LAPORAN

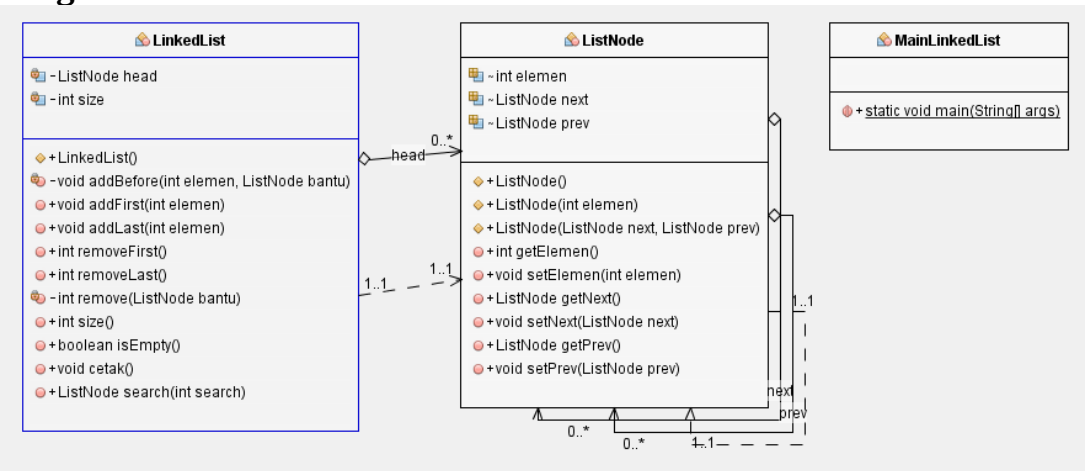
Struktur Data Linear
Praktikum 10 : Linked List

NAMA : Johanes Yogtan Wicaksono Raharja

NIM : 215314105

Program Studi INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS SANATA DHARMA

Diagram UML :



Linked List

1. ListNode

- Screenshot Listing Program dan Penjelasan Tiap Metode

```
package Modul9_SBGKB;
public class ListNode {
    int elemen;
    ListNode next;
    ListNode prev;

    //Konstruktor tanpa parameter, untuk memberi nilai default namun tak diisi
    public ListNode() {
    }
    /*Konstruktor kelas yang ada parameter, paramater next, prev maupun elemen
    akan diisikan di masing masing atribut*/
    public ListNode(int elemen) {
        this.elemen = elemen;
    }

    public ListNode(ListNode next, ListNode prev) {
        this.next = next;
        this.prev = prev;
    }
    /*Elemen disini digunakan untuk pembentukan data yang akan ditampilkan
    maupun dihapuskan dengan get untuk menampilkannya dan set untuk menyimpannya
    public int getElemen() {
        return elemen;
    }

    public void setElemen(int elemen) {
        this.elemen = elemen;
    }
    /*Next disini digunakan untuk menunjuk pada node setelahnya dengan get untuk
    menampilkan dan set untuk menyimpan*/
    public ListNode getNext() {
        return next;
    }

    public void setNext(ListNode next) {
        this.next = next;
    }
    /*Prev disini digunakan untuk menunjuk pada node sebelumnya dengan get untuk
    menampilkan dan set untuk menyimpan*/
    public ListNode getPrev() {
        return prev;
    }

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

2. LinkedList

- Screenshot Listing Program dan Penjelasan Tiap Metode

```
public class LinkedList {

    private ListNode head;
    private int size;

