POLITEKNIK NEGERI MALANG

TEKNOLOGI INFORMASI

TEKNIK INFORMATIKA



#### Nama : Muhammad Nuril Huda

#### Kelas : TI-1A

#### No : 19

#### Mata Kuliah : Algoritma dan Struktur Data

## **2.1 Pembuatan Single Linked List**

**2.1.1 Kode Program**

* Kode Program Mahasiswa19

public class Mahasiswa19 {

    String nim;

    String nama;

    String kelas;

    Double ipk;

    Mahasiswa19(){

    }

    Mahasiswa19 (String nm, String name, String kls, double ip){

        nim = nm;

        nama = name;

        kelas = kls;

        ipk = ip;

    }

    void tampilInformasi(){

        System.out.println(nim + "\t" + nama + "\t" + kelas + "\t" + ipk);

    }

}

* Kode Program Node19

public class Node19 {

    Mahasiswa19 data;

    Node19 next;

    public Node19(Mahasiswa19 data, Node19 next){

        this.data = data;

        this.next = next;

    }

}

* Kode Program SingleLinkedList19

public class SingleLinkedList19 {

    Node19 head;

    Node19 tail;

    boolean isEmpty(){

        return(head == null);

    }

    public void print(){

        if(!isEmpty()){

            Node19 tmp = head;

            System.out.println("Isi Linked List:\t");

            while (tmp != null){

                tmp.data.tampilInformasi();

                tmp = tmp.next;

            }

            System.out.println("");

        } else {

            System.out.println("Linked List Kosong");

        }

    }

    public void addFirst(Mahasiswa19 input){

        Node19 ndInput = new Node19(input,null);

        if (isEmpty()){

            head = ndInput;

            tail = ndInput;

        } else {

            ndInput.next = head;

            head = ndInput;

        }

    }

public void addLast(Mahasiswa19 input){

        Node19 ndInput = new Node19(input, null);

        if(isEmpty()){

            head = ndInput;

            tail = ndInput;

        } else {

            tail.next = ndInput;

            tail = ndInput;

        }

    }

    public void insertAfter (String key, Mahasiswa19 input){

        Node19 ndInput = new Node19(input, null);

        Node19 temp = head;

        do {

            if (temp.data.nama.equalsIgnoreCase(key)){

                ndInput.next = temp.next;

                temp.next = ndInput;

                if(ndInput.next == null){

                    tail = ndInput;

                }

                break;

            }

            temp = temp.next;

        } while (temp != null);

    }

    public void insertAt (int index, Mahasiswa19 input){

        if (index < 0){

            System.out.println("Indeks Salah");

        } else if (index == 0){

            addFirst(input);

        } else {

            Node19 temp = head;

            for (int i = 0; i<index-1; i++){

                temp = temp.next;

            }

            temp.next = new Node19(input, temp.next);

            if (temp.next.next == null) {

                tail = temp.next;

            }

        }

    }

}

* Kode Program SLL Main19

public class SLLMain19 {

    public static void main(String[] args) {

        SingleLinkedList19 sll = new SingleLinkedList19();

        Mahasiswa19 mhs1 = new Mahasiswa19("11111", "Andi", "1A", 3.75);

        Mahasiswa19 mhs2 = new Mahasiswa19("22222", "Budi", "1B", 3.60);

        Mahasiswa19 mhs3 = new Mahasiswa19("33333", "Dirga", "1C", 3.85);

        Mahasiswa19 mhs4 = new Mahasiswa19("44444", "Dina", "1D", 3.90);

        // Cetak isi awal (kosong)

        sll.print();

        // Tambah mhs4 ke depan

        sll.addFirst(mhs4);

        // Cetak

        sll.print();

        // Tambah mhs1 ke belakang

        sll.addLast(mhs1);

        // Cetak

        sll.print();

        // Sisipkan mhs3 setelah "Dirga"

        sll.insertAfter("Dirga", mhs3);

        // Sisipkan mhs2 di index ke-2

        sll.insertAt(2, mhs2);

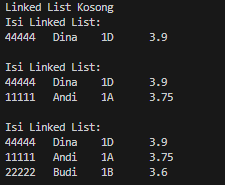
         // Cetak hasil akhir

        sll.print();

    }

}

**2.1.2 Hasil Kode Program**



**2.1.3 Pertanyaan**

1. Mengapa hasil compile kode program di baris pertama menghasilkan “Linked List Kosong”?

* Karena pada saat method print() pertama kali dipanggil, belum ada data apapun yang ditambahkan ke dalam SingleLinkedList19.

