17 Formulare vererben und Basisklassen erstellen

In diesem Kapitel lernen Sie

- wie Sie mit der Vererbungsauswahl vorhandene Formulare in Projekte aufnehmen.
- wie Sie eigene Basisklassen mit selbst definierten Eigenschaften und Methoden erstellen.
- wie Sie mit der Anweisung *Inherits* neue Klassen von Basisklassen ableiten.

Objektorientierte Programmierung (OOP) ist unter Softwareentwicklern von heute ein beliebtes Stichwort. Visual Basic 4 erweiterte die Sprache Visual Basic um einige objektorientierte Merkmale, aber nach Expertenmeinung hinkte Visual Basic den „echten" programmierorientierten Sprachen wie Microsoft Visual C++ immer noch hinter her, weil es die *Vererbung* nicht unterstützte. Vererbung ist ein Mechanismus, der es einer Klasse erlaubt, die vorhandenen Schnittstellen und Verhaltenscharakteristika einer anderen Klasse zu übernehmen. Schließlich unterstützt Visual Basic .NET die Vererbung; das heißt, Sie können in der Entwicklungsumgebung ein Formular erstellen und seine Charakteristika und Funktionalität an ein anderes Formular weitergeben. Zudem können Sie eigene Klassen definieren und Eigenschaften, Methoden und Ereignisse dieser Klassen vererben. In diesem Kapitel werden Sie mit beiden Arten der Vererbung experimentieren. Sie erfahren, wie Sie unter Verwendung eines Tools namens *Vererbungsauswahl* vorhandene Formulare in Projekte integrieren und wie Sie selbst Klassen definieren und mit der Anweisung *Inherits* neue Klassen davon ableiten. Diese Fertigkeiten werden Sie in die Lage versetzen, viele der bereits von Ihnen entwickelten Formulare und Routinen in anderen Projekten zu nutzen, sodass die Programmierung mit Visual Basic rascher und flexibler wird.

Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

Hinweise zur neuen Version: Neue Funktionen in Visual Basic .NET

Wenn Sie bereits über Erfahrungen mit Visual Basic 6 verfügen, werden Ihnen diese neuen Funktionen in Visual Basic .NET besonders auffallen:

- Die Fähigkeit, unter Verwendung des Tools *Vererbungsauswahl* in der Visual Studio-Entwicklungsumgebung Formulare zu erben.
- Klassen werden nun innerhalb der Schlüsselwörter *Public Class* und *End Class* definiert.
- Verschiedene benutzerdefinierte Klassen können nun in derselben Quelldatei gespeichert werden. (In Visual Basic 6 musste jede neue Klasse in einer eigenen Datei gespeichert werden.)
- In Visual Studio wird zum Hinzufügen von Eigenschaften zu Klassen eine neue Syntax verwendet. Die Anweisungen *Property Get*, *Property Let* und *Property Set* werden nicht mehr unterstützt.
- Das Schlüsselwort *Inherits* ermöglicht es, dass eine neue abgeleitete Klasse die Schnittstellen und Verhaltensmerkmale einer vorhandenen Klasse erbt.

Mit der Vererbungsauswahl ein Formular vererben

In der Terminologie objektorientierter Programmierung bedeutet Vererbung, dass eine Klasse die Objekte, Eigenschaften, Methoden und anderen Attribute einer anderen Klasse übernimmt. Wie bereits in Kapitel 15 erwähnt, geschieht dies in Visual Basic regelmäßig beim Erstellen neuer Formulare in der Entwicklungsumgebung. Das erste Formular eines Projekts (*Form1*) wird bei seiner Erstellung nicht physisch in das Projekt kopiert, sondern es bezieht seine Definition und seine Standardwerte von der Klasse *System.Windows.Forms.Form*. In der Tat wird diese Klasse am Beginn jedes Formulars mit dem Schlüsselwort *Inherits* angegeben, wenn ein Formular mit dem Befehl *Windows Form hinzufügen* aus dem Menü *Projekt* erstellt wird.

```
Inherits System.Windows.Forms.Form
```

Obwohl Sie sich dessen nicht bewusst waren, haben Sie die ganze Zeit schon die Vererbung zur Definition der Formulare genutzt, die zur Erstellung der Visual Basic-Anwendungen verwendet wurden.

Obwohl vorhandene Formulare auch über Programmcode vererbt werden können, haben die Entwickler von Visual Studio .NET diese Aufgabe als so wichtig erachtet, dass sie ein spezielles Tool für die Entwicklungsumgebung

Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

entworfen haben, das diesen Vorgang vereinfacht. Dieses Tool wird als *Vererbungsauswahl* bezeichnet, und Sie greifen über den Befehl *Geerbtes Formular hinzufügen* im Menü *Projekt* darauf zu. In der folgenden Übung verwenden Sie die Vererbungsauswahl, um eine zweite Kopie des Dialogfelds eines Projekts zu erstellen.

Ein einfaches Dialogfeld vererben

1. Starten Sie Visual Studio, und erstellen Sie im Ordner *C:\vbnetsfs\kap17* ein neues Visual Basic Windows-Anwendungsprojekt namens **Neu Formularvererbung.**

2. Zeigen Sie das Formular im Projekt an, und zeichnen Sie mit dem *Button*-Steuerelement zwei Schaltflächen nebeneinander an den unteren Formularrand.
3. Ändern Sie die *Text*-Eigenschaft der Schaltflächen in **OK** und **Abbrechen**.
4. Doppelklicken Sie auf die Schaltfläche *OK*, um die *Button1_Click*-Ereignisprozedur im Code-Editor anzuzeigen.
5. Geben Sie die folgende Programmanweisung ein:

```
MsgBox("Sie haben auf OK geklickt")
```

6. Zeigen Sie das Formular nochmals an, doppelklicken Sie auf die Schaltfläche *Abbrechen*, und geben Sie die folgende Programmanweisung in die *Button2_Click*-Ereignisprozedur ein:

```
MsgBox("Sie haben auf Abbrechen geklickt")
```

7. Zeigen Sie das Formular nochmals an, und weisen Sie der *Text*-Eigenschaft des Formulars die Zeichenfolge **Dialogfeld** zu.

