الباب الخامس

نشر التطبيقات التي تتكون من ثلاث طبقات من الأصناف

DEPLOYING THE THREE-TIER APPLICATION

- الفصل الخامس عشر: تجميع تطبيقات الويندوز المكونة من ثلاث طبقات
- الفصل السادس عشر: تجميع تطبيقات الويب المكونة من ثلاث طبقات

الفصل الخامس عشر

تجميع تطبيقات الويندوز المكونة من ثلاث طبقات
ASSEMBLING A THREE-TIER WINDOWS APPLICATION

أهداف الفصل:

- مراجعة أسلوب تصميم البرامج من ثلاث طبقات من الأصناف.
- تجميع كل من أصناف مجال المشكلة وأصناف التعامل مع قواعد البيانات وأصناف واجهة الاستخدام.
- استخدام واجهة استخدام متعددة وإضافة بيانات إلى قواعد البيانات.
- استخدام واجهة الاستخدام مع العديد من أصناف مجال المشكلة لكي تتفاعل مع قواعد البيانات.

عزيزي القارئ ستتعلم في هذا الفصل الذي يمثل الفصل الأول من الباب الخامس كيف تجمع كلاً من أصناف مجال المشكلة وأصناف واجهة الاستخدام وأصناف التعامل مع البيانات لكي تكون نظاماً معلوماتياً كاملاً (Complete Information System) والذي يقدم واجهة استخدام ويقدم المهام التي طورت من أجلها ويقدم أيضاً مفهوم بقاء الكائنات الذي لا غنى عنه داخل تطبيقات العميل والخادم (Client-Server Applications).

لقد تعلمنا خلال الباب الرابع من هذا الكتاب كيف نطور أصناف التعامل مع البيانات التي تقدم مفهوم بقاء كائنات أصناف مجال المشكلة حيث يوجد أكثر من أسلوب لتخزين هذه الكائنات، ولكن يجب أن تعلم أن معظم التطبيقات تستخدم قواعد بيانات علاقية ولقد صممنا لكل صنف مجال مشكلة صنف مناظر للتعامل مع البيانات، ولقد لاحظنا أن كلاً من أصناف واجهة الاستخدام وأصناف مجال المشكلة لا تتغير مع تغيير نوع مخزن البيانات (ملفات تعاقبية أو قواعد البيانات) والذي يتغير فقط هو أصناف التعامل مع البيانات.

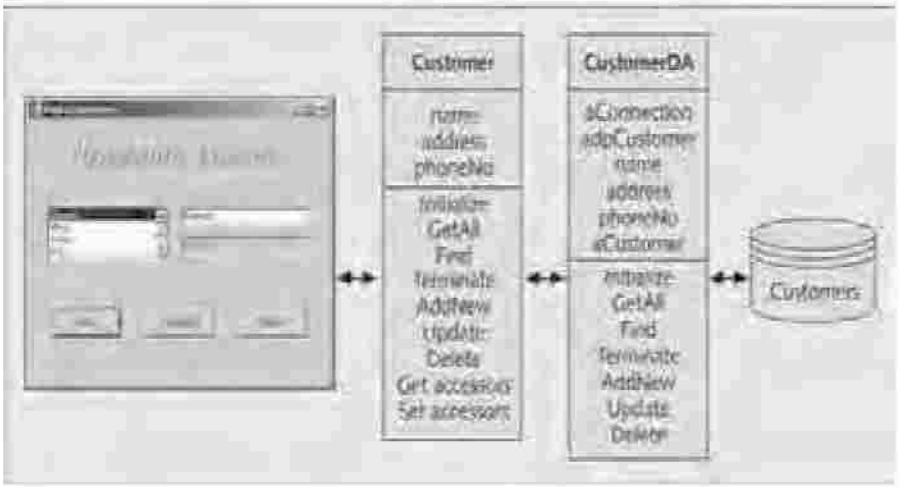
ولقد تعلمنا خلال الفصل الثالث كيف نطور أصناف واجهة الاستخدام التي تستجيب لأحداث المستخدم حيث يتم تطوير العديد من النماذج التي تتكون من عناصر معرفة سابقاً داخل لغة Visual Basic .NET مثل عناصر الأزرار والقوائم المنسدلة وقوائم الاختيار، ولقد لاحظنا كيف تتفاعل هذه الأصناف مع أصناف مجال المشكلة.

يهتم هذا الباب من الكتاب باستخدام جميع مفاهيم تطوير التطبيقات باستخدام تقنية الكائنات التي تعلمناها سابقاً وذلك لتكوين نظم معلوماتية متكاملة ، حيث يعرض الفصل الخامس عشر كيف نطور نظام شركة برادشو مارينا كتطبيق ويندوز مستقل، ثم يعرض الفصل السادس عشر كيف نطور تطبيقات الويب والتي تتكون أيضاً من ثلاث طبقات أصناف مختلفة.

مراجعة أسلوب تصميم النظم المكونة من ثلاثة أنواع من الأصناف

Reviewing Three-Tier Design

لقد تعلمنا خلال الفصل الخامس كيف نصمم التطبيقات باستخدام أسلوب ثلاثي الطبقات وهي: طبقة أصناف واجهة الاستخدام (GUI Classes)، وطبقة أصناف مجال المشكلة (BD Classes)، وطبقة أصناف التعامل مع البيانات (DA Classes)، حيث يتعامل المستخدم مع أصناف واجهة المستخدم التي بدورها تتفاعل مع أصناف مجال المشكلة وكائناتها التي بدورها تتفاعل مع أصناف التعامل مع البيانات التي تتولى تخزين كائنات أصناف مجال المشكلة داخل قواعد البيانات واسترجاعها، ويوضح الشكل رقم (١٥.١) كيفية التفاعل بين أنواع الطبقات الثلاثة للبحث عن بيانات عميل (Customer).



الشكل رقم (١٥.١). تصميم ثلاثي الطبقات لإيجاد العميل.

إن استخدام أسلوب الطبقات الثلاثة من الأصناف في تصميم النظم وتطويرها يؤدي إلى سهولة صيانة البرنامج، فإذا تم تغيير أصناف واجهة الاستخدام لا تتأثر كل من أصناف مجال المشكلة وأصناف التعامل مع البيانات، وبالمثل إذا تغيرت أصناف التعامل مع البيانات فإن كلاً من أصناف واجهة الاستخدام وأصناف مجال المشكلة لا تتأثر، فعلى سبيل المثال يمكن تغيير برنامج إدارة قواعد البيانات أو تحديثه بما يؤثر على أصناف التعامل

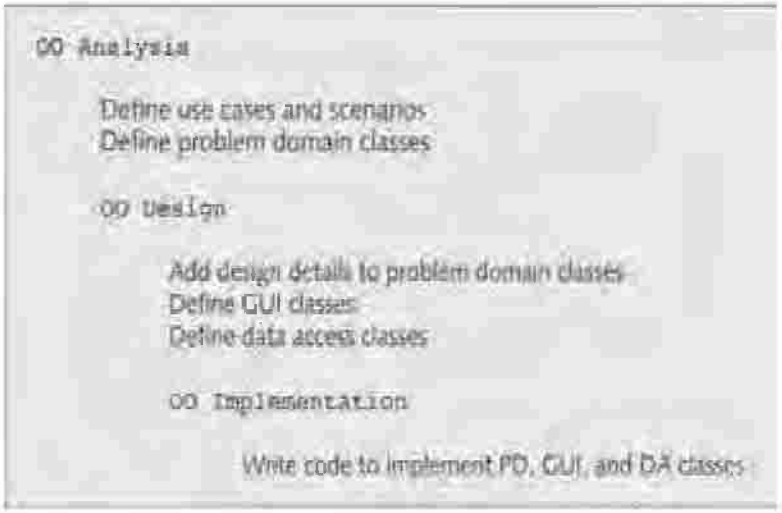
مع البيانات فقط، وكذلك يمكن تحديث واجهة الاستخدام أو تعديلها دون تغيير في كل من أصناف مجال المشكلة وأصناف التعامل مع البيانات. يقدم أيضاً أسلوب تصميم النظم وتطويرها باستخدام الطبقات الثلاثة من الأصناف طريقة سهلة لنشر التطبيق على أجهزة الشبكة، حيث يمكن تركيب أصناف واجهة الاستخدام على أجهزة الموظفين (العميل) وتركيب أصناف مجال المشكلة على خادم وتركيب قاعدة البيانات على خادم آخر، ومن ثم فإن هذا الأسلوب يتوافق مع تطبيقات العميل والخادم (Client-Server).

أسلوب تصميم النظم وتطويرها باستخدام الطبقات الثلاثة من الأصناف
Three-Tier and the Development Process

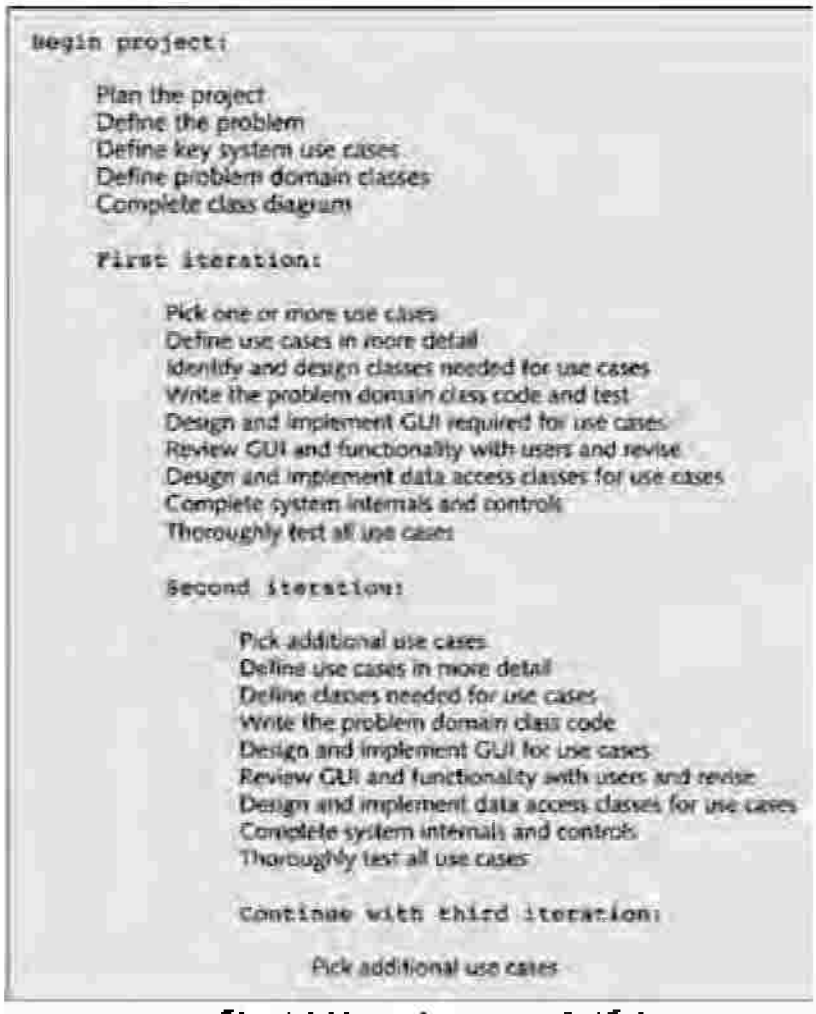
يقدم أسلوب التصميم الذي يعتمد على ثلاثة طبقات من الأصناف المذكورة سابقاً إطار عمل لعملية تطوير النظم المعتمدة على الكائنات التي تتم بشكل متكرر، حيث يتم إجراء بعض التحليل ثم إجراء بعض التصميم ثم إجراء بعض البرمجة وهكذا إلى أن يتم تطوير النظام كاملاً. تهتم مرحلة التحليل بتعريف متطلبات النظام التي يتم التعبير عنها بكل من حالات الاستخدام (Use Cases) وأصناف مجال المشكلة، ثم تحديد السيناريوهات المختلفة لحالات الاستخدام والتي تمثل وظائف النظام المقدمة للمستخدم، وكذلك يتم تعريف أصناف مجال المشكلة في البداية بتعريف الصفات الخاصة لكل صنف والإجراءات الأساسية لكل صنف، ثم إضافة تفاصيل أكثر أثناء كل دورة من دورات التحليل.

