Clases Sealed

Cuando se aplica a una clase, el modificador `sealed` impide que otras clases hereden de ella. En el ejemplo siguiente, la clase `B` hereda de la clase `A`, pero ninguna clase puede heredar de la clase `B`.

```
class A {}
sealed class B : A {}
```

También puede usar el modificador `sealed` en un método o propiedad que invalida un método o propiedad virtual en una clase base. De este modo, puede permitir que las clases deriven de su clase e impedir que invaliden determinados métodos o propiedades virtuales.

Ejemplo

En el ejemplo siguiente, `Z` hereda de `Y` pero `Z` no puede invalidar la función virtual `F` que se declara en `X` y se sella en `Y`.

```
class X
{
    protected virtual void F() { Console.WriteLine("X.F"); }
    protected virtual void F2() { Console.WriteLine("X.F2"); }
}

class Y : X
{
    sealed protected override void F() { Console.WriteLine("Y.F"); }
    protected override void F2() { Console.WriteLine("Y.F2"); }
}

class Z : Y
{
    // Attempting to override F causes compiler error CS0239.
    // protected override void F() { Console.WriteLine("Z.F"); }

    // Overriding F2 is allowed.
    protected override void F2() { Console.WriteLine("Z.F2"); }
}
```

Al definir nuevos métodos o propiedades en una clase, puede impedir que las clases derivadas los invaliden. Para ello, no los declare como virtuales.

Es un error usar el modificador abstract con una clase sellada, porque una clase abstracta debe heredarla una clase que proporcione una implementación de los métodos o propiedades abstractos.

Cuando se aplica a un método o propiedad, el modificador sealed siempre se debe usar con override.

Dado que los structs están sellados implícitamente, no puede heredarse.

Para obtener más información, vea la sección de Herencia.

Ejemplo

```
sealed class SealedClass
{
    public int x;
    public int y;
}

class SealedTest2
{
    static void Main()
    {
        SealedClass sc = new SealedClass();
        sc.x = 110;
        sc.y = 150;
        Console.WriteLine("x = {0}, y = {1}", sc.x, sc.y);
    }
}
// Output: x = 110, y = 150
```

En el ejemplo anterior, podría intentar heredar de la clase sellada mediante la instrucción siguiente:

```
class MyDerivedC: SealedClass {} // Error
```

El resultado es un mensaje de error:

```
'MyDerivedC': cannot derive from sealed type 'SealedClass'
```

Comentarios

Para determinar si se debe sellar una clase, un método o una propiedad, por lo general debe tener en cuenta los dos puntos siguientes:

- Las posibles ventajas que podrían obtener las clases derivadas con la capacidad de personalizar la clase.
- La posibilidad de que las clases derivadas modifiquen las clases de tal manera que no funcionen correctamente o del modo esperado.