C# Access Modifiers [1] [2] [3]

In C#, **access modifiers** define the visibility and accessibility of classes, structs, interfaces, and their members. They are essential for **encapsulation** and controlling how different parts of a program interact.

Main Modifiers:

- **public** – Accessible from anywhere, across assemblies.
- **private** – Accessible only within the same class or struct (default for members).
- **protected** – Accessible within the same class and derived classes.
- **internal** – Accessible only within the same assembly.
- **protected internal** – Accessible within the same assembly or from derived classes in other assemblies.
- **private protected** – Accessible only within the same assembly and from derived classes.
- **file** – Accessible only within the same source file (C# 11+).

Example Usage:

```
public class Vehicle
{
  private string model; // Only within Vehicle
  protected int speed; // Vehicle + derived classes
  internal string brand; // Same assembly
  protected internal int year; // Same assembly or derived classes
  private protected bool isElectric; // Same assembly + derived classes
  public void Drive() => Console.WriteLine("Driving...");
}

public class Car : Vehicle
{
  public void ShowDetails()
  {
    speed = 100; // Accessible (protected)
    year = 2023; // Accessible (protected internal)
    isElectric = true; // Accessible (private protected)
  }
}
```

Key Points:

- **Default accessibility**: Top-level types → internal Class/struct members → private
- **Structs** cannot have `protected`, `protected internal`, or `private protected` members because they don't support inheritance.
- **Derived classes** cannot have higher accessibility than their base class.
- **Interface members** are `public` by default.
- Use `InternalsVisibleTo` to share `internal` members with specific assemblies.

Best Practices:

- Use **private** for fields to enforce encapsulation.
- Expose only necessary members as **public**.
- Use **protected** for extensibility in inheritance.
- Limit **internal** usage to assembly-specific APIs.
- Combine modifiers (`protected internal`, `private protected`) for fine-grained control.

This structured approach ensures **security, maintainability, and clarity** in your C# codebase.

Learn more: 1 - learn.microsoft.com 2 - geeksforgeeks.org 3 - w3schools.com

See less

GeeksForGeeks
https://www.geeksforgeeks.org › c-sharp › access-modifiers-in-c-sharp

Access Modifiers in C# - GeeksforGeeks

Sep 17, 2025 · Access modifiers in C# define the scope and visibility of classes, methods, fields, constructors and other members. They determine where and how a member can be accessed in a ...

Microsoft Learn
https://learn.microsoft.com › ... › classes-and-structs › access-modifiers

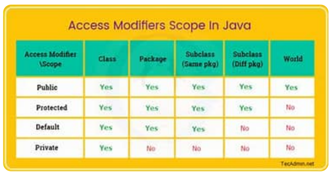
Access Modifiers - C# | Microsoft Learn

Oct 10, 2025 · All types and type members in C# have an accessibility level that controls whether they

Access modifiers

Access modifiers are keywords in object-oriented languages that set the accessibility of classes, methods, and other members. Access modifiers are a specific part of programming language syntax used t...

Wikipedia



Access Modifiers Scope In Java

Access Modifier \Scope	Class	Package	Subclass (Same pkg)	Subclass (Diff pkg)	World
Public	Yes	Yes	Yes	Yes	Yes
Protected	Yes	Yes	Yes	Yes	No
Default	Yes	Yes	Yes	No	No
Private	Yes	No	No	No	No

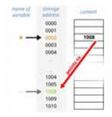
TecAdmin.net

Access modifiers are one way to achieve this goal. They **tell other users** of your code how you want them to interact with specific methods or variables.

Access modifiers play an important role to **protect** the data from unauthorized access as well as protecting it from getting manipulated.

Access modifiers are used to **implement encapsulation** of OOP. Access modifiers allow you to define who does or who doesn't have access to certain features.

Explore more

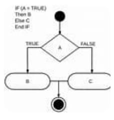


Pointer

W3Schools

C Sharp

GitHub



Conditional

Data: Wikipedia · studytonight.com · tecadmin.net · c-sharpcorner.com
Wikipedia text under CC-BY-SA license

Feedback

Deep dive into access...

difference between private and protected c#