# Experiment-1

Name: Deepanshu saini
UID: 23BCS13189
Section : 23bcs_KRG_01
Subject Name : DAA
Subject Code:23CSH-301
Date:23/07/2025

---

1. **Aim**: Analyse if the stack Is empty or full and if elements are present return the top element in the stack using templates also perform push and pop on the stack.

2. **Objective:**
   - Implement a generic stack in Java using type-safe generics.
   - Provide essential stack operations: push, pop, isEmpty.
   - Ensure retrieval of the top element only when the stack is not empty.
   - Allow stack operations for any object type without code duplication.

3. **PseudoCode:**

```
CLASS Node

   INTEGER data

   Node next

   CONSTRUCTOR Node(data)

      SET this.data = data

      SET this.next = null

   END CONSTRUCTOR

END CLASS

CLASS Stack

   Node Top

   CONSTRUCTOR Stack()

      SET Top = null
```

```
  END CONSTRUCTOR


  METHOD isEmpty()
    IF Top == null THEN
      RETURN true
    ELSE
      RETURN false
    END IF
  END METHOD
  METHOD push(data)
    CREATE newNode = new Node(data)
    IF isEmpty() THEN
      SET Top = newNode
    ELSE
      SET newNode.next = Top
      SET Top = newNode
    END IF
  END METHOD
METHOD pop()
    IF isEmpty() THEN
      PRINT "Stack is empty"
    ELSE
      SET Top = Top.next
    END IF
```

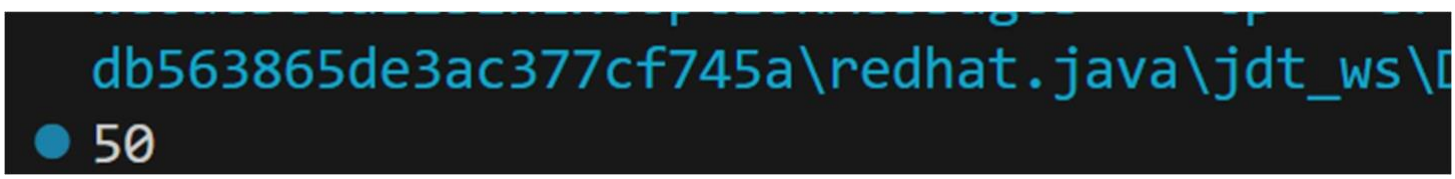
```
    END METHOD


    METHOD printStack()
      IF isEmpty() THEN
        PRINT "Stack is empty"
      ELSE
        SET current = Top
        WHILE current != null DO
          PRINT current.data
          SET current = current.next
        END WHILE
      END IF
    END METHOD
END CLASS


MAIN
    CREATE stack = new Stack()
    CALL stack.push(50)
    CALL stack.push(30)
    CALL stack.pop()
    CALL stack.printStack()
     END MAIN
```

## 4. Code:

```
public class exp1 {
public class Node{
int data;
        Node next;

        Node(int data){
this.data=data;
this.next=null;
        }
    }     class Stack{
public Node Top;

        Stack(){
this.Top=null;
        }

        public boolean isEmpty(){
if(Top==null){                return
true;            }            return
false;
        }

        public void push(int data){
Node newNode = new Node(data);
if (isEmpty()) {
                Top = newNode;
} else {
newNode.next = Top;
                Top = newNode;
            }
        }

        public void pop(){
if (isEmpty()) {
                System.out.println("Stack is empty");
            } else {
                Top = Top.next;
            }
        }

        public void printstack(){
if(isEmpty()){

          System.out.println("Stack is empty");
```

```
        }else{              Node current=Top;
while(current!=null){
System.out.println(current.data);
                current=current.next;
            }
        }
    }
  }
  public static void main(String[] args) {
exp1 stack = new exp1();
exp1.Stack s = stack.new Stack();
s.push(50);
      s.push(30);
      s.pop();
      s.printstack();
  }
}
```

5. Output:




6. Time Complexity: O(n)

7. Learning Outcomes:
   - Understand how to implement and manipulate a stack using Java generics.
   - Gain hands-on experience with type-safe data structures in Java.
   - Learn to apply core stack operations (push, pop, peek) in real scenarios.
   - Strengthen debugging and logic-building skills in stack-based problems.
   - Develop a deeper understanding of LIFO principles and memory management.