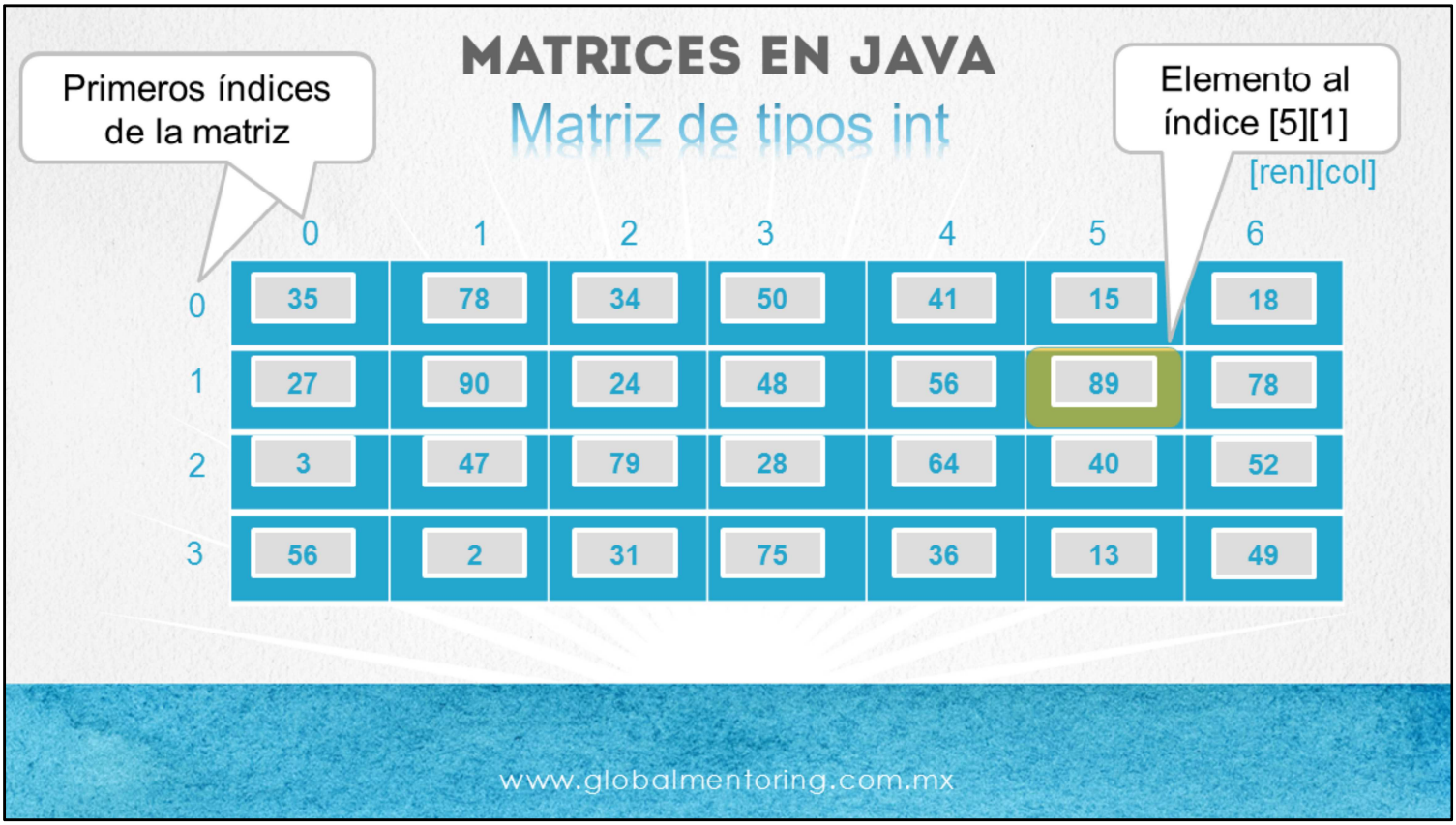
Hola, te saluda Ubaldo Acosta. Bienvenida o bienvenido nuevamente. Espero que estés listo para comenzar con esta lección.

