

3.A. Introducción y conceptos de la POO

✓ Hecho

2. Fundamentos de la Programación Orientada a Objetos.

2.3. Características.

Cuando hablamos de Programación Orientada a Objetos, existen una serie de características que se deben cumplir. Cualquier lenguaje de programación orientado a objetos las debe contemplar. Las características más importantes del paradigma de la programación orientada a objetos son:

- **Abstracción.** Es el proceso por el cual definimos las características más importantes de un objeto, sin preocuparnos de cómo se escribirán en el código del programa, simplemente lo definimos de forma general. En la Programación Orientada a Objetos la herramienta más importante para soportar la abstracción es la **clase**. Básicamente, una clase es un tipo de dato que agrupa las características comunes de un conjunto de objetos. Poder ver los objetos del mundo real que deseamos trasladar a nuestros programas, en términos abstractos, resulta de gran utilidad para un buen diseño del software, ya que nos ayuda a comprender mejor el problema y a tener una visión global de todo el conjunto. Por ejemplo, si pensamos en una clase **Vehículo** que agrupa las características comunes de todos ellos, a partir de dicha clase podríamos crear objetos como **Coche** y **Camion**. Entonces se dice que **Vehículo** es una abstracción de **Coche** y de **Camion**.
- **Modularidad.** Una vez que hemos representado el escenario del problema en nuestra aplicación, tenemos como resultado un conjunto de objetos software a utilizar. Este conjunto de objetos se crean a partir de una o varias clases. Cada clase se encuentra en un archivo diferente, por lo que la modularidad nos permite modificar las características de la clase que define un objeto, sin que esto afecte al resto de clases de la aplicación.
- **Encapsulación.** También llamada "**ocultamiento de la información**". La **encapsulación** o **encapsulamiento** es el mecanismo básico para ocultar la información de las partes internas de un objeto a los demás objetos de la aplicación. Con la encapsulación un objeto puede ocultar la información que contiene al mundo exterior, o bien restringir el acceso a la misma para evitar ser manipulado de forma inadecuada. Por ejemplo, pensemos en un programa con dos objetos, un objeto **Persona** y otro **Coche**. **Persona** se comunica con el objeto **Coche** para llegar a su destino, utilizando para ello las acciones que **Coche** tenga definidas como por ejemplo conducir. Es decir, **Persona** utiliza **Coche** pero no sabe cómo funciona internamente, sólo sabe utilizar sus métodos o acciones.
- **Jerarquía.** Mediante esta propiedad podemos definir relaciones de jerarquías entre clases y objetos. Las dos jerarquías más importantes son la jerarquía "**es un**" llamada **generalización** o **especialización** y la jerarquía "**es parte de**", llamada **agregación**. Conviene detallar algunos aspectos:
 - La generalización o especialización, también conocida como **herencia**, permite crear una clase nueva en términos de una clase ya existente (herencia simple) o de varias clases ya existentes (herencia múltiple). Por ejemplo, podemos crear la clase **CochedeCarreras** a partir de la clase **Coche**, y así sólo tendremos que definir las nuevas características que tenga.
 - La agregación, también conocida como **inclusión**, permite agrupar objetos relacionados entre sí dentro de una clase. Así, un **Coche** está formado por **Motor**, **Ruedas**, **Frenos** y **Ventanas**. Se dice que **Coche** es una agregación y **Motor**, **Ruedas**, **Frenos** y **Ventanas** son agregados de **Coche**.
- **Polimorfismo.** Esta propiedad indica la capacidad de que varias clases creadas a partir de una antecesora realicen una misma acción de forma diferente. Por ejemplo, pensemos en la clase **Animal** y la acción de expresarse. Nos encontramos que cada tipo de **Animal** puede hacerlo de manera distinta, los **Perros** ladran, los **Gatos** maullan, las **Personas** hablamos, etc. Dicho de otra manera, el polimorfismo indica la posibilidad de tomar un objeto (de tipo **Animal**, por ejemplo), e indicarle que realice la acción de expresarse, esta acción será diferente según el tipo de mamífero del que se trate.

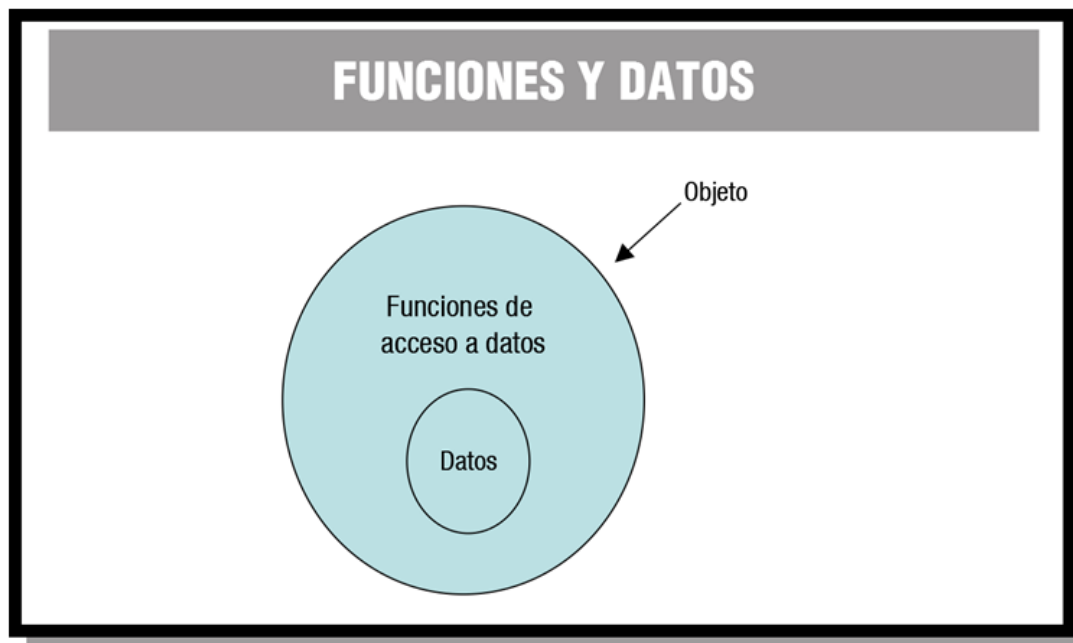


Imagen extraída de curso Programación del MECD.

◀ Orientaciones para el alumno UT03.

Ir a...

3.B. Objetos y clases. ▶