9.C. Conjuntos.

3. Conjuntos (III).

¿En qué se diferencian las estructuras `LinkedHashSet` y `TreeSet` de la estructura `HashSet`? Ya se comento antes, y es básicamente en su funcionamiento interno.

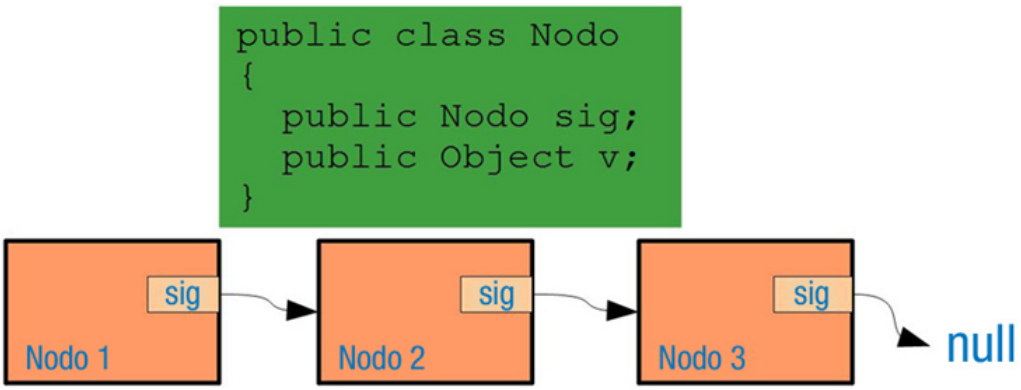

Imagen procedente de curso de Programación del MECD.

La estructura `LinkedHashSet` es una estructura que internamente funciona como una lista enlazada, aunque usa también tablas hash para poder acceder rápidamente a los elementos. Una lista enlazada es una estructura similar a la representada en la imagen de la derecha, la cual está compuesta por nodos (elementos que forman la lista) que van enlazándose entre sí. Un nodo contiene dos cosas: el dato u objeto almacenado en la lista y el siguiente nodo de la lista. Si no hay siguiente nodo, se indica poniendo nulo (`null`) en la variable que contiene el siguiente nodo.

Las listas enlazadas tienen un montón de operaciones asociadas que podremos programar, como eliminación de un nodo de la lista, inserción de un nodo al final, al principio o entre dos nodos, ordenación de nodoes, etc. Gracias a las colecciones podremos utilizar listas enlazadas sin tener que complicarnos en detalles de programación.

La estructura `TreeSet`, en cambio, utiliza internamente árboles. Los árboles son como las listas pero mucho más complejos. En vez de tener un único elemento siguiente, pueden tener dos o más elementos siguientes, formando estructuras organizadas y jerárquicas.

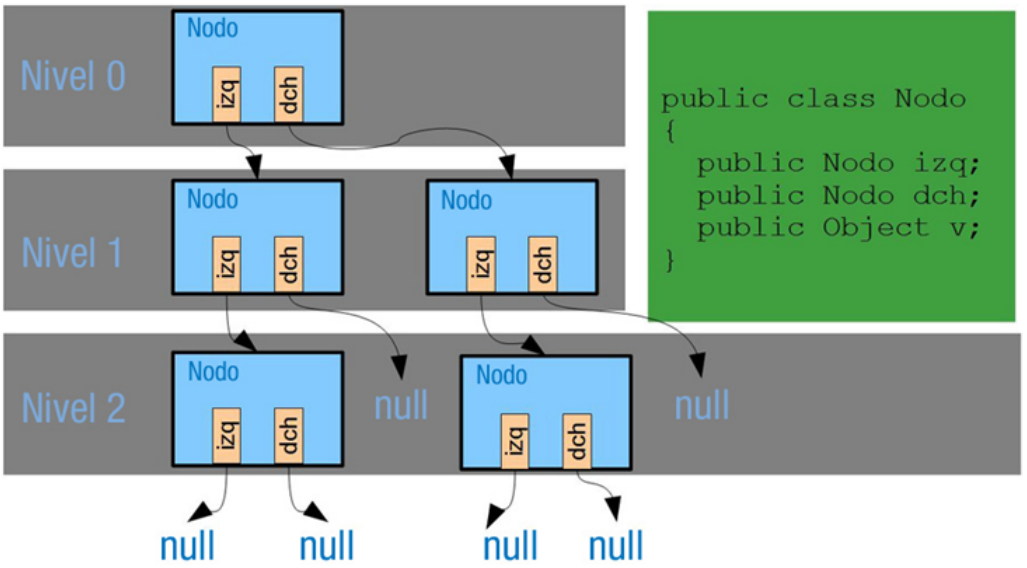

Imagen procedente de curso de Programación del MECD.

Los nodos se diferencian en dos tipos: nodos padre y nodos hijo; un nodo padre puede tener varios nodos hijo asociados (depende del tipo de árbol), dando lugar a una estructura que parece un árbol invertido (de ahí su nombre).

En la figura de la derecha se puede apreciar un árbol donde cada nodo puede tener dos hijos, denominados izquierdo (izq) y derecho (dch). Puesto que un nodo hijo puede también ser padre a su vez, los árboles se suelen visualizar para su estudio por niveles para entenderlos

mejor, donde cada nivel contiene hijos de los nodos del nivel anterior, excepto el primer nivel (que no tiene padre).

Los árboles son estructuras complejas de manejar y que permiten operaciones muy sofisticadas. Los árboles usados en los `TreeSet`, los árboles rojo-negro, son árboles auto-ordenados, es decir, que al insertar un elemento, este queda ordenado por su valor de forma que al recorrer el árbol, pasando por todos los nodos, los elementos salen ordenados. El ejemplo mostrado en la imagen es simplemente un árbol binario, el más simple de todos.

Ver más sobre árboles rojo-negro en Wikipedia.

Nuevamente, no se va a profundizar en las operaciones que se pueden realizar en un árbol a nivel interno (inserción de nodos, eliminación de nodos, búsqueda de un valor, etc.). Nos aprovecharemos de las colecciones para hacer uso de su potencial. En la siguiente tabla tienes un uso comparado de `TreeSet` y `LinkedHashSet`. Su creación es similar a como se hace con `HashSet`, simplemente sustituyendo el nombre de la clase `HashSet` por una de las otras. Ni `TreeSet`, ni `LinkedHashSet` admiten duplicados, y se usan los mismos métodos ya vistos antes, los existentes en la interfaz `Set` (que es la interfaz que implementan).

Ejemplos de utilización de los conjuntos `TreeSet` y `LinkedHashSet`.		
	Conjunto `TreeSet`.	**Conjunto `LinkedHashSet`.**

Ejemplo de uso	`TreeSet <Integer> t;`
	`t=new TreeSet<Integer>();`
	`t.add(new Integer(4));`
	`t.add(new Integer(3));`
	`t.add(new Integer(1));`
	`t.add(new Integer(99));`
	`for (Integer i:t) System.out.println(i);`
		`LinkedHashSet <Integer> t;`
		`t=new LinkedHashSet<Integer>();`
		`t.add(new Integer(4));`
		`t.add(new Integer(3));`
		`t.add(new Integer(1));`
		`t.add(new Integer(99));`
		`for (Integer i:t) System.out.println(i);`

Resultado mostrado por pantalla	1 3 4 99 (el resultado sale ordenado por valor)	4 3 1 99 (los valores salen ordenados según el momento de inserción en el conjunto)

Autoevaluación

Un árbol cuyos nodos solo pueden tener un único nodo hijo, en realidad es una lista. ¿Verdadero o falso?

- ○ Verdadero.
- ○ Falso.