

Interacción persona-computadora



8. Diseño de una aplicación Windows Forms en VB.NET

Luís Rodríguez Baena (luis.rodriguez@upsam.net)

Universidad Pontificia de Salamanca (campus Madrid)
Facultad de Informática

Programación en Windows

❑ Programación convencional (lineal)

- Acciones previsibles e independientes del entorno donde se ejecutan.
- Opciones de usuario limitadas a las posibilidades que el programador dicte.
 - ✓ El control de las opciones se hace por medio de bucles y estructuras selectivas.
- No adecuada para entornos gráficos o multitarea.
 - ✓ El número de eventos disponibles es demasiado grande para poder controlarlos todos.
 - ✓ La elección de orden del proceso de eventos es compleja.
 - ✓ La estructura de un programa lineal no facilita la espera a que se produzcan los eventos.

Programación en Windows (II)

□ Conceptos clave en la programación en Windows.

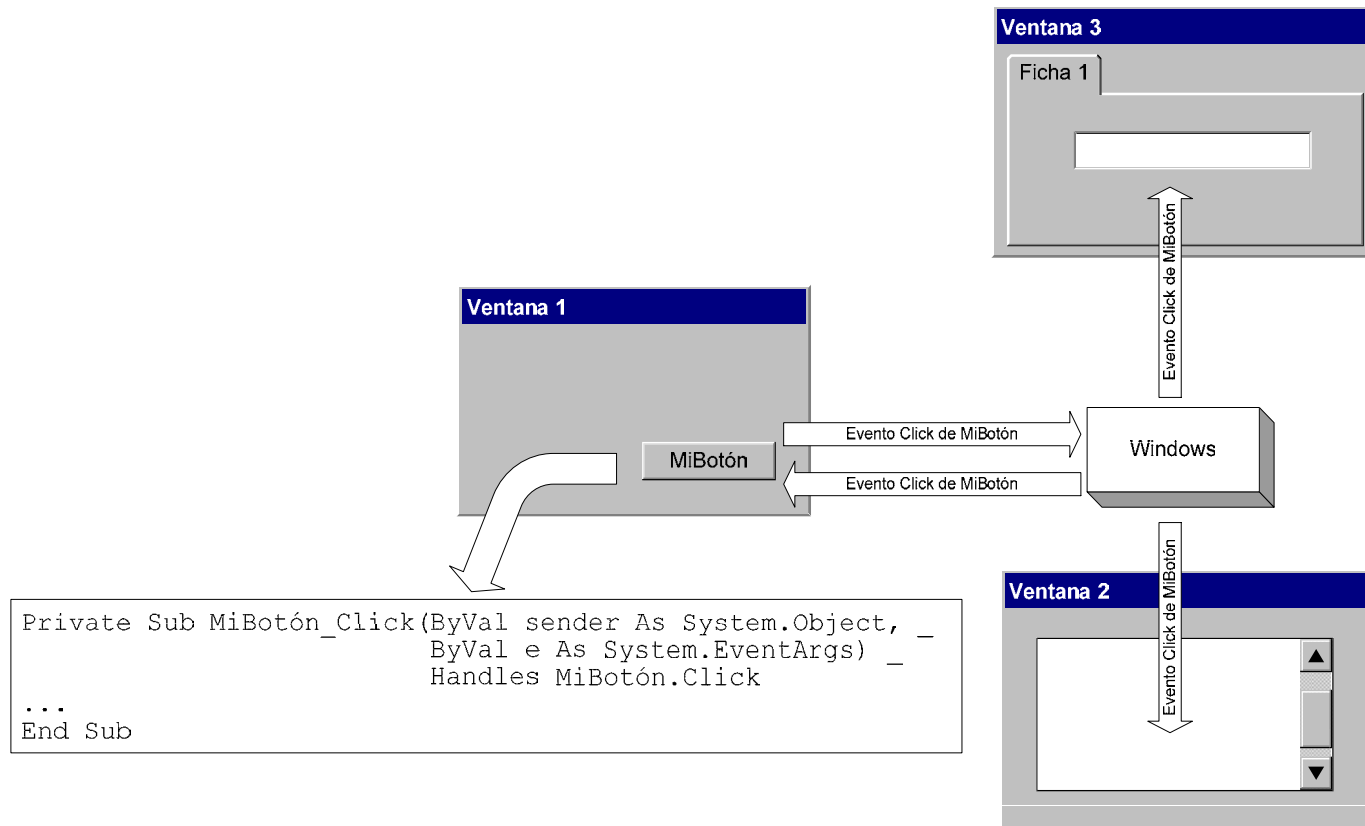
- Ventanas.
 - ✓ Región de la pantalla.
 - Ventanas de documentos, botones, listas desplegables, cuadros de diálogo.
 - ✓ El SO administra todas las ventanas asignándolas un identificador.
- Eventos.
 - ✓ Acción que se ejecuta sobre el sistema.
 - ✓ El sistema operativo rastrea continuamente las ventanas en busca de sucesos.
- Mensajes.
 - ✓ Cuando se produce un evento se envía un mensaje al sistema operativo.
 - ✓ El mensaje guarda información sobre el suceso y la ventana que lo ha producido.
 - ✓ El sistema operativo lo registra y almacena en una cola de mensajes.

Programación en Windows (III)

❑ Programación orientada a eventos

- El entorno (sistema operativo, usuario, etc.) puede actuar sobre el programa en cualquier momento.
- El programa debe responder a las acciones del entorno no proporcionadas de forma lineal.
- No se debe prever un desarrollo lineal del flujo del programa.
 - ✓ Las distintas acciones se activan como respuesta a sucesos que ocurren en el entorno.
- Al ejecutarse una aplicación basada en eventos
 - ✓ Windows rastrea las ventanas.
 - ✓ Si se detecta un evento en alguna ventana manda un mensaje al sistema operativo y lo almacena en la cola de mensajes
 - ✓ El sistema operativo lo procesa y lo transmite a las demás ventanas, indicando el evento y el identificador de la ventana que lo produce (`Handle`).
 - ✓ La aplicación busca el controlador de eventos asociado a ese evento en el control y, si existe, ejecuta el código correspondiente.

Programación en Windows (IV)



Programación en Windows (V)

❑ Estructura de un procedimiento de evento.

- Cada componente de Windows Forms tiene asociado una serie de eventos a los que responde.
 - ✓ Los controladores de eventos tienen dos argumentos:
 - `Sender`, de tipo `Object` y tiene una referencia al objeto que lo ha producido.
 - `e`, un objeto de la clase `EventArgs` o alguna de sus derivadas con información del evento.
 - ✓ El nombre corresponde con el nombre del control.
 - ✓ La cláusula `Handles` indica que métodos de eventos están asociados al procedimiento.

Programación en Windows (VI)

```
Private Sub MiEvento(ByVal sender As System.Object, _  
                    ByVal e As System.EventArgs) _  
                    Handles MiBotón.Click, MiBotón.Leave, OtroBotón.Click  
    MessageBox.Show("Se ha pulsado MiBotón, se ha pulsado OtroBotón " & _  
                  "o MiBotón a perdido el foco")  
    If sender Is MiBotón Then  
        MessageBox.Show("Se ha pulsado MiBotón o MiBotón ha perdido el foco")  
    Else  
        MessageBox.Show("Se ha pulsado OtroBotón")  
    End If  
End Sub
```



Programación en Windows (VII)

- ❑ La instrucción `AddHandler`, permite asociar un evento a un controlador de eventos determinado, distinto del procedimiento de evento predeterminado.
 - Esto permite activar y desactivar los eventos a voluntad.

`AddHandler NombreObjeto.NombreEvento, AddressOf NombreControladorDeEventos`

```
AddHandler MiBotón.Click, AddressOf MiEvento
AddHandler MiBotón.Leave, AddressOf MiEvento
AddHandler OtroBotón.Click, AddressOf MiEvento
...
Private Sub MiEvento(ByVal sender As Object, _
    ByVal e As EventArgs) 'No lleva cláusula Handles
    MessageBox.Show("Se ha pulsado MiBotón " & _
        "u OtroBotón " & _
        "o MiBotón ha perdido el foco")
    If sender Is MiBotón Then
        MessageBox.Show("Se ha pulsado MiBotón " & _
            "o mi botón ha perdido el foco")
    Else
        MessageBox.Show("Se ha pulsado OtroBotón")
    End If
End Sub
```

- ❑ La instrucción `RemoveHandler`, permite desactivar un controlador de eventos.

`RemoveHandler NombreObjeto.NombreEvento, AddressOf NombreControladorDeEventos`

Aplicaciones Windows Forms

- ❑ Se desarrolla alrededor de uno o más formularios.
- ❑ Generación automática de código.
 - Visual Studio genera código en tres sitios distintos:
 - ✓ Archivo `Application.Designer.vb`.
 - Uno por proyecto
 - Está dentro del directorio `My Project` del proyecto.
 - Incluye las características generales de la aplicación y formulario o módulo de arranque.
 - ✓ Archivo `FormX.Designer.vb`.
 - Uno por formulario.
 - Dentro del directorio de proyecto.
 - Implementación parcial de la clase `Form`.
 - Incluye el código necesario para crear y destruir los controles que se incluyan en el formulario.
 - ✓ Archivo `FormX.vb`.
 - Clase `FormX` con la declaración del resto de la clase.
 - Incluye el código de usuario para manejar la aplicación

Tareas comunes: texto

❑ Propiedad `Text`.

- Establece u obtiene el texto asociado al control.
- Presente en todos los controles que tienen texto estático o editable.
- En texto estático, el carácter `&` se utiliza para determinar la tecla de acceso.

❑ Propiedad `TextAlign`.

- Alineación del texto
- Presente en los controles `Label`, `TextBox`, `Button`, `CheckBox`, `RadioButton`, `NumericUpDown` y `DomainUpDown`.
- Para los controles `Label`, `Button`, `CheckBox` y `RadioButton` puede tomar alguno de los valores de la enumeración `ContentAlignment`.
 - ✓ `BottomCenter`, `BottomLeft`, `BottomRight`, `MiddleCenter`, `MiddleLeft`, `MiddleRight`, `TopCenter`, `TopLeft`, `TopRight`.
- Para el resto puede tomar alguno de los valores de la enumeración `HorizontalAlignment`.
 - ✓ `Center`, `Left`, `Right`.

Tareas comunes: color

- ❑ Propiedades `ForeColor` y `BackColor`.
 - Establece u obtienen el color de primer plano y el color de fondo.
 - Su valor es un dato de la estructura `System.Drawing.Color`.
- ❑ Miembros de la estructura `Color`.
 - Método estático `Color.FromArgb(rojo, verde, azul)`.
 - Método estático `Color.FromKnownColor(nombreColorConocido)`.
 - Método estático `Color.FromName(cadena)`.
 - Propiedades `R`, `G`, `B`.

```
MiBotón.BackColor = Color.FromArgb(0, 0, 255) 'Color de fondo azul
Me.BackColor = Color.FromKnownColor(KnownColor.Yellow) 'Amarillo
OtroBotón.BackColor = Color.FromName("Green") 'Color de fondo verde
Dim c As System.Drawing.Color = MiBotón.BackColor
MessageBox.Show(c.R & "-" & c.G & "-" & c.B) 'Devuelve 0-0-255
```

Tareas comunes: fuentes

❑ Propiedad `Font`.

- Hace referencia a un objeto `System.Drawing.Font`.
- En tiempo de ejecución la modificación de las características de la fuente implica la creación de una nueva instancia de la clase.

```
'Para cambiar el estilo de la fuente a negrita  
'MiBotón.Font.Bold = True no es válido  
MiBotón.Font = New Font(MiBotón.Font, FontStyle.Bold)
```

- Las fuentes de los componentes de un objeto contenedor, toman las características de los objetos contenidos.

Propiedades del objeto Font		
Propiedad	Descripción	Valores
Bold	Obtiene un valor que indica si el objeto Font está en negrita	True o False
Italic	Obtiene un valor que indica si el objeto Font está en cursiva	True o False
Name	Obtiene una representación del tipo de letra del objeto Font	Cadena
Size	Obtiene el tamaño del objeto Font	Real de simple precisión
Strikeout	Otiene un valor que indica si el objeto Font está tachado	True o False
Underline	Otiene un valor que indica si el objeto Font está subrayado	True o False
Unit	Obtiene la unidad de medida del objeto Font	Un miembro de GraphicsUnit (Inch, Millimeter, Inches, Point,...)

Tareas comunes: tamaño y posición

□ Propiedad `Location`.

- Hace referencia a una estructura de tipo `System.Drawing.Point` que identifica la posición de la esquina superior izquierda del componente.
- Estructura `Point`.
 - ✓ Constructor: `Point(X, Y)`.
 - ✓ Propiedades `X` e `Y`.
- Se puede establecer en tiempo de diseño (ventana de propiedades) o de ejecución.
 - ✓ Ejemplo

```
'Pone el botón en la esquina superior izquierda del formulario  
OtroBotón.Location = New Point(0,0)
```

Tareas comunes: tamaño y posición (II)

❑ Propiedad `Size`.

- Hace referencia a una estructura `System.Drawing.Size`.
 - ✓ Constructor: `Size(ancho, alto)`.
 - ✓ Miembros `Width` y `Height`.

```
OtroBotón.Size = New Size(100, 50)
MiBotón.Size = OtroBotón.Size
```

❑ Método `SetBounds()`.

- Establece la posición y el tamaño de un componente.

control.SetBounds(x, y, ancho, alto)

```
`Iguala el tamaño de OtroBotón a MiBotón y lo coloca en la posición 0,0
OtroBotón.SetBounds(0, 0, MiBotón.Size.Width, MiBotón.Size.Height)
`Iguala el tamaño del formulario al de la pantalla
Me.SetBounds(0, 0, Screen.PrimaryScreen.WorkingArea.Width, _
             Screen.PrimaryScreen.WorkingArea.Height)
```

Tareas comunes: tamaño y posición (III)

Propiedad `Bounds`.

- Hace referencia a una estructura de tipo `System.Drawing.Rectangle`.
 - ✓ Propiedades `X`, `Y`, `Width` y `Height`.

```
OtroBotón.Bounds = MiBotón.Bounds 'Pone a OtroBotón encima de MiBotón
```

Propiedad `ClientSize`.

- Devuelve un objeto `Size` con el tamaño del área cliente del control.

Propiedad `ClientRectangle`.

