# *j a v a*
# HASHMAP

## Que es?

- Es una implementación de la interfaz Map<K, V>.
- Almacena pares clave-valor (key - value).
- Permite acceso muy rápido (por lo general en tiempo constante, O(1)) para operaciones como get(), put(), remove().

## Cómo funciona internamente

- **Hashing de la clave**

Se calcula hashCode() de la clave a un entero.

- **Índice en la tabla**

Se toma ese hashCode, se transforma para determinar en qué "bucket" o casilla del array se coloca.

- **Buckets / celdas**

Cada posición del array es un bucket, puede contener 0, 1 o varios pares.

## Principales operaciones

| VENTAJAS | DESVENTAJAS |
|---|---|
| Muy rápido para búsquedas, inserciones y borrados (O(1) promedio) | No conserva orden (no es ordenado por clave ni por inserción) |
| Permite claves null (una vez) y valores null | Uso de memoria puede ser alto (por los buckets, nodos, estructuras auxiliares) |
| Ampliamente usado, bien optimizado en Java | En peor caso (muchas colisiones) puede degradar: operaciones pueden acercarse a O(n) |

## Principales operaciones

| metodo | que hace |
|---|---|
| put(K key, V value) | Inserta un par clave-valor. Si la clave ya existe, reemplaza el valor. |
| get(Object key) | Devuelve el valor asociado a la clave, o null si no existe |
| remove(Object key) | Elimina el par clave-valor para esa clave. |
| containsKey(Object key) | Comprueba si existe la clave. |
| containsValue(Object value) | Comprueba si existe algún par con ese valor. |
| size() | Número de pares clave-valor. |
| isEmpty() | True si no contiene ningún par. |

## Manejo de Colisiones

Colisiones en HashMap (Java):

- Si dos claves distintas caen en el mismo índice ↠ colisión.
- Manejo:
  - Si el bucket ya tiene un par, se usa una lista enlazada dentro del bucket.