Capítulo 5
Los objetos y colecciones en VBA

1. Noción de objeto

VBA es un lenguaje que permite hacer programación orientada a objetos (POO): un objeto representa una idea, concepto o entidad del mundo real, como un avión, un individuo e incluso una película. Tiene una estructura interna y un comportamiento, y se puede comunicar con sus homólogos. Los elementos que permiten describir un objeto forman lo que se llama una clase. Cada objeto que proviene de una clase es una instancia de clase. Las clases tienen propiedades, métodos y eventos.

1.1 Propiedades

El objeto es una entidad que podemos distinguir gracias a sus propiedades (su color o dimensiones, por ejemplo). Si tomamos como ejemplo un libro, este se caracteriza por sus propiedades: número de páginas, título, número de capítulos, editor, contenido, etc. Cada una de sus propiedades se puede especificar en cada libro, pero todos los libros tienen fundamentalmente las mismas propiedades. Algunas propiedades de los objetos se pueden modificar (velocidad de un coche, por ejemplo) y otras no (una marca de coche).

En programación con VBA, la sintaxis general de acceso a las propiedades de un objeto es la siguiente:

```
UnObjeto.SuPropiedad
```

Aquí se accede a la propiedad `SuPropiedad` del objeto `UnObjeto`. La combinación `UnObjeto.SuPropiedad` tendrá el mismo comportamiento que una variable clásica, que puede tomar un valor o devolver un valor con el uso del operador =.

Por ejemplo, es posible leer el contenido SQL de una consulta de Access y visualizarla con el siguiente código:

```
Sub LeerSQLConsulta()
    Dim UnaConsulta As QueryDef
    UnaConsulta.SQL = "SELECT * FROM MiTabla"
    ...
    MsgBox UnaConsulta.SQL
End Sub
```

En el capítulo Los objetos de acceso a los datos DAO y ADO, volveremos al objeto `QueryDef` con más detalle.

Una propiedad de un objeto puede ser un objeto ella misma, por lo que se podría utilizar más adelante en el código.

1.2 Métodos

Además de los elementos que caracterizan un objeto, también es posible realizar acciones con estos objetos, como por ejemplo "sentarse", "acostarse", "arreglarse", etc. Estas acciones se llaman métodos. Estas acciones se representan en forma de procedimientos y funciones en VBA, según puedan o no devolver valores. Algunas acciones pueden necesitar argumentos para funcionar (número de caracteres de un libro, con o sin espacios).

En programación con VBA, la sintaxis general de acceso a los métodos de un objeto es la siguiente:

```
UnObjeto.SuMetodo [MiArgumento]
```

En el siguiente ejemplo, se trata simplemente de refrescar el vínculo entre una tabla relacionada y la aplicación Access:

```
Sub RefrescarRelacionTabla()
    Dim UnaVariableTemporal As TableDef
    ...
    UnaVariableTemporal.RefreshLink
End Sub
```

Y, en caso de que el método sea una función, podemos almacenar el resultado en una variable, como en el siguiente ejemplo:

```
Sub Número_Campos()
    Dim Nm_Campos As Integer
    Dim MiTabla As TableDef
    ...
Nm = MiTabla.Fields.Count
End Sub
```

De la misma manera que en las llamadas a funciones o procedimientos, los argumentos se separan por comas en las llamadas a métodos del objeto.

1.3 Eventos

Para cada objeto, es posible detectar y tener en cuenta el resultado de acciones particulares externas, que llamamos eventos. Todos los eventos que tienen lugar durante la ejecución de un programa se detectan y gestionan por la máquina, pero, según la opción del desarrollador, solo algunas se pueden tratar de manera particular, como por ejemplo un clic en un botón, una casilla de selección que se marca, la apertura o cierre de una ventana, etc. Estos eventos se gestionan a través de procedimientos que algunas veces tienen argumentos y otras no.

