الدرس الحادي عشر : أساسيات البرمجة الكائنية 1

في هذا الدرس و الدروس القادمة ، سوف نتعرف سويا على أهم أساسيات البرمجة الكائنية ، بعد أن استعرضنا أهم أسس البرمجة الخطية ، و سيحوي هذا الدرس على ما يلي :

- مقدمة للبرمجة الكائنية
- الأصناف المجردة
- إعادة تعريف المشيدات
- الخصائص
- تركيب الأصناف من بعضها البعض

مقدمة للبرمجة الكائنية :

تقوم البرمجة الكائنية على تقسيم البرنامج إلى عدة كائنات ، بحيث يقوم كل كائن بجزء معين من البرنامج الكلي ، و تحتوي الكائنات على وسائل و خصائص تمكنها من أداء مهمتها باستقلالية تامة ؛ بحيث لا تعتمد على كائنات الأخرى ، بل أن تلك الكائنات لا ترى كيف قام ذاك الكائن بأداء مهمته ، و حتى أسهل عليك الفكرة تصور أن تريد أن تقود سيارة ، فما عليك أن تدير محرك السيارة ، و تتحكم في المقود و ناقل السرعة و الكوابح ، فأنت لا تعرف في الغالب كيف يقوم المحرك بدفع السيارة ، و لا كيف يقوم ناقل السرعة بمهمته ، فيمكن تشبيه السيارة ببرنامج يحتوي على عدة كائنات منها المحرك ، و ناقل السرعة ، و المقود و الكوابح ، و السائق هو بالبرنامج الرئيسي الذي يستغل تلك الكائنات في أداء مهمة يحددها هو .

هذه النسق من البرمجة يسمى البرمجة الكائنية ، ولعلك تلاحظ أن هناك فرق بين الكائن و الوسيلة ، فالوسيلة تقوم بمهمة معينة فقط ، بينما الكائن يقوم بمهمة معينة ، بالإضافة إلى أنه يملك خصائص معينة ، فعندما نرجع إلى كائن المحرك نجد أنه له خصائص مثل معدل استهلاك البنزين و كم حصان قوته ، هذا مع المهمة الرئيسية وهي دفع السيارة .

الأصناف المجردة :

تقوم عملية إنشاء الكائنات و استغلالها في معظم لغات التي تدعم البرمجة الكائنية على إنشاء صنف class مجرد يحتوي كل خصائص و مهام الكائن ، و من ثم تجسيد ذلك الصنف في كائن أو عدد لا نهائي من الكائنات تنتمي إلى ذلك الصنف .

و حتى لا نكون أكثر التصاقًا بلغة السي شارب ، سوف نأخذ هذا المثال الذي سيوضح لك كيف سوف ننشئ صنف جديد اسمه Time1 بواسطة الكلمة class ، بعد ذلك سوف نستغل هذا الصنف في إنشاء كائنات تقوم بأخذ الوقت (الساعة و الدقيقة و الثانية) ، و إرجاع الوقت منسق على صيغة القياسية أو صيغة 24 ساعة.

سوف أترك لك مهمة التعرف على كيفية إضافة صنف جديد إلى البيئة التي تبرمج عليها ، أما أنا فاستخدم برنامج يسمى Snippet Compiler ، وهو سهل جدا ، على العموم إذا لم تعرف فما عليك إلا أن تكتب جميع الأصناف في ملف واحد !!

```
using System;

// من هنا يبدأ تعريف الصنف الجديد
public class Time1 : Object

{
        // المتغيرات اللحظية الخاصة بالصنف ، وهي الساعة و الدقيقة و الثانية
        private int hour;    // 0-23
        private int minute; // 0-59
        private int second; // 0-59


        //هنا تعريف مشيد الصنف الذي سوف يقوم بوضع المتغيرات
        // اللحظية إلى الصفر
        public Time1()
        {
                SetTime(0,0,0);
        }

        //   هذه الوسيلة ستقوم الوقت على هيئة 24 ساعة
        //كذلك ستقوم بفحص و تأكد من صحة البيانات ، فإذا لم تكن صحيح سوف تجعلها صفر.
        public void SetTime(
                        int hourValue, int minuteValue, int secondValue)
        {
                hour = ( hourValue >= 0 && hourValue < 24) ? hourValue : 0;

                minute = ( minuteValue >= 0 && minuteValue < 60) ?
                        minuteValue : 0;

                second = ( secondValue >= 0 && secondValue < 60) ?
                        secondValue : 0;
        }
```

```
        // هذه الوسيلة سوف تحول الوقت إلى سلسة حرف على هيئة 24 ساعة.
        public string ToUniversalString()
        {
         return String.Format("{0:D2}:{1:D2}:{2:D2}",hour,minute,second);
        }

        //.هذه الوسيلة سوف تحول الوقت إلى سلسة حروف على هيئة 12 ساعة
        public string ToStandardString()
        {
              return String.Format("{0:D2}:{1:D2}:{2:D2} {3}",
                     ( (hour == 12 || hour == 0)? 12 : hour%12),
                     minute, second, (hour < 12 ? "AM" : "PM" ) );
        }

 } // نهاية الصنف
```

