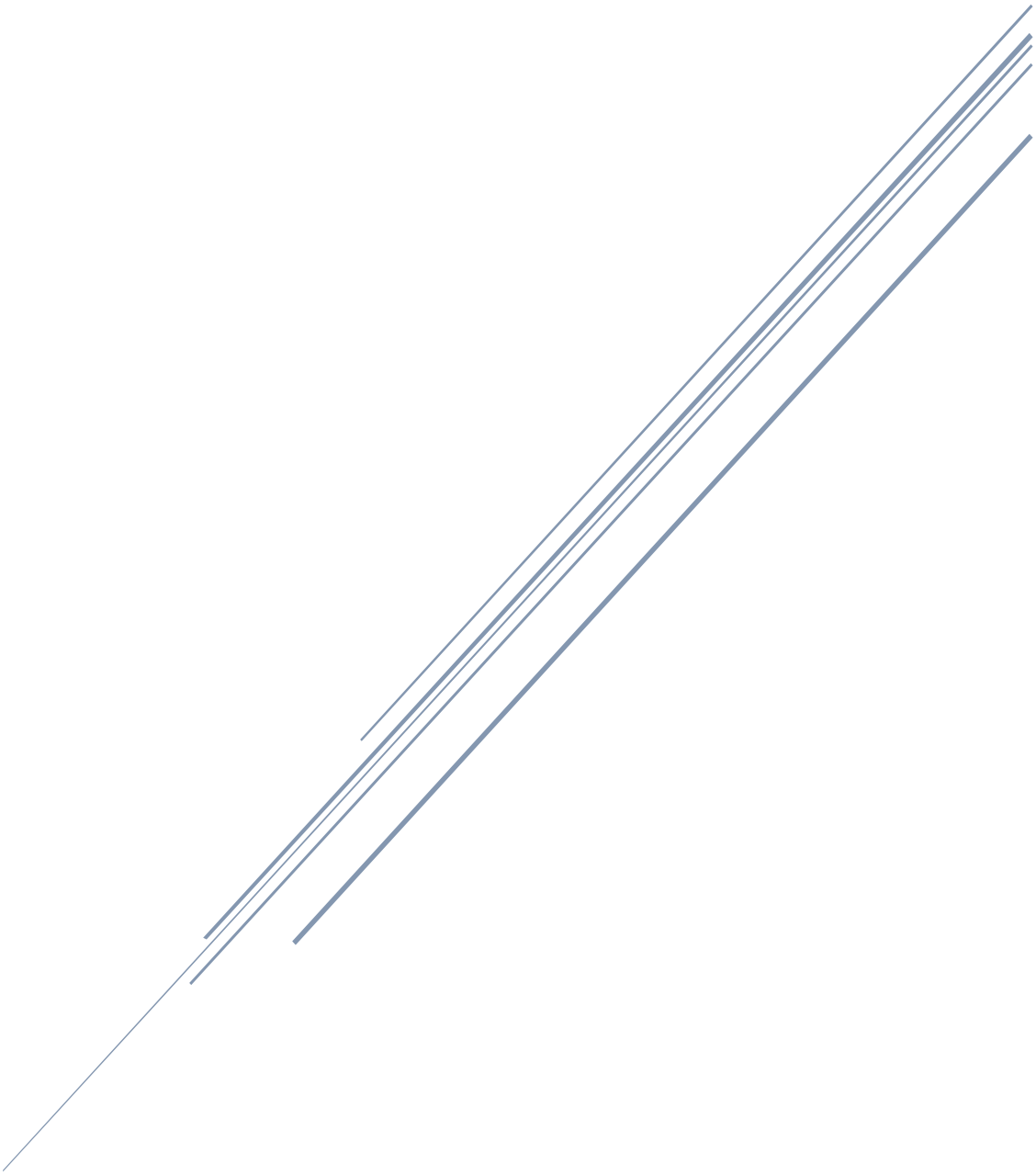
ARRAYS

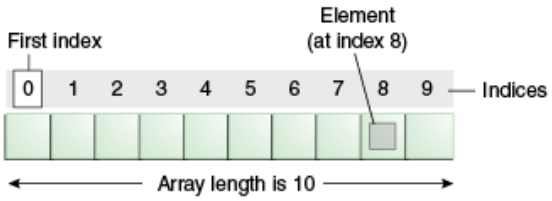
PROGRAMACIÓN



[Escuela]
[Título del curso]

Contenido

Array (Vectores)



En este tema veremos el uso de los arrays o vectores. Los arrays consisten en realizar una carga con varios datos a una misma variable, es decir, si quisiéramos almacenar las notas de n alumnos debería ser dinámico y usaríamos los arrays independientemente para cada alumno pero con una misma variable.

