Programación III
TEMA 4: Árboles binarios
Práctica nº 4 - A

Puede continuar trabajando en su proyecto Programacion3. Para esta práctica tendrá a disposición el archivo fuente de la implementación de Arbol Binario y su clase de test.

1. Analice la implementación en JAVA de la clase ArbolBinario brindada por la cátedra.

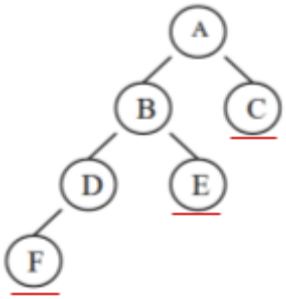
    a. Realice el diagrama del siguiente árbol. En particular indique **cómo quedan representados los nodos que son HOJA. ¿Cómo se representa el hijo izquierdo y el hijo derecho de una HOJA?**

```
ArbolBinario<Integer> arbolBinarioB=new ArbolBinario<Integer>(1);
ArbolBinario<Integer> hijoIzquierdoB=new ArbolBinario<Integer>(2);
hijoIzquierdoB.agregarHijoIzquierdo(new ArbolBinario<Integer>(3));
hijoIzquierdoB.agregarHijoDerecho(new ArbolBinario<Integer>(4));
ArbolBinario<Integer> hijoDerechoB=new ArbolBinario<Integer>(6);
hijoDerechoB.agregarHijoIzquierdo(new ArbolBinario<Integer>(7));
hijoDerechoB.agregarHijoDerecho(new ArbolBinario<Integer>(8));
arbolBinarioB.agregarHijoIzquierdo(hijoIzquierdoB);
arbolBinarioB.agregarHijoDerecho(hijoDerechoB);
```

2. Implemente en la clase ArbolBinario los métodos correspondientes a los 3 tipos de recorrido en profundidad: **PreOrder**, **InOrder** y **PostOrder**.

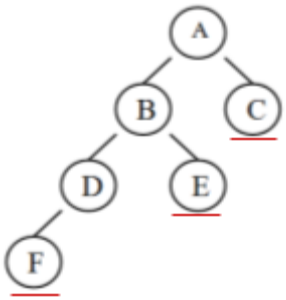
3. Agregue a la clase Arbol Binario los siguientes métodos
    a. **contarHojas():int** Devuelve la cantidad de hojas del árbol receptor.

    Ejemplo: para el árbol binario del gráfico, el resultado será 3.

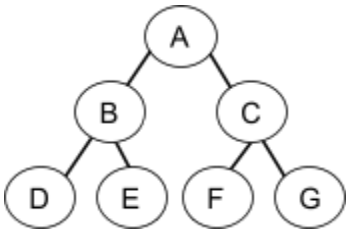


    b. **frontera():ListaGenerica<T>** Se define **frontera** de un árbol binario, a las hojas de un árbol binario recorridos de izquierda a derecha.
    NOTA: analice los 3 tipos de recorridos en profundidad de un ArbolBinario y elija el que corresponde.

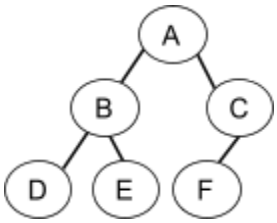
Ejemplo: para el árbol binario del gráfico, el resultado será una lista conteniendo los valores: **F, E, C**



c. **esLleno()**: **boolean.** Devuelve true si el árbol es lleno. Un árbol binario es lleno si tiene todas las hojas en el mismo nivel y además tiene todas las hojas posibles (es decir todos los nodos intermedios tienen dos hijos).

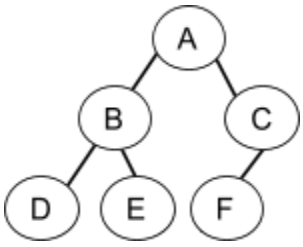


Éste árbol binario **ES** lleno

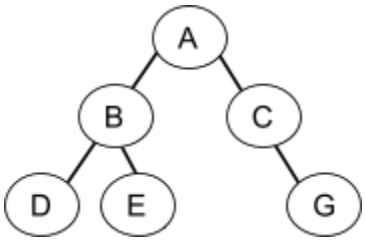


Éste árbol binario **NO ES** lleno

d. Indique cual sería la lógica de la solución (no implemente), para el siguiente método:
esCompleto(): boolean. Devuelve true si el árbol es completo. Un árbol binario de altura h es completo si es lleno hasta el nivel (h-1) y el nivel h se completa de izquierda a derecha.



Éste árbol binario **ES** completo



Éste árbol binario **NO ES** completo

4. JUnit (prueba de la implementación de los ejercicios anteriores)

a. Descargue del sitio https://github.com/junit-team/junit/releases el achivo .jar (librería recomendada version 4.7) correspondiente a JUnit ó descarguelo de la página de la cátedra.
b. Incluya dicha librería en su proyecto (cree una carpeta lib de modo que la librería quede dentro de su proyecto)

c. Ejecute la clase ArbolBinarioTest y verifique que los Test se ejecutan exitosamente.