الفصل الحادي عشر

استخدام نماذج متعددة مع أصناف مجال المشكلة

USING MULTIPLE FORMS WITH PROBLEM DOMAIN CLASSES

أهداف الفصل:

- تطوير صنف لواجهة المستخدم الرسومية يتفاعل مع صنف مجال المشكلة.
- محاكاة التفاعل مع قواعد البيانات.
- تطوير صنف لواجهة المستخدم يتفاعل مع عدة أصناف مجال المشكلة.
- التجول بين النماذج داخل نظام متكامل.
- تطوير صنف لواجهة المستخدم يتجول بين أصناف مجال مشكلة بينها علاقة ترابط (Association Relation).

لقد تعلمنا في الفصل العاشر كيف نطور صنف واجهة المستخدم الرسومية (GUI Class) الذي يستخدم لعرض نموذج يحتوي على عناصر متنوعة لواجهة المستخدم مثل العناوين وصناديق النصوص والأزرار، وكذلك تعلمنا كيفية تعريف إجراءات معالجة الأحداث.

وسوف يستمر هذا الفصل في تطوير أصناف واجهة المستخدم لكي نوضح لك كيفية استخدام كل من صنف مجال المشكلة Customer وصنف مجال المشكلة Boat الذيْن تم تطويرهما في الباب الثاني من هذا الكتاب، حيث نقوم بإنشاء صنف واجهة المستخدم لإضافة بيانات عميل جديد، وصنف ثانٍ للبحث عن عميل محدد، وصنف ثالث لإضافة مركب جديد للنظام (يشمل هذا النموذج على عناصر جديدة مثل أزرار الاختيار RadioButton والإطارات Panels).

كما يوضح آخر مثالين من هذا الفصل كيفية استخدام النماذج المتعددة (Multiple Forms) حيث نرى كيف نطور نموذجاً يحتوي على قائمة رئيسة تسهل التنقل بين نماذج متعددة. وبدلاً من استخدام أصناف منفصلة لواجهة المستخدم، سوف نستخدم نموذجاً يحتوي على قائمة رئيسة لتحميل هذه الأصناف وتنفيذها كجزء من نظام

متكامل. وسوف ترى أيضاً كيف تتنقل بين أصناف مجال مشكلة تربطها علاقة ترابط (Association Relation) بواسطة تصميم صنف لواجهة المستخدم الذي يمكّن المستخدم من إضافة بيانات عميل وبيانات المركب الخاص به. يستخدم كثير من أمثلة هذا الفصل الصفوفات لمحاكاة قواعد البيانات التي تخزن بيانات العملاء وبيانات المراكب الخاصة بهم. وبالرغم من أن أصناف التعامل مع البيانات ومعالجة قواعد البيانات سوف يتم شرحها بالتفصيل لاحقاً في هذا الكتاب، إلا أن هذا الفصل سوف يقدم بعض الإجراءات المشتركة للتعامل مع البيانات وذلك لمحاكاة التعامل مع قواعد البيانات.

تطوير صنف واجهة المستخدم الذي يتفاعل مع صنف مجال مشكلة

Developing a GUI Class that Interacts with a PD Class

سوف نطور في هذه الفقرة صنف واجهة المستخدم اسمه AddCustomer لكي يستقبل بيانات العميل (قيم صفات الصنف Customer الذي طورناه في الفصل السادس)، ثم ننشئ كائناً من الصنف Customer وإسناد هذه البيانات إليه. يوضح الشكل رقم (١١.١) نموذج استقبال بيانات عميل الذي يحتوي على عنوان لعرض اسم شركة برادشو مارينا ويحتوي أيضاً على ثلاثة عناوين وثلاثة صناديق نصية لاستقبال بيانات العميل.

الشكل رقم (١١.١)- نموذج إضافة عميل جديد.

وكما تعلمنا في الفصل العاشر، فإن إنشاء نموذج يتطلب إنشاء تطبيق ويندوز أولاً، ثم استخدام صندوق الأدوات (Toolbox) ونافذة الخصائص (Properties Window) لإضافة عناصر إلى النموذج وضبط خصائصها، مع العلم أن لغة VB .NET تولد الشفرة المطلوبة لتنفيذ ذلك النموذج أثناء عملية تصميمه، ولقد شرحنا كيفية تصميم النماذج بالتفصيل خلال الفصل العاشر، ومن ثم سوف نركز في هذا الفصل على شفرة معالجة الأحداث فقط وذلك لتوضيح مفاهيم جديدة.

يحتوي النموذج الموضح في الشكل رقم (١١,١) على ثلاثة صناديق نصية وهي (txtName و txtAddress و txtPhone) لاستقبال صفات الصنف Customer (الصفة Name، والصفة Address، والصفة Phone) وعلى ثلاثة عناوين لعنونة تلك الصناديق، كما يحتوي النموذج على ثلاثة أزرار هي (btnAdd و btnClose و btnClear) لاستقبال عميل جديد وإضافته بعد إدخال بياناته ولمسح الصناديق النصية ولغلق النموذج وإنهاء المعالجة.

إضافة متغيرات إلى الشفرة المنتجة تلقائياً
Adding Variables to the Generated Code

إن عملية تصميم النماذج التي يطلق عليها البرمجة المرئية تنتج فقط المتغيرات الضرورية لإنشاء واجهة المستخدم، ولكن توجد هناك حاجة إلى تعريف متغيرات أخرى تستخدم للتفاعل مع المستخدم عند معالجة الأحداث. هذه المتغيرات يجب أن تضاف إلى شفرة البرنامج من خلال نافذة محرر البرنامج. ولأن معظم إجراءات معالجة الأحداث تحتاج إلى هذه المتغيرات، فإنه يجب تعريفها على مستوى مجال صنف واجهة المستخدم (Class Scope) ومن ثم تستطيع جميع إجراءات الصنف أن تتعامل معها. وبالعودة إلى المثال الذي بين أيدينا، فإن الحاجة تستدعي تعريف متغير إشارة ليشير إلى كائن الصنف Customer الذي سوف يتم إنشاؤه عند استقبال بيانات عميل جديد. ونحتاج أيضاً إلى تعريف ثلاثة متغيرات من النوع String لتخزين بيانات العميل المدخلة في الصناديق النصية كما هو واضح من الأوامر التالية:

```
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
```

معالجة الأحداث

Handling Events

يحتوي هذا المثال على ثلاثة أزرار ضغط تحتاج إلى تطوير ثلاثة إجراءات معالجة ليتم استدعاؤها عند الضغط عليها ؛ وذلك لتنفيذ مهمة كل زر (إضافة عميل جديد، ومسح بيانات النموذج، وغلق النموذج). وكما تعلمنا في الفصل السابق فإن كل حدث يتم معالجته بإجراء منفصل، ومن ثم سوف نطور ثلاثة إجراءات (الإجراء btnClear_Click، والإجراء btnClose_Click، والإجراء btnAdd_Click) لمعالجة الأحداث الثلاثة (الحدث btnClear.Click، والحدث btnClose.Click، والحدث btnAdd.Click).

سوف يستدعي الإجراء btnClear_Click الإجراء الفرعي ClearForm الذي يسند قيمة نصية فارغة " " داخل جميع الصناديق النصية، ثم يستدعي الإجراء Focus من صندوق النص txtName لوضع المؤشر داخله. لاحظ أن السطر الأول في الإجراء التالي لا يأتي في سطر واحد ولذلك استخدمنا الرمز "_" الذي يشير إلى تكملة الأمر في السطر التالي:

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

Private Sub ClearForm()
      txtName.Text = ""
      txtAddress.Text = ""
      txtPhone.Text = ""
      txtName.Focus()
End Sub


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

يسترجع الإجراء btnAdd_Click قيم الصناديق النصية ثم يتأكد أن المستخدم أدخل الاسم والعنوان ورقم الهاتف. إذا اكتشف عدم وجود أي من هذه البيانات يظهر رسالة خطأ تحث المستخدم على إدخال القيمة المفقودة. ولقد استخدمنا أسلوب التحقق من طول القيمة النصية المدخلة (كائن الصنف String) لمعرفة إذا كان المستخدم أدخل قيمة نصية داخل الصناديق النصية أم لا. فإذا كان طول النص يساوي صفراً فهذا يعني أن المستخدم لم يدخل شيئاً في هذا الصندوق النصي، وإذا أدخل المستخدم القيم الثلاثة فعندئذ يتم إنشاء كائن من الصنف Customer وإظهار رسالة تفيد إضافة عميل جديد ("Customer Added")، ثم يتم مسح النموذج.

```
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 namem
      '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 a Customer instance
            aCustomer = New Customer(customerName, _
                        customerAddress, _ customerPhone)
            MessageBox.Show("Customer Added")
            'Clear the form
            ClearForm()
      End If
End Sub
```

محاكاة التعامل مع قواعد البيانات

Simulating Interaction with a Database

سوف نحتاج أحياناً إلى البحث عن أحد عملاء شركة برادشو مارينا وعرض معلوماته (عنوانه ورقم هاتفه)، وسوف نشرح في هذه الفقرة كيف نطور صنف واجهة المستخدم FindCustomer الذي يمكننا من البحث عن عميل ما وعرض معلوماته (إن وجدت) باستخدام أسلوب يتشابه مع أسلوب التفاعل مع قواعد البيانات.

يوضح الشكل رقم (١١,٢) نموذج البحث عن عميل الذي يحتوي على صندوق قائمة بأسماء العملاء (lstCustomer) وعلى صندوقي نص (txtCustomerAddress و txtCustomerPhone) وعلى ثلاثة أزرار (btnClose و btnUpdate و btnFind) وعلى عنوان يحتوي على شعار الشركة برادشو مارينا (lblLogo).

إنشاء مصفوفة عملاء

Creating Array List of Customers

لا توجد لدينا قاعدة بيانات إلى الآن تحتوي على بيانات العملاء والعقود. ولكن على أي حال سوف ننشئ هذه القاعدة في الفصل الرابع عشر وعندئذ سوف يسترجع الصنف FindCustomer سجلات العملاء من قاعدة البيانات، أما الآن فسوف يتفاعل الصنف FindCustomer مع مصفوفة بشكل يحاكي التفاعل مع قواعد البيانات. ولمحاكاة التفاعل مع قواعد البيانات سوف ينشئ الصنف FindCustomer ستة كائنات من الصنف Customer لتخزين بيانات ستة عملاء وتخزينها في مصفوفة من النوع ArrayList. ولأن الصنف ArrayList يوجد داخل الفضاء المسمى

System.Collection، يبدأ تعريف الصنف FindCustomer بجلب هذا الفضاء المسمى، ثم تعريف الكائن aCustomer من الصنف Customer لتخزين بيانات العميل الحالي، ثم تعريف الكائن customers من الصنف ArrayList للاحتفاظ بكائنات العملاء كما هو واضح من الأوامر التالية.