أما مرحلة التصميم فتهتم بإضافة تفاصيل أخرى عن كيفية تطوير النظم فعلياً حيث يهتم بكيفية تفاعل المستخدم مع النظام لتنفيذ حالات الاستخدام وذلك بتصميم أصناف واجهة الاستخدام، كما تهتم مرحلة التصميم بمفهوم بقاء كائنات أصناف مجال المشكلة (أي التعامل مع قاعدة بيانات مسؤولة عن تخزين بيانات النظام) وذلك بتصميم أصناف التعامل مع البيانات، ومن ثم فإن تصميم واجهة الاستخدام وتصميم قاعدة البيانات أشياء تخص مرحلة التصميم، أما مرحلة البرمجة فتهتم بكتابة أوامر كل من أصناف مجال المشكلة وأصناف واجهة الاستخدام وأصناف التعامل مع البيانات (انظر الشكل رقم ١٥,٢).

وكما أوضحنا سابقاً في نظام برادشو مارينا، فإن عملية التطوير تبدأ بالتخطيط للمشروع وتعريف المشكلة، ثم تعريف حالات الاستخدام وتحديد أصناف مجال المشكلة؛ ولذلك وصف الفصل الخامس متطلبات هذا النظام وقدمت حالات الاستخدام الخاصة به، ونموذج Class Diagram الخاص به، ويوضح الشكل رقم (١٥,٣) خطوات عملية تطوير النظم التكرارية ومراحلها.



الشكل رقم (١٥,٢). مراحل برمجة OO وتحليلها وتصميمها.



الشكل رقم (١٥,٣). خطوات عملية التطوير التكراري.

قد تم تحديد أصناف مجال المشكلة في نظام برادشو مارينا أولاً والمتعلقة بحالات الاستخدام، ثم قمنا بتطويرها باستخدام لغة Visual Basic .NET، ثم أضفنا إجراءات المرور وبعض الإجراءات الخاصة، ولقد أجرينا بعض الاختبارات بعد كل مرحلة، كما أضفنا مفهوم التحقق من صحة البيانات باستخدام تقنية معالجة الاستثناءات، ثم أضفنا أصنافاً فرعية وأضفنا الاستثناءات الخاصة، وأخيراً أضفنا علاقات الترابط بين هذه الأصناف دون الاهتمام إلى واجهة الاستخدام وطريقة بقاء الكائنات، ثم أجرينا بعض التحليل، ثم بعض التصميم، ثم بعض البرمجة.

وبمعنى آخر فإن فريق تطوير النظم لا يحاولون أن يطوروا جميع تفاصيل أصناف مجال المشكلة مرة واحدة، بل يقومون أولاً بتحديد الأصناف الهامة طبقاً لشكل حالات الاستخدام، ثم يقومون بتطويرها، كما يمكنهم تصميم بعض أصناف واجهة الاستخدام التي تمكن المستخدمين من التفاعل مع النظام وتطويرها، وكما رأينا خلال الباب الثالث من هذا الكتاب فإن أصناف واجهة الاستخدام تلعب دور برامج الاختبار؛ ولذلك يمكن ببساطة وضع هذه الأصناف مكان برامج الاختبار، ومن ثم يمكن عمل بعض التحليل وبعض التصميم لواجهة الاستخدام في كل دورة.

ولقد ربطنا بين كل من أصناف واجهة الاستخدام وأصناف مجال المشكلة في الفصل الحادي عشر، حيث رأينا كيف نضيف صنف واجهة الاستخدام دون تغيير في أصناف مجال المشكلة، كما أضفنا مفهوم بقاء الكائنات من خلال إضافة أصناف تحاكي أصناف التعامل مع البيانات، ثم حولنا اهتمامنا إلى تطبيق مفهوم الكائنات من خلال تصميم قاعدة بيانات علاقية وتطويرها وذلك لتخزين كائنات أصناف مجال المشكلة داخل جداول، حيث تم الربط بين هذه الجداول من خلال إضافة مفاتيح أجنبية، ومن ثم طورنا أصناف التعامل مع البيانات خلال هذه الدورة.

والآن وصلنا إلى مرحلة إتمام كل من أصناف مجال المشكلة وأصناف واجهة الاستخدام وأصناف التعامل مع البيانات التي تحقق حالات الاستخدام المصممة سابقاً، ومن ثم نريد أن نضيف بعض التحسينات التي تربط تلك الأنواع الثلاثة من الأصناف لإنتاج برنامج متكامل يمكن استخدامه، وهذا ما سوف نقوم بعمله خلال هذا الفصل.

وكما رأينا فإن عملية التطوير التكرارية تستمر حيث تبدأ المرحلة التالية على حالات استخدام أخرى، فعلى سبيل المثال فإن نظام برادشو مارينا بدأ بالتركيز على العملاء والمراكب التي يملكونها، ثم العمل على الأرصفة والمراسي، وأخيراً ركز على التقارير والاستعلامات، كما يمكننا تقسيم فريق التطوير بناء على تخصص كل فرد منهم خلال كل دورة حيث يمكن أن يهتم أحد أعضاء الفريق على تطوير أصناف مجال المشكلة وتحسينها، ويهتم الآخر بكتابة الأوامر المطلوبة لاختبار هذه الأصناف وتخزينها، بينما يركز عضو آخر على تطوير أصناف واجهة الاستخدام، وأخيراً يركز عضو آخر في تطوير أصناف البيانات، وطالما يتم التنسيق بين أعضاء الفريق بناء

على حالات الاستخدام وأصناف مجال للمشكلة، فإن ذلك يسهل تقسيم العمل عن اتباع أسلوب التطوير التكراري واستخدام أسلوب التصميم المعتمد على ثلاث طبقات من الأصناف.

تجميع أصناف مجال المشكلة والتعامل مع البيانات وواجهة الاستخدام
Combining a PD Class, a DA Class, and a GUI

يشمل المثال الأول في هذه الفقرة كلاً من صنف مجال المشكلة Customer وصنف التعامل مع البيانات CustomerDA اللذين قمنا بتطويرهما خلال الفصل الثالث عشر، كما يحتوي على صنف واجهة الاستخدام FindCustomer الذي قمنا بتصميمه خلال الفصل الحادي عشر حيث كان يقوم باسترجاع معلومات عميل محدد (انظر الشكل رقم ١٥,٤).

الشكل رقم (١٥,٤). واجهة المستخدم الرسومية FindCustomer من الفصل الحادي عشر.

مراجعة صنف مجال المشكلة Customer
Reviewing the Customer Problem Domain Class

يعتبر الصنف Customer هو صنف مجال المشكلة الأول الذي قمنا بتطويره (توجد منه عدة أجيال)، حيث بدأنا بتطويره خلال الفصل الخامس الذي كان يحتوي على دالة إنشاء تستقبل معاملات وعلى دوال مرور، ثم أضفنا له في الفصول التالية كلاً من الإجراء TellAboutSelf والعديد من إجراءات الإنشاء ومفهوم التحقق من صحة

البيانات باستخدام كيفية معالجة الاستثناءات ومميزات أخرى. ولقد طورنا علاقات الترابط بين هذا الصنف وأصناف أخرى خلال الفصل التاسع مثل علاقة الترابط بين الصنف Customer والصنف Boat، ومن ثم أضفنا بعض الصفات والإجراءات لتنفيذ هذه العلاقات.

أما صنف واجهة الاستخدام الذي قمنا بتطويره خلال الفصل الحادي عشر يتعامل فقط مع الصنف Customer، ولم نتعرض لمفهوم بقاء الكائنات خلال هذا الفصل إلى أن وصلنا إلى الفصل الثالث عشر الذي تعرض لمفهوم بقاء الكائنات بواسطة تطوير صنف التعامل مع البيانات CustomerDA الذي استخدم أولاً ملفات لتخزين بيانات العملاء، ثم استبدل الملفات بقواعد البيانات دون التعديل داخل الصنف Customer الذي لم يتأثر بأسلوب تخزين البيانات (وظيفة الصنف Customer DA فقط)، ويوضح الشكل رقم (١٥,٥) أوامر الصنف Customer.

تذكر أن هذا الصنف يحتوي على أربعة إجراءات يتم استدعاؤها من الصنف مباشرة (الإجراء Find، والإجراء GetAll، والإجراء Terminate، والإجراء Initialize)، وثلاثة إجراءات يتم استدعاؤها من كائنات هذا الصنف (الإجراء Delete، والإجراء Update، والإجراء AddNew)؛ ولذلك تتفاعل كائنات هذا الصنف مع صنف التعامل مع البيانات CustomerDA بتمرير ما يطلب منها إلى الإجراءات المناظرة داخل الصنف CustomerDA الذي يقوم بالتنفيذ الفعلي، ومن ثم فإن طلبات المستخدم من تخزين بيانات واسترجاعها ترسل من أصناف واجهة الاستخدام إلى أصناف مجال المشكلة (الصنف Customer) ومن ثم ترسل للتنفيذ فعلياً لدى أصناف التعامل مع البيانات (الصنف CustomerDA) كما رأينا خلال الفصل الثالث عشر.

