**ARRAYS**

**CREAR UN ARRAY. SINTAXIS.**

Al crear un array tenemos que realizar la siguiente sintaxis:

tipoVariable[] nombreArray;

nombreArray = new tipoVariable[tamañoArray];

Por ejemplo,

int[] intArrEdades;

intArrEdades = new int[3];

Esto se puede resumir en una única sentencia con la forma:

tipoVariable[] nombreArray = new tipoVariable[tamañoArray];

**RELLENADO DE UN ARRAY**

El array que hemos creado va a ser capaz de guardar 3 variables de tipo int. Pero no las guarda dentro del programa, sino que las guarda en posiciones de memoria, es decir, cuando creamos el array vamos a guardar un espacio en memoria, concretamente vamos a guardar tantas direcciones de memoria como tamaño tenga nuestro array, es decir, en este ejemplo vamos a guardar 3 direcciones de memoria (la 000, la 001 y la 010):

|  |  |  |
| --- | --- | --- |
| POSICIÓN 0 | POSICIÓN 1 | POSICIÓN 2 |
| 000 | 001 | 010 |

Es decir, la posición 0 del array (intArrEdades[0]) se guardará en la dirección de memoria 000, , la posición 1 del array (intArrEdades[1]) se guardará en la dirección de memoria 001 y la posición 2 del array (intArrEdades[2]) se guardará en la dirección de memoria 010.

A la hora de rellenar de forma manual un array tenemos dos opciones de sintaxis:

* Opción nombreArray[índice] = valor. En este caso lo tendremos que hacer así:

intArrEdades[0] = 18;

intArrEdades[1] = 20;

intArrEdades[2] = 21;

* Opción {valor1,valor2,valor3,…,valor n}. En este caso cambiará la sintaxis utilizada.

int[] intArrEdades = {18,20,21}

La forma de llenarse será la misma, en la posición 0 tendremos el primer valor y en la posición n-1 el último valor.

En ambos ejemplos tendremos lo siguiente:

|  |  |  |
| --- | --- | --- |
| POSICIÓN 0 | POSICIÓN 1 | POSICIÓN 2 |
| 000 | 001 | 010 |
| 18 | 20 | 21 |

**RELLANADO DE UN ARRAY DEJANDO POSICIONES SIN VALOR**

Hay que tener en cuenta que cuando inicializamos un array, todas las posiciones de este ya tienen un valor preestablecido: 0 en el caso de arrays de tipo int, 0.0 en el caso de arrays de tipo double, false en el caso de arrays de tipo boolean…

Si en la entrada de datos hemos puesto lo siguiente:

intArrEdades[0] = 18;

intArrEdades[2] = 21;

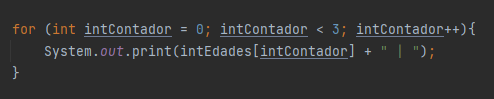
Nuestro array se habrá guardado de la siguiente manera:

|  |  |  |
| --- | --- | --- |
| POSICIÓN 0 | POSICIÓN 1 | POSICIÓN 2 |
| 000 | 001 | 010 |
| 18 | 0 | 21 |

Es decir, como no hemos inicializado el índice 1 (intArrEdades[1]) este toma el valor por defecto, es decir, el valor 0.

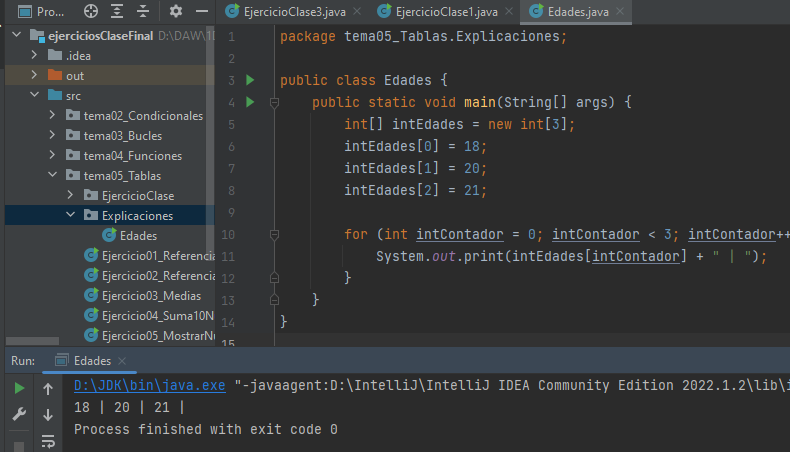
**IMPRESIÓN DE UN ARRAY**

Tenemos 3 opciones de imprimir un array:

* Bucle for. Con un bucle for, por ejemplo:

Crearemos un bucle for que inicie en 0 y que se repita mientras el iterador (intContador) sea menor que el tamaño del array (en este caso 3) y dentro de él realizaremos un System.out.print de intEdades[intContador], es decir, de intEdades[0] (iterador 0), intEdades[1] (iterador 1) e intEdades[2] (iterador 2).

De esta forma obtendremos el siguiente resultado:



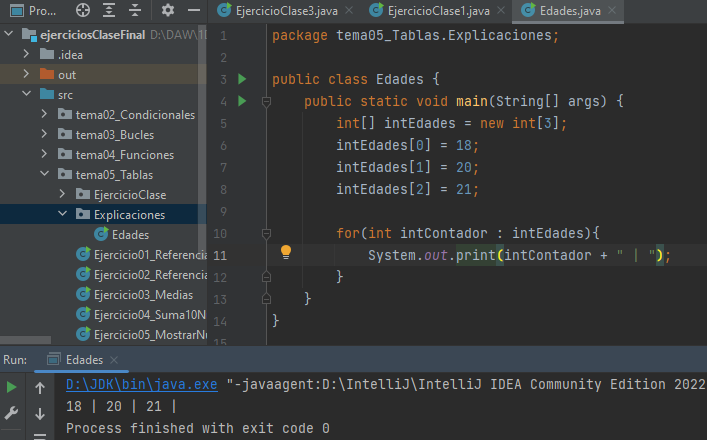
* Bucle for each. En el caso de las tablas tenemos un nuevo tipo de bucle for, este bucle a partir de una variable imprimirá toda la tabla desde el inicio hasta su fin. Su sintaxis es la siguiente:

for(declaración variable : nombreArray){

System.out.print(nombreVariable);

