**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[9]. 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.

|  |  |
| --- | --- |
|  | |
| **RESUMEN DE TIPOS DE COLECCIONES**  **EN JAVA aprenderaprogramar.com** | **De tamaño flexible:**ArrayList, LinkedList, HashSet, etc. Sólo permiten almacenar objetos. |
| **De tamaño fijo:** arrays estáticos que permiten almacenar tanto objetos como tipos primitivos. |

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

ArrayList en Java

La clase ArrayList en Java, es una clase que permite almacenar datos en memoria de forma similar a los Arrays, con la ventaja de que el numero de elementos que almacena, lo hace de forma dinámica, es decir, que no es necesario declarar su tamaño como pasa con los Arrays. Para todos aquellos que hayáis estudiado en alguna asignatura las estructuras de datos de las Pilas, Colas, Listas, Arboles (AVL, B, B+, B\*) etc. hay decir que los ArrayList "tiran por tierra" toda la teoria que hay detrás de esas estructuras de datos ya que los ArrayList nos permiten añadir, eliminar y modificar elementos (que pueden ser objetos o elementos atómicos) de forma trasparente para el programador. Antes de pasar a explicar el manejo de los ArrayList

Los principales métodos para trabajar con los ArrayList son los siguientes:

*// Declaración de un ArrayList de "String". Puede ser de cualquier otro Elemento u Objeto (float, Boolean, Object, ...)*

ArrayList<String> nombreArrayList = new ArrayList<String>();

*// Añade el elemento al ArrayList*

nombreArrayList.add("Elemento");

*// Añade el elemento al ArrayList en la posición 'n'*

nombreArrayList.add(n, "Elemento 2");

*// Devuelve el numero de elementos del ArrayList*

nombreArrayList.size();

*// Devuelve el elemento que esta en la posición '2' del ArrayList*

nombreArrayList.get(2);

*// Comprueba se existe del elemento ('Elemento') que se le pasa como parametro*

nombreArrayList.contains("Elemento");

*// Devuelve la posición de la primera ocurrencia ('Elemento') en el ArrayList*

nombreArrayList.indexOf("Elemento");

*// Devuelve la posición de la última ocurrencia ('Elemento') en el ArrayList*

nombreArrayList.lastIndexOf("Elemento");

*// Borra el elemento de la posición '5' del ArrayList*

nombreArrayList.remove(5);

*// Borra la primera ocurrencia del 'Elemento' que se le pasa como parametro.*

nombreArrayList.remove("Elemento");

*//Borra todos los elementos de ArrayList*

nombreArrayList.clear();

*// Devuelve True si el ArrayList esta vacio. Sino Devuelve False*

nombreArrayList.isEmpty();

*// Copiar un ArrayList*

ArrayList arrayListCopia = (ArrayList) nombreArrayList.clone();

*// Pasa el ArrayList a un Array*

Object[] array = nombreArrayList.toArray();

Otra cosa muy importante a la hora de trabajar con los ArrayList son los "Iteradores" ([Iterator](http://docs.oracle.com/javase/1.4.2/docs/api/java/util/Iterator.html" \t "_blank)). Los Iteradores sirven para recorrer los ArrayList y poder trabajar con ellos. Los Iteradores solo tienen tres métodos que son el *"hasNext()"* para comprobar que siguen quedando elementos en el iterador, el *"next()"*  para que nos de el siguiente elemento del iterador; y el *"remove()"* que sirve para eliminar el elemento del Iterador.

Bueno, si esto no te ha quedado muy claro, pasamos a poner el primer ejemplo. En el siguiente fragmento de código, declaramos un ArrayList de Strings y lo rellenamos con 10 Strings (Elemento i). Esto lo hacemos con el método *"add()"*. Después añadimos un nuevo elemento al ArrayList en la posición '2' (con el metodo *"add(posición,elemento)"*) que le llamaremos "Elemento 3" y posteriormente imprimiremos el contenido del ArrayList, recorriendolo con un Iterador. El fragmento de este código es el siguiente:

*// Declaración el ArrayList*

ArrayList<String> nombreArrayList = new ArrayList<String>();

*// Añadimos 10 Elementos en el ArrayList*

for (int i=1; i<=10; i++){

nombreArrayList.add("Elemento "+i);

}

*// Añadimos un nuevo elemento al ArrayList en la posición 2*

nombreArrayList.add(2, "Elemento 3");

*// Declaramos el Iterador e imprimimos los Elementos del ArrayList*

Iterator<String> nombreIterator = nombreArrayList.iterator();

while(nombreIterator.hasNext()){

String elemento = nombreIterator.next();

System.out.print(elemento+" / ");

