* Full Binary Tree
  + Each Node is either a leaf or an internal Node with exactly two non-empty children.
* Complete Binary Tree
  + If the height of the tree (number of levels) is d, then the first (top) d-1 levels are completely full. Namely, all Node positions on those levels are taken.
  + The bottom level has Nodes that are filled from left to right.
  + There is no gap on the bottom level.
* Height-Balanced Binary Tree (balanced)
  + An empty tree is height balanced.
  + A non-empty binary tree T is balanced if it satisfies 3 conditions:
    - Left sub-tree of T is balanced
    - Right sub-tree of T is balanced
    - The height difference of left sub-tree and right sub-tree is no more than one (<=1).
    - Logic to check a tree is balanced or not:
      * Check all sub-trees in the diagram, to see if all of sub-trees satisfy condition 3.
  + Boolean isBalanced(Node root) {
    - Int lh, rh;
    - If (root == null)
      * Return true
      * Lh = height(root.left)
      * Rh = height(root.right)
    - If (Math.abs(lh – Rh) <= 1 && isBalanced(root.left) && isBalanced(root.right))
      * Return true
    - Return false }
  + Int height (Node node) {
    - If (node == null)
      * Return 0
    - Return 1 + max(height(node.left), height(node.right))
* Side note: Look up B/B+ trees. R tree, R\* tree, prefix tree.