Q1.Inorder tree traversal

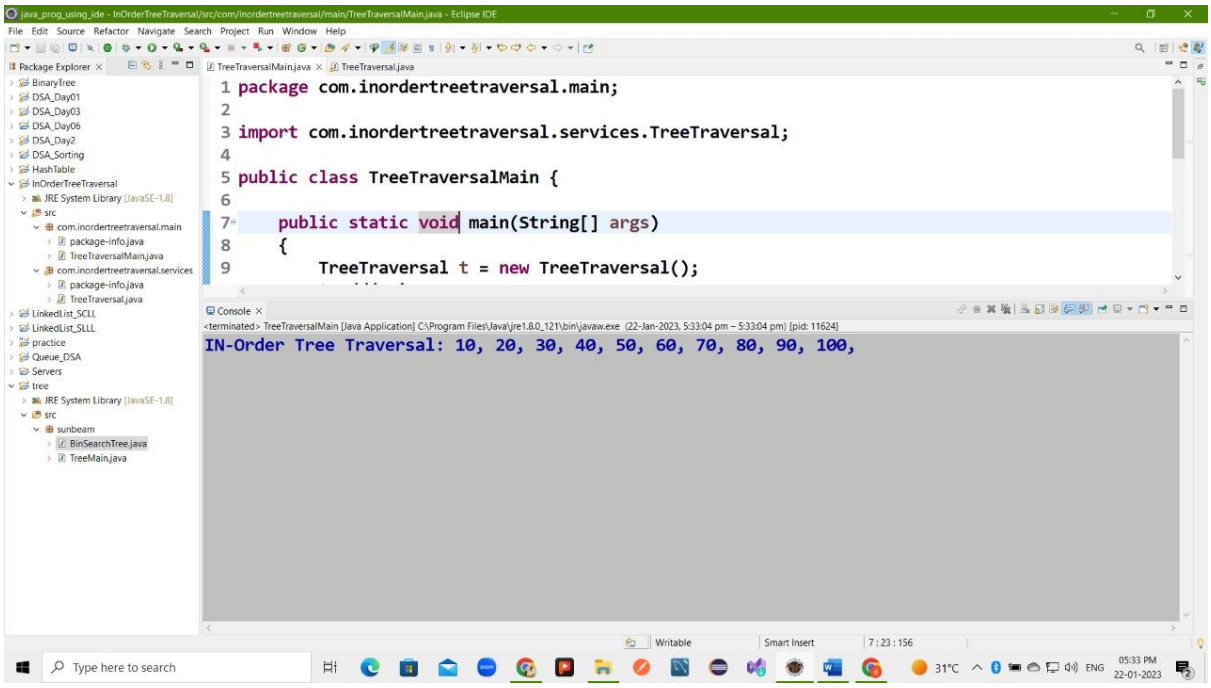
```
package com.inordertreetraversal.services;

public class TreeTraversal {

	static class Node {
		private int data;
		private Node left;
		private Node right;
		private boolean visited;

		public Node()
		{
			data = 0;
			left = null;
			right = null;
			visited = false;
		}
		public Node(int val)
		{
			data = val;
			left = null;
			right = null;
			visited = false;
		}

		@Override
		public String toString()
		{
```

```
                return "Node [data=" + data + "]";
        }
}

private Node root;

public TreeTraversal() {
        root = null;
}

public void add(int val) {
        Node newNode = new Node(val);

        if (root == null)
                root = newNode;
        else {
                Node trav = root;
                while (true) {
                        if (val < trav.data) {
                                if (trav.left == null) {
                                        trav.left = newNode;
                                        break;
                                } else
                                        trav = trav.left;
                        } else {
                                if (trav.right == null) {
                                        trav.right = newNode;
                                        break;
                                } else
                                        trav = trav.right;
                        }
```

```
                }
            }
        }
        public void inorder(Node cur) {
            if (cur == null)
                return;
            inorder(cur.left);
            System.out.print(cur.data + ", ");
            inorder(cur.right);
        }
        public void inorder() {
            System.out.print("IN-Order Tree Traversal: ");
            inorder(root);
            System.out.println();
        }
}
```

```
package com.inordertreetraversal.main;

import com.inordertreetraversal.services.TreeTraversal;

public class TreeTraversalMain {

    public static void main(String[] args)
    {
        TreeTraversal t = new TreeTraversal();
        t.add(50);
        t.add(30);
        t.add(90);
```

```
        t.add(10);
        t.add(40);
        t.add(70);
        t.add(100);
        t.add(20);
        t.add(60);
        t.add(80);

        t.inorder();
    }
}
```