Vamos a estudiar el tema de Matrices en Java.

¿Estás listo? Ok, ¡Vamos!



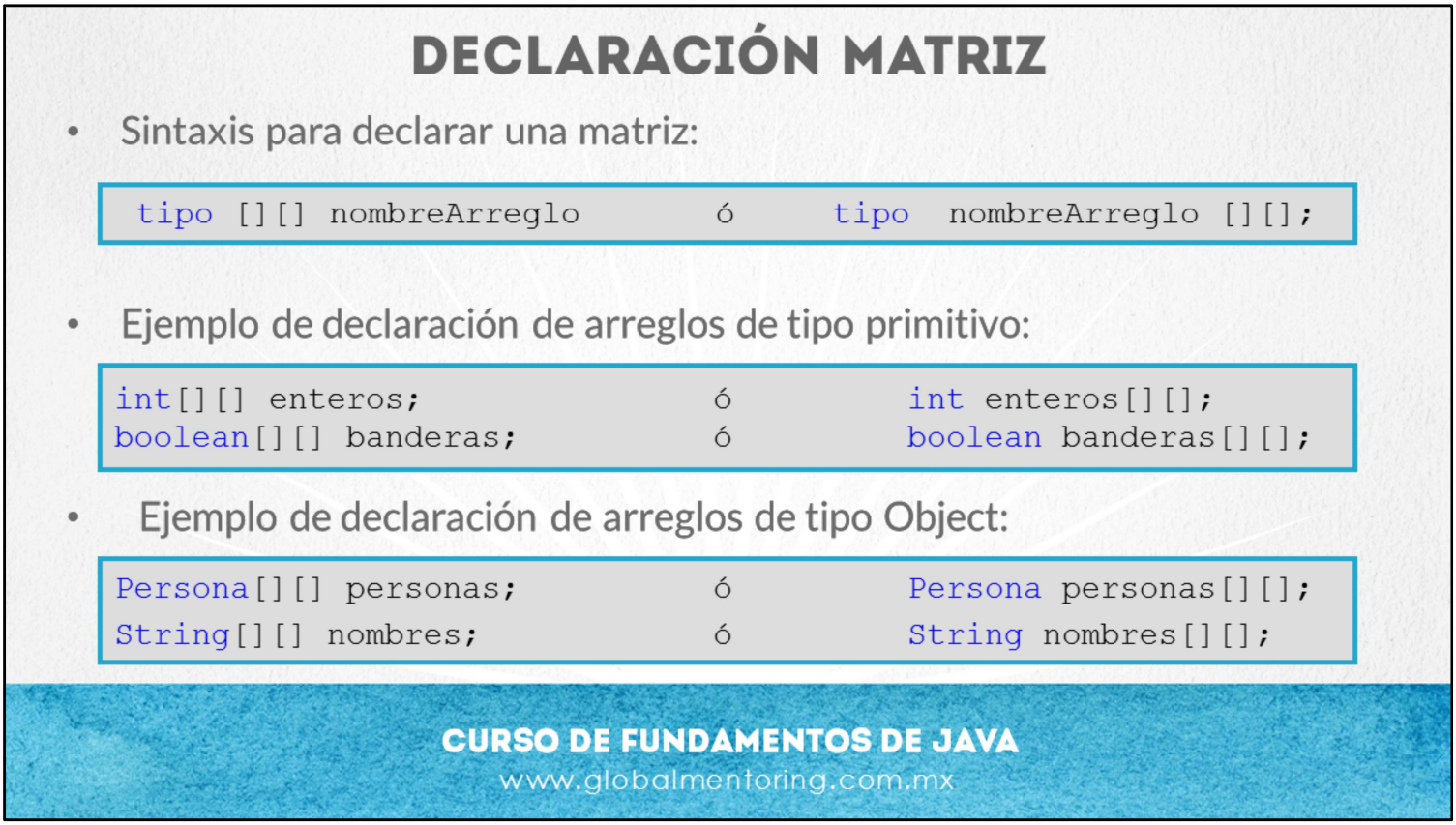
Otra estructura de datos en Java que podemos utilizar es una matriz. A diferencia de un arreglo de una dimensión, una matriz podríamos verla como dos arreglos, un arreglo maneja los renglones y otro arreglo las columnas, y al juntarlos obtenemos una matriz. Sin embargo veremos que existen muchas cosas similares con un arreglo, y una vez entendido el concepto de arreglo es más sencillo entender el concepto de matriz.

