoDShape(double w, double h, string n) {
width = w;
height = h;
name = n;
}

// Construct object with equal width and height.
public TwoDShape(double x, string n) {
width = height = x;
name = n;
}

// Construct an object from an object.
public TwoDShape(TwoDShape ob) {
width = ob.width;
height = ob.height;
name = ob.name;
}

// Properties for width, height, and name
public double width {
get { return pri_width; }
set { pri_width = value; }
}

public double height {
get { return pri_height; }
set { pri_height = value; }
}

public string name {
get { return pri_name; }
set { pri_name = value; }
}

public void showDim() {
Console.WriteLine(“Width and height are ” +
width + ” and ” + height);
}

public virtual double area() {
Console.WriteLine(“area() must be overridden”);
return 0.0;
}
}

// A derived class of TwoDShape for triangles.
class Triangle : TwoDShape {
string style; // private
```

```
// A default constructor.
public Triangle() {
style = “null”;
}

// Constructor for Triangle.
public Triangle(string s, double w, double h) :
base(w, h, “triangle”) {
style = s;
}

// Construct an isosceles triangle.
public Triangle(double x) : base(x, “triangle”) {
style = “isosceles”;
}

// Construct an object from an object.
public Triangle(Triangle ob) : base(ob) {
style = ob.style;
}

// Override area() for Triangle.
public override double area() {
return width * height / 2;
}

// Display a triangle's style.
public void showStyle() {
Console.WriteLine(“Triangle is ” + style);
}
}

// A derived class of TwoDShape for rectangles.
class Rectangle : TwoDShape {
// Constructor for Rectangle.
public Rectangle(double w, double h) :
base(w, h, “rectangle”){ }

// Construct a square.
```

```
public Rectangle(double x) :
base(x, “rectangle”) { }

// Construct an object from an object.
public Rectangle(Rectangle ob) : base(ob) { }

// Return true if the rectangle is square.
public bool isSquare() {
if(width == height) return true;
return false;
}

// Override area() for Rectangle.
public override double area() {
return width * height;
}
}

public class DynShapes {
public static void Main() {
TwoDShape[] shapes = new TwoDShape[5];

shapes[0] = new Triangle(“right”, 8.0, 12.0);
shapes[1] = new Rectangle(10);
shapes[2] = new Rectangle(10, 4);
shapes[3] = new Triangle(7.0);
shapes[4] = new TwoDShape(10, 20, “generic”);

for(int i=0; i < shapes.Length; i++) {
Console.WriteLine("object is " + shapes[i].name);
Console.WriteLine("Area is " + shapes[i].area());
Console.WriteLine(); } } } [/csharp]
```