

Informática II - Módulo 1

Arreglos Multidimensionales

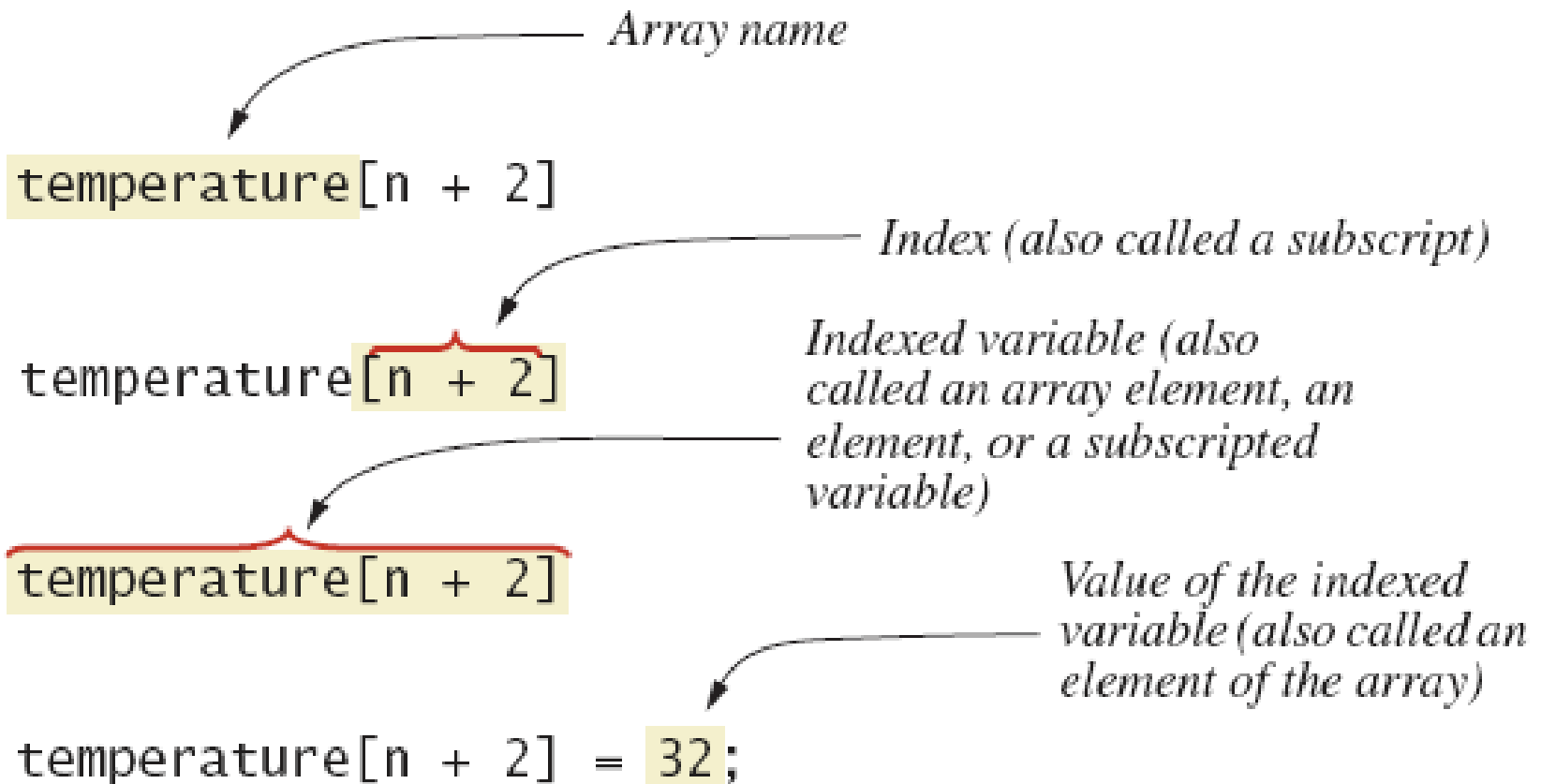


Definiendo y Leyendo Arreglos

Un arreglo es una colección de variables del mismo tipo. Nos sirven para almacenar muchas variables de un mismo tipo de dato.

```
int[] arr1;  
arr1 = new int[10];  
  
int arr2[];  
arr2 = new int[10];  
  
int[] arr3 = new int[10];  
  
int arr4[] = new int[10];  
  
int[] arr5 = {0,0,0,0,0,0,0,0,0,0};  
  
int a = 0;  
int[] arr6 = new int[]{a, a, a, a, a, a, a, a, a, a};
```

Arreglos



Método 1: Ciclo For

```
int[] numbers = {1,2,3,4,5,6,7,8,9,10};  
  
for (int i=0; i<arr1.length; i++){  
    System.out.println("Count: " + arr1[i]);  
}
```

Esta declaración implícita está creando un arreglo de 10 elementos. Cada elemento está separado por una coma (,).

