

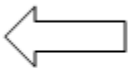
# Métodos en Java

Un **método en Java** es un conjunto de instrucciones definidas dentro de una clase, que realizan una determinada tarea y a las que podemos invocar mediante un nombre.

Algunos métodos que hemos utilizado hasta ahora:

- `Math.pow()`
- `Math.sqrt()`
- `Character.isDigit()`
- `System.out.println();`

```
public static void main(String[] args) {  
  
    double x = Math.pow(3, 7);  
  
    System.out.println(x);  
}
```

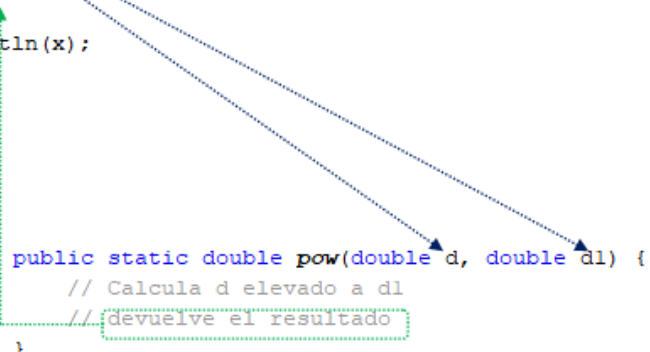


Llamada al método `Math.pow`

En la imagen vemos la llamada al método `Math.pow` para que realice la operación de elevar 3 a 7. También en la instrucción siguiente se está haciendo uso del método `println` para mostrar el valor de `x`.

Cuando se llama a un método, la ejecución del programa pasa al método y cuando éste acaba, la ejecución continúa a partir del punto donde se produjo la llamada.

```
public static void main(String[] args) {  
  
    double x = Math.pow(3, 7);  
  
    System.out.println(x);  
}
```



El método `pow` es uno de los métodos que proporciona la clase `Math` de Java

```
public static double pow(double d, double dl) {  
    // Calcula d elevado a dl  
    // devuelve el resultado  
}
```

En la imagen se muestra el flujo de ejecución del programa cuando se produce la llamada al método `Math.pow`: En la llamada al método le enviamos los valores con los que tiene que realizar la operación. La ejecución del programa continúa dentro del método. Su tarea en este caso es realizar la operación de elevar 3 a 7. Cuando termina devuelve el resultado al punto donde se invocó el método. La ejecución continúa en el método `main` desde el punto donde se produjo la llamada.


Utilizando métodos:

- Podemos construir programas modulares.
- Se consigue la reutilización de código. En lugar de escribir el mismo código repetido cuando se necesite, por ejemplo, para validar una fecha, se hace una llamada al método que lo realiza.
-

Cuando trabajamos con métodos debemos tener en cuenta lo siguiente:

- En Java un método siempre pertenece a una clase. **No podemos escribir métodos fuera de una clase.**
- No podemos escribir métodos dentro de otros métodos.
- Todo programa java tiene un método llamado **main**. La ejecución del programa empieza en este método. El método main es el punto de entrada al programa y también el punto de salida.
- Un método tiene un único punto de inicio, representando por la llave de inicio {
- La ejecución de un método termina cuando se llega a la llave final } o cuando se ejecuta una instrucción *return*.
- La instrucción return puede aparecer en cualquier lugar dentro del método, no tiene que estar necesariamente al final.
- Cuando un método finaliza, la ejecución del programa continúa a partir del punto donde se produjo la llamada al método.
- Desde dentro de un método se puede a su vez invocar a otro método.

```
public class MetodosJava {  
    public static void main(String[] args) {  
        // Método main. El programa comienza a ejecutarse en este método  
    }  
    //Resto de métodos  
    //El resto de métodos se escriben dentro de la clase y fuera de cualquier otro método  
    //Dentro de un método no se puede escribir el código de otro método.  
}
```



## 1. ESTRUCTURA GENERAL DE UN MÉTODO JAVA

La estructura general de un método Java es la siguiente:

```
[especificadores] tipoDevuelto nombreMetodo([lista parámetros]) [throws  
listaExcepciones]  
{  
    // instrucciones  
    [return valor;]  
}
```

Los elementos que aparecen entre corchetes son opcionales.

**especificadores** (opcional): determinan el tipo de acceso al método. Se verán en detalle más adelante.

**tipoDevuelto**: indica el tipo del valor que devuelve el método. En Java es imprescindible que, en la declaración de un método, se indique el tipo de dato que ha de devolver. El dato se devuelve mediante la instrucción return. Si el método no devuelve ningún valor este tipo será void.

**nombreMetodo**: es el nombre que se le da al método. Para crearlo hay que seguir las mismas normas que para crear nombres de variables.

