ArrayList en Java, con ejemplos

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.

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

```
// Declaración de un ArrayList de "String". Puede ser de cualquier otro Eleme
nto 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 parametr
o
nombreArrayList.contains("Elemento");



// Devuelve la posición de la primera ocurrencia ('Elemento') en el ArrayList
nombreArrayList.indexOf("Elemento");
```

```java
// 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). Los Iteradores sirven para recorrer los ArrayList y poder trabajar con ellos.

Iterator es un patrón de diseño de comportamiento que permite el recorrido secuencial por una estructura de datos compleja sin exponer sus detalles internos. Gracias al patrón **Iterator**, los clientes pueden recorrer elementos de colecciones diferentes de un modo similar, utilizando una única interfaz iteradora.

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;

- el *"remove()"* que sirve para eliminar el elemento del Iterador.

EJEMPLOS

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+" / ");
}
```

Ejecutando esta código obtenemos por pantalla lo siguiente:

```
Elemento 1 / Elemento 2 / Elemento 3 / Elemento 3 / Elemento 4 / Elemento 5 /
Elemento 6 / Elemento 7 / Elemento 8 / Elemento 9 / Elemento 10 /
```

Nota: Leer el arraylist sin utilizar el iterador

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 eliminaci
ones:

numElementos = nombreArrayList.size();

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

Nota: si el elementos esta varias veces, solo elimine uno, el primero que encuentre. En un segundo caso borrar el ultimo que encuentre

Como salida a este código tenemos lo siguiente:

```
El ArrayList tiene 11 elementos


... Eliminamos el primer elemento del ArrayList (Elemento 1)...


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


Imprimimos los elementos del ArrayList tras realizar las eliminaciones:

Elemento 2 / Elemento 4 / Elemento 5 / Elemento 6 / Elemento 7 / Elemento 8 /
Elemento 9 / Elemento 10 /


Numero de elementos del ArrayList tras las eliminaciones = 8
```

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.

Adicionar

Menú con las siguientes opciones

1-Llenar el arrayList

2- Adicionar nuevo elemento en posición n.

3- Listar el arrayList

4- Borrar elemento de un posición n

5- Borrar un elemento especifico

La presentación sean utilizando ventanas (Grafica)