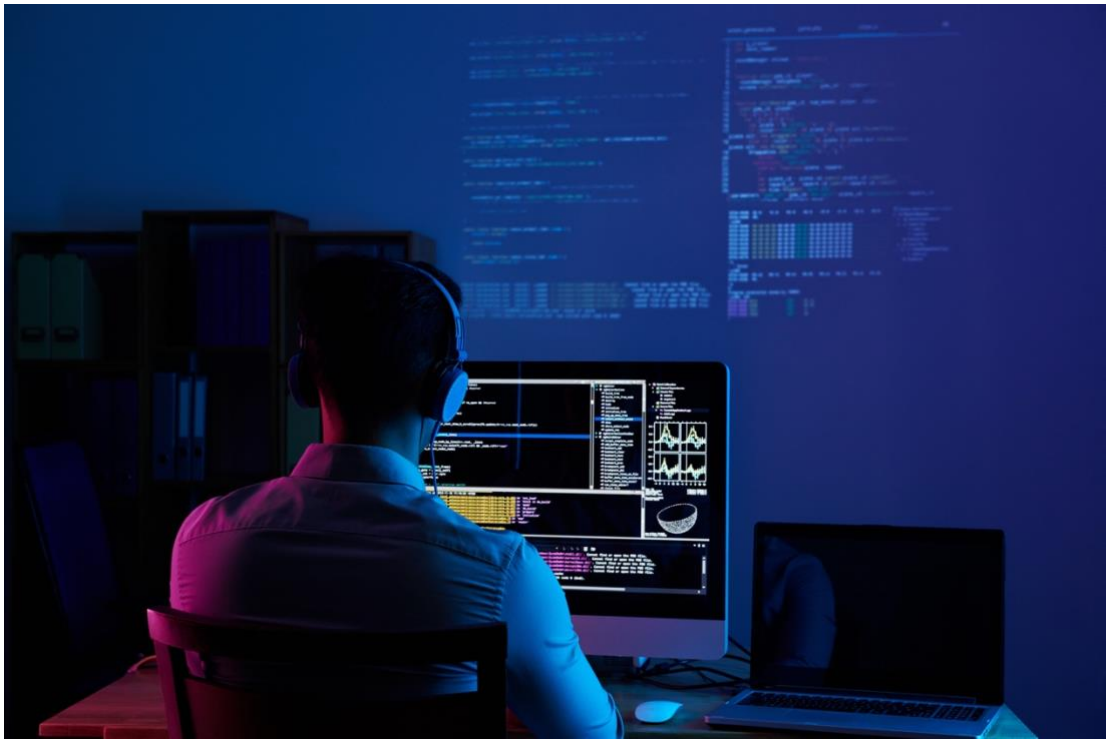
COLECCIONES – CLASE ARRAYLIST

Programación



INDICE

COLECCIONES: LA CLASE ARRAYLIST

Una colección en Java es una estructura de datos que permite almacenar muchos valores del mismo tipo; por tanto, conceptualmente es prácticamente igual que un array.

Según el uso y según si se permiten o no repeticiones, Java dispone de un amplio catálogo de colecciones: ArrayList (lista), ArrayBlockingQueue (cola), HashSet (conjunto), Stack (pila), etc. En este curso estudiaremos la colección ArrayList.

Un ArrayList es una estructura en forma de lista que permite almacenar elementos del mismo tipo (pueden ser incluso objetos); su tamaño va cambiando a medida que se añaden o se eliminan esos elementos.

Nos podemos imaginar un ArrayList como un conjunto de celdas o cajoncitos donde se guardan los valores, exactamente igual que un array convencional. En la práctica será más fácil trabajar con un ArrayList.

En temas anteriores hemos podido comprobar la utilidad del array; es un recurso imprescindible que cualquier programador debe manejar con soltura. No obstante, el array presenta algunos inconvenientes. Uno de ellos es la necesidad de conocer el tamaño exacto en el momento de su creación. Una colección, sin embargo, se crea sin que se tenga que especificar el tamaño; posteriormente se van añadiendo y quitando elementos a medida que se necesitan.

Trabajando con arrays es frecuente cometer errores al utilizar los índices; por ejemplo al intentar guardar un elemento en una posición que no existe (índice fuera de rango).

Aunque las colecciones permiten el uso de índices, no es necesario indicarlos siempre.

Por ejemplo, en una colección del tipo ArrayList, cuando hay que añadir el elemento "Amapola", se puede hacer simplemente flores.add("Amapola"). Al no especificar índice, el elemento "Amapola" se añadiría justo al final de flores independientemente del tamaño y del número de elementos que se hayan introducido ya.

