U.T.6: ESTRUCTURAS DE ALMACENAMIENTO

Hasta ahora hemos utilizado una variable para almacenar cada objeto. Ya hemos percibido la necesidad de utilizar una estructura de almacenamiento que nos permita guardar una cantidad de objetos sin necesidad de tener que declarar de antemano tantas variables, cada una con su propio nombre. Para solucionar este problema se utilizan distintos tipos de estructuras de almacenamiento: arrays, vectores, colecciones….

ARRAY UNIDIMENSIONAL

Colección de elementos del mismo tipo. Los elementos pueden ser tipos básicos (enteros, reales, char) o bien objetos (String, clases nuestras) . Los elementos se almacenan en posiciones consecutivas de memoria.

Creación:

```
tipo nombretabla[] = new tipo[limite];
tipo nombretabla[] = {valores};
```

Las tablas pueden contener elementos de cualquier tipo o clase, pero el índice y el límite deben ser siempre un entero. Cada elemento está indexado por un número de 0 a límite - 1. Cada elemento es referido con el nombre del array y la posición que ocupa dentro de éste

Ejemplos:

```
int frecuencias[]= new int[15];
Fechas fiestas[]= new Fechas[30];
String paises[]= new String[100];
```

Estas instrucciones crean e inicializan:

```
char vocales[]={'a','e','i','o','u'};
int enteros[]={3,4,5,2};
String cadenas[]={"Hola","vale","otra"};
Fechas fiestas[]={new Fechas(17,2,1998),new Fechas(3,4,2001)};
```

Para introducir un elemento en la tabla:

```
frecuencias[1]=5; // Introduce un 5 en la posición 1 del array
fiestas[5]=new Fechas(97,12,13);// Introduce una fecha en la pos 5 del array
paises[2]="Suiza"; //Introduce la cadena “Suiza” en la posición 2 del array
```

De igual manera, para acceder a un elemento de una tabla, damos el nombre de la variable tabla y el índice entre corchetes:

```
int numero=frecuencias[1]+2; // Movería un 7 a la variable numero
```

Por defecto una tabla se encuentra inicializada a los siguientes valores:

numéricos: 0
char: '\0'
boolean:false
objetos: null

Propiedades:

- Sus elementos pueden ser de cualquier tipo.
- El tamaño de una tabla se fija en el momento de su creación y no se puede cambiar después.
- Si asignamos una tabla a otra no se crea una copia de la tabla, sino que las dos tablas se refieren al mismo espacio de almacenamiento, y los cambios que se hagan en una tabla se ven reflejados en la otra.
- Cuando accedemos a una tabla se comprueban los límites, si estamos fuera: ArrayIndexOfBoundsException.
- Longitud de una tabla: **frecuencias.length**
- Las tablas pueden pasarse como parámetros a los métodos; el parámetro formal del método no tiene que especificar la longitud de la tabla que espera. Se puede calcular usando la propiedad length.

Ejemplos:

1. Programa que lea 10 números por teclado , pida otro más y compruebe cuantas veces se repite este último entre los 10 introducidos.
2. Programa que almacena cadenas de caracteres en un array y luego recorre el array y las muestra por pantalla.
3. Programa que lee desde teclado la temperatura de una semana(todos los días) y calcula la temperatura media. Permite después introducir un día y nos dice la temperatura que hizo ese día.
4. Añadir una opción al programa anterior para que escriba por pantalla los días en que se registró la temperatura máxima y la mínima.
5. Leer una fecha desde teclado (día, mes y año) y escribirla en formato largo usando un array para almacenar el nombre de los meses:
   3/5/1999……..3 de Mayo de 1999

Métodos proporcionados en java.util.Arrays:

static void sort([] a): Ordena el array pasado como parámetro en forma ascendente sólo si sus elementos son de tipos básicos (int, long, short, char, byte, boolean,float o double) o String.

static int binarySearch([] a, v): a es un array ordenado de los tipos anteriores y v es el valor a buscar. Si lo encuentra retorna el índice, sino retorna un número negativo.

static void fill([] a, v): Rellena el array con el valor indicado.

Método de String

String[] split(String separador) : Dado un String, devuelve un array de String que es el resultado de dividir dicho String en subStrings tomando como separador la cadena que se pasa como parámetro.

Ejemplo:

```
String cadena="lunes,martes,miércoles,jueves";
String [] resultado=cadena.split(",");
for(int i=0; i<resultado.length;i++)
        System.out.println(resultado[i]);
```

Este programa mostrará por pantalla:
lunes
martes
miércoles
jueves

Podemos usar varios separadores en split, poniendo éstos entre corchetes. Además hay caracteres especiales que deben ir precedidos de \ ó \\.
Ejemplo de split que parte la cadena usando como separadores: espacio, punto, coma, punto y coma y comillas: cadena.split("[.,;\"]")

Ejemplos:

6. Lee 10 números y almacénalos en un array y después ordénalos en orden ascendente. Pide un número por teclado e indica si se encuentra en el array.
7. Lee un String con varios números decimales separados por $. Transforma el String en un array de doubles (usando Split) y ordénalos en orden ascendente.

