Práctica 3: Árboles binarios

Ejercicio 1: Implementar los métodos descritos a continuación. Para ello, se proporciona el prototipo de una clase (`BinaryTreeUtils`) de utilidades que los contendrá.

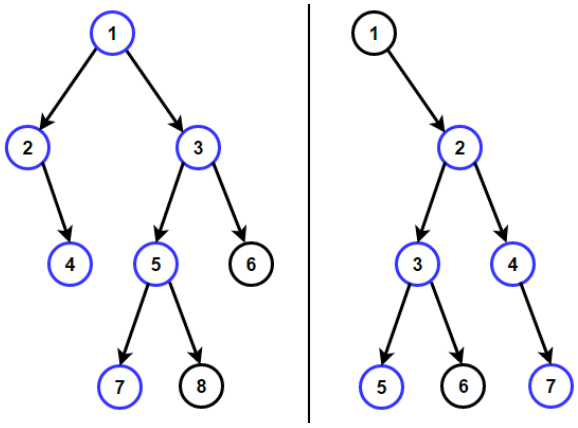
```
/**
  * Given a tree the method returns a new tree where all left children
  * become right children and vice versa
*/
public BinaryTree<E> mirror();

/**
  * Determines whether the element e is the tree or not
*/
public boolean contains (E e);

/**
  * Determines the level of a node in the tree (root is located at level
1)
*/
public int level(Position<E> p);
```

Ejercicio 2: Utilizando los ficheros disponibles en el Aula Virtual, implementar todos los métodos de las clases `ArrayBinaryTree` y `ABTPos` (implementación basada en array, donde cada nodo únicamente guarda información sobre su posición y su contenido) como estrategia alternativa de diseño de árboles binarios para tener una funcionalidad equivalente a `LinkedBinaryTree`. Para que el ejercicio se considere válido las dos clases (`LinkedBinaryTree` y `ArrayBinaryTree`) deben tener los mismos métodos y el mismo comportamiento.

Ejercicio 3: Dado un árbol binario, implementa un algoritmo eficiente que calcule su diámetro. El diámetro de un árbol binario se define como el número de nodos que hay en el camino más largo entre dos hojas cualesquiera. Por ejemplo, el árbol de la izquierda tiene diámetro 6, mientras el de la derecha tiene diámetro 5 (los nodos que forman parte del diámetro están resaltados en azul).



NOTA: Utiliza las clases proporcionadas con el material de esta práctica.