في هذا المثال قمنا أولا بتعريف الصنف Time1 بهذا السطر :

```
public class Time1 : Object
```

حيث استخدمنا الكلمة class متبوعة باسم الصنف ، بعد ذلك أشرنا إلى أن هذا الصنف يرث الصنف Object باستخدام " : " ، على العموم كل الأصناف في السي شارب ترث هذا الصنف أبى من أبى و رضي من رضي ، على العموم سوف نتناول الوراثة بشيء من التفصيل في الدورس القادمة ، لا تقلق !!

بعد ذلك قمنا بكتابة تعريف الصنف بين قوسين معكوفين { } ، وهنا يجدر الأشارة أن مدى وصول الصنف يقع بين هذين القوسين ، سنرجع إلى هذه النقطة لاحقا ، و الآن نعود لشرح كلمتي public و private ، فهتين الكلمتين تنتميان إلى متحكمات الوصول للأعضاء ، حيث تعني الأولى أن العضو سواء كان متغير لحظي أو وسيلة يمكن الوصول إليه ، فمثلا بعد إنشاء كائن من الصنف Time1 ، سنسطيع الوصول إلى الوسيلة SetTime و استخدامها ، سترى ذلك في البرنامج التالي ، أما كلمة private فهي تعني أن العضو هو خاص و لا يمكن الوصول إليه و استغلاله ، في العادة نجعل المتغيرات اللحظية خاصة و الوسائل عامة ، و لكن ربما نضع بعض الوسائل خاصة حينئذ تسمى وسائل مساعدة .

والآن جاء دور المشيد ، المهمة الآساسية للمشيد هي وضع قيمة ابتدائية لأعضاء الصنف ، و لا فرق بين المشيد و الوسيلة سوى أن المشيد لا يرجع شيء ، أي لا يستخدم كلمة return ، في مثالنا استخدمنا المشيد لجعل قيمة الساعة و الدقيقة و الثانية صفر ، أما بقية الوسائل فالتعليقات التي عليها تبين مهمتا ولا داعي للتكرار .

و الآن سوف نكتب برنامج يقوم بإنشاء كائن من النوع Time1 ، و يقوم بفحصه :

```
using System;
```

```
using System.Windows.Forms;

// هذا البرنامج سينشئ و يستخدم كائن من الصنف Time1
class TimeTest1

{
        // هنا نقطة الدخول للتطبيق
        static void Main (string [] args)
        {
                //هنا سوف يتم إنشاء كائن من الصنف Time1
                Time1 time = new Time1(); //هنا سيتم منادة مشيد الصنف Time1

                string output;

                // سيتم اختبار القيم الإبتدائية للوقت
                output = "Initial universal time is: " +
                          time.ToUniversalString() +
                 "\nInitial standard time is: " + time.ToStandardString();

                // هنا سوف نضع وقت صحيح
                time.SetTime (13,27,6);

                output += "\n\nUniversal time after SetTime is: " +
                        time.ToUniversalString() +
            "\nStandard time after SetTime is: " + time.ToStandardString();

                // هنا سنضع وقت خاطئ
                time.SetTime (99,99,99);

        output += "\n\nUniversal time after attempting invalid settings:"
                + "\n Universal time: " + time.ToUniversalString() +
                          "\n Satandard time: " + time.ToStandardString();

                // هنا سنعرض الناتج في رسالة
                MessageBox.Show(output, "Testing Calss Time1");
        }
} // نهاية الصنف TimeTest1
```

أما النتيجة التطبيق فهي كالتالي :

