Material descargable

egg

Java Collections Framework

¿Qué es un Map?

Un Map es una interfaz en Java que forma parte del Java Collections Framework y se utiliza para **almacenar pares clave-valor**, donde cada clave está asociada a un valor único.

Es una estructura de datos que permite acceder eficientemente a un valor mediante su clave correspondiente. La interfaz Map no permite claves duplicadas y cada clave se asigna a exactamente un valor.

¿Qué es un HashMap?

HashMap es una implementación de la interfaz Map en el Java Collections Framework. **Esta estructura de datos permite almacenar pares clave-valor, donde cada clave está asociada a un valor único**. Tanto la clave como el valor son objetos, y la relación entre ellos es tal que cada clave se asigna exactamente a un valor. En términos simples, un HashMap proporciona un acceso rápido a los valores a través de sus claves. La clave se utiliza para calcular un índice mediante una función hash, lo que permite un acceso eficiente y constante a los valores asociados.

Comparación con Otras Implementaciones de Map en Java

- **TreeMap**: A diferencia de HashMap, TreeMap mantiene las claves ordenadas. Esto implica que las claves se almacenan en orden ascendente, lo que facilita la iteración en orden. Las operaciones de búsqueda, inserción y eliminación tienen un rendimiento de O(log n), lo que hace que TreeMap sea más adecuado para conjuntos donde el orden es relevante.
- **LinkedHashMap**: Similar a HashMap, pero mantiene el orden de inserción. Combina la eficiencia de acceso de HashMap con la previsibilidad del orden de LinkedHashMap. Es útil cuando se necesita iterar sobre los elementos en el orden en que se insertaron.

Uso Común de los HashMap

- **Asociación de Datos:** HashMap resulta ideal cuando se necesita una asociación rápida de claves a valores, y el orden de las claves no es crítico. Se utiliza ampliamente en casos de uso como el almacenamiento de configuraciones, cachés y en general para la implementación eficiente de estructuras de datos asociativas.
- **Eficiencia en Búsquedas:** Gracias a la función hash, HashMap ofrece una eficiencia excepcional en términos de búsqueda y recuperación de valores a través de claves. Las operaciones básicas (put, get, remove) tienen un tiempo de ejecución promedio constante (O(1)).
- **Manejo de Asociaciones Uno a Uno:** Es útil cuando se necesita mantener asociaciones únicas entre claves y valores. Cada clave tiene exactamente un valor asociado.

Operaciones básicas con HashMap

En este anexo, puedes ver algunos de los distintos métodos explicados brevemente:

Método	Descripción
put(K key, V value)	Agrega un par clave-valor al HashMap. Si la clave ya existe, el valor asociado se actualiza. Si la clave es nueva, se crea un nuevo par clave-valor. Devuelve el valor anterior asociado con la clave o null si la clave no estaba presente antes.
get(Object key)	Devuelve el valor asociado con la clave especificada. Si la clave no está presente, devuelve null.
remove(Object key)	Elimina el par clave-valor asociado con la clave especificada. Devuelve el valor asociado con la clave, o null si la clave no está presente.
containsKey(Object key)	Devuelve true si la clave está presente en el HashMap. Devuelve false si la clave no está presente.
size()	Devuelve el número de pares clave-valor en el HashMap.

isEmpty()	Devuelve true si el HashMap no contiene ningún par clave-valor. Devuelve false si el HashMap contiene uno o más pares clave-valor.
clear()	Elimina todos los pares clave-valor del HashMap.
values()	Devuelve una colección de todos los valores presentes en el HashMap.
replace(K key, V value)	Reemplaza el valor asociado con la clave especificada por el nuevo valor especificado. Devuelve el valor anterior asociado con la clave o null si la clave no estaba presente antes y, por lo tanto, no se pudo reemplazar.

Ejemplo de Uso:

```java
public class HashMapExample {
    Run | Debug
    public static void main(String[] args) {
    // Crear un HashMap
    HashMap<String, Integer> hashMap = new HashMap<>();

    // Inserción de elementos
    hashMap.put(key:"Uno", value:1);
    hashMap.put(key:"Dos", value:2);
    hashMap.put(key:"Tres", value:3);

    // Obtención de elementos
    int valorDos = hashMap.get(key:"Dos");
    System.out.println("Valor asociado a 'Dos': " + valorDos);

    // Eliminación de un elemento
    int valorEliminado = hashMap.remove(key:"Dos");
    System.out.println("Valor eliminado asociado a 'Dos': " + valorEliminado);

    // Verificación de existencia de una clave
    boolean contieneTres = hashMap.containsKey(key:"Tres");
    System.out.println("¿Contiene la clave 'Tres'? " + contieneTres);
    }
}
```

💡Siempre es recomendable utilizar la **documentación oficial** para obtener más detalles sobre el material proporcionado, su uso y su implementación. Puedes acceder a la información sobre Map desde **aquí**.