# 6.D. Arrays unidimensionales.

## 1. Arrays unidimensionales.

### 1.2. Inicialización.

Rellenar un array, para su primera utilización, es una tarea muy habitual que puede ser rápidamente simplificada. Vamos a explorar dos sistemas que nos van a permitir inicializar un array de forma cómoda y rápida.

En primer lugar, una forma habitual de crear un array y rellenarlo es simplemente a través de un método que lleve a cabo la creación y el rellenado del array. Esto es sencillo desde que podemos hacer que un método retorne un array simplemente indicando en la declaración que el valor retornado es tipo[], donde tipo de dato primitivo (`int, short, float`,...) o una clase existente (`String` por ejemplo). Veamos un ejemplo:

```
static int[] ArrayConNumerosConsecutivos (int totalNumeros) {

        int[] r=new int[totalNumeros];

        for (int i=0;i<totalNumeros;i++) r[i]=i;

        return r;

}
```

En el ejemplo anterior se crea un array con una serie de números consecutivos, empezando por el cero, ¿sencillo no? Este uso suele ahorrar bastantes líneas de código en tareas repetitivas. Otra forma de inicializar los arrays, cuando el número de elementos es fijo y sabido a priori, es indicando entre llaves el listado de valores que tiene el array. En el siguiente ejemplo puedes ver la inicialización de un array de tres números, y la inicialización de un array con tres cadenas de texto:

```
int[] array = {10, 20, 30};
```

```
String[] diassemana= {"Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"};
```

Pero cuidado, la inicialización solo se puede usar en ciertos casos. La inicialización anterior funciona cuando se trata de un tipo de dato primitivo (`int, short, float, double`, etc.) o un `String`, y algunos pocos casos más, pero no funcionará para cualquier objeto.

Cuando se trata de un array de objetos, la inicialización del mismo es un poco más liosa, dado que el valor inicial de los elementos del array de objetos será `null`, o lo que es lo mismo, crear un array de objetos no significa que se han creado las instancias de los objetos. Hay que crear, para cada posición del array, el objeto del tipo correspondiente con el operador `new`. Veamos un ejemplo con la clase `StringBuilder`. En el siguiente ejemplo solo aparecerá `null` por pantalla:

```
StringBuilder[] j=new StringBuilderStringBuilder[10];
```

```
for (int i=0; i<j.length;i++) System.out.println("Valor" +i + "="+j[i]); // Imprimirá null para los 10 valores.
```

Para solucionar este problema podemos optar por lo siguiente, crear para cada posición del array una instancia del objeto:

```
StringBuilder[] j=new StringBuilder[10];
```

```
for (int i=0; i<j.length;i++) j[i]=new StringBuilder("cadena "+i);
```

**Reflexiona**

Para acceder a una propiedad o a un método cuando los elementos del array son objetos, puedes usar la notación de punto detrás de los corchetes, por ejemplo: `diassemana[0].length`. Fijate bien en el array `diassemana` anterior y piensa en lo que se mostraría por pantalla con el siguiente código:

```
System.out.println(diassemana[0].substring(0,2));
```

**Autoevaluación**

**¿Cuál es el valor de la posición 3 del siguiente array:** `String[] m=new String[10]`**?**

- ○ null.
- ○ Una cadena vacía.
- ○ Daría error y no se podría compilar.