- Devuelve un objeto `Rectangle` con el rectángulo del área cliente del control.

```
OtroBotón.Bounds=New Rectangle(0,0,Me.ClientSize.Width,Me.ClientSize.Height)  
OtroBotón.Bounds = Me.ClientRectangle 'Hace lo mismo que lo anterior
```

Tareas comunes: tamaño y posición (IV)

Propiedad	Descripción	Valores
Location	Obtiene o establece el punto superior izquierdo del control	Una estructura Point
Size	Obtiene o establece el tamaño del control	Una estructura Size
Left, Top, Width, Height	Coordenadas individuales del control (obsoletas)	Un valor entero
Right	Coordenada X del borde derecho	Un valor entero
Bottom	Coordenada Y del borde inferior	Un valor entero
Bounds	Establece u obtiene el rectángulo que identifica la posición y el tamaño del control	Una estructura Rectangle
ClientRectangle	El rectángulo del área cliente del control	Una estructura Rectangle
ClientSize	Dimensiones del área cliente del control	Una estructura Size
Anchor	Distancia desde el borde del contenedor al control (ver tutorial del entorno)	Un miembro de la enumeración AnchorStyles
Dock	Establece que bordes del control se encuentran acoplados a su contenedor	Un miembro de la enumeración DockStyles
Método	Descripción	Devuelve
BringToFront	Trae el objeto a primer plano	
SendToBack	Lleva el objeto al fondo	
SetBounds(X,Y,ancho, alto)	Define el rectángulo que define la posición y tamaño del control	
SetSize(ancho, alto)	Define el tamaño que define un control	

Tareas comunes: manejo del teclado

□ Eventos `KeyPress`, `KeyDown` y `KeyUp`.

- Se ejecutan en el siguiente orden:

- ✓ `KeyDown`
- ✓ `KeyPress`
- ✓ `KeyUp`.

□ Evento `KeyPress`.

`control_KeyPress(sender As Object, e As KeyPressEventArgs)`

- `sender` es una referencia al objeto que ha enviado el evento.
- `e` es una referencia a un objeto de la clase `System.Windows.Forms.KeyPressEventArgs`.

✓ Miembros de `KeyPressEventArgs`:

- `KeyChar`, representa el carácter que se ha pulsado.
- `Handled`, un valor lógico. Si se pone a `True`, indica que el evento se ha procesado y no hay que hacer nada más.

Tareas comunes: manejo del teclado (II)

```
Private Sub TextBox1_KeyPress(ByVal sender As Object, _
                               ByVal e As System.Windows.Forms.KeyPressEventArgs) _
                               Handles TextBox1.KeyPress
    'Procesa sólo las teclas numéricas y las teclas de control
    If Not (Char.IsDigit(e.KeyChar) Or Char.IsControl(e.KeyChar)) Then
        'El motor ignora la tecla
        e.Handled = True
    End If
End Sub

Private Sub TextBox2_KeyPress(ByVal sender As Object, _
                               ByVal e As System.Windows.Forms.KeyPressEventArgs) _
                               Handles TextBox2.KeyPress
    'Convierte los caracteres alfabéticos a mayúsculas
    If Char.IsLetter(e.KeyChar) Then
        'La propiedad SelectedText devuelve el texto seleccionado
        'Puede ser una cadena nula si no hay seleccionado ningún texto
        'En ese caso será una cadena nula situada en la posición del cursor
        TextBox2.SelectedText = Char.ToUpper(e.KeyChar)
        e.Handled = True
    End If
End Sub
```

Tareas comunes: manejo del teclado (III)

□ Eventos KeyUp y KeyDown.

- Permiten detectar las teclas especiales mediante el argumento e de la clase KeyEventArgs.

✓ Miembros de KeyEventArgs:

- Handled.
- Alt, Control, Shift.
- KeyCode. **Contiene el código de la tecla pulsada, un dato la enumeración Keys (Keys.A..Keys.Z, Keys.D0..Keys.D9, Keys.F1..Keys.F2, etc.).**

```
Private Sub TextBox2_KeyDown(ByVal sender As Object, _  
                             ByVal e As System.Windows.Forms.KeyEventArgs) _  
                             Handles TextBox2.KeyDown  
    'Detecta si se ha pulsado la tecla Shift+F1  
    If e.Shift And e.KeyCode = Keys.F1 Then  
        MsgBox("Se ha pulsado Shift+F1")  
    End If  
End Sub
```

Tareas comunes: manejo del ratón

❑ Pulsación de teclas:

- Eventos `Click`, `DoubleClick`, `MouseUp`, `MouseDown` y `MouseWheel`.

❑ Movimiento del ratón.

- Eventos `MouseMove`, `MouseEnter`, `MouseLeave` y `MouseHover`.

❑ Orden de procesamiento de eventos:

1. `MouseEnter`.
2. `MouseMove`.
3. `MouseHover/MouseDown-Click-DoubleClick/MouseWheel`.
4. `MouseUp`.
5. `MouseLeave`.

Tareas comunes: manejo del ratón (II)

- ❑ `MouseMove`, `MouseDown`, `MouseWheel` y `MouseUp` reciben un argumento de la clase `MouseEventArgs`.
 - Miembros de `MouseEventArgs`.

Miembros de <code>MouseEventArgs</code>		
Propiedad	Descripción	Valores
<code>Button</code>	Obtiene el botón del ratón que se presionó.	Un miembro de la enumeración <code>MouseButtons</code> (<code>Left</code> , <code>Middle</code> , <code>None</code> , <code>Rigth</code> , <code>XButton1</code> o <code>XButton2</code>)
<code>Clicks</code>	Obtiene el número de veces que el botón del ratón se presionó y se soltó.	Un entero con el número de veces que se pulsó y soltó el botón
<code>Delta</code>	Obtiene un recuento con signo que indica el número de pasos de trinquete que ha girado la rueda del ratón. Un paso de trinquete es una muesca de la rueda del ratón.	Entero
<code>X</code>	Obtiene la coordenada x del ratón.	Entero
<code>Y</code>	Obtiene la coordenada y del ratón.	Entero

Tareas comunes: control del foco de entrada

Propiedad	Descripción	Valores
Enabled	Obtiene o establece el estado de activado o desactivado del control	Lógico
TabStop	Determina si el control va a entrar en el orden de tabulación	Lógico
TabIndex	Determina el orden en que el control va a entrar en el orden de tabulación	Entero
Visible	Obtiene o establece si un control es visible	Lógico
CausesValidation	Determina si un control va a provocar un evento de validación	Lógico
CanFocus	Determina si un control puede tomar el foco de entrada (si Visible y Enabled están a True)	Lógico
Focused	Determina si un control tiene el foco	Lógico

Método	Descripción	Valores devueltos
Focus()	Da el foco a un control	
GetNextFocus(<i>control</i> , <i>adelante</i>)	Obtiene el siguiente o anterior control en el orden de tabulación (si <i>adelante</i> es True, obtiene el siguiente)	Control
Select()	Establece el foco en un control	

Tareas comunes: control del foco de entrada (II)

- ❑ Cuando un control entra en foco se producen los siguientes eventos:
 1. Enter.
 2. GotFocus.
 3. Leave.
 4. Validating.
 5. Validated.

```
Private Sub TextBox3_Validating(ByVal sender As Object, _  
                                ByVal e As System.ComponentModel.CancelEventArgs) _  
                                Handles TextBox3.Validating  
    'Sólo permite dejar el control si se introduce un valor numérico positivo  
    If Not IsNumeric(TextBox3.Text) OrElse Cint(TextBox3.Text) <= 0 Then  
        MessageBox.Show("Se debe introducir un valor numérico mayor a 0")  
        TextBox3.Text = String.Empty  
        e.Cancel = True  
    End If  
End Sub
```

La clase Form

- ❑ Representa una ventana o cuadro de diálogo de la aplicación.
- ❑ Desde el punto de vista de la interfaz, se utilizará como un contenedor de controles.
- ❑ Desde el punto de vista de la aplicación, será un objeto heredado de la clase `Form` y que constituye el punto de entrada de la aplicación.
 - Normalmente contendrá las declaraciones y el código de la aplicación.
- ❑ En el archivo `Formx.designer.vb...`

```
Partial Class Form1
    Inherits System.Windows.Forms.Form
    'Código generado por Visual Studio con las características del formulario
    ...
End Class
```

- ❑ En el archivo `Formx.vb ...`

```
Public Class Form1
    'Código de usuario para manejar el formulario
    ...
End Class
```

La clase Form (II)

❑ Ciclo de vida de un formulario. Eventos que intervienen.

