

Tema 6. Programación Orientada a Objetos

6-1. Introducción a la POO

1. INTRODUCCIÓN

La Programación Orientada a Objetos (POO o también OOP del inglés Object Oriented Programming) es el paradigma de programación (modelo de programación) más utilizado en la actualidad para el desarrollo de software. Surge en los años 70 del siglo XX y fue en la década de los 90 donde se popularizó su uso.

Antes de la aparición del modelo de programación orientado a objetos se utilizaba el modelo imperativo, que es el modelo que hemos seguido durante el curso hasta ahora, en el que se definen por un lado los datos y por otro lado las operaciones que vamos a realizar con esos datos. Datos y operaciones son cosas independientes.

La idea principal de la POO es el concepto de objeto que se puede entender como una entidad formada por datos y las operaciones a realizar con esos datos. En este caso los datos y las operaciones no son independientes.

Para ver las diferencias entre la programación clásica y la POO, en esta introducción comenzaremos con un repaso a los conceptos de programación imperativa, estructurada y procedimental para seguir después con los conceptos básicos de la POO.

2. PROGRAMACIÓN IMPERATIVA, ESTRUCTURADA Y PROCEDIMENTAL

El paradigma de programación (modelo de programación) más antiguo es el **modelo imperativo**.

En este modelo los programas están formados por una serie de instrucciones que describen paso a paso y de forma precisa los pasos a seguir para resolver un problema.

Las instrucciones se ejecutan unas a continuación de otras en el orden en que han sido escritas.

En sus orígenes cuando había que romper el orden secuencial de ejecución de las instrucciones para realizar un bucle o una instrucción condicional, se utilizaban saltos (instrucción *goto*). Las instrucciones de salto hacían que los programas fuesen complicados de seguir y depurar.

```
x = 0
etiqueta1:
x = x + 1
print x
if x < 10
    goto etiqueta1
```

El paradigma de **programación estructurada** es un modelo de programación basado en el modelo imperativo que usa las estructuras de control: estructuras condicionales, bucles while, for, etc. Este modelo evita las instrucciones de salto (goto) con lo que se consigue un código más sencillo de leer y de menor complejidad.

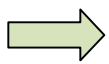
```
x = 0
do{
    x = x + 1
    print x
while(x < 10)
```

El paradigma de **programación procedimental**, también llamado programación **modular** o programación **funcional**, es un modelo de programación que utiliza el modelo de programación estructurada y se basa en dividir las tareas que debe hacer un programa en partes más pequeñas. Estas partes en las que se divide el trabajo a realizar se llaman procedimientos aunque también se les conocen como métodos, funciones, subrutinas, etc.

Un procedimiento (un método en lenguaje Java) contiene una serie de instrucciones que realizan una tarea muy concreta.

En este modelo los programas están formados por una serie de datos y una serie de métodos que se van llamando en un determinado orden para resolver el problema.

En el siguiente diagrama puedes ver la relación entre los paradigmas imperativo, estructurado y procedimental:



Todos los programas que hemos realizado hasta ahora durante el curso han seguido el modelo procedimental.

En la programación procedimental:

- Los datos y las acciones a realizar sobre ellos son cosas distintas.
- Se definen por un lado los datos y se definen por separado una serie de métodos o procedimientos que operan sobre ellas.

En el siguiente ejemplo se puede ver el estilo de los programas procedimentales:

```
public static void main(String[] args) {
```

```
String nombre="Arsenio", apellido1="Leante", apellido2="Gomez";
int edad = 30;
String nombre2;
```

Se crean los datos

```
nombre2 = nombreCompleto (nombre, apellido1, apellido2);
```

Los datos se envían a los métodos para que operen con ellos y devuelvan resultados

```
if (esMayorEdad (edad)){
    System.out.println (nombre2 + " es mayor de edad");
}else{
    System.out.println (nombre2 + " no es mayor de edad");
}
}
```

```
public static String nombreCompleto (String nom, String apel, String ape2){
    return nom + " " + apel + " " + ape2;
}
```

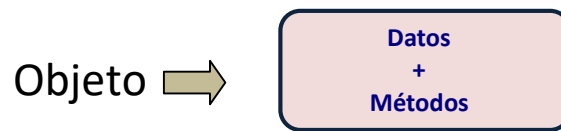
```
public static boolean esMayorEdad(int n){
    if (n>=18)
        return true;
    else
        return false;
}
```

3. PROGRAMACIÓN ORIENTADA A OBJETOS

En el paradigma de **programación orientada a objetos** los programas se organizan de forma similar a la organización de los objetos en el mundo real.

En POO cuando definimos los datos también definimos las acciones a realizar sobre ellos.

