

ACTIVIDAD LABORATORIO NO.2
LABORATORIO: IMPLEMENTACIÓN DE UN ÁRBOL BINARIO DE
BÚSQUEDA

PRESENTADO POR:
ALEJANDRO DE MENDOZA

PRESENTADO AL PROFESOR:
ING EDWIN EDUARDO MILLAN ROJAS
PROFESOR

FUNDACIÓN UNIVERSITARIA INTERNACIONAL DE LA RIOJA
FACULTAD DE INGENIERÍA – INGENIERIA INFORMÁTICA
BOGOTÁ D.C.
25 DE MARZO
2025

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

TABLA DE CONTENIDO

RESUMEN.....	3
INTRODUCCIÓN	5
DESARROLLO DEL LABORATORIO.....	6
CREACIÓN DEL PROYECTO EN NETBEANS	6
CREACIÓN DE LAS CLASES DEL PROYECTO.....	7
DEFINICIÓN SI UTILIZAR UN ARREGLO, UNA LISTA, UNA COLA O PILA.....	7
DESARROLLO Y EXPLICACIÓN DE LOS CÓDIGOS DE LAS CLASES.....	8
DESARROLLO CLASE NODOABB (NODO DEL ÁRBOL).....	9
DESARROLLO CLASE ARBOLABB	11
DESARROLLO CLASE SEGUNDOLABESTRDATOSABB.....	20
DESARROLLO CLASE PRUEBAARBOLABB	26
RESULTADOS DE CONSOLA	27
CONCLUSIÓN	28
BIBLIOGRAFÍA	29

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

RESUMEN

Para el desarrollo de este laboratorio en el aula de Estructura de Datos, tal y como indica la actividad desarrolle la implementación de un Árbol Binario de Búsqueda (ABB) funcional en Java, aplicando estructuras de datos y algoritmos para realizar operaciones clave como inserción, eliminación, distintos tipos de recorridos, entre otros. Ahora y basándome en las recomendaciones dadas en clase, utilicé NetBeans como entorno de desarrollo, dado que ofrece una integración óptima con Java. Inicialmente, trabajé con JDK 23, pero lo adapté a JDK 21 para una manipulación más eficiente. Además, empleé herramientas auxiliares como OpenJDK, la línea de comandos, el compilador de Java y WinRAR para verificar archivos JAR.

En cuanto al desarrollo y funcionalidades, ejecuté la implementación de un Árbol Binario de Búsqueda donde diseñé una clase Nodo para representar los nodos del árbol y una clase ArbolABB que implementa las operaciones del ABB y cuyas operaciones se basaron en la Inserción de nodos donde se insertaron elementos y se verificó su correcta ubicación según las reglas del ABB. Adicionalmente se ejecutó la eliminación de nodos donde se probó la eliminación de estos con distintos casos (hoja, un hijo, dos hijos). Además de mi parte se ejecuto la implementación de los tres tipos de recorrido: Preorden (raíz → izquierda → derecha), Inorden (izquierda → raíz → derecha) y Postorden (izquierda → derecha → raíz). Luego procedí con la impresión de los valores del ABB en el orden correspondiente después de su creación e inserción de datos. Después procedí con la inserción de nuevos valores en el ABB, asegurando que se mantuviera la estructura ordenada. Los elementos insertados fueron

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

correctamente ubicados según las reglas del ABB. Para luego de esto probar la eliminación de nodos bajo distintos escenarios:

- Eliminación de un nodo hoja.
- Eliminación de un nodo con un solo hijo.
- Eliminación de un nodo con dos hijos.

Para luego crear una clase de prueba para verificar el correcto funcionamiento de todas las operaciones, asegurando que los recorridos, inserciones y eliminaciones se comportaran como se esperaba.

Por último, ejecuté el proceso de compilación y generación del archivo JAR donde se compiló el proyecto en NetBeans asegurando que el JDK seleccionado fuera la versión 21, se generó el archivo ejecutable SegundoLabEstrDatosABB.jar. Además, se verificó su correcto empaquetado explorándolo con WinRAR, luego se ejecutaron las Pruebas y Ejecución del JAR donde y desde la línea de comandos, se ejecutó el JAR con: `java -jar " C:\Users\gran_\Documents\Estudio\Carrera Ingeniería Informatica\SegundoSemestre\EstructuraDeDatos\Laboratorios\Segundo\SegundoLabEstrDatosABB\dist\SegundoLabEstrDatosABB.jar"`. Y los resultados observados fueron la correcta impresión de los elementos del ABB y la inserción y eliminación de elementos reflejadas correctamente en los recorridos.

Ahora y para finalizar este resumen debo indicar que se logró una implementación funcional del ABB con todas sus operaciones esenciales donde la correcta configuración del entorno (NetBeans y JDK 21) fue clave para la generación y ejecución del JAR y se verificó que el ABB respeta sus reglas fundamentales en la inserción, eliminación, recorridos, entre otros.

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

INTRODUCCIÓN

En el presente trabajo, desarrollé la implementación de un Árbol Binario de Búsqueda (ABB), asegurando su correcta funcionalidad a través de diversos procesos clave. A lo largo de este laboratorio, me enfoqué en construir un programa robusto en Java, empleando el entorno de desarrollo NetBeans y utilizando la versión JDK 21 para garantizar compatibilidad y estabilidad.

Desde el inicio, configuré el entorno de trabajo, verificando la correcta instalación del JDK y solucionando posibles conflictos de versión. Posteriormente, implementé los recorridos básicos del ABB (preorden, inorden y posorden), asegurándome de que la estructura se mantuviera correctamente ordenada. Además, desarrollé la funcionalidad de inserción de elementos, permitiendo que nuevos valores se ubicaran en la posición adecuada dentro del árbol.