1. Evento `Load()` .

- Se produce cuando el formulario se carga por primera vez y antes de que se muestre.
- Es el lugar adecuado para meter el código necesario para inicializar variables, abrir bases de datos, dar contenido a los controles, etc.

2. Evento `Shown()`

- Se produce la primera vez que se muestra.

3. Evento `Activated()` .

- Se produce cada vez que el formulario entra en foco, ya sea por una acción del usuario o por el código del programa.
- Este evento se puede usar para la actualización del contenido con los cambios que pudieran haberse producido cuando no estaba activado.

La clase Form (III)

❑ Ciclo de vida de un formulario (*continuación*)

4. Evento `Deactivate()` .

- Se produce cuando el formulario pierde el foco.
- Puede utilizarse para actualizar el contenido de otra ventana con los datos del formulario que ha perdido el foco.

5. Evento `FormClosing()` .

- Se produce cuando se da la orden de cerrar el formulario, pero antes de que se cierre.
- Es posible cancelar la acción de cierre poniendo a `True` la propiedad `Cancel` del argumento `FormClosingEventArgs` del control.

6. Evento `FormClosed()` .

- Se produce después de haberse cerrado el formulario.
- Se puede utilizar para liberar recursos utilizados por el formulario, almacenar la información producida por él o actualizar otro formulario.

Mover y cambiar el tamaño del formulario

- ❑ Propiedades `Size`, `Location`, `Bounds`.
- ❑ Propiedades `DesktopLocation` y `DesktopBounds`.
 - Establecen la posición (un objeto `Point`) y el tamaño (un objeto `Size`) a partir del área del escritorio no ocupada por la barra de tareas.
 - ✓ Realizan acciones distintas a `Location` y `Bounds` si la barra de tareas está no esta acoplada a la parte inferior.
- ❑ Métodos `SetDesktopLocation` y `SetDesktopBounds`.
 - `SetDesktopLocation(x, y)`
 - `SetDesktopBounds(x, y, ancho, alto)`

```
'Establece la posición y el tamaño de la pantalla activa
'Screen.PrimaryScreen hace referencia a la pantalla principal
'La propiedad WorkingArea devuelve el tamaño y posición de una pantalla
Me.DesktopBounds = Screen.PrimaryScreen.WorkingArea
'Establece el tamaño del formulario a 1/4 del tamaño del escritorio
'y lo centra en el cuadrante inferior derecho del mismo
Me.SetDesktopBounds(Screen.PrimaryScreen.WorkingArea.Width / 2, _
                    Screen.PrimaryScreen.WorkingArea.Height / 2, _
                    Screen.PrimaryScreen.WorkingArea.Width / 2, _
                    Screen.PrimaryScreen.WorkingArea.Height / 2)
```

Mover y cambiar el tamaño del formulario (II)

- Métodos `CenterToScreen()` y `CenterToParent()`.
 - Centran el formulario en la pantalla y en el formulario padre (en el caso de que sea una aplicación MDI).
- Propiedad `TopMost`.
 - Asignando un valor `True`, el formulario siempre aparece por encima del resto.
- Propiedad `StartPosition`.
 - Establece la posición de inicio del formulario.

Miembros de <code>StartPosition</code>	Descripción
<code>CenterParent</code>	El formulario está centrado en los límites de su formulario principal.
<code>CenterScreen</code>	El formulario está centrado en la pantalla actual y tiene las dimensiones especificadas en el tamaño del formulario.
<code>Manual</code>	La posición del formulario viene determinado por la propiedad <code>Location</code>
<code>WindowsDefaultBounds</code>	El formulario se encuentra colocado en la ubicación predeterminada de Windows y tiene los límites establecidos por Windows de forma predeterminada.
<code>WindowsDefaultLocation</code>	El formulario se encuentra colocado en la ubicación predeterminada de Windows y tiene las dimensiones especificadas en el tamaño del formulario

Modificar el aspecto del formulario

- ❑ Propiedad `BackgroundImage`.
 - Establece la imagen de fondo del formulario.
- ❑ Propiedad `Icon`.
 - Establece el icono de la barra de títulos del formulario.
- ❑ Propiedades `ControlBox`, `MaximizeBox`, `MinimizeBox`, `HelpButton`.
 - Contienen un valor lógico que establece si el botón del menú de control, maximizar, minimizar o el botón de ayuda aparecen en el formulario.
- ❑ Propiedad `Opacity`.
 - Establece mediante un número real el nivel de transparencia de un formulario.
 - ✓ De forma predeterminada el nivel de transparencia es de 1,00.
- ❑ Propiedad `TransparencyKey`.
 - Establece el color que será transparente en el formulario.
 - ✓ `Me.TransparencyKey = Me.BackColor` 'Hace transparente el fondo del formulario.

Modificar el aspecto del formulario (II)

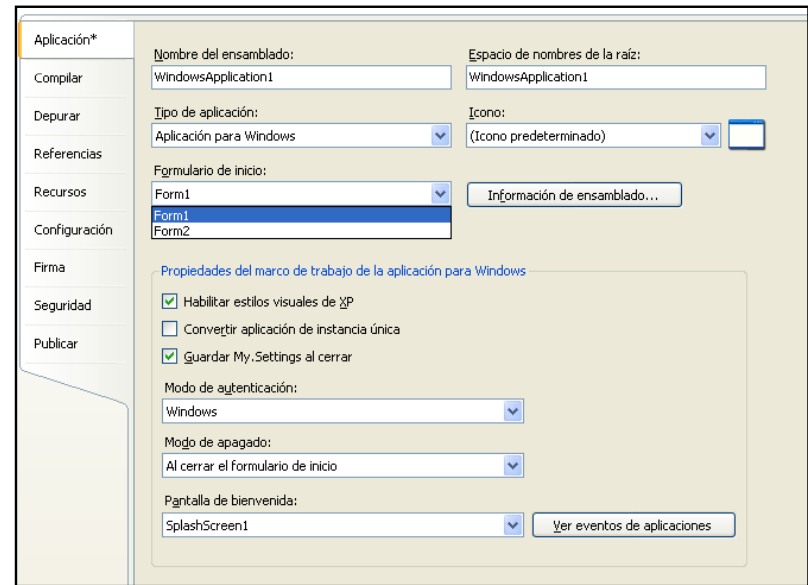
- ❑ Propiedad `FormBorderStyle`.
 - Permite tomar alguno de estos valores:

Parámetro	Descripción
None	Ninguno (ningún borde ni elemento relacionado con él). Se utiliza para los formularios de inicio.
Fixed 3D	Se utiliza cuando se desea un efecto de borde tridimensional. No se puede cambiar de tamaño. Puede incluir en la barra de título un botón de menú de control y botones Maximizar y Minimizar.
Fixed Dialog	Se utiliza para los cuadros de diálogo. Presenta un borde grueso. No se puede cambiar de tamaño. Puede incluir en la barra de título un cuadro de menú de control, y botones Maximizar y Minimizar.
Fixed Single.	No se puede cambiar de tamaño. Presenta un borde de una sola línea. Puede incluir cuadro de menú de control y botones Maximizar y minimizar. Sólo puede cambiar de tamaño con los botones Maximizar y Minimizar.
Fixed Tool Window	Se utiliza para las ventanas de herramientas. Muestra una ventana de tamaño no ajustable con un botón Cerrar y texto de barra de título con un tamaño de fuente reducido. El formulario no aparece en la barra de herramientas de Windows.
Sizable	Con frecuencia se utiliza como ventana principal. Se le puede cambiar el tamaño. Puede incluir un menú de control y botones Maximizar y Minimizar. Puede cambiar de tamaño mediante el cuadro de menú de control, los botones Maximizar y Minimizar de la barra de título, o mediante el ratón.
SizableToolWindow	Ventana de herramientas de tamaño variable. Una ventana de herramientas no aparece en la barra de tareas ni en la ventana que aparece cuando el usuario presiona ALT+TAB.

Mostrar formularios

❑ Formulario de inicio.

- Se selecciona en la página Aplicación del Diseñador de proyectos.
- Dependiendo del tipo de aplicación se puede seleccionar:
 - ✓ Para aplicaciones de consola.
 - Sub Main de un módulo.
 - ✓ Para aplicaciones Windows.
 - Cualquiera de los formularios de la lista "Formulario de inicio"
 - ✓ Biblioteca de clases.
 - No existe un objeto inicial.
- Se puede establecer mediante código en el método `Main()` mediante el método `Run` del objeto `Application`.



```
Sub Main()  
    Dim frm As New Form1  
    Application.Run(frm)  
End Sub
```

Mostrar formularios (II)

❑ Mostrar formularios secundarios no modales.

- Se debe crea una instancia del formulario y aplicar el método `Show()`.

```
'El proyecto incluye la clase Form3  
Dim frm As New Form3  
frm.Show()
```

- En el Visual Basic de .NET Framework 2.0, se puede acceder a instancias de los formularios a través del objeto `My.Forms`.

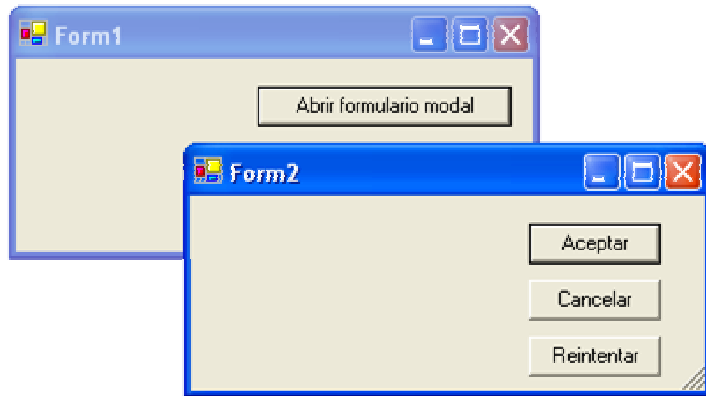
```
'El proyecto incluye la clase Form3  
My.Forms.Form2.Show()
```

❑ Mostrar formularios modales.

- Se crea una instancia del formulario y se usa el método `ShowDialog()`.
- El propietario será el formulario activo al hacer la llamada.
 - ✓ `ShowDialog()` puede pasar como argumento una referencia a otro formulario para cambiar el formulario propietario.
- `ShowDialog()` devuelve un elemento del enumerado `DialogResult`.
 - ✓ La propiedad `DialogResult` del formulario permite especificar que valor del enumerado devuelve (p.e. `Me.DialogResult = DialogResult.Yes`).
 - Al asignar esta propiedad, el formulario se cierra.

Mostrar formularios (III)

- ❑ Los formularios modales y no modales tienen distinto comportamiento:
 - En los no modales, al abrir con el método `Show`, continúa el evento que ha realizado la llamada.
 - En los modales, al abrir con el método `ShowDialog`, el evento que ha realizado la llamada se detiene hasta que se cierra el formulario modal.



```
'En el botón Aceptar de Form2
Private Sub Button1_Click
    'Esto también cierra el formulario
    Me.DialogResult = DialogResult.OK
End Sub
```

```
'En Form1...
Private Sub Button1_Click(...)...
    Dim frm As New Form2
    Dim r As DialogResult = frm.ShowDialog()
    Select Case r
        Case DialogResult.OK
            'Acciones cuando se pulsa Aceptar
        Case DialogResult.Cancel
            'Acciones cuando se pulsa Cancelar
        Case DialogResult.Retry
            'Acciones cuando se pulsa Reintentar
    End Select
End Sub
```

Compartir información entre formularios

❑ Con formularios modales.

- Se puede acceder a los controles de un formulario modal desde el formulario que lo llama.

```
Dim frm As New Form2
frm.ShowDialog()
'Accede al contenido de TextBox1 en Form2
MessageBox.Show(frm.TextBox1.Text)
```

- Se puede acceder a las variables públicas del formulario modal desde el formulario que lo llama.

```
'En Form2
Public a As Integer = 10
...
'En Form1
Dim frm As New Form2
frm.ShowDialog()
'Accede al contenido de la variable a de form2
MessageBox.Show("A = " & frm.a)
```

Compartir información entre formularios (II)

- ❑ Con formularios no modales o en las ventanas secundarias
 - Se pueden utilizar variables globales en la ventana principal o en un módulo de código.
 - ✓ Cómo no tenemos una referencia a la instancia donde está declarada la variable, hay que hacer que la variable sea compartida.

```
'En Form1
Public Shared otraVariable As Integer = 100
...
'En Form2
MessageBox.Show(Form1.otraVariable)
```

- También se pueden poner las variables en un módulo de código.

```
'En Module1
Public MásVariables as Integer = 200
...
'En Form1 o en Form2
MessageBox.Show(MásVariables)
```

Compartir información entre formularios (III)

- ❑ Acceder a la información de un formulario por medio de `My.Forms`.
 - `My.Forms` proporciona una instancia de cada formulario en el proyecto actual.
 - ✓ Para acceder a cada formulario, el nombre de la propiedad que hay que llamar es igual que el nombre de la clase que forma el formulario.
 - `My.Forms.Form1.Show()`
 - ✓ La primera vez que se accede a un formulario con `My.Forms`, se crea la instancia del mismo. Las veces siguientes, se accederá a la instancia creada anteriormente.
 - Sólo proporciona acceso a los formularios en aplicaciones Windows Forms, no en aplicaciones de consola o en formularios contenidos en DLL.
 - A partir de la instancia proporcionada es posible acceder a todos los miembros del formulario.
 - Para acceder a todos los formularios abiertos de una aplicación en un momento dado se puede utilizar la propiedad `My.Application.OpenForms`, que devuelve una colección con todos los formularios de la aplicación.

```
'Escribe en una etiqueta, el título de todos los formularios abiertos
For Each frm As Form In My.Application.OpenForms
    Label1.Text = Label1.Text & " " & frm.Text
Next
```

