# Abstract Class examples

**Q1)** Create classes to represent different animals. You can create a generic, abstract **class named Animal** so you can provide generic data fields, such as the animal's name, only once.

An Animal is generic, but each specific Animal, such as **Dog** or **Cat**,makes a unique sound.

If you code **an abstract Speak()** method in the abstract Animal class, then you require all future Animal derived classes to override the Speak() method and provide an implementation that is specific to the derived class.

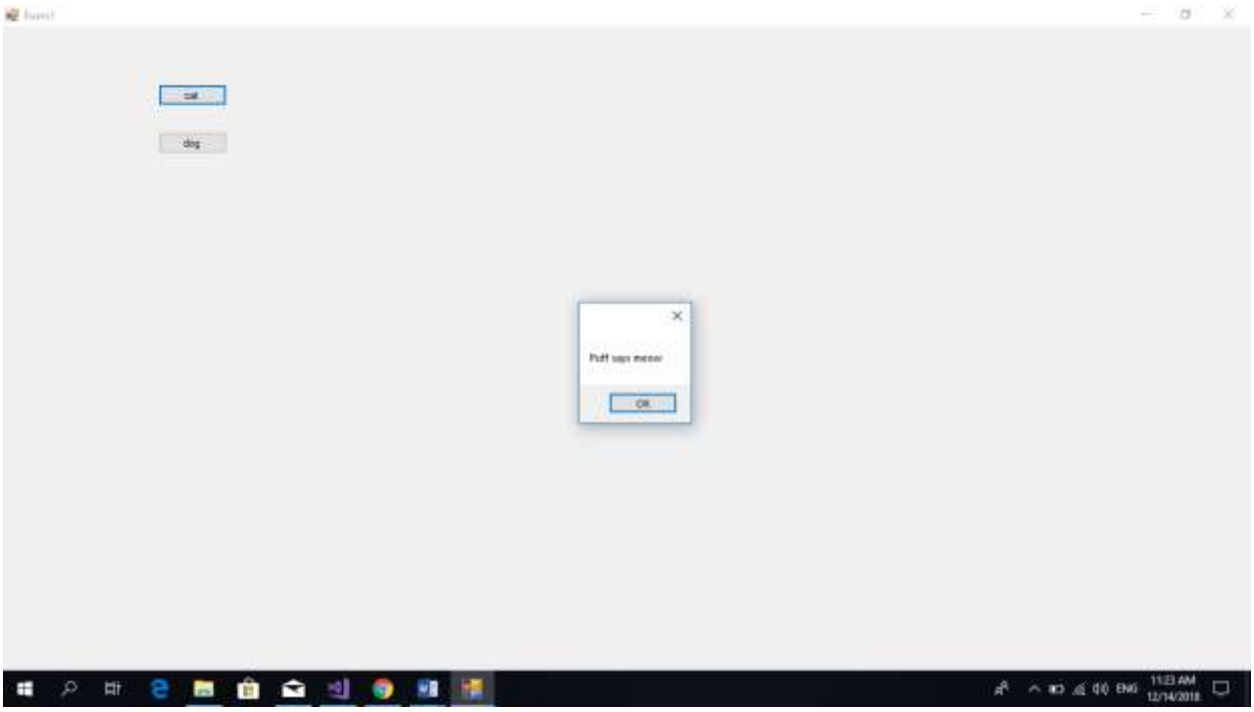
```
using System;
using System.Collections.Generic;
using System.Linq;

using System.Text;
using System.Threading.Tasks;

namespace example_of_abstract
{

    abstract class Animal
    {

        private string name;
        public Animal(string name)
        {
            this.name = name;
        }

        public string Name {
            get { return name; } }
        public abstract string Speak();

    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace example_of_abstract
{
    class Dog : Animal
    {
        public Dog(string name) : base(name)
        {
        }
        public override string Speak()
        {
            return "woof";

        }
    }

}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace example_of_abstract
{
    class Cat    : Animal    {
        public Cat(string name) : base(name)
        {
        }
        public override string Speak()
        {
            return "meow";

        }
    }
}
```

