TA_FINALSEMESTRE
EQ01

LENGUAJE NATURAL:
El método recorrerá media recursividad el árbol en búsqueda de la clave(palabra) pasada por parámetro.
Si encuentra dicha clave(palabra) finaliza la búsqueda y se realiza la siguiente operación, se extraen las páginas pares de cada clave analizada mientras se buscaba la correcta e insertando las páginas pares en una lista. En caso de no haber encontrado la clave(palabra) no se realiza operación ninguna, resultando en una lista vacía.

Nuestro método tomará de esta lista sólo las páginas pares y las agrega a la lista.
Esto lo hará con todos los objetos que estén en su recorrido de vuelta.

PRECONDICIONES:
- El árbol debe existir.

POSTCONDICIONES:
- El árbol ni sus elementos no deben alterarse
- El método retornará una lista vacía en caso de no encontrar la clave
- El método retornará una lista conteniendo las páginas pared que contienen cada clave del recorrido por el árbol.

<3
PSEUDOCÓDIGO:

TArbolBB.paresPalabra(String: clave): devuelve Lista<int>

COMIENZO
```
    Lista<Integer> soloPares <- new Lista()
    SI (esVacio()) HACER
        DEVOLVER soloPares
    SINO
        this.raiz.paresPalabras(clave, soloPares)  //Se agregan páginas pares.
        DEVOLVER soloPares
    FIN SI
```
FIN

TElementoAB.paresPalabra(String: clave, Lista: nuevaLista) : devuelve Boolean

COMIENZO

```
SI clave = this.etiqueta HACER //Se encontró la palabra.
    SI this.getDato() <> nulo HACER
        Lista<Integer> listaAux <- (Lista<Integer>) this.getDatos()
        Nodo<T> nodoActual <- this.getDato().getPrimero()
        WHILE nodoActual <> nulo HACER
            SI (nodoActual.getEtiqueta() % 2 = 0)
                nuevaLista.insertar(nodoActual.getEtiqueta())
            FIN SI
            nodoActual = nodoActual.getSiguiente()
        FIN WHILE
    FIN SI
    DEVOLVER TRUE
SI NO
  SI clave < this.etiqueta HACER //La clave buscada es menor
    SI this.hijoIzq <> nulo HACER
        SI (this.hijoIzq.paresPalabras(clave, nuevaLista)) //Si no falso
            SI this.getDato() <> nulo HACER
                Lista<Integer> listaAux <- (Lista<Integer>) this.getDatos()
                Nodo<T> nodoActual <- this.getDato().getPrimero()
                WHILE nodoActual <> nulo HACER
                    SI (nodoActual.getEtiqueta() % 2 = 0)
                        nuevaLista.insertar(nodoActual.getEtiqueta())
                    FIN SI
                    nodoActual = nodoActual.getSiguiente()
                FIN WHILE
            FIN SI
            DEVOLVER TRUE //Se puede seguir buscando
         FIN SI
     SI NO
        DEVOLVER FALSE //NO hay más palabras o se viene de no encontrarla
     FIN SI
SI NO //La clave buscada es mayor
    SI this.hijoDer <> nulo ENTONCES
        SI (this.hijoDer.paresPalabras(clave, nuevaLista))
            SI this.getDato() <> nulo HACER
                Lista<Integer> listaAux <- (Lista<Integer>) this.getDatos()
                Nodo<T> nodoActual <- this.getDato().getPrimero()
                WHILE nodoActual <> nulo HACER
                    SI (nodoActual.getEtiqueta() % 2 = 0)
                        nuevaLista.insertar(nodoActual.getEtiqueta())
                    FIN SI
```

```
                        nodoActual = nodoActual.getSiguiente()
                    FIN WHILE
                FIN SI
                DEVOLVER TRUE//Se puede seguir buscando

            FIN SI
        SI NO
            DEVOLVER falso ////NO hay más palabras o se viene de no encontrarla
        FIN SI
  FIN SI
FINAL
```