En la figura podemos observar una matriz de 4 renglones por 7 columnas, de tipo enteros, sin embargo puede ser de cualquier tipo que definamos.

Podemos recuperar el largo de los renglones con el código nombreArreglo.length y podemos obtener el largo de las columnas escribiendo nombreArreglo[0].length, es decir, que con cualquier renglón válido seleccionado podemos obtener el largo de las columnas. Esto nos va a servir posteriormente para iterar por medio de un ciclo for anidado cada uno de los elementos de la matriz.

De igual manera que un arreglo, no todos los elementos de una matriz deben contener valores. Los valores que no tengan un valor asignado tendrán el valor por default según el tipo de datos definido para la matriz.

A continuación veremos la sintaxis para declarar una matriz.



En esta lámina podemos observar la sintaxis para la declaración de una matriz. Básicamente es como declarar un arreglo excepto que vamos a utilizar doble corchete [][], los cuales se pueden utilizar en dos partes, ya sea antes del nombre de la variable o después del nombre de la variable, por ello se muestran las dos opciones en cada declaración.

Declarar una matriz es igual a declarar variables, podemos declarar matrices que almacenen tipos primitivos o que almacenen referencias a objetos. En la lámina mostramos ambos casos, primero mostramos dos ejemplos de tipo primitivo, uno de tipo int y otro de tipo boolean. Posteriormente mostramos la declaración de dos matrices que almacenarán referencias de objetos de tipo Persona y de tipo String.

Debido a que una matriz es una colección de datos , normalmente el nombre de una matriz es en plural, para que fácilmente con solo leer el nombre de la variable reconozcamos que se trata de una colección de datos, aunque más adelante veremos que se puede tratar no solamente de arreglos o matrices, sino de otras estructuras de datos.

Lo mostrado en la lámina es únicamente la declaración de la variable, veremos a continuación como inicializar una matriz, ya que hasta el momento con sólo declarar una variable de tipo matriz la JVM no sabe cuan largo y ancho es la matriz, para ello debemos inicializarlo, veamos como.

INSTANCIAR MATRICES

- Sintaxis para instanciar una matriz:

```
nombreArreglo = new tipo[renglones][columnas];
```

- Ejemplo para instanciar matrices de tipos primitivos:

```
enteros = new int[2][2] ;//Matriz tipo int: 2 renglones y 2 columnas
anderas = new boolean[3][2];//Matriz de tipo boolean: 3 ren y 2 col
```

- Ejemplo para instanciar matrices de tipo Object:

```
personas = new Persona[4][2]; //Matriz tipo Person: 4 ren y  2 col
nombres = new String[5][3]; //Matriz tipo String: 5 ren y 3 col
```

CURSO DE FUNDAMENTOS DE JAVA
www.globalmentoring.com.mx

Partiendo de las variables definidas en la lámina anterior, en esta lámina veremos la sintaxis para instanciar una matriz según el tipo de dato que estemos utilizando.

La sintaxis es muy similar a instanciar una variable de tipo object, y de hecho esta es una de las características de Java, incluso las matrices o cualquier tipo en Java que almacena una referencia hereda de la clase Object de manera directa o indirecta, por lo tanto las matrices también descienden de la clase Object.

Sin embargo, la diferencia con instanciar un objeto normal a una matriz, es que en un arreglo indicamos el número de elementos máximo que contendrá dicha matriz, tanto en número de renglones, como en el número de columnas. Previamente en la definición de la variable ya se indicó que tipo es el que va almacenar, y ahora debemos indicar que crearemos un objeto de cierto tipo y que contendrá un número de renglones y columnas según se indique.

Podemos observar en la lámina varios ejemplos de inicialización de matrices según el tipo de datos que elegimos. A continuación veremos cómo inicializar los elementos de una matriz.

