Universidad Autónoma de Zacatecas

Unidad Académica de Ingeniería Eléctrica

Programa Académico de Ingeniería de Software.

Nombre de la Práctica Árboles.

Numero de Práctica 26.

Nombre de la carrera Ingeniería de Software.

Nombre de la materia Lab. Estructuras de Datos.

Nombre del alumno Jesús Manuel Juárez Pasillas.

Nombre del docente Aldonso Becerra Sánchez.

Fecha: 27/10/2021.

Práctica 26: Árboles.

Introducción:

Los arboles binarios nos permiten almacenar datos en donde cada nodo puede ser una subraiz o una hoja del árbol, y donde cada subraiz puede tener máximo 2 hijos (izquierdo y derecho).

Como los arboles son naturalmente recursivos es muy confuso hacer los métodos sin esta característica.

Desarrollo:

Para realizar esta practica se pide realizar 3 puntos sobre los arboles binarios. Para esto los métodos que se van a estar realizando serán dentro de la clase ArbolBinario ya que de esta heredan varias clases las cuales tendrán el funcionamiento agregado.

1. Se pide realizar un método **no recursivo** el cual imprima el árbol en forma inOrden:

   ```
   • Imprime el árbol en inOrden sin utilizar la recursión.
     public void inOrdenNoRec().
   ```

   Como este método no es recursivo no requirió de un método adicional el cual lo llamara, además que sin la recursión tendremos que hacer el recorrido como si se hiciera con recursión ya que tendremos que utilizar una pila para realizar esto donde meteremos los nodos y en el orden necesario para que la salida tenga el formato de inOrden.

2. En el segundo paso se pide realizar el método con el cual se haga el recorrido de amplitud, este método se hará con forme al algoritmo descrito en la práctica.

   ```
   • Imprime el árbol haciendo el recorrido por amplitud.
     public void amplitud().
   ```

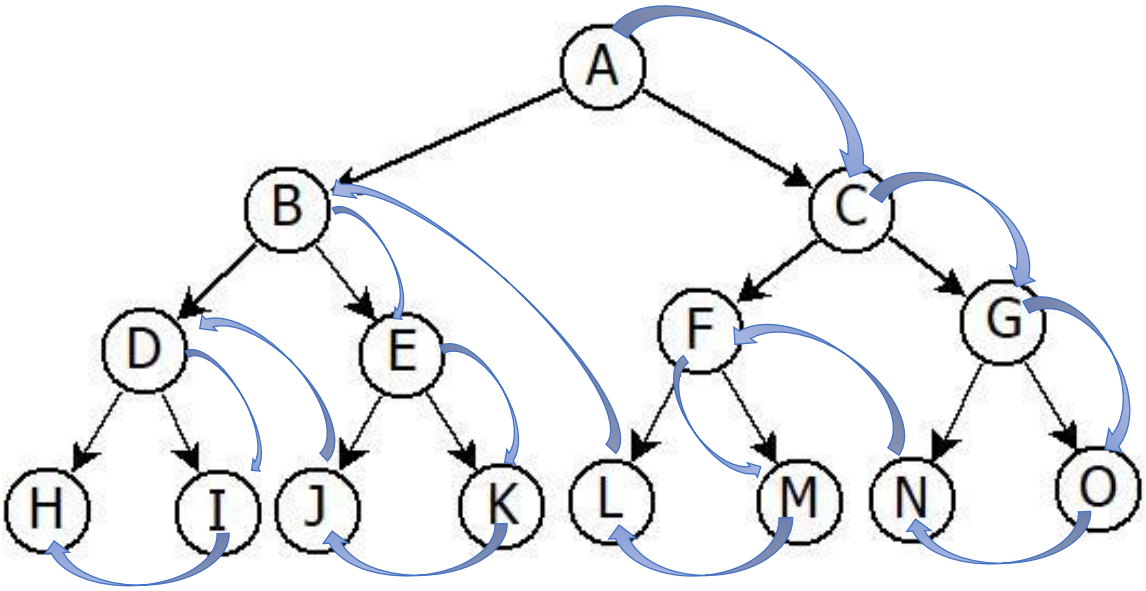
   Este método lo que hace es recorrer el árbol e imprimir los nodos por nivel, ya que primero imprime todos los nodos del primer nivel, luego los del segundo, y así hasta llegar al último nivel. Este método tampoco es recursivo.

3. En el último paso se pide realizar y explicar el método anterior (amplitud), pero en vez de que el método utilice una cola, que utilice una pila.

   ```
   • Imprime el árbol utilizando el recorrido por amplitud utilizando una pila en vez
     de una cola.
     public void amplitudPila().
   ```

El método en si tiene las mismas instrucciones que el método de amplitud, pero con la diferencia que se utiliza una pila en vez de una cola, esto ocasiona que el resultado sea completamente distinto al del método de amplitud que utiliza la cola.

