



Programación Orientada a Objetos

Autor: Ing. Ernesto Gigliotti.



Comencemos con lo que sabemos:

C

Biblioteca "Persona"

```
struct S_Persona
{
    char nombre[20];
    int edad;
};
typedef struct S_Persona
Persona;
```

**Quiero validar la carga de los
campos**

**Escribo una función que
setee el campo y **chequee** el
valor a cargar**

Función:

```
void per_setEdad(Persona* p, int edad)
{
    if(edad<=99)
        p->edad=edad;
}
```

Uso:

```
Persona p;
```

```
per_setEdad(&p, 56);
```

Función:

```
void per_setNombre(Persona* p, char* nombre)
{
    strncpy(p->nombre, nombre, 20);
}
```

Uso:

```
Persona p;
```

```
per_setNombre(&p, "juan");
```

Quiero imprimir los datos

Función:

```
void per_printPersona(Persona* p)
{
    printf("%s edad:%d\r\n", p->nombre, p->edad);
}
```

Uso:

```
Persona p;
```

```
per_setEdad(&p, 56);
per_setNombre(&p, "juan");
```

```
per_printPersona(&p);
```

Clase

- Modelo (o plantilla) para crear objetos de ese **tipo**.
- Describe el estado y el comportamiento que todos los objetos de la clase comparten.
- Formada por **atributos** y **métodos**

```
public class Persona
{
    int edad;
    String nombre;

    public void metodo1 ()
    {

    }
}
```

Objeto

- La instancia de una clase.
- En tiempo de ejecución realiza las tareas de un programa.
- Es capaz de recibir mensajes, procesar datos y enviar mensajes a otros objetos.

```
Persona juan = new Persona();
```

El operador new reserva memoria para el objeto creado

- Si hacemos:

```
Persona juan2 = juan;
```

Ambos objetos apuntarán al mismo espacio de memoria.

Lenguaje C

```
struct S_Persona
{
    char nombre[20];
    int edad;
};
typedef struct S_Persona Persona;
```

Modo de uso:

```
Persona p;

p.edad = 18;
```

CAMPOS



Lenguaje Java

```
public class Persona
{
    int edad;
    String nombre;
}
```

Modo de uso:

```
Persona p = new Persona();

p.edad = 18;
```

ATRIBUTOS

Lenguaje C

```
struct S_Persona
{
    char nombre[20];
    int edad;
};
typedef struct S_Persona Persona;
```

Modo de uso:

```
Persona p;

p.edad = 18;
```

CAMPOS



Lenguaje Python

```
class Persona:
    nombre=""
    edad=0
```

Modo de uso:

```
p = Persona()

p.edad = 18
```

ATRIBUTOS

Lenguaje C

```
struct S_Persona
{
    char nombre[20];
    int edad;
};
typedef struct S_Persona Persona;
```

Modo de uso:

```
Persona p;
p.edad = 18;
```

CAMPOS



Lenguaje C#

```
public class Persona
{
    public int edad;
    public String nombre;
}
```

Modo de uso:

```
Persona p = new Persona();
p.edad = 18;
```

ATRIBUTOS

Funciones de nuestra biblioteca

Siempre le pasamos la variable con la que trabajamos dentro de la función

```
void per_setNombre(Persona* p, char* nombre)
{
    strncpy(p->nombre, nombre, 20);
}
```

```
void per_setEdad(Persona* p, int edad)
{
    if(edad<=99)
        p->edad=edad;
}
```

```
void per_printPersona(Persona* p)
{
    printf("%s edad:%d\r\n", p->nombre, p->edad);
}
```


**Analizemos el uso
de las funciones**

Siempre le pasamos la variable con la que trabajamos dentro de la función

```
Persona p;
```

```
per_setEdad(&p, 56);
```

```
per_setNombre(&p, "juan");
```

```
per_printPersona(&p);
```

**Esto nos da a entender que
la **función** esta fuertemente
ligada a un **contexto de valores**
brindado por la **variable** "Persona"
que pasamos como argumento**

```
Persona p;  
  
per_setEdad(&p, 56);  
per_setNombre(&p, "juan")  
;  
  
per_printPersona(&p);
```

