DEPARTMENT OF

Discover. Learn. Empower.

COMPUTER SCIENCE & ENGINEERING

Experiment-1.1

Student Name: Vishal Saini
UID: 23BCS10163
Branch: B.E-C.S.E
Section/Group: 23KRG-2A
Semester: 5th
Date of Performance: 15/08/2025
Subject Name: DAA
Subject Code: 23CSH-301

1. **Aim:**
   Create Java programs to manage product details, library systems, and student information using classes, inheritance, and abstraction.

2. **Objective:** To understand stacks.

3. **Input/Apparatus Used:** Stack are implemented using templates.

4. **Procedure:**
   Step1: Create stack.
   Step2: Check underflow and overflow condition.
   Step3: Increment top to store element in stack.
   Step4: Decrement top after removing element form stack.
   Step5: Check is stack empty or not.

```java

class MyStack<T> {  2 usages
    private int capacity;  2 usages
    private int top;  6 usages
    private T[] stack;  4 usages

    public MyStack(int size) {  1 usage
        capacity = size;
        top = -1;
        stack = (T[]) new Object[size];
    }

    public void push(T element) {  3 usages
        if (isFull()) {
            System.out.println("Stack Overflow! Cannot push " + element);
        } else {
            stack[++top] = element;
            System.out.println(element + " pushed into stack.");
        }
    }

    public T pop() {  2 usages
        if (isEmpty()) {
            System.out.println("Stack Underflow! Cannot pop.");
            return null;
        } else {
            return stack[top--];
        }
    }

    public T peek() {  1 usage
        if (isEmpty()) {
            System.out.println("Stack is empty. No top element.");
            return null;
        } else {
            return stack[top];
        }
    }

```

```
42
43      public boolean isEmpty() {  3 usages
44          return top == -1;
45      }
46
47      public boolean isFull() {  2 usages
48          return top == capacity - 1;
49      }
50  }
51
52  class StackDemo {
53      public static void main(String[] args) {
54          MyStack<Integer> intStack = new MyStack<>( size: 5);
55
56          intStack.push( element: 10);
57          intStack.push( element: 20);
58          intStack.push( element: 30);
59
60          System.out.println("Top element: " + intStack.peek());
61
62          System.out.println("Popped: " + intStack.pop());
63          System.out.println("Popped: " + intStack.pop());
64
65          System.out.println("Is stack empty? " + intStack.isEmpty());
66          System.out.println("Is stack full? " + intStack.isFull());
67      }
68  }
```

5. Output:

```
10 pushed into stack.
20 pushed into stack.
30 pushed into stack.
Top element: 30
Popped: 30
Popped: 20
Is stack empty? false
Is stack full? false

Process finished with exit code 0
```