}

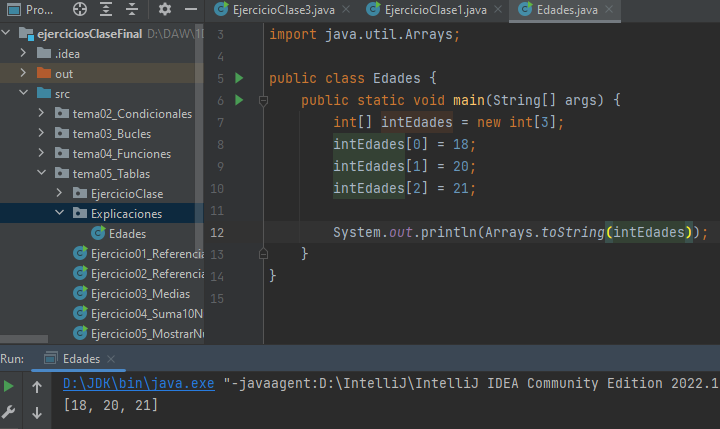
En este caso vamos a llamar a la variable intContador:



* Arrays.toString(). Por último, tenemos la posibilidad de usar uno de los métodos de la clase Arrays, concretamente el toString, este método nos va a convertir el array en un objeto de tipo String. Para ello usaremos la siguiente sintaxis:

Arrays.toString(nombreArray)

En nuestro caso será:



**REFERENCIAS**

Como ya se ha dicho los arrays no guardan los valores en sí mismos sino en direcciones de memoria y son esas direcciones de memoria lo que realmente se guarda en un array, esto es lo que se conoce como referencia.

Bien pues gracias a esto podemos hacer que 2 arrays se dirijan a la misma dirección de memoria, por ejemplo:

1. Vamos a crear un nuevo array de nombre intEdadesAuxiliar que no vamos a iniciar de primeras:



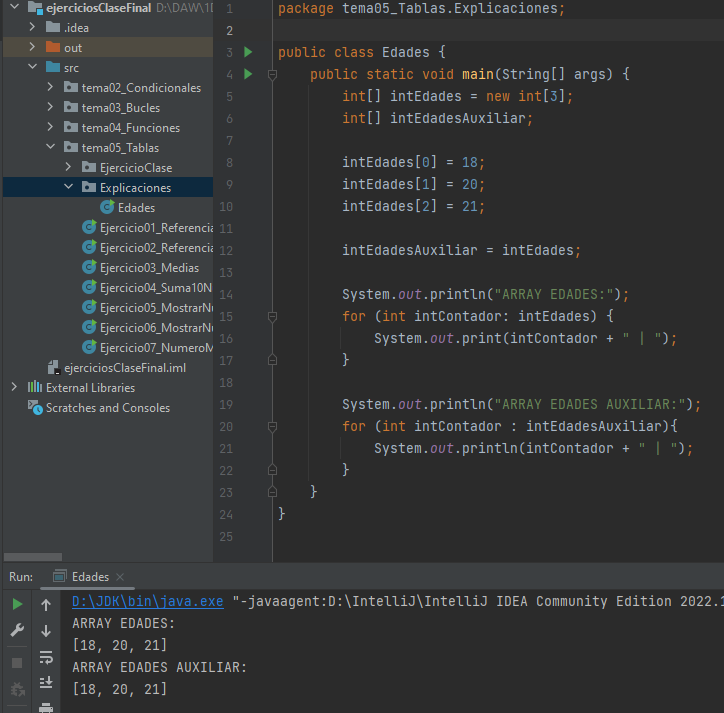
1. Una vez rellenado el array principal (intEdades) podemos hacer que este array auxiliar apunte a la misma dirección de memoria con la siguiente sintaxis:

intArrayAuxiliar = intArrayPrincipal;



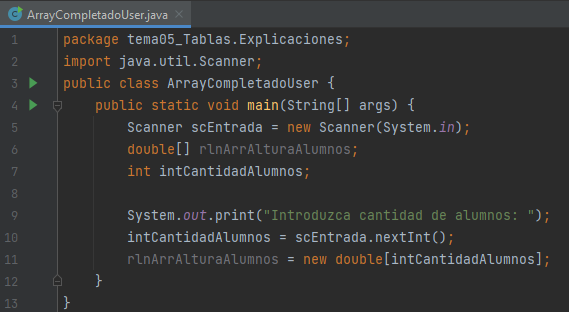
Ahora mismo ambos arrays apuntan hacía las mismas direcciones de memoria, es decir, no se han copiado los valores de las edades, sino que se han copiado las direcciones de memoria que almacenan los datos, es decir, esas posiciones 000, 001 y 010 que se comentan arriba.

Si imprimimos ambos arrays obtendremos lo siguiente:



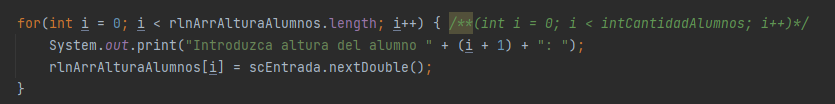
**RELLENADO DE UN ARRAY POR PARTE DEL USUARIO**

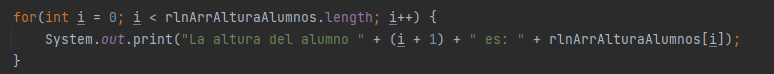
Como sucede con cualquier otro tipo de variables los arrays también pueden ser completados por usuario. Por ejemplo, un usuario puede definir el tamaño que tendrá el array:



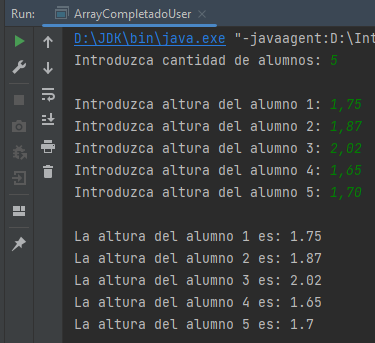
En este caso hemos creado un array de tipo String donde vamos a introducir nombres de alumnos. Además, la cantidad de alumnos, es decir, el tamaño del array lo fijará el usuario cada vez que inicie el programa a través de una variable (intCantidadAlumnos).

Para rellenar el bucle haremos uso de un bucle for que se repetirá tantas veces como cantidad de alumnos haya, para ello tenemos dos opciones que el iterador -en este caso i- vaya hasta intCantidadAlumnos o hacer uso del método length, es decir, que el iterador vaya hasta nombreArray.length -en este caso rlnArrAlturaAlumnos.length-.



