



Manejo de Colecciones en Java

Java proporciona la interfaz **Collection** y la interfaz **Map** dentro del paquete `java.util` para manejar estructuras de datos dinámicas.

Las colecciones principales son:

1. **Listas (List)** → Elementos ordenados, permite duplicados.
2. **Conjuntos (Set)** → No permite duplicados, orden variable.
3. **Mapas (Map)** → Pares clave-valor, claves únicas.

Listas (List)

Una List mantiene el orden de inserción y permite elementos duplicados.

◊ Implementaciones principales:

- **ArrayList** → Basado en array dinámico, rápido en acceso por índice.
- **LinkedList** → Basado en lista doblemente enlazada, rápido en inserciones/eliminaciones.

Ejemplo de ArrayList

```
import java.util.ArrayList;
import java.util.List;

List<String> lista = new ArrayList<>();
lista.add("A");
lista.add("B");
lista.add("A"); // Se permiten duplicados
System.out.println(lista); // [A, B, A]
```

Ejemplo de LinkedList

```
import java.util.LinkedList;

LinkedList<Integer> numeros = new LinkedList<>();
numeros.add(10);
numeros.addFirst(5);
numeros.addLast(20);
System.out.println(numeros); // [5, 10, 20]
```



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.^a ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



Conjuntos (Set)

Un Set **no permite elementos duplicados** y su orden depende de la implementación.

◊ **Implementaciones principales:**

- HashSet → No garantiza orden.
- LinkedHashSet → Mantiene el orden de inserción.
- TreeSet → Ordena automáticamente los elementos.

Ejemplo de HashSet

```
import java.util.HashSet;
import java.util.Set;

Set<String> conjunto = new HashSet<>();
conjunto.add("A");
conjunto.add("B");
conjunto.add("A"); // Ignorado porque ya existe
System.out.println(conjunto); // [A, B] (orden no garantizado)
```

Ejemplo de TreeSet

```
import java.util.TreeSet;

TreeSet<Integer> numeros = new TreeSet<>();
numeros.add(3);
numeros.add(1);
numeros.add(2);
System.out.println(numeros); // [1, 2, 3] (ordenado automáticamente)
```



Mapas (Map)

Un Map almacena **pares clave-valor**, donde las claves son únicas.

◇ Implementaciones principales:

- **HashMap** → No garantiza orden.
- **LinkedHashMap** → Mantiene el orden de inserción.
- **TreeMap** → Ordena automáticamente por clave.

Ejemplo de HashMap

```
import java.util.HashMap;
import java.util.Map;

Map<String, Integer> edades = new HashMap<>();
edades.put("Juan", 25);
edades.put("Ana", 30);
edades.put("Juan", 28); // Sobrescribe el valor de "Juan"
System.out.println(edades); // {Ana=30, Juan=28} (orden no garantizado)
```

Ejemplo de TreeMap

```
import java.util.TreeMap;

TreeMap<Integer, String> mapa = new TreeMap<>();
mapa.put(3, "Tres");
mapa.put(1, "Uno");
mapa.put(2, "Dos");
System.out.println(mapa); // {1=Uno, 2=Dos, 3=Tres} (ordenado por clave)
```





Comparación de Colecciones

Tipo	Implementación	Duplicados	Orden
List	ArrayList	✓ Sí	✓ Sí (inserción)
	LinkedList	✓ Sí	✓ Sí (inserción)
Set	HashSet	✗ No	✗ No garantizado
	LinkedHashSet	✗ No	✓ Sí (inserción)
	TreeSet	✗ No	✓ Sí (orden natural)
Map	HashMap	✗ (claves)	✗ No garantizado
	LinkedHashMap	✗ (claves)	✓ Sí (inserción)
	TreeMap	✗ (claves)	✓ Sí (ordenado por clave)

Las **listas** son ideales para datos ordenados *con duplicados*.

Los **conjuntos** son perfectos cuando no queremos duplicados.

Los **mapas** son útiles para relaciones clave-valor.



ArrayList en Java

ArrayList es una estructura de datos de la librería java.util que funciona como un **array dinámico**, permitiendo agregar, eliminar y modificar elementos sin necesidad de definir un tamaño fijo.

Importar y Crear un ArrayList

Antes de usarlo, debemos **importarlo**:

```
import java.util.ArrayList;
```

Para **crearlo**:

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

También podemos **inicializarlo** con valores:

```
ArrayList<Integer> numeros = new ArrayList<>(List.of(1, 2, 3, 4, 5));
```

Agregar Elementos

- Usamos `add()`:

```
lista.add("Manzana");
lista.add("Banana");
lista.add(1, "Naranja"); // Inserta en la posición 1
```

Acceder y Modificar Elementos

- Obtener un elemento por índice

```
String fruta = lista.get(0);
```

- Modificar un elemento:

```
lista.set(1, "Uva"); // Reemplaza el valor en la posición 1
```



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.ª ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG



Eliminar Elementos

- **Por índice:**

```
lista.remove(1); // Elimina el elemento en la posición 1
```

- **Por valor:**

```
lista.remove("Manzana"); // Elimina la primera ocurrencia de "Manzana"
```

- **Eliminar todos los elementos:**

```
lista.clear();
```

Recorrer un ArrayList

- **Usando un for tradicional**

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

- **Usando un for-each**

```
for (String fruta : lista) {  
    System.out.println(fruta);  
}
```

Otras Operaciones Comunes

Obtener el tamaño

```
int tamano = lista.size();
```

Verificar si está vacía

```
boolean estaVacia = lista.isEmpty();
```

Ordenar la lista

```
lista.sort(String::compareTo);
```

Convertir ArrayList a array

```
String[] array = lista.toArray(new String[0]);
```

Buscar un elemento

```
boolean existe = lista.contains("Banana");
```

Obtener el índice de un elemento

```
int indice = lista.indexOf("Banana");
```