**Lista de parámetros** (opcional): después del nombre del método y siempre entre paréntesis puede aparecer una lista de parámetros, también llamados **argumentos**, separados por comas. Estos parámetros son los datos de entrada que recibe el método para operar con ellos. Un método puede recibir cero o más argumentos. Se debe especificar para cada argumento su tipo. **Los paréntesis a continuación del nombre del método son obligatorios**, aunque estén vacíos.

**throws listaExcepciones** (opcional): indica las excepciones que puede generar y manipular el método.

**return:** se utiliza para devolver un valor. Algunos aspectos importantes sobre la palabra clave *return* en un método:

- La palabra clave *return* va seguida de una expresión que será evaluada para saber el valor de retorno. Esta expresión puede ser compleja o puede ser simplemente una variable de tipo primitivo o una constante.
- El tipo del valor de retorno debe coincidir con el tipoDevuelto que se ha indicado en la declaración del método.
- Si el método no devuelve nada (tipoDevuelto = void) la instrucción *return* es opcional.
- Un método puede devolver un tipo primitivo, un array, un String o un objeto.
- La instrucción *return* puede aparecer en cualquier lugar dentro del método.
- La ejecución de un método termina cuando se llega a su llave final `}` o cuando se ejecuta la instrucción *return*.

## 2. IMPLEMENTACIÓN DE MÉTODOS EN JAVA

Pasos para implementar un método:

1. Describir lo que el método debe hacer.
2. Determinar las entradas del método, es decir, lo que el método recibe.
3. Determinar los tipos de datos de las entradas.
4. Determinar lo que debe devolver el método y el tipo del valor devuelto.
5. Escribir las instrucciones que forman el cuerpo del método.
6. Prueba del método: diseñar distintos casos de prueba.

**Ejemplo de método:** método llamado *sumar* que recibe dos números enteros y calcula y devuelve su suma.

```
import java.util.Scanner;

public class Metodos1 {

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

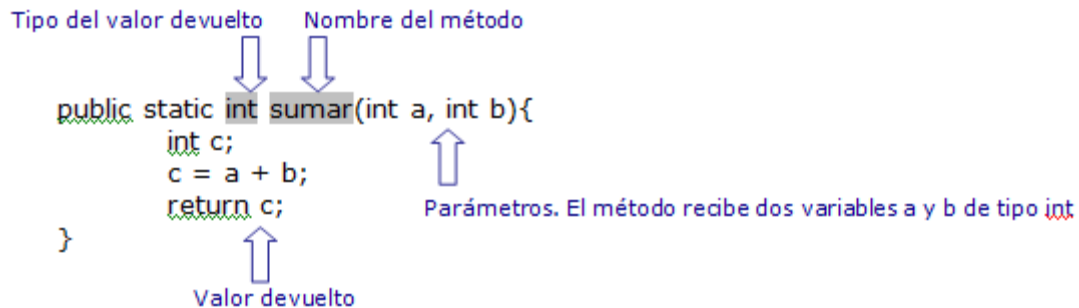
```

        Scanner sc = new Scanner(System.in);
        int numero1, numero2, resultado;
        System.out.print("Introduce primer número: ");
        numero1 = sc.nextInt();
        System.out.print("Introduce segundo número: ");
        numero2 = sc.nextInt();
        resultado = sumar(numero1, numero2);
        System.out.println("Suma: " + resultado);
    }

    //método sumar
    public static int sumar(int a, int b){
        int c;
        c = a + b;
        return c;
    }
}