Finalmente, imprimimos el array usando cualquiera de los métodos disponibles:

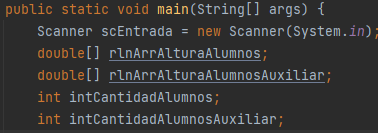
Y una ejecución completa daría este resultado por consola:

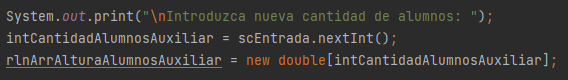


**REDIMENSIÓN DE UN ARRAY**

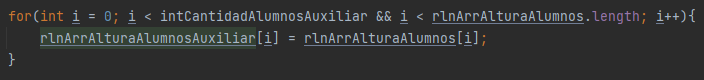
En ocasiones puede que necesitemos redimensionar un array creado previamente, por ejemplo, en nuestro ejemplo de las alturas de los alumnos porque estos pasen de 5 a 10 alumnos, en ese caso necesitaremos crear un segundo array (un array auxiliar) de tamaño 10.

1. Declararemos un segundo array y una segunda variable para guardar el tamaño (double[] rlnArralturaAlumnosAuxiliar y intCantidadAlumnosAuxiliar).



1. Una vez completado el programa como en el apartado anterior, es decir, tras la impresión de las 5 primeras alturas, le preguntaremos al usuario el nuevo número de alumnos:

Con esto habremos iniciado el array auxiliar y, por tanto, habremos reservado en memoria una cantidad de direcciones igual a intCantidadAlumnosAuxiliar.

1. Como no queremos perder los datos, vamos a copiar los datos originales en el nuevo array auxiliar.

Para ello haremos uso de un bucle for con los siguientes parámetros:

1. Iniciaremos una variable iterador (en este caso i) a 0.
2. Repetiremos el bucle mientras i sea menor a intCantidadAlumnosAuxiliar y, además, menor a la longitud del array original, es decir, rlnArrAlturaAlumnos.length (o lo que es lo mismo intCantidadAlumnos).

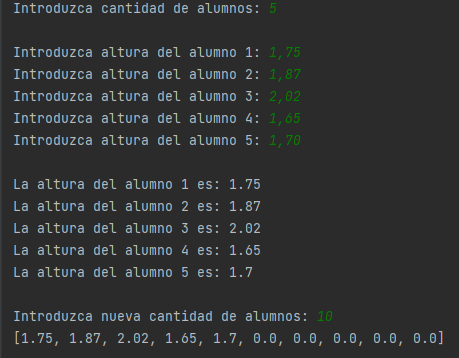
Dentro del bucle copiaremos cada dato del array original (rlnArrAlturaAlumnos[i]) en el array auxiliar (rlnArrAlturaAlumnosAuxiliar[i]), es decir, el 1.75 que estaba en rlnArrAlturaAlumnos[0] pasará a estar ahora en rlnArrAlturaAlumnosAuxiliar[0], el 1.87 que estaba en rlnArrAlturaAlumnos[1] pasará a estar ahora en rlnArrAlturaAlumnosAuxiliar[1], etc. Y, además, tendremos 5 “huecos” (desde rlnArrAlturaAlumnosAuxiliar[5] hasta rlnArrAlturaAlumnosAuxiliar[9]) que no tendrán un valor en su interior y estarán iniciados a 0.

1. Por último y para no tener espacio desaprovechado en memoria habrá que hacer que el array original apunte a las direcciones de memoria del array auxiliar e informar al programa de que el auxiliar ya no apunta a ningún lado (referencia null) para que el recolector de basura lo destruya.



Si ahora solicitamos imprimir el array veremos que se nos han copiado los 5 valores del array original pero que tenemos 5 huecos a 0.





**ARRAYS COMO PARÁMETRO DE UNA FUNCIÓN (VOID)**

Como ocurre con todos los tipos de variables, las variables de tipo array también se pueden introducir como parámetros de una función. Por ejemplo, vamos a hacer la misma redimensión que hemos hecho en el ejemplo anterior pero dentro de una función a la que llamaremos redimensionArrayAlumno(). Esta función en este caso va a ser void por lo que no vamos a devolver ningún valor al programa.

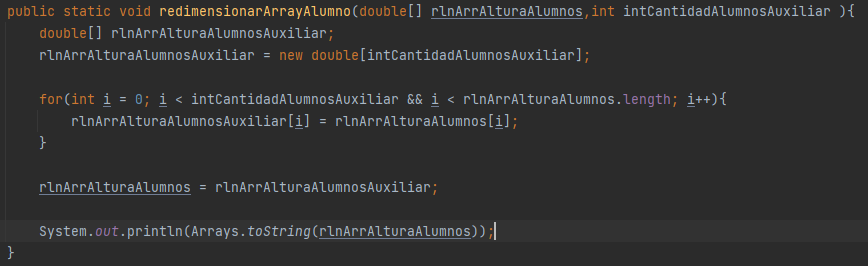
La parte inicial de nuestro programa, es decir, desde el punto de inicio hasta el momento de preguntar al usuario el nuevo número de alumnos se va a mantener idéntica al ejercicio original:



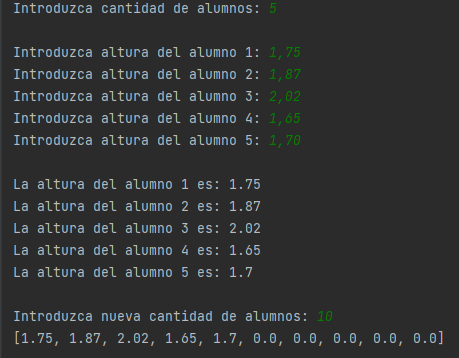
A continuación, vamos a llamar a la función y le asignaremos los parámetros de entrada del array (rlnArrAlturaAlumnos) y de la nueva cantidad de alumnos (intCantidadAlumnosAuxiliar).

Al tratarse de una función void que nos imprimirá directamente el nuevo array y con ello finalizará el programa la sintaxis en el método main será la misma que para el resto de funciones de tipo void, es decir, simplemente llamaremos a la función sin necesidad de asignarle ninguna variable que guarde el resultado:

Nuestra función va a realizar la misma tarea que se ha realizado en el anterior apartado tras la entrada de la nueva cantidad de alumnos por parte del usuario, es decir, copiar el array original en el nuevo array y, posteriormente, asignar las referencias nuevas al array original:



En este caso no es necesario igualar el array auxiliar a null porque al tratarse de un array creado dentro de una función una vez que esta acabe, el array auxiliar será “destruido”.

