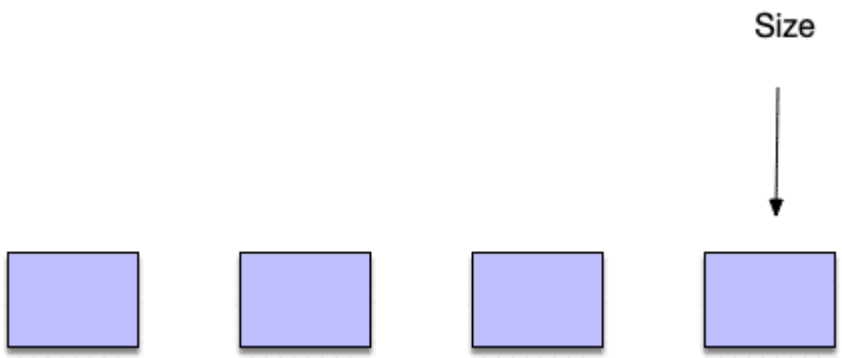
Java ArrayList for y las opciones para recorrerlos siempre vuelven locos a los principiantes y a algunos que no lo son tanto ya que cuando tenemos muchas opciones pues siempre se generan dudas a la hora de elegir. Vamos a ver cómo recorrer una lista de elementos de forma sencilla y abordar las diferentes posibilidades.

Java ArrayList for y size()

La forma más sencilla de recorrer una lista es a través de un bucle for y accediendo a la propiedad size.



Veámoslo en código:

```
package com.arquitecturajava;

import java.util.ArrayList;

public class JavaFor {

	public static void main(String[] args) {

		ArrayList<String> lista = new ArrayList<String>();
		lista.add("hola");
		lista.add("que");
		lista.add("tal");
```

```
                lista.add("estas");
                lista.add("hoy");
                for (int i=0;i<lista.size();i++) {
                        System.out.println(lista.get(i));
                }
        }
}
```

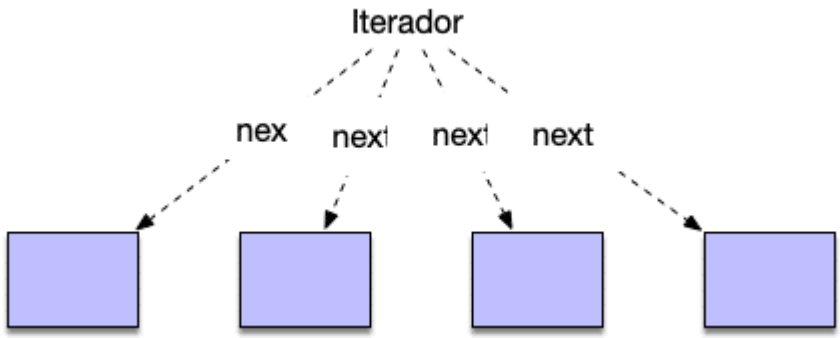
El resultado se muestra en la consola:

```
hola
que
tal
estas
hoy
```

Cuando uno empieza esta forma parece la más clara y no parece tener mucha problemática . Sin embargo algunas veces sucede que los desarrolladores al recorrer la lista no asignan correctamente el lista.size() o ponen otro valor sobre todo cuando se es muy novato. Por lo tanto puede generarse un NullPointerException. No solo eso sino que es una forma de recorrer elementos que solo nos vale para los ArrayList.

Java for e Iteradores

Una alternativa a esta situación es usar un Iterador .Un Iterador es un interface que dispone de los métodos hasNext() y next() y nos permite recorrer una colección de elementos.



Da lo mismo que sea un ArrayList que otra estructura. Por lo tanto aporta homogeneidad en las APIs.

```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.Iterator;

public class JavaFor2 {

	public static void main(String[] args) {

		ArrayList<String> lista = new ArrayList<String>();
		lista.add("hola");
		lista.add("que");
		lista.add("tal");
		lista.add("estas");
		lista.add("hoy");
		Iterator<String> it= lista.iterator();
		while(it.hasNext()) {
			System.out.println(it.next());
		}
	}
}
```

Java 5 ArrayList for

A partir de Java 5 aparece el concepto de bucle forEach y permite simplificar la forma en la que trabajamos con los Iteradores.

```
package com.arquitecturajava;
```

```
import java.util.ArrayList;

public class JavaFor3 {

	public static void main(String[] args) {

		ArrayList<String> lista = new ArrayList<String>();
		lista.add("hola");
		lista.add("que");
		lista.add("tal");
		lista.add("estas");
		lista.add("hoy");
		for (String cadena: lista) {
			System.out.println(cadena);
		}
	}
}
```

Como se puede observar la forma de recorrer una lista se simplifica sobremanera comparado con el Iterador que aunque genera homogeneidad siempre era complicado de usar. Esta es una de las opciones más recomendadas hasta la llegada de Java 8.

Java 8 y Streams.

En Java 8 tenemos un salto importante a niveles de programación ya que entran todas las capacidades funcionales y con ello se pueden simplificar el manejo de listas a través de streams .

```
package com.arquitecturajava;

import java.util.ArrayList;
```

```
public class JavaFor3 {

	public static void main(String[] args) {

		ArrayList<String> lista = new ArrayList<String>();
		lista.add("hola");
		lista.add("que");
		lista.add("tal");
		lista.add("estas");
		lista.add("hoy");
		lista.forEach(System.out::println);
	}
}
```

La simplificación es clara aun así nos quedan situaciones clásicas en las que un bucle de forEach de Java 5 nos aporta más ya que por ejemplo deseamos cambiar el contenido de los elementos del Array y luego imprimirlo.

```
package com.arquitecturajava;

import java.util.ArrayList;

public class JavaFor6 {

	public static void main(String[] args) {

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

		lista.add("hola");
		lista.add("que");
		lista.add("tal");
		lista.add("estas");
```

```
                lista.add("hoy");

                for (int i = 0; i < lista.size(); i++) {

                        lista.set(i, lista.get(i).toUpperCase());
                }

                for (String cadena : lista) {

                        System.out.println(cadena);

                }

        }
}
```

Este bloque nos recorrera el array y volverá a recorrerlo de una forma clásica para mostrar la información en pantalla actualizada.

¿Existe una forma de realizar esta operación de una forma más compacta en Java 8 ? . La realidad es que sí . Podemos usar el método replaceAll del ArrayList que recibe un interface funcional y permite actualizar todos los elementos de golpe a continuación usaremos el método forEach.

```
package com.arquitecturajava;

import java.util.ArrayList;
```

```
public class JavaFor5 {

	public static void main(String[] args) {

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

		lista.add("hola");
		lista.add("que");
		lista.add("tal");
		lista.add("estas");
		lista.add("hoy");

		lista.replaceAll(String::toUpperCase);
		lista.forEach(System.out::println);

	}
}
```

De esta forma el resultado será idéntico pero con mucho menos código.

```
HOLA
QUE
TAL
ESTAS
HOY
```

Otros Artículos relacionados

- Java ArrayList Count
- Java Iterable
- Java collections list vs set
- JDK ArrayList

Cecilio Álvarez Caules
Cecilio Álvarez Caules Oracle Java Certified Architect