Here's a Java program that performs a \*\*preorder traversal\*\* of a binary tree, printing the node values in a single line, space-separated:

Consider java program. Input format: our test code passes the root node of a binary tree to the preorder function. constraints: nodez in the tree. Output format: Print the trees preorder traversal as a single line of space seperated values

### Java Code

```java

class TreeNode {

int val;

TreeNode left, right;

// Constructor

public TreeNode(int val) {

this.val = val;

this.left = this.right = null;

}

}

public class BinaryTreePreorderTraversal {

// Preorder traversal: root -> left -> right

public static void preorder(TreeNode root) {

if (root == null) {

return;

}

// Print the current node's value

System.out.print(root.val + " ");

// Recursively traverse the left subtree

preorder(root.left);

// Recursively traverse the right subtree

preorder(root.right);

}

public static void main(String[] args) {

// Constructing the binary tree for testing

TreeNode root = new TreeNode(1);

root.left = new TreeNode(2);

root.right = new TreeNode(3);

root.left.left = new TreeNode(4);

root.left.right = new TreeNode(5);

root.right.left = new TreeNode(6);

root.right.right = new TreeNode(7);

// Calling the preorder traversal function

preorder(root);

}

}

```

### Explanation:

1. \*\*TreeNode Class\*\*: Defines the structure of the nodes in the tree.

- Each node has an integer value (`val`) and two child nodes: `left` and `right`.

2. \*\*preorder Function\*\*: Implements the preorder traversal (root -> left -> right).

- It first prints the current node's value, then recursively processes the left and right subtrees.

3. \*\*main Function\*\*: Constructs a sample binary tree and calls the `preorder` function to print the values.

### Example:

For a binary tree like:

```

1

/ \

2 3

/ \ / \

4 5 6 7

```

The output will be:

```

1 2 4 5 3 6 7

```

This code takes the root node as input and prints the preorder traversal in the expected format. You can modify the tree structure in the `main` function to fit different test cases.

Here’s a Java program that performs \*\*inorder traversal\*\* of a binary tree, printing the node values in a single line, space-separated:

### Java Code

```java

class TreeNode {

int val;

TreeNode left, right;

// Constructor

public TreeNode(int val) {

this.val = val;

this.left = this.right = null;

}

}

public class BinaryTreeInorderTraversal {

// Inorder traversal: left -> root -> right

public static void inorder(TreeNode root) {

if (root == null) {

return;

}

// Recursively traverse the left subtree

inorder(root.left);

// Print the current node's value

System.out.print(root.val + " ");

// Recursively traverse the right subtree

inorder(root.right);

}

public static void main(String[] args) {

// Constructing the binary tree for testing

TreeNode root = new TreeNode(1);

root.left = new TreeNode(2);

root.right = new TreeNode(3);

root.left.left = new TreeNode(4);

root.left.right = new TreeNode(5);

root.right.left = new TreeNode(6);

root.right.right = new TreeNode(7);

// Calling the inorder traversal function

inorder(root);

}

}

```

### Explanation:

1. \*\*TreeNode Class\*\*: Defines the structure of the nodes in the tree with integer values (`val`) and two child nodes: `left` and `right`.

2. \*\*inorder Function\*\*: Implements the inorder traversal (left -> root -> right).

- It first recursively processes the left subtree, then prints the current node’s value, and finally recursively processes the right subtree.

3. \*\*main Function\*\*: Constructs a sample binary tree and calls the `inorder` function to print the values.

### Example:

For a binary tree like:

```

1

/ \

2 3

/ \ / \

4 5 6 7

```

The output will be:

```

4 2 5 1 6 3 7

```

This code performs the inorder traversal, printing the nodes in the required order. You can modify the binary tree structure within the `main` function to test different scenarios.