Si iniciamos el programa podemos ver que obtenemos el mismo resultado:

**ARRAYS COMO PARÁMETRO DE UNA FUNCIÓN (TIPOVARIABLE[])**

Al igual que pasaba con las funciones de los tipos primitivos (int, long, double, String…) podemos tener funciones que nos devuelvan variables de tipo array, para ello emplearemos la siguiente sintaxis:

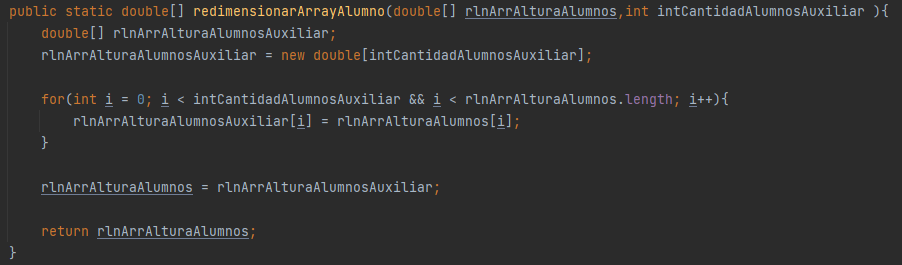
public static tipoVariable[] nombreFuncion(parámetros){

instrucciones;

return nombreArray;

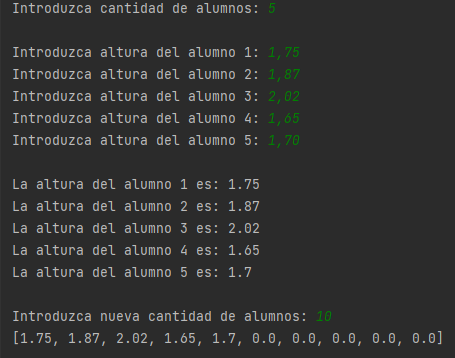
}

Por ejemplo, en nuestro caso tendríamos que utilizar la siguiente función:



En este caso lo que hará la función será asignar la nueva dimensión del array y devolver el array original (ya redimensionado al array original). Obviamente como ocurre en las variables de retorno la llamada al método debe guardarse en algún tipo de variable, en este caso como lo que estamos haciendo es redimensionar un array lo más óptimo es guardarlo en la variable rlnArrAlturaAlumno para sobreescribirla:



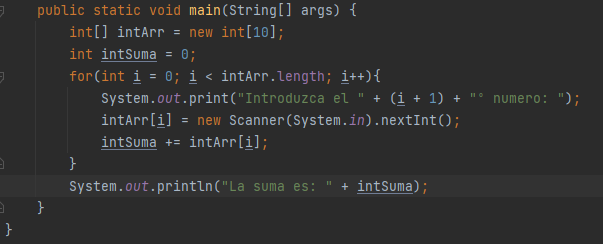
Por último, imprimimos el array en el método principal y obtenemos el mismo resultado:

**LONGITUD DE UN ARRAY**

Java contiene dentro de sus métodos básicos un método que nos permite conocer el tamaño de un array, el length cuya sintaxis es la siguiente:

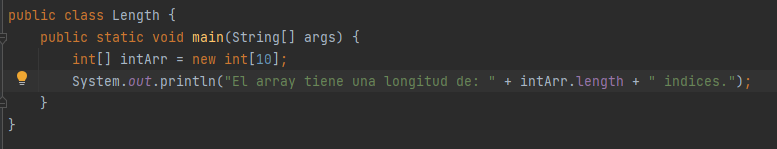
nombreArray.length

Este método nos va a solucionar muchas cosas ya que nos va a permitir a través de él fijar el final de un bucle for sin necesidad de estar pendientes del número/variable asignada a la dimensión del array.

En el siguiente ejemplo vamos a iniciar un array de enteros de una dimensión de 10 y vamos a declarar un bucle for que sirva para completar el array es decir, que repita el ciclo 10 veces.

El i < intArr.length sería igual que escribir i < 10. En este caso sólo es un bucle y en caso de modificar la dimensión la reescritura de código es mínima pero si tenemos varios bucles declarados utilizando el valor numérico (o el nombre de la variable) tendríamos que cambiar el valor en todos (aumentando el riesgo de error) por lo que es mejor usar el método length.

También nos puede servir para conocer la longitud de un array e imprimirlo a través de consola, por ejemplo:





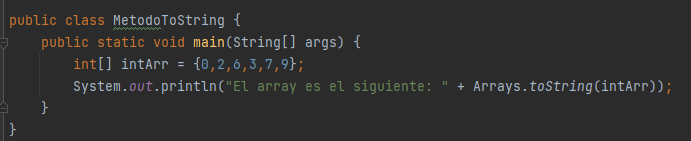
**CLASE ARRAYS: CONVERSION A CADENAS, RELLENADO Y ORDENACIÓN DE ARRAYS**

La API de Java tiene una clase que provee de diferentes métodos para trabajar con arrays.

Esta clase pertenece al paquete java.util por lo que no está dentro de los paquetes básicos de Java así que habrá que importarlo (import java.util.Arrays;).

1. Arrays.toString. Este método nos va a servir para convertir un array en una cadena de caracteres la cual podrá ser imprimible por consola. Su sintaxis es la siguiente:

Arrays.toString(nombreArray)

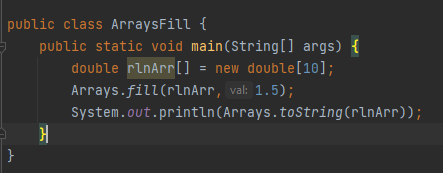
Por ejemplo:

1. Arrays.fill. Este método nos va a permitir inicializar (cambiar el valor por defecto) de los valores de un array. Se trata de un método sobrecargado por lo que nos va a permitir a través de un mismo nombre de método realizar operaciones diferentes, vamos a destacar dos:
   * Iniciar todos los indices de un array al mismo valor. Para iniciar todos los índices al mismo valor haremos uso de la siguiente sintaxis:

Arrays.fill(nombreArray, valor);

Obviamente el valor de inicialización debe ser un dato del mismo tipo que el array que hayamos creado.

Por ejemplo, vamos a crear un array donde vamos a guardar 10 numeros reales (double) y vamos a inicializar todos los índices a 1.5.



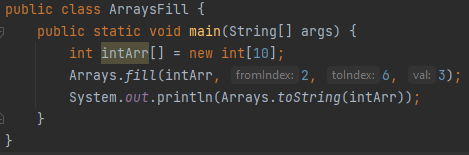
Si imprimimos este array obtendremos esto:

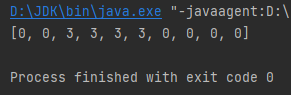


* + Inicializar solo ciertos índices. El método fill nos permite también inicializar ciertos índices del array. Para ello haremos uso de la siguiente sintaxis:

Arrays.fill(nombreArray, indiceInicio, indiceFin, valor);

Hay que tener en cuenta que el método funciona teniendo en cuenta los índices que se encuentran entre el inicio (incluido en el cambio de valor) y el fin (excluido del campo), es decir, si introducimos los índices 3 y 5, se cambiará el valor del índice 3 y del índice 4 pero no del 5.

