الفصل السادس

فهم التوارث والواجهات

UNDERSTANDING INHERITANCE AND INTERFACES

أهداف الفصل:

- تطوير شجرة توارث الصنف Boat.
- معرفة معنى كل من الصنف النهائي (Final Class) والصنف المجرد (Abstract Class).
- استعمال الكلمات المحجوزة MustInherit و NotInheritable.
- إعادة تعرف إجراء صنف الأب.
- الفرق بين الكلمة Private، والكلمة Protected.
- اكتشاف الأصناف الفرعية للصنف Lease والتعرض للإجراءات المجردة (Abstract Methods).
- فهم الواجهات (Interfaces) واستخدامها.
- تعريف استثناءات خاصة (Custom Exceptions).
- التعرف على الصنف Object وعلى مفهوم التوارث Inheritance.

لقد تعلمنا خلال الفصل السادس والفصل السابق كيف نطور ونختبر أصناف مجال المشكلة (Problem Domain Classes) ونطورها حيث طورنا صنفين من أصناف مجال مشكلة شركة برادشو مارينا (الصنف Customer، والصنف Slip)، ولكن تذكر أن نظام شركة برادشو مارينا (كما هو واضح في الفصل الخامس) يحتوي على علاقات توارث، حيث يوجد نوعين من المراكب (Boats) ويوجد نوعان من العقود (Leases). سوف نطور خلال هذا الفصل الصنف الرئيس Boat، ثم نطور الأصناف الفرعية (الصنف Sailboat، والصنف Powerboat) لهذا الصنف ونختبرها والتي سترث صفات وإجراءات الصنف Boat. بالإضافة إلى ذلك سنطور الأصناف الفرعية للصنف Lease وهي الصنف AnnualLease والصنف DailyLease.

يعتبر مفهوم التوارث (Inheritance) أقوى مفهوم في تقنية الكائنات (Object-Oriented Approach)، حيث يمكنك بسهولة تكبير صنف موجود ومن ثم يسهل عليك عملية تطوير النظم وعملية إعادة استخدام الأصناف (Reusability)، ولقد تميزت لغة VB .NET عن لغة VB 6.0 بتوفير مفهوم التوارث الحقيقي، حيث تقدم لغة VB .NET الكلمة المحجوزة Inherits التي تستخدم لإنشاء صنف فرعي (Sub Class) من صنف رئيس (Super Class). وسوف نطبق مفهوم التوارث في هذا الفصل على كل من أصناف مجال المشكلة وعلى الصنف Exception المعرف لدى لغة VB .NET.

يعتبر مفهوم التوارث أحد طرق تطبيق مفهوم تعدد تعريف الإجراءات (Polymorphism)، فعلى سبيل المثال عند تطوير الصنف الرئيس Boat وأصنافه الفرعية الصنف Sailboat والصنف Powerboat سنجد أن كلاً من كائنات الصنف Sailboat وكائنات الصنف Powerboat ستستجيب إلى الرسالة TellAboutSelf بطرق مختلفة. ولتطبيق ذلك، يمكنك إعادة تعريف الإجراء TellAboutSelf المعرف لدى الصنف Boat في كل من الصنف الفرعي Sailboat والصنف الفرعي Powerboat بالتوقيع نفسه.

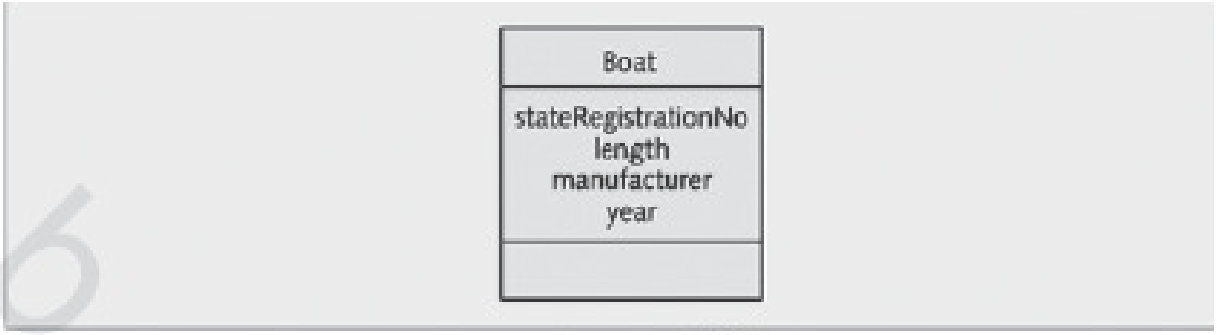
سوف نوضح أيضاً في هذا الفصل مفاهيم أخرى تتعلق بتقنية التوارث مثل تعريف الإجراءات المجردة (Abstract Methods) والواجهات والاستثناءات الخاصة مستخدمين الأصناف الفرعية للصنف Lease، حيث توجد العديد من الأساليب لكي تطلب من الأصناف أن تشتمل على تعريف إجراءات معينة وذلك للتأكد من أن كائنات هذه الأصناف ستستجيب لنفس الرسالة. أولاً، يمكن استخدام الإجراءات المجردة مع الصنف الرئيس والذي تتطلب من الصنف الفرعي أن يعرفها بداخله. ثانياً، يمكن استخدام الواجهات التي تتطلب من الأصناف أن تطور الإجراءات المعرفة داخل هذه الواجهات.

بعد ذلك ستتعلم كيف نطبق مفهوم التوارث على أصناف غير أصناف مجال المشكلة، فعلى سبيل المثال يمكن تعريف صنف استثناء خاص (Custom Exception) وذلك بوراثة الصنف Exception المعرف داخل لغة VB .NET. وأخيراً، سوف نناقش الصنف Object الذي يعتبر الصنف الأب لجميع الأصناف وجميع أصناف مجال المشكلة التي أنشأتها لنظام شركة برادشو ماورينا ترث هذا الصنف بشكل تلقائي.

تطوير شجرة توارث الصنف Boat

Implementing the Boat Generalization/Specialization Hierarchy

كما وضحنا في الفصل الخامس أن النموذج Class Diagram لنظام شركة برادشو ماورينا يحتوي على الصنف Boat والذي يحتوي على معلومات المراكب التي يجب تسجيلها في النظام مثل رقم تسجيل حالة المركب، وطول المركب، ومصنع المركب، وسنة صنع المركب. يوضح الشكل رقم (٨.١) الصنف Boat كما هو موضح في نموذج Class Diagram لشركة برادشو، كما يوضح الشكل رقم (٨.٢) شفرة تعريف الصنف Boat بلغة VB .NET.



الشكل رقم (٨,١). الصنف Boat.

```
' Boat -- an initial Boat Class
' make abstract by inserting MustInherit keyword

Public Class Boat

    'attributes
    Private stateRegistrationNo As String
    Private length As Single
    Private manufacturer As String
    Private year As Integer

    'constructor (four parameters)
    Public Sub New(ByVal aStateRegistrationNo As String, _
        ByVal aLength As Single,
        ByVal aManufacturer As String, ByVal aYear As Integer)
        'set values of attributes using accessor methods
        SetStateRegistrationNo(aStateRegistrationNo)
        SetLength(aLength)
        SetManufacturer(aManufacturer)
        SetYear(aYear)
    End Sub

    'TellAboutSelf method
    Public Overridable Function TellAboutSelf() As String
        Return (stateRegistrationNo & ", " & length & ", " & _
          manufacturer & ", " & year)
    End Function

    'Get accessor methods
    Public Function GetStateRegistrationNo() As String
        Return stateRegistrationNo
    End Function
    Public Function GetLength() As Single
        Return length
    End Function
    Public Function GetManufacturer() As String
        Return manufacturer
    End Function
    Public Function GetYear() As Integer
        Return year
    End Function

    'Set accessor methods
    Public Sub SetStateRegistrationNo(ByVal aStateRegNo As String)
        stateRegistrationNo = aStateRegNo
    End Sub
    Public Sub SetLength(ByVal aLength As Double)
        length = aLength
    End Sub
    Public Sub SetManufacturer(ByVal aManufacturer As String)
        manufacturer = aManufacturer
    End Sub
    Public Sub SetYear(ByVal aYear As Integer)
        year = aYear
    End Sub

End Class
```

الشكل رقم (٨,٢). تعريف الصنف Boat.

أولاً، تم كتابة رأس الصنف Boat، ثم تم تعريف الصفات الأربعة للصنف كما فعلت سابقاً لكل من الصنف Customer والصنف Slip. وأيضاً تم تعريف إجراء الإنشاء الذي يستقبل أربعة معاملات لإسنادها داخل الصفات الأربعة.

ثم بعد ذلك تم تعريف أربعة إجراءات وصول من النوع Setters وكذلك أربعة إجراءات وصول من النوع Getters. تذكر أيضاً أنه بالإمكان تعريف خصائص (Properties) بدلاً من إجراءات الوصول كما فعلنا في الفصل السادس، ولكن سوف نستخدم هنا في هذا الفصل إجراءات الوصول بدلاً من تعريف الخصائص. ويستدعي إجراء الإنشاء إجراءات الوصول Setters مثلما فعلنا في المثال السابق. لاحظ أن إجراءات الوصول Setters لا تحتوي على أوامر التحقق من صحة البيانات وذلك لتبسيط أوامرها. يحتوي أيضاً الصنف Boat على الإجراء TellAboutSelf الذي سوف يستخدم لاحقاً في هذا الفصل.

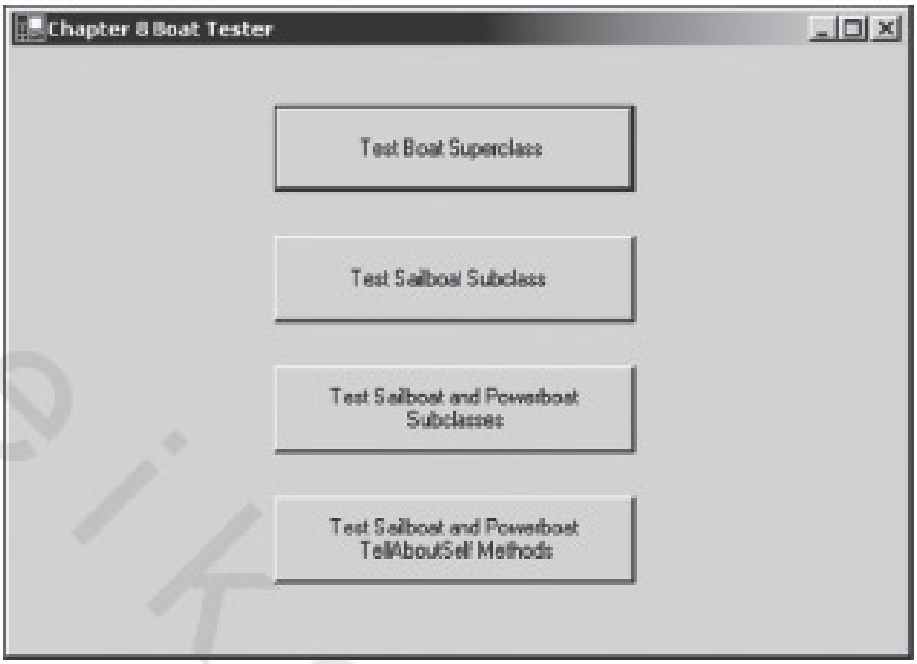
اختبار الصنف الرئيس Boat باستخدام تطبيق ويندوز
Testing the Boat Superclass with a Windows Form

لقد اختبرنا معظم أصناف مجال المشكلة السابقة مستخدمين تطبيقات Console التي تحتوي على ملف برمجي (Module) حيث يتم تنفيذ أوامر الإجراء Main الذي تم تعريفه داخل هذا الملف لاختبار أصناف مجال المشكلة، كما أنشأنا أيضاً في الفصل السادس والسابع تطبيقات Windows لبعض الأمثلة القليلة بغرض اختبار أصناف مجال المشكلة، وسوف ننشئ تطبيقات Windows خلال هذا الفصل وتعريف نموذج يحتوي على أزرار (Buttons) لاختبار جميع أصناف مجال المشكلة. يمكن الرجوع للفصل السابق لتذكر كيف ننشئ نماذج Windows. عزيزي القارئ تذكر أن الفائدة من اختبار أصناف مجال المشكلة (كما علمنا سابقاً) هو التأكد من صحة أوامر هذه الأصناف.

يوضح الشكل رقم (٨.٣) نموذجاً يحتوي على أربعة أزرار أطلقنا على الزر الأول عنوان "Test Boat Superclass" وأسندنا له الاسم btnBoat وعرفنا الإجراء btnBoat_Click الذي سيتم استدعاؤه بشكل تلقائي عند الضغط على هذا الزر والذي يحتوي على أوامر اختبار الصنف Boat (انظر الشكل رقم ٨.٤)، كما يوضح الشكل رقم (٨.٥) مخرجات هذا الإجراء.

ينشئ الإجراء السابق كائنان من الصنف Boat ثم يسترجع معلومات هذين الكائنين مستخدماً إجراءات الوصول Getters، ثم يتم عرض هذه المعلومات على شاشة Console مستخدماً WriteLine المعرف في الصنف Console.

يوضح الشكل رقم (٨.٦) نموذج Sequence Diagram الذي يصور التفاعل بين أوامر اختبار الصنف Boat. لاحظ أن هذا النموذج يوضح عملية إنشاء كائن واحد فقط من الصنف Boat. ولاحظ أيضاً أن الصنف Boat جاهز الآن للعب دور الصنف الرئيس (Superclass).



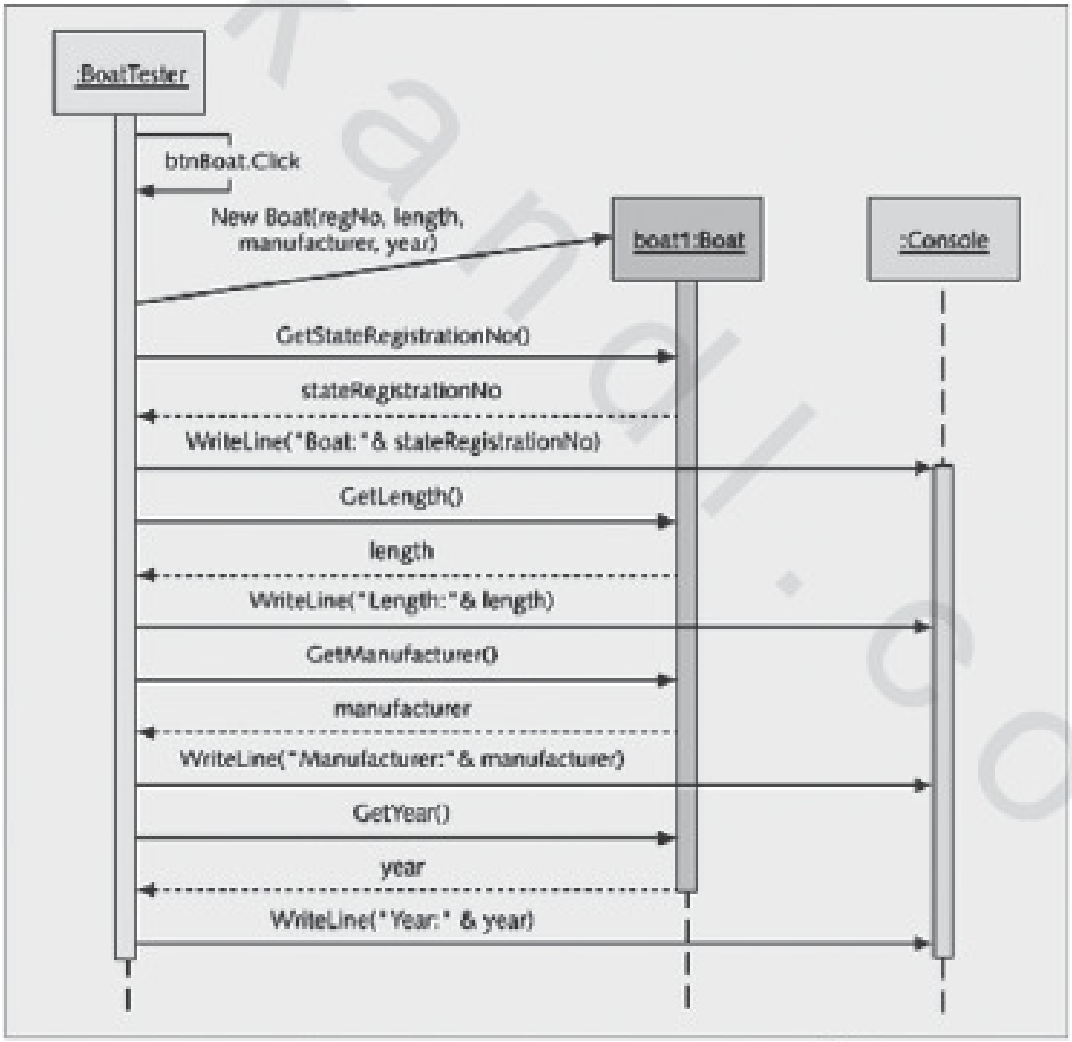
الشكل رقم (٨,٣). نموذج تطبيق Windows الخاص باختبار الصنف Boat.

```
Private Sub btnBoat_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnBoat.Click

    Console.WriteLine("Begin Boat Test:")

    'declare 2 boat references
    Dim boat1 As Boat
    Dim boat2 As Boat

    'instantiate 2 boats
    boat1 = New Boat("MO1234", 28, "Tartan", 2001)
    boat2 = New Boat("CA9876", 32, "Catalina", 2003)

    'invoke getter methods for boat1
    Console.WriteLine("Boat No: " & boat1.GetStateRegistrationNo())
    Console.WriteLine("Length: " & boat1.GetLength())
    Console.WriteLine("Manufacturer: " & boat1.GetManufacturer())
    Console.WriteLine("Year: " & boat1.GetYear())

    'invoke getter methods for boat2
    Console.WriteLine("Boat No: " & boat2.GetStateRegistrationNo())
    Console.WriteLine("Length: " & boat2.GetLength())
    Console.WriteLine("Manufacturer: " & boat2.GetManufacturer())
    Console.WriteLine("Year: " & boat2.GetYear())

End Sub
```

الشكل رقم (٨,٤). شفرة إجراء اختبار الصنف Boat المبدئي.

```
Begin Boat Test:
Boat No: MO1234
Length: 28
Manufacturer: Tartan
Year: 2001
Boat No: CA9876
Length: 32
Manufacturer: Catalina
Year: 2003
```

الشكل رقم (٨,٥). مخرجات إجراء اختبار الصنف Boat.

