Java Programs: BST Operations

1. Build BST from Array & Level Order Traversal

```
import java.util.*;

class TreeNode {
    int val;
    TreeNode left, right;

    TreeNode(int val) {
        this.val = val;
        left = right = null;
    }
}

public class BSTLevelOrder {

    static TreeNode insert(TreeNode root, int val) {
        if (root == null) {
            return new TreeNode(val);
        }
        if (val < root.val) {
            root.left = insert(root.left, val);
        } else {
            root.right = insert(root.right, val);
        }
        return root;
    }

    static TreeNode buildBST(int[] arr) {
        TreeNode root = null;
        for (int val : arr) {
            root = insert(root, val);
        }
        return root;
    }

    static void levelOrder(TreeNode root) {
        if (root == null) return;

        Queue<TreeNode> q = new LinkedList<>();
        q.add(root);

        while (!q.isEmpty()) {
            TreeNode current = q.poll();
            System.out.print(current.val + " ");

            if (current.left != null) q.add(current.left);
            if (current.right != null) q.add(current.right);
```

```
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of elements: ");
        int n = sc.nextInt();
        int[] arr = new int[n];

        System.out.println("Enter array elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        TreeNode root = buildBST(arr);
        System.out.println("Level Order Traversal of BST:");
        levelOrder(root);
    }
}
```

2. Check if a Binary Tree is a Valid BST

```
class TreeNode {
    int val;
    TreeNode left, right;

    TreeNode(int val) {
        this.val = val;
        left = right = null;
    }
}

public class CheckValidBST {

    static boolean isValidBST(TreeNode root) {
        return isValidBST(root, Long.MIN_VALUE, Long.MAX_VALUE);
    }

    static boolean isValidBST(TreeNode node, long min, long max) {
        if (node == null) return true;

        if (node.val <= min || node.val >= max) return false;

        return isValidBST(node.left, min, node.val) &&
               isValidBST(node.right, node.val, max);
    }
```

Java Programs: BST Operations

```
    public static void main(String[] args) {
        TreeNode root = new TreeNode(10);
        root.left = new TreeNode(5);
        root.right = new TreeNode(15);
        root.right.left = new TreeNode(12);
        root.right.right = new TreeNode(20);

        if (isValidBST(root)) {
            System.out.println("The tree is a valid BST.");
        } else {
            System.out.println("The tree is NOT a valid BST.");
        }
    }
}
```