الباب الثاني

تطوير أصناف مجال المشكلة

DEVELOPING PROBLEM DOMAIN CLASSES

- الفصل السادس: تطوير أصناف مجال المشكلة
- الفصل السابع: إضافة مهام جديدة لأصناف مجال المشكلة
- الفصل الثامن: فهم التوارث والواجهات
- الفصل التاسع: تطوير علاقات المرافقة بين الكائنات

الفصل السادس

تطوير أصناف مجال المشكلة

WRITING A PROBLEM DOMAIN CLASS DEFINITION

أهداف الفصل:

- تعلم كيفية تسمية مصطلحات لغة VB .NET.
- تطوير أصناف مجال المشكلة (Problem Domain Classes).
- تعريف الصفات (Attributes).
- كتابة الإجراءات والخصائص (Methods, Properties).
- تجربة صنف مجال المشكلة.
- إنشاء كائنات من الأصناف.
- كتابة الإجراء TellAboutSelf.
- كتابة صنف اختبار لواجهة المستخدم (Form).

لقد تعلمنا في الباب الأول أن تطوير الأنظمة المعتمدة على الكائنات توظف تقنية Three-Tier والتي تتكون من ثلاثة طبقات من الأصناف وهي: أصناف واجهة المستخدم (GUI Classes) والتي تمثل واجهة النظام لاستقبال بيانات وإرسال نتائج، وأصناف مجال المشكلة (PD Classes) والتي تمثل الكائنات الضرورية المتعلقة بأعمال الشركة الطالبة للنظام، وكائنات التعامل مع البيانات (DA Classes) والتي تقدم خدمات تخزين البيانات واسترجاعها. سوف نشرح هنا في الباب الثاني كيفية تطوير أصناف مجال المشكلة مستخدماً لغة VB .NET، أما الباب الثالث فسوف يركز على العمل مع أصناف واجهة المستخدم، وفي الباب الرابع من الكتاب سوف نتعلم تطوير أصناف التعامل مع البيانات. أما الباب الخامس والأخير فيركز على كيفية تطوير النظام كاملاً وذلك بربط الأنواع الثلاثة من الأصناف.

وسوف تركز في هذا الفصل على تطوير الصنف Customer وذلك بتعريف الصفات (Attributes) الخاصة به والإجراءات (Methods) المسؤولة عن تخزين قيم هذه الصفات واسترجاعها. ولاختبار الصنف Customer سوف نكتب صنفاً آخر اسمه TesterOne والذي سيحتوي على أوامر تعريف كائنات من النوع Customer واستدعاء الإجراءات من هذه الكائنات للتعامل مع قيم صفات هذه الكائنات. وسوف نكتب أيضاً صنفاً آخر اسمه TesterTwo والذي سيحتوي على نفس الأوامر الموجودة في الصنف TesterOne ولكنه سوف يستخدم واجهة رسومية (GUI) بدلاً من استخدام ملف برمجي (Module).

كيفية تسمية مصطلحات لغة VB .NET

VB .NET Naming Conventions

لقد تعرفنا خلال كل من الفصل الثالث والرابع كيف يسمي المبرمجون معرفات (Identifiers) لغة VB .NET وهي الأسماء التي تستند للأصناف وتعريفات الأصناف والصفات والإجراءات والتي يمكن تلخيصها كما يلي:

- يبدأ اسم الصفة بحرف صغير (Small)، وإذا تكوّن اسم الصفة من أكثر من كلمة يجب أن تبدأ الكلمات التالية للكلمة الأولى بحرف كبير (Capital) مثل الصفة address، والصفة phoneNo.
- يبدأ اسم الإجراء بحرف كبير والكلمات التالية للكلمة الأولى تبدأ أيضاً بحرف كبير، ولابد أن تكون أول كلمة فعلاً والكلمة التالية تكون اسماً مثل الإجراء GetPhoneNo، والإجراء SetAddress، والإجراء ComputeLease.

تطوير أصناف مجال المشكلة

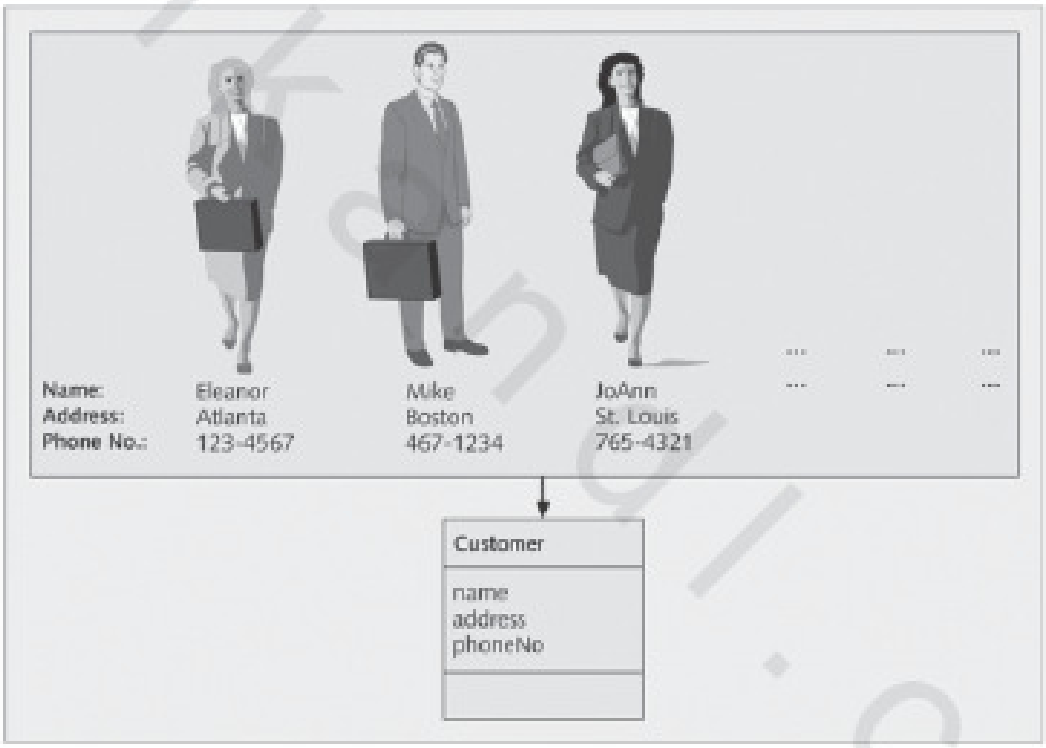
Developing a PD Class Definition

تذكر أن نظام برادشو يحتوي على العديد من الأصناف وهي: الصنف Customer، والصنف Boat، والصنف Slip، والصنف Dock، وتوجد أيضاً العديد من التفاعلات والعلاقات بينها حيث العميل يمتلك المركب، والمركب يرسى على مرسى، والعميل يؤجر المرسى. وتوجد أيضاً علاقة توارث بين بعض هذه الأصناف، فعلى سبيل المثال الصنف Sailboat والصنف Powerboat أصناف فرعية (Subclasses) للصنف Boat، أو بمعنى آخر Sailboat هو Boat، و Powerboat هو Boat.

نبدأ بتطوير نظام شركة برادشو مارينا بكتابة تعريف أصناف مجال المشكلة كلٌ على حده حيث يتطلب ذلك كتابة صفات الصنف التي تمثل خصائص كائنات الصنف والإجراءات الخاصة بالصنف والتي تمثل سلوك كائنات الصنف.

إن برنامج VB .NET الذي يمثل صنفاً معيناً يطلق عليه تعريف الصنف (Class Definition). إن أول ما سيتم تطويره في نظام برادشو هو الصنف Customer الذي يمثل عملاء شركة برادشو والموضح في الشكل رقم (٦.١). وسوف تكتب العديد من أصناف مجال المشكلة الإضافية خلال الفصول التالية.

إن الصنف Customer يحتوي على مجموعة من الصفات التي تمثل بيانات العملاء والتي تم استخلاصها خلال الفصل الخامس. الصفة name (اسم العميل)، والصفة address (عنوان العميل)، والصفة phoneNo (هاتف العميل). وسوف يتم توصيف هذه الصفات داخل تعريف الصنف، ثم تكتب البرنامج المطلوب لإنشاء كائن من الصنف Customer وإسناد قيم لصفات هذا الكائن، ثم استرجاع هذه القيم.

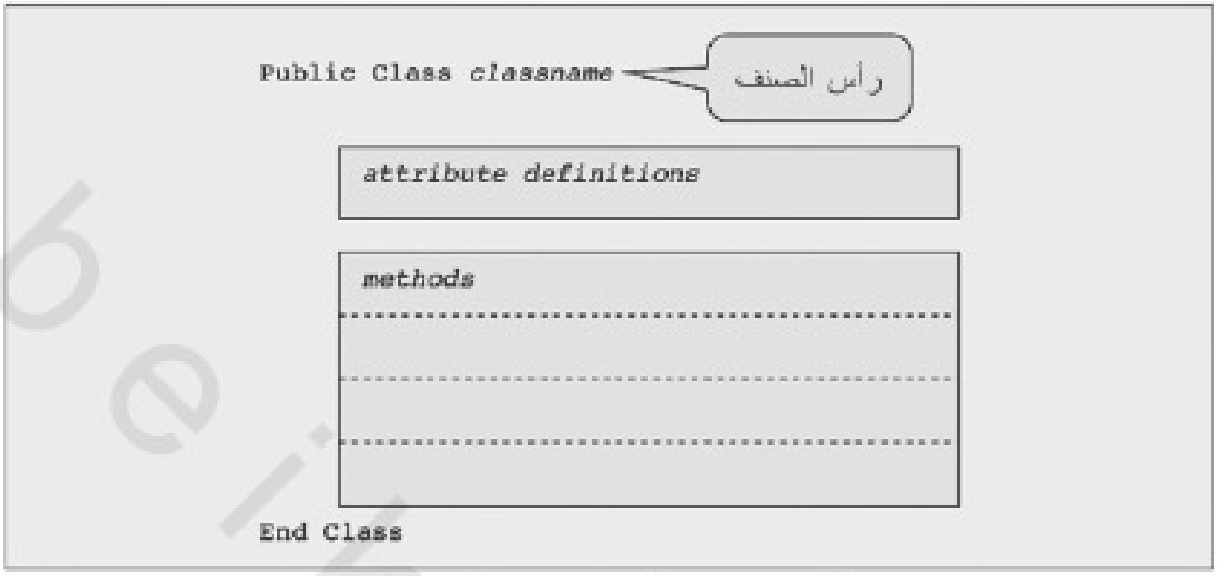


الشكل رقم (٦.١). الصنف Customer يمثل العملاء.

