```
1 /* PRACTICAL-6: Implementation of Binary Search Trees.
 3 Name: Angat Shah
 4 Enrollment No: 202203103510097
 5 Branch: B.Tech Computer Science and Engineering */
 6
 7 // CODE:
 8
 9 import java.util.Scanner;
10 class TreeNode {
11
      int data;
12
      TreeNode left;
13
      TreeNode right;
14
      public TreeNode(int data) {
15
        this.data = data;
16
        this.left = null;
17
        this.right = null;
18
      }
19 }
20 class BinarySearchTree {
      private TreeNode root;
21
22
      public BinarySearchTree() {
23
        root = null;
24
25
      public void insert(int data) {
26
        root = insertRec(root, data);
27
28
      private TreeNode insertRec(TreeNode root, int data) {
29
        if (root == null) {
30
           root = new TreeNode(data);
31
           return root;
32
         }
33
        if (data < root.data)
34
           root.left = insertRec(root.left, data);
35
        else if (data > root.data)
36
           root.right = insertRec(root.right, data);
37
38
        return root;
39
      }
40
      public void inorder() {
41
        inorderRec(root);
42
43
      private void inorderRec(TreeNode root) {
44
        if (root != null) {
45
           inorderRec(root.left);
46
           System.out.print(root.data + " ");
47
           inorderRec(root.right);
48
         }
49
50
      public void preorder() {
51
        preorderRec(root);
52
53
      private void preorderRec(TreeNode root) {
54
        if (root != null) {
55
           System.out.print(root.data + " ");
           preorderRec(root.left);
56
57
           preorderRec(root.right);
58
         }
```

```
59
 60
       public void postorder() {
 61
         postorderRec(root);
 62
 63
       private void postorderRec(TreeNode root) {
 64
         if (root != null) {
 65
            postorderRec(root.left);
            postorderRec(root.right);
 66
 67
            System.out.print(root.data + " ");
 68
         }
 69
       }
 70 }
 71
     public class practical6 {
 72
       public static void main(String[] args) {
 73
          Scanner scanner = new Scanner(System.in);
 74
         BinarySearchTree bst = new BinarySearchTree();
 75
         int choice:
 76
         do {
 77
            System.out.println("\nBinary Search Tree Operations:");
 78
            System.out.println("1. Insert");
 79
            System.out.println("2. Inorder Traversal");
            System.out.println("3. Preorder Traversal");
 80
 81
            System.out.println("4. Postorder Traversal");
 82
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");
 83
 84
            choice = scanner.nextInt();
 85
            switch (choice) {
 86
               case 1:
 87
                 System.out.print("Enter element to insert: ");
                 int element = scanner.nextInt();
 88
 89
                 bst.insert(element);
 90
                 break;
               case 2:
 91
 92
                 System.out.print("Inorder Traversal: ");
 93
                 bst.inorder();
 94
                 System.out.println();
 95
                 break;
 96
               case 3:
 97
                 System.out.print("Preorder Traversal: ");
 98
                 bst.preorder();
 99
                 System.out.println();
100
                 break;
               case 4:
101
                 System.out.print("Postorder Traversal: ");
102
103
                 bst.postorder();
104
                 System.out.println();
105
                 break;
               case 5:
106
107
                 System.out.println("Exiting...");
108
                 break;
109
               default:
110
                 System.out.println("Invalid choice. Please enter a valid option.");
111
112
          \} while (choice != 5);
113
         scanner.close();
114
       }
115 }
```