En VBA, la sintaxis más frecuente para los eventos es la siguiente:

```
Sub MiObjeto_MiEventoDestacado()
    'El código que se ejecutará cuando se detecte el evento
End Sub
```

Por ejemplo, cuando se trata de un clic en un botón llamado `MiBoton`, tendrá el código siguiente:

```
Sub MiBoton_Click()
    'código que se ejecutará
End Sub
```

Los eventos en Access se tratarán más adelante en detalle, en el capítulo Los eventos de Access.

1.4 Las colecciones

Cuando varios objetos de una misma clase se agrupan, pueden pertenecer a una misma colección de objetos. Para referirse a un elemento en particular de una colección de objetos, hay varias sintaxis, entre las siguientes:

```
Nombre_Colección!Nombre_Objeto
Nombre_Colección![nombreObjeto]
Nombre_Colección("NombreObjeto")
Nombre_Colección(variable_Nombre) 'variable_Nombre es una cadena de
caracteres que contiene el nombre del objeto
Nombre_Colección(variable_Numero) 'variable_Numero es un valor
digital que contiene el número del objeto dentro de la colección.
```

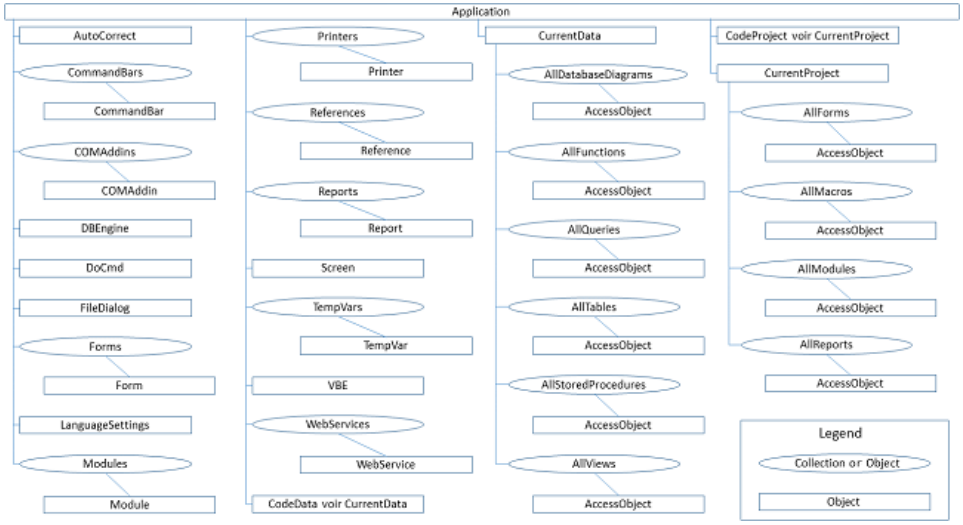
Las dos sintaxis que se utilizan más habitualmente son la tercera y la quinta, ya que permiten el uso de IntelliSense (ver capítulo VBE y seguridad en Access 2019).

Observación

Es habitual en las colecciones comenzar a contar en 0 y no en 1. Además, el número de un objeto dentro de una colección dependerá de la presencia de otros elementos antes o después de él, lo que no garantiza localizar el objeto correcto por este medio.

2. Modelo de objeto de Access

El objetivo de algunas de las secciones siguientes es mostrar el modelo jerárquico utilizado dentro de la aplicación Access, en forma de colecciones de objetos, que se van a explicar a continuación.



3. Colecciones en Access

A continuación se muestran las principales colecciones que podemos manipular con VBA en Access.

Colección	Contiene una colección de	Descripción
`COMAddins`	`COMAddin`	Colección de los **complementos COM**
`CommandBars`	`CommandBar`	Colección de las **barras de comando**
`Forms`	`Form`	Colección de los **formularios abiertos**. Ver también `CurrentProject.AllForms`

Colección	Contiene una colección de	Descripción
`Modules`	`Module`	Colección de los **módulos**
`Printers`	`Printer`	Colección de las **impresoras disponibles**
`References`	`Reference`	Colección de las **referencias a librerías**. Ver Menú: **Herramientas\ Referencias**
`Reports`	`Report`	Colección de **estados**. Ver también `CurrentProject.AllReports`
`TempVars`	`TempVar`	Colección de las **variables temporales**
`WebServices`	`WebService`	Colección de las conexiones a **servicios web**

4. Objetos de Access

A continuación se muestran los principales objetos que es posible manipular en el modelo Access.

