What are the default access modifiers in C#?

Asked 10 years, 1 month ago Active 8 months ago Viewed 232k times

376

141

What is the default access modifier for classes, methods, members, constructors, delegates and interfaces?

c# access-modifiers

edited Oct 31 '18 at 8:36
Muntasir
730 ● 1 ● 12 ● 19

asked Mar 26 '10 at 6:28
Surya sasidhar
25.4k ● 54 ● 127 ● 209

9 Answers

Active Oldest Votes

478

✓

The default access for everything in C# is **"the most restricted access you could declare for that member"**.

So for example:

```
namespace MyCompany
{
    class Outer
    {
        void Foo() {}
        class Inner {}
    }
}
```

is equivalent to

```
namespace MyCompany
{
    internal class Outer
    {
        private void Foo() {}
        private class Inner {}
    }
}
```

The one sort of exception to this is making one part of a property (usually the setter) more restricted than the declared accessibility of the property itself:

```
public string Name
{
    get { ... }
    private set { ... } // This isn't the default, have to do it explicitly
}
```

This is what the C# 3.0 specification has to say (section 3.5.1):

> Depending on the context in which a member declaration takes place, only certain types of declared accessibility are permitted. Furthermore, when a member declaration does not include any access modifiers, the context in which the declaration takes place determines the default declared accessibility.
>
> - Namespaces implicitly have public declared accessibility. No access modifiers are allowed on namespace declarations.
> - Types declared in compilation units or namespaces can have public or internal declared accessibility and default to internal declared accessibility.
> - Class members can have any of the five kinds of declared accessibility and default to private declared accessibility. (Note that a type declared as a member of a class can have any of the five kinds of declared accessibility, whereas a type declared as a member of a namespace can have only public or internal declared accessibility.)
> - Struct members can have public, internal, or private declared accessibility and default to private declared accessibility because structs are implicitly sealed. Struct members introduced in a struct (that is, not inherited by that struct) cannot have protected or protected internal declared accessibility. (Note that a type declared as a member of a struct can have public, internal, or private declared accessibility, whereas a type declared as a member of a namespace can have only public or internal declared accessibility.)
> - Interface members implicitly have public declared accessibility. No access modifiers are allowed on interface member declarations.
> - Enumeration members implicitly have public declared accessibility. No access modifiers are allowed on enumeration member declarations.

(Note that nested types would come under the "class members" or "struct members" parts - and therefore default to private visibility.)

edited Jul 25 '17 at 13:50
Hakan Fıstık
8,439 ● 8 ● 53 ● 81

answered Mar 26 '10 at 6:34
Jon Skeet
1179k ● 751 ● 8416 ● 8730

152

```
top level class: internal
method: private
members (unless an interface or enum): private (including nested classes)
members (of interface or enum): public
constructor: private (note that if no constructor is explicitly defined, a public
default constructor will be automatically defined)
delegate: internal
interface: internal
explicitly implemented interface member: public!
```

edited Aug 3 '13 at 17:19
Pieter
26.1k ● 71 ● 153 ● 228

answered Mar 26 '10 at 6:30
John Buchanan
4,066 ● 1 ● 15 ● 17

12 This doesn't make it clear that if a class is also a member (due to being a nested type) then it defaults to private. Also, members of an interface and enum are always public. – Jon Skeet Mar 26 '10 at 8:16

1 @niry No, it isn't private. It is public. It just doesn't get a slot in the implementing type, so if `class Foo : IFoo { IFoo.M() {} } ... Foo a = new Foo();` , you can't access `M` with `a.M()` , however you can access it with `(a as IFoo).M()` . (Check the spec for more info on the specifics) – M.Stramm Oct 20 '16 at 9:54

129

Short answer: minimum possible access (cf Jon Skeet's answer).

Long answer:

Non-nested **types, enumeration and delegate accessibilities** (*may only have internal or public accessibility*)

```
                     | Default   | Permitted declared accessibilities
------------------------------------------------------------------
namespace            | public    | none (always implicitly public)

enum                 | public    | none (always implicitly public)
```

```
interface             | internal  | public, internal

class                 | internal  | public, internal

struct                | internal  | public, internal

delegate              | internal  | public, internal
```

Nested type and member accessiblities

```
                      | Default   | Permitted declared accessibilities
------------------------------------------------------------------
namespace             | public    | none (always implicitly public)

enum                  | public    | none (always implicitly public)

interface             | public    | none

class                 | private   | All¹

struct                | private   | public, internal, private²

delegate              | private   | All¹

constructor           | private   | All¹

interface member      | public    | none (always implicitly public)

method                | private   | All¹

field                 | private   | All¹

user-defined operator| none      | public (must be declared public)
```

¹ All === public, protected, internal, private, protected internal

² structs cannot inherit from structs or classes (although they can, interfaces), hence protected is not a valid modifier

The accessibility of a nested type depends on its accessibility domain, which is determined by both the declared accessibility of the member and the accessibility domain of the immediately containing type. However, the accessibility domain of a nested type cannot exceed that of the containing type.

