HINTS

POO

A lo largo de la historia, han ido apareciendo diferentes paradigmas de programación. Lenguajes secuenciales como COBOL, o procedimentales como Basic o C, se centraban más en la lógica que en los datos. Otros más modernos como Java, C# y Python, utilizan paradigmas para definir los programas, siendo la Programación Orientada a Objetos la más popular.

Con el paradigma de Programación Orientado a Objetos, lo que buscamos es dejar de centrarnos en la lógica pura de los programas, para empezar a pensar en objetos, lo que constituye la base de todo esto. Ayuda muchísimo en sistemas grandes, ya que, en vez de pensar en funciones, pensamos en las relaciones o interacciones de los diferentes componentes del sistema.

¿POR QUÉ POO?

La Programación Orientada a Objetos permite que el código sea reutilizable, organizado y fácil de mantener. Sigue el principio de desarrollo de software utilizado por muchos programadores **DRY (***Don't Repeat Yourself***)**, para evitar duplicar el código y crear de esta manera programas eficientes. Además, evita el acceso no deseado a los datos, o la exposición de código propietario mediante la encapsulación y la abstracción.

BENEFICIOS DE PROGRAMACIÓN ORIENTADA A OBJETOS

- Reutilización del código.
- Convierte cosas complejas en estructuras simples reproducibles.
- Evita la duplicación de código.
- Permite trabajar en equipo gracias al encapsulamiento, ya que minimiza la posibilidad de duplicar funciones cuando varias personas trabajan sobre un mismo objeto al mismo tiempo.
- Al estar la clase bien estructurada permite la corrección de errores en varios lugares del código.
- Protege la información a través de la encapsulación, ya que solo se puede acceder a los datos del objeto a través de propiedades y métodos privados.

- La abstracción nos permite construir sistemas más complejos, y de una forma más sencilla y organizada.

La Programación Orientada a Objetos es actualmente el paradigma que más se utiliza para diseñar aplicaciones y programas informáticos. Son muchas sus ventajas, principalmente cuando necesitas resolver desafíos de programación complejos. Permite una mejor estructura de datos y reutilización del código, lo que facilita el ahorro de tiempo a largo plazo. Eso sí, para ello se requiere pensar bien en la estructura del programa, planificar al comienzo de la codificación, así como también analizar los requisitos en clases simples y reutilizables que se pueden usar para diseñar instancias de objetos.