Objeto	Descripción
`Application`	Representa la aplicación Microsoft Access activa.
`AutoCorrect`	Representa las opciones de corrección automática de Access.
`DBEngine`	Representa el motor de base de datos Microsoft Jet. Este objeto permite controlar el resto de los objetos de acceso a los datos.
`DoCmd`	Permite convertir en VBA las acciones de Macros.
`FileDialog`	Permite acceder a las funcionalidades de los cuadros de diálogo (Abrir o Guardar, por ejemplo).

Capítulo 9
Controles

Duración: 1 hora 40 minutos

Palabras clave

control, botón de comando, zona de lista, lista modificable, zona de texto, título, etiqueta, casilla de selección, casilla de opción, control multipágina, imagen, método, propiedad, evento, control ActiveX

Objetivos

Utilizar los controles principales y familiarizarse con sus propiedades, métodos y eventos.

Conocimientos preliminares

Para comprobar si dispone de los conocimientos necesarios antes de iniciar esta práctica técnica, responda al siguiente cuestionario (algunas preguntas admiten varias respuestas):

1. Un control de formulario posee:
   - **a.** eventos.
   - **b.** propiedades.
   - **c.** métodos.
2. El valor del índice del primer control es:
   - **a.** `0`
   - **b.** `1`
3. El valor del índice indica el orden:
   - **a.** de arriba abajo.
   - **b.** de izquierda a derecha.
   - **c.** en el que los controles se han añadido a la colección.
4. La colección `Controls` de un formulario concreto permite:
   - **a.** enumerar los controles.
   - **b.** definir las propiedades de los controles.
   - **c.** modificar el formulario en sí mismo.

5. Los métodos comunes a los controles TextBox, ListBox, ComboBox son:
   - **a.** `Move`
   - **b.** `SetFocus`
   - **c.** `Clear`

6. Para cambiar el texto de un control Label, se debe utilizar la propiedad:
   - **a.** `Caption`
   - **b.** `Value`
   - **c.** `Name`

7. En un ListBox, la propiedad `ListCount` permite precisar:
   - **a.** el máximo de filas que se muestran a la vez.
   - **b.** el número de registros en la lista.
   - **c.** el número de filas en las que se puede implementar un bucle de recorrido.

8. En un ListBox o ComboBox, la propiedad `ListIndex`:
   - **a.** tiene por valor `0` si no se ha seleccionado ningún elemento.
   - **b.** tiene por valor `-1` si no se ha seleccionado ningún elemento.
   - **c.** contiene un índice del elemento seleccionado en la lista.

9. La propiedad `Text` permite modificar:
   - **a.** el valor de un control ComboBox o ListBox.
   - **b.** el texto de un control TextBox.
   - **c.** el valor del elemento seleccionado en un control ComboBox o ListBox.

10. Para insertar una imagen, debe utilizarse el control:
   - **a.** Image
   - **b.** Picture
   - **c.** Shape

Solución pág. 227

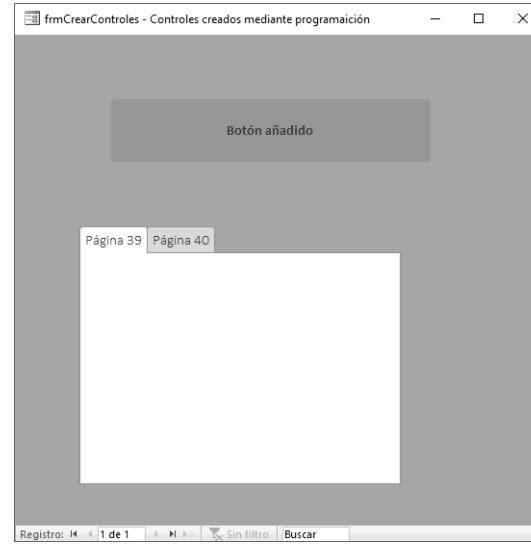
Enunciado 9.1 Añadir controles mediante programación

Duración estimada: 15 minutos


formulario **frmControles**.

