

الفصل الرابع عشر استخدام التصنيفات Classes

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

بانتهاء هذا الفصل ستتعرف على:

- ◆ مفهوم التصنيفات
- ◆ خطوات إنشاء التصنيف
- ◆ استخدام التصنيف داخل التطبيقات

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 الذي ترغب في إعادة استخدامه داخل قطعة من الكود **Code Component** تسمى **Class** وهو ما اصطلحنا على تسميته تصنيفاً، حيث يحتوى التصنيف على الإجراءات والدوال التي يمكنك استخدامها أكثر من مرة وبالتالي يمكنك تحقيق مفهوم البرمجة الموجهة بالكائنات **Object Oriented Programming (OOP)** المبنية أساساً على استخدام التصنيفات. سنقوم في هذا الفصل بالتعرف على كيفية إنشاء التصنيفات واستخدامها داخل **Visual Basic 2012**، فكن معنا.

مفهوم التصنيفات (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 تدري بالتصنيفات **Classes** والكائنات **Objects**. فعلى سبيل المثال حينما قمت بإدراج مربع نص **Text Box** من مربع الأدوات إلى النموذج، فأنت بذلك قمت بإنشاء حالة **Instance** (أو كائن **Object**) من التصنيف **Text Box** دون أن تدري. وعلى ذلك إذا قمت برسم خمسة مربعات نصوص على النموذج، تكون بذلك قد أنشأت خمس حالات من التصنيف **Text Box**. كما أن النموذج المستخدم نفسه عبارة عن حالة جديدة من التصنيف **Form**.

يطلق على حالات التصنيف "كائنات" **Objects**. فكل تصنيف عبارة عن قالب يتم منه إنشاء عدد من الكائنات متشابهة الوظيفة. فعلى سبيل المثال، التصنيف **Text Box** عبارة عن قالب لا يحتوى على أية بيانات داخل الخصائص، أما الكائنات المنبثقة من هذا التصنيف

مثل مربع النص **txtF Name** مثلاً فيحتوى على بيانات خاصة بكل خاصية من خصائصه، فهو يحتوى مثلاً على القيمة **txtF Name** 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** و **F#**، ويهدف هذا الأسلوب لفصل البرنامج إلى 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 التصنيف

يمكنك إنشاء تصنيفاتك داخل **Visual Basic 2012** بإضافة قالب التصنيف إلى مشروع موجود مسبقاً أو بإنشاء مكتبة تصنيفات **Class Library** تحتوى على تصنيف أو أكثر. تحتوى التصنيفات دائماً على كوكبة من العناصر الثلاثة الآتية:

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

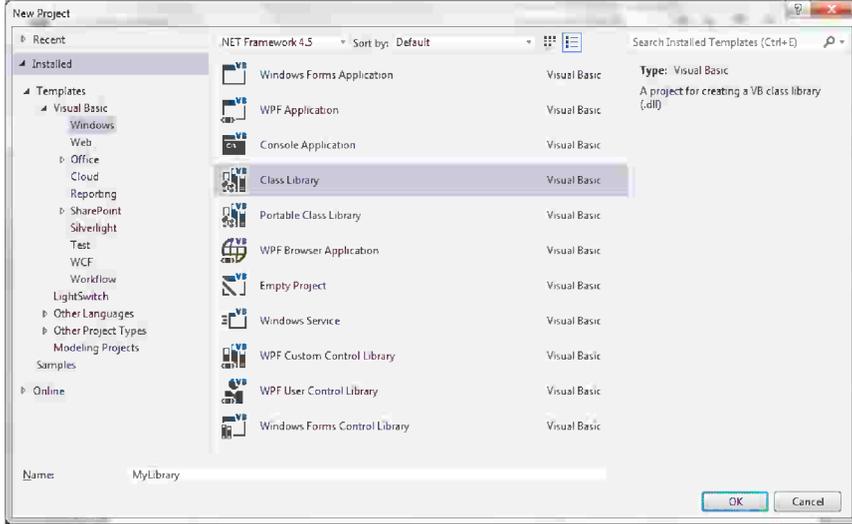
إنشاء التصنيف

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

١. افتح قائمة **File** من شريط القوائم ثم اختر **NewProject** من القائمة المنسدلة، يظهر المربع الحوارى المعتاد **New Project** (انظر شكل ١٤-١).
٢. من العمود الموجود بالجزء الأيمن من المربع الحوارى، اختر **Class Library**.
٣. قم بتعيين اسم مميز لمكتبة التصنيفات الجديدة داخل مربع النص **Name** وليكن

.MyLibrary

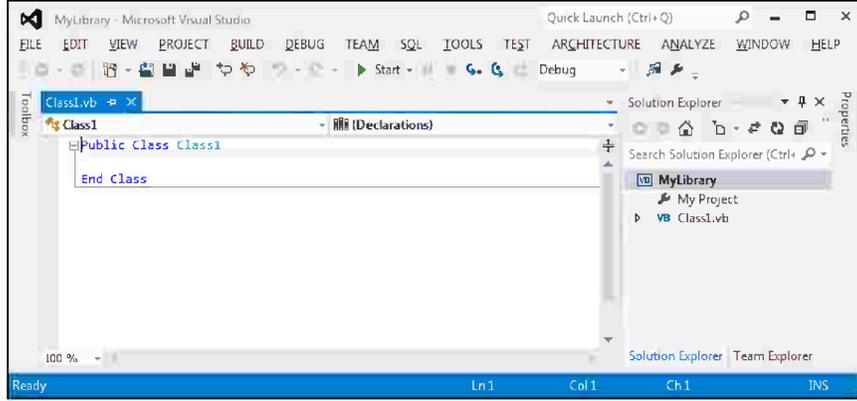
٤ . انقر زر **Ok**، تظهر مكتبة التصنيفات الجديدة داخل مشروع جديد محتويةً على ملف واحد باسم **Class1.vb** وهو الملف الذي سيتم كتابة كود التصنيف بداخله (انظر شكل ١٤-٢).



شكل ١٤-١ إنشاء مكتبة التصنيفات باستخدام المربع الحوارى **New Project**

يمكنك تضمين عدة تصنيفات داخل مكتبة التصنيفات التي قمنا بإنشائها. لأداء ذلك، افتح قائمة **Project** من شريط القوائم ثم اختر **Add Class** من القائمة المنسدلة. إلا أننا للتبسيط سنستخدم تصنيفاً واحداً فقط.





شكل ١٤-٢ مكتبة التصنيفات الجديدة داخل بيئة التطوير

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

إضافة الخصائص إلى التصنيف

بعد إنشاء الوحدة النمطية الخاصة بالتصنيف، 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 تصنيفك بطريقتين، الأولى باستخدام المتغيرات العامة **Public Variables** والثانية باستخدام إجراءات الخصائص **Property Procedures**.

المتغيرات العامة

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

```
Public FirstNameAs String
Public LastNameAs String
Public BirthDateAs String
```

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



وهكذا يمكنك فيما بعد حينما تقوم بإنشاء كائنات (حالات) من التصنيف 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 مدئ معين وإنما يتم قبول أى تاريخ.

