Sistemas Operativos 6to

Desarrollo de Aplicaciones con Lenguajes Visuales

Docente / Martin Murdaca

Clase 5

UNIDAD 2
3er trimestre 2020

1. Presentación

Hola a todos, les recuerdo que sigo armando y pensando en cómo desarrollar la primera evaluación Obligatoria.

Les recuerdo nuevamente que es importante y **obligatorio ingresar todas las semanas por lo menos una vez,** no sólo para leer y descargar todo el material de la semana sino también para interactuar o consultar dudas con los compañeros. También sigamos conectados a través de mi dirección de correo personal martinmurdaca@gmail.com.

En la clase de hoy trabajaremos con los **controles comunes y personalizados**.

A los primeros, se los conoce como **básicos**, por ser los controles que se utilizan con mayor frecuencia en todos los proyectos. Examinaremos las propiedades, métodos y eventos principales de estos controles.

Por otro lado, veremos que se pueden **agregar más controles**, a los que aparecen en la caja de herramientas. Veremos cómo llevarlo a cabo.

Finalmente trataremos los **eventos de teclado**, que junto con los eventos del mouse, son los principales elementos de interacción del usuario con el programa. Presionar una tecla o hacer clic desencadena eventos que proporcionan el medio de introducción de datos y la manera básica de desplazarse entre menús y ventanas

Desarrollo de los contenidos

Exploración de los controles de la caja de herramientas

Controles básicos o estándares

Se los conoce como básicos por ser los controles que se utilizan con mayor frecuencia en todos los proyectos. Examinemos a continuación dichos controles.

GroupBox

Es un objeto contenedor que se utiliza para agrupar objetos relacionados. Estos pueden ser radio botones, casillas de verificación, etc.

CheckBox

La casilla de verificación se usa para especificar y seleccionar varias opciones. Un control checkbox muestra una tilde cuando esta activada, y la misma desaparece cuando el control se desactiva.

Pueden usar controles **CheckBox** en grupos para mostrar múltiples opciones entre las cuales el usuario puede seleccionar una o más.

RadioButton

Un radio botón despliega una opción que puede estar activa o no. Por lo general, este se emplea para especificar varias opciones, de las cuales solamente una de ellas puede estar seleccionada.

PictureBox Utilicen este control para mostrar un gráfico. Un control PictureBox puede mostrar un gráfico desde un mapa de bits, un icono o un metarchivo, así como un metarchivo mejorado, un archivo JPEG o archivos GIF.

ComboBox

Lista desplegable combinada, es una combinación de lo controles TextBox y ListBox. Se puede ingresar información en el cuadro de texto o se puede seleccionar un elemento de la lista.

ListBox

Un control ListBox muestra una lista de elementos entre los cuales el usuario puede seleccionar uno o más. Si el número de elementos supera el número que puede mostrarse, se agregará automáticamente una barra de desplazamiento al control ListBox.

En la siguiente tabla mencionaremos las propiedades más importantes de este grupo de controles.

Tabla I Propiedades más importantes de los controles estándares

Propiedad	Se aplica a....	Descripción	
Image	PictureBox, Radio Button y Checkbox	Indica el grafico que se mostrará en el control.	
SizeMode	PictureBox	Controla como tratará la ubicación de las imágenes y el tamaño del control.	
Text	Radio Button, Checkbox y Combobox	Establece la leyenda para estos controles.	
Checked	Radio Button y Checkbox	Devuelve o establece el estado del control. Propiedad lógica. Si está en True, el control ha sido activado, de lo contrario se encuentra desactivado.	
DropDrownStyle	Combobox	Dropdown	Incluye una lista desplegable y un cuadro de texto. Se puede seleccionar un elemento de la lista o bien teclear en el cuadro de texto. (Valor predeterminado)
		Simple	Incluye un cuadro de texto y una lista fija. Por defecto este tipo de combo se dimensiona para que nada de la lista se despliegue. Se puede recorrer la lista utilizando las fechas de dirección del teclado.
		DropDownList	Este estilo permite solo la selección de elementos de la lista desplegable. La parte del cuadro de texto no está disponible.
Sorted	Combobox y Listbox	Indica si la lista de elementos que contiene se encuentra ordenada alfabéticamente.	
Ítems	Combobox y Listbox	Devuelve o establece los elementos contenidos en una lista.	
Backcolor	Todos los controles mencionados	Devuelve o establece el color de fondo de todos los objetos. Esta propiedad se comporta de igual manera que en los otros controles ya vistos. Ej: textbox, label, etc.	
Forecolor	Idem anterior	Devuelve o establece el color de las letras de todos los objetos. Esta propiedad se comporta de igual manera que en los otros controles ya vistos. Ej: textbox, label, etc.	

