# Chapter 2: Overview of OOP

- Class & Object
  - What is Class?
  - What is Object?
  - Property
  - Method
  - Event
  - Interface
- Three Main Features of OOP
  - Encapsulation
  - Inheritance
  - Polymorphism

Apply with Microsoft Visual Basic .Net 2005 or 2008



## Class and Object

- **What is Class?**
  - In general, **class** is a *blueprint* or *general form* of all the objects that have the same or similar properties, behavior, and relationship.
  - In programming language, **class** is a **reference type** or a **user defined data type** that encapsulates *data member, methods, properties and events*.
  - Example of Class
    - Student, Teacher, Customer, Employee
    - TextBox, Button, ComboBox, etc.

# Class and Object

- **Creating Class in MS-VB.Net 2008**
  - General Form

```
Public Class ClassName
    'Data Members
    'Properties
    'Methods
    'Events
    '.....................
End Class
```



# Class and Object

- Example: Create one class represent to student

```
Public Class Student
  Private intSysID As Long
  Private strCode As String
  Private strLastName As String
  Private strFirstName As String
  Private strGender As String

  Public Property LastName() As String
    Get
      Return strLastName
    End Get
    Set(ByVal value As String)
      strLastName = value
    End Set
  End Property

  Public Function GetName() As String
    Return strLastName & " " & strFirstName
  End Function
End Class
```

Data Members

Property

Method

## Class and Object

- **What is Object?**
  - In the real world, ***objects*** are *everything around you* including your cat, dog, table, friends, car, motorcycle, etc.
  - In object-oriented programming, ***object*** is an *instance of class*. Objects are created from class.
  - There are two types of object: *tangible object* and *intangible object.*
    - Tangible object: the object that we *can touch and see* such as student, dog.
    - Intangible object: the object that we *cannot touch and cannot see* such as job, study.

Chapter 2 5

## Class and Object

- Example of Object
  - In real world: Mr. Sovann, Mr. Sok, This Car, etc.
  - In VB.Net 2008:
    - Dim objStd as New Student
    - Dim objTextBox as New TextBox
    - Dim dt as New DataTable

    - objStd is an instance of Student class
    - objTextBox is an instance of TextBox class
    - dt is an instance of DataTable class

Chapter 2 6

## Class and Object

- **Property**
  - A *property* is some sort of data value in class.
  - A *property* is a named set of two matching methods called accessors.
    - The **set** accessor is used for assigning a value to the property. (write property)
    - The **get** accessor is used for retrieving a value from the property. (read property)
  - Example
    - Properties of Class Student are: ID, FirstName, LastName, Gender, Address, etc.



## Class and Object

- **Creating Property in MS-VB.Net 2008**
  - General Form

```
Public Property PropertyName()        As DataType
       Get
            ……
Read
       End Get
       Set(ByVal value As DataType)
            ……
write
       End Set
End Property
```

# Class and Object - Property

– Example: Create Class Student with two read-write properties – LastName and FirstName and the read-only property Name.

```
Public Class Student
    Private strLastName As String
    Private strFirstName As String

    Public Property LastName() As String
        Get
            Return strLastName
        End Get
        Set(ByVal value As String)
            strLastName = value
        End Set
    End Property
    Public Property FirstName() As String
        Get
            Return strFirstName
        End Get
        Set(ByVal value As String)
            strFirstName = value
        End Set
    End Property
    Public ReadOnly Property Name() As String
        Get
            Return strLastName & " " & strFirstName
        End Get
    End Property
End Class
```

Chapter 2 9

# Class and Object

- Create Object from the above class
- Suppose that you have one Windows Form named Form1

```
Private Sub Form1_Load(ByVal sender As Object, _
                       ByVal e As System.EventArgs) Handles Me.Load

    Dim x As New Student

    x.LastName = "Royal"
    x.FirstName = "Green"

    MsgBox("Welcome to " & x.LastName & " " & x.FirstName)
    MsgBox("Welcome to " & x.Name)

End Sub
```

Declare **object** instantiate to **Student** Class

The use of **Set** property is applied here

The use of **Get** property is applied here

Chapter 2 10

## Class and Object

- **Method**
  - A *method* is a subroutine or function. It is a piece of code that makes the object defined by class to do something.