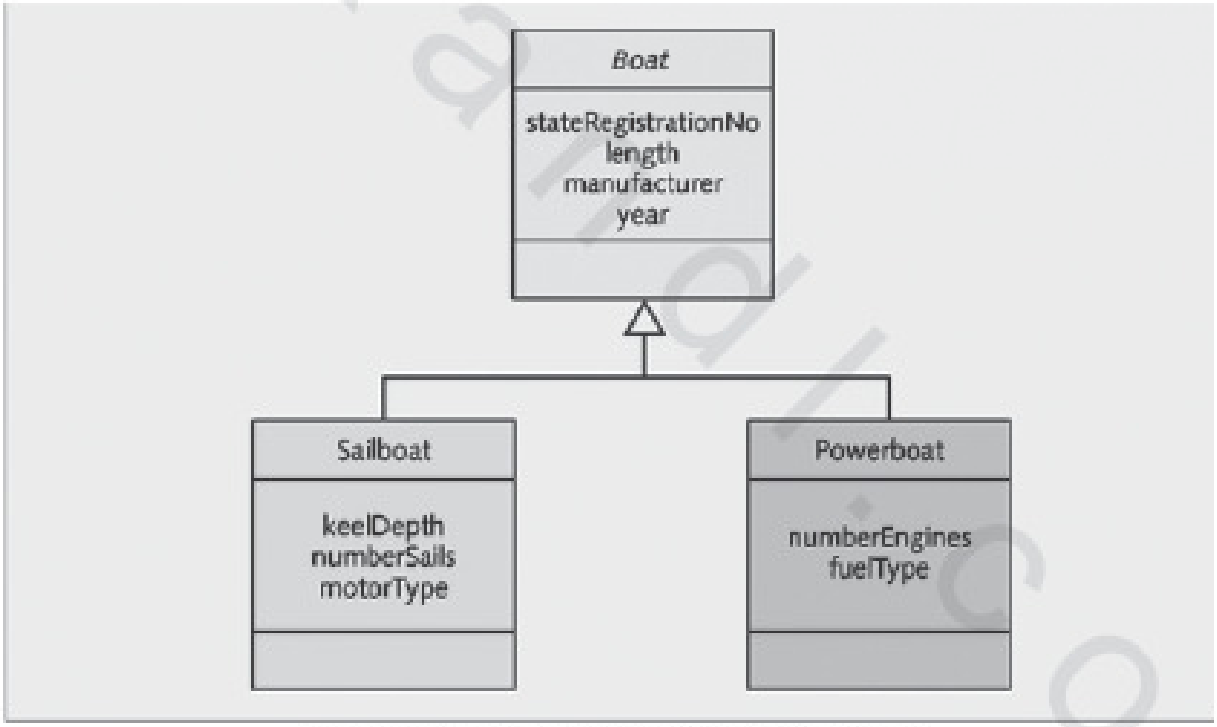


الشكل رقم (٨,٦). نموذج Sequence Diagram لإجراء اختبار الصنف Boat.

استخدام الكلمة Inherits لتعريف الصنف الفرعي Sailboat

Using the Inherits Keyword to Create the Sailboat Subclass

يوضح نموذج Class Diagram لشركة برادشو مارينا أنه توجد أنواع خاصة من المراكب التي تشترك في مجموعة من الصفات (صفات الصنف Boat) ويختص كل نوع منها بمجموعة أخرى من الصفات، فعلى سبيل المثال إن المركب الشراعي (Sailboat) يختص بصفة "عمق قاع المركب" (Keel Depth) والصفة "عدد الأشرعة" (Number of Sails) والصفة "نوع المحرك" (Motor Type) والتي يمكن أن تأخذ قيمة من ثلاث وهي: none أو inboard أو outboard. وعلى الجانب الآخر فإن المركب الآلي (Powerboat) لا يحتوي على أشرعة، ومعرفة عمق القاع غير ضروري. وعلى أي حال فإن المركب الآلي ربما يحتوي على العديد من المحركات (Number of Engines) وصفة "نوع الوقود" (Fuel Type) الذي يمكن أن يكون بنزيناً أو ديزلاً. يوضح الشكل رقم (٨,٧) شجرة التوارث (نطلق عليها أحياناً شجرة التعميم/التخصيص) للصنف Boat التي تعرض أنواع المراكب المختلفة.



الشكل رقم (٨,٧). شجرة التوارث (التعميم/التخصيص) لصنف المركب.

تذكر أن شجرة التعميم/التخصيص تعني أن الصنف الرئيس العام يحتوي على صفات وإجراءات مشتركة للأصناف الفرعية؛ لذلك فإن كائنات الأصناف الفرعية ترث الصفات والإجراءات المعرفة في الصنف الرئيس بالإضافة إلى الصفات والإجراءات المعرفة لديها. ويمثل الصنف Boat بما يحتويه من أربع صفات وثمانية إجراءات

وصول الصنف الرئيس، بينما يمثل كل من الصنف Sailboat والصنف Powerboat بما يحتويان من صفات وإجراءات خاصة بهما هما أصناف فرعية. وتذكر أيضاً أن رمز المثلث الذي يشير إلى الصنف Boat في الشكل رقم (٨.٧) يعني أن الصنف Boat هو الصنف الرئيس.

تستخدم الكلمة المحجوزة Inherits لإنشاء صنف فرعي من صنف رئيس حيث نكتب هذه الكلمة متبوعة باسم الصنف الرئيس بعد رأس الصنف الفرعي (في السطر الثاني للصنف الفرعي). فعلى سبيل المثال، لتطوير الصنف الفرعي Sailboat من الصنف الرئيس Boat نكتب الأوامر التالية:

```
Public Class Sailboat
      Inherits Boat
```

ثم نعرف الصفات والإجراءات الخاصة بالصنف الفرعي كما تعلمنا سابقاً، ولا نكرر الصفات والإجراءات الموروثة من الصنف الرئيس.

يحتوي الصنف Sailboat على ستة إجراءات وصول للصفات الثلاثة الإضافية حيث يوضح الشكل رقم (٨.٨) تعريف الصنف Sailboat كاملاً. سوف نعرف في هذا الكتاب كل صنف من أصناف مجال المشكلة داخل ملف vb. منفصل على الرغم من أن لغة VB .NET لا تتطلب ذلك حيث يمكن تعريف أكثر من صنف داخل ملف vb. واحد.

يمكنك تعريف أكثر من إجراء إنشاء داخل الصنف Sailboat بتوقيع مختلف (قوائم معاملات مختلفة)، ولكن إذا نظرنا إلى الشكل رقم (٨.٨) لوجدنا أنه يوجد إجراء إنشاء واحد والذي يستقبل سبعة معاملات لإسنادها لصفات كائنات الصنف Sailboat (أربع صفات موروثة من الصنف Boat وثلاث صفات معرفة داخل الصنف Sailboat).

```
'constructor (all seven attribute values as parameters)
Public Sub New(ByVal aStateRegistrationNo As String, _
 ByVal aLength As Double, _
 ByVal aManufacturer As String, ByVal aYear As Integer, _
 ByVal aKeelDepth As Double, ByVal aNumberSails As Integer, _
 ByVal aMotorType As String)
```

يستخدم إجراء إنشاء الصنف Sailboat الكلمة MyBase لاستدعاء إجراء الإنشاء New من الصنف الرئيس Boat وإجراء أربعة معاملات له، وذلك لإسنادها داخل الصفات الأربعة الموروثة من الصنف Boat (رقم التسجيل، والطول، والمصنِّع، وسنة الإنتاج). وبهذا الأسلوب الجيد من البرمجة يستطيع إجراء إنشاء الصنف Boat إنهاء أي معالجة مطلوبة لإسناد البيانات المستقبلة للصفات المعرفة لديه مثل التحقق من صحة البيانات المستقبلة قبل إسنادها.

```
'invoke superclass constructor passing 4 values
MyBase.New(aStateRegistrationNo, aLength, aManufacturer, aYear)
```

```
' Sailboat -- a subclass of Boat

Public Class Sailboat
   Inherits Boat

   'attributes in addition to the four
   'inherited from Boat
   Private keelDepth As Double
   Private numberSails As Integer
   Private motorType As String

   'constructor (all seven attribute values as parameters)
   Public Sub New(ByVal aStateRegistrationNo As String, _
      ByVal aLength As Double, _
      ByVal aManufacturer As String, ByVal aYear As Integer, _
      ByVal aKeelDepth As Double, ByVal aNumberSails As Integer, _
      ByVal aMotorType As String)
      'invoke superclass constructor passing 4 values
      MyBase.New(aStateRegistrationNo, aLength, aManufacturer, aYear)
      'set additional attribute values
      SetKeelDepth(aKeelDepth)
      SetNumberSails(aNumberSails)
      SetMotorType(aMotorType)
   End Sub

   'TellAboutSelf overrides but does not invoke superclass method
   Public Overrides Function TellAboutSelf() As String
      Return (GetStateRegistrationNo() & ", " & GetLength() & _
         ", " & _
      GetManufacturer() & ", " & GetYear() & ", " _
      & keelDepth & ", " & numberSails & ", " & motorType)
   End Function

   'Get accessor methods
   Public Function GetKeelDepth() As Double
      Return keelDepth
   End Function
   Public Function GetNumberSails() As Integer
      Return numberSails
   End Function
   Public Function GetMotorType() As String
      Return motorType
   End Function

   'Set accessor methods
   Public Sub SetKeelDepth(ByVal aKeelDepth As Double)
      keelDepth = aKeelDepth
   End Sub
   Public Sub SetNumberSails(ByVal aNumberSails As Integer)
      numberSails = aNumberSails
   End Sub
   Public Sub SetMotorType(ByVal aMotorType As String)
      motorType = aMotorType
   End Sub

End Class
```

الشكل رقم (٨,٨). تعريف الصنف Sailboat.

وبشكل عام يجب أن يتم استدعاء الأمر MyBase.New في بداية إجراء إنشاء الصنف الفرعي إلا إذا كان الصنف الرئيس لا يحتوي على إجراء إنشاء يستقبل معاملات، فعندئذ سيتم استدعاء إجراء الإنشاء الافتراضي الذي لا يستقبل معاملات بشكل تلقائي.

عندما ينتهي إجراء إنشاء الصنف Boat من عمله، ينتقل التحكم ثانية إلى إجراء إنشاء الصنف Sailboat ليستدعي إجراءات الوصول Setters المعرفة بداخله لإسناد قيم الصفات الثلاثة المعرفة لديه هكذا:

```
'set additional attribute values
SetKeelDepth(aKeelDepth)
SetNumberSails(aNumberSails)
SetMotorType(aMotorType)
```

عندما ينتهي تنفيذ إجراء الصنف Sailboat، عندئذ يتم إنشاء كائن من النوع Sailboat يحتوي على ١٤ إجراء وصول (بالإضافة إلى الإجراء TellAboutSelf) ويحتوي أيضاً على سبع قيم لصفاته السبعة. نستطيع الآن أن ننشئ برنامج اختبار الذي سيقوم بإنشاء كائن أو أكثر من الصنف Sailboat، ثم يستدعي أي إجراء من إجراءات الوصول الأربعة عشر دون أن يهتم هل هذا الإجراء معرف لدى الصنف Sailboat أو تم وراثته من الصنف Boat. ولا نهتم أيضاً أن نعرف شيئاً عن التركيب الداخلي لشجرة التوارث لأننا نتعامل مع كائن (وحدة متكاملة تحتوي على كل من الصفات والإجراءات).

في الحقيقة إن تقنية التوارث تعطي قوة للمبرمجين حيث لا تحتاج من المبرمج أن يعلم شيئاً عن التركيب الداخلي للصنف Boat لكي ينشئ منه صنفاً فرعياً (لا يتعامل مطلقاً مع شفرة المصدر)؛ ولذلك يمكن إنشاء صنف فرعي من صنف مترجم مكتوب بلغة مثل C# أو لغة J#. وهذا يتطلب فقط معرفة توقيع إجراء الإنشاء وأي إجراءات أخرى مطلوبة.

اختبار الصنف الفرعي Sailboat
Testing the Sailboat Subclass

يوضح الشكل رقم (٨.٩) أوامر اختبار الصنف Sailboat، كما يوضح الشكل رقم (٨.١٠) مخرجات هذه الأوامر. تنفذ هذه الأوامر استجابة للضغط على زر "Test Sailboat Subclass" الموجود في نموذج اختبار الصنف Boat. وتنشئ هذه الأوامر كائنان من الصنف Sailboat مستخدمة إجراء الإنشاء الذي يستقبل سبعة معاملات (رقم التسجيل، والطول، والمصنّع، وسنة الصنع، وعمق القاع، وعدد الأشرعة، ونوع الوقود). ثم يستخدم أوامر الوصول Getters للتحقق من أن المعاملات قد تم إسنادها للصفات بشكل صحيح، ولاحظ أن كائن الصنف

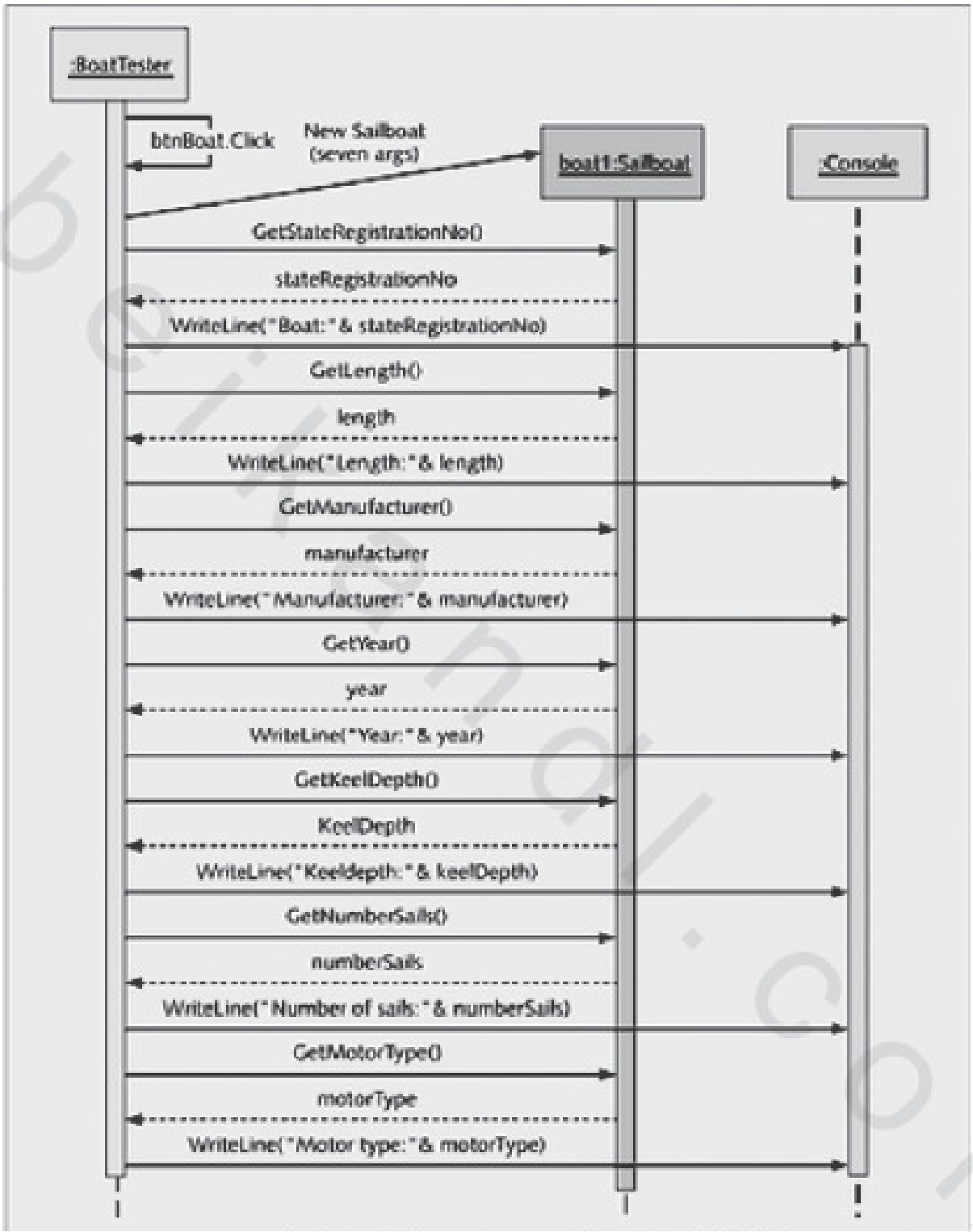
Sailboat قد ورث إجراءات الوصول لأربع صفات من الصنف Boat. ولاحظ أيضاً أنه لا يوجد أي مؤشر في أوامر الاختبار يوضح أن الصنف Sailboat هو صنف فرعي. يوضح الشكل رقم (٨,١١) نموذج Sequence Diagram الخاص بأوامر اختبار الصنف Sailboat.

```
Private Sub btnSailboat_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnSailboat.Click

   Console.WriteLine("Begin Sailboat Test:")

   'declare 2 Sailboat references
   Dim boat1 As Sailboat
   Dim boat2 As Sailboat

   'instantiate 2 Sailboats
   boat1 = New Sailboat("MO1234", 28, "Tartan", 2001, 4, 2, "outboard")
   boat2 = New Sailboat("CA9876", 32, "Catalina", 2003, 5, 3, "inboard")

   'invoke getter methods for boat1
   Console.WriteLine("Boat No: " & boat1.GetStateRegistrationNo())
   Console.WriteLine("Length: " & boat1.GetLength())
   Console.WriteLine("Manufacturer: " & boat1.GetManufacturer())
   Console.WriteLine("Year: " & boat1.GetYear())
   Console.WriteLine("Keel depth: " & boat1.GetKeelDepth())
   Console.WriteLine("Number of sails: " & boat1.GetNumberSails())
   Console.WriteLine("Motor type: " & boat1.GetMotorType())

   'invoke getter methods for boat2
   Console.WriteLine("Boat No: " & boat2.GetStateRegistrationNo())
   Console.WriteLine("Length: " & boat2.GetLength())
   Console.WriteLine("Manufacturer: " & boat2.GetManufacturer())
   Console.WriteLine("Year: " & boat2.GetYear())
   Console.WriteLine("Keel depth: " & boat2.GetKeelDepth())
   Console.WriteLine("Number of sails: " & boat2.GetNumberSails())
   Console.WriteLine("Motor type: " & boat2.GetMotorType())

End Sub
```

الشكل رقم (٨,٩). شفرة إجراء اختبار الصنف Sailboat.

```
Begin Sailboat Test:
Boat No: MO1234
Length: 28
Manufacturer: Tartan
Year: 2001
Keel depth: 4
Number of sails: 2
Motor type: outboard
Boat No: CA9876
Length: 32
Manufacturer: Catalina
Year: 2003
Keel depth: 5
Number of sails: 3
Motor type: inboard
```

الشكل رقم (٨,١٠). مخرجات إجراء اختبار الصنف Sailboat.



