Arrays unidimensionales en Java

Concepto de Array o Arreglo:

Un **array** es una colección finita de datos del mismo tipo, que se almacenan en posiciones consecutivas de memoria y reciben un nombre común.

Por ejemplo, supongamos que queremos guardar las notas de los 20 alumnos de una clase. Para almacenar las notas utilizaremos un array de 20 elementos de tipo double y en cada elemento del array guardaremos la nota de cada alumno

Podemos representar gráficamente el array de notas de la siguiente forma:

Array **notas**:

8.50	**6.35**	**5.75**	**8.50**	**· · ·**	**3.75**	**6.00**	**7.40**
notas[0]	notas[1]	notas[2]	notas[3]	· · ·	notas[17]	notas[18]	notas[19]

Para acceder a cada elemento del array se utiliza el nombre del array y un índice que indica la posición que ocupa el elemento dentro del array.

El índice se escribe entre corchetes.

El **primer elemento** del array ocupa la **posición 0**, el segundo la posición 1, etc. **En un array de N elementos el último ocupará la posición N-1**.

En el ejemplo anterior, notas[0] contiene la nota del primer alumno y notas[19] contiene la del último

Los índices deben ser enteros no negativos.

1. CREAR ARRAYS UNIDIMENSIONALES

Para crear un array se deben realizar dos operaciones:

- Declaración
- Instanciación

Declarar de un array

En la declaración se crea la **referencia** al array.

La referencia será el nombre con el que manejaremos el array en el programa.

Se debe indicar el nombre del array y el tipo de datos que contendrá.

De forma general un array unidimensional se puede declarar en java de cualquiera de estas dos formas:

```
tipo [] nombreArray;
```

tipo nombreArray[];

tipo: indica el tipo de datos que contendrá. Un array puede contener elementos de tipo básico o referencias a objetos.

nombreArray: es la referencia al array. Es el nombre que se usará en el programa para manejarlo.

Por ejemplo:

int [] ventas; //array de datos de tipo int llamado ventas

double [] temperaturas; //array de datos de tipo double llamado temperaturas

String [] nombres; //array de datos de tipo String llamado nombres

Instanciar un array

Mediante la instanciación se reserva un bloque de memoria para almacenar todos los elementos del array.

La dirección donde comienza el bloque de memoria donde se almacenará el array se asigna al nombre del array. Esto quiere decir que el nombre del array contiene la dirección de memoria donde se encuentra.

De forma general un array se instancia así:

nombreArray = new tipo[tamaño];

- nombreArray: es el nombre creado en la declaración.
- tipo: indica el tipo de datos que contiene.
- tamaño: es el número de elementos del array. Debe ser una expresión entera positiva. El tamaño del array no se puede modificar durante la ejecución del programa.
- new: operador para crear objetos. Mediante new se asigna la memoria necesaria para ubicar el objeto. Java implementa los arrays como objetos.

Por ejemplo:

ventas = new int[5]; //se reserva memoria para 5 enteros y

//se asigna la dirección de inicio del array a ventas.

Lo normal es que la declaración y la instanciación se hagan en una sola instrucción:

tipo [] nombreArray = new tipo[tamaño];

Por ejemplo:

int [] ventas = new int[5];

El tamaño del array también se puede indicar durante la ejecución del programa, es decir, en tiempo de ejecución se puede pedir por teclado el tamaño del array y crearlo:

```
Scanner sc = new Scanner(System.in);

System.out.print("Número de elementos del array: ");

int numeroElementos = sc.nextInt();

int [] ventas = new int[numeroElementos];
```

Si no hay memoria suficiente para crear el array, new lanza una excepción java.lang.OutOfMemoryError.

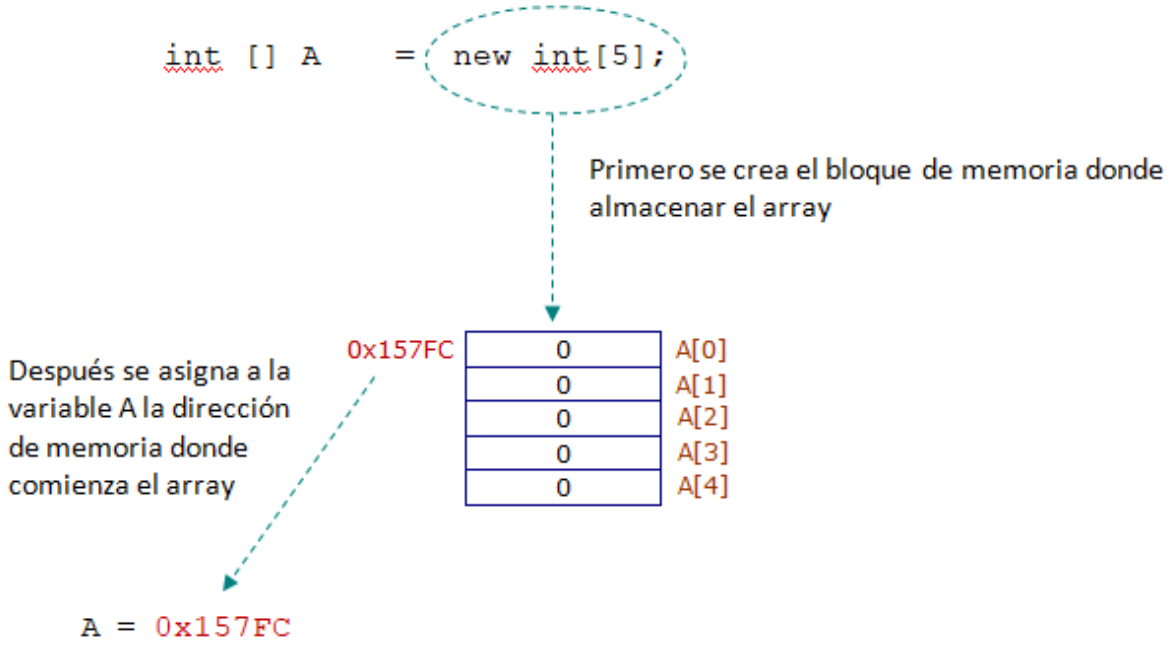
Diferencia entre la referencia y el contenido del array

Debe quedar clara la diferencia entre el nombre del array y el contenido del array.

Cuando por ejemplo se ejecuta una instrucción para crear un array de enteros llamado A:

```
int [] A = new int [5];
```

se realizan dos operaciones: primero se crea un bloque de memoria donde almacenar el array de 5 enteros y después se asigna la dirección de inicio del bloque de memoria, también llamado referencia del array, a la variable A.



El nombre del array contiene la dirección de inicio del bloque de memoria donde se encuentra el contenido del array.

2. INICIALIZAR ARRAYS UNIDIMENSIONALES

Un array es un objeto, por lo tanto, cuando se crea, a sus elementos se les asigna automáticamente un valor inicial:

Valores iniciales por defecto para un array en java:

- 0 para arrays numéricos
- '\u0000' (carácter nulo) para arrays de caracteres
- false para arrays booleanos
- null para arrays de String y de referencias a objetos.

También podemos dar otros valores iniciales al array cuando se crea.

Los valores iniciales de un array se escriben entre llaves separados por comas.

Los valores iniciales de un array deben aparecer en el orden en que serán asignados a los elementos del array.

El número de valores determina el tamaño del array.

Por ejemplo:

double [] notas = {6.7, 7.5, 5.3, 8.75, 3.6, 6.5};

se declara el array notas de tipo double, se reserva memoria para 6 elementos y se les asignan esos valores iniciales.

Ejemplos:

//creación de un array de 4 elementos booleanos

boolean [] resultados = {true,false,true,false};

//creación de un array de 7 elementos de tipo String

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

3. ACCEDER A LOS ELEMENTOS DE UN ARRAY

Para acceder a cada elemento del array se utiliza el nombre del array y un índice que indica la posición que ocupa el elemento dentro del array.

