

الفصل الرابع الفصائل والكائنات

سنقوم في هذا الفصل بالتعرف على المبادئ الأساسية للفصائل **Classes** وكيفية الإعلان عنها ومكوناتها المختلفة، حيث تعتبر الفصائل من العناصر الأساسية في تقنية البرمجة الموجهة بالكائنات **OOP**.

بالانتهاء من هذا الفصل ستكتسب المعارف وتندرب على المهارات التي تجعلك قادراً على:

- مفهوم الفصائل
- البرمجة الموجهة بالكائنات **Object Oriented Programming**
- خطوات إنشاء الفصيل
- تحديد خصائص الفصيل **Class Properties**
- استخدام الفصيل داخل التطبيقات
- إجراءات الخصائص.
- الإعلان عن الكائنات وحذفها **Object Creation**

كلما قمت بتطوير المزيد من التطبيقات، كلما اكتشفت أنك في حاجة إلى إعادة استخدام بعض الإجراءات الموجودة بأحد هذه التطبيقات داخل تطبيق آخر. وبدلاً من أن تقوم بكتابة هذه الإجراءات مرةً أخرى، ربما أوصلك تفكيرك بنسخ هذه الإجراءات من التطبيق الأصلي ثم لصقها مرةً أخرى في التطبيق الجديد. لكن يعاب على هذه الطريقة أنك إذا

قمت بلصق الإجراء داخل أكثر من تطبيق ثم اكتشفت بعد ذلك وجود خطأ ما بهذا الإجراء، وجب عليك في هذه الحالة تصحيح الخطأ في كل إجراء على حده سواءً في التطبيق الأول أو في التطبيقات الجديدة. وهذا أمرٌ ممل للغاية ويؤدي إلى العديد من المشاكل والصعاب. والأفضل في هذه الحالة هو كتابة الكود الذى ترغب فى إعادة استخدامه داخل قطعة من الكود **Code Component** تسمى **Class** وهو ما اصطلحنا على تسميته فصيلاً، حيث يحتوى الفصيل على الإجراءات والدوال التى يمكنك استخدامها أكثر من مرة وبالتالى يمكنك تحقيق مفهوم البرمجة الموجهة بالكائنات **Object Oriented Programming (OOP)** المبنية أساساً على استخدام الفصائل. سنقوم فى هذا الفصل بالتعرف على كيفية إنشاء الفصائل واستخدامها داخل **Visual Basic 2008**، فكن معنا.

مفهوم الفصائل (Classes)

قمنا فيما سبق من فصول الكتاب بإنشاء العديد من التطبيقات أو المشروعات، وأثناء ذلك قمنا بإدراج العديد من النماذج وأدوات التحكم إلى هذه التطبيقات وحينئذٍ كنا على موعد دون أن تدرى بالفصائل **Classes** والكائنات **Objects**. فعلى سبيل المثال حينما قمت بإدراج مربع نص **TextBox** من مربع الأدوات إلى النموذج، فأنت بذلك قمت بإنشاء حالة **Instance** (أو كائن **Object**) من الفصيل **TextBox** دون أن تدرى. وعلى ذلك إذا قمت برسم خمسة مربعات نصوص على النموذج، تكون بذلك قد أنشأت خمس حالات من الفصيل **TextBox**. كما أن النموذج المستخدم نفسه عبارة عن حالة جديدة من الفصيل **Form**.

يطلق على حالات الفصيل "كائنات" **Objects**. فكل فصيل عبارة عن قالب يتم منه إنشاء عدد من الكائنات المتشابهة الوظيفة. فعلى سبيل المثال الفصيل **TextBox** عبارة عن قالب لا يحتوى على أية بيانات داخل الخصائص، أما الكائنات المنبثقة من هذا

الفصيل مثل مربع النص `txtFName` مثلاً فيحتوى على بيانات خاصة بكل خاصية من خصائصه، فهو يحتوى مثلاً على القيمة `txtFName` داخل الخاصية `Name`.

البرمجة الموجهة بالكائنات

البرمجة الموجهة بالكائنات (**Object Oriented Programming (OOP)**) أسلوب تعتمد عليه العديد من لغات البرمجة مثل **Smalltalk** ، **Ada** ، **Java** ، **C++** ، **C#** و **Visual Basic**، ويهدف هذا الأسلوب لفصل البرنامج إلي أجزاء منفصلة وظيفياً وشكلياً تسمى بالكائنات **Objects** تعمل باستقلال تام وإن أرادت التعاون مع غيرها من الكائنات خاطبتها من خلال ما يسمى بواجهة التخاطب للكائن **Interface**. أهم ما يميز البرمجة بالكائنات، استخدام كائنات يمكن إعادة استخدامها بحيث تدعم المفاهيم التالية:

الاحتوائية Encapsulation : لا بد أن يحتوي الكائن على المعلومات الموصفة له ، مضافاً إليها الأساليب المستخدمة في معالجة هذه المعلومات. تسمى هذه الخاصية أحياناً بـ "اختباء البيانات" **Information Hiding**.

الوراثة Inheritance : يمكن إنشاء كائن جديد من كائن موجود و يرث الكائن الجديد كافة خصائص الكائن الأصلي والذي يسمى "والد" **parent**.

تعدد الأشكال polymorphism : رغم السماح بوجود نفس الوظيفة **method** في العديد من الكائنات (حتى بين الكائنات المشتقة من كائنات أخرى)، إلا أن كل كائن يقوم بهذه الوظيفة بشكل مختلف. مثال على ذلك العامل + يمكن استخدامه لكل من الأعداد الحقيقية والصحيحة، رغم اختلاف التمثيل الداخلي لكل منهما إلا أن البرنامج سيقوم بتنفيذ الأسلوب المناسب عند إجراء عملية الجمع الفعلية.

