

# System.ICloneable Interface

```
[ILAsm]  
.class interface public abstract ICloneable  
  
[C#]  
public interface ICloneable
```

## Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- *Version:* 2.0.x.x
- *Attributes:*
  - CLSCompliantAttribute(true)

## Summary

Implemented by classes that require control over the way in which copies of instances are constructed.

**Library:** BCL

## Description

[*Note:* System.ICloneable contains the System.ICloneable.Clone method. The consumer of an object should call this method when a copy of the object is needed.]

# 1 ICloneable.Clone() Method

```
2 [ILAsm]  
3 .method public hidebysig virtual abstract object Clone()  
  
4 [C#]  
5 object Clone()
```

## 6 Summary

7 Creates a copy of the current instance.

## 8 Return Value

9 A *System.Object* of the same type as the current instance, containing copies of the  
10 non-static members of the current instance.

## 11 Description

12 The exact behavior of this method is unspecified. The intent of the method is to provide  
13 a mechanism that constructs instances that are copies of the current instance, without  
14 regard for class-specific definitions of the term "copy".

15  
16 [*Note:* Use the *System.Object.MemberwiseClone* method to create a shallow copy of an  
17 object. For more information, see *System.Object.MemberwiseClone*.]  
18  
19

## 20 Behaviors

21 This method is required to return an instance of the same type as the current instance.  
22

## 23 How and When to Override

24 Implement this method to provide class-specific copying behavior.  
25

## 26 Usage

27 Use the *System.ICloneable.Clone* method to obtain a copy of the current instance.  
28

## 29 Example

1 The following example shows an implementation of `System.ICloneable.Clone` that  
2 uses the `System.Object.MemberwiseClone` method to create a copy of the current  
3 instance.

4  
5 [C#]

```
6 using System;
7 class MyClass:ICloneable {
8     public int myField;
9     public MyClass() {
10         myField = 0;
11     }
12     public MyClass(int value) {
13         myField = value;
14     }
15     public object Clone() {
16         return this.MemberwiseClone();
17     }
18 }
19 public class TestMyClass {
20     public static void Main() {
21         MyClass my1 = new MyClass(44);
22         MyClass my2 = (MyClass) my1.Clone();
23         Console.WriteLine("my1 {0} my2 {1}",my1.myField, my2.myField);
24         my2.myField = 22;
25         Console.WriteLine("my1 {0} my2 {1}",my1.myField, my2.myField);
26     }
27 }
```

28 The output is

29  
30 my1 44 my2 44

31  
32  
33 my1 44 my2 22