والآن أود أن أنبه كل الأصناف المعرفة لا تنتمي اسم فضاء معين - لاحظ أن تستطيع أن تكون اسم فضاء خاص بك باستخدم الكلمة namespace كما في المثال التالي :

```
namespace MyCompany.Proj1
{
    class MyClass
    {
        // code
    }
}
```

كنت أقول أن كل الأصناف التي لا تنتمي إلى اسم فضاء معين تكون في الفضاء الافتراضي ، و هو الفضاء الذي يحوي جميع الأصناف المترجمة الموجودة في الدليل الحالي لبرنامجك ؛ فعند وجود الصنف المراد استخدامه في اسم فضاء معين ، فلا بد من استخدام كلمة using متبوعة باسم ذاك الفضاء ، بطبيعة الحال في مثالنا لا نحتاج إلى ذلك لأننا نستخدم الفضاء الافتراضي.

الأمر الثاني أود أن ألفت انتبهك إلى كيفية إنشاء كائن من صنف معين ، ركز في هذا السطر :

```
Time1 time = new Time1();
```

فبعد ذكر اسم الصنف Time1 ذكرنا اسم الكائن المراد إنشائه وهو time ثم استخدمنا كلمة new لحجز الذاكرة لهذا الكائن الجديد و نادينا مشيد الصنف Time1 حتى يقوم بوضع القيم الإبتدائية لعناصر ذلك الكائن.

الأمر الثالث هو كيفية الوصول إلى الأعضاء كائن العامة في ذلك الكائن ،

انظر هذا السطر :

```
time.ToUniversalString()
```

حيث استخدمنا "." للوصول لتلك الأعضاء.

إعادة تعريف المشيدات

ولآن نرجع للمشيدات ، فكما في الوسائل نستطيع أن نعيد تعريف المشيدات لكي تستقبل أي عدد من المتغيرات ، في المثال التالي سوف نطور الصنف الذي أنشأناه سابقا بحيث يكون فيه أكثر من مشيد :

```
using System;

public class Time2

{
        private int hour;
        private int minute;
        private int second;

        //هذا المشيد يجعل جميع المتغيرات صفرًا
        public Time2()
        {
                SetTime(0,0,0);
        }

        //أما هذا المشيد فهو يستقبل الساعات ، ويجعل الباقي صفر
        public Time2( int hour)
        {
                SetTime(hour,0,0);
        }

        //أما هذا فيستقبل الساعات و الدقائق ويجعل الثواني صفرًا
        public Time2 (int hour,int minute)
        {
                SetTime(hour,minute,0);
        }

        //هذا المشيد فإنه يستقبل الساعات و الدقائق و الثواني
        public Time2 ( int hour, int minute, int second)
        {
                SetTime(hour,minute,second);
        }

        //أما هذا المشيد فإنه يستقبل كائن من نفس الصنف
        public Time2 (Time2 time)
        {
                SetTime( time.hour, time.minute, time.second);
        }

        public void SetTime
              (int hourValue, int minuteValue, int secondValue)
        {
```

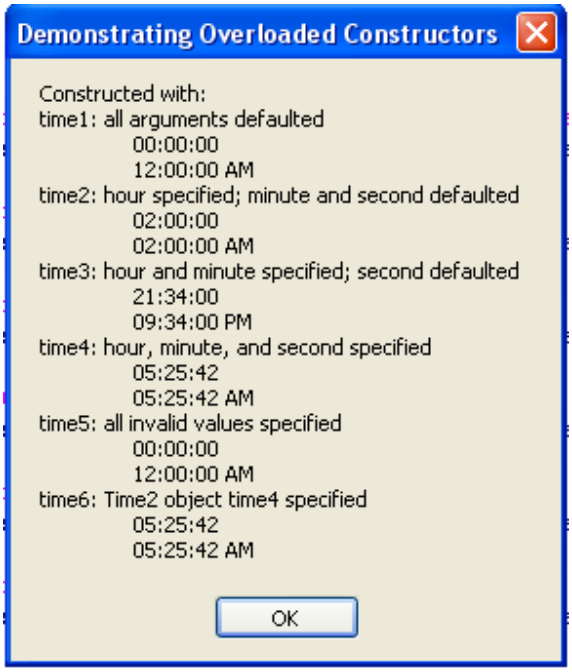
```
            hour = ( hourValue >= 0 && hourValue < 24) ? hourValue : 0;

            minute = ( minuteValue >= 0 && minuteValue < 60) ?
             minuteValue : 0;

          second = ( secondValue >= 0 && secondValue < 60) ?
            secondValue : 0;
      }

      public string ToUniversalString()
      {
            return
           String.Format("{0:D2}:{1:D2}:{2:D2}",hour,minute,second);
      }

      public string ToStandardString()
      {
            return String.Format("{0:D2}:{1:D2}:{2:D2} {3}",
                  ( (hour == 12 || hour == 0)? 12 : hour%12),
                    minute, second, (hour < 12 ? "AM" : "PM" ) );
      }
}
```

