ARRAYS – (“VECTORES”|”TABLAS”)

Un **Array** en Java es un conjunto finito de datos **del mismo tipo**, que se almacenan bajo un nombre común en lo que se conoce coloquialmente como “tabla” o “vector”.

Los Arrays pueden ser de una o más dimensiones, y al declararlos Java reserva en memoria el espacio necesario para el Array. **Es fundamental entender que un ARRAY no puede aumentar/disminuir su tamaño una vez creado**. Remarcar que en java un Arrays sólo puede contener datos de un mismo tipo; int, double, char, etc
Los Arrays son estructuras que “nacen” con un tamaño prefijado y así hemos de utilizarlas. Cuando necesitemos estructuras de datos que puedan crecer/disminuir dinámicamente tendremos que recurrir a las **Colecciones**.

Ejemplo: Array para almacenar las faltas de asistencia del alumnado en el módulo de programación.

Si el número de alumnos es constante (supongamos 30), entonces podemos usar un **array** de 30 elementos de tipo int, y en cada posición del Array guardamos las faltas mensuales de cada alumno/a.

int[] faltas; *(declaración de un Array en Java --- tipo[] nombre | tipo nombre [])*

int faltas[];

2	0	5	0	· · ·	14	0	0
faltas[0]	**faltas[1]**	**faltas[2]**	**faltas[3]**		**faltas[27]**	**faltas[28]**	**faltas[29]**

Para acceder a cada elemento del array se utiliza el nombre del array y un índice entre corchetes, que indica la posición que ocupa el elemento dentro del array. El **primer elemento** del array ocupa la **posición 0. En un array de N elementos el último ocupará la posición N-1**.

Es similar a como hemos estado accediendo en algunos ejemplos a posiciones concretas dentro de un **String** con el método **.charAt[]**, pero aún más sencillo pues sólo hay que poner el nombre del array y entre corchetes la posición a la que queremos acceder, **recordando siempre comenzar en 0**.

Declaración e instanciación de Arrays.

Los Arrays son un tipo de dato especial definido en la clase Java **Array**, incluida en **java.lang** (no hay que hacer **import** para usarlos). Cada Array que declaremos será un **objeto en el HEAP**, aunque como sucedía con la clase **String** de Java, vamos a poder instanciar un **Array** de 2 formas.

- **Declaración del Array e inicialización con valores directamente:**

  int[] faltas = {0,0,0,5,6,8,0,3,11,5,0,0,0,4,5,6,8,20,1,0,0,0,2,4,5,0,0,8,6,0}

  Java reserva espacio en memoria directamente para 30 datos de tipo int, y no hace falta llamar al constructor de clase con **new.**

- **Declaración e instanciación con su correspondiente new tipo[]**

  int[] faltas = new int[30];

  Java reserva espacio para un Array de 30 elementos de tipo int y lo rellena con el valor por defecto **0**

Recorrido de un Array.

Para recorrer todos los elementos de un array lo más sencillo es utilizar un bucle **for** junto con un índice que va desde 0 hasta el tamaño del array menos 1. El tamaño del Array se puede saber de forma sencilla e inmediata usando el método **length** de la clase **Array.**

```
public class RecorridoArray {
   public static void main(String[] args) {
      int[] faltas = {0,0,0,5,6,8,0,3,11,5,0,0,0,4,5,6,8,20,1,0,0,0,2,4,5,0,0,8,6,0};
     /* observad que la condición para que el bucle "gire" es i<faltas.length. Probad a poner <= y veréis
     que salta una Exception porque nos salimos de límites. Si dejamos que el bucle gire hasta la posición
     30 nos pasamos de frenada, pues en un Array de 30 posiciones, están organizadas de 0-29        */
      for (int i=0; i<faltas.length; i++)
      {
          System.out.print(faltas[i]+" ");
      }
   }
}
```

```
Opciones
0 0 0 5 6 8 0 3 11 5 0 0 0 4 5 6 8 20 1 0 0 0 2 4 5 0 0 8 6 0
```

