Object-Oriented Programming: Classes and Objects

Chapter 1

OBJECTIVES

In this chapter you'll learn:

- How to declare a class and use it to create an object.
- How to implement a class's behaviors as methods.
- How to implement a class's attributes as instance variables and properties.
- How to use a constructor to ensure that an object's attributes are initialized.
- How to use properties to ensure that only valid data is placed in attributes.
- How composition enables you to construct classes by reusing other classes.
- To implement classwide information with `Shared` class variables.
- To use the **Object Browser** to discover the capabilities of the classes in the .NET Framework Class Library.

Introduction

- Many applications consist of one or more classes, each containing one or more methods.
- If you become part of a development team in industry, you may work on applications that contain hundreds, or even thousands, of classes.
- In this chapter, we motivate the notion of classes with real-world examples and use complete working applications to demonstrate creating your own classes and manipulating objects of those classes.

Classes and Objects

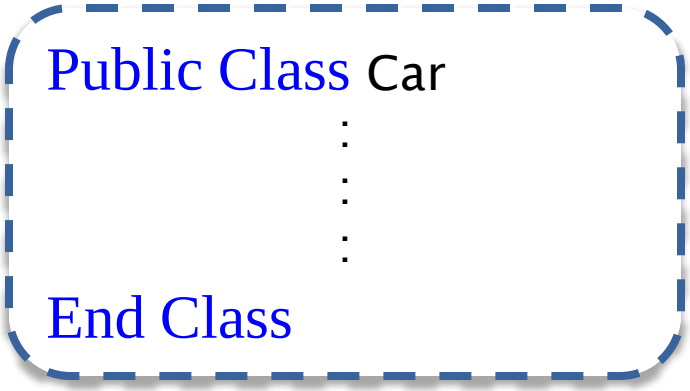
- A class is an abstract representation of something.
- An object is a usable example of the thing the class represents.
- object is a structure containing data and methods that manipulate the data.

Fields, Properties and methods

- Classes are made of **fields**, **properties** and **methods.**

- Fields and properties represent information that an object contains.
  - Fields are like variables in that they can be read or set directly.
    - Example Car color.
- Properties are retrieved and set like fields, but are implemented using property Get and property Set procedures.
- Methods represent actions that an object can perform.
  - For example, a "Car" object could have "StartEngine," "Drive," and "Stop" methods.

Class Declaration



- The keyword Public is an access modifier.
- Only `Public` classes can be reused in other projects.
- Every class declaration contains keyword Class followed immediately by the class's name.
- Every class's body ends with the keywords End Class.

Access Levels

- The access level of a declared element is the extent of the ability to access it, that is, what code has permission to read it or write to it.
  - **Public -** elements are accessible from code anywhere within the same project, from other projects that reference the project.
  - **Protected -** elements are accessible only from within the same class, or from a class derived from this class.
  - **Private -** elements are accessible only from within the same module, class, or structure.

Class Member Variable (Field)

```
Public Class Car
    Private color As String
End Class
```

- Class members declared with member access modifier Private are accessible only within the class, which gives the class complete control over how those members are used.
- Class members declared with member-access modifier Public are accessible wherever the program has a reference to an Account object.
- Member variables declared with Dim default to Private access.

Class Property

- Use property procedure when:
  - Need to control when and how a value is set or retrieved.
  - Need to validate values.
  - Setting the property causes changes to other internal variables or to the values of other properties.
- Visual Basic provides for the following property procedures:
  - A **Get** procedure returns the value of a property. It is called when you access the property in an expression.
  - A **Set** procedure sets a property to a value, including an object reference. It is called when you assign a value to the property.
- You usually define property procedures in pairs, using the Get and Set statements, but you can define either procedure alone if the property is read-only (Get Statement) or write-only (Set Statement).

Class Property

```
Dim firstName, lastName As String
Property fullName() As String
    Get
      If lastName = "" Then
          Return firstName
      Else
          Return firstName & " " & lastName
      End If

    End Get
    Set(ByVal Value As String)
        Dim space As Integer = Value.IndexOf(" ")
        If space < 0 Then
            firstName = Value
            lastName = ""
        Else
            firstName = Value.Substring(0, space)
            lastName = Value.Substring(space + 1)
        End If
    End Set
End Property
```

Class Constructor

```
Public Class Car
    Private number As Integer
    Public Sub New( ByVal  n As Integer)
        number = n
    End Sub
End Class
```

- When you create an object of a class, the class's constructor is called to initialize the object.
- Constructors must be named `New` and are generally declared `Public`.
- Constructors are implemented as `Sub` procedures, because they cannot return values.

Class Constructor

- A constructor call is required for every object that's created.
- You can provide a parameterless constructor that contains code and takes no parameters, or that takes only Optional parameters so you can call it with no arguments.

Create Object

- Each object in Visual Basic is defined by a class.
- A class describes the variables, properties, procedures, and events of an object.
- Objects are instances of classes; you can create as many objects you need once you have defined a class.

```
Dim VariableName As [New] ClassName
```

- You can also specify Public, Protected, Friend, Protected Friend, Private, Shared, or Static in the declaration.