Dispone del formulario **frmControles**. Este formulario contiene un botón de comando llamado **cmdAñadirControles**. Programe el evento "al hacer clic" de este botón que efectúe las operaciones que se enumeran a continuación:

1. Abrir el formulario existente **frmCrearControles**.
2. Suprimir todos los controles eventuales que pueda haber en ese formulario.
3. Añadir un botón de comando y configuración de las distintas propiedades de formato de ese botón (Top, Left, Width, Height, BackColor, etc.)
4. Añadir un control Pestaña y configurar las distintas propiedades de formato.



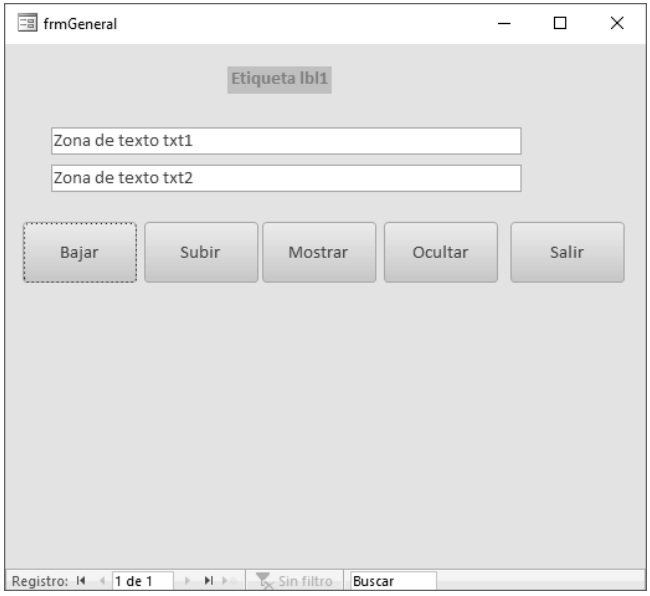
Solución pág. 227

Enunciado 9.2 Ocultar y volver a mostrar los controles

Duración estimada: 15 minutos

Formulario **frmGeneral**

Dispone del formulario **frmGeneral**. Cree tres procedimientos de evento que permitan, respectivamente: ocultar todos los controles excepto el control «Mostrar» (**cmdOcultar**), mostrar todos los controles ocultos (**cmdMostrar**) y cerrar el formulario (**cmdSalir**).



Pista

Nombre de los controles empleados:

- zona de título: ***lbl1*** *(Label)*

- zonas de texto: ***txt1***, ***txt2*** *(TextBox)*

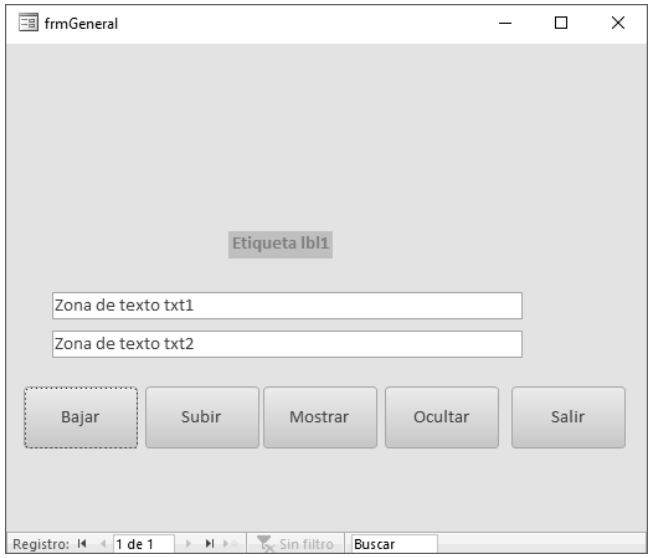
- botones de comando: ***cmdBajar***, ***cmdSubir***, ***cmdMostrar***, ***cmdOcultar***, ***cmdSalir*** *(CommandButton)*

Solución pág. 228

Enunciado 9.3 Desplazar los controles

Duración estimada: 10 minutos

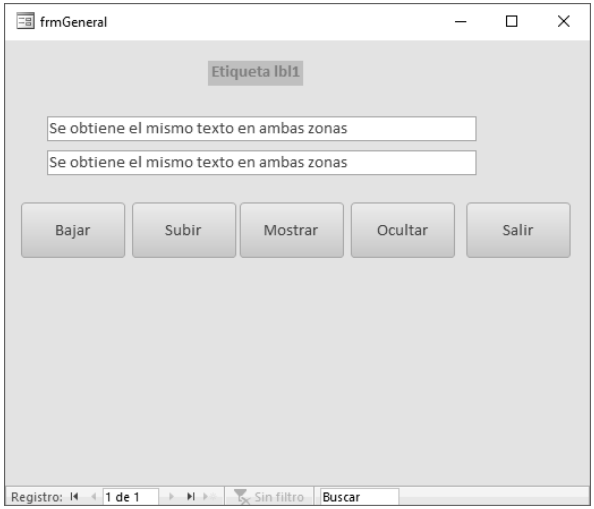
A partir del formulario anterior y de los controles **Subir** y **Bajar**, cree los procedimientos de evento que permitan subir o bajar todos los controles (**cmdBajar** y **cmdSubir**, respectivamente). Ejemplo:



Solución pág. 229

Enunciado 9.4 Hacer que un control dependa de otro

Duración estimada: 5 minutos
Cree el procedimiento de evento de **txt1** que transcriba en la segunda zona de texto todos los caracteres que se escriben en la primera zona de texto. Ejemplo:



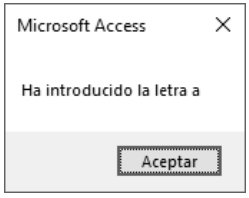
Pista

Utilice el evento "al soltar la tecla" (KeyUp) preferentemente en lugar de "al cambiar" (Change).

Solución pág. 230

Enunciado 9.5 Hacer que los controles reaccionen a una entrada de datos por teclado

Duración estimada: 10 minutos
Cree el procedimiento de evento que muestre un mensaje cuando la letra «a» se introduzca en la zona de texto **txt2**.



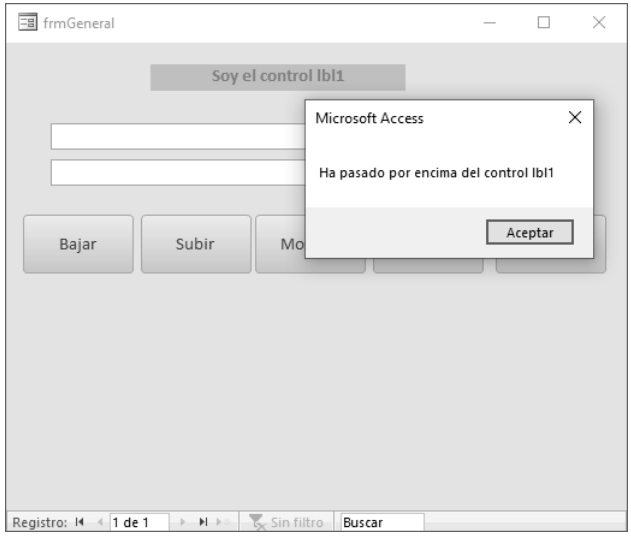
Pista

Utilice el evento "al pulsar tecla" (`KeyPress`) .

Solución pág. 230

Enunciado 9.6 Hacer que los controles reaccionen al ratón

Duración estimada: 10 minutos
Cree el procedimiento de evento que muestre un mensaje cuando el puntero del ratón pase por encima del control **Etiqueta lbl1** del formulario y que cambie el título de esa etiqueta.



Solución pág. 231