```
using static System.Console;//needed for using WriteLine
abstract class Shape
{
    //is needed so that we can switch on objects of type Shape
    //classes are used to express "is a" type of relationship
    //interfaces are used to express "can be used as" type of relationship
}
class Rectangle:Shape//rectangle is a kind of shape, so Shape is the parent class
{
    private double width, height;//instance variables
    public Rectangle(double w, double h)
    {
        width = w;height = h;//set values of instance variables
    }
    //properties can be expression-bodied members
    public double Width { get => width; }//Width property gets the value of the width backing field
    public double Height { get => height; }//Height property gets the value of the height backing field
}
class Circle: Shape//circles are kinds of shapes, so Shape is the parent class
{
    private double radius;//radius is the distance from center to a point on the circumference
    public Circle(double r)=>radius=r;//set value of radius instance variable
    public double Radius { get => radius; }//Get value of radius field through Radius property
}
class Program
{
    static void Main()
    {
        Shape shape = new Rectangle(10,5);
        switch(shape)//it's possible to switch on different data types
        {
            //if a rectnagle has width=height, it's really a square
            case Rectangle r when (r.Width == r.Height)://check whether it's a square
                WriteLine($"Perimeter={4 * r.Width}: Square");
                break;
            case Rectangle r://when width and height are different, P=2width+2height
                WriteLine($"Perimeter={2 * r.Width + 2 * r.Height}:Rectangle");
                break;
            case Circle c://circumference of circle is 2 * pi * radius
                WriteLine($"Circumference:{2 * 3.1459 * c.Radius}:Circle");
                break;
        }
    }
}
//for a square, 2width+2height is the same as 4width or 4height, since height and width are equal
```