En esta tabla hemos presentado las propiedades más importantes que ustedes debe conocer acerca de estos nuevos controles. A continuación, nos iremos explayando con algunos controles, señalando propiedades y métodos no descriptos en la tabla anterior.

Métodos comunes de los objetos de Lista

Tanto el cuadro combinado, como el cuadro de lista, presentan los siguientes métodos que permiten realizar las operaciones básicas con estos controles. Veamos cuáles son:

Add

Agrega elementos a la lista indicada. Recuerden que este procedimiento también se puede llevar a cabo en tiempo de diseño, utilizando la propiedad Items de la ventana de propiedades.

Para añadir elementos en tiempo de ejecución se utiliza la siguiente sintaxis:

NombeLista.Items.Add(Elemento)

Ej: lstClientes.Items.Add(“Ana López”) ‘ Incorpora el elemento Ana López a la lista de clientes

RemoveAt

Quita elementos de una lista. En la sintaxis que aparece a continuación ustedes verán que para eliminar un elemento de una lista, usted necesita indicar la posición del ítem a borrar.

NombreLista.Lista.RemoveAt (index)

Presten atención al siguiente ejemplo. Siguiendo con el ejemplo de la lista de clientes. Pensemos que la misma se compone de la siguiente manera:

lstClientes	**Posición**
Ana López	0
Juan Pérez	1
Sandra Torres	2
Carolina Ferreira	3

La siguiente lista se compone de 4 elementos, aunque debe tener en cuenta que el índice del primer elemento es el cero.

Entonces, si pretendemos borrar el cliente Juan Pérez, la línea de código sería la siguiente:

LstClientes.Items.RemoveAt(1)

Remove

Otro método para eliminar elementos de una lista es el método Remove, quien necesita como parámetro, el elemento a borrar, no su posición dentro de la lista como lo explicado en el método anterior.

NombreLista.Lista.Remove (Elemento)

Siguiendo con el ejemplo anterior para borrar al cliente Juan Pérez la sintaxis seria la que sigue…

LstClientes.Items.Remove ("Juan Pérez")

Clear

Elimina todos los elementos de la lista. La sintaxis es la siguiente:

Nombrelista.Clear 'este método no necesita argumento porque limpia la lista completamente

Ej: para vaciar la lista de clientes:

LstClientes.Clear

PROPIEDADES COMUNES DE LISTA

Las siguientes propiedades son accesibles desde el código, no se encuentran disponibles desde la ventana de propiedades.

SelectedIndex

Retorna el valor numérico de la selección actual. El primer ítem de la lista esta indexado con el número (0), y si no se encuentra ningún ítem seleccionado, el valor de retorno es (-1).

Ej: si deseáramos eliminar el cliente que se encuentra seleccionado, la línea de código sería la siguiente:

lstClientes.Items.RemoveAt(lstSocios.SelectedIndex)

lstClientes	Posición
Ana López	0
Juan Pérez	1
Sandra Torres	2
Carolina Ferreira	3

La lista lstClientes, con el ítem seleccionado, para borrarlo.

Usamos la propiedad SelectedIndex, pero como vimos desde el comienzo del seminario una propiedad debe ir junto al objeto que le da origen. Por eso escribimos lstClientes.SelectedIndex.

lstClientes	Posición
Ana López	0
Sandra Torres	2
Carolina Ferreira	3

La lista lstClientes después de haber borrado el elemento indicado.

Count

La propiedad Count presenta el número de elementos de la lista.