Podemos considerar un **objeto** como un conjunto de **datos + métodos**.



Los métodos definen las operaciones que se pueden realizar con los datos del objeto.

Podemos definir la programación orientada a objetos como un paradigma de programación (una forma de programar) en el que **el código se organiza en unidades llamadas clases de las que se crean objetos que interactúan entre ellos** para conseguir resolver el problema.

La principal diferencia entre el modelo procedimental y el orientado a objetos es que en el modelo procedimental los datos y las operaciones que se realizan sobre ellos se definen por separado. En la POO los datos llevan asociados una serie de operaciones a realizar.

El ejemplo anterior escrito ahora utilizando orientación a objetos en Java sería así:

1. Se declara la **clase Persona** que contiene los datos de la persona y las operaciones (métodos) a realizar con esos datos:

```
public class Persona {
    private String nombre;
    private String apellido1;
    private String apellido2;
    private int edad;

    public void inicializar(String n, String a1, String a2, int e){
        nombre = n;
        apellido1 = a1;
        apellido2 = a2;
        edad = e;
    }

    public String nombreCompleto() {
        return nombre + " " + apellido1 + " " + apellido2;
    }

    public boolean esMayorEdad() {
        if (edad >= 18){
            return true;
        }else{
            return false;
        }
    }
}
```

} **Datos o Atributos**

} **Métodos**

2. Se escribe el código para utilizar la clase:

```
public class EjemploPoo{
    public static void main(String[] args) {
        Persona p = new Persona();

        p.inicializar("Arsenio", "Leante", "Gomez", 30);

        String nombre2 = p.nombreCompleto();

        if(p.esMayorEdad()){
            System.out.println(nombre2 + " es mayor de edad");
        }else{
            System.out.println(nombre2 + " no es mayor de edad");
        }
    }
}
```


→ **Se crea un objeto Persona**

← **Se utilizan sus métodos**

En el ejemplo se ha creado un objeto llamado p de esta forma:

```
Persona p = new Persona();
```

En java cualquier variable u objeto que se crea debe tener un tipo. En este caso el tipo de p es *Persona*.

 p es un objeto de tipo Persona

Una vez creado el objeto podemos invocar a sus métodos.

```
p.inicializar("Arsenio", "Leante", "Gomez", 30);
```

Para llamar a un método se escribe **objeto.metodo()**, es decir, realizamos una operación sobre el objeto mediante un método.

En la actualidad existen una gran cantidad de lenguajes orientados a objetos. Algunos de ellos son:

C++	Delphi	PHP	Ruby	GOLANG
C#	Java	Python	Smalltalk	
VB.NET				

4. VENTAJAS DE LA POO FRENTE A LA PROGRAMACIÓN PROCEDIMENTAL

La programación orientada a objetos se impuso como el estilo de programación dominante como evolución de la programación procedimental clásica.

Mediante la programación procedimental se pueden alcanzar excelentes resultados, pero cuando se aplica a problemas complejos, los resultados son menos satisfactorios. Algunos de los problemas que presenta el modelo tradicional de desarrollo de sistemas de información surgen:

- Cuando la complejidad del sistema o su ambigüedad no permite un análisis inicial completo.
- Si el sistema desarrollado no se adapta a las necesidades de los usuarios, por falta de entendimiento inicial entre usuarios y diseñadores.
- Requiere excesivos recursos dedicados a su mantenimiento.
- Resulta difícil llevar a cabo su modificación por cambios del entorno o de los requisitos iniciales.
- Cuando necesita incorporar el tratamiento de nuevos tipos de datos como imágenes, sonido, vídeo, etc.

La POO toma las mejores ideas incorporadas a la programación procedimental y las combina con nuevos y potentes conceptos que permiten organizar los programas en una forma más efectiva.

Algunas ventajas sobre la programación tradicional son:

- Los problemas se pueden descomponer en partes más pequeñas. Esto facilita la resolución de problemas complejos.
- Los programas son más comprensibles. Mediante la POO se obtiene un código más legible que refleja de una forma clara la relación entre cada concepto a desarrollar y cada objeto que interviene en dicho desarrollo.
- Reutilización de código. La herencia, es una de las claves de la POO para reutilizar y reducir código.
- Los programas son más resistentes al cambio. La modificación de un método no afectará a los otros ni a la estructura de datos.
- Los programas son más fáciles de mantener y modificar.

5. CARACTERÍSTICAS DE LA POO

Las características principales que definen la programación orientado a objetos son:

- **Abstracción**
- **Encapsulamiento**
- **Herencia**
- **Polimorfismo**

ABSTRACCIÓN

La abstracción es la capacidad de extraer del mundo real las características (atributos y métodos) significativas que intervendrán en el programa.