مكونات الفصيل

يمكنك إنشاء فصائلك داخل Visual Basic 2008 بإضافة قالب الفصيل إلى مشروع موجود مسبقاً أو بإنشاء مكتبة فصائل Class Library تحتوى على فصيل أو أكثر. تحتوى الفصائل دائماً على كوكبة من العناصر الثلاثة الآتية:

- الخصائص Properties وتستخدم في تخصيص القيم إلى الفصيل واسترجاعها.
- الطرق Methods وهى الدوال أو الإجراءات العامة المعرفة داخل الفصيل.
- الأحداث Events كما تنشأ الأحداث من الأدوات الموجودة داخل النماذج، فمن الممكن أن يقوم الكائن المنبثق من فصيلك بإنشاء الأحداث أيضاً. كما تحتوى الفصائل افتراضياً على حدثين، أحدهما هو الحدث New الذى ينشأ تلقائياً بمجرد إنشاء حالة جديدة (كائن) من حالات الفصيل، والآخر هو الحدث Finalize الذى ينشأ تلقائياً بمجرد تدمير الكائن أو حذفه.

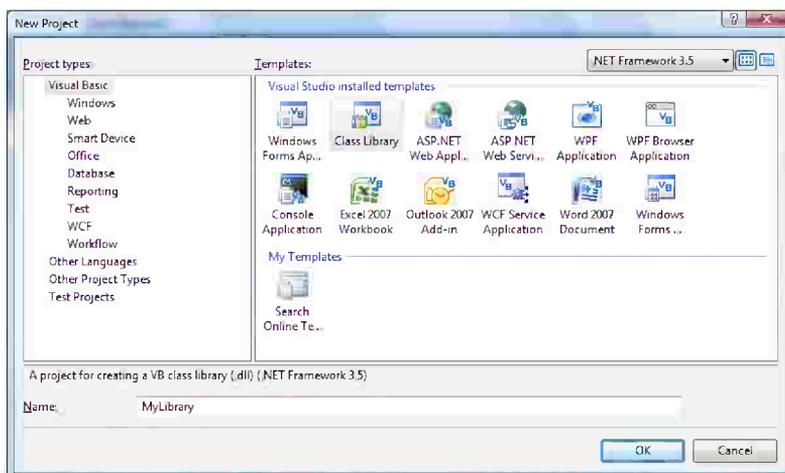
إنهاء الفصيل

يتم إنشاء تعريف الفصيل داخل وحدة نمطية خاصة بالفصيل أو ما يسمى Class Module الذى يتشابه كثيراً مع الوحدة النمطية الخاصة بالنموذج غير أنه يحتوى على تعريفات وامتغيرات فقط بينما لا يحتوى على أدوات تحكم أو واجهات. سنقوم فيما يلى بالتعرف على كيفية إنشاء فصيل جديد واستخدامه من خلال مثال عملي يحتوى على نوع بيانات جديد يتضمن بيانات العاملين فى شركة كمبيوساينس. تابع معنا الخطوات الآتية:

١. افتح قائمة File من شريط القوائم ثم اختر New Project من القائمة المنسدلة، يظهر المربع الحوارى المعتاد New Project (انظر شكل ٤-١).
٢. من عمود القوالب Templates الموجود بالجزء الأيمن من المربع الحوارى، اختر Class Library.

٣. قم بتعيين اسم مميز لمكتبة الفصائل الجديدة داخل مربع النص **Name** وليكن **.MyLibrary**.

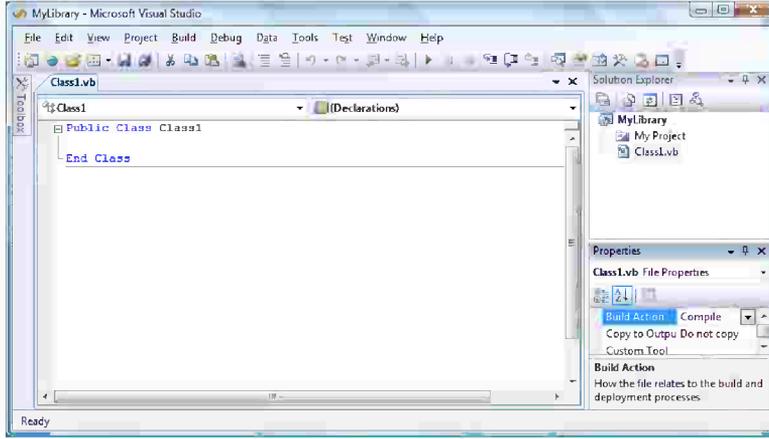
٤. انقر زر **OK**، تظهر مكتبة الفصائل الجديدة داخل مشروع جديد محتويةً على ملف واحد باسم **Class1.vb** وهو الملف الذي سيتم كتابة كود الفصيل بداخله (انظر شكل ٤-٢).



شكل ٤-١ إنشاء مكتبة الفصائل باستخدام المربع الحوارى **New Project**

يمكنك تضمين عدة فصائل داخل مكتبة الفصائل التى قمنا بإنشائها. لأداء ذلك، افتح قائمة **Project** من شريط القوائم ثم اختر **Add Class** من القائمة المنسدلة. إلا أننا للتبسيط سنستخدم فصيلاً واحداً فقط.





شكل ٤-٢ مكتبة الفصائل الجديدة داخل بيئة التطوير

٥. لتغيير اسم الفصيل إلى اسم معبر بدلاً من **Class1**، قم بتغيير كلمة **Class1** داخل نافذة الوحدة النمطية الخاصة بالفصيل إلى اسم معبر وليكن **clsEmployee**، حيث تعني **cls** كلمة **Class** كي يسهل التعرف على الفصيل فيما بعد داخل الكود. وهكذا يصبح السطر الأول بالوحدة النمطية هو **Public Class clsEmployee** بدلاً من **Public Class Class1**.

٦. لتغيير اسم ملف الفصيل، انقر اسم الملف داخل نافذة المستكشف مرتين بينهما فاصل بسيط ثم قم بكتابة الاسم الجديد وليكن **clsEmployee.vb**. وبهذا نكون قد انتهينا من البنية التحتية للفصيل الجديد.

