Binary Tree:

```
import java.util.Scanner;

public class BinaryTree {
	static class Node {
		int data;
		Node left, right;

		Node(int data) {
			this.data = data;
			left = right = null;
		}
	}

	Scanner input = new Scanner(System.in);

	Node create() {
		System.out.print("Enter data: ");
		int data = input.nextInt();
		Node newNode = new Node(data);
		if (data == -1) {
			return null;
		}
		System.out.println("Enter Left child");
		newNode.left = create();
		System.out.println("Enter Right child");
		newNode.right = create();
		return newNode;
	}

	void preOrder(Node root) {
		if (root == null) {
			return;
		}
		System.out.print(root.data + " ");
		preOrder(root.left);
		preOrder(root.right);
	}

	void inOrder(Node root) {
		if (root == null) {
			return;
		}
		inOrder(root.left);
```

```
        System.out.print(root.data + " ");
        inOrder(root.right);
    }

    void postOrder(Node root) {
        if (root == null) {
            return;
        }
        postOrder(root.left);
        postOrder(root.right);
        System.out.print(root.data + " ");
    }

    public static void main(String[] args) {
        BinaryTree tree = new BinaryTree();
        Node root = tree.create();
        System.out.print("PreOrder: ");
        tree.preOrder(root);
        System.out.print("\nInOrder: ");
        tree.inOrder(root);
        System.out.print("\nPostOrder: ");
        tree.postOrder(root);

    }
}
```

Binary Search Tree:

```
public class BinarySearchTree {
    static class Node {
        int key;
        Node left, right;
        Node(int data) {
            this.key = data;
            left = right = null;
        }
    }

    Node root = null;

    void insertNode(int key) {
        root = insert(root, key);
    }

    Node insert(Node root, int key) {
        if (root == null) {
            root = new Node(key);
            return root;
        }

        if (key < root.key) {
            root.left = insert(root.left, key);
        } else if (key > root.key) {
            root.right = insert(root.right, key);
        }
        return root;
    }

    public static void main(String[] args) {
        BinarySearchTree tree = new BinarySearchTree();
        tree.insertNode(5);
        tree.insertNode(3);
        tree.insertNode(2);
        tree.insertNode(4);
        tree.insertNode(7);
        tree.insertNode(6);
        tree.insertNode(8);
    }
}
```