6.3 Argumentos variables

1. Introducción

Se puede dar el caso de que se necesite crear un método que tome una cantidad variable de argumentos, según su uso preciso. Por ello, Java en su versión 5 incorporó los llamados ***varargs*** que es la abreviatura de argumentos de longitud variable (*Variable-Length Arguments*). Un método que toma una cantidad variable de argumentos se denomina ***método varargs***.

2. Sintaxis de Varargs

Un parámetro de longitud variable se especifica por tres puntos (...).

```
public static int add(int... nums)
```

Esta sintaxis le dice al compilador que se puede llamar al método *add* con cero o más argumentos. Además, hace que *nums* se declare implícitamente como un array de tipo int[]. Por lo tanto, dentro del método *add* se accede a *nums* usando la sintaxis de array normal:

```
package tema6_3_ArgumentosVariables;

public class Varargs1 {

    public void show() {

        System.out.println(add());
        System.out.println(add(1));
        System.out.println(add(new int[] { 1 }));// También se le puede pasar un
array
        System.out.println(add(1, 2));
        System.out.println(add(1, 2, 3));
        System.out.println(add(1, 2, 3, 4));
        System.out.println(add(new int[] { 1, 2, 3, 4 }));

    }

    public int add(int... nums) {

        int sum = 0;

        for (int i = 0; i < nums.length; i++) {
            sum += nums[i];
        }
```

```
        return sum;

    }

    public static void main(String[] args) {

        new Varargs1().show();

    }

}
```

Si nos fijamos en el ejemplo, el método *add* se llama con diferentes números de argumentos, incluyendo ningún argumento. Los argumentos se colocan automáticamente en un array y se pasan a *nums*. En el caso de que no haya argumentos, la longitud del array es cero.

3. Métodos con parámetros normales y parámetros variables

Un método puede tener parámetros normales junto con un parámetro de longitud variable. Sin embargo, el parámetro de longitud variable debe ser el último parámetro declarado por el método y solamente se acepta un varargs por método. Por ejemplo, esta declaración de método es perfectamente aceptable:

```
int compute(int x,int y,double z,int ... values)
```

En este caso, los tres primeros argumentos utilizados en una llamada al método *compute* se corresponden con los primeros tres parámetros. Entonces, se supone que los argumentos restantes pertenecen a *values*.

Aquí hay un ejemplo que toma un argumento normal y un argumento de longitud variable:

```
package tema6_3_ArgumentosVariables;

public class Varargs2 {

    public void show() {

        System.out.println(operate(2));
        System.out.println(operate(2, 1));
        System.out.println(operate(2, 1, 2));
        System.out.println(operate(2, 1, 2, 3));
        System.out.println(operate(2, 1, 2, 3, 4));

    }

    public int operate(int mult, int... summands) {

        int sum = 0;

        for (int i = 0; i < summands.length; i++) {
            sum += summands[i];
        }
```

```
        return sum * mult;
    }

    public static void main(String[] args) {

        new Varargs2().show();

    }

}
```

Recuerda, el parámetro varargs debe ser el último. Por ejemplo, la siguiente declaración es incorrecta:

```
int compute(int x,int y,double z,int ... values,boolean exit)//error
```

Además, solamente se acepta un varargs por método. Por ejemplo, esta declaración tampoco es válida:

```
int compute(int x,int y,double z,int ... values,double ... nums)//error
```