إجراءات الخصائص

تعتبر هذه الطريقة أكثر مرونة من الطريقة السابقة وذلك لقدرة 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 فصلٍ 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 بالمرّة. 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 يلي:

Private Shared dSalary As Decimal

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

Public Property Salary() As Decimal

اكتب كلمة **Get** ثم اضغط مفتاح الإدخال، تقوم بيئة التطوير نيابةً عنك بكتابة الهيكل العام لإجراء الخاصية كما يلي (انظر شكل ١٤-٣):

Public Property Salary() As Decimal

Get

End Get

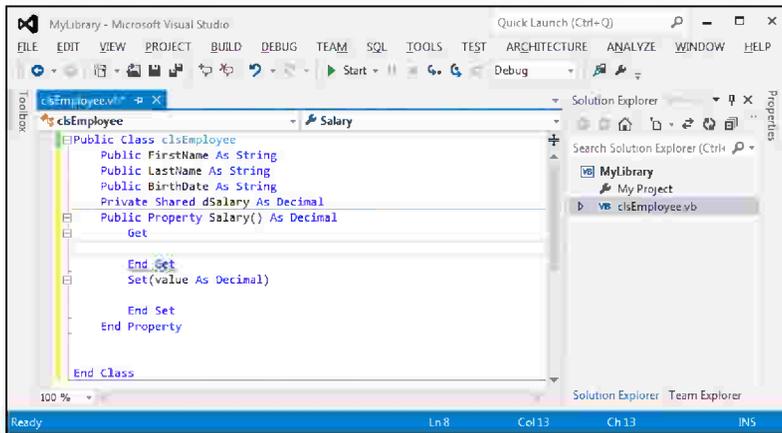
Set(ByVal Value As Decimal)

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 ١٤-٣ يقوم Visual Basic بإنشاء هيكل إجراء الخاصية نيابةً عنك بمجرد ضغط مفتاح الإدخال.

خاصية الاسم الكامل *FullName*

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

```
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.

Supervisor الخاصية المسئول

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 التصنيف 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 التصنيف `clsEmployee` التي لا تحتوي 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 التصنيف

كما تعلم فإن الكائنات تقوم بتنفيذ الأحداث من خلال الوظائف `Methods` أو الدوال مثل الوظيفة `Add()` المصاحبة لمربع السرد أو الوظيفة `Focus()` المصاحبة لزر الأمر وهكذا. يمكنك بنفس الطريقة إنشاء الوظائف داخل تصنيفك من خلال إضافة الإجراءات والدوال العامة إلى التصنيف. لتوضيح ذلك في تصنيفنا الحالي، 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 التالي إلى نافذة كود التصنيف

`:clsEmployee`

```
Public Function ChangeSalary(salPer As Decimal)
```

```
If (salPer > 1) Or (salPer < -1) Then
```

```
salPer = salPer / 100
```

```

End If
Salary = Salary * (1 + salPer)
End Function

```

إضافة حدث إلى التصنيف

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

دعنا الآن نقوم بإضافة الحدث **DataError** الذي ينتج عن إدخال المستخدم لقيمة سالبة بالمتغير **Salary**. قم بالإعلان عن الحدث في جزء الإعلانات العلوى بنافذة كود التصنيف **clsEmployee** كما يلي:

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

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

```
Public Property Salary() As Decimal
```

```
Get
```

```
Return dSalary
```

```
End Get
```

```
Set(ByVal Value As Decimal)
```

```
If Value >= 0 Then
```

```
dSalary = Value
```

```
Else
```

```
RaiseEventDataError("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:

```
RaiseEventDataEvent("Invalid Birth Date")
```



إجراء إنشاء التصنيف

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

```
Public Shared ClassInstanceCountAs Long = 0
Private Shared NextInstanceIDAs Integer = 1
Public ReadOnlyInstanceIDAs Integer
```

حيث:

- سنقوم باستخدام المتغير **ClassInstanceCount** لتخزين عدد حالات (كائنات) التصنيف **clsEmployee**. وهو كمتغير مشترك **Shared Variable** يكون موجود فقط داخل مستوى التصنيف 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 التصنيف **clsEmployee**.

يعتبر الإجراء **Sub New** إجراء الإنشاء الافتراضي للتصنيف والذى يتم تنفيذه كلما تم إنشاء حالة جديدة من حالات التصنيف. لإنشاء هذا الإجراء، اختر **New** من مربع الأحداث بنافذة كود التصنيف ثم قم بإضافة الكود التالى للإجراء:

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

يمكنك بالطبع إضافة أى كود آخر إلى الإجراء.

