

# Introducción al paradigma de objetos

## Objetivos

---

- Comprender que es un Paradigma.
- Comprender que es la POO (Programación Orientada a Objetos).
- Conocer la importancia de los objetos.
- Conocer las clases y métodos.

## Contenidos:

Paradigma. Diferencias entre Programación Estructurada y Programación Orientada a Objetos. Objetos y Clases. Atributos de clases. Comportamiento (métodos). Parámetros, Argumentos y valor de retorno en métodos. Identidad de un objeto. Constructores. Uso del this. Creación de objetos.

## ¿Qué es un paradigma de programación?

Comenzamos un nuevo curso y lo primero que nos proponemos es explicarte qué es el paradigma de la programación orientada a objetos. En esta unidad veremos los conceptos básicos de este enfoque de creación de software, cuáles son sus características y las ventajas con respecto a la programación estructurada

En el módulo Técnicas de Programación te presentamos a la programación estructurada, en donde se definen funciones y procedimientos. En este módulo nos dedicaremos al estudio de la **Programación Orientada a Objetos (POO)**.

**Cambiamos entonces de paradigma.... ¿Y esto qué significa? ¿Nos servirá lo que aprendimos de técnicas? ¡Por supuesto!**

Nos servirán muchas cosas como, por ejemplo, las estructuras de datos y las estructuras de control. Seguiremos utilizando las definiciones de variables, tipos de datos estándares, arreglos, matrices, pilas, colas y listas. También utilizaremos los condicionales y los ciclos. Es decir, que seguiremos utilizando muchas de las herramientas que vimos en Técnicas de programación, pero ahora adaptadas a una nueva forma de crear analizar, codificar e imaginar la programación de un sistema.

Comencemos, entonces, definiendo lo que es un paradigma de programación....

SIGUIENTE>

## Definición de Paradigma de programación

Podemos definir a un paradigma de programación como un enfoque particular o criterio para la creación de software.

Hay diferentes paradigmas de programación, que determinan distintos estilos de programación y diferentes formas de resolver un problema.

Para mostrar más claro los diferentes enfoques de los paradigmas, te presentamos un mismo ejemplo abordado desde ambos paradigmas.

Supongamos que tenemos que ingresar por teclado el nombre, apellido y la edad de un alumno. El sistema no solicitará más datos si se ingresa una edad igual a cero.

## PARADIGMA ESTRUCTURADO Botón Desplegable

SEUDOCÓDIGO A MODO DE EJEMPLO:

INICIO PROGRAMA

VARIABLES

NOMBRE : CADENA  
APELLIDO : CADENA  
EDAD : ENTERO

PROCEDIMIENTO ObtenerPersona ()

Mostrar ("Ingrese nombre")  
Ingresar (NOMBRE)  
Mostrar ("Ingrese apellido")  
Ingresar (APELLIDO)  
Mostrar ("Ingrese edad")  
Ingresar (EDAD)  
Hasta que EDAD > 0

FINPROCEDIMIENTO

FINPROGRAMA

En nuestro ejemplo anterior de programación en **versión estructurada**, se realiza la definición de los datos, por medio de variables. Luego, se define un procedimiento (en nuestro caso ObtenerPersona) y éste se encarga de recibir los datos del alumno para incorporar al programa.

Es decir, que se definen los datos (variables: nombre, apellido y edad) y su comportamiento (recibir los datos por pantalla) de manera separada. Si alguna definición de datos cambia, deberemos cambiar las funciones o procedimientos definidos en el programa.

## PARADIGMA ORIENTADO A OBJETOS Botón Desplegable

SEUDOCÓDIGO A MODO DE EJEMPLO:

INICIO Principal()

// Definimos el inicio del programa

```

    AlumnoNuevo = nuevo Alumno()                                // Llamamos a la clase Alumno, generando
                                                                // el objeto AlumnoNuevo
    AlumnoNuevo.obtener()                                       // Utilizamos el método obtener() de
Alumno

FIN Principal

CLASE Alumno                                                    // Creamos la clase alumno

    Atributos                                                    // Definimos sus atributos o propiedades
        NOMBRE : CADENA                                          // Son tres Nombre, Apellido y Edad
        APELLIDO: CADENA
        EDAD : ENTERO

    METODO Obtener ( )                                           // Dentro de la clase creamos
        Mostrar ("Ingrese nombre")                               // el método Obtener
        Ingresar (NOMBRE)
        Mostrar ("Ingrese apellido")
        Ingresar (APELLIDO)
        Mostrar ("Ingrese edad")
        Ingresar (EDAD)
        Hasta que EDAD > 0
    FIN METODO

FIN CLASE

```

En cambio, en la versión de Objetos, habrás notado que hay algo nuevo.

