Arreglos

La sintaxis para declarar e inicializar un array será:

Tipo\_de\_variable[ ] Nombre\_del\_array = new Tipo\_de\_variable[dimensión];

También podemos alternativamente usar esta declaración:

Tipo\_de\_variable[ ] Nombre\_del\_array; Nombre\_del\_array = new Tipo\_de\_variable[dimensión];

El tipo de variable puede ser cualquiera de los admitidos por Java y que ya hemos explicado. Ejemplos de declaración e inicialización con valores por defecto de arrays usando todos los tipos de variables Java, serían:

− byte[ ] edad = new byte[4];

− short[ ] edad = new short[4];

− int[ ] edad = new int[4];

− long[ ] edad = new long[4];

− float[ ] estatura = new float[3];

− double[ ] estatura = new double[3];

− boolean[ ] estado = new boolean[5];

− char[ ] sexo = new char[2];

− String[ ] nombre = new String[2];

Aclarar que los valores por defecto son los siguientes:

a) Para números el valor cero “0”.

b) Para cadenas y letras el valor vacío.

c) Para booleanos el valor false.

En caso de que queramos inicializarlos con valores propios, haremos esto:

Para números enteros

int[ ] edad = {45, 23, 11, 9}; //Array de 4 elementos De la misma forma procederíamos para los otros tipos de enteros : byte, short, long.

Para números reales

double[ ] estatura = {1.73, 1.67, 1.56}; //Array de 3 elementos

De la misma forma procederíamos para el tipo float, pero teniendo en cuenta que los números deberán llevar al final la letra “f” o “F”. Por ejemplo 1.73f o 1.73F.

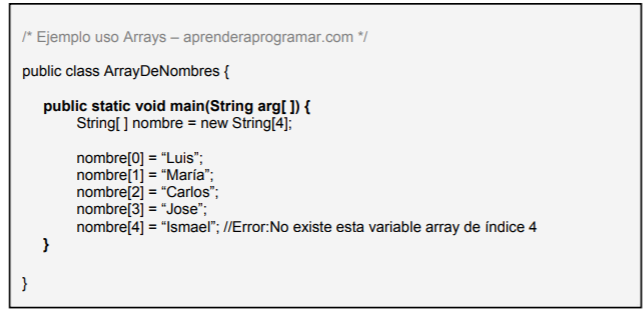
Para booleanos

boolean[ ] = {true,false}; //Array de 2 elementos

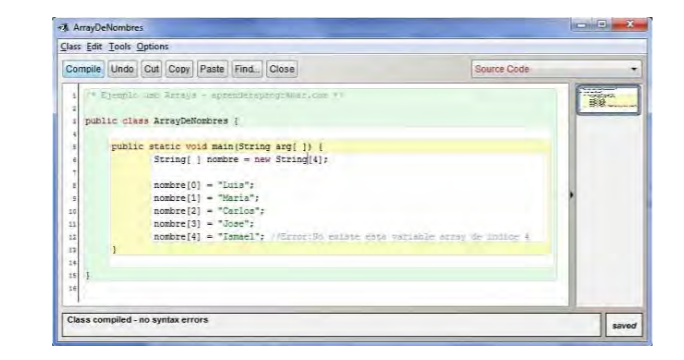
Cuando creamos un array de nombre “a” y de dimensión “n” (int[ ] a = new int[n]) estamos creando n variables que son a[0], a[1], a[2], ..., a[n-1]. Los arrays se numeran desde el elemento cero, que sería el primer elemento, hasta el n-1 que sería el último elemento. Es decir, si tenemos un array de 5 elementos, el primer elemento sería el cero y el último elemento sería el 4. Esto conviene tenerlo en cuenta porque puede dar lugar a alguna confusión. Disponer de un valor con índice cero puede ser de utilidad en situaciones como considerar cada variable asociada a una hora del día, empezando a contar desde la hora cero hasta la 23 (total de 24 horas), cosa que es habitual en algunos países. En lugar de 1, 2, 3, ..., 24 estaríamos usando 0, 1, 2, ..., 23.

Vamos a trabajarlo sobre el ordenador en un programa y ver qué pasaría si declaramos un array de tamaño “n” y vamos asignando un valor a cada variable del array desde la posición cero hasta la posición “n-1”. Adicionalmente vamos a comprobar qué ocurre si tratamos de asignarle valor a la variable de posición “n”.

El código fuente del programa (escríbelo en BlueJ) es el siguiente:



Procedemos a la compilación del programa pulsando en el botón de Compile y después a su ejecución:



Obtenemos una respuesta (interpretación) que es la siguiente:



Una vez escrito el programa, cuando compilamos observamos que no hay ningún error de sintaxis, es decir, no faltan puntos y comas, las llaves están bien equilibradas, las palabras reservadas del lenguaje son las correctas, etc. Por tanto, aparentemente el programa debería ejecutarse y darnos un resultado. Sin embargo cuando cuando interpretamos el programa observamos un error “ArrayIndexOutBoundsException”. Este mensaje de error significa “desbordamiento de array”, es decir, que tratamos de acceder a una posición del array que no existe. En nuestro caso, hemos tratado de acceder a la posición 4, que no está definida (sólo se ha definido hasta la posición 3). En esto Java es muy estricto y no admite este tipo de error lógico del programador. Siguiendo el análisis del error que nos aparece en la ventana leemos lo siguiente “ArrayDeNombres.java:12”, que nos está indicando que el error está en el archivo que contiene el programa de nombre “ArrayDeNombres.java”, y más concretamente en la línea 12.

Ahora ampliaremos conocimientos con respecto a la instrucción public static void main(String arg[ ]). Esta instrucción es un estándar en Java, es decir, es la forma habitual de encabezar un programa que se ejecuta en consola. Ahora trataremos de entender un poco mejor el significado de este encabezado. Vamos a analizar el significado de (String arg[ ]). En esta instrucción el método main admite que se le pase un “array de Strings” cuando se proceda a la ejecución del programa si así lo deseamos. String arg[] significa “un array de Strings” cuyo nombre es arg. Dicho de otra manera, al ejecutar el programa podemos indicar “parámetros de entrada” de forma opcional. Si especificamos parámetros de entrada, éstos parámetros quedan asignados al array arg tomando el orden arg[0], arg[1].