INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS (POO)

Diferencias entre Programación Estructurada y POO

Programación Estructurada

La programación estructurada nació como solución a los problemas que presentaba la programación no estructurada, la cual se empleó durante mucho tiempo antes de la invención de la programación estructurada.

Un programa no estructurado es un programa procedimental: las instrucciones se ejecutan en el mismo orden en que han sido escritas. Sin embargo, este tipo de programación emplea la instrucción "goto". Una instrucción "goto" permite pasar el control a cualquier otra parte del programa, es decir “hacer saltos a otra sección del código”. Cuando se ejecuta una instrucción "goto" la secuencia de ejecución del programa continúa a partir de la instrucción indicada en el salto.

Programación orientada a objetos

La programación orientada a objetos o POO (OOP – Object Oriented Programming según sus siglas en inglés), es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas informáticos. Está basado en varias técnicas que son sus pilares conceptuales, incluyendo herencia, abstracción, polimorfismo y encapsulamiento. Su uso se popularizó a principios de la década de los años 1990. En la actualidad, existen variedad de lenguajes de programación que soportan la orientación a objetos.

Los objetos son entidades que tienen un determinado estado formado por sus **atributo**, tienen **comportamiento** a través de sus métodos, y poseen identidad:

- El estado está compuesto de datos, será uno o varios atributos a los que se habrán asignado valores concretos.
- El comportamiento está definido por los métodos o mensajes a los que sabe responder dicho objeto, es decir, qué operaciones se pueden realizar con él.
- La identidad es una propiedad de un objeto que lo diferencia del resto, dicho con otras palabras, es su identificador (concepto análogo al de identificador de una variable o una constante).

Un objeto se construye a partir de la clase que lo define. La clase es la plantilla del objeto, un molde.

Un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos. A su vez, los objetos disponen de mecanismos de interacción llamados métodos, que favorecen la comunicación entre ellos. Esta comunicación favorece a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separa el estado y el comportamiento.

Beneficios de la Programación Orientada a Objetos

La programación orientada a objetos aporta a los desarrolladores los medios de enfrentarse a sus nuevos retos, con:

- Una organización modular muy cercana a la realidad, pues un sistema se piensan en los “objetos que interactúan entre sí”.
- Procesos de creación, puesta a punto y mantenimiento de los componentes, más sencillos y rápidos.
- La reutilización y evolución de los componentes existentes o de aquellos que provienen de proveedores de software de terceros.
- Una integración sencilla para su funcionamiento en entornos gráficos.
- Una lógica de codificación compatible con las aplicaciones distribuidas, que reparten sus contenidos en varias máquinas.
- Un desacoplamiento de la aplicación, que permite un trabajo en equipo más eficaz y productivo. Esto significa que no existe código repetido y solapado (que hace lo mismo), en distintas partes de un sistema.

Y los principios básicos de la POO son:

- **Abstracción**: denota las características esenciales de un objeto del mundo real, objeto tangible o intangible, donde se capturan sus comportamientos. El comportamiento correcto en POO consiste en establecer relaciones entre objetos lo más abstractas posible. Por ejemplo, un origen de datos puede provenir de una entrada por teclado, del contenido de un archivo, de una conexión de red, etc. Si determinada funcionalidad de un objeto recibe como argumento una sucesión de datos, será interesante abstraerse de su origen, considerándola como un flujo "genérico de datos" (*stream*), sin pensar si viene del teclado o de un archivo, etc.

- **Encapsulamiento**: significa reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. El encapsulamiento consiste en crear un tipo de **caja negra** que contiene internamente el comportamiento del objeto, y externamente un conjunto de comandos que van a permitir manipularlo desde el exterior, esto implica que el usuario del objeto jamás podrá acceder internamente a sus datos, sólo lo hará desde los comandos.

- **Principio de ocultación**: cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una interfaz a otros objetos que especifica cómo pueden interactuar unos con otros.

- **Herencia**: Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. Por ejemplo, crear una clase de base, llamada *Empleado*, que contiene la información común a todos los empleados de una empresa. Después crear una clase llamada *Ejecutivo*, que heredará de la clase base *Empleado*. *Ejecutivo* heredará los miembros (datos y funcionalidad) de *Empleado* y agregará a esta lista de miembros los aspectos específicos del tipo ”Ejecutivo”. En este caso, se dice que *Ejecutivo* extiende *Empleado* o que *Ejecutivo* hereda de *Empleado*. *Ejecutivo* es la clase derivada y *Empleado* es la clase base.

- **Polimorfismo**: permite tener comportamientos similares, asociados a objetos distintos. El polimorfismo de los objetos está muy relacionado con la herencia. La raíz etimológica de la palabra conduce a pensar de manera natural que el objeto puede adoptar varias formas. Para entender en qué medida esto es posible, imaginemos que tenemos un objeto *Ingeniero* que hereda de la clase *Ejecutivo* y, por tanto, un objeto *Ingeniero* es un tipo de *Ejecutivo*. Gracias al polimorfismo, en cualquier sitio donde se espere un objeto *Ejecutivo* se podrá utilizar un objeto *Ingeniero*.