هيكل تعريف الصنف

Class Definition Structure

في لغة VB .NET يحتوي تعريف الصنف على رأس الصنف (Class Header) متبوعاً بتعريف صفات الصنف ثم تعريف إجراءات الصنف. ويتكون رأس الصنف من سطر واحد والذي يعرف الصنف وبعض خصائصه. وينتهي تعريف الصنف بالأمر end class. يوضح الشكل رقم (٦.٢) هيكل تعريف الأصناف بلغة VB .NET.



الشكل رقم (٦,٢). هيكل تعريف الصنف في لغة VB .NET.

يكتب رأس الصنف Customer هكذا:

```
Public Class Customer
```

يوضح محرر نصوص لغة VB .NET الكلمات المحجوزة باللون الأزرق ولكنها تظهر في هذا الكتاب بخط Courier أسود عريض. تشير الكلمة المحجوزة Public أن هذا الصنف له Public Accessibility أي نستطيع استخدام هذا الصنف من أي مكان في البرنامج. وتشير الكلمة المحجوزة Class أن هذا السطر هو رأس الصنف والكلمة Customer تشير إلى اسم الصنف.

تعريف الصفات
Defining Attributes

يحتوي الصنف Customer على ثلاث صفات وهي: الصفة name، والصفة address، والصفة phoneNo والذي يمكن توصيفها كما نوصف المتغيرات، وإذا استرجعنا الفصل الثالث نجد أن المتغير يتم تعريفه مستخدماً الكلمة المحجوزة Dim ثم اسم المتغير ثم الكلمة المحجوزة As ثم نوع البيانات المناسب. ويمكن توصيف صفات الأصناف بنفس الأسلوب عدا استبدال الكلمة المحجوزة Dim بالكلمة المحجوزة Private، ولتعريف صفات الصنف Customer، يستخدم نوع البيانات String كما يلي:

```
' attributes
Private name As String
Private address As String
Private phoneNo As String
```

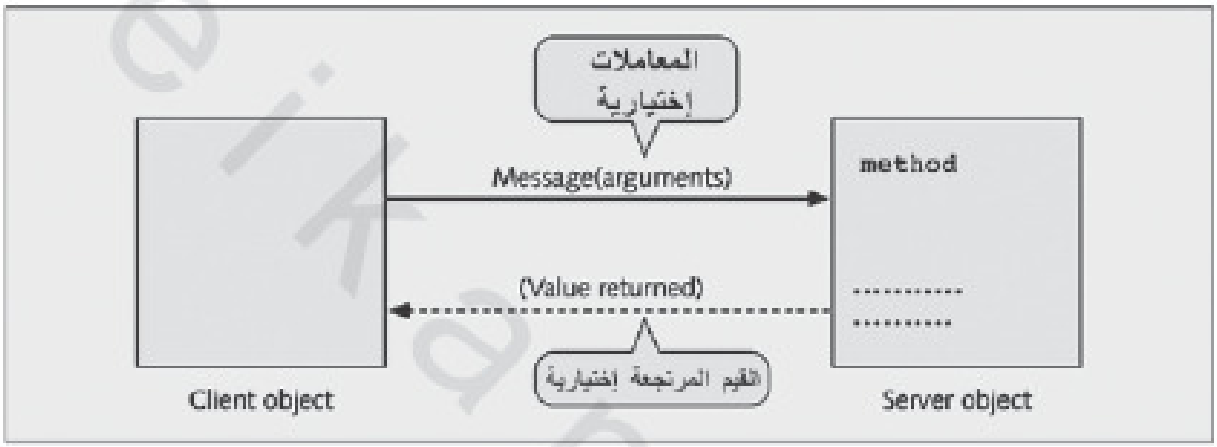
عند تعريف صفات الأصناف لابد من تعريف مدى الوصول إلى المتغير (Accessibility) حيث يوجد أربع مستويات للوصول وهي: Public (عام)، و Private (خاص)، و Protected (محمي)، و Friend (صديق). إن استخدام الكلمة المحجوزة Public يعني أن أي صنف يستطيع الوصول إلى الصفة مباشرة، بينما الكلمة المحجوزة Private يمنع الوصول المباشر ولكن يمكن الوصول للصفة داخل الصنف فقط الذي تم تعريف هذه الصفة فيه، أما الكلمة Protected فتسمح للأصناف الفرعية أن تصل للصفة المعرفة في الصنف الأب مباشرة، والكلمة Friend تسمح للأصناف المعرفة داخل assembly مشترك أن تصل لهذه الصفة. يتكون assembly من مشروع أو أكثر تم إصدارها في تطبيق واحد، مع العلم أنك تستطيع أن تسند المشروع الخاص بك وجميع أصنافه إلى assembly محدد.

يتم اختيار مستوى الوصول المناسب بناء على نوع المتغير الذي تريد توصيفه، فعلى سبيل المثال إن الكلمة Public عادة تستخدم لتعريف متغيرات ثابتة (Constants) والتي تحتاج قيمتها بواسطة أصناف أخرى. لقد قرأنا في الفصل الأول على مفهوم الكبسلة (Encapsulation) والذي يعني إخفاء التركيب الداخلي لصفات الصنف عن الكائنات الأخرى، ويمكن تطبيق هذا المفهوم عند تعريف صفات صنف ما بواسطة الكلمة المحجوزة Private لكي نقيد الوصول إلى هذه الصفة، وهذا يؤدي إلى الكبسلة وإخفاء المعلومات عن الآخرين (Information Hiding) ويمنع الوصول إليها مباشرة. وعلى أي حال لابد من تعريف إجراءات الوصول (Accessor Methods) والتي يمكن استدعاؤها بواسطة الآخرين لتخزين قيم الصفات المعرفة من النوع Private واسترجاعها.

تطوير الإجراءات والخصائص
Writing Methods and Properties

إن كائنات مجال المشكلة لا تعمل بمفردها ولكن تتفاعل مع الكائنات الأخرى في النظام بتقديم العديد من الخدمات (بواسطة تنفيذ الإجراءات المعرفة لديها). ولقد رأيت سابقاً أن الكائنات تتفاعل مع بعضها بعضاً لإنجاز مهام النظام مثل تفاعل الكائنات في الواقع وذلك عن طريق إرسال رسائل لتنفيذ المهام. ويمكن تمثيل علاقة الكائن المرسل للرسالة (طالب الخدمة) والكائن المستقبل للرسالة (منفذ الخدمة) بعلاقة العميل والخادم حيث يصبح الكائن الطالب للخدمة هو العميل (Client Object) والكائن المنفذ للخدمة هو الخادم (Server Object). يوضح الشكل رقم (٦.٣) العلاقة بين كل من كائن العميل وكائن الخادم في نموذج العميل–الخادم.

يرسل العميل رسالة إلى الخادم طالباً منه تنفيذ إجراء لديه، وربما يرسل قيماً مع الرسالة أولاً في شكل معاملات (Arguments)، عندئذ ينفذ الخادم الإجراء المناظر للرسالة لإنجاز المهمة المطلوبة وربما يعيد قيمة للعميل. فعلى سبيل المثال، لقد كتبنا أوامر في الفصل الرابع تستدعي إجراءات معرفة في أصناف مكتبة لغة VB .NET مثل الصنف Math والصنف String، وبتطبيق نموذج العميل–الخادم نجد أن هذه الأوامر تمثل جانب العميل مستدعياً إجراءات من الخادم.



الشكل رقم (٦,٣). التفاعل بين الصنف العميل والصنف الخادم.

يتم تعريف الإجراءات (Methods) مستخدماً نوعين من الإجراءات وهما: إجراء فرعي (Sub Procedure)، وإجراء دالة (Function Procedure)، والفرق الوحيد بينهما هو أن الإجراء الفرعي لا يعيد قيمة أما إجراء الدالة فيعيد قيمة.

يبدأ تعريف الإجراء الفرعي برأس الإجراء (Procedure Header) متبوعاً بأوامر الإجراء كما هو واضح في الشكل رقم (٦,٤).

```
accessibility Sub procedurename(parameter list)

    method statements

End Sub
```

الشكل رقم (٦,٤). هيكل الإجراء Sub في لغة VB .NET.

يحتوي رأس الإجراء الموضح في الشكل رقم (٦.٤) على أربعة أجزاء:

١- مستوى الوصول للإجراء (Accessibility) ويأخذ واحدة من أربع قيم وهي: Public أو Private أو Protected أو Friend والتي تم الحديث عنها عند تعريف الصفات. وبشكل عام إن مستوى الوصول للإجراءات يكون Public (عام) لأنك تريد الكائنات الأخرى أن تستدعي هذا الإجراء.

٢- الكلمة Sub والتي تشير إلى أن هذا الإجراء من النوع الإجراء الفرعي.

٣- اسم الإجراء (procedurename)، ولقد تعلمنا أن اسم الإجراء يبدأ بحرف كبير متبوعاً بكلمات تبدأ بحرف كبير ودائماً ما يبدأ اسم الإجراء بفعل يشير إلى ما سيفعله الإجراء. فعلى سبيل المثال الاسم RecordPayment و ComputeServiceCharge.

٤- قائمة المعاملات (Parameter List) والتي تشير إلى قائمة المعاملات التي تمرر إلى المتغيرات المعرفة داخل أقواس إجراء الخادم التي ستستقبل قيم المعاملات التي تمرر للإجراء، ولابد أن تتوافق أنواع البيانات الخاصة بقائمة المعاملات مع متغيرات الإجراء حيث لا يجوز أن تمرر قيمة من النوع Integer إلى متغير من النوع String. وإذا لم تمرر معاملات للإجراء نترك القائمة فارغة ولا نضع شيئاً بين الأقواس.

ويتشابه تعريف إجراء الدالة مع الإجراء الفرعي حيث يبدأ التعريف برأس الدالة متبوعاً بأوامر الدالة كما هو واضح في الشكل رقم (٦.٥).

```
accessibility Function procedurename(parameter list) As datatype

    method statements

End Function
```

الشكل رقم (٦.٥). هيكل إجراء الدالة في لغة VB .NET.

يحتوي رأس الدالة على خمسة أجزاء:

١- مستوى الوصول (Accessibility) لإجراء الدالة والتي تأخذ كلمة من أربع: Public أو Private أو Protected أو Friend والتي تم وصفها سابقاً.

٢- الدالة (Function) والتي تشير إلى أن هذا هو إجراء الدالة.

٣- اسم الإجراء (procedurename) والتي تشير إلى اسم الإجراء.

٤- قائمة المعاملات (Parameter List) وهي قائمة بتعريف المتغيرات والتي ستستقبل قيم المعاملات (Arguments) المستقبلية عند استدعاء الإجراء.

٥- As datatype الذي يوضح نوع القيمة التي ستعود من الإجراء بعد تنفيذها حيث يجب أن يكتب نوع البيانات (Integer أو Double أو String إلخ).

إن الصنف Customer يحتوي على ثلاث صفات، كل واحد منها عرف بمستوى مرور خاص (Private) والتي لا تمكّن الكائنات الأخرى من الوصول إليها مباشرة، ولذلك لابد من تعريف إجراءات مرور (Accessor Methods) تمكّن الكائنات الأخرى من إسناد (أو استرجاع) قيم هذه الصفات.

ويطلق على إجراءات الوصول اسم الإجراءات القياسية (Standard Methods) والتي لا تظهر في نموذج Class Diagram لأن المطورين يفترضون وجودهم بشكل افتراضي. وعلى النظير فإن الإجراءات التي تكتب لإنجاز مهام أخرى يطلق عليها اسم الإجراءات الخاصة (Custom Methods) والتي يجب أن تظهر في نموذج Class Diagram. ويوضح لك الفصل التالي كيفية برمجة الإجراءات الخاصة.

يوجد نوعان من إجراءات الوصول وهما: إجراءات استرجاع قيمة صفة (get attribute value) وإجراءات إسناد قيمة الصفة (set attribute value). ويطلق على النوع الأول اسم إجراءات الوصول للاسترجاع (Get Accessor Methods) أو ببساطة اسم Getters ولابد أن تبدأ بكلمة Get متبوعة باسم الصفة، ولذلك سوف تكتب ثلاثة إجراءات من هذا النوع وهي: GetName و GetAddress و GetPhoneNo للصنف Customer. وبالمثل، الإجراءات التي تستخدم لتغيير قيمة صفة يطلق عليها إجراءات الوصول للإسناد أو ببساطة Setters. ويجب أن يبدأ اسم هذه الإجراءات بكلمة Set متبوعةً باسم الصفة. وسوف تكتب أيضاً ثلاثة إجراءات من هذا النوع لصفات الصنف Customer وهي: SetName و SetAddress و SetPhoneNo، وسوف يستخدم نوع الإجراء الفرعي لتعريف الإجراءات Setters لأنها لا تعيد قيمة، أما النوع إجراء الدالة فسيستخدم لتعريف الإجراءات من النوع Getters لأنها تعيد قيمة. الصيغة العامة لتعريف إجراء من النوع Getters:

```
// getter format
Public Function GetAttributeName() As attributeDataType
      Return attributeName
End Function
```

بينما الصيغة العامة لتعريف إجراء من النوع Setters:

```
// setter format
Public Sub SetAttributeName(ByVal attributeDataType parameterName)
      attributeName = parameterName
End Sub
```

لاحظ أن رأس الإجراء Getter يحتوي على نوع البيانات الخاصة بالصفة ويحتوي على قائمة معاملات فارغة، ويحتوي أيضاً على أمر واحد والذي يتضمن الكلمة المحجوزة Return لكي تعيد قيمة الصفة إلى الكائن الذي يستدعي هذا الإجراء. ويستطيع هذا الإجراء إرجاع قيمة واحدة فقط حتى ولو كان متغير إشارة يشير إلى كائن يحتوي على العديد من المتغيرات. ويوضح الشكل رقم (٦.٦) ثلاثة إجراءات من النوع Getters لصفات الصنف Customer.

```
'get accessor methods
Public Function GetName() As String
   Return name
End Function

Public Function GetAddress() As String
   Return address
End Function

Public Function GetPhoneNo() As String
   Return phoneNo
End Function
```

الشكل رقم (٦.٦). إجراءات الوصول Getters للصنف Customer.

يستدعي الكائن (العميل) الإجراء Setter لإسناد قيمة المعامل المرسل إلى صفة ما في كائن. وأحياناً يحتوي هذا الإجراء على أوامر للتحقق من صحة المعاملات المرسلة قبل إسنادها للصفات والتي سيتم عرضها بالتفصيل في الفصل السابع. تحتوي الإجراءات Setters في هذا الفصل على أمر واحد فقط والذي يسند القيمة المستقبلة للصفة حيث يسرد الشكل رقم (٦.٧) الإجراءات الثلاث المسؤولة عن إسناد قيم للصفات الثلاثة الخاصة بالصنف Customer.

```
'set accessor methods
Public Sub SetName(ByVal aName As String)
   name = aName
End Sub

Public Sub SetAddress(ByVal anAddress As String)
   address = anAddress
End Sub

Public Sub SetPhoneNo(ByVal aPhoneNo As String)
   phoneNo = aPhoneNo
End Sub
```

الشكل رقم (٦.٧). إجراءات الوصول Setters للصنف Customer.

كما يمكنك استبدال كل من الإجراء Getter والإجراء Setter بتطوير خاصية (Property) لكل صفة. والخاصية تشبه الإجراء وتحتوي على كل من الشق Get والشق Set ولذلك تبدو للكائن (العميل) مثل الصفة (أي متغير). يبدأ تعريف الخاصية برأس الخاصية والذي يشير إلى أن هذا تعريف خاصية وتنتهي بالكلمة End Property، ولأن تعريف الخاصية يحتوي على كل من الشق Get والشق Set، فإن رأس الخاصية يحتوي على نوع البيانات الخاص بقيمة الصفة العائدة كما هو واضح في الشكل رقم (٦,٨). ويعيد الجزء Get قيمة الصفة عن طريق الأمر Return متبوعاً باسم الصفة، بينما يسند الجزء Set المعامل المستقبل للصفة وذلك عن طريق إسناد المعامل لاسم الصفة.

```
' property named CustomerName
   Public Property CustomerName() As String
      Get
         Return name
      End Get
      Set(ByVal aName As String)
         name = aName
      End Set
   End Property
```

الشكل رقم (٦,٨). الخاصية CustomerName.

عزيزي القارئ لقد أنهيت الآن تعريف الصنف Customer (الموضح في الشكل رقم ٦,٩) وذلك بتعريف ثلاثة صفات، ثم تعريف ثلاثة إجراءات من النوع Getters لاسترجاع قيم هذه الصفات، وتعريف ثلاثة إجراءات من النوع Setters لإسناد قيم هذه الصفات، وأيضاً تم تعريف خاصية للصفة name.

تجربة صنف مجال المشكلة

Testing a PD Class

لمحاكاة تفاعل الكائنات لإنجاز مهام النظام بإرسال الرسائل، يمكن تعريف صنف اختبار اسمه TesterOne لكي يستدعي الإجراءات المعرفة داخل الصنف Customer. ويوضح الشكل رقم (٦,١٠) أوامر الصنف TesterOne حيث يتشابه هذا الصنف مع الأصناف التي تم تطويرها في كل من الفصل الثالث والفصل الرابع.

```
' Chapter 6 Customer class Example 1
Public Class Customer

     'attributes
     Private name As String
     Private address As String
     Private phoneNo As String

     'get accessor methods
     Public Function GetName() As String
          Return name
     End Function
     Public Function GetAddress() As String
          Return address
     End Function
     Public Function GetPhoneNo() As String
          Return phoneNo
     End Function

     'set accessor methods
     Public Sub SetName(ByVal aName As String)
          name = aName
     End Sub
     Public Sub SetAddress(ByVal anAddress As String)
          address = anAddress
     End Sub
     Public Sub SetPhoneNo(ByVal aPhoneNo As String)
          phoneNo = aPhoneNo
     End Sub

     ' property named CustomerName
     Public Property CustomerName() As String
        Get
          Return name
        End Get
        Set(ByVal aName As String)
          name = aName
        End Set
     End Property

End Class
```

الشكل رقم (٦,٩). شفرة برنامج تعريف الصنف Customer.

```
Chapter 6 TesterOne class Example 1

Module TesterOne

   Sub Main()
      Dim firstCustomer As Customer = New Customer() ' create instance

      ' use property to populate name
      firstCustomer.CustomerName = "Eleanor"

      ' invoke set accessors to populate attributes
      firstCustomer.SetName("Eleanor")
      firstCustomer.SetAddress("Atlanta")
      firstCustomer.SetPhoneNo("123-4567")

      ' define variables to contain attribute values retrieved
      Dim customerName, customerAddress, customerPhoneNo As String

      ' use property to retrieve name
      customerName = firstCustomer.CustomerName

      ' invoke get accessors to retrieve attributes
      customerName = firstCustomer.GetName()
      customerAddress = firstCustomer.GetAddress()
      customerPhoneNo = firstCustomer.GetPhoneNo()

      ' display the retrieved attribute values
      Console.WriteLine("The name is " + customerName)
      Console.WriteLine("The address is " + customerAddress)
      Console.WriteLine("The phone is " + customerPhoneNo)

   End Sub

End Module
```

الشكل رقم (٦,١٠). شفرة برنامج الاختبار TesterOne.

في هذا المثال، يلعب الصنف TesterOne دور العميل، ويلعب الصنف Customer دور الخادم حيث يحتوي الصنف TesterOne على الإجراء Main والذي سيتم استدعاؤه مباشرة عند تنفيذ البرنامج. تنشئ الأوامر الموجودة داخل هذا الإجراء كائناً من الصنف Customer باسم firstCustomer حيث تستخدم الخاصية CustomerName لإسناد اسم العميل، ثم تستدعي الإجراءات من النوع Setters لإسناد قيم ابتدائية لصفات الكائن firstCustomer، على أن تسترجع وتظهر قيم صفات الكائن عن طريق استدعاء الخاصية CustomerName واستدعاء إجراءات الوصول Getters، وسوف نشرح أوامر الإجراء Main بالتفصيل خلال الفقرات التالية.

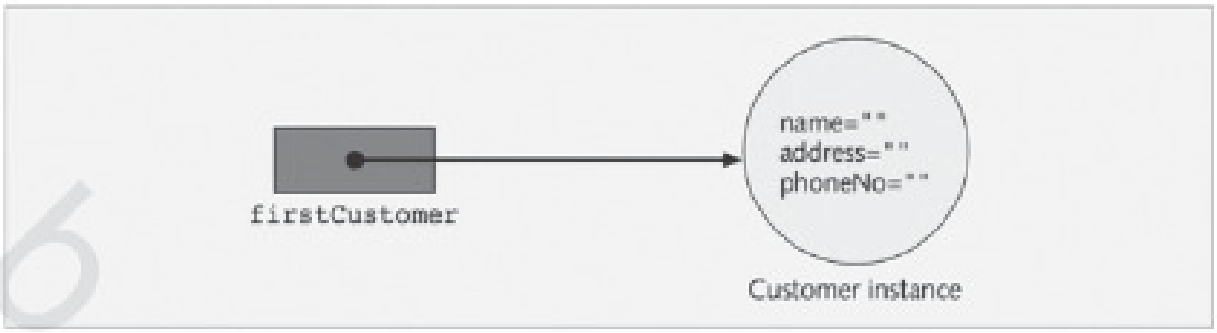
إنشاء كائن
Creating an Instance

لقد تعلمنا في الفصل الثالث أن المتغيرات في لغة VB .NET تنقسم إلى نوعين وهما: متغيرات أولية ومتغيرات إشارة. تُعرَّف المتغيرات الأولية مستخدمةً أنواع البيانات الأولية، وفي الحقيقة تحتوي هذه المتغيرات على البيانات، أما متغيرات الإشارة على الجانب الآخر فيتم تعريفها مستخدمةً اسم صنف ما بدلاً من نوع البيانات وتحتوي على عنوان كائن من هذا الصنف؛ ولذلك يقال إن هذا المتغير يشير إلى هذا الكائن. ولقد تعلمنا في الفصل الرابع أن الكائنات تأخذ مساحة من الذاكرة وتحتوي على بيانات (قيم الصفات)، ولقد أنشأنا سابقاً أيضاً كائنات من الصنف String والصنف DateTime، وفي مثال الصنف TesterOne أنشئ كائن من الصنف Customer حيث عرف متغير من النوع Customer لكي يشاور على هذا الكائن، ولقد استدعيت إجراءات الوصول Setters لإسناد بيانات العميل (الصفة name، والصفة address، والصفة phoneNo).

لقد تم تعريف كائن من الصنف Customer في المثال السابق على خطوتين حيث تم توصيف متغير الإشارة firstCustomer من النوع Customer، ثم تم إنشاء كائن من الصنف Customer مستخدماً الكلمة المحجوزة New، ولقد تم دمج هذين الأمرين في أمر واحد هكذا:

```
Dim firstCustomer As Customer = New Customer() ' create instance
```

وبذلك سيشاور المتغير firstCustomer على الكائن الذي تم إنشاؤه من الصنف Customer كما هو واضح في الشكل رقم (٦.١١).



الشكل رقم (٦,١١). كائن من الصنف Customer.

لاحظ أن صفات الكائن الواضح في الشكل رقم (٦,١١) لا تحتوي على أي قيم؛ ولذلك لابد من كتابة أوامر لإسناد قيم هذه الصفات مستدعيةً إجراءات الوصول Setters أو مستدعيةً الخصائص، ولقد استدعينا في هذا المثال الخاصية CustomerName لإسناد الاسم داخل الصفة name. وتذكر أن الخاصية تحتوي على كل من شق Set لإسناد قيمة للصفة والشق Get لاسترجاع قيمة من الصفة، ولقد استدعينا الشق Set هنا في هذا المثال لإسناد الاسم هكذا:

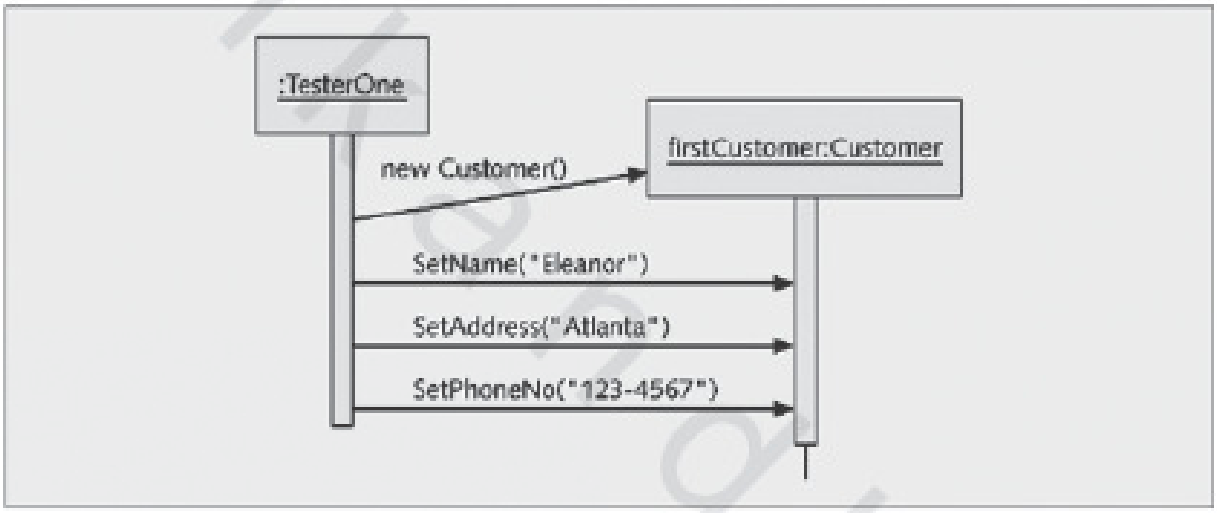
```
' use property to populate name
firstCustomer.CustomerName = "Eleanor"
```

ذكرنا سابقاً أن الخاصية تبدو للمستخدم على أنها صفة وليست إجراءاً (لا تحتاج إلى أقواس)؛ ولذلك فإن الأمر السابق يبدو وكأننا نسند القيمة "Eleanor" إلى متغير اسمه CustomerName داخل الكائن firstCustomer، وعلى أي حال إن هذا الأمر يستدعي الشق Set المعرف داخل الخاصية CustomerName.

ويمكنك أيضاً استدعاء إجراءات الوصول Setters لإسناد القيم لصفات الكائن firstCustomer حيث تمرر القيم كمعاملات لهذه الإجراءات والتي بدورها تسندها إلى الصفات، ويجب أن تلاحظ أنها استدعيت هذه الإجراءات من الكائن firstCustomer وليس من الصنف Customer حيث إن إجراءات الوصول ترتبط بالكائن وليس بالصنف حيث يوجد لكل كائن الصفات الخاصة به (الصفة name، والصفة address، والصفة phoneNo) هكذا:

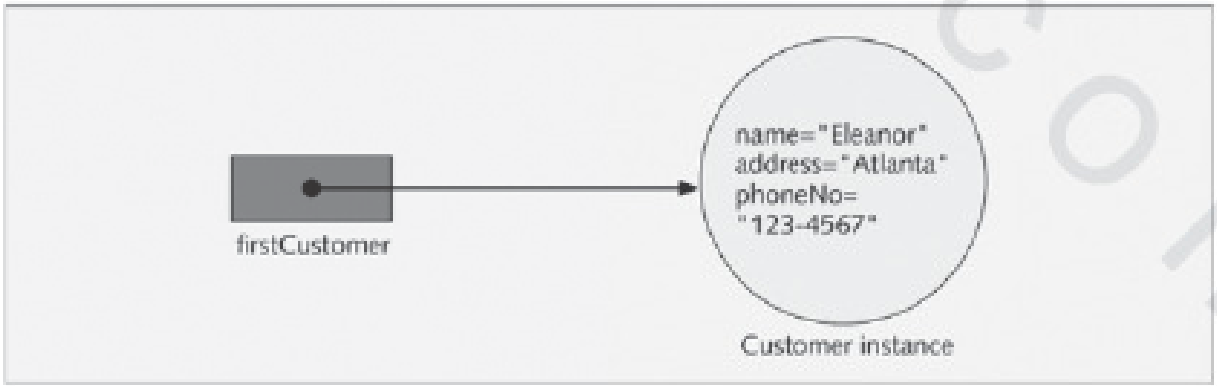
```
' invoke set accessors to populate attributes
firstCustomer.SetName("Eleanor")
firstCustomer.SetAddress("Atlanta")
firstCustomer.SetPhoneNo("123-4567")
```

لقد درسنا سابقاً نماذج Sequence Diagram في الفصل السابق وعرفنا أنها تستخدم لتوضيح التفاعل بين الكائنات في النظام. يوضح الشكل رقم (٦.١٢) نموذج Sequence Diagram الموجود في التفاعل بين الصنف TesterOne والصنف Customer لإنشاء بيانات كائن من الصنف Cusomer وإسنادها. تذكر أن الخطوط الأفقية تمثل رسائل من كائن إلى آخر والذي يعني أن كائن العميل TesterOne يرسل رسائل للكائن الخادم Customer طالباً منه إنجاز مهام (تنفيذ إجراءات معرفة لديه). وكما هو ملاحظ من الشكل رقم (٦.١٢) تم إنشاء كائن من الصنف Customer أولاً ثم استدعاء إجراءات الوصول Setters لإسناد قيم صفات الكائن. سوف نتعرف بعد قليل على الإجراء Constructor وهو إجراء من نوع خاص حيث يمكن إسناد قيم الصفات مع نفس أمر إنشاء الكائن في خطوة واحدة.



الشكل رقم (٦.١٢). نموذج Sequence Diagram لإنشاء كائن من الصنف Customer.

الآن يحتوي الكائن firstCustomer على قيم الصفات كما هو واضح في الشكل رقم (٦.١٣).



الشكل رقم (٦.١٣). الكائن يحتوي على قيم الصفات.

ولكي نتحقق من صحة عمل إجراءات الوصول Setters، يمكننا كتابة أوامر مسؤولة عن استرجاع قيم الصفات من الكائن وعرضها على الشاشة. ولعمل ذلك، نعرّف ثلاثة متغيرات من النوع String (لاستقبال قيم الصفات) هكذا:

```
' define variables to contain attribute values retrieved
Dim customerName, customerAddress, customerPhoneNo As String
```

ثم بعد ذلك نكتب الأوامر المسؤولة عن استرجاع قيم الصفات من الكائن مستخدمين إجراءات الوصول Getters لاسترجاع صفة name وصفة address وصفة phoneNo وإسنادها للمتغيرات الثلاث السابق تعريفها. ويوضح الشكل رقم (٦.١٤) هذه الأوامر مع شرح تفصيلي لها.

الشكل رقم (٦.١٤). تنفيذ إجراءات الوصول Getters للصنف Customer.

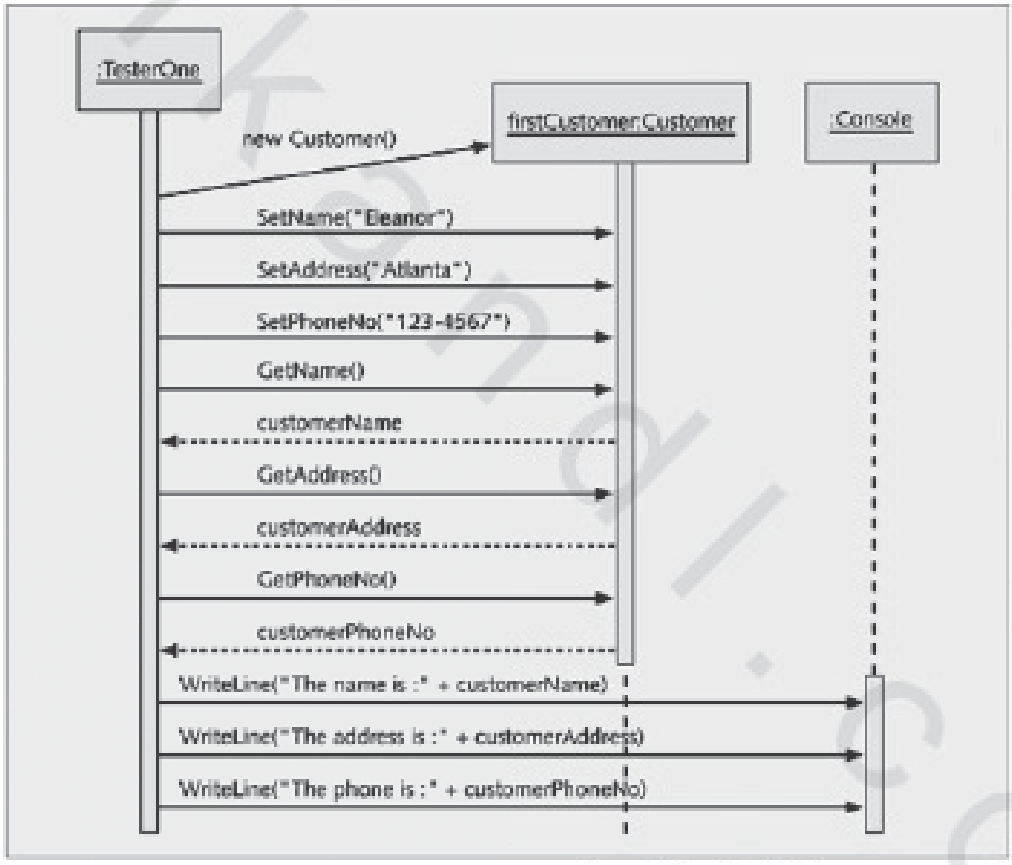
ويمكنك أيضاً استدعاء شق Get المعرف داخل الخاصية CustomerName لاسترجاع قيمة الصفة name وذلك بدلاً من استدعاء الإجراء GetName هكذا:

```
' use property to retrieve name
customerName = firstCustomer.CustomerName
```

بقي لنا معرفة آخر مهمة يقوم بها الصنف TesterOne وهي عرض قيم الصفات بعد استرجاعها حيث نستدعي الإجراء WriteLine المعرف داخل الصنف Console وذلك لعرض جملة نصية على الشاشة والتي تعرفنا عليها في كل من الفصل الثالث والرابع، حيث نمرر الجملة النصية المراد عرضها بين أقواس الإجراء (التي تتكون في هذه الحالة من نص ثابت والقيمة المسترجعة من صفة الكائن) هكذا:

```
' display the retrieved attribute values
Console.WriteLine("The name is " + customerName)
Console.WriteLine("The address is " + customerAddress)
Console.WriteLine("The phone is " + customerPhoneNo)
```

يحتوي الشكل رقم (٦.١٥) على نموذج Sequence Diagram يوضح التفاعل بين كل من الصنف TesterOne والصنف Cutomer، وكل من الصنف TesterOne والصنف Console لعرض قيم الصفات المسترجعة. لاحظ أن النموذج يوضح القيم المسترجعة من إجراءات الوصول Getters على شكل خط أفقي متقطع. يلعب الصنف TesterOne دور العميل ويلعب كل من الصنف Cutomer والصنف Console دور الخادم.



الشكل رقم (٦.١٥). نموذج Sequence Diagram للبرنامج TesterOne.

يوضح الشكل رقم (٦.١٦) مخرجات أوامر الإجراء main للصف TesterOne حيث يظهر الإجراء WriteLine سطراً عند استدعائه كل مرة.

```
The name is Eleanor
The address is Atlanta
The phone is 123-4567
```

الشكل رقم (٦,١٦). مخرجات البرنامج TesterOne.

إنشاء عدة كائنات
Creating Multiple Instance

كما هو معلوم لنا أن كائنات الصنف Customer تمثل عملاء شركة برادشو مارينا، ولقد عرفنا هذا الصنف سابقاً وأنشأنا منه كائناً واحداً فقط خلال المثال السابق (العميل Eleanor). ولأن شركة برادشو مارينا تتعامل مع العديد من العملاء، فيمكن إنشاء العديد من الكائنات من هذا الصنف وإسناد بيانات هذه العملاء داخل صفات هذه الكائنات كما هو واضح في الشكل رقم (٦,١٧). يوضح هذا الشكل أن الصنف Customer يمثل عملاء شركة برادشو مارينا، وأن تعريف الصنف تم استنتاجه من نموذج Class Diagram، وأنه يمكن إنشاء العديد من الكائنات مستخدماً هذا التعريف. ولذلك إن كل كائن من النوع Customer يمثل أحد عملاء شركة برادشو، وهم في هذا المثال: العميل Eleanor، والعميل Mike، والعميل JoAnn.

يوضح الشكل رقم (٦,١٨) صنف اختبار آخر اسمه TesterTwo والذي يستخدم لإنشاء ثلاثة كائنات من الصنف Customer المعرف سابقاً، وذلك لتمثيل العملاء الثلاثة الموضحين في الشكل رقم (٦,١٧). يحتوي برنامج الصنف TesterTwo على إجراءات الوصول Setters لإسناد بيانات العملاء الثلاثة، ثم استدعاء إجراءات الوصول Getters لاسترجاع بعض بيانات العملاء، وأخيراً يستدعي إجراء الوصول Setter لتغيير بعض بيانات العميل الثالث، وبعد ذلك استرجاع وعرض البيانات المعدلة.

لقد أنشأنا في الصنف TesterOne متغير إشارة واحد فقط يشير إلى كائن واحد من النوع Customer، أما في الصنف TesterTwo نحتاج إلى تعريف متغيريْ إشارة إضافيين للإشارة إلى كائنين آخرين من النوع Customer، والذي يمكن أن نسميه باسم SecondCustomer واسم ThirdCustomer هكذا:

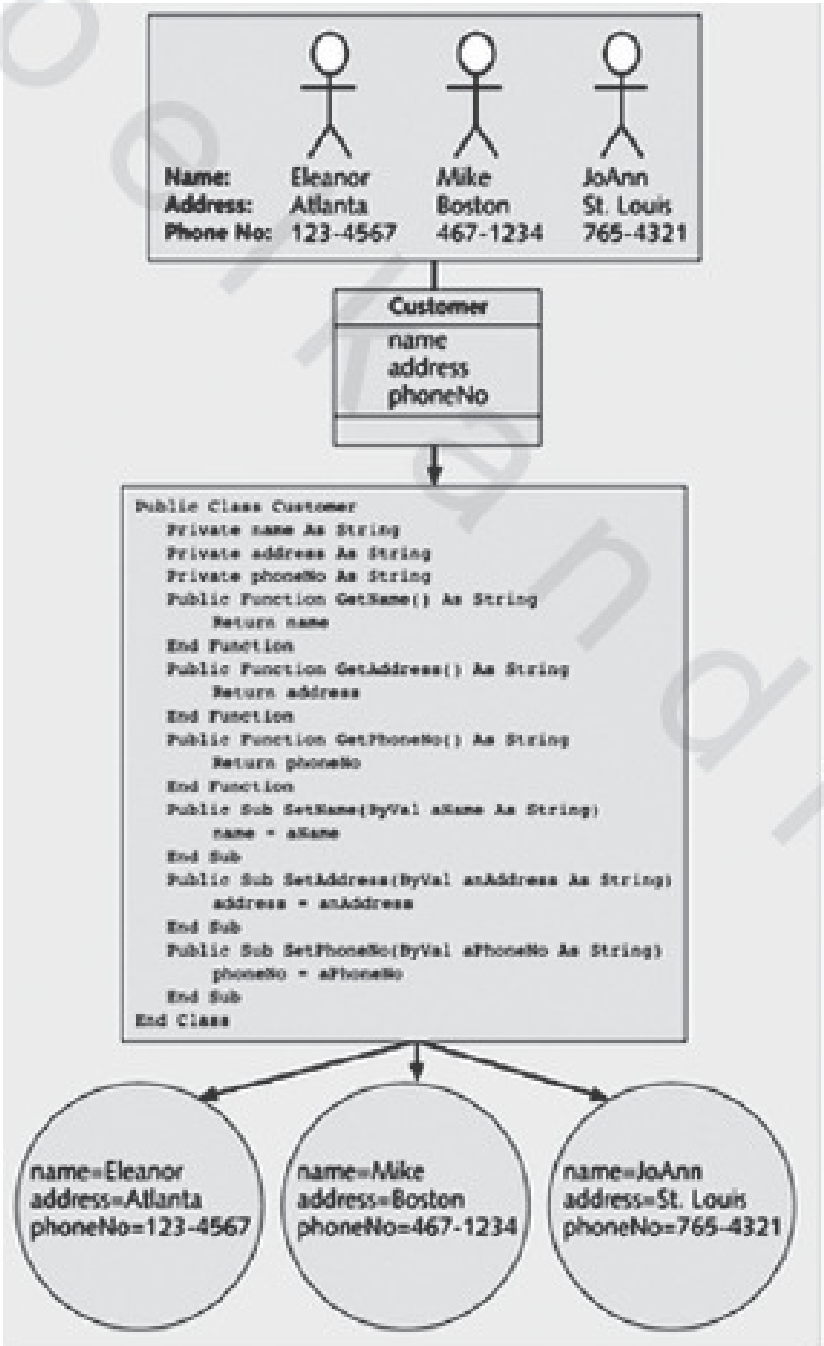
```
Dim firstCustomer, secondCustomer, thirdCustomer As Customer
```

لقد كتبت أمر واحد لإنشاء كائن من النوع Customer:

```
firstCustomer = New Customer()
```

ولإنشاء كائنين إضافيين، نكتب الأمرين التاليين:

```
secondCustomer = New Customer()
thirdCustomer = New Customer()
```



عملاء شركة برادشوا

الصنف
Customer

الملف
Customer.vb
الذى يمثل تعريف الصنف
Customer

كائنات من الصنف
Customer

الشكل رقم (٦,١٧). كائنات الصنف Customer.

```
' Chapter 6 TesterTwo class Example 2

Module TesterTwo

   Sub Main()
      Dim firstCustomer, secondCustomer, thirdCustomer As Customer

      firstCustomer = New Customer()       ' create three customers
      secondCustomer = New Customer()
      thirdCustomer = New Customer()

      firstCustomer.SetName("Eleanor")     ' populate first instance
      firstCustomer.SetAddress("Atlanta")
      firstCustomer.SetPhoneNo("123-4567")

      secondCustomer.SetName("Mike")       ' populate second instance
      secondCustomer.SetAddress("Boston")
      secondCustomer.SetPhoneNo("467-1234")

      thirdCustomer.SetName("JoAnn")       ' populate third instance
      thirdCustomer.SetAddress("St. Louis")
      thirdCustomer.SetPhoneNo("765-4321")

      ' display names of all three customers
      Console.WriteLine(firstCustomer.GetName())
      Console.WriteLine(secondCustomer.GetName())
      Console.WriteLine(thirdCustomer.GetName())

      ' change phone no for 3rd customer & redisplay
         thirdCustomer.SetPhoneNo("818-1000")
      Console.WriteLine("changed 3rd phone is " +
         thirdCustomer.GetPhoneNo())

   End Sub

End Module
```

الشكل رقم (٦,١٨). شفرة برنامج الاختبار TesterTwo.