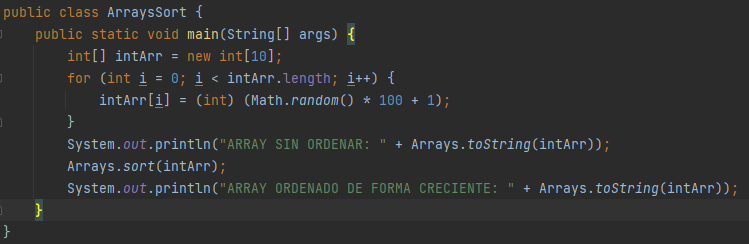
Por ejemplo, vamos a crear otro array de 10 números enteros y vamos a iniciar los índices del 2 al 5 con el valor 3.

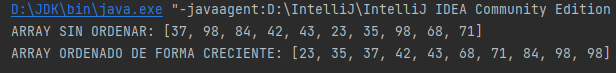


Los índices 2, 3, 4 y 5 se han inicializado en 3 (el resto permanecen en el valor 0 por defecto) pero en la llamada al método hemos tenido que introducir el parámetro de indiceFin como 6.

1. Arrays.sort. Este método nos va a servir para ordenar un array en sentido creciente. Hay que tener en cuenta que ordenar un array a través de este método consume bastantes recursos y que es una operación lenta por lo que habrá que pensar si es necesario o no realizar una ordenación. Su sintaxis es la siguiente:

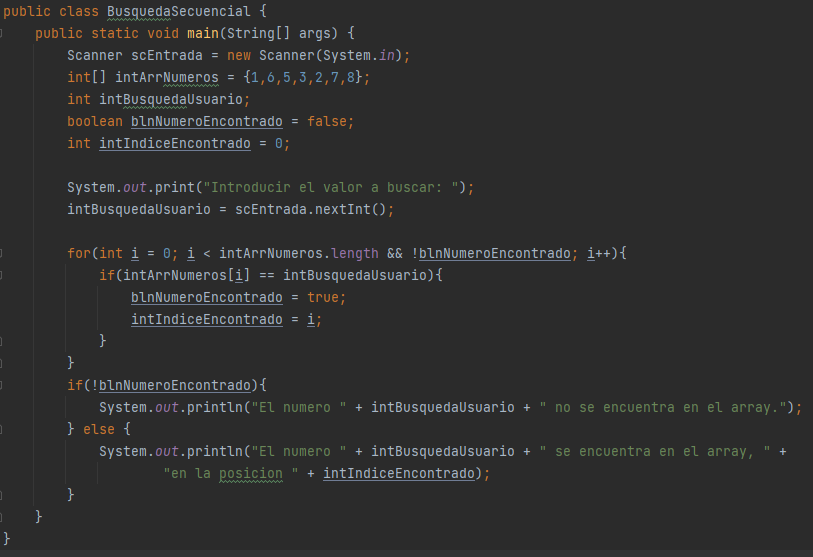
Arrays.sort(nombreArray);

Vamos a crear un array de 10 índices, los cuales se van a rellenar de forma automática dentro de un bucle for. Cuando este proceso termine se va a imprimir el array sin ordenar, luego se va a ordenar el array de forma creciente y, por último, se va a imprimir el array ordenado por consola.



**BÚSQUEDAS EN UN ARRAY NO ORDENADO**

Si no tenemos el array ordenado debemos usar una búsqueda secuencial para buscar el valor dentro de un array. Esta búsqueda consistirá en recorrer el array intentando encontrar si alguno de los valores del array es igual al valor que hemos mandado buscar.



Declaramos un array y lo iniciamos con una serie de valores. Y creamos las siguientes variables:

* intBusquedaUsuario: Nos servirá para guardar el numero que quiere buscar el usuario.
* blnNumeroEncontrado: Boolean que utilizaremos como bandera para romper el bucle for que se explicará después. Se inicializa en false.
* intIndiceEncontrado: Variable que guardará el índice del array donde se encuentra el valor que quiere encontrar el usuario. Se inicializa en 0.

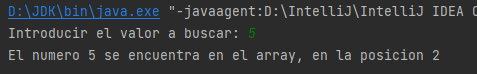
Preguntamos al usuario el valor que quiere buscar y lo guardamos en intBusquedaUsuario.

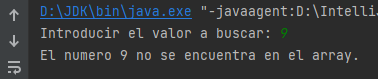
Vamos a recorrer el array para intentar buscar el valor que ha pedido el usuario para ello este bucle se deberá mantener mientras nuestro contador sea menor al tamaño del array y, además, para evitar pérdida de tiempo colocamos una bandera. Esta bandera, que inicialmente está en false, pasará a true si el número es encontrado y provocará la salida del bucle.

Por ejemplo, el usuario nos ha pedido encontrar el valor 5 en el array:

* Iteración = 0 🡪 Como el número en intArrNumeros[0] es diferente de 5, el condicional no se activa.
* Iteración = 1 🡪 Como el número en intArrNumeros[1] es diferente de 5, el condicional no se activa.
* Iteración = 2 🡪 Como el número en intArrNumeros[2] es el número 5, el condicional se activa, la variable blnNumeroEncontrado pasa a true (lo que romperá el bucle) y dentro de la variable intIndiceEncontrado se guarda el valor que tiene el contador del for en ese momento, es decir, 2.

Por último, se produce la impresión del resultado, como en este ejemplo sí que se ha encontrado el número, se informa al usuario de que se ha encontrado el valor y de la posición del mismo.



Si buscáramos un número que no se encuentra en el array, por ejemplo el 9, el for llegaría a su fin y la variable booleana se mantendría en false dando este mensaje:

**BÚSQUEDAS EN UN ARRAY ORDENADO**

En el caso de que tengamos un array ordenado podemos hacer el mismo tipo de búsqueda secuencial que en el anterior o podemos hacer uso de uno de los métodos de la clase Arrays, en este caso el método binarySearch cuya sintaxis es la siguiente:

Arrays.binarySearch(nombreArray, valorABuscar);

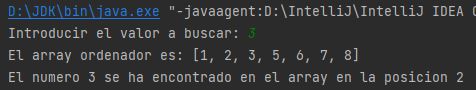
Se trata de una búsqueda dicotómica o binaria que lo primero que realiza es buscar si valorABuscar se encuentra en el elemento central de la tabla, en función de si este elemento central es mayor o menor a valorABuscar se iniciará una nueva búsqueda en la primera mitad del array (si el valorABuscar es menor que el elemento central) o en la segunda mitad (si el valorABuscar es mayor que el elemento central) este proceso se repite infinitas veces hasta que se encuentra el valor o hasta que no se puede dividir más el array (en caso de que el valor no exista).

