Contenido

1. NÚMERO VARIABLE DE PARÁMETROS

El compilador de Java permite definir un método que admita un número variable de parámetros, es decir, es posible definir un método que puede recibir ninguno o más valores para un determinado parámetro. Para ello, se utiliza el modificador '`...`' (denominado modificador varargs, de **var**iable **arg**ument**s**), afectando al **último parámetro de la lista** de parámetros del método.

No se permite especificar más de una vez el modificador varargs. El parámetro afectado por este modificador puede pertenecer a cualquier tipo de dato y, en el cuerpo del método, se comporta como un array[1] unidimensional de ese tipo de dato.

Para invocar a un método con un número variable de parámetros:

- Si no se quieren pasar parámetros al método, basta con omitirlos en la invocación. En el cuerpo del método el parámetro variable tendrá el valor `null`.

- Si se quieren pasar uno o más parámetros, deben escribirse separados por comas en la invocación. El compilador guardará secuencialmente cada parámetro en una posición del array de parámetros variables.

El siguiente programa recibe un número variable de parámetros de tipo `int` y realiza la suma de los mismos:

```
// Dado el siguiente procedimiento...
void add(int ... numbers)
   {
   if (numbers!= null)
      {
      int result = 0;
      for(int n in numbers)
         {
         result += n;
         }
      System.out.print(result);
      }
   else
      {
      System.out.print("No se han especificado números para sumar");
      }
   }

// ...la invocación sin parámetros se realiza omitiéndolos:
add();

// Y la invocación con parámetros se realiza pasando los valores como parámetros
// discretos, separados por coma:
add(1, 2, 3);
```

[1] Un array es una colección de datos del mismo tipo y de tamaño prefijado. Se puede acceder a los datos almacenados en el array a partir de un índice de base cero, que representa la posición del dato en la colección.

2. PARÁMETROS POR DEFECTO

El compilador de Java permite especificar valores por defecto para los parámetros de una función o procedimiento. De ese modo, si el programador omite el valor del parámetro en la sentencia de invocación, se considera que el valor recibido es el valor que tiene asignado el parámetro como valor por defecto.

La especificación de valores por defecto para los parámetros se realiza mediante una asignación en la lista de parámetros del método, como en el siguiente ejemplo:

```
// Dado el siguiente procedimiento...
void testMethod(string message = "Hola")
   {
   System.out.print(message);
   }

// ...la invocación sin parámetro se realiza omitiéndolo. El parámetro tomará el
// valor por defecto especificado en el prototipo del procedimiento ("Hola").
testMethod();

// Y la invocación con parámetros se realiza normalmente. En este caso, se toma
// como parámetro real el valor utilizado en la sentencia de invocación y se
// ignora el parámetro por defecto:
testMethod("Mensaje de prueba");
```

3. INVOCACIÓN MEDIANTE PARÁMETROS CON NOMBRE

Es posible escribir la sentencia de invocación de un método de forma que cada parámetro real vaya precedido del nombre del parámetro formal. La inclusión del nombre del parámetro es opcional y puede facilitar la lectura cuando la rutina invocada tienen varios parámetros.

Al hacer uso de parámetros con nombre, es posible cambiar el orden de los parámetros en la llamada, aunque es una práctica poco recomendable.

Para realizar una invocación mediante parámetros con nombre, basta con preceder el parámetro real del nombre del parámetro formal, separado por un carácter `':'` (dos puntos):

```
// Dado el siguiente procedimiento...
void testMethod(string message, int param2)
   {
   // ...
   }

// La invocación mediante parámetros con nombre se realiza precediendo el
// parámetro real del nombre del nombre del parámetro formal.
testMethod(message: "hola", param2: 69);
```