الشكل رقم (١١،٢). نموذج البحث عن عميل.

```
Imports System.Collections
Public Class FindCustomer
      Inherits System.Windows.Forms.Form

      'Declare customer reference variable
      Private aCustomer As Customer
      'Declare array to simulate customer database
      Private customers As New ArrayList()
```

يعمل الصنف ArrayList مثل المصفوفة العادية ولكن له حجم متغير حيث يحتوي على الإجراء Add الذي يستخدم لإضافة عنصر في نهاية المصفوفة الحالية. توضح الأوامر التالية تعريف الإجراء CreateCustomers الذي يستخدم لإنشاء ستة كائنات من الصنف Customer (ستة عملاء) وإضافتها داخل المصفوفة customers.

```
Private Sub CreateCustomers()
      'Create customer instances - simulate database
      customers.Add(New Customer("Eleanor", "Atlanta", "123-4567"))
      customers.Add(New Customer("Mike", "Boston", "467-1122"))
      customers.Add(New Customer("JoAnn", "St. Louis", "765-4321"))
      customers.Add(New Customer("Dave", "Atlanta", "321-4567"))
      customers.Add(New Customer("Brian", "Boston", "467-1234"))
      customers.Add(New Customer("Dan", "St. Louis", "587-4321"))
End Sub
```

يتم استدعاء الإجراء CreateCustomers بواسطة الإجراء PopulateListBox الذي يستخدم لاستخلاص أسماء العملاء (كائنات الصنف Customer) لإضافة أسمائهم إلى قائمة الاختيار في النموذج كما هو واضح من الأوامر التالية:

```
Private Sub PopulateListBox()
      'Create the customer instances
      CreateCustomers()
      '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
End Sub
```

يتم استدعاء الإجراء PopulateListBox من إجراء إنشاء صنف واجهة المستخدم FindCustomer (الإجراء New) كجزء من عملية تهيئة النموذج. لاحظ أن إجراء إنشاء الصنف FindCustomer يوجد داخل الأوامر المخفية في نافذة محرر البرنامج والتي يجب توسعتها لرؤية الأوامر البرمجية المنتجة آلياً بواسطة مصمم نماذج الويندوز وتعديل هذا الإجراء كما هو واضح في الأوامر التالية:

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

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

      'Add any initialization after the InitializeComponent() call

      'Populate the customer list box
      PopulateListBox()
End Sub
```

معالجة الأحداث
Handling Events

عندما يضغط المستخدم على زر Find أو زر Update أو زر Close يتم استدعاء إجراء معالجة الحدث المرتبط بكل منها (الإجراء btnFind_Click والإجراء btnUpdate_Click والإجراء btnClose_Click على التوالي).

إيجاد عميل
Finding a Customer

يهدف الإجراء btnFind_Click إلى البحث عن العميل الذي يظهر اسمه داخل قائمة اختيار العملاء، ثم يسترجع كلاً من العنوان ورقم الهاتف الخاص به ويعرضهما. تذكر أن الصنف ListBox يحتوي على خاصية اسمها SelectedIndex التي تعيد فهرس العنصر المختار من قائمة الاختيار. وبذلك نستطيع أن نستفيد من توافق فهرس المصفوفة التي تخزن العملاء (كائنات الصنف Customer) مع فهرس قائمة الاختيار حيث تم تخزين العملاء داخل كل من قائمة الاختيار والمصفوفة بنفس الترتيب. وبمعنى آخر إن العميل الأول في المصفوفة يظهر كأول اسم في قائمة الاختيار، والعميل الثاني في المصفوفة يظهر كثاني اسم في قائمة الاختيار، وهكذا. يسترجع الإجراء btnFind_Click (كما هو واضح من الأوامر التالية) فهرس اسم العميل المختار من قائمة اختيار أسماء العملاء، ثم يسترجع كائن الصنف Customer من المصفوفة customers مستخدماً نفس هذا الفهرس. ومن ثم يسترجع عنوان العميل مستدعياً الإجراء GetAddress ورقم هاتف العميل مستدعياً الإجراء GetPhoneNo من هذا الكائن، ثم يقوم بعرض هذه المعلومات.

```
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()
End Sub
```

تعديل بيانات عميل
Updating a Customer

يستخدم صنف واجهة المستخدم FindCustomer أيضاً لتعديل بيانات العميل، حيث يمكن اختيار العميل من قائمة اختيار العملاء ثم الضغط على الزر Find فعندئذ ستظهر بيانات هذا العميل داخل الصناديق النصية، ثم نقوم بتغيير العنوان أو رقم الهاتف، ثم نقوم بالضغط على الزر Update.

يتبع الإجراء btnUpdate_Click نفس أسلوب الإجراء btnFind_Click، كما هو واضح من الأوامر التالية حيث يستخدم الخاصية SelectedIndex لاسترجاع كائن الصنف Customer المناسب، ثم استدعاء الإجراء SetAddress ممرراً إليه قيمة صندوق النص txtAddress لتغيير عنوان العميل واستدعاء الإجراء SetPhoneNo ممرراً إليه قيمة صندوق النص txtPhoneNo لتغيير رقم هاتف العميل. ثم تظهر رسالة تفيد تعديل بيانات العميل "Customer Updated"، ويتم مسح صندوق نص العنوان وصندوق نص رقم الهاتف.

```
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 ArrayList
      aCustomer = customers(i)
      'Set this customer's address and phone to the values
      'entered by the user
      aCustomer.SetAddress(txtCustomerAddress.Text)
      aCustomer.SetPhoneNo(txtCustomerPhone.Text)
      MessageBox.Show("Customer Updated")
      'Clear address and phone text fields
      txtCustomerAddress.Text = ""
      txtCustomerPhone.Text = ""
End Sub
```

إن أوامر الإجراء btnClose_Click هي أوامر إجراء زر إغلاق نموذج الصنف AddCustomer نفسها.

تطوير صنف واجهة مستخدم رسومية يتفاعل مع العديد من أصناف مجال المشكلة

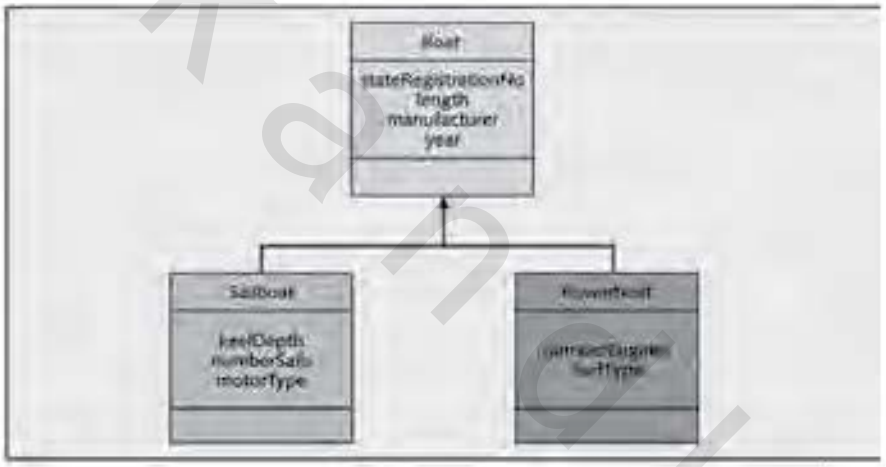
Developing a GUI Class that Interacts with Multiple PD Class

سوف نطور في هذه الفقرة صنفاً لواجهة المستخدم اسمه AddBoat الذي يستخدم لإضافة بيانات مركب إلى نظام برادشو مارينا، ومن ثم سيتفاعل مع الصنف Boat والصنف Sailboat والصنف Powerboat بإنشاء كائنات من كل صنف من هذه الأصناف (راجع الباب الثاني من هذا الكتاب).

تذكر أن الصنف Boat يمثل الصنف الرئيس لكل من الصنف الفرعي Sailboat والصنف الفرعي Powerboat، كما هو واضح في الشكل رقم (١١,٣). يحتوي الصنف Boat على الصفات المشتركة بين كل من الصنفين (الصفة StateRegistrationNo والصفة Length والصفة Manufacture والصفة Year)، بينما يحتوي كل صنف فرعي على الصفات الخاصة به فقط. فعلى سبيل المثال يحتوي الصنف Sailboat على الصفة keepDepth والصفة numberSails والصفة motorType، بينما يحتوي الصنف Powerboat على الصفة numberEngines وعلى الصفة fuelType.

يوضح الشكل رقم (١١,٤) نموذج الصنف AddBoat. لاحظ أن زر الاختيار Sailboat هو الزر المختار حالياً وإن الصناديق النصية تظهر طبقاً لنوع المركب المختار حالياً، ومن ثم فإن صفات المركب الشراعي هي التي تظهر حالياً داخل النموذج.

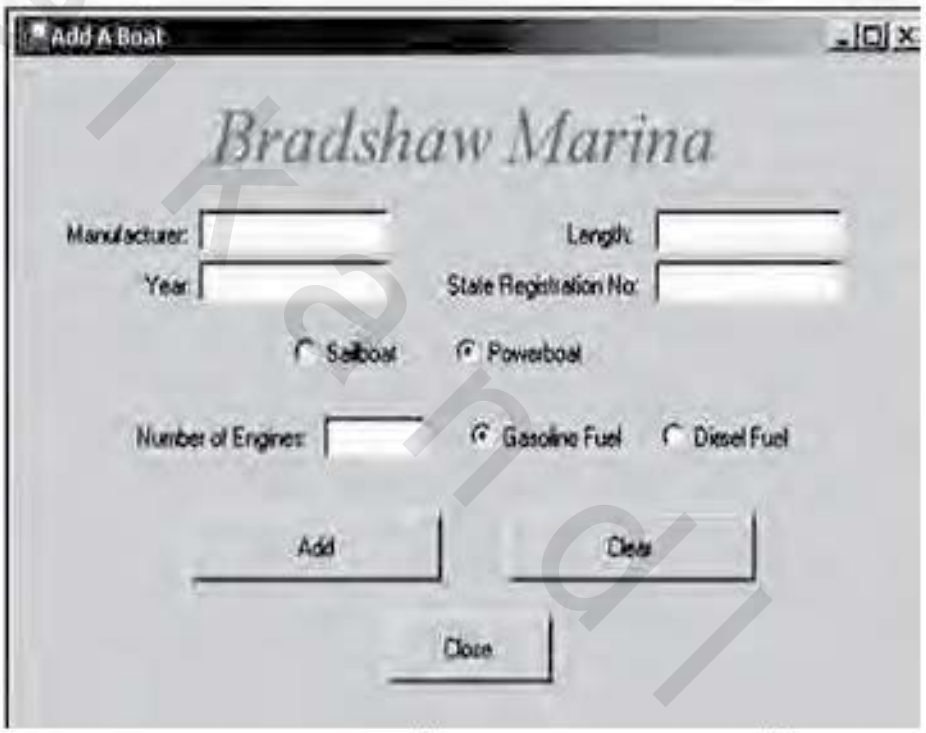
يحتوي الجزء العلوي من النموذج على عناوين وصناديق نصية لاستقبال الصفات الأربعة للصنف Boat (الصفات المشتركة لكل من المركب الآلي والمركب الشراعي)، أما الجزء الأوسط من النموذج فيحتوي على زري اختيار (rbSailboat و rbPowerboat) وإطارين (pnlSailboat و pnlPowerboat). وعند إدخال بيانات المركب يجب على المستخدم أن يختار زر الاختيار Sailboat أو زر الاختيار Powerboat، مع العلم أن زر الاختيار Sailboat هو الاختيار الافتراضي.



الشكل رقم (١١,٣). علاقة الوراثة بين الصنف Boat والصنف Sailboat والصنف Powerboat.

الشكل رقم (١١,٤). نموذج إضافة مركب شراعي (عندما يتم اختيار Sailboat).

يحتوي كل إطار على صناديق نصية وأزرار اختيار تتعلق بالصفات الخاصة بنوع المركب المرتبط به. فمثلاً إذا اختار المستخدم زر الاختيار Sailboat فعندئذ يظهر الإطار pnlSailboat كما هو موضح في الشكل رقم (١١.٤). وإذا اختار المستخدم زر الاختيار Powerboat فعندئذ يظهر الإطار pnlPowerboat كما هو واضح في الشكل رقم (١١.٥). يمكّنك هذا الأسلوب من تغيير محتويات الإطارات وبالتالي تغيير مظهر النموذج بشكل تلقائي.



الشكل رقم (١١.٥). نموذج إضافة مركب آلي (عندما يتم اختيار Powerboat).

وكما هو واضح من الشكل رقم (١١.٤) فإن إطار المركب الشراعي pnlSailboat يحتوي على صندوقين نصيين لاستقبال عدد الأشرعة وعمق المركب (Number of sails و Keel Depth) ويحتوي على ثلاثة أزرار اختيار لتحديد نوع المحرك (Outboard Engine و Inboard Engine و No Engine)، مع العلم أن الاختيار Inboard Engine هو الاختيار الافتراضي.

وكما هو واضح أيضاً من الشكل رقم (١١.٥) فإن إطار المركب الآلي pnlPowerboat يحتوي على صندوق نصي لاستقبال عدد المحركات (Number of Engines) وعلى زري اختيار لمعرفة نوع الوقود (Gasoline Fuel و Diesel Fuel)، مع العلم أن الاختيار Gasoline Fuel هو الاختيار الافتراضي. أما الجزء السفلي من النموذج (كما هو واضح من الشكلين السابقين) فيحتوي على ثلاثة أزرار مثل أزرار النموذج AddCustomer.

معالجة الأحداث
Handling Events

يختلف هذا المثال عن ما سبق في استجابته للأحداث، حيث يجب أن نستجيب إلى حدث التأشير (Checking) أو إلغاء التأشير (Unchecking) لكل من زر الاختيار Sailboat وزر الاختيار Powerboat، هذا بالإضافة إلى أحداث الضغط على الزر Add والزر Clear والزر Close.

تذكر عزيزي القارئ من الفصل العاشر أن حدث تأشير أو إلغاء تأشير زر الاختيار (Radio Button) يؤدي إلى إصدار حدث واحد هو CheckedChanged. ومن ثم سوف نكتب إجراء معالجة حدث زر الاختيار radSailboat (يسمى radSailboat_CheckedChanged) والذي يستخدم الصفة Checked المعرفة داخل الصنف RadioButton لمعرفة هل زر الاختيار radSailboat مؤشر عليه أم لا. فإذا كان زر الاختيار radSailboat مؤشراً عليه فسيتم إخفاء الإطار pnlPowerboat وإظهار الإطار pnlSailboat.

وبالمثل فإن الإجراء radPowerboat_checkedChanged يحدد هل زر الاختيار radPowerboat مؤشر عليه أم لا، فإذا كان زر الاختيار radPowerboat مؤشراً عليه فعندئذ يتم إخفاء الإطار pnlSailboat وإظهار الإطار pnlPowerboat.

