**FUNCIONES**

Las funciones son elementos Java que utilizaremos principalmente para dos cosas: evitar la duplicidad de código y favorecer el mantenimiento de la aplicación.

La sintaxis básica de una función es la siguiente:

public static tipoFuncion nombreFuncion (parámetros){

instrucciones;

return valor;

}

De forma general diferenciamos dos tipos de funciones:

* **Funciones que no devuelven un valor.** Son las funciones de tipo void. Estas funciones reciben una serie de parámetros con los cuales realizan una serie de instrucciones y dan lugar a unos resultados que no son devueltos al método que llamó a la función.
* **Funciones que sí devuelven un valor.** Son las funciones de los tipos primitivos (int, long, boolean, String, array…). Estas funciones reciben una serie de parámetros y realizan una serie de instrucciones que dan lugar a un o unos resultados que son devueltos al método que llamó a la función.

**LLAMADA A LAS FUNCIONES**

Para llamar a una función debemos realizar la siguiente sintaxis:

nombreFuncion (parametroEntrada1, parametroEntrada2, ...);

Estos parámetros de entrada son variables que hemos creado e iniciado en el método que llama a la función. Son los parámetros con los que trabajará la función para producir unos resultados. Se deben introducir en el mismo orden en el que los escribiremos en la declaración de la función.

En el caso de que se trate de una función que devuelve un valor la llamada de la función se debe realizar asignada a una variable que guarde el valor en su interior.

Por ejemplo:



Llamada a una función de nombre calculadora a la que se le pasan 3 parámetros: dos números decimales y un valor de opción elegida (para la operación a realizar). Esta función trabajará con estos datos y dará lugar a unos resultados (función de tipo void).



Llamada a una función de nombre numeroMayor a la que se le pasan 3 parámetros: tres números enteros. Esta función trabajará con estos datos y dará lugar a un resultado que devolverá al método main y se guardará en la variable intNumMayor (función de tipo int).

**FUNCIONES DE TIPO VOID**

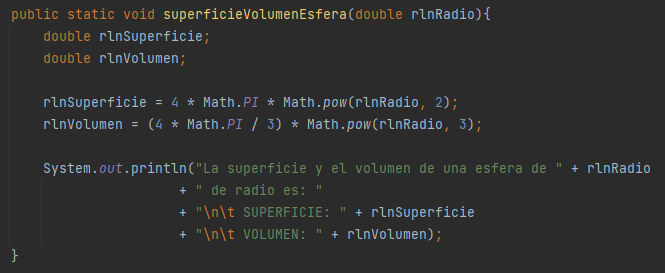
Son funciones que, como se ha dicho más arriba, no van a devolver ningún resultado al método que ha realizado la llamada de la función, por lo tanto, no van a tener ningún return y el programa terminará su ejecución dentro de ellas.

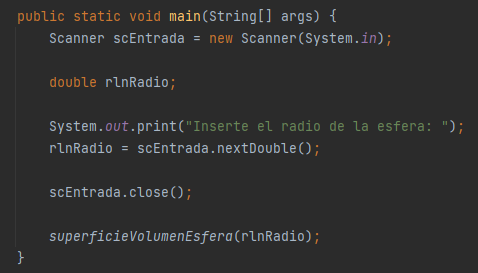
Su sintaxis es:

public static void nombreFuncion (parámetros){

instrucciones;

}

Por ejemplo, vamos a crear una función que calcule el volumen y la superficie de una esfera:

Esta función será llamada desde el método main y se le pasará como parámetro un valor para el radio:

Comenzamos la ejecución del programa preguntándole al usuario el radio de la esfera que quiere calcular.

Obtenemos ese radio y llamamos a la función superficieVolumenEsfera pasándole como parámetro el radio (rlnRadio).

Ya en la función realizamos los cálculos necesarios para conocer la superficie y el volumen de la esfera con el radio que nos ha dado el usuario e imprimimos por consola los resultados.