   Sie verfügen damit über ein einfaches Formular, das als Grundlage eines Dialogfelds dienen kann. Sie können dieses Basisformular anpassen und zur Ausführung verschiedener Aufgaben verwenden, indem Sie einfach die Steuerelemente hinzufügen, die für eine bestimmte Aufgabe benötigt werden.

   Sie werden nun üben, wie sich das Formular vererben lässt. Der erste Schritt besteht hierbei im Erstellen (Kompilieren) des Projekts, weil nur Formulare, die in .EXE- oder .DLL-Dateien kompiliert wurden, vererbt werden können. Jedes Mal, wenn das Basisformular neu kompiliert wird, werden die am Basisformular vorgenommenen Änderungen an das abgeleitete (geerbte) Formular weitergegeben.

8. Klicken Sie im Menü *Erstellen* auf den Befehl *Projektmappe erstellen*.

   Visual Basic kompiliert das Projekt und erstellt eine .EXE-Datei.

9. Klicken Sie im Menü *Projekt* auf den Befehl *Geerbtes Formular hinzufügen*.

   Daraufhin wird das in Abbildung 17.1 dargestellte Dialogfeld angezeigt.

Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

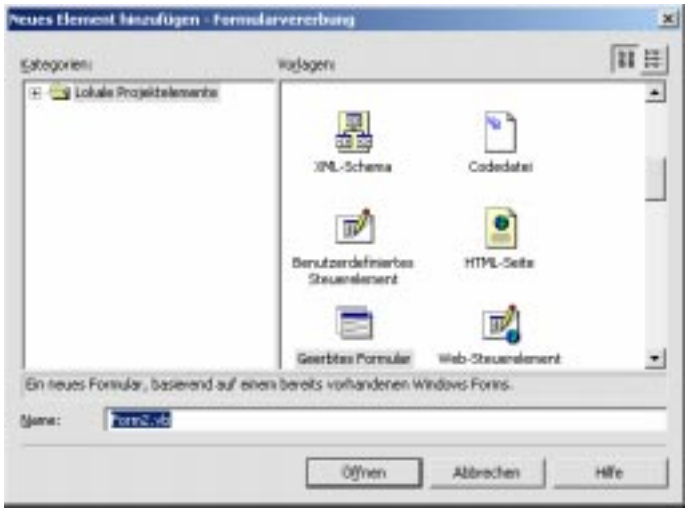

Abbildung 17.1
Das Dialogfeld *Neues Element hinzufügen*

Wie gewöhnlich listet Visual Studio alle Vorlagen auf, die Sie in ein Projekt aufnehmen könnten, und nicht nur die mit Vererbung in Beziehung stehenden Vorlagen. Da im Bereich *Vorlagen* die Option *Geerbtes Formular* bereits ausgewählt ist, müssen Sie hier keine Einstellungen ändern.

In dem Textfeld *Name* am unteren Rand des Dialogfelds können Sie dem geerbten Formular einen Namen zuweisen. Dieser Name wird im Projektmappen-Explorer angezeigt und als Dateiname des Formulars verwendet.

10 Klicken Sie auf *Öffnen*, um die Standardeinstellungen für das neue, geerbte Formular zu übernehmen.

Visual Studio zeigt das Dialogfeld *Vererbungsauswahl* an, das in Abbildung 17.2 dargestellt ist.

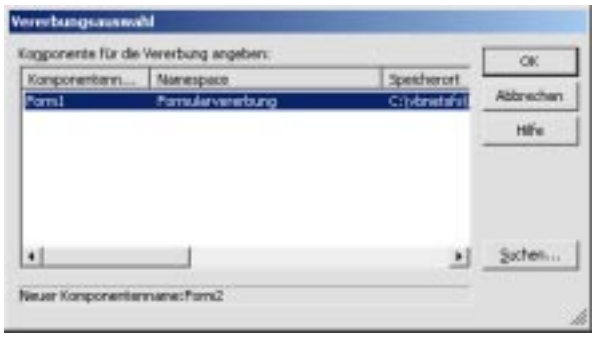

Abbildung 17.2
Das Dialogfeld *Vererbungsauswahl*

Dieses Dialogfeld enthält alle vererbbaren Formulare des aktuellen Projekts. Wenn Sie nach anderen vererbbaren Formularen suchen möchten, klicken Sie auf die Schaltfläche *Durchsuchen* und suchen auf Ihrer Festplatte nach der .DLL-Datei. (Wenn Sie ein Formular erben möchten, das nicht Teil des

Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

aktuellen Projekts ist, dann muss das Formular in eine .DLL-Datei kompiliert worden sein.)

11 Klicken Sie im Dialogfeld *Vererbungsauswahl* auf *Form1* und dann auf *OK*.

Visual Studio fügt den Eintrag *Form2.vb* in den Projektmappen-Explorer ein und zeigt das geerbte Formular im Windows Forms-Designer an. Beachten Sie, dass das Formular genauso aussieht wie das Formular, das Sie zuvor erstellt haben, abgesehen davon, dass die beiden Schaltflächen kleine Symbole enthalten, die anzeigen, dass diese Objekte aus dem geerbten Formular stammen.