أيضاً لقد استدعينا ثلاثة إجراءات مرور Setters لإنشاء بيانات العميل الأول هكذا:

```
firstCustomer.SetName("Eleanor")    ' populate first instance
firstCustomer.SetAddress("Atlanta")
firstCustomer.SetPhoneNo("123-4567")
```

واضح في الأوامر التالية:

```
secondCustomer.SetName("Mike")    ' populate second instance
secondCustomer.SetAddress("Boston")
secondCustomer.SetPhoneNo("467-1234")

thirdCustomer.SetName("JoAnn")    ' populate third instance
thirdCustomer.SetAddress("St. Louis")
thirdCustomer.SetPhoneNo("765-4321")

' display all names of all three customers
Console.WriteLine(firstCustomer.GetName())
Console.WriteLine(secondCustomer.GetName())
Console.WriteLine(thirdCustomer.GetName())
```

لاحظ أن الصنف TesterTwo أنشأ ثلاثة كائنات منفصلة من النوع Customer، وكل واحد منها له شخصيته المستقلة، وصفاته المستقلة، والمقدرة على الاستجابة إلى الرسائل المستقبلية. فعلى سبيل المثال، إذا طلبت من الكائن SecondCustomer إسناد القيمة "Mike" إلى الصفة name، فلابد من استدعاء إجراء الوصول SetName المسؤول عن إسناد القيمة "Mike" إلى الصفة name للكائن الثاني.

كما يمكن كتابة أوامر مسؤولة عن استرجاع قيمة الصفة name وإظهارها للكائنات الثلاثة حيث يمكن ربط كل من الإجراء GetName مع الإجراء WriteLine وتكوين أمر مركب واحد حيث تمرر القيمة العائدة من الإجراء GetName إلى الإجراء WriteLine. ولاحظ للمرة الثانية أن الأوامر التالية تستخدم متغير الإشارة firstCustomer لاستدعاء الإجراء GetName للكائن الأول، واستخدام كل من المتغير SecondCustomer والمتغير ThirdCustomer لكي نستدعي الإجراء GetName للكائن الثاني والكائن الثالث.

```
' display names of all three customers
Console.WriteLine(firstCustomer.GetName())
Console.WriteLine(secondCustomer.GetName())
Console.WriteLine(thirdCustomer.GetName())
```

لقد تعلمت كيف تستدعي إجراء الوصول Setter لإسناد قيمة إلى صفة داخل كائن، كما تعلمت كيف تغير قيمة صفة داخل كائن عن طريق استدعاء إجراء الوصول Setter أيضاً. توضح الأوامر التالية كيفية تغيير صفة phoneNo للكائن الثالث إلى القيمة 818-1000، ثم استرجاع الرقم وعرضه للتأكد من صحة تغير رقم الهاتف.

```
' change phone no for 3rd customer & redisplay
thirdCustomer.SetPhoneNo("818-1000")
Console.WriteLine("changed 3rd phone is " + thirdCustomer.GetPhoneNo())
```

يوضح الشكل رقم (٦,١٩) مخرجات برنامج الصنف TesterTwo حيث توضح السطور الثلاثة الأولى أسماء العملاء الثلاثة، والسطر الأخير يعرض رقم الهاتف المعدل للعميل الثالث.

```
Eleanor
Mike
JoAnn
changed 3rd phone is 818-1000
```

الشكل رقم (٦,١٩): مخرجات برنامج الاختبار TesterTwo.

إجراء إنشاء الكائن
Writing a Constructor Method

لقد تعلمنا كيف ننشئ كائنات من الصنف Customer وكيف تستدعى إجراءات الوصول Setters لإسناد قيم في صفات الكائن، وبالرغم من أن هذه الطريقة تعمل جيداً إلا أنه يمكن دمج هاتين الخطوتين في خطوة واحدة حيث يمكن تعريف إجراء Constructor داخل الصنف Customer والذي يتم استدعاؤه تلقائياً عند إنشاء كائن جديد مستخدماً الكلمة المحجوزة New. ويعتبر الإجراء Constructor إجراءاً فريداً من نوعه حيث يجب أن يسمى بالاسم New، ويجب تعريفه من نوع الإجراء Sub لأنه لا يعيد قيمة بعد تنفيذ أوامره، ويجب أن تعلم أن لغة VB .NET تضيف إجراء إنشاء افتراضي في حالة عدم كتابة هذا الإجراء داخل الصنف والذي لا يحتوي على أي أوامر، وإنما فقط يحتوي على رأس الإجراء والأمر End Sub، فعلى سبيل المثال إن إجراء الإنشاء الافتراضي للصنف Customer يبدو كما هو واضح في البرنامج التالي:

```
Public Sub New()
End Sub
```

تستطيع أيضاً إضافة إجراء إنشاء خاص بك داخل الصنف لاستقبال قيم تسند لصفات الكائن عند إنشائه، وفي هذه الحالة نطلق على هذه الإجراء اسم Parameterized Constructor، فعلى سبيل المثال إذا أردنا إضافة إجراء إنشاء يتقبل قيماً لإسنادها داخل صفات الكائن عند إنشائه للصنف، فلابد للكائن أن يستقبل ثلاثة معطيات لإسنادها للصفات الثلاثة وهي: صفة name، وصفة address، وصفة phoneNo. وعندئذ يستدعي هذا الإجراء إجراءات الوصول Setters لإسناد قيم الصفات، وبذلك يوفر استدعاء تلك الإجراءات إجراءاً لإسناد قيم صفات الكائن. يكتب إجراء الإنشاء للصنف Customer هكذا:

```
'constructor (3 parameters)
Public Sub New(ByVal aName As String, ByVal anAddress As String, ByVal
      aPhoneNo As String)
      SetName(aName)
      SetAddress(anAddress)
      SetPhoneNo(aPhoneNo)
End Sub
```

لاحظ أن اسم الإجراء New، والذي يتم تعريفه من النوع Public، يستقبل ثلاثة عوامل من النوع String وهي: المعامل aName، والمعامل anAddress، والمعامل aPhoneNo، وذلك لاستقبال القيم الثلاثة المراد إسنادها داخل صفات الكائن عند إنشائه. ولاحظ أيضاً أن جسم الإجراء New يستدعي إجراءات الوصول Setters ممرراً لها المعاملات المستقبلة لإسنادها داخل صفات الكائن.

ويمكن كتابة الإجراء السابق بطريقة أخرى حيث يمكن إسناد المعاملات المستقبلية مباشرة إلى الصفات دون استدعاء إجراءات الوصول Setters كما هو واضح في المثال التالي :

```
Public Sub New(ByVal aName As String, ByVal anAddress As String, ByVal
    aPhoneNo As String)
    name = aName
    address = anAddress
    phoneNo = aPhoneNo
End Sub
```

سوف نتعلم في الفصل التالي كيف نضيف أوامر التحقق من صحة البيانات قبل إسنادها لصفات الكائن داخل إجراءات الوصول Setters، وعند إضافة الأوامر المسؤولة عن ذلك فلابد من استدعاء تلك الإجراءات من داخل إجراء الإنشاء للقيام بالتحقق من صحة البيانات قبل إسنادها لصفات الكائن عند إنشائه.

ولتجربة إجراء الإنشاء المضاف للصنف Customer، فيمكن كتابة صنف جديد TesterThree والذي سوف يحتوي على أوامر أقل وذلك لاستدعاء إجراءات الوصول Setters من داخل إجراء الإنشاء Parameterized Constructor. يمكن تعريف متغيرات إشارة كما فعلنا داخل الصنف TesterTwo لثلاثة كائنات من الصنف Customer هكذا :

```
Dim firstCustomer, secondCustomer, thirdCustomer As Customer
```

ثم إضافة الأوامر المسؤولة عن إنشاء ثلاثة كائنات من الصنف Customer وذلك باستدعاء إجراء الإنشاء عند إنشاء الكائنات بشكل تلقائي وتمرير قيم صفات الكائنات كمعاملات للإجراء، ولاحظ أنك لا تحتاج إلى استدعاء إجراءات الوصول Setters التي سيتم استدعاؤها داخل إجراء الإنشاء :

```
' create three customers
firstCustomer = New Customer("Eleanor", "Atlanta", "123-4567")
secondCustomer = New Customer("Mike", "Boston", "467-1234")
thirdCustomer = New Customer("JoAnn", "St. Louis", "765-4321")
```

وأخيراً كما فعلنا سابقاً نستدعي إجراءات الوصول Getters لاسترجاع قيم صفات الكائنات وعرضها على الشاشة مباشرة دون اللجوء إلى تعريف متغيرات للاحتفاظ بها حيث يمكن استدعاء إجراءات الوصول Getters داخل إجراءات WriteLine، وذلك يعني تمرير القيمة العائدة من الإجراءات Getters إلى الإجراءات WriteLine.

```
' display names of all three customers
Console.WriteLine(firstCustomer.GetName())
Console.WriteLine(secondCustomer.GetName())
Console.WriteLine(thirdCustomer.GetName())
```

يوضح الشكل رقم (٦.٢٠) برنامج الصنف TesterThree كاملاً، ويوضح الشكل رقم (٦.٢١) مخرجات هذا البرنامج.

```
' Chapter 6 TesterThree class Example 3

Module TesterThree

   Sub Main()
      Dim firstCustomer, secondCustomer, thirdCustomer As Customer

      ' create three customers
      firstCustomer = New Customer("Eleanor", "Atlanta", "123-4567")
      secondCustomer = New Customer("Mike", "Boston", "467-1234")
      thirdCustomer = New Customer("JoAnn", "St. Louis", "765-4321")

      ' display names of all three customers
      Console.WriteLine(firstCustomer.GetName())
      Console.WriteLine(secondCustomer.GetName())
      Console.WriteLine(thirdCustomer.GetName())

   End Sub

End Module
```

الشكل رقم (٦.٢٠). شفرة برنامج الاختبار TesterThree.vb.

```
Eleanor
Mike
JoAnn
```

الشكل رقم (٦.٢١). مخرجات برنامج الاختبار TesterThree.

