Table 5-1. C# Access Modifiers

C# Access Modifier May Be Applied To Meaning in Life

public : Types or type members : Public items have no access restrictions. A public member can be accessed from an object, as well as any derived class. A public type can be accessed from other external assemblies.

Private: Type members or nested types : Private items can only be accessed by the class (or structure) that defines the item.

Protected: Type members or nested types: Protected items can be used by the class which defines it, and **any child class.** However, protected items cannot be accessed from the outside world using the C# dot operator.

Internal: Types or type members : Internal items are accessible only within **the current assembly**. Therefore, if you define a set of internal types within a .NET class library, other assemblies are not able

to make use of them.

Protected internal: Type members or nested types: When the protected and internal keywords are Combined on an item, the item is accessible within the defining assembly, the defining class, and by derived classes.

The Default Access Modifiers

By default, **type members** are *implicitly* ***private***while **types** are *implicitly* ***internal***

the private, protected, and protected internal access modifiers can be

**applied to a *nested type*.**

public class SportsCar

{

// OK! Nested types can be marked private.

private enum CarColor

{

Red, Green, Blue

}

}

Here, it is permissible to apply the private access modifier on the nested type. However, nonnested

types (such as the SportsCar) can only be defined with the public or internal modifiers. Therefore, the

following class definition is illegal:

// Error! Nonnested types cannot be marked private!

private class SportsCar

{}