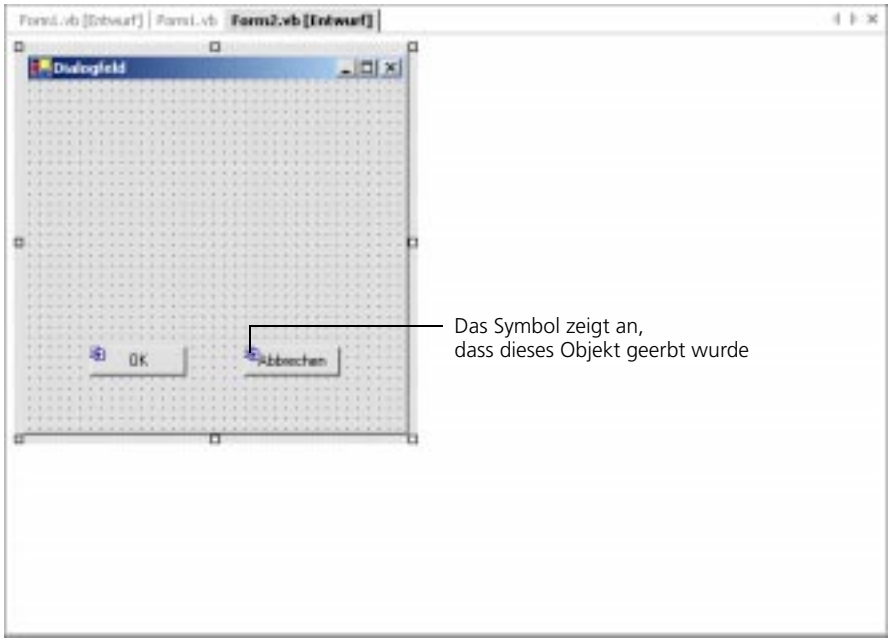

Abbildung 17.3 Das geerbte Formular im Windows Forms-Designer

Es kann mitunter schwierig sein, ein geerbtes Formular vom Basisformular zu unterscheiden (die winzigen Symbole werden leicht übersehen), daher sollten Sie stets auf die Fensterregisterkarten der Entwicklungsumgebung achten und den Projektmappen-Explorer zur Auswahl der Formulare verwenden.

Sie werden nun einige neue Elemente in das geerbte Formular einfügen.

Das geerbte Formular anpassen

1 Fügen Sie mithilfe des *Button*-Steuerelements eine dritte Schaltfläche in *Form2* (das geerbte Formular) ein.

2 Geben Sie als Wert der *Text*-Eigenschaft dieser Schaltfläche **Hier klicken!** ein.

3 Doppelklicken Sie auf diese Schaltfläche.

Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

4. Geben Sie die folgende Anweisung in die *Button3_Click*-Ereignisprozedur ein:

```
MsgBox("Dies ist das geerbte Formular!")
```

5. Zeigen Sie *Form2* wieder an, und doppelklicken Sie dann auf die Schaltflächen *OK* und *Abbrechen* dieses Formulars.

   Sie können die Ereignisprozeduren dieser geerbten Objekte nicht anzeigen oder bearbeiten, ohne einige zusätzliche Schritte auszuführen, die über den Rahmen dieses Kapitels hinausgehen. Allerdings können Sie dem Formular neue Objekte hinzufügen und es auf diese Weise anpassen.

6. Vergrößern Sie das Formular.

   Sie können auch andere Charakteristika des Formulars ändern, wie z.B. seine Größe und Position. Beachten Sie, dass in der Dropdown-Liste *Objekt* im Eigenschaftenfenster das Formular angezeigt wird, von dem das gerbte Formular abgeleitet ist.

   Sie legen nun *Form2* als Startobjekt fest.

7. Klicken Sie im Projektmappen-Explorer auf das Symbol der Anwendung, und klicken Sie dann im Menü *Projekt* auf den Befehl *Eigenschaften*.

   Das Dialogfeld *Eigenschaftsseiten* wird angezeigt.

8. Klicken Sie auf die Dropdown-Liste *Startobjekt*, dann auf *Form2* und anschließend auf *OK*.

   Nun führen Sie das Projekt aus.

9. Klicken Sie auf die Schaltfläche *Starten*.

   Wie in Abbildung 17.4 zu sehen, wird das geerbte Formular geöffnet.

Abbildung 17.4
Das geerbte Formular wird zuerst angezeigt

10. Klicken Sie auf die Schaltfläche *OK*.

    Das geerbte Formular führt die Ereignisprozedur aus, die es von *Form1* geerbt hat, und zeigt das in Abbildung 17.5 dargestellte Meldungsfeld an.

Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

Abbildung 17.5
Das geerbte Formular erzeugt dieses Meldungsfeld

11 Klicken Sie auf *OK* und dann auf die Schaltfläche *Hier klicken!*.

Das Formular *Form2* zeigt die Meldung an, dass es sich um das geerbte Formular handelt.

Das geerbte Formular wurde durch ein neues Objekt ergänzt, sodass es neben den zwei geerbten Schaltflächen über eine dritte Schaltfläche verfügt. Gratulation! Sie haben durch den Einsatz des Dialogfelds *Vererbungsauswahl* Ihre ersten Schritt im Bereich Vererbung unternommen.

12 Klicken Sie auf *OK*, um das Meldungsfeld zu schließen, und klicken Sie dann auf die Schaltfläche *Schließen*, um das Programm zu beenden.

Das Programm wird beendet, und die Entwicklungsumgebung wird wieder angezeigt.

Eigene Basisklassen erstellen

Das Tool *Vererbungsauswahl* handhabte den Vererbungsvorgang in der letzten Übung, indem es eine neue Klasse namens *Form2* im Projekt definierte. Zur Erstellung der Klasse *Form2* stellte die *Vererbungsauswahl* eine Verbindung zwischen der Klasse *Form1* des Projekts *Formularvererbung* und dem neuen Formular her. Die Klasse *Form2* sieht im Code-Editor wie folgt aus.

```
Public Class Form2
    Inherits Visual_Inheritance.Form1
.
.
.
    Private Sub Button3_Click(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles Button3.Click
        MsgBox("Dies ist das geerbte Formular!")
    End Sub
End Class
```

1 Neben der *Inherits*-Anweisung am Anfang des Formularcodes ist auch die *Button3_Click*-Ereignisprozedur, die Sie hinzugefügt haben, ein Element der neuen Klasse. Sie erinnern sich, dass die Klasse *Form1* selbst ihre grundlegenden Merkmale von der Klasse *System.Windows.Forms.Form* erbt. Daher zeigt die letzte Übung, dass eine abgeleitete Klasse (*Form2*) ihre Funktionalität von einer anderen abgeleiteten Klasse (*Form1*) erben kann, die wiederum ihre Kernfunktionalität von einer Basisklasse (*Form*) erbt, die Bestandteil des Namespace *System.Windows.Forms* des .NET Framework ist.

