**1.1 ¿Qué es la programación orientada a objetos?**

En **Visual Basic 6.0** y en otros programas, se le llama *objeto* a todo lo que se ve en una ventana típica de Windows; los objetos son por ejemplo un botón de comando, una caja de texto, una imagen, en general todo objeto visible que puedas ver en la pantalla.

Se les llaman *objetos* porque cada uno de ellos poseen *propiedades*, *eventos* y *métodos*. Un *botón de comando* tiene *propiedades* tales como: **Caption** (Titulo) que indica el texto que tiene el botón, también tiene las *propiedades***Width** (Anchura) y **Height** (Altura) que establecen la anchura y altura del botón.

Todos los objetos con que trabajemos en **Visual Basic 6.0**  poseen *propiedades*, *métodos* y *eventos*, aunque algunos objetos pueden tener *propiedades*, *métodos* y *eventos* que otros objetos no pueden tener, por ejemplo, un objeto **TextBox** (caja de texto) tiene la propiedad **Text** (texto) que indica el texto que contiene la caja, de tal manera, es obvio que un botón de comando no pueda tener esta propiedad, así como el evento **Change** (Cambio) que poseen las cajas de textos que indican cuando el usuario esta cambiando el contenido de la caja, también esta claro que un botón de comando **(CommandButton)** no pueda tener este evento.

Resumiendo todo lo dicho anteriormente, se puede definir la **Programación Orientada a Objetos** **(POO)** como aquella en la que trabajamos con objetos visibles cada uno de los cuales poseen sus propios *eventos*, *métodos* y *propiedades*.

**1.2 Programas orientados a eventos**

Es lógico que para que un programa se pueda llamar *orientado a eventos* debe haber sido creado en un lenguaje de programación *orientado a objetos*, ya que cada *objeto* espera a algún evento que realice el usuario sobre él.

Los *programas orientados a eventos* son los programas típicos de Windows, Linux, Beos, que esperan a que el usuario realice alguna acción, ya sea con el *mouse* o con el *teclado* para realizar alguna función, por ejemplo, la calculadora de **Windows** espera a que el usuario haga clic (*evento clic*) con el mouse sobre uno de los botones que contienen los números para ponerlo en la caja de texto, o bien, espera a que el usuario pulse un número desde el teclado para ponerlo en la caja de texto.

**1.3 Programas para el entorno de Windows**

**Visual Basic 6.0** está orientado a la realización de programas para **Windows**, pudiendo incorporar todos los elementos de este entrono informático: ventanas, botones, cajas de diálogo y de texto, botones de opción y de selección, barras de desplazamiento, gráficos, menús, etc.

Prácticamente todos los elementos de interacción con el usuario de los que dispone **Windows** **95/98/XP/2000/NT** pueden ser programados en **Visual Basic 6.0** de un modo muy sencillo. En ocasiones bastan unas pocas operaciones con el ratón y la introducción a través del teclado de algunas sentencias para disponer de aplicaciones con todas las características de **Windows** **95/98/XP/2000/NT**.

**- 1.3.1 Modo de diseño y modo de ejecución**

La aplicación **Visual Basic** de Microsoft puede trabajar de dos modos distintos: En *modo diseño* y en *modo de ejecución*. En *modo diseño* el usuario construye interactivamente la aplicación, colocando *controles* en el formulario, definiendo sus *propiedades*, y desarrollando funciones para gestionar los *eventos*.

La aplicación se prueba en *modo de ejecución*. En este caso el usuario actúa sobre el programa (introduce eventos) y prueba cómo responde el programa. Hay algunas *propiedades* de los *controles* que deben establecerse en modo de diseño, pero muchas otras pueden cambiarse en tiempo de ejecución desde el programa escrito en **Visual Basic 6.0**.

- **1.3.2 Formularios y Controles**

Cada uno de los elementos gráficos que pueden formar parte de una aplicación típica de **Windows** es un tipo de *control*: botones, cajas de dialogo y de texto, cajas de selección desplegables, los botones de selección y de opción, las barras de desplazamiento horizontales y verticales, los gráficos, los menús, y muchos otros elementos son *controles* para **Visual Basic 6.0**. Cada *control* debe tener un **nombre** a través del cual se puede hacer referencia a el en el programa. **Visual Basic 6.0** asigna **nombres** por defecto a los *controles*. **Visual Basic** permite al usuario cambiar los *nombres por defecto*.

En **Visual Basic** 6.0 un *formulario* es una ventana. Un *formulario* puede ser considerado como una especie de contenedor para los *controles*. Una aplicación puede tener uno o varios *formularios* (ventanas), pero un único *formulario* puede ser suficiente para la creación de una aplicación sencilla. Los *formularios* deben también tener un *nombre* que permita hacerse referencia a él ó del él.

**- 1.3.3 Objetos, Propiedades y Clase**

A los *controles* que colocamos en un *formulario* que poseen *propiedades*, *métodos* y *eventos* se les llaman *objetos* y a las características propias de esos *objetos* se les llaman *propiedades*.

La **clase** es la entidad genérica a la que pertenece un *control*, por ejemplo, en un programa puede haber varios botones, cada uno de los cuales es un *objeto* que pertenece a una *clase* de los controles **(CommandButton)**. Cada *formulario* y cada tipo de *control* tienen un conjunto de *propiedades* que definen su aspecto gráfico (tamaño, color, posición en la ventana, tipo y tamaño de letra, etc.) y su forma de responder a las acciones (*eventos*) del usuario. Cada *propiedad* tiene un nombre que viene ya definido por el lenguaje.

Por lo general, las *propiedades* de un *objeto* son datos que tienen valores lógicos **(true, false)** o numéricos concretos, propios de ese *objeto* y distintos de las de otros *objetos* de su *clase*. Así pues, cada *clase*, tipo de *objeto* o *control* tienen su conjunto de propiedades, y cada *objeto* o *control* tienen valores determinados para las *propiedades* de su *clase*.

Casi todas las *propiedades* de los *objetos* pueden establecerse en tiempo de *diseño* y también casi siempre en tiempo de *ejecución*. En este segundo caso se accede a sus valores por medio de las sentencias del programa en forma análoga a como se accede a cualquier variable en un lenguaje de programación. Para ciertas *propiedades* ésta es la única forma de acceder a ellos. Por supuesto **Visual Basic 6.0** permite crear distintos tipos de variables, como verá más adelante.

Para acceder a una *propiedad* de un *objeto* se hace por medio del *nombre del objeto* **(name),** seguido de un *punto* (**.**) y el *nombre de la propiedad.* Por ejemplo, para cambiar el color de fondo de una caja de texto (TextBox) llamada Text1 se haría de la siguiente manera:

Ejemplo:

Text1**.BackColor** = vbRed

Donde **Text1** es el nombre del *control*, **BackColor**es el nombre de la propiedad que permite cambiar el color del fondo de la caja de texto y **vbRed** es el valor que se le ha asignado a la *propiedad* **BackColor**,que en este caso es el color rojo de **Visual Basic 6.0**.

**- 1.3.4 Nombre de objetos**

En principio cada *objeto* de **Visual Basic 6.0** debe tener un nombre, por medio del cual se hace referencia a dicho *objeto*. El nombre **(name)** puede ser el que el usuario desee, e incluso **Visual Basic 6.0** proporciona *nombres por defecto* para los diversos *controles*. Estos *nombres por defecto* hacen referencia al tipo de *control* y van seguido de un número que se incrementa a medida que se van introduciendo más *controles* de ese mismo tipo en el formulario, por ejemplo, **Text1** para una caja de texto, **Text2** para otra caja de texto, **Command1** para un botón de comando, **Command2** para otro botón de comando, etc.

Los *nombres por defectos* no son adecuados porque sólo hacen referencia al tipo de *control*, pero no al uso que de dicho control está haciendo el programador. Por ejemplo, si se agregan dos botones a una aplicación **(Command1, Command2)** uno para salir de la aplicación y otro para guardar los cambios hechos en la aplicación, sería recomendable que el botón de **cerrar** lleve por nombre “cmdCerrar” y el botón de **guardar** “cmdGuardar” y no los *nombres por defecto* **Command1** y **Command2**, ya que en ocasiones no sabremos para que utilizamos uno y para que utilizamos el otro.

Para asignar los nombres a los *controles* existe una convención ampliamente aceptada que es la siguiente: se utilizan siempre tres letras en minúscula que indican el tipo de *control,* seguido de otras letras (la primera en mayúscula) libremente escogidas por el usuario, que tienen que hacer referencia al uso que se va a dar a ese *control.*

La **tabla 1.1** muestra las abreviaturas de los *controles* más usuales, junto con la nomenclatura inglesa de la que se derivan.

|  |  |  |  |
| --- | --- | --- | --- |
| **Abreviatura** | **Control** | **Abreviatura** | **Control** |
| chk | CheckBox | cbo | Combo y Drop-ListBox |
| cmd | CommandButton | dir | DirListBox |
| drv | DriveListBox | fil | FileListBox |
| frm | Form | fra | Frame |
| hsb | HorizontalScrollBar | img | Image |
| lbl | Label | lin | Line |
| lst | List | mnu | Menu |
| opt | OptionButton | pct | PictureBox |
| shp | Shape | txt | TExtEditBox |
| tmr | Timer | vsb | VerticalScrollBar |

Tabla 1.1. Abreviaturas para los controles más usuales.

**- 1.3.5 Eventos**

Ya se ha dicho que las acciones del usuario sobre un programa se llaman *eventos*. Son *eventos* típicos: hacer clic sobre un botón, el hacer doble clic sobre un fichero para abrirlo, el arrastrar un icono, el pulsar una tecla o combinación de teclas, el elegir una opción de un menú, el escribir en una caja de texto o simplemente mover el mouse.

Cada vez que se produce un *evento* sobre un determinado tipo de *control*, **Visual Basic 6.0** arranca una determinada *función* o *procedimiento* que realiza la acción programada por el usuario para ese *evento* concreto. Estos *procedimientos* se llaman con un nombre que se forma a partir del nombre del *objeto* y el nombre del *evento*, separados por el carácter (**\_**) underscore. Por ejemplo, el *evento* clic de un botón de comando:

**Private Sub** Command1\_Click ( )

**End Sub**

Donde **Private Sub**  indica la declaración del procedimiento, **Command1**especifica el nombre del *control*, el carácter (**\_**) underscore indica la separación entre el nombre del *control* y el nombre del *evento*, **Click** es el nombre del *evento* que especifica que la acción de ese procedimiento se ejecutará cuando el usuario haga clic sobre el botón, los paréntesis **( )** se utilizan para otras declaraciones que veremos más adelante y **End Sub** indica el fin del *procedimiento*.

**- 1.3.6 Métodos**

Los *métodos* son funciones que también son llamadas desde el programa, pero a diferencia de los *procedimientos* no son programadas por el usuario, sino que vienen ya pre-programadas con el lenguaje de programación. Los *métodos* realizan tareas típicas, previsibles y comunes para todas las aplicaciones. Cada tipo de *objeto* o de *control* tienen sus propios *métodos*. Por ejemplo, los formularios poseen un *método* llamado **Hide** que permite ocultar el formulario y otro *método* llamado **Show** que permite mostrarlo después de haber sido ocultado.

Para hacer referencia a un *método* basta con indicar el *nombre del objeto* **(name)** y el *nombre del método* separados por un *punto.*

Ejemplo: Para ocultar un formulario

**Form1.Hide**

Dónde **Form1**indica el nombre del *control* que en este caso es un formulario y **Hide**especifica el nombre del *procedimiento*.