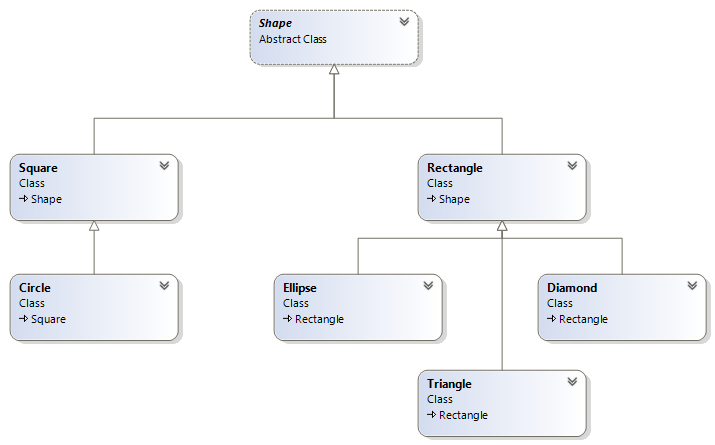
In this exercise you will be using inheritance to promote code re-use.

There are three ways of building classes: By building everything, by using an existing class and add to it and by composing with other classes

# You must follow the specifications exactly (To demo to instructor at the end of this class)

Because the required code statements for each type is very few, you may choose to implement all the classes in a single code file.



There are seven classes participating in this application as shown in the diagram above. Each of the seven classes is fully described below.

You will need to “google” how to declare an abstract property.

It is possible to override a member in the base class that is decorated with the virtual, abstract or override keyword

# The Shape class

This abstract class comprises of four members. This is abstract because there is an abstract member (property Area). There is no formulae to compute the area of a shape object.

|  |
| --- |
| **Shape**  **Abstract Class** |
| **Properties** |
| **+ <<C# property set absent>>Name : string**  **+ <<C# property abstract get>>Area : double** |
| **Methods** |
| **+ <<constructor>> Shape(string name)**  **+ ToString() : string** |

#### Description of class members

##### Fields:

There are no fields

##### Properties:

There are two properties in which one of them is auto-implemented.

**Name** – this string property represents the name of this object. The getter is public and the setter is private.

**Area** – this property represents the area of this shape. This computed property is an abstract property that is implemented in its derived classes. This is abstract because this property does not make sense (what is the expression for the area of a shape object) in this class, only in its derived classes. There is no set accessor.

Some decorators for classes are

**abstract** it cannot be instantiated

**sealed** it cannot be the parent of any class

**static** all members are static, cannot be instantiated

**partial** implementation spans multiple files

##### Constructor:

**public Shape(string name)** – This constructor takes a string parameter and assigns it to the appropriate property.

##### Methods:

**public override string ToString()** – This is a public method overrides the corresponding method in the object class to return the Name and the Area of this object. You get to decide how the properties will be display to the user.

Not all base classes are abstract

**Object** is the parent class of all classes and it is not abstract.

**Square** is the parent class of **Circle** and it is not abstract.

Similarly **Rectangle** is the parent class of **Ellipse**, **Triangle** and **Diamond** and it is not declared abstract.

# The Square class

This class inherits from the Shape class and it supplies four members of its own: one auto-implemented and one computed property and two constructors.

|  |
| --- |
| **Square**  **Class**  **Shape** |
| **Properties** |
| **# «property get, set absent» Length : double**  **+ «property get» Area : double** |
| **Methods** |
| **+ «constructor» Square(double length)**  **+ «constructor» Square(string name, double length)** |

#### Description of class members

##### Fields:

There are no fields

##### Properties:

There are two properties in which one of them is auto-implemented and the other is computed.

**Length** – this double property represents the length of this object. Both the getter and the setter are protected.

**Area** – this property represents the area of this shape. This property overrides the represented member in the Shape class. It returns the square of the length.

##### Constructor:

**public Square(double length)** – This constructor takes one parameter, it invokes the base constructor with the string "square" and assigns the argument to the appropriate property. This is the primary constructor that is used in object instantiation.

**public Square(string name, double length)** – This constructor takes two parameters; it invokes the base constructor with the first one and assigns the second one to the appropriate property. This constructor is used by derived class.

##### Methods:

There are no methods

# The Circle class

This class inherits from the Square class and it comprises of two members: a computed property that override the same one in the parent class and a constructor.

|  |
| --- |
| **Circle**  **Class**  **Square** |
| **Properties** |
| **+ «property get» Area : double** |
| **Methods** |
| **+ «constructor» Circle(double length)** |

#### Description of class members

##### Fields:

There are no fields

##### Properties:

There is a calculated property.

