Arrays

An *array* is a collection of variables of the same type, referred to by a common name.

One-Dimensional Arrays

To declare a one-dimensional array (o matriz), you can use this general form:

type array-name[ ] = new *type*[*size*];

El tipo de elemento determina el tipo de datos de cada elemento contenido en la matriz o array. El número de elementos que la matriz tendrá se determina por tamaño.
Dado que las matrices se implementan como objetos, la creación de una matriz es un proceso de dos pasos.
Primero, declaras una **variable** de tipo **referencia a un** array (Por ejemplo una v**ariable** int almacena un valor entero las **variables** de tipo **referencia** a objetos en cambio almacenan direcciones y no valores directamente).
Segundo, reservas memoria para el array, assigning a reference to that memory to the array variable. Thus, arrays in Java are dynamically allocated using the **new** operator.

Here is an example. The following creates an **int** array of 10 elements and links it to an array reference variable named **sample**:

```
int sample[] = new int[10];
```

It is possible to break the preceding declaration in two. For example:

```
int sample[]; //declaramos que sample es una matriz de enteros, es una referencia, todavía no es un objeto

sample = new int[10];  //instanciamos en objeto sample y reservamos memoria para 10 elementos enteros
```

In this case, when **sample** is first created, it refers to no physical object. It is only after the second statement executes that **sample** is linked with an array.

An individual element within an array is accessed by use of an index. An *index* describes the position of an element within an array. In Java, all arrays have zero as the index of their first element. Because **sample** has 10 elements, it has index values of 0 through 9. To index an array, specify the number of the element you want, surrounded by square brackets. Thus, the first element in **sample** is **sample[0]**, and the last element is **sample[9]**.

Debido a que las matrices se implementan como objetos, cada matriz tiene asociada una variable de instancia de longitud que contiene la cantidad de elementos que la matriz puede contener. (En otras palabras, la longitud contiene el tamaño de la matriz). Aquí hay un programa que muestra esta propiedad: sample.length

Ejemplo 1:

```
public class PruebasArrays {
   public static void main(String[] args) {
      int ejemplo[] = new int[10];
      int i;
System.out.println("el nº de elementos de ejemplo es  "+ejemplo.length);
      for (i = 0; i < 10; i = i + 1) {
         ejemplo[i] = i;
      }
      for (i = 0; i < 10; i = i + 1) {
         System.out.println("This is ejemplo[" + i + "]: "+ ejemplo[i]);
      }
   }
}
```

Ejemplo 2:

```
public class MinMax {
   public static void main(String args[]) {
      int numeros[] = new int[10];
      int min, max;
      int valor;

      for (int i = 0; i < numeros.length; i++) {
         valor =(int) (Math.random()*20001)-10000;
         numeros[i]=valor;
      }
      min = max = numeros[0];
      for (int i = 1; i < 10; i++) {
                  if (numeros[i] < min) {
                              min = numeros[i];
```

```
                }
                if (numeros[i] > max) {
                        max = numeros[i];
                }
        }
        System.out.println("min and max: " + min + " " + max);
}}
```

Arrays can be initialized when they are created:

```
public class MinMax2 {
   public static void main(String args[]) {
      int nums[] = {99, -10, 10123, 18, -978, 5623, 463, -9, 287, 49};
      int min, max;
      min = max = nums[0];
      for (int i = 1; i < nums.length; i++) {
                if (nums[i] < min) {
                        min = nums[i];
                }
                if (nums[i] > max) {
                        max = nums[i];
                }
      }
      System.out.println("Min and max: " + min + " " + max);
   }
}
```

Ejercicios:

i. Realiza un programa que lea desde teclado 10 números enteros, para posteriormente mostrarlos en el orden inverso al que fueron introducidos.

ii. Rellena un array de 100 casillas con números aleatorios comprendidos entre 1 y 100 (ambos incluidos). Muestra posteriormente los valores almacenados en las casillas impares del array.

iii. Solicita las 20 notas con decimales de los alumnos de una clase. Calcula y muestra la nota media obtenida por el grupo. Muestra después cuántos alumnos tienen una nota superior o igual a la media. Ayuda para redondear `Math.rint(numero*cifras)/cifras;`

iv. Rellena un array de 100 casillas con números aleatorios comprendidos entre 1 y 100 (ambos incluidos). Posteriormente pide un valor al usuario comprendido entre 1 y 100 (insiste en la lectura del valor hasta que el número sea válido y esté comprendido entre 1 y 100) y muestra en pantalla si el valor dado aparece en el array o no, y si aparece indica en qué posiciones del array aparece.

v. Dado un array de 10 casillas relleno con números aleatorios comprendidos entre 1 y 10 (ambos incluidos).
   - Muestra su contenido en una línea de pantalla.
   - Intercambia el valor de la primera casilla con el valor de la última casilla
   - Muestra el contenido actual del array en una línea de pantalla.

vi. Rellena un array con el resultado del cálculo del factorial de los números del 1 al 20 y muestra posteriormente su contenido en pantalla.
   NOTAS:
   Para el cálculo del factorial de un número debes utilizar el factorial que ya has calculado para el número anterior.
   En una variable de tipo int no cabe el factorial de 20.
   20!=2.432.902.008.176.640.000
   9,223,372,036,854,775,807

vii. Realiza un programa que lea desde teclado 10 números, pero que no permita introducir números repetidos. El programa acabará cuando haya obtenido 10 números distintos que mostrará posteriormente en pantalla.

Sorting an Array

Debido a que una matriz unidimensional organiza los datos en una lista lineal indexable, es la estructura de datos perfecta para la clasificación.

Objetivo: ordenar una matriz. Hay una serie de algoritmos de clasificación diferentes.
. There are the quick sort, the shaker sort, and the shell sort, to name just three. Sin embargo, el más conocido, el más simple y el más fácil de entender se llama el tipo de burbuja. Aunque la clasificación de burbuja no es muy eficiente (de hecho, su rendimiento es inaceptable para clasificar matrices grandes), se puede usar de manera efectiva para clasificar matrices pequeñas.

```
public class Burbuja {
```

```
    public static void main(String args[]) {
        int nums[] = {99, -10, 100123, 18, -978, 5623, 463, -9, 287, 49};
        int a, b, t;
        int size;
        size = 10; // número de elementos que hay que ordenar
        // nums.length
        System.out.print("Original array is:");
        for (int i = 0; i < size; i++) {
            System.out.print(" " + nums[i]);
        }
        System.out.println();
        // Algoritmo de la burbuja.
        for (a = 0; a < size-1; a++) {
            for (b = a+ 1; b <size; b++) {
                if (nums[a] > nums[b]) {
                    t = nums[b];
                    nums[b] = nums[a];
                    nums[a] = t;
                }
            }
        }
        System.out.print("Sorted array is:");
        for (int i = 0; i < size; i++) {
            System.out.print(" " + nums[i]);
        }
        System.out.println();
    }
}
```

viii. Realiza un programa que lea 44 números para posteriormente mostrarlos en orden descendente.

ix. Dado un array de 10 posiciones numérico enteras:

- Rellénalo de números aleatorios comprendidos entre 1 y 100 (ambos incluidos)
- Muestra su contenido en una línea de pantalla.
- Ordénalo de menor a mayor **utilizando un algoritmo de ordenación**.
- Muestra su contenido actual en una línea de pantalla.
- Ordénalo de mayor a menor **sin utilizar un algoritmo de ordenación**.
- Muestra su contenido actual en una línea de pantalla.