Lab 7: Stack

Objective(s)

- Stack implementation using arrays
- Stack implementation using Linked List

Tool(s)/Software

Java programming language with NetBeans IDE.

Description

1- Stack Implementation in Java (Using **Array**)

```
public class StackArray {

    int top;
    int[] Stack;

    public StackArray(int capacity) {
        Stack = new int[capacity];
        top = -1;
    }
```

STACK BASIC FUNCTIONS:

- **isEmpty():** Examines whether the stack is empty or not
- **isFull():** Examines whether the stack is full or not
- **Push(int value):** Add a value at the top of the stack
- **Top()/peek():** Return/Read the value at the top of the stack
- **Pop():** Remove the value from the top of the stack
- **Display():** Displays all the elements in the stack
- **makeEmpty():** Delete all elements from stack
- **size():** Return the number of elements in the stack

```
public boolean isEmpty() { return (top==-1);}
public int size() { return top+1;}
public int top() {
    if(isEmpty())
    {
        System.out.println("Empty Stack.");
        return -1;
    }

    return Stack[top];
}
```

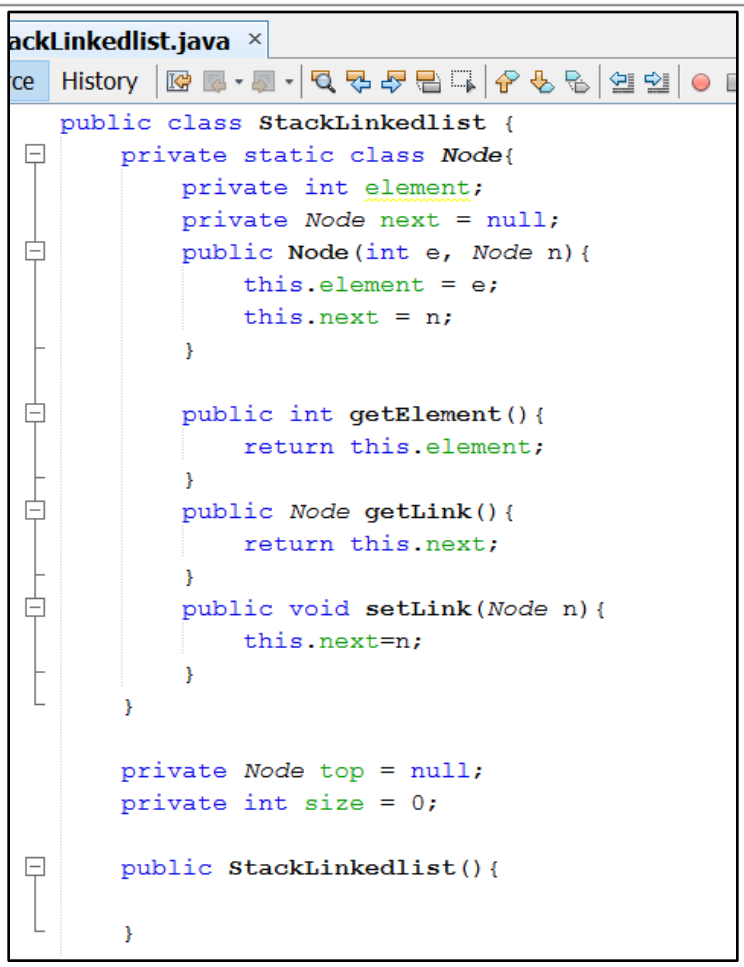
```
public void display() {
    System.out.println("----- Display Method ------");

    if (isEmpty()) {
        System.out.println("Empty Stack.");
        return;
    }

    for (int k = top; k >= 0; k--) {
        System.out.println(Stack[k]);
    }
    System.out.println();
}
```

```
public void push(int value)
{
     if (isFull()) {
        System.out.println("Stack Overflow..");
        return;
    }

     top=top+1;
     Stack[top]=value;

    System.out.println(value+" PUSHED into stack");
}
```

Task 1: Implement isFull() method?

```
public void pop() {
    if (isEmpty()) {
        System.out.println("Stack Underflow..");
        return;
    }
    int temp = Stack[top];
    top = top - 1;
    System.out.println(temp + " POPPED from stack");
}
```

2- Stack Implementation in Java (Using **LinkedList**)



ackLinkedlist.java ×

ce History

```
public class StackLinkedlist {
    private static class Node{
        private int element;
        private Node next = null;
        public Node(int e, Node n){
            this.element = e;
            this.next = n;
        }

        public int getElement(){
            return this.element;
        }
        public Node getLink(){
            return this.next;
        }
        public void setLink(Node n){
            this.next=n;
        }
    }

    private Node top = null;
    private int size = 0;

    public StackLinkedlist(){

    }
```

STACK BASIC FUNCTIONS:

- **isEmpty():** Examines whether the stack is empty or not
- **Push(int value):** Add a value at the top of the stack
- **Top()/peek():** Return/Read the value at the top of the stack
- **Pop():** Remove the value from the top of the stack
- **Display():** Displays all the elements in the stack
- **makeEmpty():** Delete all elements from stack
- **size():** Return the number of elements in the stack

```
public void top(){
    System.out.println("\n---- Top() method ----");
    System.out.println("Top Element of Stack: " + top.element);
}

public boolean isEmpty(){
    return size == 0;
}
```

```
public void display(){
    System.out.println("\n---- display method() -----");
    if (top == null){
        System.out.println(" Stack is Empty ");
        return;
    }
    Node temp = top;
    System.out.print("Stack elements: ");
    while(temp != null){
        System.out.print(temp.element + " ");
        temp = temp.getLink();
    }
    System.out.println();
}
```

```
public void push(int value){
    System.out.println("\n----- PUSH method() -----");
    Node newNode = new Node(value, null);
    newNode.setLink(top);
    top = newNode;
    size++;
    System.out.println(value + " - push to Stack ");
}
```

```
public void pop(){
    System.out.println("\n---- POP method()-----");
    if (isEmpty()){
        System.out.println("Stack is underflow (Stack Empty) ");
        return;
    }
    System.out.println(top.element + "  is POP from the STACK");
    top = top.next;
    size--;
}
```

Tasks/Assignments(s)

1- Create **StackArray** class in Java that implements Stack using Array and implement all the methods for the following stack operations: *push, pop, top, size, isEmpty, isFull, display* and *makeEmpty*.

In the main, create object **S1** from **StackArray,** push some values and print the top after each push operation.

2- Create **StackLinkedList** class in Java that implements Stack using Singly Linked List and implement all the methods for the following stack operations: *push, pop, top, size, isEmpty, display* and *makeEmpty.*

In the main, create object **S2** from **StackLinkedList**, pop all elements from stack **S1** and push them into **S2**.

Deliverables(s)

You are required to implement and deliver a Java program(s) as described in the previous section.