DECLARACIÓN DE UN OBJETO ARRAYLIST

La declaración genérica de un ArrayList se puede hacer con un formato similar al siguiente:

ArrayList nombreDeLista;

Como se puede observar, de esta manera no se indica el tipo de datos que va a contener.

Suele ser recomendable especificar el tipo de datos que va a contener la lista para que así se empleen las operaciones y métodos adecuados para el tipo de datos manejado. Para especificar el tipo de datos que va a contener la lista se debe indicar entre los caracteres '<' y '>' la clase de los objetos que se almacenarán:

ArrayList<nombreClase> nombreDeLista;

En caso de almacenar datos de un tipo básico de Java como char, int, double, etc, se debe especificar el nombre de la clase asociada: Character, Integer, Double, etc.

Ejemplos:

ArrayList<String> listaPaises;

ArrayList<Integer> edades;

CREACIÓN DE UN ARRAYLIST

Para crear un ArrayList se puede seguir el siguiente formato:

nombreDeLista = new ArrayList();

Como suele ser habitual, se puede declarar la lista a la vez que se crea:

ArrayList<nombreClase> nombreDeLista = new ArrayList();

Por ejemplo:

ArrayList<String> listaPaises = new ArrayList();

La clase ArrayList forma parte del paquete java.util, por lo que hay que incluir en la parte inicial del código la importación de ese paquete (import java.util.ArrayList;).

AÑADIR ELEMENTOS AL FINAL DE LA LISTA

El método add de la clase ArrayList posibilita añadir elementos. Los elementos que se van añadiendo, se colocan después del último elemento que hubiera en el ArrayList. En primer elemento que se añada se colocará en la posición 0.

boolean add(Object elementoAInsertar);

Ejemplos:

```
ArrayList<String> listaPaises = new ArrayList();
listaPaises.add("España");   //Ocupa la posición 0
listaPaises.add("Francia");  //Ocupa la posición 1
listaPaises.add("Portugal"); //Ocupa la posición 2
//Se pueden crear ArrayList para guardar datos numéricos de igual manera
ArrayList<Integer> edades = new ArrayList();
edades.add(22);
edades.add(31);
edades.add(18);
```

INSERTAR ELEMENTOS EN UNA DETERMINADA POSICIÓN

Con los ArrayList también es posible insertar un elemento en una determinada posición desplazando el elemento que se encontraba en esa posición, y todos los siguientes, una posición más.

Para ello, se emplea también el método add indicando como primer parámetro el número de la posición donde se desea colocar el nuevo elemento:

void add(int posición, Object elementoAInsertar);

Ejemplo:

```
ArrayList<String> listaPaises = new ArrayList();
listaPaises.add("España");
listaPaises.add("Francia");
listaPaises.add("Portugal");      //El orden hasta ahora es: España, Francia, Portugal
listaPaises.add(1, "Italia");     //El orden ahora es: España, Italia, Francia, Portugal
```

Si se intenta insertar en una posición que no existe, se produce una excepción (IndexOutOfBoundsException)

SUPRIMIR ELEMENTOS DE LA LISTA

Si se quiere que un determinado elemento se elimine de la lista se puede emplear el método remove al que se le puede indicar por parámetro un valor int con la posición a suprimir, o bien, se puede especificar directamente el elemento a eliminar si es encontrado en la lista.

Object remove(int posición)

boolean remove(Object elementoASuprimir)

Se puede ver en el siguiente ejemplo los dos posibles usos:

```
ArrayList<String> listaPaises = new ArrayList();
listaPaises.add("España");
listaPaises.add("Francia");
listaPaises.add("Portugal");
//El orden hasta ahora es: España, Francia, Portugal
listaPaises.add(1, "Italia");
//El orden ahora es: España, Italia, Francia, Portugal
listaPaises.remove(2);
//Eliminada Francia, queda: España, Italia, Portugal
listaPaises.remove("Portugal");
//Eliminada Portugal, queda: España, Italia
```

CONSULTA DE UN DETERMINADO ELEMENTO DE LA LISTA

El método get permite obtener el elemento almacenado en una determinada posición que es indicada con un parámetro de tipo int:

Object get(int posición)

Con el elemento obtenido se podrá realizar cualquiera de las operaciones posibles según el tipo de dato del elemento (asignar el elemento a una variable, incluirlo en una expresión, mostrarlo por pantalla, etc). Por ejemplo:

```
System.out.println(listaPaises.get(3));
//Siguiendo el ejemplo anterior, mostraría: Portugal
```

MODIFICAR UN ELEMENTO CONTENIDO EN LA LISTA

Es posible modificar un elemento que previamente ha sido almacenando en la lista utilizando el método set. Como primer parámetro se indica, con un valor int, la posición que ocupa el elemento a modificar, y en el segundo parámetro se especifica el nuevo elemento que ocupará dicha posición sustituyendo al elemento anterior.