EJEMPLO 1: Supongamos una falta colectiva en un día de huelga de estudiantes. Podríamos modificar el contenido del Array para añadir una falta a cada alumno/a y volver a mostrar el contenido del Array.

```
public class Array1 {
   public static void main(String[] args) {
      int[] faltas = {0,0,0,5,6,8,0,3,11,5,0,0,0,4,5,6,8,20,1,0,0,0,2,4,5,0,0,8,6,0};

      for (int i=0; i<faltas.length;i++)
      {
         faltas[i]++;
         System.out.print(faltas[i]+" ");
      }
   }
}
```

SALIDA:

```
Opciones
1 1 1 6 7 9 1 4 12 6 1 1 1 5 6 7 9 21 2 1 1 1 3 5 6 1 1 9 7 1
```

EJEMPLO 2: Otro sencillo Ejemplo sobre el Array **faltas** sería recorrerlo para calcular el promedio de faltas de los alumnos del grupo

```
public class Array2 {
   public static void main(String[] args) {
      int[] faltas = {0,0,0,5,6,8,0,3,11,5,0,0,0,4,5,6,8,20,1,0,0,0,2,4,5,0,0,8,6,0};
      float totalFaltas=0;
      for (int i=0; i<faltas.length; i++)
      {
         totalFaltas+=faltas[i];  // es lo mismo que totalfaltas = totalFaltas + faltas[i];
      }
      float mediaFaltas = totalFaltas/faltas.length;

      System.out.println("La media de faltas por alumno es: " + mediaFaltas);
   }
}
```

SALIDA:

```
Opciones
La media de faltas por alumno es: 3.5666666
```

EJEMPLO 3: Un tercer ejemplo sobre el Array **faltas** podría ser recorrerlo para conocer los/as alumnos/as que más faltas tienen.

```
public class Array3 {
   public static void main(String[] args) {
      int[] faltas = {0,11,0,5,6,8,0,3,11,5,20,7,0,4,5,6,8,20,1,0,2,0,2,4,5,0,9,8,6,2};
      int maxFaltas=faltas[0];
     // variable para calcular cuál es el máximo de faltas. Se irá comparando posición a posición
      for (int i=0; i<faltas.length; i++)
      {
         if (faltas[i] > maxFaltas) {
            maxFaltas = faltas[i];
         }
      }
      // recorremos de nuevo el Array para averiguar en que posiciones se ha encontrado maxFaltas
      int tot = 0;        // vble para calcular cuantos alumnos/as tienen el máximo de faltas
      for (int i=0; i<faltas.length; i++)
      {
         if (faltas[i] == maxFaltas) {
            System.out.println ("El alumno/a en la posición " + i + " tiene "+ maxFaltas + " faltas");
            tot++;
         }
      }
      System.out.println ("En total, " + tot + " alumnos tienen " + maxFaltas + " faltas");
   }
}
```

SALIDA:

```
Opciones
El alumno/a en la posición 10 tiene 20 faltas
El alumno/a en la posición 17 tiene 20 faltas
En total, 2 alumnos tienen 20 faltas
```