Operaciones con Arrays

- Búsqueda de un elemento

Búsqueda secuencial (lineal): Se utiliza cuando el array no está ordenado y consiste en recorrer todo el array, desde el principio hasta el fin buscando el elemento

Búsqueda lineal ordenada: buscamos un elemento en un array ordenado de forma creciente, paramos porque encontramos el elemento o porque encontramos uno mayor que el que buscamos.

- Ordenación de un array unidimensional: Hay muchos algoritmos que ordenan arrays. No todos ordenan de la misma manera ni todos son igual de eficientes. Dependiendo de las circunstancias del array unos lo son más que otros. Los algoritmos existentes son: burbuja, cocktail sort, insertion sort, merge sort, quicksort.

  Método de la burbuja: Más fácil pero menos eficiente.

```
public static void burbuja(int v[], int TAM)
{
int j,k,aux;
for(j=1;j<TAM;j++)  // Doy tantas vueltas como elementos tenga mi array menos 1
        for (k=0; k<TAM-j;k++) // En cada vuelta llevo el elemento mayor al final del  array.
                if (v[k]>v[k+1])  // Si un elemento es mayor que el siguiente, los intercambio
                        {
                        aux=v[k];
                        v[k]=v[k+1];
                        v[k+1]=aux;
                        }
}
```

  Método de quicksort: Más complicado, pero más eficiente

```
public static void quicksort(int A[], int izq, int der) {

 int pivote=A[izq]; // tomamos primer elemento como pivote
 int i=izq; // i realiza la búsqueda de izquierda a derecha
 int j=der; // j realiza la búsqueda de derecha a izquierda
 int aux;

 while(i<j){          // mientras no se crucen las búsquedas
   while(A[i]<=pivote && i<j) i++; // busca elemento mayor que pivote
   while(A[j]>pivote) j--;        // busca elemento menor que pivote
   if (i<j) {                  // si no se han cruzado
      aux= A[i];               // los intercambia
      A[i]=A[j];
      A[j]=aux;
   }
 }
```

```
  A[izq]=A[j]; // se coloca el pivote en su lugar de forma que tendremos
  A[j]=pivote; // los menores a su izquierda y los mayores a su derecha
  if(izq<j-1)
    quicksort(A,izq,j-1); // ordenamos subarray izquierdo
  if(j+1 <der)
    quicksort(A,j+1,der); // ordenamos subarray derecho
}
```

El método sort de la clase Arrays, utiliza el método quicksort. Cuando el array es de elementos de tipos básicos utilizamos el método sort.

- Borrado de un elemento: se busca ese elemento y a partir de su posición se recoloca todo el array.

Ejemplo:

8. Codificar un programa que lea tantos enteros como desee el usuario (máximo 20) y los introduce en un array. Luego introducir un número y buscarlo secuencialmente.

9. Igual que antes, pero antes ordenar el array y buscar un número haciendo una búsqueda ordenada.

10. Crear un array de 100 enteros aleatorios entre 1 y 100. Ordenarlos mediante burbuja y quicksort. Comprobar que método es más rápido.

ARRAY BIDIMENSIONAL

Array de dos dimensiones, los datos están organizados en filas y columnas.

Una matriz de enteros se declararía así:

```
int matriz[][] = new int[4][5];
```

Para referirnos a la fila 2 de la matriz:

```
matriz[2];
```

Para referirnos al elemento que ocupa la posición 1, 2:

```
matriz[1][2];
```

Podemos intercambiar dos filas:

```
fila=matriz[i];
matriz[i]=matriz[j];
matriz[j]=fila;
```

Ejemplo

Lee una matriz de números marca aquellos puntos de la matriz cuyo valor es mayor que la media de los que le rodean.

```
import java.io.*;

class Oro{
	private int maxFila,maxCol;
	private char [][] mapa;
	private double[][] datos;
	static final char blanco=' ';
	static final char estrella='*';


	public Oro(int f, int c){
		maxFila=f;
		maxCol =c;
		datos = new double [maxFila][maxCol];
		mapa=new char[maxFila][maxCol];
	}

	public void leer() throws IOException{
```

```
                BufferedReader f= new BufferedReader( new InputStreamReader(
System.in));
                for (int i=0; i < maxFila; i++)
                    for (int j=0; j< maxCol;j++)
                        datos[i][j]=Double.valueOf(f.readLine().trim()).doubleValue();

                }

        public void evaluar(){
                double punto,media;
                for (int i=0; i<maxFila;i++)
                        for (int j=0; j<maxCol;j++)
                                mapa[i][j]=blanco;
                for (int i=1; i<maxFila-1;i++)
                        for (int j=1; j<maxCol-1;j++)
                        {
                                punto=datos[i][j];
                                media=(datos[i-1][j]+datos[i+1][j]+ datos[i][j-1]+
datos[i][j+1])/4;
                                if (punto > media) mapa[i][j] = estrella;
                        }
        }

        public void imprimir(){
                System.out.print("  ");
                for (int j=0; j< maxCol; j++)
                                System.out.print(j+ " ");
                System.out.println();
                for (int i=0; i < maxFila; i++)
                {
                        System.out.print(i+ " ");
                        for (int j=0; j<maxCol; j++)
                                        System.out.print(mapa[i][j]+" ");
                        System.out.println();
                }
                }
        }

        class buscar{

        public static void main( String[] args) throws IOException{
        Oro mina= new Oro(4,4);
        mina.leer();
        mina.evaluar();
        mina.imprimir();
        }}
```