Uno de los retos más interesantes fue la eliminación de nodos, ya que abordé distintos casos, como la eliminación de un nodo sin hijos, con un solo hijo y con dos hijos, comprobando cada escenario mediante pruebas exhaustivas. Para validar la implementación, desarrollé una clase de prueba que permitió verificar el comportamiento esperado de cada operación.

Una vez finalizada la implementación, generé y empaqueté el proyecto en un archivo JAR, asegurándome de que pudiera ejecutarse de manera independiente desde la línea de comandos. Durante este proceso, enfrenté y resolví problemas relacionados con la versión del JDK, garantizando que el programa se ejecutara sin errores.

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

Respetado profesor este trabajo no solo fortaleció mi comprensión sobre los Árboles Binarios de Búsqueda y sus aplicaciones, sino que también me permitió afianzar habilidades en gestión de entornos de desarrollo, depuración de código y ejecución de programas en Java. Con este laboratorio, debo indicar no solo implementé un ABB funcional, sino que también logré optimizar su uso y comprobación en un entorno real.

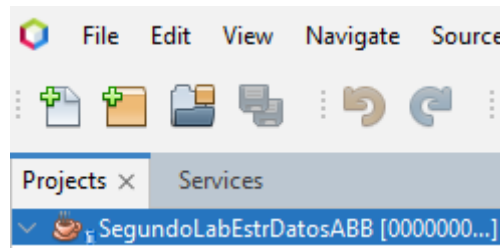
DESARROLLO DEL LABORATORIO

En este caso mi enfoque y como se dio un breve abrebocas en la introducción como escenario real implementé y probé un Árbol Binario de Búsqueda (ABB) en Java, ejecutando operaciones clave como recorridos (preorden, inorden y posorden), inserción y eliminación de elementos. Además, desarrollé una clase de prueba para validar la funcionalidad del árbol y generé un archivo JAR ejecutable. Durante el proceso, resolví problemas de compatibilidad con el JDK y aseguré el correcto desempeño del programa en distintos entornos. A continuación, el desarrollo completo:

Creación Del Proyecto En NETBEANS

Entonces para este desarrollo lo primero de mi parte fue abrir la plataforma de NetBeans y crear el proyecto de Java que en este caso recibe el nombre de “SegundoLabEstrDatosABB”. A continuación, muestro la imagen respectiva de la creación del proyecto en el lenguaje de Java:

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	



Creación De Las Clases Del Proyecto

Luego de proceder a crear el proyecto entonces lo primero fue crear sus clases dentro del paquete de Sources y las clases que procedí a crear son:

- NodoABB → Representa un nodo del árbol.
- ArbolABB → Implementa la lógica del Árbol Binario de Búsqueda.
- SegundoLabEstrDatosABB (Main) → Contiene la ejecución y pruebas.
- PruebaArbolABB → Contiene las pruebas a ejecutar del desarrollo.

Definición Si Utilizar Un Arreglo, Una Lista, Una Cola O Pila

Ahora y luego de proceder a crear las clases lo siguiente fue plantearme el hecho si quería para este desarrollo desarrollar un arreglo, una lista, una cola o pila, a continuación, sus descripciones:

- Arreglo (int [] datos): Para un número fijo de elementos por nodo.
- Lista (ArrayList<Integer> datos): Para una cantidad variable de elementos.
- Cola (Queue<Integer> datos) o Pila (Stack<Integer> datos): Si el orden de inserción/importancia es relevante.

En este caso procedí a desarrollar el código usando una lista o un ArrayList<Integer> para que cada nodo pueda almacenar múltiples valores. Considero es la mejor opción para entrar a mostrar los cambios y se apliquen evitando el ad hoc. Además de las siguientes ventajas:

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

- Redimensionamiento Dinámico: A diferencia de los arrays tradicionales, ArrayList ajusta automáticamente su tamaño cuando se agregan o eliminan elementos, evitando problemas de capacidad fija.
- Fácil Manipulación de Datos: Métodos como `add()`, `remove()` y `.contains()` simplifican la gestión de los elementos, lo que hizo en este caso más sencillo insertar y eliminar nodos en el árbol.
- Acceso Rápido a Elementos: Acceder a un elemento por su índice es más eficiente ($O(1)$) en comparación con estructuras como LinkedList, lo que facilitó en este caso ciertas operaciones dentro del árbol.
- Compatibilidad con Java Collections: ArrayList es compatible con otras estructuras de datos y métodos de Java, lo que facilita su integración con las funciones de impresión y recorrido del árbol.
- Menos Sobrecarga de Memoria que LinkedList: Aunque cada elemento es un objeto Integer, ArrayList consume menos memoria en comparación con LinkedList, ya que no almacena referencias adicionales a nodos previos y siguientes.
- Iteración Eficiente: Usar `for-each` o `Iterator` sobre un ArrayList es más rápido que recorrer un LinkedList, lo que benefició los recorridos del árbol binario.

En resumen, el uso de `ArrayList<Integer>` mejoró la eficiencia, simplificó la manipulación de datos y redujo la complejidad del manejo del árbol binario de búsqueda y por eso su utilización para este desarrollo.

Desarrollo Y Explicación De Los Códigos De Las Clases

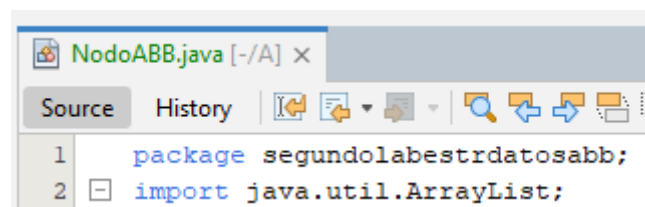
Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

Una vez definí la implementación de una lista en este desarrollo me enfoqué en la implementación y análisis detallado de los códigos que conforman la ejecución de un Árbol Binario de Búsqueda (ABB) en Java. Abordé aspectos clave como la estructura de las clases, los métodos utilizados y su funcionalidad dentro del programa. Por lo que mi énfasis se basa en comprender el comportamiento del árbol en operaciones como inserción, eliminación y recorridos, entre otros., facilitando así el aprendizaje sobre estructuras de datos y su aplicación en programación y es por esto que a continuación, voy a brindar la explicación del desarrollo de cada una de las clases y la ejecución completa del árbol desarrollado.