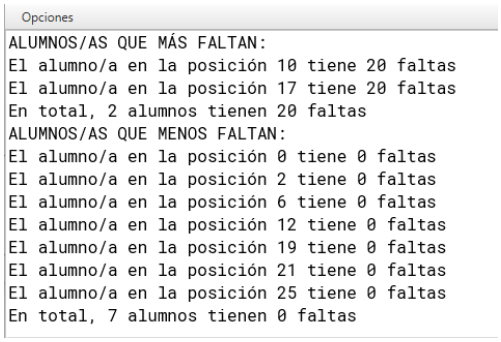
EJEMPLO 4: Un último Ejemplo. En este caso averiguaremos quienes son los alumnos/as que tienen más faltas y los que menos. También **optimizaremos un poco el código,** haciendo uso de las llamadas a las funciones del API **Math.max y Math.min**.

```
public class Array4 {
   public static void main(String[] args) {
      int[] faltas = {0,11,0,5,6,8,0,3,11,5,20,7,0,4,5,6,8,20,1,0,2,0,2,4,5,0,9,8,6,2};
      int maxFaltas, minFaltas;
      maxFaltas = minFaltas = faltas[0];
      for (int i=0; i<faltas.length; i++)
      {
         maxFaltas = Math.max(maxFaltas,faltas[i]);
         minFaltas = Math.min(minFaltas,faltas[i]);
      }
      // Nuevos bucles para averiguar en que posiciones se encuentran maxFaltas y minFaltas

      System.out.println ("ALUMNOS/AS QUE MÁS FALTAN: ");
      int totMax = 0;
      for (int i=0; i<faltas.length; i++)
      {
         if (faltas[i] == maxFaltas) {
            System.out.println ("El alumno/a en la posición " + i + " tiene "+ maxFaltas + " faltas");
            totMax++;
         }
      }
      System.out.println ("En total, " + totMax + " alumnos tienen " + maxFaltas + " faltas");

      System.out.println ("ALUMNOS/AS QUE MENOS FALTAN: ");
      int totMin = 0;
      for (int i=0; i<faltas.length; i++)
      {
         if (faltas[i] == minFaltas) {
            System.out.println ("El alumno/a en la posición " + i + " tiene "+ minFaltas + " faltas");
            totMin++;
         }
      }
      System.out.println ("En total, " + totMin + " alumnos tienen " + minFaltas + " faltas");
   }
}
```

SALIDA:



Bucle especial FOR EACH para Arrays.

- Es una simplificación del clásico bucle for para el caso en que necesitamos recorrer un Array en modo consulta o listado. **No permite hacer modificaciones sobre los elementos del Array.**
- Permite recorrer uno a uno y hacía adelante los elementos del Array de una forma muy sencilla, pero al ser un bucle que no usa índice, y no permite modificar el contenido del Array, **sus usos están limitados a labores de listado-consulta-cálculos de/con los valores del Array.**

```
for (int f:faltas) {
    System.out.println(f);    // Imprimiría directamente todas las posiciones del Array notas
}
```

Equivalente a:

```
for (int i=0;i<faltas.length;i++)
{
    System.out.println(faltas[i]);
}
```

El bucle FOR EACH **no sólo** se usará en Arrays, sino también en **Colecciones**, y con otras variantes, por lo que siempre que podamos usarlo debemos hacerlo para acostumbrarnos.

APLICACIÓN DE FOR EACH A LOS EJEMPLOS 1-4

- **En el Ejemplo inicial para listar simplemente los 30 valores del Array faltas:**

```
for (int f:faltas)
{
   System.out.print( f +" ");
}
```

- **En el Ejemplo 1 para ponerles una falta por huelga a todos los alumnos/as.**

  No se puede usar For Each para MODIFICACIONES. En este caso el Ejemplo exige modificar los valores del Array para sumarle una falta a cada posición. Deberíamos hacerlo con un bucle **for** normal:

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

- **En el Ejemplo 2 para calcular la media de faltas de todo el alumnado:**

```
int totalFaltas=0;
for (int f:faltas)
{
   totalFaltas+=f;
}
float mediaFaltas = totalFaltas/faltas.length;
```

- **En los Ejemplos 3 y 4 para el primer bucle que recorre el array buscando el máximo/mínimo número de faltas.**

```
for (int f:faltas)
{
   System.out.print (f + " ");
   maxFaltas = Math.max(maxFaltas,f);
   minFaltas = Math.min(minFaltas,f);
}
```

Como se ha comentado, el bucle FOR EACH no utiliza índice para recorrer el Array, pero eso no impide que podamos construir nuestro propio índice. También hay métodos avanzados para poder obtener el índice de una posición de Array en un recorrido con FOR EACH, pero hay que utilizar **clases más avanzadas de Java** que no son de nuestro interés en este momento.