NombreLista.Items.Count

Ej: si quisiéramos saber la cantidad de elementos de la lista de clientes, y mostrarla en un cuadro de mensaje, el código sería el siguiente:

MessageBox.Show (”La lista posee “ & stclientes.Items.Count & “elementos”)

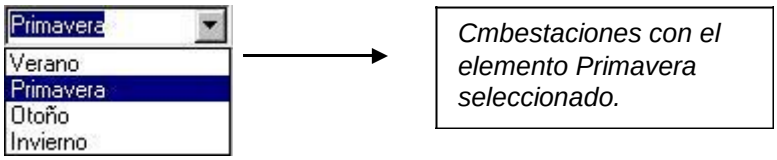
Text (Sólo se aplica al cuadro combinado)

Normalmente, la forma más sencilla de obtener el valor del elemento seleccionado actualmente es mediante la propiedad **Text**. Esta propiedad se corresponde con el contenido de la parte de cuadro de texto del control en tiempo de ejecución. Puede ser un elemento de la lista o una cadena de texto escrita por el usuario en el cuadro de texto.

La propiedad **Text** contiene el elemento seleccionado actualmente en el cuadro combinado.

Por Ej: si deseo mostrar en una etiqueta el elemento seleccionado de un combo determinado, las líneas de código serían las siguientes:

Lblestacion=cmbestaciones.Text



En este caso, la etiqueta lblestacion, mostrará la leyenda Primavera.

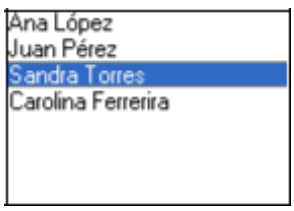
SelectedItem

Esta propiedad devuelve el elemento de la lista seleccionado actualmente.

NombreLista.SelectedItem

Esta propiedad muestra lo mismo que la propiedad Text, pero es aplicable tanto en el objeto ComboBox como en un ListBox.

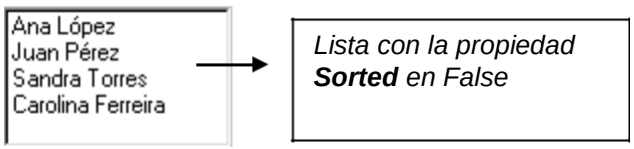
Para mostrar el elemento seleccionado de la lista lstClientes…

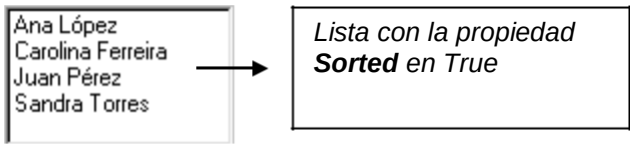


MessageBox.Show (lstClientes.SelectedItem)

Sorted

Para hacer que una lista aparezca ordenado en forma alfanumérica, se coloca la opción *True* en la propiedad Sorted. Para mostrar la lista en el orden en que fueron agregados, se establece dicha propiedad en *False*.





Agregar controles

Además de los controles que aparecen en el cuadro de herramientas, podemos añadir nuevos controles desde el comando **Elegir Elemento…** del menú contextual de la barra.

Agregaremos tres controles se pueden utilizar para manejar archivos (FileListBox, DirListBox, DriveListBox).

DRIVELISTBOX

Permite al usuario seleccionar una unidad válida de disco en tiempo de ejecución. Este despliega una lista con todas las unidades disponibles del sistema y reacciona a los clics del mouse para permitirle moverse entre ellos.

Utilizaremos este control para crear cuadros de diálogo que permitan al usuario abrir un archivo de una lista de un disco en cualquier unidad disponible.

La propiedad más importante de este control es la propiedad **Drive**. Si tienen en su formulario un control DriveListBox llamado Drive1, pueden usarlo para determinar la unidad actual en tiempo de ejecución, por ejemplo almacenando la misma en una variable:

```
Unidadactual=drive1.Drive
```

O bien, pueden hacer que un cuadro de lista de directorios(DirListBox) despliegue las carpe-tas de la unidad actualmente seleccionada, agregando la siguiente línea de código al evento Change del cuadro de lista de unidades:

```
Dir1.Path=Drive1.Drive
```

DIRLISTBOX

Muestra una lista jerárquicamente ordenada de directorios y subdirectorios (término que en Windows conocemos como carpetas y subcarpetas) y permite que el usuario con el clic del mouse pueda navegar entre ellos.

