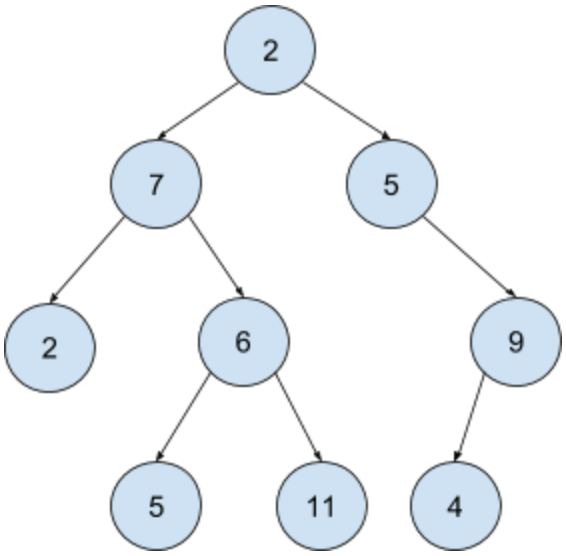
Viernes 28 de Abril - 13,30 horas

Sea la clase **NivelArbol** que tiene una variable de instancia arbol con un ArbolBinario <Integer> ya inicializado. Usted debe implementar el método **minEnNiveldeAB (int n): ArbolBinario <Integer>** que devuelve el subarbol **hoja** con menor valor en el nivel n del árbol. De haber más de uno devuelve el primero encontrado. Considere que n es un nivel válido del árbol. Sin embargo, puede suceder que no existan hojas en ese nivel, en ese caso, debe devolver null. Realice el recorrido por niveles.

Para el siguiente árbol:

- Si el nivel es 1, debe devolver null, puesto que 7 y 5 no son hojas.
- Si el nivel es 2, debe devolver 2, ya que es la única hoja.
- Si el nivel es 3, debe devolver 4, puesto que es el menor entre 5, 11 y 4.



Una posible solución (con un recorrido por niveles):

```
public class NivelArbol {
    private ArbolBinario<Integer> arbol;

    public ArbolBinario<Integer> minEnNiveldeAB (int n) {
        ColaGenerica<ArbolBinario<Integer>> cola = new ColaGenerica<>();
        int nivel = 0;
        ArbolBinario<Integer> menor = null;
        ArbolBinario<Integer> actual;

        cola.encolar(arbol);
        cola.encolar(null);
        while (!cola.esVacia()) {
            actual = cola.desencolar();
            if (actual == null) {
                nivel++;
                if (!cola.esVacia()) {
                    cola.encolar(null);
                }
            }
            else {
                if (nivel == n) {
                    if (actual.esHoja()) {
                        if (menor == null ||
                           menor.getDatoRaiz() < actual.getDatoRaiz()) {
                            menor = actual;
                        }
                    }
                }
                else {       /* arbol no vacio y nivel menor a n */
                    if (!actual.getHijoIzquierdo().esVacio()) {
                        cola.encolar(actual.getHijoIzquierdo());
                    }
                    if (!actual.getHijoDerecho().esVacio()) {
                        cola.encolar(actual.getHijoDerecho());
                    }
                }
            }
        }
        return menor;
    }
}
```

Otra posible solución (con recorrido en profundidad):

```
public class NivelArbol {

    private ArbolBinario<Integer> arbol;

    public ArbolBinario<Integer> minEnNiveldeAB (int n) {
        return this.minEnNiveldeAB(arbol, n);
    }

    private ArbolBinario<Integer> minEnNiveldeAB(ArbolBinario<Integer> arbol, int n) {
        if (arbol.esVacio()) {
            return null;
        }
        if (arbol.esHoja()) {
            if (n == 0) {
                return arbol;
            }
            else {
                return null;
            }
        }
        ArbolBinario<Integer> resultadoIzq=
this.minEnNiveldeAB(arbol.getHijoIzquierdo(), n-1);
        ArbolBinario<Integer> resultadoDer =
this.minEnNiveldeAB(arbol.getHijoDerecho(), n-1);
        if (resultadoIzq == null) {
            return resultadoDer;
        }
        if (resultadoDer == null) {
            return resultadoIzq;
        }
        if (resultadoDer.getDatoRaiz() < resultadoIzq.getDatoRaiz()) {
            return resultadoDer;
        }
        else {
            return resultadoIzq;
        }
    }
}
```