قبل أن أنتقل إلى البرنامج الذي سيختبار تلك المشيدات و يعرض النتيجة ، أود أن أشير أنه إذا كان هناك كائنين من نفس الصنف فإنهما يستطيعان أن يريا جميع أعضاء الخاصة ذاك الصنف ، فمن هنا استطعنا في المشيد الخامس استخدام المتغيرات الخاصة للكائن الذي من نفس الصنف ، و الآن ننتقل إلى البرنامج :

```
using System;
using System.Windows.Forms;

class TimeTest1

{
      static void Main (string [] args)
      {
            Time2 time1, time2, time3, time4, time5, time6;

            time1 = new Time2();           // 00:00:00
            time2 = new Time2(2);          // 02:00:00
            time3 = new Time2(21,34);      // 21:34:00
            time4 = new Time2(5,25,42);    // 05:25:42
            time5 = new Time2(27,74,99);   // 00:00:00
            time6 = new Time2(time4);      // 05:25:42

            String output = "Constructed with: " +
            "\ntime1: all arguments defaulted " +
       time1.ToUniversalString()+ "\n\t" +  time1.ToStandardString();

            output += "\ntime2: hour specified; minute and second " +
                  "defaulted " + time2.ToUniversalString() + "\n\t" +
                   time2.ToStandardString();
```

```
            output += "\ntime3: hour and minute specified; second " +
                      "defaulted " + time3.ToUniversalString() + "\n\t" +
                       time3.ToStandardString();

            output += "\ntime4: hour, minute, and second specified " +
        time4.ToUniversalString() + "\n\t" + time4.ToStandardString();

            output += "\ntime5: all invalid values specified " +
        time5.ToUniversalString() + "\n\t" + time5.ToStandardString();

            output += "\ntime6: Time2 object time4 specified " +
        time6.ToUniversalString() + "\n\t" + time6.ToStandardString();


            MessageBox.Show(output,
                "Demonstrating Overloaded Constructors");
        }
}
```

و النتيجة هي التالي :



الخصائص

الآن نرجع إلى المتغيرات اللحظية في الصنف ، قلنا أنه في الغالب نجعلها خاصة ، طيب لو أردنا أن نجعل متغير عام في الصنف فماذا سنفعل ؟ ربما أسهل حل هو أن نجعله public و نريح رؤسنا ، ولكن ألن يشكل ذلك بعض الخطورة الأمنية على الصنف ؟ تصور أن

المبرمج الذي يستخدم ذلك الصنف قام بوضع قيمة خاطئة لذاك المتغير مما يؤدي إلى عدم عمل كائن من ذاك الصنف بشكل صحيح ، مما يضطره إلى تعديل في شفرة ذاك الصنف أو على الأقل معرفة كيفية تم إنشاء ذاك الصنف ؛ و هذا يهدم أحد أسس البرمجة الكائنية ألا و هو الإستقلاليةو إخفاء تفاصيل البرمجية ، إذا ما رأيك أن نقوم بعملية ترشيح قبل أن نضع القيمة أو أن نخذ قيمة من أي متغير لحظي ، و بهذا نكون واثقين بأن كل شيء سيسير على أحسن وجه .