En este punto finaliza el programa. Es decir, una vez llamada la función, el programa ha continuado su ejecución dentro de la función y no ha vuelto al main.

**FUNCIONES QUE RETORNAN VALORES**

Son las funciones que van a recibir una serie de parámetros, van a trabajar con ellos y van a producir una serie de resultados los cuales van a ser devueltos al método que llamó a la función para que siga trabajando con ellos.

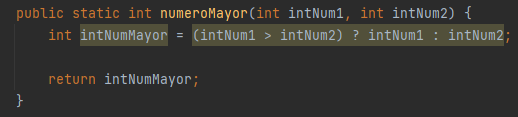
Su sintaxis es:

public static tipoFuncion nombreFuncion (parámetros){

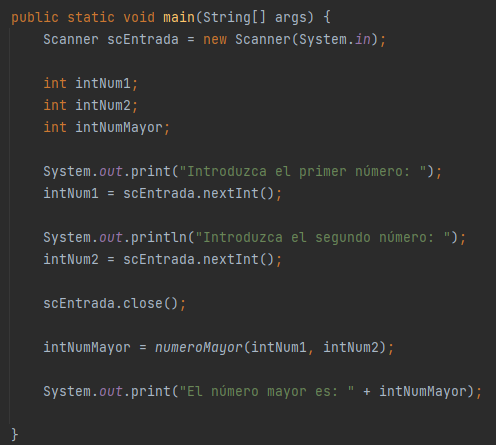
instrucciones;

return valor;

}

Por ejemplo, una función en la que pasamos dos números enteros y nos va a devolver cuál es el número mayor:

Esta función será llamada desde el método main:



En este caso el programa comienza solicitando 2 números al usuario que se guardan en dos variables int (intNum1, intNum2).

A continuación, se va a realizar la llamada a la función numeroMayor, como en este caso estamos ante una función que produce un return, es conveniente (no obligatorio) asignar el valor que va a ser devuelto a una variable, en este caso intNumMayor. A esta función se le pasan como parámetros los dos números enteros que nos ha introducido el usuario.

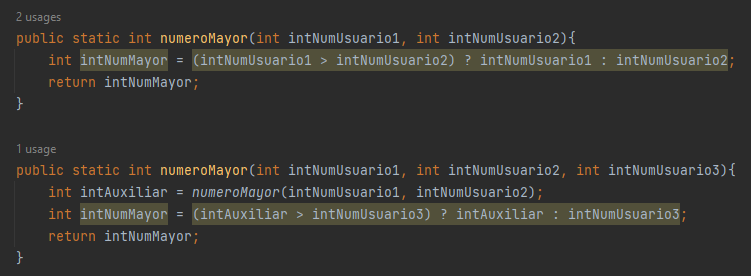
Llegados a este punto se ejecuta la función la cual tiene en su interior un operador ternario que determina cuál de los dos números pasados por parámetro es mayor, ese número se guarda en una variable y es devuelto al método main para continuar la ejecución del programa.

Finalmente se imprime por consola el resultado.

**SOBRECARGA DE MÉTODOS O DE FUNCIONES**

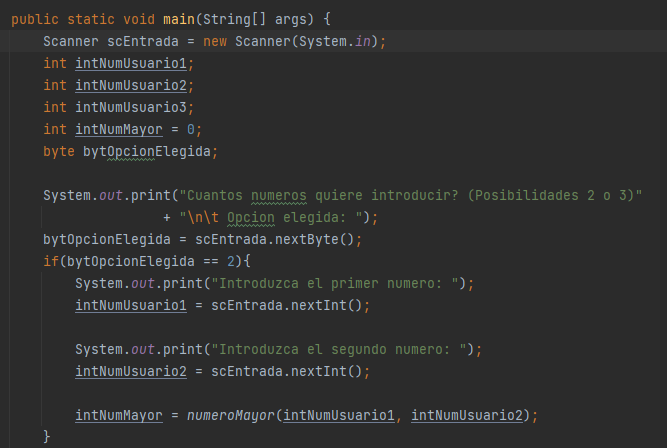
La sobrecarga de funciones se produce cuando con un mismo nombre de función, pero con diferente número o tipo de parámetros podemos realizar instrucciones diferentes.