INICIALIZAR LOS ELEMENTOS DE UNA MATRIZ

- Sintaxis para inicializar los elementos de una matriz:

```
nombreArreglo[índice_renglon][índice_columna] = valor;
```

- Ejemplo para inicializar los elementos de una matriz de tipo entero:

```
enteros[0][0] = 15; //Se asigna el valor de 15 en el ren=0 y col=0
enteros[1][0] = 13; //Se asigna el valor de 13 en el ren=1 y col=0
```

- Ejemplo para inicializar los elementos de un arreglos de tipo Object:

```
personas[0][0] = new Persona(); //Se asigna objeto en ren=0 y col=0
personas[1][1] = new Persona("Pedro","Lara");//Se asigna en ren=1 y col=1
nombres[0][0] = new String("Juan"); //Se asigna String en ren=0, col=0
nombres[2][1] = new String("Sara"); //Se asigna String en ren=2, col=1
```

En la lámina podemos observar la inicialización de los elementos de una matriz. Lo que debemos hacer para ir agregando elementos a una matriz, es seleccionar un renglón y una columna con los índices respectivos que queremos ir inicializando.

Por ello, es importante saber que a diferencia de un arreglo, en una matriz utilizaremos dos índices para determinar la posición de un elemento, y que los primeros índices tanto del renglón como de la columna inician en cero. También es importante saber que cuando indicamos una posición primero se indica el renglón y después la columna, siempre en ese orden. Por ejemplo, el primer elemento de una matriz será el elemento [0][0] y el largo de una matriz realmente son dos, el primero lo determinaremos por el nombreMatriz.length lo que nos regresa el largo de renglones, y posteriormente podemos saber el largo de las columnas seleccionando cualquier renglón, por ejemplo: nombreMatriz[0].length.

Al igual que en un arreglo, sólo podemos agregar elementos hasta el máximo de elementos menos uno, por ejemplo, si son renglones, seria nombreMatriz.lenght -1 y si fuera el máximo de columnas sería nombreMatriz[i].lenght -1, donde i es el renglón que se está trabajando. Si nos pasamos del índice máximo tanto en renglones o columnas y queremos agregar un elemento fuera de la cantidad máxima de elementos nos arrojará un error, por ello debemos saber cual es el máximo número de elementos tanto en renglones como en columnas.

Podemos observar en la lámina varios ejemplos de cómo agregar elementos a nuestra matriz. Podemos agregarlos de manera manual, es decir, uno a uno cada elemento, o podemos ir agregando los elementos de manera más dinámica utilizando dos contadores de elementos que han sido agregados, tanto para los renglones como para las columnas, de tal forma que podamos saber si ya hemos llegado al límite de elementos agregados o no.

En esta lámina podemos observar más claramente que no siempre estarán llenos todos los elementos de una matriz, por ejemplo si la matriz de enteros es de 2 renglones por 2 columnas, entonces sólo hemos llenado 2 de los 4 elementos disponibles, esto quiere decir que 2 elementos tendrán el valor por default, en este caso el valor de 0. Por ello en muchas ocasiones será conveniente tener contadores para poder conocer cuántos elementos se han inicializado en nuestra matriz, lo cual es distinto al número de elementos máximo que soporta nuestra matriz tanto en los renglones como en las columnas disponibles de la matriz.

En el ejercicio de esta lección veremos cómo inicializar los elementos de nuestras matrices.

EXTRAER ELEMENTOS DE UNA MATRIZ

- Sintaxis para extraer los elementos de una matriz:

```
variableReceptora = nombreArreglo[índice_renglón][índice_columna];
```

- Ejemplo para extraer los elementos de una matriz de tipo entero:

```
int i = enteros[0][0]; //Extraemos valor almacenado en ren 0 y col 0
int j = enteros[1][0]; //Extraemos valor almacenado en ren 1 y col 0
```

