ADSI

Arreglos Java - POO

Instructor: Gustavo Adolfo Rodríguez Q.

garodriguez335@misena.edu.co

ADSI

ARREGLOS CON JAVA

Un array o arreglo, es una estructura de datos que nos permite almacenar un conjunto de datos de un mismo tipo. El tamaño de los arrays se declara en un primer momento y no puede cambiar en tiempo de ejecución como puede producirse en otros lenguajes, razón por la cual se dice que *los arreglos en Java son dinámicos, pero no extensibles.* El tamaño de un arreglo permanecerá invariante a través de todo su ciclo de vida.

El acceso a los elementos de un arreglo se realiza mediante la utilización de índices. Los índices para un arreglo de ***n*** elementos en Java van desde ***0*** hasta ***n-1***.

1. DECLARACION DE UN ARREGLO

En Java, para declarar un array o arreglo de elementos se debe utilizar la siguiente estructura:

Tipo_de_Datos nombre_arreglo []

Por ejemplo, para declarar un arreglo de datos de tipo int utilizamos:

```
int arrayInt[];
```

Para declara un arreglo de datos de tipo booleano utilizamos:

```
boolean arrayBoolean[];
```

Para declara un arreglo de datos de tipo String utilizamos:

```
String arrayString[];
```

2. INSTANCIACIÓN DE UN ARREGLO

En Java, los arreglos en sí mismos son tratados como objetos, pero su contenido si depende del tipo de dato que se haya declarado.

Puesto que los arreglos en Java son considerados como objetos, para su instanciación también se utiliza el operador ***new.***

Por ejemplo, para instanciar el arreglo de datos de tipo int declarado anteriormente debemos utilizar:

```
arrayInt = new int[5];
```

En este caso, hemos creado un arreglo de datos de tipo int con 5 posiciones.

Para instanciar el arreglo de datos de tipo boolean declarado anteriormente debemos utilizar:

```
arrayBoolean = new boolean[10];
```

En este caso hemos creado un arreglo de datos de tipo boolean con 10 posiciones.

Para instanciar el arreglo de datos de tipo String declarado anteriormente debemos utilizar:

```
arrayString = new String[7];
```

En este caso hemos creado un arreglo de datos de tipo String con 7 posiciones.

NOTA: Es necesario tener presente que cada vez que instanciamos un arreglo, estamos construyendo simplemente la estructura que contendrá dentro de sí a los elementos, ***más no estamos instanciando cada uno de los elementos dentro de esa estructura.***

Para completar el proceso, debemos realizar la inicialización o instanciación, según sea el caso, de los elementos del arreglo. Este proceso se describe a continuación.

3. INICIALIZACION DE LOS ELEMENTOS DE UN ARREGLO

Recordemos que en Java existen básicamente dos tipos de datos los cuales son:

- Datos primitivos: short, int, log, float, double, boolean, char
- Datos complejos u objetos: String, Integer, Scanner, Random, List, Map, Point, etc.

Recordemos además que cada vez que instanciamos un arreglo, estamos creando simplemente la estructura para almacenar los elementos más no los elementos como tal.

Si no se realiza el proceso de inicialización del arreglo, cada uno de los elementos tomará un valor denominado ***valor por defecto*** , los valores por defecto para cada tipo de datos se muestran en la siguiente tabla:

TIPO DE DATO	VALOR POR DEFECTO
byte	0x00
short	0
char	'\u0000'
int	0
long	0L
float	0.0F
double	0.0D
boolean	false
Objeto	**null**

Para asignar un valor diferente al valor por defecto a los elementos de un arreglo debemos acceder a este elemento y realizar la asignación pertinente, por ejemplo:

```
String[] arregloCadenas;         //declaración del arreglo
arregloCadenas = new String[4]; //instanciación del arreglo
arregloCadenas[0] = "SENA";      //inicialización índice 0
arregloCadenas[1] = "CENTRO";    //inicialización índice 1
arregloCadenas[2] = "COMERCIO"; //inicialización índice 2
arregloCadenas[3] = "SERVICIOS";//inicialización índice 3
```

En el ejemplo anterior, se ha creado un arreglo de 4 elementos de tipo String, puesto que String es un tipo complejo o tipo objeto, el valor por defecto asignado a cada elemento en el momento de la instanciación del arreglo es ***null***. Sin embargo, hemos cambiado el ***null*** por "SENA", "CENTRO", "COMERCIO", "SERVICIOS" para cada uno de los elementos del arreglo respectivamente.

4. INDIZACIÓN DE UN ARREGLO

Como se ha dicho anteriormente, un arreglo tiene un tamaño establecido en el momento de la instanciación o creación. Por tanto, tendremos una estructura de datos como aparece en la siguiente imagen:

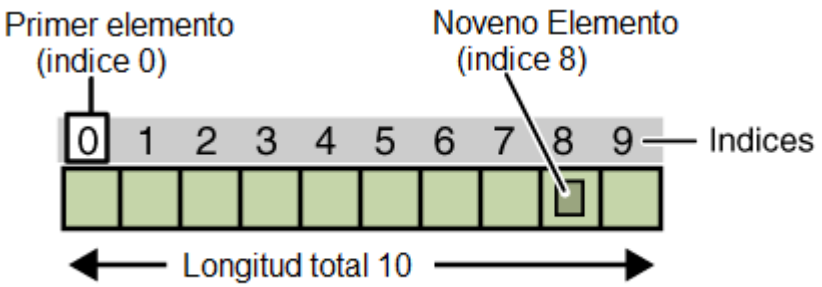
índice **0**	índice 1	índice 2	índice ...	índice ...	índice **n - 1**

elemento **1**	elemento **2**	elemento **3**	....	...	elemento **n**

Si un arreglo ha sido creado con un tamaño **n**, entonces sus índices van desde **0** hasta **n-1**.

