to use Access Modifiers In C#.

C# Curator Mar 20 2019

20 34 660.3k

Access Modifiers In C#

Access modifiers in C# are used to specify the scope of accessibility of a member of a class or type of the class itself. For example, a public class is accessible to everyone without any restrictions, while an internal class may be accessible to the assembly only.

Why to use access modifiers?

Access modifiers are an integral part of object-oriented programming. 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.

In C# there are 6 different types of Access Modifiers.

Modifier	Description
public	There are no restrictions on accessing public members.
private	Access is limited to within the class definition. This is the default access modifier type if none is formally specified
protected	Access is limited to within the class definition and any class that inherits from the class

protected internal	Access is limited to the current assembly and types derived from the containing class. All members in current project and all members in derived class can access the variables.
private	

```
using System;
namespace AccessModifiers
{
    class Program
    {
        class AccessMod
        {
            public int num1;
        }
        static void Main(string[] args)
        {
            AccessMod ob1 = new AccessMod();
            //Direct access to public members
            ob1.num1 = 100;
            Console.WriteLine("Number one value in main {0}", ob1.num1);
            Console.ReadLine();
        }
    }
}
```

public modifier

The public keyword is an access modifier for types and type members. Public access is the most permissive access level.

There are no restrictions on accessing public members.

Accessibility

- Can be accessed by objects of the class
- Can be accessed by derived classes

private modifier

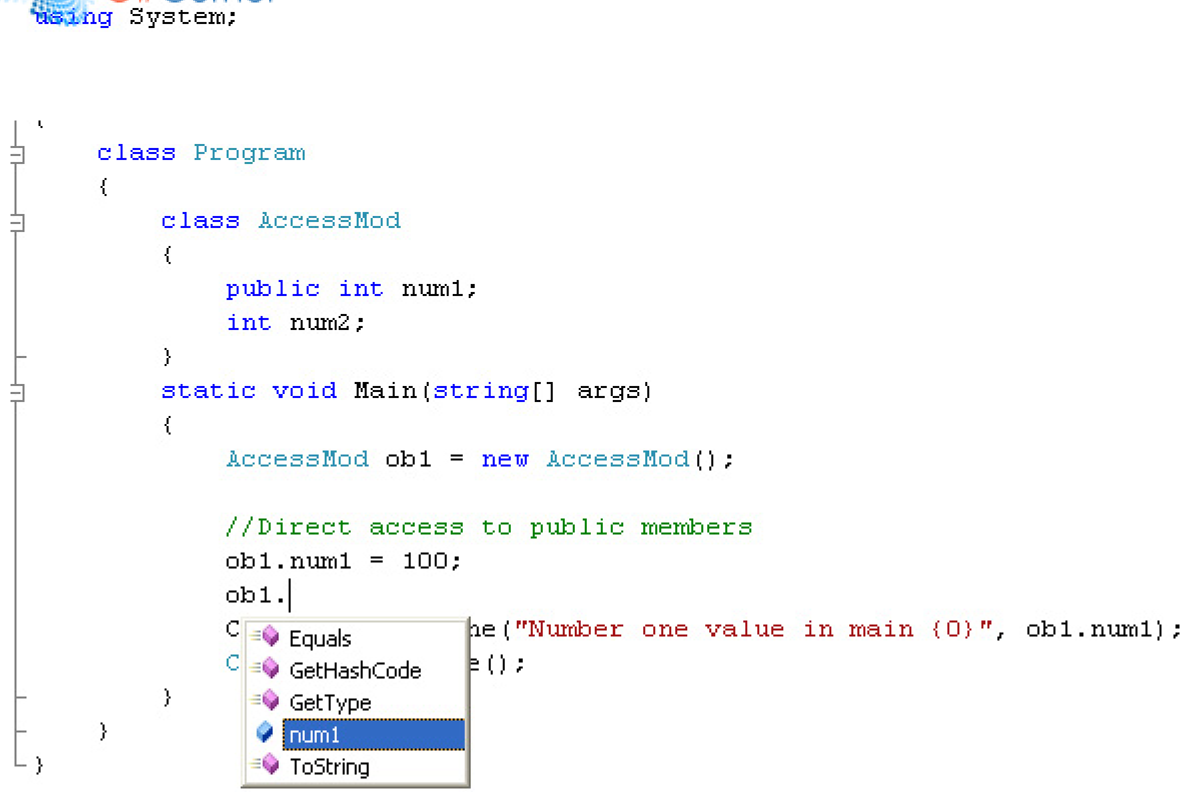
Private members are accessible only within the body of the class or the struct in which they are declared.

