Arrays

UACM SLT

Ejercicio.

- Crear un programa en Java que realice el ordenamiento burbuja.

```
procedimiento DeLaBurbuja (a(0), a(1), a(2), ..., a(n-1))
    para i ← 1 hasta n hacer
        para j ← 0 hasta n − i hacer
            si a(j) > a(j+1) entonces
                aux ← a(j)
                a(j) ← a(j+1)
                a(j+1) ← aux
            fin si
        fin para
    fin para
fin procedimiento
```

Para quien no conozca el método, consiste en ordenar un arreglo de tamaño N de mayor a menor o viceversa.

Parte 2 de Arrays

- Supongamos.
- int notas[][];
- notas = new int[3][12];//notas está compuesto por 3 arrays
- //de 12 enteros cada uno
- notas[0][0]=9;//el primer valor es 9

Longitud de un array

- Los arrays poseen un método que permite determinar cuánto mide un array. Se trata de length

  - System.out.println(notas.length);

La clase Arrays

- En el paquete java.utils se encuentra una clase estática llamada Arrays. Una clase estática permite ser utilizada como si fuera un objeto (como ocurre con Math). Esta clase posee métodos muy interesantes para utilizar sobre arrays.
  - Su uso es:
    - Arrays.método(argumentos);
- **fill**
- Permite rellenar todo un array unidimensional con un determinado valor. Sus argumentos son el array a rellenar y el valor deseado:
  - int valores[]=new int[23];
  - Arrays.fill(valores,-1);//Todo el array vale -1

- También permite decidir desde que índice hasta qué índice rellenamos:
  - Arrays.fill(valores,5,8,-1);//Del elemento 5 al 7 valdrán -1
- **equals**
- Compara dos arrays y devuelve true si son iguales. Se consideran iguales si son del mismo tipo, tamaño y contienen los mismos valores.
- **sort**
- Permite ordenar un array en orden ascendente. Se pueden ordenar sólo una serie de elementos desde un determinado punto hasta un determinado punto.
- int x[]={4,5,2,3,7,8,2,3,9,5};
- Arrays.sort(x);//Estará ordenado
- Arrays.sort(x,2,5);//Ordena del 2º al 4º elemento

- **binarySearch**
- Permite buscar un elemento de forma ultrarrápida en un array ordenado (en un array desordenado sus resultados son impredecibles). Devuelve el índice en el que está colocado el elemento. Ejemplo:
- int x[]={1,2,3,4,5,6,7,8,9,10,11,12};
- Arrays.sort(x);
- System.out.println(Arrays.binarySearch(x,8));//Da 7

- El método **System.arraysCopy**
- La clase System también posee un método relacionado con los arrays, dicho método permite copiar un array en otro. Recibe cinco argumentos: el array que se copia, el índice desde que se empieza a copia en el origen, el array destino de la copia, el índice desde el que se copia en el destino, y el tamaño de la copia (número de elementos de la copia).

- int uno[]={1,1,2}; int dos[]={3,3,3,3,3,3,3,3,3};
- System.arraycopy(uno, 0, dos, 0, uno.length);
- for (int i=0;i<=8;i++)
- {
- System.out.print(dos[i]+" ");
- } //Sale 112333333

CLASE STRING

- Para Java las cadenas de texto son objetos especiales. Los textos deben manejarse creando objetos de tipo String. Ejemplo:
  - String texto1 = “¡Prueba de texto!”;
- Las cadenas pueden ocupar varias líneas utilizando el operador de concatenación “+”.
  - String texto2 =”Este es un texto que ocupa “ +
  - “varias líneas, no obstante se puede “+
  - “perfectamente encadenar”;

- También se pueden crear objetos String sin utilizar constantes entrecomilladas, usando otros constructores:
  - char[] palabra = {‘P’,’a’,’l’,’b’,’r’,’a’};//Array de char
  - String cadena = new String(palabra);
  - byte[] datos = {97,98,99};
  - String codificada = new String (datos, “8859_1”);
- En el último ejemplo la cadena codificada se crea desde un array de tipo byte que contiene números que serán interpretados como códigos Unicode. Al asignar, el valor 8859_1 indica la tabla de códigos a utilizar.

Comparación entre objetos String

- Los objetos String no pueden compararse directamente con los operadores de comparación. En su lugar se deben utilizar estas expresiones:
- cadena1.equals(cadena2). El resultado es true si la cadena1 es igual a la cadena2. Ambas cadenas son variables de tipo String.
- cadena1.equalsIgnoreCase(cadena2). Como la anterior, pero en este caso no se tienen en cuenta mayúsculas y minúsculas.
- s1.compareTo(s2). Compara ambas cadenas, considerando el orden alfabético. Si la primera cadena es mayor en orden alfabético que la segunda devuelve 1, si son iguales devuelve 0 y si es la segunda la mayor devuelve -1. Hay que tener en cuenta que el orden no es el del alfabeto español, sino que usa la tabla ASCII, en esa tabla la letra ñ es mucho mayor que la o.
- s1.compareToIgnoreCase(s2). Igual que la anterior, sólo que además ignora las mayúsculas (disponible desde Java 1.2)

String.valueOf

- Este método pertenece no sólo a la clase String, sino a otras y siempre es un método que convierte valores de una clase a otra. En el caso de los objetos String, permite convertir valores que no son de cadena a forma de cadena. Ejemplos:
  - String numero = String.valueOf(1234);
  - String fecha = String.valueOf(new Date());
- En el ejemplo se observa que este método pertenece a la clase String directamente, no hay que utilizar el nombre del objeto creado (como se verá más adelante, es un método estático).