الشكل رقم (٨,١١). نموذج Sequence Diagram لإجراء اختبار الصنف Sailboat.

إضافة الصنف الفرعي الآخر Powerboat

Adding a Second Subclass—Powerboat

الصنف Powerboat هو صنف فرعي آخر للصنف Boat حيث يمكن إضافة الصنف Powerboat دون أن يؤثر ذلك على كل من الصنف Boat أو الصنف Sailboat. يوضح الشكل رقم (٨,١٢) شفرة أوامر الصنف Powerboat.

```
' Powerboat -- a subclass of Boat

Public Class Powerboat
   Inherits Boat

   'attributes in addition to the four
   'inherited from Boat
   Private numberEngines As Integer
   Private fuelType As String

   'constructor (all six attribute values as parameters)
   Public Sub New(ByVal aStateRegistrationNo As String, _
    ByVal aLength As Double, _
    ByVal aManufacturer As String, ByVal aYear As Integer, _
    ByVal aNumberEngines As Integer, ByVal aFuelType As String)
      'invoke superclass constructor passing four values
      MyBase.New(aStateRegistrationNo, aLength, aManufacturer, aYear)
      'set additional attribute values
      SetNumberEngines(aNumberEngines)
      SetFuelType(aFuelType)
   End Sub
   'TellAboutSelf method overrides then invokes superclass method
   'then concatenates two additional attribute values
   Public Overrides Function TellAboutSelf() As String
      Return (MyBase.TellAboutSelf() & ", " _
      & numberEngines & ", " _
      & fuelType)
   End Function

   'Get accessor methods
   Public Function GetNumberEngines() As Integer
      Return numberEngines
   End Function
   Public Function GetFuelType() As String
      Return fuelType
   End Function

   'Set accessor methods
   Public Sub SetNumberEngines(ByVal aNumberEngines As Integer)
      numberEngines = aNumberEngines
   End Sub
   Public Sub SetFuelType(ByVal aFuelType As String)
      fuelType = aFuelType
   End Sub

End Class
```

الشكل رقم (٨,١٢). تعريف الصنف Powerboat.

يعرف الصنف Powerboat صفاته numberEngines و fuelType ومن ثم سوف يستقبل إجراء الإنشاء الخاص به ستة معاملات (أربعة معاملات للصفات الأربعة الموروثة من الصنف Boat ومعاملان لهاتين الصفتين). سوف يستدعي إجراء إنشاء الصنف Powerboat إجراء إنشاء الصنف Boat (كما فعل إجراء إنشاء الصنف Sailboat) مستخدماً الأمر MyBase.New ويمر إليه أربعة معاملات، ثم يستدعي إجراءات الوصول Setters المعرفة داخل الصنف Powerboat لإسناد الصفتين المعرفتين داخل الصنف Powerboat.

```
'constructor (all six attribute values as parameters)
Public Sub New(ByVal aStateRegistrationNo As String, _
 ByVal aLength As Double, _
 ByVal aManufacturer As String, ByVal aYear As Integer, _
 ByVal aNumberEngines As Integer, ByVal aFuelType As String)
    'invoke superclass constructor passing four values
    MyBase.New(aStateRegistrationNo, aLength, aManufacturer, aYear)
    'set additional attribute values
    SetNumberEngines(aNumberEngines)
    SetFuelType(aFuelType)
End Sub
```

والآن بعدما عرفت الأصناف الفرعية Sailboat و Powerboat من الصنف الرئيس Boat، يمكنك استخدامها كيف تشاء، فمثلاً يمكن كتابة برنامج اختبار لتعرُّف كائنات من الصنفين، ثم تستدعي الأوامر اللازمة لاسترجاع معلوماتها ثانية. يوضح الشكل رقم (٨.١٣) أوامر الاختبار اللازمة لإنشاء كائنات من الصنفين وذلك داخل الإجراء btnSailAndPower_Click، ويوضح الشكل رقم (٨.١٤) مخرجات هذه الأوامر، كما يوضح الشكل رقم (٨.١٥) جزءاً من نموذج Sequence Diagram الذي يوضح التفاعل بين الكائنات لتنفيذ تلك الأوامر. سوف نناقش الإجراء TellAboutSelf لاحقاً في هذه الفصل.

وفي حالة إنشاء كائنات من الصنف Powerboat أو الصنف Sailboat، فيمكنك استدعاء إجراءات الوصول Getters الأربعة الموروثة من الصنف Boat من أي منهما، وعلى أي حال فإن كائنات الصنف Sailboat تستجيب لثلاثة إجراءات وصول زيادة، بينما كائنات الصنف Powerboat تستجيب لإجراءين وصول زيادة ومختلفة.

التعرف على الصنف المجرد والصنف النهائي
Understanding Abstract and Final Classes

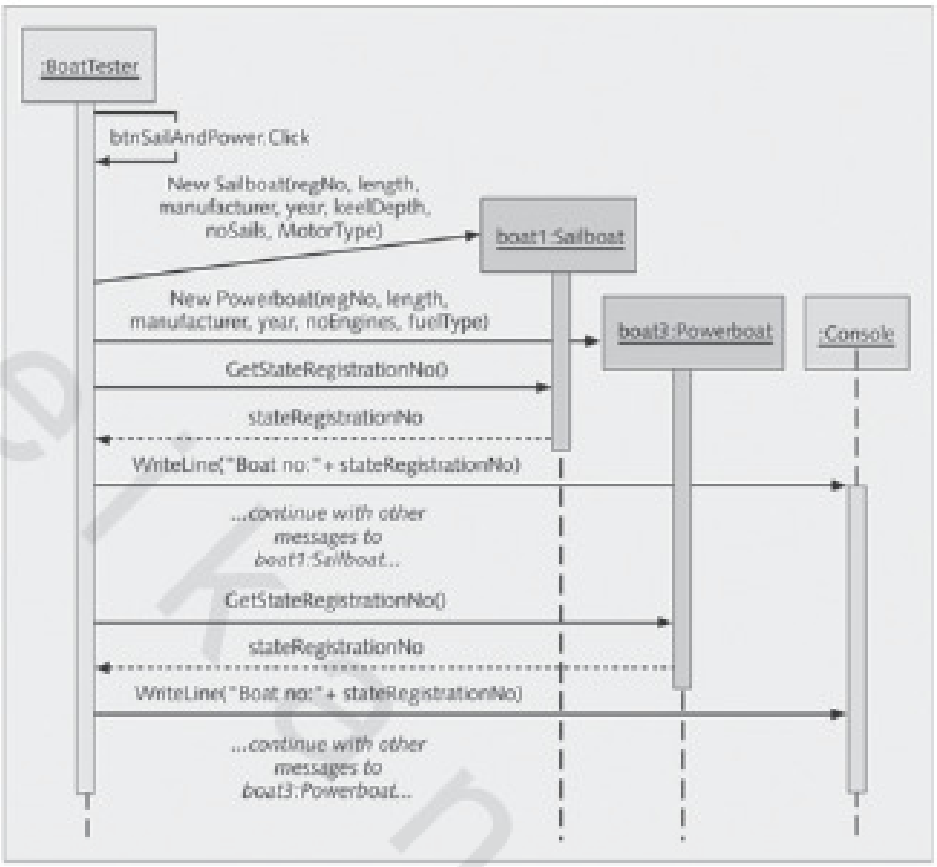
لقد عرفنا سابقاً كلاً من الصنف الرئيس Boat والصنف الفرعي Sailboat والصنف الفرعي Powerboat ثم أنشأنا كائنات منها جميعاً؛ ولذلك يطلق عليها اسم الأصناف المعدة للاستخدام (Concrete Class) وهي الأصناف التي يتم تعريفها ويمكن استخدامها في إنشاء كائنات، كما يمكن أيضاً استخدامها كأصناف رئيسة لتعريف أصناف فرعية منها.

```
Private Sub btnSailAndPower_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnSailAndPower.Click

    Console.WriteLine("Begin Sail and Powerboat Test:")

    'declare 2 Sailboat and 2 Powerboat references
    Dim boat1 As Sailboat
    Dim boat2 As Sailboat
    Dim boat3 As Powerboat
    Dim boat4 As Powerboat

    'instantiate 2 Sailboats and 2 Powerboats
    boat1 = New Sailboat("MD1234", 28, "Tartan", 2001, 4, 2, "outboard")
    boat2 = New Sailboat("CA9876", 32, "Catalina", 2003, 5, 3, "inboard")
    boat3 = New Powerboat("GA4567", 34, "Bayliner", 2002, 2, "gas")
    boat4 = New Powerboat("MD5678", 24, "Searay", 2003, 2, "gas")

    'invoke getter methods for Sailboat boat1 (7 attributes)
    Console.WriteLine("Boat No: " & boat1.GetStateRegistrationNo())
    Console.WriteLine("Length: " & boat1.GetLength())
    Console.WriteLine("Manufacturer: " & boat1.GetManufacturer())
    Console.WriteLine("Year: " & boat1.GetYear())
    Console.WriteLine("Keel depth: " & boat1.GetKeelDepth())
    Console.WriteLine("Number of sails: " & boat1.GetNumberSails())
    Console.WriteLine("Motor type: " & boat1.GetMotorType())

    'invoke getter methods for Sailboat boat2 (7 attributes)
    Console.WriteLine("Boat No: " & boat2.GetStateRegistrationNo())
    Console.WriteLine("Length: " & boat2.GetLength())
    Console.WriteLine("Manufacturer: " & boat2.GetManufacturer())
    Console.WriteLine("Year: " & boat2.GetYear())
    Console.WriteLine("Keel depth: " & boat2.GetKeelDepth())
    Console.WriteLine("Number of sails: " & boat2.GetNumberSails())
    Console.WriteLine("Motor type: " & boat2.GetMotorType())

    'invoke getter methods for Powerboat boat3 (6 attributes)
    Console.WriteLine("Boat No: " & boat3.GetStateRegistrationNo())
    Console.WriteLine("Length: " & boat3.GetLength())
    Console.WriteLine("Manufacturer: " & boat3.GetManufacturer())
    Console.WriteLine("Year: " & boat3.GetYear())
    Console.WriteLine("Number engines: " & boat3.GetNumberEngines())
    Console.WriteLine("Fuel type: " & boat3.GetFuelType())

    'invoke getter methods for Powerboat boat4 (6 attributes)
    Console.WriteLine("Boat No: " & boat4.GetStateRegistrationNo())
    Console.WriteLine("Length: " & boat4.GetLength())
    Console.WriteLine("Manufacturer: " & boat4.GetManufacturer())
    Console.WriteLine("Year: " & boat4.GetYear())
    Console.WriteLine("Number engines: " & boat4.GetNumberEngines())
    Console.WriteLine("Fuel type: " & boat4.GetFuelType())

End Sub
```

الشكل رقم (٨,١٣). شفرة إجراء اختبار لكل من الصنف Powerboat و Sailboat.

```
Begin Sail and Powerboat Test:
Boat No: MO1234
Length: 28
Manufacturer: Tartan
Year: 2001
Keel depth: 4
Number of sails: 2
Motor type: outboard
Boat No: CA9876
Length: 32
Manufacturer: Catalina
Year: 2003
Keel depth: 5
Number of sails: 3
Motor type: inboard
Boat No: GA4567
Length: 34
Manufacturer: Bayliner
Year: 2002
Number engines: 2
Fuel type: gas
Boat No: MO5678
Length: 24
Manufacturer: Searay
Year: 2003
Number engines: 2
Fuel type: gas
```

الشكل رقم (٨,١٤)- مخرجات إجراء اختبار لكل من الصنف Powerboat و Sailboat.



الشكل رقم (٨,١٥). جزء من نموذج Sequence Diagram لإجراء اختبار الصنف Sailboat و Powerboat.

استخدام الكلمة MustInherit
Using the MustInherit Keyword

في بعض الأحيان تقتضي الحاجة أن نعرِّف صنفاً بغرض الوراثة فقط، أي بغرض أن يحتوي على كل من الصفات والإجراءات المشتركة لأصنافه الفرعية؛ ولذلك لا يمكن أن ننشئ كائناً من هذا الصنف لأنه سيكون كائناً ناقصاً. وفي هذه الحالة نطلق على هذا الصنف اسم الصنف المجرد (Abstract Class) وهو الصنف الذي يمكن توريثه ولكن لا يمكن إنشاء كائن منه. وإذا رجعنا إلى المراكب التي يمكن تأجيرها لعملاء شركة برادشو مارينا نجد أنها نوعان فقط (المركب الشراعي Sailboat، والمركب الآلي Powerboat)؛ لذلك لا توجد حاجة لتعريف كائنات من الصنف Boat إلا لاختباره فقط، ولهذا سيصبح صنفاً مجرداً. يكتب الصنف المجرد في نموذج Class Diagram بحروف مائلة.

تقدم لغة VB .NET الكلمة المحجوزة MustInherit لتعريف صنف مجرد، ولتحويل الصنف Boat إلى صنف مجرد تستخدم الكلمة MustInherit في رأس الصنف هكذا:

```
Public MustInherit Class Boat
```

يمكن للصنف الفرعي Sailboat والصنف الفرعي Powerboat أن يرثا الصنف Boat دون تغيير، ولكن لا يمكن تعريف أوامر اختبار لتعريف كائن من الصنف Boat. فعندئذ سيعطي مترجم لغة VB .NET خطئاً يقول أنه لا يمكن استخدام الكلمة New مع صنف معرف بالكلمة MustInherit؛ ولذلك لا تحاول إنشاء كائن من صف مجرد، ولكن يمكنك فقط إنشاء صنف فرعي منه ثم إنشاء كائن من هذا الصنف الفرعي. إن الأصناف المجردة مفيدة جداً لتعريف الصفات والإجراءات المشتركة بين أصنافها الفرعية وبهذا يتحقق مفهوم "إعادة الاستخدام" (Reusability) لأن الأصناف الفرعية تعيد استخدام الصفات والإجراءات المعرفة داخل الصنف المجرد. ولكن تذكر أنه لا يشترط أن يكون الصنف الرئيس (Super Class) صنفاً مجرداً، فعلى سبيل المثال إن الصنف Slip هو صنف رئيس ومعرف منه الصنف الفرعي CoveredSlip حيث تريد شركة برادشو مارينا إنشاء كائنات من كل من الصنف Slip والصنف الفرعي CoveredSlip.

استخدام الكلمة NotInheritable
Using the NotInheritable Keyword

توضح العملية السابقة أن عملية إنشاء أصناف فرعية من أصناف رئيسة عملية سهلة ولا تتطلب حتى معرفة شفرة أوامر الصنف الرئيس، ولكن في بعض الأحيان يتطلب الأمر إنشاء صنف لا يمكن توريثه، فعلى سبيل المثال افترض في نظام الرواتب أنه يوجد صنف اسمه Paycheck يحتوي على إجراء اسمه Pay Amount لدفع قيمة الرواتب. فإذا وفرنا إمكانية توريث الصنف Paycheck للآخرين، فعندئذ يمكن تعريف صنف فرعي منه وإعادة تعريف الإجراء Pay Amount لتغيير طريقة حساب الرواتب؛ ولذلك من أجل الأمان نحتاج في بعض الأحيان ألا نسمح لصنف ما أن يتم توريثه بواسطة الآخرين.

تستخدم الكلمة NotInheritable لتوصيف صنف لا يمكن توريثه، فعلى سبيل المثال إذا أرادت شركة برادشو مارينا ألا يتم توريث الصنف Powerboat، نكتب الأوامر التالية:

```
Public NotInheritable Class Powerboat
    Inherits Boat
```

إعادة تعريف إجراء الصنف الرئيس
Overriding a Superclass Method

إن إمكانية إعادة تعريف (Overriding) إجراء موروث من الصنف الرئيس داخل الصنف الفرعي تعطي قوة هائلة للبرمجة المعتمدة على الكائنات، حيث يتم استدعاء الإجراء المعرف داخل الصنف الفرعي بدلاً من الإجراء الموروث من الصنف الرئيس ولو كان الإجراءان لهما التوقيع نفسه. وتذكر أن توقيع الإجراء (Method Signature)

يشمل على اسم الإجراء ونوع البيان العائد وقائمة المعاملات المستقبلة. فعلى سبيل المثال يمثل الأمر التالي توقيع الإجراء SetNumberSails الذي يستقبل معامل Integer والمعرف داخل الصنف SailBoat.

```
Public Sub SetNumberSails(ByVal aNumberSails As Integer)
```

إن مفهوم إعادة تعريف الإجراءات (Method Overriding) يسمح بتغيير سلوك الصنف الرئيس حيث يسمح للصنف الفرعي بإعادة تعريف إجراءات الصنف الرئيس، ويجب عليك أن تلاحظ أن هذا المفهوم يختلف عن مفهوم تعدد استخدام الإجراءات (Overloading) الذي ناقشناه في الفصل السابق والذي يعني تعريف إجراء ما داخل صنف ما بنفس الاسم ولكن بتوقيع مختلف، بينما مفهوم إعادة تعريف الإجراءات فيعني تعريف الإجراء نفسه بنفس التوقيع في كل من الصنف الرئيس والصنف الفرعي.

