Universidad Complutense de Madrid.

Facultad de Ingeniería Informática.

MARP I.

**Implementación de un árbol 2-3-4**

**con las operaciones de**

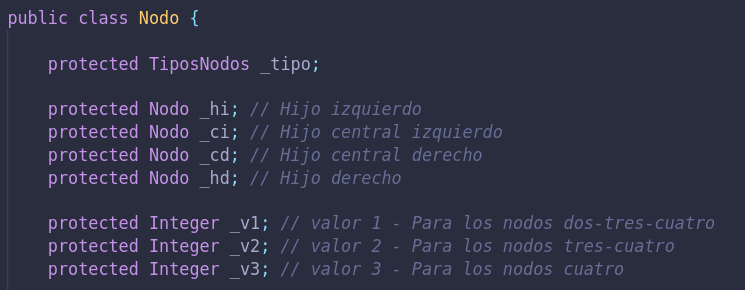
**buscar, insertar y borrar una clave.**

Víctor Manuel Cavero Gracia

**Desarrollo del código fuente.**

He implementado el árbol 2-3-4 en el lenguaje **Java**. Para comenzar me he definido una **clase *Nodo****,* esta consta de una variable *\_tipo* que es del **enumerado** **TiposNodos** (sus posibilidades son NODODOS, NODOTRES y NODOCUATRO) además de 3 instancias de la clase *Integer (****\_v1, \_v2 y \_v3****)* para guardar sus valores y se tienen punteros a sus nodos hijos *(****\_hi, \_ci, \_cd y \_hd****)*.

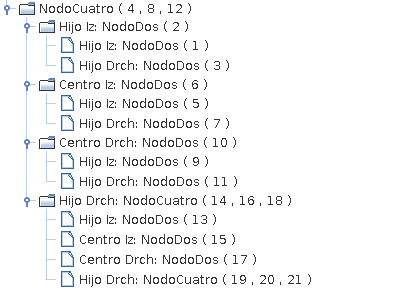
Aclaración: Los **NODODOS** usan las variables \_hi para el hijo izquierdo y \_hd para su hijo derecho, los **NODOTRES**  incluyen *\_ci* para su hijo central y los **NODOCUATRO** utilizan además *\_cd* para el nuevo hijo.

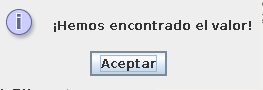


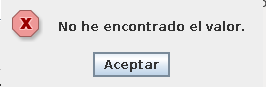
Para la representación he utilizado el paquete *javax.swing* y en concreto para la mostrar el árbol de manera correcta he usado la clase *JTree*.

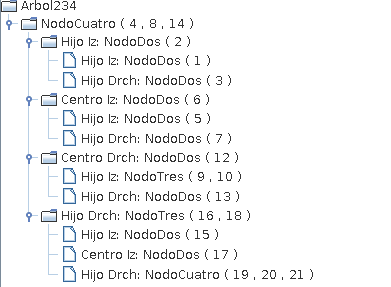
En cuanto al archivo principal (**Arbol234.java**) este incluye todo el código de los algoritmos de **búsqueda, inserción y borrado**. Solo consta de un atributo *\_raiz* de tipo *Nodo* a partir del cual se genera el árbol completo. En los comentarios del archivo se explica claramente la casuística de las funciones y el funcionamiento de la solución planteada.

**Casos de prueba sencillos.**

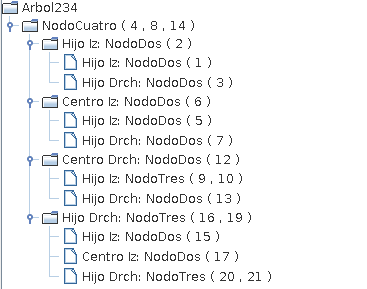
Inserción creciente desde 1 hasta 21.

Buscar el valor 13 en el árbol nos dará este aviso.

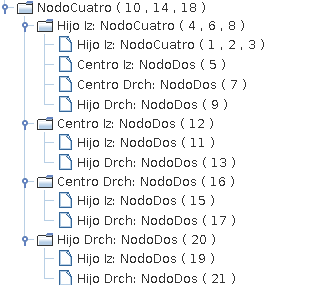
Buscar el valor 0 en el árbol nos dará.

Borrar 11.

Borrar 18.



Nuevo árbol, inserción decreciente desde 21 hasta 1.



**Casos de prueba voluminosos.**

Para la obtención de los casos de prueba voluminosos he generado números aleatorios y realizada la operación en cuestión a temporizar. Como los valores obtenidos en mili-segundos eran demasiado pequeños tuve que ejecutar la operación 1000 veces y realizar la media de todas ellas. He obtenido datos en todos los casos hasta llegar a 5000 elementos en el árbol.

Como podemos observar gracias a las graficas obtenidas se cumple con mayor o menor similitud que el coste real de las operaciones que debería tener en su caso peor y en el caso promedio una complejidad O(log n).

