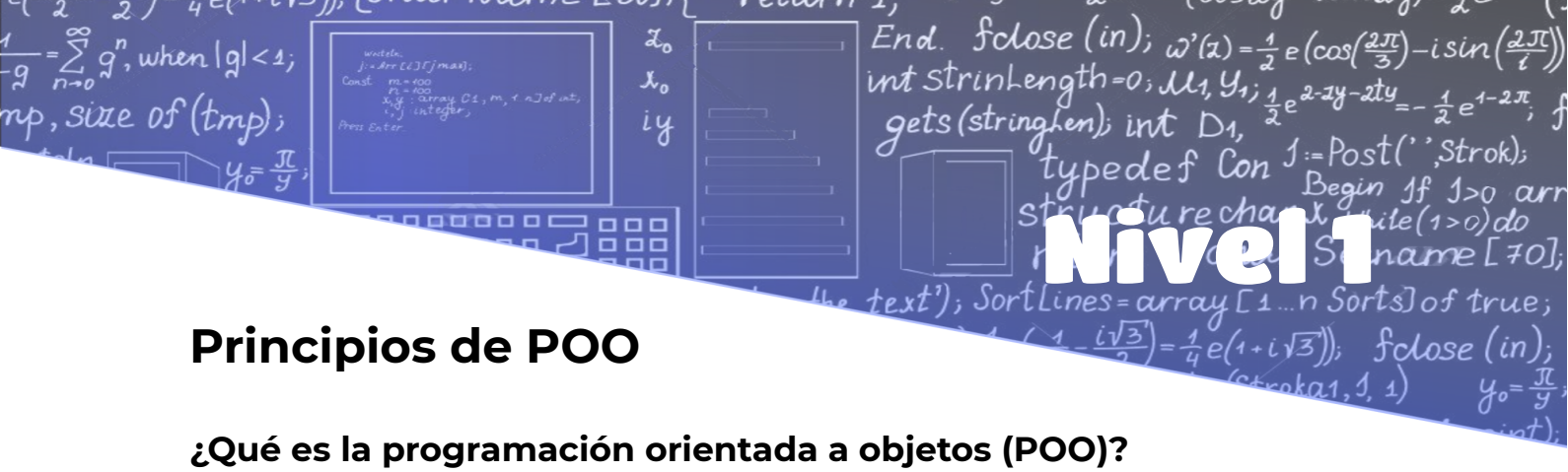



# Principios de POO

**¿Qué es la programación orientada a objetos (POO)?**

Es uno de los paradigmas de la programación que permite que el código sea reutilizable, organizado y fácil de mantener.

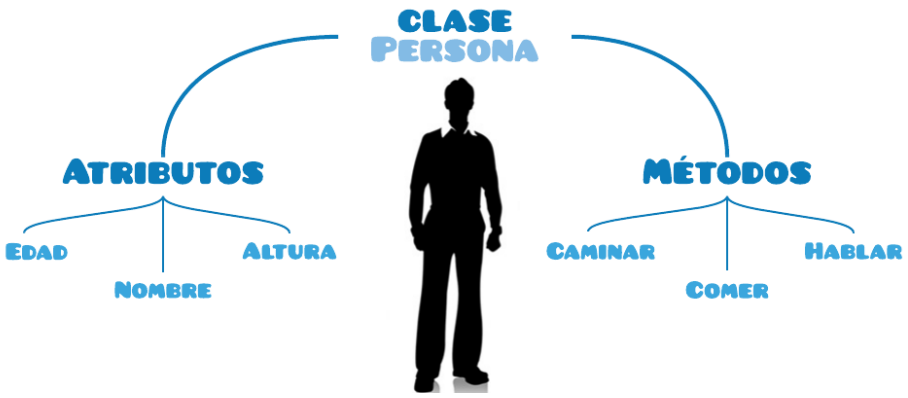
Con el paradigma de **Programación Orientado a Objetos** lo que buscamos es dejar de centrarnos en la lógica pura de los programas, para empezar a pensar en objetos y sus relaciones [1].

Sus elementos principales son las clases y objetos, muchas veces se crea confusión al diferenciarlos por lo que primero veremos su definición.

## Clase

Las clases se pueden entender como plantillas a partir de las cuales se pueden crear objetos.

