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.]

ICloneable.Clone() Method

[ILAsm]

```
.method public hidebysig virtual abstract object Clone()
```

[C#]

```
object Clone()
```

Summary

Creates a copy of the current instance.

Return Value

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

Description

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

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

Behaviors

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

How and When to Override

Implement this method to provide class-specific copying behavior.

Usage

Use the `System.ICloneable.Clone` method to obtain a copy of the current instance.

Example

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

[C#]

```
using System;
class MyClass:ICloneable {
    public int myField;
    public MyClass() {
        myField = 0;
    }
    public MyClass(int value) {
        myField = value;
    }
    public object Clone() {
        return this.MemberwiseClone();
    }
}
public class TestMyClass {
    public static void Main() {
        MyClass my1 = new MyClass(44);
        MyClass my2 = (MyClass) my1.Clone();
        Console.WriteLine("my1 {0} my2 {1}",my1.myField, my2.myField);
        my2.myField = 22;
        Console.WriteLine("my1 {0} my2 {1}",my1.myField, my2.myField);
    }
}
```

The output is

```
my1 44 my2 44


my1 44 my2 22
```