Programación Orientada a Objetos (POO)

Visual Basic .NET

¿Qué es POO?

- No es un lenguaje de programación sino una forma de programar, una metodología.
- Intenta llevar al código lo mismo que encontramos en el mundo real.
- Cuando miramos a nuestro alrededor, ¿qué vemos?
- Vemos OBJETOS
  - Podemos distinguir un objeto por que pertenece a una **clase**.
  - Distinguimos un perro de un coche porque pertenecen a clases diferentes.
  - Distinguimos un televisor de otro porque aunque pertenezcan a la misma clase, cada objeto es diferente.
- Esta es el modelo que intenta seguir la POO para estructurar un sistema.

Características de la POO: Abstracción

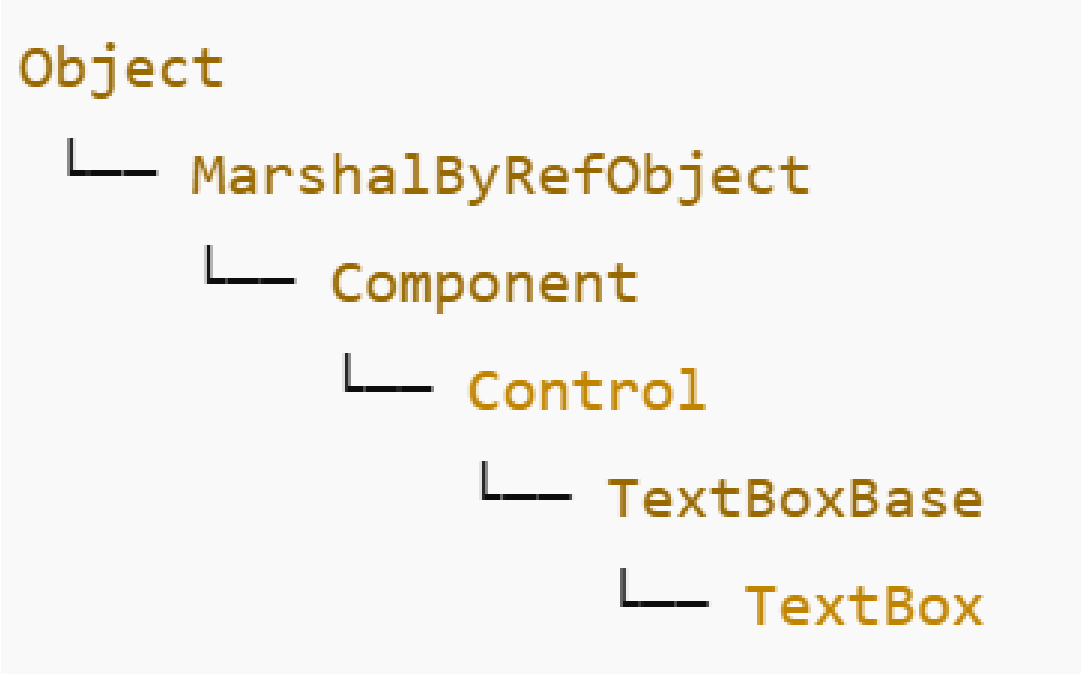
- La abstracción es algo conceptual. Persigue seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes.
- Define nuevos tipos de entidades en el mundo real.
- Depende de la perspectiva del observador.
- Se centra en la visión externa de un objeto, Ej.: Pastor, pekinés o bulldog son derivados de la idea abstracta “perro”.

Características de la POO: Jerarquía y herencia

- Las abstracciones forman una jerarquía.
- Seccionamos el código en partes más pequeñas que al unirlas hacen el trabajo.
- Ej.: Máquinas de locomoción→de tierra →sin motor→con pedales→para niños → triciclo.
- Cada parte hereda propiedades y comportamientos de su antecesor.
- Ventajas: Evita tener que escribir el mismo código una y otra vez ya que al definir una categoría (que llamaremos clase) que pertenece a otra, automáticamente atribuimos las características de la primera sin tener que definirlas de nuevo

Ejemplo de Jerarquía real

- **Object**: Clase de más alto nivel
- **MarshalByRefObject**: Permite que los objetos interactúen con otros dominios de aplicación
- **Component**: Permite que el objeto se use como componente en un formulario o diseñador
- **Control**: Define propiedades visuales como tamaño, color y eventos.
- **TextBoxBase**: Clase base para controles de texto editables.
- **TextBox**: Control



Características de la POO: Encapsulación y ocultamiento (I)

- ¿Puede alguien modificar algo de un objeto? Si el objeto estuviese bien definido, los datos tendrían que ser accesibles y sólo deberían ser modificados mediante un método del objeto.
- La encapsulación se encarga de mantener ocultos los procesos internos que se necesitan para hacer lo que sea que haga.
- El programador accede sólo a lo que necesita.