Desarrollo Clase NodoABB (Nodo Del Árbol)

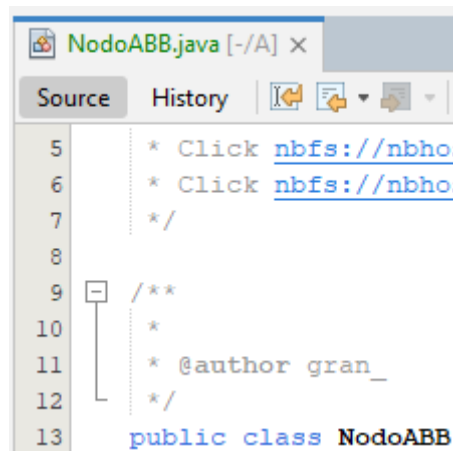
A continuación, el desarrollo completo con su código y la respectiva explicación de este para esta clase:

1. Lo primero que hice para la creación del código de esta clase fue definir el las librerías a importar y en este caso utilice `import java.util.ArrayList;` que Importa la clase `ArrayList` del paquete `java.util.` y por otra parte `ArrayList<Integer>` es una estructura de datos dinámica que permite almacenar múltiples enteros en un nodo. Es la lista como lo indique en su anterioridad. A continuación, la respectiva imagen:



2. Luego procediendo con la explicación del código tenemos la siguiente imagen:

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	



```

5      * Click nbfs://nbho:
6      * Click nbfs://nbho:
7      */
8
9      /**
10     *
11     * @author gran_
12     */
13     public class NodoABB

```

Donde cómo se puede denotar en la imagen brinda el nombre de la clase y declara una clase pública llamada **NodoABB**. Esta clase representa un nodo en un Árbol Binario de Búsqueda (ABB), donde cada nodo contiene una lista de enteros (`ArrayList<Integer> datos`) en lugar de un solo valor y dos referencias (izquierda y derecha) a nodos hijos.

- Ahora procedo a determinar los atributos de la clase, donde como indique anteriormente `ArrayList<Integer> datos`, declara una lista de enteros para almacenar múltiples valores en el mismo nodo y que es útil en caso de duplicados o estructuras donde un nodo debe contener más de un dato.

```
ArrayList<Integer> datos; // Lista para almacenar múltiples valores
```

- Por otra parte, se declara el **NodoABB** izquierda, derecha el cual declara dos referencias a otros nodos del árbol, donde:

- izquierda almacena el subárbol izquierdo.
- derecha almacena el subárbol derecho. A continuación, la respectiva imagen:

```
NodoABB izquierda, derecha;
```

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

5. Ahora procedí a desarrollar el constructor de la clase con la función “public NodoABB (int dato)”, donde definí un constructor que recibe un valor entero (dato) y este se ejecuta automáticamente cuando se crea un nuevo nodo. A continuación, la respectiva imagen:

```

17
18
19
public NodoABB(int dato)
{

```

6. Ahora procedí con la inicialización de la lista y los hijos en donde ingreso el código de “datos = new ArrayList<> ();”, el cual crea una nueva instancia de ArrayList<Integer>, inicializando la lista vacía. Por otra parte, “datos.add(dato);”, agrega el primer valor (dato) a la lista de enteros del nodo y “izquierda = derecha = null;”, que inicializa los hijos del nodo como null, indicando que aún no tiene conexiones con otros nodos. A continuación, la imagen respectiva:

```

18
19
20
21
22
23
24
public NodoABB(int dato)
{
    datos = new ArrayList<>();
    datos.add(dato);
    izquierda = derecha = null;
}

```

Y con esto finalizo la explicación del código de la clase NodoABB.

Desarrollo Clase ArbolABB

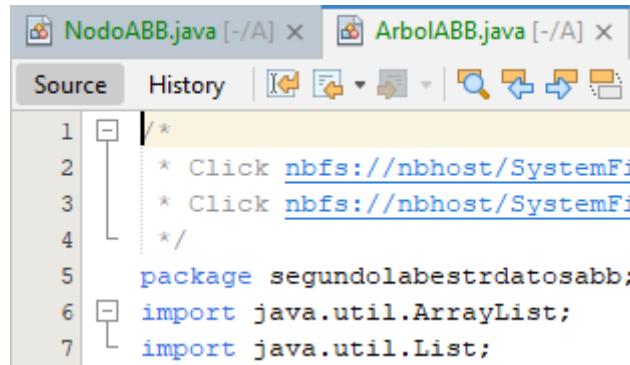
Ahora que ya he definido y explicado el código de la clase NodoABB procedo a definir y explicar el desarrollo de la clase del árbol denotada con el nombre ArbolABB:

1. Lo primero entonces es importar las clases necesarias de java.util, donde ArrayList como ya he explicado me sirve para manejar listas dinámicas de

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

elementos. Y por otra parte List, para definir listas de manera más genérica.