في الحقيقة هذه الخاصية موجودة في لغتنا السي شارب ، و تسمى الخصائص فخاصية للوضع set و خاصية للأخذ get ، و حتى لا نخرج عن أسلوبنا في الشرح خذ هذا الصنف المعدل من Time1 الذي يحوي على خصائص الأخد و الوضع للساعات و الدقائق و الثواني:

```
using System;

public class Time3

{
        private int hour;
        private int minute;
        private int second;


        public Time3()
        {
                SetTime(0,0,0);
        }


        public void SetTime(
                                int hourValue, int minuteValue, int secondValue)
        {
                Hour =hourValue;
                Minute = minuteValue;
                Second = secondValue;
        }


        //خصائص الساعة
        public int Hour
        {
                get
                {
                        return hour;
                }
                set
                {
                        hour = ( value >= 0 && value < 24) ? value : 0;
                }
        }
```

```
        //خصائص الدقائق
        public int Minute
        {
                get
                {
                        return minute;
                }
                set
                {
                        minute = ( value >= 0 && value < 60) ? value : 0;
                }
        }

        // خصائص الثواني
        public int Second
        {
                get
                {
                        return second;
                }
                set
                {
                        second = ( value >= 0 && value < 60) ? value : 0;
                }
        }

        public string ToUniversalString()
        {
                return
                  String.Format("{0:D2}:{1:D2}:{2:D2}",hour,minute,second);
        }

        public string ToStandardString()
        {
                return String.Format("{0:D2}:{1:D2}:{2:D2} {3}",
                        ( (hour == 12 || hour == 0)? 12 : hour%12),
                        minute, second, (hour < 12 ? "AM" : "PM" ) );
        }
}
```

والآن جاء دور التشريح !! و لنأخذ خصائص الثواني :

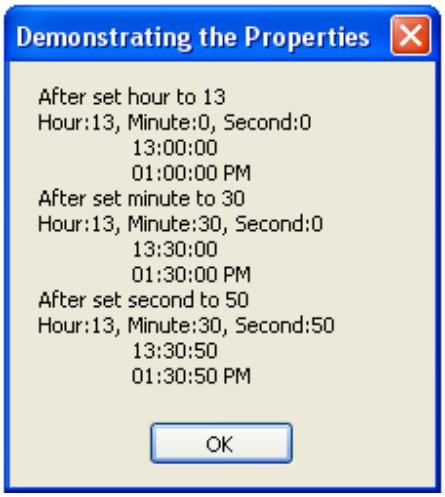
```
public int Second
        {
                get
                {
                        return second;
                }
                set
                {
                        second = ( value >= 0 && value < 60) ? value : 0;
                }
        }
```

فكما ترى أعلنا عن متغير عام اسمه Second ليكون واجهة للمتغير الخاص second ، ثم استعملنا كلمة get لترجع قيمة المتغير الخاص ، بطبيعة الحال يمكنك أن تكتب و تنسق الكيفية التي ستقدم بها قيمة هذا المتغير في هذا المكان ، ولكن لا نحتاج إلى هذا في مثالنا المبسط ، و بعد ذلك قمنا باستخدام كلمة set حتى نضمن أن الذي يوضع هو صحيح ، و كما يظهر فإن كلمة value تعني القيمة التي سوضع لهذا المتغير .

بعد أن كتبنا الصنف الجديد ، سوف نكتب برنامج يقوم باختبارة و هو كالتالي :

```
using System;
using System.Windows.Forms;


class TimeTest3

{
      static void Main (string [] args)
      {

            Time3 time = new Time3();

            //13 هنا سنضع الساعة
            time.Hour = 13;

            String output = "After set hour to 13 " +
            String.Format(":{0}, Minute:{1}, Second:{2}",time.Hour,
            time.Minute, time.Second) + "\n\t"+
            time.ToUniversalString() + "\n\t" + time.ToStandardString();

            //30 أما هنا سنضع الدقائق
            time.Minute = 30;

            output += "\nAfter set minute to 30 " +
            String.Format(":{0}, Minute:{1}, Second:{2}",time.Hour,
            time.Minute, time.Second) + "\n\t"+
            time.ToUniversalString() + "\n\t" + time.ToStandardString();

            //50 و هنا سنضع الثواني
            time.Second = 50;

            output += "\nAfter set second to 50 " +
            String.Format(":{0}, Minute:{1}, Second:{2}",time.Hour,
            time.Minute, time.Second) + "\n\t"+
            time.ToUniversalString() + "\n\t" + time.ToStandardString();


            MessageBox.Show(output, "Demonstrating the Properties");
      }
}
```

و الناتج هو :



تستطيع أن تلعب بالبرنامج السابق و تختبر بقية الخيارات بأن تضع قيم خاطئة ، و ترى النتيجة بفسك .