Ustedes pueden pasar la propiedad **Path** mientras el programa se está ejecutando para identificar la ruta actual, o también sincronizar un DirListbox con un FileListBox, para que este último despliegue los archivos de la carpeta actual, de la siguiente manera:

File1.Path=Dir1.Path

FILELISTBOX

Se utiliza este control para listar archivos en el directorio especificado por su propiedad PATH en tiempo de ejecución. Pueden usar este control para mostrar una lista de archivos en el directorio actual, o pueden establecer la propiedad PATTERN para mostrar solamente cierto tipo de archivos (por ejemplo: imágenes de mapa de bits) seleccionar un archivo o un grupo de archivos.

El archivo que se encuentra seleccionado es referenciado por la propiedad **FileName**, la cual se encuentra disponible únicamente en tiempo de ejecución. Al concatenar la propiedad Path del cuadro de directorio junto a la propiedad FileName del cuadro de lista de archivos sincronizados, usted tiene identificado el archivo, el cual puede almacenarse en una variable de tipo String, tal como sigue.

Nombre_archivo=dir1.Path & "\" & file1.FileName

Ej: si estoy haciendo referencia a la imagen agua.bmp de la carpeta Windows, obtendríamos los siguientes datos:

Dir1.Path= C:\Windows
File1.FileName= agua.bmp

Si tratamos de unir esto, quedaría así: C:\Windowsagua.bmp

Sin duda, nos está faltando la barra invertida (\) que separa la carpeta del archivo, por esto tuvimos que añadirlo en nuestra línea de programación con ayuda del &.

Por lo tanto, ahora quedará así: C:\Windows\agua.bmp

Timer (temporizador)

Puede ejecutar código en tiempos regulares por eventos de tiempo. Por ejemplo si desean crear una pantalla de presentación de un programa, la cual solo debe perma-necer abierta por unos cuantos segundos, ya que transcurridos estos se debe ocultar automáticamente, sin necesidad que el usuario provoque ningún evento.

Las propiedades más importantes de este control, se presentan en la siguiente tabla:

Tabla II Propiedades del control Timer

Propiedad	**Descripción**
Interval	Esta propiedad se mide en milisegundos representado por un valor en-

	tre 1 a 65.535. Ej. 10.000 son 10 seg. Y el máximo es 1'
Enabled	Esta propiedad determina si el control de tiempo será invocado por un evento de tiempo.

El ejemplo del reloj digital presentado en un objeto Label se crea de la siguiente forma:

Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick

Label1.Text = **TimeOfDay**

End Sub

Luego, debe establecer la propiedad ENABLED como TRUE e INTERVAL como 1000 (1 seg).

Eventos del Teclado

Los eventos del teclado, junto con los eventos del *mouse*, son los principales elementos de interacción del usuario con el programa. Presionar una tecla o hacer clic desencadena eventos que proporcionan el medio de introducción de datos y la manera básica de desplazarse entre menús y ventanas.

Aunque Windows como sistema operativo ya proporciona un medio para realizar todas esas acciones, algunas veces es útil o necesario modificarlas o mejorarlas. Los eventos KeyPress, KeyUp y KeyDown le permiten efectuar esas modificaciones o mejoras

Programar su aplicación para que responda a los eventos del teclado se conoce como escribir un *controlador de teclado*. Un controlador de teclado puede funcionar en dos niveles: al nivel de control y al nivel de formulario. El controlador de *nivel de control* le permite programar un control específico. Por ejemplo, si desea que al pulsar la tecla ENTER sobre un cuadro de texto, éste se desplace al siguiente. En un controlador de *nivel de formulario*, el formulario reacciona primero a los eventos del teclado. Por ejemplo programar la acción de una tecla de Función.

Con estos eventos del teclado puede escribir código para tratar la mayoría de las teclas de un teclado estándar.

