ARRAYS MULTIDIMENSIONALES

Los arrays Java pueden tener más de una dimensión, por ejemplo, un array de enteros de dos dimensiones se declararía:

```
int [] [] k;
```

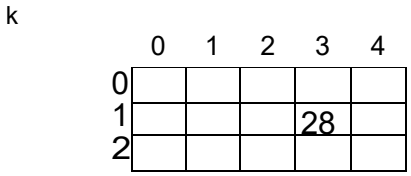
A la hora de asignarle tamaño se procedería como en los de una dimensión, indicando en los corchetes el tamaño de cada dimensión:

```
k= new int[3][5];
```

Igualmente, para acceder a cada una de las posiciones del array se utilizaría un índice por dimensión:

```
k[1] [3] =28;
```

Si imaginamos un array de dos dimensiones como una tabla organizada en filas y columnas, el array anterior tendría el aspecto indicado siguiente.



Array bidimensional

Recorrido de un array multidimensional

Para recorrer un array multidimensional podemos utilizar la instrucción *for* tradicional, empleando para ello tantas variables de control como dimensiones tenga el array. La siguiente instrucción almacenaría en cada una de las posiciones del array *k,* definido anteriormente, la suma de sus índices:

```
for (int i=0;i<k.length;i++) //recorre primera dimensión
      for (int j=0; j<k.[i].length ; j++) //recorre segunda
      dimensión
      k[i][j]=i+j;
```

Igualmente, se puede utilizar también una instrucción ***for-each*** para recuperar el contenido de un array multidimensional, simplificando enormemente dicha tarea. El siguiente ejemplo mostraría en pantalla los valores almacenados en el array *k* anterior:

```
for (int [] n:k) //recorre primera dimensión
      for (int p:n) //recorre segunda dimensión
           System.out.println("valor: "+p);
```

Arrays multidimensionales irregulares

Es posible definir un array multidimensional en el que el número de elementos de la segunda y sucesivas dimensiones sea variable, indicando inicialmente sólo el tamaño de la primera dimensión:

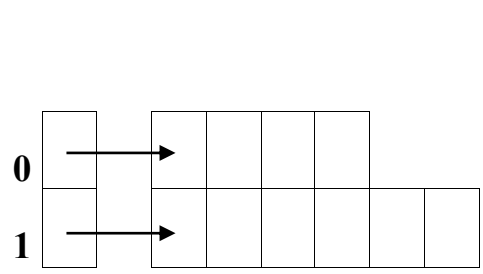
```
int [] [] irreg = new int [2] [];
```

Un array de estas características equivale a un array de arrays, donde cada elemento de la primera dimensión almacenará su propio array:

```
irreg[0]= new int[4];

irreg[1]= new int[6];
```

La figura siguiente ilustra gráficamente la situación.



Array bidimensional con dimensiones de diferente tamaño

Un array de estas características sería complicado de recorrer empleando un bucle *for* normal, en cambio, mediante *for-each* su recorrido sería exactamente igual que con los arrays multidimensionales estándar. El siguiente ejemplo recorrería el array anterior:

```
for(int[] n:irreg) //recorre primera dimensión
      for(int p:n) //recorre segunda dimensión
      System.out.println{"valor: "+p);
```

MÉTODOS CON NUMERO VARIABLE DE ARGUMENTOS

Cuando hemos explicado el funcionamiento del paso de parámetros al invocar a un método, se dijo que el número de argumentos utilizados en la llamada viene determinado por el número de parámetros que el método acepta. Esto significa que si, por ejemplo, un método tiene declarados dos parámetros, la llamada al método deberá efectuarse con dos argumentos.

A partir de la versión Java 5 es posible definir métodos que reciban un número variable de argumentos, para ello es necesario declarar un parámetro que recoja todos los argumentos de número variable. La declaración de un parámetro de estas características debe hacerse utilizando el siguiente formato:

tipo ... nombre_parámetro

Por ejemplo, un parámetro declarado de la siguiente forma en un método:

String ... cadenas

representaría cualquier número de argumentos de tipo String.

Un parámetro declarado de esta manera es realmente un array en el que se almacenan los argumentos recibidos y cuyos valores deben ser todos del mismo tipo.

El siguiente método corresponde a un método que calcula la suma de todos los números recibidos en el parámetro:

```
//puede ser invocado con cualquier número de
//argumentos de tipo entero
public int sumador (int ... varios){
      int s=0;
```

```
        for(int i:varios){
              s+=i;
        }
        return s;

}
```

El siguiente programa utiliza el método *sumado()* para realizar diversos cálculos de sumas enteras:

```
public class Cálculos {
  public static void main (String []args){
      System.out.println ( sumador (3,7,9,11)); //30
      System.out.println (sumador (25,4,6)); //30
      System.out.println (sumador (8,19));//27
   }
  public static int sumador (int ... varios){
      int s=0;
      for (int i:varios){
               s+=i;
      }
      return s;
   }
}
```

Un método puede declarar tanto parámetros estándares como parámetros para número variable de argumentos. En estos casos, los segundos deben aparecer declarados al final de la lista de parámetros:

public void método1(int k, String s, int ... nums){} //correcto public
void metodo2(int p, String ... cads, long f){} //incorrecto

La definición de métodos con un número variable de argumentos resulta muy útil para todas aquellas situaciones en donde la cantidad de datos que deba recibir el método en la llamada no esté determinada.