ARRAYS DE MÁS DE 1 DIMENSIÓN (MATRICES/TABLAS BIDIMENSIONALES).

Un Array bidimensional utiliza dos índices para localizar cada elemento. Podemos "visualizar" un Array de 2 dimensiones como una **TABLA** en la que los datos quedan organizados en **filas** y **columnas**.

Vamos a modificar el Array con el que venimos trabajando **faltas**[], para que nos permita almacenar las faltas del alumnado de forma independiente CADA MES. En realidad necesitaría 1 Array de 30 elementos para cada mes (9 Arrays en total). Algo así:

```
int  faltasOctubre[] = new int[30];
int faltasNoviembre[] = new int[30];

…
int faltasJunio[] = new int[30];
```

Manejar 9 Arrays independientes para algo así es inadecuado. Se debe usar una única estructura Bidimensional:

int faltasCurso[][] = new int[9][30]

IMPORTANTE: El primer índice siempre será el número de "filas" y el segundo el de "columnas".

EJEMPLO 1 ARRAYS BIDIMENSIONALES. Recorrido y salida por pantalla.

```
public class ArraysBidim1 {
   public static void main(String[] args) {
      int[][]faltasCurso ={
                        {1,0,0,5,6,8,0,3,11,5,0,12,0,4,5,6,8,20,1,0,9,0,2,4,5,0,0,8,6,0},
                        {0,0,2,5,6,5,0,3,0,5,0,0,0,4,5,6,8,20,1,0,0,0,2,4,5,3,0,8,6,0},
                        {0,2,0,5,6,8,0,3,6,5,0,5,0,4,5,6,8,20,1,0,12,0,2,4,5,0,0,8,6,0},
                        {0,0,0,5,6,8,0,3,12,5,0,0,0,4,5,6,8,20,1,0,3,0,2,4,5,0,0,8,8,0},
                        {0,33,0,5,6,8,0,3,6,5,0,4,0,4,5,6,8,20,1,0,5,0,2,4,5,0,7,8,4,0},
                        {0,0,0,5,6,11,0,3,14,5,0,7,0,4,5,6,8,20,1,0,14,0,2,4,5,0,0,8,6,0},
                        {0,7,0,5,6,13,0,3,12,5,0,0,0,4,5,6,8,20,1,0,8,0,2,4,5,12,0,5,6,0},
                        {0,0,0,5,6,8,0,3,9,5,0,21,0,4,5,6,4,11,19,0,5,0,2,4,5,0,0,8,6,0},
                        {0,9,0,5,6,8,0,3,11,5,0,0,0,4,5,6,8,20,1,0,0,0,2,7,5,4,0,8,6,0},
                        };
      for (int f=0; f<9; f++)   //aprenderemos a sustituir ese 9 con el método length
      {
         for (int c=0; c< 30; c++)    //aprenderemos a sustituir ese 30 con el método length
         {
             System.out.print(faltasCurso[f][c]+" ");
         }
         System.out.println();
      }
   }
}
```

Uso de length en arrays de 2 dimensiones.

Para no poner los valores **9 y 30** debemos usar el método **.length,** que nos devuelve el número de elementos de un Array, pero ahora el Array tiene 2 dimensiones.

faltasCurso.length me devolverá el número de "filas". **9 en este caso.**

Para saber el número de elementos que hay en cada fila hemos de poner:

faltasCurso[0].length
faltasCurso[1].length
…
faltasCurso[8].length

… esto nos ayudará mucho**,** porque en **Java l**os Arrays bidimensionales pueden tener "filas" de distinto tamaño, y así recurrimos al método length[] para plantear bien los bucles de recorrido del Array sin temor a salirnos de límites y que nos salte una **Exception.**

EQUIVALENTE AL EJEMPLO ANTERIOR, PERO MUCHO MEJOR CODIFICADO AL NO USAR VALORES CONCRETOS:

```
for (int f=0; f<faltasCurso.length; f++)
{
   for (int c=0; c<faltasCurso[f].length; c++)
   {
        System.out.print(faltasCurso[f][c]+" ");
   }
   System.out.println();
}
```

