# internal (C# Reference)

Article • 01/25/2022 • 2 minutes to read

The `internal` keyword is an access modifier for types and type members.

> This page covers `internal` access. The `internal` keyword is also part of the protected internal access modifier.

Internal types or members are accessible only within files in the same assembly, as in this example:

C#

```csharp
public class BaseClass
{
    // Only accessible within the same assembly.
    internal static int x = 0;
}
```

For a comparison of `internal` with the other access modifiers, see Accessibility Levels and Access Modifiers.

For more information about assemblies, see Assemblies in .NET.

A common use of internal access is in component-based development because it enables a group of components to cooperate in a private manner without being exposed to the rest of the application code. For example, a framework for building graphical user interfaces could provide `Control` and `Form` classes that cooperate by using members with internal access. Since these members are internal, they are not exposed to code that is using the framework.

It is an error to reference a type or a member with internal access outside the assembly within which it was defined.

## Example 1

This example contains two files, `Assembly1.cs` and `Assembly1_a.cs`. The first file contains an internal base class, `BaseClass`. In the second file, an attempt to instantiate `BaseClass` will produce an error.

C#

```
// Assembly1.cs
// Compile with: /target:library
internal class BaseClass
{
   public static int intM = 0;
}
```

C#

```
// Assembly1_a.cs
// Compile with: /reference:Assembly1.dll
class TestAccess
{
   static void Main()
   {
      var myBase = new BaseClass();   // CS0122
   }
}
```

## Example 2

In this example, use the same files you used in example 1, and change the accessibility level of `BaseClass` to `public`. Also change the accessibility level of the member `intM` to `internal`. In this case, you can instantiate the class, but you cannot access the internal member.

C#

```
// Assembly2.cs
// Compile with: /target:library
public class BaseClass
{
   internal static int intM = 0;
}
```

C#

```
// Assembly2_a.cs
// Compile with: /reference:Assembly2.dll
public class TestAccess
{
   static void Main()
   {
      var myBase = new BaseClass();   // Ok.
      BaseClass.intM = 444;    // CS0117
   }
}
```

## C# Language Specification

For more information, see Declared accessibility in the C# Language Specification. The language specification is the definitive source for C# syntax and usage.

## See also

- C# Reference
- C# Programming Guide
- C# Keywords
- Access Modifiers
- Accessibility Levels
- Modifiers
- public
- private
- protected