

Private Constructors (C# Programming Guide)

Article • 01/12/2022

A private constructor is a special instance constructor. It is generally used in classes that contain static members only. If a class has one or more private constructors and no public constructors, other classes (except nested classes) cannot create instances of this class. For example:

C#

```
class NLog
{
    // Private Constructor:
    private NLog() { }

    public static double e = Math.E; //2.71828...
}
```

The declaration of the empty constructor prevents the automatic generation of a parameterless constructor. Note that if you do not use an access modifier with the constructor it will still be private by default. However, the `private` modifier is usually used explicitly to make it clear that the class cannot be instantiated.

Private constructors are used to prevent creating instances of a class when there are no instance fields or methods, such as the `Math` class, or when a method is called to obtain an instance of a class. If all the methods in the class are static, consider making the complete class static. For more information see [Static Classes and Static Class Members](#).

Example

The following is an example of a class using a private constructor.

C#

```
public class Counter
{
    private Counter() { }

    public static int currentCount;
```

```
    public static int IncrementCount()
    {
        return ++currentCount;
    }
}

class TestCounter
{
    static void Main()
    {
        // If you uncomment the following statement, it will generate
        // an error because the constructor is inaccessible:
        // Counter aCounter = new Counter();    // Error

        Counter.currentCount = 100;
        Counter.IncrementCount();
        Console.WriteLine("New count: {0}", Counter.currentCount);

        // Keep the console window open in debug mode.
        Console.WriteLine("Press any key to exit.");
        Console.ReadKey();
    }
}
// Output: New count: 101
```

Notice that if you uncomment the following statement from the example, it will generate an error because the constructor is inaccessible because of its protection level:

C#

```
// Counter aCounter = new Counter();    // Error
```

See also

- [The C# type system](#)
- [Constructors](#)
- [Finalizers](#)
- [private](#)
- [public](#)