Apuntes Java - Mapas

Profesor: Luis Javier López López

Índice

Mapas en Java

Introducción

En Java, los mapas son implementaciones de la interfaz **Map** que proporciona una colección de pares clave-valor, donde cada clave es única. Algunas de las implementaciones más comunes son **HashMap**, **TreeMap**, y **LinkedHashMap**. Aquí te daré una explicación detallada sobre cómo funcionan los mapas en Java:

Algunas características clave de los Mapas en Java son:

1. **Colección de pares clave-valor:** Un mapa es una colección de elementos, donde cada elemento está compuesto por un par clave-valor. Cada clave es única dentro del mapa.
2. **No permite duplicados de claves:** No puede haber dos claves iguales en un mapa. Si intentas agregar una clave que ya existe, el valor asociado a esa clave se actualiza.
3. **Interfaz Map:** La interfaz Map define métodos para trabajar con mapas en Java, proporcionando operaciones como agregar, eliminar, buscar y recuperar elementos.
4. **Implementaciones diversas:**Java ofrece varias implementaciones de la interfaz Map, cada una con sus propias características. Algunas de las implementaciones más comunes son *HashMap, TreeMap y LinkedHashMap.*
5. **Eficiencia en la búsqueda:** La mayoría de las implementaciones de mapas están diseñadas para realizar búsquedas eficientes. Por ejemplo, *HashMap* utiliza tablas de dispersión para lograr acceso rápido a los elementos.
6. **Ordenación (depende de la implementación):** Algunas implementaciones de mapas, como *TreeMap y LinkedHashMap*, ofrecen cierto grado de ordenación en los elementos. *TreeMap* ordena naturalmente o mediante un comparador, mientras que *LinkedHashMap* mantiene el orden de inserción.
7. **Admite valores nulos (depende de la implementación):** Algunas implementaciones, como *HashMap*, permiten que tanto las claves como los valores sean nulos. Otras implementaciones, como *TreeMap*, tienen restricciones en la nulidad.
8. **Iteración sobre claves, valores y pares clave-valor:** La interfaz *Map* proporciona métodos para iterar sobre las claves (*keySet()*), los valores (*values()*) y los pares clave-valor (*entrySet()*), facilitando la manipulación y procesamiento de los elementos del mapa.
9. **Tamaño dinámico:** Los mapas pueden crecer o decrecer dinámicamente según la cantidad de elementos que contienen. No es necesario especificar el tamaño inicial.
10. **Funciones hash (depende de la implementación):** Algunas implementaciones, como HashMap, utilizan funciones hash para distribuir eficientemente las claves en la estructura de datos subyacente, mejorando el rendimiento de las operaciones de búsqueda y recuperación.

Implementaciones comunes de la interfaz 'Map'

Las implementaciones más comunes de la interfaz Map en Java son:

- **HashMap**:
  - Basado en la tabla de dispersión (hash table).
  - No garantiza el orden de los elementos.
  - Permite claves y valores nulos.

```
Map<String, Integer> hashMap = new HashMap<>();
```

- **TreeMap**:
  - Implementación basada en árboles de búsqueda balanceados.
  - Los elementos se ordenan naturalmente o mediante un comparador proporcionado.

```
Map<String, Integer> treeMap = new TreeMap<>();
```

- **LinkedHashMap**:
  - Mantiene el orden de inserción de los elementos.
  - Combina las características de *HashMap* y *LinkedList*.

```
Map<String, Integer> linkedHashMap = new LinkedHashMap<>();
```

Principales Métodos

Estos son algunos de los métodos más comunes de la interfaz **Map** en Java:

- **put(K key, V value)**: Agrega un par clave-valor al mapa. Si la clave ya existe, actualiza el valor asociado con esa clave.

```
Map<String, Integer> map = new HashMap<>();
map.put("Juan", 25);
map.put("María", 30);
```

- **get(Object key)**: Devuelve el valor asociado con la clave especificada. Si la clave no está presente, devuelve null.

```
Integer edadJuan = map.get("Juan");
```

- **remove(Object key)**: Elimina el par clave-valor asociado con la clave especificada.

```
map.remove("Juan");
```

- **containsKey(Object key)**: Verifica si el mapa contiene la clave especificada.

```
if (map.containsKey("Juan")) {
    // Hacer algo si la clave está presente}
```

- **containsValue(Object value)**:Verifica si el mapa contiene al menos un valor igual al especificado.

```
if (map.containsValue(30)) {
    // Hacer algo si el valor está presente  }
```

- **keySet()**:Devuelve un conjunto de todas las claves en el mapa.

```
Set<String> keys = map.keySet();
```

- **values()**:Devuelve una colección de todos los valores en el mapa.

```
Collection<Integer> values = map.values();
```

- **entrySet()**:Devuelve el número de elementos en la lista.

```
Set<Map.Entry<String, Integer>> entrySet = map.entrySet();
for (Map.Entry<String, Integer> entry : map.entrySet()) {
    System.out.println("Clave: " + entry.getKey() + ", Valor: " +
entry.getValue());
}
```