Cuando pensamos en el término “persona” se nos viene a la mente la idea general de lo que es una persona, un ser humano que tiene una edad, nombre, altura, etc.[2] Esa es la clase, mientras que las características/variables ligadas a la clase se conocen como **Atributos.**

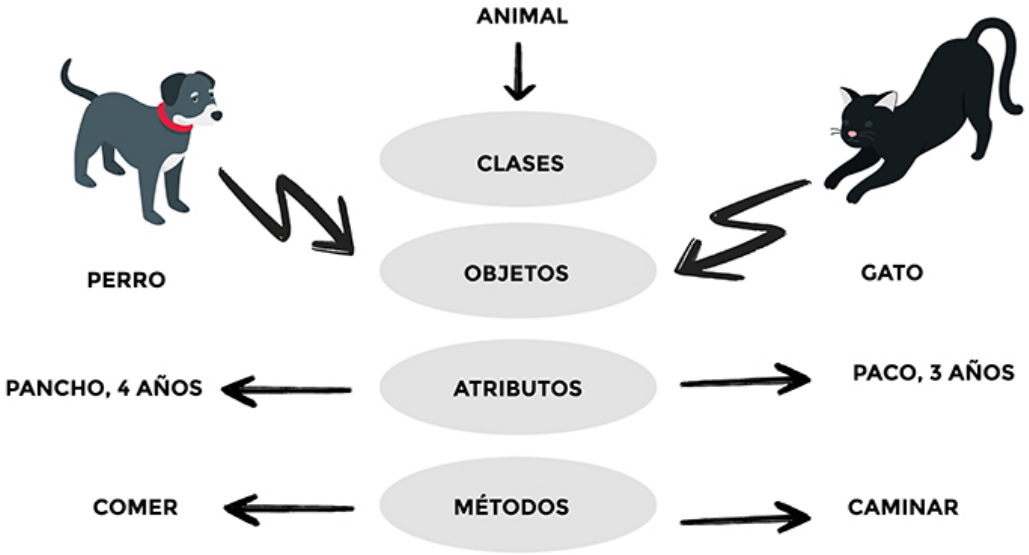



Otro de los conceptos principales de la clase son los **métodos** o funciones que es capaz de realizar la clase, siguiendo con la analogía de la clase ‘persona’ sus métodos pueden ser caminar, hablar, comer, etc.

Así como existen métodos únicos para cada clase, existe un método universal llamado **constructor,** es el encargado de instanciar el objeto dentro de la clase, otra forma de verlo es como la función que se encarga de asignarle un valor a los atributos de la clase a su vez que le indica al programa que el objeto pertenece a esa clase.

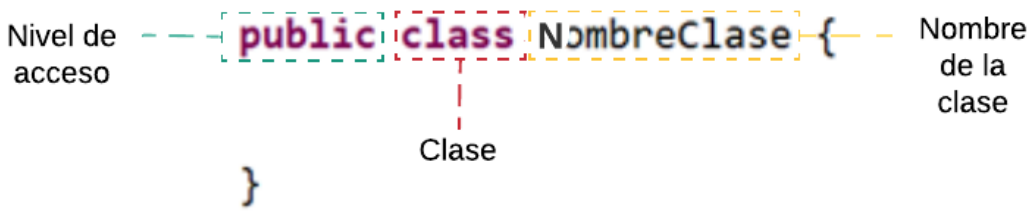
## Objeto

"Un objeto(también llamado **instancia**) es creado a partir de una clase(el molde, boceto...). El proceso de creación de este objeto se denomina **instanciación**".[3]
En otras palabras, un objeto se crea cuando a una variable le asignó la instancia de una clase, en el proceso de instancia se definen los valores específicos de los atributos de la clase.

