The following six accessibility levels can be specified using the access modifiers:

* [public](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/public): Access is not restricted.
* [protected](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/protected): Access is limited to the containing class or types derived from the containing class.
* [internal](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/internal): Access is limited to the current assembly.
* [protected internal](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/protected-internal): Access is limited to the current assembly or types derived from the containing class.
* [private](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/private): Access is limited to the containing type.
* [private protected](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/private-protected): Access is limited to the containing class or types derived from the containing class within the current assembly.

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

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.

namespace MyCompany

{

class Outer

{

void Foo() {}

class Inner {}

}

}

is equivalent to

namespace MyCompany

{

internal class Outer

{

private void Foo() {}

private class Inner {}

}

}