A continuación, la imagen respectiva:



```

1  /*
2   * Click nbfs://nbhost/SystemFi
3   * Click nbfs://nbhost/SystemFi
4   */
5  package segundolabestrdatosabb;
6  import java.util.ArrayList;
7  import java.util.List;

```

- Ahora entonces procedo a declarar la clase public class ArbolABB, la cual define la clase ArbolABB, que representa un Árbol Binario de Búsqueda (ABB). Adicionalmente empleo el código “private NodoABB raiz;” el cual me declara la raíz del árbol, que es de tipo NodoABB e inicialmente será null, indicando que el árbol está vacío.

```

public class ArbolABB
{
    private NodoABB raiz;

```

- Ahora procedo entonces a declarar la raíz del árbol, que es de tipo NodoABB, la cual inicialmente será null, indicando que el árbol está vacío. Y “private List<Integer> ultimosInsertados;”, y “private List<Integer> nuevosInsertados;”, que declaran dos listas para almacenar los valores insertados, donde “ultimosInsertados” guarda un historial de todos los valores insertados en el árbol, y “nuevosInsertados” solo almacena los valores insertados en la última ejecución.

```

public class ArbolABB
{
    private NodoABB raiz;
    private List<Integer> ultimosInsertados; // Lista para almacenar los últimos elementos insertados
    private List<Integer> nuevosInsertados; // Solo para la última ejecución

```

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

4. Entonces procedo ahora con la creación del constructor **con el código**

“public ArbolABB()”, que es el Constructor de la clase ArbolABB, donde:

- this.raiz = null;
- this.ultimosInsertados = new ArrayList<>();
- this.nuevosInsertados = new ArrayList<>();

Entonces como se puede denotar se inicializa la raíz en null y las listas de valores insertados como nuevas instancias de ArrayList<Integer>.

```
public ArbolABB()
{
    this.raiz = null;
    this.ultimosInsertados = new ArrayList<>();
    this.nuevosInsertados = new ArrayList<>();
}
```

5. Ahora procedo a desarrollar el Método para Mostrar Elementos el cual se basa en el código “public void mostrarElementos ()”, el cual es un método público que imprime los elementos del árbol y con “System.out.print("Elementos en el árbol");”, me imprime el mensaje “Elementos en el árbol” antes de mostrar los valores. Y donde el código “mostrarElementos(raiz);” y “System.out.println();”, llama al método recursivo mostrarElementos(NodoABB nodo) para recorrer el árbol e imprimir sus valores.

Ahora y por otra parte procedo a utilizar “private void mostrarElementos (NodoABB nodo)”, el cual es un método recursivo que me muestra los valores en preorden (raíz → izquierda → derecha). Y “if(nodo == null){return;} y me indica que si el nodo es null, finaliza la ejecución. Ahora y para finalizar utilizo el código “System.out.print(nodo.datos + " ");” para

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

imprimir los valores almacenados en el nodo y “mostrarElementos(nodo.izquierda);” y “mostrarElementos(nodo.derecha);”, llaman recursivamente a los subárboles izquierdo y derecho.

```
public void mostrarElementos()
{
    System.out.print("Elementos en el arbol");
    mostrarElementos(raiz);
    System.out.println();
}
private void mostrarElementos(NodoABB nodo)
{
    if(nodo == null)
    {
        return;
    }
    System.out.print(nodo.datos + " "); //Imprime el valor del nodo
    mostrarElementos(nodo.izquierda); //Llama al subarbol izquierdo
    mostrarElementos(nodo.derecha); //Llama al subarbol derecho
}
```

- Entonces ahora, ejecuto el método para Limpiar la Lista de Últimos Insertados y para esto utilizo “public void limpiarUltimosInsertados()” que es un método público que limpia la lista nuevosInsertados antes de una nueva inserción múltiple. Y ejecuto “nuevosInsertados.clear();” que limpia la lista eliminando todos los elementos previos.

```
// Método para limpiar la lista de nuevos insertados antes de cada inserción múltiple
public void limpiarUltimosInsertados()
{
    nuevosInsertados.clear();
}
```

- Ahora entonces procedo con el método para Insertar Elementos con el código “public void insertar (int dato)” el cual es un método público que inserta un nuevo elemento en el árbol. Donde “raiz = insertarRec (raiz, dato);” llama al método recursivo insertarRec (NodoABB nodo, int dato) para insertar el dato en el lugar correcto. Y “ultimosInsertados.add (dato);” y

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

“nuevosInsertados.add (dato);”, guardan el dato insertado en ambas listas (ultimosInsertados y nuevosInsertados).

```
// Método para insertar un elemento en el árbol
public void insertar(int dato)
{
    raiz = insertarRec(raiz, dato);
    ultimosInsertados.add(dato); // Agrega el dato a la lista de últimos insertados
    nuevosInsertados.add(dato); // Solo almacena los insertados en esta ejecución
}
```

8. Luego entonces procedo con el método recursivo para insertar datos que denote con el código “private NodoABB insertarRec(NodoABB nodo, int dato)”, el cual es un **método recursivo** que inserta un dato en el árbol siguiendo las reglas del ABB y con if (nodo == null) {return new NodoABB(dato);} determino si el nodo actual es null, y en dado caso me crea un nuevo nodo con el dato y por otra parte if (dato == nodo.datos.get(0)) { nodo.datos.add(dato);} indica que si el dato ya existe, lo añade a la lista del nodo actual. Y para finalizar “else if (dato < nodo.datos.get(0)) {nodo.izquierda = insertarRec(nodo.izquierda, dato);}else {nodo.derecha = insertarRec(nodo.derecha, dato);} que me indica que si el dato es menor, se inserta en el subárbol izquierdo y si el dato es mayor, se inserta en el subárbol derecho. A continuación la respectiva imagen del código implementado:

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

```
private NodoABB insertarRec(NodoABB nodo, int dato)
{
    if (nodo == null)
    {
        return new NodoABB(dato);
    }
    if (dato == nodo.datos.get(0))
    { // Si el valor ya existe, lo agregamos a la lista
        nodo.datos.add(dato);
    }
    else if (dato < nodo.datos.get(0))
    {
        nodo.izquierda = insertarRec(nodo.izquierda, dato);
    }
    else
    {
        nodo.derecha = insertarRec(nodo.derecha, dato);
    }
    return nodo;
}
```

