Stack using arrays

Write a Java program to implement the Stack using arrays. Write push(), pop(), and display() methods to demonstrate its working.

Input Format

3 1 53 1 68 1 20 2 2 2 3 4

Constraints

size of stack should be positive

Output Format

pushed element 53 pushed element 68 pushed element 20 Popped element 20 Popped element 68 Popped element 53 Stack Empty

Sample Input 0

```
3
1
53
1
68
1
20
2
2
2
3
4
```

Sample Output 0

```
pushed element 53
pushed element 68
pushed element 20
Popped element 20
Popped element 68
Popped element 53
Stack Empty
```

```java
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {
 static int top = -1;
 static int arr[] = new int[50];
 public void push(int num, int ele) {
   if (top == num - 1) {
     System.out.println("Stack is overflow");
   } else {
     top++;
     arr[top] = ele;
     System.out.println("pushed element "+ele);
   }
 }
 public void pop() {
   if (top == -1) {
      System.out.println("Stack is underflow");
   } else {
      System.out.println("Popped element " + arr[top]);
      top--;
   }
 }
 public void display(int[] arr2, int num) {
   if (top < 0) {
     System.out.print("Stack Empty");
 } else {
     System.out.print("ELEMENTS : ");
     for (int i = top; i >= 0; i--) {
       System.out.print(arr2[i] + " ");
     }
 }
 System.out.println();
}
public static void main(String[] args) {
 Solution su = new Solution();
 Scanner sc = new Scanner(System.in);
 int num, opt;
 int ele;
 num = sc.nextInt();
 Boolean kl = true;
 while (kl) {
  opt = sc.nextInt();
  switch (opt) {
  case 1:
   ele = sc.nextInt();
   su.push(num, ele);
   break;
 case 2:
   su.pop();
   break;
 case 3:
   su.display(arr, num);
   break;
 case 4:
   kl = false;
   break;
 default:
 break;
 }
}
}
}
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
3
1
53
1
68
1
20
2
2
2
3
4
```

Your Output (stdout)

```
pushed element 53
pushed element 68
pushed element 20
Popped element 20
Popped element 68
Popped element 53
Stack Empty
```

Expected Output

```
pushed element 53
pushed element 68
pushed element 20
Popped element 20
Popped element 68
Popped element 53
Stack Empty
```