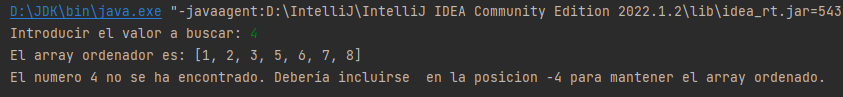
Como respuesta este método nos da un valor, el cual es:

* Positivo si encuentra el valor e informa del índice que guarda ese valor.
* Negativo si no encuentra el valor e informa del índice donde tendría que colocarse el elemento buscado para que la tabla continúe ordenada.

Vamos a usar el mismo programa que para el no ordenado, pero en este caso ordenaremos el array, para ello usaremos el método Arrays.sort().

Si buscamos un número que se encuentre en el array, por ejemplo, el 3 obtendremos este resultado:



Si buscamos un número que no exista en el array, por ejemplo, el 4 obtendremos este otro resultado:

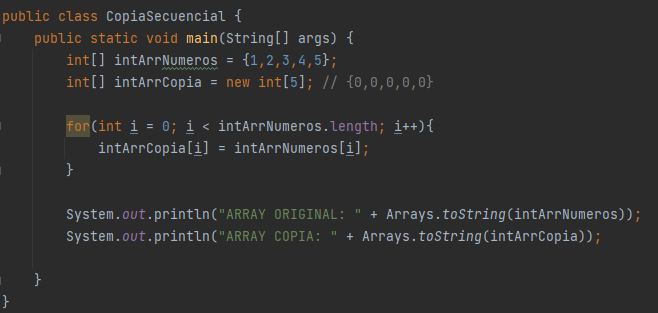
Este resultado lo que nos informa es la de posición (no índice) que debería ocupar el número, en este caso el 4 debería estar entre el 3 (posición 3 e índice 2) y el 5 (posición 4 actual e índice 3), es decir, el 4 debería ocupar la posición 4 del array y desplazar del 5 en adelante un espacio en el array.

**COPIA DE UN ARRAY**

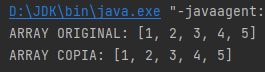
Para copiar un array podemos utilizar diferentes formas:

* Copia secuencial. Consiste en copiar uno a uno los valores de un array original en un nuevo array, para ello haremos uso de un for.

Por ejemplo, vamos a declarar e iniciar un array que vamos a llenar con 5 números y luego vamos a declarar un nuevo array de 5 posiciones (este array estará formado por 5 valores a 0).



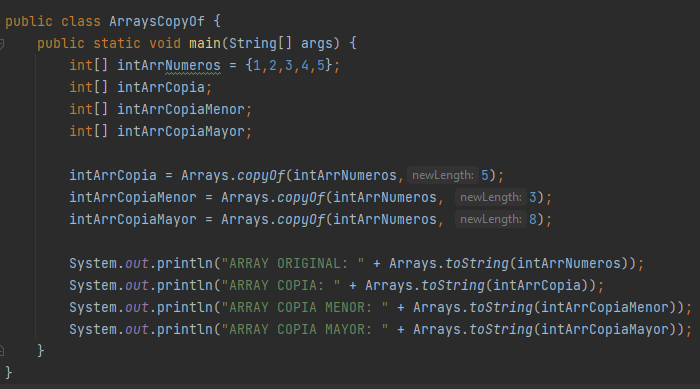
A través de un bucle for vamos copiando posición a posición los valores del intArrNumeros en intArrCopia.

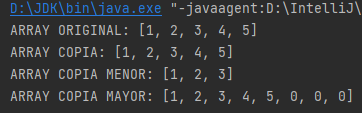
Si imprimimos los arrays obtendremos este resultado:

* Arrays.copyOf. La clase Arrays nos da un método que realiza una copia secuencial de un array y que tiene la siguiente sintaxis:

Arrays.copyOf(nombreArrayOriginal,longitudNuevoArray)

Este método nos va a permitir varias cosas:

* + Vamos a poder crear un array de la misma longitud. Se creará un array de la misma longitud (en el ejemplo será intArrCopia).
  + Vamos a poder crear un array de una longitud menor. En este caso el nuevo array sólo tendrá los valores que ocupan las posiciones de la nueva longitud (en el ejemplo será intArrCopiaMenor).
  + Vamos a poder crear un array de una longitud mayor. En esta situación el método creará un array con todos los valores del array original y, además, creará N valores nuevos hasta rellenar la nueva longitud, estos valores estarán inicializados al valor por defecto, es decir, 0 (en el ejemplo será intArrCopiaMayor).

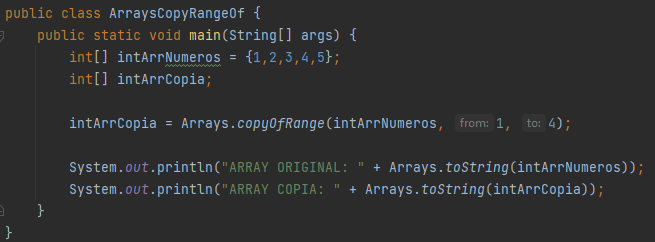
Si imprimimos por consola los arrays conseguiremos esto:

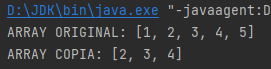
* Arrays.copyOfRange. Este otro método de la clase Arrays va a permitir copiar sólo N valores dentro de un rango. Tiene la siguiente sintaxis:

Arrays.copyOfRange(nombreArrayOriginal, indiceInicio, indiceFin)

Con este método vamos a copiar los valores desde el indiceInicio (incluido en la copia) hasta el indiceFin (excluido de la copia).

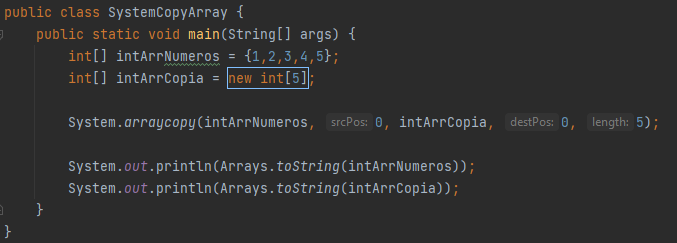
Por ejemplo, en el ejemplo anterior vamos a copiar entre el índice 1 y el índice 4, es decir, los índices 1, 2 y 3 que contienen los números 2, 3 y 4.