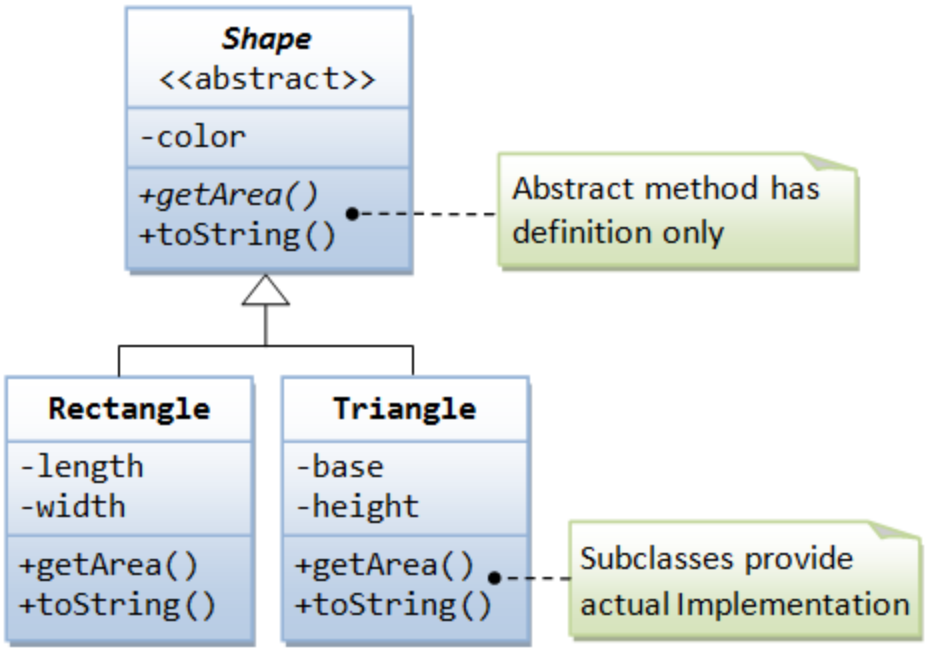
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace example_of_abstract
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {


        }

        private void button2_Click(object sender, EventArgs e)
        {
            Dog spot = new Dog("Spot");
            MessageBox.Show(spot.Name + " says " + spot.Speak());

        }

        private void button1_Click(object sender, EventArgs e)
        {
            Cat puff = new Cat("Puff");

            MessageBox.Show(puff.Name + " says " + puff.Speak());
        }
    }
}
```

Output


cat
dog
Puff says meow
OK

# Abstract Class examples

**Q2)** Convert the following UML class diagram to C# program, test the program .




```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApp6
{
    abstract class Shape {
        private string color;
        public Shape(string color)  {  this.color = color;  }
        public abstract double getArea();
        public virtual string toString() { return "im in shape class"; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApp6
{
    class Rectangle : Shape
    {
        private int length, width;
        public Rectangle(string color, int length, int width):base(color){
            this.length=length;
            this.width=width;
            }
        public override double getArea() { return length * width; }
        public override string toString() { return "im in rectangle class"; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApp6
{
    class Triangle: Shape
    {
        private int Base, height;
        public Triangle(string color, int Base, int height) : base(color)
        {
            this.Base = Base;
            this.height = height;
        }
        public override double getArea() { return 0.5*Base * height; }
        public override string toString() { return "im in Triangle class"; }

    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp8
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }

        private void Form1_Load(object sender, EventArgs e)
        {
       textBox1.Text = "im in shape class its bstract i cannot create object of it ";
            Rectangle r = new Rectangle("red", 7, 6);
            textBox2.Text = r.toString() + Environment.NewLine +
               "Area = "+ r.getArea().ToString();
           Triangle t = new Triangle("blue", 9, 6);
            textBox3.Text = t.toString() + Environment.NewLine +
               "Area = " + t.getArea().ToString();
        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {

        }
    }
}
```


Form1
abstract class shape
class rectangle
Area = 42
class triangle
Area = 27