Empecemos a ver un poco sobre la forma de uso de los arrays

DECLARACIÓN DE ARRAYS

```
int[] ArrayEnteros = new int[50];
float[] ArrayReales = new float[50];
boolean[] Valo r = new boolean[50];
String[] CadenaMultiple = new String[50];
```

Primero identificamos con los primeros corchetes que es un tipo array, luego al asignarle el valor new <tipo>[<Longitud>] estamos especificando cuantos datos va a poder recibir nuestro array

```
byte[] anArrayOfBytes;
short[] anArrayOfShorts;
long[] anArrayOfLongs;

float[] anArrayOfFloats;

double[] anArrayOfDoubles;
boolean[] anArrayOfBooleans;
char[] anArrayOfChars;
String[] anArrayOfStrings;
```

```
//this form is discouraged
float anArrayOfFloats[];



Ejemplo:

//create an array of integers
anArray = new int[10];
```

ASIGNACIÓN DE ARRAYS

```
anArray[0] = 100; // initialize first
element
anArray[1] = 200; // initialize second
element
anArray[2] = 300; // and so forth
```

Para asignar generalmente usaremos los bucles aunque no es algo obligatoria ya que se pueden asignar los valores según los necesitemos. Como ejemplo haremos la asignación de todas las variables declaradas mediante un bucle for

```
int i;
for(i=0;i<50;i++){
      ArrayEnteros[i]=i;
      ArrayReales[i]=i;
      Valor[i]=true;
      CadenaMultiple[i]="Cadena";}
```

El “i” dentro de los corchetes indica la posición de memoria donde se almacenará el valor que se le pase, en este caso “i” empieza desde “0” tal como los arrays que siempre empieza su posición de memoria desde 0 aunque podríamos hacer que recauda sus valores desde otra posición. Como vemos el bucle va desde “0” hasta “49” lo cual consta de 50 vueltas y llena a todos los arrays con los datos que les pasemos, si llegaríamos a la vuelta “51” surgiría un error de desbordamiento de memoria ya que los arrays lo declaramos sólo hasta 50 espacios de memoria

IMPRESIÓN DE ARRAYS

Para imprimir los valores de un array se usa la misma lógica de cuando lo almacenamos, lo cual se puede apreciar seguidamente

Supongamos que queremos imprimir el ArrayEnteros pero sólo queremos los 7 primeros valores, lo haríamos de la siguiente forma

```
for(i=0;i<7;i++){
    System.out.println(ArrayEnteros[i]);}
```

Resultado

```
0
1
2
3
4
5
6
```

Como podemos ver imprime sólo los 7 primeros valores, así podríamos imprimir los valores de las posiciones de memoria que deseemos

Ejercicio EjArray01: (Entregar)

Realizar un programa que cree muestre los primeros 100 números por pantalla. Este programa debe de estar realizado exclusivamente con Arrays, tipo de dato donde se almacenara previamente los datos.

Array Bidimensional

En Java es posible crear arrays con más de una dimensión, pasando de la idea de lista, vector o matriz de una sola fila a la idea de matriz de m x n elementos, estructuras tridimensionales, tetradimensionales, etc. La sintaxis será:

Tipo_de_variable[ ][ ] ….. [ ] Nombre_del_array = new Tipo_de_variable[dimensión1] [dimensión2]….[dimensiónN];

```
/* Ejemplo declaración - aprenderaprogramar.com */
int[][]  matriz = new int[3][2];

    O alternativamente

int[][]  matriz;
matriz = new int[3][2];
```

El número de elementos sería: 3 x 2 = 6, dónde 3 es el número de filas y 2 es el número de columnas.

Ahora procedemos a cargar la matriz con valores:

```
matriz[0][0]    =    1;
matriz[0][1]    =    2;
matriz[1][0]    =    3;
matriz[1][1]    =    4;
matriz[2][0]    =    5;
matriz[2][1]    =    6;
```

```
class MultiDimArrayDemo {
    public static void main(String[] args) {
        String[][] names = {
            {"Mr. ", "Mrs. ", "Ms. "},
            {"Smith", "Jones"}
        };
        // Mr. Smith
        System.out.println(names[0][0] + names[1][0]);
        // Ms. Jones
        System.out.println(names[0][2] + names[1][1]);
    }
}
```

```
int[][]  matriz = {{1,2},{3,4},{5,6}};
```

donde {1,2} corresponde a la fila 1, {3,4} a la fila 2 y {5,6} a la fila 3, y los números separados por coma dentro de cada fila, corresponden a las columnas. En este caso, los números (1, 3, 5) de cada una de las filas corresponden a la primera columna y los números (2, 4, 6) atañen a la segunda columna.

Para obtener el número de filas de la matriz, podemos recurrir a la propiedad “length” de los arrays, de la siguiente manera:

```
int filas = matriz.length;
```

Para el caso del número de columnas sería de la siguiente forma:

```
int columnas = matriz[0].length;
```

También Java nos permite la posibilidad de clonar una matriz, es decir, crear una matriz nueva a partir de otra matriz, siguiendo esta sintaxis:
String[][] nuevaMatriz = matriz.clone();

donde clone() es un método especial, que permite la clonación de arrays de cualquier dimensión en Java. De esta manera “nuevaMatriz” y “matriz” son 2 matrices distintas pero con los mismos valores. Hablaremos del método clone más adelante.

Ejercicio EjArray02: (Entregar)

Vamos a plantear y resolver un ejercicio: queremos almacenar en una matriz el número de alumnos con el que cuenta una academia, ordenados en función del nivel y del idioma que se estudia. Tendremos 3 filas que representarán al Nivel básico, medio y de perfeccionamiento y 4 columnas en las que figurarán los idiomas (0 = Inglés, 1 = Francés, 2 = Alemán y 3 = Ruso). Se pide realizar la declaración de la matriz y asignarle unos valores de ejemplo a cada elemento.

Copying Arrays

The `System` class has an *`arraycopy`* method that you can use to efficiently copy data from one array into another:

```
public static void arraycopy(Object src, int srcPos,
                             Object dest, int destPos, int length)
```