

## Ejercicio 4 – Árboles

Diseñe un algoritmo recursivo y eficiente que, dado un árbol n-ario de enteros, determine el camino del árbol (lista de enteros) desde la raíz a una hoja no vacía tal que no contenga ningún elemento par y que la suma de sus etiquetas sea mínima. En caso de que no exista ningún camino que cumpla la condición, el algoritmo debe devolver null.

**Tiempo estimado: 45 min.**

**Puntuación: 25%.**



## Solución 1

```
private static record Tupla(Integer suma, List<Integer> camino) {
    public static Tupla of(List<Integer> camino) {
        Integer suma = camino.stream().reduce(0, (x,y)-> x+y);
        return new Tupla(suma, camino);
    }
    public static Tupla of(Integer suma, List<Integer> camino) {
        return new Tupla(suma, camino);
    }
}

public static List<Integer> solucion_rekursiva(Tree<Integer> tree) {
    Tupla t = recursivo(tree);
    return t!=null?t.camino():null;
}

private static Tupla recursivo(Tree<Integer> tree){
    return switch(tree) {
        case TEmpty<Integer> t -> null;

        case TLeaf<Integer> t && t.label()%2==0 -> null;

        case TLeaf<Integer> t && t.label()%2!=0 -> {
            List<Integer> copia = new ArrayList<>();
            copia.add(t.label());
            yield Tupla.of(t.label(), copia);
        }

        case TNary<Integer> t && t.label()%2==0 -> null;

        case TNary<Integer> t && t.label()%2!=0-> {
            Tupla minCamino = t.elements().stream()
                .map(ch -> recursivo(ch))
                .filter(x -> x!=null)
                .min(Comparator.comparing(x -> x.suma()))
                .orElse(null);

            Tupla res = null;
            if (minCamino!=null) {
                minCamino.camino.add(0, t.label());
                res = Tupla.of(minCamino.suma() + t.label(),
                    minCamino.camino);
            }
            yield res;
        }
    };
};
```