

HASHMAP JAVA

Para la presente guía, usaremos como ejemplo el siguiente código:

```
import java.util.HashMap;
import java.util.Map;
public class LlaveValor {
    public static void main(String[] args) {
        // 1. Crear un HashMap
        //solo recibe Objetos no primitivos
        HashMap<String, Integer> edades = new
HashMap<>();
        System.out.println("1. HashMap creado: " +
edades);

        System.out.println("-----
-----");
        // 2. Agregar elementos (put)
        edades.put("Ana", 25);
        edades.put("Luis", 30);
        edades.put("Carlos", 22);
        System.out.println("2. Elementos agregados:
" + edades);

        System.out.println("-----
-----");
        // 3. Obtener un elemento (get)
        int edadDeAna = edades.get("Ana");
        System.out.println("3. La edad de Ana es: "
+ edadDeAna);

        System.out.println("-----
-----");
```

```

-----");
    // 4. Actualizar un elemento
    // Si la clave ya existe, put() la
sobrescribe
    edades.put("Carlos", 28);
    System.out.println("4. Edad de Carlos
actualizada: " + edades);

System.out.println("-----
-----");
    // 5. Eliminar un elemento (remove)
    edades.remove("Luis");
    System.out.println("5. Elemento 'Luis'
eliminado: " + edades);

System.out.println("-----
-----");
    // 6. Recorrer el HashMap
    System.out.println("6. Recorriendo el
HashMap:");
    for (Map.Entry<String, Integer> entrada :
edades.entrySet()) {
        System.out.println("Nombre: " +
entrada.getKey() + ", Edad: " +
entrada.getValue());
    }
}
}

```

¿Qué es un HashMap?

Un HashMap es una de las implementaciones más utilizadas de la interfaz Map en Java. Pertenece al framework de Colecciones de Java y se utiliza para almacenar datos en pares de "clave-valor" (key-value).

Su principal característica es que utiliza una técnica llamada *hashing* para almacenar y recuperar elementos de manera muy eficiente. En lugar de buscar un elemento secuencialmente, el HashMap calcula un código único (un "hash") a partir de la clave y lo utiliza para determinar dónde se debe guardar el valor asociado.

Características principales:

- **Pares Clave-Valor:** Cada elemento en un HashMap consta de una clave única y un valor asociado.
- **Claves Únicas:** No permite claves duplicadas. Si se intenta insertar una clave que ya existe, el valor asociado a esa clave se sobrescribirá con el nuevo.
- **Permite un Nulo como Clave:** Se puede utilizar null como clave una única vez.
- **Permite Múltiples Valores Nulos:** Se pueden almacenar valores null sin restricciones.
- **Sin Orden Garantizado:** El HashMap no garantiza ningún orden específico para los elementos. El orden en que se insertan no será necesariamente el orden en que se recuperan.

Guía Paso a Paso Basada en el Código

A continuación, se desglosa el funcionamiento del HashMap siguiendo los pasos del código de ejemplo.

1. Crear un HashMap

Para empezar a utilizar un HashMap, primero se debe crear una instancia del mismo.

```
// solo recibe Objetos no primitivos
HashMap<String, Integer> edades = new HashMap<>();
System.out.println("1. HashMap creado: " + edades);
System.out.println("-----");
```

Explicación:

- **HashMap<String, Integer>:** Aquí se declara una variable llamada edades que será de tipo HashMap.
- **Los parámetros entre los paréntesis angulares < > definen los tipos de datos que almacenará:**
 - **String:** Será el tipo de dato para las **claves**.
 - **Integer:** Será el tipo de dato para los **valores**.

- **Importante:** Un HashMap solo puede almacenar objetos, no tipos de datos primitivos. Por eso se utiliza Integer (la clase envolvente) en lugar de int (el tipo primitivo).
 - `new HashMap<>()`: Esta es la instrucción que crea una nueva instancia vacía del HashMap en la memoria.
 - **Salida:** 1. HashMap creado: {}. El resultado muestra un mapa vacío, representado por las llaves {}.
-

2. Agregar Elementos con put()

Una vez creado el mapa, se pueden añadir elementos utilizando el método `put()`. Este método asocia una clave a un valor.

Java

```
edades.put("Ana", 25);
edades.put("Luis", 30);
edades.put("Carlos", 22);
System.out.println("2. Elementos agregados: " + edades);
System.out.println("-----");
```

Explicación:

- `edades.put("Ana", 25)` :: Se añade un nuevo par al mapa. La clave es el String "Ana" y el valor asociado es el Integer 25.
 - Internamente, Java calcula un *hash code* para la clave "Ana" y utiliza ese código para decidir dónde almacenar el valor 25.
 - Lo mismo ocurre con "Luis" y "Carlos".
 - **Salida:** 2. Elementos agregados: {Ana=25, Luis=30, Carlos=22}. Se observa que el mapa ahora contiene los tres pares clave-valor. El orden de la salida puede variar.
-

3. Obtener un Elemento con get()

Para recuperar un valor, se utiliza el método `get()`, proporcionando la clave del elemento que se desea obtener.

