
Programación orientada a objetos: Objetos

Tema 4



Contenidos

1. Introducción a la orientación a objetos
2. El modelo Orientado a Objetos
3. Creación y uso de clases y objetos en Java

Introducción a la orientación a Objetos

- ❑ La Programación orientada a objetos (POO) consiste en organizar y construir programas a partir de modelos que representan la interacción de los objetos del mundo real.
- ❑ Es una nueva forma de organizar el conocimiento.
- ❑ En POO lo importante es entender “que es” y “que hace” un objeto, más que el decidir “como” y en que lenguaje OO se implementará.

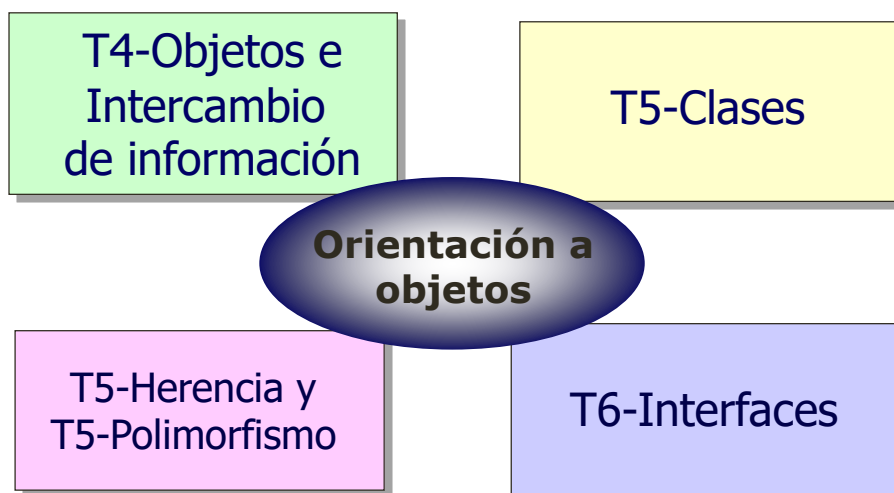
“La POO constituye una simulación de un modelo del mundo real”

Ventajas de la orientación a objetos

- ❑ Permite crear modelos más **próximos al mundo real**, más estables y preparados para los cambios.
- ❑ Facilita la **reutilización del código**.
- ❑ Facilita el **trabajo en equipo** y **mantenimiento** del software
- ❑ **Mejora la productividad y calidad** del software.

El Modelo orientado a objetos

Conceptos fundamentales



Objeto – Estructura y Función

- Identidad (¿Quién soy?) = Identificador único
- Estado (¿Cómo soy?) = **Atributos**
- Comportamiento (¿Qué se hacer?) = **Métodos**
- Procedencia (¿Cuál es mi origen?) = **Clase**
- Comunicación (¿Qué entiendo?) = **Mensajes**

Objeto - Ejemplo



Objeto - Ejemplos

❑ Modelo físico



❑ Modelo informático



**Clase
Coche**

Atributos

marca
modelo
potencia
combustible
velocidad
máximaVelocidad
aceleración

Métodos

pararMotor()
arrancarMotor()
Frenar()
cambiarMarcha()
acelerar()
girar()

❑ Modelo físico



❑ Modelo informático



**Clase
Bombilla**

Atributos

tipo
consumo
potencia
duración

Métodos

encender()
apagar()
brillar()
atenuar()

Clase - Definición

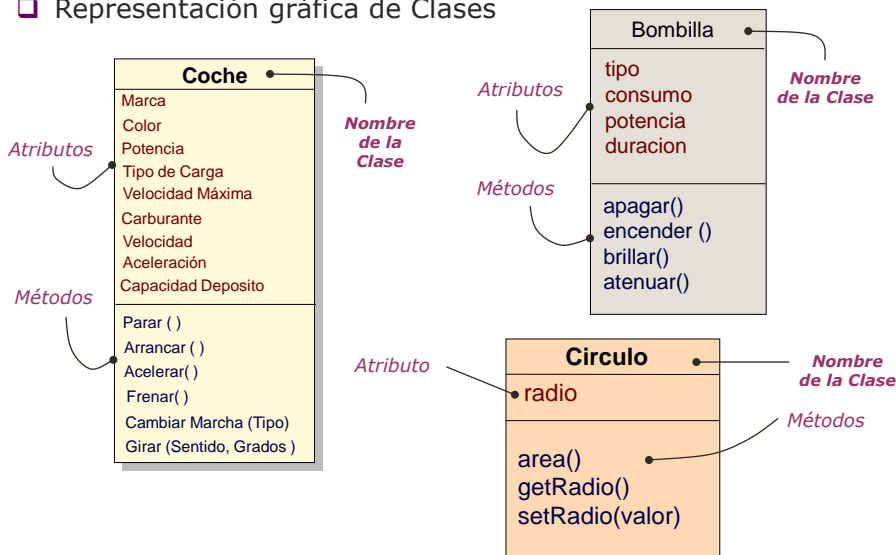
- ❑ Una **Clase** describe un conjunto de objetos que comparten:
 - ✓ mismas propiedades: **Atributos**
 - ✓ mismo comportamiento: **Métodos**
 - ✓ mismas relaciones con otros objetos: **Intercambio de información**
- ❑ En un programa, una **Clase** es un componente de software que actúa como una **plantilla** para fabricar tipos particulares de objetos que tienen los mismos atributos y métodos.
- ❑ Los **Objetos** creados a partir de una Clase se llaman **instancias** de esa Clase, son la plantilla con los atributos rellenos.