KeyDown-KeyUp

Estos dos eventos ocurren cuando el usuario presiona (KeyDown) o suelta (KeyUp) una tecla mientas un objeto tiene el enfoque.

Estos procedimientos de evento nos brindan mayor información que el evento KeyPress, por eso trabajaremos con ellos.

Sintaxis

Private Sub *objeto*_**KeyDown(**ByVal sender As Object, ByVal e As System.Windows.Forms.KeyEventArgs**)**
Private Sub *objeto*_**KeyUp(***códigoTecla* **As Integer**, *mayús* **As Integer)**
La sintaxis de los eventos KeyDown y KeyUp consta de las siguientes partes:

Parte	Descripción
sender	Identifica el objeto que recibe la pulsación de teclas
KeyEventArgs	*keyEventArgs*.Key: un valor entero que representa system-specific. *keyEventArgs*.Shift: un valor lógico que representa si la tecla Shift fue pulsada. *keyEventArgs*.Ctrl: un valor lógico que representa si la tecla Ctrl fue pulsada. *keyEventArgs*.Alt: un valor lógico que representa si la tecla Alt fue pulsada.

Para ambos eventos, el objeto que tiene el enfoque recibe todas las pulsaciones de tecla.
Los eventos KeyDown y KeyUp pueden aplicarse a:

- Teclas de función.
- Teclas de desplazamiento.
- Combinaciones de teclas.

Utilicen los procedimientos de evento KeyDown y KeyUp si necesita responder a presionar y soltar una tecla.

Propiedad KeyPreview

Devuelve o establece un valor que determina si los eventos de teclado (KeyDown, KeyUp y KeyPress) de los formularios se invocan antes que los eventos de teclado de los controles.

Sintaxis

objeto.**KeyPreview** [= *booleano*]

La sintaxis de la propiedad **KeyPreview** consta de las siguientes partes:

Parte	Descripción
Objeto	El nombre del formulario al cual se quiere activar esta propiedad.
booleano	Una expresión booleana que especifica cómo se reciben los eventos, como se describe en Valores.

Valores

Los valores admitidos para *booleano* son:

Valor	Descripción
True	El formulario recibe los eventos de teclado primero y después el control activo.
False	(Predeterminado) El control activo recibe los eventos de teclado y el formulario no.

TAREA

Desarrolle las siguientes actividades, la primera es un ejercicio complementario con los temas de la clase anterior, y la segunda un ejercicio donde trabajará con un control nuevo el *ComboBox* que encontrarán explicado en el guion de esta semana.

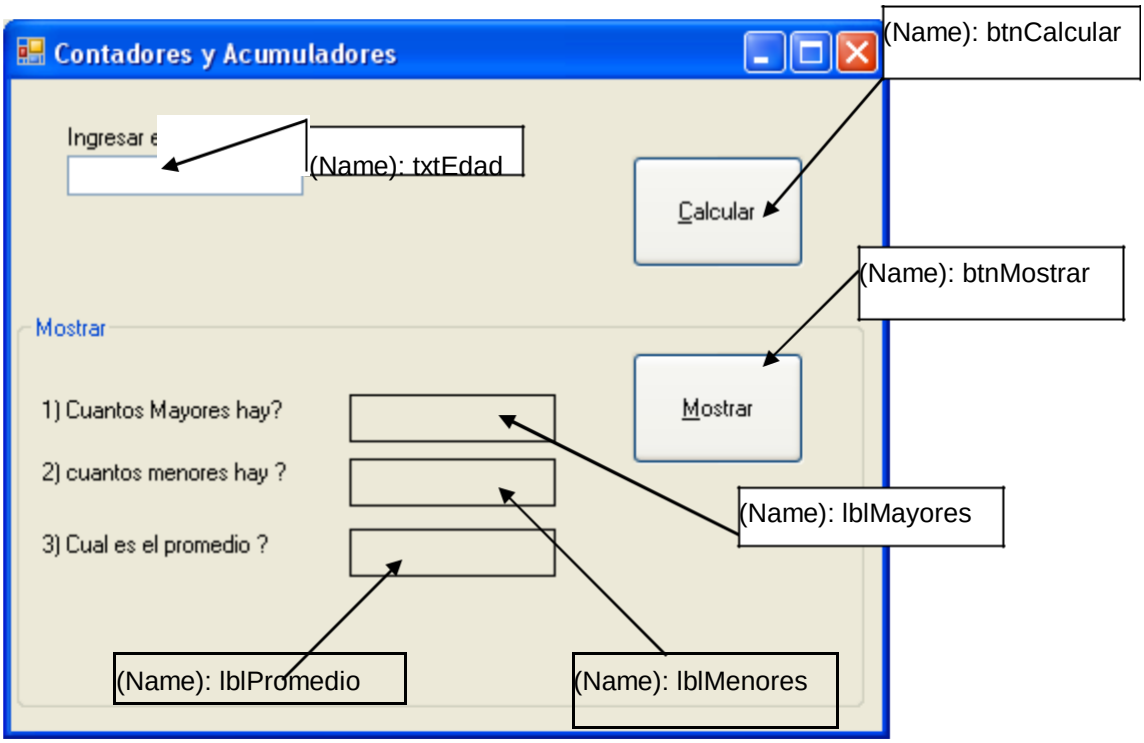
La entrega no es obligatoria, pero quienes quieran, pueden subir su resolución al *Foro de consultas de la Unidad 2*.

