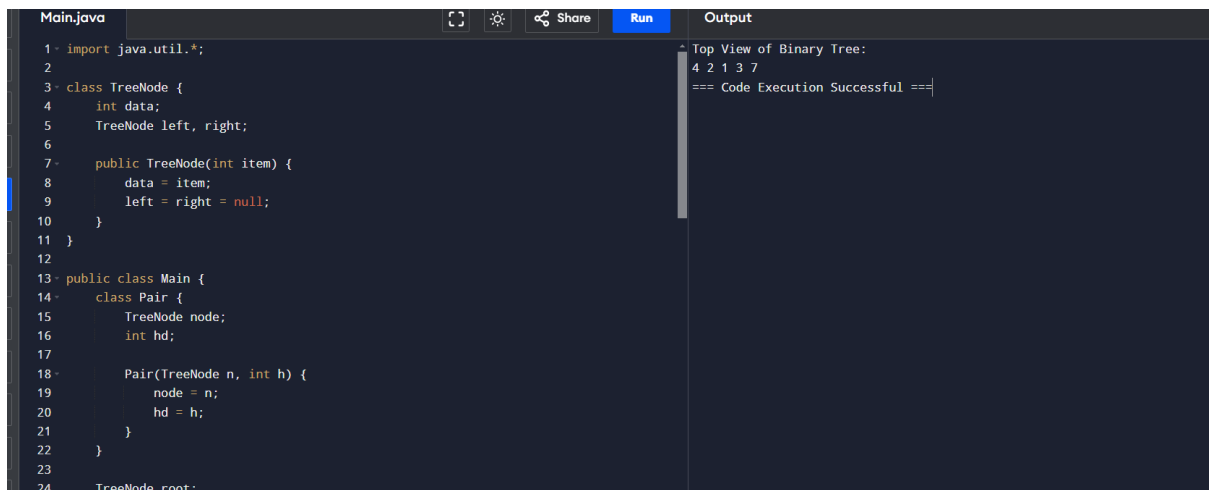


DSA CODING PRACTICE – 10

BINARY TREE

1.Top view of the BST



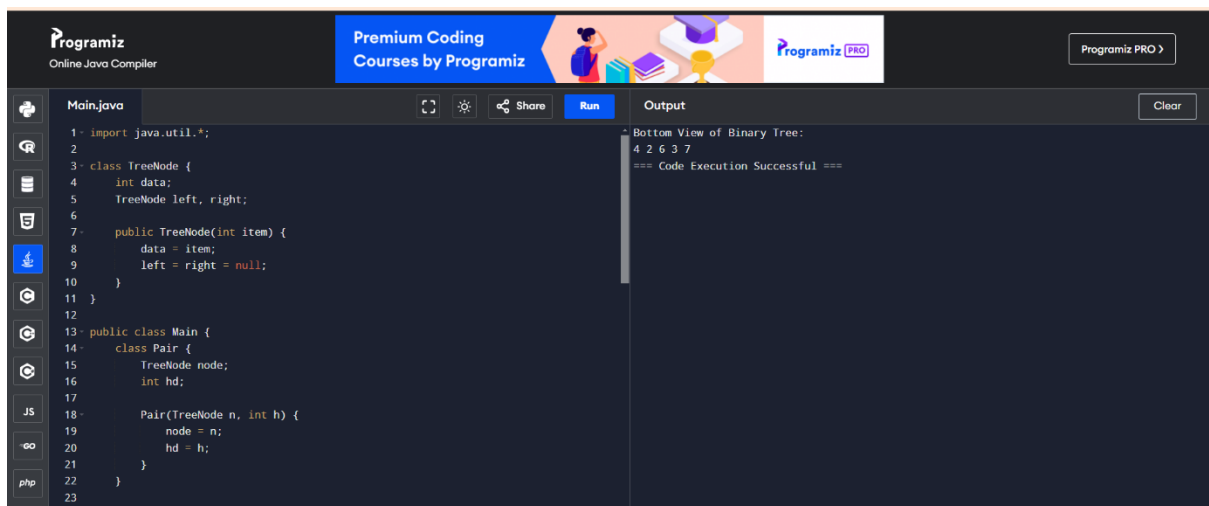
The screenshot shows a Java IDE with a file named 'Main.java'. The code defines a 'TreeNode' class with 'data', 'left', and 'right' attributes, and a 'Main' class with a 'Pair' inner class. The 'Pair' class has a 'Pair(TreeNode n, int h)' method. The output window displays the top view of the binary tree as '4 2 1 3 7' and a success message '=== Code Execution Successful ==='.

```
1- import java.util.*;
2
3- class TreeNode {
4-     int data;
5-     TreeNode left, right;
6
7-     public TreeNode(int item) {
8-         data = item;
9-         left = right = null;
10    }
11 }
12
13- public class Main {
14-     class Pair {
15-         TreeNode node;
16-         int hd;
17
18-         Pair(TreeNode n, int h) {
19-             node = n;
20-             hd = h;
21-         }
22     }
23
24     TreeNode root;
```

Output

```
Top View of Binary Tree:
4 2 1 3 7
=== Code Execution Successful ===
```

2.Bottom view of the BST



The screenshot shows the Programiz online Java compiler interface. The code is identical to the previous one. The output window displays the bottom view of the binary tree as '4 2 6 3 7' and a success message '=== Code Execution Successful ==='. The interface includes a header with the Programiz logo and a sidebar with various icons.

```
1- import java.util.*;
2
3- class TreeNode {
4-     int data;
5-     TreeNode left, right;
6
7-     public TreeNode(int item) {
8-         data = item;
9-         left = right = null;
10    }
11 }
12
13- public class Main {
14-     class Pair {
15-         TreeNode node;
16-         int hd;
17
18-         Pair(TreeNode n, int h) {
19-             node = n;
20-             hd = h;
21-         }
22     }
23
24     TreeNode root;
```

Output

```
Bottom View of Binary Tree:
4 2 6 3 7
=== Code Execution Successful ===
```

3.Right view of BST

Main.java	Output
<pre>16 17 void rightView(TreeNode root) { 18 if (root == null) { 19 return; 20 } 21 22 Queue<TreeNode> queue = new LinkedList<>(); 23 queue.add(root); 24 25 while (!queue.isEmpty()) { 26 int levelSize = queue.size(); 27 28 for (int i = 0; i < levelSize; i++) { 29 TreeNode current = queue.poll(); 30 31 if (i == levelSize - 1) { 32 System.out.print(current.data + " "); 33 } 34 35 if (current.left != null) { 36 queue.add(current.left); </pre>	<pre>^ Right View of Binary Tree: 1 3 7 === Code Execution Successful ===</pre>