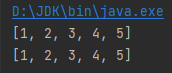


Si imprimimos por consola:

* System.arraycopy. Es un método de la clase System que permite copiar arrays. Funciona igual que el Arrays.copyOf. Tiene la siguiente sintaxis:

System.arraycopy(nombreArrayOriginal, posicionInicio, nombreNuevoArray, posición inicio, longitudNuevoArray)

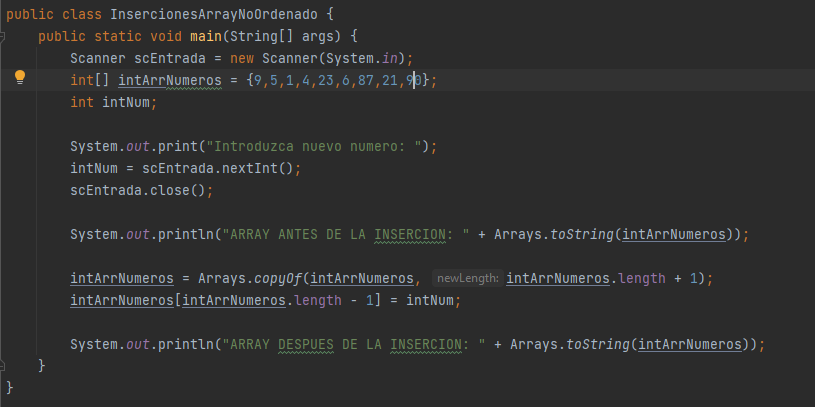




**INSERCIONES EN UN ARRAY NO ORDENADO**

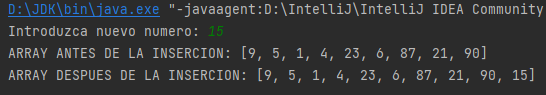
Para realizar una inserción dentro de un array no ordenado lo primero que deberemos hacer es una copia de ese array con una longitud de uno más y, a continuación, en ese nuevo índice que se ha creado (que será igual a la longitud – 1) introducir el nuevo valor.

Por ejemplo, tenemos un array de 9 posiciones y queremos llenar una 10ª posición:



Declaramos un array con 9 números y solicitamos al usuario un 10º.

1. Creamos una copia del array original con un espacio más, es decir, la nueva longitud del array será la longitud del original + 1 (intArrNumeros.length + 1).
2. En el nuevo índice que hemos creado que será el índice de mayor número es decir, el de la longitud del nuevo array – 1 (intArrNumeros.length - 1) incluimos el número del usuario.

Si imprimimos el array antes y después de la inserción conseguimos esto:

**INSERCIÓN EN UN ARRAY ORDENADO**

Para realizar una inserción dentro de un array no ordenado lo primero que deberemos hacer es un nuevo array que funcionará de auxiliar y tendrá una longitud de uno más que el array original.

A continuación, calcularemos el índice donde debe insertarse el array a través de una búsqueda binaria (Arrays.binarySearch(arrayOriginal, valorIntroducir).

Este índice podrá ser positivo (si el valor existe en el array) o negativo (si no existe). En caso de ser negativo lo tendremos que convertir en positivo y, además, restarle 1.

Haremos uso del método System.arraycopy() para hacer una copia del array original en el array auxiliar en dos partes:

* Desde el índice 0 hasta el índice que hemos calculado.

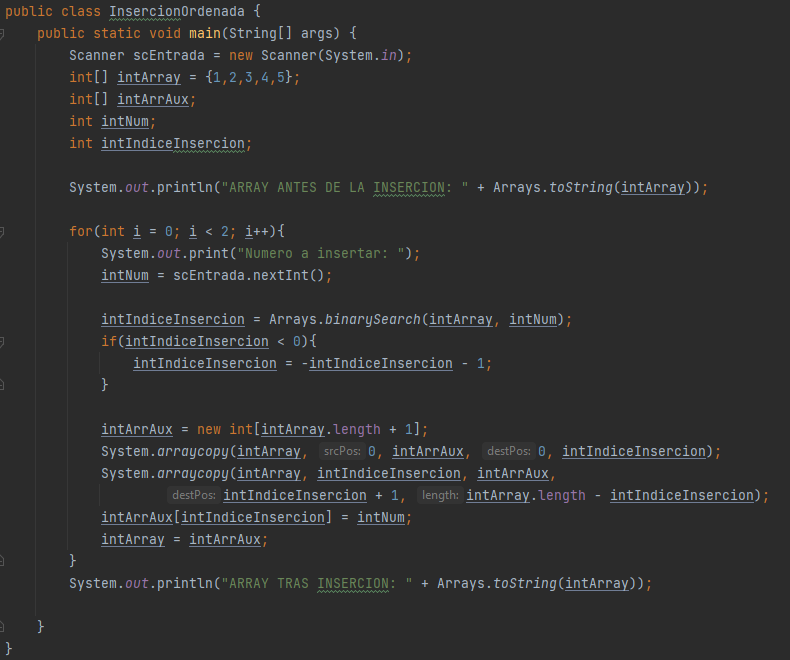
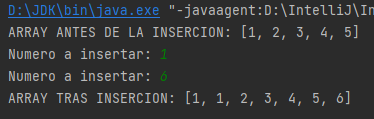
System.arraycopy(arrayOriginal, 0, arrayAuxiliar, 0, indiceInsercion);

* Desde el índice siguiente al que hemos calculado hasta el final.

System.arraycopy(arrayOriginal, indiceInsercion, arrayAuxiliar, indiceInsercion + 1, arrayOriginal.length - indiceInsercion);

Introduciremos el nuevo valor en el array en el espacio en “blanco” que tendremos y, por último, haremos un cambio de referencias.

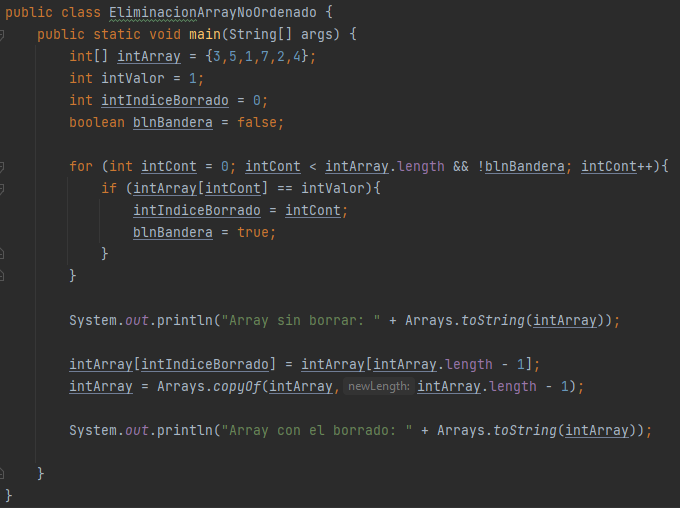
Vamos a hacer un ejemplo con 2 inserciones, una con un numero que existe en el array y otro con un número que no existe.