Object set(int posición, Object nuevoElemento)

Por ejemplo, si en el ejemplo de la lista de países se desea modificar el que ocupe la posición 1 (segundo en la lista) por "Alemania":

```
listaPaises.set(1, "Alemania");
```

BUSCAR UN ELEMENTO

La clase ArrayList facilita mucho las búsquedas de elementos gracias al método indexOf que retorna, con un valor int, la posición que ocupa el elemento que se indique por parámetro.

int indexOf(Object elementoBuscado)

Si el elemento se encontrara en más de una posición, este método retorna la posición del primero que se encuentre. El método lastIndexOf obtiene la posición del último encontrado.

Ejemplo que comprueba si Francia está en la lista, y muestra su posición.

```
String paisBuscado = "Francia";
int pos = listaPaises.indexOf(paisBuscado);
if(pos!=-1)
   System.out.println(paisBuscado + " se ha encontrado en la posición: "+pos);
else
   System.out.println(paisBuscado + " no se ha encontrado");
```

En caso de que no se encuentre en la lista el elemento buscado, se obtiene el valor -1.

RECORRER EL CONTENIDO DE LA LISTA

Es posible obtener cada uno de los elementos de la lista utilizando un bucle con tantas iteraciones como elementos contenga, de forma similar a la empleada con los arrays convencionales. Para obtener el número de elementos de forma automática se puede emplear el método size() que devuelve un valor int con el número de elementos que contiene la lista.

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

También se puede emplear el otro formato del bucle for en el que se va asignando cada elemento de la lista a una variable declarada del mismo tipo que los elementos del ArrayList:

```
for(String pais:listaPaises)
    System.out.println(pais);
```

PRINCIPALES MÉTODOS DE ARRAYLIST

Las operaciones más comunes que se pueden realizar con un objeto de la clase ArrayList son las siguientes:

MÉTODO	DESCRIPCIÓN
add(X)	Añade el objeto X al final. Devuelve true.
add(posición, X)	Inserta el objeto X en la posición indicada.
clear()	Elimina todos los elementos.
contains(X)	Comprueba si la colección contiene al objeto X. Devuelve true o false.
get(posicion)	Devuelve el elemento que está en la posición indicada.
indexOf(X)	Devuelve la posición del objeto X. Si no existe devuelve -1
isEmpty()	Devuelve true si la lista está vacía y false en caso de tener algún elemento.
remove(posicion)	Elimina el elemento que se encuentra en la posición indicada. Devuelve el elemento eliminado.
remove(X)	Elimina la primera ocurrencia del objeto X. Devuelve true si el elemento está en la lista.
set(posición, X)	Sustituye el elemento que se encuentra en la posición indicada por el objeto X. Devuelve el elemento sustituido.
size()	Devuelve el número de elementos (int)
toArray()	Devuelve un array con todos y cada uno de los elementos que contiene la lista.

ARRAYLIST DE OBJETOS

Una colección ArrayList puede contener objetos que son instancias de clases definidas por el programador. Esto es muy útil sobre todo en aplicaciones de gestión para guardar datos de alumnos, productos, libros, etc.

En el siguiente ejemplo, definimos una lista de gatos. En cada celda de la lista se almacenará un objeto de la clase Gato.

```
public class Gato {
	private String nombre;
	private String color;
	private String raza;

	public Gato(String nombre, String color, String raza) {
		this.nombre = nombre;
		this.color = color;
		this.raza = raza;
	}

	public String getNombre() {
		return nombre;
	}

	public String getRaza() {
		return raza;
	}

	public String toString() {
		return "Nombre: " + this.nombre + "\nColor: " + this.color + "\nRaza: " +
		this.raza;
	}
}
```

```
import java.util.ArrayList;

public class Main {
	public static void main(String[] args) {
		ArrayList<Gato> g = new ArrayList<Gato>();
		g.add(new Gato("Garfield", "naranja", "mestizo"));
		g.add(new Gato("Pepe", "gris", "angora"));
		g.add(new Gato("Mauri", "blanco", "manx"));
		g.add(new Gato("Ulises", "marrón", "persa"));

		System.out.println("\nDatos de los gatos:\n");
		for (Gato gatoAux: g) {
			System.out.println(gatoAux+"\n");
		}
	}
}
```