الفصل الثامن

خاصية التوريث INHERITANCE

فى هذا الفصل Programming Concepts

8

في هذا الفصل نتناول أشهر خصائص مفهوم البرمجة بواسطة الأهداف OOP وهي خاصية التوريث Inheritance وكيفية تحقيقها وما يرتبط بها من أوامر وخصائص وذلك من خلال النقاط التالية :

- ما معنى خاصية التوريث Inheritance
- كيف كان يتم تطبيق خاصية التوريث في الإصدار السابق vb6
  كيف يتم تطبيق خاصية التوريث في vb.net
- مثال لتطبيق خاصية التوريث inheritance
- الكلمات المستعملة مع خاصية التوريث Inheritance
- تغيير الخصائص و الدوال في الفصيلة المشتقة overriding
- ما هي خاصية polymorphism

معنى خاصية التوريث Inheritance

هي خاصية قوية جداً في مفهوم البرمجة باستعمال الأهداف OOP وتستعمل لتقليل إعادة كتابة الأوامر، وهي أنه باستعمال خاصية التوريث تستطيع إنشاء فصيلة (CLASS) تحتوى على خصائص Proprities و دوال methods ثم استعمالها كأساس لفصائل أخرى وبالتالي لا نحتاج لكتابة ما كتبناه في تعريف الفصيلة الأولي (والتي تسمى basic class) مع كل فصيلة جديدة .

وتسمى الفصائل الجديدة التي تستورث فصيلة قديمة فصائل مشتقة derived class وغالباً يتم التعامل مع الفصائل classes بأن نستورث فصيلة ونكمل عليها .

ولتوضيح ذلك نضرب لك المثال التالي:

نفرض أنك تريد تعريف فصيلتان الأولي لتعريف بيانات الطالب و الثانية لتعريف بيانات المدرس فبدون استعمال خاصية التوريث سنضطر لإنشاء الفصيلتان برغم من تشابه الفصيلتان واحتوائهما على بيانات متشابه فمثلاً فصيلة الطالب تحتوى على البيانات التالية :

- الكود ، الاسم ، العنوان ، التليفون ، المؤهل ، تاريخ الميلاد ، وكثير من البيانات

وتحتوى على الدوال التالية :

- دالة تسجيل بيانات الطالب
- دالة حذف بيانات الطالب
- دالة تعديل بيانات الطالب (ودوال أخرى)

وعند النظر إلي فصيلة المدرس سوف نجد أنها تشتمل علي كثير من البيانات و الدوال الأعضاء في فصيلة الطالب بالاضافة لبعض الأعضاء الجديدة .

وبالتالي باستعمال خاصية التوريث علينا توريث فصيلة الطالب لفصيلة المدرس وإضافة الجديد إليها مما يوفر علينا إعادة كل شيء ويمكن تكرار هذه الفكرة مع فصائل جديدة أخرى مما يوفر الكثير من الوقت بالاضافة ان هذا الاسلوب يسمح لك باستعمال فصائل classes تم اعدادها وتم اختبارها.

التوريث في VB.Net

أما VB.NET فقد وفر إمكانية التوريث الحقيقية ، حيث لا يسمح لك بإنشاء فصائل كاملة بكل دوالها (تعتبر كفصائل أساس Base class) ثم توريث هذه الفصائل بكل دوالها ومتغيراتها لفصائل جديدة وذلك يوفر عليك الكثير وهذا يعني أنه يمكنك أن تنشأ form جديدة بناء علي form قديمة لأن form في حد ذاتها فصيلة.

وفي نفس الوقت يمكنك تنفيذ الفكرة القديمة (Interface) في VB.NET ولتوضيح كيفية تحقيق عملية التوريث في VB.NET تابع المثال التالي:

1- ابدأ تطبيق وحدد نوع ConsoleApplication كما سبق

2- اكتب سطور تعريف فصيلة class بالاسم Person كما في السطور التالية:

```
1.  Public Class person
2.  Public Pname, Paddress As String
3.  Public Sub SetData(ByVal name_v As String, ByVal    address_v As String)
4.  Pname = name_v
5.  Paddress = address_v
6.  End Sub
7.  Public Function getName()
8.  Return Pname
9.  End Function
10. Public Function getAddress()
11. Return Paddress
12. End Function
13. End class
```