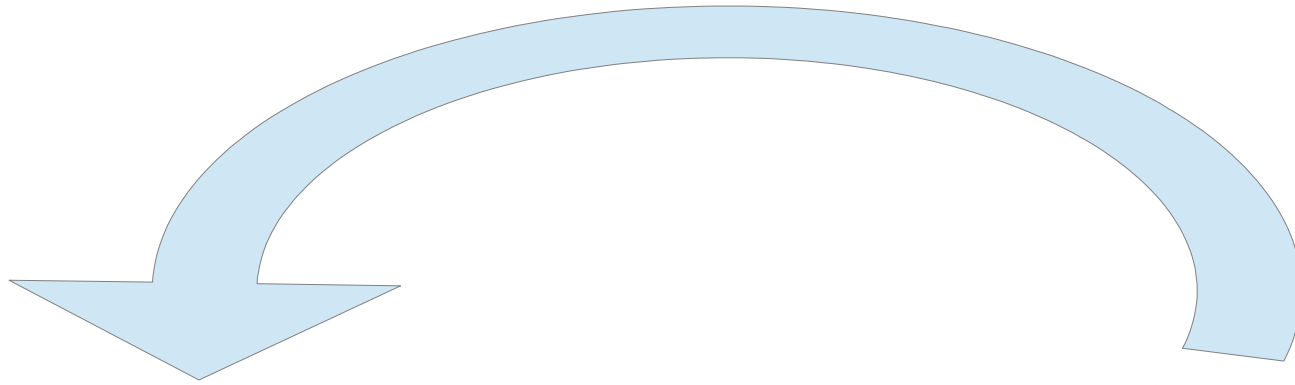
```
> juan edad:56
```

```
Persona p2;  
  
per_setEdad(&p2, 18);  
per_setNombre(&p2, "Pedro");  
  
per_printPersona(&p2);
```

```
> Pedro edad:18
```

Según la variable pasada como argumento, el resultado de las funciones será diferente

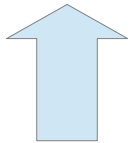
**En Programación Orientada
a Objetos, se realiza una relación
más profunda entre la variable (objeto)
y las funciones que la necesitan
(métodos)**



`p.per_setEdad(&p, 56);`

Al usar objetos, dejamos de pasar como argumento el “objeto”

```
p.per_setEdad(56);  
p.per_setNombre("juan");  
p.per_printPersona();
```



**Los métodos se ejecutan en el
contexto del objeto**

```
p.setEdad(56);  
p.setNombre("juan");  
p.printPersona();
```

**Ya no es necesario el prefijo en el
nombre de la función**

Las funciones se definen dentro de la clase y se llaman “métodos**”**

Lenguaje C

```
void per_setNombre(Persona* p, char* nombre)
{
    strncpy(p->nombre, nombre, 20);
}

void per_setEdad(Persona* p, int edad)
{
    if(edad<=99)
        p->edad=edad;
}

void per_printPersona(Persona* p)
{
    printf("%s edad:%d\r\n", p->nombre, p->edad);
}
```

FUNCIONES



MÉTODOS

Lenguaje Java

p.setEdad(56);

```
class Persona {  
    String nombre;  
    int edad;  
  
    public void setNombre(String nombre)  
    {  
        this.nombre = nombre;  
    }  
  
    public void setEdad(int edad)  
    {  
        this.edad = edad;  
    }  
  
    public void printPersona()  
    {  
        System.out.println(this.nombre+" edad:"+this.edad);  
    }  
}
```

**En Java no
recibimos el obj
como primer
argumento,
pero existe como
"this"**

Lenguaje Python

p.setEdad(56)

```
class Persona:  
    nombre=""  
    edad=0
```

```
def setNombre(self, nombre):  
    self.nombre=nombre
```

```
def setEdad(self, edad):  
    self.edad=edad
```

```
def printPersona(self):  
    print(self.nombre + " edad:" + str(self.edad))
```

En Python
seguimos
recibiendo el obj
como primer
argumento
(Aunque no lo
pasamos al
llamar al método)