Hay una clase (Alumno) que contiene atributos (Nombre, Apellido y Edad) y métodos (Obtener()).

Es decir que, en este mismo componente (en la clase (Alumno)), vamos a tener los datos y la forma en que se manipulan esos datos mediante el método (Obtener()).

Luego, en el programa principal, se crea un objeto (AlumnoNuevo) en base a una clase (Alumno).

También habrás notado que no hay llamadas a funciones, sino que hay referencias a los métodos definidos en la clase Alumno utilizando el objeto AlumnoNuevo.obtener(). Es decir que creamos un Objeto a partir de una clase y luego usamos el objeto para llamar a los métodos.

En el mismo objeto se encuentran los datos y sus comportamientos, de tal manera que, ante un cambio en los datos, no se deberá cambiar el programa principal, sino que los cambios deberán realizarse solamente en la clase.

SIGUIENTE>

# El paradigma de la Programación Orientada a Objetos

En programación un paradigma, es un estilo, una forma, un modo, una práctica y hasta una visión del desarrollo de un sistema, cuando nos referimos a objetos y clases.

Aunque esta definición teórica no contribuya mucho a la realización de código, nunca olvides que un Paradigma no deja de marcar y ser una guía de los pasos a seguir en cualquier acción a realizar.

Entonces, la programación orientada a **objetos** es un enfoque de programación que tiene un estilo en el cual cada programa es pensado y escrito como un objeto y además, se forma por una serie o un conjunto de componentes, que también son **objetos** y que con sus datos (**atributos o propiedades**) y con la posibilidad de realizar acciones (**métodos**) cooperan y se relacionan para lograr el funcionamiento de la aplicación completa.

A continuación, te presentamos un gráfico que compara la **programación estructurada** y la **programación orientada a objetos**:



SIGUIENTE>

## ¿Cuáles son las diferencias de la POO con el paradigma estructurado?

La principal diferencia con la programación estructurada tradicional es que ahora deberás pensar simultáneamente en el conjunto de atributos que tendrá una clase y en los métodos para poder tratarlos. Es decir, que los datos son tan importantes como su comportamiento y deben ir íntimamente entrelazados.

En cambio, en el paradigma estructurado, los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida.

SIGUIENTE>

## Y entonces, ¿que son los Objetos ?

Antes de ver la definición de objeto, te pedimos que te tomes un momento para observar los objetos y seres vivos que hay a tu alrededor, por ejemplo, una mesa, una silla, tu perro, cualquier elemento.

Podrás notar que a cada uno de ellos se le puede asociar un conjunto de características, por ejemplo: color, peso, dimensiones, raza, etc.

En programación, el concepto de **objeto** agrupa cosas y seres vivos, para lo cual podemos indicar que cada objeto podrá tener un comportamiento propio generado por él mismo o inducidos por otros objetos: el perro ladra, camina, salta, etc. Así, cada objeto tendrá sus propias características y comportamiento.

Un **Objeto** es un elemento que posee características, denominadas atributos, y comportamientos, denominados métodos, que puede interactuar con otros objetos del sistema, enviando mensajes, y tiene asociado una identidad que lo distingue entre todos los objetos del sistema.

Mas técnicamente hablando:

- **Objeto:** es la entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos).

### EJEMPLO

“Mi perro es de color marrón” ← ese es el dato

“Mi perro puede ladrar” ← ese es el método

En muchos casos, corresponden a los objetos reales del mundo que nos rodea, y también a objetos internos del sistema (del programa). Esta entidad **objeto** tiene como “estructura” a una **clase**.

### EJEMPLO

“Un caniche es de la clase perro” ← Caniche es el objeto, Perro es la clase.