في لغة C#

```
public class person
{
    public string Pname, Paddress;
    public void  SetData( string name_v ,   string address_v )
{
Pname = name_v;
Paddress = address_v;
}
    public string  getName()
    {
    return Pname;
    }
    public string  getAddress()
    {
        return Paddress;
    }
}
```

فى لغة Java

```
class Person
{
    public String Pname,Paddress;
    public void Set_Data(String Name_V,String address_V)
      {

        Pname=Name_V;
        Paddress=address_V;
    }

    public String Get_Name ( )
      {
       return Pname;
       }

    public String Get_Address ( )
      {
          return Paddress;
       }
  }
public class InheritApp1 {

    public static void main(String[] args) {
     } }
```

- في هذه السطور تم إنشاء فصيلة بالاسم person وتم تعريف متغيرات.
- وتم كتابة فهي فصيلة متكاملة تحتوى على المتغيرات Pname, Paddress.
- وتحتوى على الدالة () SetData التى تقوم بتسجيل بيانات الفصيلة.
- وتحتوى على الدالة ()getName التى تعيد قيمة المتغير pname الخاص بالفصيلة والخاص باسم الشخص وكذلك الدالة ()getAddress التى تعيد قيمة المتغير paddress .
- وعند إنشاء فصيلة جديدة وتوريثها هذه الفصيلة Person يتم استعمال متغيراتها ودوالها مباشرة دون الحاجة إلى إنشائها مرة أخرى .

وتوريث هذه الفصيلة لفصيلة جديده يتم كما يلي:

3- أكتب سطور فصيلة جديدة بالاسم student

```
Public Class student
Inherits person
Dim degree As Integer
Public Sub setdegree(ByVal degree_v As Integer)
degree = degree_v
End Sub
Public Function retdegree()
Return degree
End Function
End Class
```

فى لغة C#

```
57    public class student: person
58    {
59    int degree;
60        public void  setdegree(int  degree_v)
61    {
62        degree = degree_v;
63     }
64
65        public int  retdegree()
66        {
67            return degree;
68        }
69
70    }
```

فى لغة Java

```
25  class Student extends Person
26  {
27  int degree;
28    public void Set_Degre(int degree_V)
29    {
30
31     degree=degree_V;
32
33  }
34  public int RetDegree()
35  {
36    return degree;
37  }
38
39  }
```

في هذه السطور:

- تم إنشاء فصيلة جديدة بالاسم student.
- وتم استعمال العبارة inherits person ومعناها قم بتوريث الفصيلة person بكل ما فيها للفصيلة الجديدة (student) و بالتالي أصبحت نفس إعضاء الفصيلة القديمة person اعضاء فى الفصيلة الجديدة student و هذا يسمح لك باستعمالها مباشرة وذلك كما يلي :

فى الدالة الرئيسية (Sub Main قم بكتابة سطور استعمال كلا من الفصيلتين person,student كما فى السطور التالية:

```
1. Dim vname, vaddrs As String
2. Dim no As New person()
3. no.SetData("azab", "Cairo")
4. vname = no.getName()
5. vaddrs = no.getAddress()
6. Console.WriteLine(vname)
7. Console.WriteLine(vaddrs)
8. Dim st As New student()
9. St.setdegree(100)
10. Dim vd As Integer
11. dim vname2,vaddress2 as string
12. vd = st.retdegree()
13. st. SetData("omr", "Giza")
14. vname = st.getName()
15. vaddress2 = st.getAddress()
16. Console.WriteLine(vd)
17. Console.WriteLine(vname)
18. Console.WriteLine(vaddress2)
```

فى لغة C#

```
12        static void Main(string[] args)
13        {
14
15   string vname, vaddrs ;
16  person no = new person ();
17      no.SetData("azab", "Cairo");
18  vname = no.getName();
19  vaddrs = no.getAddress();
20  Console.WriteLine(vname);
21  Console.WriteLine(vaddrs);
22      student st =new student();
23
24      st.setdegree(100);
25      int  vd ;
26  string vaddress2;
27      vd = st.retdegree();
28      st. SetData("omr", "Giza");
29      vname = st.getName();
30      vaddress2 = st.getAddress();
31      Console.WriteLine(vd);
32      Console.WriteLine(vname);
33      Console.WriteLine(vaddress2);
34
35  }
```

