**import** java.util.LinkedList;

**import** java.util.Queue;

**public** **class** BinaryTree {

**public** **static** **class** Node{

**int** data;

        Node left;

        Node right;

**public** Node(**int** data){

**this**.data = data;

**this**.left = **null**;

**this**.right = **null**;

        }

      }

**public** Node root;

**public** BinaryTree(){

        root = **null**;

      }

**public** **int** findMaximumWidth() {

**int** maxWidth = 0;

**int** nodesInLevel = 0;

          Queue<Node> queue = **new** LinkedList<Node>();

**if**(root == **null**) {

              System.out.println("Tree is empty");

**return** 0;

          }

**else** {

              //Add root node to queue as it represents the first level

              queue.add(root);

**while**(queue.size() != 0) {

                  nodesInLevel = queue.size();

                  maxWidth = Math.max(maxWidth, nodesInLevel);

**while**(nodesInLevel > 0) {

                     Node current = queue.remove();

**if**(current.left != **null**)

                         queue.add(current.left);

**if**(current.right != **null**)

                         queue.add(current.right);

                     nodesInLevel--;

                  }

              }

          }

**return** maxWidth;

      }

**public** **static** **void** main(String[] args) {

          BinaryTree bt = **new** BinaryTree();

          //Add nodes to the binary tree

          bt.root = **new** Node(1);

          bt.root.left = **new** Node(2);

          bt.root.right = **new** Node(3);

          bt.root.left.left = **new** Node(4);

          bt.root.left.right = **new** Node(5);

          bt.root.right.left = **new** Node(6);

          bt.root.right.right = **new** Node(7);

          bt.root.left.left.left = **new** Node(8);

          System.out.println("Maximum width of the binary tree: " + bt.findMaximumWidth());

      }  }