Compartir información entre formularios (IV)

- ❑ Ejemplo: intercambiar información entre dos formularios con `My.Forms`.



```
Public Class Form1
    'Cada vez que se pulsa el botón, el contenido del textbox pasa a form2
    Private Sub Button1_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) _
        Handles Button1.Click
        My.Forms.Form2.TextBox1.Text = TextBox1.Text
        My.Forms.Form2.Show()
    End Sub
End Class

Public Class Form2
    'Al cerrar Form2, el contenido del TextBox para a Form1
    Private Sub Form2_FormClosing(ByVal sender As Object, _
        ByVal e As _
        System.Windows.Forms.FormClosingEventArgs) _
        Handles Me.FormClosing
        My.Forms.Form1.TextBox1.Text = TextBox1.Text
    End Sub
End Class
```

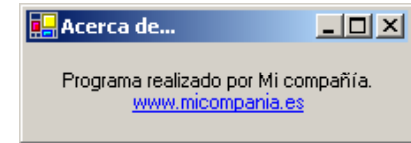
Clase Label

- ❑ Representa un campo de texto estático.
 - Propiedad `Text`.
 - ✓ Obtiene o establece el texto de la etiqueta en modo de diseño o ejecución.
 - ✓ Propiedad `TextAlign`.
 - Alineación del texto. Puede contener alguno de los valores del enumerado `ContentAlign` (`BottomCenter`, `BottomLeft`, `BottomRight`, `MiddleCenter`, `MiddleLeft`, `MiddleRight`, `TopCenter`, `TopLeft`, `TopRight`).
 - Aspecto, tamaño, posición.
 - ✓ Propiedades `Location`, `Size`, `BackColor`, `ForeColor`, `Font`.
 - ✓ Propiedad `BorderStyle`.
 - Puede tomar alguno de los valores de la enumeración `BorderStyle`: `Fixed3D`, `FixedSingle` o `None`.
 - ✓ Propiedad `AutoSize`.
 - Un valor lógico `True` hace que el tamaño de la etiqueta se adecue al tamaño del texto.
 - `True` es el valor por omisión.
 - ✓ Propiedad `Autoelipsis`.
 - Si esta a `True`, visualiza puntos suspensivos si el texto de la etiqueta se extiende más allá de su longitud.
 - ✓ Propiedad `Image`.
 - Permite establecer una imagen en la etiqueta. La propiedad `ImageAlign` permite establecer la alineación de dicha imagen.

Clase LinkLabel

❑ Hereda de Label.

- Permite establecer un enlace en la etiqueta.
- Propiedad LinkArea.
 - ✓ Establece el área de la imagen.
 - ✓ Se le debe asignar un objeto de la clase LinkArea.
 - Constructor de LinkArea: LinkArea(inicio, longitud).
- Evento LinkClicked.
 - ✓ Se produce cuando se pulsa sobre el enlace.



```
LinkLabel1.Text = "Programa realizado por Mi compañía. www.micompania.es"  
LinkLabel1.LinkArea = New LinkArea(35, 53)  
...  
Private Sub LinkLabel1_LinkClicked(ByVal sender As System.Object, _  
    ByVal e As System.Windows.Forms.LinkLabelLinkClickedEventArgs) _  
    Handles LinkLabel1.LinkClicked  
    'La orden Shell permite ejecutar un archivo del sistema  
    Shell("explorer http://www.micompania.es")  
End Sub
```

Clase TextBox

- ❑ Representa un campo de texto editable por el usuario.
 - Propiedad `Text`.
 - ✓ Permite obtener o establecer el texto del control.
 - Propiedad `TextAlign`.
 - ✓ Permite establecer la alineación del texto. Su contenido es un miembro del enumerado `HorizontalAlignment` (`Right`, `Center`, `Left`).
 - Cuadros multilínea.
 - ✓ La propiedad `Multiline` permite cuadros multilínea.
 - ✓ La propiedad `WordWrap` permite el salto de línea automático.
 - ✓ La propiedad `ScrollBars`, añade barras de desplazamiento al cuadro de texto multilinea.
 - ✓ Todas las líneas se almacenan en el array de cadenas `Lines`.

Clase TextBox (II)

❑ Modificar el contenido.

- Cada vez que se modifica el texto del control se produce el evento `TextChanged()`.
- La propiedad `Modified` se pone a `True` si el cuadro se ha modificado.
- Propiedad `CharacterCasing`, permite convertir el texto en a mayúsculas (`Upper`) o minúsculas (`Lower`).
- Propiedad `PasswordChar`.
 - ✓ Todo el texto escrito se visualiza como el carácter asignado a esa propiedad.
- Propiedad `UseSystemPasswordChar`.
 - ✓ Todo el texto escrito se visualiza como el carácter que el sistema utiliza como carácter de contraseña.
- Método `AppendText(cadena)`.
 - ✓ Añade la cadena al final del cuadro de texto.
- Propiedad `ReadOnly`.
 - ✓ Impide modificar el cuadro de texto.

Clase TextBox (III)

❑ Autocompletar el contenido de un TextBox.

- La propiedad `AutoCompleteMode` permite indicar si queremos que se autocomplete el contenido de los escrito en un TextBox:
 - ✓ `None`, no se autocompleta.
 - ✓ `Append`, al teclear los primeros caracteres añaden los que faltan.
 - ✓ `Sugest`, despliega una lista con las posibles opciones a completar.
 - ✓ `Sugest Append`, añade los que faltan y despliega la lista.
- La propiedad `AutoCompleteSource`, indica el origen de los datos a autocompletar.
 - ✓ `FileSystem` Especifica el sistema de archivos como origen.
 - ✓ `HistoryList` Incluye los URL en la lista de historial.
 - ✓ `RecentlyUsedList` Incluye los URL de la lista de las direcciones usadas recientemente.
 - ✓ `AllUrl` Especifica el equivalente de `HistoryList` y `RecentlyUsedList` como el origen.
 - ✓ `AllSystemSources` Especifica el equivalente de `FileSystem` y `AllUrl` como el origen.
 - ✓ `FileSystemDirectories` Especifica que sólo los nombres de directorio y no los nombres de archivo se finalizarán automáticamente.
 - ✓ `CustomSource` Especifica que se utilizarán las cadenas que formen la propiedad `AutoCompleteCustomSource`.

Clase TextBox (IV)

❑ Control del punto de inserción y del texto seleccionado.

- La propiedad `SelectionStart` permite obtener o establecer el punto de inserción.
 - ✓ Un valor 0 indica que el punto de inserción está antes del primer carácter.
- La propiedad `SelectionLength` permite obtener o establecer la longitud del texto seleccionado.
 - ✓ Un valor igual a 0 indica que no hay seleccionado ningún carácter.
- La propiedad `SelectedText` obtiene o establece el texto seleccionado.
 - ✓ Una cadena nula elimina indica que no hay texto seleccionado o elimina el mismo.
- Los métodos `Cut()`, `Copy()` o `Paste()`, permiten cortar, copiar o pegar el texto.
- El método `SelectAll()` selecciona todo el contenido del cuadro de texto.
- El método `Select(inicio, longitud)` permite seleccionar una porción de texto.

