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.

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

1. **class** Program
2. {
3. **class** AccessMod
4. {
5. **public** **int** num1;
6. }
7. **static** **void** Main(**string**[] args)
8. {
9. AccessMod ob1 = **new** AccessMod();
10. //Direct access to public members
11. ob1.num1 = 100;
12. Console.WriteLine("Number one value in main {0}", ob1.num1);
13. Console.ReadLine();
14. }
15. }

private modifier

Private access is the least permissive access level.

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
* **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();
* }
* }

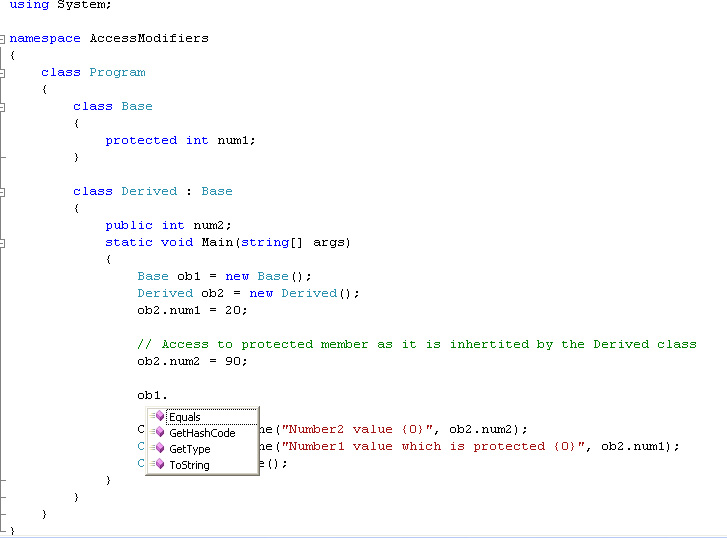
The above program will give compilation error, as access to private is not permissible. In the below figure you can see the private member num2 is not available

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.

1. **class** Program
2. {
3. **class** Base
4. {
5. **protected** **int** num1;
6. }
7. **class** Derived : Base
8. {
9. **public** **int** num2;
10. **static** **void** Main(**string**[] args)
11. {
12. Base ob1 = **new** Base();
13. Derived ob2 = **new** Derived();
14. ob2.num1 = 20;
15. // Access to protected member as it is inherited by the Derived class
16. ob2.num2 = 90;
17. Console.WriteLine("Number2 value {0}", ob2.num2);
18. Console.WriteLine("Number1 value which is protected {0}", ob2.num1);
19. Console.ReadLine();
20. }
21. }
22. }



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.

**Accessibility**

In same assembly (public)

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

In other assembly (internal)

* Cannot be accessed by object
* Cannot be accessed by derived classes
* namespace **Internal\_Access\_Specifier**
* {
* class access
* {
* // String Variable declared as internal
* internal string name;
* public void print()
* {
* **Console**.**WriteLine**("\nMy name is " + name);
* }
* }
* class **Program**
* {
* static void **Main**(string[] args)
* {
* access ac = new access();
* **Console**.**Write**("Enter your name:\t");
* // Accepting value in internal variable
* ac.name = **Console**.**ReadLine**();
* ac.print();
* **Console**.**ReadLine**();
* }
* }
* }
* namespace Tutlane
* {
* class User
* {
* internal string Name;
* internal string Location;
* internal int Age;
* internal void GetUserDetails()
* {
* Console.WriteLine("Name: {0}", Name);
* Console.WriteLine("Location: {0}", Location);
* Console.WriteLine("Age: {0}", Age);
* }
* }
* class Program
* {
* static void Main(string[] args)
* {
* User u = new User();
* u.Name = "Bibekananda";
* u.Name = " Bibekananda ";
* u.Location = "BBSR";
* u.Age = 32;
* u.GetUserDetails();
* Console.WriteLine("\nPress Enter Key to Exit..");
* Console.ReadLine();
* }
* }
* }

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.
* class User
* {
* protected internal string Name;
* protected internal string Location;
* protected internal int Age;
* protected internal void GetUserDetails()
* {
* Console.WriteLine("Name: {0}", Name);
* Console.WriteLine("Location: {0}", Location);
* Console.WriteLine("Age: {0}", Age);
* }
* }
* class Program
* {
* static void Main(string[] args)
* {
* User u = new User();
* u.Name = "Bibekananda";
* u.Name = "Panigrahi";
* u.Location = "BBSR";
* u.Age = 32;
* u.GetUserDetails();
* Console.WriteLine("\nPress Enter Key to Exit..");
* Console.ReadLine();
* }
* }