إعادة تعريف الإجراء TellAboutSelf الخاص بالصنف Boat
Overriding the Boat TellAboutSelf Method

نعرض مفهوم إعادة التعريف مستخدمين الإجراء TellAboutSelf لكل من الصنف Boat والصنف Sailboat والصنف Powerboat. لقد طورنا هذا الإجراء سابقاً داخل الصنف Slip في الفصل السابق. ويمكن تعريف هذا الإجراء داخل الأصناف لتوفير إمكانية للكائنات المعرفة منها أن تسترجع معلومات كاملة عن صفاتها (مشتملاً الصنف Boat). وإذا أردت أن تعطي إمكانية إعادة تعريف الإجراء TellAboutSelf داخل الصنف الفرعي Sailboat فيجب إضافة الكلمة Overridable مع رأس الإجراء TellAboutSelf داخل الصنف الرئيس Boat؛ وبذلك تسمح للصنف Sailboat أن يعبر عن نفسه بالطريقة التي يراها مناسبة. توضح شفرة الأوامر التالية تعريف الإجراء TellAboutSelf المعرف داخل الصنف Boat (وكما هو واضح أيضاً في الشكل رقم ٨.٢).

```
'TellAboutSelf method
Public Overridable Function TellAboutSelf() As String
    Return (stateRegistrationNo & ", " & length & ", " & _
      manufacturer & ", " & year)
End Function
```

عندما تختبر الصنف Boat كصنف معد للاستخدام (Concrete Class) فيمكنك تعريف كائن واستدعاء الإجراء TellAboutSelf منه والذي سيعيد معلومات حول صفاته الأربعة فقط (رقم التسجيل، والطول، والمصنِّع، وسنة الصنع). وفي حالة عدم إعادة تعريف الإجراء داخل الصنف Sailboat وتم استدعاء هذا الإجراء من كائن

الصنف Sailboat فسوف يتم استدعاء الإجراء TellAboutSelf المعرف في الصنف Boat والذي يعيد معلومات حول هذه الصفات الأربعة فقط. والسؤال هنا هو كيف يتم تعريف المعلومات الإضافية لكائن الصنف Sailboat (الصفات الثلاثة الإضافية)؟

لحل هذه المشكلة يجب إعادة تعريف الإجراء TellAboutSelf داخل الصنف Sailboat لإضافة معلومات الصفات الثلاثة إلى المعلومات السابقة مستخدمين الكلمة المحجوزة Overrides بدلاً من كلمة Overridable. وعندئذ سيتم استدعاء الإجراء TellAboutSelf المعرف داخل الصنف Sailboat (كما هو واضح مسبقاً في الشكل رقم ٨.٨).

```
'TellAboutSelf overrides but does not invoke superclass method
Public Overrides Function TellAboutSelf() As String
    Return (GetStateRegistrationNo() & ", " & GetLength() & ", " & _
    GetManufacturer() & ", " & GetYear() & ", " _
    & keelDepth & ", " & numberSails & ", " & motorType)
End Function
```

يستدعي هذا الإجراء إجراءات الوصول Getters الأربعة الموروثة من الصنف Boat لاسترجاع قيم الصفات الأربعة المعرفة داخل الصنف Boat ثم يستدعي إجراءات الوصول Getters الثلاثة المعرفة داخل الصنف Sailboat لاسترجاع قيم صفاته الإضافية الثلاثة.

إعادة تعريف إجراء الصنف الرئيس واستدعاؤه
Overriding and Invoking a Superclass Method

في بعض الأحيان نحتاج أن نعيد تعريف إجراء ما (معرف في صنف رئيس) داخل صنف فرعي لنزيد ما يقوم هذا الإجراء بعمله. فعلى سبيل المثال إن الإجراء TellAboutSelf المعرف داخل الصنف الرئيس Boat يحتوي على أوامر تعيد قيم الصفات الأربعة المشتركة بين جميع أنواع المراكب، وكما هو ملاحظ فإن الإجراء TellAboutSelf السابق المعرف داخل الصنف الفرعي Sailboat أعاد تعريف هذه الأوامر، ثم احتوى على عدد من الأوامر الأخرى الإضافية.

سوف نستخدم طريقة أخرى لتعريف الإجراء TellAboutSelf داخل الصنف Powerboat (غير الأسلوب المتبع في تعريف إجراء TellAboutSelf الخاص بالصنف Sailboat)، حيث سيتم استدعاء الإجراء TellAboutSelf من الصنف الرئيس Boat مستخدماً الكلمة المحجوزة MyBase. سنقوم بكتابة الأمر MyBase.TellAboutSelf() داخل الإجراء TellAboutSelf الخاص بالصنف Powerboat لاستدعاء الإجراء TellAboutSelf من الصنف Boat وتنفيذ أوامره أولاً وإعادة نص يحتوي على قيم الصفات الأربعة ثم تعيد بقية أوامر الإجراء TellAboutSelf الخاص بالصنف Powerboat ولصق قيمة الصنفين المعرفين داخل الصنف Powerboat وإعادة نص يحتوي على قيمة الصفات

الستة. توضح الأوامر التالية تعريف الإجراء TellAboutSelf الخاص بالصنف Powerboat (كما هو واضح مسبقاً في الشكل رقم ٨.١٢).

```
'TellAboutSelf method overrides then invokes superclass method
'then concatenates two additional attribute values
Public Overrides Function TellAboutSelf() As String
    Return (MyBase.TellAboutSelf() & ", " _
    & numberEnginas & ", " _
    & fuelType)
End Function
```

اختبار أسلوبيْ إعادة تعريف الإجراءات

Testing Two Method-Overriding Approaches

يعرض الإجراء الأخير لنموذج اختبار الصنف Boat أوامر اختبار الإجراء TellAboutSelf لكل من الصنف الفرعي Sailboat والصنف الفرعي Powerboat كما هو واضح في الشكل رقم (٨.١٦)، حيث ينشئ الإجراء كائنان من كل من الصنف Powerboat والصنف Sailboat، ثم يستدعي الإجراء TellAboutSelf من كل كائن على حده.

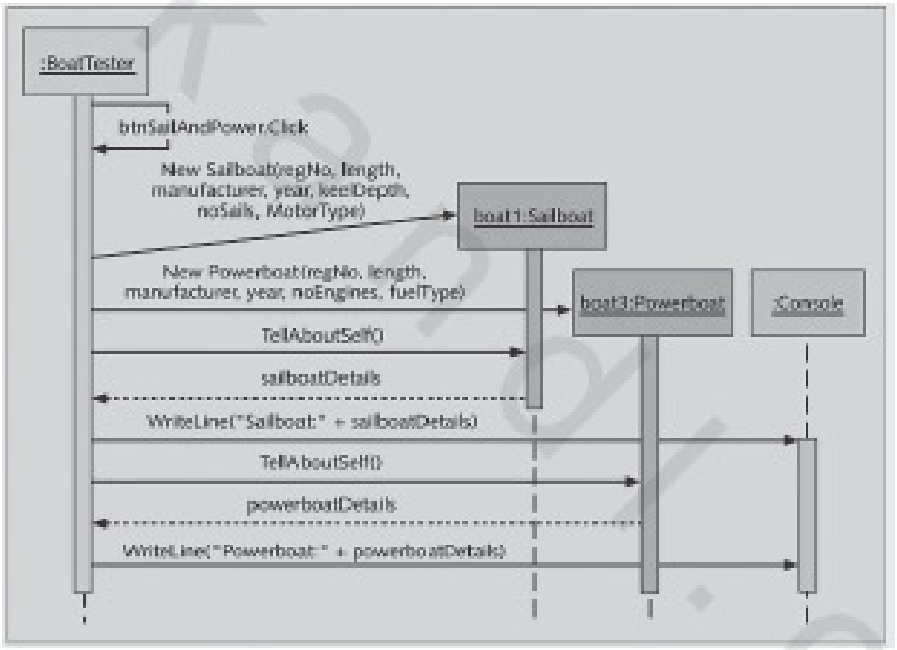
```
Private Sub btnTellAboutSelf_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnTellAboutSelf.Click

   Console.WriteLine("Begin Sail and Powerboat TellAboutSelf Test:")

   'declare 2 Sailboat and 2 Powerboat references
   Dim boat1 As Sailboat
   Dim boat2 As Sailboat
   Dim boat3 As Powerboat
   Dim boat4 As Powerboat
   'instantiate 2 Sailboats and 2 Powerboats
   boat1 = New Sailboat("MO1234", 28, "Tartan", 2001, 4, 2, "outboard")
   boat2 = New Sailboat("CA9876", 32, "Catalina", 2003, 5, 3, "inboard")
   boat3 = New Powerboat("GA4567", 34, "Bayliner", 2002, 2, "gas")
   boat4 = New Powerboat("MO5678", 24, "Searay", 2003, 2, "gas")

   'invoke TellAboutSelf methods for Sailboats and Powerboats
   Console.WriteLine("Sailboat: " & boat1.TellAboutSelf())
   Console.WriteLine("Sailboat: " & boat2.TellAboutSelf())
   Console.WriteLine("Powerboat: " & boat3.TellAboutSelf())
   Console.WriteLine("Powerboat: " & boat4.TellAboutSelf())

End Sub
```

الشكل رقم (٨.١٦). شفرة اختبار الإجراء TellAboutSelf لكل من الصنف Sailboat والصنف Powerboat.

يوضح الشكل رقم (٨,١٧) مخرجات هذا الإجراء مشتملاً على قيم الصفات السبعة لكائنات الصنف Sailboat وقيم الصفات الستة لكائنات الصنف Powerboat، كما يوضح الشكل رقم (٨,١٨) نموذج Sequence Diagram الجزئي لهذا الإجراء. لاحظ كيف أصبح التفاعل سهلاً بعدما أصبح كل كائن قادر على التعبير عن نفسه.

```
Begin Sail and Powerboat TellAboutSelf Test:
Sailboat: MO1234, 28, Tartan, 2001, 4, 2, outboard
Sailboat: CA9876, 32, Catalina, 2003, 5, 3, inboard
Powerboat: GA4567, 34, Bayliner, 2002, 2, gas
Powerboat: MO5678, 24, Searay, 2003, 2, gas
```

الشكل رقم (٨,١٧). مخرجات اختبار الإجراء TellAboutSelf لكل من الصنف Sailboat والصنف Powerboat.



الشكل رقم (٨,١٨). نموذج Sequence Diagram لإجراء اختبار TellAboutSelf للصنف Sailboat والصنف Powerboat.

إعادة التعريف وتعدد تعريف الإجراءات والرابط المتغير
Overriding, Polymorphism, and Dynamic Binding

لقد لاحظنا في المثال السابق أن مفهوم إعادة تعريف الإجراءات (Overriding) يمثل أحد أساليب تطبيق مفهوم تعدد تعريف الإجراءات (Polymorphism) الذي يعني وقوع سلوك مختلف من كائنات لأصناف مختلفة عند استقبال الرسالة نفسها، حيث لاحظنا أن كائنات الأصناف الفرعية لديها طريقتها الخاصة للتعبير عن نفسها وذلك

بإعادة تعريف الإجراء TellAboutSelf الموروث عن الصنف الأب. ومن ثم سوف يصبح نظام المعالجة أكثر بساطة لأن كل كائن لدية التعريف الخاص به (سواء استفاد من الإجراء الموروث أم لا).

افترض أن لديك نظام رواتب لشركة ما وأن هذا النظام يحتوي على صنف Employee الذي يحتوي على الإجراء CalculatePay المسؤول عن حساب راتب الموظف، وتوجد أيضاً أصناف فرعية ترث هذا الصنف التي تمثل أنواع الموظفين المختلفة مثل الصنف HourlyEmployee (موظف يعمل بنظام الساعات) والصنف SalaryEmployee (موظف الراتب الكامل) والصنف ManagementEmployee (موظف يعمل في الإدارة)، فيجب أن يتم إعادة تعريف إجراءات حساب الراتب CalculatePay في كل من هذه الأصناف لحساب راتب كل موظف طبقاً لنظام توظيفه، ومن ثم عند إجراء عملية حساب الرواتب سوف يسأل النظام كائنات الموظفين لحساب الراتب الخاص بهم (إرسال رسالة CalculatePay لكل كائن) دون الحاجة إلى معرفة نوع كل كائن لكل موظف، وبذلك فإن مسؤولية حساب الراتب سوف توكل لكل كائن على حده مما يؤدي إلى تبسيط المعالجة في النظام، كما أن عملية صيانة النظام سوف تصبح أكثر بساطة، فلو افترضنا إضافة نوع موظف جديد إلى النظام يتطلب ذلك إضافة صنف جديد دون التأثير على الأصناف الأخرى، أو افترض حدوث تغيير في طريقة حساب راتب نوع موظف ما، عند ذلك سوف يتم التعديل من إجراء هذا الصنف فقط دون التأثير على الأصناف الأخرى.

تستخدم لغة VB .NET طريقة "الربط المتغير" (Dynamic Binding) لكي تميز الإجراء المطلوب تنفيذه أثناء تشغيل النظام وذلك عند وجود أكثر من تعريف لهذا الإجراء داخل أصناف تربطها علاقة توارث، حيث إن اتخاذ قرار اختيار الإجراء أثناء وقت التشغيل يعطي مرونة عند إضافة أصناف فرعية جديدة والتي ستعيد تعريف إجراءات الأصناف الرئيسة. افترض أن شركة برادشو مارينا أضافت نوعاً جديداً من المراكب الآلية (مركب شخصي)، فسوف يتطلب ذلك إضافة صنف فرعي جديد Personal Watercraft مع إعادة تعريف الإجراء TellAboutSelf. فعندئذ يستطيع أي كائن أن يتفاعل مع كائنات الصنف Personal Watercraft بإرسال الرسالة TellAboutSelf.

الفرق بين درجة الوصول الخاص والمحمي
Understanding Private versus Protected Access

اتسمت صفات الصنف Boat بدرجة الوصول الخاص (مستخدمين كلمة الوصول Private) والذي يعني أنه ليس من المسموح لأي كائن أن يتعامل معها مباشرة، ولكي يتمكن أي كائن من تغيير أو الحصول على قيمها فيجب أن يستدعى إجراءات الوصول Getters/Setters المعرفة داخل الصنف Boat، وهذا ما أطلق عليه مفهوم الكبسلة (Encapsulation) أو إخفاء البيانات (Information Hiding)، وبمعنى آخر فإن تعريف الصفات من النوع الخاص يؤدي إلى حماية تكامل البيانات حيث لا يسمح بتغيير قيمها إلا عن طريق إجراءات الوصول Setters التي

تحتوي على الأوامر المنطقية اللازمة للتحقق من صحة القيم قبل إسنادها إلى الصفات، وبالمثل لا نسمح بالحصول على قيم هذه الصفات إلا عن طريق إجراءات الوصول Getters.

كما أن درجة الوصول الخاص (Private) تمنع كائن الصنف الفرعي أن يتعامل مباشرة مع الصفات الموروثة من الصنف الرئيس، ومن ثم يجب استدعاء إجراءات الوصول Getters للحصول على قيم هذه الصفات. لقد لاحظنا أن الإجراء TellAboutSelf المعرف داخل الصنف الفرعي Sailboat يستدعي إجراءات الوصول Getters للحصول على قيم الصفات الموروثة كما هو واضح في الأوامر التالية:

```
'TellAboutSelf overrides but does not invoke superclass method
'invokes getter methods of Boat
Public Overrides Function TellAboutSelf() As String
    Return (GetStateRegistrationNo() & ", " & GetLength() & ", " & _
    GetManufacturer() & ", " & GetYear() & ", " _
    & keelDepth & ", " & numberSails & ", " & motorType)
End Function
```

يمكنك تعريف صفات الصنف الرئيس من النوع المحمي (Protected) وذلك للسماح للصنف الفرعي أن يتعامل معها مباشرة دون الحاجة إلى استدعاء إجراءات الوصول، فعلى سبيل المثال يمكن إعادة تعريف صفات الصنف Boat من النوع المحمي مستخدماً الكلمة المحجوزة (Protected) هكذا.

```
Public Class Boat

    'attributes
    Protected stateRegistrationNo As String
    Protected length As Single
    Protected manufacturer As String
    Protected year As Integer
```

وبذلك يمكن إعادة تعريف الإجراء TellAboutSelf المعرف في الصنف Sailboat للتعامل مباشرة مع قيم الصفات هكذا:

```
'TellAboutSelf overrides but does not invoke superclass method
'directly accesses Boat protected attribute values
Public Overrides Function TellAboutSelf() As String
    Return (stateRegistrationNo & ", " & length & ", " & _
    manufacturer & ", " & year & ", " _
    & keelDepth & ", " & numberSails & ", " & motorType)
End Function
```

لاحظ أن أول أربع صفات داخل الإجراء TellAboutSelf معرفة داخل الصنف Boat من النوع المحمي (Protected) والصفات الثلاثة الأخرى ظلت معرفة داخل الصنف Sailboat من النوع الخاص (Private)، كما أن برنامج الاختبار الذي ينشئ كائناً من الصنف Sailboat سيظل كما هو بدون تغيير.

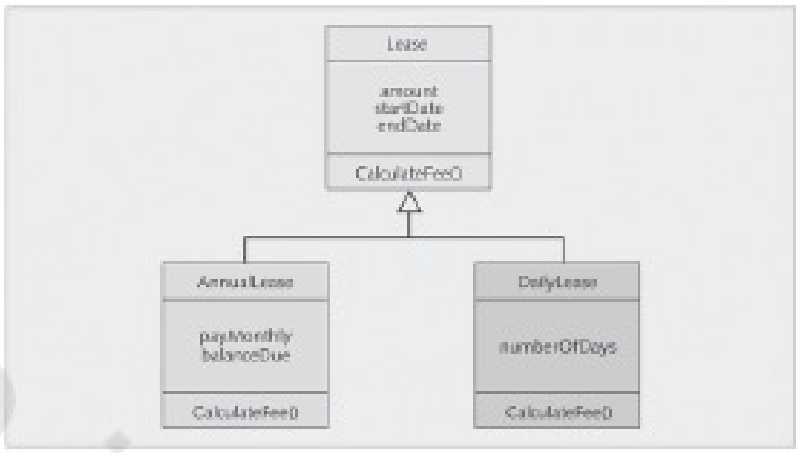
بالرغم من أنه يمكنك تعريف الصفات من النوع العام مستخدماً الكلمة المحجوزة (Public) مما يؤدي إلى السماح لأي كائن أن يتعامل معها مباشرة بالتغيير أو الاسترجاع من أي كائن، ولكن عزيزي القارئ احذر أن تستخدم هذا الأسلوب السيئ والذي يؤدي إلى الاعتداء على أهم مفهوم تقدمه البرمجة المعتمدة على الكائنات وهو مفهوم إخفاء المعلومات، كما يمكنك أيضاً تعريف الصفات من النوع الصديق (مستخدماً الكلمة المحجوزة Friend أو الكلمات Protected Friend معاً) لتغيير التعامل المباشر مع الصفات داخل البرنامج الخاص بك فقط، ولكن يجب عليك أيضاً عدم اتباع هذا الأسلوب من البرمجة.