**Area** – this property overrides the represented member in the parent class. represents the area of this shape. The getter returns **π \* Length \* Length**.

##### Constructor:

**public Circle(double length)** – This constructor takes one parameter. It invokes the base constructor with the string "circle" and the length.

##### Methods:

There are no methods

# The Rectangle class

This class inherits from the Shape class and it comprises of three members:

|  |
| --- |
| **Rectangle**  **Class**  **Shape** |
| **Properties** |
| **# «property get, set» Width : double**  **# «property get, set» Height : double**  **+ «property get» Area : double** |
| **Methods** |
| **+ «constructor» Rectangle(double height, double width)**  **+ «constructor» Rectangle(string name, double height, double width)** |

#### Description of class members

##### Fields:

There are no fields

##### Properties:

There are three properties.

**Width** – this double property represents the width of this object. Both the getter and the setter are protected.

**Height** – this double property represents the height of this object. Both the getter and the setter are protected.

**Area** – this property represents the area of this shape. It overrides the represented member in the parent class. The getter returns **Width** **\* Height**.

##### Constructor:

**public Rectangle(double length, double width)** – This constructor takes two parameters. It invokes the base constructor with the string "rectangle" and assigns the last two to the appropriate properties. This is the primary constructor used in class instantiation.

**public Rectangle(string name, double length, double width)** – This constructor takes three parameters, it invokes the base constructor with the first one the assigns the last two to the appropriate properties. This constructor is used by derived class.

##### Methods:

There are no methods

# The Ellipse class

This class inherits from the Rectangle class and it comprises of two members:

|  |
| --- |
| **Ellipse**  **Class**  **Rectangle** |
| **Properties** |
| **+ «property get» Area : double** |
| **Methods** |
| **+ «constructor» Ellipse(double height, double width)** |

#### Description of class members

##### Fields:

There are no fields

##### Properties:

There is only one calculated property.

**Area** – this property represents the area of this shape. It overrides the represented member in the parent class. The getter returns **π \* Width \* Height**.

##### Constructor:

**public Ellipse(double height, double width)** – This constructor takes two parameters, it invokes the base constructor with the string "ellipse" and the two arguments. There is no code in the constructor.

##### Methods:

There are no methods

# The Triangle class

This class inherits from the Rectangle class and it comprises of two members:

|  |
| --- |
| **Triangle**  **Class**  **Rectangle** |
| **Properties** |
| **+ «property get» Area : double** |
| **Methods** |
| **+ «constructor» Triangle(double height, double width)** |

#### Description of class members

##### Fields:

There are no fields

##### Properties:

There is only one calculated property.

**Area** – this property represents the area of this shape. The getter returns **Width \* Height \* 0.5**.

##### Constructor:

**public Triangle(double length, double height)** – This constructor takes two parameters, it invokes the base constructor with the string "triangle" and the two arguments. There is no code in the constructor.

##### Methods:

There are no methods

# The Diamond class

This class inherits from the Rectangle class and it comprises of two members:

|  |
| --- |
| **Diamond**  **Class**  **Rectangle** |
| **Properties** |
| **+ «property get» Area : double** |
| **Methods** |
| **+ «constructor» Diamond(double height, double width)** |

#### Description of class members

##### Fields:

There are no fields

##### Properties:

There is only one calculated property.

**Area** – this property represents the area of this shape. It overrides the represented member in the parent class. The getter returns **Width \* Height**.

##### Constructor:

**public Diamond(double height, double width)** – This constructor takes two parameters, it invokes the base constructor with the string diamond and the two arguments. This is no code in the constructor.

##### Methods:

There are no methods

### Test Harness

Copy the following lines to the Main method in the Program.cs file.

//although Shape is an abstract is can be used as a reference type

//any child class of Shape is also a Shape

List<Shape> shapes = new List<Shape>();

shapes.Add(new Square(2));

shapes.Add(new Rectangle(2, 3));

shapes.Add(new Circle(2));

shapes.Add(new Triangle(4, 6));

shapes.Add(new Ellipse(2, 3));

shapes.Add(new Diamond(2, 3));

shapes.Add(new Square(5));

shapes.Add(new Rectangle(5, 4));

shapes.Add(new Circle(1));

shapes.Add(new Triangle(7, 8));

foreach (var s in shapes)

{

Console.WriteLine(s);

}