Por ejemplo, un arreglo de enteros de 10 posiciones, tendrá los índices 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9

Los arrays se numeran desde el elemento ***cero***, que sería el primer elemento, hasta el ***tamaño-1*** que sería el último elemento. Es decir, para el caso del arreglo de diez elementos, el primer elemento sería el cero y el último elemento sería el nueve, como se muestra en la siguiente figura:



5. ATRIBUTOS DE UN ARREGLO EN JAVA

Para conocer exactamente, a dimensión o longitud del arreglo podemos utilizar el atributo ***lenght*** que nos devolverá un número entero.

Cuando intentamos acceder a un elemento fuera del rango de índices del arreglo, Java nos lanzará una excepción del tipo

java.lang.ArrayIndexOutOfBoundsException

6. ACCESO A LOS ELEMENTOS DE UN ARREGLO

Para acceder a los elementos de un arreglo se debe utilizar el nombre del arreglo y a continuación, encerrado entre corchetes, el índice al cual queremos acceder. Por ejemplo

```
int[] enteros = new int[5];

enteros[0] = 5;
enteros[1] = 12;
enteros[2] = 3;
enteros[3] = 56;
enteros[4] = 0;
```

En este caso, hemos creado un arreglo de enteros y hemos asignado valores a cada una de sus posiciones.

Tenga en cuenta que para este caso particular, el arreglo tiene una longitud 5 t los índices van desde el 0 hasta el 4.

A continuación se muestra un programa que crea un arreglo de enteros, inicializa cada una de las posiciones del arreglo e imprime los valores almacenados en la consola.

```
/**
 *
 * @author Jaime
 */
class ArrayDemo {
     public static void main(String[] args) {
          int[] unArrego;              // declaramos un array de enteros
          unArrego = new int[10];      // instanciamos el array de 10 posiciones
          unArrego[0] = 100; // inicializamos el primer elemento
          unArrego[1] = 200; // inicializamos el segundo elemento
          unArrego[2] = 300; // inicializamos el tercer elemento
          unArrego[3] = 400; // etc
          unArrego[4] = 500;
          unArrego[5] = 600;
          unArrego[6] = 700;
          unArrego[7] = 800;
          unArrego[8] = 900;
          unArrego[9] = 1000;

          System.out.println("Elemento en indice  0: " + unArrego[0]);
          System.out.println("Elemento en indice  1: " + unArrego[1]);
          System.out.println("Elemento en indice  2: " + unArrego[2]);
          System.out.println("Elemento en indice  3: " + unArrego[3]);
          System.out.println("Elemento en indice  4: " + unArrego[4]);
          System.out.println("Elemento en indice  5: " + unArrego[5]);
          System.out.println("Elemento en indice  6: " + unArrego[6]);
          System.out.println("Elemento en indice  7: " + unArrego[7]);
          System.out.println("Elemento en indice  8: " + unArrego[8]);
          System.out.println("Elemento en indice  9: " + unArrego[9]);
     }
}
```

En la vida real, para imprimir los valores de cada una de las posiciones del arreglo, se utilizan sentencias cíclicas como se verá a continuación.

7. RECORRIDO DE UN ARREGLO

Para realizar el recorrido de un arreglo podemos utilizar un ciclo ***for*** que inicie en **0** y termine en **lenght -1** como en el siguiente ejemplo:

```
for (int i = 0; i < unArrego.length; i++) {
    int j = unArrego[i];
    System.out.println("numero posicion "+i+" vale "+ j);
}
```

O también podemos utilizar el método abreviado de Java para recorrer arreglos como se muestra en el siguiente ejemplo:

```
char[] car = new char[3];

for (char c : car) {
    System.out.println("char" + c);
}
```

Esta forma es mucho más corta, pero no nos permite referirnos a una posición específica del arreglo.

8. ARREGLOS MULTIDIMENSIONALES

Los arreglos multidimensionales en Java están permitidos y son considerados como un arreglo de otros arreglos. Por ejemplo, para crear un arreglo bidimensional de cadenas de caracteres e inicializar cada uno de sus elementos utilizamos:

```
String[][] bidimensional;                    //declaración

bidimensional = new String[3][4];            //instanciación

bidimensional[0][0] = "fila 0 columna 0";    //inicialización
bidimensional[0][1] = "fila 0 columna 1";    //inicialización
bidimensional[0][2] = "fila 0 columna 2";    //inicialización
bidimensional[0][3] = "fila 0 columna 3";    //inicialización

bidimensional[1][0] = "fila 1 columna 0";    //inicialización
bidimensional[1][1] = "fila 1 columna 1";    //inicialización
bidimensional[1][2] = "fila 1 columna 2";    //inicialización
bidimensional[1][3] = "fila 1 columna 3";    //inicialización

bidimensional[2][0] = "fila 2 columna 0";    //inicialización
bidimensional[2][1] = "fila 2 columna 1";    //inicialización
bidimensional[2][2] = "fila 2 columna 2";    //inicialización
bidimensional[2][3] = "fila 2 columna 3";    //inicialización
```

Las líneas de código anteriores podrían reemplazarse por las siguientes obteniendo un resultado idéntico pero mediante la utilización de ciclos ***for*** así:

```
String[][] bidimensional;                    //declaración
bidimensional = new String[3][4];            //instanciación

for (int i = 0; i < bidimensional.length; i++) {//primera dimension
    String[] strings = bidimensional[i];
    for (int j = 0; j < strings.length; j++) {  //segunda dimension
        strings[j] = "fila "+i+" columna "+j;   //asignar el valor

    }
}
```