Java ArrayList. Estructura dinámica de datos

DECLARACIÓN Y CREACIÓN DE UN ARRAYLIST

De forma general un ArrayList en Java se crea de la siguiente forma:

```
ArrayList nombreArray = new ArrayList();
```

Esta instrucción crea el ArrayList *nombreArray* vacío.
Un arrayList declarado así puede contener objetos de cualquier tipo.
Por ejemplo:

```
ArrayList a = new ArrayList();
a.add("Lenguaje");
a.add(3);
a.add('a');
a.add(23.5);
```

Los elementos del arrayList a son:
"Lenguaje" 3 'a' Y 23.5
Es decir, **un ArrayList puede contener objetos de tipos distintos**.

En este ejemplo, el primer objeto que se añade es el String "Lenguaje". El resto no son objetos. Son datos de tipos básicos pero el compilador los convierte automáticamente en objetos de su **clase envolvente** (clase contenedora o wrapper) antes de añadirlos al array.
Un array al que se le pueden asignar elementos de distintos tipos puede tener alguna complicación a la hora de trabajar con él. Por eso, una alternativa a esta declara
ción es indicar el tipo de objetos que contiene. En este caso, el array solo podrá contener objetos de ese tipo.
De forma general:

```
ArrayList<tipo> nombreArray = new ArrayList<tipo>();
```

tipo debe ser una clase. Indica el tipo de objetos que contendrá el array.
No se pueden usar tipos primitivos. Para un tipo primitivo se debe utilizar su clase envolvente.
Por ejemplo:

```
ArrayList<Integer> numeros = new ArrayList<Integer>();
```

Crea el array *numeros* de enteros.

MÉTODOS DE ARRAYLIST

Algunos métodos que proporciona ArrayList son:

MÉTODO	DESCRIPCIÓN
size()	Devuelve el número de elementos (int)
add(X)	Añade el objeto X al final. Devuelve true.
add(posición, X)	Inserta el objeto X en la posición indicada.exception si no existe la posicion IndexOutOfBoundsException
get(posicion)	Devuelve el elemento que está en la posición indicada.Empieza posicion 0.exception si no existe la posicion IndexOutOfBoundsException

remove(posicion)	Elimina el elemento que se encuentra en la posición indicada. Devuelve el elemento eliminado. exception si no existe la posicion IndexOutOfBoundsException
remove(X)	Elimina la primera ocurrencia del objeto X. Devuelve true si el elemento está en la lista.
clear()	Elimina todos los elementos.
set(posición, X)	Sustituye el elemento que se encuentra en la posición indicada por el objeto X. Devuelve el elemento sustituido.exception si posicion no existe
contains(X)	Comprueba si la colección contiene al objeto X. Devuelve true o false.
indexOf(X)	Devuelve la posición del objeto X. Si no existe devuelve -1

Los puedes consultar todos en:
http://docs.oracle.com/javase/9/docs/api/java/util/ArrayList.html

RECORRER UN ARRAYLIST

Podemos recorrerlo de forma clásica con un **bucle for:**

```
for(int i = 0;i<array.size();i++){
        System.out.println(array.get(i));
}
```

Con un **bucle foreach:**
Si suponemos el array de enteros llamado *numeros:*

```
for(Integer i: numeros){
 System.out.println(i);
}
```

Si el array contiene objetos de tipos distintos o desconocemos el tipo:

```
for(Object o: nombreArray){

System.out.println(o);
}
```

Utilizando un **objeto Iterator**.

http://docs.oracle.com/javase/9/docs/api/java/util/Iterator.html
La ventaja de utilizar un Iterador es que no necesitamos indicar el tipo de objetos que contiene el array.
Iterator tiene como métodos:
hasNext: devuelve true si hay más elementos en el array.
Next: devuelve el siguiente objeto contenido en el array.
Ejemplo:

```
ArrayList<Integer> numeros = new ArrayList<Integer>();
.....
//se insertan elementos
.....
```

```
Iterator it = numeros.iterator();//se crea el iterador it para el array numeros
while(it.hasNext()) //mientras queden elementos
System.out.println(it.next());//se obtienen y se muestran
```