El **objeto** inicia su participación activa en el sistema cuando es llamada la **clase** que contiene su “estructura” por medio de una instancia. Este proceso, en realidad, constituye técnicamente una “instancia de una clase” y explicaremos el significado de “instancia” en las siguientes páginas, después de conocer bien el concepto de clase.

## EJEMPLO

Para crear un “Caniche” necesito la clase “Perro” ← genera la instancia de la clase Perro y crea al

objeto Caniche

Podríamos dejar como resumen de estos párrafos que un objeto es un ejemplo, un derivado o una muestra de una clase, que tiene todas o algunas de las características de la clase que representa y que puede o no interactuar con otros objetos.

## Clases

### ¿Y cual es la relación entre las clases y los objetos?

Las Clases son definiciones de las propiedades y comportamientos que podrá tener un tipo de **objeto**. Instanciar una clase, realiza la lectura de estas definiciones y la creación de un **objeto** a partir de ellas.

Sigamos con los casos de los objetos de la vida real y el ejemplo de un perro.

Nosotros sabemos qué es un perro porque conocemos la idea de un perro. Todos los perros presentan las características y tienen el mismo comportamiento. Ahora bien, cada perro en particular se diferencia porque tiene diferentes atributos, por ejemplo, son de distinta raza, de distinto tamaño, de distinto peso, etc. Por lo tanto, la idea de un perro es el equivalente a una Clase.

Podemos ver a **una clase** como un modelo o plantilla que representa entidades o conceptos.

En una clase se definen los datos y el comportamiento que tendrá la entidad que representa. Los datos se denominan atributos, mientras que el comportamiento está definido por funcionalidades o comportamientos que se denominan métodos. Una clase se utiliza para crear objetos, es por eso que también se dice que un objeto es una instancia de una clase.

## EJEMPLO

La clase perro, puede indicar como atributo: pelo, color, altura, peso etc. que son todas propiedades pertenecientes a los perros.

También puede indicar si ladra, corre, come, duerme etc y esos son los métodos que pertenecen a esa clase.

Una clase es un modelo abstracto que define cómo será un objeto real.

Y en programación JAVA, como se escribe una clase? Probemos ahora con el ejemplo de un Automóvil.

### **Ejemplo Práctico** Botón Desplegable

```
public class Auto {                                // este es un ejemplo de como escribir la clase Auto

    String marca = "Ford";                        // comienza con una lista de atributos o
propiedades
    String modelo = "Falcón";
    int año = 1975;
    int puertas = 4;

    public Auto() {                                // posee un método constructor que lo hace
arrancar
        arrancar();
    }

    public String arrancar(){                      // el método arrancar avisa cuando
        return "en marcha"; // el auto está "en marcha"
    }
}
```

En el código, definimos la clase auto, con sus atributos marca, modelo, año etc y un método Auto() que invoca a otro método que lo hacer arrancar(). Por ahora, este detalle no es tan real como debería y no estamos haciendo un código funcional, solamente el ejemplo muestra como sería la forma de la escritura de la clase.

Tenemos la clase y el objeto ¿donde está?. En el ejemplo anterior no generamos un nuevo objeto y por ende el objeto no está codificado, eso lo veremos en las siguientes páginas.

SIGUIENTE>

### Diseño de clases

Como te contamos antes, una clase es un modelo, un esqueleto, una estructura para generar objetos. La clase está compuesta por atributos (datos, variables), también llamadas variables de instancia, que nos indican en qué estado se encuentra cada objeto, y métodos (funcionalidades) que indican cuales comportamientos posee el objeto.

Las **clases** habitualmente se denotan con nombres generales y globales como Animal, Arbol, Socio, Barco etc.

Toda clase está compuesta por **atributos y métodos**. Los atributos son características de los objetos. Cuando defines un atributo tendrás que especificar su nombre y su tipo.

SIGUIENTE>

## Atributos y Métodos

### Atributos

Cuando definimos atributos dentro de una clase, éstos toman el nombre técnico de **atributos de instancia**, porque cuando se crea un objeto en memoria de la computadora, éste objeto tendrá una copia tanto de los atributos como también de sus métodos de la clase inicial.