Clase TextBox (V)

Un cuadro de texto

Un cuadro de texto

Un texto

Un textotexto

Un textotexto

Un texto

```
TextBox1.SelectionStart = 3
```

```
TextBox1.SelectionLength = 10
```

```
TextBox1.SelectedText = ""
```

```
TextBox1.SelectionLength = 5
```

```
TextBox1.Copy()
```

```
TextBox1.SelectionStart = TextBox1.TextLength
```

```
TextBox1.Paste()
```

```
TextBox1.SelectAll()
```

```
TextBox1.Select(3, 5)
```

```
TextBox1.Cut()
```

Clase Button

- ❑ Desciende de la clase `System.Windows.Forms.ButtonBase`, de la que también descienden la clase `RadioButton` y `CheckBox`.
- ❑ La propiedad `Text`, establece la etiqueta del control y permite asignar una tecla de acceso.
- ❑ Apariencia del botón.
 - La propiedad `Image`, permite establecer una imagen para el control.
 - ✓ La propiedad `ImageAlign`, permite establecer su posición en el control y puede tomar alguno de los valores de `ContentAlignment` (véase página 12).
 - La propiedad `BackgroundImage` permite repetir una imagen a lo largo del área que ocupa el control.
- ❑ Botones por omisión.
 - Las propiedades `AcceptButton` y `CancelButton` **del formulario** permiten especificar que botón se ejecutará al pulsar ENTER o al pulsar ESC.
 - La propiedad `DialogResult`, permite especificar que valor devolverá el botón al ser pulsado en un cuadro de diálogo modal.
 - ✓ Hace lo mismo que establecer la propiedad `DialogResult` del formulario.

Clase PictureBox

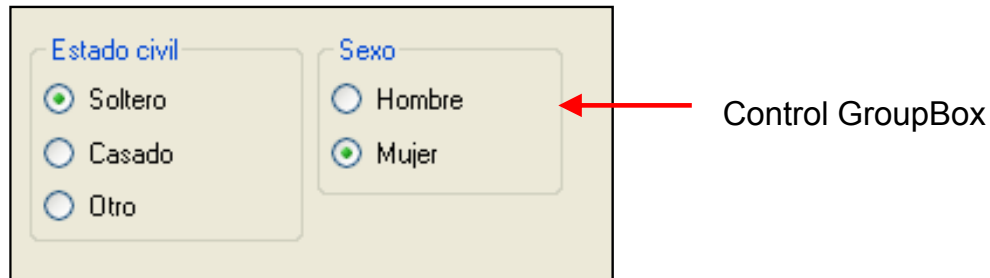
- ❑ Actúa como contenedor de imágenes de mapas de bits.
- ❑ La imagen se guarda en un objeto de la clase `Image` accesible mediante la propiedad `Image` de la clase.
 - El método `FromFile(espArchivo)` devuelve un objeto de la clase `Image` a partir de un archivo `.bmp`, `.jpg`, `.ico`, `.gif`, `.wmf` o `.png` contenido en disco.

```
PictureBox1.Image = FromFile("C:\imágenes\MiImagen.jpg")
```
 - Para eliminar dicho contenido hay que asignar a la propiedad `Image` de `PictureBox` el literal `Nothing`.

```
PictureBox1.Image = Nothing
```
- ❑ Tamaño y posición de la imagen en el control
 - La propiedad `SizeMode` permite acomodar la imagen en el control. Puede tomar alguno de los siguientes valores:
 - ✓ `AutoSize`. El tamaño del control se adecua al de la imagen.
 - ✓ `CenterImage`. La imagen se centra en el control, recortándola si es necesario.
 - ✓ `Normal`. La imagen se sitúa en la esquina superior izquierda del control, recortándola si es necesario.
 - ✓ `StretchImage`. La imagen se adapta al tamaño del control.

Clase RadioButton

- ❑ Representa un grupo de opciones excluyentes.
 - Se pueden crear grupos independientes siempre que aparezcan en contenedores distintos.
 - ✓ Los controles contenedores son el objeto `Form`, el objeto `Panel` y el objeto `GroupBox`.



- ❑ La propiedad `Checked` devuelve un valor lógico dependiendo del estado del control.
- ❑ Eventos del control.
 - Además del evento `Click` presente en casi todos los controles, es posible controlar el evento `CheckedChanged` que se produce cuando la propiedad `Checked` cambia de estado.

Clase RadioButton (II)

❑ Aspecto del control.

- Propiedad `Image`.
- Las propiedades `TextAlign`, `ImageAlign` y `CheckAlign` pueden tomar algún valor del enumerado `ContentAlignment`.
- Propiedad `FlatStyle`.
 - ✓ Puede tomar alguno de los valores del enumerado `FlatStyle`:
 - `Standard`. Aspecto tridimensional.
 - `Flat`. Aspecto plano.
 - `Popup`. Aspecto plano hasta que el cursor pasa por encima.
 - `System`. Toma el aspecto predeterminado del sistema.
- Propiedad `Appearance`.
 - ✓ `Normal`. Toma el aspecto normal.
 - ✓ `Button`. Toma el aspecto de botón.



Control RadioButton (III)

☐ Manejo del control.

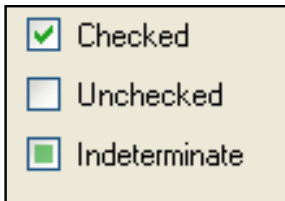
- Cambiar el tratamiento de la persona (Don o Doña) según esté marcado el RadioButton Hombre o el RadioButton Mujer.

```
Public tratamiento As String
...
Private Sub Sexo_CheckedChanged(ByVal sender As System.Object, _
                                ByVal e As System.EventArgs) _
                                Handles radHombre.CheckedChanged, _
                                radMujer.CheckedChanged

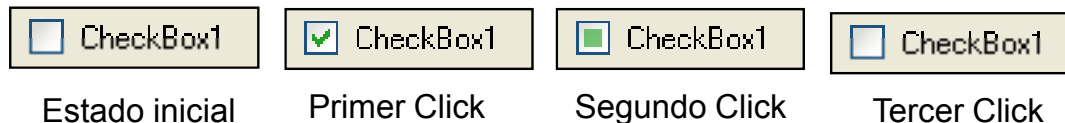
    If sender Is radHombre Then
        Tratamiento = "Don"
    Else
        Tratamiento = "Doña"
    End If
End Sub
```

Clase CheckBox

- Representa un grupo de opciones no excluyentes.
 - El control puede devolver de forma predeterminada tres estados distintos: `Checked`, `Unchecked` o `Indeterminate`.