مراجعة صنف التعامل مع البيانات CustomerDA
Reviewing the Customer Data Access Class

قدم الفصل الثالث عشر الصنف CustomerDA لتحقيق مفهوم بقاء كائنات الصنف Customer حيث تم تطوير عدة أجيال من هذا الصنف باستخدام أكثر من أسلوب في تخزين البيانات بداية من استخدام الملفات التعاقبية والملفات المفهرسة وأخيراً قواعد البيانات، كما استمر الفصل الرابع عشر في تطوير أصناف التعامل مع البيانات واستخدام أنواع بيانات أكثر تعقيداً حيث تم إصدار نسخة جديدة من الصنف CustomerDA الذي يربط كلاً من العميل بالمركب الذي يملكه، ولأن المثال الحالي يتعامل فقط مع بيانات العملاء فسيتم استخدام الصنف CustomerDA المطور خلال الفصل الثالث عشر والموضح في الشكل رقم (١٥,٦).

```
'Chapter 15 Example 1
' Customer Class with added methods for DA Class
' From Chapter 13
Imports System.Data
Imports System.Data.OleDb
Public Class Customer
    'attributes
    Private name As String
    Private address As String
    Private phoneNo As String

    '------------Begin Data Access Shared Methods-----
    Public Shared Sub Initialize()
        CustomerDA.Initialize()
    End Sub
    Public Shared Function Find(ByVal PhoneNo As String) As Customer
        Return CustomerDA.Find(PhoneNo)
    End Function
    Public Shared Function GetAll() As ArrayList
       Return CustomerDA.GetAll
    End Function
    Public Shared Sub Terminate()
        CustomerDA.Terminate()
    End Sub
    '------------End Data Access Shared Methods--------
    '------------Begin Data Access Instance Methods-----
    Public Sub AddNew()
        CustomerDA.AddNew(Me)
    End Sub
    Public Sub Update()
        CustomerDA.Update(Me)
    End Sub
    Public Sub Delete()
        CustomerDA.Delete(Me)
    End Sub
    '------------End Data Access Instance Methods-----
    '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
```

الشكل رقم (١٥,٥). تعريف صنف العميل الذي يتفاعل مع CustomerDA.

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

     property named CustomerName
    Public Property CustomerName() As String
        Get
            Return name
        End Get
        Set(ByVal aName As String)
            name = aName
        End Set
    End Property
    '-----------------Begin Constructors--------------
    'default constructor
    Public Sub New()
    End Sub
    'constructor () parameters)
    Public Sub New(ByVal aName As String, _
            ByVal anAddress As String, ByVal aPhoneNo As String)
        name = aName
        address = anAddress
        phoneNo = aPhoneNo

    End Sub
    '-----------------End Constructors--------------
End Class
```

تابع الشكل رقم (١٠,٥).

```
'Chapter 15 Example 1
' Chapter 13 - Example ------DA classes using dbms
' for single table example with CustomerTable
Imports System.Data.OleDb
Imports System.Collections
Public Class CustomerDA
    Shared customers As New ArrayList() ' Customer references
    Shared aCustomer As Customer

    'Declare a connection. Eliminate the hard-coded path to
    'the database by putting the database file in the
    'project's Bin folder

    Shared cnnCustomer As New
    OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0; " & _
    "Data Source=CustomerDatabase.mdb")

    'Declare variables for Customer attribute values
    Shared name, address, phoneno As String

    'Initialize Method (chapter 13 example)
    'no connection passed as reference
    Public Shared Sub Initialize()
        Try
            ' Try to open the connection
            cnnCustomer.Open()
        Catch e As Exception
            Console.WriteLine(e.ToString)
        End Try
    End Sub

    ' Terminate Method (chapter 13 example)
    Public Shared Sub Terminate()
        Try
            cnnCustomer.Close()
            cnnCustomer = Nothing
        Catch e As Exception
            Console.WriteLine(e.Message.ToString)
        End Try
    End Sub

    ' AddNew Method --Throws DuplicateException if exists
    Public Shared Sub AddNew(ByVal aCustomer As Customer)

        ' Get customer information
        name = aCustomer.GetName
        address = aCustomer.GetAddress
        phoneno = aCustomer.GetPhoneNo
```

الشكل رقم (١٥,٦). تعريف صنف CustomerDA للمثال رقم ١.

```
        ' Declare a string SQL statement
        Dim sqlInsert As String = "INSERT INTO customerTable " & _
        "VALUES ('" & name & "', '" & address & "', '" & phoneno _
                        & "')"
        Dim adpcustomer As New OleDbDataAdapter()

        Try
            Dim c As Customer = Find(phoneno)
            Throw New DuplicateException(" Customer Exists ")
        Catch e As NotFoundException
            Try     ' Assign Insert Commands and Execute
                adpcustomer.InsertCommand = _
                                New OleDbCommand(sqlInsert)
                adpcustomer.InsertCommand.Connection = _
                                cnnCustomer
                adpcustomer.InsertCommand.ExecuteNonQuery()
            Finally
            End Try
        End Try
    End Sub

    ' Find Method--Throws NotFoundException if Not Found
    Public Shared Function Find(ByVal key As String) As Customer

        Dim acustomer As New Customer()
        acustomer = Nothing
        Dim dsCustomer As New DataSet()
        Try
            ' Define the SQL statement using the phoneno key
            Dim sqlQuery As String = _
                  "SELECT Name, Address, PhoneNO " & _
            "FROM CustomerTable WHERE phoneNo = '" & key & "'"
            Dim adpCustomer As New _
             OleDbDataAdapter(sqlQuery, cnnCustomer)
            adpCustomer.Fill(dsCustomer, "CustTable")
            If dsCustomer.Tables("CustTable").Rows.Count > 0 Then
               Dim custRow As DataRow
               custRow = _
                            dsCustomer.Tables("custTable").Rows(0)
               name = custRow.Item("Name")
               address = custRow.Item("address")
               phoneno = custRow.Item("phoneno")
               acustomer = _
                            New Customer(name, Address, phoneno)
            Else
               Throw New NotFoundException("Not Found")
            End If
            dsCustomer = Nothing
        Catch e As OleDb.OleDbException
            Console.WriteLine(e.Message.ToString)
        End Try
        Return acustomer

    End Function
```

تابع الشكل رقم (١٥،٦).

```
' GetAll Method
Public Shared Function GetAll() As ArrayList
    Dim dsCustomer As New DataSet()
    Dim sqlQuery As String = "SELECT Name, Address, PhoneNo " _
            & "FROM CustomerTable"
    Try
        Dim adpCustomer As New _
          OleDbDataAdapter(sqlQuery, conCustomer)
        adpCustomer.Fill(dsCustomer, "CustTable")
        If dsCustomer.Tables("CustTable").Rows.Count > 0 Then
            Dim dsRow As DataRow
            ' Clear the array list
            customers.Clear()
            For Each dsRow In dsCustomer.Tables("CustTable").Rows
                name = dsRow("Name")
                address = dsRow("Address")
                phoneno = dsRow("PhoneNo")
                Dim aCustomer As New _
                 Customer(name, address, phoneno)
                customers.Add(aCustomer)
            Next
        Else
            ' No records in database
        End If
        dsCustomer = Nothing
    Catch e As Exception
        Console.WriteLine(e.ToString)
        Throw New NotFoundException("Not Found")
    End Try
    Return customers

End Function

' Update Method
Public Shared Sub Update(ByVal aCustomer As Customer)

    ' Get Customer information
    name = aCustomer.GetName
    address = aCustomer.GetAddress
    phoneno = aCustomer.GetPhoneNo

    Dim sqlUpdate As String = "Update CustomerTable " & _
     "SET Name = '" & name & "', Address = '" & address & _
            "'," & "PhoneNo = '" & phoneno & "' " & _
     "WHERE PhoneNo = '" & phoneno & "'"
    Try
        Dim c As Customer = Customer.Find(phoneno)
        Dim adpcustomer As New OleDbDataAdapter()
        adpcustomer.UpdateCommand = New _
                    OleDbCommand(sqlUpdate)
        adpcustomer.UpdateCommand.Connection = conCustomer
        adpcustomer.UpdateCommand.ExecuteNonQuery()
    Catch e As Exception
        Console.WriteLine(e.ToString)
    End Try
End Sub
```

تابع الشكل رقم (١٥,٦).

```
    'Delete Method--Throws NotFoundException if not found
    Public Shared Sub Delete(ByVal aCustomer As Customer)
        Dim phoneno, sqlDelete As String
        phoneno = aCustomer.GetPhoneNo
        sqlDelete = "DELETE FROM CustomerTable " & _
          "WHERE PhoneNo = '" & phoneno & "'"
        Try
            Dim c As Customer = Customer.Find(phoneno)
            Dim adpCustomer As New OleDbDataAdapter()
            adpCustomer.DeleteCommand = New _
                        OleDbCommand(sqlDelete)
            adpCustomer.DeleteCommand.Connection = pubCustomer
            adpCustomer.DeleteCommand.ExecuteNonQuery()
        Catch e As Exception
            Console.WriteLine(e.ToString)
        End Try
    End Sub
End Class
```

تابع الشكل رقم (١٥,٦).

لقد شرحنا الصنف CustomerDA بالتفصيل خلال الفصل الثالث عشر حيث يحتوي على الإجراءات اللازمة والمذكورة سابقاً للصنف Customer للإضافة والاسترجاع والتعديل وحذف عميل من قاعدة البيانات وإلقاء استثناء عند حدوث خطأ ما عند تنفيذ الإجراء.

لا نحتاج أن نجري أي تغيير على الصنف CustomerDA حيث يمكن استخدامه كما هو داخل المثال الحالي، مع العلم أننا وضعنا ملف قاعدة البيانات CustomerDatabase داخل المجلد bin لعدم كتابة مسار الملف صراحة داخل أوامر الصنف CustomerDA، إذ يكوّن كل من صنف مجال المشكلة Customer وصنف التعامل مع البيانات CustomerDA وصنف واجهة الاستخدام التركيب اللازم لتطوير تطبيق يعتمد على ثلاث طبقات من الأصناف بحيث تستدعي أوامر أصناف واجهة الاستخدام إجراءات أصناف مجال المشكلة التي بدورها تستدعي إجراءات أصناف التعامل مع البيانات التي بدورها تقوم بتنفيذ الأوامر وإعادة الناتج إلى أصناف مجال المشكلة (الصنف Customer) والتي بدورها تعيدها إلى أصناف واجهة الاستخدام. والآن إن كلاً من الصنف Customer والصنف CustomerDA متكاملين وجاهزين لاستخدامهما داخل برنامج المثال الحالي.

تعديل صنف واجهة الاستخدام FindCustomer

Updating the FindCustomer GUI

يعتمد صنف واجهة استخدام هذا المثال على صنف واجهة الاستخدام FindCustomer المقدم خلال الفصل الحادي عشر (المثال رقم ٤) الذي كان يتفاعل مع الصنف Customer والصنف CustomerDA (الذي كان يحاكي عملية تخزين البيانات واسترجاعها). يعتبر الصنف FindCustomer جاهزاً تقريباً لكي يعمل مع كل من الصنف Customer والصنف CustomerDA، ولكن يحتاج فقط إلى بعض التعديلات البسيطة.

يبدأ الصنف FindCustomer كما فعلنا سابقاً بتوصيف متغير إشارة من الصنف Customer (aCustomer) ومصفوفة لمتغيرات إشارة من الصنف Customer (customers) لكي يحتوي على البيانات العائدة من قاعدة البيانات هكذا:

```
'Chapter 15 Example 1
Imports System.Collections
Public Class FindCustomer
    Inherits System.Windows.Forms.Form

    'Declare customer reference variable
    Private aCustomer As Customer

   'Declare array of customers (no longer simulated)
   Private customers As New ArrayList()
```

وعلى أي حال، لا يحتوي صنف واجهة الاستخدام FindCustomer على قائمة رئيسة مثل مثال الفصل الحادي عشر؛ ولذلك يجب أن يتم تهيئة قاعدة البيانات داخل هذا الصنف بدلاً من القائمة الرئيسة؛ ولذلك يجب استدعاء الإجراء Initialize من الصنف Customer داخل إجراء إنشاء الصنف FindCustomer الذي بدوره سوف يستدعي الإجراء Initialize من الصنف CustomerDA المسؤول فعلاً عن تهيئة قاعدة البيانات، وكما فعلنا سابقاً يجب إضافة أوامر استرجاع أسماء العملاء من قاعدة البيانات وإسنادها للقائمة داخل إجراء الإنشاء وذلك بعد تهيئة الاتصال بقاعدة البيانات هكذا:

```
'Add statements to generated code to:
'initialize the database and populate the list box
Public Sub New()
    MyBase.New()

   'This call is required by the Windows Form Designer.
   InitializeComponent()
   'Add any initialization after the InitializeComponent() call

   'Initialize the database
   Customer.Initialize()
   'Populate the customer list box
   PopulateListBox()
End Sub
```

يستدعي الإجراء PopulateListBox الإجراء GetAll من الصنف Customer لاسترجاع جميع كائنات الصنف Customer من قاعدة البيانات، ولاحتمال عدم وجود كائنات داخل قاعدة البيانات فلقد تم استدعاء الإجراء GetAll داخل التركيب Try-Catch.

إن إسناد القيمة صفر للفهرس المختار (SelectedIndex) يمنع حدوث خطأ أثناء التشغيل إذا ضغط المستخدم على الزر Find قبل اختيار عنصر من القائمة كما هو واضح في الأوامر التالية:

```
Private Sub PopulateListBox()
    'Get the customers from the database
    'Add try-catch block to trap NotFoundException
    Try
        customers = Customer.GetAll()
        'Add the name of each customer to the list
        Dim i As Integer
        For i = 0 To customers.Count - 1
            aCustomer = customers(i)
            lstCustomer.Items.Add(aCustomer.GetName())
        Next
        'Set the selected index to 0
        lstCustomer.SelectedIndex = 0
    Catch nfe As NotFoundException
        MessageBox.Show("No records found in database")
    End Try
End Sub
```

لاحظ أن هذه الأوامر لم تتغير عن أوامر مثال الفصل الحادي عشر عدا استدعاء الإجراء GetAll بدلاً من إنشاء كائنات الصنف Customer داخل الأوامر. لا يتطلب إجراء معالجة الحدث Find أي تغيير عن مثال الفصل الحادي عشر حيث يتم استخدام فهرس القائمة لاسترجاع بيانات العميل، ولاحظ أن الإجراء Find لم يتم استدعاؤه من الصنف customers لأن جميع الكائنات تم استرجاعها وتخزينها داخل المصفوفة Customer بواسطة الإجراء GetAll سابقاً:

```
Private Sub btnFind_Click _
(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles btnFind.Click
    Dim i As Integer
    'Identify the selected index in the list box
    i = lstCustomer.SelectedIndex
    'Find this customer in the array list
    aCustomer = customers(i)
    'Retrieve and display this customer's address and phone
    txtCustomerAddress.Text = aCustomer.GetAddress()
    txtCustomerPhone.Text = aCustomer.GetPhoneNo()
End Sub
```

أما إجراء معالجة الحدث Update فيتم استدعاء الإجراء Update من الصنف Customer الذي يقوم بتعديل بيانات العميل داخل قاعدة البيانات، ولأن بيانات العميل ربما لا توجد داخل قاعدة البيانات فيجب استدعاء هذا الإجراء داخل التركيب Try-Catch.

```
Private Sub btnUpdate_Click _
(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles btnUpdate.Click
    Dim i As Integer
    'Identify the selected index in the list box
    i = lstCustomer.SelectedIndex
    'Find this customer in the array list
    aCustomer = customers(i)
    'Set this customer's address to the value entered by
    'the user.
    aCustomer.SetAddress(txtCustomerAddress.Text)
    'Update this customer in the database
    'Use try-catch block to trap possible "not found" error
    Try
        aCustomer.Update()
        MessageBox.Show("Customer Updated")
    Catch ee As NotFoundException
        MessageBox.Show("This customer could not be found")
    End Try

    'Clear address and phone text fields
    txtCustomerAddress.Text = ""
    txtCustomerPhone.Text = ""
   End Sub
```

تذكر أيضاً أن رقم هاتف العميل يستخدم كمفتاح أساسي داخل قاعدة البيانات؛ ولذلك لا يمكن تعديله، ومن ثم لا يمكن تعديل هذه الصفة، كما تم إسناد القيمة False للصفة Enabled الخاصة بصندوق نص رقم الهاتف لعدم إمكانية تغيير رقم الهاتف كما هو واضح في الشكل رقم (١٥,٧).

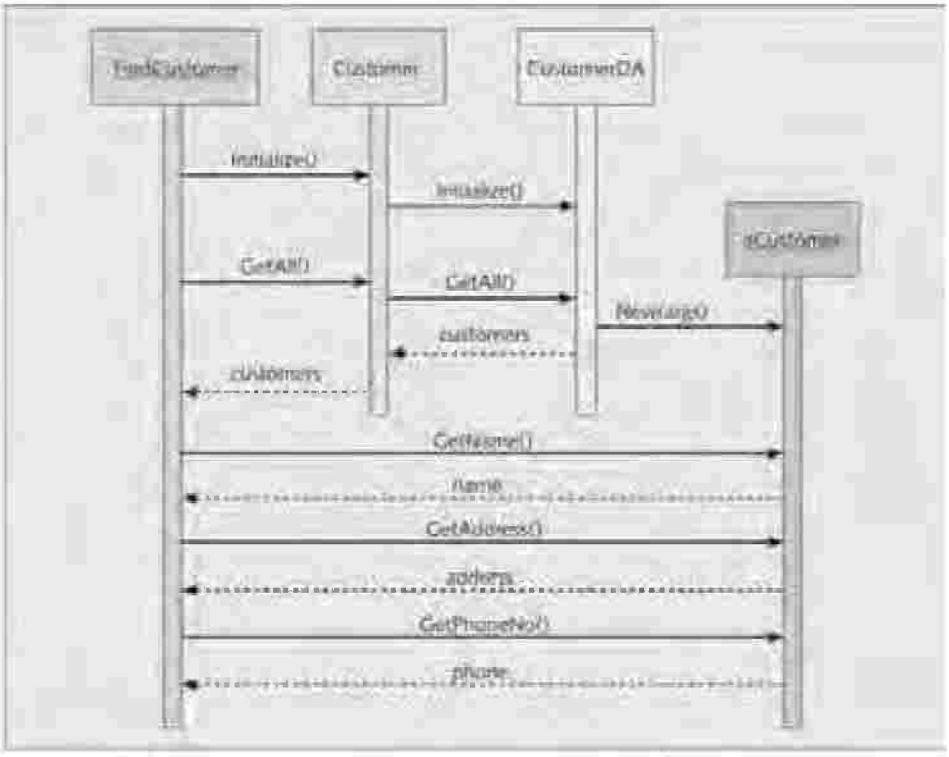
أما التعديل الأخير داخل الصنف FindCustomer فهو إجراء معالجة الحدث Close حيث يجب غلق الاتصال بقاعدة البيانات داخل هذا الإجراء لعدم وجود قائمة رئيسة لهذا البرنامج:

```
Private Sub btnClose_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnClose.Click
    'Add call to terminate database for Chapter 15
    Customer.Terminate()
    Me.Close()
End Sub
```

الشكل رقم (١٥،٧). واجهة المستخدم الرسومية FindCustomer للمثال رقم ١.

والآن قد اكتمل البرنامج الذي يتكون من ثلاث طبقات من الأصناف حيث يطلب صنف واجهة الاستخدام FindCustomer مصفوفة العملاء من صنف مجال المشكلة Customer الذي بدوره يطلب المصفوفة من صنف التعامل مع البيانات CustomerAD الذي يتصل بقاعدة البيانات ليسترجع بيانات عميل تلو الآخر وإنشاء كائنات مناظرة وإضافتها داخل المصفوفة، ثم إرجاع هذه المصفوفة إلى الصنف Customer الذي يعيدها إلى الصنف FindCustomer، والآن يستطيع الصنف FindCustomer أن يسترجع بيانات أي عميل من المصفوفة مباشرة باستخدام إجراءات المرور وفهرس المصفوفة. يوضح الشكل رقم (١٥،٨) نموذج Sequence Diagram الذي يوضح التفاعل بين الطبقات الثلاثة من الأصناف.

تذكر أن الصنف CustomerDA يلقي كلاً من الاستثناء NotFoundException والاستثناء DuplicateException؛ ولذلك يجب أن يتم إضافتهما داخل هذا البرنامج ليعمل بنجاح (ارجع إلى الصنف CustomerDA داخل الفصل الثالث عشر).

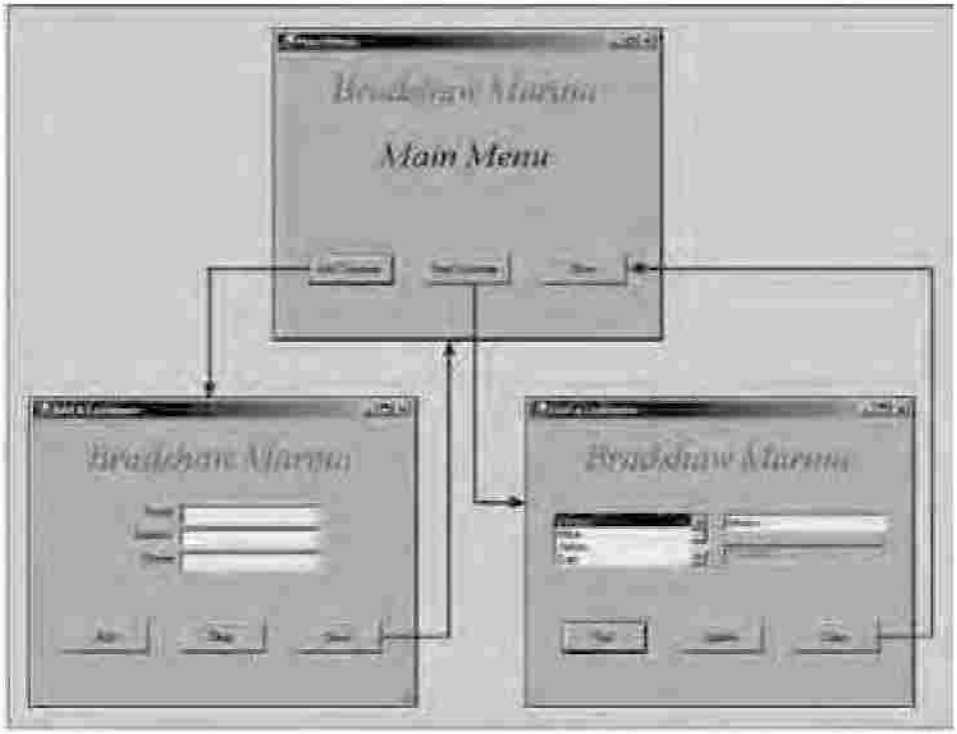


الشكل رقم (١٥,٨). النموذج العام FindCustomer الذي يعرض تصميم ثلاثي الطبقات.

استخدام واجهات متعددة وإضافة بيانات إلى قاعدة البيانات
Using Multiple GUIs and Adding an Instance to the Database

يستخدم المثال التالي واجهات استخدام متعددة، كما يضيف البيانات (كائنات أصناف مجال المشكلة) إلى قاعدة البيانات باستخدام صنف التعامل مع البيانات. يعتمد هذا المثال على المثال رقم ٤ الموضح داخل الفصل الحادي عشر حيث تسمح شاشة القائمة الرئيسة للمستخدم أن يبحث عن عميل أو أن يضيف عميلاً جديداً. سوف يتم تعديل الصنف FindCustomer المذكور في المثال رقم ١ قليلاً لكي يتوافق مع القائمة الرئيسة لهذا المثال، كما سيتم تعديل كل من الصنف AddCustomer وصنف القائمة الرئيسة MainMenu المذكورين في الفصل الحادي عشر لكي يعملان مع صنف التعامل مع البيانات. كما يجب إضافة كل من الاستثناء NotFoundException والاستثناء DuplicateException إلى البرنامج. يوضح الشكل رقم (١٥,٩) أصناف واجهة الاستخدام وهي: الصنف MainMenu، والصنف FindCustomer، والصنف AddCustomer.

لاحظ أن كلاً من الصنف Customer والصنف CustomerDA المذكورين في المثال السابق لا يحتاجان إلى أي تعديل كي يتم استخدامهما داخل هذا المثال، حيث إنهما يشتملان بالفعل على إمكانية إضافة عميل جديد إلى قاعدة البيانات، ولذلك سيتم إضافتهما دون تعديل هذا المثال.



الشكل رقم (١٥,٩). واجهة المستخدم الرسومية MainMenu و FindCustomer و AddCustomer للمثال رقم ٢.

مراجعة القائمة الرئيسة

Reviewing the MainMenu GUI

يشبه صنف واجهة الاستخدام MainMenu الصنف المذكور داخل الفصل الحادي عشر، ولكن نحتاج أن نجري بعض التعديلات، وبما أن أحد مهام صنف القائمة الرئيسة هي بدء الاتصال بقاعدة البيانات وإنهائها، وبما أن البرنامج سيحتوي على صنف التعامل مع البيانات فسيتم استدعاء الإجراء Initialise والإجراء Terminate من صنف مجال المشكلة Customer بدلاً من الصنف CustomerData حيث يتم استدعاء الإجراء Initialise داخل إجراء الإنشاء.

```
Public Sub New()
    MyBase.New()
    'This call is required by the Windows Form Designer.
    InitializeComponent()

    'Initialise the database
    Customer.Initialise()

End Sub
```

وبالمثل سيتم استدعاء الإجراء Terminate من الصنف Customer داخل إجراء معالجة الحدث Close btnClose_Click() لغلق الاتصال بقاعدة البيانات هكذا:

```
Private Sub btnClose_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnClose.Click
    'Terminate connection to the database
    Customer.Terminate()
    Me.Close()
End Sub
```

يبقى كل من الإجراء btnAdd_Click والإجراء btnFind_Click لصنف واجهة الاستخدام MainMenu كما هما دون تعديل، أي مسؤولين عن إنشاء نوافذ عميل جديد وإظهارها والبحث عن عميل موجود) هكذا:

```
Private Sub btnAdd_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnAdd.Click
    'Hide the main menu form
    Me.Hide()
    'Create and display AddCustomer form
    Dim frmAddCustomerGUI = New AddCustomer()
    frmAddCustomerGUI.ShowDialog()
    'Make the main menu form visible
    Me.Show()
End Sub

Private Sub btnFind_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnFind.Click
    'Hide the main menu form
    Me.Hide()
    'Create and display FindCustomer form
    Dim frmFindCustomerGUI = New FindCustomer()
    frmFindCustomerGUI.ShowDialog()
    'Make the main menu form visible
    Me.Show()
End Sub
```

مراجعة صنف واجهة الاستخدام AddCustomer

Reviewing the AddCustomer GUI

لقد قدمنا الصنف AddCustomer بداية مع الصنف FindCustomer داخل الفصل الحادي عشر، ولكن يجب إجراء بعض التعديلات على هذا الصنف لكي يعمل مع صنف التعامل مع البيانات. تنحصر التعديلات المطلوبة داخل الإجراء btnAddClick حيث يجب إضافة التركيب Try-Catch واستدعاء الإجراء AddNew من صنف التعامل مع البيانات CustomerDA بدلاً من الصنف CustomerData.

يستقبل صنف واجهة الاستخدام AddCustomer بيانات المستخدم المراد إضافته، ثم التحقق من صحة البيانات المدخلة، ثم إنشاء كائن من الصنف Customer (aCustomer)، ثم استدعاء الإجراء AddNew من خلال الكائن aCustomer لإضافته في قاعدة البيانات هكذا:

```
Private Sub btnAdd_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnAdd.Click
    'Get values from the text boxes
    customerName = txtName.Text
    customerAddress = txtAddress.Text
    customerPhone = txtPhone.Text
    'Validate that the user has entered values for name,
    'address and phone
    If customerName.Length = 0 Or _
        customerAddress.Length = 0 Or _
        customerPhone.Length = 0 Then
            MessageBox.Show("Please Enter All Data")
    Else
        'Data is valid -- create Customer instance
        aCustomer = New Customer(customerName, _
                    customerAddress, customerPhone)
        'Ask the new customer instance to add itself
        'Use try catch block in case duplicate exception
        'is thrown by AddNew() method
           Try
               aCustomer.AddNew()
               MessageBox.Show("Customer Added")
               'Clear the form
               ClearForm()
           Catch ee As DuplicateException
              MessageBox.Show _
                  ("That customer already exists in the database")
               'Do not clear the form so user can see and fix the error
               txtName.Focus()
           End Try
    End If
End Sub
```

تذكر أن الإجراء AddNew المعرف داخل الصنف Customer يستخدم أمر SQL INSERT لإضافة سجل جديد لقاعدة البيانات، ولاحظ وجود التركيب Try-Catch لاحتمال استقبال الاستثناء DuplicateException الذي ربما يحدث عند إضافة بيانات عميل جديد يملك رقم هاتف موجود مسبقاً داخل قاعدة البيانات، فإذا تم استقبال هذا الاستثناء فسيتم إظهار رسالة للمستخدم. لا نحتاج إلى أي تغيير إضافي على تعريف الصنف AddCustomer المذكور داخل الفصل الحادي عشر.

تعديل صنف واجهة الاستخدام FindCustomer
Updating the FindCustomer GUI

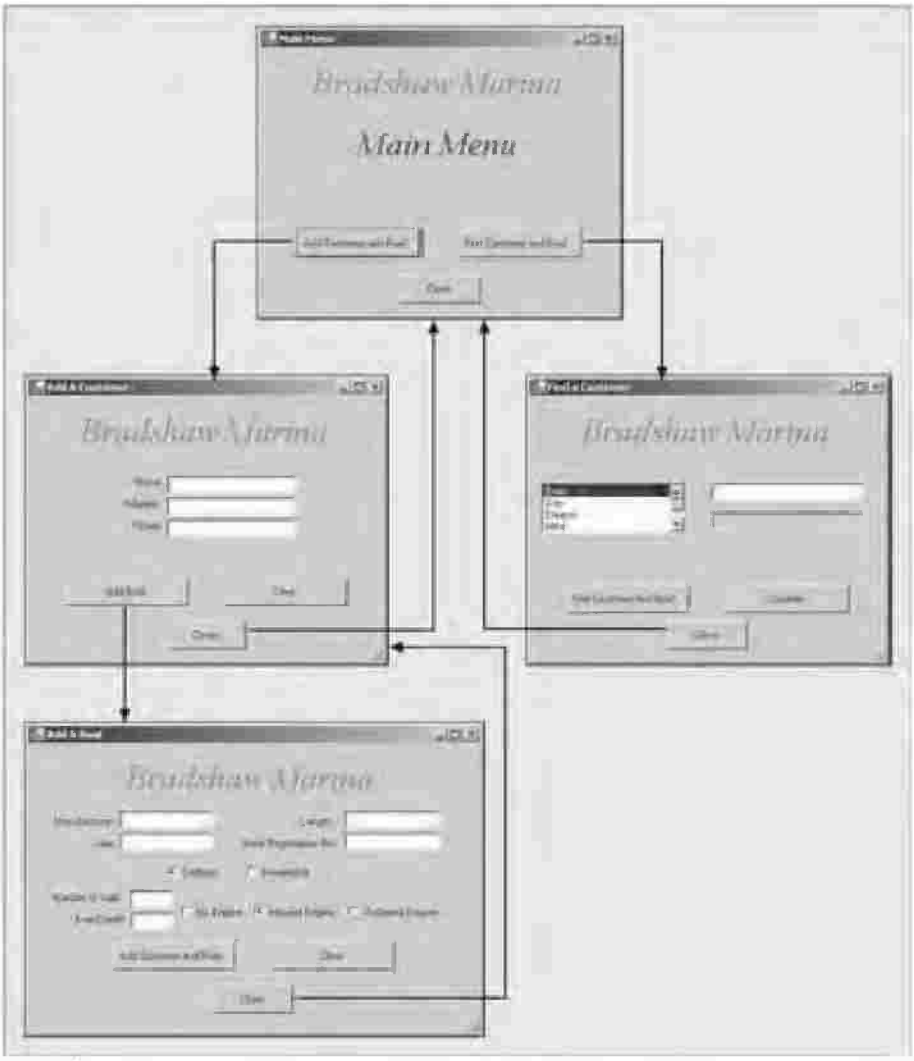
إن صنف واجهة الاستخدام FindCustomer المعرف داخل المثال السابق لا يحتاج إلى أي تعديل غير الذي يمكنه من العمل مع الصنف MainMenu. تذكر أن تهيئة الاتصال بقاعدة البيانات وإنهائه مسؤولية صنف واجهة الاستخدام MainMenu بدلاً من الصنف FindCustomer ؛ ولذلك يجب حذف كل من استدعاء الإجراء Initialize والإجراء Terminate من تعريف الصنف FindCustomer.

استخدام واجهة استخدام متعددة مع أصناف مجال مشكلة متعددة
Using GUIs with Multiple Problem Domain Classes

سوف نتعلم في هذه الفقرة كيف نطور برنامجاً يحتوي على عدة واجهات استخدام (نوافذ) وعدة أصناف مجال مشكلة وعدة أصناف تتعامل مع البيانات، حيث يحتوي على عدة واجهات الاستخدام (MainMenu و FindCustomer و AddCustomer و AddBoat) التي تم تطويرها خلال الفصل الحادي عشر، وكذلك يحتوي على كل من الأصناف Customer و Boat و CustomerDA و BoatDA و CustomerAndBoatConnect التي تم تطويرها خلال الفصل الرابع عشر عدا وجود بعض الاختلافات في عناوين الأزرار، حيث يوضح الشكل رقم (١٥.١٠) واجهات استخدام هذا البرنامج وعلاقة بعضها مع بعض، مع العلم أن هذه الأصناف تحتاج إلى تعديلات طفيفة لكي تعمل سوياً داخل البرنامج الذي يتكون من ثلاثة أنواع من الأصناف.

مراجعة الصنف Customer والصنف Boat وأصناف التعامل مع البيانات من الفصل الرابع عشر
Reviewing the Customer and Boat PD and DA Classes from Chapter 14

سوف يتم استخدام أصناف الفصل الرابع عشر (الصنف Customer، والصنف Boat، والصنف CustomerAndBoatDatabaseConnect) خلال هذا المثال دون أدنى تعديل. يحتوي الصنف Customer والصنف Boat على صفات وإجراءات مسؤولة عن ربط العميل بالمركب الذي يملكه، وتذكر أن إقامة علاقة ربط بين كائن الصنف Customer وكائن الصنف Boat يتطلب إضافة متغير إشارة من الصنف Customer داخل الصنف Boat وإضافة متغير إشارة من الصنف Boat داخل الصنف Customer، وأيضاً إضافة إجراءات مرور تسمح بالتعامل مع تلك الصفات. إن درجة العلاقة بين العميل والمركب تساوي "واحد-إلى-واحد"، أي كل عميل يملك مركباً واحداً فقط وكل مركب يملكه عميل واحد فقط، ولقد قدمنا هذه العلاقة خلال الفصل التاسع وقمنا ببرمجتها خلال الفصل الرابع عشر حيث يستطيع صنف التعامل مع البيانات CustomerDA أن يصل إلى بيانات المركب الذي يملكه، ومن ثم يتأكد من إضافة بياناته (أو حذفه) من قاعدة البيانات عند إضافة بيانات عميل (أو حذفها).



الشكل رقم (١٥,١٠). واجهة المستخدم الرسومية MainMenu و FindCustomer و AddCustomer للمثال رقم ٣.

يستخدم المثال الأخير في هذا الفصل قاعدة بيانات تسمى CustomerAndBoatDatabase.mdb التي تحتوي على جدولين وهما: الجدول CustomerTable، والجدول BoatTable. يحتوي الجدول BoatTable على المفتاح الأجنبي CustomerPhone الذي يسهل عملية الربط بين الجدولين، كما يجب أن نلاحظ أن قاعدة البيانات هذه لم تتغير عن الفصل الرابع عشر.

تعديل صنف واجهة الاستخدام MainMenu
Modifying the MainMenu GUI

كما رأينا خلال المثال السابق، فإن صنف واجهة الاستخدام MainMenu يفتح ويغلق الاتصال بقاعدة البيانات التي تحتوي على جدولين وهما: الجدول CustomerTable، والجدول BoatTable. وتذكر أنه يوجد هناك

اتصال مشترك لهذين الجدولين، وبالمثل سيحتوي إجراء إنشاء الصنف MainMenu على أمر استدعاء الإجراء Initialize المعرف داخل هذا الاتصال المشترك CustomerAndBoatDatabaseConnect وذلك لإنشاء اتصال مشترك ثم يتم إرسال هذا الاتصال كمعامل للإجراء Initialize الخاص بكل من الصنف Customer والصنف Boat.

```
'Chapter 15 Example 3
Imports System.Data.OleDb
Imports System.Collections

Public Class MainMenu
    Inherits System.Windows.Forms.Form

    'Added for Chapter 15
    'Declare variable for database connection
    Dim aConnection As OleDbConnection

    'Add statements to hidden code to initialize the
    'connection to the database

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call
        aConnection = CustomerAndBoatDatabaseConnect.Initialize()
        Customer.Initialize(aConnection)
        Boat.Initialize(aConnection)

    End Sub
```

يغلق الصنف MainMenu الاتصال بقاعدة البيانات داخل الإجراء btnClose_Click هكذا:

```
Private Sub btnClose_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnClose.Click
       'Terminate added for Chapter 15
       Customer.Terminate()
       Boat.Terminate()
       Me.Close()
End Sub
```

لا نحتاج إلى إجراء أي تعديلات أخرى حيث إن كلاً من الإجراء btnFind_Click والإجراء btnAdd_Click مسؤولين عن إنشاء النموذج المناسب وإظهاره.

```
Private Sub btnAdd_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnAdd.Click
        'Hide the main menu form
        Me.Hide()
        'Create and display the AddCustomer form
        Dim frmAddCustomerGUI = New AddCustomer()
        frmAddCustomerGUI.ShowDialog()
        'Make the main menu form visible
        Me.Show()
End Sub

Private Sub btnFind_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnFind.Click
        'Hide the main menu form
        Me.Hide()
        'Create and display the FindCustomer form
        Dim frmFindCustomerGUI = New FindCustomer()
        frmFindCustomerGUI.ShowDialog()
        'Make the main menu form visible
        Me.Show()
End Sub
```

تعديل صنف واجهة الاستخدام FindCustomer

Modifying the FindCustomer GUI

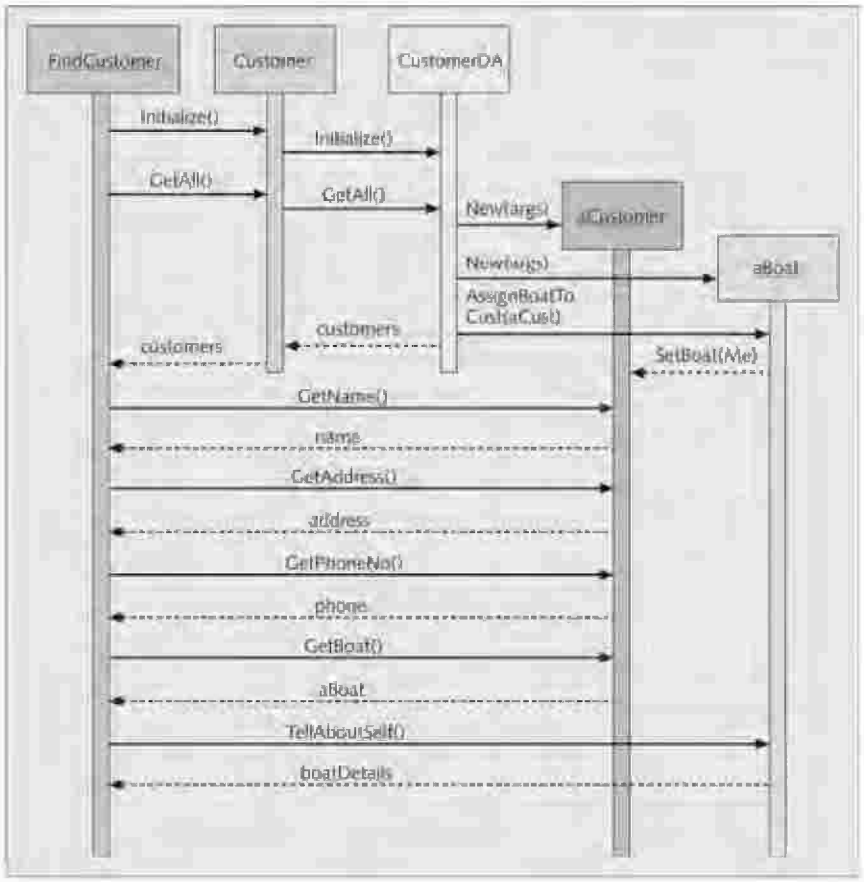
يتطلب صنف واجهة الاستخدام FindCustomer تعديلات طفيفة جداً عن المثال السابق، حيث يجب أن يستخدم علاقة الربط التي تربط العميل بالمركب الذي يملكه؛ ولذلك يجب إضافة متغير إشارة للصنف Boat داخل تعريف هذا الصنف:

```
'Chapter 15 Example 3
Imports System.Collections
Public Class FindCustomer
   Inherits System.Windows.Forms.Form

   'Declare array to hold customer instances from database
   Private customers As New ArrayList()

   'Declare customer and boat reference variables
   Private aCustomer As Customer
   Private aBoat As Boat
```

وبالمثل فإذا ضغط المستخدم على الزر "Find Customer and Boat"، فإن الإجراء btnFind_Click لا يسترجع اسم العميل وعنوانه ورقم هاتفه ويعرضها فحسب، بل يسترجع بيانات المركب الذي يملكه العميل الحالي ويقوم بعرضها داخل العنوان lblBoatInfo:

```
Private Sub btnFind_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnFind.Click
        Dim i As Integer
        'Identify the selected index in the list box
        i = lstCustomer.SelectedIndex
        'Find this customer in the ArrayList
        aCustomer = customers(i)
        'Retrieve and display this customer's address and phone
        txtCustomerAddress.Text = aCustomer.GetAddress
        txtCustomerPhone.Text = aCustomer.GetPhoneNo
        'Retrieve this customer's boat reference
        aBoat = aCustomer.GetBoat()
        'Display the boat information
        lblBoatInfo.Text = "Customer Boat (4 attributes only) " _
         + aBoat.TellAboutSelf()
End Sub
```

لا يوجد هناك تغيير آخر على باقي تعريف الصنف FindCustomer، ويوضح الشكل رقم (١٥,١١) نموذج Sequence Diagram الخاص بالبحث واسترجاع بيانات العميل والمركب الذي يملكه وعرضها.



الشكل رقم (١٥,١١). النموذج التتابعي FindCustomer للمثال رقم ٣.

مراجعة صنف واجهة الاستخدام AddCustomer من الفصل الحادي عشر
Reviewing the AddCustomer GUI from Chapter 11

يشبه صنف واجهة الاستخدام AddCustomer لهذا المثال صنف واجهة الاستخدام AddCustomer المقدم خلال الفصل الحادي عشر، حيث كان عمل الصنف هو التحقق من صحة بيانات العميل المدخلة بواسطة المستخدم، ثم إنشاء كائن من الصنف Customer، ثم إرسال هذا الكائن إلى صنف واجهة الاستخدام AddBoat، فعندئذ يسترجع صنف واجهة الاستخدام AddBoat بيانات العميل من هذا الكائن وإضافتها داخل الجدول CustomerTable وإضافة بيانات المركب المدخلة داخل الجدول BoatTable.

لاحظ أن صنف واجهة الاستخدام AddCustomer لا يتعامل مباشرة مع قاعدة البيانات، ولكن يرسل كائن الصنف Customer المراد إضافته داخل قاعدة البيانات إلى الصنف AddBoat المسؤول عن إضافة كل من بيانات العميل والمركب داخل جدولي قاعدة البيانات، ومن ثم نضمن أن بيانات العميل لم تضاف بشكل منفرد بل يجب من توفر بيانات المركب الذي يملكه هذا العميل ليتم إضافتها سوياً.

ولأن صنف واجهة الاستخدام AddCustomer لا يتعامل مباشرة مع قاعدة البيانات فإن تعريف هذا الصنف يتشابه تماماً مع التعريف المقدم داخل الفصل الحادي عشر (انظر الشكل رقم ١٥,١٢)، ولاحظ أن الأوامر المنتجة تلقائياً بواسطة برنامج مصمم نماذج الويندوز لا يتطلب أي تعديل؛ ولذلك تم حذفه.

تعديل صنف واجهة الاستخدام AddBoat
Modifying the AddBoat GUI

يتشابه صنف واجهة الاستخدام AddBoat لهذا المثال أيضاً مع صنف واجهة الاستخدام AddBoat المقدم في الفصل الحادي عشر. تذكر أن إجراء الإنشاء لهذا الصنف يستقبل متغير إشارة للصنف Customer كمعامل، وكان أيضاً يتحقق من صحة بيانات المركب المدخلة بواسطة المستخدم، ثم ينشئ كائناً من الصنف Customer الذي تم استقباله، ثم يضيف بيانات المركب إلى الجدول BoatTable ويضيف بيانات المركب إلى الجدول CustomerTable.

قبل شرح الأوامر المسؤولة عن تنفيذا هذا السيناريو، يجب أن تلاحظ أن جدول BoatTable يخزن البيانات المشتركة بين الصنف Sailboat والصنف Powerboat فقط (أي صفات الصنف Boat فقط)، مع أن صنف واجهة الاستخدام AddBoat يستقبل بيانات المراكب الشراعية والمركب الآلية ويتحقق من صحتها، ولكن يجب تخزين البيانات المشتركة فقط. ولتخزين بيانات المراكب الشراعية (كائنات الصنف Sailboat) وكائنات المراكب الآلية (كائنات الصنف Powerboat) يجب إنشاء صنفين للتعامل مع البيانات لكل من الصنف Sailboat والصنف Powerboat وإنشاء جدولين منفصلين داخل قاعدة البيانات.

```
'Chapter 15 Example 3
'Same as Chapter 11
Public Class AddCustomer
    Inherits System.Windows.Forms.Form

    'Declare customer reference variable
    Private aCustomer As Customer
    'Declare string variables for name, address and phone
    Private customerName, customerAddress, customerPhone As String

    'Declare reference for AddBoat form
    Private frmAddBoatGUI As AddBoat

    'Windows Form Designer generated code goes here

    Private Sub btnAddBoat_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnAddBoat.Click
        'Get values from the text boxes
        customerName = txtName.Text
        customerAddress = txtAddress.Text
        customerPhone = txtPhone.Text
        'Validate that the user has entered values for name, address _
        and phone
        If customerName.Length = 0 Or _
         customerAddress.Length = 0 Or _
         customerPhone.Length = 0 Then
            MessageBox.Show("Please Enter All Data")
         Else
            'Create a customer instance
            aCustomer = New Customer(customerName, customerAddress, _
                        customerPhone)
            'Clear the form
            ClearForm()
            'Hide the AddCustomer form
            Me.Hide()
            'Create and display the AddBoat form
            frmAddBoatGUI = New AddBoat(aCustomer)
            frmAddBoatGUI.ShowDialog()
            'Make the AddCustomer form visible
            Me.Show()
        End If
    End Sub
    Private Sub ClearForm()
       txtName.Text = ""
       txtAddress.Text = ""
       txtPhone.Text = ""
       txtName.Focus()
    End Sub

    Private Sub btnClear_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnClear.Click
        ClearForm()
    End Sub

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

الشكل رقم (١٥.١٢). واجهة المستخدم الرسومية AddCustomer من الفصل الحادي عشر.

يبدأ صنف واجهة الاستخدام AddBoat كالعادة بتعريف إجراء الإنشاء الذي يستقبل متغير إشارة للصنف Customer، ثم يقوم بإسناده لمتغير معرف على مستوى الصنف (aCustomer) لكي تتمكن جميع إجراءات الصنف من التعامل معه.

```
'Chapter 15 Example 3
Public Class AddBoat
    Inherits System.Windows.Forms.Form
    'Declare a Customer reference variable
    Private aCustomer As Customer

    'Within the generated code:
    '  1. Modify the parameter list for the constructor method to
    '       accept a customer reference argument.
    '  2. Add a statement to the generated code to store the
    '       Customer instance that is passed to the constructor.

    Public Sub New(ByRef thisCustomer As Customer)

       MyBase.New()

       'This call is required by the Windows Form Designer.
       InitializeComponent()

       'Add any initialization after the InitializeComponent() call

       'Store customer reference in the private class variable
       aCustomer = thisCustomer

End Sub
```

كما أن كلاً من الإجراء radSailboat_CheckChanged والإجراء radPowerboat_CheckChanged والإجراء btnAdd_Click ظلا دون تغيير عن مثال الفصل الحادي عشر.

```
Private Sub radSailboat_CheckedChanged _
(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles radSailboat.CheckedChanged
       'If sailboat radio button is checked display the sailboat panel
       If radSailboat.Checked = True Then
          pnlPowerboat.Visible = False
          pnlSailboat.Visible = True
       End If
End Sub

Private Sub radPowerBoat_CheckedChanged _
(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles radPowerBoat.CheckedChanged
    'If powerboat radio button is checked display the powerboat panel
    If radPowerBoat.Checked = True Then
       pnlSailboat.Visible = False
       pnlPowerboat.Visible = True
    End If
End Sub
```

```
Private Sub btnAdd_Click _
(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles btnAdd.Click
    'Declare variables for boat attributes
    Dim boatLength As Double
    Dim year As Integer
    Dim manufacturer, stateRegistration As String
    'Get reg. no. and manufacturer from text boxes
    stateRegistration = txtStateRegNo.Text
    manufacturer = txtManufacturer.Text
    'Validate that theuser has entered reg. no. and manuracturer
    If manufacturer.Length = 0 Or _
     stateRegistration.Length = 0 Then
        MessageBox.Show("Please Enter All Data")
    Else
        'Ensure that the value for boat length is numeric
        If Not IsNumeric(txtLength.Text) Then
           MessageBox.Show("Boat Length must be numeric")
        Else
           boatLength = txtLength.Text
           'Ensure that the value for year is numeric
           If Not IsNumeric(txtYear.Text) Then
              MessageBox.Show("Year must be numeric")
           Else
              year = txtYear.Text
              'Invoke the appropriate add method
              If radSailboat.Checked = True Then
                 AddSailboat(stateRegistration, boatLength, _
                    manufacturer, year)
              Else
                 AddPowerboat(stateRegistration, boatLength, _
                    manufacturer, year)
             End If
           End If
        End If
    End If
End Sub
```

ومثل المثال السابق، فإن كلاً من الإجراء AddSailboat والإجراء AddPowerboat يتحققان أولاً من صحة بيانات كل نوع مركب مدخلة بواسطة المستخدم، ولكن يتم تخزين الصفات المشتركة بين النوعين؛ ولذلك يتم تعريف كائن من الصنف Boat بدلاً من تعريف كائن من الصنف Sailboat أو كائن من الصنف Poweboat، ثم يتم إنشاء علاقة الترابط بين كائن الصنف Customer وكائن الصنف Boat بواسطة استدعاء الإجراء AssignBoatToCustomer المعرف داخل الصنف Boat، أما مهمة إضافة كل من بيانات العميل وبيانات المركب داخل قاعدة البيانات فهي مسؤولية الإجراء AddNew للصنف Customer. ولكن تذكر أن عمل هذا الإجراء هو استدعاء الإجراء المناظر له AddNew المعرف داخل الصنف CustomerDA الذي يستخلص بيانات العميل بما فيها المركب

الذي يملكه، ثم يقوم بإضافتها داخل الجدول CustomerTable، ثم يستدعي الإجراء AddNew المعرف داخل الصنف BoatDA الذي يقوم بإضافة بيانات المركب داخل الجدول BoatTable. إن هذا الأسلوب يضمن إمكانية تكامل البيانات حيث لا يمكن أن يتم تخزين عميل دون مركب أو العكس، كما أن أوامر إضافة البيانات تتواجد داخل التركيب Try-Catch لاكتشاف حدوث أي أخطاء.

```
'Method to add a sailboat and customer
'only the four boat attributes are actually stored in the database
Private Sub AddSailboat(ByVal aStateRegNo As String, _
ByVal aBoatLength As Double, ByVal aManufacturer As String, _
ByVal aYear As Integer)
    'Declare variables for sailboat attributes
    Dim numberOfSails As Integer
    Dim keelDepth As Double
    Dim motorType As String

    'Ensure that the number of sails is numeric
    If Not IsNumeric(txtNumberOfSails.Text) Then
        MessageBox.Show("Number of sails must be numeric")
    Else
        numberOfSails = txtNumberOfSails.Text
        'Ensure that the keel depth is numeric
        If Not IsNumeric(txtKeelDepth.Text) Then
            MessageBox.Show("Keel depth must be numeric")
        Else
            keelDepth = txtKeelDepth.Text
            'Determine type of engine
            If radNoEngine.Checked = True Then
                motorType = "None"
            Else
                If radInboardEngine.Checked = True Then
                    motorType = "Inboard"
                Else
                    If radOutboardEngine.Checked = True Then
                        motorType = "Outboard"
                    End If
                End If
            End If
            'Create a Boat instance - sailboat attributes are not
            'persistent
            Dim aBoat As Boat
            aBoat = New Boat(aStateRegNo, aBoatLength, _
                   aManufacturer, aYear)

            'Establish the association in both directions
            aBoat.AssignBoatToCustomer(aCustomer)

            'Add the customer and the boat to the database
            'The CustomerDA.AddNew invokes Boat.AddNew
            Try
```

```
                aCustomer.AddNew()
                MessageBox.Show("Customer and Boat Added")
            Catch de As DuplicateException
                MessageBox.Show("Sorry: " + de.Message.ToString)
            End Try
            'Clear the form
            ClearForm()
        End If
    End If
End Sub
```

يتطلب الإجراء AddPowerboat تغيرات مشابهة، كما أن كلاً من الإجراء btnClear_Click والإجراء btnClose_Click لا يتطلبان أي تعديل.