}

Como se observa en el resultado tenemos repetido el elemento "Elemento 3" dos veces y esto lo hemos puesto a proposito para mostrar el siguiente ejemplo. Ahora para seguir trabajando con los ArrayList, lo que vamos ha hacer es mostrar el numero de elementos que tiene el ArrayList y después eliminaremos el primer elemento del ArrayList y los elementos del ArrayList que sean iguales a "Elemento 3", que por eso lo hemos puesto repetido. El "Elemento 3" lo eliminaremos con el metodo "remove()" del iterador. A continuación mostramos el código que realiza lo descrito:

*// Recordar que previamente ya hemos declarado el ArrayList y el Iterator de la siguiente forma:*

*// ArrayList<String> nombreArrayList = new ArrayList<String>();*

*// Iterator<String> nombreIterator = nombreArrayList.iterator();*

*// Obtenemos el numero de elementos del ArrayList*

int numElementos = nombreArrayList.size();

System.out.println("nnEl ArrayList tiene "+numElementos+" elementos");

*// Eliminamos el primer elemento del ArrayList, es decir el que ocupa la posición '0'*

System.out.println("n... Eliminamos el primer elemento del ArrayList ("+nombreArrayList.get(0)+")...");

nombreArrayList.remove(0);

*// Eliminamos los elementos de ArrayList que sean iguales a "Elemento 3"*

System.out.println("n... Eliminamos los elementos de ArrayList que sean iguales a "Elemento 3" ...");

nombreIterator = nombreArrayList.iterator();

while(nombreIterator.hasNext()){

String elemento = nombreIterator.next();

if(elemento.equals("Elemento 3"))

nombreIterator.remove(); *// Eliminamos el Elemento que hemos obtenido del Iterator*

}

*// Imprimimos el ArrayList despues de eliminar los elementos iguales a "Elemento 3"*

System.out.println("nImprimimos los elementos del ArrayList tras realizar las eliminaciones: ");

nombreIterator = nombreArrayList.iterator();

while(nombreIterator.hasNext()){

String elemento = nombreIterator.next();

System.out.print(elemento+" / ");

}

*// Mostramos el numero de elementos que tiene el ArrayList tras las eliminaciones:*

numElementos = nombreArrayList.size();

System.out.println("nNumero de elementos del ArrayList tras las eliminaciones = "+numElementos);

Si os fijais, hemos eliminado 3 elementos del ArrayList de dos formas distintas, preguntando por la posición que ocupa un elemento en el ArrayList y preguntando por el contenido de algún elemento del ArrayList. Como se observa es muy importante saber manejar los Iteradores ya que con ellos podemos tratar los elementos del ArrayList.

Bueno todo lo que hemos visto esta muy bien, pero por lo general los ArrayList se suelen utilizar con objetos más que con estructuras atómicas de datos, ya que los ArrayList en Java son estructuras muy potentes y sencillas de manejar.

Ahora vamos a poner un ejemplo de la utilización de ArrayList con Objetos. Para ello nos vamos ha crear una clase llamada "PartidoFutbol" que utilizaremos para crearnos objetos de esa clase que almacenaremos en el ArrayList. En primer lugar mostramos la clase "PartidoFutbol" que es la siguiente:

package ArrayList\_Objetos;

public class PartidoFutbol {

private String equipoLocal;

private String equipoVisitante;

private int golesLocal;

private int golesVisitante;

public String getEquipoLocal() {

return equipoLocal;

}

public void setEquipoLocal(String equipoLocal) {

this.equipoLocal = equipoLocal;

}

public String getEquipoVisitante() {

return equipoVisitante;

}

public void setEquipoVisitante(String equipoVisitante) {

this.equipoVisitante = equipoVisitante;

}

public int getGolesLocal() {

return golesLocal;

}

public void setGolesLocal(int golesLocal) {

this.golesLocal = golesLocal;

}

public int getGolesVisitante() {

return golesVisitante;

}

public void setGolesVisitante(int golesVisitante) {

this.golesVisitante = golesVisitante;

}

}

Lo que vamos ha hacer ahora en este ejemplo, es leer desde un fichero de texto, una serie de partidos de fútbol que guardaremos en un ArrayList de objetos "PartidoFutbol". Al utilizar el ArrayList, nos da igual el numero de partidos de fútbol que haya en el fichero de texto ya que los ArrayList como vimos al principio son dinámicos. Para este caso os adelante que hay 50 partidos de fútbol en el fichero, pero nos daria igual cuantos podría haber.

