/////Program1//////

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace CSharp220716\_E5

{

//A class for two dimensional shape

class TwoDShape

{

double pri\_Width;

double pri\_Height;

string name;

//Default Constructor

public TwoDShape()

{

Width = Height = 0.0;

name = "null";

}

//Constructor for TwoDShape

public TwoDShape(double w, double h,string n)

{

Width = w;

Height = h;

name = n;

}

//Constructor for equal Width and Height

public TwoDShape(double x,string n)

{

Width = Height = x;

name = n;

}

//Construct a copy of TwoDShape object

public TwoDShape(TwoDShape ob)

{

Width = ob.Width;

Height = ob.Height;

name = ob.name;

}

//properties for Width

public double Width

{

get

{

return pri\_Width;

}

set

{

pri\_Width = value < 0 ? -value : value;

}

}

//properties for Height

public double Height

{

get

{

return pri\_Height;

}

set

{

pri\_Height = value < 0 ? -value : value;

}

}

//properties for name

public string Name

{

get

{

return name;

}

set

{

name = value;

}

}

public void ShowDim()

{

Console.WriteLine("Width and Height are " +

Width + " and " + Height);

}

public virtual double Area()

{

Console.WriteLine("Area() must be overriden");

return 0.0;

}

}

class Triangle : TwoDShape

{

string Style; //private

/\* A default constructor. This automatically

invokes the default constructor of TwoDShape \*/

public Triangle()

{

Style = "null"; //init the derived class

}

//Constructor

public Triangle(string s, double w, double h) : base(w, h, "triangle")

{

Style = s; //init the derived class

}

//Constructor

public Triangle(double x) : base(x, "triangle")

{

Style = "isosceles"; //init the derived class

}

//Constructor

public Triangle(Triangle ob) : base(ob)

{

Style = ob.Style; //init the derived class

}

//override Area()

public override double Area()

{

return Width \* Height / 2;

}

//Display a Triangle Style

public void ShowStyle()

{

Console.WriteLine("Triangle is " + Style);

}

}

class Rectangle : TwoDShape

{

string Style; //private

/\* A default constructor. This automatically

invokes the default constructor of TwoDShape \*/

public Rectangle()

{

Style = "null"; //init the derived class

}

//Constructor

public Rectangle(string s, double w, double h) : base(w, h, "rectangle")

{

Style = s; //init the derived class

}

//Constructor

public Rectangle(double x,double y) : base(x,y, "rectangle")

{

Style = "isosceles"; //init the derived class

}

//Constructor

public Rectangle(double x) : base(x, "rectangle")

{

Style = "isosceles"; //init the derived class

}

//Constructor

public Rectangle(Rectangle ob) : base(ob)

{

Style = ob.Style;

}

//Constructor

public bool IsSquare()

{

if(Width == Height)

{

return true;

}

else

{

return false;

}

}

//override Area()

public override double Area()

{

return Width \* Height / 2;

}

}

class Shapes

{

static void Main(string[] args)

{

TwoDShape []shapes = new TwoDShape[5];

shapes[0] = new Triangle("right", 8.0, 12.0);

shapes[1] = new Rectangle(10.0);

shapes[2] = new Rectangle(10.0,4.0);

shapes[3] = new Triangle(9.0);

shapes[4] = new TwoDShape( 8.0, 12.0, "generic");

for(int i =0;i < shapes.Length; i++)

{

Console.WriteLine("object is " + shapes[i].Name);

shapes[i].ShowDim();

Console.WriteLine("Area is " + shapes[i].Area());

Console.WriteLine();

}

Console.ReadLine();

}

}

}

//////////////Program2//////////

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace CSharp230716\_E1

{

//A class for two dimensional shape

abstract class TwoDShape

{

double pri\_Width;

double pri\_Height;

string name;

//Default Constructor

public TwoDShape()

{

Width = Height = 0.0;

name = "null";

}

//Constructor for TwoDShape

public TwoDShape(double w, double h,string n)

{

Width = w;

Height = h;

name = n;

}

//Constructor for equal Width and Height

public TwoDShape(double x,string n)

{

Width = Height = x;

name = n;

}

//Construct a copy of TwoDShape object

public TwoDShape(TwoDShape ob)

{

Width = ob.Width;

Height = ob.Height;

name = ob.name;

}

//properties for Width

public double Width

{

get

{

return pri\_Width;

}

set

{

pri\_Width = value < 0 ? -value : value;

}

}

//properties for Height

public double Height

{

get

{

return pri\_Height;

}

set

{

pri\_Height = value < 0 ? -value : value;

}

}

//properties for name

public string Name

{

get

{

return name;

}

set

{

name = value;

}

}

public void ShowDim()

{

Console.WriteLine("Width and Height are " +

Width + " and " + Height);

}

//now area is abstract

public abstract double Area();

}

class Triangle : TwoDShape

{

string Style; //private

/\* A default constructor. This automatically

invokes the default constructor of TwoDShape \*/

public Triangle()

{

Style = "null"; //init the derived class

}

//Constructor

public Triangle(string s, double w, double h) : base(w, h, "triangle")

{

Style = s; //init the derived class

}

//Constructor

public Triangle(double x) : base(x, "triangle")

{

Style = "isosceles"; //init the derived class

}

//Constructor

public Triangle(Triangle ob) : base(ob)

{

Style = ob.Style; //init the derived class

}

//override Area()

public override double Area()

{

return Width \* Height / 2;

}

//Display a Triangle Style

public void ShowStyle()

{

Console.WriteLine("Triangle is " + Style);

}

}

class Rectangle : TwoDShape

{

string Style; //private

/\* A default constructor. This automatically

invokes the default constructor of TwoDShape \*/

public Rectangle()

{

Style = "null"; //init the derived class

}

//Constructor

public Rectangle(string s, double w, double h) : base(w, h, "rectangle")

{

Style = s; //init the derived class

}

//Constructor

public Rectangle(double x,double y) : base(x,y, "rectangle")

{

Style = "isosceles"; //init the derived class

}

//Constructor

public Rectangle(double x) : base(x, "rectangle")

{

Style = "isosceles"; //init the derived class

}

//Constructor

public Rectangle(Rectangle ob) : base(ob)

{

Style = ob.Style;

}

//Constructor

public bool IsSquare()

{

if(Width == Height)

{

return true;

}

else

{

return false;

}

}

//override Area()

public override double Area()

{

return Width \* Height / 2;

}

}

class Shapes

{

static void Main(string[] args)

{

TwoDShape []shapes = new TwoDShape[4];

shapes[0] = new Triangle("right", 8.0, 12.0);

shapes[1] = new Rectangle(10.0);

shapes[2] = new Rectangle(10.0,4.0);

shapes[3] = new Triangle(9.0);

for(int i =0;i < shapes.Length; i++)

{

Console.WriteLine("object is " + shapes[i].Name);

shapes[i].ShowDim();

Console.WriteLine("Area is " + shapes[i].Area());

Console.WriteLine();

}

Console.ReadLine();

}

}

}