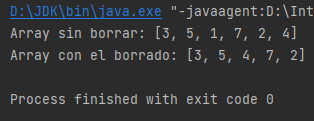


**ELIMINACIÓN EN UN ARRAY NO ORDENADO**

Para eliminar un dato de una tabla no ordenada lo que vamos a hacer es conseguir el índice de borrado, es decir, el lugar donde se encuentra el valor que queremos borrar. Para ello haremos una búsqueda secuencial del índice con un bucle for.

A continuación, en el lugar del array que ocupe ese índice de borrado copiaremos el valor del último número del array (intArray.length – 1) y haremos una copia del array con 1 menos de longitud.





Lo que ha hecho nuestro programa es borrar el 1 sobrescribiendo sobre él el valor del último índice del array (es decir, el 4) y luego ha reducido en 1 el tamaño del array.

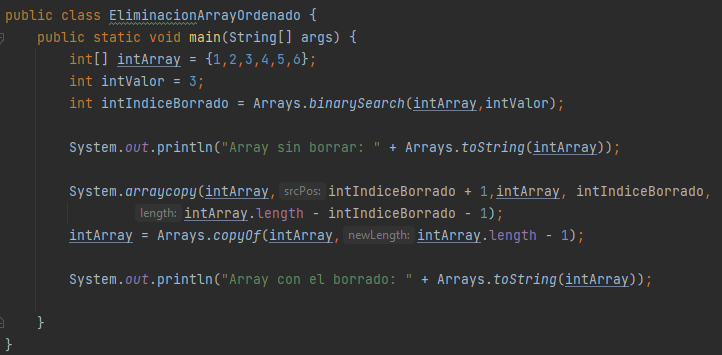
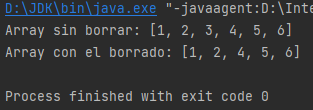
**ELIMINACIÓN EN UN ARRAY ORDENADO**

Para eliminar valores en un array ordenado vamos a volver a averiguar el índice de borrado con un binarySearch y, a continuación, vamos a hacer una sobreescritura de todos los elementos siguientes con el método arrayscopy de la clase System. Utilizando la siguiente sintaxis:

System.arraycopy(intArray, intIndiceBorrado + 1, intArray, intIndiceBorrado, intArray.length – intIndiceBorrado – 1);

Es decir, vamos a hacer una copia de intArray desde intIndiceBorrado + 1 (posición siguiente al índice de borrado) en intArray desde intIndiceBorrado y este nuevo array va a tener una longitud igual a la diferencia entre la longitud y el índice de borrado y 1 menos (el espacio que ocupaba el índice de borrado).

A continuación, haremos una copia del array con uno menos de longitud.



Lo que ha hecho nuestro programa en este caso es encontrar el índice en el que se encuentra el 3, es decir, el índice 2 y, a continuación, copiar el bloque de índices 3 – 6 en los índices 2 – 5.

Por último, se ha borrado el último valor del array recortando su longitud.

**COMPARACIÓN DE DOS TABLAS**

Para comparar dos tablas no podemos usar el operador de comparación (==) ya que si hacemos esta operación:

int[] intArray = {1,2,3};

int[] intArray2 = {1,2,3};

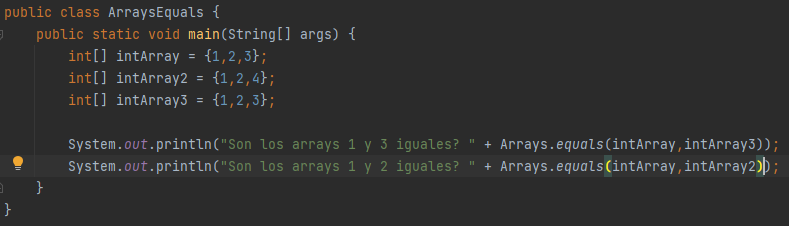
System.out.println(intArray == intArray2);

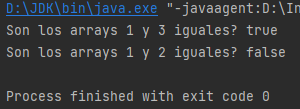
Obtendríamos un false porque aunque los valores son exactamente idénticos, lo que estaríamos comparando serían las posiciones de memoria (las referencias) que no son las mismas.

Para comparar tablas debemos de usar el método equals de la clase Arrays que tiene la siguiente sintaxis:

System.out.println(Arrays.equals(intArray, intArray2));

Esta función lo que va a realizar es una comparación valor a valor de los arrays para comprobar si son idénticos o no.



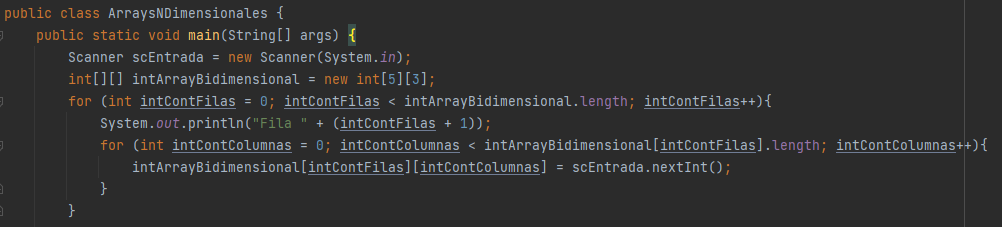


**TABLAS N-DIMENSIONALES**

Las tablas se pueden crear desde 1 hasta N dimensiones, por ejemplo, una tabla a la que llamemos int[][] intArray tendrá 2 dimensiones (cada corchete representa una dimensión).

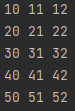
Para rellenar estas arrays necesitaremos de anidación de for. Tantos como dimensiones tenga el array.

Por ejemplo, vamos a rellenar un array de 2 dimensiones de 5 x 3 de capacidad (5 filas y 3 columnas)

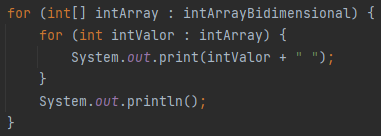


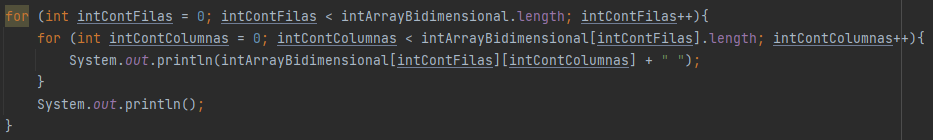
De forma general cuando rellenemos un array de 2 dimensiones utilizaremos esta misma sintaxis.

Para imprimir el array tenemos 2 opciones:

* Utilizar el método deepToString de la clase arrays que nos dará este resultado por consola:
* Utilizar un doble bucle for (o for each) para conseguir un resultado de tabla:

Si utilizamos un for each utilizaremos esta sintaxis:



Si utilizamos un for normal usaremos esta otra: