APRENDERAPROGRAMAR.COM

ARRAYS O ARREGLOS. EJEMPLOS CON OBJETOS Y TIPOS PRIMITIVOS. CAMPO LENGTH. RESUMEN TIPOS DE COLECCIONES JAVA. (CU00669B)

Sección: Cursos
Categoría: Curso “Aprender programación Java desde cero”

Fecha revisión: 2029

Resumen: Entrega nº69 curso Aprender programación Java desde cero.

Autor: Alex Rodríguez

ARRAYS, ARREGLOS O FORMACIONES EN JAVA

Algunas clases que hemos citado, como ArrayList o LinkedList, se denominan colecciones de tamaño flexible porque permiten modificar dinámicamente el número de ítems que contienen, es decir, ampliarlo o reducirlo. A veces nos referiremos a estas colecciones como arrays dinámicos porque son similares a los arrays dinámicos que se emplean en otros lenguajes de programación.

Los arrays dinámicos son muy potentes porque permiten crear colecciones de tamaño variable que podemos agrandar o empequeñecer en función de nuestras necesidades.

Sin embargo, cuando se conoce el número de elementos en una colección y éste va a ser invariable (por ejemplo, los 12 meses del año), será **más eficiente** utilizar una colección de tamaño fijo a la que denominamos array estático, arreglo estático, formación o vector. Si utilizamos el término array o arreglo, a secas, entenderemos que hacemos alusión a un array estático. El uso de arrays estáticos tiene ventajas e inconvenientes:

VENTAJAS	INCONVENIENTES
- **Acceso y operación con elementos más eficiente.** - **Permiten almacenar tanto objetos como tipos primitivos directamente.**	- Rigidez. No se pueden redimensionar (aunque sí copiar a otros arrays estáticos de mayor tamaño).

La sintaxis a emplear con arrays la exponemos a continuación.

```
private TipoPrimitivoUObjeto [ ] nombreDelArray;   //Declaración: reserva espacio de memoria

nombreDelArray = new TipoPrimitivoUObjeto [numero]; //Creación del array

private int [ ] miArrayDeNumeros = { 2, -3, 4, 7, -10 }; //Sintaxis ejemplo declarar y crear un array en una línea

// Ejemplo de uso con tipos primitivos aprenderaprogramar.com
private int [ ] cochesHorasDelDia;
cochesHorasDelDia = new int [24]; // Creamos un array de enteros con índices entre el 0 y el 23
cochesHorasDelDia [9] = 4521; //Ejemplo de asignación
cochesHorasDelDia [hora] = 4521; //Ejemplo de asignación
cochesHorasDelDia [23]++ ; //Ejemplo de asignación que equivale a cochesHorasDelDia[23] +=1;
private boolean[ ] superado;
superado = new boolean [1000];
superado [832] = false;
```

```
//Ejemplo de uso con tipos objeto aprenderaprogramar.com
private Persona [ ] Grupo3A;
Grupo3A = new Persona [50]; //Creamos un array de objetos Persona con índices entre el 0 y el 49
private String [ ] nombre;
nombre = new String [200];
nombre [37] = "Juan Antonio";

//Declarar el array y crear el objeto en una misma línea
Tipo [ ] nombreDelArray = new Tipo [número];

//Ejemplo de código que podríamos incluir en un método main aprenderaprogramar.com
int [ ] arrayEnteros = {2, 3, 1, 7, -1}; //El 2 es el elemento con índice 0 y el -1 el elemento con índice 4
String [ ] misNombres = new String [10];
misNombres [4] = "José Alberto Pérez";
System.out.println (misNombres[4]);
```

La numeración de los índices de los arrays va desde cero hasta (número de elementos – 1). Tener en cuenta que la variable que es el nombre del array, p.ej. misNombres, lo que contiene es un puntero o referencia al objeto que es en sí el array. **Un array en Java puede considerarse un "objeto especial".** Se crea con la sentencia *new* como el resto de objetos, pero sin embargo no hay una clase específica en Java que defina el tipo de los arrays. Dada una declaración del tipo *int [ ] cochesPorHora = new int [24];*, hay ciertos errores habituales frente a los que hay que estar atentos:

a) Pensar que los índices van de 0 a 24. Falso: van de 0 a 23.

b) Pensar que el número de elementos total es 23. Falso: son 24.

c) Usar cochesPorHora [24]. El índice 24 no existe y el uso de esa expresión daría lugar a un error en tiempo de ejecución del tipo "ArrayIndexOutOfBoundsException".