إن هذه الكلمات المحجوزة لا تستخدم مع المتغيرات المحلية (Local Variables) لأنها ببساطة لا يمكن التعامل معها خارج حدود الإجراء المعرفة داخله، وتبقى في الذاكرة أثناء تنفيذه فقط. بينما يمكن استخدام تلك الكلمات المحجوزة (Private أو Protected أو Friend) لتعريف الإجراءات حيث إن الكلمة Private تتيح للإجراء أن يتم استدعاؤه داخل الصنف المعرف داخله فقط، ولا تسمح حتى لإجراءات الصنف الفرعي أن تستدعي هذا الإجراء، ولكن الإجراءات المعرفة من النوع المحمي (مستخدمة الكلمة Protected) يمكن استدعاؤها داخل أوامر الصنف الفرعي.

تعريف الأصناف الفرعية للصنف Lease والإجراءات المجردة
Introducing the Lease Subclasses and Abstract Methods

أوضحنا في الفصل الخامس أن نموذج Class Diagram الخاص بنظام شركة برادشو مارينا يحتوي على ثلاث شجرات توارث (شجرة الصنف Boat وشجرة الصنف Slip وشجرة الصنف Lease)، ولقد تعلمنا المفاهيم الأساسية للتوارث (Inheritance) مع الأصناف الفرعية للصنف Boat سابقاً. وسوف تكمل عرض مفاهيم أخرى عن التوارث في هذا الفصل مستعينين بالأصناف الفرعية للصنف Lease كما تتميز شجرة توارث الصنف Lease بأن أصنافها تحتوي على متغيرات إشارة تشير إلى كائنات أصناف مجال المشكلة بدلاً من احتوائها على متغيرات أولية (Primitive Variables) أو احتوائها على كائنات الصنف String. وتحتوي صفات أصناف شجرة توارث الصنف Lease على كائنات من الصنف DateTime التي لا يخلو نظام من التعامل معه (لمعالجة التاريخ والوقت)، وقد قدمنا إجراءات الصنف DateTime في الفصل الرابع.

يوضح الشكل رقم (٨.١٩) شجرة توارث الصنف Lease حيث يتفرع من هذا الصنف صنفين فرعيين وهما: الصنف AnnualLease والصنف DailyLease.



الشكل رقم (٨,١٩). شجرة توارث الصنف Lease.

يوضح الشكل رقم (٨,٢٠) تعريف الصنف الرئيس Lease والذي يحتوي على ثلاث صفات وهي: صفة Amount وصفة Start Date وصفة End Date وقد تم تعريفه من نوع الصنف المجرد (Abstract Class) مستخدماً الكلمة MustInherit في رأس الصنف (يمكنك أن تفترض أنك اختبرت هذا الصنف قبل جعله صنفاً مجرداً).

```
' Lease -- an initial abstract Lease Class
' with an abstract method

Public MustInherit Class Lease

   'attributes
   Private amount As Double
   Private startDate As DateTime
   Private endDate As DateTime

   'constructor (1 parameter)
   Public Sub New(ByVal aStartDate As DateTime)
      SetStartDate(aStartDate)
      SetEndDate(Nothing)       ' endDate set by subclass
      SetAmount(0)
   End Sub

   'TellAboutSelf method
   Public Overridable Function TellAboutSelf() As String
      Return (startDate & ", " & endDate & ", " & amount)
   End Function

   'custom method CalculateFee based on slip width
   'abstract method all subclasses must implement
   Public MustOverride Function CalculateFee(ByVal aWidth As Integer) _
   As Single

   'Get accessor methods
   Public Function GetStartDate() As DateTime
      Return startDate
   End Function
   Public Function GetEndDate() As DateTime
      Return endDate
   End Function
   Public Function GetAmount() As Double
      Return amount
   End Function

   'Set accessor methods
   Public Sub SetStartDate(ByVal aStartDate As DateTime)
      startDate = aStartDate
   End Sub
   Public Sub SetEndDate(ByVal anEndDate As DateTime)
      endDate = anEndDate
   End Sub
   Public Sub SetAmount(ByVal anAmount As Double)
      amount = anAmount
   End Sub
End Class
```

الشكل رقم (٨,٢٠). تعريف الصنف Lease.

يحتوي الصنف Lease على صفتين من النوع DateTime وصفة من النوع الرقمي Double، بينما يستقبل إجراء الإنشاء معاملاً واحداً فقط عبارة عن متغير إشارة من النوع DateTime والذي يمثل تاريخ بداية العقد. ومن ثم يسند إجراء الإنشاء الصفة End Date بالكلمة المحجوزة Nothing، والصفة Amount بصفر مستدعياً إجراءات الوصول Setters. وسوف يسند كل صنف على حده تاريخ نهاية العقد وقيمة إيجار العقد بعد حسابها.

إضافة إجراء مجرد إلى الصنف Lease
Adding an Abstract Method to Lease

في بعض الأحيان نحتاج إلى تعريف إجراء محدد داخل جميع الأصناف الفرعية لصنف رئيس ما. فعلى سبيل المثال إن الأصناف الفرعية للصنف Lease يجب أن تحتوي على الإجراء CaculateFee المسؤول عن حساب قيمة العقد. وحيث إن حساب قيمة العقد تختلف على حسب نوع العقد، فإن ذلك يتطلب تعريف هذا الإجراء داخل جميع الأصناف الفرعية للصنف Lease وإذا لم يُعرَّف هذا الإجراء داخل أحد الأصناف الفرعية للصنف Lease سيحدث خطأ عند استدعاء هذا الإجراء لحساب قيمة هذا العقد.

ولكي تلزم جميع الأصناف الفرعية لصنف ما أن تطور إجراءاً محدداً يجب عليك أن تعرِّف هذا الإجراء داخل الصنف الرئيس من النوع المجرد وهو إجراء لا يحتوي على أوامر ويجب على الأصناف الفرعية إعادة تعريفه. يجب أن تلاحظ أن الإجراء المجرد يجب أن يعرف داخل صنف مجرد وتستخدم لغة VB .NET الكلمة المحجوزة MustOverride لتعريف إجراء مجرد.

عزيزي القارئ لاحظ أن الصنف Lease يحتوي على الإجراء المجرد CaculateFee المعرف مستخدماً الكلمة MustOverride ويستقبل عرض المرسى (متغير من النوع Integer) ثم يعيد قيمة حساب العقد (من النوع Integer). كما يجب أن تلاحظ أن هذا الإجراء لا يحتوي على أي أوامر (كما هو واضح في الأوامر التالية) لأن كل صنف فرعي يحسب قيمة العقد بطريقة مختلفة بناء على نوعه وشروطه.

```
'custom method CalculateFee based on slip width
'an abstract method all subclasses must implement
Public MustOverride Function CalculateFee(ByVal aWidth As Integer) _
As Single
```

تطوير الصنف الفرعي AnnualLease
Implementing the AnnualLease Subclass

يحتوي نموذج Class Diagram الموضح في الشكل رقم (٨.١٩) على الأصناف الفرعية AnnualLease (عقد سنوي) و DailyLease (عقد يومي) واللذان يمثلان أنواع العقود في شركة برادشو مارينا. يحتوي الصنف

AnnualLease على الصفة balanceDue من النوع Single التي تخزن القيمة المتبقية على العميل من قيمة العقد، والصفة payMonthly من النوع Boolean التي تحدد هل العميل سوف يدفع قيمة العقد نقداً أم سوف يقسط قيمة العقد على ١٢ شهراً. فإذا احتوت هذه الصفة على القيمة True فذلك يعني أن الصفة balanceDue ستحتوي بداية على مجموع ١١ دفعة على افتراض أن العميل دفع الدفعة الأولى عند توقيع العقد، وإذا احتوت على القيمة False فذلك يعني أن الصفة balanceDue ستحتوي على القيمة صفر.

يوضح الشكل رقم (٨.٢١) على تعريف الصنف AnnualLease، وبالرغم من أن هذا الصنف يحتوي على صفتين فقط فإن إجراء الإنشاء وبعض الإجراءات الخاصة به تعتبر أكثر تعقيداً عن الأمثلة السابقة.

عزيزي القارئ تأكد من وجود الأمر Inherits Lease في السطر الثاني، وتأكد من تعريف الصفتين الزائدتين (الصفة balanceDue والصفة payMonthly) داخل الصنف AnnualLease، ولاحظ أيضاً أن إجراء الإنشاء يستقبل ثلاثة معاملات وهي: المعامل تاريخ بداية العقد aStartDate، والمعامل عرض المرسى aSlipWidth، والمعامل أسلوب الدفع isPayMonthly. نحتاج إلى عرض المرسى لحساب قيمة العقد فقط، حيث لا توجد حاجة أخرى لهذا المعامل داخل الكائن. ويعتبر هذا أول مثال في هذا الكتاب الذي يوضح عدم إسناد قيمة مستقبلة لإجراء إنشاء إلى صفه. ونحتاج أيضاً معرفة أسلوب الدفع لحساب القيمة المتبقية على العميل والتي تمثل صفة لكائن العقد السنوي (balanceDue). توضح الأوامر التالية تعريف رأس إجراء إنشاء الصنف AnnualLease:

```
'constructor (three values as parameters)
Public Sub New(ByVal aStartDate As DateTime, _
    ByVal aSlipWidth As Integer, _
    ByVal isPayMonthly As Boolean)
```

يعتبر إجراء إنشاء الصنف AnnualLease أصعب بكثير من الأمثلة السابقة حيث يستدعي الأمر الأول إجراء إنشاء الصنف الرئيس (في هذه الحالة الصنف Lease) مرسلاً معامل تاريخ بداية العقد. عندئذ يسند صفة تاريخ بداية العقد ويسند Nothing إلى تاريخ نهاية العقد ويسند القيمة صفر إلى صفة قيمة العقد Amount كما ذكرنا سابقاً.

ثم يرجع التحكم إلى إجراء إنشاء الصنف AnnualLease الذي يحسب تاريخ نهاية العقد بإضافة سنة إلى تاريخ بداية العقد باستخدام الإجراء AddYears المعرف داخل الصنف DateTime ثم يستدعي الإجراء SetEndDate المعرف داخل الصنف الرئيس Lease لإسناد صفة تاريخ نهاية العقد:

```
'calculate end date
Dim yearLater as DateTime = aStartDate.AddYears(1)
'invoke superclass method to set end date
SetEndDate(yearLater)
```

```
' AnnualLease -- a subclass of Lease

Public Class AnnualLease
   Inherits Lease

   'attributes in addition to those inherited from Lease
   Private balanceDue As Single
   Private payMonthly As Boolean

   'constructor (three values as parameters)
   Public Sub New(ByVal aStartDate As DateTime, _
      ByVal aSlipWidth As Integer, _
      ByVal isPayMonthly As Boolean)
      'invoke superclass constructor
      MyBase.New(aStartDate)
      'calculate end date
      Dim yearLater as DateTime = aStartDate.AddYears(1)
      'invoke superclass method to set end date
      SetEndDate(yearLater)
      'invoke superclass SetAmount method after getting
      'fee amount from CalculateFee method
      SetAmount(CalculateFee(aSlipWidth))
      'set payMonthly
      SetPayMonthly(isPayMonthly)
      'set balance due if applicable
      If payMonthly Then
         setBalanceDue(GetAmount() - GetAmount()/12)
      Else
         SetBalanceDue(0)
      End If
   End Sub

   'custom method CalculateFee based on slip width
   Public Overridden Function CalculateFee(ByVal aWidth As Integer) _
   As Single
      Dim fee As Single
      Select Case aWidth
         Case 10
            fee = 800
         Case 12
            fee = 900
         Case 14
            fee = 1100
         Case 16
            fee = 1500
         Case Else
            Fee = 0
      End Select
      Return fee
   End Function

   'TellAboutSelf overrides then invokes superclass method
   Public Overrides Function TellAboutSelf() As String
      Return MyBase.TellAboutSelf() & ", " _
      & balanceDue & ", " & payMonthly
   End Function

   'Get accessor methods
   Public Function GetBalanceDue() As Single
      Return balanceDue
   End Function
   Public Function GetPayMonthly() As Boolean
      Return payMonthly
   End Function

   'Set accessor methods
   Public Sub SetBalanceDue(ByVal aBalanceDue As Single)
      balanceDue = aBalanceDue
   End Sub
   Public Sub SetPayMonthly(ByVal isPayMonthly As Boolean)
      payMonthly = isPayMonthly
   End Sub

End Class
```

الشكل رقم (٨,٢١). تعريف الصنف AnnualLease.

ثم يستدعي الإجراء SetAmount المعرف داخل الصنف الرئيس Lease لإسناد قيمة العقد وذلك بعد حسابها بواسطة استدعاء الإجراء CaculateFee المعرف داخل الصنف AnnualLease. يوضح هذا المثال أن استدعاء الإجراءات ربما يتداخل (ينفذ من الداخل إلى الخارج) حيث يتم استدعاء الإجراء CaculateFee الذي يعيد القيمة التي تمرر كمعلومة لإجراء SetAmount:

```
'invoke superclass SetAmount method after getting
'fee amount from CalculateFee method
SetAmount(CalculateFee(aSlipWidth))
```

ثم تسند آخر الأوامر قيمة إلى الصفة payMonthly ثم تحسب قيمة وتسند إلى الصفة balanceDue والتي تعتمد طريقة حسابها على قيمة الصفة payMonthly مستخدماً الأمر If. إذا كان العميل سيدفع قيمة العقد على ١٢ شهراً فسوف تكون قيمة balanceDue تساوي ١١ دفعة، أما إذا كان غير ذلك تكون قيمة balanceDue تساوي صفراً:

```
'set payMonthly
SetPayMonthly(isPayMonthly)
'set balance due if applicable
If payMonthly Then
    setBalanceDue(GetAmount() - GetAmount()/12)
Else
    SetBalanceDue(0)
End If
```

لقد عرفنا الإجراء LeaseSlip داخل الصنف Slip في الفصل السابع والذي كان مسؤولاً عن حساب قيمة العقد بناء على عرض المرسى وذلك لتبسيط المثال، ولكن في النظام الكامل لشركة برادشو مارينا كان المسؤول عن حساب قيمة العقد الأصناف الفرعية للصنف Lease، ومن ثم أعدنا تعريف الإجراء CaculateFee في جميع الأصناف وعرفناه في الصنف الرئيس كإجراء مجرد (Abstract Method)، ويتم حساب قيمة العقد بنفس الأسلوب المتبع في الصنف Slip مستخدماً الأمر Select Case، حيث يتم استدعاء هذا الإجراء داخل إجراء إنشاء الصنف AnnualLease لإسناد قيمة العقد Amount. ويجب أن تلاحظ إن رأس الإجراء CaculateFee يحتوي على الكلمة Overrides لأنها تعيد تعريف الإجراء المجرد في الصنف الرئيس Lease:

```
'custom method CalculateFee based on slip width
Public Overrides Function CalculateFee(ByVal aWidth As Integer) _
As Single
    Dim fee As Single
    Select Case aWidth
```

```
        Case 10
            fee = 800
        Case 12
            fee = 900
        Case 14
            fee = 1100
        Case 16
            fee = 1500
        Case Else
            fee = 0
    End Select
    Return fee
End Function
```

تطوير الصنف الفرعي DailyLease

Implementing the DailyLease Subclass

أما أنواع العقد الآخر الذي يمكن أن يتم في شركة برادشو ماريتا هو العقد اليومي (يمثله الصنف DailyLease)، والذي يمكّن العميل من إيجار المرسى لمدة محددة تتراوح بين عدة أيام إلى عدة شهور. يوضح الشكل رقم (٨,٢٢) التعريف الكامل للصنف DailyLease الذي يحتوي على صفة واحدة إضافية وهي الصفة Number of Days التي تخزن عدد أيام الإيجار والتي تستنتج بطرح تاريخ بداية العقد من تاريخ نهاية العقد. وبالطبع لا توجد أقساط شهرية ولا توجد قيمة متبقية.

```
Public Class DailyLease
   Inherits Lease

   'attribute in addition to those inherited from Lease
   Private numberOfDays As Integer

   'constructor (three values as parameters)
   Public Sub New(ByVal aStartDate As DateTime, _
      ByVal anEndDate As DateTime, _
      ByVal aSlipWidth As Integer)
      'invoke superclass constructor
      MyBase.New(aStartDate)
      'calculate number of days
      Dim diff As System.TimeSpan
      diff = anEndDate.Subtract(aStartDate)
      Dim days as Integer = diff.Days()
      SetNumberOfDays(days)
      'invoke superclass method to set end date
      SetEndDate(anEndDate)
      'invoke superclass SetAmount method after getting
      'fee amount from CalculateFee method
      SetAmount(CalculateFee(aSlipWidth))
   End Sub
```

الشكل رقم (٨,٢٢). تعريف الصنف DailyLease مع الإجراء CalculateFee.

```
'custom method CalculateFee based on slip width and number of days
Public Overrides Function CalculateFee(ByVal aWidth As Integer) _
As Single
   Dim fee As Single
   Select Case aWidth
      Case 10
         fee = 20 * numberOfDays
      Case 12
         fee = 25 * numberOfDays
      Case 14
         fee = 30 * numberOfDays
      Case 16
         fee = 35 * numberOfDays
      Case Else
         fee = 0
   End Select
   Return fee
End Function

'TellAboutSelf overrides then invokes superclass method
Public Overrides Function TellAboutSelf() As String
   Return MyBase.TellAboutSelf() & ", " _
   & numberOfDays
End Function

'Get accessor method
Public Function GetNumberOfDays() As Integer
   Return numberOfDays
End Function

'Set accessor method
Public Sub SetNumberOfDays(ByVal aNumberOfDays As Integer)
   numberOfDays = aNumberOfDays
End Sub

End Class
```

تابع الشكل رقم (٨,٢٢).