فى لغة Java

```
42  public class InhertApp1 {
43
44      public static void main(String[] args) {
45          String vname,vaddress;
46          Person no=new Person();
47          no.Set_Data("Azat","Cairo");
48          vname=no.Get_Name ( );
49          vaddress=no.Get_Address ( );
50          System.out.println(vname);
51          System.out.println(vaddress);
52
53  Student st=new Student();
54  st.Set_Degree(100);
55  int vd;
56  String vaddress2;
57  vd=st.RetDegree();
58  st.Set_Data("Omar","Giza");
59
60      vname=st.Get_Name ( );
61          vaddress2=st.Get_Address ( );
62
63          System.out.println("vd:"+vd);
64       System.out.println("vname:"+vname);
65          System.out.println("vaddress:"+vaddress);
66
67          }
68  }
```

في هذه السطور

- في السطر رقم 1 تم تعريف المتغيرات vname, vaddrs.
- في السطر رقم 2 تم تعريف المتغير no(هدف) في الفصيلة person.
- فى السطور 3 و 4 و 5 و 6 و 7 تم التعامل مع هدف الفصيلة person باستدعاء الدوال والتعامل معها كما سبق.
- في السطر رقم 8 تم تعريف المتغير st من الفصيلة student.
- في السطر رقم 9 تم استدعاء الدالة الجديدة المضافة للفصيلة بالاسم setdegree.
- في السطر رقم 10 و 11 تم الاعلان عن متغيرات.
- في السطر رقم 13 تم استدعاء الدالة SetData() المعرفة داخل الفصيلة الأساسية Person فبالرغم من عدم وجود الدالة داخل الفصيلة الجديدة إلا أننا استعملناها لأنها معرفة فى الفصيلة التى استورثناها بالاستعمال الأمر Inherits.

- في السطر رقم 14 و 15 تم استدعاء دوال الفصيلة الأساسية Person فبالرغم من عدم وجود هذه الدوال داخل الفصيلة الجديدة ألا إننا استعملناها لانها معرفة فى الفصيلة التى استورثناها بالاستعمال الأمر Inherits.

من هذا المثال نلاحظ استعمال الخصائص المستورثه من الفصيلة person مع الفصيلة student بالرغم من عدم تعريفها أو عدم الإعلان عنها والسطور التالية تعرض النص الكامل للبرنامج.

```
Module Module1
    Sub Main()
        Dim vname, vaddrs As String
        Dim no As New person()
        no.SetData("azab", "Cairo")
        vname = no.getName()
        vaddrs = no.getAddress()
        Console.WriteLine(vname)
        Console.WriteLine(vaddrs)

        Dim st As New student()
        st.setdegree(100)
        Dim vd As Integer
        Dim vname2, vaddress2 As String
        vd = st.retdegree()
        st.SetData("omr", "Giza")
        vname = st.getName()
        vaddress2 = st.getAddress()
        Console.WriteLine(vd)
        Console.WriteLine(vname)
        Console.WriteLine(vaddress2)

    End Sub
    Public Class person
        Public Pname, Paddress As String

        Public Sub SetData(ByVal name_v As String, ByVal address_v As String)
            Pname = name_v
            Paddress = address_v
        End Sub
        Public Function getName()
            Return Pname
```

```
    End Function
    Public Function getAddress()
      Return Paddress
    End Function
  End Class
  Public Class student
    Inherits person
    Dim degree As Integer
    Public Sub setdegree(ByVal degree_v As Integer)
      degree = degree_v
    End Sub
    Public Function retdegree()
      Return degree
    End Function
  End Class
End Module
```

فى لغة C#

```
using System;
namespace ConsoleApplication5
{
/// <summary>
/// Summary description for Class1.
/// </summary>
class Class1
{

[STAThread]
static void Main(string[] args)
{

string vname, vaddrs ;
person no = new person ();
no.SetData("azab", "Cairo");
vname = no.getName();
vaddrs = no.getAddress();
Console.WriteLine(vname);
Console.WriteLine(vaddrs);
student st =new student();

st.setdegree(100);
int  vd ;
string vaddress2;
```

```
vd = st.retdegree();
st. SetData("cmr", "Giza");
vname = st.getName();
vaddress2 = st.getAddress();
Console.WriteLine(vd);
Console.WriteLine(vname);
Console.WriteLine(vaddress2);

}
}
public class person
{
public string Pname, Paddress;
public void  SetData( string name_v       string
address_v  )
{
Pname = name_v;
Paddress = address_v;
}
public string  getName()
{
return Pname;
}
public string  getAddress()
{
return Paddress;
}
}

public class student: person
{
int degree;
public void  setdegree(int  degree_v)
{
degree = degree_v;
}

public int  retdegree()
{
return degree;
}

}
}
```

فى لغة Java

```
class Person
{
public String Pname,Paddress;
public void Set_Data(String Name_V,String
address_V)
{

Pname=Name_V;
Paddress=address_V;
}

public String  Get_Name ( )
{
return Pname;
}

public String  Get_Address ( )
{
return Paddress;
}
}

class Student  extends Person
{
int degree;
public void Set_Degree(int degree_V)
{

degree=degree_V;

}
public int  RetDegree()
{
return degree;
}

}
public class InhertAppl {
```

```
public static void main(String[] args) {
String vname,vaddress;
Person no=new Person();
no.Set_Data("Azab","Cairo");
vname=no.Get_Name ( );
vaddress=no.Get_Address ( );
System.out.println(vname);
System.out.println(vaddress);

Student st=new Student();
st.Set_Degree(100);
int vd;
String vaddress2;
vd=st.RetDegree();
st.Set_Data("Omar","Giza");

vname=st.Get_Name ( );
vaddress2=st.Get_Address ( );
System.out.println("vd:"+vd);
System.out.println("vname:"+vname);
System.out.println("vaddress:"+vaddress);
}
}
```

فى لغة VB.NET

كلمات التوريث Inheritance Keywords

هناك مجموعة من الكلمات التي تستعمل مع عملية توريث الفصائل Inheritance وذلك لتحقيقها والتحكم فيها ، فمثلاً يمكنك منع توريث فصيلة أو العكس ،أو يمكنك استعمال فصيلة بدون توريثها لغيرها حل ذلك من خلال الكلمات المستعملة لهذا العرض ونعرضها كما يلي :

الكلمات Mustinheritable،Notinheritable

الأصل في الفصائل التي تنشئها أنها تعمل بمفردها أي يمكن استعمالها مباشرة بتعريف متغير (هدف) من الفصيلة ، وكذلك يمكن توريثها لغيرها ،ولكن يمكن تغيير ذلك كما يلي :

الأمر Notinheritable

هذا الأمر عند ما يكتب أمام الفصائل عند إنشائها يعني أنها غير قابلة للتوريث ويتم ذلك عند إنشاء الفصيلة بالصورة التالية :

```
public Notinheritable class Employee

End class
```

في السطور السابقة تم تعريف (الإعلان عن) فصيلة جديدة (class) بالاسم Employee ولكن الجديد هو كتابة الأمر notinheritable الذى لا يسمح بتوريث هذه الفصيلة لفصيلة أخرى ، ولكن يسمح فقط بتعريف متغير (هدف) من الفصيلة للتعامل معها ، وبالتالي لا يمكنك عند إنشاء فصيلة جديدة كتابة الأمر inherits واسم هذه الفصيلة.

الأمر Mustinherit

يؤدى هذا الأمر إلي نتيجة عكس الأمر Notinheritable فهو لا يسمح باستعمال الفصيلة المحددة مباشرة بل لابد من توريثها لفصيلة أخرى لتعريف متغير واستعمالها فمثلاً عند إنشائك فصيلة بالشكل التالي :

```
public Mustinherit class student

End class
```

في هذه الحالة لا يمكنك استعمال هذه الفصيلة مباشرة بتعريف متغير.

فلا يمكنك كتابة السطر التالي :

```
Dim ST AS New student ( )
```

هذا السطر يطلب تعريف المتغير ST من نوع الفصيلة Student ونظراً لوجود الأمر Mustinherit فلابد من توريث هذه الفصيلة ثم استعمال الفصيلة الجديدة .