إضافة الخصائص إلى الفصيل *Class Properties*

بعد إنشاء الوحدة النمطية الخاصة بالفصيل، يمكنك الآن إضافة الخصائص إليها، وهذه الخصائص تتشابه كثيراً مع تلك الموجودة بالأدوات المعتادة والتي تعاملت معها حتى الآن، فهي تستخدم في تسجيل البيانات واسترجاعها. يمكنك في الواقع إضافة الخصائص إلى فصيلك بطريقتين، الأولى باستخدام المتغيرات العامة **Public Variables** والثانية باستخدام إجراءات الخصائص **Property Procedures**.

المتغيرات العامة

تعتبر هذه الطريقة أبسط طرق إنشاء الخصائص. ففي حالة الفصيل `clsEmployee` الذى بين أيدينا، يمكنك إضافة بعض الخصائص البسيطة من خلال المتغيرات العامة. قم بكتابة الكود التالى داخل نافذة كود الفصيل وأسفل السطر `Public Class clsEmployee` مباشرةً:

```
Public FirstName As String
Public LastName As String
Public BirthDate As String
```

قد يحتوى الفصيل على متغيرات خاصة يتم استخدامها داخل الفصيل فقط، ولكن تبقى المتغيرات العامة هى الوحيدة التى يمكن مشاهدتها والتعامل معها خارج نطاق الفصيل.



وهكذا يمكنك فيما بعد حينما تقوم بإنشاء كائنات (حالات) من الفصيل أن تقوم بتخصيص القيم إلى خصائص هذه الكائنات كما يلي:

```
MyObject.FirstName = "Waleed"
MyObject.LastName = "Abdelrazek"
MyObject.BirthDate = #18/1/1976#
lblEmployeeInfo.Text = MyObject.FirstName & " " &
MyObject.LastName
```

ويعاب على هذه الطريقة عدم القدرة على التحقق من صحة البيانات التى يتم إدخالها، فطالما كانت البيانات المدخلة من نفس نوع بيانات المتغير، سيتم قبول هذه البيانات. فلا يمكنك مثلاً اشتراط أن يكون تاريخ الميلاد الممثل فى الخاصية `BirthDate` فى مدى معين وإنما يتم قبول أى تاريخ.

إجراءات الخصائص

تعتبر هذه الطريقة أكثر مرونة من الطريقة السابقة وذلك لقدرتها على تنفيذ كود معين بمجرد استدعاء الخاصية وهو القصور الموجود بالطريقة السابقة كما أسلفنا منذ قليل. يمكنك كتابة إجراءات الخصائص بنفس طريقة كتابة الدوال العادية، إلا أنها تعمل بالنسبة للمستخدم بنفس طريقة عمل الخصائص العادية، فهى تمكن المستخدمين من استرجاع

بيانات الخاصية من خلال الأداة **Get** وكذلك تخصيصها بالبيانات من خلال الأداة **Set** وسوف نقوم بشرح الأدوات بالتفصيل لاحقاً في فصل آخر من فصول الكتاب عند حديثنا عن إنشاء أدوات التحكم المخصصة.

وكما ذكرنا من قبل يمكنك استخدام المتغيرات العامة في إنشاء الخصائص البسيطة والتي لا يكون هناك أى قيود على قيمها. لكن تخيل أننا نرغب في إنشاء الخاصية **Salary** التي تحتوى على مرتب الموظف، إذا قمنا بإنشاء هذه الخاصية من خلال متغير عام، فقد يقوم المستخدم بإدخال قيمة سالبة للمرتب وهذا غير صحيح بالمرة. لذا يكون من الضروري إنشاء مثل هذه الخاصية من خلال إجراء وذلك لكتابة الكود اللازم للتأكد من صحة البيانات المدخلة إلى الخاصية.

لإنشاء الخاصية **Salary** باستخدام إجراء، سنحتاج إلى تعريف متغير خاص يحتوى على قيمة الخاصية داخل الخاصية نفسها دون أن يظهر إلى المستخدم العادى وليكن باسم **dSalary** كما يوجد بالإجراء جزأين أساسيين، أحدهما الجزء **Get** الذى يمكن المستخدم من استرجاع قيمة الخاصية والموجودة أساساً داخل المتغير **dSalary** والجزء **Set** الذى يستقبل القيمة من المستخدم ويقوم بتمريرها إذا كانت صحيحة إلى الخاصية **dSalary**. قم بالإعلان عن المتغير **dSalary** أسفل مجموعة المتغيرات العامة التى قمت بإدخالها منذ قليل كما يلى:

Private Shared dSalary As Decimal

والآن قم بالإعلان عن إجراء الخاصية **Salary** كما يلى:

Public Property Salary() As Decimal

اضغط مفتاح الإدخال، تقوم بيئة التطوير نيابةً عنك بكتابة الهيكل العام لإجراء الخاصية كما يلى (انظر شكل ٤-٣):

Public Property Salary() As Decimal

Get

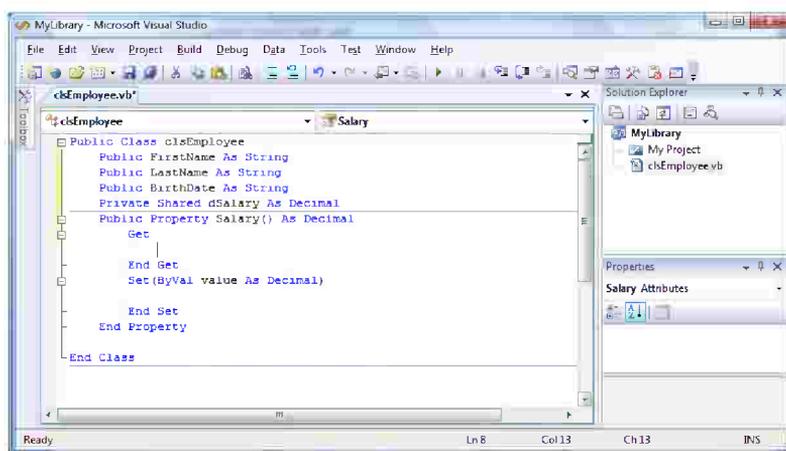
End Get

Set(ByVal Value As Decimal)

End Set
End Property

قم بإدخال كود الإجراء اللازم لإدخال قيمة الخاصية من خلال الجزء **Set** أو استرجاعها من خلال الجزء **Get** كما يلي:

```
Public Property Salary() As Decimal  
Get  
    Return dSalary  
End Get  
Set(ByVal Value As Decimal)  
    dSalary = Value  
End Set  
End Property
```



شكل ٤-٣ يقوم Visual Basic بإنشاء هيكل إجراء الخاصية نيابةً عنك بمجرد ضغط مفتاح الإدخال.