2 Angesichts der Tatsache, dass Klassen derart grundlegende Bausteine in Visual Basic .NET-Programmen sind, stellt sich die Frage, wie neue Klassen

Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

erstellt werden und wie diese neuen Klassen an nachfolgend abgeleitete Klassen vererbt werden. Daher beschäftigt sich der restliche Teil dieses Kapitels mit der neuen Syntax für das Erstellen von Klassen in Visual Basic .NET und mit der Frage, wie diese benutzerdefinierten Klassen an nachfolgende Klassen vererbt werden. Gleichzeitig erfahren Sie, wie nützlich das Erstellen eigener Klassen sein kann.

In Erläuterungen zum Thema Vererbung und Erstellung benutzerdefinierter Klassen kann es zu Begriffsverwirrungen kommen. Eine Reihe sehr intelligenter Informatiker hat über diese objektorientierten Programmierkonzepte einige Jahre lang nachgedacht, und es sind zahlreiche Begriffe und Definitionen für die hier behandelten Konzepte im Umlauf. Wenn Sie sich allerdings an die hier verwendete Terminologie halten, dann werden Sie feststellen, dass die Erstellung von Klassen und das Vererben derselben in Visual Basic .NET recht einfach ist und dass Sie eine Menge sinnvoller Arbeit erledigen können, indem Sie Ihren Projekten einfach einige Zeilen Programmcode hinzufügen. Dann wollen wir einmal anfangen.

Eine neue Klasse zu einem Projekt hinzufügen

Mithilfe einer benutzerdefinierten Klasse können Sie eigene Objekte in einem Programm erstellen, d.h. Objekte, die ebenso über Eigenschaften, Methoden und Ereignisse verfügen, wie die Objekte, die Sie mit den Steuerelementen aus der Toolbox in Windows Forms erstellen. Um dem Projekt eine neue Klasse hinzuzufügen, klicken Sie im Menü *Projekt* auf den Befehl *Klasse hinzufügen* und definieren die Klasse dann durch Programmcode und einige neue Visual Basic-Schlüsselwörter.

In der folgenden Übung erstellen Sie ein Programm, das neue Mitarbeiter zur Angabe des Vornamens, Nachnamens und Geburtsdatums auffordert. Sie speichern diese Daten in den Eigenschaften einer neuen Klasse namens *Person*, und Sie definieren in dieser Klasse eine Methode, die das aktuelle Alter des neuen Mitarbeiters berechnet. Sie können aus diesem Projekt lernen, wie Sie eigene Klassen erstellen und wie Sie diese Klassen in den Ereignisprozeduren Ihrer Programme verwenden.

Das Projekt *Person-Klasse* erstellen

1. Klicken Sie im Menü *Datei* auf den Befehl *Projektmappe schließen*, und erstellen Sie dann eine neue Visual Basic Windows-Anwendung namens **Neu Person-Klasse**.

2. Fügen Sie mithilfe des *Label*-Steuerelements ein langes Bezeichnungsfeld am oberen Rand des Formulars ein.

3. Zeichnen Sie mit dem *TextBox*-Steuerelement zwei breite Textfelder unter das Bezeichnungsfeld.

4. Zeichnen Sie mit dem *DateTimePicker*-Steuerelement ein *DateTimePicker*-Objekt unter die beiden Textfelder.

Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

Sie haben das *DateTimePicker*-Steuerelement zur Eingabe von Daten in Kapitel 3 verwendet. Lesen Sie in Kapitel 3 nach, wenn Sie sich die Methoden und Eigenschaften dieses Steuerelements noch einmal ansehen möchten.

5. Zeichnen Sie mit dem *Button*-Steuerelement eine Schaltfläche unter das *DateTimePicker*-Objekt.
6. Legen Sie für die Objekte des Formulars die folgenden Eigenschaften fest:

Objekt	Eigenschaft	Einstellung
Label1	Text	"Geben Sie Vornamen, Nachnamen und Geburtsdatum der Mitarbeiter ein"
TextBox1	Text	"Vorname"
TextBox2	Text	"Nachname"
Button1	Text	"Datensatz anzeigen"
Form1	Text	"Person-Klasse"

7. Ihr Formular sollte nun wie in Abbildung 17.6 aussehen.

Abbildung 17.6
Das Formular der Anwendung *Person-Klasse*

Dies ist die Benutzeroberfläche für ein Formular, das einen neuen Mitarbeiterdatendatz in einem Unternehmen erfasst. (Das Formular ist nicht mit einer

Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

Datenbank verbunden, daher kann jeweils nur ein Datensatz gespeichert werden.) Sie fügen dem Projekt nun eine Klasse hinzu, um den Datensatz zu speichern.

8 Klicken Sie im Menü *Projekt* auf den Befehl *Klasse hinzufügen*.

Visual Studio zeigt das Dialogfeld *Neues Element hinzufügen* an, das in Abbildung 17.7 dargestellt ist.