1. Jelaskan kegunaan variable temp secara umum pada setiap method!

* Variabel temp digunakan sebagai penunjuk (pointer) untuk menelusuri node satu per satu dalam linked list, tanpa mengubah head asli.

1. Lakukan modifikasi agar data dapat ditambahkan dari keyboard!

import java.util.Scanner;

public class SLLMain19 {

    public static void main(String[] args) {

        Scanner sc = new Scanner (System.in);

        SingleLinkedList19 sll = new SingleLinkedList19();

        System.out.print("Masukkan jumlah mahasiswa: ");

        int jumlah = sc.nextInt();

        sc.nextLine();

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

            System.out.println("\nMahasiswa ke-" + (i + 1));

            System.out.print("NIM   : ");

            String nim = sc.nextLine();

            System.out.print("Nama  : ");

            String nama = sc.nextLine();

            System.out.print("Kelas : ");

            String kelas = sc.nextLine();

            System.out.print("IPK   : ");

            double ipk = sc.nextDouble();

            sc.nextLine();

            Mahasiswa19 mhs = new Mahasiswa19(nim, nama, kelas, ipk);

            sll.addLast(mhs);

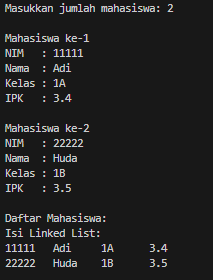
        }

        System.out.println("\nDaftar Mahasiswa:");

        sll.print();

    }

}



## **2.2 Modifikasi Elemen pada Single Linked List**

**2.2.1 Kode Program**

* Kode Program SingleLinkedList19

public class SingleLinkedList19 {

    Node19 head;

    Node19 tail;

    boolean isEmpty(){

        return(head == null);

    }

    public void print(){

        if(!isEmpty()){

            Node19 tmp = head;

            System.out.println("Isi Linked List:\t");

            while (tmp != null){

                tmp.data.tampilInformasi();

                tmp = tmp.next;

            }

            System.out.println("");

        } else {

            System.out.println("Linked List Kosong");

        }

    }

public void addFirst(Mahasiswa19 input){

        Node19 ndInput = new Node19(input,null);

        if (isEmpty()){

            head = ndInput;

            tail = ndInput;

        } else {

            ndInput.next = head;

            head = ndInput;

        }

    }

    public void addLast(Mahasiswa19 input){

        Node19 ndInput = new Node19(input, null);

        if(isEmpty()){

            head = ndInput;

            tail = ndInput;

        } else {

            tail.next = ndInput;

            tail = ndInput;

        }

    }

    public void insertAfter (String key, Mahasiswa19 input){

        Node19 ndInput = new Node19(input, null);

        Node19 temp = head;

        do {

            if (temp.data.nama.equalsIgnoreCase(key)){

                ndInput.next = temp.next;

                temp.next = ndInput;

                if(ndInput.next == null){

                    tail = ndInput;

                }

                break;

            }

            temp = temp.next;

        } while (temp != null);

    }

public void insertAt (int index, Mahasiswa19 input){

        if (index < 0){

            System.out.println("Indeks Salah");

        } else if (index == 0){

            addFirst(input);

        } else {

            Node19 temp = head;

            for (int i = 0; i<index-1; i++){

                temp = temp.next;

            }

            temp.next = new Node19(input, temp.next);

            if (temp.next.next == null) {

                tail = temp.next;

            }

        }

    }

    public void getData (int index) {

        Node19 tmp = head;

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

            tmp = tmp.next;

        }

        tmp.data.tampilInformasi();

    }

    public int indexOf (String key){

        Node19 tmp = head;

        int index = 0;

        while (tmp != null && !tmp.data.nama.equalsIgnoreCase(key)) {

            tmp = tmp.next;

            index++;