EJEMPLOS DE USO DE ARRAYLIST

Ejemplo 1:

```
ArrayList<String> nombres = new ArrayList<String>();
nombres.add("Ana");
nombres.add("Luisa");
nombres.add("Felipe");
System.out.println(nombres);//[Ana, Luisa, Felipe]
nombres.add(1, "Pablo");
System.out.println(nombres);// [Ana, Pablo, Luisa, Felipe]
nombres.remove(0);
System.out.println(nombres);// [Pablo, Luisa, Felipe]
nombres.set(0,"Alfonso");
System.out.println(nombres);// [Alfonso, Luisa, Felipe]
String s = nombres.get(1);
String ultimo = nombres.get(nombres.size() - 1);
System.out.println(s + " " + ultimo);// Luisa Felipe
```

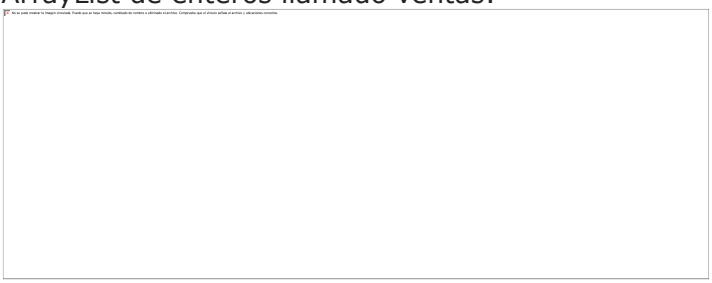
Ejemplo 2: Escribe un programa que lea números enteros y los guarde en un ArrayList hasta que se lea un 0 y muestra los números leídos, su suma y su media.

```
import java.util.*;

public class ArrayList2 {

 public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  ArrayList<Integer> numeros = new ArrayList<Integer>();
  int n;
  do {
   System.out.println("Introduce números enteros. 0 para acabar: ");
   System.out.println("Numero: ");
   n = sc.nextInt();
   if (n != 0)
    numeros.add(n);
  }while (n != 0);

  System.out.println("Ha introducido: " + numeros.size() + " números:");

  //mostrar el arrayList completo
  System.out.println(numeros);

  //recorrido usando un iterador para mostrar un elemento por línea
  Iterator it = numeros.iterator();
  while(it.hasNext())
    System.out.println(it.next());

  //recorrido usando foreach para sumar los elementos
```

```
  double suma = 0;
  for(Integer i: numeros){
   suma = suma + i;
  }
  System.out.println("Suma: " + suma);
  System.out.println("Media: " + suma/numeros.size());
 }
}
```

COPIAR UN ARRAYLIST

El nombre de un ArrayList contiene la referencia al ArrayList, es decir, la dirección de memoria donde se encuentra el ArrayList, igual que sucede con los arrays estáticos.
Si disponemos de un ArrayList de enteros llamado ventas:



La instrucción:
ArrayList<Integer> ventas1 = ventas;
No copia el array ventas en el nuevo array ventas1 sino que **crea un alias**:

De esta forma tenemos dos formas de acceder al mismo ArrayList: mediante la referencia ventas y mediante la referencia ventas1.
Para hacer una copia podemos hacerlo de forma manual elemento a elemento o se puede pasar la referencia del ArrayList original al constructor del nuevo:
ArrayList<Integer> ventas1 = new ArrayList(ventas);

ARRAYLIST COMO PARÁMETRO DE UN MÉTODO

Un ArrayList puede ser usado como parámetro de un método. Además un método puede devolver un ArrayList mediante la sentencia return.

Ejemplo: Método que recibe un ArrayList de String y lo modifica invirtiendo su contenido:

```
import java.util.*;


public class ArrayList4 {


 public static void main(String[] args) {
  ArrayList<String> nombres = new ArrayList();
  nombres.add("Ana");
  nombres.add("Luisa");
  nombres.add("Felipe");
  nombres.add("Pablo");
  System.out.println(nombres);
  nombres = invertir(nombres);
  System.out.println(nombres);
 }

 public static ArrayList<String> invertir(ArrayList<String> nombres) {
  // Crea una lista para el resultado del método
  ArrayList<String> resultado = new ArrayList();
  // Recorre la lista de nombres en orden inverso
  for (int i = nombres.size() - 1; i >= 0; i--) {
   // Añade cada nombre al resultado
   resultado.add(nombres.get(i));
  }
  return resultado;
 }
}
```