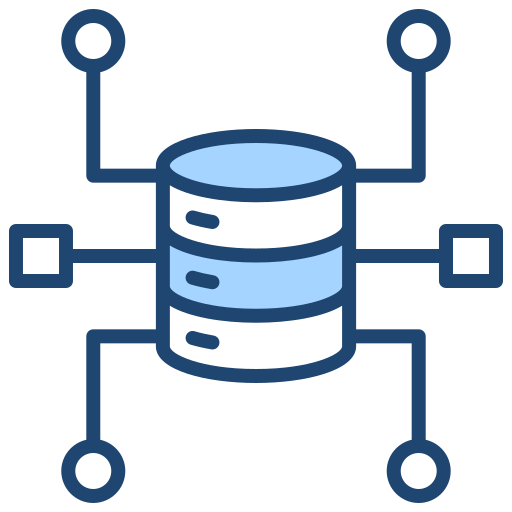
**ACTIVIDAD 4**

*Estructuras de Datos*

Docente: Adalberto Emmanuel Rojas Perea

Alumno: Osbaldo Damián Rubio Morales



**Objetivo:** Comprender la estructura y funcionamiento de los árboles binarios, implementar y manipular árboles binarios en Java asi como aplicar los conceptos aprendidos en un caso práctico: gestión de empleados.

El proyecto consiste en implementar un árbol binario de búsqueda como sistema para la gestión de empleados. Cada empleado se representa mediante un nodo del árbol, el cual contiene un identificador único (ID) y el nombre del empleado.

**Funcionamiento del árbol binario de búsqueda**

Un árbol binario de búsqueda tiene la siguiente propiedad:

* Todos los nodos del subárbol izquierdo de un nodo contienen IDs menores que el ID del nodo padre.
* Todos los nodos del subárbol derecho contienen IDs mayores que el ID del nodo padre.

Gracias a esta organización, las operaciones de búsqueda, inserción y eliminación se realizan de manera mucho más eficiente que en una lista tradicional.

**Operaciones implementadas**

1. **Inserción de empleados**

Los empleados se insertan en el árbol de acuerdo con su ID. Si el ID es menor al del nodo actual, se coloca en el subárbol izquierdo; si es mayor, se coloca en el subárbol derecho. Esto mantiene el orden del árbol.

1. **Búsqueda de empleados**

El árbol permite localizar rápidamente un empleado a partir de su ID. La búsqueda no requiere recorrer todos los elementos, sino que se guía según si el ID buscado es mayor o menor que el nodo actual.

1. **Eliminación de empleados**

Para eliminar un empleado se consideran tres casos:

* + Si el nodo no tiene hijos, simplemente se elimina.
  + Si tiene un solo hijo, el nodo se reemplaza por su hijo.
  + Si tiene dos hijos, el nodo se reemplaza por el sucesor más pequeño de su subárbol derecho, garantizando que el árbol mantenga su orden.

1. **Recorridos del árbol**

Se implementaron tres tipos de recorridos:

* + **Inorden:** muestra los empleados en orden ascendente de IDs.
  + **Preorden:** primero muestra el nodo raíz y luego los subárboles.
  + **Postorden:** recorre primero los subárboles y al final el nodo raíz.

**Caso práctico: gestión de empleados**

Se probó el sistema con una lista de empleados, cada uno con un ID único.

* Al insertar todos los empleados en el árbol, se puede observar cómo la estructura queda ordenada automáticamente según los IDs.
* Se realizaron búsquedas de empleados, mostrando resultados inmediatos para IDs existentes y mensajes de no encontrado para IDs inexistentes.
* Se eliminó un empleado y el árbol se reorganizó de forma correcta sin perder el orden.

**Ventajas frente a la búsqueda secuencial**

En una lista simple, la búsqueda de un empleado requiere revisar uno por uno hasta encontrarlo, lo cual en el peor de los casos toma tiempo lineal.

En cambio, en un árbol binario de búsqueda, las operaciones de inserción, búsqueda y eliminación se realizan en promedio en tiempo logaritmico.