9. Entonces procedo con el método para mostrar los últimos Insertados con el código “public void mostrarUltimosInsertados()” el cual me muestra los valores insertados en la última ejecución y con “System.out.println (“Últimos elementos insertados: ” + nuevosInsertados);” me imprime la lista nuevosInsertados.

```
// Método para mostrar los últimos elementos insertados en el árbol
public void mostrarUltimosInsertados()
{
    System.out.println("Últimos elementos insertados: " + nuevosInsertados);
}
```

10. Es entonces que procedo con el desarrollo de los recorridos del Árbol iniciando con public void inOrden() que es el **método público** que me inicia el recorrido inorden. Ahora private void inOrderRec(NodoABB nodo) que es el método recursivo que me recorre el árbol en orden (izquierda → raíz → derecha).

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

Ahora y por otra parte utilizo el método public void preOrden() que es el método público que me inicia el recorrido preorden y “private void preOrdenRec(NodoABB nodo)”, que se encarga de ejecutar el método recursivo para recorrer el árbol en preorden (raíz → izquierda → derecha). Para finalmente implementar el código “public void posOrden ()” que es el método público que me inicia el recorrido posorden donde, “private void posOrdenRec(NodoABB nodo)”, ejecuta el método recursivo para recorrer el árbol en posorden (izquierda → derecha → raíz).

```

public void inOrden()
{
    inOrderRec(raiz);
    System.out.println();
}

private void inOrderRec(NodoABB nodo)
{
    if (nodo != null)
    {
        inOrderRec(nodo.izquierda);
        System.out.print(nodo.datos + " ");
        inOrderRec(nodo.derecha);
    }
}

public void preOrden()
{
    preOrdenRec(raiz);
    System.out.println();
}

private void preOrdenRec(NodoABB nodo)
{
    if (nodo != null)
    {
        System.out.print(nodo.datos + " ");
        preOrdenRec(nodo.izquierda);
        preOrdenRec(nodo.derecha);
    }
}

public void posOrden()
{
    posOrdenRec(raiz);
    System.out.println();
}

private void posOrdenRec(NodoABB nodo)
{
    if (nodo != null)
    {
        posOrdenRec(nodo.izquierda);
        posOrdenRec(nodo.derecha);
        System.out.print(nodo.datos + " ");
    }
}

```

11. Continuando con el desarrollo del laboratorio y la implementación del Árbol Binario de Búsqueda (ABB), procedí a desarrollar el código para la eliminación de elementos. Para ello, implementé el método público eliminar (int dato), encargado de iniciar el proceso de eliminación de un nodo con el

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

valor especificado. Este método invoca a `eliminarRec(NodoABB nodo, int dato)`, una función recursiva que localiza y elimina el nodo correspondiente dentro del árbol. La búsqueda del nodo a eliminar se realiza comparando el valor con el del nodo actual, donde, si el dato es menor, se busca en el subárbol izquierdo y para eso utilizo el código `"nodo.izquierda = eliminarRec(nodo.izquierda, dato);"`. Ahora y por otra parte si es mayor, se busca en el subárbol derecho con el código `"nodo.derecha = eliminarRec(nodo.derecha, dato);"`

Ahora y en adición si el nodo contiene múltiples valores, solo se elimina uno de ellos con la siguiente instrucción `"if (nodo.datos.size () > 1){ nodo.datos.remove(0); return nodo;}"`. Y en caso de que el nodo tenga un solo hijo, se reemplaza directamente con ese hijo con el siguiente código `"if (nodo.izquierda == null) {return nodo.derecha; } else if (nodo.derecha == null) {return nodo.izquierda;}"`

Finalmente, si el nodo tiene dos hijos, se sustituye su valor por el menor del subárbol derecho para mantener la estructura del ABB:

- `nodo.datos.set (0, encontrarMinimo(nodo.derecha).datos.get(0));`
- `nodo.derecha = eliminarRec(nodo.derecha, nodo.datos.get(0));`

A continuación, se presentan las imágenes correspondientes que ilustran el proceso.

```
//Borrado de elementos
public void eliminar(int dato)
{
    raiz = eliminarRec(raiz, dato);
}
```

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

```

private NodoABB eliminarRec(NodoABB nodo, int dato)
{
    if (nodo == null)
    {
        return nodo;
    }

    if (dato < nodo.datos.get(0))
    {
        nodo.izquierda = eliminarRec(nodo.izquierda, dato);
    }
    else if (dato > nodo.datos.get(0))
    {
        nodo.derecha = eliminarRec(nodo.derecha, dato);
    }
    else
    {
        // ♦ Luego imprimir que se va a eliminar el nodo
        System.out.println("\n***Eliminando el dato***: " + dato + "\n");

        // ♦ Si el nodo tiene múltiples valores, eliminar solo uno de la lista
        if (nodo.datos.size() > 1)
        {
            nodo.datos.remove(0); // Elimina solo una ocurrencia del valor
            return nodo;
        }

        // ♦ Si solo hay un valor en la lista, aplicar eliminación normal
        System.out.println("Eliminando nodo con valor: " + dato);

        // Caso 1: Nodo sin hijos o con un solo hijo
        if (nodo.izquierda == null)
        {
            return nodo.derecha;
        }
        else if (nodo.derecha == null)
        {
            return nodo.izquierda;
        }

        // Caso 2: Nodo con dos hijos -> Encontramos el sucesor inmediato (mínimo del subárbol derecho)
        nodo.datos.set(0, encontrarMinimo(nodo.derecha).datos.get(0));
        nodo.derecha = eliminarRec(nodo.derecha, nodo.datos.get(0));
    }
    return nodo;
}

// Método auxiliar para encontrar el nodo con el valor mínimo en un subárbol
private NodoABB encontrarMinimo(NodoABB nodo)
{
    while (nodo.izquierda != null)
    {
        nodo = nodo.izquierda;
    }
    return nodo;
}

// Método auxiliar para encontrar el nodo con el valor mínimo en un subárbol
private NodoABB encontrarMinimo(NodoABB nodo)
{
    while (nodo.izquierda != null)
    {
        nodo = nodo.izquierda;
    }
    return nodo;
}