FullName خاصية الاسم الكامل

كى يتمكن المستخدم فى أى وقت من استرجاع الاسم الكامل للموظف، سنقوم بإنشاء خاصية للقراءة فقط **Read Only** باسم **FullName** تقوم أساساً بدمج الاسم الأول مع الاسم الأخير كما يلي:

```
ReadOnly Property FullName() As String  
Get
```

```

Return Trim(FirstName & " " & LastName)
End Get
End Property

```

خاصية القسم *Department*

بنفس الطريقة سنقوم بإنشاء خاصية القسم الذى ينتمى إليه الموظف داخل الشركة وذلك باستخدام متغير خاص داخلى باسم `nDeptNum`. قم بإضافة إعلان المتغير فى قسم الإعلانات الموجود فى بداية الملف كما يلى:

```
Private Shared nDeptNum As Integer
```

قم الآن بإدخال إجراء الخاصية كما يلى:

```
Public Property Department() As String
```

```

Get
    Select Case nDeptNum
        Case 1
            Return "Sales"
        Case 2
            Return "Technical Support"
        Case 3
            Return "Accounting"
        Case Else
            Return "Compuscience"
    End Select
End Get
Set(ByVal Value As String)
    Select Case Trim(UCase(Value))
        Case "SALES"
            nDeptNum = 1
        Case "TECKNECAL SUPPORT"
            nDeptNum = 2
        Case "ACCOUNTING"
            nDeptNum = 3
        Case Else
            nDeptNum = 0
    End Select
End Set
End Property

```

وكما ترى يتم تخزين الأقسام داخل أرقام إلا أن المستخدم يقوم باسترجاعها فى صورة سلاسل بيانات من خلال العبارة **Select Case**. وقد قمنا هنا باستخدام ثلاثة أقسام فقط من أقسام الشركة للتبسيط، ولكن عملياً تحتاج إلى تخزين الأقسام داخل قاعدة بيانات كى تتيح للمستخدم الاختيار من بينها أو حتى تعريف أقسام جديدة.

خاصية المسئول *Supervisor*

لأن مسئول القسم أو مدير القسم أو حتى مدير الشركة ما هو إلا أحد الموظفين الموجودين بالشركة، فمن الضروري إنشاء خاصية تميزه عن بقية الموظفين. لذلك قم بتعريف متغير خاص من نوع الفصيل باسم **objSupervisor** كما يلي:

```
Private objSupervisor As clsEmployee
```

قم بإضافة إجراء الخاصية كما يلي:

```
Property Supervisor() As clsEmployee
```

```
Get
```

```
Return objSupervisor
```

```
End Get
```

```
Set(ByVal Value As clsEmployee)
```

```
objSupervisor = Value
```

```
End Set
```

```
End Property
```

إذا حاولت فيما بعد الوصول إلى الخاصية **Supervisor** لإحدى حالات الفصيل **clsEmployee** التى لا تحتوى على قيمة لهذه الخاصية، ستحصل على خطأ. يمكنك التغلب على ذلك من خلال الكود باستخدام عبارة **If ... Else** كما يلي:



```
If objEmp.Supervisor Is Nothing Then code Else code
```

إضافة دالة إلى الفصيل

كما تعلم فإن الكائنات تقوم بتنفيذ الأحداث من خلال الطرق **Methods** أو الدوال مثل الوظيفة **Add** المصاحبة لمربع السرد أو الوظيفة **Focus** المصاحبة لزر الأمر وهكذا. يمكنك بنفس الطريقة إنشاء الطرق داخل فصيلك من خلال إضافة الإجراءات والدوال

العامه إلى الفصيل. لتوضيح ذلك فى فصيلنا الحالى، سنقوم بإنشاء دالة باسم **ChangeSalary** تعمل على تغيير مرتب الموظف من خلال نسبة مئوية سالبة أو موجبة يتم تمريرها إلى هذه الدالة. قم بإدخال الكود التالى إلى نافذة كود الفصيل **:clsEmployee**

```
Public Function ChangeSalary(ByVal salPer As Decimal)
    If (salPer > 1) Or (salPer < -1) Then
        salPer = salPer / 100
    End If
    Salary = Salary * (1 + salPer)
End Function
```

إضافة حدث إلى الفصيل

تتسبب الكائنات دائماً كما تعلم فى ظهور الأحداث **Events** مثل الحدث **Click** الذى ينشأ من نقر المستخدم لزر الأمر. يمكنك من خلال الأحداث تنفيذ جزء من الكود يسمى "معالج الحدث" أو **Event Handler**. ولكى تقوم بإنشاء حدث جديد داخل الفصيل، تحتاج إلى شيئين، الأول هو تعريف الحدث داخل الفصيل والثانى استخدام العبارة **RaiseEvent** لاستدعاء هذا الحدث وقت الحاجة.

دعنا الآن نقوم بإضافة الحدث **DataError** الذى ينتج عن إدخال المستخدم لقيمة سالبة بالمتغير **Salary**. قم بالإعلان عن الحدث فى جزء الإعلانات العلوى بنافذة كود الفصيل **clsEmployee** كما يلى:

```
Public Event DataError(ByVal sErrorMsg As String)
```

قم أيضاً بتعديل كود الخاصية **Salary** كما يلى:

```
Public Property Salary() As Decimal
    Get
        Return dSalary
    End Get
    Set(ByVal Value As Decimal)
        If Value >= 0 Then
            dSalary = Value
```

```
Else
    RaiseEvent DataError("Salary cannot be negative.")
End If
End Set
End Property
```

وهكذا إذا قام المستخدم بإدخال قيمة سالبة للمرتب، يتم إظهار رسالة الخطأ الموضحة داخل الخاصية.