Esto significa que con un número grande de empleados (por ejemplo, 1000), la diferencia de eficiencia es muy significativa:

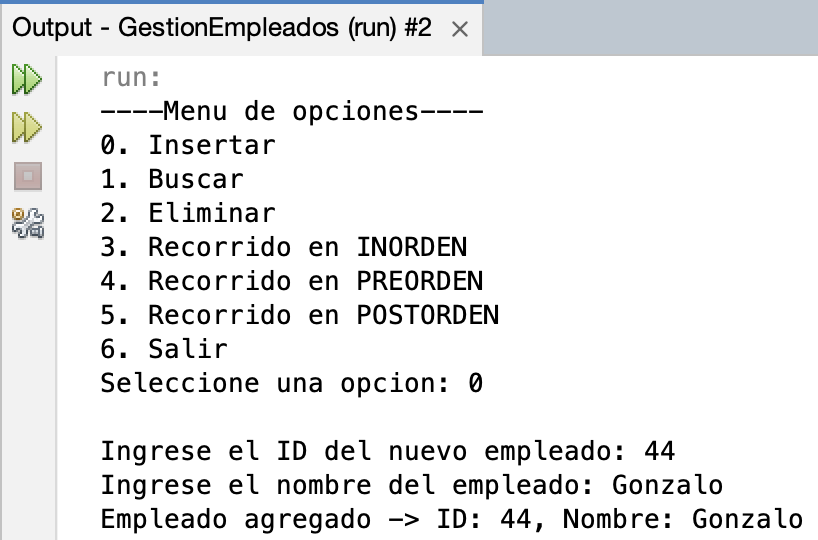
* Con búsqueda secuencial podrían ser necesarias hasta 1000 comparaciones.
* Con un árbol binario bastarían alrededor de 10 comparaciones.

**Conclusión**

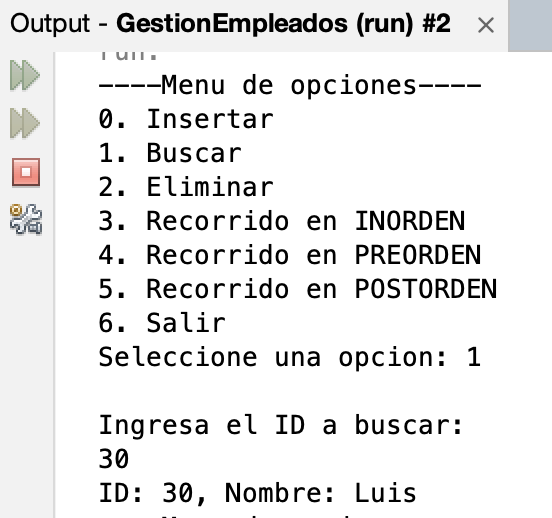
El uso de un árbol binario de búsqueda permite una gestión más rápida y eficiente de empleados, especialmente cuando se trabaja con grandes cantidades de datos. La estructura garantiza que las operaciones de inserción, búsqueda y eliminación se realicen de manera ordenada y con menor costo computacional en comparación con métodos tradicionales como la búsqueda secuencial en listas.

Evidencias de la ejecución del código:

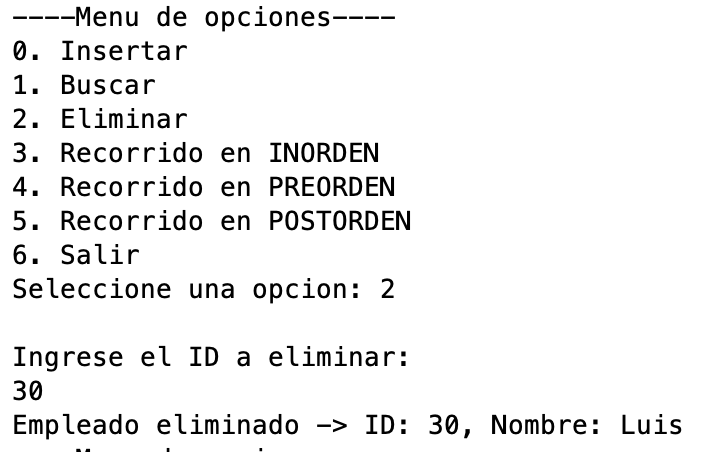
Insertar:



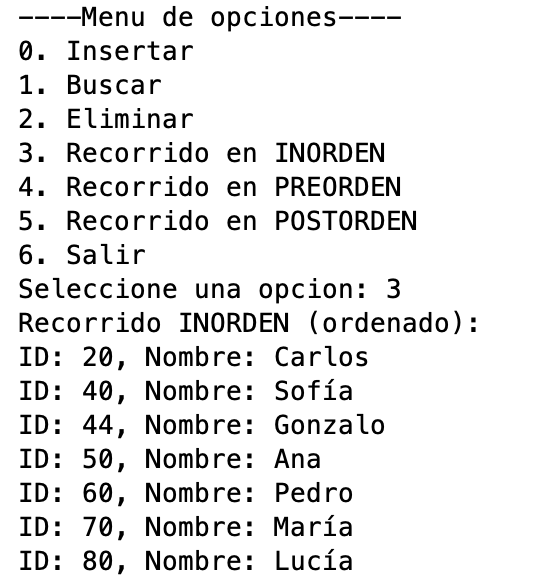
Buscar:



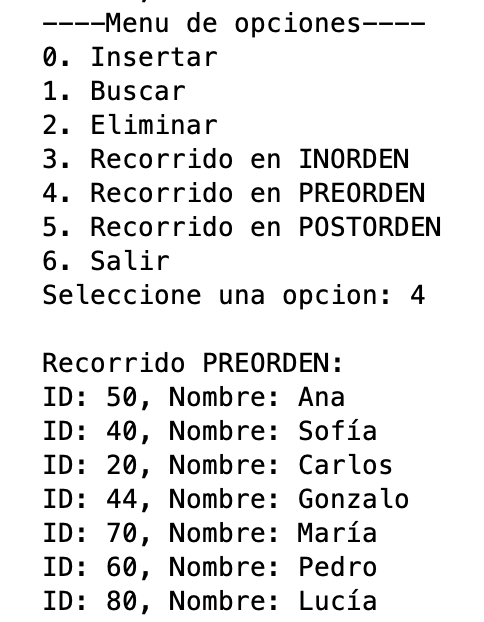
Eliminar:



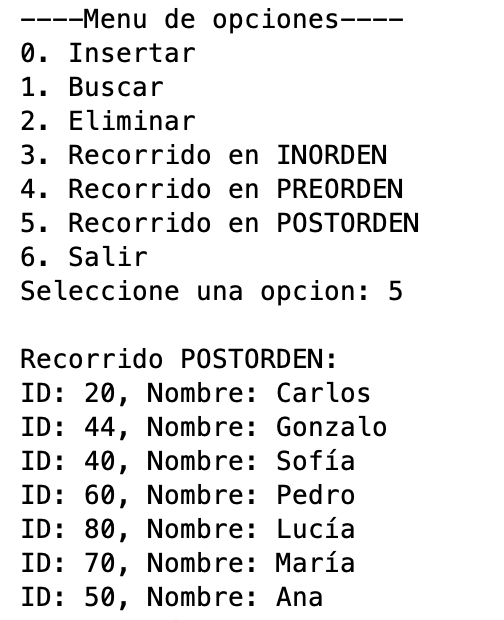
Recorrido en INORDEN:



Recorrido en PREORDEN:



Recorrido en POSTORDEN:



Reflexión:

Los árboles binarios son estructuras de datos fundamentales en programación porque permiten organizar la información de forma jerárquica, facilitando operaciones como búsqueda, inserción y eliminación de manera eficiente. Su importancia radica en que reducen el tiempo de acceso a los datos en comparación con estructuras lineales como listas o arreglos.

En cuanto a sus aplicaciones, se utilizan en:

* **Sistemas de bases de datos**, para optimizar consultas mediante árboles balanceados.
* **Algoritmos de búsqueda y ordenamiento**, como los árboles binarios de búsqueda.
* **Inteligencia artificial**, en estructuras como los árboles de decisión.

En resumen, los árboles binarios no solo ayudan a trabajar con datos de manera más eficiente, sino que también son la base de estructuras más complejas que se aplican en problemas reales de software.