Características de la POO: Encapsulación y ocultamiento (II)

- Un objeto se comunica con el resto del mundo mediante su interfaz → la lista de métodos y propiedades que hace público.
- Ventaja: Lo que hace el usuario puede ser controlado internamente, incluso los errores, evitando que todo colapse.

Clase

- Sabemos que “ave” se refiere a algo que cumple unas determinadas características, se comporta de una forma concreta → tiene plumas, pico, dos patas… → “ave” es una palabra que nos permite clasificar las cosas.
- “ave” identifica a un grupo, no a un ave en concreto.
- Es un concepto, una abstracción.
- La clasificación (crear clases) es la tarea de agrupar cosas según sus características y sus capacidades.
- Cada objeto debe pertenecer a una clase específica, de acuerdo a sus características y si esa clase no existe, se crea.

Objeto

- Una clase es una serie de código que define a todos los elementos relacionados.
- Es una plantilla, un molde a partir de la que se crean los objetos.
- La clase posee todas las variables y funciones que deben tener un objeto.
- El objeto tendrá valores propios en esas variables cuando sea creado.
- Un objeto no es una clase, sino el resultado de crear un ejemplar de esa clase.

Clases en VB

- La plataforma .net tiene miles de clases (Math, button…) preconstruidas que utilizamos para desarrollar aplicaciones.
- Formato de una clase en Visual Basic:

```
Class NombreClase
    constructor
    variables
    métodos
End Class
```

- La palabra clave Class siempre encabeza la definición de una clase.

Objetos en VB

- Una clase es una acción declarativa y no ocurre nada hasta que no creamos su objeto.
- A la creación de un ejemplar de una clase se le denomina instanciación. Al objeto, instancia.
- En Visual Basic usamos los objetos para llevar a cabo todos los trabajos.

Objetos en VB

- Usaremos la palabra clave new para construir el objeto.
- Hay clases que requieren argumentos.
- Ejemplo: `Dim punto as new Point (2,2)`
- New hace 3 cosas:
  - Crea una nueva instancia de la clase.
  - Localiza espacio en la memoria para el nuevo objeto.
  - Llama a un método especial de la clase, el constructor.
  - VB se ocupa en la propia ejecución del programa de destruir los objetos no utilizados. Es lo que se llama “recolección de basura”.

Atributos/variables

- Forman la estructura interna de los objetos de una clase. Son los que hacen a un objeto diferente de otro.
- Se declaran exactamente igual que cualquier otra variable pudiendo incluso inicializarse con un valor.

```
Class MiClase
  Private edad as Short
  Public DNI as String =“00000000A”
End class
```

Constructores

- Son métodos que se llaman automáticamente al crear un objeto de una clase.
- Su misión principal es iniciar nuevos objetos de una clase.
- En Visual Basic los constructores siempre serán métodos llamados **New** y:
  - No pueden retornar valores.
  - Pueden tomar parámetros.
  - Pueden sobrecargarse.
  - Si no se define uno se añade automáticamente uno por defecto (sin parámetros) aunque se recomienda que siempre se codifique el constructor por defecto.
  - Si creamos uno o varios constructores no se añade el constructor por defecto. Si quisiéramos también el constructor básico (sin parámetros) lo tendríamos que codificar.

Métodos

- Un método es un conjunto de declaraciones asociadas a un nombre.
- Se ejecutarán todas cuando se llame al método.
- Forman lo que se denomina interfaz o medio de acceso a la estructura interna de las clases.
- Definen las operaciones que pueden realizarse con las propiedades.