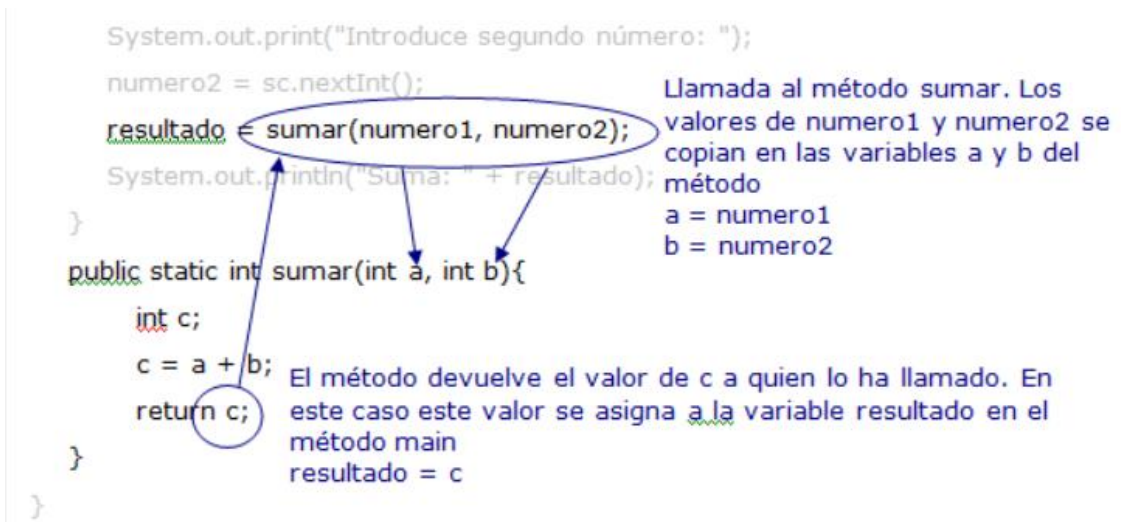
```

El método se llama sumar. Recibe dos números enteros a y b que son los parámetros de entrada. El método devuelve un valor de tipo int.



El flujo de ejecución del programa es el siguiente:

1. En el método *main* cuando se produce la llamada al método *sumar*, **los valores de las variables numero1 y numero2 se copian en las variables a y b respectivamente. Las variables a y b son los parámetros del método.**
2. El flujo de ejecución del programa pasa al método *sumar*.
3. El método suma los dos números y guarda el resultado en *c*.
4. El método devuelve mediante la instrucción *return* la suma calculada.
5. Finaliza la ejecución del método.
6. El flujo de ejecución del programa continúa en el método *main* a partir de dónde se produjo la llamada al método *sumar*.
7. El valor devuelto por el método es lo que se asigna a la variable *resultado* en el método *main*.



Si un método devuelve un valor, como sucede en el método *sumar*, la llamada al método puede estar incluida en una expresión que utilice el valor devuelto.

En el ejemplo anterior, el método *sumar* devuelve un valor entero que después vamos a mostrar. En este caso podríamos haber hecho la llamada al método directamente dentro del `System.out.println`:

```
System.out.println("Suma: " + sumar(numero1, numero2));
```

**Ejemplo de programa Java** que contiene un método con varios return:

Programa que lee por teclado un año y calcula y muestra si es bisiesto.

Para realizar el cálculo se utiliza un método llamado *esBisiesto*. El método *esBisiesto* tiene un parámetro de tipo entero llamado *a*. Será el encargado de recibir el valor del año *a* a comprobar si es bisiesto o no. El método devuelve un valor de tipo boolean. Devuelve *true* si el año recibido es bisiesto y *false* si no lo es. Como el método devuelve un dato de tipo boolean se puede llamar dentro de una instrucción *if*.

```
import java.util.Scanner;
```

```
public class Main {
```

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

```
        Scanner sc = new Scanner(System.in);
        int año;
        System.out.print("Introduce año: ");
        año = sc.nextInt();
        if(esBisiesto(año)){ //llamada al método
            System.out.println("Bisiesto");
        }else{
            System.out.println("No es bisiesto");
        }
    }
```

```
    /*
     * método que calcula si un año es o no bisiesto
     */
    public static boolean esBisiesto(int a){
        if(a%4==0 && a%100!=0 || a%400==0)
```

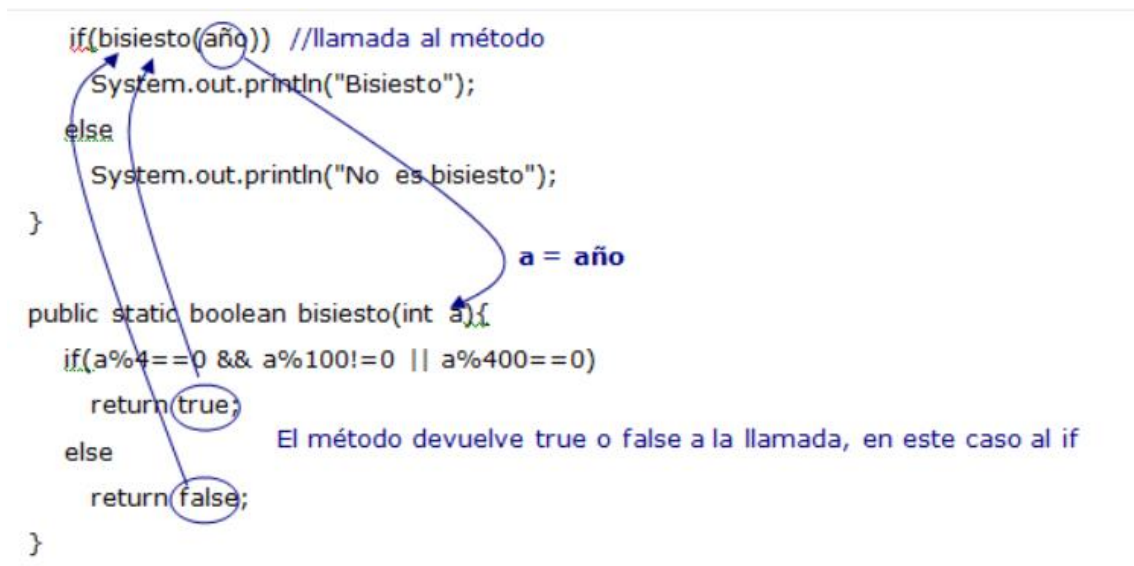
```