كما في السطور التالية :

```
public class instructor
  Inherits student

End class
```

بهذا تم تعريف فصيلة جديدة وتوريث الفصيلة student لها ،و بالتالي يمكن تعريف متغير منها كما يلي :

```
Dim NS AS  New instructor ( )
```

تغيير الدوال في الفصيلة الجديدة overriding proprities & Methods

فى لغة VB.NET

من الإمكانيات المتاحة في مفهوم OOP البرمجة بواسطة الأهداف خاصية تسمى Overriding وهذه الخاصية تعنى إمكانية كتابة دوال جديده في الفصائل الجديدة بنفس اسم الدوال الموجودة في الفصيلة الأساس (Base class) التي تم توريثها أي أن الدالة موجودة في الفصيلة الأساسية ومع ذلك تم إنشائها مرة أخرى بنفس الاسم في الفصيلة الجديدة و بالتالي يتم إلغاء الدالة من الفصيلة الجديدة واعتماد الدالة الجديدة وهذا يسمى تركيب دالة على أخرى overriding ولتوضيح ذلك ننشأ فصيلة بها دالة ثم نستورث هذه الفصيلة و ننشأ الدالة السابقة في الفصيلة الجديدة و يظهر ذلك من السطور التالية:

```
1: public class one
2: public overridable function say( )
3:   MSGbox "THIS Msg from  class one   "
4:  End function
5: End class
6: Public class two
7:inherits one
8:  public overrides function say( )
9:    Msgbox "this Msg from class two..."
10:  end function
11: end class
```

فى لغة C#

```
public class one
{
    public virtual void say( )
        {
    Console.WriteLine ("THIS Msg from  class one   " );
        }
}
public class two:one
{
    public override void say( )
    {
        Console.WriteLine ("THIS Msg from  class two   " );
    }
}
```

فى لغة Java

Override.java

```
1  class One
2  {
3     public void Say()
4     {
5     System.out.println("This Msg from class One");
6
7     }
8  }
9  class Two extends One
10 {
11    public void Say()
12    {
13    System.out.println("This Msg from class Two");
14
15    }
16 }
17 public class Override {
18
19    public static void main(String[] args) {
20
21
22    }
23 }
```

في هذه السطور:

- في السطر رقم 1 تم تعريف فصيلة جديدة بالاسم one.
- في السطر رقم 2 تم تعريف دالة بالاسم ()say مع وضع الكلمة overiridable أمام الدالة وهذا يعني إمكانية التركيب عليها أي إنشاء دالة بنفس الاسم في الفصيلة المشتقة ،ثم تم إنشاء الدالة ()Say في الفصيلة الجديدة برسالة جديدة وإنهاء الفصيلة.
- في السطر رقم 6 تم إنشاء فصيلة بالاسم two.
- في السطر رقم 8 تم إنشاء دالة للفصيلة الجديدة بنفس الاسم say وهو اسم الدالة الموجودة في الفصيلة الأساس التي تم توريثها(one) مع وضع الكلمة overrides أمام الدالة ليعني انها بديل الدالة موجودة في الفصيلة الأساس.

- في السطر رقم 7 تم توريث الفصيلة القديمة one .
- بهذا يصبح للفصيلة الأساس (one) دالة بالاسم say .
- ويصبح للفصيلة الجديدة (two) دالة بالاسم say .
- وعند تعريف متغير من كل منهما واستدعاء الدالة say يتم استدعاء الدالة الخاصة بكل منهما بالرغم من وجود خاصية التوريث بها.

ولتوضيح ذلك تابع ما يلي :

بعد كتابة سطور الفصيلتان اكتب السطور التالية فى الدالة الرئيسية للبرنامج Sub Main()

```
1: Dim a as new one ( )
2: Dim b as new two ( )
3: a. say ( )
4: b. say ( )
```

فى لغة C#

```
8       class Class1
9       {
10          /// <summary>
11          /// The main entry point for the application.
12          /// </summary>
13          [STAThread]
14          static void Main(string[] args)
15          {
16  one a = new one ( );
17  two b = new two ( );
18   a. say  ( );
19   b. say ( );
20
21          }
22      }
```

في لغة Java

```
17 public class Override {
18
19   public static void main(String[] args) {
20
21     One a=new One();
22     Two b=new Two();
23     a.Say();
24     b.Say();
25   }
```

في هذه السطور:

- في السطر رقم 1،2 المتغيرات a , b من الفصائل one ,two.
- في السطر رقم 3 تم استدعاء الدالة say مع المتغير a لمأخوذ في الفصيلة one وبالتالي يتم استدعاء الدالة الأولي وتظهر الرسالة الأولي.
- في السطر رقم 4 يتم استدعاء الدالة say مع المتغير a المعرف من الفصيلة b وبالتالي يتم استدعاء الدالة الثانية وتظهر الرسالة الثانية بالرغبة في توريث الفصيلة الأولي المحتوية علي الدالة say للفصيلة الثانية الا إن إنشاء دالة جديدة بنفس الاسم أدي إلي إلغاء الدالة say مع الفصيلة الجديدة واستعمال الدالة الخاصة بها.

الأمر Mustoverride

هذا الامر عندما يوضع أمام دالة في الفصيلة فإنه يعني عدم إمكانية استدعاء هذه الدالة مباشرة بل لابد من تعريف دالة جديده بنفس الاسم حتى يمكن استعمالها وبالتالي فهذه الدالة تكتب للتعريف فقط.

درجات حماية أعضاء الفصيلة Access Modifiers

عند الإعلان عن أعضاء الفصيلة سواء بيانات data أو دوال Methods لاحظنا إننا كنا نكتب كلمة public قبل الإعلان هذه الكلمة تحدد درجة حماية للاعضاء وتسمى Access Modifiers وكلمة public تعنى عام أى يمكن التعامل معه مباشرة من خارج الفصيلة بدون الاحتياج لى دالة من الفصيلة للتعامل معه.

ربما يكون هذا الشرح غير واضح لأنه نظرى ولكن تعال أولا نعرض لك الكلمات المختلفة التى يمكن أن تستعمل مكان الكلمة public لتحديد درجة حماية الاعضاء والتى تسمى كما أشرنا Access Modifiers وهى:

1- الكلمة Public كما أشرنا تستعمل لتحديد أن هذه الاعضاء عامة يمكن استعمالها من خارج الفصيلة أو يمكن الوصول اليها من خارج الفصيلة.
2- الكلمة Private عكس الكلمة Public وتعنى خاص وتعنى أن هذه الاعضاء لا يمكن الوصول اليها من خارج الفصيلة فقط يتم التعامل معها من داخل الفصيلة و من خلال أعضاء الفصلة.
3- الكلمة Protected هى نفس عمل الكلمة private عدا فى حالة الفصائل المشتقة تستطيع التعامل معها.

بعد سرد معانى الكلمات الثلاثة تعال نوضح ذلك عمليا

1- ارجع إلى البرنامج السابق والفصيلة Person والتى تم فيها الإعلان عن متغيرات سبقت بالكلمة Public كما فى السطر التالى:

```
Public Pname, Paddress As String
```

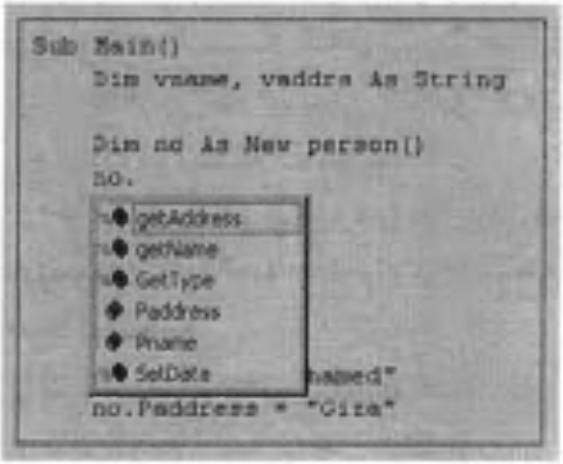
فى هذه الحالة يمكن التعامل مع هذه المتغيرات خارج الفصيلة بمجرد تعريف هدف Object من الفصيلة class كما فى السطور التالية:

```
Dim no As New person()
no.Pname = "mohamed"
no.Paddress = "Giza"
```

فى هذه السطور تم التعامل مباشرة مع المتغيرين Pname,Paddress.

وعندما يتم الإعلان عن الأعضاء بالكلمة Public تظهر هذه الأعضاء مع الهدف عند

استعماله ، إرجع الى تعريف الهدف Object ثم استعماله تلاحظ بمجرد كتابة اسم الهدف object ثم كتابة نقطة . تظهر الأعضاء الممكن التعامل معهم كما فى الشكل (1-8).



