C# Access Modifiers / Specifiers

C# Access modifiers or specifiers are the keywords that are used to specify accessibility or scope of variables and functions in the C# application.

C# provides five types of access specifiers.

1. Public
2. Protected
3. Internal
4. Protected internal
5. Private

We can choose any of these to protect our data. Public is not restricted and Private is most restricted. The following table describes about the accessibility of each.

Access Specifier	Description
Public	It specifies that access is not restricted.
Protected	It specifies that access is limited to the containing class or in derived class.
Internal	It specifies that access is limited to the current assembly.
protected internal	It specifies that access is limited to the current assembly or types derived from the containing class.
Private	It specifies that access is limited to the containing type.

Now, let's create examples to check accessibility of each access specifier.

1) C# Public Access Specifier

It makes data accessible publicly. It does not restrict data to the declared block.

Example

```
1. using System;
2. namespace AccessSpecifiers
```

```
3.  {
4.      class PublicTest
5.      {
6.          public string name = "Shantosh Kumar";
7.          public void Msg(string msg)
8.          {
9.              Console.WriteLine("Hello " + msg);
10.         }
11.     }
12.     class Program
13.     {
14.         static void Main(string[] args)
15.         {
16.             PublicTest publicTest = new PublicTest();
17.             // Accessing public variable
18.             Console.WriteLine("Hello " + publicTest.name);
19.             // Accessing public function
20.             publicTest.Msg("Peter Decosta");
21.         }
22.     }
23. }
```

Output:

```
Hello Shantosh Kumar
Hello Peter Decosta
```

2) C# Protected Access Specifier

It is accessible within the class and has limited scope. It is also accessible within sub class or child class, in case of inheritance.

Example

```
1.  using System;
2.  namespace AccessSpecifiers
3.  {
4.      class ProtectedTest
5.      {
6.          protected string name = "Shashikant";
7.          protected void Msg(string msg)
8.          {
9.              Console.WriteLine("Hello " + msg);
10.         }
11.     }
12.     class Program
13.     {
```

```
14.     static void Main(string[] args)
15.     {
16.         ProtectedTest protectedTest = new ProtectedTest();
17.         // Accessing protected variable
18.         Console.WriteLine("Hello "+ protectedTest.name);
19.         // Accessing protected function
20.         protectedTest.Msg("Swami Ayyer");
21.     }
22. }
23.}
```

Output:

```
Compile time error
'ProtectedTest.name' is inaccessible due to its protection level.
```

Example2

Here, we are accessing protected members within child class by inheritance.

```
1.  using System;
2.  namespace AccessSpecifiers
3.  {
4.      class ProtectedTest
5.      {
6.          protected string name = "Shashikant";
7.          protected void Msg(string msg)
8.          {
9.              Console.WriteLine("Hello " + msg);
10.         }
11.     }
12.     class Program : ProtectedTest
13.     {
14.         static void Main(string[] args)
15.         {
16.             Program program = new Program();
17.             // Accessing protected variable
18.             Console.WriteLine("Hello " + program.name);
19.             // Accessing protected function
20.             program.Msg("Swami Ayyer");
21.         }
22.     }
23.}
```

Output:

```
Hello Shashikant
Hello Swami Ayyer
```

3) C# Internal Access Specifier

The internal keyword is used to specify the internal access specifier for the variables and functions. This specifier is accessible only within files in the same assembly.

Example

```
1.  using System;
2.  namespace AccessSpecifiers
3.  {
4.      class InternalTest
5.      {
6.          internal string name = "Shantosh Kumar";
7.          internal void Msg(string msg)
8.          {
9.              Console.WriteLine("Hello " + msg);
10.         }
11.     }
12.     class Program
13.     {
14.         static void Main(string[] args)
15.         {
16.             InternalTest internalTest = new InternalTest();
17.             // Accessing internal variable
18.             Console.WriteLine("Hello " + internalTest.name);
19.             // Accessing internal function
20.             internalTest.Msg("Peter Decosta");
21.         }
22.     }
23. }
```

Output:

```
Hello Shantosh Kumar
Hello Peter Decosta
```

4) C# Protected Internal Access Specifier

Variable or function declared **protected internal** can be accessed in the assembly in which it is declared. It can also be accessed within a derived class in another assembly.

Example

```
1.  using System;
2.  namespace AccessSpecifiers
3.  {
4.      class InternalTest
5.      {
```

```
6.        protected internal string name = "Shantosh Kumar";
7.        protected internal void Msg(string msg)
8.        {
9.            Console.WriteLine("Hello " + msg);
10.       }
11.    }
12.    class Program
13.    {
14.        static void Main(string[] args)
15.        {
16.            InternalTest internalTest = new InternalTest();
17.            // Accessing protected internal variable
18.            Console.WriteLine("Hello " + internalTest.name);
19.            // Accessing protected internal function
20.            internalTest.Msg("Peter Decosta");
21.        }
22.    }
23. }
```

Output:

```
Hello Shantosh Kumar
Hello Peter Decosta
```

5) C# Private Access Specifier

Private Access Specifier is used to specify private accessibility to the variable or function. It is most restrictive and accessible only within the body of class in which it is declared.

Example

```
1. using System;
2. namespace AccessSpecifiers
3. {
4.     class PrivateTest
5.     {
6.         private string name = "Shantosh Kumar";
7.         private void Msg(string msg)
8.         {
9.             Console.WriteLine("Hello " + msg);
10.        }
11.    }
12.    class Program
13.    {
14.        static void Main(string[] args)
15.        {
16.            PrivateTest privateTest = new PrivateTest();
```

```
17.         // Accessing private variable
18.         Console.WriteLine("Hello " + privateTest.name);
19.         // Accessing private function
20.         privateTest.Msg("Peter Decosta");
21.     }
22.   }
23. }
```

Output:

```
Compile time error
'PrivateTest.name' is inaccessible due to its protection level.
```

C# Private Specifier Example 2

```
1.  using System;
2.  namespace AccessSpecifiers
3.  {
4.      class Program
5.      {
6.          private string name = "Shantosh Kumar";
7.          private void Msg(string msg)
8.          {
9.              Console.WriteLine("Hello " + msg);
10.         }
11.         static void Main(string[] args)
12.         {
13.             Program program = new Program();
14.             // Accessing private variable
15.             Console.WriteLine("Hello " + program.name);
16.             // Accessing private function
17.             program.Msg("Peter Decosta");
18.         }
19.     }
20. }
```

Output:

```
Hello Shantosh Kumar
Hello Peter Decosta
```