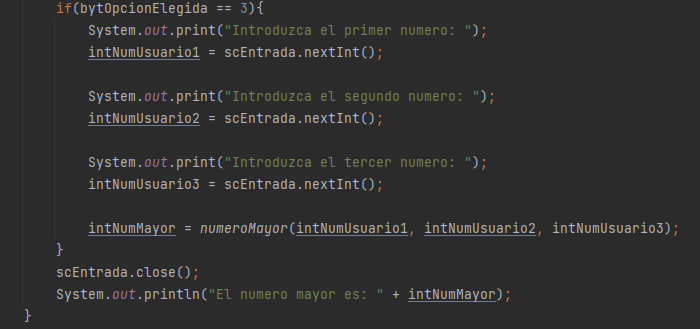
Por ejemplo, vamos a sobrecargar la función del número mayor:



Tenemos 2 funciones del mismo nombre, pero con diferentes parámetros, en la primera función (la original) tenemos 2 números y en la segunda tenemos 3 números.

Ambas funciones van a ser llamadas desde el método main:



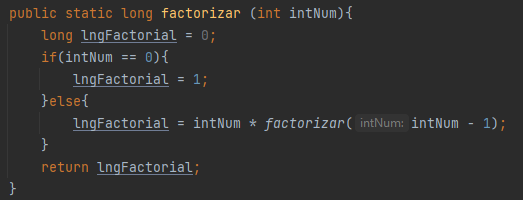


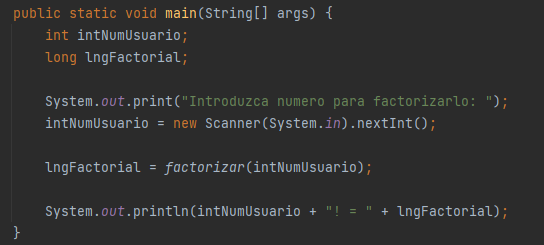
En este caso iniciamos el programa preguntándole al usuario si quiere introducir dos o tres números, en función de su respuesta se activará un condicional u otro y con ello se activará una u otra función.

La función de dos parámetros realizará lo mismo que en el caso anterior pero la función de tres parámetros primero realizará una llamada a la función de dos parámetros y guardará el primer número mayor dentro de un auxiliar el cual luego comparará con el tercer número.

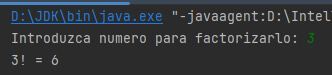
FUNCIONES RECURSIVAS

Son funciones que se llaman a si mismas para dar lugar a un resultado.

Por ejemplo, vamos a hacer una función que calcule el factorial de un número de forma recursiva:



Por ejemplo, el usuario nos pide el factorial de 3, lo que hará esta función es:

* Paso 1 🡪 intNum = 3 🡪 lngFactorial = 3 \* factorizar (2);
* Paso 2 🡪 intNum = 2 🡪 lngFactorial = 2 \* factorizar(1);
* Paso 3 🡪 intNum = 1 🡪 lngFactorial = 1 \* factorizar(0);
* Paso 4 🡪 intNum = 0 🡪 lngFactorial = 1;
* Paso 5 🡪 lngFactorial = 1 🡪 lngFactorial = 1 \* 1 = 1;
* Paso 6 🡪 lngFactorial = 1 🡪 lngFactorial = 1 \* 2 = 2;
* Paso 7 🡪 lngFactorial = 2 🡪 lngFactorial = 2 \* 3 = 6;
* Paso 8 🡪 Se devuelve el valor de lngFactorial.