Métodos en Java, funciones y procedimientos. Cómo hacerlos y usarlos

Los métodos en Java, las funciones y los procedimientos, especialmente en Java, son una herramienta indispensable para programar. Java nos permite crear o hacer nuestros propios métodos y usarlos sencillamente, como también nos facilita hacer uso de los métodos de otras librerías (funciones matemáticas, aritméticas, de archivos, de fechas, etc. Cualquiera que sea el caso, los métodos nos permiten automatizar tareas que requiramos con frecuencia y que además se puedan generalizar por medio de parámetros o argumentos.

Aprender a crear métodos en Java y usarlos correctamente es de gran importancia, separar nuestro código en módulos y según las tareas que requerimos. En java un método debe contener la implementación de una utilidad de nuestra aplicación, esto nos pide que por cada utilidad básica (abrir, cerrar, cargar, mover, etc.) sería adecuado tener al menos un método asociado a ésta, pues sería muy complejo usar o crear un método que haga todo de una sola vez. Por esto es muy buena idea separar cada tarea en una función o método (según corresponda).

En Java es mucho más común hablar de métodos que de funciones y procedimientos. En programación, un método, una función y un procedimiento no son lo mismo, veamos la diferencia:

¿Funciones, métodos o procedimientos?

Es muy común entre programadores que se hable indistintamente de estos tres términos, sin embargo, poseen diferencias fundamentales.

Funciones:

Las funciones son un conjunto de líneas de código (instrucciones), encapsulados en un bloque, usualmente reciben parámetros, cuyos valores utilizan para efectuar operaciones y adicionalmente devuelven/retornan un valor. En otras palabras, una función puede recibir parámetros o argumentos (algunas no reciben nada), hace uso de dichos valores recibidos como sea necesario y devuelve un valor usando la instrucción **return**. Si no devuelve algo, entonces no es una función (se suele llamar procedimiento en este caso).

Métodos:

Los métodos y las funciones (y/o procedimientos) en Java pueden realizar las mismas tareas, es decir, son funcionalmente idénticos, pero su diferencia radica en la manera en que hacemos uso de uno u otro (el contexto). Un método también puede recibir valores, efectuar operaciones con estos y retornar valores, sin embargo, un método está asociado a un objeto **siempre**, básicamente un método es una función/procedimiento que pertenece a un objeto o clase, mientras que una función/procedimiento existe por sí sola, sin necesidad de un objeto para ser usada. **Nota:** En Java se debe hablar de métodos y no de funciones/ procedimientos, pues en Java estamos siempre obligados a crear un

objeto para usar el método. Para que sea una función/procedimiento, esta debe ser *static*, para que no requiera de un objeto para ser llamada.

Procedimientos:

Los procedimientos son básicamente un conjunto de instrucciones que se ejecutan sin retornar ningún valor, y que pueden recibir o no argumentos. En el contexto de Java un procedimiento es básicamente un método cuyo tipo de retorno es *void*, que no nos obliga a utilizar una sentencia return.

Crear un método en Java

La sintaxis para declarar un método tipo función es muy simple, veamos:

```
[acceso] [modificador] tipoDevuelto nombreFuncion([tipo
nombreArgumento,[tipo nombreArgumento]...])
{
        /*
                * Bloque de instrucciones
        */

        return valor;
}
```

El primer componente corresponde al **modificador de acceso**, que puede ser public, private o protected, este último es opcional, si no ponemos nada, se asume el modificador de acceso por defecto.

El segundo componente es el **modificador** que puede ser final o static (o ambas), también es opcional. Recordemos que un método o función siempre retorna algo, por lo tanto es obligatorio declararle un **tipo** (el tercer componente de la sintaxis anterior), puede ser entero (int), booleano (boolean), o cualquiera que consideremos, incluso tipos complejos. Si no es función, pondremos void.

Luego debemos darle un **nombre** a dicha función/método/procedimiento, para poder identificarla y llamarla (invocarla) durante la ejecución. Después, en el interior del paréntesis, podemos poner los **argumentos** o parámetros.

Cuando hemos acabado con la definición de la "firma" o prototipo del método, definimos su funcionamiento entre llaves. Todo lo que esté dentro de las llaves, es parte del cuerpo del método y este se ejecuta hasta llegar a una instrucción *return* o cuando se alcanza la última sentencia a ejecutar.

Return y void

En algunas ocasiones, no es necesario que el método estático tenga que devolver un valor al finalizar su ejecución. En este caso, el tipo de dato que debe indicar en la cabecera de declaración del método es el tipo `void` y la sentencia `return` no viene seguida de ninguna expresión. Este return solitario se puede omitir.

Sintaxis:

```
return;
```

Ejemplo:

Seguidamente se muestra un ejemplo de declaración y uso de un método tipo función, que devuelve el cubo de un valor numérico real con una sentencia `return`:

```
/**
 * Demostracion del metodo cubo
 */
public class PruebaCubo {
    public static void main (String [] args){
        System.out.println("El cubo de 7.5 es: " + cubo(7.5));
        // llamada dentro de la sentencia System.out.println
    }

    public static double cubo (double x) {
        // declaracion
        return x*x*x;
    }
}
```

Si no hay sentencia `return` dentro de un método, su ejecución continúa hasta que se alcanza el final del método y entonces se devuelve la secuencia de ejecución al lugar dónde se invocó al método.

Un método cuyo tipo de retorno no es `void` necesita siempre devolver algo. Si el código de un método contiene varias sentencias `if` debe asegurarse de que cada una de las posibles opciones devuelve un valor. En caso contrario, se generaría un error de compilación.

Por ejemplo:

```
/**
 * Demostracion de la funcion esPositivo
 */
public class PruebaPositivo {
    public static void main (String [] args) {
        for (int i=5; i>=-5; i--)
            System.out.println(i + " es positivo: " + esPositivo(i));
    }

    public static boolean esPositivo(int x) {
        if (x<0) return false;
        if (x>0) return true;
    }
}
```

Ejemplo de intento de compilación del código anterior:

```
$>javac PruebaPositivo.java
pruebaPositivo.java:14: missing return statement
```

Como se ha dicho antes, si no hay sentencia `return` dentro de un método, su ejecución continúa hasta que se alcanza el final del método y entonces se devuelve la secuencia de ejecución al lugar dónde se invocó al método.

En el siguiente código se incluye un ejemplo de método que no devuelve un valor (de tipo `void`):

```
/**
 * Demostracion del metodo tabla
 */
public class PruebaTabla {
    public static void main (String [] args){
        tabla(4);
        tabla(7);
    }

    public static void tabla (int n) {
        // ejemplo de llamada
        // de tipo void
        System.out.println("Tabla de multiplicar del numero " + n);
        for (int i=0; i<=10; i++)
            System.out.println(n + " x " + i + " = " + producto(n,i));

        return; // No devuelve ningun valor
    }

    public static int producto (int a, int b) {
        return a*b;
    }

}
```