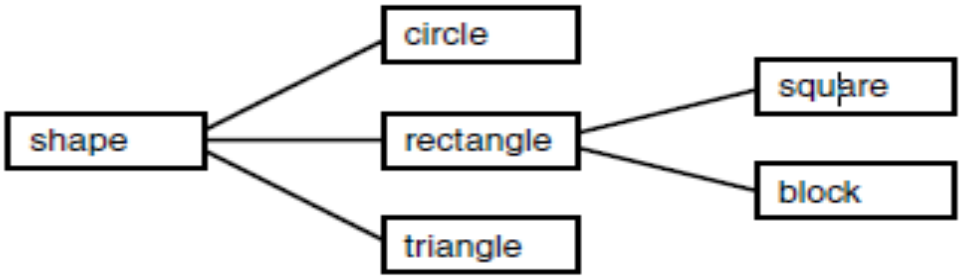
Inheritance

- Inheritance allows a class to be derived from another class known as the **base class**. The class that you derive is called **the derived class**.

تَسْمحُ الوراثة باشتقاق صنف مِنْ صنفِ آخرِ والمعروف بالصنفِ **الأساسي**.و إنّ الصنفَ الذي يَشتقُّ يُدْعَى الصنفَ **المُشتَقَّ**.

The **concept** of inheritance gives us the ability to create a new class based on an existing class (reuse the data and methods of a class by the derived class)
A number of new classes can inherit from an original class; however, only one class can be inherited from.

يمكن أن يرث عدد من الاصناف من صنف واحد ولكن لايمكن لصنف واحد أن يرث من أكثر من واحد.

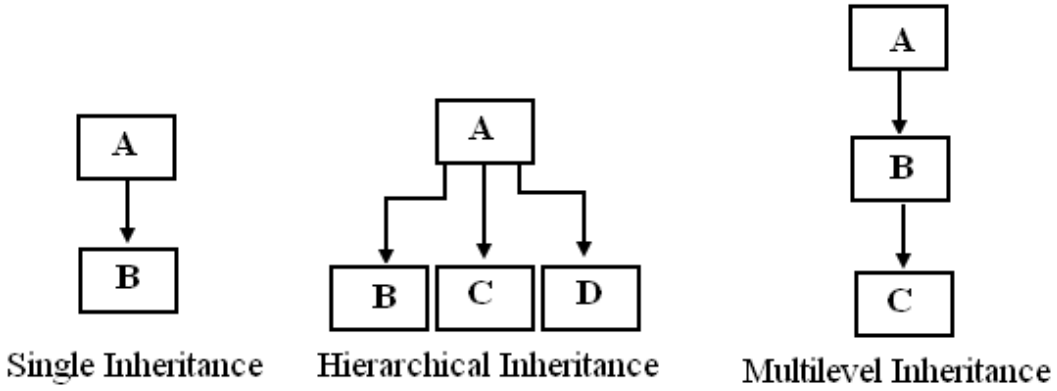


Inheriting relationships

Several basic terms are commonly used with inheritance:

Base class	The original class.
Parent class	Another name for a base class.
Derived class	A new class, created by inheriting from a base class.
Child class	Another name for a derived class.
Single inheritance	C# supports only **single inheritance**. A derived class created from only one base class. .
Multiple inheritance	C# does not support **multiple inheritance.** Aderived class created from two or more base classes.

Forms (Types) of Inheritance



Implementing Inheritance

The syntax of inheritance is:

```
class   <derived class name> : <base class name>
 {
        //   members of derived class
 }
```

The colon (:) is used to indicate that a class is derived from an existing class. For example: **class C2 : C1**

Here, **C1** is the base class of **C2**, and therefore, **C2** inherits all members of **C1**. In this case, **C2** is the derived class of **C1**.

Example:

```
class E
 {
     public void E_Name()
       { - - - - - - - - - - - - -   }
 }
class S : E
 {
    public void S_Name ()
       {  - - - - - - - - - - -     }
  }
class B
{          static void Main()
            {      S obj = new S( );
                   obj . E_Name( ) ;
```

```
            obj . S_Name ( ) ;
        }
}
```

This code declares two classes, **E** and **S**. The **S** class is inherited from the **E** class and, therefore, inherits the method declared by the base class. An instance of the derived class is created to access the members of the base class.

Protect members:

A base class s protected members can be accessed by members of that class & by members of its derived class.

Notes :

١. الـ Private members لاتورث، أي أن المُتغيرات والدوال تكون مُعروفة على مُستوى الـ class الذي تنتمي اليه فقط.

٢. الـ Protected members يتم الوصول لهل فقط من الـ members للـ class المعرفة فيه وكذلك من الـ members لل class المشتق من الـ class الذي عرفت فيه. أي أن المُتغيرات والدوال تكون مُعروفة على مُستوى الـ class الذي تنتمي اليه و الـ class الذي ترث منه فقط

٣. الـ Public members يمكن الوصول لها من أي مكان. أي أن المُتغيرات والدوال تكون مُعروفة على مُستوى جميع اجزاء البرنامج.

Example:

```
class B      //base class
   {
      protected int i, j;
      public void set(int x, int y)
      {
         i = x;
         j = y;
      }
      public void show()
      {
         Console.WriteLine("{0}  {1}", i, j);
      }
         class D:B      //derived class
```

```
    {
        int k;
        public void setk()
        {
            k =i*j;
        }
        public void showk()
        {
            Console .WriteLine ("{0}",k);
        }
    }
    static void Main(string[] args)
    {
        D ob=new D();
        ob.set (5,6);
        ob .show ();
        ob.setk ();
        ob .showk ();

    }
}
```

The Keyword new

The derived class implicitly inherits all the members of the base class. However, **you can hide any method of the base class** so that the derived class cannot access the method. To do so, you declare another method by a signature that is same as that of the base class method. When a compiler finds such a method, it generates a warning. To suppress this warning, you use the **new** keyword. This makes the base class method inaccessible to the derived class.

ممكن تعريف دالة في الـ class المشتق لها نفس الاسم لدالة في الـ class الأساس. في هذه الحالة يجب إخفاء الدالة التي تحمل نفس الاسم في الـ class الأساس عن الـ class المشتق لتجنب ظهور رسالة تحذير. رسالة التنبيه تظهر فقط للتنبيه لاخفاء الدالة بالـ class الأساس ولكي يتم الاخفاء نستخدم الكلمة المفتاحية new مع الدالة المتكرر اسمها في الـ class المشتق.

In the above example, if you need to declare a method with the name **E_Name**() in the **S** class, you need to include the new keyword in the method declaration statement. The next example uses the **new** keyword.

```
class S : E
{
```

```
    new public void E_Name()
      { - - - - - - - - - - -     }

     public void S_Name ()
      {  - - - - - - - - - - -     }

}
```

In the following example, the class ExtendedCounter inherits from the class Counter. The derived method Tick reuses the same method of its base class by invoking the Tick method of its parent. The keyword new is required in the derived Tick method to hides the one in the base class. To avoid a recursive call, the invocation (الاستدعاء) of **Tick** is preceded (يُسبَق) by the keyword **base**.

```
class Counter {
public bool Tick( ) { ... }
...
}
class ExtendedCounter : Counter {
        public new bool Tick( )
        {
        ...
        base.Tick( ); // Reuse the Tick method from Counter
        ...
        }
}
```

Examples for inheritance:

Example1 :

```
using System;
class Point
{
   protected int x;
   protected int y;
}

class DerivedPoint: Point
{
   static void Main()
   {
      DerivedPoint dp = new DerivedPoint();
```

```
        // Direct access to protected members:
        dp.x = 10;
        dp.y = 15;
        Console.WriteLine("x = {0}, y = {1}", dp.x, dp.y);
    }
}
```

Example2 :

```
using System;
namespace Inheritance
{
class Program
{
  class Person
   {
     public void showDetails(string Name, int Age)
       {
          Console.WriteLine(" My Name is: " + Name + " and My Age is: "
+ Age);
       }
   }
   class Programmer : Person
    {
       public void Print1()
        {
            Console.Write("Iam a Programmer" );
        }
     }
    class Engineer : Person
     {
       public void Print2()
        {
            Console.Write("Iam an Engineer " );
        }
     }
    class Doctor : Person
    {
       public void Print3()
         {
            Console.Write("Iam a Doctor " );
         }
```

```
        }
static void Main(string[] rags)
{
    Programmer Pro = new Programmer();
    Engineer Eng = new Engineer();
    Doctor Doc = new Doctor();
    Pro.Print1();
    Pro.showDetails("Hassan Kareem", 23);
    Eng.Print2();
    Eng.showDetails("Ahmad Khader", 25);
    Doc.Print3();
    Doc.showDetails("Ameen Ali", 24);
    Console.ReadKey();
}
}
}
```

Example3 :

```
public class student
{
  protected  int roll_number;

  public void get_number(int a)
    {   roll_number = a;   }

  public void put_number( )
    {      Console .WriteLine ("Roll Number: {0}" , roll_number);  }
}
public class test : student  // first level derivation
{
   protected  double  sub1;
   protected  double  sub2;
   public void get_marks(double x, double  y)
    {    sub1 = x;  sub2 = y;  }

   public void put_marks( )
    {
Console .WriteLine ("Mark in SUB1 = {0}, Mark in SUB2 = {1}
",sub1, sub2);
    }
}
```

```
    public class result : test  // second level derivation
    {
    double  total;           // private by default
    public  void display ( )
    {
         total = sub1 + sub2;
         put_number( );
         put_marks( );
         Console .WriteLine ("Total = {0}", total);
    }
static void Main(string[] args)
{
       result   student_1 = new result( );
       student_1.get_number(111);
       student_1.get_marks(75.0,59.5);
       student_1.display( );
    }
}
```

H.W.

Write C# program using the inheritance feature of OOP:
Consider the base class **Shape** has the following members:
1- Two **protected integer** variables **Width** and **Length**
2- **A set method** of width and Length members.
The derived class **Rectangle** has the following member:
1- **getArea method** to find the area of the rectangle.