El índice se escribe entre corchetes.

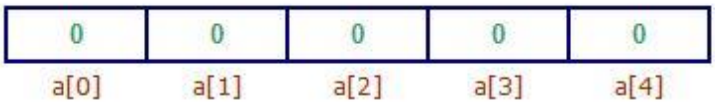
Se puede utilizar como índice un valor entero, una variable de tipo entero o una expresión de tipo entero.

El **primer elemento** del array ocupa la **posición 0**, el segundo la posición 1, etc. **En un array de N elementos el último ocupará la posición N-1**.

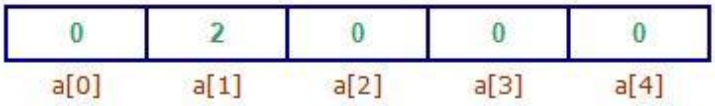
Un elemento de un array se puede utilizar igual que cualquier otra variable. Se puede hacer con ellos las mismas operaciones que se pueden hacer con el resto de variables (incremento, decremento, operaciones aritméticas, comparaciones, etc).

Por ejemplo:

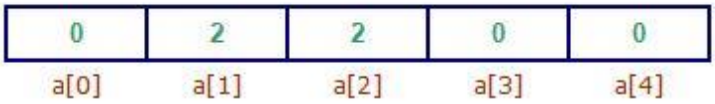
```
int m = 5;
int [] a = new int[5];
```



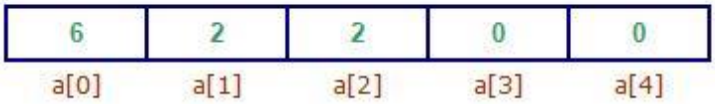

```
a[1] = 2;
```



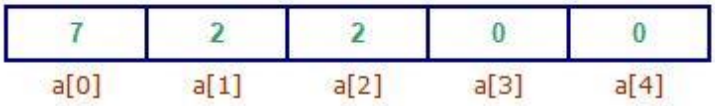

```
a[2] = a[1];
```



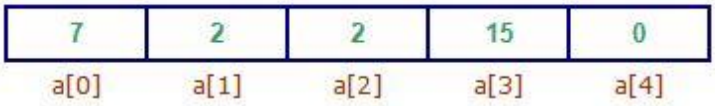

```
a[0] = a[1] + a[2] + 2;
```




```
a[0]++;
```




```
int m = 5;
a[3] = m + 10;
```



Si se intenta acceder a un elemento que está fuera de los límites del array (un elemento con índice negativo o con un índice mayor que el que corresponde al último elemento del array) **el compilador no avisa del error**. El error se producirá durante la ejecución. En ese caso se lanza una excepción java.lang.ArrayIndexOutOfBoundsException.

Se puede saber el número de elementos del array mediante el atributo **length**.

Podemos utilizar length para comprobar el rango del array y evitar errores de acceso.

Por ejemplo, introducimos por teclado un valor y la posición del array donde lo vamos a guardar:

```
Scanner sc = new Scanner(System.in);

int i, valor;

int [] a = new int[10];

System.out.print("Posición: ");

i = sc.nextInt(); //pedimos una posición del array

if (i >= 0 && i < a.length) {  //si la posición introducida está dentro de los límites del array
   System.out.print("Valor: ");
   valor = sc.nextInt();  //pedimos el valor
```

```
   a[i] = valor;
}else{
   System.out.println("Posición no válida");
}
```

4. RECORRER UN ARRAY UNIDIMENSIONAL

Para recorrer un array se utiliza una instrucción iterativa, normalmente una instrucción for, aunque también puede hacerse con while o do..while, utilizando una variable entera como índice que tomará valores desde el primer elemento al último o desde el último al primero.

