UNIDAD TEMÁTICA 4

Arboles Binarios I

TRABAJO DOMICILIARIO 1

Desarrolla los siguientes algoritmos (método de árbol y método de nodo):

- 1. Desarrolla un algoritmo que devuelva la altura de un árbol binario
 - altura(): devuelve un entero

```
public class TArbolBB<T> implements IArbolBB<T> {
    /**
    * Obtener la altura de un árbol.
    * @return altura del árbol.*/
    @Override
    public int obtenerAltura() {
        if (raiz != null) {
            return raiz.obtenerAltura();
        }
        return 0;
}
```

```
class TElementoAB<T> implements IElementoAB<T> {
    /**
    * Obtener la altura de un árbol.
    * Retorna la altura del árbol cuya raíz es el nodo actual.
    * @return altura del subárbol.*/
    @Override
    public int obtenerAltura() {
        int T1 = -1;
        int T2 = -1;
        if (this.hijoIzq != null) {
            T1 = this.hijoIzq.obtenerAltura();
        }
        if (this.hijoDer != null) {
            T2 = this.hijoDer.obtenerAltura();
        }
        if (T1 > T2) {
            return T1 + 1;
        } else {
            return T2 + 1;
        }
    }
}
```

- 2. Desarrolla un algoritmo que devuelva el tamaño de un árbol binario
 - tamaño(): devuelve un entero

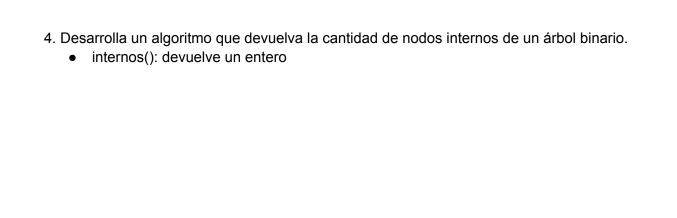
```
public class TArbolBB<T> implements IArbolBB<T> {
    /**
    * Obtener el tamaño de un árbol.
    * @return tamaño del árbol.
    */
    @Override
    public int obtenerTamanio() {
        if (raiz != null) {
            return raiz.obtenerTamanio();
        }
        return 0;
    }
```

```
class TElementoAB<T> implements IElementoAB<T> {
    /**
    * Obtener la tamaño de un árbol.
    * Retorna el tamaño del árbol cuya raíz es el nodo actual.
    * @return tamaño del subárbol.
    */
    @Override
    public int obtenerTamanio() {
        int T1 = 0;
        int T2 = 0;
        if (hijoIzq != null) {
            T1 += hijoIzq.obtenerTamanio();
        }
        if (hijoDer != null) {
            T2 += hijoDer.obtenerTamanio();
        }
        return T1 + T2 + 1;
    }
}
```

- 3. Desarrolla un algoritmo que devuelva la cantidad de hojas de un árbol binario
 - hojas(): devuelve un entero

```
public class TArbolBB<T> implements IArbolBB<T> {
    /**
    * Obtener la cantidad de hojas.
    *
    * @return int
    */
    @Override
    public int obtenerCantidadHojas() {
        if (esVacio()) {
            return 0;
        } else {
            return raiz.obtenerCantidadHojas();
        }
}
```

```
class TElementoAB<T> implements IElementoAB<T> {
    /**
    * Obtener la cantidad de hojas.
    *
    * @return int
    */
    @Override
    public int obtenerCantidadHojas() {
        if (hijoIzq == null && hijoDer == null) {
            return 1;
        } else if (hijoIzq == null && hijoDer != null) {
            return hijoDer.obtenerCantidadHojas();
        } else if (hijoIzq != null && hijoDer == null) {
            return hijoIzq.obtenerCantidadHojas();
        } else {
            return hijoIzq.obtenerCantidadHojas() +
        hijoDer.obtenerCantidadHojas();
    }
}
```



- 5. Desarrolla un algoritmo que devuelva la cantidad de nodos completos (ambos hijos no nulos) de árbol binario.
 - completos(): devuelve un entero

```
public class TArbolBB<T> implements IArbolBB<T> {
    /**
    * Retorna cantidades de nodos internos completos
    * @return cantidad de nodos internos completos*/
    @Override
    public int internosCompletos() {
        if (esVacio()) {
            return 0;
        } else {
            return raiz.internosCompletos();
        }
    }
}
```

```
class TElementoAB<T> implements IElementoAB<T> {
    /**
    * Retorna cantidades de nodos internos completos
    * @return cantidad de nodos internos completos*/
    @Override
    public int internosCompletos() {
        int completos = 0;
        if (this.hijoIzq == null && this.hijoDer == null) {
            return 0;
        }
        if (this.hijoIzq != null) {
                completos += this.hijoIzq.internosCompletos();
        }
        if (this.hijoDer != null) {
                 completos += this.hijoDer.internosCompletos();
        }
        if (this.hijoIzq != null && this.hijoDer != null) {
                 completos += 1;
        }
        return completos;
}
```

- 6. Desarrolla un algoritmo que devuelva la cantidad de nodos de un cierto nivel de un árbol binario.
 - enNivel (nivel de tipo entero): devuelve un entero