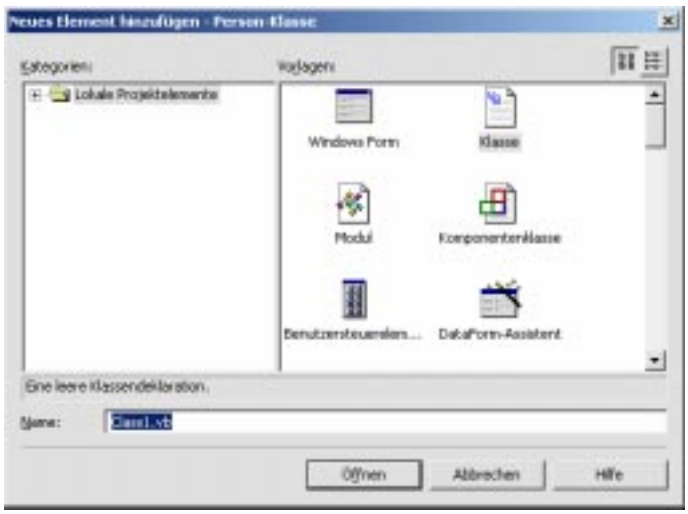

Abbildung 17.7 Das Dialogfeld *Neues Element hinzufügen*

Im Dialogfeld *Neues Element hinzufügen* können Sie die Klasse benennen. Berücksichtigen Sie bei der Namenswahl, dass Sie in einem neuen Klassenmodul mehrere Klassen speichern können. Sie sollten daher einen Namen angeben, der aussagekräftig ist.

9 Geben Sie in das Textfeld *Name* den Namen **Person** ein, und klicken Sie dann auf *Öffnen*.

Visual Studio öffnet ein leeres Klassenmodul im Code-Editor und zeigt im Projektmappen-Explorer die Datei *Person.vb* an (siehe Abbildung 17.8).

Abbildung 17.8 Das Klassenmodul wird im Projektmappen-Explorer angezeigt

Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

Sie werden nun die Definition der Klasse in das Klassenmodul eingeben und einige neue Visual Basic-Schlüsselwörter kennen lernen. Sie führen drei Schritte aus: Klassenvariablen deklarieren, Eigenschaften definieren und Methoden definieren.

Klassenvariablen deklarieren

0 Geben Sie nach der Anweisung *Public Class Person* die folgenden Variablendeklarationen ein:

```
Private Name1 As String
Private Name2 As String
```

Sie deklarieren hier zwei Variablen, die ausschließlich innerhalb dieses Klassenmoduls verwendet werden, um die Werte zweier Eigenschaften zu speichern, die Zeichenfolgen enthalten. Die Variablen wurden mit dem Schlüsselwort *Private* deklariert, weil es eine Programmierkonvention unter Visual Basic-Programmiern ist, interne Klassenvariablen als *Private* zu definieren; mit anderen Worten, sie sollen außerhalb des Klassenmoduls nicht verfügbar sein.

Eigenschaften definieren

1 Geben Sie nach den Variablendeklarationen die folgende Programmanweisung ein, und drücken Sie ↵

```
Public Property FirstName() As String
```

Mit dieser Anweisung wird in der Klasse eine Eigenschaft namens *FirstName* definiert, die vom Typ *String* ist. Sobald Sie ↵ drücken, fügt Visual Studio eine Codestruktur für die übrigen Elemente der Eigenschaftsdeklaration ein. Erforderlich sind ein Get-Block, der bestimmt, was die Programmierer sehen, wenn sie den Wert der Eigenschaft FirstName abrufen, ein Set-Block, der bestimmt, was passiert, wenn der Eigenschaft FirstName ein Wert zugewiesen oder ihr Wert geändert wird, und die Anweisung End Property, die das Ende der Eigenschaftsprozedur markiert.

In Visual Basic 6 enthielten Eigenschaftsprozeduren *Property Get-*, *Property Let-* und *Property Set-*.Codeblöcke. Diese Syntax wird nicht mehr unterstützt.

2 Geben Sie die unten in Fettschrift gedruckten Anweisungen in diese Codestruktur ein, sodass sie der unten abgedruckten Prozedur entspricht:

```
Public Property FirstName() As String
    Get
        Return Name1
    End Get
    Set(ByVal Value As String)
        Name1 = Value
    End Set
End Property
```


Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

Das Schlüsselwort *Return* gibt an, dass die String-Variable *Name1* zurückgegeben wird, wenn auf die Eigenschaft FirstName Bezug genommen wird. Der *Set-Block* weist der Variablen *Name1* eine Zeichenfolge zu, wenn der Wert der Eigenschaft festgelegt wird. Beachten Sie hier insbesondere die Variable *Value*, die in Eigenschaftsprozeduren eingesetzt wird, um den Wert zu repräsentieren, der der Klasse zugewiesen wird, wenn eine Eigenschaft festgelegt wird. Obwohl diese Syntax seltsam aussehen mag, können Sie darauf vertrauen, dass hier tatsächlich die Art und Weise dargestellt wird, wie Eigenschaftseinstellungen in Steuerelementen erzeugt werden, auch wenn bei etwas komplexeren Eigenschaften zusätzliche Programmanweisungen hier eingefügt würden, um die Gültigkeit von Werten zu testen oder Berechnungen auszuführen.

3. Geben Sie nach der Anweisung *End Property* eine zweite Eigenschaftsprozedur für die Eigenschaft *LastName* der Klasse ein. Diese Prozedur sollte wie nachfolgend abgedruckt aussehen (Sie müssen die in Fettschritt gedruckten Zeilen eingeben):

```
Public Property FirstName() As String
    Get
        Return Name1
    End Get
    Set(ByVal Value As String)
        Name1 = Value
    End Set
End Property
```

Diese Eigenschaftsprozedur ähnelt der Ersten, verwendet jedoch die zweite String-Variable (*Name2*), die Sie am Anfang der Klasse deklariert haben.

Sie haben damit die beiden Eigenschaften der Klasse *Person* definiert. Nun wenden wir uns einer Methode namens *Age* zu, die anhand des eingegebenen Geburtsdatums das aktuelle Alter des Mitarbeiters ermittelt.