Este ciclo iterará sobre cada elemento del arreglo numbers, asignando el contenido en la variable **int item**.

```
Count is: 1  
Count is: 2  
Count is: 3  
Count is: 4  
Count is: 5  
Count is: 6  
Count is: 7  
Count is: 8  
Count is: 9  
Count is: 10
```

Método 2: Ciclo For-each

```
int[] numbers = {1,2,3,4,5,6,7,8,9,10};  
for (int item : numbers) {  
    System.out.println("Count is: " + item);  
}
```

Este ciclo iterará sobre cada elemento del arreglo numbers, asignando el contenido en la variable **int item**.

```
Count is: 1  
Count is: 2  
Count is: 3  
Count is: 4  
Count is: 5  
Count is: 6  
Count is: 7  
Count is: 8  
Count is: 9  
Count is: 10
```

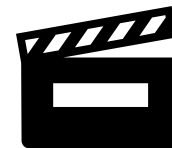
Métodos

```
public class SampleClass
{
    public static void incrementArrayBy2(double[] anArray)
    {
        for (int i = 0; i < anArray.length; i++)
            anArray[i] = anArray[i] + 2;
    }
    <The rest of the class definition goes here.>
}
```

Arreglos multidimensionales

Los arreglos multidimensionales son **estructuras de datos** que permiten almacenar grupos de información definidos.

Un arreglo multidimensional es un arreglo de arreglos.



Arreglos multidimensionales

Usando el índice de la fila (3) y la columna (2) podemos acceder el elemento deseado.

		Column index 2					
Row index 3	Indices	0	1	2	3	4	5
	0	\$1050	\$1055	\$1060	\$1065	\$1070	\$1075
	1	\$1103	\$1113	\$1124	\$1134	\$1145	\$1156
	2	\$1158	\$1174	\$1191	\$1208	\$1225	\$1242
	3	\$1216	\$1239	\$1262	\$1286	\$1311	\$1335
	4	\$1276	\$1307	\$1338	\$1370	\$1403	\$1436
	5	\$1340	\$1379	\$1419	\$1459	\$1501	\$1543
	6	\$1407	\$1455	\$1504	\$1554	\$1606	\$1659
	7	\$1477	\$1535	\$1594	\$1655	\$1718	\$1783
	8	\$1551	\$1619	\$1689	\$1763	\$1838	\$1917
	9	\$1629	\$1708	\$1791	\$1877	\$1967	\$2061

`table[3][2]` has a value of 1262



Por convención, el primer índice corresponde a las filas y el segundo índice a la columna.

Declaración de arreglos multidimensionales

La forma general de declarar una matriz es la siguiente:

```
//Method 1: Declaration and instantiation
int[][] x1 = new int[3][3];
int x2[][] = new int[3][3];

// Method 2: Implicit initialization
int[][] x3 = {{1,2,3},{4,5,6},{7,8,9}};

//Method 3: Declaration and instantiation
int[][] x4 = new int[3][];
x4[0] = new int[3];
x4[1] = new int[3];
x4[2] = new int[3];
```

Declaración General

```
data_type[][] variable_name = new data_type[rows][columns];
```

`data_type`: String, char, Scanner, int

`variable_name`: Identificador mediante el cual nos vamos a referir a la variable

`rows y columns`: Cantidad de filas y columnas

Arreglos multidimensionales

Para acceder a cada uno de los elementos, debemos utilizar los **índices** pertinentes al elemento que queremos utilizar.

	0	1	2	3	4
0	[0][0]	[0][1]	[0][2]	[0][3]	[0][4]
1	[1][0]	[1][1]	[1][2]	[1][3]	[1][4]
2	[2][0]	[2][1]	[2][2]	[2][3]	[2][4]
3	[3][0]	[3][1]	[3][2]	[3][3]	[3][4]
4	[4][0]	[4][1]	[4][2]	[4][3]	[4][4]
5	[5][0]	[5][1]	[5][2]	[5][3]	[5][4]

Declaración de arreglos

```
//matrix with:  
// rows = 6  
// columns = 5  
int[][] matrix = new int[6][5];  
matrix[4][3] = 1;
```

	0	1	2	3	4
0					
1					
2					
3					
4				1	
5					

Ejemplo: Calendario

Diseña un método `createCalendar()` que instancie y retorne un arreglo multidimensional que sirva para representar el calendario de una semana, como se puede ver en la siguiente imagen.