A continuación se muestra el código de la lectura de los partidos de fútbol y como los almacenamos en un ArrayList. Por curiosidad, un partido de fútbol esta guardado en el fichero con la siguiente estructura: *equipoLocal::equipoVisitante::golesLocal::golesVisitante* *// Nos creamos un ArrayList de objetos de la Clase "PartidoFutbol"*

System.out.println("... Nos creamos un ArrayList de objetos de la Clase "PartidoFutbol" ...");

ArrayList<PartidoFutbol> partidos = new ArrayList<PartidoFutbol>();

*// Instanciamos el fichero donde estan los datos*

File fichero = new File(nombreFichero);

Scanner s = null;

try {

*// Leemos el contenido del fichero*

System.out.println("... Leemos el contenido del fichero ...");

s = new Scanner(fichero);

*// Obtengo los datos de los partidos de fútbol del fichero*

while (s.hasNextLine()){

String linea = s.nextLine(); *// Obtengo una linea del fichero (un partido de fútbol)*

String [] cortarString = linea.split("::"); *// Obtengo los datos del partido de futbol*

PartidoFutbol partido = new PartidoFutbol(); *// Creo un objeto de la clase "PartidoFutbol"*

*// Pongo los atributos al objeto "partido"*

partido.setEquipoLocal(cortarString[0]);

partido.setEquipoVisitante(cortarString[1]);

partido.setGolesLocal(Integer.parseInt(cortarString[2]));

partido.setGolesVisitante(Integer.parseInt(cortarString[3]));

*// Añadimos el objeto "partido" al ArrayList*

partidos.add(partido);

}

} catch (Exception e) {

e.printStackTrace();

} finally{

try {

if (s != null)

s.close();

} catch (Exception e2) {

e2.printStackTrace();

}

}

System.out.println("... Guardados "+partidos.size()+" partidos de fútbol ...");

Con este código que mostramos tambien se puede ver de que forma se puede leer un fichero de texto y tratar los datos  . Una vez ejecutado este código tenemos guardado en el ArrayList *"partidos"*, 50 partidos de fútbol. Ahora recorremos el ArrayList con un Iterator y mostramos por pantalla todos los resultados de fútbol. Esto lo hacemos igual que antes, con el Iterator:

System.out.println("n... Resultados de los partidos de Futbol ...");

Iterator<PartidoFutbol> itrPartidos = partidos.iterator();

while(itrPartidos.hasNext()){

PartidoFutbol partido = itrPartidos.next();

System.out.println(partido.getEquipoLocal() + " "

+ partido.getGolesLocal() + "-"

+ partido.getGolesVisitante() + " "

+ partido.getEquipoVisitante());

}

Como salida a la ejecución de este código tenemos lo siguiente:

... Resultados de los partidos de Futbol ...

Valencia 1-0 Betis

Betis 1-0 Osasuna

Atletico Madrid 0-0 Zaragoza

Zaragoza 2-2 Valencia

Mallorca 0-1 Deportivo

Deportivo 1-0 Atletico Madrid

.

.

.

Mallorca 1-4 Malaga

Malaga 0-2 Atletico Madrid

Ahora vamos a eliminar del ArrayList, todos los partidos que no tengan como resultado un empate. De esta forma trabajaremos de nuevo con el iterator. Esto lo hacemos de la siguiente forma:

*// Eliminamos los partidos de futbol del ArrayList, cuyo resultado*

*// no sea un empate*

System.out.println("n... Tamaño del ArrayList antes de eliminar partidos de futbol = "+partidos.size()+" ...");

System.out.println("n... Eliminamos los partidos de futbol cuyo resultado no sea un empate ...");

itrPartidos = partidos.iterator();

while(itrPartidos.hasNext()){

PartidoFutbol partido = itrPartidos.next();

*// Si los goles no son iguale, eliminamos el partido*

if(partido.getGolesLocal() != partido.getGolesVisitante())

itrPartidos.remove();

}

*// Imprimimos los elementos del ArrayList*

System.out.println("n... Tamaño del ArrayList despues de eliminar partidos de futbol = "+partidos.size()+" ...");

System.out.println("n... Resultados de los partidos de Futbol con empate ...");

itrPartidos = partidos.iterator();

while(itrPartidos.hasNext()){

PartidoFutbol partido = itrPartidos.next();

System.out.println(partido.getEquipoLocal() + " "

+ partido.getGolesLocal() + "-"

+ partido.getGolesVisitante() + " "

+ partido.getEquipoVisitante());

}

Como resultado al ejecutar este código tenemos lo siguiente, que como se observa el ArrayList solo se ha quedado con los partidos de fútbol en los que ha habido un empate:

... Tamaño del ArrayList antes de eliminar partidos de futbol = 50 ...

... Eliminamos los partidos de futbol cuyo resultado no sea un empate ...

... Tamaño del ArrayList despues de eliminar partidos de futbol = 16 ...

... Resultados de los partidos de Futbol con empate ...

Atletico Madrid 0-0 Zaragoza

Zaragoza 2-2 Valencia

Alaves 0-0 Barcelona

Getafe 1-1 Athletic

Villarreal 1-1 Sevilla

Betis 0-0 Zaragoza

Valencia 2-2 Deportivo

Deportivo 1-1 Betis

Barcelona 2-2 Valencia

Getafe 1-1 Mallorca

Malaga 0-0 Alaves

Cadiz 1-1 Villarreal

Sevilla 0-0 Cadiz

Zaragoza 1-1 Deportivo

Barcelona 2-2 Zaragoza

Real Sociedad 1-1 Betis