```java
package binarytree;

// Java program for printing vertical order of a given binary tree
import java.util.TreeMap;
import java.util.Vector;
import java.util.Map.Entry;

  class VerticalOrderBtree
{
    // Tree node
    static class Node
    {
        int key;
        Node left;
        Node right;

        // Constructor
        Node(int data)
        {
            key = data;
            left = null;
            right = null;
        }
    }

    // Utility function to store vertical order in map 'm'
    // 'hd' is horizontal distance of current node from root.
    // 'hd' is initially passed as 0
    static void getVerticalOrder(Node root, int hd,
                                    TreeMap<Integer,Vector<Integer>> m)
    {
        // Base case
        if(root == null)
            return;

        //get the vector list at 'hd'
        Vector<Integer> get = m.get(hd);

        // Store current node in map 'm'
        if(get == null)
        {
            get = new Vector<>();
            get.add(root.key);
        }
        else
            get.add(root.key);

        m.put(hd, get);

        // Store nodes in left subtree
        getVerticalOrder(root.left, hd-1, m);

        // Store nodes in right subtree
        getVerticalOrder(root.right, hd+1, m);
    }

    // The main function to print vertical order of a binary tree
    // with the given root
    static void printVerticalOrder(Node root)
    {
        // Create a map and store vertical order in map using
        // function getVerticalOrder()
        TreeMap<Integer,Vector<Integer>> m = new TreeMap<>();
        int hd =0;
        getVerticalOrder(root,hd,m);
        // Traverse the map and print nodes at every horigontal
        // distance (hd)
        for (Entry<Integer, Vector<Integer>> entry : m.entrySet())
        {
            System.out.println(entry.getValue());
        }
    }

    // Driver program to test above functions
    public static void main(String[] args) {

        // TO DO Auto-generated method stub
        Node root = new Node(1);
        root.left = new Node(2);
        root.right = new Node(3);
        root.left.left = new Node(4);
        root.left.right = new Node(5);
        root.right.left = new Node(6);
        root.right.right = new Node(7);
        root.right.left.right = new Node(8);
        root.right.right.right = new Node(9);
        System.out.println("Vertical Order traversal is");
        printVerticalOrder(root);
    }
}
// This code is contributed by Sumit Ghosh
```