ملخص الفصل
Chapter Summary

- لقد تعلمنا من الفصول السابقة كيف نصمم التطبيقات باستخدام أسلوب ثلاثي الطبقات: طبقة أصناف واجهة الاستخدام (GUI Classes)، وطبقة أصناف مجال المشكلة (PD Classes)، وطبقة أصناف التعامل مع البيانات (DA Classes). إن استخدام أسلوب الطبقات الثلاثة من الأصناف في تصميم النظم وتطويرها يؤدي إلى سهولة صيانة البرنامج، فإذا تم تغيير أصناف واجهة الاستخدام مثلاً فلا تتأثر كل من أصناف مجال المشكلة وأصناف التعامل مع البيانات. ولقد تعلمنا خلال هذا الفصل كيفية تجميع كل من أصناف مجال المشكلة وأصناف التعامل مع قواعد البيانات وأصناف واجهة الاستخدام داخل تطبيق واحد.
- كما يقدم أسلوب التصميم الذي يعتمد على ثلاثة أنواع من الأصناف إطار عمل لعملية تطوير النظم المعتمدة على الكائنات التي تتم بشكل متكرر، حيث يتم إجراء بعض التحليل، ثم إجراء بعض التصميم، ثم إجراء بعض البرمجة وهكذا إلى أن يتم تطوير النظام كاملاً.
- اشتمل المثال الأول على كل من صنف مجال المشكلة Customer وصنف التعامل مع البيانات CustomerDA اللذين قمنا بتطويرهما خلال الفصل الثالث عشر، كما اشتمل على صنف واجهة الاستخدام FindCustomer الذي قمنا بتصميمه خلال الفصل الحادي عشر، حيث كان يقوم باسترجاع معلومات عميل محدد ولا يقوم بتعديل داخل قاعدة البيانات.
- اشتمل المثال الثاني على واجهات استخدام متعددة، وإضافة بيانات (كائنات أصناف مجال المشكلة) إلى قاعدة البيانات باستخدام صنف التعامل مع البيانات.

- استخدم المثال الثالث فوائد متعددة وصنفي مجال مشكلة بينهما علاقة ترابط، وكذلك صنفي التعامل مع البيانات، وأخيراً قاعدة بيانات تحتوي على جدولين، ويقوم الصنف CustomerDA باسترجاع بيانات العملاء والمراكب التي يملكونها من قاعدة البيانات، ثم يقوم بتحويلها إلى كائنات مترابطة.

أسئلة المراجعة

Review Questions

١- اشرح كيف تكون صيانة النظام مبسطة مع التصميم ثلاثي الطبقات.

٢- اشرح كيف يعطي التصميم ثلاثي الطبقات فائدة في توزيع التطبيق عبر الشبكة.

٣- اشرح كيف يمكن استخدام تصميم ثلاثي الطبقات في تعريف المهام المنفذة في كل دورة لعملية التطوير.

٤- راجع الشكل رقم (١٥,٢). في أي مرحلة من دورة حياة التطبيق يجب تعريف الكود المطلوب لصنف مجال المشكلة الذي يحتوي على إجراءات البناء والمرور؟

٥- راجع مرة أخرى الشكل رقم (١٥,٢). أي مرحلة من دورة حياة التطبيق يجب تعريف الكود المطلوب للإجراءات الثابتة لصنف مجال المشكلة مثل Initialize و Find و GetAll؟

٦- في أي مرحلة من دورة حياة التطبيق يجب تعريف الكود المطلوب لأصناف واجهة المستخدم الرسومية؟

٧- اشرح كيفية تعريف أصناف مجال المشكلة PD وأصناف واجهة المستخدم الرسومية GUI وأصناف مرور البيانات DA وبرمجتها واختبارها مشتملاً على دورات متعددة من التحليل والتصميم والبرمجة خلال دورة استكمال إحدى حالات الاستخدام.

٨- راجع النموذج التتابعي المعروض في الشكل رقم (١٥,٨). لماذا يحتوي على الصنف Customer (دون خط تحته)، بينما يتضمن الصنف aCustomer على خط تحته في النموذج نفسه؟

٩- يوضح الشكل رقم (١٥,٨) التفاعل المطلوب لإنشاء كائنات من الصنف Customer من قاعدة البيانات ومن ثم يستطيع المستخدم أن يجدها؛ ولذلك لم يتم استدعاء كل من الإجراء AddNew والإجراء Update. إذا قام المثال بتعديل أو إضافة كائن لعميل جديد، فأين يتم إرسال رسائل لكل من Customer و aCustomer و CustomerDA من وجهة نظر واجهة الاستخدام؟ ولماذا؟

١٠- ما الاختلاف في الوظيفة بين إجراءات مجال المشكلة setter وإجراء Update؟

١١- ما الاختلاف في الوظيفة بين بناء صنف مجال المشكلة وإجراء AddNew؟

١٢- تتطلب واجهة استخدام الفصل الحادي عشر تعديلاً بسيطاً في الفصل الحالي. قم بسرد هذه التغييرات لكل من النموذج MainMenu والنموذج FindCustomer والنموذج AddCustomer.

١٣- لا تحتوي واجهة الاستخدام FindCustomer الخاصة بآخر مثال داخل هذا الفصل على كل من الأصناف الفرعية Sailboat و Powerboat، كما يوجد مثال مشابه بالفصل الحادي عشر احتوى على هذه الأصناف واحتوى أيضاً على محاكاة تخزين الكائنات. اذكر التعديلات المطلوبة لكي يشمل هذا المثال كلاً من هذين الصنفين وكذلك إمكانية تخزين بياناتهما.

أسئلة المناقشة
Discussion Questions

١- ناقش الإستراتيجية المتبعة لتطوير برنامج برادشو مارينا وهي: تطوير أصناف مجال المشكلة، ثم تطوير أصناف واجهة الاستخدام، ثم تطوير أصناف التعامل مع البيانات. لماذا نقوم بتطوير أصناف واجهة الاستخدام قبل تطوير أصناف التعامل مع البيانات؟ ولماذا يجب أن يركز كل فريق على أحد الأنشطة فقط؟

٢- يعيد الإجراء GetAll المعرف داخل الصنف CustomerDA في المثال رقم ٣ مصفوفة من العملاء والمراكب التي يملكونها وكذلك يفعل كل من الإجراء Find والإجراء AddNew. لماذا يتعامل الصنف CustomerDA مع العملاء والمراكب معاً ولا يتعامل مع العملاء فقط؟ وما هي المهام الأخرى التي يجب أن تضاف لهذا الصنف لاسترجاع بيانات أخرى؟ وما هي الإجراءات التي يجب تعريفها لمعالجة هذه المهام؟ (لاحظ علاقة الترابط بين كل من الصنف Customer والصنف Lease).

٣- بالإشارة إلى السؤال رقم ٢، ما هي الإجراءات الأخرى التي يجب أن تضاف إلى الصنف BoatDA والصنف SlipDA والصنف DockDA والصنف LeaseDA؟ وإلى مدى سوف تساعد نماذج Use cases والسيناريوهات المختلفة الخاصة بها؟

٤- ناقش الإستراتيجية التالية: "أن كل إجراء Setter يجب أن يستدعي إجراء Update عند التنفيذ لكي يعدل قيمة الصفة مباشرة داخل قاعدة البيانات" من حيث المشاكل التي تظهر مع إجراء الإنشاء، وكذلك من حيث إدارة قواعد البيانات.

مشاريع الفصل
Projects

١- قم بتطوير نظام إدارة بيانات الطلبة الذي يتطلب تطوير الصنف Student الذي يحتوي على الصفة ID والصفة name والصفة phone number والصفة grade point average والصفة major. قم بتصميم نموذج Class Diagram موضحاً كلاً من الصنف Student والصنف StudentDA والإجراءات المطلوبة لهما. ثم قم بتطوير النظام

المعتمد على الأنواع الثلاثة من الأصناف مع قاعدة البيانات الخاصة به حيث يجب أن يحتوي النظام على نموذج لإضافة بيانات الطلبة ونموذج للبحث عن بيانات طالب وتعديلها. ستحتاج أن تطور قاعدة بيانات تحتوي على جدول واحد وعلى بيانات اختبار وعلى الصنف Student والصنف StudentDA. طور برنامج اختبار لكي تختبر الأصناف والتعامل مع قاعدة البيانات.

٢- استرجع مثال الإنسان والحيوان الأليف من الفصل التاسع. قم بتطوير قاعدة البيانات التي تحتوي على جدولين (الجدول Person لتخزين بيانات الناس، والجدول Pet لتخزين بيانات الحيوانات المدللة)، ويجب أن يحتوي الجدول Pet على مفتاح أجنبي مناظر Person ID للربط بين الجدولين. ثم عرف الصنف Person والصنف Pet والصنف PersonDA والصنف PetDA، ولاحظ أنك ستحتاج إلى تعريف الصنف PersonAndPetDatabaseConnect مثلما فعلنا في أمثلة الفصل الرابع عشر. أضف عينة بيانات إلى كلا الجدولين (بيانات أربعة أشخاص وأربعة حيوانات)، وافترض أن الشخص ربما لا يملك حيواناً أليفاً واحداً أو أكثر، وأن الحيوان الأليف يجب أن يكون مملوكاً لشخص واحد فقط. اكتب برنامج اختبار لكي تختبر كلاً من الإجراء GetAll والإجراء Find والإجراء Delete لهذه الأصناف.

٣- قم بتصميم وبرمجة أصناف واجهة المستخدم الخاصة بمثال الصنف Person والصنف Pet للمشروع رقم ٢ لكي تختبر الإجراء AddNew والإجراء Find والإجراء Update لهذه الأصناف. ويجب أن تشمل أصناف واجهة الاستخدام على واجهة MainMenu وواجهة AddPerson وواجهة AddPet وواجهة FindAPerson وواجهة FindPersonAndPet. ولا يجب أن يملك الشخص حيواناً أليفاً ليتواجد بالنظام ولكن عكس ذلك لا يتواجد حيوان أليف إلا يملكه شخص؛ ولذلك سوف يعيد الإجراء FindPerson بيانات الإنسان فقط حتى لو كان لا يملك هذا الإنسان حيواناً أليفاً. كما يجب أن يقدم صنف واجهة الاستخدام FindPersonAndPet معلومات عن الشخص والحيوان الأليف الذي يملكه. كما يجب أن يحتوي صنف واجهة الاستخدام FindPersonAndPet على زر يمكّن المستخدم من تعديل بيانات المالك، وكذلك صنف واجهة الاستخدام AddCustomer، أما الصنف AddPet فيجب أن يسمح للمستخدم أن يختار اسم الشخص الذي يملك الحيوان الأليف الجاري إدخال بياناته.

٤- ما هي التعديلات المطلوبة لتعديل مثال السؤال رقم ٣ لكي تجعل علاقة الترابط بين الصنف Person والصنف Pet علاقة "واحد-إلى-العديد" (حيث يملك كل شخص العديد من الحيوانات الأليفة). قم بتطوير صنف واجهة الاستخدام FindPetAndPerson الذي يقوم بعرض أكثر من حيوان أليف للشخص الواحد، ثم قم بإضافته للمشروع السابق (لاحظ أنه لا يجب تعديل قاعدة البيانات ولكن يجب تعديل الصنف Person والصنف PersonDA).