TAMBIÉN podemos iterar ese Array Bidimensional muy fácilmente con un **FOR EACH** anidado.

```
for (int fila[]:faltasCurso){
    for (int falta:fila){
      System.out.print(falta + " ");
    }
    System.out.println();
}
```

Sencillo, aunque no hay que olvidar las limitaciones de los bucles FOR EACH:

- No permite modificar los valores del Array.
- Sin uso de clases avanzadas no tenemos acceso al índice (posición) de los elementos.
- Sólo se puede recorrer el ARRAY hacia "adelante" y de 1 en 1.

EJEMPLO 2. ARRAYS BIDIMENSIONALES. Media de faltas mensual.

Si deseamos conocer la media de faltas de cada mes podemos ampliar el **Ejemplo 2 de ARRAYS** y pasarlo a 2 dimensiones. No es difícil, sólo hay que añadir un bucle externo que mueva las **9 filas (meses)**

```
public class ArraysBidim2 {
   public static void main(String[] args) {
      int[][]faltasCurso ={
                  {1,0,0,5,6,8,0,3,11,5,0,12,0,4,5,6,8,20,1,0,9,0,2,4,5,0,0,8,6,0},
                  {0,0,2,5,6,5,0,3,0,5,0,0,0,4,5,6,8,20,1,0,0,0,2,4,5,3,0,8,6,0},
                  {0,2,0,5,6,8,0,3,6,5,0,5,0,4,5,6,8,20,1,0,12,0,2,4,5,0,0,8,6,0},
                  {0,0,0,5,6,8,0,3,12,5,0,0,0,4,5,6,8,20,1,0,3,0,2,4,5,0,0,8,8,0},
                  {0,33,0,5,6,8,0,3,6,5,0,4,0,4,5,6,8,20,1,0,5,0,2,4,5,0,7,8,4,0},
                  {0,0,0,5,6,11,0,3,14,5,0,7,0,4,5,6,8,20,1,0,14,0,2,4,5,0,0,8,6,0},
                  {0,7,0,5,6,13,0,3,12,5,0,0,0,4,5,6,8,20,1,0,8,0,2,4,5,12,0,5,6,0},
                  {0,0,0,5,6,8,0,3,9,5,0,21,0,4,5,6,4,11,19,0,5,0,2,4,5,0,0,8,6,0},
                  {0,9,0,5,6,8,0,3,11,5,0,0,0,4,5,6,8,20,1,0,0,0,2,7,5,4,0,8,6,0},
                  };
      for (int f=0; f< faltasCurso.length; f++)
      {
         float totalFaltas=0;
         for (int c=0; c< faltasCurso[f].length; c++)
         {
            totalFaltas+=faltasCurso[f][c];
         }
         float mediaFaltas = totalFaltas/ faltasCurso[f].length;

        System.out.println("La media de faltas por alumno en el mes " + f + " es: " + mediaFaltas);
      }
   }
}
```

… si queremos que muestre el mes correspondiente en letra:

```
      String mes = new String();
      for (int f=0; f< faltasCurso.length; f++)
      {
         float totalFaltas=0;
         for (int c=0; c< faltasCurso[f].length; c++)
         {
            totalFaltas+=faltasCurso[f][c];
         }
         float mediaFaltas = totalFaltas/ faltasCurso[f].length;

         switch (f){
         case 0: mes = "OCTUBRE";
               break;
         case 1: mes = "NOVIEMBRE";
               break;
         case 2: mes = "DICIEMBRE";
               break;
         case 3: mes = "ENERO";
```

```
            break;
        case 4: mes = "FEBRERO";
            break;
        case 5: mes = "MARZO";
            break;
        case 6: mes = "ABRIL";
            break;
        case 7: mes = "MAYO";
            break;
        case 8: mes = "JUNIO";
            break;
        }
        System.out.println("La media de faltas por alumno en el mes " + mes + " es: " + mediaFaltas);
    }
```