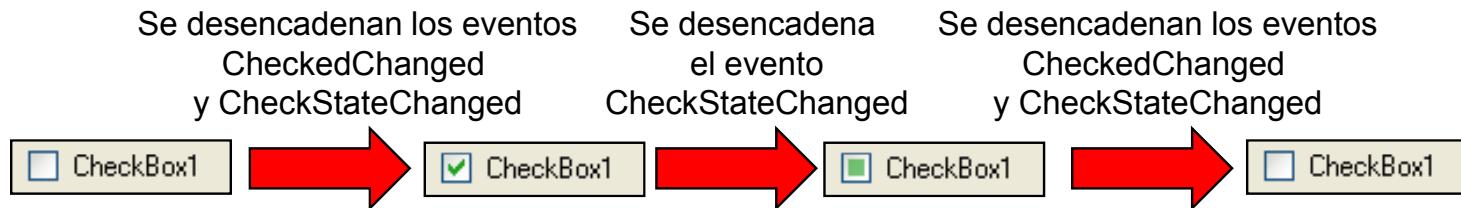


- La propiedad lógica `ThreeState` permite determinar si el control va a tener esos tres estados o únicamente `Checked` y `Unchecked`.
- La propiedad `Autocheck` permite pasar automáticamente de un estado a otro.
 - ✓ Con las propiedades `ThreeState` y `AutoCheck` a `True...`



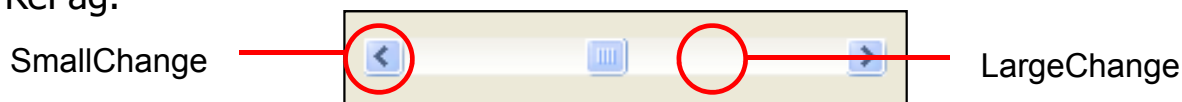
Clase CheckBox (II)

- ❑ El estado del control se puede obtener o establecer a través de las propiedades:
 - `Checked`, devuelve un valor lógico según esté o no marcado el control.
 - ✓ Devuelve `True` cuando el estado del control es `Checked` o `Indeterminate`.
 - `CheckState`, devuelve un valor del enumerado `CheckState`:
 - ✓ `CheckState.Checked`, `CheckState.Indeterminate` y `CheckState.Unchecked`.
- ❑ Eventos (se desencadenan en este orden):
 - `CheckedChanged`, se desencadena cuando cambia el estado de la propiedad `Checked`.
 - `CheckStateChanged`, se desencadena cuando cambia el valor de la propiedad `CheckState`.
 - `Click`, se desencadena cuando el usuario pulsa sobre el control.



Clases HScrollBar y VScrollBar

- ❑ Proporcionan barras de desplazamiento horizontal y vertical.
 - Se pueden utilizar para proporcionar desplazamiento en controles que no las incluyan o para asignar de forma gráfica valores numéricos.
- ❑ Propiedad `Value`.
 - Proporciona un valor entero representado por el cuadro de desplazamiento de la barra.
- ❑ Propiedades `Minimum` y `Maximum`.
 - Valores máximo y mínimo permitidos en la barra.
- ❑ Propiedad `SmallChange`.
 - Representa el incremento o decremento que se produce en la propiedad `Value` cuando se pulsa en las flechas de desplazamiento o se utilizan las teclas del cursor.
- ❑ Propiedad `LargeChange`.
 - Representa el incremento o decremento que se produce en la propiedad `Value` cuando se pulsa en el recorrido de la barra o se utilizan las teclas AvPág o RePág.



Clases HScrollBar y VScrollBar (II)

❑ Evento Scroll.

- Se produce cuando se desplaza el cuadro de desplazamiento mediante el ratón o el teclado.

❑ Evento ValueChanged.

- Se produce cuando cambia la propiedad Value.

❑ Enlazar un control ScrollBar a un cuadro de texto.



'Cuando cambia el valor de la barra de desplazamiento cambia el contenido
'del cuadro de texto

```
Private Sub HScrollBar1_ValueChanged(ByVal sender As Object, _  
                                     ByVal e As System.EventArgs) _  
                                     Handles HScrollBar1.ValueChanged  
    TextBox1.Text = HScrollBar1.Value  
End Sub
```

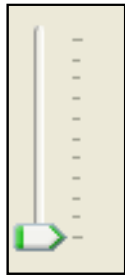
Clases HScrollBar y VScrollBar (III)

❑ Enlazar un cuadro de texto a un control ScrollBar.

```
Private Sub TextBox1_TextChanged_1(ByVal sender As System.Object, _
                                ByVal e As System.EventArgs) _
                                Handles TextBox1.TextChanged
    'Sólo se modifica el valor si es numérico
    If IsNumeric(TextBox1.Text) Then
        If CInt(TextBox1.Text) > HScrollBar1.Maximum Then
            'Si el valor introducido es mayor que el máximo
            'el valor se queda en el máximo
            HScrollBar1.Value = HScrollBar1.Maximum
        ElseIf CInt(TextBox1.Text) < HScrollBar1.Minimum Then
            'Si el valor introducido es menor que el mínimo
            'el valor se queda en el minimo
            HScrollBar1.Value = HScrollBar1.Minimum
        Else
            'En caso contrario se modifica el valor de la barra
            HScrollBar1.Value = CInt(TextBox1.Text)
        End If
    End If
End Sub
```

Clase TrackBar

- ❑ Proporciona una funcionalidad similar a las barras de desplazamiento.
- ❑ Dispone además de las siguientes propiedades.
 - Propiedad `Orientation`.
 - ✓ Permite definir la orientación (`Horizontal` o `Vertical`) del control.
 - `TickFrequency`.
 - ✓ Un entero que permite establecer la distancia entre las marcas del control.
 - `TickStyle`.
 - ✓ Un miembro del enumerado `TickStyle` que especifica que marcas aparecerán en el control.



- `BottomRight` (valor predeterminado). Las marcas aparecen abajo o a la derecha según la orientación del control.
- `None`. No aparecen las marcas.
- `Both`. Las marcas aparecen a ambos lados.
- `TopLeft`. Las marcas aparecen arriba o a la izquierda.

Clase NumericUpDown

- ❑ Proporciona un mecanismo para introducir valores numéricos.
 - Está formado por un cuadro de texto y dos flechas.



- ❑ La propiedad `Value` establece o devuelve el valor del cuadro de texto asociado.
 - Propiedad `Maximum` y `Minimum`.
 - Propiedad `Increment`. Establece o devuelve el incremento o decremento al pulsar cualquiera de las flechas.
 - Propiedad `ReadOnly`. Un valor `True` impide al usuario modificar los valores del cuadro de texto.
- ❑ Aspecto del control.
 - Propiedades `TextAlign` (se puede alinear a la izquierda derecha o centro) y `UpDownAlign` (se puede alinear a la izquierda o a la derecha).
 - Propiedad `DecimalPlaces`.
 - Propiedad `Hexadecimal`.
- ❑ Métodos `UpButton` y `DownButton`.
 - Realizan el mismo efecto que pulsar las teclas arriba o abajo.
- ❑ El cambio de valor del control se intercepta mediante el evento `ValueChanged`.

Clase DomainUpDown

- ❑ Tiene una funcionalidad y un aspecto similar a la clase `NumericUpDown`.
 - Permite establecer una cadena de texto a partir de una serie de elementos.
- ❑ Los elementos del control se guardan en la propiedad `Items`.
 - Se pueden insertar en tiempo de diseño en la ventana de propiedades.
 - Se pueden insertar en tiempo de ejecución mediante el método `Add` de la colección `Items`.
 - Se pueden eliminar en tiempo de ejecución mediante la propiedad `Remove`.
 - La propiedad `SelectedItem` devuelve el valor del elemento seleccionado del control.
 - La propiedad `SelectedIndex` devuelve o establece el índice del elemento seleccionado del control.
- ❑ Propiedad `Wrap`.
 - Un valor `True`, permite realizar un ciclo por dichos elementos (del último de la lista pasará al primero).
- ❑ El evento `SelectedItemChanged` se produce cuando cambia el valor de la propiedad `SelectedItem`.
 - Si el valor se cambia mediante el cuadro de texto se produce el evento `TextChanged`.
 - ✓ También cambia la propiedad `SelectedItem` (y se desencadena el evento `SelectedItemChanged`) a `-1`.

