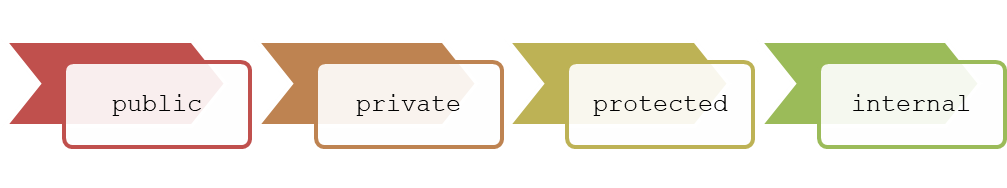
**Access Modifiers In C# Programming**

**Access Modifiers**

C# provides you with access modifiers that allow you to specify which classes can access the data members of a particular class.

**In C#, there are four commonly used access modifiers.**



**These are described as follows:**

**public:** The public access modifier provides the most permissive access level.  
The members declared as public can be accessed anywhere in the class as well as from other classes.

**The following code declares a public string variable called Name to store the name of the person which can be publicly accessed by any other class:**

class Employee

{

// No access restrictions.

public string Name = “Adil”;

}

**private:** The private access modifier provides the least permissive access level. Private members are accessible only within the class in which they are declared.

Following code declares a variable called \_salary as private, which means it cannot be accessed by any other class except for the Employee class.

class Employee

{

// Accessible only within the class

private float \_salary;

}

**protected:**  
The protected access modifier allows the class members to be accessible within the class as well as within the derived classes.

The following code declares a variable called Salary as protected, which means it can be accessed only by the Employee class and its derived classes:

class Employee

{

// Protected access

protected float Salary;

}

**internal:** The internal access modifier allows the class members to be accessible only within the classes of the same assembly. An assembly is a file that is automatically generated by the compiler upon successful compilation of a .NET application. The code declares a variable called NumOne as internal, which means it has only assembly-level access.

public class Sample

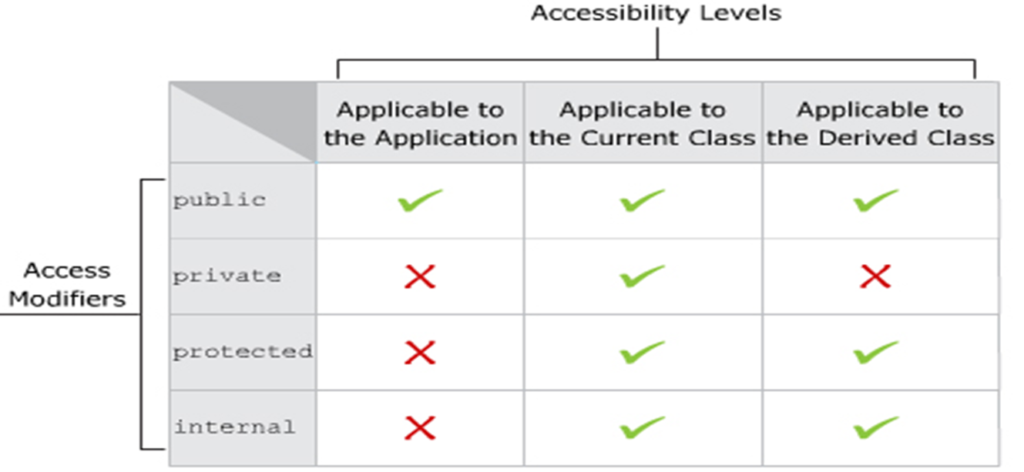
{

// Only accessible within the same assembly

internal static intNumOne = 3;

}

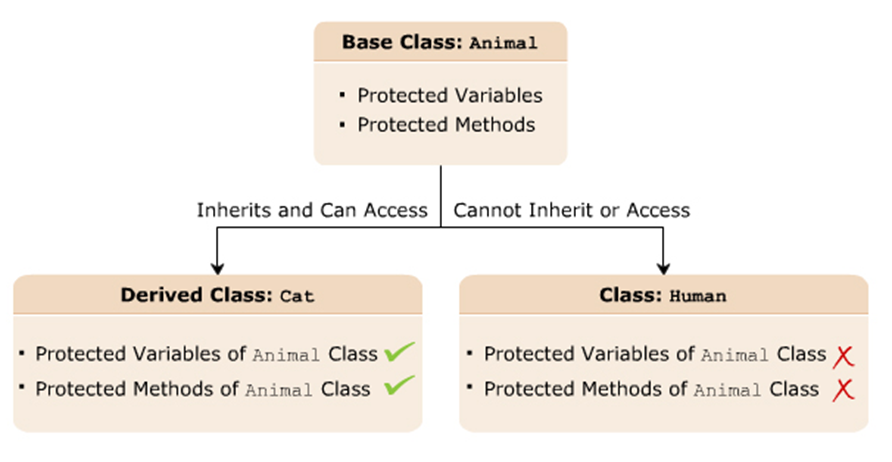
**The following figure displays the various accessibility levels:**



**Protected Access Modifier**

* The **protected** access modifier protects the data members that are declared using this modifier.
* The **protected** access modifier is specified using the protected keyword.
* Variables or methods that are declared as protected are accessed only by the class in which they are declared or by a class that is derived from this class.

**The following figure displays an example of using the protected access modifier:**

**Protected Access Modifier**

**The following syntax declares a protected variable:**

protected <data\_type> <VariableName>;

**where,**  
**data\_type**: Is the data type of the data member.  
**VariableName**: Is the name of the variable.

**The following syntax declares a protected method:**

protected <return\_type> <MethodName>(argument\_list);

**where,**  
**return\_type**: Is the type of value the method will return.  
**MethodName**: Is the name of the method.  
**argument\_list**: Is the list of parameters.

**The following code demonstrates the use of the protected access modifier:**

class Animal

{

protected string Food;

protected string Activity;

}

class Cat:Animal

{

static voidMain(String[]args)

{

cat objCat=newCat();

objCat.Food="Mouse";

objCat.Activity="lazearound";

Console.WriteLine("The Cat loves to eat"+objCat.Food+".");

Console.WriteLine("The Catloves to"+objCat.Activity+".");

}

}

In Above code:

* Two variables are created in the class Animal with the protected keyword.
* The class Cat is inherited from the class Animal.
* The instance of the class Cat is created that is referring the two variables defined in the class Animal using the dot (.) operator.
* The protected access modifier allows the variables declared in the class Animal to be accessed by the derived class Cat.

**Output**

Cat loves to eat Mouse.

The Cat loves to laze around