Clases y Objetos

Programación Orientada a Objetos (CC2008) - 2026

Clases y Objetos

Semestre 02, 2025

POO

El mundo se percibe como una colección de objetos que interactúan para resolver problemas.

Bombilla

Atributos: consumo, estado (encendido/apagado).

Métodos: encender(), apagar().

Televisor

Atributos: tamaño, resolución, color.

Métodos: encender(), apagar(), cambiarCanal(), ajustarVolumen().

Los atributos se van a representar como variables a las que se les define un tipo de datos y un valor.

El comportamiento se va a representar como métidos a los que les definimos su tipo de retorno y parámetros.

Partes de un método

```
public double calcularArea(double base, double altura) {
    return base * altura;
}
```

- Modificador: `public`
- Tipo de retorno: `double`
- Nombre: `calcularArea`
- Parámetros: `base`, `altura`
- Cuerpo: instrucciones

Clases

Una clase es una plantilla o modelo que define las características y comportamientos comunes de un conjunto de objetos.

En ella se definen:

- Atributos (Estado)
- Métodos (Comportamiento)

Perro

```
class Perro {
    // atributos
    private String nombre;

    // metodos
    public void ladrar() {
        System.out.println("¡Guau!");
    }
```

```
}
```

Constructores

Un constructor es un método especial en una clase que se llama automáticamente cuando se crea una nueva instancia de esa clase.

Su función principal es inicializar los atributos del objeto recién creado.

Características

- Mismo nombre que la clase.
- No tienen tipo de retorno.
- Se llaman automáticamente.
- Se pueden sobrecargar.

```
public Perro(String nombre) {
    this.nombre = nombre;
}
```

Perro (final)

```
class Perro {
    // atributos
    private String nombre;

    // constructor
    public Perro(String nombre) {
        this.nombre = nombre;
    }

    // metodos
    public void ladrar() {
```

```
        System.out.println("¡Guau!");
    }
}
```

Objetos

Un objeto es una entidad que se genera en base a una clase. Se dice que es una instancia concreta de una clase.

Es la unidad básica en la programación orientada a objetos (POO).

Instanciar

Instanciar una clase es el proceso de crear un nuevo objeto en base a ella. Se utiliza la palabra reservada **new**

miPerro

```
Perro miPerro = new Perro("Scooby Doo");
miPerro.ladrar();
```

Clase vs Objeto

Clase: estructura común

```
CuentaBancaria
- saldo: double
```

Objeto: instancia concreta

```
cuentaJuan → saldo = Q500.00
cuentaMaria → saldo = Q1500.00
```

Encapsulamiento

- Encapsular = proteger el estado interno
- Agrupa datos y operaciones dentro de la clase
- Oculta la implementación interna y expone una interfaz pública clara.
- Seguridad
- Flexibilidad

Caja Negra

Modificadores de Visibilidad

- `public` : accesible desde cualquier parte
- `private` : solo accesible dentro de la clase
- `protected` : Más adelante cuando veamos herencia.

```
private String nombre;
public int contador = 0;
```

Los atributos suelen declararse como privados, y se accede a ellos mediante métodos públicos getters y setters.

CuentaBancaria

```
public class CuentaBancaria {
    private double saldo;
```

```
    // getters
    public double getSaldo() {
        return saldo;
    }

    // setters
    public void setSaldo(double cantidad) {
        saldo = cantidad;
    }
}
```

UML

Definición

- UML = Unified Modeling Language
- **Estándar**
- Herramienta para representar gráficamente:
  - Clases
  - Objetos
  - Relaciones

Fue creado en los años 90 por Grady Booch, Ivar Jacobson y James Rumbaugh como una unificación de varios métodos de modelado.

Ayuda a equipos de desarrollo a entender la estructura y el comportamiento de un sistema sin leer código.

Facilita la identificación de clases, atributos, métodos y relaciones antes de escribir una sola línea de código.

Al diseñar el sistema visualmente, es más fácil identificar problemas de arquitectura.

Diagrama de clase básico

```
+----------------------------+
| CuentaBancaria             |
+----------------------------+
| - saldo: double            |
| - numero: Int              |
+----------------------------+
| + depositar(monto): void   |
| + retirar(monto): void     |
+----------------------------+
```