```

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

12. Y finalmente procedo con el método para buscar un elemento con el código “public boolean buscar (int valor) “, el cual es un **método público** que busca un valor en el árbol. Por lo que utilizo “private boolean buscarNodo (NodoABB nodo, int valor)”, que es el método recursivo que verifica si el valor está en el árbol. Ahora con “if (nodo = null) {return false;} verifico que, si el nodo es null, el valor no está en el árbol. Y finalmente con el código “if (nodo.datos.contains(valor)){return true;} determino que si el nodo contiene el valor, retorna true. A continuación, la respectiva imagen:

```
private boolean buscarNodo(NodoABB nodo, int valor)
{
    if (nodo == null)
    {
        return false;
    }
    if (nodo.datos.contains(valor))
    {
        return true;
    }
    return valor < nodo.datos.get(0) ? buscarNodo(nodo.izquierda, valor) : buscarNodo(nodo.derecha, valor);
}
```

Y con esto finalizo la explicación del código de la clase ArbolABB.

Desarrollo Clase SegundoLabEstrDatosABB

Ahora que ya he definido y explicado el código de la clase NodoABB y Árbol ABB procedo a definir y explicar el desarrollo de la clase del main denotada con el nombre “**SegundoLabEstrDatosABB**”:

1. Entonces para iniciar este desarrollo lo primero es declarar la clase principal y lo hago con el siguiente código “public class SegundoLabEstrDatosABB {“, el cual define una clase pública llamada SegundoLabEstrDatosABB, que será el punto de entrada del programa.

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

```

-  /*
   * Click nbfs://nbhost/SystemFileSys
   * Click nbfs://nbhost/SystemFileSys
   */
package segundolabestrdatosabb;

-  /**
   *
   * @author gran_
   */
public class SegundoLabEstrDatosABB
{

```

2. Ahora procedo a desarrollar el método main con el siguiente código “public static void main (String [] args){“, donde main es el método principal de Java, donde comienza la ejecución del programa y String[] args permite recibir argumentos desde la línea de comandos (no se usan en este código).

```

public static void main(String[] args)
{

```

3. Entonces ejecuto la creación del árbol binario de búsqueda con el siguiente código “ArbolABB abb = new ArbolABB ();” donde se crea un objeto abb de la clase ArbolABB, que representa un Árbol Binario de Búsqueda (ABB).

```

ArbolABB abb = new ArbolABB();

```

4. Ahora procedo con la Inserción de valores en el árbol y para eso utilizo el código abb.insertar() y los valores insertados son los siguientes:

- a) abb.insertar(10);
- b) abb.insertar(5);
- c) abb.insertar(15);
- d) abb.insertar(10); // Se almacena en la lista del nodo con 10
- e) abb.insertar(20);

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

f) `abb.insertar(5);` // Se almacena en la lista del nodo con 5

Donde la explicación de la inserción de los valores en el árbol binario de búsqueda es el siguiente:

a) `abb.insertar(10);` → Inserta el nodo raíz con valor 10.

b) `abb.insertar(5);` → Inserta 5 como hijo izquierdo de 10.

c) `abb.insertar(15);` → Inserta 15 como hijo derecho de 10.

d) `abb.insertar(10);` → Parece que se almacena en una lista dentro del nodo 10, en lugar de rechazarlo o manejar duplicados de otro modo.

e) `abb.insertar(20);` → Inserta 20 como hijo derecho de 15.

f) `abb.insertar(5);` → Similar al 10, se almacena en la lista del nodo 5.

```
// Insertamos valores en el árbol
abb.insertar(10);
abb.insertar(5);
abb.insertar(15);
abb.insertar(10); // Se almacena en la lista del nodo con 10
abb.insertar(20);
abb.insertar(5); // Se almacena en la lista del nodo con 5
```

5. Ahora procedo a desarrollar los recorridos del árbol sin imprimir con un mensaje indicando que los recorridos se realizarán, pero sin mostrar resultados (aunque en este código no se están ejecutando todavía). Y con el siguiente código “`System.out.print("\n1. RECORRIDO EN PREORDEN, INORDEN Y POSORDEN (SIN IMPRIMIR)"); System.out.println();`”

```
//Recorrido en preorden, inorden y posorden (sin imprimir).
System.out.print("\n1.RECORRIDO EN PREORDEN, INORDEN Y POSORDEN (SIN IMPRIMIR)");
System.out.println();
```

6. Entonces y luego de es esta y paso entro a verificar la actividad a realiza r y con base en lo indicado en esta ahora procedo a mostrar los elementos del árbol a través de los siguientes códigos:” `System.out.println("\n2.`

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

IMPRESION DE LOS ELEMENTOS DEL ARBOL");" y "abb.mostrarElementos();", donde "abb.mostrarElementos();", imprime los valores almacenados en el árbol.

```
//Mostrar los elementos del árbol
System.out.println("\n2.IMPRESION DE LOS ELEMENTOS DEL ARBOL");
abb.mostrarElementos();
```

7. Continuando con base en la actividad se nos pide ahora ejecutar un proceso de inserción de más elementos entonces procedo con los siguientes códigos: System.out.println("\n3. INSERCIÓN DE ELEMENTOS AL ARBOL"); abb.limpiarUltimosInsertados();

