

**C#-7**

The screenshot shows a Microsoft Visual Studio interface with the following details:

- Title Bar:** Displays "PassbyValueDemo2.cs", "ClassDemo.cs", and "ConsoleApp".
- Toolbars:** Standard Visual Studio toolbars for "Common Tools" and "Edit" are visible.
- Code Editor:** The main window contains C# code for a console application. The code defines a class `ClassDemo` with a `Main` method that creates a `Student` object and prints its properties.
- Code Navigation:** A vertical green bar on the left indicates the current line of code being edited (line 10). A pink dashed rectangle highlights the entire class definition from the opening brace of `ClassDemo` to its closing brace.
- IntelliSense:** A tooltip labeled "ConsoleApp.ClassDemo" appears near the highlighted code, providing information about the current context.

```
5     using System.Threading.Tasks;
6
7     namespace ConsoleApp
8     {
9         class ClassDemo
10        {
11            static void Main()
12            {
13                Student student; //declare
14                student = new Student(1001); //
15                Console.WriteLine(student.id);
16                Console.WriteLine(student.name);
17
18                Student student1 = new Student(100, "Shashi");
19                Console.WriteLine(student1.id);
20                Console.WriteLine(student1.name);
21            }
22        }
23    }
24
```

PassbyValueDemo2.cs ClassDemo.cs X ConsoleApp

ConsoleApp

```
5     using System.Threading.Tasks;
6
7     namespace ConsoleApp
8     {
9         class ClassDemo
10        {
11            static void Main()
12            {
13                Student student; //declare
14                student = new Student(1001); //
15                Console.WriteLine(student.id);
16                Console.WriteLine(student.name);
17
18                Student student1 = new Student(100, "Shashi");
19                Console.WriteLine(student1.id);
20                Console.WriteLine(student1.name);
21            }
22        }
23    }
24
```

PassbyValueDemo2.cs ClassDemo.cs ConsoleApp

ConsoleApp

```
7  namespace ConsoleApp
8  {
9      class Student
10     {
11         public int id { get; set; }
12         public string name { get; set; }
13         public Student(int id)
14         {
15             if(id > 5) I
16             {
17                 id = id + 5;
18             }
19             this.id = id;
20         }
21     }
22     public Student(int id, string name)
23     {
24         this.id = id;
25         this.name = name;
26     }
}
```

10 references  
6 references  
5 references  
2 references  
1 reference

No issues found

here i'm passing

Now what will be the output ?

so i'm passing 1001

so as it is greater than 5, so it will add 5 to it and 1006 gets stored in  
the id

```
namespace ConsoleApp
{
    class ClassDemo
    {
        static void Main()
        {
            Student student; //declare
            student = new Student(1001); //
            Console.WriteLine(student.id);
            Console.WriteLine(student.name);

            Student student1 = new Student(100, "Shashi");
            Console.WriteLine(student1.id);
            Console.WriteLine(student1.name);
        }
    }
}
```

But when i come down here  
i'm passing 2 parameters here

The screenshot shows the code editor in Visual Studio with the file 'ConsoleApp.cs' open. The code defines a class 'Student' with two constructors. The first constructor takes an integer 'id' and performs a check if it's greater than 5, then adds 5 to it before assigning it to 'this.id'. The second constructor takes 'id' and 'name' and assigns them directly to 'this.id' and 'this.name' respectively. A red box highlights the second constructor, and a red arrow points from it to a text annotation.

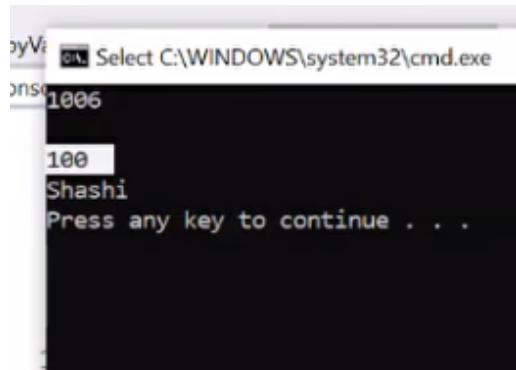
```
9     class Student
10    {
11        public int id { get; set; }
12        public string name { get; set; }
13        public Student(int id)
14        {
15            if(id > 5)
16            {
17                id = id + 5;
18            }
19            this.id = id;
20        }
21
22        public Student(int id, string name)
23        {
24            this.id = id;
25            this.name = name;
26        }
27    }
28
29
```

So it will come down to 2 argument constructor which is taking id and name

so here i'm directly taking id and name and storing it

Where is the business Rule ?  
there is no business rule

So, i was expecting it to be as 105 and shashi because of the fact that the above one has a bussiness rule but when i looked it in the output



```
100
Shashi
Press any key to continue . . .
```

you can see it is still 100, because i have implemented the rule in one of the constructor , so i need to copy paste the same logic in 2nd constructor also, so that like same bussiness rule can be implemented

```
PassbyValueDemo2.cs  ClassDemo.cs  ConsoleApp
ConsoleApp
public int id; get; set;
public string name { get; set; }

public Student(int id)
{
    if(id > 5)
    {
        id = id + 5;
    }
    this.id = id;
}

public Student(int id, string name)
{
    if (id > 5)
    {
        id = id + 5;
    }
    this.id = id;
    this.name = name;
}
```

Server Explorer    Toolbox

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help

Search

Debug Any CPU Start

121 % No issues found

Output Package Manager Console Error List Immediate Window

Build succeeded

The screenshot shows a Microsoft Visual Studio IDE window. The title bar says "PassbyValueDemo2.cs" and "ClassDemo.cs". The main editor area contains C# code for a class named "ConsoleApp.Student". Two identical blocks of code are highlighted with red boxes and arrows pointing to them from a text annotation.

```
public int id { get; set; }  
5 references  
public string name { get; set; }  
2 references  
public Student(int id)  
{  
    if(id > 5)  
    {  
        id = id + 5;  
    }  
    this.id = id;  
}  
1 reference  
public Student(int id, string name)  
{  
    if (id > 5)  
    {  
        id = id + 5;  
    }  
    this.id = id;  
    this.name = name;  
}
```

Now dont you think  
i'm writing too much of code here  
Same code again and again

So instead of doing this what i can do is

```
PassbyValueDemo2.cs*  X  ClassDemo.cs  ConsoleApp
ConsoleApp
public int id { get; set; }
5 references
public string name { get; set; }
2 references
public Student(int id)
{
    if(id > 5)
    {
        id = id + 5;
    }
    this.id = id;
}
1 reference
public Student(int id, string name)
{
    this.name = name;
}

0 references
class PassbyValueDemo2
{
```

i can say that id initialization  
already this guy is doing

So i can call this  
Constructor

Now How will you call the object what represent a object here -----  
**this** always represent the current object

The screenshot shows a Microsoft Visual Studio IDE window with the following details:

- Project:** ConsoleApp
- File:** PassbyValueDemo2.cs
- Code:** A C# class definition for `ConsoleApp.Student`. It includes a constructor that takes an `int id` and a `string name`, and a constructor that takes an `int id` and sets the `name` to "test".
- Completion List:** A tooltip is open over the `this()` keyword in the second constructor's parameter list, showing suggestions. The suggestion `Student(int id)` is highlighted.
- Annotations:** Handwritten-style text and arrows are overlaid on the screen:
  - so i'll put colon here and say this()**: An arrow points from this annotation to the `this()` keyword in the completion list.
  - look here it is already asking me student so i can pass id to this constructor**: Another arrow points from this annotation to the same `this()` keyword in the completion list.
- Toolbars and Menus:** Standard Visual Studio toolbars and menus are visible at the top and bottom of the interface.
- Status Bar:** The status bar at the bottom shows "121 %", "1", "0", and other standard status indicators.

```
11
12     public string name { get; set; }
13
14     public Student(int id)
15     {
16         if(id > 5)
17         {
18             id = id + 5;
19         }
20         this.id = id;
21     }
22
23     public Student(int id, string name):this(id)
24     {
25         this.name = name;
26     }
27
28
29 class PassbyValueDemo2
30 {
```

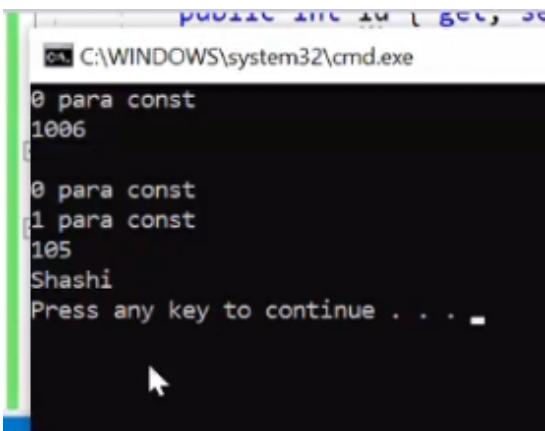
Now whenever i'm calling the second constructor which is taking 2 parameter by default it will call the constructor which is taking 1 parameter

here you can see this is executing this particular thing after the execution of this this particular constructor get called

PassbyValueDemo2.cs   X   ClassDemo.cs   ConsoleApp

ConsoleApp

```
11
12     public string name { get; set; }
13
14     public Student(int id)
15     {
16         if(id > 5)
17         {
18             id = id + 5;
19         }
20         this.id = id;
21         Console.WriteLine("0 para const");
22     }
23
24     public Student(int id, string name):this(id)
25     {
26         this.name = name;
27         Console.WriteLine("1 para const");
28     }
29
30 }
```



```
public int 2u L 866, 36
C:\WINDOWS\system32\cmd.exe
0 para const
1006
0 para const
1 para const
105
Shashi
Press any key to continue . . .
```

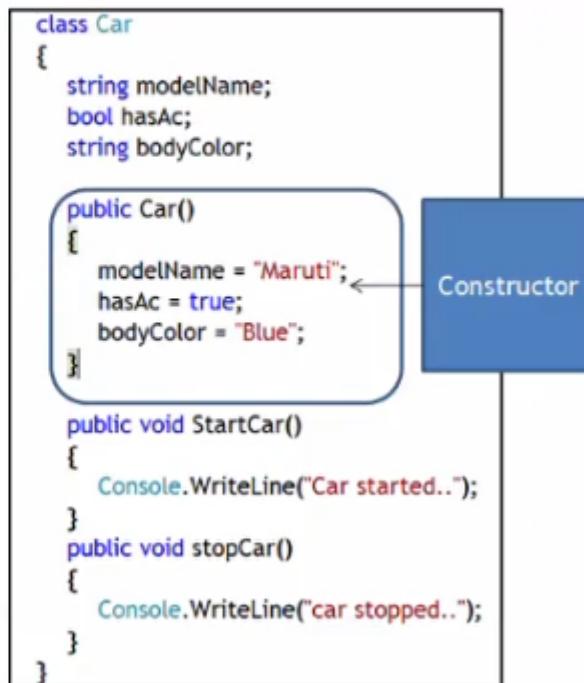
Now you can see as i told you, when i pass 2 parameter so it will call the 0 parameter 1st and then 1 parameter constructor gets called

### **So we are Chaining the Constructor here**

So, by using this you can always call the object or the constructor of the same class

## Constructor

- A constructor is a special method that is called to initialize default values to data members (fields) of a class
- It does not create object
- Constructor's name is same as that of the class.
- Constructor does not return anything



## Default Constructor

- Features of a default constructor
  - Public accessibility
  - Same name as the class
  - No return type—not even **void**
  - Expects no arguments
  - Initializes all fields to **zero, false or null**
- Constructor Example:

```
class Date
{
    public Date( )
    { ... }
```

## Explicitly Providing the Default Constructor

- Default constructor will be supplied by compiler
- But, the default constructor might be inappropriate
  - If so, do not use it; write your own!
  - If you are providing default constructor, then compiler will not supply the default constructor

```
class Date
{
    private int ccyy;
    private int mm;
    private int dd;

    public Date( )
    {
        ccyy = 1970;
        mm = 1;
        dd = 1;
    }
}
```

Remember Constructors can be private

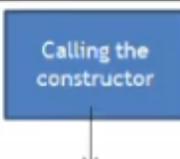
## Calling Constructor

- Initialize the object's fields with default values by using a constructor
  - Use the name of the class followed by parentheses followed by 'new' keyword
  - As default value, zero is assigned to numerical fields (int, double, long etc.), null is assigned to reference types (string or any class variables) and false is assigned to boolean types

```
class Program
{
    static void Main()
    {
        Car marutiCar = new Car();
        marutiCar.StartCar();
        marutiCar.stopCar();

    }
}
```

Calling the constructor



# Access Specifiers

This id for the Class level

## Access Specifiers for a Class

- Class has two access specifiers

Access Specifier	Description
internal	A class declared with this access specifier is accessible from anywhere within the containing program, but not accessible from outside that program
public	A class declared with this access specifier is accessible from anywhere within the containing program as well as accessible from outside that program

