**Properties:**

* Another type of class member is the *property.*
* As a general rule, a property combines a field with the methods that access it.
* Properties are similar to indexers.
* A property consists of a name along with **get** and **set** accessors.
* The accessors are used to get and set the value of a variable.
* The key benefit of a property is that its name can be used in expressions and assignments like a normal variable, but in actuality the **get** and **set** accessors are automatically invoked.
* This is similar to the way that an indexer’s **get** and **set** accessors are automatically used.
* The general form of a property is shown here:

***type name***

{

**get**

{

// get accessor code

}

**set**

{

// set accessor code

}

}

* Here, *type* specifies the type of the property, such as **int**, and *name* is the name of the property.
* Once the property has been defined, any use of *name* results in a call to its appropriate accessor.
* The **set** accessor automatically receives a parameter called **value** that contains the
* value being assigned to the property.

**Example:**

// A simple property example.

using System;

class SimpProp

{

int prop; // field being managed by MyProp

public SimpProp()

{

prop = 0;

}

// This is the property that supports access to the private instance variable prop. It allows only positive values.

public int MyProp

{

get

{

return prop;

}

set

{

if(value >= 0)

prop = value;

}

}

}

// Demonstrate a property.

class PropertyDemo

{

public static void Main()

{

SimpProp ob = new SimpProp();

Console.WriteLine("Original value of ob.MyProp: " + ob.MyProp);

ob.MyProp = 100; // assign value

Console.WriteLine("Value of ob.MyProp: " + ob.MyProp);

// Can't assign negative value to prop.

Console.WriteLine("Attempting to assign -10 to ob.MyProp");

ob.MyProp = -10;

Console.WriteLine("Value of ob.MyProp: " + ob.MyProp);

}

}