## **Access Modifiers**Modifier Akses

**Access modifiers** are keywords that you add to a type or a type member's declaration to specify its accessibility from other types and other assemblies (compiled .dll or .exe files). They allow you to promote **encapsulation**, which is the principle of hiding the internal implementation details of an object and exposing only what is necessary for other parts of the program to interact with it.Modifier akses adalah kata kunci yang ditambahkan ke deklarasi tipe atau anggota tipe untuk menentukan tingkat aksesibilitasnya dari tipe lain dan perakitan lain (berkas .dll atau .exe yang telah dikompilasi). Modifier akses memungkinkan Anda untuk meningkatkan enkapsulasi, yaitu prinsip menyembunyikan detail implementasi internal suatu objek dan hanya mengekspos apa yang diperlukan agar bagian lain dari program dapat berinteraksi dengannya.

Here are the primary access modifiers in C#, ordered roughly from most accessible to least accessible:Berikut adalah modifikator akses utama dalam C#, diurutkan secara kasar dari yang paling mudah diakses hingga yang paling sulit diakses:

### **public**

* **Fully accessible.**
* A public type or member can be accessed by any other code, regardless of the assembly it resides in.Tipe atau anggota publik dapat diakses oleh kode lain, terlepas dari assembly tempatnya berada.
* This is the implicit accessibility for members of an enum or an interface.Ini adalah aksesibilitas implisit untuk anggota dari sebuah enum atau antarmuka.

### **internal**

* Accessible **only within the containing assembly** or from **friend assemblies**.Hanya dapat diakses di dalam assembly yang mengandungnya atau dari assembly teman.
* This is the default accessibility for non-nested types (classes, structs, enums, interfaces) if no access modifier is explicitly specified.Ini adalah tingkat aksesibilitas default untuk tipe non-bersarang (kelas, struktur, enum, antarmuka) jika tidak ada modifikator akses yang secara eksplisit ditentukan.
* It's commonly used for components that are part of a library but not intended for direct use by external consumers of that library.Hal ini umumnya digunakan untuk komponen yang merupakan bagian dari perpustakaan tetapi tidak dimaksudkan untuk digunakan secara langsung oleh pengguna eksternal perpustakaan tersebut.

### **private**

* Accessible **only within the containing type**.Hanya dapat diakses di dalam tipe yang mengandungnya.
* This is the default accessibility for members (fields, methods, properties, etc.) of a class or struct if no access modifier is explicitly specified.Ini adalah tingkat akses default untuk anggota (bidang, metode, properti, dll.) dari sebuah kelas atau struktur jika tidak ada modifikator akses yang secara eksplisit ditentukan.
* private members are the internal implementation details of a class and cannot be accessed from outside that class, even by derived classes.Anggota privat adalah detail implementasi internal dari sebuah kelas dan tidak dapat diakses dari luar kelas tersebut, bahkan oleh kelas turunan.

### **protected**

* Accessible **only within the containing type or its subclasses (derived types)**.Hanya dapat diakses di dalam tipe induk atau subkelasnya (tipe turunan).
* protected members allow derived classes to access or extend behavior while keeping it hidden from unrelated types.Anggota yang dilindungi memungkinkan kelas turunan untuk mengakses atau memperluas perilaku sambil tetap menyembunyikannya dari jenis yang tidak terkait.

### **protected internal**

* This is a **union** of protected and internal accessibility.Ini adalah gabungan antara aksesibilitas yang dilindungi dan aksesibilitas internal.
* A member that is protected internal is accessible:Anggota yang dilindungi secara internal dapat diakses:
  + Within the containing type.Di dalam jenis yang mengandung.
  + From subclasses, regardless of their assembly.Dari subkelas, terlepas dari perakitan mereka.
  + From any code within the same assembly.Dari kode apa pun dalam assembly yang sama.

**private protected**

* This is an **intersection** of protected and internal accessibility.Ini adalah perpotongan antara aksesibilitas yang dilindungi dan aksesibilitas internal.
* A member that is private protected is accessible:Anggota yang dilindungi secara pribadi dapat diakses:
  + Only within the containing type.Hanya dalam tipe yang mengandung.
  + From subclasses *that reside within the same assembly*.Dari subkelas yang berada dalam assembly yang sama.
* This makes it *less accessible* than protected alone (which allows subclasses in any assembly) or internal alone (which allows any type in the same assembly). It's used for highly constrained internal access for derived types.Hal ini membuatnya kurang dapat diakses dibandingkan dengan akses protected saja (yang memungkinkan subkelas di assembly mana pun) atau internal saja (yang memungkinkan tipe apa pun di assembly yang sama). Fitur ini digunakan untuk akses internal yang sangat terbatas untuk tipe turunan.

### **file (from C# 11)**

* Accessible **only from within the same file**.Hanya dapat diakses dari dalam file yang sama.
* This modifier can *only* be applied to **type declarations** (classes, structs, interfaces, enums, delegates).Modifier ini hanya dapat diterapkan pada deklarasi tipe (kelas, struktur, antarmuka, enum, delegasi).
* It's primarily intended for use by source generators, allowing them to create helper types that are truly local to a single generated file, preventing naming conflicts or accidental exposure.Ini terutama ditujukan untuk digunakan oleh generator sumber, memungkinkan mereka untuk membuat tipe bantu yang benar-benar lokal pada satu file yang dihasilkan, sehingga mencegah konflik penamaan atau paparan yang tidak disengaja.