        }

        if (tmp == null){

            return -1;

        } else {

            return index;

        }

    }

public void removeFirst(){

        if (isEmpty()){

            System.out.println("Linked List Masih Kosong, Tidak Dapat Dihapus!");

        } else if (head == tail) {

            head = tail = null;

        } else {

            head = head.next;

        }

    }

    public void removeLast(){

        if (isEmpty()){

            System.out.println("Linked List Masih Kosong, Tidak Dapat Dihapus!");

        } else if (head == tail){

            head = tail = null;

        } else {

            Node19 temp = head;

            while (temp.next != tail) {

                temp = temp.next;

            }

            temp.next = null;

            tail = temp;

        }

    }

public void remove(String key) {

        if (isEmpty()) {

            System.out.println("Linked List Masih Kosong, Tidak Dapat Dihapus!");

        } else {

            Node19 temp = head;

            while (temp != null) {

                if ((temp.data.nama.equalsIgnoreCase(key)) && (temp == head)) {

                    this.removeFirst();

                    break;

                } else if (temp.data.nama.equalsIgnoreCase(key)) {

                    temp.next = temp.next.next;

                    if (temp.next == null) {

                    tail = temp;

                    }

                 break;

                }

                temp = temp.next;

            }

        }

    }

    public void removeAt(int index) {

        if (index == 0) {

            removeFirst();

        } else {

            Node19 temp = head;

            for (int i = 0; i < index - 1; i++) {

                temp = temp.next;

            }

            temp.next = temp.next.next;

            if (temp.next == null) {

                tail = temp;

            }

        }

    }

}

* Kode Program SLLMain19

import java.util.Scanner;

public class SLLMain19 {

    public static void main(String[] args) {

        Scanner sc = new Scanner (System.in);

        SingleLinkedList19 sll = new SingleLinkedList19();

        System.out.print("Masukkan jumlah mahasiswa: ");

        int jumlah = sc.nextInt();

        sc.nextLine();

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

            System.out.println("\nMahasiswa ke-" + (i + 1));

            System.out.print("NIM   : ");

            String nim = sc.nextLine();

            System.out.print("Nama  : ");

            String nama = sc.nextLine();

            System.out.print("Kelas : ");

            String kelas = sc.nextLine();

            System.out.print("IPK   : ");

            double ipk = sc.nextDouble();

            sc.nextLine();

            Mahasiswa19 mhs = new Mahasiswa19(nim, nama, kelas, ipk);

            sll.addLast(mhs);

        }

        System.out.println("\nDaftar Mahasiswa:");

        sll.print();

        System.out.println("Data Index 1 : ");

        sll.getData(1);

        System.out.println();

        System.out.println("Data Mahasiswa an Nuril Berada Pada Index : " +sll.indexOf("Nuril"));

        System.out.println();

        sll.removeFirst();

        sll.removeLast();

        sll.print();

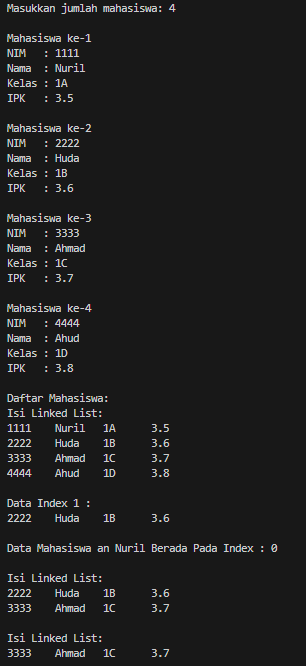
        sll.removeAt(0);

        sll.print();

    }

}

**2.2.2 Hasil Kode Program**

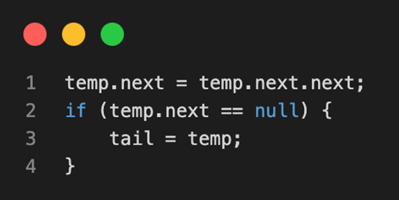


**2.2.3 Pertanyaan**

1. Mengapa digunakan keyword break pada fungsi remove? Jelaskan!

* Break digunakan untuk menghentikan proses pencarian begitu data yang dicari telah ditemukan dan dihapus. Tanpa break, program akan terus melanjutkan perulangan meskipun node yang dimaksud sudah dihapus