لا يقتصر الحدث السابق على المرتب فقط، وإنما يمكنك استخدامه للتأكد من صحة البيانات الأخرى كتاريخ الميلاد مثلاً. فيمكنك استخدام الحدث مع الخاصية Birthdate كما يلي:



```
RaiseEvent DataEvent("Invalid Birth Date")
```

إجراء إنشاء الفصيل

حينما يتم إنشاء حالة جديدة من الفصيل أى تعريف كائن جديد ينتمى إلى الفصيل، يتم استدعاء إجراء يسمى "إجراء الإنشاء" أو Constructor حيث يمكنك كتابة الكود الذى ترغب فى تنفيذه بمجرد تعريف الكائن. فربما أردت معرفة عدد الكائنات التى تنتمى إلى الفصيل مثلاً. لتوضيح ذلك، قم بإضافة التعريفات التالية إلى قسم الإعلانات بنافذة كود الفصيل clsEmployee:

```
Public Shared ClassInstanceCount As Long = 0
Private Shared NextInstanceID As Integer = 1
Public ReadOnly InstanceID As Integer
```

حيث:

- سنقوم باستخدام المتغير ClassInstanceCount لتخزين عدد حالات (كائنات) الفصيل clsEmployee. وهو كمتغير مشترك Shared Variable يكون موجود فقط داخل مستوى الفصيل نفسه، وبالتالي ترى كل حالة من الحالات نفس النسخة من المتغير.
- سنقوم باستخدام المتغير NextInstanceID لتخزين القيمة التى سيتم تخصيصها للمتغير InstanceID والتى تعبر عن الحالة التالية التى سيتم إنشائها.

- سنقوم باستخدام المتغير InstanceID لتخزين رقم الحالة الحالية من الفصيل .clsEmployee

يعتبر الإجراء Sub New إجراء الإنشاء الافتراضي للفصيل والذي يتم تنفيذه كلما تم إنشاء حالة جديدة من حالات الفصيل. لإنشاء هذا الإجراء، اختر New من مربع الأحداث بنافذة كود الفصيل ثم قم بإضافة الكود التالي للإجراء:

```
Public Sub New()
    InstanceID = NextInstanceID
    NextInstanceID += 1
    ClassInstanceCount += 1
End Sub
```

يمكنك بالطبع إضافة أى كود آخر إلى الإجراء.

إجراء هدم الفصيل

يمكنك بنفس الطريقة استخدام الإجراء Finalize الذى يتم تنفيذه تلقائياً بمجرد حذف أحد الكائنات التى تنتمى للفصيل أو هدمه. لإنشاء هذا الإجراء، اختر Finalize من مربع الأحداث بنافذة كود الفصيل ثم قم بإضافة الكود التالي للإجراء:

```
Protected Overrides Sub Finalize()
    MyBase.Finalize()
    ClassInstanceCount - = 1
End Sub
```

حيث يقوم الإجراء بتقليل عدد الكائنات بمقدار ١ وهو الكائن الذى تم حذفه.

وبهذا يكون الكود الكامل للفصيل كما يلى:

```
Public Class clsEmployee
    Public FirstName As String
    Public LastName As String
    Public BirthDate As String

    Private Shared dSalary As Decimal
    Private Shared nDeptNum As Integer
    Private objSupervisor As clsEmployee
    Public Shared ClassInstanceCount As Long = 0
    Private Shared NextInstanceID As Integer = 1
```

```
Public ReadOnly InstanceID As Integer
Public Event DataError(ByVal sErrorMsg As String)

Public Property Salary() As Decimal
    Get
        Return dSalary
    End Get
    Set(ByVal Value As Decimal)
        If Value >= 0 Then
            dSalary = Value
        Else
            RaiseEvent DataError("Salary cannot be negative.")
        End If
    End Set
End Property
ReadOnly Property FullName() As String
    Get
        Return Trim(FirstName & " " & LastName)
    End Get
End Property
Public Property Department() As String
    Get
        Select Case nDeptNum
            Case 1
                Return "Sales"
            Case 2
                Return "Technical Support"
            Case 3
                Return "Accounting"
            Case Else
                Return "Compuscience"
        End Select
    End Get
    Set(ByVal Value As String)
        Select Case Trim(UCase(Value))
            Case "SALES"
                nDeptNum = 1
            Case "TECKNECAL SUPPORT"
                nDeptNum = 2
```

```

        Case "ACCOUNTING"
            nDeptNum = 3
        Case Else
            nDeptNum = 0
        End Select
    End Set
End Property
Property Supervisor() As clsEmployee
    Get
        Return objSupervisor
    End Get
    Set(ByVal Value As clsEmployee)
        objSupervisor = Value
    End Set
End Property
Public Function ChangeSalary(ByVal salPer As Decimal)
    If (salPer > 1) Or (salPer < -1) Then
        salPer = salPer / 100
    End If
    Salary = Salary * (1 + salPer)
End Function

Public Sub New()
    InstanceID = NextInstanceID
    NextInstanceID += 1
    ClassInstanceCount += 1
End Sub

Protected Overrides Sub Finalize()
    MyBase.Finalize()
    ClassInstanceCount -= 1
End Sub
End Class

```

استخدام الفصيل داخل التطبيقات

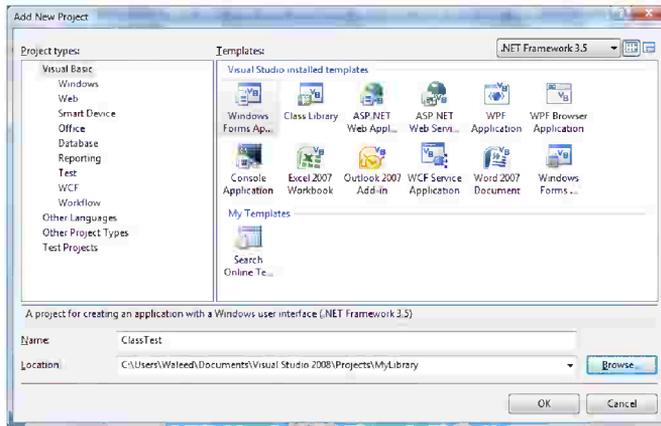
لا فائدة من إنشاء الفصيل الجديد إلا إذا قمت باستخدامه داخل أحد التطبيقات. ولكي تستخدم الفصيل داخل التطبيق، يجب أن تقوم بإنشاء مرجع إلى المشروع الذي يحتوي على الفصيل داخل المشروع الذي ترغب في استخدام الفصيل بداخله. سنقوم أولاً بإضافة مشروع جديد إلى الحل **Solution** الحالي ثم تعيين مشروع البدء وإضافة المرجع إلى الفصيل وإنشاء الكائنات الجديدة، فكن معنا.

