

Objetivos

Unidad 2: Diseño y Construcción de Estructuras de Datos

La actividad planteada en esta hoja de trabajo contribuye al desarrollo del siguiente objetivo específico:

OE2.1. Aplicar apropiadamente una metodología de diseño de estructuras de datos abstractas.

OE2.2. Utilizar correctamente tipos de datos genéricos en el diseño de nuevas estructuras de datos.

Enunciado

Defina el TAD para la estructura de datos árbol binario de búsqueda e implementela en Java haciendo uso de generics.

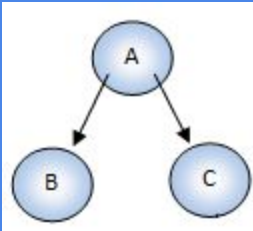
1. **[50pts]** Diligencie la tabla del TAD con todos sus campos: **[1pto]** nombre, **[10pts]** objeto abstracto, **[15pts]** invariante, **[10pts]** operaciones principales, estableciendo entradas, salidas y el tipo de operación (constructora, modificadora o analizadora) y **[14pts]** especificación detallada de dichas operaciones (resumen, precondiciones y poscondiciones).
2. **[50pts]** Implemente en Java la estructura de datos diseñada en el punto anterior haciendo uso de generics.

Su seguimiento se evaluará con relación a esta [rúbrica](#).

Este seguimiento se realizará en parejas como máximo.

Desarrollo:

grupo: Camilo Cordoba y Camilo Escobar

TAD:	<Árbol Binario de Búsqueda>	
		
Inv:	<ul style="list-style-type: none">El root = A y sus hijos son B y C. Donde $\langle B_n \langle A_n \langle C_n \rangle \rangle$	
Operaciones Primitivas:	<p>Insertar: $ABB\langle T \rangle$ (Modificadora)---> un elemento adicional a los que ya tenía el árbol.</p> <p>Eliminar: $ABB\langle T \rangle$ (Modificadora)---> del árbol se eliminó ese nodo con ese atributo.</p> <p>Buscar: $ABB\langle T \rangle$ (analizadora)---> Nodo x.</p> <p>PosOrden: $ABB\langle T \rangle$ (anilizadora) ---> List</p> <p>PreOrden: $ABB\langle T \rangle$ (anilizadora) ---> List</p> <p>InOrden: $ABB\langle T \rangle$ (anilizadora) ---> List</p> <p>Peso: $ABB\langle T \rangle$ (analizadora) ---> entero n</p> <p>Altura: $ABB\langle T \rangle$ (analizadora) ---> entero n</p>	

insertarNodo(Nodo n)

“Permite ingresar un nodo al árbol binario de búsqueda.”

Pre: $ABB<T>$ con (x,y,z) atributos .

Pos: Nodo $n \in ABB<T>$ árbol binario donde $n < Root$ v $n > Root$.

eliminarNodo(var x)

“Permite eliminar un nodo del árbol binario de búsqueda.”

Pre: valor $x \in \exists_{Nodo}$ del árbol.

Pos: En el árbol el nodo a eliminar no está.

buscarNodo(var x)

“Permite buscar un nodo en el árbol binario de búsqueda.”

Pre: True.

Pos: Nodo $n \in ABB<T>$.

preOrden($ABB<T>$)

“Recorrido en secuencia $<A_n, B_n, C_n \dots>$ ”

Pre: True.

Pos: Lista ordenada de forma preOrden.

posOrden($ABB<T>$)

“Recorrido en secuencia $<B_n, C_n, A_n, \dots>$ ”

Pre: True.

Pos: Lista ordenada de forma posOrden..

inOrden($ABB<T>$)

“Recorrido en secuencia $<B_n, A_n, C_n, \dots>$ ”

Pre: True.

Pos: Lista ordenada de forma inOrden.

peso($ABB<T>$)

“Permite determinar el peso de un árbol binario de búsqueda.”

Pre: True.

Pos: El peso del $ABB<T>$.

altura(ABB<T>)

“Permite determinar la altura de un árbol binario de búsqueda.”

Pre: True.

Pos: Altura del ABB<T>.