```java
package binarytree;

// Java Program to print Bottom View of Binary Tree
import java.util.*;
import java.util.Map.Entry;

// Tree Node22 class
class Node22
{
    int data; //data of the Node22
    int hd; //horizontal distance of the Node22
    Node22 left, right; //left and right references

    // Constructor of tree Node22
    public Node22(int key)
    {
        data = key;
        hd = Integer.MAX_VALUE;
        left = right = null;
    }
}

//Tree class
class Tree
{
    Node22 root; //root Node22 of tree

    // Default constructor
    public Tree() {}

    // Parameterized tree constructor
    public Tree(Node22 Node22)
    {
        root = Node22;
    }

    // Method that prints the bottom view.
    public void bottomView()
    {
        if (root == null)
            return;

        // Initialize a variable 'hd' with 0 for the root element.
        int hd = 0;

        // TreeMap which stores key value pair sorted on key value
        Map<Integer, Integer> map = new TreeMap<>();

        // Queue to store tree Node22s in level order traversal
        Queue<Node22> queue = new LinkedList<Node22>();

        // Assign initialized horizontal distance value to root
        // Node22 and add it to the queue.
        root.hd = hd;
        queue.add(root);

        // Loop until the queue is empty (standard level order loop)
        while (!queue.isEmpty())
        {
            Node22 temp = queue.remove();

            // Extract the horizontal distance value from the
            // dequeued tree Node22.
            hd = temp.hd;

            // Put the dequeued tree Node22 to TreeMap having key
            // as horizontal distance. Every time we find a Node22
            // having same horizontal distance we need to replace
            // the data in the map.
            map.put(hd, temp.data);

            // If the dequeued Node22 has a left child add it to the
            // queue with a horizontal distance hd-1.
            if (temp.left != null)
            {
                temp.left.hd = hd-1;
                queue.add(temp.left);
            }
            // If the dequeued Node22 has a right child add it to the
            // queue with a horizontal distance hd+1.
            if (temp.right != null)
            {
                temp.right.hd = hd+1;
                queue.add(temp.right);
            }
        }

        // Extract the entries of map into a set to traverse
        // an iterator over that.
        Set<Entry<Integer, Integer>> set = map.entrySet();

        // Make an iterator
        Iterator<Entry<Integer, Integer>> iterator = set.iterator();

        // Traverse the map elements using the iterator.
        while (iterator.hasNext())
        {
            Map.Entry<Integer, Integer> me = iterator.next();
            System.out.print(me.getValue()+" ");
        }
    }
}

// Main driver class
  class BottomView
{
    public static void main(String[] args)
    {
        Node22 root = new Node22(20);
        root.left = new Node22(8);
        root.right = new Node22(22);
        root.left.left = new Node22(5);
        root.left.right = new Node22(3);
        root.right.left = new Node22(4);
        root.right.right = new Node22(25);
        root.left.right.left = new Node22(10);
        root.left.right.right = new Node22(14);
        Tree tree = new Tree(root);
        System.out.println("Bottom view of the given binary tree:");
        tree.bottomView();
    }
}
```