Accessibility

- Cannot be accessed by object
- Cannot be accessed by derived classes

Example: In the following example num2 is not accessible outside the class.

```
using System;
namespace AccessModifiers
{
    class Program
    {
        class AccessMod
        {
            public int num1;
            int num2;
        }
        static void Main(string[] args)
        {
            AccessMod ob1 = new AccessMod();
            //Direct access to public members
            ob1.num1 = 100;
            //Access to private member is not permitted
            ob1.num2 = 20;
            Console.WriteLine("Number one value in main {0}", ob1.num1);
            Console.ReadLine();
        }
    }
}
```



```
using System;

    class Program
    {
        class AccessMod
        {
            public int num1;
            int num2;
        }
        static void Main(string[] args)
        {
            AccessMod ob1 = new AccessMod();

            //Direct access to public members
            ob1.num1 = 100;
            ob1.
            C                  he("Number one value in main {0}", ob1.num1);
            C                  e();
        }
    }
}
```

protected modifier

A protected member is accessible from within the class in which it is declared, and from within any class derived from the class that declared this member.

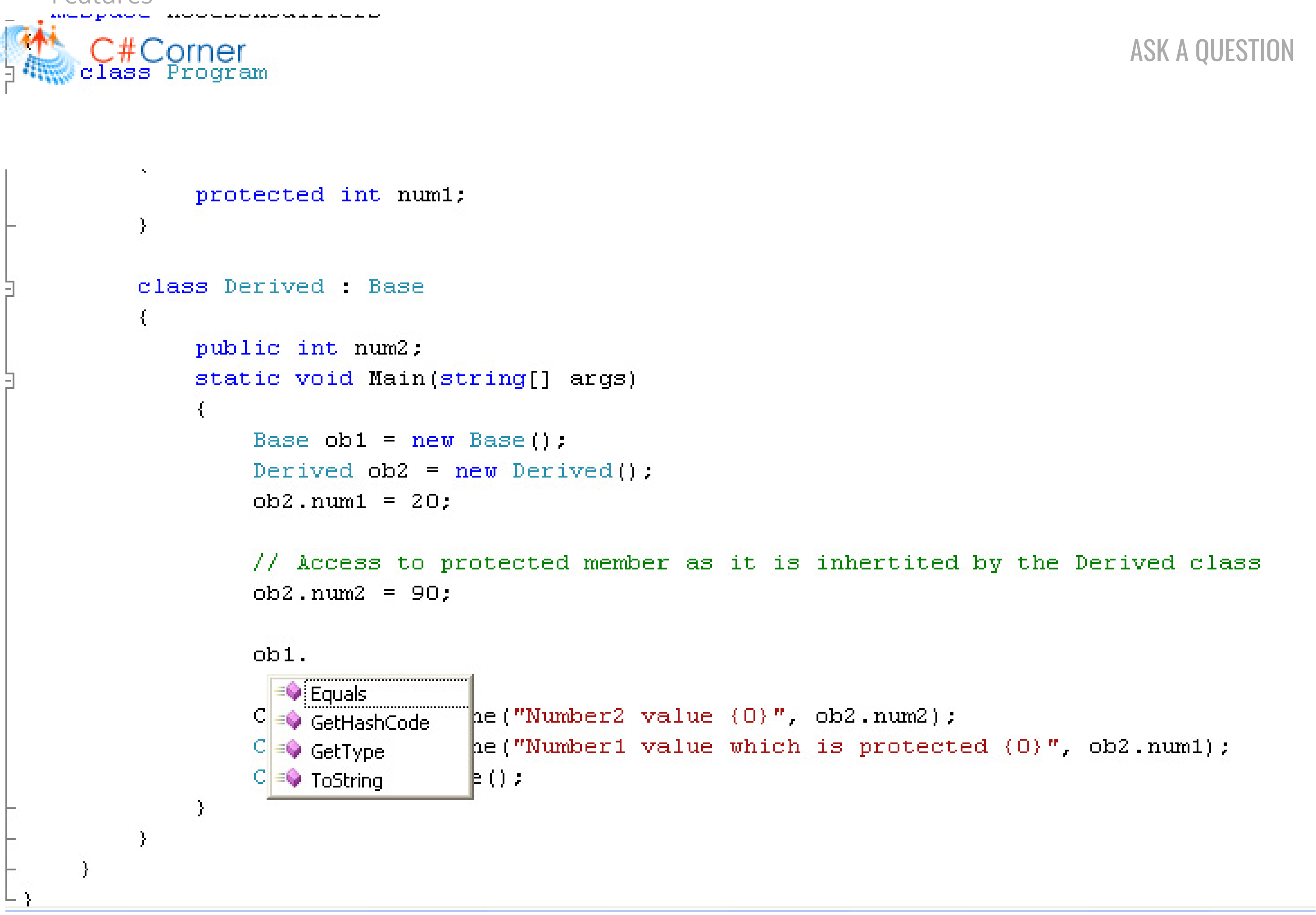
A protected member of a base class is accessible in a derived class only if the access takes place through the derived class type.

