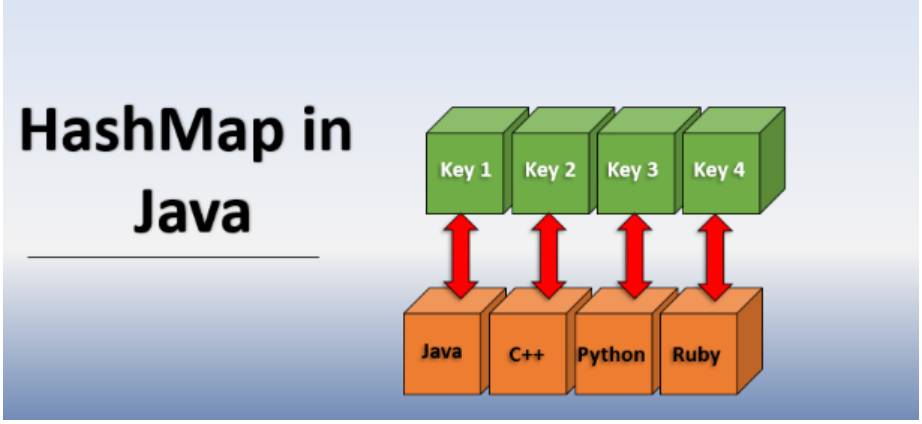
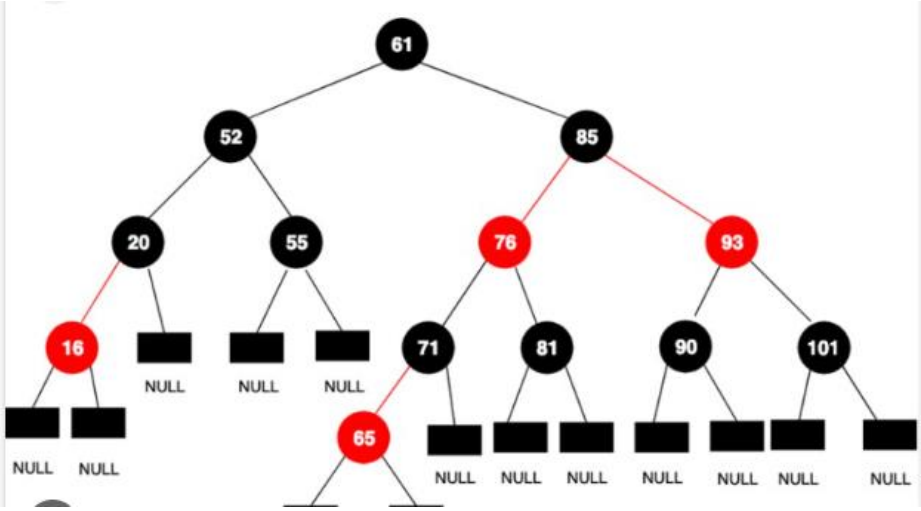
Un **HashMap** o (Map en JAVA) es un mapeo de clave-valor, lo que significa que cada clave se asigna exactamente a un valor y que podemos usar la clave para recuperar el valor correspondiente de un HashMap. A diferencia de las colecciones, que guardan los valores de forma ordenada y cuyo indice es simpre un número entero (int), los HashMap guardan sus valores con un indice definido por el programador cómo por ejemplo un String.



A TreeMap es una implementación basada en árbol de Map interfaz. Se extiende el AbstractMap clase e implementa el NavigableMap, Cloneable, y Serializable interfaces. A TreeMap permite un acceso rápido a los valores utilizando las claves como criterio de búsqueda.

A TreeMap almacena pares clave-valor en un árbol rojo-negro, que es un árbol de búsqueda binario auto equilibrado que mantiene algunas propiedades para garantizar que sea equilibrado y eficiente. Estas propiedades aseguran que la altura del árbol sea siempre logarítmica con respecto al número de nodos, lo que significa que las operaciones en el árbol son rápidas y consistentes.

LinkedHashMap: esta implementación almacena las claves en función del orden de inserción. Es, simplemente, un poco más costosa que HashMap



Similitudes entre HashMap y TreeMap

`HashMap` y `TreeMap` Son dos estructuras de datos diferentes que tienen diferentes propiedades y comportamientos. Comencemos discutiendo algunas de las similitudes entre los dos:

1. `HashMap` y `TreeMap` son miembros de Java Collections Framework e implementan `java.util.Map` interfaz.
2. Implementaciones de `HashMap` y `TreeMap` no están sincronizados, lo que significa que no es seguro para subprocesos de forma predeterminada.
3. Para evitar el acceso no sincronizado accidental al mapa, `HashMap` y `TreeMap` se puede envolver usando el `Collections.synchronizedSortedMap()` método.
4. Los iteradores devueltos por `HashMap` y `TreeMap` Los métodos iteradores son rápidos. Iteradores a prueba de fallos lanza `ConcurrentModificationException` si el mapa se modifica

estructuralmente después de que se crea el iterador sin usar el iterador `remove()` método.