Q2.stack implementation using array

package com.stackusingarray.main;

import java.util.Scanner;

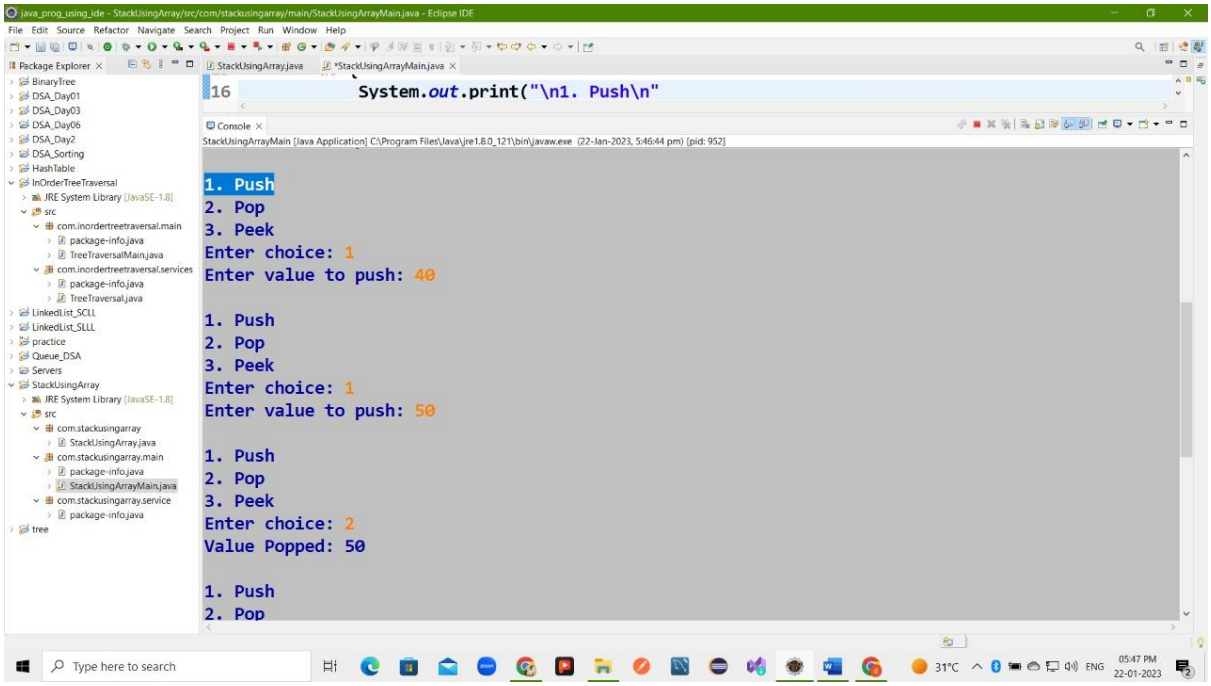
import com.stackusingarray.StackUsingArray;