Cada fila servirá para representar 30 minutos del día, y cada columna representará 1 día de la semana.

	LUN	MAR	MIÉ	JUE	VIE	SÁB	DOM
GMT-06	13	14	15	16	17	18	19
01:00							
02:00							
03:00							
04:00							
05:00							
06:00							
07:00							
08:00							
09:00							
10:00							

```
public static String[][] createCalendar(){
    int rows = 48;      // 48 slots of 30 minutes in a day
    int columns = 7;    // 7 days in a week

    String[][] calendar = new String[rows][columns];
    return calendar;
}
```

Leer dimensión de una matriz

Para obtener el numero de renglones de una matriz podemos utilizar el atributo `length`:

```
nombre.length
```

Para obtener el numero de columnas de una matriz debe podemos utilizar:

```
nombre[row_index].length
```

Matrices

La estructura compañera de las matrices está compuesta por dos ciclos **for anidados**.

La mayoría de los problemas de matrices tienen un código similar al siguiente:

```
for(int row = 0; row < matrix.length; row++){  
    for(int column = 0; column < matrix[row].length; column++) {  
        // Code here  
    }  
}
```

Inicialización de arreglos

Inicialización utilizando ciclos:

```
int[][] matrix = new int[3][3];  
int contador = 1;  
  
for(int row = 0; row < matrix.length; row++){  
    for(int column = 0; column < matrix[row].length; column++) {  
        matrix[row][column] = contador++;  
    }  
}
```

1	2	3
4	5	6
7	8	9

En el calendario, diseña un método para agendar un evento en el calendario.

Deberás recibir:

- La referencia al calendario
- El nombre del evento
- El día
- La hora

Si el evento se puede agendar, agrégalo al calendario y devuelve TRUE. De lo contrario, deberás retornar FALSE.

Agendar eventos

	LUN	MAR	MIÉ	JUE	VIE	SÁB	DOM
GMT-06	13	14	15	16	17	18	19
01:00							
02:00							
03:00							
04:00							
05:00							
06:00							
07:00							
08:00							
09:00							
10:00							

```
public static boolean addEvent(String[][] calendar, String event, int day, int time){  
    // error!  
    if (calendar == null || calendar.length <= time || calendar[time].length <= day) {  
        return false;  
    }  
    if (calendar[time][day] != null)  
        return false;  
  
    calendar[time][day] = event;  
    return true;  
}
```

Incluye también un parámetro duración que permita enviar la duración del evento a agendar (en minutos). Asume que la duración se recibirá en múltiplos de 30.

Reto!

	LUN	MAR	MIÉ	JUE	VIE	SÁB	DOM
GMT-06	13	14	15	16	17	18	19
01:00							
02:00							
03:00							
04:00							
05:00							
06:00							
07:00							
08:00							
09:00							
10:00							

```
public static boolean addEvent(String[][] calendar, String event, int day, int  
time, int duration){
```

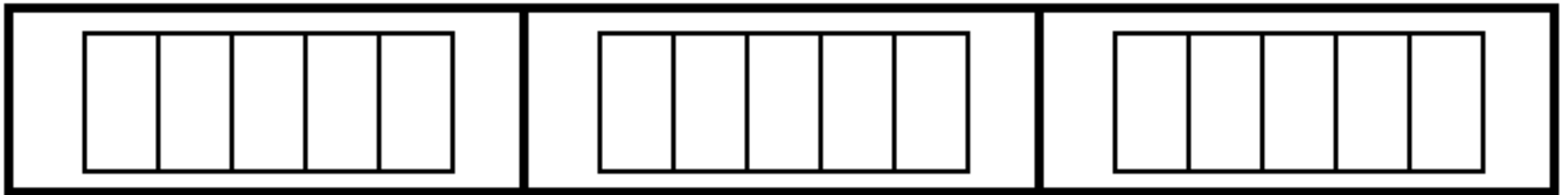
```
}
```

Representación de una matriz

Los arreglos multidimensionales se representan internamente como arreglos de una dimensión.

Ejemplo:

```
int[][] table = new int[3][5];
```



Ragged Arrays

No todos los elementos del arreglo tienen que ser del mismo tamaño.