Note: CIL also has the provision for *protected and internal* (as opposed to the existing protected "or" internal), but to my knowledge this is not currently available for use in C#.

See:

http://msdn.microsoft.com/en-us/library/ba0a1yw2.aspx
http://msdn.microsoft.com/en-us/library/ms173121.aspx
http://msdn.microsoft.com/en-us/library/cx03xt0t.aspx
(Man I love Microsoft URIs...)

edited Aug 11 '17 at 17:10
gsamaras
63.2k ● 30 ● 132 ● 222

answered Jul 4 '10 at 18:12
Ben Aston
41.4k ● 47 ● 168 ● 283

11

Have a look at Access Modifiers (C# Programming Guide)

> **Class and Struct Accessibility**
>
> Classes and structs that are declared directly within a namespace (in other words, that are not nested within other classes or structs) can be either public or internal. Internal is the default if no access modifier is specified.
>
> Struct members, including nested classes and structs, can be declared as public, internal, or private. Class members, including nested classes and structs, can be public, protected internal, protected, internal, private protected or private. The access level for class members and struct members, including nested classes and structs, is private by default. Private nested types are not accessible from outside the containing type.
>
> Derived classes cannot have greater accessibility than their base types. In other words, you cannot have a public class B that derives from an internal class A. If this were allowed, it would have the effect of making A public, because all protected or internal members of A are accessible from the derived class.
>
> You can enable specific other assemblies to access your internal types by using the `InternalsVisibleToAttribute`. For more information, see Friend Assemblies.
>
> **Class and Struct Member Accessibility**
>
> Class members (including nested classes and structs) can be declared with any of the six types of access. Struct members cannot be declared as protected because structs do not support inheritance.

> Normally, the accessibility of a member is not greater than the accessibility of the type that contains it. However, a public member of an internal class might be accessible from outside the assembly if the member implements interface methods or overrides virtual methods that are defined in a public base class.
>
> The type of any member that is a field, property, or event must be at least as accessible as the member itself. Similarly, the return type and the parameter types of any member that is a method, indexer, or delegate must be at least as accessible as the member itself. For example, you cannot have a public method M that returns a class C unless C is also public. Likewise, you cannot have a protected property of type A if A is declared as private.
>
> User-defined operators must always be declared as public and static. For more information, see Operator overloading.
>
> Finalizers cannot have accessibility modifiers.
>
> **Other Types**
>
> Interfaces declared directly within a namespace can be declared as public or internal and, just like classes and structs, interfaces default to internal access. Interface members are always public because the purpose of an interface is to enable other types to access a class or struct. No access modifiers can be applied to interface members.
>
> Enumeration members are always public, and no access modifiers can be applied.
>
> Delegates behave like classes and structs. By default, they have internal access when declared directly within a namespace, and private access when nested.

edited Aug 22 '19 at 10:52
Amit Joshi
10.4k ● 11 ● 46 ● 93

answered Mar 26 '10 at 6:31
Adriaan Stander
143k ● 26 ● 256 ● 267

8

Class is **Internal** by default.

- Class members are **private** by default.

Interface is **Internal** by default.

- Interface members are **public** by default. (Interfaces won't allow us to specify any kind of accessibility to it's members.)

  Note: If you try to specify any access specifier to interface's members then, it shows compile error.