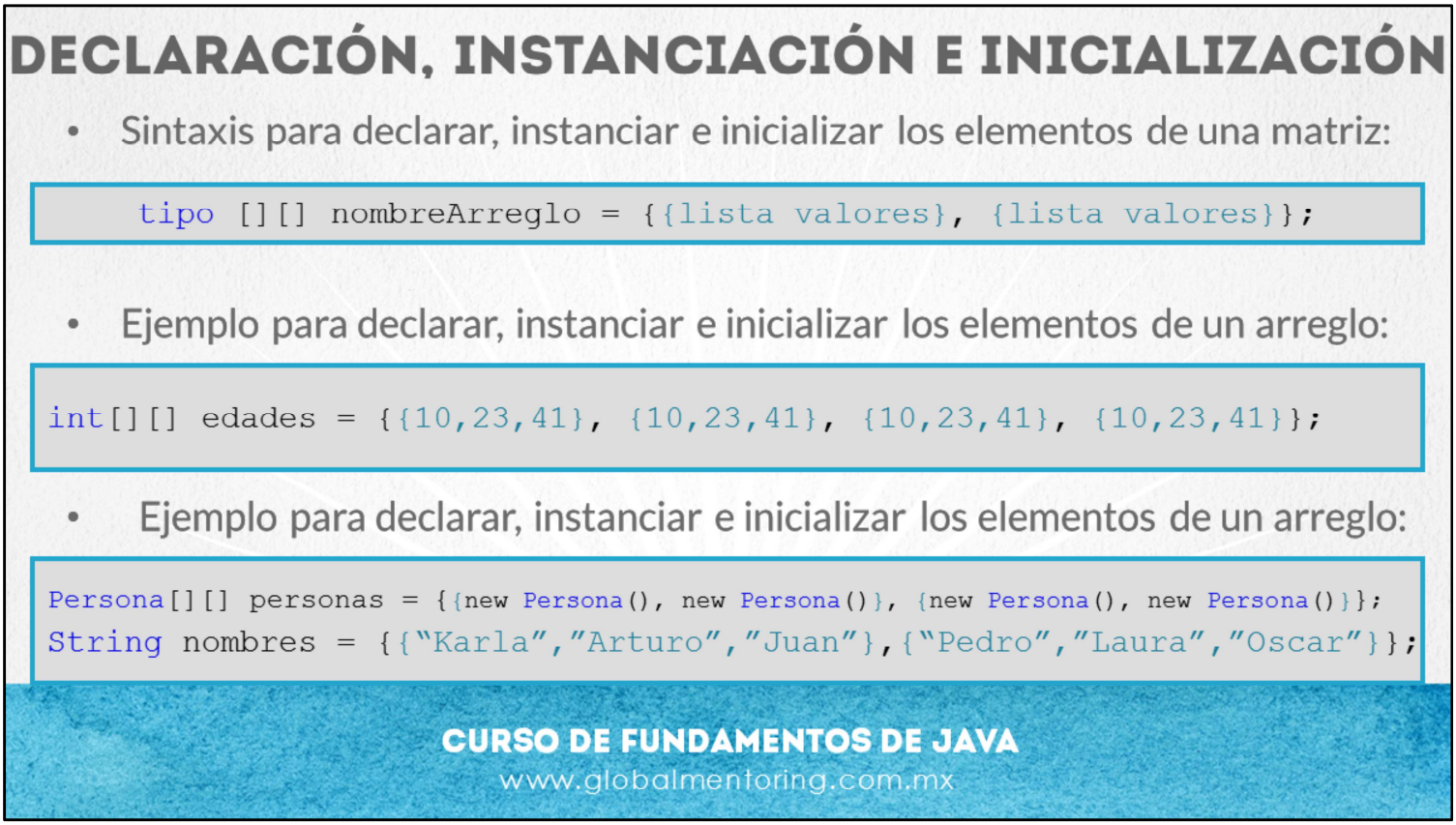
- Ejemplo para extraer los elementos de una matriz de tipo Object:

```
Persona p1 = personas[0][0];//Extraemos valor de ren 0 y col 0
Persona p2 = personas[1][0];//Extraemos valor de ren 1 y col 0
String nombre1 = nombres[0][0]; //Extraemos valor de ren 0 y col 0
String nombre2 = nombres[1][1]; //Extraemos valor de ren 1 y col 1
```

Para leer o extraer los elementos almacenados en una matriz basta con indicar el nombre de la matriz e indicar dos índices, tanto el renglón como la columna del elemento que queremos extraer, esto regresará el elemento de los índices proporcionados.