```
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
```

تعريف الإجراء ClearForm
Writing the ClearForm Method

إن كلاً من الإجراء btnClear_Click والإجراء ClearForm يعملان مثل إجراءات النموذج AddCustomer، كما هو واضح من الأوامر التالية حيث يستدعي الإجراء btnClear_Click الإجراء ClearForm الذي يسند نصاً فارغاً

("") لصناديق النص، ثم يؤشر أزرار الاختيار بالقيم الافتراضية لها، وأخيراً يضع المؤشر داخل صندوق النص Manufacturer.

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

Private Sub ClearForm()
      txtManufacturer.Text = ""
      txtLength.Text = ""
      txtYear.Text = ""
      txtStateRegNo.Text = ""
      txtNumberOfSails.Text = ""
      txtKeelDepth.Text = ""
      txtNumberOfEngines.Text = ""
      'Set default values of radio buttons
      radSailboat.Checked = True
      radInboardEngine.Checked = True
      radGasoline.Checked = True
      'Set the input focus
      txtManufacturer.Focus()
End Sub
```

تعريف الإجراء btnAdd_Click

Writing the btnAdd_Click Method

يسترجع الإجراء btnAdd_ClicK قيمة Manufacturer وقيمة Length وقيمة Year وقيمة Registration من الصناديق النصية المناظرة لها في النموذج، ثم يؤدي عملية تحقق بسيطة للتأكد من صحة البيانات، ثم يستدعي الإجراء AddSailboat أو الإجراء AddPowerboat وذلك بناء على نوع المركب المختار حالياً؛ وذلك لإكمال عملية إضافة المركب. يتم التحقق من صحة القيمة Manufacturer والقيمة Registration بنفس طريقة التحقق من اسم العميل وعنوانه داخل النموذج AddCustomer، فإذا كان طول حروف النص المدخل يساوي صفراً ستظهر رسالة خطأ تفيد عدم إدخال قيمة لهذا الحقل، أما إذا كانت القيم المدخلة لها صحيحة فيتحقق الإجراء من صحة قيمة Length وقيمة Year وذلك باستدعاء الإجراء IsNumeric الذي يستخدم لمعرفة هل القيمة الممررة له يمكن تحويلها إلى قيمة رقمية أم لا. فإذا لم تكن القيمة المدخلة رقمية فعندئذ تظهر رسالة خطأ تفيد ذلك. فإذا نجحت عملية التحقق من صحة البيانات يتم استدعاء الإجراء AddSailboat أو الإجراء AddPowerboat بناء على نوع المركب المختار (لاحظ أن البيانات المشتركة بين كل من المركب الشراعي والمركب الآلي والمعرفة داخل الصنف الأب Boat سوف تمرر لكلا الإجراءين) كما هو واضح من الأوامر التالية.

```
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
      'Validate that the user has entered reg. no. and manufacturer
      If manufacturer.Length = 0 Or _
       stateRegistration.Length = 0 Then
            MessageBox.Show("Please Enter All Date")
      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 appropriate add method
                        If radSailboat.Checked = True Then
                              AddSailboat(stateRegistration, _
                              boatLength, manufacturer, year)
                        Else
                              AddPowerboat(stateRegistration, _
                              boatLength, manufacturer, yesr)
                        End If
                  End If
            End If
      End If
End Sub
```

تعريف الإجراء AddSailboat

Writing the AddSailboat Method

كما هو واضح من الأوامر التالية، يستقبل الإجراء AddSailboat القيمة Manufacturer والقيمة Length والقيمة Year والقيمة Registration كمعاملات للإجراء، ثم يستخلص قيمة Number of Sails (عدد الأشرعة) وقيمة Keel Depth (عمق المركب) من الصناديق النصية، ثم يتحقق من صحتها مستدعياً الإجراء IsNumeric لمعرفة هل القيم المدخلة يمكن تحويلها إلى قيم رقمية أم لا. فإذا كانت القيم المدخلة غير صحيحة ستظهر رسالة خطأ تفيد ذلك.

وبعد التحقق من صحة بيانات عدد الأشرعة وعمق المركب، نكتب ثلاثة أوامر If لمعرفة أي نوع محرك تم اختياره (أي زر اختيار تم اختياره) ومن ثم نسند القيمة المناسبة داخل المتغير motorType. وأخيراً يتم تعريف كائن من الصنف الفرعي Sailboat وإظهار رسالة تفيد نجاح عملية الإضافة (Sailboat Added).

```
'Method to add a sailboat
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
      'Declare a sailboat reference variable
      Dim aSailboat As Sailboat

      '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 sailboat instance
                  aSailboat = New Sailboat(aStateRegNo, aBoatLength, _
                              aManufacturer, aYear, keelDepth, _
                              numberOfSails, motorType)
                  MessageBox.Show("Sailboat Added")
                  ClearForm()
            End If
     End If
End Sub
```

تعريف الإجراء AddPowerboat

Writing the AddPowerboat Method

كما هو واضح من الأوامر التالية، يعمل الإجراء AddPowerboat مثل الإجراء AddSailboat ماعدا أن عدد المحركات (Number of Engines) هي التي يتم استخلاصها من النموذج. وسيتم التعامل مع زري اختيار فقط (الزر radGasoline والزر radDiesel) بدلاً من ثلاثة أزرار اختيار.

```
'Method to add a powerboat
Private Sub AddPowerboat(ByVal aStateRegNo As String, _
ByVal aBoatLength As Double, ByVal aManufacturer As String, _
ByVal aYear As Integer)
      'Declare variables for powerboat attributes
      Dim numberOfEngines As Integer
      Dim fuelType As String
      'Declare a powerboat reference variable
      Dim aPowerboat As Powerboat

      'Ensure that the number of engines is numeric
      If Not IsNumeric(txtNumberOfEngines.Text) Then
            MessageBox.Show("Number of engines must be numeric")
      Else
            numberOfEngines = txtNumberOfEngines.Text
            'Determine type of fuel
            If radGasoline.Checked = True Then
                  fuelType = "Gasoline"
            Else
                  If radDiesel.Checked = True Then
                        fuelType = "Diesel"
                  End If
            End If
            'Create a Powerboat instance
            aPowerboat = New Powerboat(aStateRegNo, aBoatLength, _
                        aManufacturer, aYear, numberOfEngines, _
                        fuelType)
            MessageBox.Show("Powerboat Added")
            ClearForm()
      End If
End Sub
```

التجول بين النماذج في نظام متكامل

Navigating Multiple Forms in an Integrated System

يتكون نظام برادشو مارينا إلى الآن من نماذج منفصلة (مثل النموذج AddCustomer والنموذج AddBoat) وغير مرتبطة حيث يؤدي كل نموذج مهمته بشكل منفصل. ولكن من المستحسن أن نجعل النظام متكاملاً وذلك بربط هذه النماذج المنفصلة من خلال تطوير أوامر رئيسة (Main Menu) تتكون من مجموعة من الأزرار للمهام المتاحة، وبذلك يكون من السهل على المستخدم النهائي أن يتنقل من مهمة (مثل FindCustomer) إلى مهمة ثانية (مثل AddCustomer). عند الضغط على أحد هذه الأزرار يتم إنشاء النموذج المناسب (إنشاء كائن من أحد أصناف واجهة المستخدم) ثم عرضه. وبعد إتمام مهمة النموذج والضغط على زر Close نعود ثانية إلى القائمة الرئيسة. سوف نتعلم في هذه الفقرة كيفية تطوير قائمة رئيسة وكيف ندمج النماذج المنفصلة السابقة داخل نظام موحد ومتكامل. ولكن سوف نتعلم أولاً كيف نطور صنفاً يحاكي التعامل مع البيانات (Data Access Class) حيث سنستخدم جميع نماذج النظام إجراءات هذا الصنف لمحاكاة التفاعل مع قواعد البيانات.

