LAPORAN PRAKTIKUM

STRUKTUR DATA LINIER

MODUL 11

Dosen Pengampu

JB. Budi Darmawan S.T., M.Sc.

DISUSUN OLEH :

AGUSTINUS KEVIN YUDIPRATAMA

235314029

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS SANATA DHARMA

YOGYAKARTA

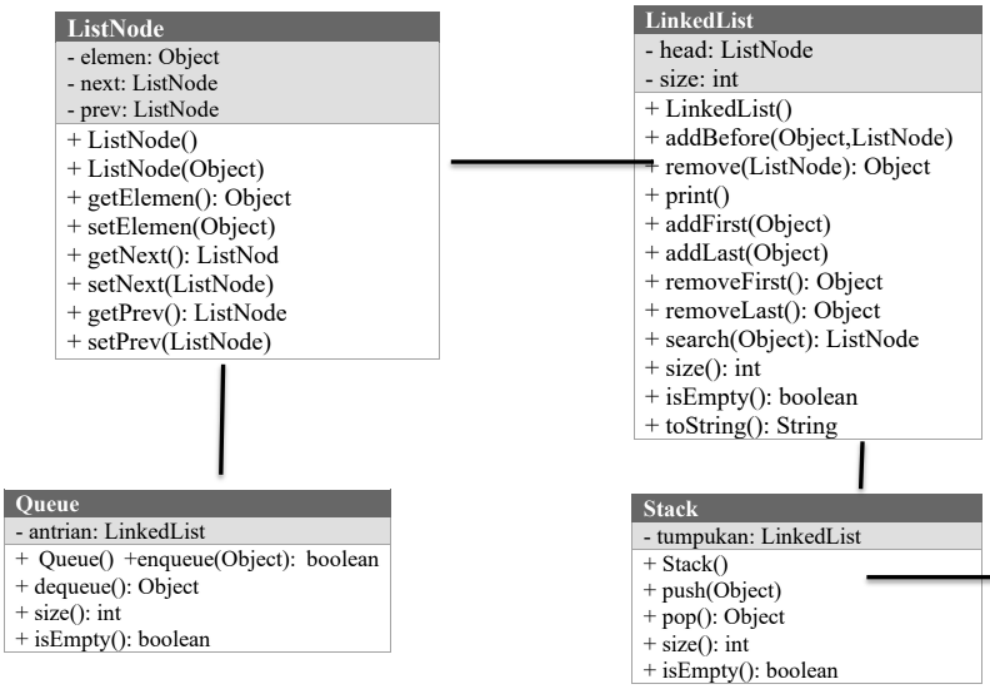
2024

A. TUJUAN PRAKTIKUM

Mampu menguasai dan mengimplementasikan pengelolaan struktur data stack dengan struktur data statis dan struktur data dinamis bertipe Object

Mampu menguasai dan mengimplementasikan pengelolaan struktur data queue dengan struktur data statis dan struktur data dinamis bertipe Object

B. DIAGRAM UML



C. SOURCE CODE

Class mainnya

queue

```java
package vscode.Modul11;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        QueueDinamis antrian = new QueueDinamis();  // Without specifying type
        antrian.enqueue(x:14);
        antrian.enqueue(x:25);
        antrian.enqueue(x:58);

        while (!antrian.isEmpty()) {
            System.out.println(antrian.dequeue());
        }
    }
}
```


Stack

```java
package vscode.Modul11;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        Stack<Integer> myStack = new Stack<>();

        myStack.push(element:10);
        myStack.push(element:20);
        myStack.push(element:30);



        while (!myStack.isEmpty()) {
            System.out.println(myStack.pop());
        }
    }
}
```

LindkedList

```java
package vscode.Modul11;

import java.util.NoSuchElementException;

public class LinkedList {
    private ListNode head;
    private int size;

    public LinkedList(){
        head = new ListNode();
        head.setNext(head);
        head.setPrev(head);
        size = 0;
    }

    public void addBefore(Object elemen, ListNode bantu){
        ListNode baru = new ListNode(elemen);
        baru.setNext(bantu);
        baru.setPrev(bantu.getPrev());
        bantu.getPrev().setNext(baru);
        bantu.setPrev(baru);
        size++;
    }
    public Object remove (ListNode bantu){
        if (isEmpty()) {
            throw new NoSuchElementException();
        }
        Object elemen = bantu.getElemen();
        bantu.getPrev().setNext(bantu.getNext());
        bantu.getNext().setPrev(bantu.getPrev());
        size--;
        return elemen;
    }
    public void print() {
        ListNode bantu = head.getNext();
        while (bantu != head) {
            System.out.println(bantu.getElemen());
            bantu = bantu.getNext();
        }
    }
    public void addFirst(Object x) {
        addBefore(x, head);
    }

    public void addLast(Object x) {
        addBefore(x, head);
    }
    public Object removeLast(){
        return remove(head.getPrev());
    }
    public ListNode search(Object key){
        ListNode bantu = head;
        while (bantu!= head) {
            if (bantu.getElemen() == key) {
                return bantu;
            }
            bantu = bantu.getNext();
        }
        return null;
    }
    public int size(){
        return size;
    }
    public boolean isEmpty(){
        return head.getNext() == head;
    }
    @Override
    public String toString(){
        if (isEmpty()) {
            return "";
        }
        StringBuilder result = new StringBuilder();
        ListNode bantu = head.getNext();
        while (bantu != head) {
            result.append(bantu.getElemen().toString());
            bantu = bantu.getNext();
            if (bantu != head) {
                result.append(str:"null");

            }

        }
       return toString();
    }
}
```