Repasemos:

En JAVA, un atributo o propiedad de una clase, se define igual que una variable y para ello necesita del tipo de dato que tendrá, por ejemplo:

**EJEMPLO** Código JAVA:

```
String nombre;  
String apellido;  
int    edad;
```

estos atributos se crearán dentro de una clase a la cual pertenecerán y al momento de crear una instancia de esta clase (llamarla para interactuar con ella), generará un objeto con los mismos atributos. Para ello los atributos se inicializan dentro de la clase:

**EJEMPLO** Código JAVA:

```
personas Medicos = new personas(); // instancia de la clase personas para crear al objeto  
Medicos
```

```
public class personas {           // Clase personas que será instanciada para crear un  
objeto.  
  
String nombre = "Juan";          // Atributos nombre, apellido, edad que pasarán a ser  
parte  
String apellido = "Perez";       // del objeto  
int    edad = 23;  
  
}
```

¿No entiendo, y donde se escribe una clase? ¿Cuando tengo que escribirla y como se relaciona con el objeto?

La clase que vimos en la página anterior, es un ejemplo sencillo y sin ninguna funcionalidad real, de como se escribe una clase Auto.

Como ese código representa a la clase Auto en lenguaje JAVA, la escritura debe realizarse en NetBeans y para ello debemos generar previamente un proyecto.

Te invitamos a ver el siguiente video que muestra paso a paso, como crear el proyecto, la clase y el objeto de esa clase para que puedas ir incorporando el concepto práctico de la



escritura en la POO. El video tiene un lenguaje sencillo y algunos términos, no son los que estrictamente deban ser utilizados. Hemos dado mayor importancia a que se entienda lo explicado que a la técnica teórica.

Link al Video: <https://youtu.be/QVcZ8nFzyXY>

Video del Prof. Daniel Balbi. Docente del plan Codo a Codo.

## Comportamiento (métodos)

Con los métodos estamos definiendo el comportamiento que tendrá y/o podrá realizar un objeto.

**En nuestro ejemplo siguiente, los métodos son 2: MiNombre y MiEdad**

Cuando se crea un objeto (tomando como base la clase a la cual pertenece) se genera una copia de los métodos de la clase, en cada objeto creado. A esto se lo denomina método de instancia.

Es importante que sepas que junto con el nombre del método, la declaración lleva información del tipo de devolución de los datos que posee el método, el número y el tipo de los parámetros necesarios, y qué otras clases y objetos pueden llamar al método.

Los métodos pueden poseer argumentos (parámetros) o no. En caso de no tenerlos solo se escriben los paréntesis vacíos, en caso de tenerlos se define el conjunto de argumentos de cualquier método en una lista de declaraciones de variables delimitadas por comas donde cada declaración de variable se realiza indicando el tipo de dato y el nombre, por ejemplo: *String Nombre*.

## **EJEMPLO** Código JAVA:

Veamos entonces como escribir la creación del objeto “Medicos” mediante la instancia a la clase “personas”:

### **Archivo medicos.java**

```
public class medicos {  
    public static void main(String[] args) {  
        personas Medicos = new personas();           // creo el objeto Medicos en base a  
personas  
        Medicos.MiNombre();                           // Llamo al método MiNombre en Medicos  
        Medicos.MiEdad();                             // Llamo al método MiEdad en Medicos  
    }  
}
```

Como podrás notar, los métodos los llamamos desde Medicos con la escritura de Medicos.MiNombre() o Medicos.MiEdad() ya que como anteriormente creamos el objetos Medicos con todos los atributos y métodos de la clase personas.

Ahora veremos como está conformada la clase “personas” para albergar los atributos (Nombre, Apellido y Edad) y los métodos MiNombre y MiEdad que emiten por pantalla los datos de los atributos.

### **EJEMPLO** Código JAVA:

#### **Archivo personas.java**

```
public class personas {  
  
    String Nombre = "Juan";           // creamos los atributos de la clase  
    String Apellido = "Perez";        // en este caso Nombre, Apellido y Edad  
    int edad = 48;  
  
    public void MiNombre() {          // método que muestra por pantalla el Nombre  
        System.out.println("Mi nombre es: " + Apellido + ", " + Nombre);  
    }  
  
    public void MiEdad(){             // método que muestra por pantalla la Edad  
        System.out.println("Y tengo " + edad + " años");  
    }  
}
```

Para ver más sobre creación de métodos te invitamos a ver el siguiente video que muestra paso a paso, como crear el proyecto, la clase y el objeto de esa clase para que puedas ir incorporando el concepto práctico de la escritura en la POO. El video tiene un lenguaje sencillo y algunos términos, no son los que estrictamente deban ser utilizados. Hemos dado mayor importancia a que se entienda lo explicado que a la técnica teórica.