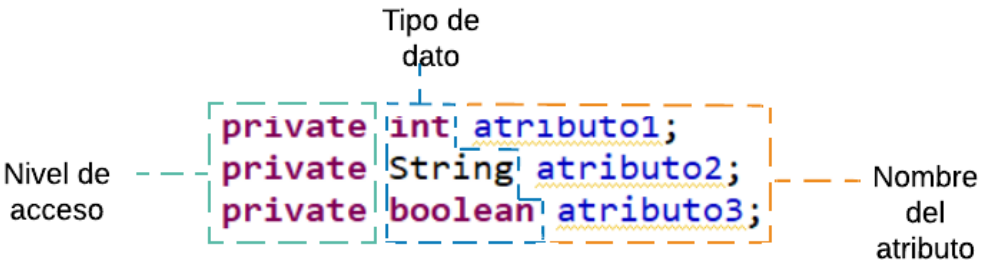



## Aplicación en Java

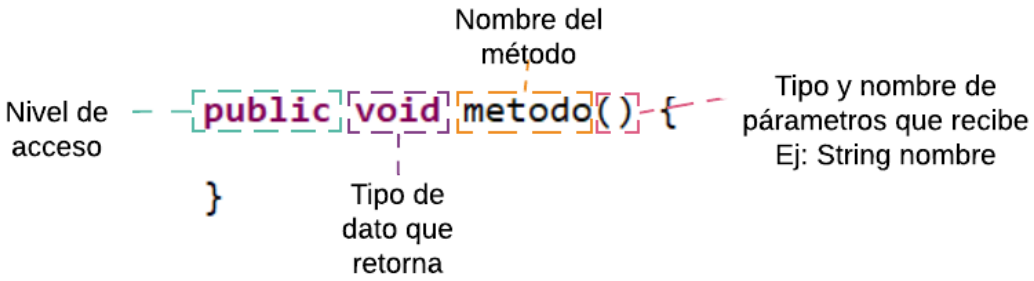
- **Definir una clase:** Para definir una clase en java se debe indicar el nivel de acceso de la clase *(Ver anexo 1)*.



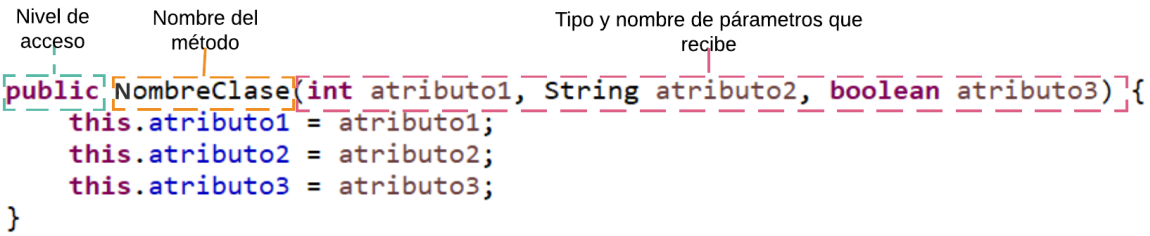
- **Definición de atributos:** deben ser inicializados indicando su tipo de dato *(anexo 2)* y su nivel de acceso *(anexo 1)*.




- **Definición de métodos:** para instanciar un método se definen tres factores, el nivel de protección, el tipo de dato que retorna la función (en caso de que no retorna nada se usa void).



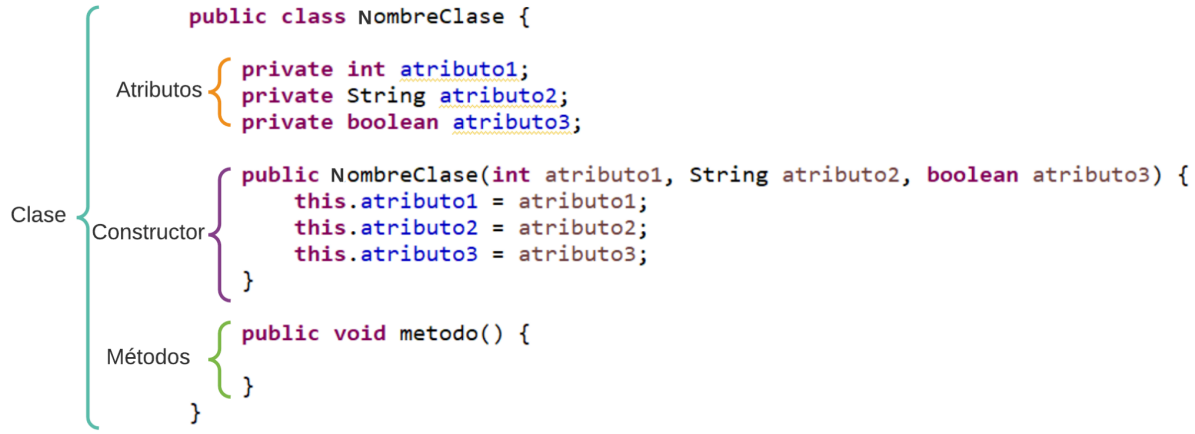

- ***Definición de un constructor:*** cuando defino un constructor se debe tener en cuenta que debe ser público para que se puedan instanciar objetos y su nombre debe ser el mismo que el de la clase.




