Accessibility Domain (C# Reference)

Article • 09/15/2021 • 2 minutes to read

The accessibility domain of a member specifies in which program sections a member can be referenced. If the member is nested within another type, its accessibility domain is determined by both the accessibility level of the member and the accessibility domain of the immediately containing type.

The accessibility domain of a top-level type is at least the program text of the project that it is declared in. That is, the domain includes all of the source files of this project. The accessibility domain of a nested type is at least the program text of the type in which it is declared. That is, the domain is the type body, which includes all nested types. The accessibility domain of a nested type never exceeds that of the containing type. These concepts are demonstrated in the following example.

Example

This example contains a top-level type, `T1`, and two nested classes, `M1` and `M2`. The classes contain fields that have different declared accessibilities. In the `Main` method, a comment follows each statement to indicate the accessibility domain of each member. Notice that the statements that try to reference the inaccessible members are commented out. If you want to see the compiler errors caused by referencing an inaccessible member, remove the comments one at a time.

C#

```csharp
public class T1
{
    public static int publicInt;
    internal static int internalInt;
    private static int privateInt = 0;

    static T1()
    {
        // T1 can access public or internal members
        // in a public or private (or internal) nested class.
        M1.publicInt = 1;
        M1.internalInt = 2;
        M2.publicInt = 3;
        M2.internalInt = 4;
```

```
            // Cannot access the private member privateInt
            // in either class:
            // M1.privateInt = 2; //CS0122
        }

        public class M1
        {
            public static int publicInt;
            internal static int internalInt;
            private static int privateInt = 0;
        }

        private class M2
        {
            public static int publicInt = 0;
            internal static int internalInt = 0;
            private static int privateInt = 0;
        }
    }

    class MainClass
    {
        static void Main()
        {
            // Access is unlimited.
            T1.publicInt = 1;

            // Accessible only in current assembly.
            T1.internalInt = 2;

            // Error CS0122: inaccessible outside T1.
            // T1.privateInt = 3;

            // Access is unlimited.
            T1.M1.publicInt = 1;

            // Accessible only in current assembly.
            T1.M1.internalInt = 2;

            // Error CS0122: inaccessible outside M1.
            //    T1.M1.privateInt = 3;

            // Error CS0122: inaccessible outside T1.
            //    T1.M2.publicInt = 1;

            // Error CS0122: inaccessible outside T1.
            //    T1.M2.internalInt = 2;

            // Error CS0122: inaccessible outside M2.
            //    T1.M2.privateInt = 3;
            // Keep the console open in debug mode.
            System.Console.WriteLine("Press any key to exit.");
            System.Console.ReadKey();
        }
    }
```

C# Language Specification

For more information, see the C# Language Specification. The language specification is the definitive source for C# syntax and usage.

See also

- C# Reference
- C# Programming Guide
- C# Keywords
- Access Modifiers
- Accessibility Levels
- Restrictions on Using Accessibility Levels
- Access Modifiers
- public
- private
- protected
- internal