Objetos de la
Clase Coche



Clase - Ejemplo

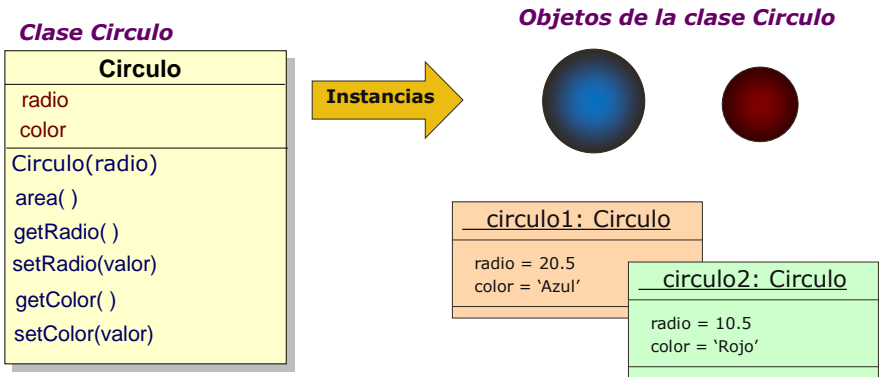
Representación gráfica de Clases



Clases y Objetos

Ejemplo:

La instanciación de la clase Circulo creará objetos circulo con valores específicos para sus atributos



Todos los objetos (*instancias*) de la clase *Circulo* tienen los mismos métodos y los mismos atributos, pero con diferentes valores

Tipo de Visibilidad

- Toda Clase *encapsula* sus *atributos* y *métodos*, con ciertos criterios de visibilidad y manipulación respecto a otras Clases.

| Tipo visibilidad | Descripción |
|---|---|
| + Público (public) | Puede ser usados por cualquier otra clase |
| - Privado (private) | Pueden ser usados solo por la Clase que los define |
| # Protegido (protected) | Pueden ser usados por: <ul style="list-style-type: none">- la Clase que los define y por las subclases derivadas- ó clases de su mismo package |

IMPORTANTE:

- **atributos** los definiremos como **private** y
- **métodos** como **public-protected** para
 - Acceder/devolver su valor: `get()` y
 - modificar su valor: `set()`

Creación y uso de clases y objetos en Java

Definición de una clase Java

```
public class Nombreclase{

    // Atributos o variables
    Tipo_visibilidad    tipoDatos    atributo1;
    Tipo_visibilidad    tipoDatos    atributo2;
    ...

    // Métodos
    tipoVisibilidad    tipoDatosDevuelto
        nombre_método( tipoDatos    parámetro,...){
        .....
    }
    .....
}


```

Ejemplo de Definición de una clase Java

□ Dada la siguiente definición de clase en Java:


```
public class Circulo{
    private double    radio;        // atributo privado radio del circulo
    public String    color;        // atributo público color del circulo

    public double    getRadio(){    // método que devuelve el radio
        return radio;
    }

    public void    setRadio(double valor){ //método que asigna valor al radio
        radio = valor;
    }

    public double    area(){        // método que calcula el área
        return    (Math.PI * radio * radio);
    }
}


```



- La definición de la clase se guarda en un fichero de texto llamado **Circulo.java** (Crear dentro del proyecto Netbeans, **File\New file\Java\Java Class** sin main())

Creación de objetos en Java

- Para crear un objeto de la clase `Circulo` se usa la siguiente instrucción:

`Circulo c1 = new Circulo();`

Método Constructor: crea la instancia de la clase.
Se llama igual que la clase

Variable de referencia
a un objeto circulo

El operador `new` crea una nuevo **objeto** `c1` (variable de referencia a un objeto) de la **clase** `Circulo`, llamando al **método constructor** `Circulo()`.

Acceso a información del objeto

- Para acceder al contenido de un objeto (sus atributos y métodos) se usa el **punto** `.`

```
String color = c1.color;  
Double radio = c1.getRadio();  
c1.setRadio(25.5);
```

A través del **punto** `.` tendremos acceso según el tipo de visibilidad de cada elemento (`public`, `private`, `protected`)

En este caso el atributo `color` sería `public`

Ejemplo: Uso de objetos en Java

```
public Class Principal{  
    public static void main( String args[ ] ) {  
        // Creamos dos círculos c1 y c2  
        Circulo c1 = new Circulo();  
        Circulo c2 = new Circulo();  
        c1.setRadio(4); //Modificamos el radio de c1 a 4  
        c2.setRadio(10.5); //Modificamos el radio de c2 a 10.5  
  
        // Calculamos y mostramos el área de los círculos c1 y c2;  
        System.out.println("Area Circulo 1: " + c1.area() );  
        System.out.println("Area Circulo 2: " + c2.area() );  
        // Modificamos el radio del círculo c1 y obtenemos el área;  
        c1.setRadio(7.0);  
        System.out.println("Area Circulo 1: " + c1.area() );  
    }  
}
```

- La definición de la clase se guarda en un fichero de texto llamado **Principal.java** (Crear dentro del mismo proyecto Netbeans, **File\New file\Java\Java Main Class**)