Comencemos a programar con

VBA - Access

Entrega 08

Continuando con las Clases

Eduardo Olaz

Aclaraciones sobre el código del último capítulo.

Una clase no deja de ser un conjunto de Datos y Procedimientos dentro de un Módulo de Clase.

El nombre de la clase vendrá dado por el nombre con el que grabemos ese módulo.

Así en el ejemplo grabamos previamente el módulo de la clase **CPersona**, con ese nombre **Cpersona**.

Ya comenté que a los datos, accesibles de forma pública, se les llama **Propiedades** de la clase. A los procedimientos que son públicos se les llama **Métodos** de la clase.

Recordemos el código de la clase **Cpersona**

He modificado la línea : **If FechaNacimiento <> 0 Then**

```
Option Compare Database
Option Explicit

Public Nombre As String
Public Apellido1 As String
Public Apellido2 As String
Public FechaNacimiento As Date
Public Telefono As String

Public Function Edad() As Long
    ' Que conste que esta función no es exacta
    If FechaNacimiento <> 0 Then
        Edad = (Date - FechaNacimiento) / 365.2425
    End If
End Function

Public Function NombreCompleto() As String
    NombreCompleto = Nombre _
            & " " & Apellido1 _
            & " " & Apellido2
End Function
```

Las dos primeras líneas son similares a las que podemos encontrar en todos los módulos de Access. Os recuerdo que la primera es para indicar que los criterios de comparación entre cadenas serán los definidos por Access, y la segunda es para exigir la declaración Explícita de las variables.

En cuanto a las Propiedades tenemos 5,

Nombre Apellido1 Apellido2 FechaNacimiento Teléfono

Y tenemos 2 procedimientos públicos (**Métodos**):

Edad NombreCompleto

Una consideración:

Tanto **Edad** como **NombreCompleto** también se podrían considerar como propiedades de la clase **CPersona**, **de sólo lectura**.

No obstante existen otros métodos específicos para crear las propiedades.

Estos métodos son los procedimientos **Property**. Hablaremos de ellos en otra entrega próxima cuando estudiemos las clases con más de profundidad.

Realmente estas propiedades, y métodos no revisten ninguna especial complicación. **Edad** tras comprobar que el dato pasado sea diferente que cero realiza un simple cálculo matemático de resta y división, devolviendo el resultado. Si el parámetro fuera igual a cero, devolverá el **valor por defecto** de un **Long**, que también es **cero**.

En cuanto a **NombreCompleto**, realiza una concatenación de los contenidos de las tres primeras propiedades, separándolas con Espacios en Blanco (`" "`) y devuelve su resultado.

Ya tenemos definida la clase **CPersona**, pero y ahora ¿qué podemos hacer con ella?.

Ya os he comentado que una clase sirve para crear **Objetos**.

Y ¿en qué se diferencia una clase de un objeto?.

Aunque no sea una comparación exacta, la que puede haber entre los planos de ingeniería de un automóvil, y el automóvil construido.

O entre el código genético de un ser y el ser vivo real.

Si mediante el código de la clase **CPersona**, creamos el objeto **Pepe**, y el objeto **Juan**, (se les llamaría **Ejemplares de la Clase CPersona**), ambos tienen en común el código en el que se basan para existir, me refiero a los objetos informáticos, pero en concreto estos dos **Ejemplares** cambian en al menos, la propiedad **Nombre**.

Por ejemplo, yo tengo un hermano gemelo, que se llama **Enrique**.

Si utilizáramos mis datos y los de mi hermano para atribuir las propiedades al Objeto **Yo** y al Objeto **MiHermano**, de la case **CPersona**, diferirán en dos propiedades, el **Nombre** y el **Teléfono**, pero la clase en la que se basarán será la misma.

Es decir, el Objeto es la entidad creada, y la clase es el código en el que se basa ese objeto.

Para crear un objeto en VBA, se han de hacer dos cosas

1. Se declara la variable que va a manejar ese objeto, como del tipo de la clase del objeto. `Dim Yo As CPersona`

   Tras esta declaración se define que la variable **Yo** va a ser un objeto del tipo del generado por la clase **CPersona**.

   Pero en realidad todavía no es nada. Por ejemplo, si **CPersona** fueran los planos de un coche, tras esta declaración **Yo** sería algo así como la orden de fabricación de ese coche. Todavía sólo existe en un papel, aunque se sabe que se tiene intención de fabricarlo. ¿Y cómo se fabrica?

2. Utilizando la instrucción **New**. `Set Yo = New CPersona`

   Esta línea sería equivalente a decir **Haz** que **Yo** sea una **CPersona Nueva**. Tras esto **Yo** saldría nuevecito y reluciente del taller de montaje de los modelos **CPersona**.

   Pero **Yo** quién es. Es sólo un **Ejemplar** de la clase **CPersona**, También podemos decir que es una **Instancia** de la clase CPersona ó un **Objeto** de la clase. Lo

que hemos hecho, es mediante la instrucción **Set**, asignar una **Referencia de Objeto** a la Variable **Yo**.

¡Pobre **Yo**! Es todas estas cosas, en la jerga informática, pero en realidad ¿quién es?

¡Es sólo un **indocumentado**!

Y es un indocumentado porque no tiene ni nombre ni apellidos ni teléfono, y su fecha de nacimiento, así como su edad es Cero.

Vamos, que es un **Cero a la izquierda**.

Pero eso sí, es un flamante y nuevecito **objeto**.

Como he indicado la instrucción **Set** asigna una referencia de Objeto a la variable **Yo**. En realidad ¿qué hace?. Con **New CPersona** creamos una zona de memoria en la que ubicamos el objeto, y con **Set** conectamos la variable **Yo** a esa zona de memoria.

Si la declaración de la variable **Yo** la hubiéramos hecho de esta forma:

```
Dim Yo As New CPersona
```

Nos podremos ahorrar el paso de la asignación mediante **Set**.

Con `As New CPersona` se declara e instancia simultáneamente la variable **Yo**.

Pero no siempre es aconsejable declarar una variable objeto e instanciarla a la vez.

Por cierto, **Set** permite otras cosas, por ejemplo que haya **2 variables diferentes** apuntando al mismo objeto real.

Vamos a hacer un experimento:

Si no lo has hecho, créate un módulo de clase e introduce en él, el código de la clase **CPersona**.

Graba ese módulo de clase con el nombre **CPersona** y créate un módulo estándar.

En ese módulo estándar escribe los siguiente:

```
Option Compare Database
Option Explicit

Public Sub PruebaDeLaClaseCPersona()
    Dim Friki As CPersona
    Dim Alienigena1 As CPersona
    ' Hay un segundo alienígena que lo vamos _
      a declarar y crear simultáneamente
    Dim Alienigena2 As New CPersona
    ' Creamos una instancia del Friki
    Set Friki = New CPersona
    'Todavía no es nadie, vamos a identificarlo
    With Friki
        .Nombre = "Carlos Jesús"
        .Apellido1 = "el Curandero"
        .Apellido2 = "Friki"
        .FechaNacimiento = #1/24/1945#
```

```
    .Telefono = "696969696"
End With
' Ya no es un Sin Papeles
' está documentado y nos saluda.

Debug.Print " - - - - - - - - - - - - - - - - - - -"
Debug.Print "Hola terrícolas"
Debug.Print "Soy " & Friki.NombreCompleto
Debug.Print "Tengo " & Friki.Edad & " años"
Debug.Print "Y he venido a salvar al Mundo"

' Como todo el mundo sabe _
  los alienígenas son capaces _
  de apoderarse de las mentes de los Frikis
' Vamos a crear al Alienigena1 _
  no sólo con los datos del Friki _
  sino siendo el verdadero Friki
Set Alienigena1 = Friki
' y lo podemos comprobar
Debug.Print " - - - - - - - - - - - - - - - - - - -"
Debug.Print "Brrlp. Brrlp"
Debug.Print "Ahora habla el alienígena"
Debug.Print "que se ha hecho Uno con el Friki:"
Debug.Print "Soy un alienígena"
Debug.Print "en el cuerpo de " _
                & Alienigena1.NombreCompleto
' Al Alinígena1 no le gusta _
  lo que ha encontrado en el Friki
' Por ello recupera su verdadera personalidad
With Alienigena1
    .Nombre = "Christopher"
    .Apellido1 = "el Mensajero"
    .Apellido2 = "de Raticulín"
    .FechaNacimiento = #1/1/1492#
End With

'Tras recuperar Alienigena1 su personalidad _
 ¿Qué ha pasado con el Friki?
 ' que al ser la misma persona que el alienígena _
   también ha cambiado
Debug.Print " - - - - - - - - - - - - - - - - - - -"
Debug.Print "Transferida la personalidad"
```

```
        Debug.Print "del Alienígena al Friki"
        Debug.Print "Nuevo mensaje del Friki:"
        Debug.Print "Hola terrestres"
        Debug.Print "Soy " & Friki.NombreCompleto
        Debug.Print "Tengo " & Friki.Edad & " años"
        Debug.Print "Y en el pié llevo una pila"
        Debug.Print "de 2 millones de voltios"

        ' El Alienigena2 todavía está en el hiper-espacio
        ' Hagamos que pise la tierra
        With Alienigena2
            .Nombre = "Micael"
            .Apellido1 = "el Vengador"
            .Apellido2 = "de Gamínedes"
            .FechaNacimiento = #12/31/999#
        End With

        ' Y trasvasa su personalidad al Friki
        Set Friki = Alienigena2
        Debug.Print " - - - - - - - - - - - - - - - - - - -"
        Debug.Print "Brrlp. Brrlp"
        Debug.Print "Ahora el Friki es el segundo Alienígena:"
        Debug.Print "Soy " & Friki.NombreCompleto
        Debug.Print "Tengo " & Friki.Edad & " años"
        Debug.Print "Y castigaré a los impíos"

        ' Harto de tanta tontería, apago el televisor _
          y destruyo las referencias de los objetos creados
        Set Friki = Nothing
        Set Alienigena1 = Nothing
        Set Alienigena2 = Nothing
    End Sub
```

Si ejecutamos el procedimiento **PruebaDeLaClaseCPersona** nos mostrará en la ventana inmediato los siguiente:

```
- - - - - - - - - - - - - - - - - - -
Hola terrícolas
Soy Carlos Jesús el Curandero Friki
Tengo 60 años
Y he venido a salvar al Mundo
 - - - - - - - - - - - - - - - - - -
Brrlp. Brrlp
Ahora habla el alienígena
```

```
que se ha hecho Uno con el Friki:
Soy un alienígena
en el cuerpo de Carlos Jesús el Curandero Friki
 - - - - - - - - - - - - - - - - - -
Transferida la personalidad
del Alienígena al Friki
Nuevo mensaje del Friki:
Hola terrestres
Soy Christopher el Mensajero de Raticulín
Tengo 513 años
Y en el pié llevo una pila
de 2 millones de voltios
 - - - - - - - - - - - - - - - - - -
Brrlp. Brrlp
Ahora el Friki es el segundo Alienígena:
Soy Micael el Vengador de Gamínedes
Tengo 1005 años
Y castigaré a los impíos
```

Antes de que se termine el procedimiento, existen tres variables que hacen referencias a objetos de la clase **`CPersona`**.

Tras las líneas

```
Set Friki = Nothing
Set Alienigena1 = Nothing
Set Alienigena2 = Nothing
```

Destruimos los objetos, al eliminar las referencias que se hacen a ellos.

> *Un objeto existirá en memoria mientras quede alguna variable que haga referencia a él; es decir **Apunte** al objeto.*

Si tenemos declarado el objeto **`Friki`** y hacemos:

```
Set Alienigena1 = Friki
Set Alienigena2 = Friki
```

En ese momento, en realidad sólo habrá un objeto, el que hacía referencia la variable **`Friki`**, y tres variables que hacen referencia a él.

Cualquier cambio en las propiedades que hagamos en cualquiera de las variables, se reflejará inmediatamente en las otras dos, ya que estamos cambiando las propiedades del mismo objeto.

Recordemos que **es un único objeto con tres referencias diferentes** (no caigo qué, pero esta frase me recuerda a algo…)

Sigamos analizando el código del final de la entrega anterior

El plato fuerte del código era el procedimiento **PruebaConClase**, que ejecutaba una serie de operaciones con instancias de la clase **CPersona** (Objetos) que cargaba y descargaba en la colección **Empleados**.

Os recuerdo el argumento del código.

La variable **Empleados** del tipo colección estaba declarada como pública.

A continuación instanciábamos la variable **Empleados**, mediante la instrucción **New**, como hemos hecho en el caso de las clases.

Al fin y al cabo, una colección es un objeto y su existencia se rige por sus reglas.

Luego declarábamos **psnEmpleado** y creábamos con **New**, a la vez que declarábamos, la variable **Empleado** como del tipo **CPersona**.

Asignábamos datos a las propiedades de la variable **psnEmpleado** usando `With` y `End With`.

Añadíamos el objeto que instancia la variable **psnEmpleado** en la colección **Empleados**, mediante su método **Add**.

A continuación volvíamos a reconstruir el objeto al que instancia la variable **psnEmpleado**.

Para ello volvemos a utilizar **New**. Asignamos datos a sus propiedades y volvemos a añadirlo a la colección.

¿Por qué vuelvo a llamar a **Set psnEmpleado = New Cpersona**?

Si no hubiera hecho, el primer elemento de la colección, sería el mismo que el objeto que instancia la variable **psnEmpleado**. Por ello al cambiar los datos en la variable, también se cambiarían en el elemento de la colección. Es decir, **Empleados(1)** apuntaría al mismo objeto que el que es apuntado por **psnEmpleado**, por lo que, al ser el mismo, tendría las mismas propiedades, como hemos visto con el **Friki** y los **Alienígenas**.

En unas líneas posteriores, hacemos asignaciones de Objetos a Variables de objeto mediante Set, como hemos visto en el punto anterior.

Quiero resaltar que cuando hemos cargado los datos en la colección, con cada elemento le hemos asignado una clave. Esto nos permite acceder a los elementos de la colección **Empleados** mediante su **índice** o su **clave**.

Esto lo podemos ver en las líneas

```
Set Empleado = Empleados(2)
     - - - -
          ó
Set Empleado = Empleados("Martínez")
     - - - -
```

Voy a fijarme en el último procedimiento, **VaciaColeccion**.

Fijaos que este procedimiento es llamado al final del procedimiento **PruebaConClase**.

```
Public Sub VaciaColeccion(ByRef Coleccion As Collection)
    Dim i As Long
    For i = Coleccion.Count To 1 Step -1
        Coleccion.Remove (i)
    Next i
End Sub
```

¿Qué hace este procedimento?

Primero observad que el parámetro **Colección**, tiene delante la palabra **ByRef**.

ByRef hace que se pase al procedimiento la propia colección, al contrario que si se hubiera utilizado **ByVal**, que haría que lo que se pasara como parámetro al procedimiento fuera una copia del parámetro

El bucle **For . . . Next** hace que se repita el código que hay entre **For** y **Next**, mientras **i** sea mayor ó igual que **1**.

Me explico:

```
For i = Coleccion.Count To 1 Step -1
```

Hace que **i** vaya tomando como valores **n**, **n-1**, **n-2**, . . , **2** ,**1** siendo n el número de elementos de la colección, devuelto por la propiedad **Count** de la misma.

La variable **i** va cambiando de valor con cada vuelta, y el incremento ó decremento lo establece la cantidad que sigue a **Step**; en este caso **-1**, con lo que decrece de uno en uno. Si hubiera sido **2**, la variable hubiera crecido de 2 en 2. después de alcanzar i el valor que se indica a continuación de **To**, en este caso **1**, el bucle se detiene.

Si no se indica **Step Incremento**, la variable que sigue a **For Variable =** … irá creciendo de **1** en **1**.

En este ejemplo, el código que contiene el bucle es

```
Coleccion.Remove (i)
```

Remove es el método de la colección que elimina el elemento **i** de la misma.

En el ejemplo empieza por el último elemento y acaba el bucle cuando ha eliminado el primero.

Tras completar los ciclos la colección estará vacía, se saldrá del bucle y del procedimiento.