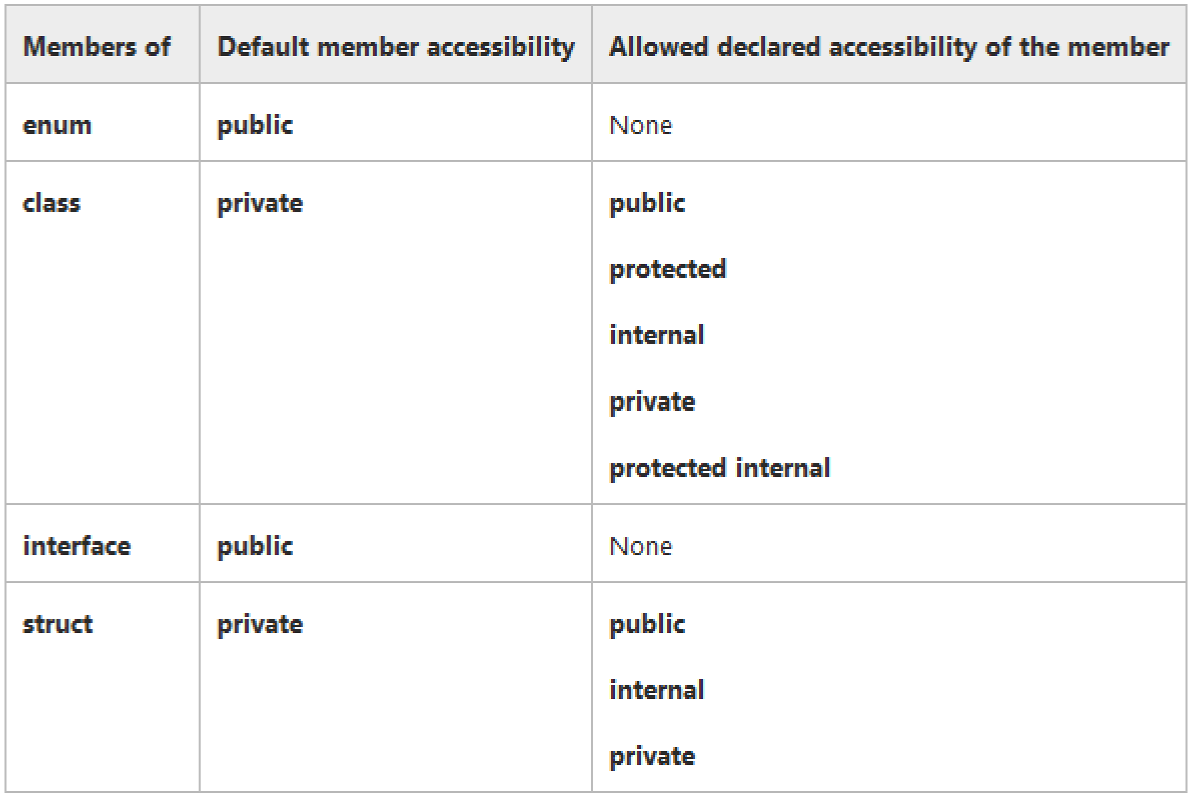
Struct is **Internal** by default.

- Struct members are **private** by default.

answered Oct 15 '18 at 7:36
Charming ZooZoo
656 ● 6 ● 13

5

I would like to add some documentation link. Check out more detail here.



Members of	Default member accessibility	Allowed declared accessibility of the member
enum	**public**	None
class	**private**	**public** **protected** **internal** **private** **protected internal**
interface	**public**	None
struct	**private**	**public** **internal** **private**

answered Oct 1 '16 at 11:17
Asif Mushtaq
2,483 ● 1 ● 23 ● 53

1 class is by default internal and not private. – Baahubali Dec 22 '16 at 2:53

1 Where I wrote class is private? – Asif Mushtaq Apr 5 '17 at 16:05

This table is valid only for nested types. – BlueSilver Jun 21 '17 at 11:04

class is internal by default and class inside namespace can not be private. but class inside a class(nested class) can be private – Arun Oct 12 '17 at 5:27

Interface's access modifier is **Internal** by default. – Charming ZooZoo Oct 15 '18 at 7:07

3

Simplest answer is the following.....

All members in C# always take the LEAST accessible modifier possible by default.

That is why all top level classes in an assembly are "internal" by default, which means they are public to the assembly they are in, but private or excluded from access to outside assemblies. The only other option for a top level class is public which is more accessible. For nested types its all private except for a few rare exceptions like members of enums and interfaces which can only be public. Some examples. In the case of top level classes and interfaces, the defaults are:

class Animal same as **internal class Animal**

interface Animal same as **public interface Animal**

In the case of nested classes and interfaces (inside types), the defaults are:

class Animal same as **private class Animal**

interface Animal same as **private interface Animal**

If you just assume the default is always the most private, then you do not need to use an accessors until you need to change the default. Easy.

answered Oct 11 '18 at 4:49
Stokely
827 ● 8 ● 6

1

Internal is the default modifier

answered May 14 '10 at 11:12
renjucool

302 ● 3 ● 19

0

Namespace level: `internal`

Type level: `private`

answered Mar 26 '10 at 6:31

leppie

105k ● 16 ● 179 ● 283

Highly active question. Earn 10 reputation in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.