Actividad Nº 8

Consigna: se ingresan las edades de varias personas, para luego realizar los siguientes cálculos.

Se pide:

- Mostrar cuantos son mayores de edad
- Mostrar cuantos son menores
- Mostrar cual es el promedio de edades.

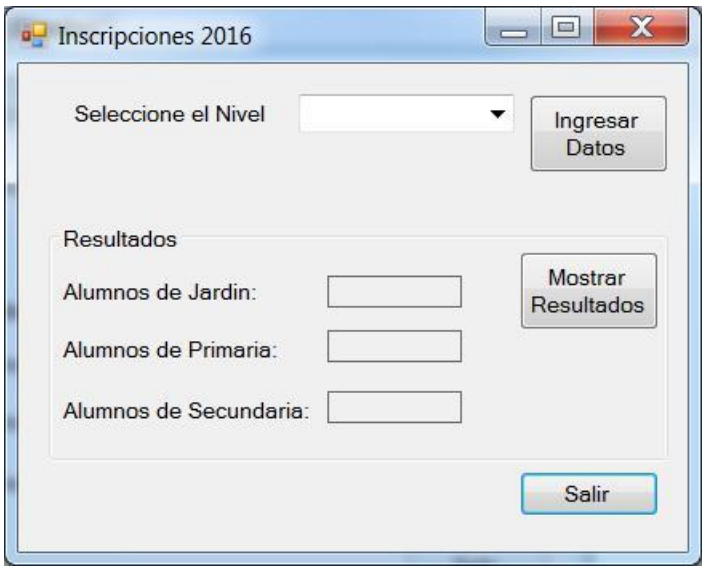


Actividad Nº 9

Consigna: en la administración de un colegio se van registrando los alumnos para los tres niveles escolares: Jardín, Primaria y Secundaria.

Se pide:

- Calcular y Mostrar cuantos alumnos ingresaron por Nivel.
- Programar el botón para Salir del proyecto.



Cierre de la clase

A lo largo de esta clase nos familiarizamos con el resto de los controles de la barra de herramientas. Examinamos de todos ellos: sus propiedades y métodos más importantes.

Conociendo las características de cada uno, tendrán una visión más objetiva a la hora de decidir con qué controles armarán la interfaz de sus programas.

Lo esencial es que comprendan que todos estos controles, ya sean las casillas de verificación, los radio botones, los marcos, listas desplegables, lista de unidades, directorios, etc. son objetos visualmente atractivos, pero también eficientes a la hora de ser utilizados por el usuario de su programa. Puesto que un persona que maneja Windows, no tendrá inconveniente alguno de utilizarlos eficientemente, sin necesidad de haber adquirido ningún conocimiento previo, deslindando de este modo, posibles errores operativos.

¡¡¡Les deseo a todos, que tengan una excelente semana!!

Martin Murdaca