Es importante no pasarnos del numero máximo de elementos, tanto en renglones como en columnas, de lo contrario regresará un error.

Podemos observar en la lámina varios ejemplos con matrices que almacenan tipos primitivos o tipos Object, en ambos casos la sintaxis es la misma, sólo debemos tener una variable que reciba el valor extraído de matriz respectiva según los índices especificados. Más adelante veremos ejemplos de cómo extraer los elementos utilizando un ciclo for para recorrer cada uno de los elementos de una matriz.



Existe otra forma de declarar matrices, y al mismo tiempo instanciar e inicializar cada uno de sus elementos. Esta es una sintaxis distinta en la forma en que se asignan los valores.

Podemos observar en la lámina varios ejemplos, tanto con tipos primitivos, así como tipos Object. Y lo que podemos observar es que en la misma línea de código se declara la variable, se instancia la matriz y se inicializa cada uno de sus valores.

Sin embargo, esta sintaxis no siempre es posible utilizarla ya que necesitaríamos saber de antemano todos los elementos que van a ser almacenados en la matriz, y en muchas ocasiones no tenemos esta información desde un inicio, pero si tenemos esta información antes de crear nuestra matriz, entonces es posible utilizar esta sintaxis simplificada.

EJEMPLO DE MANEJO DE MATRICES

Ejemplo de uso de matrices:

```
1  public class EjemploMatrices {
2
3      public static void main(String[] args) {
4          //1. Declaramos un arreglo de enteros
5          int edades[][];
6          //2. Instanciamos el arreglo de enteros
7          edades = new int[3][2];
8          //3. Inicializamos los valores del arreglo de enteros
9          edades[0][0] = 30;
10         edades[0][1] = 15;
11         edades[1][0] = 20;
12         edades[1][1] = 45;
13         edades[2][0] = 5;
14         edades[2][1] = 38;
15
16         //Imprimimos los valores a la salida estándar
17         //4. leemos los valores de cada elemento del arreglo
18         System.out.println("Arreglo enteros indice 0-0: " + edades[0][0]);
19         System.out.println("Arreglo enteros indice 0-1: " + edades[0][1]);
20         System.out.println("Arreglo enteros indice 1-0: " + edades[1][0]);
21         System.out.println("Arreglo enteros indice 1-1: " + edades[1][1]);
22         System.out.println("Arreglo enteros indice 2-0: " + edades[2][0]);
23         System.out.println("Arreglo enteros indice 2-1: " + edades[2][1]);
24     }
25 }
```

Podemos observar en la lámina un ejemplo para el uso de matrices

Desde la declaración (línea 5), el instanciamiento (línea 7), la inicialización de valores (líneas 9-14), y finalmente la lectura de los valores (líneas 18-23).

Podemos observar que en este caso, todos los valores del arreglo están inicializados, sin embargo si hubiera valores que no se hayan inicializado su valor será el valor por default del tipo declarado, en el caso del arreglo de tipo int el valor por default es 0 y en el caso del objeto tipo Persona su valor por default será null ya que es un tipo Object.

Más adelante realizaremos este ejercicio para poner en práctica estos conceptos.

