Introducci› n_POO

Su fundamento es la definici› n de **clase** para la creaci› n de **objetos**

CLASE: es una abstracci› n, una generalidad de una entidad del mundo real. Es la esencia de un conjunto, lo com‚ n a todos ellos, lo que los determina, lo que los define y les da la singularidad para ser eso y no otra cosa. Son las caracter“sticas esenciales del conjunto que le hacen definirse como lo que es. Es una plantilla, que vamos a utilizar para crear objetos concretos. Es el tipo del que nos vamos a servir para la creaci› n de objetos.

OBJETO: es la **instancia** de una clase. Es la concreci› n de una clase, representa una entidad especifica especifica y determinada del mundo real. Contiene las caracter“sticas especificas que lo diferencia de otro ente que perteneciendo al mismo conjunto. Es la concreci› n de las generalidades en en un ente que le va a permitir diferenciarse de otro que pertenezca al mismo conjunto..

La clase Alumno recoger·· la caracter“sticas que lo conformen: nombre, direcci› n, edad,..No tendr·· informaci› n sobre litros, temperatura…

INSTANCIA, es la representaci› n concreta y especifica de una clase

Una clase est·· compuesta por miembros (atributos (propiedades), m\todos(comportamiento)).

```
Class   CCuenta
      'Atributos
      Private nombre As String
      Private cuenta As String
      Private saldo As Double
      Private tipoDeInter\s As Double
      ...
      ...
      'M\todos
       Public Sub New()
       End Sub
       Public Sub New(nom As String, cue As String, _
                      sal As Double, tipo As Double)
            asignarNombre(nom)
            asignarCuenta(cue)
           i ngreso(sal)
            aignarTipoDeInters(tipo)
      End Sub

End Class
```

Los atributos generalmente se declaran Private, para que no se pueda acceder a ellos desde fuera de la clase. Esta ocultaci›n es lo que se denomina **ENCAPSULAMIENTO**, con esto se protege la estructura interna del objeto y se obliga a programar pensando y creando objetos para utilizar su interfaz publica.

Los m\todos la mayor“a de ellos se declaran **p blicos** para poder acceder desde cualquier punto del proyecto.

Los m\todos, son los que le dan funcionalidad al objeto. Son los procedimientos (Sub o Function). Si existe mas de un m\todo con el mismo nombre, pero que difieren en el n mero de par·metros o en el tipo de alguno de ellos, se dice que el m\todo est·· **SOBRECARGADO**

Para acceder a los m\todos de una clase crearemos previamente un objeto y utilizaremos la sintaxis

```
Dim objeto As Calumno = New Calumno(xxx,xxx,xxx)
objeto.metodo(xxx)
```

El **CONSTRUCTOR**, es un m\todo esencia y especial de la clase que es llamado autom·ticamente siempre que se crea un objeto de esa clase. Su funci›n es iniciar el objeto. Se distingue f·cilmente porque tiene el nombre **New** y no puede retornar un valor (procedimiento de tipo **Sub**).

```
Public Sub New()
End Sub
```

Un *constructor por omisi›n* de una clase, es un constructor sin par·metros que no hace nada. El objeto ser·· iniciado con los valores predeterminados por el sistema (los atributos num\ricos a ceros y los alfanum\ricos y las referencias a objetos a Nothing).

Los constructores, salvo en casos excepcionales, deben declararse siempre p blicos para que puedan ser invocados desde cualquier parte

Podemos a‹adir tambi\n, un constructor a la clase con el fin de poder iniciar los atributos de cada nuevo objeto con unos valores determinados pasados como argumentos en el instante en el que se solicita crearlo: Este seria un ejemplo de un m\todo que est·· **sobrecargado**

Hemos dicho que los atributos se declaran private y los m\todos la mayor“a de ellos se declaran **p blicos** Cuando creamos objetos cada objeto que creemos de esa clase mantiene su propia copia de los atributos para almacenar sus datos particulares; pero, de los m\todos s›lo hay una copia para todos los objetos, ya que es comunes a todos ellos

SHARED: Si un atributo hiciera referencia al curso(¿?) de los alumnos y que esta fuera igual para todos los objetos de esta clase, el *atributo de la clase* almacena informaci›n com‡n a todos los objetos de esa clase; no tiene sentido que se guarde la misma informaci›n en todos objetos. Para especificar un atributo com‡n a todos los objetos de su clase, cuando se declare hay que anteponer la palabra reservada **Shared** al nombre del mismo. (**Static** en otros lenguajes)

```
Private Shared tipoDeInter\s As Double
```