… una versión mejorada y resumida sin switch, empleando un Array unidimensional para los 9 meses:

```
public class ArraysBidim3 {
   public static void main(String[] args) {
      int[][]faltasCurso ={
                  {1,0,0,5,6,8,0,3,11,5,0,12,0,4,5,6,8,20,1,0,9,0,2,4,5,0,0,8,6,0},
                  {0,0,2,5,6,5,0,3,0,5,0,0,0,4,5,6,8,20,1,0,0,0,2,4,5,3,0,8,6,0},
                  {0,2,0,5,6,8,0,3,6,5,0,5,0,4,5,6,8,20,1,0,12,0,2,4,5,0,0,8,6,0},
                  {0,0,0,5,6,8,0,3,12,5,0,0,0,4,5,6,8,20,1,0,3,0,2,4,5,0,0,8,8,0},
                  {0,33,0,5,6,8,0,3,6,5,0,4,0,4,5,6,8,20,1,0,5,0,2,4,5,0,7,8,4,0},
                  {0,0,0,5,6,11,0,3,14,5,0,7,0,4,5,6,8,20,1,0,14,0,2,4,5,0,0,8,6,0},
                  {0,7,0,5,6,13,0,3,12,5,0,0,0,4,5,6,8,20,1,0,8,0,2,4,5,12,0,5,6,0},
                  {0,0,0,5,6,8,0,3,9,5,0,21,0,4,5,6,4,11,19,0,5,0,2,4,5,0,0,8,6,0},
                  {0,9,0,5,6,8,0,3,11,5,0,0,0,4,5,6,8,20,1,0,0,0,2,7,5,4,0,8,6,0},
                  };

      String[] mes = {"OCTUBRE","NOVIEMBRE","DICIEMBRE","ENERO","FEBRERO","MARZO","ABRIL","MAYO","JUNIO"};

      for (int f=0; f< faltasCurso.length; f++)
      {
         float totalFaltas=0;
         for (int c=0; c < faltasCurso[f].length; c++)
         {
            totalFaltas+=faltasCurso[f][c];
         }
         float mediaFaltas = totalFaltas/ faltasCurso[f].length;
         System.out.println("La media de faltas por alumno en el mes " + mes[f] + " es: " + mediaFaltas);
      }
   }
}
```

CONCLUSIONES:

En este documento se introduce el tipo de dato **Array** en un contexto formativo de **iniciación a la programación. Se** propone inicializarlos y recorrerlos manualmente, para afianzar y practicar a su vez el manejo de bucles.

Java proporciona una clase específica para el **manejo avanzado** de Arrays, en la que encontraremos interesantes **métodos estáticos**. Es la clase **java.util.Arrays .** Necesitamos hacer **import java.util.Arrays** para poder usarla.

El trabajo con Arrays se simplifica y potencia usando las clases que proporciona el API. **Este es un camino que todo programador/a debe seguir**. Entender los conceptos básicos, en este caso cómo son las estructuras de datos de tipo Array, e ir profundizando poco a poco y cada vez más en sus posibilidades y herramientas. Siempre habrá algo en el API de Java o en librerías de terceros que potenciará nuestras competencias como programadores/as y **nos ayudará a lograr codificar problemas más complejos, pero de forma más sencilla**