محاكاة صنف التعامل مع البيانات

Simulating s Data Access Class

سوف نتعلم في الباب الرابع من هذا الكتاب كيف نصمم أصناف التعامل مع البيانات التي تتعامل مع قواعد بيانات علاقية (Relational Database)، ثم بعد ذلك نرى كيف نستخدم لغة SQL لكي نخزن بيانات شركة برادشو مارينا ونسترجعها، وكما تعلم أننا نركز في هذا الفصل على التكامل بين عدة نماذج منفصلة، وبذلك سوف نحاكي فقط التفاعل مع قواعد البيانات بواسطة استخدام مصفوفة عملاء ومصفوفة مراكب. تركز هذه الفقرة على التعامل مع مصفوفة العملاء، وسوف يتم التعرف على مصفوفة المراكب لاحقاً. عزيزي القارئ تذكر أن الأنظمة المعتمدة على الكائنات تستخدم أسلوب تصميم طبقات الأصناف الثلاثة (Three-Tier):

أولاً: طبقة أصناف واجهة المستخدم (GUI Classes) التي تقدم واجهة استخدام رسومية لاستقبال البيانات وعرضها.

ثانياً: طبقة أصناف مجال المشكلة (PD Classes) التي تمثل كائنات أعمال مجال المشكلة وعملياتها.

ثالثاً: طبقة أصناف التعامل مع البيانات (DA Classes) التي تقدم خدمات تخزين البيانات واسترجاعها.

إن أهم ميزة تتصف بها تقنية تصميم طبقات الأصناف الثلاثة هي انفصال عمل كل طبقة عن الأخرى، فمثلاً ليس من الضروري معرفة كيفية تخزين البيانات واسترجاعها بواسطة أصناف التعامل مع البيانات لكل من طبقة أصناف مجال المشكلة وطبقة أصناف واجهة المستخدم، بل تحتاج فقط أن تستدعي إجراءات تخزين هذه البيانات واسترجاعها، وبالمثل فإن أصناف التعامل مع البيانات لا تعلم شيئاً عن عمل أصناف واجهة المستخدم. يؤدي هذا الانفصال إلى سهولة عملية صيانة النظام في المستقبل لأن التعديل في أحد الطبقات لا يتطلب تعديلاً في الطبقات الأخرى. فعلى سبيل المثال إن تغيير نوع قاعدة البيانات يؤثر فقط على أصناف التعامل مع البيانات دون تأثير على كل من طبقة أصناف واجهة المستخدم وطبقة أصناف مجال المشكلة.

يحتوي صنف التعامل مع البيانات CustomerData الذي نقدمه في هذا الموضوع على أربعة إجراءات وهي: الإجراء Initialize، والإجراء GetAll، والإجراء AddNew، والإجراء Update. سوف تستدعي أصناف واجهة المستخدم هذه الإجراءات دون أن تعلم كيفية تعريفها. وهذا يعني أنه من الممكن أن نغير تعريف هذه الإجراءات لاحقاً لتتعامل مع قاعدة بيانات (بدلاً من التعامل مع المصفوفات)، وبذلك ستبقى أوامر استدعاء تلك الإجراءات داخل أصناف واجهة المستخدم كما هي دون تغيير. نلخص فيما يلي الغرض من هذه الإجراءات الأربعة:

- الإجراء Initialize : يهدف هذا الإجراء إلى تنفيذ مهام الإعداد للاتصال بقاعدة البيانات.
- الإجراء GetAll : يهدف هذا الإجراء إلى استرجاع جميع العملاء المخزنين في قاعدة البيانات.
- الإجراء AddNew : يهدف هذا الإجراء إلى تخزين عميل جديد (كائن الصنف Customer) داخل قاعدة البيانات.
- الإجراء Update : يهدف هذا الإجراء إلى تعديل بيانات عميل موجود (كائن من الصنف Customer) داخل قاعدة البيانات.

التعرف على الصنف CustomerData
Understanding the CustomerData Class

سوف نعرف إجراءات التعامل مع البيانات لهذا المثال داخل الصنف CustomerData ، ولأن هذا الصنف يتعامل مع مصفوفة بدلاً من قاعدة البيانات ستبدأ أوامر هذا الصنف بالأمر Imports الذي يجلب الفضاء المسمى System.Collections. ثم يتم تعريف المصفوفة customers من النوع ArrayList لتخزين كائنات الصنف Customer ، ثم تعريف متغير إشارة aCustomer من النوع Customer. ويجب أن تلاحظ أن كلاً من المتغير customers والمتغير aCustomer تم تعريفهما على مستوى الصنف ومن النوع Shared ، كما هو واضح من الأوامر التالية.

```
'CustomerData class definition
Imports System.Collections
Public Class CustomerData

      'Declare array to simulate customer database
      Private Shared customers As New ArrayList()

      'Declare Customer reference variable
      Private Shared aCustomer As Customer
```

ينشئ الإجراء Initialize ستة كائنات من الصنف Customer (أي ستة عملاء) وتخزينها في المصفوفة customers ، كما هو واضح من الأوامر التالية (تذكر من المثال FindCustomer أن الإجراء Add المعرف داخل الصنف ArrayList يستخدم لإلحاق عنصر في نهاية المصفوفة) ، مع العلم أن الإجراء Initialize تم توصيفه من النوع Shared لكي يتم استدعاؤه مباشرة من خلال الصنف CustomerData.

```
Public Shared Sub Initialize()
      'Create customer instances - simulate database
      customers.Add(New Customer("Eleanor", "Atlanta", "123-4567"))
      customers.Add(New Customer("Mike", "Boston", "467-1122"))
```

```
        customers.Add(New Customer("JoAnn", "St. Louis", "765-4321"))
        customers.Add(New Customer("Dave", "Atlanta", "321-4567"))
        customers.Add(New Customer("Brian", "Boston", "467-1234"))
        customers.Add(New Customer("Dan", "St. Louis", "587-4321"))
End Sub
```

وكما هو واضح من الأوامر التالية، فإن الإجراء GetAll يعيد مصفوفة تحتوي على جميع العملاء (المتغير customers).

```
Public Shared Function GetAll() As ArrayList
        Return customers
End Function
```

وعندما نريد أن نضيف بيانات عميل جديد إلى قاعدة البيانات، يجب أن نستدعي الإجراء AddNew الذي يستقبل كائناً من الصنف Customer، ومن ثم سوف يستدعى الإجراء Add المعرف داخل الصنف ArrayList لإضافة هذا الكائن داخل المصفوفة customers، كما هو واضح من الأوامر التالية.

```
Public Shared Function AddNew(ByRef newCustomer As Customer)
        customers.Add(newCustomer)
End Function
```

وعندما نريد أن نعدل بيانات عميل موجود داخل قاعدة البيانات، يجب أن نستدعي الإجراء Update الذي يستقبل فهرس العميل داخل المصفوفة والعميل الجديد (كائن من الصنف Customer)، ومن ثم سوف يستبدل الأجراء Update الكائن القديم بالكائن الجديد عند هذا الفهرس داخل المصفوفة Customers كما هو واضح من الأوامر التالية.

```
Public Shared Function Update(ByVal index As Integer, _
ByRef thisCustomer As Customer)
        customers(index) = thisCustomer
End Function
```

تكامل مجموعة من النماذج
Integrating Multiple Forms

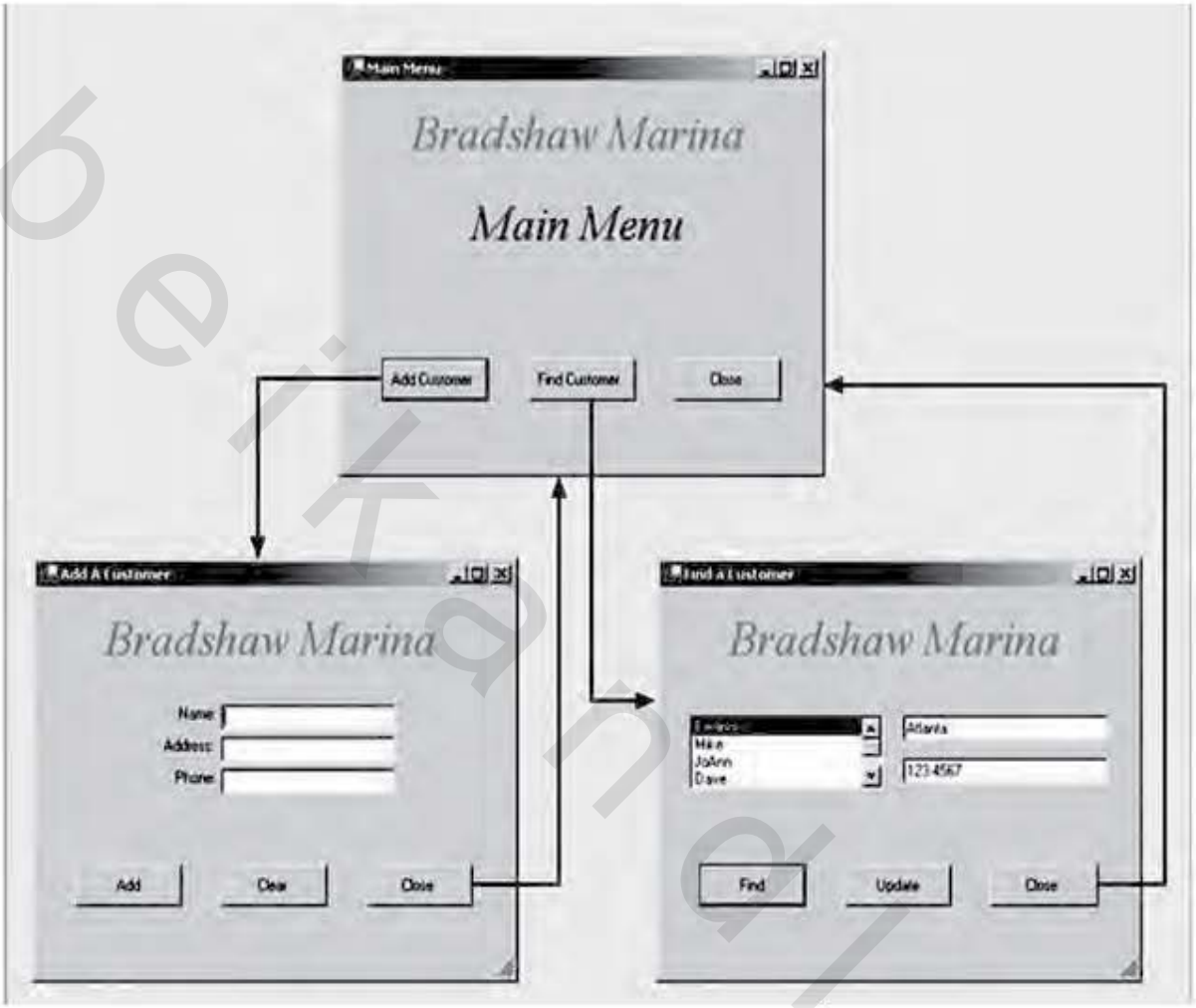
سوف ندمج داخل هذا المثال نماذج متعددة داخل نظام واحد يحتوي على نموذج قائمة رئيسة (Main Menu) بالإضافة إلى النموذج AddCustomer والنموذج FindCustomer الذين تم تطويرهما سابقاً في هذا الفصل، علماً بأن

نموذج البداية هو نموذج القائمة الرئيسة الموضح في الشكل رقم (١١,٦) الذي يظهر عند بداية تنفيذ النظام، ومنه سوف يتجول المستخدم بين النماذج الأخرى بالضغط على الزر المناسب.