الشكل (1-8)

فى هذا الشكل تلاحظ ظهور جميع أعضاء الفصيلة Person لتعريفهم بالكلمة Public.

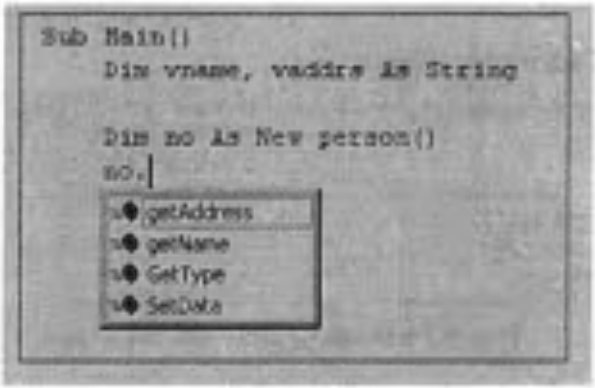
2- استبدل الكلمة Public بالكلمة Private عند تعريف المتغيرات كما فى السطر التالى:

```
Private Pname, Paddress As String
```

فى لغة C#

```
Private string  Pname, Paddress;
```

جرب التعامل مع هدف الفصيلة بنفس الطريقة السابقة بالتعامل المباشر مع المتغيرات تلاحظ أن المتغيرات لا تظهر ضمن قائمة الأعضاء كما فى الشكل (2-8):



الشكل (2-8)

فى هذا الشكل تلاحظ عدم وجود المتغيرات التى تم الإعلان عنها داخل الفصيلة person بالرغم من أنهم أعضاء بها ، ذلك للاعلان عنهم بالكلمة Private .

فى هذه الحالة لا يمكن الوصول إليهم من خارج الفصيلة إلا من خلال دالة Method عضوة فى الفصيلة كما سبق فى الدوال التى عرفناها كما فى السطور التالية:

```
Public Function getName()
   Return Pname

End Function

Public Function getAddress()
   Return Paddress
End Function
```

ملحوظة:

لا يتم الإعلان عن الدوال بالكلمة Private وإلا لن يمكن التعامل معها من خارج الفصيلة وبهذا تفقد أهميتها إلا إذا كنت تريد انشاء دوال للتعامل معها داخل المصنوفة فقط.

متى تستعمل خاصية التوريث ومتى لا نستعملها ؟

هذا السؤال ربما يرد إلى ذهنك عندما تعمل مع الفصائل وإجابته كما يلي:

إذا كانت الوظيفة المطلوبة تحقيقها ليست بكبيرة بحيث لا تحتاج فصيلة جديدة تقوم فيها بإنشاء دوال و تعريف متغيرات فلا تستخدم خاصية التوريث بل تكتب الأوامر التي تحقق العملية فقط ، مثال لذلك إذا كان لديك أداة نص textbox و تريد تحويل لون الأرقام السالبة عندما تكتب به إلى اللون الأحمر فلا تقوم بتوريث فصيلة textbox الى فصيلة جديدة ، وتعريف دالة لذلك ، ولكن اكتب الأوامر التي تحقق العملية لأنها لا تحتاج إلى فصيلة.

إذا كانت الفصيلة الجديدة (التي تستورث فصيلة قديمة) بها لن تستفيد كثيراً من الفصيلة الأساس (المستورثة) فقط سوف تستورثها لإنشاء دوالها من جديد بخاصية override فهذه الحالة يكون التوريث فيها غير جيد.

ولكن يمكنك إنشاء فصيلة بدوال بدون كتابة سطور الدوال (interface) ثم توريثها و بالتالي ننشأ الفصائل الجديدة حسب تركيب (interface) وخاصة إذا كان هناك أكثر من فصيلة سوف نشترك في التركيب فقط.

إذا كانت الفصيلة الجديدة (التي تستورث) تحتاج إلي دوال الفصيلة القديمة بنفس السطور المكتوبة فيها وتحتاج إلي الخواص والمتغيرات ولكن تريد الإضافة إليها فهذه هي الحالة المناسبة للتوريث وهو الاستفادة الكبيرة من الفصيلة المستورثها وإضافة الجديد بما يناسب العمل .