COLECCIÓN ARRAYLIST VS HASHMAP

DIFERENCIAS

**1.** La principal diferencia entre estas dos colecciones, es que, ArrayList es una estructura de datos basada en índices la cual está respaldada por una matriz dinámica, mientras que HashMap es una estructura de datos de tablas(mapa) de hash, la cual permite recuperar valores almacenados.

**2.** **Interface Implementada:** ArrayList implementa la interfaz de lista, mientras que HashMap es una implementación de la interfaz de mapa.

**clase ArrayList:**

clase pública ArrayList

extiende AbstractList

implementa List, RandomAccess, Cloneable, Serializable

**clase HashMap:**

clase pública HashMap

extiende AbstractMap

implementa Map, Cloneable, Serializable

**3. Mantenimiento del orden de inserción**: ArrayList mantiene el orden de inserción mientras que HashMap no mantiene el orden de inserción. Lo que significa que ArrayList devuelve los elementos de la lista en el mismo orden, mientras que HashMap no mantiene ningún orden, por lo que devuelve pares clave-valor de cualquier tipo de orden.

Ejemplo:

import java.util.\*;

class GFG {

public static void main(String args[])

{

// Creating ArrayList

ArrayList<String> list

= new ArrayList<String>();

// Adding object in ArrayList

list.add("A");

list.add("B");

list.add("C");

list.add("D");

// Invoking ArrayList object

System.out.println("ArrayList: " + list);

// Creating HashMap

HashMap<Integer, String> hm

= new HashMap<Integer, String>();

// Adding object in HashMap

hm.put(1, "A");

hm.put(2, "B");

hm.put(3, "C");

hm.put(4, "D");

// Invoking HashMap object

// It might or might not display elements

// in the insertion order

System.out.print("HasMap: " + hm);

}

}

Salida:

ArrayList: [A, B, C, D]

HasMap: {1 = A, 2 = B, 3 = C, 4 = D}

**4.** **Consumo de memoria:** ArrayList almacena los elementos solo como valor y mantiene internamente la indexación de cada elemento. Mientras que HashMap almacena elementos con pares clave y valor, es decir, dos objetos. Entonces HashMap toma más memoria comparativamente.

**Lista de arreglo:**

// El valor de la cadena se almacena en ArrayList

list.add ("A");

**HashMap:**

// Dos valores de cadena almacenados

// como el par de valores clave en HashMap

hm.put (1, "A");

**5. Duplicados:** ArrayList permite elementos duplicados, mientras que HashMap no permite claves duplicadas pero sí valores duplicados.

Ejemplo:

import java.util.\*;

class GFG {

public static void main(String args[])

{

// Creating ArrayList

ArrayList<String> list

= new ArrayList<String>();

// Adding object in ArrayList

list.add("A");

list.add("B");

// Add duplicates

list.add("A");

list.add("A");

// Invoking ArrayList object

System.out.println("ArrayList: " + list);

// Creating HashMap

HashMap<Integer, String> hm

= new HashMap<Integer, String>();

// Adding object in HashMap

hm.put(1, "A");

hm.put(2, "B");

// Add duplicates key

// Change value if index exist

hm.put(3, "A");

hm.put(3, "A");

// Add duplicates values

// allow duplicates value

hm.put(4, "A");

hm.put(5, "A");

// Invoking HashMap object

System.out.print("HasMap: " + hm);

}

}

Salida:

ArrayList: [A, B, A, A]

HasMap: {1 = A, 2 = B, 3 = A, 4 = A, 5 = A}

**6. Facilidad al recuperar un elemento:** en ArrayList, un elemento se puede recuperar fácilmente especificando el índice del mismo. Pero en HashMap, los elementos se obtienen mediante su clave correspondiente. Significa que la clave debe recordarse siempre.

El método get (index) de ArrayList siempre proporciona una complejidad de tiempo O (1) Mientras que get (key) de HashMap puede ser O (1) en el mejor caso y O (n) en el peor de los casos.

Ejemplo:

import java.util.\*;

class GFG {

public static void main(String args[])

{

// Creating ArrayList

ArrayList<String> list

= new ArrayList<String>();

// Adding object in ArrayList

list.add("A");

list.add("B");

list.add("C");

list.add("D");

// Invoking ArrayList object

System.out.println("First Element of ArrayList: "

+ list.get(0));

System.out.println("Third Element of ArrayList: "

+ list.get(2));

// Creating HashMap

HashMap<Integer, String> hm

= new HashMap<Integer, String>();

// Adding object in HashMap

hm.put(1, "A");

hm.put(2, "B");

hm.put(3, "C");

hm.put(4, "D");

// Invoking HashMap object

System.out.println("HashMap value at Key 1: "

+ hm.get(1));

System.out.println("HashMap value at Key 3: "

+ hm.get(3));

}

}

Salida:

Primer elemento de ArrayList: A

Tercer elemento de ArrayList: C

Valor HashMap en la Clave 1: A

Valor de HashMap en la clave 3: C

7. **Almacenamiento de nulos:** en ArrayList, se puede almacenar cualquier cantidad de elementos nulos. Mientras esté en HashMap, solo se permite una clave nula, pero los valores pueden ser de cualquier número.

Ejemplo:

import java.util.\*;

class GFG {

public static void main(String args[])

{

// Creating ArrayList

ArrayList<String> list

= new ArrayList<String>();

// Adding object in ArrayList

list.add("A");

// add first null value

list.add(null);

list.add("C");

// add two null value again

list.add(null);

list.add(null);

// Invoking ArrayList object

System.out.println("ArrayList: "

+ list);

// Creating HashMap

HashMap<Integer, String> hm

= new HashMap<Integer, String>();

// Adding object in HashMap

hm.put(1, "A");

hm.put(2, "B");

// add null key

hm.put(null, "C");

// add again null key

// which replace value of first

// insert null key value

hm.put(null, null);

// add second null value

hm.put(3, null);

// Printing the HashMap

System.out.println("HasMap: " + hm);

}

}

Salida:

ArrayList: [A, nulo, C, nulo, nulo]

HasMap: {nulo = nulo, 1 = A, 2 = B, 3 = nulo}

SIMILITUDES

* ArrayList y HashMap, **ambos no están sincronizados**. Entonces, para usarlos en el entorno de subprocesos múltiples, primero debe sincronizarse.
* **Tanto** ArrayList como HashMap **permiten nulo** . ArrayList permite valores nulos y HashMap permite claves y valores nulos
* **Tanto** ArrayList como HashMap **permiten duplicados** , ArrayList permite elementos duplicados y HashMap permite valores duplicados.
* **Tanto** ArrayList como HashMap **se pueden recorrer a través de Iterator** en Java.
* **Tanto Somewhere use array** , ArrayList está respaldado por array y Array también implementa internamente HashMap
* **Ambos son el método use get ()** , el **método** ArrayList.get () funciona basado en el índice y el método HashMap.get () toma un parámetro key\_element del tipo de objeto y se refiere a la clave cuyo valor asociado se supone que debe buscarse, por lo que ambos proporcionan constante- tiempo de ejecución.