**ArrayList**

Un ArrayList es una lista dinámica en Java que permite almacenar y manipular elementos de forma flexible. Se comporta como un array, pero su tamaño puede crecer o reducirse automáticamente.

**Características principales:**

✔ Permite elementos duplicados.  
✔ Mantiene el orden de inserción.  
✔ Acceso rápido a los elementos mediante índice.  
✔ Crece dinámicamente según se añadan elementos.

**Métodos más importantes de ArrayList**

|  |  |
| --- | --- |
| **Método** | **Descripción** |
| add(elemento) | Agrega un elemento al final de la lista. |
| add(index, elemento) | Inserta un elemento en una posición específica. |
| get(index) | Devuelve el elemento en la posición indicada. |
| set(index, elemento) | Modifica el elemento en la posición dada. |
| remove(index) | Elimina el elemento en la posición dada. |
| remove(elemento) | Elimina la primera ocurrencia del elemento en la lista. |
| clear() | Elimina todos los elementos de la lista. |
| size() | Devuelve la cantidad de elementos en la lista. |
| isEmpty() | Retorna true si la lista está vacía, false si contiene elementos. |
| contains(elemento) | Devuelve true si el elemento está en la lista. |
| indexOf(elemento) | Devuelve la primera posición del elemento o -1 si no está. |
| lastIndexOf(elemento) | Devuelve la última posición del elemento o -1 si no está. |
| subList(inicio, fin) | Devuelve una sublista desde inicio hasta fin (exclusivo). |
| toArray() | Convierte el ArrayList en un array. |

**Ejemplo de uso:**

import java.util.ArrayList;

public class EjemploArrayList {

public static void main(String[] args) {

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

nombres.add("Ana");

nombres.add("Juan");

nombres.add("Carlos");

nombres.add(1, "Elena"); // Inserta "Elena" en la posición 1

System.out.println("Lista de nombres: " + nombres);

System.out.println("Primer elemento: " + nombres.get(0));

System.out.println("Tamaño de la lista: " + nombres.size());

nombres.remove("Juan"); // Elimina "Juan" de la lista

System.out.println("Lista después de eliminar: " + nombres);

}

}

**HashMap**

Un HashMap almacena elementos en pares **clave-valor**, lo que permite búsquedas rápidas mediante la clave. Es ideal para implementar diccionarios y tablas de datos.

**Características principales:**

✔ La clave es única, pero los valores pueden repetirse.  
✔ No garantiza el orden de los elementos.  
✔ Acceso eficiente a los valores mediante sus claves.

**Métodos más importantes de HashMap**

|  |  |
| --- | --- |
| **Método** | **Descripción** |
| put(clave, valor) | Inserta o actualiza un par clave-valor. |
| get(clave) | Obtiene el valor asociado a una clave. |
| remove(clave) | Elimina una entrada del mapa según su clave. |
| containsKey(clave) | Devuelve true si la clave existe en el mapa. |
| containsValue(valor) | Devuelve true si el valor existe en el mapa. |
| size() | Devuelve el número de elementos en el HashMap. |
| isEmpty() | Devuelve true si el HashMap está vacío. |
| clear() | Elimina todas las entradas del HashMap. |
| keySet() | Devuelve un Set con todas las claves del HashMap. |
| values() | Devuelve una colección con todos los valores del HashMap. |
| entrySet() | Devuelve un conjunto de pares (clave, valor). |

**Ejemplo de uso:**

import java.util.HashMap;

public class EjemploHashMap {

public static void main(String[] args) {

HashMap<Integer, String> personas = new HashMap<>();

personas.put(1, "Luis");

personas.put(2, "Marta");

personas.put(3, "Elena");

System.out.println("Persona con clave 2: " + personas.get(2));

System.out.println("¿Existe la clave 4? " + personas.containsKey(4));

personas.remove(3);

System.out.println("Mapa después de eliminar clave 3: " + personas);

}

}

**Diferencias clave entre ArrayList y HashMap**

|  |  |  |
| --- | --- | --- |
| **Característica** | **ArrayList** | **HashMap** |
| Almacenamiento | Solo valores. | Claves y valores. |
| Acceso a elementos | Por índice (get(index)). | Por clave (get(clave)). |
| Permite duplicados | Sí. | No en claves, pero sí en valores. |
| Orden | Mantiene el orden de inserción. | No garantiza orden. |
| Aplicaciones | Listas ordenadas, pilas, colas. | Diccionarios, bases de datos en memoria. |

**Conclusión**

* Usa **ArrayList** cuando necesites una lista ordenada con acceso por índice.
* Usa **HashMap** cuando necesites almacenar datos en pares clave-valor y acceder rápidamente por clave.