إضافة المشروع الجديد

حينما قمنا بإنشاء مكتبة الفصائل السابقة باسم **MyLibrary**، تم إنشاء مشروع بنفس الاسم داخل حل **Solution** بنفس الاسم أيضاً. وبدلاً من أن نقوم بإنشاء حل جديد، سنقوم بإضافة مشروع جديد إلى نفس الحل. لأداء ذلك، تابع معنا الخطوات الآتية:

١. قم بحفظ التعديلات التي قمت بإجرائها على الحل باختيار **Save All** من القائمة **.File**.

٢. افتح قائمة **File** من شريط القوائم ثم اختر **Add** ثم **New Project** من القوائم المنسدلة، يظهر المربع الحوارى الشهير **Add New Project** (انظر شكل ٤-٤).



شكل ٤-٤ المربع الحوارى **Add New Project**.

٣. اختر **Windows Forms Application** من عمود القوالب **Templates** ثم قم بتعيين اسم مناسب للمشروع الجديد داخل مربع النص **Name** وليكن **ClassTest** ثم انقر زر **Ok**، تلاحظ إضافة المشروع الجديد إلى الحل **.MyLibrary**

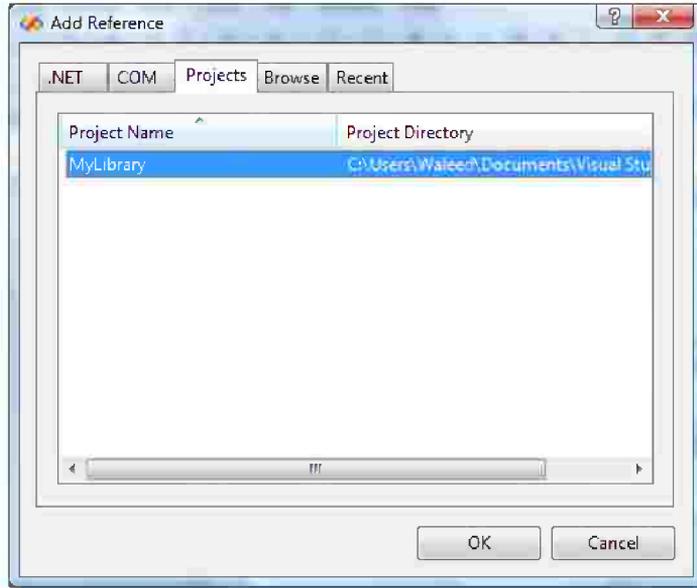
تعيين مشروع البدء

أصبح الآن لدينا مشروعان داخل الحل، أحدهما المشروع **MyLibrary** الذى يحتوى على الفصيل والآخر هو المشروع **ClassTest** الذى سنقوم باستخدام الفصيل بداخله. وإذا أمعنت النظر داخل نافذة مستكشف الحل لوجدت أن المشروع **MyLibrary** يظهر بخط ثقيل بينما يبدو المشروع الجديد بخط عادى خفيف وهذا يدل على أن المشروع الأول هو مشروع البدء أى المشروع الذى يتم تنفيذه بمجرد بدء تشغيل الحل. ولأن الفصيل لا يمكن أن يعمل بمفرده، فنود أن نبدأ التنفيذ بالمشروع الجديد. لأداء ذلك، انقر المشروع **ClassTest** داخل نافذة المستكشف بزر الفأرة الأيمن ثم اختر **Set as StartUp Project** من القائمة الموضوعية وبالتالي يكون المشروع **ClassTest** هو مشروع البدء ودلالة ذلك ظهوره بخط ثقيل وظهور المشروع الآخر بخط خفيف.

إضافة مرجع الفصيل

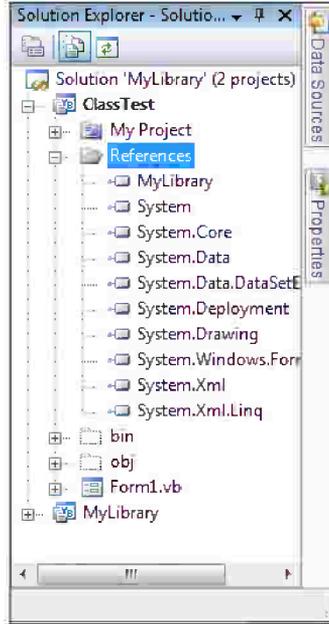
على الرغم من أن المشروع الجديد **ClassTest** جزء من الحل **MyLibrary** الذى يحتوى على مشروع الفصيل، إلا أنك مطالب بإضافة مرجع إلى مشروع الفصيل داخل المشروع الجديد. لإضافة هذا المرجع، تابع معنا الخطوات الآتية:

١. من نافذة المستكشف، انقر المجلد **References** الموجود أسفل المشروع **ClassTest** بزر الفأرة الأيمن ثم اختر **Add Reference** من القائمة الموضوعية (انقر الزر  بشريط أدوات نافذة الحل لإظهار المجلد **References** إذا لم يكن ظاهراً بالفعل)، يظهر المربع الحوارى **Add Reference** (انظر شكل ٤-٥).



شكل ٤-٥ المربع الحوارى Add Reference.