Eine Methode definieren

- Geben Sie nach der Eigenschaftsprozedur *LastName* die folgende Funktionsdefinition ein:

```
Public Function Age(ByVal Birthday As Date) As Integer
    Return Int(Now.Subtract(Birthday).Days / 365.25)
End Function
```

Um eine Methode in der Klasse zu definieren, die eine bestimmte Aktion ausführt, fügen Sie der Klasse eine Function- oder Sub-Prozedur hinzu. Obwohl viele Methoden keine Argumente benötigen, erfordert die hier definierte Methode *Age* das Argument *Birthday* vom Typ *Date*, um ihre Berechnungen ausführen zu können. Die Methode subtrahiert mithilfe der Methode *Subtract* das Geburtsdatum des Mitarbeiters vom aktuellen Datum der Systemuhr, dividiert das als Anzahl von Tagen vorliegende Ergebnis durch 365,25 (die ungefähre Länge eines Jahres in Tagen) und gibt diesen Wert zurück. Die

Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

Funktion *Int* wandelt diesen Wert in eine Ganzzahl um, und diese Zahl wird, wie bei jeder anderen Funktion, über die *Return*-Anweisung der aufrufenden Prozedur übergeben. (Nähere Informationen zu Funktionsdefinitionen finden Sie in Kapitel 10.)

Die Klassendefinition ist damit vollständig. Sie kehren nun zu *Form1* zurück und verwenden die neue Klasse in einer Ereignisprozedur.

Auch wenn es in diesem Beispiel fehlt, es ist sinnvoll, in realen Projekten Klassenmodule mit irgendeiner Art von Typüberprüfung auszustatten, damit Eigenschaften und Methoden, die nicht korrekt verwendet werden, keine Laufzeitfehler auslösen, die die Programmausführung unterbrechen.

Ein auf der neuen Klasse basierendes Objekt erstellen

1. Klicken Sie im Projektmappen-Explorer auf *Form1.vb*, und klicken Sie dann auf die Schaltfläche *Designer anzeigen*.

   Die Benutzeroberfläche von *Form1* wird angezeigt.

2. Doppelklicken Sie auf die Schaltfläche *Datensatz anzeigen*, um die *Button1_Click*-Ereignisprozedur im Code-Editor anzuzeigen.

3. Geben Sie die folgenden Programmanweisungen ein:

```
Dim Employee As New Person()
Dim DOB As Date

Employee.FirstName = TextBox1.Text
Employee.LastName = TextBox2.Text
DOB = DateTimePicker1.Value.Date

MsgBox(Employee.FirstName & " " & Employee.LastName _
  & " ist " & Employee.Age(DOB) & " Jahre alt.")
```

Diese Routine speichert die vom Benutzer eingegebenen Werte in einem Objekt namens *Employee*, das als Objekt des Typs *Person* deklariert ist. Das Schlüsselwort *New* zeigt an, dass sofort eine neue Instanz des *Employee*-Objekts erzeugt werden soll. Sie haben in diesem Buch schon oft Variablen deklariert. Dies ist jedoch das erste Mal, dass Sie eine Variable deklarieren, die auf einer von Ihnen erstellten Klasse basiert. Die Routine deklariert eine *Date*-Variable namens *DOB*, in der das vom Benutzer eingegebene Datum gespeichert wird, und den Eigenschaften *FirstName* und *LastName* des *Employee*-Objekts werden die Namen zugewiesen, die von den *TextBox*-Objekten des Formulars zurückgegeben werden. Der vom *DateTimePicker*-Objekt zurückgegebene Wert wird in der Variablen *DOB* gespeichert. Mit der letzten Programmanweisung wird ein Meldungsfeld angezeigt, das die Werte der Eigenschaften *FirstName* und *LastName* sowie das von der *Age*-Methode berechnete Alter des Mitarbeiters enthält. Die *Age*-Methode gibt eine Ganzzahl zurück, wenn ihr die Variable *DOB* übergeben wird. Diese Routine zeigt, dass es einfach ist, eine eigene Klasse in einer Ereignisprozedur zu verwenden, nachdem Sie sie in einem Klassenmodul definiert haben.

Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

Sie finden das vollständige Programm *Person-Klasse* im Ordner *C:\vbnetsfs\kap17\Person-Klasse*.

4. Klicken Sie auf die Schaltfläche *Starten*, um das Programm auszuführen.

   In der Entwicklungsumgebung wird die Benutzeroberfläche des Programms *Person-Klasse* angezeigt, das bereit zur Erfassung von Eingaben ist.

5. Geben Sie Ihren Vornamen in das Textfeld *Vorname* und Ihren Nachnamen in das Textfeld *Nachname* ein.

6. Klicken Sie auf den Dropdown-Pfeil im *DateTimePicker*-Objekt, und blättern Sie zu Ihrem Geburtsdatum.

   Die Auswahl eines in der Vergangenheit liegenden Datums ist schneller, wenn Sie im geöffneten *DateTimePicker*-Dialogfeld in die Jahreszahl klicken. Daraufhin werden winzige Bildlaufleisten angezeigt, mit denen Sie jeweils ein ganzes Jahr vor- oder zurückblättern können. Sie können auch den gewünschten Monat rasch auswählen, indem Sie zuerst auf die Monatsangabe und in dem daraufhin angezeigten Popup-Menü dann auf den Monat klicken.

   Ihr Formular sollte nun wie in Abbildung 17.9 aussehen.

Abbildung 17.9 Vorname, Nachname und Geburtsdatum wurden eingegeben

7. Klicken Sie auf die Schaltfläche *Datensatz anzeigen*.

   Das Programm speichert die eingegebenen Vor- und Nachnamen als Eigenschaftseinstellungen und berechnet mithilfe der Methode *Age* das aktuelle Alter des Mitarbeiters. Das Ergebnis wird einem Meldungsfeld angezeigt.