Link al Video: <https://youtu.be/MMl8GlaVVuI>

Video gentileza Prof. Daniel Balbi. Docente del plan Codo a Codo.

### Entonces, Que es un método?

Un método es el código definido para realizar una acción que se incluye dentro de una clase y puede ser parte de un Objeto cuando ese Objeto haya sido creado mediante la instancia a la clase, en nuestro ejemplo veíamos esta instancia y creación del objeto en la línea de código:

### **EJEMPLO** Código JAVA:

```
personas Medicos = new personas();
```

en donde el objeto Medicos se crea a partir de la instancia de la clase personas.


## Parámetros, Argumentos y Valor de Retorno en Métodos.

Aunque es bastante habitual encontrar confusiones con respecto a los conceptos de parámetro y argumento, hay que significar que son, conceptualmente hablando, los opuestos en la tarea del traslado de la información de un método a otro.

### EJEMPLO Código JAVA:

En el caso del método que calcula los números primos, sería :

```
public static boolean primo(int n){  
    for (i = n; i>1; i--){  
        if ( n%i == 0)  
            return false;  
    }  
    return true;  
}
```

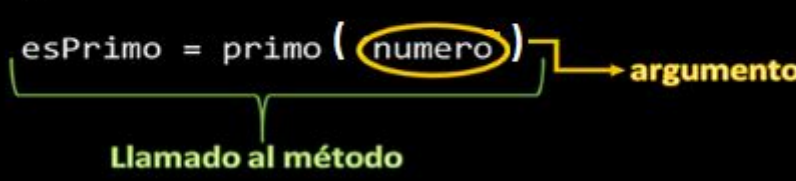


Podemos definir entonces que el **parámetro** es el **valor** que **recibe** un **método** para su procesamiento.

En cambio, un argumento:

### EJEMPLO Código JAVA:

```
public static void main(String[] args){  
    int numero;  
    boolean esPrimo;  
    // Se lee el numero del usuario  
  
    esPrimo = primo ( numero )  
    }  
}
```



Sería el “opuesto” a la recepción de un dato como es el parámetro, ya que el **argumento** es el **valor** que se **envía** a un **método** para ser procesado.

En muchos casos, un objeto se “comunica” con sus métodos o con los métodos de otros objetos mediante el envío de argumentos que el método recibe como parámetro, esta comunicación que es “unilateral”, ya que se envía un dato que es recibido por un método, tiene también su opción “bidireccional”, cuando el método puede retornar valores y modificar el estado (valor de sus atributos) de otro objeto o simplemente de una variable.

Veamos un ejemplo, en donde

1. crearemos un objeto calculadora,
2. en base a una clase calculo,
3. que tendrá un método que suma dos valores
4. que le enviaremos como argumentos
5. y recibirá como parámetros.
6. El resultado, nos llegará como un retorno de valor del método
7. y los guardaremos en una variable.

**EJEMPLO** Código JAVA:

**Archivo sumas.java**

```
public static void main(String[] args) {  
    calculo Calculadora = new calculo();           //1. Crearemos un objeto calculadora,  
                                                    //2,en base a una clase calculo,  
  
    double resultado;                             //creamos la variable para guardar el resultado  
  
    resultado = Calculadora.sumar(50.38 , 80.56); //3. método que suma dos valores  
  
    System.out.println(resultado);                // mostramos el resultado  
}
```

**EJEMPLO** Código JAVA:

**Archivo calculo.java**

```
public class calculo {                               // clase calculo,
```

```

public double sumar(double valor1,double valor2) { //3. un método que suma dos valores
                                                    //y recibe como parámetros.
    return(valor1+valor2);                          //6. retorno de valor del método
                                                    }
}

```

Repasemos el código, nos encontramos primero con la creación de un objeto de la clase calculo que llama al método sumar pasándole como argumentos los valores a sumar, en este caso los números a sumar son: 50.38 y 80.56 . El separador decimal es el punto.