Java

```
int edadDeAna = edades.get("Ana");
System.out.println("3. La edad de Ana es: " + edadDeAna);
System.out.println("-----");
```

Explicación:

- `edades.get("Ana")`: Este método busca la clave "Ana" dentro del HashMap.
 - Si la encuentra, devuelve el valor asociado a ella (en este caso, 25). Si no la encuentra, devolvería `null`.
 - La operación es muy rápida porque, en lugar de recorrer toda la lista, calcula el *hash* de "Ana" y va directamente a la ubicación donde debería estar ese valor.
 - **Salida:** 3. La edad de Ana es: 25.
-

4. Actualizar un Elemento

Si se utiliza el método `put()` con una clave que ya existe en el mapa, el valor anterior asociado a esa clave será reemplazado por el nuevo.

Java

```
// Si la clave ya existe, put() la sobrescribe
edades.put("Carlos", 28);
System.out.println("4. Edad de Carlos actualizada: " + edades);
System.out.println("-----");
```

Explicación:

- `edades.put("Carlos", 28)`: El HashMap detecta que la clave "Carlos" ya existe.
 - En lugar de crear una nueva entrada, simplemente actualiza el valor asociado a "Carlos", cambiando 22 por 28.
 - **Salida:** 4. Edad de Carlos actualizada: {Ana=25, Luis=30, Carlos=28}. El valor de Carlos ha sido modificado.
-

5. Eliminar un Elemento con `remove()`

Para quitar un par clave-valor del mapa, se utiliza el método `remove()`, especificando la clave del elemento a eliminar.

Java

```
edades.remove("Luis");
System.out.println("5. Elemento 'Luis' eliminado: " + edades);
System.out.println("-----");
```

Explicación:

- `edades.remove("Luis");` El método busca la clave "Luis" y, si la encuentra, elimina tanto la clave como su valor asociado del `HashMap`.
 - Si la clave no existiera, el método no haría nada y devolvería `null`.
 - **Salida:** 5. Elemento 'Luis' eliminado: {Ana=25, Carlos=28}. El par Luis=30 ha sido eliminado con éxito.
-

6. Recorrer el HashMap

Dado que un `HashMap` no tiene un orden definido, no se puede recorrer con un bucle `for` tradicional basado en índices. En su lugar, se utilizan otras técnicas. El ejemplo muestra una de las más comunes: iterar sobre su conjunto de entradas (`entrySet`).

Java

```
System.out.println("6. Recorriendo el HashMap:");
for (Map.Entry<String, Integer> entrada : edades.entrySet()) {
    System.out.println("Nombre: " + entrada.getKey() + ", Edad: " +
        entrada.getValue());
}
```

Explicación:

- `edades.entrySet()`: Este método devuelve una vista de todos los pares clave-valor que contiene el mapa. Cada par es un objeto de tipo `Map.Entry`.
- `for (Map.Entry<String, Integer> entrada : ...)`: Se utiliza un bucle "for-each" para iterar sobre cada `Map.Entry` (que hemos llamado `entrada`) del conjunto.
- `entrada.getKey()`: Este método devuelve la clave del par actual (por ejemplo, "Ana").
- `entrada.getValue()`: Este método devuelve el valor del par actual (por ejemplo, 25).

Salida:

6. Recorriendo el HashMap:

Nombre: Ana, Edad: 25

Nombre: Carlos, Edad: 28

- El bucle imprime cada clave y su valor correspondiente hasta que se han recorrido todos los elementos del mapa.

TALLER A RESOLVER CON HASHMAP

Diseñe y desarrolle los siguientes programas de acuerdo a los requerimientos dados.

Inventario de Tienda: Crea un `HashMap<String, Double>` donde la clave sea el nombre del producto y el valor su precio. Imprime el precio de un producto específico y luego calcula el valor total de todo el inventario sumando los valores.

Traductor Simple: Crea un diccionario `HashMap<String, String>` (Palabra Español -> Palabra Inglés). Pide al programa que traduzca una palabra dada; si no existe, muestra un mensaje de "Palabra no encontrada".

Gestión de Grupos (Valor = ArrayList): Un sistema para organizar estudiantes por cursos. La clave es el nombre del curso y el valor es una **lista** de nombres de estudiantes.

Sistema de Notas Completo (Valor = HashMap): Un sistema para guardar las notas de varios estudiantes en distintas materias. La clave principal es el nombre del estudiante, y el valor es **otro mapa** que contiene (Materia -> Nota).

Ejercicios adicionales

1. Cree un programa que pida al usuario el nombre de un producto existente en una tienda, luego, que le muestre el precio del producto. El usuario debe poder elegir de entre por lo menos cinco productos.
2. Cree un programa que almacene los documentos y nombres de diez usuarios, donde a cada documento, corresponda su respectivo nombre. Imprima todos los nombres de los usuarios usando el ciclo `foreach`.
3. Cree un programa que solicite al usuario el nombre de uno de los cinco continentes, luego, muestre cinco, países del continente seleccionado por el usuario. Use el ciclo `for`.
4. Cree un programa que cumpla con los siguientes requerimientos: En una clínica, se requiere un programa, donde el usuario pueda consultar el día de su cita mediante su documento. La cita debe tener día y fecha. Si el usuario consulta, el programa debe mostrarle sus nombres, seguidos del día y hora de su cita. Una vez hecha la consulta, el programa le debe mostrar al usuario un mensaje preguntándole si desea cambiar el día ó fecha de su cita, de ser así el programa debe realizar tal cambio y mostrarle al usuario que el cambio solicitado ha sido exitoso.