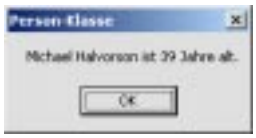

Abbildung 17.10 Das Meldungsfeld zeigt die eingegebenen Daten an

Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

8. Klicken Sie auf *OK*, um das Meldungsfeld zu schließen, und probieren Sie dann unterschiedliche Daten aus. Klicken Sie jedes Mal, nachdem Sie ein anderes Geburtsdatum eingegeben haben, auf die Schaltfläche *Datensatz anzeigen*.
9. Wenn Sie das Programm genügend ausprobiert haben, klicken Sie im Formular auf die Schaltfläche *Schließen*.

   Die Entwicklungsumgebung wird wieder angezeigt.

Einen Schritt weiter: Eine Basisklasse vererben

Wie am Anfang dieses Kapitels versprochen, lernen Sie hier einen weiteren Trick in Bezug auf benutzerdefinierte Klassen und Vererbung kennen. Ebenso wie es möglich ist, Formularklassen zu vererben, können Sie reguläre Klassen vererben, die Sie selbst mit dem Befehl *Klasse hinzufügen* in einem Klassenmodul definiert haben. Zum Vererben einer Basisklasse (oder übergeordneten Klasse) wird die *Inherits*-Anweisung verwendet, um die zuvor definierte Klasse in eine neue Klasse einzubinden. Sie können der abgeleiteten (untergeordneten) Klasse zusätzliche Eigenschaften oder Methoden hinzufügen, um sie gegenüber der Basisklasse auszuzeichnen.

In der folgenden Übung ändern Sie das Projekt *Person-Klasse* ab, indem Sie eine zweite benutzerdefinierte Klasse zu dem Klassenmodul *Person* hinzufügen. Diese neue Klasse namens *Teacher* wird die Eigenschaften *FirstName* und *LastName* sowie die Methode *Age* von der Klasse *Person* erben und über eine zusätzliche Eigenschaft namens *Grade* verfügen, in der gespeichert wird, in welcher Klasse der neue Lehrer unterrichtet.

Das Schlüsselwort *Inherits* verwenden

1. Klicken Sie im Projektmappen-Explorer auf das Klassenmodul *Person.vb*, und klicken Sie dann auf die Schaltfläche *Code anzeigen*.
2. Blättern Sie im Code-Editor ganz nach unten, sodass sich die Einfügemarke unter der Anweisung *End Class* befindet.

   Wie bereits weiter oben erwähnt, können Sie mehrere Klassen in ein Klassenmodul aufnehmen, solange jede Klasse durch die Anweisungen *Public Class* und *End Class* begrenzt wird. Sie definieren in diesem Klassenmodul eine Klasse namens *Teacher* und verwenden das Schlüsselwort *Inherits*, um die Methode und die Eigenschaften der von Ihnen definierten *Person*-Klasse darin aufzunehmen.
3. Geben Sie die folgende Klassendefinition in den Code-Editor ein. (Sie geben die in Fettschrift gedruckten Anweisungen ein, die übrigen Anweisungen werden automatisch von Visual Basic eingefügt.)

Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

```
Public Class Teacher
    Inherits Person
    Private Level As Short

    Public Property Grade() As Short
        Get
            Return Level
        End Get
        Set(ByVal Value As Short)
            Level = Value
        End Set
    End Property
End Class
```

Die *Inherits*-Anweisung verknüpft die *Person*-Klasse mit dieser Klasse, wobei alle Variablen, Eigenschaften und Methoden in die neue Klasse aufgenommen werden. Würde sich die *Person*-Klasse in einem anderen Modul oder Projekt befinden, dann könnten Sie ihren Speicherort durch eine Namespace-Angabe identifizieren; genauso wie Sie Klassen identifizieren, wenn Sie die *Imports*-Anweisung am Anfang von Programmen verwenden, die Klassen aus den Klassenbibliotheken des .NET Framework benutzen. Im Grunde genommen wurde die *Teacher*-Klasse als spezieller Typ der *Person*-Klasse definiert. Neben den Eigenschaften *FirstName* und *LastName* besitzt die *Teacher*-Klasse die Eigenschaft *Grade*, die den Schülerjahrgang verzeichnet, der von einem Lehrer unterrichtet wird.

Sie werden die neue Klasse nun in der *Button1_Click*-Ereignisprozedur verwenden.

4. Zeigen Sie die *Button1_Click*-Ereignisprozedur von *Form1* an.

Statt eine neue Variable für die *Teacher*-Klasse zu deklarieren, wird hier die Variable *Employee* verwendet. Unterschiedlich ist nur, dass jetzt der Variablen auch der Wert der *Grade*-Eigenschaft zugewiesen werden kann.

5. Ändern Sie die *Button1_Click*-Ereignisprozedur wie folgt ab (Sie müssen die in Fettschrift dargestellten Zeilen ändern):

```
Dim Employee As New Teacher()
Dim DOB As Date

Employee.FirstName = TextBox1.Text
Employee.LastName = TextBox2.Text
DOB = DateTimePicker1.Value.Date
Employee.Grade = InputBox("Welche Klasse unterrichten Sie?")

MsgBox(Employee.FirstName & " " & Employee.LastName _
  & " unterrichtet die Klasse " & Employee.Grade)
```

In diesem Beispiel wurde die Berechnung des aktuellen Alters entfernt (die *Age*-Methode wird nicht eingesetzt), damit die im Meldungsfeld ausgegebenen Daten möglichst übersichtlich sind. Sie müssen die in einer Klasse defi-

Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

nierten Eigenschaften und Methoden nicht unbedingt im Programmcode verwenden.

Sie finden das vollständige Programm *Klassenvererbung* im Ordner *C:\vbnetsfs\kap17\Klassenvererbung*.

6. Klicken Sie auf die Schaltfläche *Starten*, um das Programm auszuführen.

   Das Formular zur Eingabe der Mitarbeiterdaten wird auf dem Bildschirm angezeigt.

