ENTORNOS DE DESARROLLO UD-3

UML

ORIENTACIÓN A OBJETOS

ÍNDICE

[INTRODUCCIÓN 3](#_Toc127731929)

[CONCEPTOS DE LA ORIENTACIÓN A OBJETOS 3](#_Toc127731930)

[VENTAJAS DE LA ORIENTACIÓN A OBJETOS 4](#_Toc127731931)

[CLASES, ATRIBUTOS, MÉTODOS 4](#_Toc127731932)

[VISIBILIDAD 5](#_Toc127731933)

[OBJETOS. INSTANCIACIÓN 5](#_Toc127731934)

# INTRODUCCIÓN

A la hora de crear software tenemos dos opciones:

* **Enfoque estructurado**. Se basa en la programación estructurada, basada en funciones que actuaban como estructuras de datos y que funcionaban en conjunto para resolver el problema de partida.
* **Enfoque orientado a objetos**. En este caso el proceso se centra en simular los elementos de la realidad asociada al problema de la forma más cercana posible. Esta abstracción recibe el nombre de objeto, los cuales se dice que son instancias de una clase y tienen 2 características:
  + **Atributos**. Son los datos que lo caracterizan.
  + **Operaciones**. Definen el comportamiento del objeto y se utilizan para modificar el estado (valor) de los atributos.

Cuando se utiliza la POO ocurren 3 sucesos:

1. Primero, los objetos se crean a medida que se necesitan.
2. Segundo, los mensajes se mueven de un objeto a otro (o del usuario a un objeto) a medida que el programa procesa información o responde a la entrada del usuario.
3. Tercero, cuando los objetos ya no se necesitan, se borran y se libera la memoria.

# CONCEPTOS DE LA ORIENTACIÓN A OBJETOS

* **Abstracción**. Permite capturar las características y comportamientos similares de un conjunto de objetos con el objetivo de darles una descripción formal. Es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a definir un conjunto de clases que permitan modelar la realidad, o el problema que se quiere atacar.

Por ejemplo, cualquiera de las clases que componen un programa.

* **Encapsulación**. Organiza los datos y métodos de una clase, evitando el acceso a datos por cualquier otro método distinto a los definidos. El estado de los objetos sólo puede ser modificado desde métodos de la propia clase.

Por ejemplo, los métodos getters y setters.

* **Modularidad**. Consiste dividir una aplicación en partes más pequeñas independientes entre sí y de la aplicación.

Por ejemplo, cualquiera de los métodos de una clase.

* **Principio de ocultación**. La implementación de una clase sólo es conocida por los responsables de su desarrollo. Gracias a la ocultación, esta podrá ser modificada para mejorar su algoritmo de implementación sin tener repercusión en el resto del programa.

Por ejemplo, los métodos de cada clase que sólo pueden ser llamados a través de los objetos de la clase.

* **Polimorfismo**. Consiste en reunir bajo el mismo nombre comportamientos diferentes. El modo en el que actuará un objeto en cada momento depende del mensaje que se le envíe al solicitar la ejecución de sus métodos.
* **Herencia**. Relación que se establece entre objetos en los que unos utilizan las propiedades y comportamientos de otros formando una jerarquía. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen.

Por ejemplo, una clase padre llamada Persona que tiene una serie de atributos como nombre, edad, altura… y unos métodos como saludar() o hablar() y, por otro lado, unas clases hijas llamadas Profesor y Alumnos que heredan de Persona y, por tanto, heredan los atributos y métodos de ésta.

* **Recolección de basura**. Técnica por la cual el programa se encarga de destruir automáticamente los objetos que hayan quedado sin referencia y, por tanto, desvincular su memoria asociada. Característica disponible en algunos lenguajes orientados a objetos, no en todos.

# VENTAJAS DE LA ORIENTACIÓN A OBJETOS

La POO tiene principalmente las siguientes ventajas:

* Desarrollos más rápidos y fáciles gracias a la reusabilidad del código.
* Aumento en la calidad de los sistemas.
* Mayor facilidad de modificación y mantenimiento.
* Permite hacer aplicaciones escalables.

# CLASES, ATRIBUTOS, MÉTODOS

Las clases tienen dos objetivos principales: definir abstracciones y facilitar la modularidad.

Estas clases están formadas por:

* **Nombre**: Identifica a cada clase del programa.
* **Atributos**: Que son las características de la clase. Estos atributos son las variables y constantes de la clase.
* **Protocolo**: Conjunto de métodos de una clase (incluido el constructor). Los métodos son las acciones que puede realizar la clase. Estos métodos pueden ser funciones (devuelven un valor) y procedimientos (no devuelven valores).

Se denominan **miembros** al conjunto de atributos y métodos de una clase.

# VISIBILIDAD

A la hora de diseñar una clase hay que definir dónde va a ser visible la clase, sus parámetros (atributos) y sus métodos.

Tenemos 3 tipos de visibilidad:

* **Público**: se puede acceder desde cualquier clase y cualquier parte del programa.
* **Privado**: sólo se puede acceder desde la propia clase.
* **Protegido**: se puede acceder desde la propia clase y de las clases que heredan de esta.

Como norma general, se siguen las siguientes reglas:

* Los **atributos** son privados. Para que nadie pueda acceder a ellos de forma directa sino a través de métodos de la clase creados para cumplir esa función.
* Los **métodos**:

1. Si definen la funcionalidad de la clase: son públicos. Para que otras clases puedan acceder a ellos y entonces poder acceder a los atributos.
2. Si ayudan a implementar parte de la funcionalidad: son privadas si no se utilizan desde clases derivadas o protegidas si se utilizan desde clases derivadas.

Hay que diferenciar entre las interfaces de acceso y los detalles de implementación:

* **Interfaz**. Es la cabecera de los métodos.
* **Implementación**. Es el desarrollo interno de los métodos.

# OBJETOS. INSTANCIACIÓN

Cuando se crea un objeto de una clase, se dice que estamos creando una instancia de una clase.

Un objeto se define por tres cosas:

* **Estado**: Definido como el conjunto de valores concretos que tienen sus parámetros.
* **Comportamiento**: Formado por los métodos del objeto.
* **Tiempo** **de vida**: Comienza a la hora de crearlo (new) hasta que se elimina con el fin de liberar memoria.

La encapsulación y el ocultamiento permiten que los atributos de un objeto no sean accedidos de forma accidental.

Existen un tipo de clases que no se pueden instanciar, son las clases abstractas.