Nama : Ahmad Siddiq Priaji

NIM : 22/496854/PA/21370

Kelas : KOMB

**Link List**

Node.java

public class Node {

    int data;

    Node next;

    //constructor for initialization

    Node(int data) {

        this.data = data;

    }

    //print data

    public void displayLink() {

        System.out.print(" (" + this.data + ") ");

    }

}

LinkListInit.java

public class LinkListInit {

    Node first;

    //constructor

    LinkListInit() {

        first = null;

    }

    //check whether list is empty or not

    public boolean isEmpty() {

        return (first == null);

    }

    //insert data from the front of the list

    public void insertFirst(int data) {

        Node newNode = new Node(data);

        newNode.next = first;

        first = newNode;

    }

    //insert data from the back of the list

    public void insertLast(int data) {

        if (first == null) insertFirst(data);

        else {

            Node temp = first;

            while (temp.next != null) {

                temp = temp.next;

            }

            temp.next = new Node(data);

        }

    }

    //delete the first data

    public void deleteFirst() {

        Node temp = first;

        first = first.next;

        temp.next = null;

    }

    //delete the last data

    public void deleteLast() {

        Node temp = first;

        while (temp.next.next != null) {

            temp = temp.next;

        }

        temp.next = null;

    }

    //print the list

    public void displayList() {

        System.out.print("List (first--›last): ");

        Node current = first;

        while (current != null) {

            current.displayLink();

            current = current.next;

        }

        System.out.println("");

    }

}

TestLinkList.java

import java.util.Scanner;

public class TestLinkList {

    public static void main(String[] args) {

        LinkListInit theList1 = new LinkListInit();

        LinkListInit theList2 = new LinkListInit();

        //defining the size of the lists

        Scanner in = new Scanner(System.in);

        int nodeNum1;

        int nodeNum2;

        int tempNum;

        System.out.print("First list size? ");

        nodeNum1 = in.nextInt();

        //initializing and displaying the lists

        for (int i = 0; i < nodeNum1; i++) {

            System.out.print("Insert number: ");

            tempNum = in.nextInt();

            theList1.insertLast(tempNum);

        }

        theList1.displayList();

        System.out.print("Second list size? ");

        nodeNum2 = in.nextInt();

        for (int i = 0; i < nodeNum2; i++) {

            System.out.print("Insert number: ");

            tempNum = in.nextInt();

            theList2.insertFirst(tempNum);

        }

        theList2.displayList();

        //deleting elements of the lists

        System.out.print("\nDeleting the first node of the first list");

        theList1.deleteFirst();

        theList1.displayList();

        System.out.print("\nDeleting the last node of the second list");

        theList2.deleteLast();

        theList2.displayList();

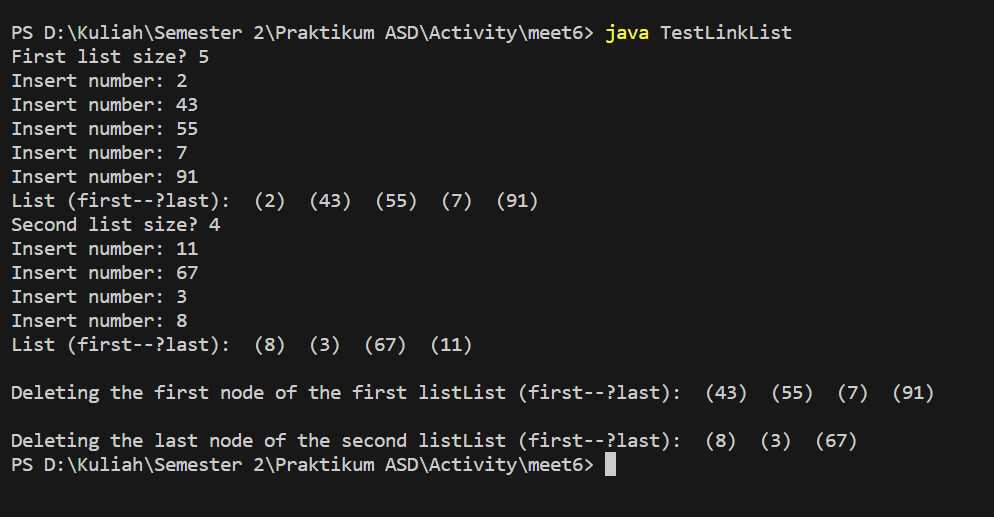
        //tambahan close Scanner 'in'

        in.close();

    }

}

Hasil:



**Stack**

StackInit.java

import java.util.Arrays;

class StackInit { //contains stack methods

    private final int maxSize; //size of stack array

    private int[] stackArray; //initialize array

    private int top; //top of stack

    public StackInit(int s) { //constructor

        maxSize = s; //set array size

        stackArray = new int[maxSize]; //create array

        top = -1; //no items yet

    }

    public void push(int j) { //put item on top of stack

        stackArray[++top] = j; //increment top, insert item

    }

    public double pop() { //take item from top of stack

        return stackArray[top--]; //access item, decrement top

    }

    public boolean isEmpty() { //true if stack is empty

        return (top == -1);

    }

    public void printStack() {

        System.out.println(Arrays.toString(stackArray));

    }

} // end class StackInit

Stack.java

// Stack.java

// demonstrates stack

import java.io.IOException; //exception for I/0

import java.util.Scanner; //for input

public class Stack {

    public static void main(String[] args) throws IOException {

        int stackSize; //stack size

        int stackNum; //number to be inserted in stack

        Scanner in = new Scanner(System.in);

        System.out.print("How many integer? ");

        stackSize = in.nextInt(); //insert stack size

        StackInit theStack = new StackInit(stackSize); //make new stack

        for (int i = 0; i < stackSize; i++) {

            System.out.print("Enter number: ");

            stackNum = in.nextInt(); //insert number

            theStack.push(stackNum); //push element onto stack

        }

        theStack.printStack(); //print Stack

        while (!theStack.isEmpty()) { //until it is empty, delete item from stack

            double value = theStack.pop();

            System.out.print(value); //display the popped item

            System.out.print(" ");

        }

        System.out.println("");

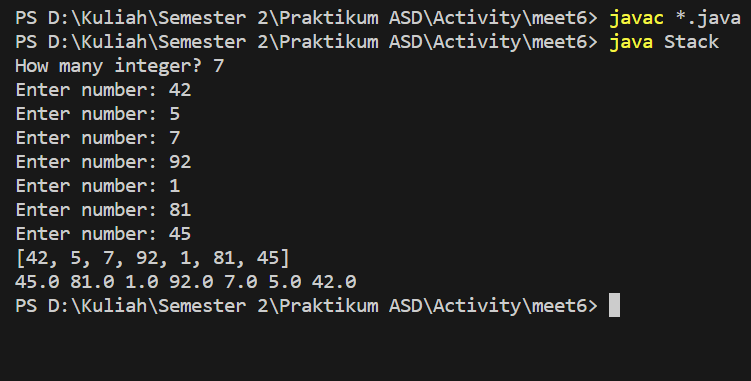
        //tambahan close Scanner 'in'

        in.close();

    } // end main()

} //end class Stack

Hasil:



**Queue**

QueueInit.java

import java.util.Arrays;

class QueueInit { //contains queue methods

    private int maxSize;

    private int[] queueArray;

    private int front;

    private int rear;

    private int nItems;

    public QueueInit(int s) { //constructor

        maxSize = s;

        queueArray = new int[maxSize];

        front = 0;

        rear = -1;

        nItems = 0;

    }

    public void enqueue(int j) { //put item at rear of queue

        if (rear == maxSize - 1) {

            rear = -1;

        } //deal with wraparound

        queueArray[++rear] = j; //increment rear and insert

        nItems++; //one more item

    }

    public int dequeue() { //take item from front of queue

        int temp = queueArray[front++]; //get value and increment front

        if (front == maxSize) {

            front = 0;

        } //deal with wraparound

        nItems--; //one less item

        return temp;

    }

    public boolean isEmpty() { //true if queue is empty

        return (nItems == 0);

    }

    public boolean isFull() { //true if queue is full

        return (nItems == maxSize);

    }

    public void printQueue() {

        System.out.println(Arrays.toString(queueArray));

    }

}

Queue.java

import java.io.IOException;

import java.util.Scanner;

public class Queue {

    public static void main(String[] args) throws IOException {

        int queueSize; //for queue size

        int numTemp; //for inserted number

        int numChoice = 0; //for command

        Scanner in = new Scanner(System.in); //for input

        System.out.print("Enter queue size: ");

        queueSize = in.nextInt();

        QueueInit theQueue = new QueueInit(queueSize); //set queue

        while (numChoice != 3) {

            System.out.println("\n 1: Enqueue \t 2 : Dequeue \t 3 : End");

            System.out.print("Enter command: ");

            numChoice = in.nextInt();

            if (numChoice == 1) {

                if (theQueue.isFull()) {

                    System.out.println("Queue is full");

                } else {

                    System.out.print("Enter number: ");

                    numTemp = in.nextInt();

                    theQueue.enqueue(numTemp);

                }

            }

            else if (numChoice == 2) {

                if (theQueue.isEmpty()) {

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

                } else {

                    numTemp = theQueue.dequeue();

                    System.out.println("Dequeue number: " + numTemp);

                }

            }

            else if (numChoice != 3) {

                System.out.println("Wrong command");

            }

        } //end main()

        //tambahan close Scanner 'in'

        in.close();

    } //end class Queue

}

Hasil:

