**Programación Orientada a Objetos**

**Tomado de:** [**https://elibro-net.bdigital.sena.edu.co/es/ereader/senavirtual/106461**](https://elibro-net.bdigital.sena.edu.co/es/ereader/senavirtual/106461)[**https://elibro-net.bdigital.sena.edu.co/es/ereader/senavirtual/70431?page=17**](https://elibro-net.bdigital.sena.edu.co/es/ereader/senavirtual/70431?page=17)

**DEFINICIÓN**

La programación orientada a objetos es un paradigma de programación totalmente diferente al método clásico de programación, el cual utiliza objetos y su comportamiento para resolver problemas y generar programas y aplicaciones informáticas.   
  
Con la programación orientada a objetos (Poo) se aumenta la modularidad de los programas y la reutilización de los mismos. Además, la Poo se diferencia de la programación clásica porque utiliza técnicas nuevas como el polimorfismo, el encapsulamiento, la herencia, etc.   
  
Generalmente, los lenguajes de última generación permiten la programación orientada a objetos, así como la programación clásica. Por esta razón, puede entenderse la Poo como una evolución de la programación clásica (programación estructurada).

**CICLO DE DESARROLLO DEL SOFTWARE PROGRAMACIÓN ORIENTACIÓN A OBJETOS: ANÁLISIS, DISEñO Y PROGRAMACIÓN ORIENTADA A OBJETOS**

La programación orientada a objetos rompe definitivamente con el paradigma de la programación estructurada, surge el concepto de “objeto”. Todos los programas se reducen a objetos que tienen una serie de atributos, y, asociados, un comportamiento o procedimientos llamados “métodos”. Estos objetos, que son instancias de clases, interaccionarán unos con otros y de esa manera se diseñarán aplicaciones y programas.   
  
Hoy en día es difícil programar sin utilizar programación orientada a objetos (Poo). Esta forma de programar es más cercana al modo en que expresaríamos las cosas en la vida real que la que se da en otros tipos de programación clásicos. Los analistas y programadores deben pensar las cosas de una manera distinta para plasmar dichos conceptos en programas en términos de objetos, atributos y métodos.

Tomemos Como ejemplo introductorio se puede tomar un burro.

El burro en mención (Pepito) es un objeto con características específicas. Claro está, en el establo pueden existir otros burros con diferente edad, color, peso y medidas.   
  
  
De esta manera se puede definir la CLASE Burro como una abstracción a todos los burros existentes en el establo; las características son las VARIABLES MIEMBRO O ATRIBUTOS DE LA CLASE y las acciones son los MÉTODOS MIEMBRO DE LA CLASE o COMPORTAMIENTO DE LA CLASE.  
  
Del anterior ejemplo podemos deducir la definición de clase y objeto. Una clase es una plantilla para crear objetos que constituye una abstracción del mundo real, como la clase Burro, la clase persona, la clase cuenta, etcétera. Las clases se nombran como sustantivos en singular y poseen variables que definen la información que se desea almacenar y métodos que definen las acciones que se desean realizar. Las variables se nombran como sustantivos y los métodos como verbos en infinitivo.

**Propiedades de la programación orientada a objetos**

La programación objetual define un conjunto de propiedades básicas que los lenguajes de programación orientados a objetos deben cumplir:   
• Abstracción.   
• Encapsulamiento.   
• Ocultamiento de información.   
• Sobrecarga.   
• Polimorfismo.   
• Herencia.   
• Reutilización.   
En el desarrollo de este capítulo abordaremos detalladamente estas propiedades. Adicionalmente, se introducirán otros conceptos relacionados con el área.

**Abstracción**   
  
La abstracción es la característica más básica de la POO y puede definirse como la acción de identificar las cualidades y acciones que un objeto puede realizar, lo que permite diferenciar los conceptos de clase y objeto.

**Encapsulamiento y ocultamiento de la información**

El encapsulamiento es la propiedad que tienen las clases de agrupar las características y las acciones relacionadas con una abstracción bajo una misma unidad de programación. Por su parte, el ocultamiento de la información protege los objetos restringiendo y controlando el acceso en la clase que los define.   
  
Esto permite al programador definir exactamente cuáles variables y métodos serán visibles para otras clases, asegurándose de que el estado interno de un objeto no pueda ser cambiado de forma inesperada por una entidad externa.

**Especificadores o modificadores de acceso**   
  
Los especificadores o modificadores condicionan el acceso de las variables, de los métodos de una clase, definiendo su nivel de ocultamiento. Los especificadores pueden ser:   
• (+) Público: El elemento puede ser accedido desde cualquier clase. Si es un dato miembro, cualquier clase puede acceder al elemento. Si es un método, cualquier clase puede invocarlo.   
• (-) Privado: Solo se puede acceder al elemento desde métodos miembros de la clase donde se encuentra definido, o solo puede invocarse desde otro método de la misma clase.   
• (#) Protegido: Proporciona acceso público para las clases derivadas (véase sección de herencia) y acceso privado para el resto de clases.   
• ( ) Sin modificador: Se puede acceder al elemento desde cualquier clase que esté en la misma ubicación (carpeta) donde se define la clase.

**Sobrecarga de métodos**

La sobrecarga permite a una misma clase tener varios métodos con el mismo nombre, inclusive con el mismo valor de retorno, pero con diferentes parámetros.

**Polimorfismo**

El concepto de polimorfismo se aplica a varios objetos que comparten un mé- todo con el mismo nombre. Cuando este método es invocado depende del objeto que lo contiene; por lo tanto tendrá un comportamiento diferente para cada instancia.