تركيب الأصناف من بعضها البعض

و الآن قد يتبادر إلى ذهنك سؤال وهو أيمكن أن أستخدم كائن من صنف ما في تكوين صنف آخر ؟ بكلمات أخرى أأستطيع تركيب صنف من عدة كائنات بحيث تصبح في الأخير صنف واحد ؟ و الجواب نعم يمكنك ذلك فبدل أن تكتب الشفرة من جديد يمكنك استعمالها جاهزة من صنف سابق ، و حتى أوضح لك هذه الفكرة خذا المثال التالي ، و هو عبارة عن صنفين الصنف الأول وهو Date أما الثاني فهو Employee أي الموظف ، فالصنف الموظف يأخذ كائنين من الصنف Date - أي التاريخ – تاريخ الميلاد و تاريخ الإلتحاق بالعمل ، فالصنف الموظف يتكون من كائنين من الصنف التاريخ و بعض الوسائل الأخرى أنظر المثال :

```
using System;

//صنف التاريخ
public class Date
{
        private int month;        // 1-12
        private int day;          // 31-1 يعتمد على الشهر
        private int year;         // أي سنة

        //   مشيد صنف التاريخ ، والذي سيتأكد من الشهر
        //   و سينادي وسيلة لتأكد من الأيام
        public Date ( int theMonth,int theDay, int theYear)
        {
```

```
            if (theMonth > 0 && theMonth <= 12) // هنا يتم فحص الأشهر
                  month = theMonth;

            else // إذا كان قيمة الشهر غير صحيح سيتم وضع قيمة واحد فيه، و التنبيه على ذلك
            {
               month =1;
               Console.WriteLine("Month {0} invalid. Set to month 1.",
                theMonth);
            }

            year = theYear;
            day = CheckDay (theDay); //  هنا يتم تأكد من الأيام
      }

      //هذه الوسيلة ستتأكد من الأيام
      private int CheckDay( int testDay)
      {
            // سنستعمل مصفوفة تحوي عدد الأيام في كل شهر
            int [] daysPerMonth ={0,31,28,31,30,31,30,31,31,30,31,30,31};

            //هنا يتم فحص الأيام إذا كانت في حدود الشهر
            if (testDay > 0 && testDay <= daysPerMonth [month])
                  return testDay;

            //هنا نتأكد إذا كانت السنة كبيسة
            if (month == 2 && testDay ==29 &&
              (year % 400 == 0 || (year % 4 == 0 && year % 100 !=0 ) ) )
                  return testDay;

            // إذا كانت القيمة خاطئة سيضعها واحد ، ويعطي رسالة عن ذلك
            Console.WriteLine("Day {0} invalid. Set to day 1.",testDay);
            return 1;
      }

      //هذه الوسيلة ستعطي التاريخ منسق
      public string ToDateString ()
      {
            return month + "/" + day + "/" + year;
      }
}
```

والآن ننتقل إلى صنف الموظف و البرنامج الذي سيختبرها:

```
using System.Windows.Forms;
using System;

public class  Employee
{
      private string firstName;
      private string lastName;
      private Date birthDate; // مرجع لكائن من صنف التاريخ
      private Date hireDate;  // مرجع لكائن من صنف التاريخ
      //  مشيد صنف الموظف
      public Employee (string first, string last, int birthMonth, int
```

```
        birthDay, int birthYear, int hireMonth, int hireDay, int hireYear)
        {
                firstName = first;
                lastName = last;
                // إنشاء كائنات من صنف التاريخ
                birthDate = new Date(birthMonth, birthDay, birthYear);
                hireDate = new Date (hireMonth, hireDay, hireYear);
        }

        public string ToEmployeeString()
        {
                return lastName + ", " + firstName + " Hired: " +
                  hireDate.ToDateString() + " Birth day: " +
                  birthDate.ToDateString();
        }
}

//-----------------------------------------------------------------------
//   هنا نبدأ البرنامج الذي سيختبر الصنف الجديد
class compositionTest
{
        static void Main (string[] args)
        {
        Employee e = new Employee ("Bob", "Jones",7,24,1949, 3, 12, 1988);

        MessageBox.Show (e.ToEmployeeString(),"Testing Class Employee");

        }
}
```

و النتيجة هي :

و الآن كالعادة الباقي عليك لكي تختبر الباقي و تتأكد بنفسك ، في الحقيقة فإن تركيب الأصناف سوف نرجع له و نقارنه مع الوراثة في الدروس القادمة بإذن الله تعالى .

و الآن أود أن أخبرك بأن الدرس قد انتهى !!

زايد السعيدي

منتدى الإبداع الإسلامي