الشكل رقم (١١,٦). نموذج القائمة الرئيسة لنظام شركة براندو مارينا.

يوضح الشكل رقم (١١,٧) ترتيب ظهور النماذج عند استدعاء مهمة إيجاد عميل أو عند استدعاء مهمة إضافة عميل جديد. وعندما يضغط المستخدم على الزر "Find Customer" في نموذج القائمة الرئيسة يظهر النموذج FindCustomer، حيث يمكن للمستخدم أن يختار اسم العميل من قائمة الأسماء ثم الضغط على زر "Find" ليظهر كل من عنوان العميل المختار ورقم هاتفه. أما إذا ضغط المستخدم على الزر "Close" سيختفي النموذج FindCustomer ويظهر نموذج القائمة الرئيسة مرة أخرى.

أما إذا ضغط المستخدم على زر "Add Customer" في نموذج القائمة الرئيسة، سيختفي نموذج القائمة الرئيسة ويظهر النموذج AddCustomer حيث يمكن للمستخدم أن يكتب بيانات عميل جديد، ثم يضغط على الزر "Add"، فستذذ ستظهر رسالة تفيد نجاح إضافة العميل الجديد "Customer Added"، ثم يضغط المستخدم على الزر OK لغلق الرسالة، ثم يضغط المستخدم على الزر Close لغلق النموذج AddCustomer والعودة إلى نموذج القائمة الرئيسة.



الشكل رقم (١١,٧). سيناريو البحث عن عميل وإضافته.

تصميم نموذج القائمة الرئيسة (Main Menu)

Designing the Main Menu

يوجد نموذج القائمة الرئيسة النماذج الأخرى ويعتبر كعنصر بداية (Startup Object) للمشروع ككل. يحتوي نموذج القائمة الرئيسة على ثلاثة أزرار (الزر btnAdd، والزر btnFind، والزر btnClose). ويعمل إجراء معالجة الزر btnClose مثل إجراءات الأمثلة السابقة. ويختفي إجراء معالجة الزر btnAdd نموذج القائمة الرئيسة وذلك باستدعاء الإجراء Hide، ثم ينشئ كائناً من صنف واجهة المستخدم AddCustomer، ثم يقوم بعرضه بواسطة استدعاء الإجراء ShowDialog.

يظهر الإجراء ShowDialog النموذج بطريقة تسمى Modal والتي تتسبب في إغلاق أو إخفاء النموذج قبل تنفيذ الأمر التالي لأمر ShowDialog. وهذا يعني أن النموذج AddCustomer يجب أن يغلق أو يختفي قبل ظهور

نموذج آخر داخل النظام. وبعد إغلاق النموذج AddCustomer يجب أن نظهر نموذج القائمة الرئيسة مرة ثانية باستدعاء الإجراء Show. نستخدم الإجراء Show بدلاً من إجراء ShowDialog لإعادة ظهور نموذج القائمة الرئيسة بعد إخفائه (لأنه يوجد بالفعل في الذاكرة ولكن لا يظهر على الشاشة)، كما هو واضح من الأوامر التالية.

```
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
```

وبالمثل عند الضغط على زر "Find Customer" يتم إنشاء كائن من صنف واجهة المستخدم FindCustomer وإظهاره بعد إخفاء نموذج القائمة الرئيسة، ولأن النموذج FindCustomer سيظهر أيضاً بطريقة Modal فإن نموذج القائمة الرئيسة لا يمكن أن يظهر إلا بعد غلق أو إخفاء النموذج FindCustomer، كما هو واضح من الأوامر التالية.

```
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
```

كما هو واضح من الأوامر التالية، يقوم نموذج القائمة الرئيسة بتنفيذ مهمة هامة أخرى وهي استدعاء الإجراء Initialize المسؤول عن تهيئة مصفوفة العملاء (customers) حيث يتم استدعاء هذا الإجراء من إجراء إنشاء صنف القائمة الرئيسة (الإجراء New). تذكر أن إجراء إنشاء صنف القائمة الرئيسة يوجد داخل جزء الأوامر المخفي داخل نافذة محرر أوامر البرنامج؛ لذلك يجب عليك أن تظهره أولاً بتوسعة هذا الجزء. وتذكر أيضاً أن الإجراء Initialize يجب أن يستدعى بواسطة اسم الصنف CustomerData مباشرة CustomerData.Initialize() لأنه معرف من النوع Shared.

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

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

      'Add any initialization after the InitializeComponent() call
      'Initialize the simulated database
      CustomerData.Initialize()
End Sub
```

البحث عن عميل
Finding a Customer

يتشابه النموذج FindCustomer كثيراً مع المثال الثاني الذي عرفناه في هذا الفصل، ولكن على أي حال يوجد بينهما اختلافان هامان وهما: أولاً، عندما نضغط على الزر Close يعيد إظهار نموذج القائمة الرئيسة. ثانياً، يستطيع النموذج FindCustomer إيجاد العملاء اللذين تم إضافتهم بواسطة النموذج AddCustomer لأن الصنف CustomerData يشارك بقية الأصناف في المصفوفة customers التي تخزن العملاء. هذا يعني أن كل عميل يضاف إلى هذه المصفوفة customers (قاعدة البيانات المؤقتة) يصبح متاحاً للبحث عنه بواسطة النموذج FindCustomer.

يتطلب النموذج FindCustomer الجديد تعديلاً واحداً فقط عن النموذج القديم حيث كان صنف النموذج FindCustomer القديم يحتوي على إجراء CreateCustomers الذي يتم استدعاؤه بواسطة الإجراء PopulateListBox المسؤول عن إنشاء ستة كائنات من الصنف Customer وتخزينها داخل المصفوفة. أما النموذج FindCustomer الجديد فيحتاج فقط أن يستدعي الإجراء GetAll المعرف داخل صنف التعامل مع البيانات CustomerData وذلك للحصول مباشرة على مصفوفة العملاء التي تحتوي على أحدث بيانات للعملاء. يعرض الشكل رقم (١١,٨) تعريف الصنف FindCustomer الجديد، وبالطبع لا يحتوي هذا الشكل على الأوامر البرمجية التي تنتج آلياً بواسطة مصمم نماذج الويندوز.

```
Imports System.Collections
Public Class FindCustomer
      Inherits System.Windows.Forms.Form

      'Declare customer reference variable
      Private aCustomer As Customer
      'Declare array to simulate customer database
      Private customers As New ArrayList()
      'Add a statement to the generated code to invoke the
      'PopulateListBox method
      'NOTE: Windows Form Designer generated code goes here

      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 simulated database
            aCustomer = customers(i)
            'Retrieve and display this customer's address and phone
            txtCustomerAddress.Text = aCustomer.GetAddress()
            txtCustomerPhone.Text = aCustomer.GetPhoneNo()
      End Sub
      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 simulated database
            aCustomer = customers(i)
            'Set this customer's address and phone to the values
            'entered by the user
            aCustomer.SetAddress(txtCustomerAddress.Text)
            aCustomer.SetPhoneNo(txtCustomerPhone.Text)
            'Update this customer in the simulated database
            CustomerData.Update(i, aCustomer)
            MessageBox.Show("Customer Updated")
            'Clear address and phone text fields
            txtCustomerAddress.Text = ""
            txtCustomerPhone.Text = ""
      End Sub

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

      Private Sub PopulateListBox()
            'Retrieve all customer data from simulated database
            customers = CustomerData.GetAll()
            'Add the name of each customer retrieved to the list
            Dim i As Integer
            For i = 0 To customers.Count - 1
                  aCustomer = customers(i)
                  lstCustomer.Items.Add(aCustomer.GetName())
            Next
      End Sub
End Class
```

الشكل رقم (١١,٨). تعريف الصنف FindCustomer.

إضافة عميل جديد
Adding a Customer

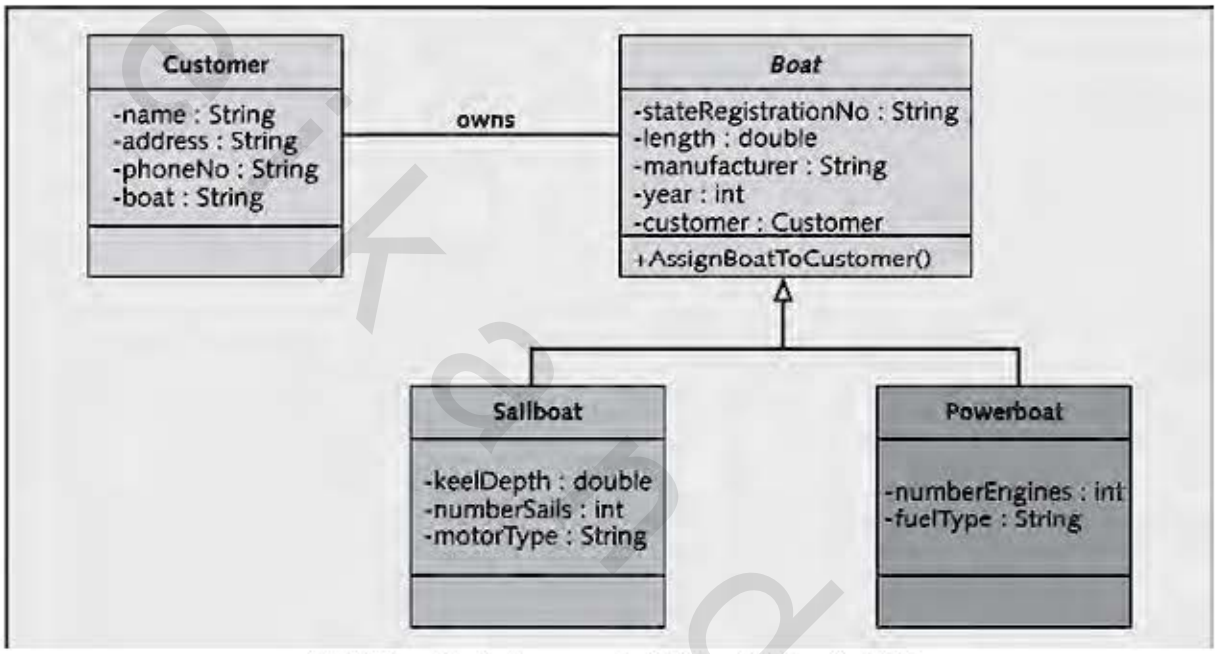
إن النموذج AddCustomer الجديد المسؤول عن إضافة عميل جديد يحتاج أيضاً إلى تعديل بسيط وهو استدعاء الإجراء AddNew من صنف التعامل مع البيانات CustomerData داخل إجراء معالجة حدث الضغط على الزر Add ؛ وذلك لإضافة كائن من الصنف Customer داخل المصفوفة customers التي تحتوي على العملاء، كما هو واضح من الأوامر التالية.

