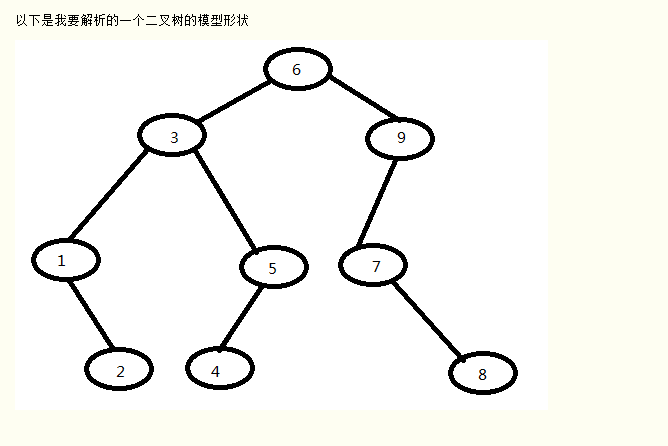
**使用队列，层次遍历二叉树。在上一层遍历完成后，下一层的所有节点已经放到队列中，此时队列中的元素个数就是下一层的宽度。以此类推，依次遍历下一层即可求出二叉树的最大宽度。**



**package** facehandjava.tree;  
  
**import** java.util.LinkedList;  
**import** java.util.Queue;  
  
**public class** WidthTree {  
 **public static** Node init() {*//注意必须逆序建立，先建立子节点，再逆序往上建立，因为非叶子结点会使用到下面的节点，而初始化是按顺序初始化的，不逆序建立会报错* Node J = **new** Node(8, **null**, **null**);  
 Node H = **new** Node(4, **null**, **null**);  
 Node G = **new** Node(2, **null**, **null**);  
 Node F = **new** Node(7, **null**, J);  
 Node E = **new** Node(5, H, **null**);  
 Node D = **new** Node(1, **null**, G);  
 Node C = **new** Node(9, F, **null**);  
 Node B = **new** Node(3, D, E);  
 Node A = **new** Node(6, B, C);  
 **return** A; *//返回根节点* }  
  
 **public static void** main(String[] args) {  
 Node root = WidthTree.*init*();  
 System.***out***.println(**"树的宽度"**);  
 **int** L = *WidthTree*(root);  
 System.***out***.println(L);  
 }  
  
 **public static int** WidthTree(Node node) {  
 Queue<Node> queue = **new** LinkedList<>();  
 queue.add(node);  
 **int** max = 0;  
 **while** (!queue.isEmpty()) {  
 **int** size = queue.size();  
 max = max > size ? max : size;  
 **while** (size > 0) {  
 Node temp = queue.remove();  
 size--;  
 **if** (temp.getLeftNode() != **null**) {  
 queue.add(temp.getLeftNode());  
 }  
 **if** (temp.getRightNode() != **null**) {  
 queue.add(temp.getRightNode());  
 }  
 }  
 }  
 **return** max;  
 }  
}