    /*Konstruktor kelas tanpa parameter berfungsi untuk memberi nilai default
    apabila tidak diisi(membaut sebuah kepala*/
    public LinkedList() {
        head = new ListNode();
        head.setNext(head);
        head.setPrev(head);
        size = 0;
    }
    /*AddBefore akan diprosaes ketika metode add akan dijalankan, metode ini aka
    membuat sebuah proses pembuatan objek, dilanjutkan dengan pemanggilan kelas
    listNode untuk proses next dan prevnya hingga pertambahan size ketika diisi*/
    private void addBefore(int elemen, ListNode bantu) {
        ListNode baru = new ListNode(elemen);
        baru.next = bantu;
        baru.prev = bantu.prev;
        bantu.prev.next = baru;
        bantu.prev = baru;
        size++;
    }
    /*Add first untuk menambah elemen di list pertama dam
    addLast untuk menambah elemen di list terakhir, keduaduanya akan
    di proses di metode addBefore*/
    public void addFirst(int elemen) {
        addBefore(elemen, head.getNext());
    }
    public void addLast(int elemen) {
        addBefore(elemen, head);
    }
    /*Add first untuk menghapus elemen di list pertama dam
    addLast untuk menghapus elemen di list terakhir, keduaduanya akan
    di proses di metode addBefore*/
    public int removeFirst() {
        return remove(head.getNext());
    }
    public int removeLast() {
        return remove(head.getPrev());
    }
    /*remove akan diproses ketika metode removeFirst dan last dijalankan, metode
    membaut sebuah proses if, ketika bantu sama dengan headnya maka dicegah
    erorrnya, dilanjutkan dengan pemanggilan kelas listNode untuk proses next
    prevnya hingga pengurangan size ketika dihapus dengan retrun elemen*/
    private int remove(ListNode bantu) {
        if(bantu == head)
            throw new NoSuchElementException();
        int elemen = bantu.getElemen();
```

```
        bantu.prev.next = bantu.next;
        bantu.next.prev = bantu.prev;
        bantu.next = null;
        bantu.prev = null;
        size--;
        return elemen;
    }
    //size untuk mengembalikan nilai size
    public int size() {
        return size;
    }
    /*isEmpty untuk mengembalikan data apabila tidak sama dengan percabangan*/
    public boolean isEmpty() {
        if (head == head.getNext()) {
            return true;
        } else {
            return false;
        }
    }
    /*cetak untuk menampilkan data yang telah di add maupun di remove sebelummya
    dengan head.nectnya itu dimasukkan ke bantu ListNode dilanjutkan perulangan
    apabila bantu tidak sama dengan head baru berhenti disitu akan ditampilkan
    data di bantu elemen*/
    public void cetak() {
        ListNode bantu = head.next;
        while (bantu != head) {
            System.out.print(bantu.elemen+" ");
            bantu = bantu.next;
        }
    }
    /*search untuk mencari data di dalam senarai yang telah dimasukkan datanya
    di dalam metode ini terdapat head.getNext yang akan dimasukkan di bantu di
    listNode, dilanjutkan perulangan while apabil bantu tidak sama dengan head
    barulah berhenti, didalam perulangan terdaopat juga percabangan yang
    jikalau ketemu akan menampilkan kalimat ketemu.
    */
    public ListNode search(int search) {
        ListNode bantu = head.getNext();
        while (bantu != head) {
            if (bantu.getElemen() == search) {
                System.out.println("Angka Ada di Senarai");
                return bantu;
            } else {
                bantu = bantu.getNext();
            }
        }
        return null;
    }
}
```

Implementasi

- Screenshot kelas main

```
package Modul9_SBGKB;
import java.util.NoSuchElementException;
import java.util.Scanner;
public class output {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        LinkedList senarai = new LinkedList();

        System.out.println("Model Senarai ADD");
        senarai.addFirst(8);
        senarai.cetak();
        System.out.println("");
        senarai.addFirst(15);
        senarai.cetak();

        System.out.println("");
        senarai.addLast(26);
        senarai.cetak();

        System.out.println("");
        senarai.addLast(14);
        senarai.cetak();

        System.out.print("\nCari angka di senarai : ");
        int cari = sc.nextInt();
        senarai.search(cari);
        try {
            System.out.println("-------------------------");
            System.out.print("Model Senarai Remove");
            senarai.removeLast();
            System.out.println("");
            senarai.cetak();
            senarai.removeLast();
            System.out.println("");
            senarai.cetak();

            senarai.removeFirst();
            System.out.println("");
            senarai.cetak();

            senarai.removeFirst();
            System.out.println("");
            senarai.cetak();

            senarai.removeFirst();
            System.out.println("");
            senarai.cetak();