FUNCIONES



MÉTODOS

Lenguaje C#

p.setEdad(56);

```
class Persona
{
    String nombre;
    int edad;

    public void setNombre(String nombre)
    {
        this.nombre = nombre;
    }

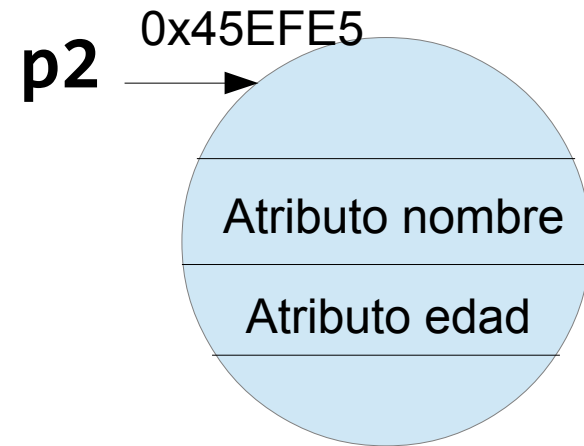
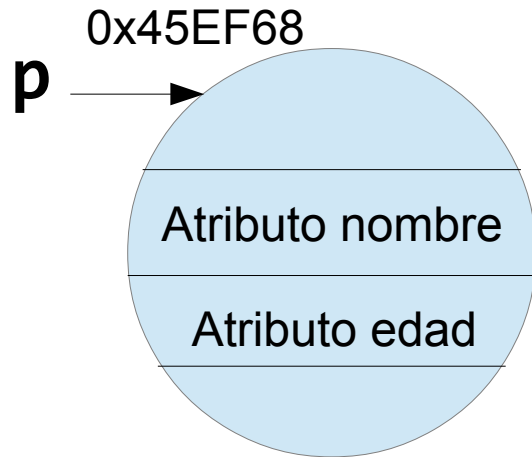
    public void setEdad(int edad)
    {
        this.edad = edad;
    }

    public void printPersona()
    {
        Console.WriteLine(this.nombre+" edad:"+this.edad);
    }
}
```

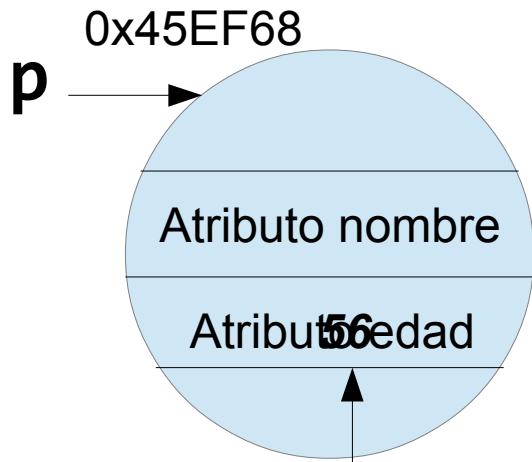
**En C# no
recibimos el obj
como primer
argumento,
pero existe como
"this"**

Ejemplo Java y C#

```
Persona p = new Persona(); Persona p2 = new Persona();
```