Diferencias entre HashMap y TreeMap

Ahora que hemos visto lo que `HashMap` y `TreeMap` están en Java, resumamos algunas de sus diferencias clave:

1. Implementación

- `HashMap` es una implementación basada en tabla hash de `Map` interfaz, mientras `TreeMap` es una implementación basada en árbol de `Map` interfaz.
- `HashMap` utiliza una función hash para asignar claves a valores, mientras que `TreeMap` utiliza un árbol rojo-negro para almacenar pares clave-valor.
- `HashMap` puede almacenar cualquier tipo de clave, mientras `TreeMap` solo permite claves que sean comparables o que tengan un comparador personalizado.

2. Orden

- `HashMap` no ofrece ninguna garantía sobre el orden de los elementos del `Map`. Significa que no podemos asumir ningún orden mientras iteramos sobre claves y valores de un `HashMap`.
- `TreeMap` proporciona un orden ordenado de los elementos en el `Map` según su orden natural o un comparador personalizado. Significa que siempre podemos esperar un orden consistente al iterar sobre claves y valores de un `TreeMap`.

3. Valores nulos

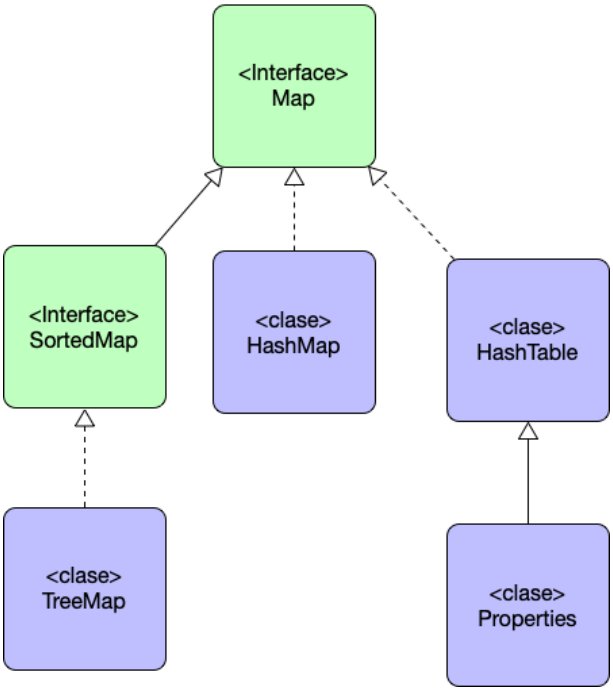
- `HashMap` permite almacenar como máximo una clave nula y muchos valores nulos. Significa que podemos usar null como clave o valor válido en un `HashMap`.
- `TreeMap` no permite una clave nula, pero puede contener muchos valores nulos. Significa que no podemos usar null como clave válida en un `TreeMap`, pero podemos usarlo como un valor válido.

4. Rendimiento

- `HashMap` generalmente funciona mejor que `TreeMap` en términos de complejidad de tiempo y es adecuado para operaciones de búsqueda rápida. La complejidad de tiempo promedio para operaciones comunes como obtener, colocar, eliminar, contiene clave, etc., es `O (1)` por `HashMap` y `O(log(n))` por `TreeMap`, dónde `n` es el número de elementos en el `Map`.
- Sin embargo, `HashMap` puede funcionar peor que `TreeMap` en términos de complejidad espacial. La complejidad espacial para `HashMap` es `O (n + m)`, dónde `m` es el número de depósitos o ranuras en la matriz, mientras que la complejidad del espacio para `TreeMap` es `O(n)`, dónde `n` es el número de elementos en el `Map`.
- Además, `HashMap` puede sufrir una degradación del rendimiento cuando hay muchas colisiones o códigos hash duplicados. En esos casos, `TreeMap` puede ofrecer un mejor rendimiento ya que no depende del hash.

Java Map

El concepto de Mapa o diccionario define los conceptos de clave y valor. Un diccionario siempre contiene una palabra y luego su definición o descripción. En un mapa en Java pasa algo similar disponemos del tipo clave y luego de su valor. Vamos a ver las clases que están relacionadas con el concepto de diccionario.



En este caso existen 4 clases ligadas a los diccionarios:

HashMap: La implementación por defecto y la que se usa más habitualmente

HashTable: La implementación legacy que existe desde Java 1.0 y está sincronizada. De esta clase hereda la clase de **Properties** que es usada para cargar parejas de claves valor desde ficheros de texto.

TreeMap: Un diccionario ordenado que implementa no solo la interface Map sino la interface SortedMap que permite operaciones ligadas al ordenamiento.