LAB4

NAME:GHULAM MUSTAFA CMS ID:023-22-0099

DESCRIPTION:THIS CODE WILL IMPLEMENTS ALL THE REQUIRED METHODS.

TASK1

public class ArrayStack

{

private int arr[];

private int size;

private int top;

ArrayStack(int size)

{

arr=new int [size];

this.size=size;

top=-1;

}

public void push(int value )

{

if(isFull())

{

System.*out*.println("Sorry! Stack is overflow");

}

else

{

if (top != size)

{

arr[++top] = value;

}

}

}

public int pop()

{

if(isEmpty())

{

System.*out*.println("Sorry! Stack is Underflow");

return -1;

}

else

{

int popedElement=arr[top];

top--;

return popedElement;

}

}

public boolean isEmpty()

{

return top==-1;

}

public int peek()

{

return top;

}

public boolean isFull()

{

return top==size-1;

}

TASK2

DESCRIPTION:THIS CODE WILL IMPLEMENTS MAIN METHOD OF ABOVE CLASS.

public static void main(String[] args)

{

ArrayStack s1=new ArrayStack(4);

s1.push(10);

s1.push(20);

s1.push(30);

s1.push(40);

s1.peek();

System.*out*.println( "The poped elements are:");

System.*out*.println(s1.pop());

System.*out*.println( s1.pop());

System.*out*.println(s1.pop());

System.*out*.println( s1.pop());

}

}

TASK3

DESCRIPTION:THIS CODE WILL IMPLEMENTS ALL THE REQUIRED METHODS.

public class LinkedListStack

{

public static class Node

{

int data;

Node next;

Node(int data)

{

this.data=data;

}

}

Node head=null;

Node top=null;

public void push(int value)

{

Node new\_node=new Node(value);

if(head==null) {

head = new\_node;

top=new\_node;

}

else {

top.next = new\_node;

top = new\_node;

}

}

public boolean isEmpty()

{

return head==null;

}

public int pop() {

int popedElement;

if (isEmpty()) {

System.*out*.println("Stack is underflow");

return -1;

} else {

Node cur = head;

popedElement = top.data;

if(head==top){

top=null;

head=null;

}

while (cur.next != top) {

cur = cur.next;

}

cur.next=null;

top=cur;

}

return popedElement;

}

public int peek()

{

if(isEmpty())

{

System.*out*.println("Stack is Empty");

return -1;

}

return top.data;

}

public void display()

{

Node temp2=head;

if(isEmpty())

{

System.*out*.println("Sorry! Stack is Empty");

}

else

{

while(temp2!=null)

{

System.*out*.println(temp2.data);

temp2=temp2.next;

}

}

}

TASK4

DESCRIPTION:THIS CODE WILL IMPLEMENTS MAIN METHOD OF ABOVE CLASS.

public static void main(String args[])

{

LinkedListStack list1=new LinkedListStack();

list1.push(15);

list1.push(20);

list1.push(25);

System.*out*.println("The peek element is:" +list1.peek());

System.*out*.println("The poped elements are:");

System.*out*.println(list1.pop());

System.*out*.println(list1.pop());

System.*out*.println(list1.pop());

System.*out*.println(list1.pop());

}

}

TASK7

DESCRIPTION:THIS CODE WILL IMPLEMENTS ALL THE REQUIRED METHODS.

package Browser;

public class Browser {

private String[] backwardStack;

private String[] forwardStack;

private int backwardTop;

private int forwardTop;

Browser(int size){

backwardStack=new String[size];

forwardStack=new String[size];

backwardTop = forwardTop = -1;

}

public void visitPage(String site){

if(backwardTop<backwardStack.length-1){

backwardStack[++backwardTop] = site;

forwardTop = -1;

System.out.println(site);

}else {

System.out.println("History full");

}

}

public void back(){

if(backwardTop>=1){

String currSite = backwardStack[backwardTop--];

forwardStack[++forwardTop]=currSite;

System.out.println("Navigating back to "+backwardStack[backwardTop]);

}else

System.out.println("No history available");

}

public void forward(){

if(forwardTop>=0){

String nextPage = forwardStack[forwardTop--];

backwardStack[++backwardTop]=nextPage;

System.out.println("Navigating forward to "+nextPage);

}else System.out.println("Can't go forward");

}

}

TASK9

DESCRIPTION:THIS CODE WILL IMPLEMENTS ALL THE REQUIRED METHODS.

import java.util.Scanner;

public class StackMin

{

int arr[];

int stackMin;

int top=-1;

int size;

int popedElement;

StackMin()

{

System.*out*.println("Enter the size of stack:");

Scanner sc = new Scanner(System.*in*);

size = sc.nextInt();

arr = new int[size];

stackMin = Integer.*MAX\_VALUE*;

}

public void push(int num)

{

if (top == size - 1)

{

System.*out*.println("Stack is overflow");

} else

{

arr[++top] = num;

if(num < stackMin)

{

stackMin=num;

}

}

}

public int getMin()

{

return stackMin;

}

public int pop()

{

if(top==-1)

{

System.*out*.println("Stack is underflow");

return -1;

}

else

{

popedElement=arr[top];

top--;

return popedElement;

}

}

public static void main(String args[])

{

StackMin s1=new StackMin();

s1.push(10);

s1.push(9);

s1.push(11);

System.*out*.println("The minimum value in stack is:"+s1.getMin());

System.*out*.println("The poped value is:"+s1.pop());

}

}