This is for the Method level

## Access Specifiers for Class Members

Access Specifier	Description
private	Access limited to the containing type. By default access is private
protected	Access is limited to the containing class or types derived from the containing class in the containing or any other program
internal	Access limited to the same or derived or any class, but in the containing program
protected internal	Available in the same or in the derived or any class in the containing program (like internal) and only in the derived classes in another program (like protected)
public	Accessible from anywhere, in the containing or derived or any other in the containing program or any other program

try this yourself, add new file AccessDemo.cs

i'm checking within the **same class** can i Access all of them, if i can access all of them you have to say yes if can't access you have to say no

So 1st

lets create a Access Class

AccessDemo.cs\* ✘ X ConsoleApp\*

ConsoleApp

ConsoleApp.AccessClass

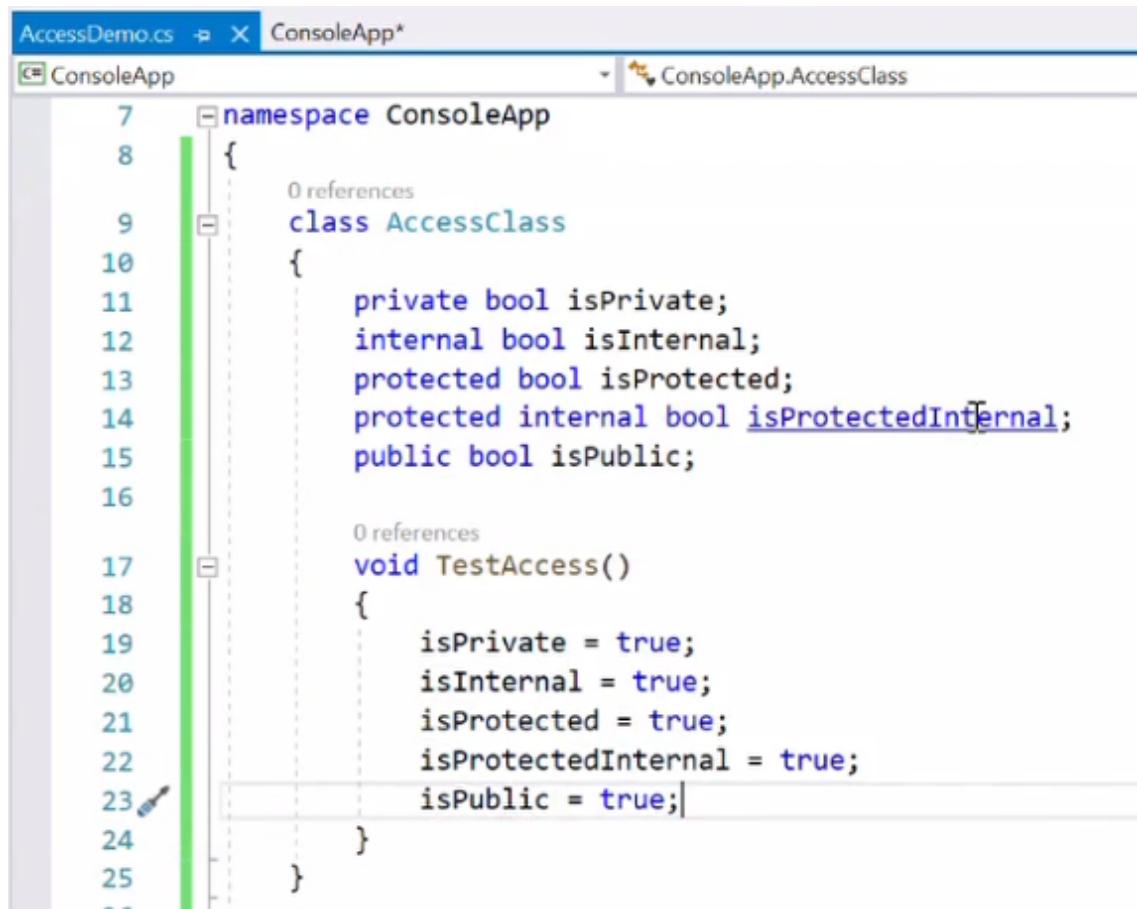
```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp
8  {
9      class AccessClass
10     {
11         private bool isPrivate;
12         internal bool isInternal;
13         protected bool isProtected;
14         protected internal bool isProtectedInternal;
15         public bool isPublic;
16     }
17
18     class AccessDemo
19     {
20     }
21
22 }
```

0 references

0 references

121 %

No issues found



```
AccessDemo.cs  X  ConsoleApp*
ConsoleApp      ConsoleApp.AccessClass

7  namespace ConsoleApp
8  {
9      class AccessClass
10     {
11         private bool isPrivate;
12         internal bool isInternal;
13         protected bool isProtected;
14         protected internal bool isProtectedInternal;
15         public bool isPublic;
16
17         void TestAccess()
18         {
19             isPrivate = true;
20             isInternal = true;
21             isProtected = true;
22             isProtectedInternal = true;
23             isPublic = true;
24         }
25     }
~~
```

meaning within the same class irrespective of weather it is private or public or whatever i can access them

	Class	Same Assembly	Same Assembly Inherited	Other Assembly Non-Inh	Other Assembly Inherited	Any
Private	Y					
Internal	Y					
Protected	Y					
Protected Internal	Y					
Public	Y					

## Same Assembly

The screenshot shows the Microsoft Visual Studio IDE interface. On the left is the code editor with the file `AccessDemo.cs` open. The code defines a `ConsoleApp` namespace containing two classes: `AccessClass` and `AccessDemo`. Both classes are highlighted with red boxes. The `AccessClass` box encloses lines 9 through 13, and the `AccessDemo` box encloses lines 14 through 17. A red arrow points from the text "here both the classes are present in same Assembly called ConsoleApp right" to the `AccessClass` box. Another red arrow points to the `AccessDemo` class definition below it. On the right side of the interface is the Solution Explorer window, which lists the project `ConsoleApp` and its files, including `AccessDemo.cs`, `App.config`, and several demo files like `ArrayDemo1.cs` and `BreakContinueDemo.cs`.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApp
8 {
9     class AccessClass
10    {
11    }
12}
13
14 class AccessDemo
15 {
16}
17
18}
```

here both the classes are present in same Assembly called ConsoleApp right

Assembly name:  
ConsoleApp

Target framework

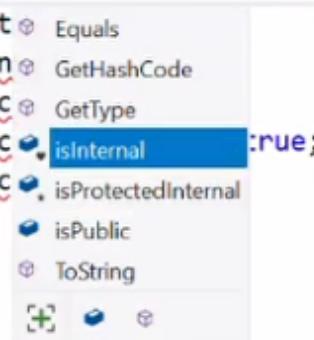
```
AccessDemo.cs*  X
ConsoleApp          ConsoleApp.AccessDemo
9   class AccessClass
10  {
11      private bool isPrivate;
12      internal bool isInternal;
13      protected bool isProtected;
14      protected internal bool isProtectedInternal;
15      public bool isPublic;
16
17      void TestAccess()
18      {
19          isPrivate = true;
20          isInternal = true;
21          isProtected = true;
22          isProtectedInternal = true;
23          isPublic = true;
24      }
25
26
27  class AccessDemo
28  {
29      void TestAccess()
30  }
```