Sobre un objeto que es un elemento de un array se pueden invocar métodos. Por ejemplo *System.out.println (persona[17].getNombre());*. Por otro lado, un elemento de un array puede almacenar un objeto anónimo. Por ejemplo *persona [47] = new Persona ("Juan", "Pérez Hernández", 39, 1.75);*, donde los elementos entre paréntesis son los parámetros requeridos por el constructor de la clase Persona.

Para recorrer arrays, dado que se conoce el número de iteraciones, es habitual usar for tradicionales.

CAMPO LENGTH PARA SABER EL NÚMERO DE ELEMENTOS DE UN ARRAY

La sintaxis *nombreDelArray.length* nos devuelve un entero (int) con el número de elementos que forman el array. Fíjate que después de length no aparecen paréntesis, lo que indica que no estamos invocando un método, sino accediendo a un atributo del array. El acceso a este atributo es posible porque el API de Java mantiene este atributo como público: **si fuera privado no podríamos acceder a él.** Ejemplos de uso podrían ser los siguientes:

```
System.out.println ("El número de elementos en el array misNombres es de " + misNombres.length );
for (int i = 0; i < misNombres.length; i++) {   System.out.println ("Nombre " + i + ": " + misNombres[i]; }
```

En este ejemplo usamos i <, es decir, menor estricto, porque length nos devuelve siempre un valor n+1 para unos índices comprendidos entre 0 y n.

USO DE CICLOS FOR EACH CON ARRAYS

Es posible usar ciclos for – each con arrays. Por ejemplo:

```
//Ejemplo aprenderaprogramar.com
for (int tmpItem : cochesPorHora) { System.out.println ("Número: " + tmpItem); }

String [ ] misNombres = new String [10];
For (String tmpObjeto : misNombres) { System.out.println  (tmpObjeto); }
//Nota: en el caso de que un ítem objeto no tenga contenido por pantalla saldrá null
```

El for each tiene ventajas como el poder recorrer una colección de la que se desconoce su tamaño, pero también inconvenientes como carecer de una variable contadora en el ciclo. Si quisiéramos contar tendríamos que introducir una variable contadora, cosa que con el for normal ya tenemos con la propia definición del bucle. Dado que en un array podemos conocer con facilidad el número de elementos de que consta y disponer de forma automática de la variable contadora, será más habitual el uso de for tradicionales con arrays que el uso del for extendido.

EJERCICIO

Crea una clase con el método main donde declares una variable de tipo array de Strings que contenga los doce meses del año, en minúsculas y declarados en una sola línea. A continuación declara una variable mesSecreto de tipo String, y hazla igual a un elemento del array (por ejemplo mesSecreto = mes[3]. El programa debe pedir al usuario que adivine el mes secreto y si acierta mostrar un mensaje y si no pedir que vuelva a intentar adivinar el mes secreto. Puedes comprobar si tu código es correcto consultando en los foros aprenderaprogramar.com.

Un ejemplo de ejecución del programa podría ser este:

Adivine el mes secreto. Introduzca el nombre del mes en minúsculas: febrero

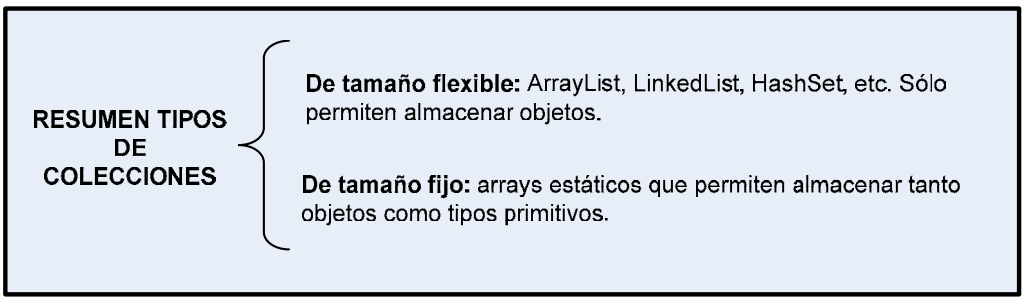
No ha acertado. Intente adivinarlo introduciendo otro mes: agosto

No ha acertado. Intente adivinarlo introduciendo otro mes: octubre

¡Ha acertado!

RESUMEN DE TIPOS DE COLECCIONES JAVA

El siguiente esquema resume los tipos de colecciones disponibles en Java.



No hay que usar con preferencia uno u otro tipo. Para cada caso, conviene estudiar las circunstancias y elegir el más adecuado.

Próxima entrega: CU00670B

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:

http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=68&Itemid=188