7. Geben Sie Ihren Vornamen in das Textfeld *Vorname* und Ihren Nachnamen in das Textfeld *Nachname* ein.

8. Klicken Sie auf das *DateTimePicker*-Objekt, und wählen Sie Ihr Geburtsdatum aus.

9. Klicken Sie auf die Schaltfläche *Datensatz anzeigen*.

   Das Programm speichert Vor- und Nachnamen als Eigenschaftseinstellungen und zeigt dann das in Abbildung 17.11 dargestellte Eingabefeld an, das den neuen Lehrer zur Angabe der unterrichteten Klasse auffordert.

Abbildung 17.11 Das Eingabefeld fordert einen Wert für die Eigenschaft *Grade* an

10. Geben Sie **3** ein, und klicken Sie auf *OK*, um das Eingabefeld zu schließen.

    Die Anwendung speichert die Zahl 3 in der neuen *Grade*-Eigenschaft und zeigt die Werte der Eigenschaften *FirstName*, *LastName* und *Grade* in einem Meldungsfeld an. Daraufhin wird das in Abbildung 17.12 dargestellte Meldungsfeld angezeigt.

Abbildung 17.12 Der eingegebene Wert wird im Meldungsfeld angezeigt

11. Probieren Sie das Programm noch mit anderen Werten aus, wenn Sie möchten, und klicken Sie dann im Formular auf die Schaltfläche *Schließen*.

    Das Programm wird beendet, und die Entwicklungsumgebung wird wieder angezeigt. Sie haben Ihre Arbeit mit Klassen und Vererbung in diesem Kapitel abgeschlossen. Prima!

Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

Weitere Experimente mit objektorientierter Programmierung

Wenn Ihnen dieser Vorgeschmack auf die objektorientierte Programmierung Spaß gemacht hat, dann können Sie sich umso mehr auf Visual Basic .NET freuen, die erste Version von Visual Basic, die es verdient, eine wahrhaft objektorientierte Programmiersprache genannt zu werden. Insbesondere sollten Sie Ereignisse zu Ihren Klassendefinitionen hinzufügen, Vorgabewerte für Eigenschaften definieren und das *Überladen von Methoden* (ein neues Feature zur Unterstützung der Polymorphie) ausprobieren. Dieses und andere OOP-Features können in der Online-Hilfe zu Visual Basic näher erforscht werden oder durch die Lektüre eines Buches zum Thema Visual Basic .NET Programmierung. (Siehe Anhang B Weitere Informationsquellen, enthält eine Literaturliste.)

Zusammenfassung

Sie möchten	dann gehen Sie wie folgt vor:
die Oberfläche und die Funktionalität eines vorhandenen Formulars erben,	Klicken Sie im Menü *Projekt* auf den Befehl *Geerbtes Formular hinzufügen*, geben Sie einen Namen für das geerbte Formular an, und klicken Sie auf *Öffnen*. Wählen Sie im Dialogfeld *Vererbungsauswahl* das gewünschte Formular aus, und klicken Sie auf *OK*. (Hinweis: Basisformulare können nur dann vererbt werden, wenn sie in eine .EXE- oder .DLL-Datei kompiliert wurden. Wenn Sie ein Formular erben möchten, das nicht Teil des aktuellen Projekts ist, dann muss das Formular in eine .DLL-Datei kompiliert worden sein.)
ein geerbtes Formular anpassen,	Fügen Sie Toolbox-Steuerelemente dem Formular hinzu, und definieren Sie die Eigenschaften. Beachten Sie, dass Sie die Eigenschaften geerbter Formularobjekte nicht ändern können. Diese Objekte sind durch kleine Symbole gekennzeichnet und sie sind deaktiviert.
eigene Basisklassen definieren,	Klicken Sie im Menü *Projekt* auf den Befehl *Klasse hinzufügen*, geben Sie einen Klassennamen ein, und klicken Sie auf *Öffnen*. Definieren Sie die Klasse mit Programmanweisungen in einem Klassenmodul.
Variablen einer Klasse deklarieren,	Verwenden Sie das Schlüsselwort Private, damit die Klassenvariablen außerhalb der Klasse nicht sichtbar sind. Beispiel: `Private Name1 As String` ▶

Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

Sie möchten	dann gehen Sie wie folgt vor:
eine neue Eigenschaft in einer Klasse definieren,	Definieren Sie eine als *Public* deklarierte Eigenschaftsprozedur in der Klasse. Beispiel:

```
Public Property FirstName() As String
    Get
        Return Name1
    End Get
    Set(ByVal Value As String)
        Name1 = Value
    End Set
End Property
```

eine neue Methode in einer Klasse definieren,	Definieren Sie eine Function- oder Sub-Prozedur in der Klasse. Beispiel:

```
Public Function Age(ByVal Birthday As Date) _
  As Integer
    Return Int(Now.Subtract(Birthday).Days _
      / 365.25)
End Function
```

eine Objektvariable vom Typ einer benutzerdefinierten Klasse deklarieren,	Geben Sie die Schlüsselwörter *Dim* und *New*, einen Variablennamen und die benutzerdefinierte Klasse in einer Programmanweisung an. Beispiel:

```
Dim Employee As New Person()
```

Eigenschaften einer Objektvariablen festlegen,	Verwenden Sie die normale Syntax zum Festlegen von Objekteigenschaften. Beispiel:

```
Employee.FirstName = TextBox1.Text
```

eine Basisklasse an eine neue Klasse vererben,	Definieren Sie eine neue Klasse, und verwenden Sie das Schlüsselwort *Inherits*, um die Definitionen einer Basisklasse darin einzubinden. Beispiel:

```
Public Class Teacher
    Inherits Person
    Private Level As Short

    Public Property Grade() As Short
        Get
            Return Level
        End Get
        Set(ByVal Value As Short)
            Level = Value
        End Set
    End Property
End Class
```


Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7

Michael Halvorson: Microsoft Visual Basic .NET Schritt für Schritt: Microsoft Press 2002. ISBN: 3-86063-782-7