        } catch (NoSuchElementException e) {
            System.out.println("LinkedList Kosong!");

        }
    }
}
```

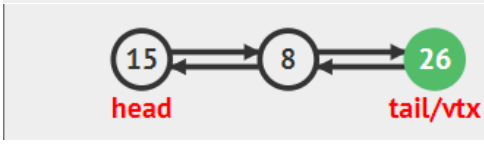
- Screenshot output

```
run:
Model Senarai ADD
8
15 8
15 8 26
15 8 26 14
Cari angka di senarai : 8
Angka Ada di Senarai
--------------------------
Model Senarai Remove
15 8 26
15 8
8
LinkedList Kosong!
BUILD SUCCESSFUL (total time: 4 seconds)
```

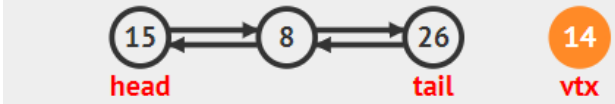
- Ilustrasi dari (Doubly) Linked List
  AddFirst(8)
  Buat elemen baru disini 8, setelah itu melakukan pengecekan head yang isinya null, setelah itu masukin elemen ke head, tailnya juga berada di head

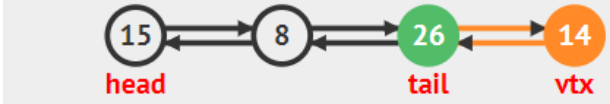
  AddFisrt(15)
  Buat elemen baru disini 15, setelah itu elemen, next akan menunjuk ke head dan prev akan menunjuk ke elemen hingga head menunjuk ke elemen untuk dimasukkan

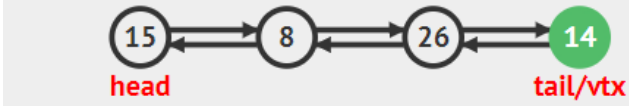
AddLast(26)
Buat elemen baru disini 26, setelah itu tail next akan menunjuk ke elemen dan prex menunjuk kembali ke tail, tail menunjuk ke elemen untuk dimasukkan



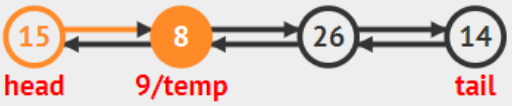
AddLast(14)
Buat elemen baru disini 14, setelah itu tail next akan menunjuk ke elemen dan prex menunjuk kembali ke tail, tail menunjuk ke elemen untuk dimasukkan



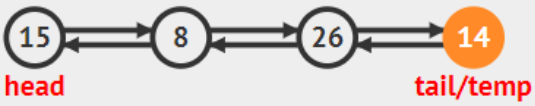


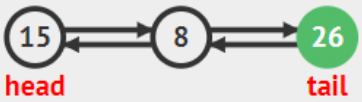


Search(cari)
Kita akan mencari 9 yang berawal dari head, setelah itu dibandingkan 8 dengan 15 ternyata tidak cocok dan dilanjutkan, setelah itu next menuju ke elemen berikutya untuk melakukan pencarian ternyata ditemukan 8

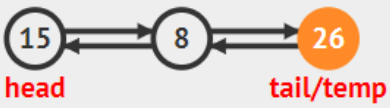


removeLast()
elemen diatur untuk berada di tailnya, setelah itu tail akan membentuk tail.prev, lalu membentuk nexct dan tailnya dijadikan null





removeLast()
elemen diatur untuk berada di tailnya, setelah itu tail akan membentuk tail.prev, lalu membentuk nexct dan tailnya dijadikan null



removeFirst()
head akan menjadi nect elemen, setelah itu head akan menuju next ke elemen, dilanjutkan dengan menghapus head dan di prev ke head

removeFirst()

removeFirst()