```
AccessDemo.cs*  X  ConsoleApp  ConsoleApp.AccessDemo
1 class AccessDemo
2 {
3     0 references
4     void TestAccess()
5     {
6         0 references
7         isPrivate = true;
8         isInternal = true;
9         isProtected = true;
10        isProtectedInternal = true;
11        isPublic = true;
12    }
13 }
```

you can see i'm getting error  
because i'm not creating instance of  
this access class so you cant access them  
directly

So if you want to access any of these you have to create instance of this particular access class

```
0 references
class AccessDemo
{
    0 references
    void TestAccess()
    {
        AccessClass myClass = new AccessClass();
        myClass.
    }
}
```



if you write  
myclass. you can check what you can  
access

```
AccessDemo.cs  X
ConsoleApp          ConsoleApp.AccessDemo
20
21     isInternal = true,
22     isProtected = true;
23     isProtectedInternal = true;
24     isPublic = true;
25 }
26
27 class AccessDemo
28 {
29     void TestAccess()
30     {
31         AccessClass myClass = new AccessClass();
32         myClass.isInternal = true;|
33         myClass.isProtectedInternal = true;
34     myClass.isPublic = true;|
35     }
36 }
37
38
```

	Class	Same Assembly	Same Assembly Inherited	Other Assembly Non-Inh	Other Assembly Inherited	Any
Private	Y	N				
Internal	Y	Y				
Protected	Y	N				
Protected Internal	Y	Y	+			
Public	Y	Y				

# Same Assembly Inherited

```
AccessDemo.cs  X
ConsoleApp          ConsoleApp.AccessDemo
20
21     isInternal = true,
22     isProtected = true;
23     isProtectedInternal = true;
24     isPublic = true;
25 }
26
27 class AccessDemo : AccessClass
28 {
29     void TestAccess()
30     {
31         isInternal = true;
32         isProtected = true;
33         isProtectedInternal = true;
34         isPublic = true;
35     }
36 }
37
38 }
```

this is how you inherit

So this is the advantage of Inheritance  
If you are inheriting your class

So from the base class what i can access -> internal members, protected members, protected internal members, and public members

You Cannot access your private members here

	Class	Same Assembly	Same Assembly Inherited	Other Assembly Non-Inh	Other Assembly Inherited	Any
Private	Y	N	N			
Internal	Y	Y	Y			
Protected	Y	N	Y			
Protected Internal	Y	Y	Y			
Public	Y	Y	Y			

## Other Assembly Non-Inherited

other Assembly means this class is not present here but whereas present in some other assembly

