**Part 49 - C# Tutorial - Access Modifiers - Private, Public, Protected**

In this session we will learn, about  different access modifiers that can be applied for type members.   
  
  
[To better understand this part, please make sure you have already watched video on, Difference between types and type members.](http://csharp-video-tutorials.blogspot.com/2012/07/part-48-c-tutorial-difference-between.html)  
  
  
**There are 5 different access modifiers in c#.**  
**1.** Private  
**2.** Protected  
**3.** Internal  
**4.** Protected Internal  
**5.** Public  
  
  
**Private members** are available **only with in the containing type**, where as public members are available **any where**. There is **no restriction**.

In the example below, **\_id**is private, so this member is only available with in the Customer class (Containing Type). It is a compile time error to access \_id outside of the Customer Class.  
  
The following line in the MainClass will generate a compiler error stating, 'Customer.\_id' is inaccessible due to its protection level.  
CustomerInstance.\_id = 101;  
  
  
On the other hand, since Id is a public member, you can access this member any where, even outside of the Customer class. Infact, we invoke the Id property of the Customer class, in the Main() method as shown below.  
CustomerInstance.Id = 101;  
  
  
**Example for Private and Public Members:**  
public class Customer  
{  
    private int \_id;  
  
  
    public int Id  
    {  
        get  
        {  
            return \_id;  
        }  
        set  
        {  
            \_id = value;  
        }  
    }  
}  
  
  
public class MainClass  
{  
    private static void Main()  
    {  
        Customer CustomerInstance = new Customer();  
        CustomerInstance.Id = 101;  
          
        // Compiler Error : 'Customer.\_id' is inaccessible due to its protection level  
        // CustomerInstance.\_id = 101;  
    }  
}  
  
  
**Protected Members** are available, with in the **containing type** and to the types that **derive from the containing type**. Let me explain with an example.   
  
  
**Protected Access Modifier Example:**  
using System;  
public class Customer  
{  
    protected int ID = 101;  
  
  
    public void PrintID()  
    {  
        //Protected member ID is accessible with in Customer class  
        Console.WriteLine(this.ID);  
    }  
}  
public class CorporateCustomer : Customer  
{  
    public void PrintCustomerID()  
    {  
        CorporateCustomer corporateCustomerInstance = new CorporateCustomer();  
        // Can access the base class protected instance member using the derived class object  
        Console.WriteLine(corporateCustomerInstance.ID);  
        // Can access the base class protected instance member using this or base keyword  
        Console.WriteLine(this.ID);  
        Console.WriteLine(base.ID);  
    }  
}  
public class RetailCustomer  
{  
    public void PrintCustomerID()  
    {  
        RetailCustomer retailCustomerInstance = new RetailCustomer();  
        //RetailCustomer class is not deriving from Customer class, hence it is an error

    //to access Customer class protected ID member, using the retailCustomerInstance  
        //Console.WriteLine(retailCustomerInstance.ID); //Error  
          
        //Both these below lines also produce the same Error  
        //Console.WriteLine(this.ID); // Error  
        //Console.WriteLine(base.ID); // Error  
    }  
}  
  
  
Customer class defines a protecetd member ID. CorporateCustomer class derives from the Customer class, so protected ID member is accessible in the Customer class (Containg Type) and also from the CorporateCustomer class (Derived Type).   
  
  
With in the PrintID() method in the Customer class, Protected member ID is accessible.  
Console.WriteLine(this.ID);  
  
  
**There are 3 ways to access, the base class protected member in the derived class as shown below.**  
**1.** Using the derived class object.  
Console.WriteLine(corporateCustomerInstance.ID);  
**2.** Using the this keyword  
Console.WriteLine(this.ID);  
**3.** Using the base keyword  
Console.WriteLine(base.ID);  
  
  
On the other hand, RetailCustomer class is not deriving from Customer class, hence it's a compile time error to access Customer class protected ID member.