Accessibility

- Cannot be accessed by object

```
using System;
namespace AccessModifiers
{
    {
        class Base
        {
            protected int num1;
        }
        class Derived : Base
        {
            public int num2;
            static void Main(string[] args)
            {
                Base ob1 = new Base();
                Derived ob2 = new Derived();
                ob2.num1 = 20;
                // Access to protected member as it is inherited by the Derived class
                ob2.num2 = 90;
                Console.WriteLine("Number2 value {0}", ob2.num2);
                Console.WriteLine("Number1 value which is protected {0}", ob2.num1);
                Console.ReadLine();
            }
        }
    }
}
```

In the above program we try to access protected member in main it is not available as shown in the picture below that num1 is not listed in intellisense.



```
namespace AccessModifiers
class Program

            protected int num1;
        }

        class Derived : Base
        {
            public int num2;
            static void Main(string[] args)
            {
                Base ob1 = new Base();
                Derived ob2 = new Derived();
                ob2.num1 = 20;

                // Access to protected member as it is inhertited by the Derived class
                ob2.num2 = 90;

                ob1.
                C              ne("Number2 value {0}", ob2.num2);
                C              ne("Number1 value which is protected {0}", ob2.num1);
                C              e();
            }
        }
    }
}
```

internal modifier

The internal keyword is an access modifier for types and type members. We can declare a class as internal or its member as internal. Internal members are accessible only within files in the same assembly (.dll).

In other words, access is limited exclusively to classes defined within the current project assembly.

In same assembly (public)

In other assembly (internal)

- Cannot be accessed by object
- Cannot be accessed by derived classes

protected internal modifier

The protected internal accessibility means protected OR internal, not protected AND internal.

In other words, a protected internal member is accessible from any class in the same assembly, including derived classes.

The protected internal access modifier seems to be a confusing but is a union of protected and internal in terms of providing access but not restricting. It allows:

- Inherited types, even though they belong to a different assembly, have access to the protected internal members.
- Types that reside in the same assembly, even if they are not derived from the type, also have access to the protected internal members.

Default access

A default access level is used if no access modifier is specified in a member declaration. The following list defines the default access modifier for certain C# types:

enum: The default and only access modifier supported is public.
class: The default access for a class is private. It may be explicitly defined using any of the access modifiers.
interface: The default and only access modifier supported is public.
struct: The default access is private with public and internal supported as well.

Note: Interface and enumeration members are always public and no access modifiers are allowed.

Conclusion

I hope that this article would have helped you in understanding accessibility modifiers. Your feedback and constructive contributions are welcome.

access modifiers default access internal private protected protected internal public

C# Curator TOP 100

This is a C# Corner community account used by curators.

https://www.c-sharpcorner.com/members/puran-mehra

71 17.4m 1

View Previous Comments

20 34

Type your comment here and press Enter Key (Minimum 10 characters)

Nicely Written C# Curator. We also have this article as video on C# Corner : > https://www.c-sharpcorner.com/article/access-modifiers-in-c-sharp/

Kapil Gaur

Mar 20, 2019

293 5.9k 116.9k

0 0 Reply

C# Curator Mahesh Chand I think you should mention this line "default access modifier for Class is internal, Internal if it is directly declared inside the namespace and Private if it is nested." . can you please change this line? so freshers can learn right thing. thank you.

OK thanks. I guess I can add this note at the top somewhere.

Nice article for freshers...
Abinash Hota
1742 19 0
Jan 02, 2019
1 0 Reply

Good...........,........
Hamid Khan
583 2.4k 180.7k
Sep 19, 2017
4 0 Reply

Class: The default access for a class is internal. It may be explicitly defined using any of the access modifiers.
bheema munagala
1711 50 0
Aug 04, 2017
3 0 Reply

Very good work keep sharing
Subash
301 5.7k 33.4k
Mar 17, 2017
3 0 Reply

Pretty good stuff but I could also find it on the http://msdn.com site as well.
Darko Markovski
1760 1 0
Jun 17, 2016
3 0 Reply

nice
Ashish Srivastava
779 1.4k 69.3k
Apr 29, 2016
2 0 Reply

Hi Puran, default access modifier for class is internal not private
Syed Md. Kamruzzaman
1702 59 0
Jan 29, 2016
3 0 Reply

wow, very easy to understand. thanks a lot
Syed Md. Kamruzzaman
Jan 29, 2016

cus
C# Corner
Features

ASK A QUESTION CONTRIBUTE

OUR TRAINING

Unity Game Development
Learn Unity game design & 2D & 3D game development.

TRENDING UP

01 C# Coding Standards 😎

02 Microservices Using ASP.NET Core

03 Building High Performance Back End (SQL Server)

04 ASP.NET MVC Request Life Cycle

05 Filtration, Sorting, And Pagination In Angular 7 Using Web API And SQL Server

cus

Features

C# Corner

07 All About C# Immutable Classes

ASK A QUESTION CONTRIBUTE

09 Implement CRUD Operations With Sorting, Searching And Paging Using EF Core In ASP.NET Core

10 What Does “var” Mean In C#?

View All

About Us Contact Us Privacy Policy Terms Media Kit Sitemap Report a Bug FAQ Partners C# Tutorials