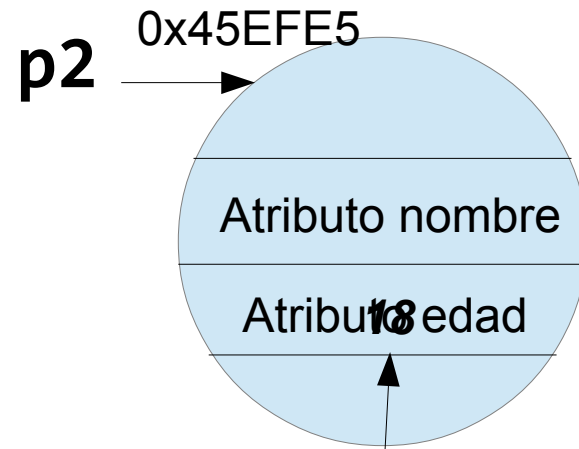
Java y C#



`p.setEdad(56);`

```
class Persona
{
    String nombre;
    int edad;

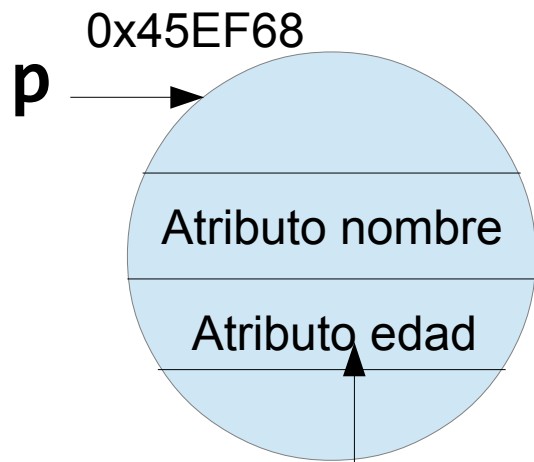
    public void setEdad(int edad)
    {
        this.edad = edad;
    }
}
```



`p2.setEdad(18);`

```
class Persona
{
    String nombre;
    int edad;

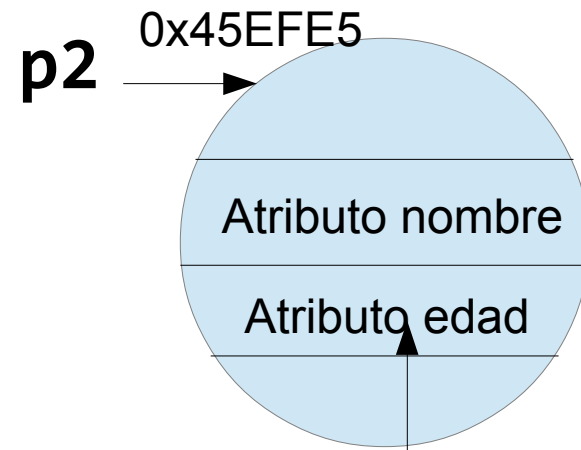
    public void setEdad(int edad)
    {
        this.edad = edad;
    }
}
```



`p.setEdad(56);`

```
class Persona:  
    nombre=""  
    edad=0  
  
def setEdad(self, edad):  
    self.edad=edad
```

Python



`p2.setEdad(18);`

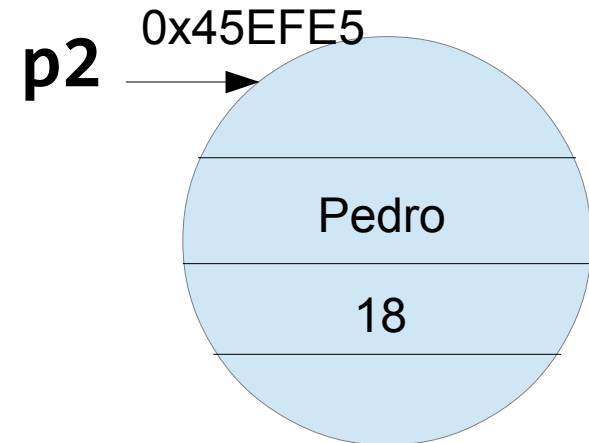
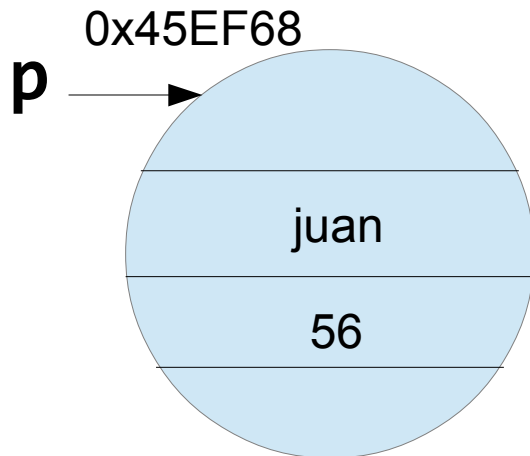
```
class Persona:  
    nombre=""  
    edad=0  
  
def setEdad(self, edad):  
    self.edad=edad
```