An·logamente, un m\todo declarado **Shared** es un m\todo de la clase, por lo tanto no se ejecuta para un objeto particular, sino que se utiliza para actuar sobre un atributo **Shared** declarado privado, o bien para realizar alguna operaci›n gen\rica al margen de los objetos de la clase.

```
Public Shared Sub setTipoDeInter\s(xxx)
        xxx
End Sub
```

Para acceder a un miembro compartido de una clase (miembro Shared; static en otros lenguajes) se puede utilizar un objeto de la clase , o bien el nombre de la clase, lo cual es l›gico porque nos estamos refiriendo, no a un objeto en particular de dicha clase sino a todos los objetos que el programa haya creado de la misma. Se utilizar·· la sintaxis:

```
clase.metodo()
```

El m\todo **WriteLine** miembro de la clase **Console** del espacio de nombres **System** es p‡blico (**Public**) y compartido (**Shared**).

```
System.Console.WriteLine( … )
          clase  . m\todo
```

En la clase Leer, crearemos todos los m\todos de tipo Shared, para que me permita hacer las lecturas de todos los tipos de datos…. sin necesidad de crear objetos

```
        variable  =  Leer.Entero()
Class  Leer
        ....
        Public  Shared  Function  datoInt()  As  Integer
                Try
                        Return Int32.Parse(Console.ReadLine())
                Catch e As FormatException
                        Return Int32.MinValue ' valor m·s peque‹o
                End Try
        End Function
End  Class
```

```
variable = Leer.datoInt()
```

Existen en VB un nuevo elemento es la **Property**

```
Public Property nombrePropiedad() As tipo
        Get.
                ' Aqu“ se devuelve el valor del atributo
                Return dpto
        End Get.
        Set(Value As tipo)
                ' Aqu“ se asigna el valor del atributo
                dpto =
        End Set
End Property
```

```
Private dptoE As String
Public Property Dpto() As String
        Get
                Return dptoE
        End Get
        Set(Dato As String)
                dptoE = Dato
        End Set
End Property
```

miObjeto.Dpto = "Inform·tica" Invoca al **SET**

System.Console.WriteLine(**miObjeto.Dpto**) Invoca al **GET**

Ojo!! observar que al llamarle a la propiedad no se ponen los par\ntesis.

Visual Basic .NET pone a nuestra disposici›n una instrucci›n, que al igual que Sub o Function, nos permiten declarar un procedimiento que tiene un trato especial, este es el caso de **Property**.

La forma de usar Property es muy parecido a como se declara una funci›n, pero con un tratamiento especial, ya que dentro de esa declaraci›n hay que especificar por un lado lo que se debe hacer cuando se quiera recuperar el valor de la propiedad y por otro lo que hay que hacer cuando se quiere asignar un nuevo valor.

¿C›mo se declara un procedimiento Property?

Si queremos que **Nombre** sea realmente una propiedad (un procedimiento del tipo Property) para que podamos hacer ciertas comprobaciones tanto al asignar un nuevo valor como al recuperar el que ya tiene asignado, tendremos que crear un procedimiento como el que te muestro a continuaci›n:

```
' variable (campo) privado para guardar el nombre
Private elNombre As String

Public Property nombreEmpleado() As String
   ' la parte Get es la que devuelve el valor de la propiedad
   Get
```

```
        Return elNombre
    End Get
    ' la parte Set es la que se usa al asignar el nuevo valor
    Set(ByVal Value As String)
        If Value <> "" Then
            elNombre = Value
        End If
    End Set
End Property
```

Un procedimiento del tipo Property, tiene dos bloques internos:

El primero es el bloque **Get**, que ser··el c〉digo que se utilice cuando queramos recuperar el valor de la propiedad. Se pondr··:

ctNombre.Text = objEmpleado.**nombreEmpleado**

El segundo es el bloque **Set**, que ser··el c〉digo que se utilice cuando queramos asignar un nuevo valor a la propiedad.

Como puedes comprobar, el bloque Set recibe un par·metro llamado **Value** que es del mismo tipo que la propiedad, en este caso de tipo String. Value representa el valor que queremos asignar a la propiedad. Se pondr··:

objEmpleado.**nombreEmpleado** = "Ane Muguruza"

objEmpleado.**nombreEmpleado** = ctNombre.Text

F"jate que al declarar la propiedad, no se indica ning" n par·metro.