

C# - Inheritance

One of the most important concepts in object-oriented programming is inheritance. Inheritance allows us to define a class in terms of another class, which makes it easier to create and maintain an application. This also provides an opportunity to reuse the code functionality and speeds up implementation time.

When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an existing class. This existing class is called the **base** class, and the new class is referred to as the **derived** class.

The idea of inheritance implements the **IS-A** relationship. For example, mammal **IS A** animal, dog **IS-A** mammal hence dog **IS-A** animal as well, and so on.

Base and Derived Classes

A class can be derived from more than one class or interface, which means that it can inherit data and functions from multiple base classes or interfaces.

The syntax used in C# for creating derived classes is as follows –

```
<access-specifier> class <base_class> {  
    ...  
}  
  
class <derived_class> : <base_class> {  
    ...  
}
```

Consider a base class Shape and its derived class Rectangle –

```
using System;  
  
namespace InheritanceApplication {  
    class Shape {  
        public void setWidth(int w) {  
            width = w;  
        }  
        public void setHeight(int h) {  
            height = h;  
        }  
        protected int width;  
        protected int height;  
    }  
  
    // Derived class  
    class Rectangle: Shape {  
        public int getArea() {  
            return (width * height);  
        }  
    }  
}
```

[Live Demo](#)

```

    }
}
class RectangleTester {
    static void Main(string[] args) {
        Rectangle Rect = new Rectangle();

        Rect.setWidth(5);
        Rect.setHeight(7);

        // Print the area of the object.
        Console.WriteLine("Total area: {0}", Rect.getArea());
        Console.ReadKey();
    }
}
}

```

When the above code is compiled and executed, it produces the following result –

```
Total area: 35
```

Initializing Base Class

The derived class inherits the base class member variables and member methods. Therefore the super class object should be created before the subclass is created. You can give instructions for superclass initialization in the member initialization list.

The following program demonstrates this –

```

using System;

namespace RectangleApplication {
    class Rectangle {

        //member variables
        protected double length;
        protected double width;

        public Rectangle(double l, double w) {
            length = l;
            width = w;
        }
        public double GetArea() {
            return length * width;
        }
        public void Display() {
            Console.WriteLine("Length: {0}", length);
            Console.WriteLine("Width: {0}", width);
            Console.WriteLine("Area: {0}", GetArea());
        }
    }
} //end class Rectangle
class Tabletop : Rectangle {
    private double cost;

```

Live Demo

```

    public Tabletop(double l, double w) : base(l, w) { }

    public double GetCost() {
        double cost;
        cost = GetArea() * 70;
        return cost;
    }
    public void Display() {
        base.Display();
        Console.WriteLine("Cost: {0}", GetCost());
    }
}
class ExecuteRectangle {
    static void Main(string[] args) {
        Tabletop t = new Tabletop(4.5, 7.5);
        t.Display();
        Console.ReadLine();
    }
}
}

```

When the above code is compiled and executed, it produces the following result –

```

Length: 4.5
Width: 7.5
Area: 33.75
Cost: 2362.5

```

Multiple Inheritance in C#

C# does not support multiple inheritance. However, you can use interfaces to implement multiple inheritance. The following program demonstrates this –

```

using System;

namespace InheritanceApplication {
    class Shape {
        public void setWidth(int w) {
            width = w;
        }
        public void setHeight(int h) {
            height = h;
        }
        protected int width;
        protected int height;
    }

    // Base class PaintCost
    public interface PaintCost {
        int getCost(int area);
    }
}

```

Live Demo

```

// Derived class
class Rectangle : Shape, PaintCost {
    public int getArea() {
        return (width * height);
    }
    public int getCost(int area) {
        return area * 70;
    }
}

class RectangleTester {
    static void Main(string[] args) {
        Rectangle Rect = new Rectangle();
        int area;

        Rect.setWidth(5);
        Rect.setHeight(7);
        area = Rect.getArea();

        // Print the area of the object.
        Console.WriteLine("Total area: {0}", Rect.getArea());
        Console.WriteLine("Total paint cost: ${0}" , Rect.getCost(area));
        Console.ReadKey();
    }
}

```

When the above code is compiled and executed, it produces the following result –

```

Total area: 35
Total paint cost: $2450

```