```
// Stack Using Array
public class StackUsingArray {
    private static final int MAX = 3;
    private int[] stack;
    private int top;
    public StackUsingArray() {
        stack = new int[MAX];
        top = -1;
    }
    public void push(int value) {
        if (top == MAX - 1) {
            System.out.println("Stack is full (Overflow).");
        } else {
            stack[++top] = value;
            System.out.println(value);
        }
    }
    public void pop() {
        if (top == -1) {
            System.out.println("Stack is empty (Underflow).");
        } else {
            System.out.println("Popped element: " + stack[top--]);
        }
    }
    public void peek() {
```

```
    if (top == -1) {
        System.out.println("Stack is empty.");
    } else {
        System.out.println("Top element: " + stack[top]);
    }
}
public boolean isEmpty() {
    return top == -1;
}
public boolean isFull() {
    return top == MAX - 1;
}
public void display() {
    if (isEmpty()) {
        System.out.println("Stack is empty.");
    } else {
        System.out.print("Stack elements: ");
        for (int i = top; i >= 0; i--) {
            System.out.print(stack[i] + " ");
        }
        System.out.println();
    }
}
public static void main(String[] args) {
    StackUsingArray stack = new StackUsingArray();
```

```
        System.out.println("=== Stack Implementation Using Arrays ===\n");
        stack.push(10);
        stack.push(20);
        stack.push(30);
        stack.display();
        stack.peek();
        stack.pop();
        stack.display();
        System.out.println("\nTesting overflow condition:");
        stack.push(40);
        stack.push(50);
    }
}
```

Output:

```
=== Stack Implementation Using Arrays ===


10
20
30
Stack elements: 30 20 10
Top element: 30
Popped element: 30
Stack elements: 20 10
Testing overflow condition:
40
```

Stack is full (Overflow).

```
//stack using linkedlist
class Node {
    int data;
    Node next;
    public Node(int data) {
        this.data = data;
        this.next = null;
    }
}

public class StackUsingLinkedList {
    private Node top;
    public StackUsingLinkedList() {
        this.top = null;
    }
    public void push(int value) {
```

```
    Node newNode = new Node(value);
    newNode.next = top;
    top = newNode;
    System.out.println(value);
}
public void pop() {
    if (top == null) {
        System.out.println("Stack is empty (Underflow).");
    } else {
        System.out.println("Popped element: " + top.data);
        top = top.next;
    }
}
public void peek() {
    if (top == null) {
        System.out.println("Stack is empty.");
    } else {
        System.out.println("Top element: " + top.data);
    }
}
public boolean isEmpty() {
    return top == null;
}
public void display() {
    if (isEmpty()) {
```

```
        System.out.println("Stack is empty.");
    } else {
        Node temp = top;
        System.out.print("Stack elements: ");
        while (temp != null) {
            System.out.print(temp.data + " ");
            temp = temp.next;
        }
        System.out.println();
    }
}
public void clear() {
    top = null;
    System.out.println("Stack cleared successfully!");
}
public static void main(String[] args) {
    StackUsingLinkedList stack = new StackUsingLinkedList();
    System.out.println("=== Stack Implementation Using Linked List ===\n");
    stack.push(100);
    stack.push(200);
    stack.push(300);
    stack.display();
    stack.peek();
    stack.pop();
    stack.display();
```

```
        System.out.println("\nAdditional Operations:");
        System.out.println("Is stack empty? " + stack.isEmpty());
        stack.push(400);
        stack.push(500);
        stack.display();
        stack.clear();
        stack.display();
        System.out.println("\nTesting underflow condition:");
        stack.pop();
    }
}
```

Output:

```
=== Stack Implementation Using Linked List ===

100
200
300
Stack elements: 300 200 100
Top element: 300
Popped element: 300
Stack elements: 200 100

Additional Operations:
Is stack empty? false
400
```

500

Stack elements: 500 400 200 100

Stack cleared successfully!

Stack is empty.

Testing underflow condition:

Stack is empty (Underflow)