```
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)
            'Add customer to simulated database
            CustomerData.AddNew(aCustomer)
            MessageBox.Show("Customer Added ")
            'Clear the form
            ClearForm()
      End If
End Sub
```

تطوير صنف واجهة المستخدم الرسومية ليتجول في علاقة الترابط بين أصناف مجال المشكلة
Developing a GUI Class That Navigates a PD Association

لقد تعلمنا في الباب الثاني من هذا الكتاب أنه يوجد نوعان من العلاقات بين أصناف مجال المشكلة وهما: أولاً، علاقة التوارث (Inheritance) حيث رأينا أن الصنف الرئيس Boat يرثه كل من الصنف الفرعي Sailboat والصنف الفرعي Powerboat. ثانياً، علاقة الترابط (Association) حيث رأينا أن الصنف Customer تربطه علاقة ترابط "واحد إلى واحد" بالصنف Boat. يوضح الشكل رقم (١١,٩) جزءاً من نموذج Class Diagram الذي يحتوي على مثل هذه العلاقات حيث يوضح أن الصنف Customer يحتوي على متغير إشارة من النوع Boat، وأن الصنف

Boat يحتوي على متغير إشارة من النوع Customer. تذكر أن الإجراء AssignBoatToCustomer يقيم العلاقة بين كائن من الصنف Customer وكائن من الصنف Boat بإسناد متغير الإشارة الموجود داخل كائن الصنف Customer بكائن الصنف Boat وكذلك العكس.



الشكل رقم (١١,٩). علاقة الصنف Customer بالصنف Boat وأنواعه.

سوف تتعلم في هذه الفقرة كيف نستفيد من هذه العلاقات في تطوير واجهة للمستخدم أكثر فائدة، حيث نجري بعض التعديلات البسيطة على الصنف AddCustomer والصنف FindCustomer والصنف AddBoat لكي نطور نظاماً متعدد النماذج الذي يمكّننا من إضافة عميل جديد والمركب الخاص به في آنٍ واحد، وسوف نتمكن أيضاً من البحث عن عميل وعرض بياناته وبيانات المركب الذي يملكه في آنٍ واحد وذلك بواسطة علاقة الترابط بين العميل والمركب.

تعريف الصنف CustomerAndBoatData

Understanding the CustomerAndBoatData Class

لقد أوضحنا في الموضوع السابق كيف نحاكي قاعدة بيانات العملاء باستخدام مصفوفة من العملاء، وسوف نستخدم هذا الأسلوب نفسه في هذا المثال وذلك بتعريف مصفوفتين لتخزين بيانات العملاء وبيانات المراكب التي يملكونها، ولقد عرفنا أربعة إجراءات داخل صنف التعامل مع بيانات العملاء وهي: الإجراء Initialize لإنشاء مصفوفة تحتوي على كائنات الصنف Customer، والإجراء GetAll لإرجاع مصفوفة كائنات

الصنف Customer، والإجراء AddNew لإضافة عميل جديد (كائن من الصنف Customer) داخل مصفوفة العملاء، والإجراء Update لتعديل بيانات عميل (قيم صفات كائن من الصنف Customer) داخل مصفوفة العملاء.

سوف نستخدم الأسلوب نفسه لمحاكاة قاعدة البيانات للعملاء والمراكب حيث يستدعي الإجراء Initialize الآن كلاً من الإجراء InitializeCustomer والإجراء InitializeBoat كما هو واضح من الأوامر التالية.

```
Imports System.Collections
Public Class CustomerAndBoatData

      'Declare array lists to simulate customer and boat database
      Private Shared customers As New ArrayList()
      Private Shared boats As New ArrayList()

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

      Public Shared Sub Initialize()
            InitializeCustomer()
            InitializeBoat()
      End Sub

Private Shared Sub InitializeCustomer()
      'Create customer instances - simulate database
      customers.Add(New Customer("Eleanor", "Atlanta", "123-4567"))
      customers.Add(New Customer("Mike", "Boston", "467-1122"))
      customers.Add(New Customer("JoAnn", "St. Louis", "765-4321"))
      customers.Add(New Customer("Dave", "Atlanta", "321-4567"))
      customers.Add(New Customer("Brian", "Boston", "467-1234"))
      customers.Add(New Customer("Dan", "St. Louis", "587-4321"))
End Sub
```

ينشئ الإجراء InitializeBoat ثلاثة كائنات من الصنف SailBoat وثلاثة كائنات من الصنف PowerBoat، ثم يضيفها إلى المصفوفة boats ثم يستدعي الإجراء AssignBoatToCustomer المعرف داخل الصنف Boat لإقامة العلاقة بين العملاء والمراكب التي يملكونها، وللحفاظ على بساطة هذا المثال أسندنا المركب الأول للعميل الأول وأسندنا المركب الثاني إلى العميل الثاني وهكذا كما هو واضح من الأوامر التالية.

```
Private Shared Sub InitializeBoat()
      'Create boat instances
      boats.Add(New Sailboat _
            ("MO34561", 28, "Tartan", 1998, 2, 4.11,"Inboard"))
      boats.Add(New Sailboat _
            ("MO98765", 28, "J-Boat", 1986, 4, 5.0, "None"))
      boats.Add(New Sailboat_
```

```
            ("MO12345", 26, "Ranger", 1976, 7, 4.5,"Outboard"))
        boats.Add(New Powerboat _
            ("MO445566", 20, "Bayliner", 2001, 2,"Gasoline"))
        boats.Add(New Powerboat _
            ("MO223344", 24, "Tracker", 1996, 1,"Diesel"))
        boats.Add(New Powerboat _
            ("MO457812", 19, "Ranger", 2001, 1,"Gasoline"))

        'Assign boats to customers
        Dim i As Integer
        For i = 0 To customers.Count - 1
            aCustomer = customers(i)
            aBoat = boats(i)
            aBoat.AssignBoatToCustomer(aCustomer)
        Next
End Sub
```

والآن بعد تنفيذ الإجراء Initialize يصبح لدينا مصفوفتان وهما: المصفوفة الأولى customers التي تحتوي على ستة عملاء (كائنات من الصنف Customer)، والمصفوفة الثانية boats التي تحتوي على ستة مراكب (كائنات من الصنف Sailboat والصنف Powerboat)، مع العلم أن كل عميل يشير إلى المركب الذي يملكه والعكس صحيح.

لم تتغير كل من أوامر الإجراء GetAll والإجراء Update عن مثيلاتهما في الأمثلة السابقة، وسوف يتم استبدال الإجراء AddNew الذي يضيف كائن الصنف Customer إلى مصفوفة العملاء بالإجراء AddNewBoat، وبدلاً من إضافة كائن الصنف Customer إلى مصفوفة العملاء فقط سيتم إضافة كائن الصنف Boat أيضاً إلى مصفوفة المراكب (المركب الذي يملكه هذا العميل)، ومن ثم سوف يستقبل هذا الإجراء معاملان وهما: كائن الصنف Customer، وكائن الصنف Boat كما هو واضح من الأوامر التالية.

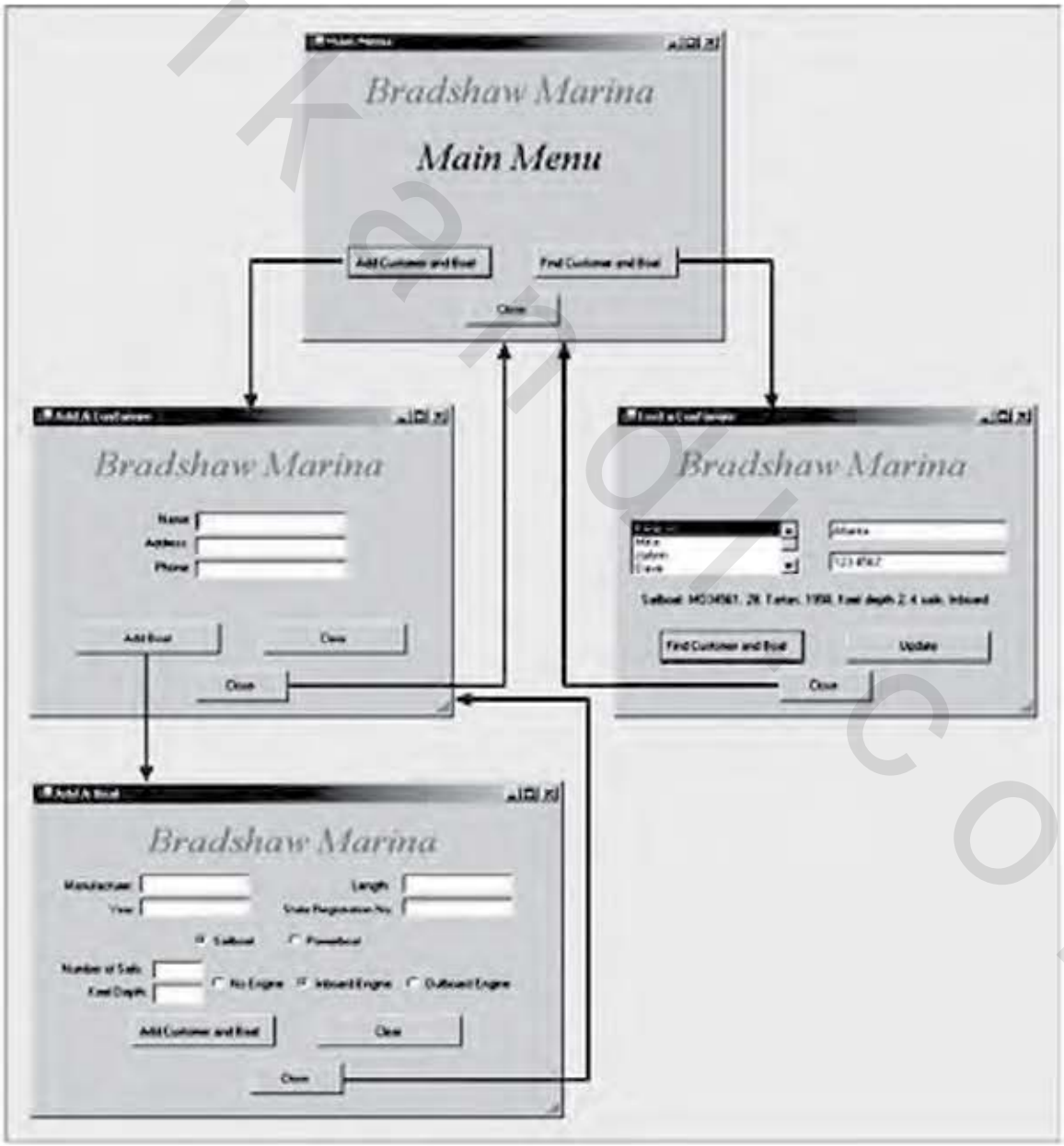
```
Public Shared Function AddNewBoat(ByRef newBoat As Boat, _
ByRef newCustomer As Customer)
        'Add the boat
        boats.Add(newBoat)
        'Add the customer
        customers.Add(newCustomer)
End Function
```

تصميم سيناريو ظهور النماذج
Designing the GUI Sequence

يوضح الشكل رقم (١١.١٠) سيناريو ظهور نماذج إضافة عميل جديد ومركب جديد حيث يمكن للمستخدم أن يضغط على الزر "Add Customer And Boat" من نموذج القائمة الرئيسة الذي يقوم بإظهار نموذج AddCustomer، ثم يقوم المستخدم بإدخال بيانات العميل الجديد، ثم يضغط على زر "Add Boat" الذي يظهر

النموذج AddBoat، ثم يقوم المستخدم بإدخال بيانات المركب الجديد (مثلما رأينا في الفصل السابق)، ثم يضغط على زر "Add Customer And Boat" المتواجد داخل النموذج AddBoat، فعندئذ تظهر رسالة تفيد إضافة عميل جديد ومركب بنجاح ("Customer And Boat Added").

لا توجد تغيرات كثيرة في نموذج القائمة الرئيسة عن المثال السابق عدا تعديل في بعض العناوين مثل عناوين الأزرار، كما أدخلنا بعض التعديلات على كل من الصنف AddCustomer والصنف AddBoat.



الشكل رقم (١١,١٠). سيناريو إضافة عميل والمركب الذي يملكه.

إضافة عميل
Adding a Customer

لكي نضيف عميلاً جديداً إلى النظام نضغط على زر "Add Customer And Boat" المتواجد داخل نموذج القائمة الرئيسة الذي يتسبب في إظهار النموذج AddCustomer الذي أجرينا عليه ثلاثة تعديلات بسيطة عن المثال السابق هكذا:

١- لقد غيرنا عنوان الزر "Add" إلى "Add Boat"، كما غيرنا اسم الزر من btnAdd إلى btnAddBoat.

٢- عرفنا متغير إشارة من صنف واجهة الاستخدام AddBoat على مستوى الصنف AddCustomer هكذا:

```
'Declare reference for AddBoat form
Private frmAddBoatGUI As AddBoat
```

٣- لقد أضفنا أوامر إلى الإجراء btnAddBoat_Click الذي يتم استدعاء استجابته إلى الضغط على الزر "Add Boat" لكي ننشئ كائناً من النموذج AddBoat وإظهاره، وعندما ينشئ كائناً من النموذج AddBoat يمرر له كائن الصنف Customer بواسطة إجراء إنشاء (الإجراء New) الذي تم تعديله أيضاً كما هو واضح من الأوامر التالية.

```
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
```

إضافة مركب جديد
Adding a Boat

بعد إدخال اسم العميل وعنوانه ورقم هاتفه ثم الضغط على الزر "Add Boat" الذي يقوم بإنشاء كائن من الصنف Customer، ثم عرض النموذج AddBoat والذي أجري عليه ثلاثة تعديلات:

١- تذكر أن إجراء إنشاء الصنف AddBoat يجب أن يعدل ليقوم باستقبال متغير إشارة لكائن الصنف Customer الذي يمثل صاحب المركب.

٢- ولأن متغير إشارة كائن الصنف Customer نحتاج إليه لاحقاً لإنجاز العلاقة بين العميل والمركب، فيجب الاحتفاظ به في متغير آخر يعرف على مستوى الصنف AddBoat (المتغير aCustomer) هكذا:

```
Public Class AddBoat
      Inherits System.Windows.Forms.Form
      Private aCustomer As Customer

      Public Sub New(ByRef thisCustomer As Customer)
            MyBase.New()


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

          'Add any initialization after InitializeComponent()
          'call

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

End Sub
```

٣- وبعد إنشاء كائن من الصنف SailBoat (أو كائن من PowerBoat بناء على نوع المركب المختار)، يقيم الإجراء AssignBoatToCustomer العلاقة بين هذا الكائن وكائن الصنف Customer، ثم يستدعي الإجراء AddSailBoat (أو الإجراء AddPowerBoat)، ثم الإجراء AddNewBoat من صنف التعامل مع البيانات ممرراً إليه مؤشر كائن الصنف SailBoat (أو كائن الصنف PowerBoat) ومؤشر كائن الصنف Customer. يضيف الإجراء AddNewBoat المعرف داخل صنف التعامل مع البيانات كائن الصنف SailBoat (أو كائن الصنف PowerBoat) وكائن الصنف Customer إلى المصفوفات كما هو واضح من الأوامر التالية.

```
'Method to add a sailboat to simulated 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
      'Declare a sailboat reference variable
      Dim aSailboat As Sailboat

      '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 Sailboat instance
                 aSailboat = New Sailboat(aStateRegNo, aBoatLength, _
                        aManufacturer, aYear, keelDepth, _
                        numberOfSails, motorType)
                        'Establish the relationship in both directions

                        aSailboat.AssignBoatToCustomer(aCustomer)
                        'Add sailboat and customer to simulated database
                        CustomerAndBoatData.AddNewBoat(aSailboat, aCustomer)
                        MessageBox.Show("Customer and Sailboat Added")
                        'Clear the form
                        ClearForm()
                  End If
            End If
      End Sub
```

البحث عن عميل والمركب الذي يملكه

Finding a Customer and Boat

لقد رأينا سابقاً كيف يعمل صنف واجهة الاستخدام FindCustomer للبحث عن عميل وعرض عنوانه ورقم هاتفه حيث يختار المستخدم اسم العميل ثم يضغط على الزر Find ، فعندئذ يظهر كل من عنوان العميل ورقم هاتفه ،

ولأنه توجد علاقة بين العميل والمركب نستطيع عرض بيانات العميل والمركب الذي يملكه في آنٍ واحد. سوف نتعلم في هذا الموضوع كيف نعدل الصنف FindCustomer لعرض بيانات كل من العميل والمركب الذي يملكه معاً.

سوف نعدل تصميم صنف واجهة الاستخدام FindCustomer ليحتوي على عنصر Label (lblBoatInfo) الذي يستخدم لعرض بيانات المركب الذي يملكه العميل المراد البحث عنه ، وكما فعلنا سابقاً يستخدم الإجراء btnFind_Click الخاصية SelectedIndex لمعرفة فهرس اسم العميل المختار في القائمة ثم استخدامه لاسترجاع كائن الصنف Customer المناظر له في مصفوفة العملاء، ثم نستدعي إجراءات المرور لاسترجاع كل من عنوان العميل ورقم هاتفه، ثم عرضهما في الصناديق النصية كما هو واضح من الأوامر التالية.

```
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()
```

ثم نستدعي إجراء المرور GetBoat من كائن الصنف Customer لاسترجاع كائن الصنف Boat كما هو واضح من الأوامر التالية.

```
'Retrieve this customer's boat reference
aBoat = aCustomer.GetBoat()
```

ثم نستدعي الإجراء TellAboutSelf من كائن الصنف Boat لاسترجاع معلومات المركب وإسنادها إلى الخاصية Text للعنوان lblBoatInfo كما هو واضح من الأوامر التالية.

```
'Display the boat information
lblBoatInfo.Text = aBoat.TellAboutSelf()
```

إن استخدام الإجراء TellAboutSelf هنا يمثل تطبيقاً هاماً لمفهوم "تعدد تعريف الإجراءات" (Polymorphism) حيث إننا نستدعي هذا الإجراء من متغير إشارة من النوع Boat ولا نعلم هل الكائن المشار إليه بهذا المتغير معرف من الصنف SailBoat أم من الصنف PowerBoat، ولأن كلا الصنفين يحتويان على نفس الإجراء فكل منهما سيستجيب

إلى هذا الاستدعاء بأسلوب مختلف عن الآخر حيث يستجيب كائن الصنف SailBoat بإرجاع بيانات عن المركب الشراعي، بينما يستجيب كائن الصنف PowerBoat بإرجاع بيانات عن المركب الآلي.

يعرف الصنف الرئيس Boat الإجراء TellAboutSelf الذي يعيد الصفات المعرفة لديه، ويعرف كل من الصنف Sailboat والصنف Powerboat الإجراء TellAboutSelf الذي يستدعي أولاً الإجراء TellAboutSelf المعرف في الصنف Boat لاستقبال قيم الصفات المشتركة ثم يزيد عليها قيم الصفات الخاصة به. توضح الأوامر التالية تعريف الإجراء TellAboutSelf المعرف داخل الصنف Sailboat حيث يتم استدعاء الإجراء TellAboutSelf من الصنف الرئيس Boat باستخدام الكلمة المحجوزة MyBase، ثم يتم إضافة الكلمة "Sail Boat:" في مقدمة النص العائد وإضافة قيم صفات المركب الشراعي في نهاية النص العائد وإعادة النص كاملاً.