**Nota:** cuando usamos 'this.atributo1' hacemos referencia a los atributos de la clase mientras que 'atributo1' hace referencia al valor que se recibe como parámetro.

**Estructura general de una clase**

Recopilando las definiciones anteriores, la estructura de una clase luciría así:




## Método *main*

Antes de ver la forma de instanciar un objeto se debe conocer el método *main*, *este* es el punto de partida del programa; únicamente el código escrito en él se ve reflejado al correr el programa.
Por tanto si queremos instanciar un objeto debemos tener una clase que contenga el método main.

**Nota**
La clase que contenga el método *main puede tener cualquier nombre*

```
public class Main {

    public static void main(String[] args) {

    }

}
```

Por defecto el método main se define de esa manera

- **Definición de un Objeto:**

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

        NombreClase nombreObjeto = new NombreClase(1,"Objeto", true);

    }
}
```

Tipo de dato — Nombre del Objeto — Llamado del constructor

## Ejemplo

Para entender mejor los conceptos vistos tenemos el siguiente

proyecto compuesto por dos clases:

**-Clase Persona**

Esta contiene la definición de su clase junto con sus atributos, constructor y métodos.

```
public class Persona {
    private String nombre;
    private int edad;
    private double estatura;
    private String estado;

    public Persona(String nombre, int edad, double estatura) {
        this.nombre = nombre;
        this.edad = edad;
        this.estatura = estatura;
    }

    public void comer() {
        this.estado = "comiendo";
    }

    public void hablar() {
        this.estado = "hablando";
    }

    public void correr() {
        this.estado = "corriendo";
    }
}
```

**-Clase Main**

Se crea el objeto 'Juan' llamando el constructor de la clase 'Persona' y luego se llama el método comer con el que le estamos indicando al programa que el atributo 'estado' = "comiendo".

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

        Persona juan = new Persona("Juan Rodriguez", 19, 1.72);

        juan.comer();
    }
}
```

Nombre Edad Estatura

Llamado del método 'comer'

## Anexos

*Anexo 1. Niveles de acceso en Java[4]*

| MODIFICADOR | CLASE | PACKAGE | SUBCLASE | TODOS |
|---|---|---|---|---|
| **public** | Sí | Sí | Sí | Sí |
| **protected** | Sí | Sí | Sí | No |
| **No especificado** | Sí | Sí | No | No |
| **private** | Sí | No | No | No |

*Anexo 2.Tipos de datos primitivos en Java[5]*

| Nombre | Tipo | Byte | Rango de valores |
|---|---|---|---|
| boolean | Lógico | 1 | true ó false |
| char | Carácter simple | 2 | Un caracter |
| byte | Entero | 1 | -128 a 127 |
| short | Entero | 2 | -32768 a 32767 |
| int | Entero | 4 | -2.147.483.648 a 2.147.483.649 |
| long | Entero | 8 | -9.223.372.036.854.775.808 a 9.223.372.036.854.775.808 |
| float | Numérico con coma flotante | 4 | $34x10^{-38}$ a $34x10^{38}$ |
| double | Numérico con coma flotante | 8 | $1.8x10^{-308}$ a $1.8x10^{308}$ |

## Referencias


1. Styde Limited. (2015). Styde.net; Styde.net. https://styde.net/que-es-la-programacion-orientada-a-objetos/

2. *Clases y Objetos*. (2011). Programación Orientada Por Objetos En Ruby. https://makeitrealcamp.gitbook.io/programacion-orientada-por-objetos-en-ruby/clases-y-objetos

3. Image.pdf. (2021). *Image.pdf*. Google Docs. https://drive.google.com/file/d/1ChoJPc0Yhpq_QZhLSwjntqhum46WJ07Z/view

4. Rodríguez, A. (2021). *public, private y protected Java.Tipos de modificadores de acceso. Visibilidad en clases, subclases. (CU00693B)*. Aprenderaprogramar.com. https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=665:public-private-y-protected-javatipos-de-modificadores-de-acceso-visibilidad-en-clases-subclases-cu00693b&catid=68&Itemid=188

5. Diego Giraldo Giraldo. (2016). Tipos de datos primitivos [YouTube Video]. In *YouTube*. https://www.youtube.com/watch?v=VrOIAzlT7Os

6. *Clase String: Representando una cadena*. (2021, July 4). Manual Web. http://www.manualweb.net/java/clase-string-representando-una-cadena/