Ejemplo:

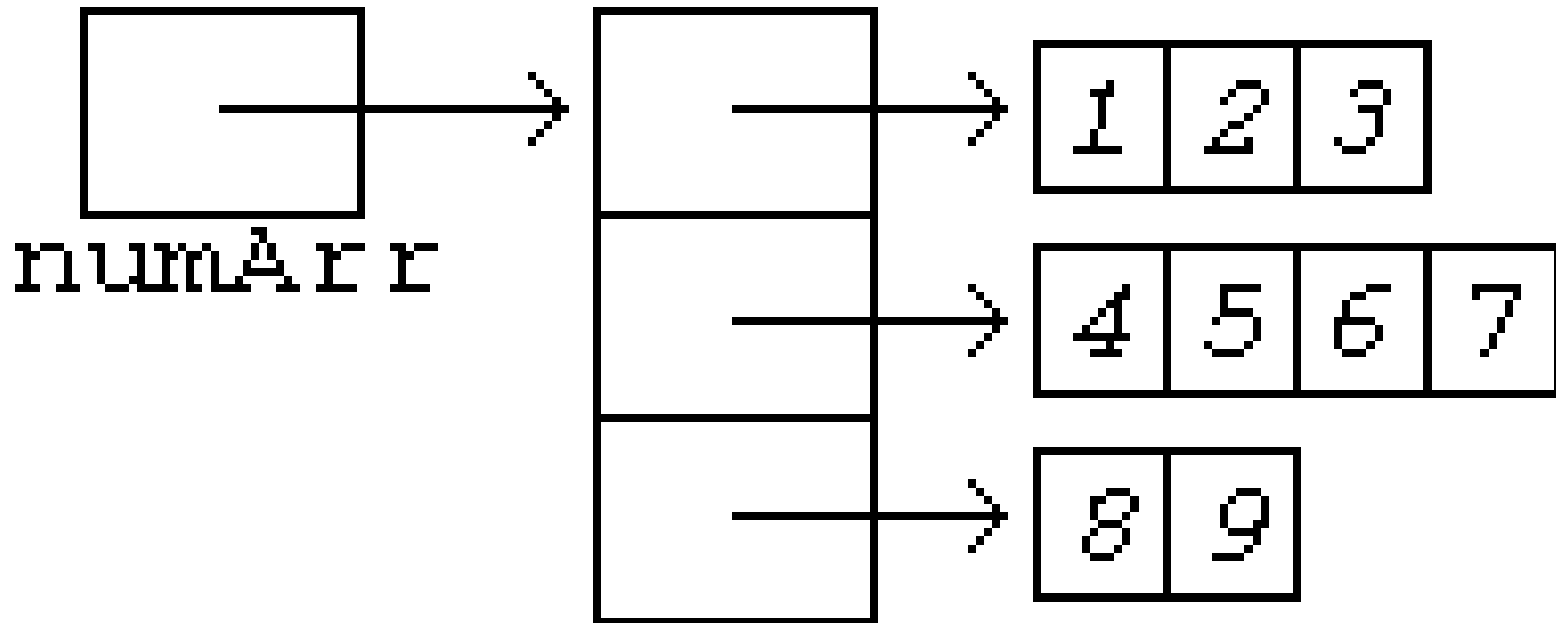
```
int[][] b;  
b = new int[3][];  
b[0] = new int[5]; //First row, 5 elements  
b[1] = new int[7]; //Second row, 7 elements  
b[2] = new int[4]; //Third row, 4 elements
```

Los ragged arrays también pueden ser inicializados implícitamente.

Las siguientes dos sentencias son idénticas:

```
int[][] b;  
b = new int[3][];  
b[0] = new int[5]; //First row, 5 elements  
b[1] = new int[7]; //Second row, 7 elements  
b[2] = new int[4]; //Third row, 4 elements  
  
int[][] c = {{0,0,0,0,0},           //First row, 5 elements  
             {0,0,0,0,0,0,0},      //Second row, 7 elements  
             {0,0,0,0}};           //Third row, 4 elements
```

Ragged Arrays



Layout of the same array

Ejemplo

Declara e inicializa una matriz de enteros para representar los días del mes de Junio 2020. Asígnale el número de día a cada elemento.

Junio de 2020						<	>
L	M	X	J	V	S	D	
1	2	3	4	5	6	7	
8	9	10	11	12	13	14	
15	16	17	18	19	20	21	
22	23	24	25	26	27	28	
29	30						