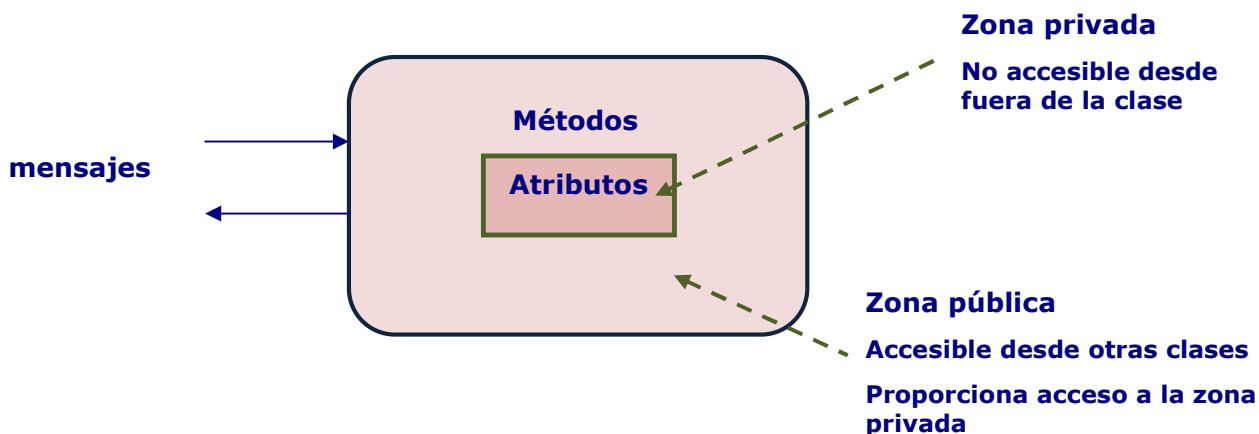
La clave de la POO está en abstraer los métodos y los datos comunes a un conjunto de objetos y almacenarlos en una clase.

Por ejemplo, para hacer una aplicación para el cálculo de nóminas la abstracción consiste en identificar todas las clases que van a intervenir y para cada clase especificar los atributos y métodos necesarios. En un programa de cálculo de nóminas una clase *Trabajador* podría tener como atributos el nombre o la fecha de nacimiento pero no es necesario que tenga atributos como color de pelo, raza, religión, etc.

ENCAPSULAMIENTO

Esta característica permite ver un objeto como una caja negra que contiene en su interior toda la información relacionada con dicho objeto.

Mediante el encapsulamiento se separan los aspectos externos del objeto (las partes a las que pueden acceder otros objetos o zona pública) de los detalles de implementación internos (ocultos a otros objetos o zona privada).



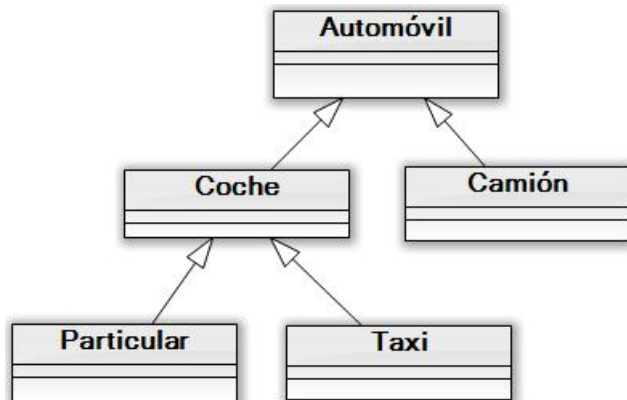
Los objetos se pueden manipular, pero su estructura interna permanecerá oculta a los demás. Sólo son visibles las partes necesarias para utilizar un objeto (interfaz pública). Las demás partes son ocultas (privadas). No es necesario saber cómo está hecho el objeto ni los datos que contiene, para poder utilizarlo.

HERENCIA

La herencia permite definir una clase a partir de otra ya existente. Una clase podrá heredar de otra un conjunto de propiedades (métodos y atributos) y a esa nueva clase se le puede añadir atributos o métodos, ocultar métodos heredados o redefinir métodos heredados.

Esta característica está muy relacionada con la **reutilización de código** en la POO.

Llamaremos **clases derivadas** a las que heredan de otra clase, a la que denominaremos **clase base**. A través de la herencia se crea una jerarquía de clases.



Automóvil es una Clase Base

Camión es una clase derivada de Automóvil
Coche es una clase derivada de Automóvil y además es la clase base de las clases Particular y Taxi

Particular y taxi son clases derivadas de Coche

POLIMORFISMO

Es la propiedad por la que una misma operación se realiza de forma distinta dependiendo de la clase a la que pertenece el objeto que la realiza.