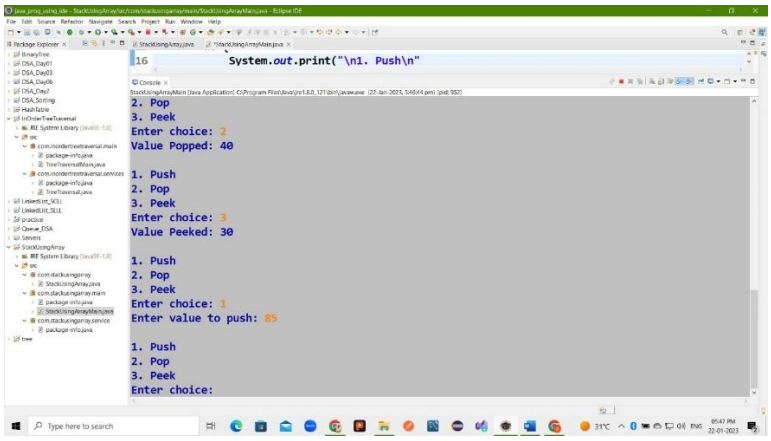
```
public class StackUsingArrayMain {

	public static void main(String[] args)
	{
		int choice, val;

		Scanner sc = new Scanner(System.in);
		StackUsingArray s = new StackUsingArray(9);
		System.out.println(".....Implementation Stack Using Array.....");
		do {
			System.out.print("\n\n1. Push\n"
					+ "2. Pop\n"
					+ "3. Peek\n"
					+ "Enter choice: ");
			choice = sc.nextInt();
			switch (choice) {
			case 1:
				if (s.isFull())
					System.out.println("Oop's...!! Stack is Full.");
				else
				{
					System.out.print("Enter value to push: ");
					val = sc.nextInt();
					s.push(val);
				}
				break;
			case 2:
				if (s.isEmpty())
					System.out.println("Stack is Empty.");
				else
```

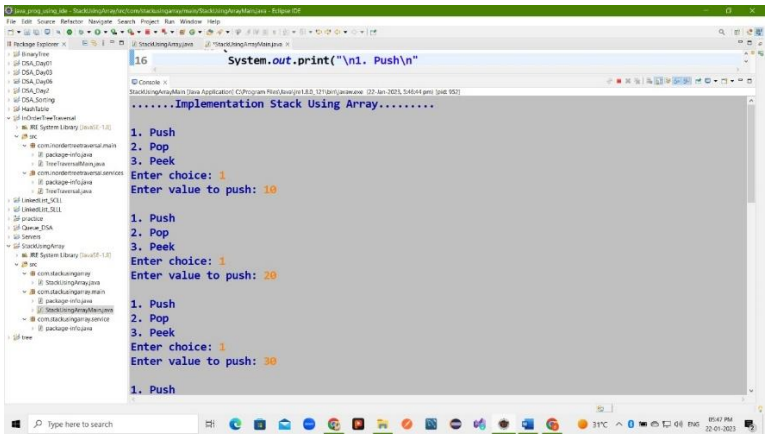
```
                {
                        val = s.peek();
                        s.pop();
                        System.out.println("Value Popped: " + val);
                }
                break;
            case 3:
                if (s.isEmpty())
                        System.out.println("Oop's...!! Stack is Empty.");
                else {
                        val = s.peek();
                        System.out.println("Value Peeked: " + val);
                }
                break;
            }
        } while (choice != 0);
    }
}
```

```
package com.stackusingarray;

public class StackUsingArray {

        private int[] arr;
        private int top;

        public StackUsingArray (int size) {
                top = -1;
                arr = new int[size];
```

```
        }

        public void push(int val) {
            top++;
            arr[top] = val;
        }

        public int peek() {
            return arr[top];
        }

        public void pop() {
            top--;
        }

        public boolean isEmpty() {
            return top == -1;
        }

        public boolean isFull() {
            return top == arr.length-1;
        }
}
```


java_prog_using_ide - StackUsingArray/src/com/stackusingarray/main/StackUsingArrayMain.java - Eclipse IDE
System.out.print("\n1. Push\n"
1. Push
2. Pop
3. Peek
Enter choice: 1
Enter value to push: 40
1. Push
2. Pop
3. Peek
Enter choice: 1
Enter value to push: 50
1. Push
2. Pop
3. Peek
Enter choice: 2
Value Popped: 50
1. Push
2. Pop


System.out.print("\n1. Push\n"
2. Pop
3. Peek
Enter choice: 2
Value Popped: 40
1. Push
2. Pop
3. Peek
Enter choice: 3
Value Peeked: 30
1. Push
2. Pop
3. Peek
Enter choice: 1
Enter value to push: 85
1. Push
2. Pop
3. Peek
Enter choice:


System.out.print("\n1. Push\n"
.....Implementation Stack Using Array.....
1. Push
2. Pop
3. Peek
Enter choice: 1
Enter value to push: 10
1. Push
2. Pop
3. Peek
Enter choice: 1
Enter value to push: 20
1. Push
2. Pop
3. Peek
Enter choice: 1
Enter value to push: 30
1. Push