CS2002-1

Lab Programs by
DISHA KUMARI
NNM24IS074

Submitted to: Dr. Martis

LAB PROGRAM 4:

A local library wants to maintain a stack of book IDs for tracking recently returned books. They ask you to create a Java program to simulate this stack with the following requirements:

1. **Constructors (2 types):**
    - **One constructor should take an array of integers (book IDs) and directly initialize the stack.**
    - **Another constructor should take only the maximum size (number of elements) and create an empty stack.**
2. **Push Operation (Overloaded):**
    - **push(int x) → Push a single element.**
    - **push(int x, int y) → Push two elements together.**
3. **Pop Operation (Overloaded):**
    - **pop() → Pop a single element.**
    - **pop(int n) → Pop n elements (for example, 2 elements at once).**
4. **Display Operation (Overloaded):**
    - **display() → Show the entire stack.**
    - **display(int n) → Show only the top n elements.**
5. **Demonstrate all these operations in the main function with a menu-driven or direct sequence of operations.**

GITHUB LINK: https://github.com/Disha-kumari/Java_Progs

https://github.com/Disha-kumari/Java_Progs/tree/main/prog_4

CODE:

```
package library;
public class Stack{

private int arr[];
private int top;
private int capacity;



Stack(int size) {
  capacity = size;
  arr = new int[capacity];
  top = -1;
} Stack(int[] inputArr) {
  capacity = inputArr.length;
  arr = new int[capacity];
  for (int i = 0; i < capacity; i++) {
    arr[i] = inputArr[i];
  }
  top = capacity - 1;
}

void push(int x) {
  if (top >= capacity - 1) {
        System.out.println("Stack Overflow" + x);
        return;
    }
    arr[++top] = x;
    System.out.println("Pushed: " + x);
}

void push(int x, int y) {
    push(x);
    push(y);
}



int pop() {
    if (top == -1) {
        System.out.println("Stack Underflow");
        return -1;
    }
    return arr[top--];
}

void pop(int n) {
    if (n <= 0) {
        System.out.println("Invalid number of elements to pop");
        return;
    }

    for (int i = 0; i < n; i++) {
        int val = pop();

        if (val == -1) break;

        System.out.println("Popped: " + val);

    }

}


void display() {

    if (top == -1) {

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

        return;

    }


    System.out.println("Stack contents (Top to Bottom):");

    for (int i = top; i >= 0; i--) {



        System.out.println(arr[i]);

    }

}


void display(int n) {

    if (top == -1) {

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

        return;

    }


    if (n <= 0) {

        System.out.println("Invalid number of elements");

        return;
    }

    System.out.println("Top " + n + " elements:");
    for (int i = top; i >= 0 && i > top - n; i--) {
      System.out.println(arr[i]);
    }
  }
}

package library;
import java.util.Scanner;

public class LibraryStackDemo {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        Stack st = null;
        System.out.println("Choose how to initialize stack");
        System.out.println("1. Empty stack with size");
        System.out.println("2. Stack initialized with array");
        int choice = sc.nextInt();

        if (choice == 1) {
          System.out.print("Enter size of stack: ");
          int size = sc.nextInt();
          st = new Stack(size);
```

```
} else if (choice == 2) {
    System.out.print("Enter number of elements: ");
    int n = sc.nextInt();
    int arr[] = new int[n];
    System.out.println("Enter elements: ");
    for (int i = 0; i < n; i++) {
        arr[i] = sc.nextInt();
    }
    st = new Stack(arr);
} else {
    System.out.println("Invalid choice. Exiting.");
    return;
}

int option;
do {
    System.out.println("\n Stack Menu ");
    System.out.println("1. Push one element");
    System.out.println("2. Push two elements");
    System.out.println("3. Pop one element");
    System.out.println("4. Pop multiple elements");
    System.out.println("5. Display full stack");
    System.out.println("6. Display top n elements");
    System.out.println("7. Exit");
    System.out.print("Enter your choice: ");
    option = sc.nextInt();

    switch(option) {
        case 1:
```

```
        System.out.print("Enter element: ");
        int val = sc.nextInt();
        st.push(val);
        break;

      case 2:
        System.out.print("Enter two elements: ");
        int v1 = sc.nextInt();
        int v2 = sc.nextInt();
        st.push(v1, v2);
        break;

      case 3:
        int popped = st.pop();
        if (popped != -1)
          System.out.println("Popped: " + popped);
        break;

      case 4:
System.out.print("Enter number of elements to pop: ");
        int n = sc.nextInt();
        st.pop(n);
        break;

      case 5:
        st.display();
        break;

      case 6:
```

```
                System.out.print("Enter number of top elements: ");
                int topN = sc.nextInt();
                st.display(topN);
                break;

            case 7:
                System.out.println("Exiting...");
                break;

            default:
                System.out.println("Invalid choice");
        }
    } while(option != 7);
  }
}
```

OUTPUT:

```
Choose how to initialize stack
1. Empty stack with size
2. Stack initialized with array
2
Enter number of elements: 3
Enter elements:
4
5
6
```

```
 Stack Menu
1. Push one element
2. Push two elements
3. Pop one element
4. Pop multiple elements
5. Display full stack
6. Display top n elements
7. Exit
Enter your choice: 3
Popped: 6

 Stack Menu
1. Push one element
2. Push two elements
3. Pop one element
4. Pop multiple elements
5. Display full stack
6. Display top n elements
7. Exit
Enter your choice: 1
Enter element: 7
Pushed: 7

 Stack Menu
1. Push one element
2. Push two elements
3. Pop one element
4. Pop multiple elements
```

```
5. Display full stack

6. Display top n elements

7. Exit

Enter your choice: 5

Stack contents (Top to Bottom):

7

5

4


 Stack Menu

1. Push one element

2. Push two elements

3. Pop one element

4. Pop multiple elements

5. Display full stack

6. Display top n elements

7. Exit

Enter your choice: 6

Enter number of top elements: 2

Top 2 elements:

7

5


 Stack Menu

1. Push one element

2. Push two elements

3. Pop one element

4. Pop multiple elements

5. Display full stack
```

```
6. Display top n elements

7. Exit

Enter your choice: 7

Exiting...
```