1. **import** java.util.LinkedList;
2. **import** java.util.Queue;
4. **public** **class** BinaryTree {
6. //Represent the node of binary tree
7. **public** **static** **class** Node{
8. **int** data;
9. Node left;
10. Node right;
12. **public** Node(**int** data){
13. //Assign data to the new node, set left and right children to null
14. **this**.data = data;
15. **this**.left = **null**;
16. **this**.right = **null**;
17. }
18. }
20. //Represent the root of binary tree
21. **public** Node root;
23. **public** BinaryTree(){
24. root = **null**;
25. }
27. //findMaximumWidth() will find out the maximum width of the given binary tree
28. **public** **int** findMaximumWidth() {
29. **int** maxWidth = 0;
31. //Variable nodesInLevel keep tracks of number of nodes in each level
32. **int** nodesInLevel = 0;
33. //queue will be used to keep track of nodes of tree level-wise
34. Queue<Node> queue = **new** LinkedList<Node>();
36. //Check if root is null, then width will be 0
37. **if**(root == **null**) {
38. System.out.println("Tree is empty");
39. **return** 0;
40. }
41. **else** {
42. //Add root node to queue as it represents the first level
43. queue.add(root);
45. **while**(queue.size() != 0) {
47. //Variable nodesInLevel will hold the size of queue i.e. number of elements in queue
48. nodesInLevel = queue.size();
49. //maxWidth will hold maximum width.
50. //If nodesInLevel is greater than maxWidth then, maxWidth will hold the value of nodesInLevel
51. maxWidth = Math.max(maxWidth, nodesInLevel);
53. //If variable nodesInLevel contains more than one node
54. //then, for each node, we'll add left and right child of the node to the queue
55. **while**(nodesInLevel > 0) {
56. Node current = queue.remove();
57. **if**(current.left != **null**)
58. queue.add(current.left);
59. **if**(current.right != **null**)
60. queue.add(current.right);
61. nodesInLevel--;
62. }
63. }
64. }
65. **return** maxWidth;
66. }
68. **public** **static** **void** main(String[] args) {
70. BinaryTree bt = **new** BinaryTree();
71. //Add nodes to the binary tree
72. bt.root = **new** Node(1);
73. bt.root.left = **new** Node(2);
74. bt.root.right = **new** Node(3);
75. bt.root.left.left = **new** Node(4);
76. bt.root.left.right = **new** Node(5);
77. bt.root.right.left = **new** Node(6);
78. bt.root.right.right = **new** Node(7);
79. bt.root.left.left.left = **new** Node(8);
81. //Display the maximum width of given tree
82. System.out.println("Maximum width of the binary tree: " + bt.findMaximumWidth());
83. }
84. }

OUTPUT:

