Polymorphism

There are **two powerful aspects** (سمات) to inheritance:

a- **Code reuse**.
b- **Polymorphism**: ***Poly*** means **many** and ***morph*** means **form**. Thus, polymorphism refers to being able to use **many forms** of a type without regard to the details**, (**i.e. polymorphism refers to the ability of a single type or class to take many forms.**)**

When a derived class inherits from a base class, it gains all the methods, fields, properties and events of the base class. To change the data and behavior of a base class, you have two choices:

عندما الصنف المشتق يَرِثُ مِنْ الصنف الأساسي، يَكْسبُ كُلّ الدوال والحقول والملكيات وأحداث الصنفِ الأساسيِ.

لتَغيير البياناتِ وسلوكِ الصنف أساسي، لدينا خياران:

1- You can replace the base member with a new derived member. The **new** keyword is used to create a new definition of that **method**, **field**, or **property** on a derived class.

يُمْكِن أَنْ تَستبدل العضو الأساسي بالعضو الجديد المُشتَق. الكلمة المحجوزة **new** تُستَعملُ لتكوين تعريف جديد للدالة أو الحقل، أَو الملكية في الصنف المُشتَقّ.

```
public class A
{
   public void DoWork() { }
   public int F;

    public int WorkProperty
   {
      get { return 0; }
   }
}

public class B : A
{
   public new void DoWork() { }
   public new int F;
   public new int WorkProperty
   {
      get { return 0; }
   }

}
```

Those base class members are called ***hidden members***. Hidden class members can still be called if an instance of the derived class is **cast** to an instance of the base class. For example:

أعضاء الصنفِ الأساسيِ يسمون أعضاء مخفيينَ ***hidden members***. أعضاء الصنفِ المخفيينِ يُمْكِنُ أَنْ يستدعون إذا تم إحلال الكائن من نوع الصنف المشتق (instance of the derived class) بعملية الـ **cast** إلى الكائن من نوعِ الصنفِ الأساسيِ. على سبيل المثال:

```
B ob1 = new B( );
ob1.DoWork( );  // Calls the new method.

A ob2 = (A) ob1;    //casting
ob2.DoWork();  // Calls the old method.
```

2- Or, you can override a virtual base member.

In order for an instance of a derived class to completely take over a class member from a base class, the base class has to declare that member as virtual (افتراضي).

لكي يَتمكّن الكائن من نوع الصنف المُشتَق من السَيْطَرَة بالكامل على عضو مِنْ الصنف الأساسي، يَجِبُ أَنْ يُعلنَ العضو كـ virtual في الصنف الأساسي .

This is accomplished by adding the virtual keyword before the return type of the member. A derived class then has the option of using the override keyword, instead of **new**, to replace the base class implementation with its own. For example:

هذا ينجز بإضافة الكلمة المحجوزةِ virtual قبل return type للعضو. في الصنف المُشتَقّ يمكن استخدام الكلمة المحجوزة override (اختياري) بدلاً مِنْ **new** ، لاستبدال تطبيق (implementation) الصنفِ الأساسيِ بـ (implementation) الصنف المشتق. على سبيل المثال

```
public class A
{
   public virtual void DoWork() { }
   public virtual int WorkProperty
   {
      get { return 0; }
   }
}
public class  B:A
{
```

```
    public override void DoWork() { }
    public override int WorkProperty
    {
        get { return 0; }
    }
}
```

Fields cannot be virtual; only methods, properties, events and indexers can be virtual.

Virtual methods and properties allow you to plan ahead for future expansion.

Virtual تسمح لنا للتخطيط مسبقا للتوسعيات المستقبلية.

Virtual members remain virtual indefinitely. If class A declares a virtual member, and class B derives from A, and class C derives from B, class C inherits the virtual member, and has the option to override it, regardless of whether class B declared an override for that member.

Virtual members تبقى Virtual إلى مالانهاية بغض النظر عن عدد الأصناف الموروثة.

إذا تم إعلان الصنف A بشكل virtual member والصنف B أشتق من A والصنف C أشتق من B. فان C يرث virtual member ولديه الخيار لعمل override لهذا العضو بغض النظر عما إذا كان B عمل له override أم لا.

For example:

```
public class A
{
    public virtual void DoWork() { }
}

public class B : A
{
    public override void DoWork() { }
}


public class C : B
{
    public override void DoWork() { }
}
```

Example:

```
class base1
   {
      public virtual void who()
      {
         Console.WriteLine("base");
      }
      class derived1 : base1
      {
         public override void who()
         {
            Console.WriteLine("derived1");

         }
         class derived2 : base1
         {
            public override void who()
            {
               Console.WriteLine("derived2");
            }
         }

         static void Main(string[] args)
         {
            base1 b = new base1();
            derived1 d1 = new derived1();
            derived2 d2 = new derived2();
            base1 bref;
            bref = b;
            bref.who();
            bref = d1;
            bref.who();
            bref = d2;
            bref.who();
         }
      }
   }
```

Abstract Classes الأصناف المجردة من التمثيل

An abstract method has declaration only (no implementation). It creates a method name and signature that must be implemented in all derived classes.