```
AccessDemo.cs  X
ConsoleApp          ConsoleApp.AccessClass

1 reference
9   class AccessClass
10  {
11    private bool isPrivate;
12    internal bool isInternal;
13    protected bool isProtected;
14    protected internal bool isProtectedInternal;
15    public bool isPublic;
16
17    0 references
18    void TestAccess()
19  }
20
21
22  0 references
23  class AccessDemo : AccessClass
24  {
25    0 references
26    void TestAccess()
27    {
28      isInternal = true;
29      isProtected = true;
30      isProtectedInternal = true;
31      isPublic = true;
32    }
33
34  }
35
36
```

121 %    No issues found

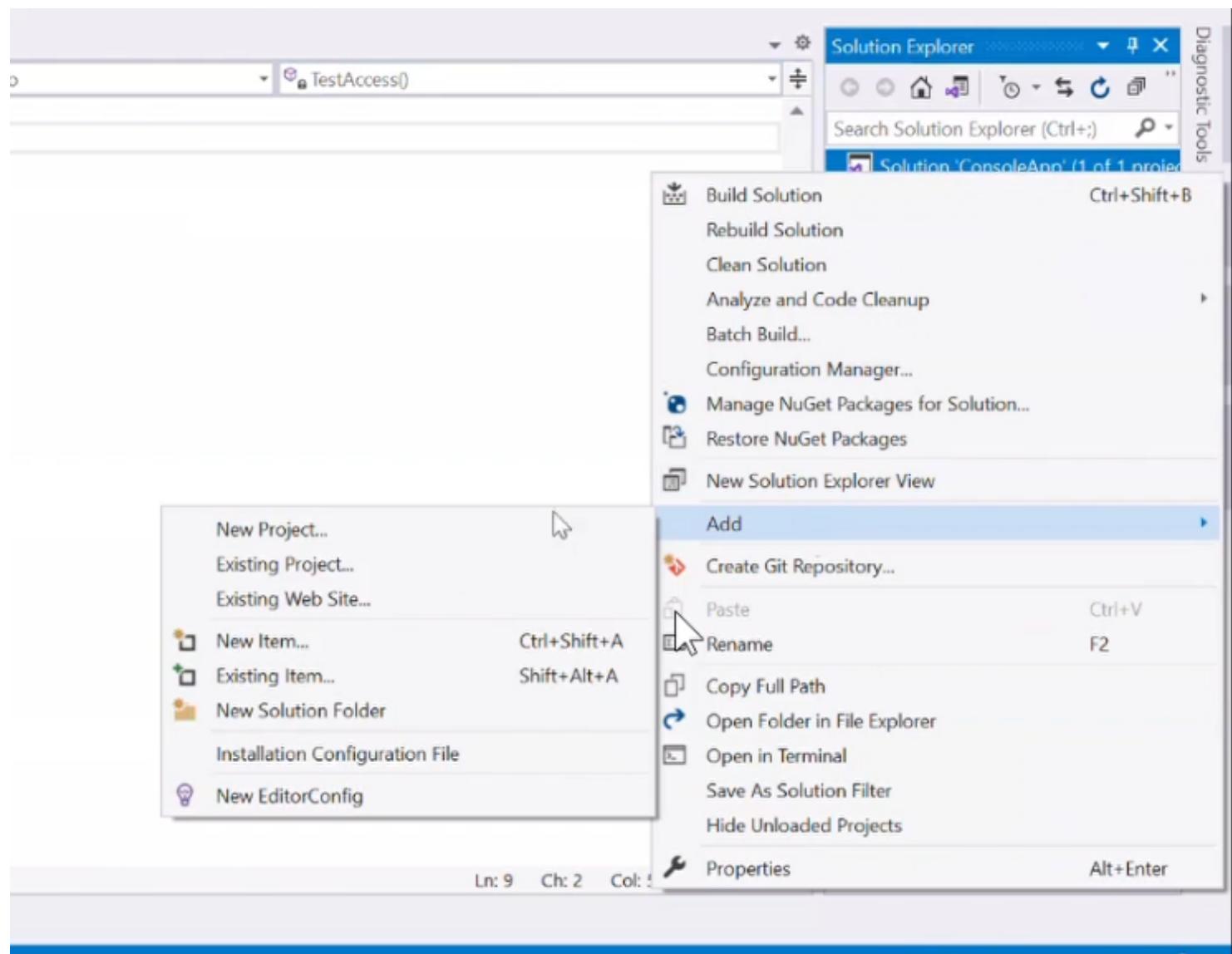
Output Package Manager Console Error List Immediate Window

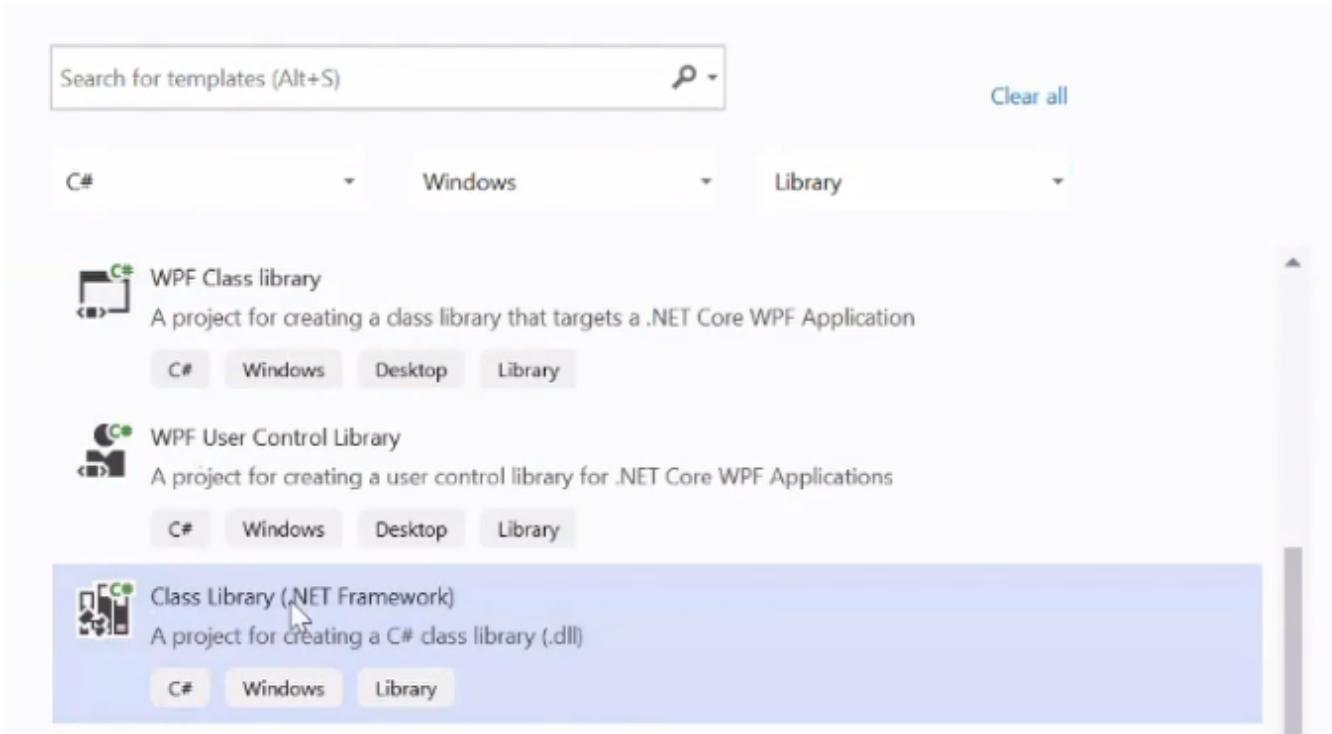
Ready

Type here to search

other Assembly means this class is not present here but whereas present in some other assembly / Project

right click on solution ---> add new project---->this time i'll not take it as a console  
rather i'll take it as a liabrary





## Class Library (.NET Framework)

C#

Windows

Library

Project name

AccessTest

Location

C:\Shashi\Training\Clients\2021\Mphasis\Oct\_Batch\Datas\ConsoleApp\ConsoleApp

...

Framework

.NET Framework 4.5

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays a code editor with two tabs: "Class1.cs\*" and "AccessDemo.cs\*". The "AccessDemo.cs\*" tab is active, showing the following C# code:

```
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace AccessTest
8  {
9      class AccessClass
10     {
11         private bool isPrivate;
12         internal bool isInternal;
13         protected bool isProtected;
14         protected internal bool isProtectedInternal;
15         public bool isPublic;
16
17         void TestAccess()
18         {
19             isPrivate = true;
20             isInternal = true;
21             isProtected = true;
22             isProtectedInternal = true;
23             isPublic = true;
24         }
25     }
26 }
```

The code editor includes standard VS features like line numbers, code folding, and status bars at the bottom indicating "Ln: 25 Ch: 3 Col: 6 TABS CRLF".

To the right of the code editor is the "Solution Explorer" window, which lists the solution structure:

- Solution 'ConsoleApp' (2 of 2 projects)
  - AccessTest
    - Properties
    - References
    - Class1.cs
  - ConsoleApp

The "Diagnostic Tools" ribbon tab is selected. The taskbar at the bottom shows the Windows Start button, a search bar, and various pinned application icons.

Now, this and

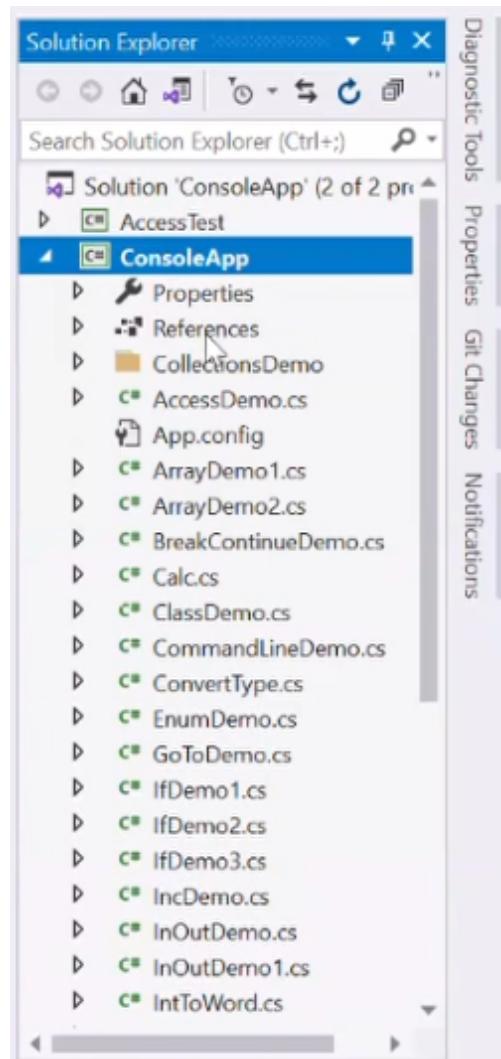
A screenshot of the Microsoft Visual Studio IDE interface. The main window shows the code editor with a file named 'AccessDemo.cs' open. The code defines a class 'AccessDemo' that inherits from 'AccessClass'. Inside the class, there is a method 'TestAccess()' containing four boolean assignments: 'isInternal', 'isProtected', 'isProtectedInternal', and 'isPublic', all set to true. The 'Solution Explorer' pane on the right displays the project structure for 'ConsoleApp', which includes two projects: 'AccessTest' and 'ConsoleApp', with files like 'Class1.cs' and 'ConsoleApp.cs'. The 'Taskbar' at the bottom shows various pinned application icons, and the system tray indicates the date and time as 22-10-2021 11:55 AM.

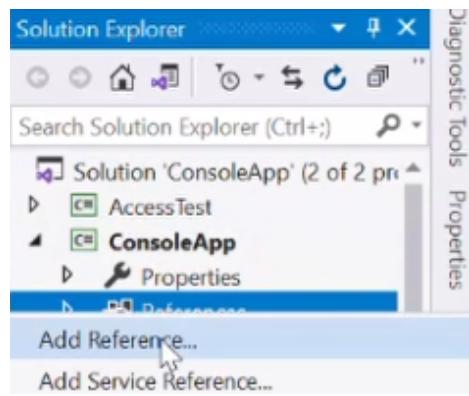
```
8
9
10
11 class AccessDemo : AccessClass
12 {
13     void TestAccess()
14     {
15         isInternal = true;
16         isProtected = true;
17         isProtectedInternal = true;
18         isPublic = true;
19     }
20 }
21
22
```

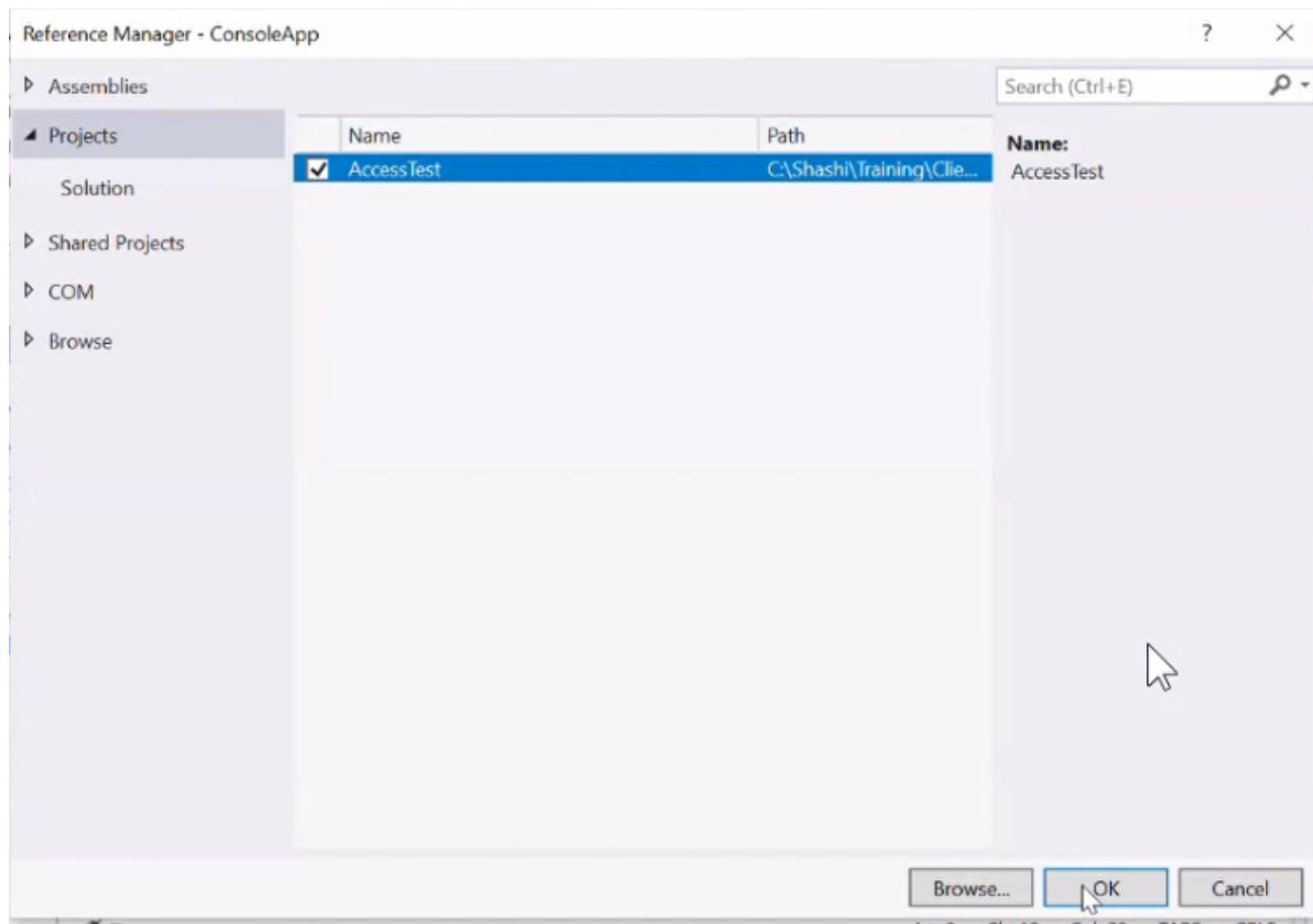
this,

Both are 2 different places

so to access AccessTest in ConsoleApp we need to add reference to them  
right click on reference







Now i have added a reference i'll come back, and create a reference of it ,  
it will act just like another name space ,

Now my AccessClass is present inside AccessTest project

The screenshot shows the Visual Studio IDE interface. On the left is the code editor for 'AccessDemo.cs' with the following code:ConsoleApp
AccessDemo.cs\* X
ConsoleApp.AccessDemo
TestAccess()

8
9
10
11 class AccessDemo
12 {
13 void TestAccess()
14 {
15 AccessClass
16 Generate property 'AccessDemo.AccessClass'
17 Generate field 'AccessDemo.AccessClass'
18 Generate read-only field 'AccessDemo.AccessClass'
19 Generate local 'AccessClass'
20 Generate parameter 'AccessClass'
21 Use expression body for methods
22 }
23
24 public object AccessClass { get; private set; }
25
26 void TestAccess()
27 }A tooltip from the code editor is displayed over the 'AccessClass' identifier at line 15, showing the following options:

- Generate property 'AccessDemo.AccessClass'
- Generate field 'AccessDemo.AccessClass'
- Generate read-only field 'AccessDemo.AccessClass'
- Generate local 'AccessClass'
- Generate parameter 'AccessClass'
- Use expression body for methods

The tooltip also includes a preview of the generated code:

```
CS0103 The name 'AccessClass' does not exist in the current context
Lines 12 to 13
{
    public object AccessClass { get; private set; }

    void TestAccess()
```

The Solution Explorer on the right shows the project structure:

```
Solution Explorer
Search Solution Explorer (Ctrl+Shift+F)
Solution 'ConsoleApp' (2 of 2 prj)
    AccessTest
    ConsoleApp
        Properties
        References
            Analyzers
            AccessTest
            Microsoft.CSharp
            System
            System.Core
            System.Data
            System.Data.DataSetE
            System.Net.Http
            System.Xml
            System.Xml.Linq
            CollectionsDemo
            AccessDemo.cs
            App.config
            ArrayDemo1.cs
            ArrayDemo2.cs
            BreakContinueDemo.cs
            Calc.cs
            ClassDemo.cs
            CommandLineDemo.cs
```

but it is saying to generate the class

Why because in AccessClass i don't have any public class

The screenshot shows a Visual Studio code editor with the file `AccessClass.cs` open. The code defines a class `AccessClass` with various access modifiers for its fields. A green vertical bar on the left indicates code coverage. A yellow lightbulb icon on line 9 suggests a potential issue. The status bar at the bottom shows "No issues found".

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AccessTest
{
    class AccessClass
    {
        private bool isPrivate;
        internal bool isInternal;
        protected bool isProtected;
        protected internal bool isProtectedInternal;
        public bool isPublic;

        void TestAccess()
        {
            isPrivate = true;
            isInternal = true;
            isProtected = true;
            isProtectedInternal = true;
            isPublic = true;
        }
    }
}
```

SO to access this AccessClass outside you need to make it public

AccessClass.cs X AccessDemo.cs\*

AccessTest

AccessTest.AccessClass

```
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace AccessTest
8  {
9      public class AccessClass
10     {
11         private bool isPrivate;
12         internal bool isInternal;
13         protected bool isProtected;
14         protected internal bool isProtectedInternal;
15         public bool isPublic;
16
17         void TestAccess()
18         {
19             isPrivate = true;
20             isInternal = true;
21             isProtected = true;
22             isProtectedInternal = true;
23             isPublic = true;
24         }
25     }
26 }
```

121 % No issues found

Output Package Manager Console Error List ... Immediate Window

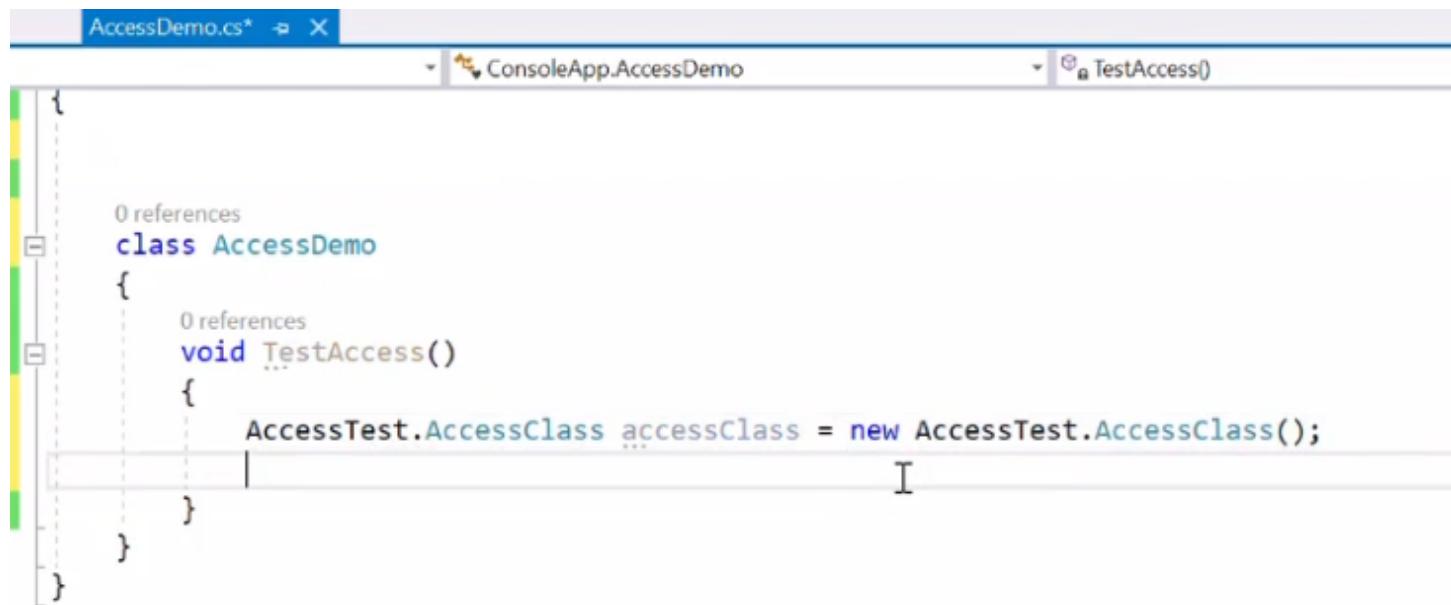
Now look at this i have AccessClass

AccessDemo.cs\* X

ConsoleApp.AccessDemo

```
{  
    0 references  
    class AccessDemo  
    {  
        0 references  
        void TestAccess()  
        {  
            AccessTest.  
        }  
    }  
}
```

