EXPERIMENT – 1

Student Name: Gaganjot Singh **UID: 22BCS14843**
Branch: BE-CSE **Section/Group:** 22BCS_JT_802-B
Semester: 5th **Date of Performance:** 18 July 2024
Subject Name: Design &Analysis of Algorithms **Subject Code:** 22CSH-311

1. **Aim:** Analyze 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 operations on the stack.

2. **Objective:** The objective is to implement a generic stack in Java that supports essential operations such as pushing elements onto the stack, popping elements off it, checking if the stack is empty, retrieving the top element without removal, and displaying all elements. The program includes a menu-driven interface using Scanner for user interaction, allowing users to perform stack operations until they choose to exit.

3. **Algorithm**
   - Start
   - Define a generic class Stack<T> with an ArrayList<T> to store elements and a maxSize to limit the stack size.
   - Implement a constructor that initializes the ArrayList with a specified size and sets maxSize.
   - Implement push(T element) to add elements to the stack if it's not full; print a message if full.
   - Implement pop() to remove and return the top element; print a message if stack is empty.
   - Implement isEmpty() to check if the stack is empty.
   - Implement isFull() to check if the stack is full based on maxSize.
   - Implement top() to return the top element without removing it; print a message ifstack is empty.
   - Implement display() to print all elements in the stack.
   - Use a Scanner for user input to perform stack operations (push, pop, isEmpty, isFull,top, display) until user chooses to exit.
   - End

4. Implementation/Code:

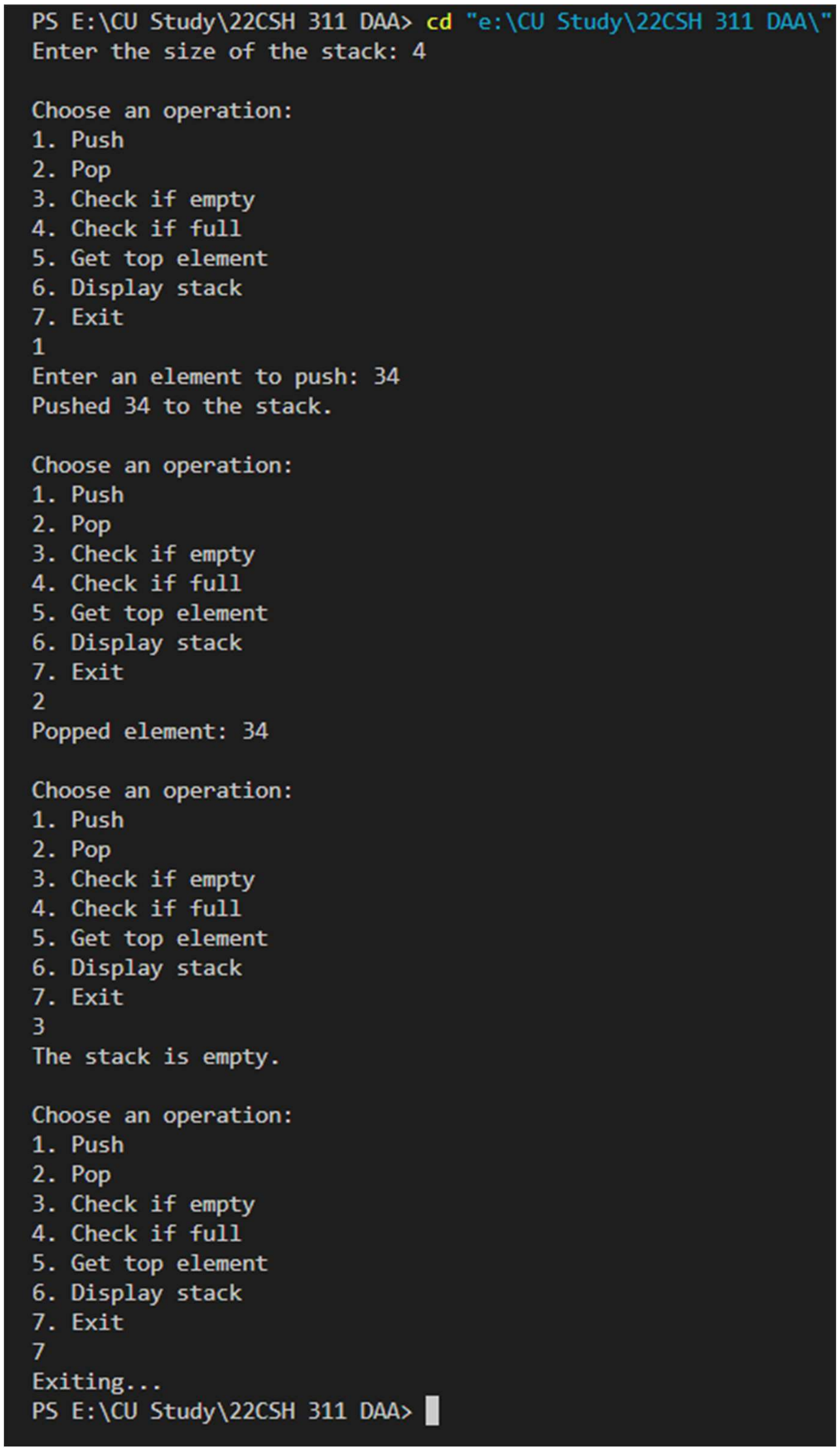
```
import java.util.*;
class Stack<T> {
    private ArrayList<T> elements;
    private int maxSize;
    public Stack(int size) {
        elements = new ArrayList<>(size);
        this.maxSize = size;
    }
    public void push(T element) {
        if (isFull()) {
            System.out.println("Stack is full");
        } else {
            elements.add(element);
            System.out.println("Pushed " + element + " to the stack.");
        }
    }
    public T pop() {
        if (isEmpty()) {
            System.out.println("Stack is empty");
            return null;
        } else {
            return elements.remove(elements.size() - 1);
        }
    }
    public boolean isEmpty() {
        return elements.isEmpty();
    }
    public boolean isFull() {
        return elements.size() == maxSize;
    }
    public T top() {
        if (isEmpty()) {
            System.out.println("Stack is empty.");
            return null;
        } else {
            return elements.get(elements.size() - 1);
        }
    }
    public void display() {
        System.out.println("Stack elements: " + elements);
    }
}

public class GenericStack {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the size of the stack: ");
        int size = scanner.nextInt();
        Stack<Integer> stack = new Stack<>(size);

        while (true) {
            System.out.println("\nChoose an operation:");
            System.out.println("1. Push");
            System.out.println("2. Pop");
            System.out.println("3. Check if empty");
            System.out.println("4. Check if full");
            System.out.println("5. Get top element");
            System.out.println("6. Display stack");
            System.out.println("7. Exit");

            int choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    System.out.print("Enter an element to push: ");
                    int element = scanner.nextInt();
                    stack.push(element);
                    break;
                case 2:
                    Integer poppedElement = stack.pop();
                    if (poppedElement != null) {
                        System.out.println("Popped element: " + poppedElement);
                    }
                    break;
                case 3:
                    if (stack.isEmpty()) {
                        System.out.println("The stack is empty.");
                    } else {
                        System.out.println("The stack is not empty.");
                    }
                    break;
                case 4:
                    if (stack.isFull()) {
                        System.out.println("The stack is full.");
                    } else {
                        System.out.println("The stack is not full.");
                    }
                    break;
                case 5:
                    Integer topElement = stack.top();
                    if (topElement != null) {
                        System.out.println("Top element: " + topElement);
                    }
                    break;
                case 6:
                    stack.display();
                    break;
                case 7:
                    System.out.println("Exiting...");
                    scanner.close();
                    System.exit(0);
                    break;
                default:
                    System.out.println("Invalid choice. ");
                    break;
            }
        }
    }
}
```

