Programación III

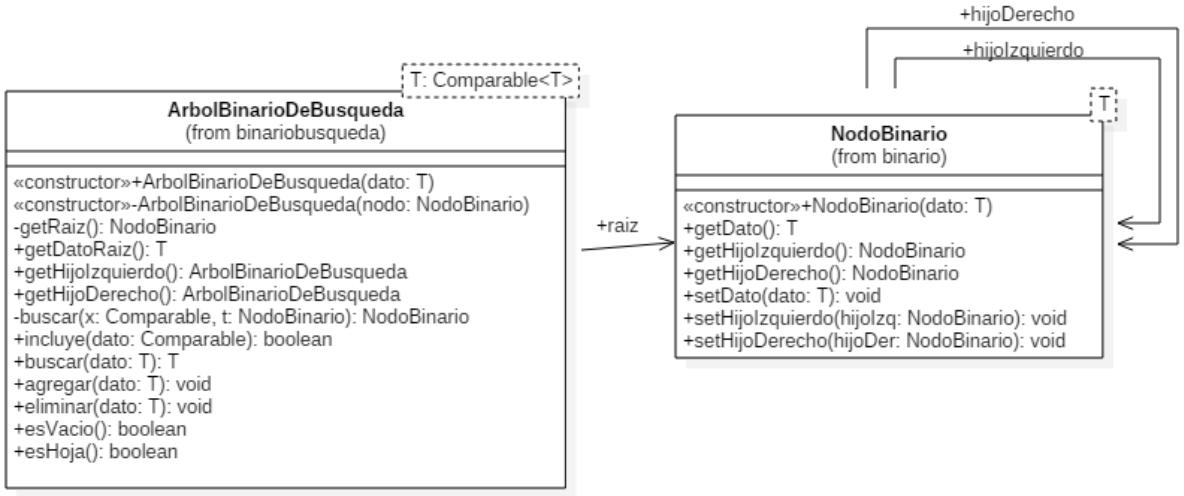
TEMA 6-B: ABB (Árbol Binario de Búsqueda)

Puede continuar trabajando en su proyecto Programacion3. El archivo zip descargado desde la página de la cátedra no es un proyecto eclipse, por tanto:

1. descomprima el archivo zip
2. sobre la carpeta **src** de su proyecto Programacion3 haga click con el botón derecho del mouse y seleccione la opción *Import* > *FileSystem*.
3. Haga click en "Browse" y busque la carpeta descomprimida y seleccione la carpeta **src** (haga click para que aparezca el check seleccionado)
4. Haga click en el botón finalizar

1. **Conceptos básicos de ABB**
   - **a.** Muestre en papel (dibuje) las transformaciones que sufre un árbol binario de búsqueda (inicialmente vacío) al insertar cada uno de los siguientes elementos: 3, 1, 4, 6, 8, 2, 5, 7.
   - **b.** Muestre cómo queda el árbol al eliminar los elementos: 7, 1 y 6.
   - **c.** A partir de un árbol binario de búsqueda nuevamente vacío, muestre las transformaciones que sufre al insertar cada uno de los siguientes elementos: 5, 3, 7, 1, 8, 4, 6.
   - **d.** Dibuje como queda el árbol al eliminar los elementos: 5, 3 y 7.
   - **e.** ¿Qué puede concluir sobre la altura del árbol a partir de a) y c)?

2. **Considere** la siguiente especificación de la clase **ArbolBinarioDeBusqueda** (con la representación hijo izquierdo e hijo derecho):



ACLARACIÓN IMPORTANTE:
A diferencia de las prácticas anteriores donde el tipo de dato almacenado es T (genérico), un árbol binario de búsqueda **requiere** que el tipo de dato almacenado sea **"comparable"** con otros elementos, por tanto en su definición se requiere **<T extends Comparable<T>>** (lo cual indica que la clase representada por la variable T implemente la interface Comparable).

Las clases que implementan la interface *java.lang.Comparable<T>* permiten que sus instancias se puedan comparar entre sí. Para lograr esto, deben implementar el método compareTo(T), el cual retorna el resultado de comparar el receptor del mensaje con el parámetro recibido. Este valor se codifica con un entero, el cual presenta la siguiente característica:

= 0: si el objeto receptor es igual al pasado en el argumento.
> 0: si el objeto receptor es mayor que el pasado como parámetro.
< 0: si el objeto receptor es menor que el pasado como parámetro.