٢. نشط التبويب **Projects** داخل المربع الحوارى إذا لم يكن هو التبويب النشط.
٣. انقر المشروع **MyLibrary** ثم انقر زر **Ok** لإغلاق المربع الحوارى ولاحظ إضافة المشروع **MyLibrary** إلى المجلد **References** بالمشروع **ClassTest** (انظر شكل ٤-٦).



شكل ٤-٦ ظهور المشروع MyLibrary بقائمة مراجع المشروع ClassTest.

الإعلان عن الكائنات وحذفها Object Creation

لاستخدام الفصيل `clsEmployee` الذى قمت بتعريفه من قبل، يجب أن تقوم بإنشاء حالات جديدة من هذا الفصيل داخل تطبيقاتك. يمكنك تعريف هذه الحالات (الكائنات) بنفس طريقة تعريف المتغيرات العادية وكلمة `New` كما يلي:

```
Dim objEmp As New clsEmployee
```

حيث تتسبب كلمة `New` فى حجز الكائن فعلياً. فإذا أردت عدم حجز الكائن وإنما مجرد الإعلان عنه، يمكنك عدم استخدام `New` على أن تستخدمها فيما بعد إذا أردت حجز مساحة للكائن كما فى الكود التالى:

```
Dim objEmp As clsEmployee
```

```
'.....
```

فيما بعد ،

```
Set objEmp = New clsEmployee
```

حذف الكائنات

يمكنك في أي وقت حذف أحد الكائنات لتفريغ موارده المصاحبة وذلك بتخصيص القيمة **Nothing** للكائن كما يلي:

objTemp = Nothing

استخدام الفصيل داخل التطبيق

الخطوة الأولى التي تحتاجها لاستخدام الفصيل **clsEmployee** داخل المشروع **ClassTest** تتمثل في تعريف حالات جديدة تنتمي إلى الفصيل. لأداء ذلك، تابع معنا الخطوات الآتية:

١. من نافذة المستكشف، انقر النموذج **Form1.vb** داخل المشروع **ClassTest** بزر الفأرة الأيمن ثم اختر **View Code** من القائمة الموضعية لإظهار نافذة الكود الخاصة بالنموذج.

٢. قم بكتابة الإعلان التالي داخل نافذة الكود:

```
Dim WithEvents objEmp1 As New MyLibrary.clsEmployee  
Dim WithEvents objEmp2 As New MyLibrary.clsEmployee  
Dim WithEvents objEmp3 As New MyLibrary.clsEmployee
```

وقد استخدمنا كلمة **WithEvents** داخل عبارة **Dim** كي يتمكن الكائن من الاستفادة من الأحداث المخصصة المعرفة داخل الفصيل (الحدث **DataError** في هذه الحالة).

كتابة الأحداث

لأننا استخدمنا كلمة **WithEvents** أثناء الإعلان عن الكائنات، فمن الطبيعي أن يكون هناك حدث مخصص داخل الفصيل وهو الحدث **DataError** الذي يظهر بمجرد إدخال المستخدم قيمة سالبة للخاصية **Salary**. يمكنك الآن تعيين كود الحدث **DataError** الخاص بكل كائن من كائنات الفصيل التي قمنا بالإعلان عنها منذ قليل. لأداء ذلك، تابع معنا الخطوات الآتية:

١. من نافذة كود النموذج Form1.vb اختر الكائن الأول objEmp1 من مربع الكائنات ثم اختر DataError من مربع الأحداث وقم بإدخال كود الحدث وهو عبارة عن مربع رسالة كما يلي:

```
Private Sub objEmp1_DataError(ByVal sErrorMsg As String)
Handles objEmp1.DataError
    MessageBox.Show(sErrorMsg, "Error!",
        MessageBoxButtons.OK, MessageBoxIcon.Warning)
End Sub
```

٢. قم بتكرار الخطوة السابقة مع الكائن objEmp2 والكائن objEmp3 ليصبح كود الحدثين كما يلي:

```
Private Sub objEmp2_DataError(ByVal sErrorMsg As String)
Handles objEmp2.DataError
    MessageBox.Show(sErrorMsg, "Error!",
        MessageBoxButtons.OK, MessageBoxIcon.Warning)
End Sub
```

```
Private Sub objEmp3_DataError(ByVal sErrorMsg As String)
Handles objEmp3.DataError
    MessageBox.Show(sErrorMsg, "Error!",
        MessageBoxButtons.OK, MessageBoxIcon.Warning)
End Sub
```

تعيين خصائص كائنات الفصيل

لتعيين خصائص الكائنات الثلاثة التي قمنا بتعريفها في البند السابق، تأكد من اختيار Form1 بقائمة الكائنات ثم اختر New من قائمة الأحداث وقم بتعديل كود الإجراء كما يلي:

```
Public Sub New()
    MyBase.New()
```

'This call is required by the Windows Form Designer.

InitializeComponent()

'Add any initialization after the InitializeComponent() call

objEmp1.FirstName = "Waleed"

objEmp1.LastName = "Abdelrazek"

```
objEmp1.BirthDate = #1/18/1976#  
objEmp1.Salary = 180  
objEmp1.Department = "Technical Support"
```

```
objEmp2.FirstName = "Sobhy"  
objEmp2.LastName = "Omran"  
objEmp2.BirthDate = #2/3/1972#  
objEmp2.Salary = 180  
objEmp2.Department = "Sales"
```

```
objEmp3.FirstName = "Mokhtar"  
objEmp3.LastName = "Samy"  
objEmp3.BirthDate = #12/1/1975#  
objEmp3.Salary = 180  
objEmp3.Department = "Accounting"
```

```
objEmp1.Supervisor = objEmp2  
objEmp2.Supervisor = objEmp3
```

End Sub

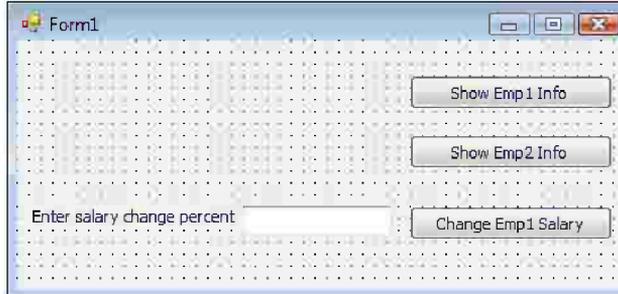
وكما ترى فقد قمنا بتعيين خصائص كل كائن على حده. كما قمنا في السطرين الأخيرين بتخصيص قيمة خاصية **Supervisor** لكل من الكائن الأول والكائن الثاني.