Por ejemplo, el siguiente fragmento de programa Java declara un array de 7 elementos de tipo double y le asigna valores iniciales. A continuación recorre el array, utilizando la instrucción for, para mostrar por pantalla el contenido del array.

```
double[] notas = {2.3, 8.5, 3.2, 9.5, 4, 5.5, 7.0}; //array de 7 elementos
for (int i = 0; i < 7; i++) {
    System.out.print(notas[i] + " "); //se muestra cada elemento del array
}
```

Para evitar errores de acceso al array es recomendable **utilizar length para recorrer el array completo**.

Por ejemplo:

```
double[] notas = {2.3, 8.5, 3.2, 9.5, 4, 5.5, 7.0}; //array de 7 elementos
for (int i = 0; i < notas.length; i++) {
    System.out.print(notas[i] + " "); //se muestra cada elemento del array
}
```

Ejemplo de recorrido de un array unidimensional en java:

Programa que lee por teclado la nota de los alumnos de una clase y calcula la nota media del grupo. También muestra los alumnos con notas superiores a la media. El número de alumnos se lee por teclado.

Este programa crea un array de elementos de tipo double que contendrá las notas de los alumnos. El tamaño del array será el número de alumnos de la clase.

Se realizan **3 recorridos** sobre el array, el primero para asignar a cada elemento las notas introducidas por teclado, el segundo para sumarlas y el tercero para mostrar los alumnos con notas superiores a la media.

```
import java.util.Scanner;
```

```
public class Recorrido2 {

   public static void main(String[] args) {

      Scanner sc = new Scanner(System.in);
      int numAlum, i;
      double suma = 0, media;

      do {
         System.out.print("Número de alumnos de la clase: ");
         numAlum = sc.nextInt();
      } while (numAlum <= 0);

      double[] notas = new double[numAlum]; //se crea el array

      // Entrada de datos. Se asigna a cada elemento del array
      // la nota introducida por teclado
      for (i = 0; i < notas.length; i++) {
         System.out.print("Alumno " + (i + 1) + " Nota final: ");
         notas[i] = sc.nextDouble();
      }

      // Sumar todas las notas
      for (i = 0; i < notas.length; i++) {
         suma = suma + notas[i];
      }

      // Calcular la media
```

```
        media = suma / notas.length;


        // Mostrar la media
        System.out.printf("Nota media del curso: %.2f %n", media);


        // Mostrar los valores superiores a la media
        System.out.println("Listado de notas superiores a la media: ");
        for (i = 0; i < notas.length; i++) {
          if (notas[i] > media) {
            System.out.println("Alumno numero " + (i + 1)+ " Nota final: " + notas[i]);
          }
        }
    }
}
```

Recorrer un Array en java con foreach. Bucle for para colecciones

A partir de java 5 se incorpora una instrucción for mejorada para recorrer arrays y contenedores en general.

Permite acceder secuencialmente a cada elemento del array.

La sintaxis general es:

```
for(tipo nombreDeVariable : nombreArray){
  ......
}
```

tipo: indica el tipo de datos que contiene el array.

nombreDeVariable: variable a la que en cada iteración se le asigna el valor de cada elemento del array. Esta definida dentro del for por lo que solo es accesible dentro de él.

nombreArray: es el nombre del array que vamos a recorrer.

Mediante este bucle solo podemos acceder a los elementos del array. No podemos hacer modificaciones en su contenido.

Ejemplo: El siguiente programa crea un array temperatura de 10 elementos. Lee por teclado los valores y a continuación los muestra por pantalla usando un bucle foreach.

```
import java.util.Scanner;

public class Recorrerforeach1 {

   public static void main(String[] args) {

      Scanner sc = new Scanner(System.in);
      double [] temperatura = new double[10];
      int i;

      for(i = 0; i < temperatura.length; i++){
         System.out.print("Elemento " + i + ": ");
         temperatura[i] = sc.nextDouble();
      }

      for(double t: temperatura){
         System.out.print(t + " ");
      }
      System.out.println();
   }
}
```