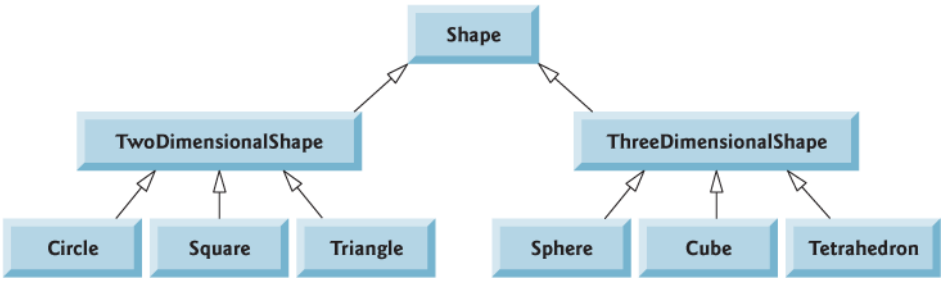

1) **(`Shape` Hierarchy)** Implement the `Shape` hierarchy of the figure.
2) Omit the `Triangle` and `Tetrahedron` classes.
3) The `Shape class` should contain read-only `int` properties `X` and `Y` that indicate the position of the `Shape` on the plane. The `class` has only one constructor that can initialize the values of `X` and `Y`.
4) The `Shape class` should contain a read-only `virtual` property `Name` that returns a `string`, indicating the `Name` of the Shape. This property, by default, throws a `NotImplementedException.`
5) Each `TwoDimensionalShape` should contain read-only `virtual` property `Area` to calculate the area of the two-dimensional shape. In addition, this `class` contains two `protected int` **fields** called `Dimension1` and `Dimension2` that get initialized in the constructor. (so there is only one constructor that takes `x`, `y`, `dimension1`, and `dimension2` as the arguments). This property, by default, throws a `NotImplementedException.`
6) Each `ThreeDimensionalShape` should have read-only `virtual` properties `Area` and `Volume` to calculate the surface area and volume, respectively, of the three-dimensional shape. These properties, by default, throw `NotImplementedException`.
7) `ThreeDimensionalShape class` contains three `protected int` **fields** called `Dimension1`, `Dimension2`, and `Dimension3` that get initialized in the constructor (so there is only one constructor takes `x`, `y`, `dimension1`, `dimension2`, and `dimension3` as the arguments).
8) In addition to implementing all the `virtual` members, the following derived classes have other properties:
   a) The `Circle` class includes a public `Radius` property that assigns the value of the radius to both `Dimension1` and `Dimension2`.
   b) The `Square class` includes a public `Side` property that assigns the value of the side to both `Dimension1` and `Dimension2`.
   c) The `Sphere` class includes a public `Radius` property that assigns the value of the radius to `Dimension1`, `Dimension2`, and `Dimension3`.
   d) The `Cube` class includes a public `Side` property that assigns the value of the side to both `Dimension1`, `Dimension2`, and `Dimension3`.
9) The following describes the overrides for the `ToString()` method:
   a) The `Shape` class's `ToString()` method returns the value of `X` and the value of `Y`, separated by comma and enclosed in parenthesis. (e.g., `(1, 2)` if X = 1 and Y = 2)

b) Both `Circle` and `Sphere` classes's `ToString()` methods return the `Shape ToString()`, followed by the word "radius: " and then the value of the `Radius`. For example, if X = 1, Y = 2, and Radius = 3, you will see: `(1, 2) radius: 3`

c) Both `Square` and `Cube` classes's `ToString()` methods return the `Shape ToString()`, followed by the word "side: " and then the value of the `Side`. For example, if X = 1, Y = 2, and Side = 4, you will see: `(1, 2) side: 4`

In the `Shape` class, create a class method named `Compute` that takes an array of `Shape` references to objects of each concrete class in the hierarchy. In the loop that processes all the shapes in the array, first display the `Name` on one line. Next, display the `ToString()` of that shape on the following line. Finally, determine whether each shape is a `TwoDimensionalShape` or a `ThreeDimensionalShape`. If a shape is a `TwoDimensionalShape`, display its `Area` on a separate line. Otherwise, if a shape is a `ThreeDimensionalShape`, display its `Area` on one line, followed by the `Volume` on the next line. Show up to 10 digits (use the `"G10"` string formatter).