PROGRAMACIÓN ORIENTADA A OBJETOS

La programación Orientada a objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación.

Con la POO tenemos que aprender a pensar las cosas de una manera distinta, para escribir nuestros programas en términos de objetos, propiedades, métodos y otras cosas que veremos rápidamente para aclarar conceptos y dar una pequeña base que permita soltarnos un poco con este tipo de programación; es otra forma de descomponer problemas. Este nuevo método de descomposición es la descomposición en objetos; vamos a fijarnos no en lo que hay que hacer en el problema, sino en cuál es el escenario real del mismo, y vamos a intentar simular ese escenario en nuestro programa.

Los lenguajes de programación tradicionales no orientados a objetos, como C, Pascal, BASIC, o Modula-2, basan su funcionamiento en el concepto de procedimiento o función. Una función es simplemente un conjunto de instrucciones que operan sobre unos argumentos y producen un resultado. De este modo, un programa no es más que una sucesión de llamadas a funciones, ya sean éstas del sistema operativo, proporcionadas por el propio lenguaje, o desarrolladas por el mismo usuario. En el caso de los lenguajes orientados a objetos, como es el caso de C++ y Java, el elemento básico no es la función, sino un ente denominado precisamente objeto.

Por ejemplo vamos a pensar en un carro para tratar de modelizarlo en un esquema de POO. Diríamos que el carro es el elemento principal que tiene una serie de características, como podrían ser el color, el modelo la marca.

Además tiene una serie de funcionalidades asociadas, como pueden ser ponerse en ar marcha, parar o aparcar.

Pues en un esquema POO el carro sería el objeto, las propiedades serían las características como el color o el modelo y los métodos serían las funcionalidades asociadas como ponerse en marcha o parar.

Este objeto se podrá utilizar en los programas, por ejemplo en un programa que gestione un taller de coches utilizarás objetos coche. Los programas Orientados a objetos utilizan muchos objetos para realizar las acciones que se desean realizar y ellos mismos también son objetos. Es decir, el taller de coches será un objeto que utilizará objetos coche, herramienta, mecánico, recambios, etc.

Programación y lenguaje orientada a objetos

La programación orientada a objetos es una "filosofía", un modelo de programación, con su teoría y su metodología, mientras que un lenguaje orientado a objetos es un lenguaje de programación que permite el diseño de aplicaciones orientadas a objetos.

Características de la P.O.O.:

- ***Abstracción:***

Cada vez que pronunciamos una palabra, realmente lo que hacemos es asociar ese sonido (o ese conjunto de garabatos al escribir) con una serie de cosas.

Decimos que una ave es tal cosa, que una silla es tal otra, etc.

Cuando vamos a aplicar la POO, lo primero que debemos hacer es cumplir con una vieja máxima de guerra: *Divide y Vencerás*. Es decir, lo que hacemos es seccionar nuestro código en grupos de código más pequeño que, al unirlos, hacen el trabajo. Un buen ejemplo de abstracción es el cuerpo humano, aunque el cuerpo es una unidad, está dividido en lo que conocemos por ***sistemas*** (el sistema respiratorio, el sistema linfático, cardiovascular, etc., etc.). Estos sistemas, a su vez están compuestos por otros más pequeños: los órganos, y así sucesivamente. **La abstracción nos permite dividir nuestro programa en distintos objetos que se agrupan para formar cosas más complejas.**

Pero ¿qué es realmente la abstracción? Básicamente es la capacidad de separar los elementos (al menos mentalmente) para poder verlos de forma singular. Como cuando describimos el cuerpo humano y decimos cabeza, brazo(s), pierna(s), etc.

- ***Encapsulación:***

También conocida como **ocultamiento**. Cuando muchas personas se acuestan a ver televisión no se preocupan del modo como éste funciona, o lo que hace para cambiar de canal o aumentar el volumen, sólo saben que al presionar un botón ocurre la magia.

La encapsulación se encarga de mantener ocultos los procesos internos que necesita para hacer lo que sea que haga, dándole al programador acceso **sólo a lo que necesita**. Esto da dos ventajas iniciales:

* Lo que hace el usuario puede ser controlado internamente (incluso sus errores), evitando que todo colapse por una intervención *indeseada* (tú no quieres que tu mamá, que no tiene ni idea de electrónica, abra tu televisor y empiece a jugar con los circuitos para cambiar los canales *manualmente* ¿verdad?).

* La segunda ventaja es que, al hacer que la mayor parte del código esté *oculto*, puedes hacer cambios y/o mejoras sin que eso afecte el modo como los usuarios van a utilizar tu código. Sólo tienes que mantener igual la forma de acceder a él (en el caso del control de la tele, que los botones sigan siendo los mismos y que el botón de "apagado" no cambie el volumen). Por cierto, estas *puertas de acceso* que das a los usuarios son lo que se conoce como *interfaz*.

- ***Herencia:***

Uno de los elementos más interesantes de la P.O.O. La herencia es la capacidad que tiene una clase de *derivar* las propiedades y métodos de otra. Tratemos de explicarlo con un ejemplo:

Decimos que una gallina es un ave; esto quiere decir que las gallinas tienen características comunes con otras aves (pico, plumas, etc.), es decir que la gallina hereda las características comunes de todas las aves. Pero además, resulta que un ave es un animal, lo que significa que también comparte características comunes al caballo, el perro, el hombre y cualquier otra cosa que pueda ser clasificada como animal.

La herencia nos permite, entre otras cosas, evitar tener que escribir el mismo código una y otra vez, puesto que al definir que una categoría (que en programación llamaremos ***clase***) pertenece a otra, automáticamente estamos atribuyéndoles las características generales de la primera, sin tener que definirlas de nuevo.

¿QUÉ ES UN OBJETO?

Un objeto es la representación en un programa de un concepto, y contiene toda la información necesaria para abstraerlo: datos que describen sus atributos y operaciones que pueden realizarse sobre los mismos; por lo cual no es más que un conjunto de variables (o datos) y métodos (o funciones) relacionados entre sí.
Los objetos en programación se usan para modelar objetos o entidades del mundo real (el objeto carro). Un objeto es, por tanto, la representación en un programa de un concepto, y contiene toda la información necesaria para abstraerlo: datos que describen sus atributos y operaciones que pueden realizarse sobre los mismos

CLASE

Una *clase*, es simplemente una abstracción que hacemos de nuestra experiencia sensible. El ser humano tiende a agrupar seres o cosas -objetos- con características similares en grupos -clases-. Así, aun cuando existen por ejemplo multitud de vasos diferentes, podemos reconocer un vaso en cuanto lo vemos, incluso aun cuando ese modelo concreto de vaso no lo hayamos visto nunca. El concepto de *vaso* es una abstracción de nuestra experiencia sensible.
Quizás el ejemplo más claro para exponer esto lo tengamos en las taxonomías; los biólogos han dividido a todo ser (vivo o inerte) sobre la tierra en distintas clases.

ARQUITECTURA DE TRES CAPAS

- ***Capa de presentación***

Esta capa es la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso.
Dicha capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener la característica de ser “amigable” para el usuario, generalmente se presentan como formularios.

- ***Capa de negocio***

Aquí es donde, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen las reglas que deben cumplirse.
Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él.
Es importante aclarar que no toda lógica de negocio es la misma, algunas no requieren un frecuente acceso a los datos, pero una interface de usuario robusta necesitaría de la lógica de negocios para la validación en la entrada de campos, cálculos en tiempo real u otras interacciones de usuarios.

- ***Capa de datos***

Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.