- a) abb.insertar(7);
- b) abb.insertar(9);
- c) abb.insertar(25);

Donde abb.limpiarUltimosInsertados (); Limpia el registro de los últimos elementos insertados.abb.insertar (7); → Inserta 7 en el árbol.cabb.insertar (9); → Inserta 9 en el árbol. abb.insertar (25); → Inserta 25 en el árbol. A continuación, la imagen del código:

```
//Insertar elementos al árbol
System.out.println("\n3.INSERCIÓN DE ELEMENTOS AL ARBOL");
abb.limpiarUltimosInsertados();
abb.insertar(7);
abb.insertar(9);
abb.insertar(25);
```

8. Ahora con el fin de dar claridad a los resultados, procedo a mostrar los últimos elementos insertados con abb.mostrarUltimosInsertados();, donde

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

este código muestra únicamente los últimos valores agregados (7, 9, 25) que como vimos en su anterioridad fueron los elementos que agregue.

```
// Mostrar únicamente los últimos elementos insertados
abb.mostrarUltimosInsertados();
```

9. Y con el fin de dar la totalidad de los elementos luego de la inserción entonces procedo a mostrar todos los elementos del árbol luego de esta y lo hago con los siguientes códigos "System.out.println("\n***Impresion De Todos Los Elementos Del Arbol Incluyendo Los Insertados***"); que me imprime el título y abb.mostrarElementos();", donde se imprimen los elementos del árbol después de las nuevas inserciones.

```
//Mostrar los elementos del árbol con la inserción
System.out.println("\n***Impresion De Todos Los Elementos Del Arbol Incluyendo Los Insertados***");
abb.mostrarElementos();
```

10. Ahora como en el laboratorio nos piden efectuar un proceso de eliminación considero lo primero y para dar claridad es mostrar estado del árbol antes de eliminar y para todos los recorridos, y lo hago con los siguientes códigos:
- a) System.out.println("\n4. PROCESO DE BORRADO DE ELEMENTOS");
 - b) System.out.println("Estado actual del arbol antes de eliminar:");
 - c) System.out.print("Recorrido Inorden antes de eliminar: "); abb.inOrden();
 - d) System.out.print("Recorrido Preorden antes de eliminar: ");
abb.preOrden();
 - e) System.out.print("Recorrido Posorden antes de eliminar: ");
abb.posOrden();

Donde se imprimen los recorridos en diferentes órdenes antes de eliminar un elemento.

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

```
// Estado del árbol antes de eliminar
System.out.println("\n4.PROCESO DE BORRADO DE ELEMENTOS");
System.out.println("Estado actual del arbol antes de eliminar:");
System.out.print("Recorrido Inorden antes de eliminar: "); abb.inOrden();
System.out.print("Recorrido Preorden antes de eliminar: "); abb.preOrden();
System.out.print("Recorrido Posorden antes de eliminar: "); abb.posOrden();
```

11. Ahora que ya he dado claridad ahora si procedo a eliminar un nodo, y en este caso elegí el nodo con el valor 10 y ejecuto el proceso con el siguiente código "abb.eliminar(10);", donde su explicación es la siguiente:

- Si 10 no tiene hijos, se elimina directamente.
- Si 10 tiene un hijo, el hijo lo reemplaza.
- Si 10 tiene dos hijos, se reemplaza con el sucesor inorden (el menor valor del subárbol derecho). A continuación la imagen:

```
//En este caso se va a proceder a eliminar el valor 10,
//Si se requiere eliminar otro valor cambiar en la línea de abajo 10 por el valor requerido
abb.eliminar(10);
```

12. Ahora y para dar claridad procedo a mostrar elementos después de la eliminación con los siguientes código "System.out.println("***Impresion De Todos Los Elementos Del Arbol Despues De Eliminar***"); que me imprime el titulo y abb.mostrarElementos(); donde se imprime el árbol luego de eliminar el nodo 10.

```
System.out.println("***Impresion De Todos Los Elementos Del Arbol Despues De Eliminar***");
abb.mostrarElementos();
```

13. Y finalmente procedo a dar claridad sobre el estado del árbol después de la eliminación donde se imprimen los recorridos nuevamente para ver cómo cambió la estructura del árbol tras eliminar 10. Y este proceso lo ejecuto con los siguientes códigos:

- System.out.println("\nEstado actual del árbol después de eliminar:");

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

- b) `System.out.print("Recorrido Inorden después de eliminar: ");`
`abb.inOrden();`
- c) `System.out.print("Recorrido Preorden después de eliminar: ");`
`abb.preOrden();`
- d) `System.out.print("Recorrido Posorden después de eliminar: ");`
`abb.posOrden();`

```
// Estado del árbol después de eliminar
System.out.println("\nEstado actual del arbol despues de eliminar:");
System.out.print("Recorrido Inorden despues de eliminar: "); abb.inOrden();
System.out.print("Recorrido Preorden despues de eliminar: "); abb.preOrden();
System.out.print("Recorrido Posorden despues de eliminar: "); abb.posOrden();
```

Y con este último análisis doy por finalizada la explicación de la clase main de nombre “**SegundoLabEstrDatosABB**”.

Desarrollo Clase PruebaArbolABB

Ahora con base en la actividad a desarrollar se ha pedido el desarrollo de esta clase, la cual le coloque el nombre de **PruebaArbolABB.java** la cual es un programa en Java que se encarga de probar las funcionalidades del Árbol Binario de Búsqueda (ABB) desarrollado, asegurando que las operaciones de inserción, eliminación, recorrido y búsqueda, entre otros., funcionen correctamente.