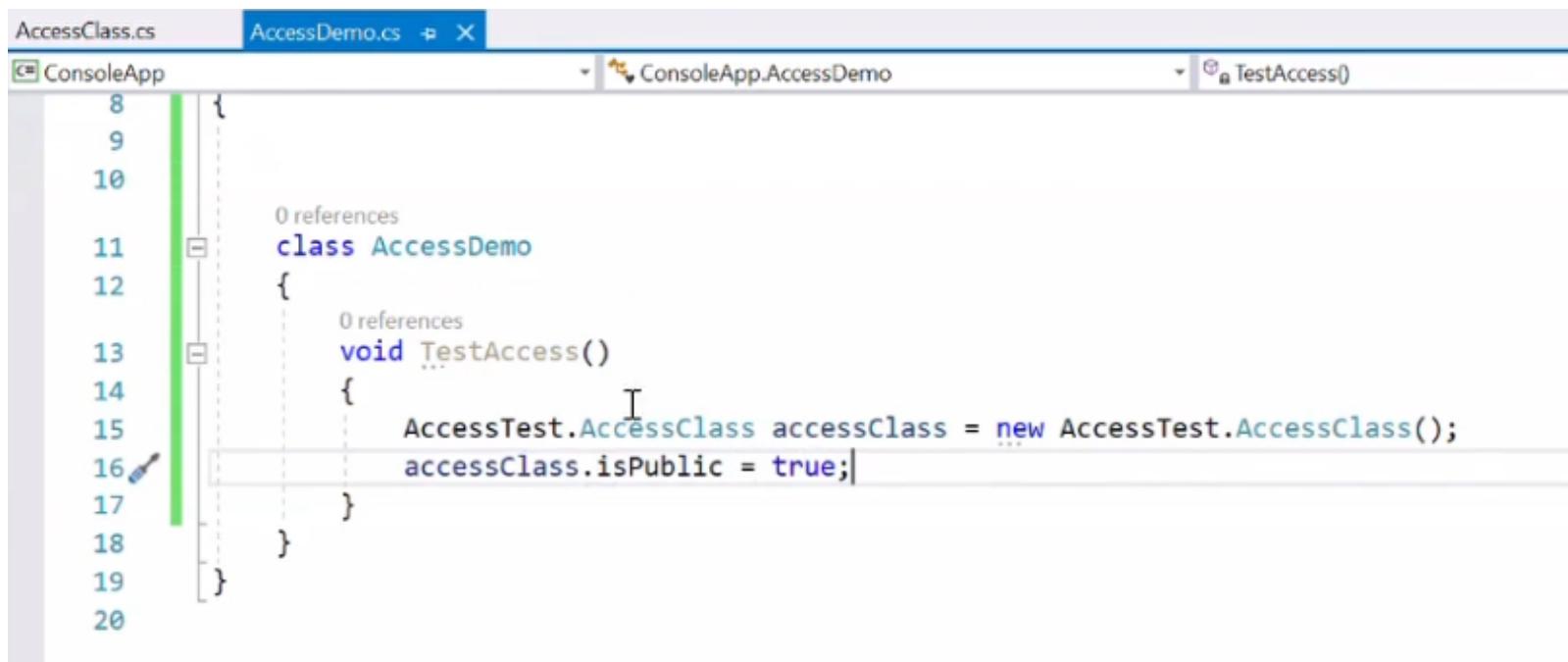
AccessClass class AccessTest.AccessClass



The screenshot shows the Microsoft Visual Studio IDE interface. The title bar displays "AccessDemo.cs\*" and the solution name "ConsoleApp.AccessDemo". The current tab is "TestAccess()". The code editor contains the following C# code:

```
0 references
class AccessDemo
{
    0 references
    void TestAccess()
    {
        AccessTest.AccessClass accessClass = new AccessTest.AccessClass();
    }
}
```

So if you are creating an instance and it is not inherited



```
8
9
10
11 class AccessDemo
12 {
13     void TestAccess()
14     {
15         AccessTest.AccessClass accessClass = new AccessTest.AccessClass();
16         accessClass.isPublic = true;
17     }
18 }
19
20
```

	Class	Same Assembly	Same Assembly Inherited	Other Assembly Non-Inh	Other Assembly Inherited	Any
Private	Y	N	N	N		
Internal	Y	Y	Y	N		
Protected	Y	N	Y	N		
Protected Internal	Y	Y	Y	N		
Public	Y	Y	Y	Y		

## Other Assembly Inherited

Now lets try to put colon and inherit

A screenshot of a C# code editor showing a class definition. The code is as follows:

```
8
9
10
11 class AccessDemo : AccessTest.AccessClass
12 {
13     void TestAccess()
14     {
15     }
16 }
17
18 }
```

The code editor interface includes tabs for "AccessClass.cs" and "AccessDemo.cs\*", a status bar showing "ConsoleApp.AccessDemo", and a vertical ruler on the left side.

Now as soon as i try to inherit in some other assembly

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar says "AccessDemo.cs\*" and the solution name is "ConsoleApp.AccessDemo". The code editor displays the following C# code:

```
{  
    0 references  
    class AccessDemo : AccessTest.AccessClass  
    {  
        0 references  
        void TestAccess()  
        {  
            isProtected = true;  
            isProtectedInternal = true; I  
            isPublic = true;  
        }  
    }  
}
```

These 3 things can be accessed

	Class	Same Assembly	Same Assembly Inherited	Other Assembly Non-Inh	Other Assembly Inherited	Any
Private	Y	N	N	N	N	
Internal	Y	Y	Y	N	N	
Protected	Y	N	Y	N	Y	
Protected Internal	Y	Y	Y	N	Y	
Public	Y	Y	Y	Y	Y	

now you have to check for any