Explicación del funcionamiento:



Recorrido en amplitud utilizando pila: **A, C, G, O, N, F, M, L, B, E, K, J, D, I, H.**

Al utilizar una pila en este recorrido nos da como resultado que recorre el árbol en forma:

Primero recorre la raíz, luego el hijo derecho y al final el hijo izquierdo. Como en este ejemplo varios hijos son subraiz (ósea que tienen hijos) esto vuelven a aplicar el mismo patrón que sería raíz, derecha y luego izquierda, y así se va sucesivamente.

Nota: Toda la documentación del proyecto esta agregada en la carpeta “doc” dentro de la carpeta del proyecto (“edylab_2021_26/doc”).

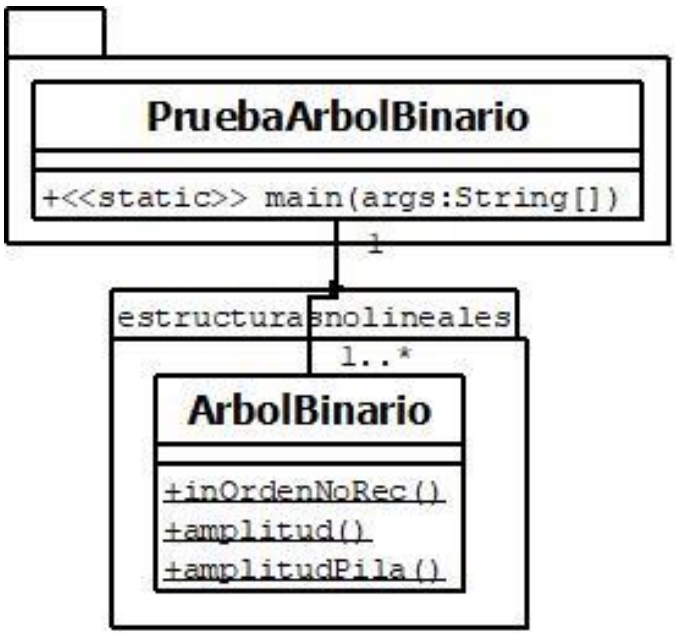
Capturas del programa funcionando:

Para hacer la prueba se introdujo el árbol que venía en la práctica.

```
InOrden con recursion:
H D I B J E K A L F M C N G O
InOrden sin recursion:
H D I B J E K A L F M C N G O
Amplitud:
A B C D E F G H I J K L M N O
Amplitud utilizando una pila:
A C G O N F M L B E K J D I H
```

Código agregado:

Para esta práctica solo se agregaron 3 métodos nuevos a la clase ArbolBinario. La prueba fue echa dentro de la clase ya existente PruebaArbolBinario donde solo se comento las pruebas anteriores y se agregaron las líneas para invocar estos métodos.



Pre-evaluación:

Pre-Evaluación para prácticas de Laboratorio de Estructuras de Datos	PRE-EVALUACIÓN DEL ALUMNO
CUMPLE CON LA FUNCIONALIDAD SOLICITADA.	Sí
DISPONE DE CÓDIGO AUTO-DOCUMENTADO.	Sí
DISPONE DE CÓDIGO DOCUMENTADO A NIVEL DE CLASE Y MÉTODO.	Sí
DISPONE DE INDENTACIÓN CORRECTA.	Sí
CUMPLE LA POO.	Sí
DISPONE DE UNA FORMA FÁCIL DE UTILIZAR EL PROGRAMA PARA EL USUARIO.	Sí
DISPONE DE UN REPORTE CON FORMATO IDC.	Sí
LA INFORMACIÓN DEL REPORTE ESTÁ LIBRE DE ERRORES DE ORTOGRAFÍA.	Sí
SE ENTREGÓ EN TIEMPO Y FORMA LA PRÁCTICA.	Sí
INCLUYE LA DOCUMENTACIÓN GENERADA CON JAVADOC.	Sí
INCLUYE EL CÓDIGO AGREGADO EN FORMATO UML.	Sí
INCLUYE LAS CAPTURAS DE PANTALLA DEL PROGRAMA FUNCIONANDO.	Sí
LA PRÁCTICA ESTÁ TOTALMENTE REALIZADA (ESPECIFIQUE EL PORCENTAJE COMPLETADO).	100%
Observaciones:	

Conclusión:

Tener que recorrer un árbol sin hacer uso de la recursión, los métodos tienden a ser un poco mas complicados ya que se tiene que utilizar alguna pila o cola con la cual almacenar los nodos necesarios y en el orden necesario para obtener la salida que queremos.

El recorrido en amplitud es muy útil a la hora de realizar operaciones por nivel en el árbol binario.