VB : Classes and Objects

New World()

Overview

- Classes
- Objects
- Fields
- Properties
- Methods
- Constructors

Classes

- **Classes enable us to combine variables (state) and procedures (behavior) to represent the things important to our application**
  - A customer, a playing card, an invoice, a project, etc.
- **The infrastructure of our application is also represented by classes**
  - Forms in a client application, pages in a Web application, data and schema in a database, etc.
- **All code must be written inside a class**
  - A Module is a special case of a class
- **Classes contain:**
  - Fields
    - Variables that are specific to the class
  - Methods and Properties
    - Functions that operate on the data stored in the variables
  - Constructors
    - Special functions that run when an object is created
- **Members**
  - Fields, methods and properties are collectively call *members* of the class

Objects

- **Classes define types**
  - Fields
  - Properties
  - Methods
- **Objects are instances of a type**
  - Use New keyword to create instance
  - You can create multiple instances
  - Each instance holds different state
  - Each instance has same behavior

```
Dim e1 As New Employee()
e1.Name = "Rob"
Dim e2 As New Employee() With { _
    .Name = "Scott" }
```

Fields

- **Fields are variables of a class**
- **Fields are declared using the Public or Private keywords instead of Dim**
  - Public: field can be accessed from outside the class
  - Private: field cannot be access from outside the class
    - Private field data often exposed using Properties

```
Class Employee
    Public Name As String
    Public Salaried as Boolean
    Private _birthDate As DateTime
End Class
```

Properties

- **Provide access to internal fields of a class**
- **Gives developer control over access**
  - Read-only properties, validation, etc
- **Properties can also be calculated**
  - Example: age property calculated from a birth date field
- **Some features of .NET will work with properties but not with public fields**
  - Data binding for example

```
Class Employee
    Private _name As String

    Public Property Name As String
        Get
            Return _name
        End Get
        Set(value As String)
            If Not String.IsNullOrEmpty(value) Then
                _name = value
            End If
        End Set
    End Property
End Class
```

```
Class Employee
    Private _birthDate As DateTime

    Public ReadOnly Property Age As Integer
        Get
            Dim diff = DateTime.Now - _birthDate
            Return diff.Days \ 365
        End Get
    End Property
End Class
```

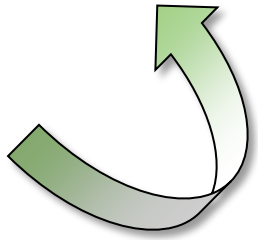
Automatic Properties

- **Define a property with an implicit backing field and get/set operations**
  - Used when you don't need special code in get/set blocks

```
Class Employee
    Private _name As String

    Public Property Name As String
        Get
            Return _name
        End Get
        Set(value As String)
            _name = value
        End Set
    End Property
End Class
```

```
Class Employee
    Public Property Name As String
End Class
```



Methods

- Procedures inside a class are called methods
- Just like Fields, they can be Private or Public
- Everything else is the same as previously discussed

```
Public Function GetDescription()
    Return String.Format( _
        "{0} is{1} a salaried employee", _
        Name, IIf(Salaried, "", " not"))
End Function
```

Constructors

- **Special methods to create objects**
  - Set default values
- **Multiple constructors allowed**
  - Overloaded methods must take different arguments

```
Class Employee
    Public Property Name As String
    Public Property Salaried As Boolean

    Public Sub New()
        Name = "<empty>"
        Salaried = False
    End Sub

    Public Sub New(ByVal name As String, _
        ByVal salaried As Boolean)

        Me.Name = name
        Me.Salaried = salaried
    End Sub
End Class
```

Summary

- **Defining classes**
  - Fields
  - Properties
  - Methods
  - Constructors
- **Creating objects**