تجربة الخصائص

لتجربة الخصائص التي قمنا بتعيينها، سنقوم بإضافة بعض أدوات التحكم إلى النموذج. يشتمل شكل ٤-٧ على النموذج بعد إضافة أدوات التحكم. ونوضح فيما يلي كيفية إضافة أدوات التحكم إليه ثم تجربته. تابع معنا الخطوات الآتية:

١. قم بإضافة زر أمر إلى النموذج **Form1.vb**. قم بتغيير اسمه إلى

btnShowInfo1 وعنوانه إلى **Show Emp1 Info** (انظر شكل ٤-٧).



شكل ٧-٤ نموذج المشروع بعد إضافة عناصره المختلفة.

٢. انقر الزر نقراً مزدوجاً ثم قم بإدخال الكود التالي داخل إجراء احتواء حدث نقر الزر:

```
Private Sub btnShowInfo1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnShowInfo1.Click
    Dim sTemp As String
    sTemp = "Employee: " & objEmp1.FullName & vbCrLf
    sTemp += "Salary: " & FormatCurrency(objEmp1.Salary) &
vbCrLf
    sTemp += "Departement: " & objEmp1.Department & vbCrLf
    sTemp += "Supervisor: " & objEmp1.Supervisor.FullName &
vbCrLf
    sTemp += "Instance: " & objEmp1.InstanceID
    sTemp += " Of " & objEmp1.ClassInstanceCount
    MessageBox.Show(sTemp, "objEmp1 Info")
End Sub
```

٣. قم بإضافة زر أمر آخر إلى النموذج. قم بتغيير اسمه إلى btnShowInfo2 وعنوانه إلى Show Emp2 Info.
٤. انقر الزر نقراً مزدوجاً ثم قم بإدخال الكود التالي داخل إجراء احتواء حدث نقر الزر:

```
Private Sub btnShowInfo2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnShowInfo2.Click
    Dim sTemp As String
    sTemp = "Employee: " & objEmp2.FullName & vbCrLf
```

```
sTemp += "Salary: " & FormatCurrency(objEmp2.Salary) &
vbCrLf
sTemp += "Departement: " & objEmp2.Department & vbCrLf
sTemp += "Supervisor: " & objEmp2.Supervisor.FullName &
vbCrLf

sTemp += "Instance: " & objEmp2.InstanceID
sTemp += " Of " & objEmp2.ClassInstanceCount
MessageBox.Show(sTemp, "objEmp2 Info")
End Sub
```

٥. قم بإضافة مربع نص إلى النموذج. قم بتغيير اسمه إلى `txtSalPer`.
٦. قم بإضافة أداة عنوان يسار مربع النص السابق. قم بتغيير عنوان الأداة إلى `.Enter Salary Change Percent:`
٧. قم بإضافة زر أمر ثالث يمين مربع النص. قم بتغيير اسمه إلى `btnChangeSalary` وعنوانه إلى `Change Emp1 Salary`.
٨. انقر الزر الجديد نقرًا مزدوجاً ثم قم بإدخال الكود التالي داخل إجراء حدث نقر الزر:

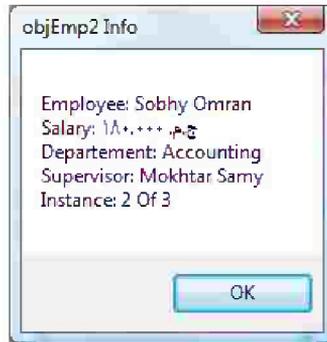
```
Private Sub btnChangeSalary_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnChangeSalary.Click
objEmp1.ChangeSalary(txtSalPer.Text)
End Sub
```

٩. قم بتشغيل التطبيق ولاحظ ما يلي:
- انقر زر `Show Emp1 Info`، يظهر مربع رسالة يحتوي على جميع خصائص الموظف الأول (انظر شكل ٤-٨).



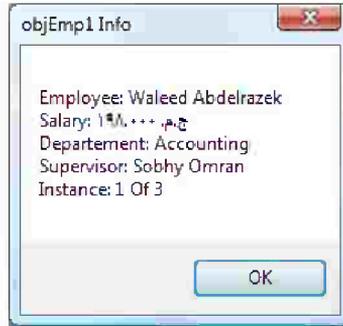
شكل ٤-٨ عرض جميع خصائص الموظف الأول.

- انقر زر **Show Emp2 Info**، يظهر مربع رسالة يحتوى على جميع خصائص الموظف الثاني (انظر شكل ٤-٩).



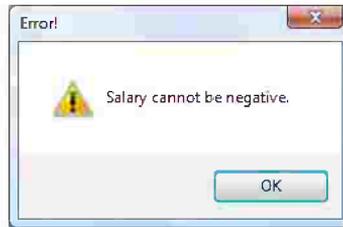
شكل ٤-٩ عرض جميع خصائص الموظف الثاني

- قم بإدخال رقم يعبر عن نسبة زيادة المرتب أو إنقاصه داخل مربع النص ثم انقر زر **Change Emp1 Salary** وأعد انقر زر **Show Emp1 Info** مرةً أخرى، تلاحظ تغيير مرتب الموظف إلى القيمة الجديدة (انظر شكل ٤-١٠).



شكل ٤-١٠ تغيير مرتب الموظف الأول بنسبة ١٠%

- وأخيراً قم بتغيير قيمة الخاصية **Salary** لأحد الموظفين كي تكون سالبة ثم قم بتشغيل التطبيق لتحصل على رسالة الخطأ الموضحة بشكل ٤-١١.



شكل ٤-١١ ظهور رسالة الخطأ لأن المرتب قيمة سالبة.