Como podrás observar los argumentos se separan con una coma en el caso que sean varios.

Por otro lado, nos encontramos en la clase con el método **sumar** que recibe ambos parámetros, define con double después del public, **que retornará un valor del tipo doble.:**

```

public double sumar(double valor1,double valor2){

```

y entre los paréntesis, indica que recibirá dos parámetros, también del tipo double con los nombres valor1 y valor2, los que serán procesados y devueltos al objeto que los llamó con la orden:

```

    return(valor1+valor2);

```

Y como obtengo el valor de la suma?

En la primer porción de código, cuando se llama al método sumar con:

```

resultado = Calculadora.sumar(50.38 , 80.56);

```

se está solicitando que el valor retornado por el cálculo sea guardado en la variable resultado.

Cuando un método no enviará ninguna respuesta, debe contener la palabra **void** indicando esta situación:

**EJEMPLO** Código JAVA:

```

public void MiNombre() {
    System.out.println("Mi nombre es: " + Apellido + ", " + Nombre);
}

```

## Identidad de un Objeto

### Identidad

La identificación o identidad es la propiedad que permite diferenciar a un objeto y distinguirse de otros. Generalmente esta propiedad es tal, que da nombre al objeto. Tomemos por ejemplo el "verde" como un objeto concreto de una clase color; la propiedad que da identidad única a este objeto es precisamente su "color" verde. Tanto es así que para

nosotros no tiene sentido usar otro nombre para el objeto que no sea el valor de la propiedad que lo identifica.

**EJEMPLO** Código JAVA:

```
color verde = new color();
```

En programación la identidad de todos los objetos sirve para comparar si dos objetos son iguales o no.

La identidad tiene su fundamento en el soporte que da un tipo de clase a su utilización para la creación de varios objetos, con lo cual, así como hemos creado el objeto verde, utilizando la misma clase “color” podemos crear otro objeto con la identidad “rojo”, y aunque ambos objetos tendrán inicialmente los mismos atributos y métodos, la identidad es lo que separa los datos y ejecución de métodos en memoria:

**EJEMPLO** Código JAVA:

```
color rojo = new color();
```

## Constructores

Un Constructor es un método de las clases, el cual es llamado automáticamente cuando se crea un objeto a partir de esa clase.

Por ser métodos, los constructores también aceptan parámetros. Cuando en una clase no especificamos ningún tipo de constructor, el compilador añade uno de uso público por omisión sin parámetros, el cual NO hace nada.

**EJEMPLO** Código JAVA:

```
public class Auto {  
    String marca = "Ford";  
    String modelo = "Falcón";  
    int año = 1975;  
    int puertas = 4;  
  
    public Auto() { // Podemos ver en esta línea la creación del constructor  
                    // de la clase Auto.  
    }  
}
```

Características de los Constructores

1. Un constructor, tiene el mismo nombre de la clase a la cual pertenece.
2. No retorna ningún valor, por lo cual no debe especificarse ningún tipo de dato.
3. Debe declararse como public, aunque en algunos casos puede ser de otro tipo.
4. Si la clase tiene algún constructor, el constructor por defecto deja de existir. En ese caso, si queremos que haya un constructor sin parámetros tendremos que declararlo explícitamente.

Su función es inicializar el objeto y sirve para asegurarnos que los objetos siempre contengan valores válidos.

## Uso del This

Hay ocasiones en las que resulta útil asegurarnos que nos estamos refiriendo al objeto desde el que se está ejecutando un método.

Para ello se implementó en JAVA el uso de la palabra reservada “this”, que en estas ocasiones se puede usar la referencia especial desde el objeto actual.

Esta referencia se suele usar para pasar una referencia al objeto actual como un parámetro que fue recibido para nuestros métodos.