## **Examples of Access Modifiers**

Let's illustrate these with some code examples:

**Type Accessibility:**

|  |
| --- |
| // Class1 is internal (default accessibility for non-nested types) // It cannot be accessed by code outside this assembly. class Class1 { }  // Class2 is public and can be accessed by code from any assembly. public class Class2 { } |

**Member Accessibility:**

|  |
| --- |
| class ClassA {  int x; // x is private (default accessibility for class/struct members)  // Only code within ClassA can access x. }  class ClassB {  internal int x; // x is internal  // Any code within the same assembly as ClassB can access x. } |

**Inheritance and protected:**

|  |
| --- |
| class BaseClass {  void Foo() { } // Foo is private (default)  // Only code within BaseClass can call Foo.   protected void Bar() { } // Bar is protected  // Code within BaseClass AND any subclasses can call Bar. }  class Subclass : BaseClass {  void Test1()  {  // Foo(); // Error: Cannot access Foo because it is private in BaseClass.  }   void Test2()  {  Bar(); // OK: Can access Bar because it is protected and Subclass is a subclass.  } } |

## **Friend Assemblies**

You can allow internal members to be accessible from other assemblies by using the System.Runtime.CompilerServices.InternalsVisibleTo assembly attribute. This is often called creating **"friend assemblies.**"Anda dapat mengizinkan anggota internal agar dapat diakses dari perakitan lain dengan menggunakan atribut perakitan System.Runtime.CompilerServices.InternalsVisibleTo. Hal ini sering disebut sebagai pembuatan “perakitan teman.”

|  |
| --- |
| // In AssemblyA's AssemblyInfo.cs or .csproj: [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("FriendAssemblyB")]  // Now, code in FriendAssemblyB can access internal members of AssemblyA. |

If the friend assembly has a strong name (a cryptographic identity), you must specify its full public key in the attribute.Jika perakitan teman memiliki nama yang kuat (identitas kriptografis), Anda harus menentukan kunci publik lengkapnya dalam atribut tersebut.

## **Accessibility Capping**

A type's accessibility acts as an **upper cap** for the accessibility of its declared members. This means a member cannot be *more* accessible than its containing type.Aksesibilitas suatu tipe bertindak sebagai batas atas untuk aksesibilitas anggota-anggotanya yang dideklarasikan. Artinya, suatu anggota tidak dapat memiliki aksesibilitas yang lebih tinggi daripada tipe yang menampungnya.

The most common scenario is an internal type with public members:Skenario yang paling umum adalah tipe internal dengan anggota publik:

|  |
| --- |
| // Class C is internal by default class C {  public void Foo() { } // Foo is declared public, but... } |

Because C is internal, its public member Foo is effectively also internal. You cannot access Foo from outside the assembly where C is defined, even if Foo itself is declared public. A common reason for marking Foo as public in such a scenario is to facilitate refactoring; if C later becomes public, Foo will automatically become truly public without requiring a code change. Karena C bersifat internal, anggota publiknya Foo secara efektif juga bersifat internal. Anda tidak dapat mengakses Foo dari luar assembly tempat C didefinisikan, meskipun Foo sendiri dinyatakan sebagai publik. Alasan umum untuk menandai Foo sebagai publik dalam skenario ini adalah untuk memudahkan refactoring; jika C nantinya menjadi publik, Foo akan secara otomatis menjadi benar-benar publik tanpa memerlukan perubahan kode.

## **Restrictions on Access Modifiers** Pembatasan pada Modifier Akses

The C# compiler enforces strict rules to prevent inconsistent use of access modifiers:Penerjemah C# menerapkan aturan ketat untuk mencegah penggunaan yang tidak konsisten dari modifikator akses.

* **Overriding Members:** When overriding a base class function (using override), the accessibility of the overridden function **must be identical** to the base class's virtual member. You cannot, for example, override a protected method with a public one. Mengganti Anggota Kelas Dasar: Saat mengganti fungsi kelas dasar (menggunakan override), aksesibilitas fungsi yang diganti harus sama dengan anggota virtual kelas dasar. Anda tidak dapat, misalnya, mengganti metode protected dengan metode public.

|  |
| --- |
| class BaseClass { protected virtual void Foo() {} } class Subclass1 : BaseClass { protected override void Foo() {} } // OK // class Subclass2 : BaseClass { public override void Foo() {} } // Error: Cannot change accessibility |

* (A specific exception exists: if overriding a protected internal method from another assembly, the override must simply be protected.)(Ada pengecualian khusus: jika meng-override metode internal yang dilindungi dari assembly lain, override tersebut harus dilindungi.)
* **Subclass Accessibility:** A derived class cannot be *more* accessible than its base class. It can be equally or less accessible.Aksesibilitas Subkelas: Sebuah kelas turunan tidak dapat memiliki tingkat aksesibilitas yang lebih tinggi daripada kelas dasarnya. Kelas turunan dapat memiliki tingkat aksesibilitas yang sama atau lebih rendah.

|  |
| --- |
| internal class A {} // Base class is internal // public class B : A {} // Error: Cannot derive public class B from internal class A |

* This prevents exposing internal implementation details through a public interface.Hal ini mencegah pengungkapan detail implementasi internal melalui antarmuka publik.

Properly applying access modifiers is fundamental to designing maintainable, secure, and robust software systems in C#. They are a primary mechanism for enforcing encapsulation and defining clear contracts between different parts of your codebase.Penggunaan modifikator akses yang tepat merupakan hal yang mendasar dalam merancang sistem perangkat lunak yang dapat dipelihara, aman, dan tangguh dalam C#. Modifikator akses merupakan mekanisme utama untuk menerapkan enkapsulasi dan mendefinisikan kontrak yang jelas antara berbagai bagian dalam basis kode Anda.