Encapsulation Sample Help

Sample Description: **Encapsulation**

Points of Interest

- **Comparing MouseUp Envents**
- **Comparing Move Methods**

Control

Gauge

For Help on Help, Press F1

Encapsulation

The encapsulation sample is designed to demonstrate the concepts of packaging related data and methods together, but also to highlight differences between how encapsulation principles can be implemented between Envelop and Visual Basic. Encapsulation simplifies the task of code maintenance. Encapsulation not only simplifies the interaction between objects, it also reduces development time when you modify the way an object carries out its task.

This Encapsulation sample was constructed using a main form, with two individual forms that each contain a gauge control. Step-by-step instructions for running the sample are located on the main form. The Envelop Way button displays a form that was constructed using object-oriented techniques. The Visual Basic Way button displays a form that was constructed using typical Visual Basic programming techniques.

Comparing MouseUp Envents

Visual Basic Way

In the VisualBasicWay form, a Gauge control was added to the form as usual. The control was automatically named "Gauge1". At this point, a MouseUp event was defined as follows:

```
Sub Gauge1_MouseUp(button As Integer, shift As Integer, x As Single, y As Single)
    Gauge1.Max = Gauge1.Width
    Gauge1.Value = x
End Sub
```

Notice that the name Gauge1 must be used in the creation of the method, as well as to reference the properties of the control. This means that the code is specific to this particular control. When copies of the control are made, the code behind each control must be changed to reflect the new name of the control.

Envelop Way

In the EnvelopWay form, a Gauge was added, however it was "abstracted" with the Abstract Object icon on the main tool bar. This means that a copy of the Gauge control was made and given a name. The MouseUp event was defined as follows:

```
Sub MouseUp(button As Integer, shift As Integer, x As Single, y As Single)
     Max = Width
     Value = x
End Sub
```

Notice that there is no control name in the code or in the event handler. This method has actually been added to the abstracted gauge, not to the form. When copies of this gauge are made on the form, each one will work without any modification.

Comparing Move Methods

Visual Basic Way

The Move Left and Move Right buttons on the main form call the following code. Notice how you must include the entire name of the form as well as the exact name of each of the controls, just to get a desired behavior.

```
        Sub BtnVisualBasicRight_Click()

              VisualBasicWay.Gauge1.Move VisualBasicWay.ScaleWidth - VisualBasicWay.Gauge1.Width - 
VisualBasicWay.Gauge1.Left, VisualBasicWay.Gauge1.Top, VisualBasicWay.Gauge1.Width, VisualBasicWay.Gauge1.Height

        End Sub

        Sub BtnVisualBasicLeft_Click()

              VisualBasicWay.Gauge1.Move VisualBasicWay.ScaleWidth - (VisualBasicWay.Gauge1.Left + 
VisualBasicWay.Gauge1.Width), VisualBasicWay.Gauge1.Top, VisualBasicWay.Gauge1.Width, VisualBasicWay.Gauge1.Height

        End Sub
```

Notice that you can invoke one of Visual Basic's standard methods from another form, such as the Move method. You can't invoke a newly created method on another form unless you define your method as a global.

Envelop Way

There are a couple of methods, MoveGaugesLeft and MoveGaugesRight, on the EnvelopWay form. The main form calls these methods from the Left and Right buttons as follows:

```
         Sub BtnEnvelopLeft_Click()
              EnvelopWay.MoveGaugesLeft
         End Sub
```

The EnvelopWay form has the corresponding method as defined below:

```
         Sub MoveGaugesLeft
                Controls.MoveLeft
         End Sub
```

Notice how this method simply sends a MoveLeft instruction to all controls on the form. If a control has a corresponding method, it is executed, otherwise it is ignored by the control. The abstracted gauge, named EnvelopGauge, has the following method defined on it:

```
         Sub MoveLeft
                Move Parent.ScaleWidth - (Width + Left), Top, Width, Height
         End Sub
```

Notice how only the Move method does not need to be qualified and it gets its parent form's (EnvelopWay form) sizes using the "Parent" qualifier. When the Left button on the main form is clicked, a custom method on the EnvelopWay form is executed. This method basically executes an instruction on all the controls on the form. The gauge control named EnvelopGauge contains that method and executes it by getting width information from its parent form.

This example demonstrates that although both products encapsulate methods and data, only the object-oriented product does it in a generic and truly reusable way. You have also seen an example of how to abstract an object and "copy and specialize" it from its parent. The Controls.MoveLeft instruction shows an example of polymorphism.