```
'TellAboutSelf overrides and invokes superclass method
Public Overrides Function TellAboutSelf() As String
       Return ("Sailboat: " & _
       MyBase.TellAboutSelf() & ", Keel depth " _
       & keelDepth & ", " & numberSails & " sails, " _
       & motorType)
End Function
```

توضح الأوامر التالية تعريف الإجراء TellAboutSelf داخل الصنف الرئيس Boat حيث يتم تكوين نص من قيم الصفات المعرفة على مستوى الصنف Boat وإعادته.

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

ملخص الفصل
Chapter Summary

- إن عملية تصميم النماذج التي يطلق عليها البرمجة المرئية تنتج المتغيرات الضرورية فقط لإنشاء واجهة الاستخدام، ولكن توجد هناك حاجة إلى تعريف متغيرات أخرى تستخدم للتفاعل مع المستخدم عند معالجة الأحداث والتي يجب أن تضاف إلى شفرة البرنامج، ولأن معظم إجراءات معالجة الأحداث تحتاج إلى هذه المتغيرات، فإنه يجب تعريفها على مستوى مجال صنف واجهة الاستخدام (Class Scope).

- يمكننا التحقق من طول القيمة النصية المدخلة (كائن الصنف String) لمعرفة إذا كان المستخدم أدخل قيمة نصية داخل الصناديق النصية أم لا ، فإذا كان طول النص يساوي صفراً فهذا يعني أن المستخدم لم يدخل شيئاً في هذا الصندوق النصي.
- يعمل الصنف ArrayList مثل المصفوفة العادية ولكن له حجم متغير حيث يحتوي على الإجراء Add الذي يستخدم لإضافة عنصر إلى نهاية المصفوفة الحالية.
- لقد أوضحنا العديد من أمثلة هذه الوحدة استخدام المصفوفات لمحاكاة قواعد البيانات التي تخزن بيانات العملاء وبيانات المراكب الخاصة بهم.
- يمكن استخدام كل من أزرار الاختيار RadioButton والإطارات Panels لتغيير مظهر النموذج وذلك عند حدوث أحداث تصدر بواسطة المستخدم.
- يمكننا استخدام النماذج المتعددة (Multiple Forms) من تطوير نموذج يحتوي على قائمة رئيسة التي تسهل التنقل بين نماذج متعددة ، وبدلاً من استخدام أصناف واجهة استخدام منفصلة لإضافة بيانات عميل مثلاً أو تعديلها ، يمكن أن نستخدم نموذجاً يحتوي على قائمة رئيسة لتحميل هذه الأصناف وتنفيذها كجزء من نظام متكامل.
- يستخدم الإجراء ShowDialog لإظهار النموذج بطريقة تسمى Modal والتي تتسبب في إغلاق أو إخفاء النموذج قبل تنفيذ الأمر التالي لأمر ShowDialog ، وهذا يعني أن النموذج يجب أن يغلق أو يخفى قبل ظهور نموذج آخر داخل النظام المتعدد النماذج.
- يمكننا تصميم صنف واجهة استخدام لإضافة عميل جديد والمراكب الخاصة به في آنٍ واحد وذلك بربط صنف واجهة الاستخدام AddCustomer بصنف واجهة الاستخدام AddBoat.
- يمكننا تصميم صنف واجهة استخدام للبحث عن عميل ما وعرض معلوماته إن وجدت وعرض بيانات المراكب التي يملكها في آنٍ واحد وذلك بواسطة علاقة الترابط بين العميل والمركب.

أسئلة المراجعة

Review Questions

١- السلسلة النصية التي طولها صفر تعني :

(أ) نص غير قابل للاستعمال لأنه لا يمكن أن يكون لديك نص بدون طول.

(ب) لا توجد بيانات داخل النص.

(ج) ستحصل على استثناء.

(د) تحتاج إلى إنشاء نص.

٢- قائمة المصفوفة مثل مصفوفة الاستثناءات :

(أ) قائمة يجب أن تستعمل بأداة التحكم ListBox.

(ب) قائمة المصفوفة متغيرة أكثر منها ثابتة في الحجم.

(ج) يمكنك فرز قائمة المصفوفة.

(د) جميعها متشابهة.

٣- للانتقال من نموذج إلى نموذج داخل نظام النماذج المتعددة، ولإظهار نموذج واحد فقط في أي وقت، يمكنك .....:

(أ) استخدام إجراء ShowDialog لإظهار النموذج الأول، وفي الوقت المناسب يتم إخفاء هذا النموذج وإظهار نموذج آخر.

(ب) استخدام إجراء Show لإظهار نموذج في أي وقت.

(ج) استدعاء إجراء في نموذج الآخر.

(د) لا شيء مما سبق.

٤- إجراء ShowDialog .....:

(أ) يعرض صندوق نص.

(ب) لا يمكن استخدامه عند وجود نماذج متعددة معقدة في التطبيق الواحد.

(ج) يتأكد من أن النموذج الذي تم عرضه اختفى أو تم غلقه قبل تنفيذ الجملة القادمة.

(د) لا يوجد

٥- إجراء التعامل مع البيانات الذي بدأ عرضه في هذا الفصل .....:

(أ) يحاكي التفاعل مع قاعدة البيانات.

(ب) هو المسؤول عن إنشاء نسخة محاكاة قاعدة البيانات

(ج) يمكن أن يستدعى بتأهيله باسم الصنف.

(د) جميع ما سبق.

٦- الصفوف الثلاثة في التصميم ثلاثي الطبقات .....:

(أ) AWT و GUI و DA.

(ب) GUI و PD و OS.

(ج) GUI و PD و DA.

(د) لا شيء مما سبق.

٧- أصناف مجال المشكلة .....:

(أ) كائنات العمل وعملياته

(ب) تحاكي قاعدة البيانات

(ج) تم إنشاؤها بواسطة قائمة رئيسة.

(د) لا شيء مما سبق.

٨- أصناف التعامل مع البيانات .....:

(أ) كائنات العمل وعملياته.

(ب) تحاكي قاعدة البيانات

(ج) تم إنشاؤها بواسطة قائمة رئيسة.

(د) لا شيء مما سبق.

٩- يمكنك تمرير نسخة بالرجوع إلى .....:

(أ) إجراء البناء.

(ب) إجراء الصنف.

(ج) إجراء نسخة.

(د) جميع ما سبق.

١٠- إجراء GetAll المعروض في هذا الفصل .....:

(أ) يرجع كائن نص.

(ب) إرجاع نسخة العميل Customer.

(ج) إرجاع نسخة المركب Boat.

(د) لا شيء مما سبق.

١١- نوع البيانات لصفة المركب في صنف Customer .....:

(أ) إما Sailboat أو Powerboat.

(ب) الكائن.

(ج) المركب.

(د) لا شيء مما سبق.

١٢- يستقر إجراء TellAboutSelf في الصنف :

(أ) Boat.

(ب) Sailboat.

(ج) Powerboat.

(د) جميع ما سبق.

١٣- يستقر إجراء GetBoat في الصنف :

(أ) Customer.

(ب) Boat.

(ج) Main Menu.

(د) لا شيء مما سبق.

١٤- تستخدم خاصية SelectedIndex لحساب :

(أ) صندوق نص.

(ب) صندوق قائمة.

(ج) عنوان.

(د) لا شيء مما سبق.

١٥- يستخدم الإجراء IsNumeric في :

(أ) تحويل النص إلى قيمة رقمية.

(ب) تحويل الرقم إلى قيمة نصية.

(ج) الاختبار سواء القيمة النصية تحولت إلى قيمة رقمية.

(د) جميع ما سبق.

١٦- للتغير المستمر في ظهور النموذج، تستطيع عمل :

(أ) أزرار الاختيار والإطارات.

(ب) عبر التبويب وصفحات التبويب.

(ج) جميع ما سبق.

(د) لا شيء مما سبق.

١٧- عندما تستخدم Array List لمحاكاة التفاعل بقاعدة بيانات العميل (customer)، فهي تخزن.....:

(أ) إشارة لقاعدة بيانات العميل الفعلية.

(ب) نسخة من قاعدة بيانات العميل الفعلية.

(ج) مؤشرات كائنات العميل ليست كائنات العميل.

(د) كائن الصنف Customer.

١٨- ما هي الفائدة الرئيسة للتصميم ثلاثي الطبقات والمكون من أصناف واجهة المستخدم الرسومية GUI، وأصناف مجال المشكلة PD، وأصناف التعامل مع البياناتDA؟

(أ) استقلال كل طبقة عن الأخرى.

(ب) البرنامج الذي لا يتكون من جميع الطبقات لا يستطيع أن يستخدم في حل المشكلة.

(ج) أصناف في كل طبقة لديها علاقة مباشرة بالأصناف الموجودة في باقي الطبقات.

(د) أصناف تستطيع التفاعل مع الواجهة العامة المشتركة.

١٩- عند عرض النموذج كـ.....، يجب إخفاؤه أو غلقه قبل الجملة البرمجية التي ستنفذ في الإجراء القادم.

(أ) قائمة رئيسة.

(ب) صندوق حوار رئيس.

(ج) صندوق حوار نموذجي.

(د) كائن خاص.

٢٠- في الجملة البرمجية Private shared customers As New ArrayList()، الكلمة المحجوزة Shared والتي تعني أن المتغير Customers هو.....:

(أ) يمكن الوصول له فقط من إجراءات الصنف العامة.

(ب) أنه مشترك مع نماذج برمجة أخرى في النظام.

(ج) سهولة الوصول حتى بعد انتهاء البرنامج.

(د) الاشتراك مع برامج تعمل في الوقت نفسه.

أسئلة المناقشة

Discussion Questions

١- اذكر الأسباب التي تجبر المبرمج أن يكتب أوامر لغة VB .NET بنفسه لإنشاء أصناف واجهة الاستخدام بدلاً من الاعتماد فقط على أداة مصمم النماذج.

٢- يمكن استبدال صنف واجهة الاستخدام AddBoat بصنفي واجهة استخدام منفصلين (للمركب الشراعي والمركب الآلي). قم بتصميم هذين النموذجين، ثم اذكر مميزات استخدام هذين النموذجين عن التصميم المعرف داخل هذا الفصل.

٣- في حال إمكانية استخدام زر الاختيار بدلاً من صندوق النص، هل تعتقد أن ذلك أفضل؟ لماذا ولماذا لا؟

٤- لقد تعلمت داخل هذا الفصل كيف تصمم نموذجاً يتم غلقه عند الضغط على الزر Close والرجوع إلى القائمة الرئيسة. قم بتصميم نموذج يحتوي على زر Close الذي يغلق هذا النموذج ويقوم بفتح نموذج آخر (أي لا يرجع إلى القائمة الرئيسة). فعلى المثال قم بفتح النموذج "FindCustomer" من النموذج "AddBoat".

٥- تقوم أمثلة هذا الفصل بإظهار نموذج في الوقت الواحد. ما هي الظروف التي تحتاج فيها التعامل مع أكثر من نموذج في نفس الوقت؟

مشاريع الفصل

Projects

١- قم بنسخ محتويات المجلد Project1 من المجلد Chap11\Projects\Proj01\Project1 المتواجد داخل ملفات بيانات الكتاب إلى المجلد Chap11\Projects\Proj01 داخل مجلد العمل الخاص بك. قم بتعريف صنف فرعي للصنف Boat يسمى Rowboat والذي يحتوي على الصفة width والصفة maximumNumberOfPassengers وعلى الإجراء TellAboutSelf. يجب أن يحتوي صنف واجهة الاستخدام AddRowBoat على صفات كل من الصنف Boat والصنف RowBoat. قم بإنشاء صنف اختبار لإنشاء كائنين من الصنف RowBoat، ثم قم باسترجاع قيم الصفات الخاصة بها وعرضها؟

٢- قم بنسخ محتويات المجلد Project4 من المجلد Chap11\Projects\Proj04\Project4 المتواجد داخل ملفات بيانات الكتاب إلى المجلد Chap11\Projects\Proj04 داخل مجلد العمل الخاص بك. لاحظ أن هذا المشروع يحتوي على صنف مجال المشكلة Customer فقط. قم بتطوير النموذج FindCustomer الذي يستخدم الحدث SelectedIndexChanged للصنف ListBox عندما يتم اختيار عميل من القائمة. وتأكد أن العميل الأول في القائمة تم اختياره عند فتح النموذج، وأن المستخدم لا يستطيع أن يغير بيانات العملاء بنص فارغ. وما أفضل طريقة يمكن اتباعها: حدث الضغط على زر من النوع Push button، أو الاعتماد على حدث تغير القائمة؟ ولماذا؟