1. Jelaskan kegunaan kode dibawah pada method remove



* Kode temp.next = temp.next.next; digunakan untuk menghapus sebuah node pada linked list. Caranya adalah dengan melewatkan node yang ingin dihapus. Jadi, jika temp adalah node sebelum node yang ingin dihapus, maka baris ini akan menyambungkan temp langsung ke node setelah node yang dihapus, sehingga node yang dihapus tidak lagi menjadi bagian dari daftar. Selanjutnya kondisi if berfungsi untuk memastikan bahwa jika node yang dihapus adalah node terakhir atau tail, maka kita harus memperbarui tail agar menunjuk ke node sebelumnya, yaitu temp. Hal ini penting supaya tail tetap menunjuk ke ujung linked list yang benar, dan tidak ke node yang sudah dihapus. Jika tidak dilakukan, bisa menyebabkan kesalahan saat menambahkan atau mengakses data di akhir list.

## **Tugas**

**Kode Program**

* Kode Program Antrian19

public class Antrian19 {

    String nim;

    String nama;

    String prodi;

    String kelas;

    Antrian19 (String nim, String nama, String prodi, String kelas){

        this.nim = nim;

        this.nama = nama;

        this.prodi = prodi;

        this.kelas = kelas;

    }

    void tampilInformasi(){

        System.out.println(nim + "\t" + nama + "\t" + prodi + "\t" + kelas);

    }

}

* Kode Program AntrianNode19

public class AntrianNode19 {

    Antrian19 data;

    AntrianNode19 next;

    AntrianNode19 (Antrian19 data, AntrianNode19 next){

        this.data = data;

        this.next = next;

    }

}

* Kode Program AntrianMethod19

public class AntrianMedthod19 {

    AntrianNode19 head;

    AntrianNode19 tail;

    int jumlahAntrian = 0;

    int kapasitasMaksimal = 4;

    boolean isEmpty(){

        return(head == null);

    }

    boolean isFull() {

        return jumlahAntrian >= kapasitasMaksimal;

    }

    public void print(){

        if (!isEmpty()){

            AntrianNode19 tmp = head;

            System.out.println("Daftar Antrian:");

            System.out.println("NIM\tNAMA\tPRODI\tKELAS");

            while (tmp != null){

                tmp.data.tampilInformasi();

                tmp = tmp.next;

            }

            System.out.println("");

        } else {

            System.out.println("Antrian Kosong");

        }

    }

public void enqueue (Antrian19 input){

        if (isFull()) {

            System.out.println("Antrian Sudah Penuh. Tidak Dapat Menambahkan " + input.nama);

            return;

        }

        AntrianNode19 ndInput = new AntrianNode19(input, null);

        if (isEmpty()){

            head = ndInput;

            tail = ndInput;

        } else {

            tail.next = ndInput;

            tail = ndInput;

        }

        jumlahAntrian++;

        System.out.println(input.nama + " Berhasil Ditambahkan Ke Antrian");

    }

    public void dequeue (){

        if (isEmpty()){

            System.out.println("Antrian Masih Kosong");

        } else {

            String namaDipanggil = head.data.nama;

            if (head == tail){

                head = tail = null;

            } else {

                head = head.next;

            }

            jumlahAntrian--;

            System.out.println(namaDipanggil + " Dipanggil");

        }

    }

public void removeAll (){

        if (isEmpty()) {

            System.out.println("Antrian sudah kosong.");

        } else {

            head = tail = null;

            jumlahAntrian = 0;

            System.out.println("Semua antrian telah dikosongkan.");

        }

    }

    public void lihatAntrianTerdepan (){

        if (isEmpty()){

            System.out.println("Antrian Masih Kosong");

        } else {

            System.out.println("Data Antrian Paling Depan: ");

            System.out.println("NIM\tNAMA\tPRODI\tKELAS");

            head.data.tampilInformasi();