        return true;
    else
        return false;
    }
}

```

Flujo de ejecución:

1. En el método main se llama al método esBisiesto dentro de la instrucción if.
2. El valor de la variable año se copia en la variable a.
3. El flujo de ejecución pasa al método esBisiesto.
4. El método comprueba si el año recibido es o no bisiesto.
5. El método devuelve true si el año es bisiesto o false si no lo es.
6. Finaliza la ejecución del método.
7. El flujo de ejecución continúa en el método main en la instrucción if donde se produjo la llamada al método.
8. El valor true o false devuelto por el método es lo que determina si la condición es cierta o no.



**Ejemplo de programa Java:** Método que no devuelve ningún valor.

El método *cajaTexto* tiene un parámetro de tipo String.

El String que recibe lo muestra por pantalla rodeado con un borde.

Este método no devuelve nada, solo se limita a mostrar el String por pantalla.

Cuando un método no devuelve nada hay que indicar **void** como tipo devuelto.

Como el método no devuelve nada no es necesario escribir la sentencia return.

El método acaba cuando se alcanza la llave final.

```
import java.util.Scanner;
```

```
public class MetodoVoid {
```

```

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String cadena;
        System.out.print("Introduce cadena de texto: ");
        cadena = sc.nextLine();
    }
}

```

```

    cajaTexto(cadena); //llamada al método
}

// método que muestra un String rodeado por un borde
public static void cajaTexto(String str){
    int n = str.length(); //longitud del String
    for (int i = 1; i <= n + 4; i++){ //borde de arriba
        System.out.print("#");
    }
    System.out.println();
    System.out.println("# " + str + " #"); //cadena con un borde en cada lado
    for (int i = 1; i <= n + 4; i++){ //borde de abajo
        System.out.print("#");
    }
    System.out.println();
}
}
}

```

### 3. INVOCANDO FUNCIONES Y PROCEDIMIENTOS EN JAVA

Ya hemos visto cómo hacer funciones en Java, cómo se crean y cómo se ejecutan, ahora veamos cómo usar un método, función o procedimiento.

```
nombre([valor,[valor]...]);
```

Como puedes notar es bastante sencillo invocar o llamar funciones en Java, sólo necesitas el nombre del método, función o procedimiento y enviarle el valor de los parámetros. Hay que hacer algunas salvedades respecto a esto.

#### Detalles para invocar métodos funciones y procedimientos

- No importa si se trata de un método en Java o de una función o de un procedimiento, sólo debes ocuparte de enviar los parámetros de la forma correcta para invocarlos.
- El nombre debe coincidir exactamente al momento de invocar, pues es la única forma de identificarlo.
- El orden de los parámetros y el tipo debe coincidir. Hay que ser cuidadosos al momento de enviar los parámetros, debemos hacerlo en el mismo orden en el que fueron declarados y deben ser del mismo tipo (número, texto u otros).
- Cada parámetro enviado también va separado por comas.
- Si una función no recibe parámetros, simplemente no ponemos nada al interior de los paréntesis, pero SIEMPRE debemos poner los paréntesis.
- Invocar una función sigue siendo una sentencia común y corriente en Java, así que ésta debe finalizar con ';' como siempre.
- El valor retornado por un método o función puede ser asignado a una variable del mismo tipo, pero no podemos hacer esto con un procedimiento, pues no retornan valor alguno.
- Una función puede llamar a otra dentro de sí misma o incluso puede ser enviada como parámetro a otra (mira el siguiente ejemplo).

## Ejercicios de Métodos en java

- 1.- Crear un método que muestre los datos (nombre, apellido y la edad) de un usuario introducido por teclado.
- 2.- Diseñe un método que imprima los datos de una persona ingresados por teclado e indicar si es mayor o menor de edad.
- 3.- En un método mostrar la tabla de multiplicar de cualquier número ingresado por teclado.
- 4.- En un supermercado se hace un 20% de descuento a los clientes cuya compra supere los 100 en las áreas de frutas, verduras y dulces. ¿Cuál será el total que pagará una persona por su compra?, se debe mostrar el nombre del cliente, producto, precio, cantidad, descuento y total a pagar....