يستقبل إجراء إنشاء الصنف DailyLease ثلاثة معاملات تعني: المعامل aStartDate (تاريخ بداية العقد)، والمعامل anEndDate (تاريخ نهاية العقد)، والمعامل aSlipWidth (عرض المرسى). يستنتج الصنف AnnualLease تاريخ نهاية العقد بينما يستنتج الصنف DailyLease عدد أيام العقد.

يستدعي إجراء إنشاء الصنف DailyLease أولاً إجراء إنشاء الصنف الرئيس Lease ممرراً إليه تاريخ بداية العقد، ثم يستدعي الإجراء Subtract المعرف داخل الصنف DateTime لكن يطرح تاريخ بداية العقد من تاريخ نهاية العقد ويستقبل كائناً من الصنف TimeSpan ثم يستدعي الإجراء Days المعرف داخل الصنف TimeSpan للحصول على عدد أيام العقد:

```
'calculate number of days
Dim diff As System.TimeSpan
diff = anEndDate.Subtract(aStartDate)
Dim days as Integer = diff.Days()
SetNumberOfDays(days)
```

تستدعي آخر أوامر إجراء إنشاء إجراءات الوصول Setters المعرفة داخل الصنف الرئيس لتسند قيم صفة تاريخ نهاية العقد وصفة قيمة العقد. يحسب الإجراء CaculateFee، الذي يعيد تعريف الإجراء المعرف داخل الصنف الرئيس، قيمة العقد بناء على عرض المرسى وعدد أيام العقد، حيث يمرر له معامل عرض المرسى من إجراء الإنشاء ولا يتم الاحتفاظ بقيمته. لاحظ أن عدد أيام العقد يتم حسابها وإسنادها بواسطة إجراء الإنشاء. ولاحظ أيضاً أن عدد أيام العقد تستخدم في حساب قيمة العقد ولا تمرر للإجراء CaculateFee لأنها صفة داخل الصنف DailyLease، ومن ثم يمكن للإجراء CaculateFee أن يتعامل معها مباشرة.

اختبار الصنف الفرعي DailyLease والصنف الفرعي AnnualLease
Testing the AnnualLease and DailyLease Classes

بعد أن طورنا الأصناف الفرعية للصنف Lease يمكننا اختبارها بواسطة الإجراءات المصاحبة لأزرار النموذج الموضح في الشكل رقم (٨.٢٣) حيث يوضح الشكل رقم (٨.٢٤) إجراء اختبار الصنف AnnualLease.

ولاختبار الصنف AnnualLease سوف ننشئ كائناً من النوع DateTime للاحتفاظ بتاريخ بداية العقد لعقد من النوع السنوي. ثم ننشئ كائنان من الصنف AnnualLease الأول يكون الدفع فيه شهرياً (الصفة payMonthly تساوي True) والثاني يكون الدفع فيه نقداً (الصفة payMonthly تساوي False)، هذا بالإضافة إلى اختلاف عرض المرسى في كلا العقدين. يوضح الشكل رقم (٨.٢٥) مخرجات هذا الإجراء مستدعياً إجراءات الوصول Getters لاسترجاع معلومات العقدين.

الشكل رقم (٨.٢٣). نموذج اختبار الصنف Lease.

```
Private Sub btnAnnualLease_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnAnnualLease.Click

    'instantiate some test dates
    Dim newDate As New DateTime(2004, 3, 15)
    Dim anotherDate As New DateTime(2004, 4, 10)

    'instantiate two annual leases
    Dim lease1 As New AnnualLease(newDate, 14, True)
    Dim lease2 As New AnnualLease(anotherDate, 16, False)

    'retrieve information about the annual lease
    Console.WriteLine("AnnualLease 1 information")
    Console.WriteLine("Dates: " & lease1.GetStartDate() & " " _
     & lease1.GetEndDate())
    Console.WriteLine("Amount/Balance/PayMonthly: " _
     & lease1.GetAmount() & " " & lease1.GetBalanceDue() & " " _
     & lease1.GetPayMonthly())

    'retrieve information about second annual lease
    Console.WriteLine("AnnualLease 2 information")
    Console.WriteLine("Dates: " & lease2.GetStartDate() & " " _
     & lease2.GetEndDate())
    Console.WriteLine("Amount/Balance/PayMonthly: " _
     & lease2.GetAmount() & " " & lease2.GetBalanceDue() & " " _
     & lease2.GetPayMonthly())

End Sub
```

الشكل رقم (٨,٢٤). شفرة إجراء اختبار الصنف AnnualLease.

```
AnnualLease 1 information
Dates: 3/15/2004 3/15/2005
Amount/Balance/PayMonthly: 1100 1008.333 True
AnnualLease 2 information
Dates: 4/10/2004 4/10/2005
Amount/Balance/PayMonthly: 1500 0 False
```

الشكل رقم (٨,٢٥). مخرجات إجراء اختبار الصنف AnnualLease.

ولاختبار الصنف DailyLease سوف ننشئ ثلاثة كائنات من الصنف DateTime وذلك لاستخدامها لتخزين تواريخ بداية العقد ونهاية العقد للعقد من النوع اليومي. ثم يتم إنشاء كائنات من الصنف DailyLease. ونستخدم أيضاً عرض مرسى مختلف لكل عقد. يوضح الشكل رقم (٨,٢٦) إجراء اختبار الصنف DailyLease، كما يوضح الشكل رقم (٨,٢٧) مخرجات هذا الإجراء حيث تستدعي إجراءات الوصول Getters لاسترجاع معلومات العقدين وطباعتها على الشاشة.

```
Private Sub btnDailyLease_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnDailyLease.Click

    'instantiate some test dates
    Dim newDate As New DateTime(2004, 3, 15)
    Dim anotherDate As New DateTime(2004, 4, 10)
    Dim thirdDate As New DateTime(2004, 4, 21)

    'instantiate two daily leases
    Dim lease1 As New DailyLease(newDate, anotherDate, 14)
    Dim lease2 As New DailyLease(anotherDate, thirdDate, 16)

    'retrieve information about first daily lease
    Console.WriteLine("DailyLease 1 information")
    Console.WriteLine("Dates: " & lease1.GetStartDate() _
     & " " & lease1.GetEndDate())
    Console.WriteLine("Amount/Number of Days: " & lease1.GetAmount() _
     & " " & lease1.GetNumberOfDays())

    'retrieve information about second daily lease
    Console.WriteLine("DailyLease 2 information")
    Console.WriteLine("Dates: " & lease2.GetStartDate() _
     & " " & lease2.GetEndDate())
    Console.WriteLine("Amount/Number of Days: " & lease2.GetAmount() _
     & " " & lease2.GetNumberOfDays())

End Sub
```

الشكل رقم (٨,٢٦). شفرة إجراء اختبار الصنف DailyLease.

```
DailyLease 1 information
Dates: 3/15/2004 4/10/2004
Amount/Number of Days: 780 26
DailyLease 2 information
Dates: 4/10/2004 4/21/2004
Amount/Number of Days: 385 11
```

الشكل رقم (٨,٢٧). مخرجات إجراء اختبار الصنف DailyLease.

فهم المكونات المجردة واستخدامها

Understanding and Using Interfaces

إن تعريف إجراء من النوع المجرد ليس هو الطريقة الوحيدة لإجبار الأصناف الفرعية أن تعرّف هذا الإجراء داخلها (مثل الإجراء CaculateFee)، وإن طلب وجود إجراء ما داخل أصناف فرعية لصنف ما يستلزم من مستخدمي هذا الصنف (المبرمجين الآخرين الذين سيستخدمون هذا الصنف) أن يعرّفوا هذا الإجراء داخل الأصناف الفرعية لهذا الصنف لكي لا يحدث خطأ عند استدعاء هذا الإجراء من كائنات الأصناف الفرعية.

كما توجد طريقه أخرى لإجبار مستخدم صنف ما أن يعرف إجراء ما، وذلك عن طريق تعريف الواجهات (Interfaces). الواجهة هي مكون VB .NET يحتوي على إجراءات مجردة وثوابت التي يجب إعادة تعريفها بواسطة الصنف الذي سيستخدم هذه الواجهة. إن مفهوم الواجهات بسيط جداً والذي تلخصه العبارة "كيفية

استخدام شيء ما يعني كيفية التعامل مع واجهة هذا الشيء"، وعلى سبيل المثال أنت وأنا نقود سيارة عن طريق واجهة السيارة (عجلة القيادة، ودواسة الفرامل، ودواسة البنزين، وهكذا)، حيث نتوقع جميعاً وجود هذه الواجهة في جميع أنواع السيارات. ولذلك إنه لمن المرغوب فيه وضع واجهة محددة ومعرفة للتعامل مع الأشياء. فعلى سبيل المثال، إذا قيل لك هل تستطيع قيادة سيارة مارس روفر Mars Rover ستجاوب بنعم إذا كانت تحتوي على عجلة قيادة، ودواسة بنزين، إلخ.

في النظم المعتمدة على الكائنات نستخدم أو نتحكم في كائن ما لصنف ما بواسطة إرسال رسائل له بناء على توقيعات الإجراءات المعرفة لديه؛ ولذلك يمكن تعريف هذه الإجراءات داخل واجهة كإجراءات مجردة. ثم إذا أردنا لمبرمج آخر أن يستخدم هذا الصنف، فيجب علينا تعريف الواجهة أولاً ثم نوصف أن هذا الصنف سوف يطور هذه الواجهة. والآن يعلم هذا المبرمج جيداً ما تستطيع كائنات هذا الصنف أن تفعله (الإجراءات المعرفة لدى الواجهة).

إن مفهوم تطوير الأنظمة المعتمدة على المكونات (Component-based Development) يشير إلى أن المكونات داخل النظام تتفاعل من خلال واجهاتها حتى لو اختلفت تقنيات تلك المكونات. وطالما علمنا كيف نتصل مع مكون (معرفة واجهته) ونتفاعل معه فإن ذلك يعني إمكانية استخدامه داخل النظام. وللعلم فإن هذا التفاعل لا يتطلب معرفة تقنية هذا المكون ولا معرفة التركيب الداخلي له. تمثل الواجهات تقنية تعريف كيفية استخدام المكونات ومن ثم تلعب دوراً هاماً في تطوير الأنظمة المعتمدة على الكائنات

عادة ما نشرح واجهات لغة VB .NET داخل سياق مفهوم التوارث حيث نطلق على الأصناف التي تطور إجراءات واجهة ما أصنافاً ترث الإجراءات. لأن صنف لغة VB .NET لا يستطيع أن يرث إلا من صنف واحد فقط ولكن يستطيع أن يطور واجهة أو أكثر؛ لذلك يمكن تطبيق مفهوم التوارث المتعدد (Multiple Inheritance) مستخدماً هذا الأسلوب، حيث يعني التوارث المتعدد إمكانية السماح لصنف فرعي أن يرث من أكثر من صنف رئيس.

التوارث المتعدد يعني وجود صنف فرعي واحد داخل شجرتي توارث أو أكثر في الوقت نفسه. فعلى سبيل المثال، تذكر مثال السيارة مارس روفر المذكور سابقاً. إن هذه السيارة تمثل نوعاً (صنفاً فرعياً) للمركبة، وفي الوقت نفسه تمثل نوعاً آخر (صنفاً فرعياً) من سفينة الفضاء. في حالة البرمجة مع بيئة تدعم التوارث المتعدد، يكون صعباً على مترجم اللغة تنسيق بعض المهام مثل أي من الصفات يحيث ورائتها وأي الإجراءات يجب أن يعاد تعريفها.

تسمح بعض لغات البرمجة المعتمدة على الكائنات (مثل لغة C++) لك أن تستخدم التوارث المتعدد. ولكن مصممو لغة VB .NET قرروا ألا يضيفوا التوارث المتعدد لصعوبته وكذلك كل من لغة C# ولغة Java لم تقدما إمكانية التوارث المتعدد. ومن ثم يمكننا استخدام الواجهات كحل بديل لتطوير التوارث المتعدد مع لغة VB .NET واللغات الأخرى. كما يمكنك اكتشاف أن استخدام الواجهات في تطوير النظم المعتمدة على المكونات يعد أكثر أهمية.

إنشاء واجهة VB .NET
Creating a VB .NET Interface

تستطيع أن تطور واجهة VB .NET مثلما تطور الأصناف التي تشتمل على رأس وإجراءات مجردة التي يجب أن يعاد تعريفها داخل أي صنف سوف يستخدم هذه الواجهة. على سبيل المثال نتوقع أن الأصناف الفرعية للصنف Lease يجب أن تحتوي على الإجراء CaculateFee؛ ولذلك يمكن إنشاء واجهة تسمى ILeaseInterface وتحتوي على الإجراء المجرد CaculateFee (بدلاً من احتواء الصنف الرئيس Lease على هذا الإجراء المجرد). ولقد جرى العرف أن يبدأ اسم الواجهة بالحرف "I" واستخدام الكلمة Interface داخل اسم الواجهة لتمييزه عن أسماء الأصناف.

تعرف الواجهة داخل ملف الشفرة، حيث يتم ترجمته مثل الصنف بالضبط. تستخدم الواجهة ILeaseInterface الكلمة المحجوزة Interface (بدلاً من كلمة Class) متبوعة باسم الواجهة. يوضح الشكل رقم (٨,٢٨) تعريف الواجهة ILeaseInterface. يحتوي هذا التعريف على رأس إجراء واحد دون استخدام الكلمة Public والكلمة MustOverride غير المسموح استخدامهما في هذه الحالة. لاحظ أن الإجراء لا يحتوي على أي أوامر حيث إن جميع أوامر الإجراء لابد أن تكون داخل الصنف الذي سيطور هذه الواجهة.

```
Public Interface ILeaseInterface
   'All lease classes must include CalculateFee method
   Function CalculateFee(ByVal aWidth As Integer) As Single
End Interface
```

الشكل رقم (٨,٢٨). تعريف الواجهة ILeaseInterface.

وفي حالة تطوير واجهة ما وترجمتها، يمكن لأي صنف أن يبرمج هذه الواجهة وإعادة تعريف إجراءاتها. لاحظ أن هذه الواجهة تحتوي على إجراء واحد فقط. ولكن غالباً ما تحتوي الواجهة على قائمة من الإجراءات كما يمكن تطوير العديد من الواجهات داخل تطبيق VB .NET.

يستطيع الصنف AnnualLease أن يبرمج الواجهة ILeaseInterface مستخدماً الكلمة المحجوزة Implements متبوعة باسم الواجهة داخل رأس تعريف الصنف. ويجب أن تعلم أن الصنف AnnualLease ما زال يرث الصنف Lease. توضع الأوامر التالية رأس تعريف الصنف AnnualLease:

```
Public Class AnnualLease
    Inherits Lease
    Implements ILeaseInterface
```

يجب عليك حذف الإجراء المجرد CaculateFee من الصنف الرئيس Lease لأن الواجهة ILeaseInterface تتطلب إعادة تعريف هذا الإجراء داخل الصنف AnnualLease. يجب أيضاً تعديل رأس الإجراء CaculateFee داخل الصنف AnnualLease لتوضيح أن هذا الإجراء أصبح إعادة تعريف للإجراء المجرد والمعرف داخل الواجهة ILeaseInterface كما هو موضح في الأوامر التالية:

```
'custom method CalculateFee based on slip width
'required by ILeaseInterface
Public Function CalculateFee(ByVal sWidth As Integer) As Single _
Implements ILeaseInterface.CalculateFee
```

برمجة أكثر من واجهة
Implementing More Than One Interface

تستطيع أصناف VB .NET أن تبرمج أكثر من واجهة. افترض مثلاً أن شركة برادشو مارينا قررت أنه يجب على جميع أصناف مجال المشكلة برمجة واجهة ICompanyInterface تحتوي على الإجراء TellAboutSelf. بهذا الأسلوب يطمئن فريق المبرمجين أن جميع كائنات أصناف مجال المشكلة ستستجيب للرسالة TellAboutSelf. يوضح الشكل رقم (٨,٢٩) أوامر تعريف الواجهة ICompanyInterface.

```
Public Interface ICompanyInterface
   'all company classes must include a TellAboutSelf method
   Function TellAboutSelf() As String
End Interface
```

الشكل رقم (٨,٢٩). تعريف الواجهة ICompanyInterface.

يمكن استخدام الصنف DailyLease لتوضيح كيف يبرمج صنف أكثر من واجهة، حيث يتم كتابة الكلمة Implements ثم أسماء الواجهات المراد برمجتها (مع وجود فاصلة بين الأسماء) هكذا:

```
' DailyLease -- a subclass of Lease
' Implements two interfaces

Public Class DailyLease
    Inherits Lease
    Implements ILeaseInterface, ICompanyInterface
```

والآن مطلوب تعريف كل من الإجراء CaculateFee (برمجة الواجهة ILeaseInterface) والإجراء TellAboutSelf (برمجه الواجهة ICompanyInterface) داخل الصنف DailyLease، حيث يجب تعديل الإجراء CaculateFee كما فعلنا داخل الصنف AnnualLease. ويجب تعديل الإجراء TellAboutSelf لكي نشير إلى برمجة الواجهة ICompanyInterface هكذا :

```
'TellAboutSelf overrides then invokes superclass method
'required by ICompanyInterface
Public Overrides Function TellAboutSelf() As String _
Implements ICompanyInterface.TellAboutSelf
```

يوضح الشكل رقم (٨,٣٠) التعريف الكامل للصنف DailyLease بعد تعديله.

```
' DailyLease -- a subclass of Lease
' Implements two interfaces

Public Class DailyLease
   Inherits Lease
   Implements ILeaseInterface, ICompanyInterface

   'attribute in addition to those inherited from Lease
   Private numberOfDays As Integer

   'constructor (three values as parameters)
   Public Sub New(ByVal aStartDate As DateTime, _
      ByVal anEndDate As DateTime, _
      ByVal aSlipWidth As Integer)
      'invoke superclass constructor
      MyBase.New(aStartDate)
      'calculate number of days
      Dim diff As System.TimeSpan
      diff = anEndDate.Subtract(aStartDate)
      Dim days as Integer = diff.Days()
      SetNumberOfDays(days)
      'invoke superclass method to set end date
      SetEndDate(anEndDate)
      'invoke superclass SetAmount method after getting
      'fee amount from CalculateFee method
      SetAmount(CalculateFee(aSlipWidth))
   End Sub

   'custom method CalculateFee based on slip width and number of days
   'required by ILeaseInterface
   Public Function CalculateFee(ByVal aWidth As Integer) As Single _
```

الشكل رقم (٨,٣٠). تعريف الصنف DailyLease بعد التعديل.

```
    Implements ILeaseInterface.CalculateFee
       Dim fee as Single
       Select Case aWidth
          Case 10
             fee = 20 * numberOfDays
          Case 12
             fee = 25 * numberOfDays
          Case 14
             fee = 30 * numberOfDays
          Case 16
             fee = 35 * numberOfDays
          Case Else
             fee = 0
       End Select
       Return fee
    End Function

    'TellAboutSelf overrides then invokes superclass method
    'required by ICompanyInterface
    Public Overrides Function TellAboutSelf() As String _
    Implements ICompanyInterface.TellAboutSelf
       Return MyBase.TellAboutSelf() & ", " _
       & numberOfDays
    End Function

    'get accessor method
    Public Function GetNumberOfDays() As Integer
       Return numberOfDays
    End Function

    'set accessor method
    Public Sub SetNumberOfDays(ByVal aNumberOfDays As Integer)
       numberOfDays = aNumberOfDays
    End Sub

End Class
```

تابع الشكل رقم (٨,٣٠).