Métodos **java.util.Arrays (Más información en https://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html)**

binarySearch	Busca un valor que le pasamos por parámetro, devuelve su posición. Debe estar ordenado.	Un array y un valor. Los dos del mismo tipo. Estos pueden ser un byte, char, double, float, int, long, short u objecto.	Devuelve un int
copyOf	Copia un array y lo devuelve en un nuevo array.	Un array y la longitud. Si se pasa del tamaño del array original, rellena los con ceros las posiciones sobrantes. Estos pueden ser un byte, char, double, float, int, long, short u objecto.	Devuelve array del mismo tipo
copyOfRange	Copia un array y lo devuelve en un nuevo array. Le indicamos la posición de origen y de destino.	Un array, posición origen y destino. Estos pueden ser un byte, char, double, float, int, long, short u objecto.	Devuelve array del mismo tipo
equals	Indica si dos arrays son iguales.	Dos arrays del mismo tipo.	Devuelve true o false
fill	Rellena un array con un valor que le indiquemos como parámetro.	Un array y el valor a rellenar. Estos pueden ser un byte, char, double, float, int, long, short u objeto.	No devuelve nada
sort	Ordena el array de Menor a Mayor.	Un array. Estos pueden ser un byte, char, double, float, int, long, short u objecto.	No devuelve nada
toString	Muestra en forma de String el contenido de un array pasado como parámetros	Un array. Estos pueden ser un byte, char, double, float, int, long, short u objecto.	Devuelve String con el contenido del array.

Ejemplo uso métodos clase java.util.Arrays.

```
import java.util.Arrays;
public class EjemploUtilArrays {

   public static void main(String[] args) {
      int num[]={5,49,52,25,87,19,30,7,44,1};

      int num2[]=Arrays.copyOf(num, 4);
      //Copia el array num hasta la quinta posicion(este ultimo no incluido), devuelve un array
      for (int n:num2){
         System.out.print(n+" ");
      }
      System.out.println();

      int num3[]=Arrays.copyOfRange(num, 2, 6);
      //Copia el array num de la tercera hasta la octava posicion, devuelve un array
      for (int n:num3){
         System.out.print(n+" ");
      }
      System.out.println();

      //Compara si num1 y num2 no son iguales
      System.out.println("Es num1 igual a num2 ?: "+Arrays.equals(num, num2));

      System.out.println("Metodo fill");
      Arrays.fill(num3, 5);
      for (int n:num3){
         System.out.print(n+" ");
      }
      System.out.println();

      System.out.println("Metodo toString");
      System.out.println(Arrays.toString(num));

      Arrays.sort(num);
      for (int n:num){
         System.out.print(n+" ");
      }

      System.out.println("la posición que ocupa el 25 es: " + Arrays.binarySearch(num, 25));
   }
}
```

STREAMS y ARRAYS.

Avanzando y profundizando un poco más, cualquier programador/a Java acabará encontrando en su camino el tipo de dato **Stream,** que aparece **a partir de Java 8**.

El tipo **Stream**, y la posibilidad de manejarlo con **expresiones Lambda (programación funcional)** es una herramienta muy potente para los programadores/as avanzados/as. No es el momento de pisar aún esos terrenos, pero se deja la "pista" por si en adelante algún alumno/a desea profundizar e investigar por su cuenta.

Tan sólo apuntar, que aunque los Streams están más orientados al trabajo con Colecciones Java, **también es posible convertir un Array en Stream** y manejarlo como tal. Con todas las posibilidades que esto aporta. Después siempre podemos devolverlo a su formato original de Array. **Veamos un sencillo ejemplo:**

```
import java.util.Arrays;

public class ArrayStream{
  public static void main(String[] args) {

    String [] alumnos= {"Luis","Angela", "Tania","Ana","Mar","Lucas","Oscar"};

    Arrays.stream(alumnos).forEach(System.out::println);
  }
}
```

La ventaja de utilizar un Stream es que con **programación funcional** creamos un flujo de trabajo y podemos añadir operaciones intermedias como puede ser **filter** para filtrar determinados nombres del array:

```
public class ArrayStream{
  public static void main(String[] args) {

    String [] alumnos= {"Luis","Angela", "Tania","Ana","Mar","Lucas","Oscar"};

    Arrays.stream(alumnos).filter(n->n.length()<4).forEach(System.out::println);
  }
}
```

Se pueden hacer cosas muy potentes con Arrays y Colecciones convertidas a Stream, pero ahora posiblemente sea muy pronto para explorar esos terrenos.