Métodos

- En Visual Basic hay dos formas de crear un método:
  - Se usa la instrucción **Sub** si el método no devuelve un valor.
  - Se usa la instrucción **Function** si el método devuelve un valor.
- La definición de un método tiene 3 partes básicas:
  - El nombre.
  - Los parámetros si los tiene.
  - El cuerpo.
- Visual Basic permite la sobrecarga de métodos, es decir, varios métodos con el mismo nombre en una clase mediante la instrucción **Overloads**.

Herencia

- Es la mejor solución para la reutilización de código.
- Suponemos una clase personas y desarrollamos una aplicación de nóminas.
- En las nóminas hay empleados y los empleados son personas.
- Todo empleado tiene nombre y apellidos, pero no toda persona tiene un sueldo.
- ¿Qué hay más fácil que definir la clase Empleado a partir de la clase Persona? ¿cómo se aprovecha este código?

Herencia

```
Class Empleado
   Inherits Persona
   Private Sueldo as Integer
   ......
End Class
```

- Por su parte, una clase hija, se puede trasformar en clase base de otra nueva.

```
Class Gerente
   Inherits Empleado
   Private Area as String
   ......
End Class
```

- No todos los empleados son gerentes. Los gerentes tienen un sueldo (Empleado) y un nombre y apellidos (Persona) y un área, propia de la clase.
- No sólo se reutiliza el código, sino que el código resultante tiene mayor fiabilidad ya que se minimiza la codificación.

Overridable / Overrides

- Mediante **Overridable** declaramos en una superclase los métodos que pueden ser reemplazados en una subclase.
- Es en la subclase donde mediante **Overrides** se especifica qué métodos serán reemplazados.

Alcance

- Sabemos que la encapsulación es la capacidad de ocultar los miembros de una clase (propiedades y métodos).
- Para evitar que un programador acceda a propiedades o métodos de una clase se restringe la accesibilidad de los datos definiéndolos como **Private**.
- Cuando se crea un objeto de una clase, desde fuera del objeto sólo podemos acceder a miembros públicos (**Public**).
- Como regla general las variables son privadas y los métodos públicos (a no ser que se quiera impedir su uso).

Modificadores de acceso

- **Friend**: Permite el acceso de cualquier clase del mismo ensamblado.
- **Private**: **Es el modificador que el lenguaje pone por defecto.** Lo que se declara con Private sólo puede ser accedido por la propia clase. Afecta también a las herencias.
- **Public**: Lo que se define con Public es accesible desde cualquier otra clase, incluídas las subclases.
- **Protected**: Lo que se define con Protected limita el acceso a las subclases de esa clase.

Clases estáticas

- Crea variables y métodos de clase.
- Son accesibles al poner el nombre de la clase y un punto.
- Por el contrario, las variables y métodos de instancia son accesibles desde la creación del objeto.
- Una clase puede ser estática, pero sus métodos y variables deberán ser también estáticos. Se deben definir con el modificador **Shared.**
- No permitirá crear objetos de ella. No podemos crear objetos de la clase Math, por ejemplo.
- Ejemplo: `Math.Pow(2,3);`

Espacios de nombres

- Visual Basic permite definir paquetes o conjuntos de clases. Se llaman espacios de nombres.
- El espacio de nombres de una clase coincide con el nombre del proyecto.
- En cada proyecto se importan directamente algunos espacios de nombres.
- Si queremos usar una clase concreta primero debemos saber cuál es para cargar su ensamblado en el árbol de referencias.
- Un ensamblado es la dll donde está compilada la clase.

Espacios de nombres

- Después llamaremos a su espacio de nombres (namespace) concreto empleando la sentencia **imports nombre_del_espaciodenombres**
- Para acceder a una clase dentro del espacio de nombres sin usar imports se puede usar la notación **nombre_del_espaciodenombres.nombre_de_la_clase**

Interfaces

- Una interfaz en Visual Basic (y en programación en general) es como un contrato que dice:"Cualquier clase que me implemente debe tener ciertos métodos o propiedades.”
- Una interfaz define qué debe hacer una clase, pero no cómo lo hace.
- Sirven para:
  - Unificar el comportamiento entre clases distintas.
  - Organizar el código de forma más clara y flexible.

talio

SISTEMAS
SOLUCIONES
FORMACIÓN
INGENIERÍA
SERVICIOS GESTIONADOS

www.talio.it

Alameda Mazarredo, 69 - 8º A 48009 BILBAO T.+34 94 651 99 90