استخدام الاستثناءات الخاصة

Using Custom Exceptions

ليست أصناف مجال المشكلة فقط هي التي يمكن توريثها. يمكن تطبيق مفهوم التوارث (Inheritance) على الأصناف المعرفة داخل لغة VB .NET والتي لم توصف مستخدماً الكلمة NotInheritable. على سبيل المثال، يمكن اشتقاق صنف فرعي من الصنف Exception المعرف داخل اللغة لإنشاء استثناء خاص بالتطبيق.

لقد تعلمنا كيف نستخدم الصنف Exception في الفصل السابع، حيث كان الصنف Slip يلقى استثناء (كائناً من الصنف Exception) عند استقبال بيانات غير صحيحة. ولكن من المفيد أن نضع معلومات أكثر تفصيلاً عن الخطأ عند إلقاء الاستثناء؛ ولذلك ليكن تعريف استثناء خاص لتقديم معلومة كاملة عن سبب حدوث الخطأ. ولعمل ذلك يمكن تعريف صنف فرعي من الصنف Exception، ثم نعرف صفات إضافية لتخزين سبب الخطأ. بعد ذلك نعيد تعريف الإجراءات المطلوب استدعاؤها بواسطة كائنات هذا الصنف.

تعريف الصنف LeasePaymentException

Defining LeasePaymentException

يسمح العقد السنوي (الصنف AnnualLease) للعملاء أن يدفعوا قيمة العقد على أقساط شهرية بعد خصم القسط الأول. ولكي نسجل بقية الأقساط، نحتاج إلى تعريف إجراء داخل الصنف AnnualLease، حيث يجب أن يتحقق هذا الإجراء من صحة قيمة القسط قبل تسجيله. فإذا كانت قيمة القسط غير صحيحة، سوف يتم إلقاء استثناء لتصحيح الخطأ؛ ولذلك يمكن تعريف استثناء خاص الذي يمكن إلقائه عند حدوث تلك المشكلة.

سوف نعرف الصنف الفرعي LeasePaymentException المشتق من الصنف Exception داخل نظام برادشو مارينا. يوضح الشكل رقم (٨,٣١) تعريف هذا الصنف كاملاً.

```
'custom exception example

Public Class LeasePaymentException
   Inherits Exception

   'attributes of the custom exception
   Dim theLease As AnnualLease
   Dim paymentAmount As Single
   Dim exceptionMessage As String

   'constructor (2 parameters)
   Public Sub New(ByVal anAmount As Single, _
   ByVal aLease As AnnualLease)
      'invoke superclass constructor
      MyBase.New("Lease Payment Exception")
      theLease = aLease
      paymentAmount = anAmount
      exceptionMessage = "LeasePaymentException for lease " & _
         theLease.GetStartDate() & _
         " with amount due " & _
         theLease.GetBalanceDue() & _
         " but payment made of " & paymentAmount
   End Sub

   'override ToString method of Exception
   'to provide more specific information about this exception
   Public Overrides Function ToString() As String
      Return exceptionMessage
   End Function

   'accessor methods can also be included to return
   'specific exception information that might be needed

End Class
```

الشكل رقم (٨,٣١). تعريف الصنف الفرعي LeasePaymentException.

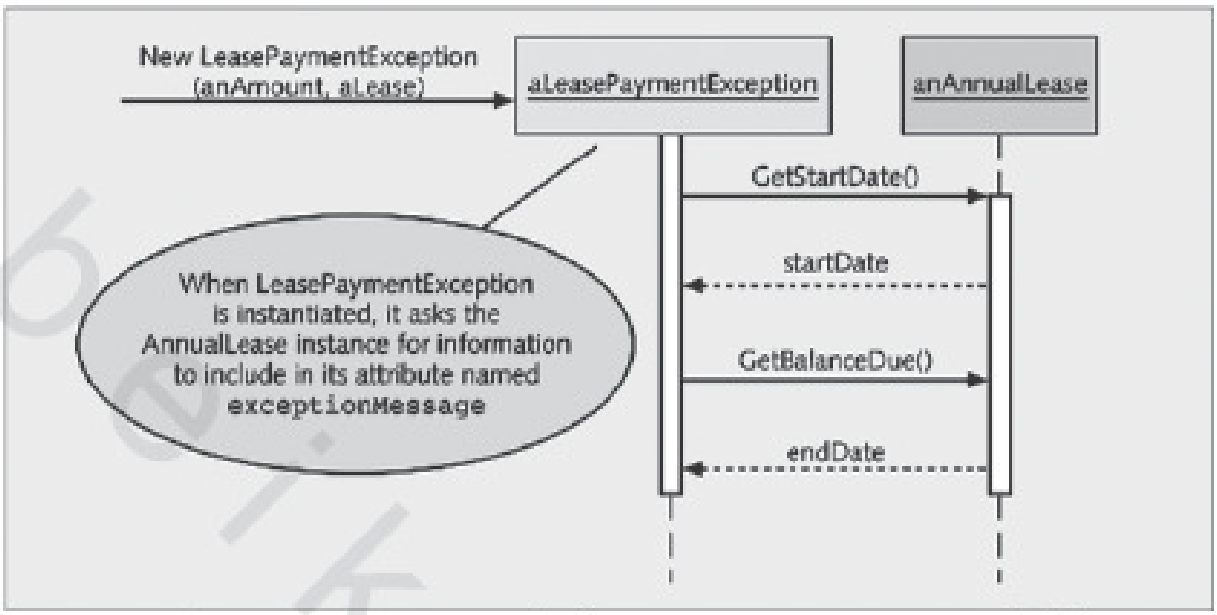
يحتوي الصنف LeasePaymentException على ثلاث صفات لتخزين قيمة القسط غير الصحيح ومتغير إشارة للصنف Lease (العقد المستقبل للصنف) ورسالة تفصيلية عن الخطأ.

لاحظ أن كائن الصنف LeasePaymentException يحتوي على متغير إشارة لكائن من الصنف Lease الذي يلقى الاستثناء عند حدوث الخطأ. إن مفهوم احتواء متغير الإشارة لكائن ما داخل صفة كائن آخر يعتبر من أهم مفاهيم برمجة النظم المعتمدة على الكائنات، حيث تنشئ علاقة بين هذين الكائنين وذلك مثل تعريف صفات من النوع DateTime داخل الصنف Lease. ولكن في هذه الحالة تحتوي الصفة على متغير إشارة لكائن من أصناف مجال المشكلة. سوف تتعلم عزيزي القارئ أهمية هذا المفهوم خلال الفصل القادم وذلك لإنشاء علاقات ترافق (Association Relationships) بين كائنات أصناف مختلفة. والآن لدينا علاقة تربط كائناً من النوع LeasePaymentException وكائناً من النوع AnnualLease.

يستقبل إجراء إنشاء الصنف LeasePaymentException قيم القسط غير الصحيح (لتخزينها داخل الصفة paymentAmount) ومؤشر إلى كائن الصنف AnnualLease (لتخزينه داخل الصفة theLease) ثم يستدعي إجراء إنشاء الصنف الرئيس Exception ممرراً إليه نص الخطأ المتوقع استقباله. ثم يتم إسناد الصفة exceptionMessage التي ستحتوي على قيم معينة تم الحصول عليها من كائن الصنف AnnualLease. وبمعنى آخر إن كائن الصنف LeasePaymentException يسأل كائن الصنف AnnualLease أن يرسل له كلاً من تاريخ بداية العقد والقيمة المتبقية على العميل لإسنادهما داخل الصفة exceptionMessage. وهذا يمثل مفهوم إرسال رسالة من كائن إلى كائن آخر للحصول على معلومات.

تذكر أن الصنف Exception يحتوي على الإجراء ToString الفرعي الذي يعيد معلومات عن الاستثناء، حيث يعيد النص الذي تم استقباله عند إنشائه بواسطة إجراء الإنشاء (كما رأينا في الفصل السابق). ويجب أن تعلم أن جميع الأصناف تستجيب للإجراء ToString (راجع موضوع الصنف Object لاحقاً في هذا الفصل). سوف يتم إعادة تعريف الإجراء ToString داخل الصنف لكي يعيد معلومات أكثر تفصيلاً حول الخطأ الذي حدث.

يوضح الشكل رقم (٨.٣٢) جزءاً من نموذج Sequence Diagram الذي يصور التفاعل بين كائنات كل من الصنف LeasePaymentException والصنف AnnualLease. لاحظ أن كائن الصنف LeasePaymentException يتقبل كائن الصنف AnnualLease ثم يطلب منه معلومات لتخزينها داخل الصفة exceptionMessage. سيتم استرجاع محتويات هذه الصفة بواسطة الإجراء الذي يستقبل هذا الاستثناء.



الشكل رقم (٨,٣٢). نموذج Sequence Diagram يوضح التفاعل بين LeasePaymentException وإجراء الإنشاء.

إلقاء الاستثناء الخاص
Throwing a Custom Exception

لقد طورنا الاستثناء الخاص (الصنف LeasePaymentException) للاستخدام داخل الصنف AnnualLease عند استقبال قيمة غير صحيحة؛ ولذلك يجب إضافة الإجراء RecordLeasePayment إلى الصنف AnnualLease لتسجيل الأقساط السنوية، حيث يستقبل هذا الإجراء قيمة القسط السنوي ثم ينشئ استثناءً خاصاً (كائناً من الصنف LeasePaymentException) ويلقيه إذا كانت قيمة القسط غير صحيحة، حيث يتم إنشاء كائن من الصنف LeasePaymentException وتمرير معاملات قيمة القسط غير الصحيحة ومتغير إشارة لكائن الصنف AnnualLease. أما إذا كانت قيمة القسط صحيحة فيتم طرحها من القيمة المتبقية على العميل balanceDue هكذا:

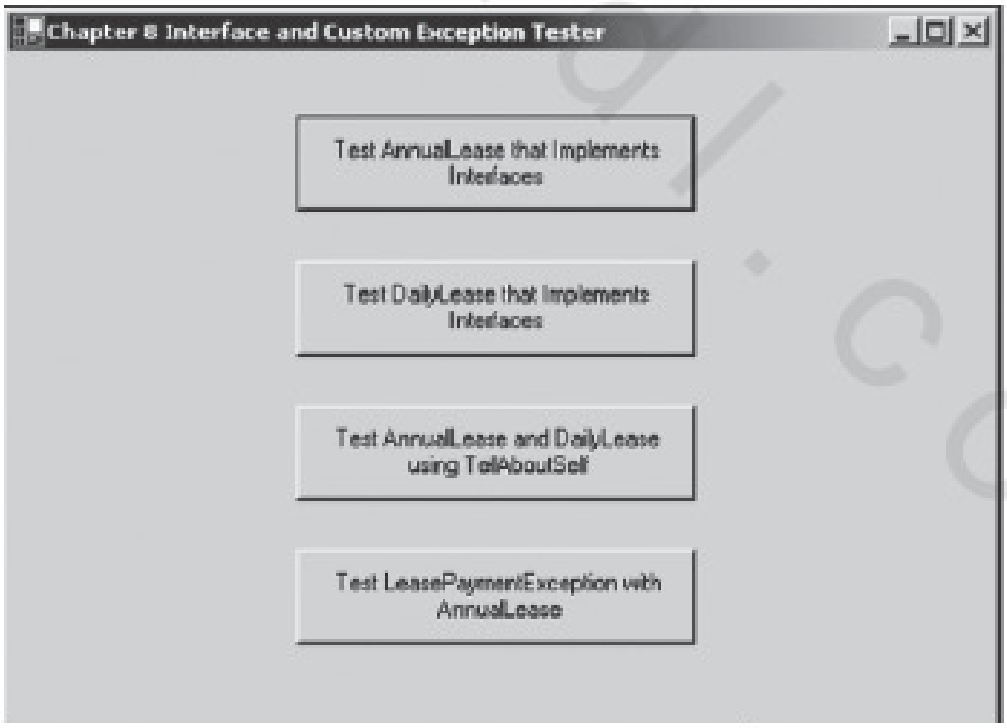
```
'custom method RecordLeasePayment to demonstrate
'LeasePaymentException (custom exception)
Public Sub RecordLeasePayment(ByVal anAmount As Single)
    If anAmount > balanceDue Then
        Throw New LeasePaymentException(anAmount, Me)
    Else
        balanceDue -= anAmount
    End If
End Sub
```

اختبار الصنف LeasePaymentException

Testing LeasePaymentException

يوضح الشكل رقم (٨,٣٣) نموذجاً بعنوان "Chapter 8 Interface and Custom Exception Tester" الذي يمكن أن يستخدم لاختبار الأصناف الفرعية للصنف Lease، حيث يوضح الشكل رقم (٨,٣٤) إجراء حدث اختبار كل من الصنف AnnualLease (خاصة الإجراء RecordLeasePayment) والصنف LeasePaymentException. ويوضح الشكل رقم (٨,٣٥) مخرجات هذا الإجراء. لقد استخدمنا تركيب Try وتركيب Catch وتركيب Finally لأن الإجراء RecordLeasePayment يلقي استثناءً عند حدوث الخطأ. ولاحظ عزيزي القارئ أن مخرجات إجراء الاختبار تحتوي على النص العائد من الاستثناء الخاص والذي يحتوي على معلومات تفصيلية عن الخطأ الذي حدث. تذكر أيضاً أن التركيب Finally سيتم تنفيذه دائماً بغض النظر عن حدوث خطأ أم لا.

إن قيمة العقد السنوي لهذا المثال تساوي ١٢٠٠ دولاراً بناء على عرض المرسى (١٤ قدماً). ولأن الصفة payMonthly تساوي True، فإن القيمة المتبقية تساوي ١٠٠٨,٣٣ دولاراً. ولقد حاولنا في هذا المثال خصم القسط الأول بقيمة ٨٠٠ دولاراً، عندئذ نجحت عملية الخصم وتبقى على العميل ٢٠٨,٣٣ دولاراً. ولكن عندما حاولنا خصم القسط الثاني (بقيمة ٨٠٠ دولاراً)، فشلت عملية الخصم لأن المتبقي أقل من القيمة المرسلة. يوضح الشكل رقم (٨,٣٦) جزءاً من نموذج Sequence Diagram الذي يضم تفاعلات أوامر إجراء الاختبار.



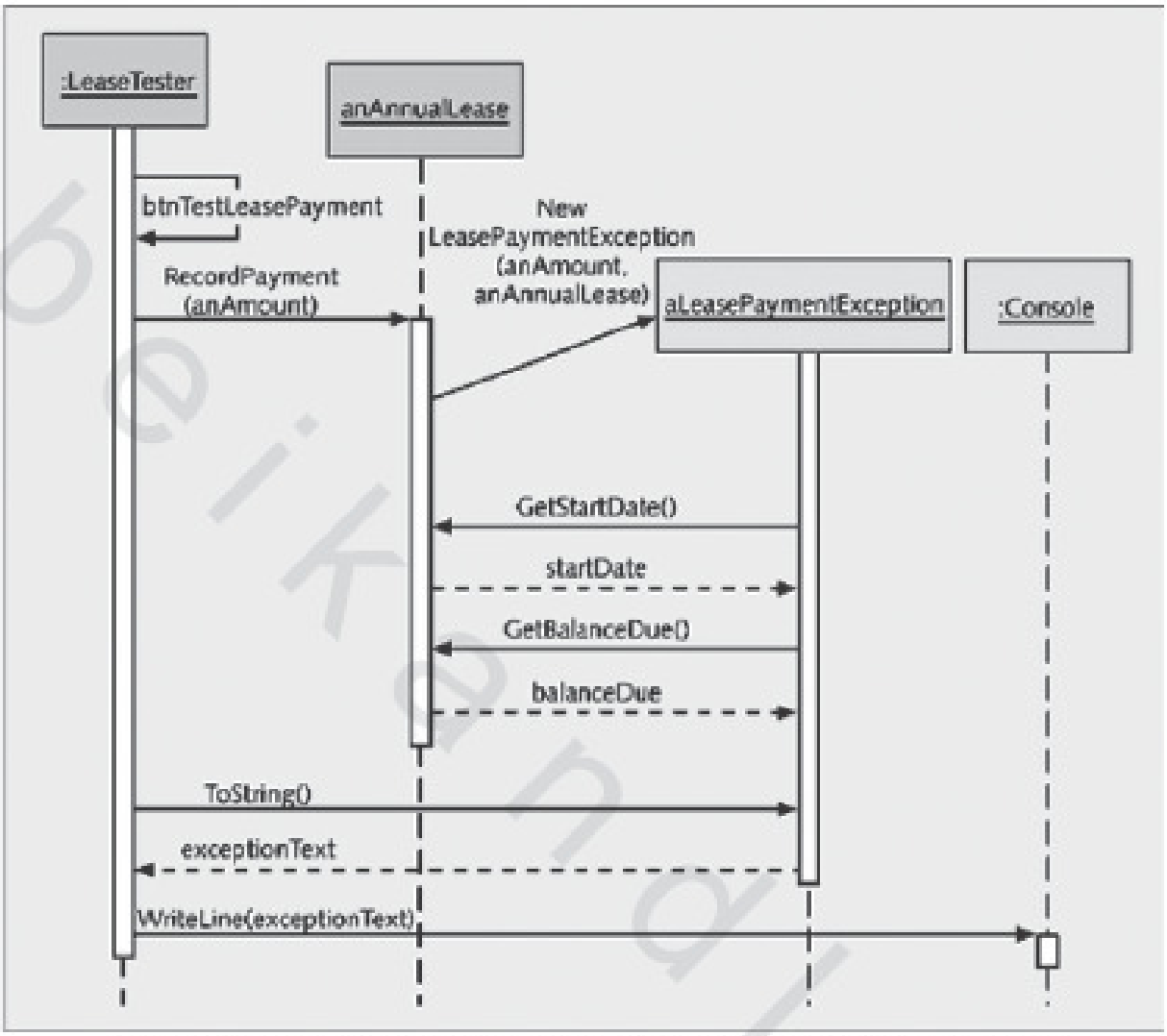
الشكل رقم (٨,٣٣). نموذج اختبار الواجهة والاستثناء الخاص.

```
Private Sub btnTestPaymentException_Click(ByVal sender _
As System.Object, ByVal e As System.EventArgs) _
Handles btnTestPaymentException.Click

   'create an annual lease amount 1100, balance due 1008.33
   Dim newDate as New DateTime(2004, 3, 15)
   Dim anotherDate as New DateTime(2004, 4, 10)
   Dim lease1 as New AnnualLease(newDate, 14, True)
   Console.WriteLine("Beginning balance is " & lease1.GetBalanceDue())

   'make a valid payment, balance will be 200.33
   Try
      lease1.RecordLeasePayment(800)
      Console.WriteLine("First payment successful. Balance is " _
      & lease1.GetBalanceDue())
   Catch theException As LeasePaymentException
      Console.WriteLine(theException.ToString())
   End Try

   'make an invalid payment, exception will be thrown by lease1

   Try
      lease1.RecordLeasePayment(800)
      Console.WriteLine("Second payment successful. Balance is " _
      & lease1.GetBalanceDue())
   Catch theException As LeasePaymentException
      Console.WriteLine(theException.ToString())
   Finally
      Console.WriteLine("End of second try catch block")
   End Try

End Sub
```

الشكل رقم (٨,٣٤). شفرة إجراء اختبار الصنف AnnualLease والصنف LeasePaymentException.

```
Beginning balance is 1008.333
First payment successful. Balance is 208.3333
LeasePaymentException for lease 3/15/2004 with amount due 208.3333 but payment made of
800
End of second try catch block
```

الشكل رقم (٨,٣٥). مخرجات إجراء اختبار الصنف LeasePaymentException.



