|  |  |
| --- | --- |
| Student Name | Uzair Hussain |
| Roll Number | 21SW085 |
| Section # | 3rd or III |
| Lab # | 9th – Stack using Array and LinkedList |

**Task#01**

**Code:**

import java.util.NoSuchElementException;

class ArrayStack<T> {

    private T[] stack;

    private int peek;       // peek is the index number

    private int capacity;

    public ArrayStack(int capacity) {

        this.capacity = capacity;

        stack = (T[]) new Object[capacity];     // we can not create an array of generic class so we type casted here

        peek = -1;

    }

    public void push(T data) {

        if (peek == capacity - 1) {

            throw new IllegalStateException("Stack is full");

        }

        stack[++peek] = data;

        peek++;

    }

    public T pop() {

        if (isEmpty()) {

            throw new NoSuchElementException("Stack is empty");

        }

        return stack[peek--];

    }

    public T peek() {

        if (isEmpty()) {

            throw new NoSuchElementException("Stack is empty");

        }

        return stack[peek];

    }

    public boolean isEmpty() {

        return peek == -1;

    }

    public int search(T data) {

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

            if (stack[i].equals(data)) {

                return peek - i + 1;

            }

        }

        return -1;

    }

}

**Output 1:**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Task 2:**

**Code:**

class LinkedStack<T> {

    private Node<T> head;

    private Node pNode;

    private int size;

    private class Node<T> {

        T data;

        Node<T> next;

        public Node(T data) {

            this.data = data;

            this.next = null;

        }

    }

    public LinkedStack() {

        head = null;

    }

    public void push(T data) {

        Node<T> newNode = new Node<>(data);

        newNode.next = head;

        head = newNode;

    }

        public T pop() {

            if (isEmpty()) {

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

            }

            T data = head.data;

            head = head.next;

            return data;

        }

        public T peek() {

            if (isEmpty()) {

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

            }

            return head.data;

        }

        public boolean isEmpty() {

            return head == null;

        }

        public int search(T data) {

            int position = 0;

            Node<T> current = head;

            for(current=head;current!=null;current=current.next){

                if(current.data.equals(data)){

                    return position;

                }

                position++;

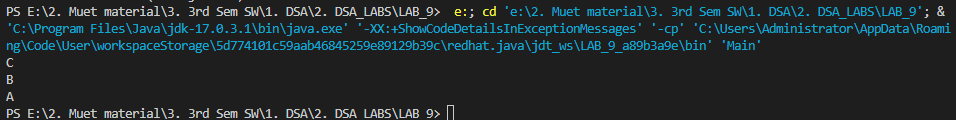
            }

            return -1;

        }

    }

**Output:**

****

**Task 3:**

**Code:**

import java.util.Scanner;

class Task\_3<T>{

        static class Node<T>{

            Scanner sc = new Scanner(System.in);

            T data;

            Node next;

            Node current;

            Node pre;

            int size ;

            Node (T data){

                this.data = data;

                this.next = null;

                this.pre = null;

            }

        }

        private Node head;

        private int size;

        private Node current;

        public void push(T data){

            Node newNode = new Node(data);

            if(head == null){

                head = newNode;

            } else if (head.next == null) {

                current = newNode;

                current.pre = head;

                head.next = current;

            }

            else {

                current.next = newNode;

                newNode.pre = current;

                current = newNode;

            }

        }

        public Object pop(){

            if(head==null){

                return null;

            }

            Object o = head.data;

            head = head.next;

            return o;

        }

        public Object peek(){

            return head.data;

        }

        public void backwarddisplay(){

            for(Node p = current ; p!=null; p = p.pre){

                System.out.println(p.data);

            }

        }

        public void forwardDisplay(){

            for(Node p = this.head ; p!=null; p = p.next){

                System.out.println(p.data);

            }

        }

}

**Output:**

**Text

Description automatically generated**

**Main Class:**

public class Main{

    public static void main(String[] args) {

        Task\_3 stack = new Task\_3();

        String[] url = {"1.facebook.com","2.linkedin.com","3.github.com"};

        for(int i = 0; i<url.length;i++){

            stack.push(url[i]);

        }

        System.out.println("Forward Display");

        stack.forwardDisplay();

        System.out.println("BackWard Display");

        stack.backwarddisplay();

            // Task Linkedstack methods main

        LinkedStack list=new LinkedStack<>();

        list.push("HP");

        list.push("Dell");

        list.push("ASUS");

        list.display();

            // Task ArrayStack methods main

        ArrayStack a=new ArrayStack<>(3);

        a.push("A");

        a.push("B");

        a.push("C");

        a.display();

}

}

**Github Repository for all Lab Tasks: (from lab 1 to continue)**

[**https://github.com/UzairHussain193/DSA\_LABS\_21SW**](https://github.com/UzairHussain193/DSA_LABS_21SW)

**The End!**