EJEMPLO RECORRER UNA MATRIZ CICLO FOR ANIDADO

Ejemplo para recorrer una matriz con un ciclo for anidado:

```
1  public class EjemploMatrices {
2
3    public static void main(String[] args) {
4
5      //1. Matriz de tipo String, notacion simplificada
6      String nombres[][] = {{"Teresa", "Cesar", "William"},{"Yesenia", "Esteban", "Maria"}};
7
8      //Largo de elementos de la matriz. Primero el no. de renglones
9      System.out.println("largo matriz renglones:" + nombres.length);
10     //Seleccionando un renglon valido nos regresa el no. de columnas
11     System.out.println("largo matriz columnas:" + nombres[0].length);
12
13     //Imprimimos los valores a la salida estándar
14     //2. Iteramos la matriz de String con un for anidado
15     for (int i = 0; i < nombres.length; i++) {
16       for(int j= 0; j < nombres[i].length; j++){
17         System.out.println("Matriz String índice : " + i + "-"+ j + " : " + nombres[i][j]);
18       }
19     }
20   }
21 }
```

```
Output - EjemploMatrices (run)
run:
largo matriz renglones:2
largo matriz columnas:3

Matriz String indice : 0-0: Teresa
Matriz String indice : 0-1: Cesar
Matriz String indice : 0-2: William
Matriz String indice : 1-0: Yesenia
Matriz String indice : 1-1: Esteban
Matriz String indice : 1-2: Maria
```

Finalmente veremos un ejemplo con una matriz de tipo String utilizando la notación simplificada, y aprovecharemos para mostrar cómo iterar los elementos de una matriz con la ayuda de un ciclo for anidado. Un ciclo for anidado es simplemente un ciclo for dentro de otro ciclo for. En este caso realizaremos esta anidación para poder recorrer cada uno de los elementos de la matriz.

En primer lugar vemos un ejemplo del uso de la notación simplificada (línea 6). En este caso es una matriz de tipo String, y en la misma línea instanciamos la matriz e inicializamos los valores de la misma. En este caso no hay que indicar el número de renglones o columnas que contendrá la matriz, este número se obtendrá directamente del número de elementos que se agreguen en la inicialización de la matriz. Cabe aclarar que en esta estructura de datos no es posible hacer más grande o más pequeña la matriz una vez declarado o como en este caso una vez inicializado. Sin embargo veremos en el siguiente curso el tema de colecciones, donde veremos estructuras como un ArrayList los cuales son estructuras que pueden crecer dinámicamente.

Una vez que hemos definido cuántos elementos tendrá la matriz, podemos obtener el no. de renglones y de columnas de nuestra matriz. Así que para comprobar estos valores, mandamos a imprimir el número de renglones con el código nombres.length (línea 9), y el número de columnas con el código nombres[0].length (línea 11). Por ello es posible combinar dos ciclos for, y utilizando dos contadores, en este caso las variables i y j, iremos iterando cada uno de los elementos de la matriz.

El primer ciclo, el más externo recorre los renglones de la matriz, y el ciclo más interno recorre las columnas de la matriz. Por ello en la salida de nuestra consola, observaremos que los primeros 3 valores el valor del índice del renglón se mantiene fijo, mientras que índice de la columna se va moviendo hasta que se acaba de iterar las columnas para ese renglón seleccionado. El fin del ciclo más interno será cuando hayamos iterado todas las columnas para el renglón seleccionado, según la condición del ciclo for interno: **j < nombres[i].length** (línea 16). Recordemos que la variable i controla los renglones, y la variable j controla las columnas. Finalmente el ciclo más externo se detendrá cuando se hayan revisado todos los renglones de la matriz, según la condición del ciclo for externo: **i < nombres.length** (línea 15). Con esto habremos iterado todos los renglones, así como cada columna de cada renglón seleccionado, y por consiguiente todos los elementos de la matriz.

Veremos más adelante la ejecución de este código para poner en práctica estos conceptos.

En Global Mentoring promovemos la Pasión por la Tecnología Java. Te invitamos a visitar nuestro sitio Web donde encontrarás cursos Java Online desde Niveles Básicos, Intermedios y Avanzados, y así te conviertas en un experto programador Java.

Además agregamos nuevos cursos para que continúes con tu preparación como programador Java profesional. A continuación te presentamos nuestro listado de cursos:

- Lógica de Programación
- Fundamentos de Java
- Programación con Java
- Java con JDBC
- HTML, CSS y JavaScript
- Servlets y JSP's
- Struts Framework
- Hibernate Framework
- Spring Framework
- JavaServer Faces
- Java EE (EJB, JPA y Web Services)
- JBoss Administration
- Android con Java
- HTML5 y CSS3

Datos de Contacto:

Sitio Web: www.globalmentoring.com.mx
Email: informes@globalmentoring.com.mx