List Node

```java
package vscode.Modul11;

public class ListNode {
    private Object elemen;
    private ListNode next, prev;

    public  ListNode(){
        this.elemen = null;
        next= null;
        prev = null;

    }

    public ListNode(Object elemen){
        this.elemen= elemen;
        this.next = null;
        this.prev = null;
    }


    public ListNode(Object elemen, ListNode next, ListNode prev) {
        this.elemen = elemen;
        this.next = next;
        this.prev = prev;
    }


    public Object getElemen() {
        return elemen;
    }


    public void setElemen(Object elemen) {
        this.elemen = elemen;
    }


    public ListNode getNext() {
        return next;
    }


    public void setNext(ListNode next) {
        this.next = next;
    }


    public ListNode getPrev() {
        return prev;
    }


    public void setPrev(ListNode prev) {
        this.prev = prev;
    }



}
```

Queue dinamis

```java
package vscode.Modul11;

import java.util.NoSuchElementException;

public class QueueDinamis {
    private LinkedList antrian;

    public QueueDinamis(){
        antrian = new LinkedList();
    }
    public boolean enqueue(Object x){
        antrian.addLast(x);
        return true;
    }
    public Object dequeue(){
        if (!isEmpty()) {
            return antrian.remove(bantu:null);

        }else {
            throw new NoSuchElementException(s:" ");
        }
    }
    public int size(){
        return antrian.size();
    }
    public boolean isEmpty(){
        return antrian.isEmpty();
    }
}
```

Queue Statis

```java
package vscode.Modul11;

import java.util.NoSuchElementException;

public class QueueStatis {

    private Object[] elemen;
    private int front;
    private int rear;
    private int size;

    // Default constructor
    public QueueStatis() {
        this(s:10); // Default size of 10
    }

    public QueueStatis(int s) {
        elemen = new Object[s];
        front = 0;
        rear = 0;
        size = 0;
    }

    public boolean enqueue(Object x) {
        if (size != elemen.length) {
            elemen[rear] = x;
            rear = (rear + 1) % elemen.length;
            size++;
            return true;
        }
        return false;
    }

    public Object dequeue() {
        if (!isEmpty()) {
            Object temp = elemen[front];
            front = (front + 1) % elemen.length;
            size--;
            return temp;
        } else {
            throw new NoSuchElementException();
        }
    }

    public int size() {
        return size;
    }

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

Stack Dinamis

```java
package vscode.Modul11;

import java.util.NoSuchElementException;
import java.util.LinkedList;

public class Stack<T> {

    private LinkedList<T> stack;

    public Stack() {
        stack = new LinkedList<>();
    }

    public void push(T element) {
        stack.addFirst(element);
    }

    public T pop() {
        if (stack.isEmpty()) {
            throw new NoSuchElementException(s:"Stack is empty");
        }
        return stack.removeFirst();
    }

    public int size() {
        return stack.size();
    }

    public boolean isEmpty() {
        return stack.isEmpty();
    }
}
```

D. OUTPUT

Queue

```
14
24
8
```

Stack

```
30
20
10
```

E. ANALISIS

a. LinkedList

- Contructor LinkedList
  Digunakan untuk menjuk ke dirinya sendiri untuk next dan prev.
- Method addBefore
  Mengatur referensi next dan prev dari simpul baru dan simpul sekitar.
- Method remove
  Mengatur ulang referensi next dan prev dari simpul sekitarnya. Jika linked list kosong, melempar pengecualian NoSuchElementException.
- Method print
  Mencetak semua elemen dalam linked list mulai dari simpul pertama hingga kembali ke kepala.
- Method addFirst
  Menyisipkan elemen baru di awal linked list dengan menggunakan addBefore
- Method addLast
  Menyisipkan elemen baru di akhir linked list dengan menggunakan addBefore
- Method removeFirst
  Menghapus dan mengembalikan elemen pertama dalam linked list dengan menggunakan remove
- Method removeLast
  Menghapus dan mengembalikan elemen terakhir dalam linked list dengan menggunakan remove
- Method search
  Mencari elemen dalam linked list. Jika ditemukan, mengembalikan referensi ke simpul tersebut, jika tidak, mengembalikan null.
- Method size
  Mengembalikan ukuran (jumlah elemen) dari linked list.
- Method isEmpty
  Memeriksa apakah linked list kosong.
- Method toString
  Mengembalikan representasi string dari linked list dengan semua elemen yang dipisahkan oleh spasi.