إجراء هدم التصنيف

يمكنك بنفس الطريقة استخدام الإجراء **Finalize** الذى يتم تنفيذه تلقائياً بمجرد حذف أحد الكائنات التى تنتمى للتصنيف أو هدمه. لإنشاء هذا الإجراء، اختر **Finalize** من مربع الأحداث بنافذة كود التصنيف ثم قم بإضافة الكود التالى 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 للتصنيف كما يلى:

```
Public Class clsEmployee  
    Public FirstNameAs String  
    Public LastNameAs String  
    Public BirthDateAs String  
    Private Shared dSalary As Decimal  
    Private Shared nDeptNum As Integer  
    Private objSupervisor As clsEmployee  
    Public Shared ClassInstanceCountAs Long = 0  
    Private Shared NextInstanceIDAs Integer = 1  
    Public ReadOnlyInstanceIDAs Integer  
    Public Event DataError(ByValsErrorMsg As String)  
    Public Property Salary() As Decimal  
        Get  
            Return dSalary  
        End Get  
        Set(ByVal Value As Decimal)  
            If Value >= 0 Then  
                dSalary = Value  
            Else  
                RaiseEventDataError("Salary can't 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(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 التصنيف داخل التطبيقات

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

إضافة المشروع الجديد

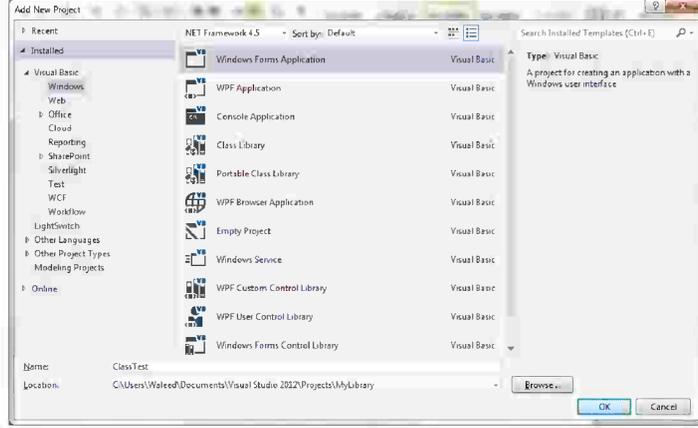
حينما قمنا بإنشاء مكتبة التصنيفات السابقة باسم **MyLibrary**، تم إنشاء مشروع بنفس الاسم داخل حل **Solution** بنفس الاسم أيضاً. وبدلاً من أن نقوم بإنشاء حل جديد، سنقوم بإضافة مشروع جديد إلى نفس الحل. لأداء ذلك، تابع معنا الخطوات الآتية:

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 ١٤-٤).