        }

    }

    public void lihatAntrianTerbelakang (){

        if (isEmpty()){

            System.out.println("Antrian Masih Kosong");

        } else {

            System.out.println("Data Antrian Paling Belakang: ");

            System.out.println("NIM\tNAMA\tPRODI\tKELAS");

            tail.data.tampilInformasi();

        }

    }

    public int jumlahAntrian(){

        return jumlahAntrian;

    }

}

* Kode Program AntrianMain19

import java.util.Scanner;

public class AntrianMain {

    public static void main(String[] args) {

        Scanner sc = new Scanner (System.in);

        AntrianMedthod19 antrn = new AntrianMedthod19();

        int pilihan;

        do {

            System.out.println("\n=== Menu Antrian Layanan Akademik ===");

            System.out.println("1. Tambah Antrian");

            System.out.println("2. Panggil Antrian");

            System.out.println("3. Lihat Semua Antrian");

            System.out.println("4. Lihat Antrian Terdepan");

            System.out.println("5. Lihat Antrian Paling Akhir");

            System.out.println("6. Jumlah Antrian");

            System.out.println("7. Kosongkan Antrian");

            System.out.println("0. Keluar");

            System.out.print("Pilih Menu: ");

            pilihan = sc.nextInt();

            sc.nextLine();

switch (pilihan) {

                case 1:

                    System.out.print("NIM: ");

                    String nim = sc.nextLine();

                    System.out.print("Nama: ");

                    String nama = sc.nextLine();

                    System.out.print("Prodi: ");

                    String prodi = sc.nextLine();

                    System.out.print("Kelas: ");

                    String kelas = sc.nextLine();

                    Antrian19 an = new Antrian19(nim, nama, prodi, kelas);

                    antrn.enqueue(an);

                    break;

                case 2:

                    antrn.dequeue();

                    break;

                case 3:

                    antrn.print();

                    break;

                case 4:

                    antrn.lihatAntrianTerdepan();

                    break;

                case 5:

                    antrn.lihatAntrianTerbelakang();

                    break;

                case 6:

                    System.out.println("Jumlah Antrian Saat Ini: " + antrn.jumlahAntrian());

                    break;

                case 7:

                    antrn.removeAll();

                    break;

                case 0:

                    System.out.println("Terima Kasih");

                    break;

                default:

                    System.out.println("Pilihan Tidak Valid");

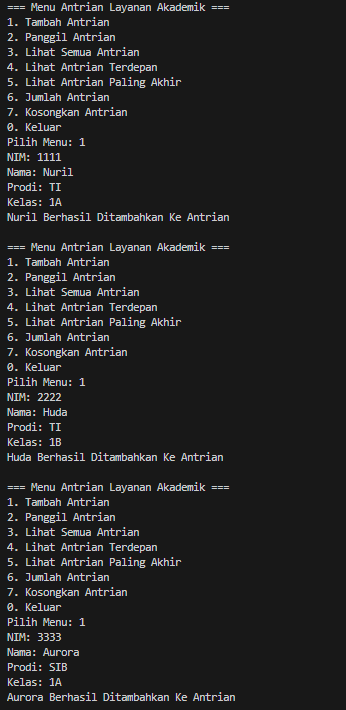
            }

        } while (pilihan!=0);

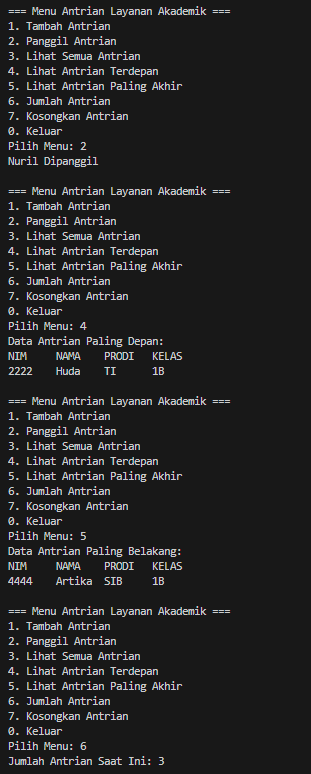
    }

}

**Hasil Kode Program**









## **Link Github:**