الشكل رقم (٨,٣٦). نموذج Sequence Diagram يوضح تفاعل إجراء اختبار الصنف LeasePaymentException.

فهم الصنف Object والواجهة

Understanding the Object Class and Inheritance

لقد أنشأنا العديد من أصناف مجال المشكلة لنظام شركة برادشو مارينا وطورنا أصنافاً فرعية لها مستخدمين الكلمة المحجوزة Inherits، حيث ترث الأصناف الفرعية إجراءات الإجراءات الرئيسة وصفاتها. ولكن من أين تكتسب أصناف مجال المشكلة وظائفها الأساسية؟ والإجابة هي أن جميع أصناف لغة VB .NET ترث من صنف يطلق عليه اسم Object.

يعرف الصنف Object الوظائف الأساسية التي تحتاجها جميع الأصناف في لغة VB .NET، حيث إن جميع شجرات التوارث في لغة VB .NET تبدأ بالصنف Object، مع أن أي صنف يتم تعريفه يرث الصنف Object بشكل تلقائي دون تدخل منك. ولكن من المفيد التعرف على الصنف Object وعلى إجراءاته.

ولقد تعاملنا بالفعل مع أحد الإجراءات المعرفة داخل الصنف Object وهو الإجراء ToString، حيث إن جميع الأصناف ترث هذا الإجراء من الصنف Object، حيث إن هذا الإجراء يعيد بشكل افتراضي كائناً من النوع string يحتوي على اسم الصنف؛ ولذلك توجد العديد من الأصناف التي أعادت تعريف هذا الإجراء. لكن تعيد معلومات أكثر تفصيلاً كما فعلنا مع صنف الاستثناء السابق. كما يتجه العديد من المبرمجين إلى إعادة تعريف هذا الإجراء داخل أصناف مجال المشكلة لتقوم عمل الإجراء TellAboutSelf.

يحتوي أيضاً الصنف Object على الإجراء Equals والإجراء GetHashCode والإجراء GetType والإجراء ReferenceEquals وإجراء الاستثناء الاعتيادي New. يمكن تعريف الأصناف الفرعية في أصناف مجال المشكلة كأصناف فرعية للصنف Object صراحة مثل الصنف Lease هكذا:

```
Public Class Lease
    Inherits Object
```

أما الأصناف الفرعية مثل الصنف AnnualLease والصنف DailyLease فلا يمكن أن ترث الصنف Object مباشرة لكنها ترث الصنف Object بالفعل لأنها ترث الصنف Lease الذي يرث من صنف Object، حيث يمكن أن تستدعي الإجراء ToString الذي يرثه كائن الصنف AnnualLease هكذا:

```
Dim myLeaseInfo As String = lease1.ToString()
```

ملخص الفصل
Chapter Summary

- تذكر أن شجرة التعميم/التخصيص تعني أن الصنف الرئيس العام يحتوي على صفات وإجراءات مشتركة للأصناف الفرعية؛ لذلك فإن كائنات الأصناف الفرعية ترث الصفات والإجراءات المعرفة في الصنف الرئيس بالإضافة إلى الصفات والإجراءات المعرفة لديها. تقدم لغة VB .NET الكلمة المحجوزة Inherits التي تستخدم لإنشاء صنف فرعي (Sub Class) من صنف رئيس (Super Class).

- يجب أن يتم استدعاء الأمر MyBase.New في بداية إجراء إنشاء الصنف الفرعي لاستدعاء إجراء إنشاء الصنف الرئيس إلا إذا كان الصنف الرئيس لا يحتوي على إجراء إنشاء يستقبل معاملات، فعندئذ سيتم استدعاء إجراء الإنشاء الافتراضي الذي لا يستقبل معاملات بشكل تلقائي.
- يمكن إضافة صنف فرعي إلى صنف رئيس دون التأثير على أي الأصناف الأخرى. ويجوز أن يحتوي الصنف الفرعي على صنف فرعي آخر.
- الصنف المجرد (Abstract Class) هو الصنف الذي يمكن توريثه ولكن لا يمكن إنشاء كائن منه. تقدم لغة VB .NET الكلمة المحجوزة MustInherit لتعريف صنف مجرد. إن الأصناف المجردة مفيدة جداً لتعريف الصفات والإجراءات المشتركة بين أصنافها الفرعية، وبهذا يتحقق مفهوم "إعادة الاستخدام" (Reusability).
- تستخدم الكلمة NotInheritable لتوصيف صنف لا يمكن توريثه الذي يطلق عليه اسم الصنف النهائي (Final Class)؛ ولذلك من أجل الأمان أو الكفاءة نحتاج في بعض الأحيان ألا نسمح لصنف ما أن يتم توريثه بواسطة الآخرين.
- في بعض الأحيان نحتاج أن نعيد تعريف إجراء ما (معرف في صنف رئيس) داخل صنف فرعي لنزيد ما يقوم هذا الإجراء بعمله وهو ما نطلق عليه مفهوم إعادة تعريف الإجراءات (Method Overriding). سيتم استدعاء الإجراء المعرف داخل الصنف الرئيس مستخدماً الكلمة المحجوزة MyBase من الإجراء المعرف داخل الصنف الفرعي هكذا: MyBase.MethodName().
- يمثل مفهوم إعادة تعريف الإجراءات (Overriding) أحد أساليب تطبيق مفهوم تعدد تعريف الإجراءات (Polymorphism) الذي يعني وقوع سلوك مختلف من كائنات لأصناف مختلفة عند استقبال الرسالة نفسها. تستخدم لغة VB .NET طريقة "الربط المتغير" (Dynamic Binding) لكي تميز الإجراء المطلوب تنفيذه أثناء تشغيل النظام وذلك عند وجود أكثر من تعريف لهذا الإجراء داخل أصناف تربطهم علاقة توارث.
- تمنع درجة الوصول الخاص (Private) كائن الصنف الفرعي أن يتعامل مباشرة مع الصفات الموروثة من الصنف الرئيس، ومن ثم يجب استدعاء إجراءات الوصول Getters للحصول على قيم هذه الصفات. ولكن يمكنك تعريف صفات الصنف الرئيس من النوع المحمي (Protected) وذلك للسماح للصنف الفرعي أن يتعامل معها مباشرة دون الحاجة إلى استدعاء إجراءات الوصول.
- توجد بالحالة الدراسية لشركة برادشو مارينا على شجرة توارث أخرى وهي شجرة توارث الصنف Lease حيث يتفرع من هذا الصنف صنفين فرعيين وهما: الصنف AnnualLease والصنف DailyLease. ويجب أن تحتوي الأصناف الفرعية للصنف Lease على الإجراء CaculateFee المسؤول عن حساب قيمة العقد حيث إن حساب قيمة العقد تختلف على حسب نوع العقد.

- في بعض الأحيان نحتاج إلى تعريف إجراء محدد داخل جميع الأصناف الفرعية لصنف رئيس ما، فعندئذ يجب تعريف هذا الإجراء داخل الصنف الرئيس كإجراء مجرد.
- تستخدم واجهة (Interface) VB .NET كمكون لإجبار أي صنف أن يعيد تعريف مجموعة من الإجراءات التي ستستخدم هذه الواجهة. يتم تعريف الواجهة مستخدماً الكلمة المحجوزة Interface (بدلاً من كلمة Class) متبوعاً باسم الواجهة ثم يتم ترجمته مثل الصنف بالضبط. ويستطيع الصنف أن يبرمج الواجهة مستخدماً الكلمة المحجوزة Implements متبوعة باسم الواجهة داخل رأس تعريف الصنف.
- يمكن تطبيق مفهوم التوارث (Inheritance) على الأصناف المعرفة داخل لغة VB .NET مستخدماً الكلمة المحجوزة Inherits. فعلى سبيل المثال، يمكن اشتقاق صنف فرعي من الصنف Exception المعرف داخل اللغة لإنشاء استثناء خاص بالتطبيق. وإذا أردنا أن نضع معلومات أكثر تفصيلاً عن الخطأ عند إلقاء الاستثناء، فيجب تعريف استثناء خاص لتقديم معلومة كاملة عن سبب حدوث الخطأ.
- إن جميع أصناف لغة VB .NET ترث صنفاً يطلق عليه اسم الصنف Object الذي يقدم جميع الوظائف الأساسية التي تحتاجها جميع أصناف لغة VB .NET، ومن ثم فإن جميع شجرات التوارث في لغة VB .NET تبدأ بالصنف Object.

المصطلحات الأساسية
Key Terms

الصنف النهائي (Final Class)	مفهوم التوارث Inheritance
الصنف المجرد (Abstract Class)	الكلمة المحجوزة Inherits
الكلمة المحجوزة Private	الكلمة المحجوزة Implements
الكلمة المحجوزة Protected	الوصول الخاص (Private)
الإجراء المجرد (Abstract Method)	النوع المحمي (Protected)
الواجهة (Interface)	الربط المتغير (Dynamic Binding)
استثناء خاص (Custom Exception)	مفهوم تعدد تعريف الإجراءات (Polymorphism)
مفهوم إعادة تعريف الإجراءات (Method Overriding)	

أسئلة المراجعة
Review Questions

١- ضع في الاعتبار الصنف الرئيس والصنف الفرعي. في هيكل التعميم والتخصيص، أيهما أعم وأيهما أخص؟

٢- وضح كيف تطبق الوراثة في التعميم والتخصيص. ومن الذي ورث من قبل الصنف الفرعي؟

٣- ارسم مثالاً لهيكل التعميم والتخصيص يعرض على Class Diagram لصنف رئيس يسمى Car وصنف فرعي يسمى SportsCar.

٤- ما هي الكلمة المحجوزة في الفيجوال بيسك .نت التي تسمح لصنف واحد أن يورث من صنف آخر؟

٥- اكتب رأس الصنف SportsCar والذي يورث من الصنف Car.

٦- اكتب رأس الصنف المجرد المسمى Car.

٧- ما الذي سيحدث من برنامج اختبار يحتوي على عبارة Dim aCar as New Car(aMake, aModel) إذا كان Car هو صنف مجرد؟

٨- أين مكان كتابة العبارة MyBase.New(someValue) في الصنف الفرعي؟

٩- ما هو الفرق بين تعدد استخدام الإجراءات Overloading وإعادة تعريف الإجراءات Overriding؟

١٠- كيف يمكن لإعادة تعريف الإجراءات Overriding أن تسمح بتعدد تعريف الإجراءات؟ أعط مثالاً.

١١- اكتب العبارة التي تستدعي إجراء الصنف الرئيس بعد إعادة تعريفه لإجراء يسمى CalculateFee.

١٢- ما هما سببا تعريف الصنف النهائي باستخدام الكلمة المحجوزة NotInheritable؟

١٣- اكتب رأس الصنف المجرد المسمى Paycheck.

١٤- اكتب رأس الصنف المسمى HourlyPaycheck الذي يرث الصنف Paycheck على أن يكون هذا الصنف نهائياً.

١٥- ما هو تأثير استبدال الكلمة Protected بالكلمة Private في توصيف صفات الصنف الرئيس؟

١٦- ما هي الكلمة المحجوزة التي تعرِّف إجراءً داخل الصنف الرئيس الذي يجب أن يكون مضموناً في الصنف الفرعي؟

١٧- إذا احتوى الصنف على إجراء مجرد، هل من الممكن أن يحتوي الصنف نفسه على كائنات من هذا الصنف؟ وضح.

١٨- اكتب رأس واجهة لواجهة تسمى IBoatInterface.

١٩- اكتب رأس صنف لصنف يسمى Sailboat، والذي يطبق IBoatInterface.

٢٠- وضح ما هي الأصناف التي يمكن أن ترث أكثر من أصناف مجال مشكلة.

٢١- وضح لماذا الاستثناءات الخاصة مهمة في أنظمة الأعمال؟

٢٢- اكتب رأس صنف لاستثناء خاص يسمى LeaseCreationException.

٢٣- افترض أن LeaseCreationException تم إلقائها بواسطة Try-Catch في برنامج اختبار الصنف DailyLease. اكتب Try-Catch الكاملة التي تلقي الاستثناء وتظهر الرسالة.

٢٤- اكتب رأس الصنف للصنف Boat الذي يرث صنف Object ويطبق IBoatInterface و ICompanyInterface.

أسئلة المناقشة

Discussion Questions

١- لقد تحدثنا كثيراً عن أهمية تطبيق مفهوم التوارث عند تطوير النظم المعتمدة على الكائنات. ناقش أهمية هذا المفهوم من وجهة نظر المبرمج.

٢- لماذا نرغب أحياناً في تعريف إجراءات محددة داخل الأصناف الفرعية التي ترث أصناف رئيسة. بمعنى آخر ما أهمية الإجراءات المجردة والواجهات؟

٣- في أي من الأوجه يتشابه استخدام الواجهات مع مفهوم التوارث، ولماذا تعتمد عملية التطوير المركب من مكونات على استخدام الواجهات؟

٤- لماذا يجب تعريف الإجراء CalculateFee من النوع العام بدلاً من الخاص في أصناف Lease؟ لاحظ أن الإجراء لا يسند قيمة نتيجة بل يقوم بحسابها وإعادتها بناء على قيمة عرض المرسى. ناقش متى ننصح باستخدام المرور العام بدلاً من المرور الخاص للإجراءات؟

مشاريع الفصل

Projects

١- أضف الصنف الفرعي RowBoat إلى المشروع ١ الخاص بالصنف Boat حيث يجب تعريف الصفات الخاصة بهذا الصنف مثل الصفة Material التي تأخذ أحد القيم (Wood أو Fiberglass أو Inflatable) والصفة Oar Type التي تأخذ أحد القيم (Paddles أو Wood Oars أو Metal Oars). كما يجب تعريف إجراءات المرور المطلوبة والإجراء TellAboutSelf الذي سيستدعي الإجراء TellAboutSelf المعرف في الصنف الرئيس ويزيد عليه، ثم اكتب برنامج يختبر هذا الصنف. صمم نموذج Sequence Diagram الخاص ببرنامج الاختبار.

٢- أضف الصنف الفرعي PersonalWatercraft أسفل الصنف Powerboat. ما هي الصفات المتوقعة لهذا الصنف الجديد؟ اكتب شفرة هذا الصنف معرّفاً على الأقل صفتان وإجراءات المرور المطلوبة، وعرف إجراء إنشاء داخل هذا الصنف ليستقبل معاملات تسند إلى صفات الكائن. واكتب برنامج اختبار لإنشاء عدة كائنات من هذا الصنف ومن الصنف Powerboat.

٣- بالرجوع إلى المشروع رقم ٢، اشرح بالتفصيل (تتبع جميع الأوامر بالترتيب) ماذا سيحدث عندما يستدعي إجراء إنشاء الصنف الفرعي PersonalWatercraft إجراء إنشاء الصنف Powerboat. أضف الإجراء TellAboutSelf إلى الصنف الفرعي PersonalWatercraft الذي سيستدعي الإجراء TellAboutSelf المعرف في

الصنف الرئيس ويزيد عليه، ثم اشرح بالتفصيل (تتبع جميع الأوامر بالترتيب) ماذا سيحدث عند استدعاء الإجراء TellAboutSelf من كائن الصنف PersonalWatercraft.

٤- أضف الصنف الفرعي OneDayLease إلى المثال رقم ٣ الخاص بالصنف Lease الذي يستخدم للإيجار اليومي حيث إن إيجار اليوم يساوي قيمة ثابتة (غير معتمدة على عرض المرسى)، مع العلم أنه لا توجد حاجة لإضافة أي صفات على الصفات المعرفة بالصنف Lease. قم بتعريف الصنف OneDayLease أسفل الصنف Lease مبرمجاً كلاً من الواجهة ILeaseInterface والواجهة ICompanyInterface، ثم اكتب برنامجاً يختبر ذلك.

٥- تذكر التحقق الذي أضيف إلى الصنف Boat في التمرين رقم ١ وقم بتعريف استثناء خاص BoatException الذي يلقى عند إسناد قيم غير صحيحة، ثم اكتب برنامجاً يختبر ذلك.