شكل ١٤-٤ المربع الحوارى Add New Project.

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

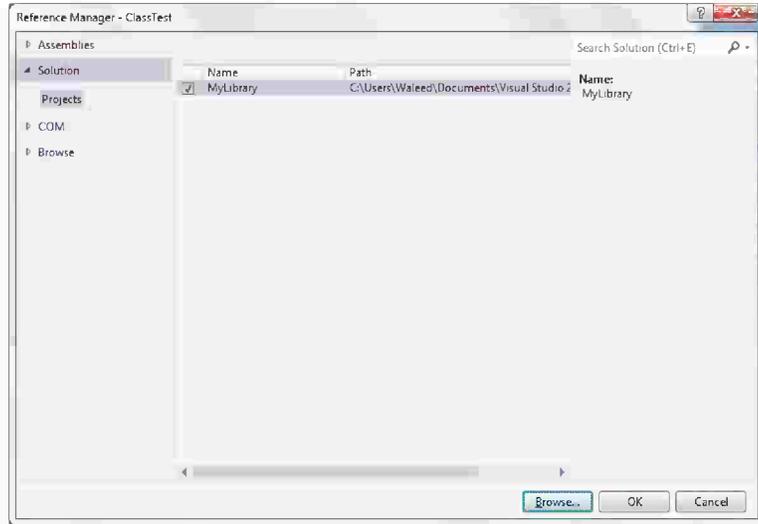
تعيين مشروع البدء

أصبح الآن لدينا مشروعان داخل الحل، أحدهما المشروع **MyLibrary** الذى يحتوى على التصنيف والآخر هو المشروع **ClassTest** الذى سنقوم باستخدام التصنيف 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 التصنيف 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 الموضعية وبالتالي يكون المشروع **ClassTest** هو مشروع البدء ودلالة ذلك ظهوره بخط ثقيل وظهور المشروع الآخر بخط خفيف.

إضافة مرجع التصنيف

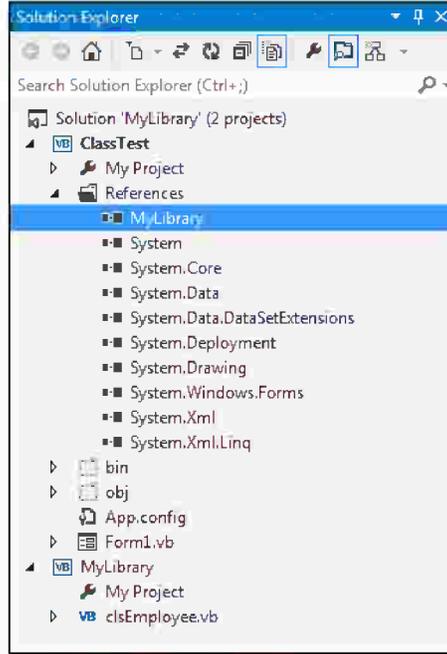
على الرغم من أن المشروع الجديد **ClassTest** جزء من الحل **MyLibrary** الذى يحتوى على مشروع التصنيف، إلا أنك مطالب بإضافة مرجع إلى مشروع التصنيف داخل المشروع الجديد. لإضافة هذا المرجع، تابع معنا الخطوات الآتية:

١. من نافذة المستكشف، انقر المجلد **References** الموجود أسفل المشروع **ClassTest** بزر الفأرة الأيمن ثم اختر **Add Reference** من القائمة الموضعية (انقر الزر  بشريط أدوات نافذة الحل لإظهار المجلد **References** إذا لم يكن ظاهراً بالفعل)، يظهر المربع الحوارى **Reference Manager** (انظر شكل ١٤ - ٥).



شكل ١٤-٥ المربع الحوارى **Reference Manager**.

٢. نشط التبويب **Solution** ثم **Projects** بالجانب الأيسر من المربع الحوارى إذا لم يكن هو التبويب النشط.
٣. انقر المشروع **MyLibrary** لاختياره ثم انقر زر **Ok** لإغلاق المربع الحوارى ولاحظ إضافة المشروع **MyLibrary** إلى المجلد **References** بالمشروع **ClassTest** (انظر شكل ١٤-٦).



شكل ١٤-٦ ظهور المشروع MyLibrary بقائمة مراجع المشروع ClassTest.

الإعلان عن الكائنات

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

