C# Properties

C# Properites doesn't have storage location. C# Properites are extension of fields and accessed like fields.

The Properties have accessors that are used to set, get or compute their values.

Usage of C# Properties

1. C# Properties can be read-only or write-only.
2. We can have logic while setting values in the C# Properties.
3. We make fields of the class private, so that fields can't be accessed from outside the class directly. Now we are forced to use C# properties for setting or getting values.

C# Properties Example

1. **using** System;
2. **public** **class** Employee
3. {
4. **private** **string** name;
6. **public** **string** Name
7. {
8. **get**
9. {
10. **return** name;
11. }
12. **set**
13. {
14. name = value;
15. }
16. }
17. }
18. **class** TestEmployee{
19. **public** **static** **void** Main(**string**[] args)
20. {
21. Employee e1 = **new** Employee();
22. e1.Name = "Sonoo Jaiswal";
23. Console.WriteLine("Employee Name: " + e1.Name);
25. }
26. }

Output:

Employee Name: Sonoo Jaiswal

C# Properties Example 2: having logic while setting value

Output:

Exception Handling in Java - Javatpoint

Employee Name: Sonoo JavaTpoint

C# Properties Example 3: read-only property

**using** System;

**public** **class** Employee

    {

**private** **static** **int** counter;

**public** Employee()

        {

            counter++;

        }

**public** **static** **int** Counter

        {

**get**

            {

**return** counter;

            }

         }

   }

**class** TestEmployee{

**public** **static** **void** Main(**string**[] args)

        {

            Employee e1 = **new** Employee();

            Employee e2 = **new** Employee();

            Employee e3 = **new** Employee();

            //e1.Counter = 10;//Compile Time Error: Can't set value

            Console.WriteLine("No. of Employees: " + Employee.Counter);

        }

    }

Output:

No. of Employees: 3

Properties & Inheritance

The properties of a Base class can be inherited to a Derived class.

1. //C# : Property : Inheritance
2. //Author: rajeshvs@msn.com

**using** System;

**class** Base

{

**public** **int** X

    {

**get**

        {

            Console.Write("Base GET");

**return** 10;

        }

**set**

        {

            Console.Write("Base SET");

        }

    }

}

**class** Derived : Base

{

}

**class** MyClient

{

**public** **static** **void** Main()

    {

        Derived d1 = **new** Derived();

        d1.X = 10;

        Console.WriteLine(d1.X);

    }

}

Properties & Polymorphism

A Base class property can be polymorphically overridden in a Derived class. But remember that the modifiers like virtual, override etc are using at property level, not at accessor level.

1. //C# : Property : Polymorphism
2. //Author: rajeshvs@msn.com

**using** System;

**class** Base

{

**public** **virtual** **int** X

    {

**get**

        {

            Console.Write("Base GET");

**return** 10;

        }

**set**

        {

            Console.Write("Base SET");

        }

    }

}

**class** Derived : Base

{

**public** **override** **int** X

    {

**get**

        {

            Console.Write("Derived GET");

**return** 10;

        }

**set**

        {

            Console.Write("Derived SET");

        }

    }

}

**class** MyClient

{

**public** **static** **void** Main()

    {

        Base b1 = **new** Derived();

        b1.X = 10;

        Console.WriteLine(b1.X);

    }

}