Miércoles 18 de Octubre - 8 horas

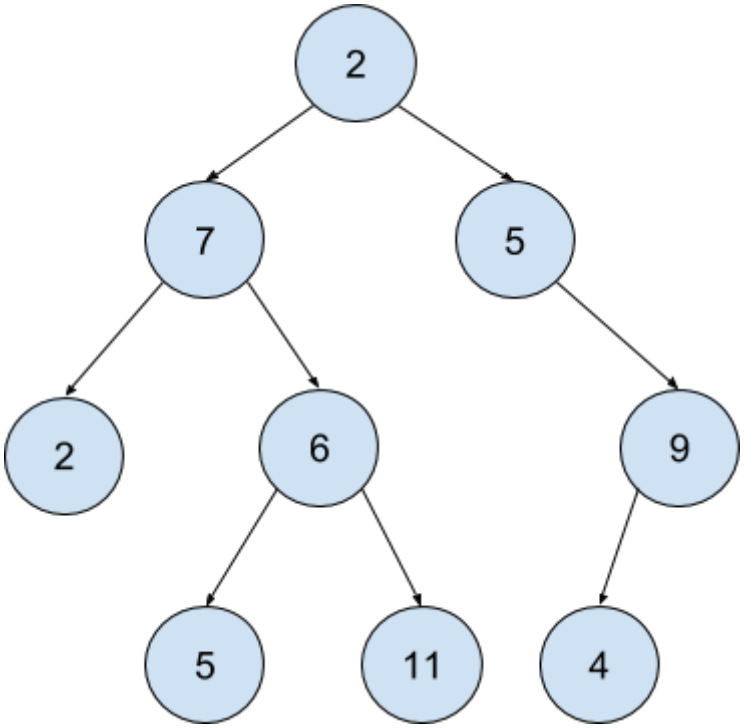
Dada una clase Java denominada **ContadorArbol** cuya función principal es proveer métodos de conteo sobre árboles binarios de Integer y que tiene como variable de instancia un **ArbolBinario<Integer>** denominado **arbol** implemente en dicha clase el método:

```
public int contadorHojasPares () {
    // retorna cantidad de hojas que contienen un número par en el árbol
    //…
}
```

Implemente su solución utilizando y recorriendo el árbol con un recorrido en postorden. Puede definir todos los métodos y variables auxiliares que considere. Todo aquel método que no esté definido en las prácticas debe ser implementado.

Por Ejemplo, para el siguiente árbol

- **contadorHojasPares() devuelve 2**



Una posible solución:

```
package tp03;

public class ContadorArbol {

private ArbolBinario<Integer> arbol;

public ContadorArbol(ArbolBinario<Integer> a){
		this.arbol=a;
	}

public int contadorHojasPares(){
		if (!arbol.esVacio()){
			Resultado r=new Resultado();
			r.setCantidad(0);
			this.contadorAuxiliar(this.arbol, r);
			return r.getCantidad();
		}
		else
		{
			return 0;
		}
	}

private void contadorAuxiliar (ArbolBinario<Integer> a, Resultado r){

		if (!a.getHijoIzquierdo().esVacio()){
			contadorAuxiliar (a.getHijoIzquierdo(), r);}
		if (!a.getHijoDerecho().esVacio()){
			contadorAuxiliar (a.getHijoDerecho(),r);}
		if(a.esHoja()){
			if (a.getDatoRaiz()%2==0){
				r.setCantidad(r.getCantidad()+1);
	}}}}
```

Otra posible solución:

```
package tp03;

public class ContadorArbol {

	private ArbolBinario<Integer> arbol;

	public ContadorArbol(ArbolBinario<Integer> a){
		this.arbol=a;
	}

public int contadorHojasPares2(){
		if (!arbol.esVacio()){

			return this.contadorAuxiliar(this.arbol );
		}
		else
		{
			return 0;
		}
	}

private int contadorAuxiliar (ArbolBinario<Integer> a){
int contador = 0;
if (!a.getHijoIzquierdo().esVacio()){
			contador = contador + contadorAuxiliar
(a.getHijoIzquierdo());}
if (!a.getHijoDerecho().esVacio()){
			contador = contador + contadorAuxiliar
(a.getHijoDerecho());}
		if(a.esHoja()){
			if (a.getDatoRaiz()%2==0){
				contador++;
	}}
	return contador;}}
```

Un posible main:

```
public static void main(String[] args) {


        ArbolBinario<Integer> a= new ArbolBinario<Integer>(2);

        ArbolBinario<Integer> b = new ArbolBinario<Integer>(7);
        ArbolBinario<Integer> c = new ArbolBinario<Integer>(5);

        ArbolBinario<Integer> d = new ArbolBinario<Integer>(6);
        d.agregarHijoIzquierdo(new ArbolBinario<Integer>(5));
        d.agregarHijoDerecho(new ArbolBinario<Integer>(11));

        b.agregarHijoIzquierdo(new ArbolBinario<Integer>(2));
        b.agregarHijoDerecho(d);

        ArbolBinario<Integer> e = new ArbolBinario<Integer>(9);
        e.agregarHijoIzquierdo(new ArbolBinario<Integer>(4));


        c.agregarHijoDerecho(e);

        a.agregarHijoIzquierdo(b);
        a.agregarHijoDerecho(c);

        ContadorArbol contador = new ContadorArbol(a);
        System.out.println(contador.contadorHojasPares());
        System.out.println(contador.contadorHojasPares2());

        }
```

Aspectos importantes que se evalúan:

- Respetar el recorrido post orden.
- Cálculo correcto de ocurrencias.
- Usar correctamente el pasaje de parámetros si los usa (se aceptan objetos, primitivos, arreglos de 1 elemento, etc. siempre y cuando estén bien utilizados).
- Revisar aspectos de a donde apunta this, árbol, y las invocaciones a los métodos que respeten los parámetros y sus tipos.
- Métodos auxiliares, si no son los de la práctica deben estar implementados.
- Firma del método, nombre de la clase ContadorArbol y variable de instancia árbol era lo único que se debía respetar y no pueden alterar esa definición.