RETORNO DE VALORES

La sentencia return se emplea para salir de la secuencia de ejecución de las sentencias de un método y, opcionalmente, devolver un valor. Tras la salida del método se vuelve a la secuencia de ejecución del programa al lugar de llamada de dicho método.

Sintaxis:

return expresion;

Declaración y uso de métodos

Un método es un trozo de código que puede ser llamado o invocado por el programa principal o por otro método para realizar alguna tarea específica. El término método en Java es equivalente al de subprograma, rutina, subrutina, procedimiento o función en otros lenguajes de programación. El método es llamado por su nombre o identificador seguido por una secuencia de parámetros o argumentos (datos utilizados por el propio método para sus cálculos) entre paréntesis. Cuando el método finaliza sus operaciones,

devuelve habitualmente un valor simple al programa que lo llama, que utiliza dicho valor de la forma que le convenga. El tipo de dato devuelto por la sentencia return debe coincidir con el tipo de dato declarado en la cabecera del método.

Sintaxis de declaración de un método:

```
[modificadores] TipoDeDato identificadorMetodo (parametros formales) {
    declaraciones de variables locales;
    sentencia_1;
    sentencia_2;
    ...
    sentencia_n;
    // dentro de estas sentencias se incluye al menos un return
}
```

La primera línea de código corresponde a la cabecera del método. Los modificadores especifican cómo puede llamarse al método, el tipo de dato indica el tipo de valor que devuelve la llamada al método y los parámetros (entre paréntesis) introducen información para la ejecución del método. Si no existen parámetros explícitos se dejan los paréntesis vacíos. A continuación, las sentencias entre llaves componen el cuerpo del método. Dentro del cuerpo del método se localiza, al menos, una sentencia return.

Un ejemplo sencillo

Seguidamente se muestra un ejemplo de declaración y uso de un método 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
    }

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

A diferencia de otros lenguajes de programación, como Pascal, en Java, la declaración del método puede realizarse en el código fuente después de la llamada al propio método. En el caso anterior, public y static son los modificadores especificados en la cabecera del método. El uso de estos dos modificadores permite que el tipo de método sea similar al de una función global de Pascal o C. El identificador double hace referencia al tipo de dato que devuelve la llamada al método, cubo es el identificador del método y x es el identificador del parámetro en la declaración de la cabecera del método (parámetro formal). Ejemplo de ejecución del código anterior y salida correspondiente por pantalla:

```
$>java PruebaCubo
El cubo de 7.5 es: 421.875
```

En Java, los métodos suelen ir asociados con los objetos o instancias en particular a los que operan (métodos de instancia). Los métodos que no necesitan o trabajan con objetos (y sí con números, por ejemplo) se denominan métodos estáticos o de clase y se declaran con el modificador static. Los métodos estáticos o de clase son equivalentes a las rutinas (funciones o procedimientos) de los lenguajes que no emplean la programación orientada a objetos. Por ejemplo, el método sqrt de la clase Math es un método estático. También lo es el método cubo del ejemplo anterior. Por otro lado, todo

programa o aplicación Java tiene un método principal main que será siempre un método estático.

¿Por qué se emplea la palabra static para los métodos de clase?. El significado o la acepción más común de la palabra estático (que permanece quieto en un lugar) parece no tener nada que ver con lo que hacen los métodos estáticos. Java emplea la palabra static porque C++ lo utiliza en el mismo contexto: para designar métodos de clase. Aprovechando su empleo en variables que tienen una única localización en memoria para diferentes llamadas a métodos, C++ lo empezó a utilizar en la designación de los métodos de clase para diferenciarlos de los métodos de instancia y no confundir al compilador. El problema es que nadie pensó en que el uso de la palabra static pudiera causar confusiones humanas.

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.

Sintaxis:

return;

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;
    }
}
```