الدوال المجردة (من نوع abstract) تحتوي فقط التوقيع (أي الإعلان فقط) وليس لها محتويات ويجب كتابة المحتويات لها في الأصناف الموروثة.

Abstract classes establish a base for derived classes, but it is not legal to instantiate an object of an abstract class. Once you declare a method to be abstract, you **prohibit(يمنع) the creation of any instances of that class.**

الأصناف المجردة من التمثيل: عبارة عن قالب (template) للأصناف (classes) التي ستشتق **ولا يمكن تكوين كائن (object)** منه **(أي يورث فقط)**.

Designating a method as abstract is accomplished by placing the `abstract` keyword at the beginning of the method definition:

abstract public void print( );

If one or more methods are abstract, the class definition must also be marked `abstract`, as in the following:

abstract public class A

إذا كان احد دوال الصنف من نوع (abstract) فان الصنف يجب أن يكون من نوع (abstract)

Example

```
using System;
 abstract class C1
   {   abstract public void print();

        public void print2()
          {
             Console.WriteLine("called from class C1");
          }
   }

class C2 : C1
 {    override public void print()
       {
          Console.WriteLine("Test Abstract");
        }
}

class tester
{
      static void Main(string[] args)
```

```
    {

 // C1 ob1 = new C1();// Error: Cannot create an instance of the abstract class 'C1'

        C2  ob = new C2();
        ob.print();
        ob.print2();
    }
}
```

Often an abstract class will include non-abstract methods. Typically, these will be marked virtual, providing the programmer who derives from your abstract class the choice of using the implementation provided in the abstract class, or overriding it. Once again, however, **all abstract methods must be overridden in order to make an instance of the (derived) class.**

إذا كان في الصنف من نوع (abstract) دوال ليست (abstract) ففي الحال المثالية من الأفضل كتابة كلمة (virtual) أمام هذه الدوال **لإتاحة** الفرصة أمام المبرمجين اللذين سيرثون من صنفك هذا لاستعمال هذه الدوال كما هي أو تحويرها باستعمال (override).
كل الدوال من نوع (abstract) يجب أن تكون (override) في الأصناف الموروثة.

Remarks

The overridden base method must have the same signature as the **override** method.

You cannot override a ***non-virtual*** or ***static*** method. The overridden base method must be ***virtual***, ***abstract***, *or* ***override***.

You cannot use the modifiers **new**, **static**, **virtual**, or **abstract** to modify an **override** method.

Sealed Classes (العقيمة) الأصناف المغلفة

In contrast to an abstract class, a sealed class does not allow classes to derive from it at all (to prevent inheritance). **You can only create instances of objects from the sealed class.** The `sealed` keyword placed before the class declaration.

بالمقارنة مع الصنفِ من نوع abstract الصنفَ من نوع sealed لا يَسْمحُ للأصنافِ بالاشتقاق منه مطلقا (لمَنْع الوراثِة أي يصبح الصنف عقيما).يمكننا فقط تكوين كائنات منه . الكلمة المحجوزة sealed توَضعْ قَبْلَ إعلانَ الصنفَ ً.

sealed class C1

If you changed the declaration of ***C1*** in Example from `abstract` to `sealed` (eliminating the `abstract` keyword from the ***`print( )`*** declaration as well), the program fails to compile. If you try to build this project, the compiler returns the following error message:

'C2' cannot inherit from sealed type 'C!'

you cannot create a new protected member in a sealed class.

لا يمكن تكوين عضو محمي (protected) جديد في صنف من نوع sealed.

Example:

```
using System;

sealed class C1
{
   public void print()
   { Console.WriteLine("In class C1"); }
}
class C2
{
    public void print()
   {
      Console.WriteLine("In class C2");
   }
}

class tester
{
   static void Main(string[] args)
   {   C1 ob1=new C1();
         ob1.print();
      C2 ob2 = new C2();
      ob2.print();

   }
```

}

Microsoft recommends using sealed when you know that you won't need to create derived classes, and also when your class consists of nothing but static methods and properties.

تَوصي مايكروسوفتُ باستعمال sealed عندما تكون متأكد من انه لا تحتاج لاشتقاق الصنف وكذلك عندما يتكون الصنف من static methods و properties .