```
Dim objEmp As New clsEmployee
```

حيث تنسب كلمة **New** في 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 في الكود التالى:

```
Dim objEmpAsclsEmployee
```

```
.....
```

فيما بعد ،

```
Set objEmp = New clsEmployee
```

حذف الكائنات

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

```
objTemp = Nothing
```

استخدام التصنيف داخل التطبيق

الخطوة الأولى التى تحتاجها لاستخدام التصنيف **clsEmployee** داخل المشروع **ClassTest** تتمثل في تعريف حالات جديدة تنتمى إلى التصنيف. 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 التالى 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** كى يتمكن الكائن من الاستفادة من الأحداث المخصصة المعرفة داخل التصنيف (الحدث **DataError** فى هذه الحالة).

كتابة الأحداث

لأننا استخدمنا كلمة **WithEvents** أثناء الإعلان عن الكائنات، فمن الطبيعى أن يكون هناك حدث مخصص داخل التصنيف وهو الحدث **DataError** الذى يظهر بمجرد إدخال المستخدم قيمة سالبة للخاصية **Salary**. يمكنك الآن تعيين كود الحدث **DataError** الخاص بكل كائن من كائنات التصنيف التى 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(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(sErrorMsg As String) Handles  
objEmp2.DataError  
MessageBox.Show(sErrorMsg, "Error!",  
MessageBoxButtons.OK, MessageBoxIcon.Warning)  
End Sub
```

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

تعيين خصائص كائنات التصنيف

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

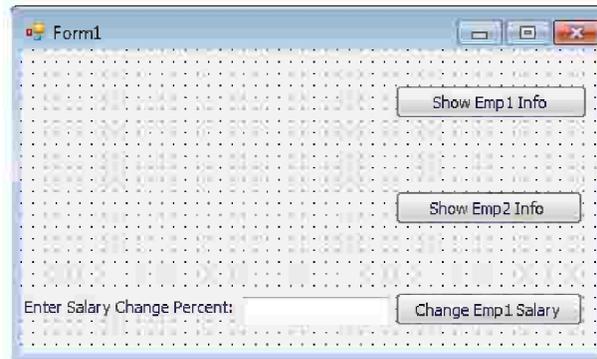
لتجربة الخصائص التي قمنا بتعيينها، سنقوم بإضافة بعض أدوات التحكم إلى النموذج.

يشتمل شكل ٧-١٤ 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 ٧-١٤).



شكل ٧-١٤ نموذج المشروع بعد إضافة عناصره المختلفة.

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

الزر:

```
Private Sub btnShowInfo1_Click(sender As Object, e As  
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 نقرًا مزدوجاً ثم قم بإدخال الكود التالي داخل إجراء احتواء حدث نقر الزر:

```
Private Sub btnShowInfo2_Click(sender As Object, e As
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 نقرأ مزدوجاً ثم قم بإدخال الكود التالي داخل إجراء حدث انقر الزر:

```
Private Sub btnChangeSalary_Click(sender As Object, e As 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 ٨-١٤).



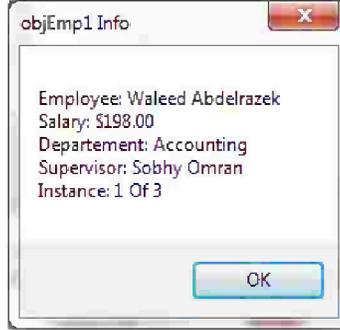
شكل ٨-١٤ عرض جميع خصائص الموظف الأول.

- انقر زر **Show Emp2 Info**، يظهر مربع رسالة يحتوي على جميع خصائص الموظف الثاني (انظر شكل ٩-١٤).



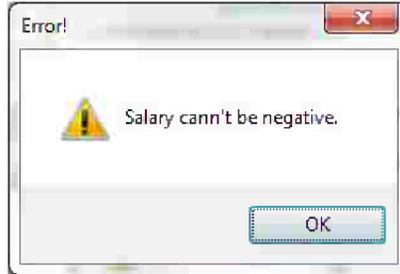
شكل ٩-١٤ عرض جميع خصائص الموظف الثاني

- قم بإدخال رقم يعبر عن نسبة زيادة المرتب أو إنقصاه 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 ١٤-١٠).



شكل ١٤-١٠ 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 ١٤-١١.



شكل ١٤-١١ ظهور رسالة الخطأ لأن المرتب قيمة سالبة.