- **Creating Method in MS-VB.Net 2008**
  - General Form

```
Private|Public|Protected Sub sub_name([parameters])
        '.....................
End Sub
```

Chapter 2 11

## Class and Object - Method

```
Private|Public|Protected Function function_name([paramters]) as DataType
    '.....................
    Return Value
End Function
```

- Example: From the above class we add one method named GetName() to return full name of student

```
Public Class Student
       '.....................
       '.....................
       Public Function GetName() As String
              Return strLastName & " " & strFirstName
       End Function
End Class
```

Chapter 2 12

# Class and Object

- **Event**
  - An *event is an action notification defined by the class.*
- **Creating Event in MS-VB.Net 2008**
  - General Form

Declare and create event

```
Public Class Class_name
    'Data Member
    Public Event Event_Name()  'use to declare event
    'Property

    Public Sub Sub_name()
      '……
      RaiseEvent Event_Name() 'use to invoke event to occur
    End Sub
End Class
```

Use object with event

```
Private WithEvents object_name As New Class_name
Private Sub object_name_Event_Name()
        '……do something here during the occurrence of event
End Sub
```



# Class and Object - Event

- Example: Insert one method named Study() to the Student class above to invoke the event

```
Public Class Student
    Private strLastName As String
    Private strFirstName As String
    Public Event onStudy()
    '……some properties
    '……some methods

    Public Sub Study()
        'do something during student study
        MsgBox("Activities of student are " & _
             "listening and asking to teacher.")
        RaiseEvent onStudy()
    End Sub
End Class
```

## Class and Object - Event

– Using the event above

```
Public Class Form1
    Public WithEvents x As Student

    Private Sub Form1_Load(ByVal sender As Object, _
       ByVal e As System.EventArgs) Handles Me.Load
        x = New Student
        x.LastName = "Royal"
        x.FirstName = "Green"
        MsgBox("Welcome to " & x.Name)
        MsgBox("Welcome to " & x.GetName)
    End Sub

    Private Sub x_onStudy() Handles x.onStudy
        MsgBox("This student is being study, " & _
              "please do not disturb him/her.")
    End Sub
End Class
```

Chapter 2 15

## Class and Object

- **Interface**
  - **Interface** in OOP is a collection of methods (only name of methods with empty body) that contract with some classes to implement it.
  - One class can implement multiple interfaces.
- **Creating Interface in VB.Net 2008**
  - General Form

```
Public Interface Interface_name
       method_name([parameter list]) as data_type
       method_name1([parameter list]) as data_type
End Interface
```

Chapter 2 16

# Class and Object - Interface

– Example: Create one interface named **IShapeMethod**. Create two classes named **Rectangle** and **Triangle** and implement to IShapeMethod Interface.

- Create Interface: **IShapeMethod**

```
Public Interface IShapeMethod
    Function Area() As Double
    Function Perimeter() As Double
End Interface
```

Interface declaretion

- Create Class: **Rectangle**
- Create Class: **Triangle**



# Class and Object - Interface

```
Public Class Rectangle
    Implements IShapeMethod
    Private intX As Double
    Private intY As Double

    Public Sub New(ByVal x As Double, ByVal y As Double)
        intX = x
        intY = y
    End Sub

    Public Function Area() As Double Implements IShapeMethod.Area
        Return (intX * intY)
    End Function

    Public Function Perimeter() As Double Implements _
                    IShapeMethod.Perimeter
        Return (intX + intY) * 2
    End Function
End Class
```

# Class and Object - Interface

```
Public Class Triangle
    Implements IShapeMethod
    Private intX As Double
    Private intY As Double
    Private intZ As Double
    Public Sub New(ByVal x As Double, ByVal y As Double, _
              ByVal z As Double)
        intX = x
        intY = y
        intZ = z
    End Sub
    Public Function Area() As Double Implements IShapeMethod.Area
        Dim p As Double = (intX + intY + intZ) / 2
        Return Math.Sqrt(p * (p - intX) * (p - intY) * _
                              (p - intZ))
    End Function

    Public Function Perimeter() As Double Implements _
                      IShapeMethod.Perimeter
        Return intX + intY + intZ
    End Function
End Class
```



# Class and Object - Interface

– Using the interface above

```
Private Sub Form1_Load(ByVal sender As Object, _
             ByVal e As System.EventArgs) _
             Handles Me.Load
        Dim mySh As IShapeMethod

        mySh = New Rectangle(10, 20)

        MsgBox("Perimeter is: " & mySh.Perimeter & _
              vbCrLf & "Area is: " & mySh.Area)

        mySh = New Triangle(3, 4, 5)

        MsgBox("Perimeter is: " & mySh.Perimeter & _
             vbCrLf & "Area is: " & mySh.Area)

End Sub
```

## Three Main Features of OOP

- **Encapsulation**
  - Encapsulation is talking about a **class** that encapsulates *data member, property, method, and event* – all encapsulates in one block. Encapsulation can use properties or methods to access data in class.
  - The need of Encapsulation is to protect or prevent data (data hiding) from accessing by other using *private, and protected*.
  - Multiple objects instantiate to one class are independence.

Chapter 2 21

## Three Main Features of OOP

- **Inheritance**
  - Inheritance between two classes (*class_1 and class_2 – Class_2 inherit to Class_1*) mean that *Class_2* can inherit any accessible properties, methods, and events of Class_1. If there are 3 accessible properties in Class_1 and 2 accessible properties in Class_2 then objects of Class_2 consists of 5 properties.
  - Inheritance provides reusability, like, adding additional features to an existing class without modifying it.

Chapter 2 22

## Three Main Features of OOP - Inheritance

- The class in which other classes are inherited is called base class, or super class. The class doing the inheriting is called the derived class (sub class).
- Consider the ISA hierarchy you will understand the concept of inheritance (specification to generalization).

- **Inheritance in VB.Net 2008**

```
Public Class Class_1
      ……
End Class

Public Class Class_2
      Inherit Class_1
      ……
End Class
```



## Three Main Features of OOP - Inheritance

- Example of Inheritance
  - Create one super class named **People** with one property (**Name**) and two methods (**Run and Walk**).
  - Create two derive class named **Student** and **Soldier**. Student consists of one property (**University**) and one method (**DisplayInfo**). Soldier consists of one property (**Region**) and one method (**Shooting**).

```
Public Class People
    Private strName As String
    Public Event onRun()

    Public Property Name() As String
        Get
            Return strName
        End Get
```

## Three Main Features of OOP - Inheritance

```
        Set(ByVal value As String)
            strName = value
        End Set
    End Property

    Public Sub Run()
        MsgBox(strName & " is running.")
        RaiseEvent onRun()
    End Sub

    Public Sub Walk()
        MsgBox("Now " & strName & " is walking.")
        Dim x As New TextBox
    End Sub
End Class
```



## Three Main Features of OOP - Inheritance

```
'Class Student
Public Class Student
    Inherits People
    Private strUniversity As String

    Public Property University() As String
        Get
            Return strUniversity
        End Get
        Set(ByVal value As String)
            strUniversity = value
        End Set
    End Property

    Public Sub DisplayInfo()
        MsgBox("Name: " & Name & vbCrLf & _
                    "University: " & strUniversity )
    End Sub
End Class
```

## Three Main Features of OOP - Inheritance

```
'Class Soldier
Public Class Soldier
    Inherits People
    Private strRegion As String
    Private intGunType As Integer
    '
    Public Property Region() As String
        Get
            Return strRegion
        End Get
        Set(ByVal value As String)
            strRegion = value
        End Set
    End Property

    Public Sub Shoot()
        MsgBox(Name & " is shooting now.")
        MsgBox("Phang! Phang! Phang!!!!!!")
    End Sub
End Class
```



## Three Main Features of OOP - Inheritance

– Using Object from Inheritance Class

```
Dim x As New Student
x.
```

DisplayInfo
Name
Run
University
Walk

## Three Main Features of OOP

- **Polymorphism**
  - Polymorphism means the ability to take more than one form. An operation may exhibit different behaviors in different instances. The behavior depends on the data types used in the operation.
  - Polymorphism is the property of object oriented languages which permits the same method to define different processes.



## Three Main Features of OOP - Polymorphism

```
Public Class Class1
        Public Function sum(ByVal x As Integer, _
                            ByVal x1 As Integer) As Integer
            Return x + x1
        End Function

        Public Function sum(ByVal x As Double, _
                            ByVal x1 As Double) As Double
            Return x + x1
        End Function

        Public Function sum(ByVal x As String, _
                    ByVal x1 As String) As String
            Return x + x1
        End Function
End Class
```