Es así como podemos realizar una nueva versión de la clase calculo, que utilizamos en la sección de parámetros y argumentos:

### **EJEMPLO** Código JAVA:

```
public class calculo {  
    double numero1;  
    double numero2;  
    public double sumar(double numero1,double numero2) {  
        this.numero1 = numero1;    // uso de this referenciando a numero1  
        this.numero2 = numero2;    // uso de this referenciando a numero1  
        return(this.numero1+this.numero2); // respuesta con this.  
    }  
}
```

Como podemos notar en el código anterior se crearon dos nuevos atributos (numero1 y numero2) que se utilizan dentro del método sumar para hacer una copia del valor que traen los parámetros al método y después con ese mismo this retornamos el valor de la suma.

Sin la utilización de este modificador this, el acceso a los atributos del método actual puede generar inconvenientes si tenemos asignados el mismo nombre a un parámetro del método.

En el ejemplo, tanto los parámetros como los atributos se llaman igual, se diferencian en el método por el uso del this, que obliga al método a tomar el valor creado por él y no el que proviene de los parámetros:

```

public class calculo {
    double numero1;
    double numero2;

    public double sumar(double numero1, double numero2) {
        this.numero1 = numero1;
        this.numero2 = numero2;

        return (this.numero1 + this.numero2);
    }
}

```

Accede al atributo por el uso de this

Lee el valor del parámetro.

Netbeans nos ayuda a leer el código, asignando el mismo color a **verde** a los atributos que se referencian con el **this** y el color **negro** a los parámetros que provienen del método.

## Creación de Objetos

En el ejemplo anterior de métodos, realizábamos la creación de un objeto con la palabra clave **New**, la sentencia era:

**EJEMPLO** Código JAVA:

```
personas Medicos = new personas();
```

Podemos leer la sentencia del ejemplo como:

*“Voy a crear un objeto Medicos del tipo o de la clase personas instanciando su método constructor personas()”*

Esta creación de un objeto, se logra instanciando la clase **personas** con la llamada a su constructor en conjunto con **new**, lo que hace que la orden de creación sea **new personas()**.

**Medicos** está representando una nueva identidad para el objeto, o sea, le asignamos un nombre y también indicamos que será del tipo “**personas**” con la sentencia **personas Medicos**.

Cuando se crea un objeto de la clase **Personas**, su variable de instancia llamada **Medicos** se inicializa de manera predeterminada con **null**, si es que la clase ya no tiene asignación de datos a los atributos.

Pero ¿qué pasa si queremos proporcionar un nombre a la hora de crear un objeto **Medicos**?

En cada clase que declaramos, se puede proporcionar de manera opcional un constructor con parámetros que pueden utilizarse para inicializar un objeto de una clase al momento de crear ese objeto.



El siguiente ejemplo mejora la clase Persona con un constructor que puede recibir un nombre y usarlo para inicializar las variables de instancia nombre, apellido y edad al momento de crear un objeto Medico:

### **EJEMPLO** Código JAVA:

Veamos entonces como escribir la creación del objeto “Medicos” mediante la instancia a la clase “personas” con un constructor que recibe tres parámetros:

#### **Archivo medicos.java**

```
public class medicos {  
    public static void main(String[] args) {  
        personas Medicos = new personas("Juan","Perez",38);// Clase con argumentos.  
        Medicos.MiNombre();  
        Medicos.MiEdad();  
    }  
}
```

### **EJEMPLO** Código JAVA:

#### **Archivo personas.java**

```
public class personas {  
    String Nombre;  
    String Apellido;  
    int edad;  
  
    public personas(String nombre, String apellido, int edad){ // Constructor con  
parámetros  
        this.Nombre = nombre;  
        this.Apellido = apellido;  
        this.edad = edad;  
    }  
    public void MiNombre() { // Método que muestra el nombre por pantalla  
        System.out.println("Mi nombre es: " + Apellido + ", " + Nombre);  
    }  
    public void MiEdad() { // Método que muestra la edad por pantalla  
        System.out.println("Y tengo " + edad + " años");  
    }  
}
```

Java requiere una llamada al constructor para cada objeto que se crea, por lo que éste es el punto ideal para inicializar las variables de instancia de un objeto.

Para reforzar y ampliar los conceptos de Objetos y del paradigma de programación orientado a objetos te recomendamos que veas el video donde el profesor Jorge Fumarola, docente del plan codo a codo, en donde explica estos temas en el siguiente link:  
<https://youtu.be/dF0LxyRRiq4>