تعريف الإجراء TellAboutSelf
Writing a TellAboutSelf Method

لقد رأينا في الأمثلة السابقة أن أصناف الاختبار تستدعي إجراءات الوصول Getters وذلك لاسترجاع قيم صفات الكائنات وعرضها، وبالرغم من صحة عمل هذه الطريقة إلا أنها ربما تحتاج إلى كتابة الكثير من الأوامر

وخصوصاً إذا كان عدد صفات الكائن المراد عرض بياناته كثير. والأكثر أهمية أنه إذا تغير عدد الصفات أو نوع البيانات فيتطلب ذلك تغيير الأوامر الموجودة لدى الكائن العميل، وبذلك ربما نحتاج إلى تغيير الشفرة المسؤولة عن استرجاع البيانات وعرضها في جميع أنحاء النظام. ولتجنب هذه المشاكل يجب اتباع التصميم الجيد الذي يقترح عزل التغييرات في صنف دون تغيير الأصناف الأخرى لتقليل متطلبات صيانة النظام.

يعتبر الإجراء TellAboutSelf اختياراً بديلاً لاستدعاء إجراءات الوصول Getters من كائن (العميل) لاسترجاع قيم الصفات، حيث يُعرَّف هذا الإجراء داخل صنف مجال المشكلة ليعيد جميع قيم صفات الكائن في شكل كائن من النوع String ويأخذ هذا الإجراء درجة الوصول Public لتمكين أي صنف من استدعائه، ويجب أن يعرف من النوع Function لأنه يعيد قيمة.

```
'TellAboutSelf method
Public Function TellAboutSelf() As String
      Dim info As String
      info = "Name = " & GetName() & _
            ", Address = " & GetAddress() & _
            ", Phone No = " & GetPhoneNo()
      Return info
End Function
```

يحتوي هذا الإجراء المعرف داخل الصنف Customer على ثلاثة أوامر. يُعرِّف الأمر الأول متغيراً من النوع String اسمه info ليحتوي على قيم صفات الكائن، ويستدعي الأمر الثاني إجراءات الوصول Getters ودمج القيم العائدة من هذه الإجراءات في كائن من النوع String وإسناده للمتغير info (لاحظ استخدام الرمز "_" لكتابة هذا الأمر في ثلاثة سطور وذلك لتوضيحه)، ويعيد الأمر الثالث المتغير info للكائن العميل (الكائن الذي يستدعي هذا الإجراء) مستخدماً الكلمة المحجوزة Return.

لاحظ أن أوامر الإجراء TellAboutSelf تستدعي الإجراءات Getters مباشرة دون وضع اسم كائن من الصنف Customer مثل aCustomer.GetName لأن الإجراء TellAboutSelf يتم استدعاؤه لكائن محدد للصنف Customer، وبذلك فإن الأوامر الموجودة داخل الإجراء TellAboutSelf تنفذ لهذا الكائن.

يوضح الشكل رقم (٦,٢٢) صنف الاختبار الرابع TesterFour والذي ينشئ كائنات من النوع Customer كما فعلنا سابقاً، ولكن على عكس الأمثلة السابقة سوف نستدعي الإجراء TellAboutSelf لكل كائن للحصول على قيم صفاته بدلاً من استدعاء إجراءات الوصول Getters، وسوف نستدعي الإجراء TellAboutSelf داخل الإجراء WriteLine. يوضح الشكل رقم (٦,٢٣) مخرجات برنامج الصنف TesterFour.

```
' Chapter 6 TesterFour class Example 4

Module TesterFour

   Sub Main()
      Dim firstCustomer, secondCustomer, thirdCustomer As Customer

      ' create three customers
      firstCustomer = New Customer("Eleanor", "Atlanta", "123-4567")
      secondCustomer = New Customer("Mike", "Boston", "467-1234")
      thirdCustomer = New Customer("JoAnn", "St. Louis", "765-4321")

      ' display all info for all three customers
      Console.WriteLine(firstCustomer.TellAboutSelf())
      Console.WriteLine(secondCustomer.TellAboutSelf())
      Console.WriteLine(thirdCustomer.TellAboutSelf())

   End Sub

End Module
```

الشكل رقم (٦,٢٢). شفرة برنامج الاختبار TesterFour.vb.

```
Name = Eleanor, Address = Atlanta, Phone No = 123-4567
Name = Mike, Address = Boston, Phone No = 467-1234
Name = JoAnn, Address = St. Louis, Phone No = 765-4321
```

الشكل رقم (٦,٢٣). مخرجات برنامج الاختبار TesterFour.

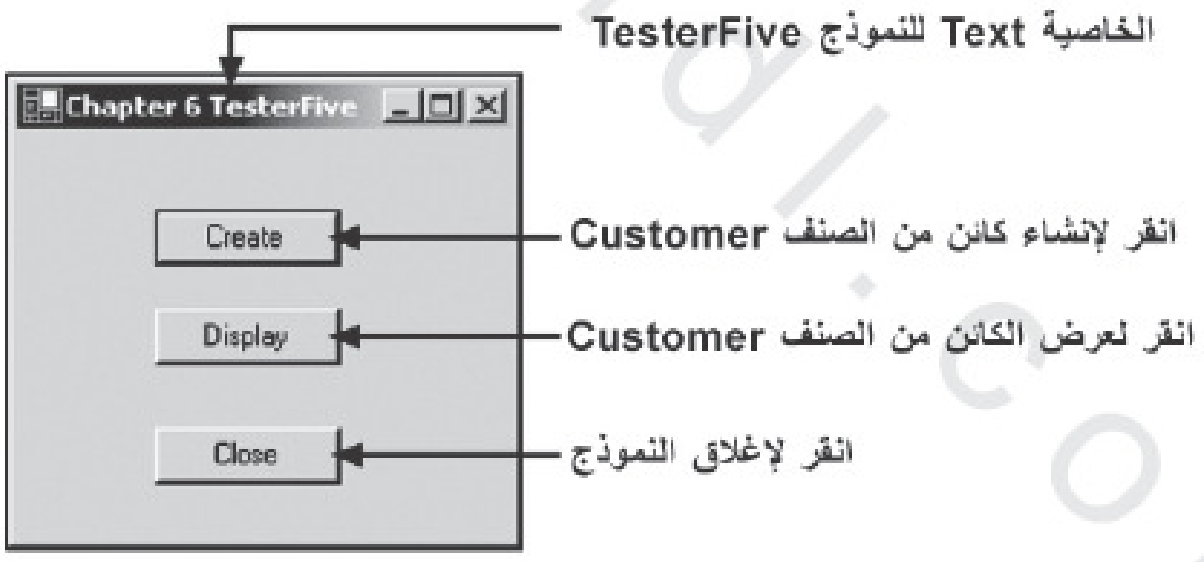
تعريف صنف اختبار من نوع نموذج
Writing a Tester Class as a Form

لقد أنشأت في الأمثلة السابقة أصناف اختبار من النوع Module وأنشأت برامج من النوع Console لتنفيذها، وإذا استرجعت معي الفصل الثاني لوجدت أنه يوجد نوع آخر من البرامج وهو تطبيقات Windows والتي تستخدم عندما نريد إنشاء واجهة رسومية للتعامل مع المستخدم (مجموعة من النماذج). ولقد رأينا أن النموذج (Form) هو كائن مرئي عبارة عن نافذة تحتوي على أزرار وكائنات مرئية أخرى.

سوف ننشئ تطبيق Windows الذي يحتوي على نموذج اسمه TesterFive الذي ينجز ما أنجزه الصنف TesterFour في المثال السابق، وقبل أن تستمر يمكنك أن تراجع موضوع "إنشاء تطبيقات Windows" في الفصل الثاني (لاحظ أيضاً أنك سوف تتعلم كيفية إنشاء أصناف مرئية في كل من الفصل العاشر والحادي عشر). الغرض من هذا المقطع هو تقديم موضوع إنشاء أصناف اختبار مرئية، والمطلوب فقط هو إضافة أوامر مع إجراءات الأحداث المعرفة سابقاً (مثل الضغط على زر) حيث سنستخدم الصنف Customer من المثال السابق دون تغيير.

يوضح الشكل رقم (٦,٢٤) النموذج form المستخدم في هذا المثال، حيث يحتوي على ثلاثة أزرار وهي: الزر "Create"، والزر "Display"، والزر "Close". ولقد سمينا هذه الأزرار الأسماء btnCreate و btnDisplay و btnClose بالترتيب وبداية الأسماء بـ "btn" تعني في لغة VB .NET أن هذه كائنات من النوع Button. عندما تضغط على أحد هذه الأزرار ينشأ حدث (Event) والذي يجب أن يكتب له إجراء دالة يستدعى عند حدوث هذا الحدث ونطلق على هذا الإجراء اسم Event Procedure أو Event Handler. في هذا المثال ينشئ إجراء الحدث Create ثلاثة كائنات من النوع Customer، ويظهر إجراء الحدث Display معلومات حول هذه الكائنات، ويغلق إجراء الحدث Close النموذج.

وكما سوف تتعلم في الفصل العاشر عند إنشاء تطبيق Windows يحتوي على نموذج، سوف تولد لغة VB .NET بعض شفرة البرنامج بالنيابة عنك، ولكن الشكل رقم (٦,٢٥) لا يوضح هذه الشفرة بل يوضح فقط إجراءات الأحداث المناظرة للأزرار الثلاثة، ولاحظ أيضاً أن متغيرات الإشارة firstCustomer و secondCustomer و ThiredCustomer تم تعريفها خارج الإجراءات الثلاثة وذلك يسمح لها التعامل مع هؤلاء المتغيرات لأنه إذا تم تعريف متغير داخل حدود إجراء فإن حدود التعامل معه لا تتعدى حدود هذا الإجراء.