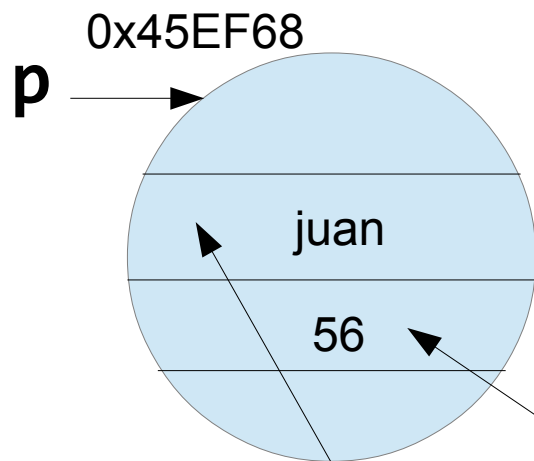



Ejemplo **Python**: imprimir

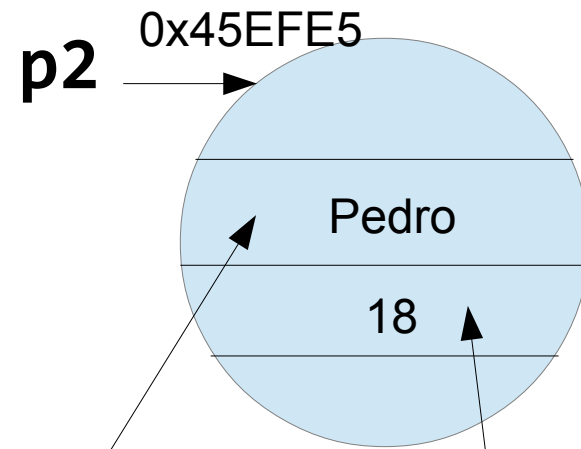
```
p = Persona()  
p.setNombre("juan")  
p.setEdad(56)
```

```
p2 = Persona()  
p.setNombre("Pedro")  
p.setEdad(18)
```





Python



`p.printPersona();`

`p2.printPersona();`

```
class Persona:  
    nombre=""  
    edad=0
```

```
def printPersona(self):
```

```
    print(self.nombre + " edad:"+str(self.edad))
```

> **juan edad:56**


> **Pedro edad:18**

Constructores

Java y C#: `Persona p = new Persona();`



Python: `p = Persona()`



- Es un método más en la clase
- No devuelve ningún valor
- Se ejecuta luego de construir el objeto en memoria
- No es necesario definirlo
- Puede recibir argumentos:
 - Inicialización de atributos al generar el objeto

Constructores

Python:

```
class Persona:  
    nombre=""  
    edad=0  
  
    def __init__(self):  
        self.nombre = ""  
        self.edad = 0
```

Java y C#:

```
class Persona {  
    String nombre;  
    int edad;  
  
    public Persona() {  
        this.nombre = "";  
        this.edad = 0;  
    }  
}
```

Atributos estáticos

- Son atributos de la clase, no de cada objeto
- No se necesita un objeto para usarlos

```
class Persona:
```

```
    a=0
```

```
    def __init__(self):  
        self.nombre = ""  
        self.edad = 0
```

Modo de uso:

```
Persona.a = 5
```

```
class Persona {  
    String nombre;  
    int edad;  
    static int a;  
  
    public Persona() {  
        this.nombre = "";  
        this.edad = 0;  
    }  
}
```

Modo de uso:

```
Persona.a = 5;
```

Atributos estáticos C#

```
class Persona
{
    String nombre;
    int edad;
    public static int a;

    public Persona() {
        this.nombre = "";
        this.edad = 0;
    }
}
```

Modo de uso:

```
Persona.a = 5;
```

Métodos estáticos

- No se ejecutan en el contexto de un objeto
- No se necesita un objeto para usarlos
- Por eso no pueden usar self/this

`class` Persona:

```
@staticmethod  
def saludo():  
    print("Hola")
```

`class` Persona {

```
public static void saludo()  
{  
    System.out.println("Hola");  
}  
}
```

Modo de uso:

Persona.saludo()

Modo de uso:

Persona.saludo();

Métodos estáticos C#

```
class Persona {  
  
    public static void saludo()  
    {  
        Console.WriteLine("Hola");  
    }  
}
```

Modo de uso:

```
Persona.saludo();
```


¡Muchas gracias!

<https://repl.it>



Este obra está bajo una licencia de Creative Commons
Reconocimiento-CompartirIgual 4.0 Internacional.