5. Output



```
PS E:\CU Study\22CSH 311 DAA> cd "e:\CU Study\22CSH 311 DAA\"
Enter the size of the stack: 4

Choose an operation:
1. Push
2. Pop
3. Check if empty
4. Check if full
5. Get top element
6. Display stack
7. Exit
1
Enter an element to push: 34
Pushed 34 to the stack.

Choose an operation:
1. Push
2. Pop
3. Check if empty
4. Check if full
5. Get top element
6. Display stack
7. Exit
2
Popped element: 34

Choose an operation:
1. Push
2. Pop
3. Check if empty
4. Check if full
5. Get top element
6. Display stack
7. Exit
3
The stack is empty.

Choose an operation:
1. Push
2. Pop
3. Check if empty
4. Check if full
5. Get top element
6. Display stack
7. Exit
7
Exiting...
PS E:\CU Study\22CSH 311 DAA>
```

6. Time Complexity :

- The overall time complexity of the stack operations in the provided Java implementation is O(n).
- While individual operations like push, pop, top, isEmpty, and isFull are O(1).
- And, the display() function iterates through all elements, resulting in O(n) time complexity for that operation.
- Thus, considering worst-case scenarios across multiple operations, the overall complexity remains O(n).

7. Learning Outcomes:

- Learned how to implement the concept of generics in Java.
- Learned about the push() and pop() operations on the stack.
- Learned about the isEmpty(), isFull(), and top() operations on the stack.
- Learned how to use ArrayList to dynamically store and manage stack elements.
- Learned to use control flow with switch-case statements to manage different user operations.