الشكل رقم (٦,٢٤). نموذج TesterFive.

```
Public Class TesterFive
   Dim firstCustomer, secondCustomer, thirdCustomer As Customer

   Private Sub btnCreate_Click(ByVal sender As System.Object, ByVal e
     As System.EventArgs) Handles btnCreate.Click
      ' create three customers
      firstCustomer = New Customer("Eleanor", "Atlanta", "123-4567")
      secondCustomer = New Customer("Mike", "Boston", "467-1234")
      thirdCustomer = New Customer("JoAnn", "St. Louis", "765-4321")
End Sub

   Private Sub btnDisplay_Click(ByVal sender As System.Object, ByVal e
     As System.EventArgs) Handles btnDisplay.Click
      ' display all info for all three customers
      MessageBox.Show(firstCustomer.TellAboutSelf())
      MessageBox.Show(secondCustomer.TellAboutSelf())
      MessageBox.Show(thirdCustomer.TellAboutSelf())
   End Sub

   Private Sub btnClose_Click(ByVal sender As System.Object, ByVal e As
     System.EventArgs) Handles btnClose.Click
      Me.Dispose()
   End Sub
End Class
```

الشكل رقم (٦,٢٥). شفرة برنامج الاختبار TesterFive.vb.

عندما تضغط على الزر Create، فإن الإجراء btnCreate_Click سوف ينفذ والذي ببساطة ينشئ ثلاث كائنات من النوع Customer مثلما فعلت في الصنف TesterFour. وعندما تضغط على الزر Display، سوف ينفذ الإجراء btnDisplay_Click الذي يظهر معلومات حول هؤلاء الكائنات، وعلى أي حال سوف تستخدم الصنف MessageBox لعرض البيانات الراجعة من الإجراء TellAboutSelf لكل كائن (لقد تعلمت كيفية استخدام الصنف MessageBox في الفصل الرابع). يوضح الشكل رقم (٦,٢٦) أول رسالة ستظهر من مخرجات البرنامج TesterFive.

الشكل رقم (٦,٢٦). مخرجات برنامج الاختبار TesterFive.

ملخص الفصل
Chapter Summary

- يسمي مبرمجو لغة VB .NET المعرفات (الأسماء التي تسند للأصناف والصفات والإجراءات) بأسماء معبرة حيث يبدأ اسم الصفة بحرف صغير (Small)، وإذا تكوَّن اسم الصفة من أكثر من كلمة يجب أن تبدأ الكلمات التالية للكلمة الأولى بحرف كبير (Capital) ويبدأ اسم الإجراء بحرف كبير والكلمات التالية للكلمة الأولى تبدأ أيضاً بحرف كبير، ولابد أن تكون أول كلمة فعلاً والكلمة التالية تكون اسماً.
- يمثل صنف مجال المشكلة صنفاً لكائنات العالم الحقيقي والذي نريد أن نمثله داخل النظام الذي سيقوم بحل هذه المشكلة حيث يجب تعريف كل أصناف مجال المشكلة.
- إن برنامج VB .NET الذي يمثل صنفاً معيناً يطلق عليه تعريف الصنف (Class Definition). يحتوي تعريف الصنف في لغة VB .NET على رأس الصنف (Class Header) متبوعاً بتعريف صفات الصنف ثم تعريف إجراءات الصنف. تمثل صفات الصنف مجموعة من المتغيرات التي يتم إنشاؤها مع كل كائن يتم إنشاؤه من الصنف.
- تستخدم إجراءات الوصول (Accessor Methods) بواسطة الإجراءات الأخرى لتخزين قيم الصفات المعرفة من النوع Private واسترجاعها حيث يطلق على الإجراءات التي تغير قيمة صفة بإجراءات الوصول للإسناد (أو ببساطة Setters)، أما نوع إجراء الدالة فسيستخدم لتعريف الإجراءات من النوع Getters لأنها تعيد قيمة. يطلق على إجراءات الوصول اسم الإجراءات القياسية (Standard Methods) والتي لا تظهر في نموذج Class Diagram لأن المطورين يفترضون وجودهم بشكل افتراضي.
- يمكنك استبدال كل من الإجراء Getter والإجراء Setter بتطوير خاصية (Property) لكل صفة. والخاصية تشبه الإجراء وتحتوي على كل من الشق Get والشق Set ؛ ولذلك تبدو للكائن (العميل) مثل الصفة (أي متغيراً).
- يمكن تمثيل علاقة الكائن المرسل للرسالة (طالب الخدمة) والكائن المستقبل للرسالة (منفذ الخدمة) بعلاقة نموذج العميل–الخادم (Client-Server Model) حيث يصبح الكائن الطالب للخدمة هو العميل (Client Object) والكائن المنفذ للخدمة هو الخادم (Server Object). يرسل العميل رسالة إلى الخادم طالباً منه تنفيذ إجراء لديه، وربما يرسل قيماً مع الرسالة أولاً في شكل معاملات (Arguments)، عندئذ ينفذ الخادم الإجراء المناظر للرسالة لإنجاز المهمة المطلوبة وربما يعيد قيمة للعميل.
- يمكن تعريف إجراء Constructor داخل الصنف ليتم استدعاؤه تلقائياً عند إنشاء كائن جديد مستخدماً الكلمة المحجوزة New. ويعتبر الإجراء Constructor إجراءاً فريداً من نوعه حيث يجب أن يسمى بالاسم New، ويجب تعريفه من نوع الإجراء Sub لأنه لا يعيد قيمة بعد تنفيذ أوامره، كما يمكن أن يستقبل هذا الإجراء معاملات لإسناد قيمها إلى الصفات مع أمر إنشاء الكائن نفسه في خطوة واحدة.

- يمكن تعريف الإجراء TellAboutSelf داخل صنف مجال المشكلة كاختيار بديل لاستدعاء إجراءات الوصول Getters من كائن العميل لاسترجاع قيم الصفات حيث يستخدم هذا الإجراء ليعيد جميع قيم صفات الكائن في شكل كائن من النوع String.

المصطلحات الأساسية

Key Terms

إجراء الوصول (Accessor Method)	معامل (Argument)
رأس الصنف (Class Header)	إجراء Constructor
تعريف الصنف (Class Definition)	كائن العميل (Client Object)
الإجراء الخاص (Custom Method)	كائن الخادم (Server Object)
حدث (Event)	خاصية (Property)
إجراء Event Procedure	الإجراء القياسي (Standard Method)
إجراء Event Handler	إجراء دالة (Function Procedure)
إجراء فرعي (Sub Procedure)	

أسئلة المراجعة

Review Questions

١- ما هو تعريف الصنف؟

٢- ما هو تعريف الصفة؟

٣- ما هي شروط تسمية الصفة؟

٤- ما هو المتغير الأولي؟

٥- ما هو متغير الإشارة؟

٦- ما وظيفة إجراء الوصول للاسترجاع Getter؟

٧- ما وظيفة إجراء الوصول للإسناد Setter؟

٨- اشرح مستويات الوصول للصفة.

٩- اشرح مستويات الوصول للإجراء.

١٠- وضح الفرق بين المعطى والمعامل.

١١- ما هو الإجراء القياسي؟

١٢- لماذا يكون إجراء الوصول للاسترجاع Getter عادة ذو قائمة معاملات فارغة؟

١٣- لماذا تكتب إجراء الوصول للإسناد Setter كإجراء فرعي بينما إجراء الوصول للاسترجاع Getter كإجراء دالة؟

١٤- ما أسباب استدعاء إجراء الإنشاء؟

١٥- كيف يتم تسمية إجراء الإنشاء؟

١٦- ما هو إجراء الإنشاء الافتراضي؟

١٧- ما هو إجراء الإنشاء ذو المعاملات؟

أسئلة المناقشة

Discussion Questions

١- لماذا لا نحتاج أن نعرف إجراءات مرور للصفات المعرفة من النوع العام (Public)؟

٢- ما الفرق بين الخاصية (Property) والصفة (Attribute)؟ وما الفرق أيضاً بين الخاصية وإجراءات المرور؟

٣- اشرح كيف يمكن لإجراء دالة أن يعيد أكثر من قيمة حتى ولو كان العائد هو متغير واحد؟

٤- ما فائدة استدعاء إجراءات المرور Setters داخل إجراء الإنشاء لإسناد قيم الصفات بدلاً من إسنادها مباشرة؟

٥- اسرد كيف يفيد استخدام الإجراء TellAboutSelf أثناء صيانة البرنامج؟

٦- افترض أن هناك كائنان معرفان من الصنف Customer حيث يشار إلى الكائن الأول بالمتغير cashCustomer ويشار إلى الكائن الثاني بالمتغير anotherCustomer. ما هي نتيجة تنفيذ الأمر التالي:

```
Customer anotherCustomer = cashCustomer;
```

مشاريع الفصل

Projects

١- افترض أنه طُلب منك تطوير نظام يحتوي على صنف مجال المشكلة Employee الذي يمثل صنف جميع موظفي الشركة. يحتوي هذا الصنف على الصفة employeeName (من النوع String) والصفة dateEmployed (من النوع DateTime) والصفة annualSalary (من النوع Double). ربما تحتاج مراجعة الصنف من الفصل الرابع.

(أ) قم بإنشاء مجلد Proj01 أسفل المجلد Chap06\Projects وهو مجلد العمل الخاص بك، ثم قم بإنشاء مشروع VB .NET من النوع Console داخل هذا المجلد.

(ب) عرف صنف مجال المشكلة Employee مشتملاً على الإجراءات القياسية وإجراء إنشاء يقبل معاملات والإجراء TellAboutSelf.

(ج) عرف الإجراء TellAboutSelf بحيث يحتوي على أوامر تشكيل الصفة dateEmployed لتصبح على الشكل "monthname day, year" والصفة annualSalary على شكل عملة.

(د) لكي نختبر هذا البرنامج، يجب تعريف صنف التجربة ProjectTester الذي ينشئ ثلاثة كائنات، ثم يستدعي الإجراء TellAboutSelf من الكائنات الثلاثة.

(هـ) قم بتشغيل البرنامج للتأكد أنه يعمل بشكل صحيح.

٢- قم بإنشاء مجلد Proj02 أسفل المجلد Chap06\Projects وهو مجلد العمل الخاص بك، ثم قم بإنشاء مشروع VB .NET من النوع Console داخل هذا المجلد باسم Project2.

(أ) استعن بالبرنامج TesterFour من هذا الفصل لتصميم صنف اختبار باسم Project2Tester وتكتبه الذي سيقوم بإنشاء ثلاث كائنات من الصنف Customer، ثم قم بإضافتها داخل مصفوفة أحادية البعد.

(ب) أسند الاسم Customers للمصفوفة.

(ج) عرف المصفوفة من النوع Customer.

(د) استخدم الملف Customer.vb من المثال الرابع الذي يحتوي على تعريف الإجراء TellAboutSelf.

(هـ) اكتب تكراراً يحتوي على أوامر استرجاع معلومات العملاء الثلاثة من المصفوفة وعرضها مستخدماً الإجراء TellAboutSelf.