A modo de ejemplo, podría tener un ABB de Alumno, sólo si Alumno es comparable con otro Alumno (ya sea por legajo, edad, dni, etc). Esto quiere decir, que Alumno debería implementar la interfaz Comparable y por tanto, implementar el método **int compareTo(Alumno otroAlumno)**.

La descripción de cada método del ABB es la siguiente:

- El constructor **ArbolBinariodeBusqueda(T dato)** inicializa un árbol que tiene como raíz un nodo binario de búsqueda. Este nodo tiene el dato pasado como parámetro y ambos hijos nulos.
- El constructor **ArbolBinariodeBusqueda(NodoBinario<T> nodo)** inicializa un árbol donde el nodo pasado como parámetro es la raíz. Este método es privado y se podrá usar en la implementación de las operaciones sobre el árbol.
- El método **getRaiz():NodoBinario <T>** retorna el nodo ubicado en la raíz del árbol.
- El método **getDatoRaiz():T** retorna el dato almacenado en el NodoBinario raíz del árbol, sólo si el árbol no es vacío.
- El método **esVacio(): boolean** indica si el árbol es vacío (no tiene dato cargado).
- Los métodos **getHijoIzquierdo():ArbolBinariodeBusqueda<T>** y **getHijoDerecho():ArbolBinariodeBusqueda<T>** retornan los árboles hijos que se ubican a la izquierda y derecha del nodo raíz respectivamente. Están indefinidos para un árbol vacío.
- El método **incluye (T dato):boolean** retorna un valor booleano indicando si el dato recibido está incluido en el árbol.
- El método **buscar (T dato):T** retorna el valor almacenado en el árbol que es igual al dato recibido.

a. Analice las implementaciones en JAVA brindadas por la cátedra y complete los siguientes métodos:
    - El método **agregar (T dato)** agrega el dato indicado al árbol. En caso de encontrar un elemento igual dentro del árbol, reemplaza el existente por el recibido.

3. MenoresOrdenados.

a. En una clase llamada ABBUtil, escriba un método llamado **menoresOrdenados**, que dado como argumento un ABB de objetos de tipo

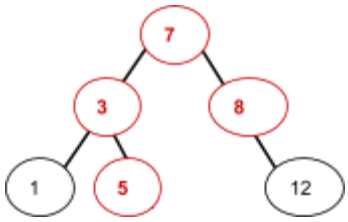
Integer, devuelve una lista ordenada con todos los valores del árbol menores a un valor dado. **Nota: NO se permite pasar los valores del árbol a una lista para luego ordenarla.**

i. Analice la estrategia que utilizará para recorrer el árbol
ii. Implemente

b. Indique qué diferencias existirían en la lógica de su código si en lugar de tratarse de un ABB, se recibiera como argumento un árbol AVL.

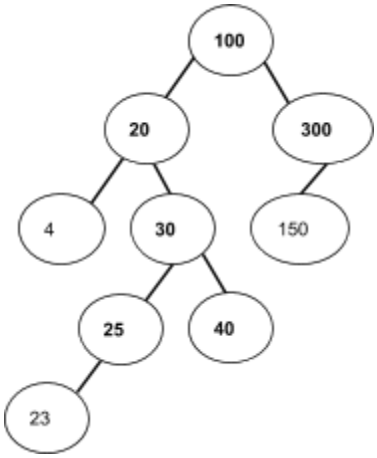
4. **sumaCaminos**. Considere un ABB de números de tipo Integer mayores a 0. Dado un valor x se quiere averiguar todos los posibles caminos a través de los cuales se obtiene dicho valor partiendo desde la raíz del árbol.
   a. Analice la estrategia que utilizará e indique cuál es el caso base.
   b. En la clase ABBUtil, agregue un método llamado sumaCaminos que devuelve lo solicitado.

Ejemplo:



sumaCaminos(ABB, 15) devuelve una lista que contiene:
[[7,3,5], [7,8]]

5. Dado un ArbolBinariodeBusqueda de números enteros positivos y mayores a cero, se quiere obtener el camino recorrido hasta llegar a un determinado valor. En caso de ir por la rama izquierda del árbol, el valor a almacenar en el camino será el valor negativo del mismo.



Es decir, en el siguiente ejemplo, si queremos encontrar el valor 25, el resultado será una lista con los valores: 100, -20, 30, -25.

Note que el nodo con valor 20 y 25 aparecen en la lista resultante como -20 y -25 debido a que se llegó a esos nodo a través de la rama izquierda del nodo con valor 100.