Ahora considero importante precisar que no voy a detallar completamente la explicación de todo el código por cuidar la extensión del trabajo y para evitar desgaste ya que el código se puede revisar perfectamente en NetBeans y se asemeja mucho al desarrollo de la clase main de nombre “**SegundoLabEstrDatosABB**”, pero si voy a explicar toda su funcionalidad mostrando todos los pasos que sigue esta funcionalidad del código. A continuación, la lista de pasos:

1. Lo primero es que se muestra un mensaje indicando el inicio de las pruebas.

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

2. Luego se crea una instancia de la clase ArbolABB, que representa el árbol binario de búsqueda.
3. Se insertan los valores en el ABB, incluyendo duplicados.
4. Se imprime el estado del árbol tras la inserción.
5. Se limpia la lista de los últimos insertados.
6. Se insertan los nuevos valores y se muestran.
7. Se realizan e imprimen los recorridos Inorden, Preorden y Posorden.
8. Se elimina un nodo específico del árbol (en este caso, el nodo con valor 10).
9. Se vuelven a realizar e imprimir los recorridos para verificar la correcta eliminación.
10. Se imprime si cada valor está en el árbol o no.
11. Se muestra un mensaje indicando el final de las pruebas.

Y con este último párrafo finalizo la explicación de la funcionalidad del código de esta clase.

RESULTADOS DE CONSOLA

A continuación, procedo a mostrar la imagen de los resultados de la consola de NetBeans con base en este desarrollo para dar claridad y entendimiento al mismo:

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

```

run:

***LABORATORIO DE IMPLEMENTACION DE UN ARBOL BINARIO DE BUSQUEDA***

1.RECORRIDO EN PREORDEN, INORDEN Y POSORDEN (SIN IMPRIMIR)

2.IMPRESION DE LOS ELEMENTOS DEL ARBOL
Elementos en el arbol[10, 10] [5, 5] [15] [20]

3.INSERCIÓN DE ELEMENTOS AL ARBOL
Ultimos elementos insertados: [7, 9, 25]

***Impresion De Todos Los Elementos Del Arbol Incluyendo Los Insertados***
Elementos en el arbol[10, 10] [5, 5] [7] [9] [15] [20] [25]

4.PROCESO DE BORRADO DE ELEMENTOS
Estado actual del arbol antes de eliminar:
Recorrido Inorden antes de eliminar: [5, 5] [7] [9] [10, 10] [15] [20] [25]
Recorrido Preorden antes de eliminar: [10, 10] [5, 5] [7] [9] [15] [20] [25]
Recorrido Posorden antes de eliminar: [9] [7] [5, 5] [25] [20] [15] [10, 10]

***Eliminando el dato***: 10

***Impresion De Todos Los Elementos Del Arbol Despues De Eliminar***
Elementos en el arbol[10] [5, 5] [7] [9] [15] [20] [25]

Estado actual del arbol despues de eliminar:
Recorrido Inorden despues de eliminar: [5, 5] [7] [9] [10] [15] [20] [25]
Recorrido Preorden despues de eliminar: [10] [5, 5] [7] [9] [15] [20] [25]
Recorrido Posorden despues de eliminar: [9] [7] [5, 5] [25] [20] [15] [10]
BUILD SUCCESSFUL (total time: 1 second)

```

CONCLUSIÓN

En el desarrollo de esta actividad, llevé a cabo la implementación y prueba de un Árbol Binario de Búsqueda (ABB), aplicando conceptos fundamentales de estructuras de datos en Java. A través de este proceso, realicé un análisis detallado del comportamiento del ABB en cuanto a la inserción, eliminación, recorridos y búsqueda de elementos, validando su correcto funcionamiento.

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

Inicialmente, estructuré el árbol y verifiqué su capacidad para almacenar y organizar los datos de manera eficiente, asegurando que cada nodo se posicionara adecuadamente dentro de la jerarquía. Posteriormente, ejecuté múltiples pruebas de inserción, incluyendo la gestión de valores repetidos, lo que me permitió evaluar cómo se manejaban dentro de la estructura.

Además, implementé y ejecuté los recorridos Inorden, Preorden y Posorden, con el objetivo de analizar la disposición de los elementos en el árbol y corroborar la correcta aplicación de estas estrategias. También realicé la eliminación de un nodo clave (10), se aclara que se puede incluir uno o varios datos para su eliminación, examinando cómo el ABB se ajusta tras esta depuración y verificando que su estructura permaneciera íntegra.

Por último, validé la funcionalidad de búsqueda dentro del árbol, asegurándome de que el algoritmo identifica correctamente la presencia o ausencia de valores específicos. Considero importante indicar que estos experimentos me permitieron evaluar la eficiencia del ABB y su aplicabilidad en la organización de datos.

En conclusión, esta actividad me permitió profundizar en el diseño e implementación de estructuras de datos avanzadas, fortaleciendo mis habilidades en programación orientada a objetos y optimización de algoritmos. Y es por esto por lo que considero que a correcta ejecución de cada una de las pruebas demuestra la solidez de la implementación y la aplicabilidad de los Árboles Binarios de Búsqueda en el procesamiento eficiente de información por lo que doy las gracias al profesor por la enseñanza impartida y recibida de mi parte.

BIBLIOGRAFÍA

Asignatura	Datos del alumno	Fecha
Estructura De Datos	Apellidos: De Mendoza	25/03/2024
	Nombre: Alejandro	

A continuación, la bibliografía implementada para el desarrollo de este laboratorio:

- ✓ Tema 5. Estructuras de datos jerárquicas. Tema 6. Distribuciones De Variables Continuas: Normal, T De Student. Tema 7. Montículos y cola de prioridad.
- ✓ Clases virtuales con el profesor Ing. Edwin Eduardo Millán Rojas.
- ✓ Material de estudio del aula
- ✓ ChatGPT para complementar el desarrollo del código en la búsqueda y solución de errores de compilación.