

LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA
JOBSHEET 11 LINKED LIST



RANI MIFTAHUS SAÁDAH

244107020057

TI_1E

PROGRAM STUDI D_IV TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
PRAKTIKUM 2025

1.1 Pembuatan Single Linked List

Mahasiswa25.java

```
public class Mahasiswa25 {
    String nim;
    String nama;
    String kelas;
    double ipk;

    public Mahasiswa25() {

    }

    public Mahasiswa25(String nim, String nama, String kelas,
double ipk) {
        this.nim = nim;
        this.nama = nama;
        this.kelas = kelas;
        this.ipk = ipk;
    }

    public void tampilInformasi() {
        System.out.println(nama + "\t " + nim + "\t " + kelas +
"\t" + ipk );
    }
}
```

Node25.java

```
public class Node25 {

    Mahasiswa25 data;
    Node25 next;

    public Node25(Mahasiswa25 data, Node25 next) {
        this.data = data;
        this.next = next;
    }
}
```

SingleLinkedList25.java

```
public class SingleLinkedList25 {
    Node25 head;
    Node25 tail;

    boolean isEmpty() {
        return (head == null);
    }

    public void print() {
        if (!isEmpty()) {
            Node25 tmp = head;
            System.out.println("Isi Linked List:");
            while (tmp != null) {
                tmp.data.tampilInformasi();
                tmp = tmp.next;
            }
            System.out.println("");
        } else {
            System.out.println("Linked List kosong");
        }
    }
}
```

```

    }

    public void addFirst(Mahasiswa25 input) {
        Node25 ndInput = new Node25(input, null);
        if (isEmpty()) {
            head = ndInput;
            tail = ndInput;
        } else {
            ndInput.next = head;
            head = ndInput;
        }
    }

    public void addLast(Mahasiswa25 input ) {
        Node25 ndInput = new Node25(input, null);
        if (isEmpty()) {
            head = ndInput;
            tail = ndInput;
        } else {
            tail.next = ndInput;
            tail = ndInput;
        }
    }

    public void insertAfter(String key, Mahasiswa25 input) {
        Node25 ndInput = new Node25(input, null);
        Node25 tmp = head;
        do {
            if (tmp.data.nama.equalsIgnoreCase(key)) {
                ndInput.next = tmp.next;
                tmp.next = ndInput;
                if (ndInput.next == null) {
                    tail = ndInput;
                }
                break;
            }
            tmp = tmp.next;
        } while (tmp != null);
    }

    public void insertArt(int index, Mahasiswa25 input) {
        if (index < 0 ) {
            System.out.println("indeks salah");
        } else if (index == 0){
            addFirst(input);
        } else {
            Node25 tmp = head;
            for (int i = 0; i < index - 1; i++) {
                tmp = tmp.next;
            }
            tmp.next = new Node25(input, tmp.next);
            if (tmp.next.next == null) {
                tail = tmp.next;
            }
        }
    }
}

```

SLLMain25.java

```
public class SLLMain25 {
    public static void main(String[] args) {
        SingleLinkedList25 sll = new SingleLinkedList25();

        Mahasiswa25 mhs1 = new Mahasiswa25("24212200",
        "Alvaro", "1A", 4.0);
        Mahasiswa25 mhs2 = new Mahasiswa25("23212201", "Bimon",
        "2B", 3.8);
        Mahasiswa25 mhs3 = new Mahasiswa25("22212202",
        "Cintia", "3C", 3.5);
        Mahasiswa25 mhs4 = new Mahasiswa25("21212225", "Dirga",
        "4D", 3.6);

        sll.print();
        sll.addFirst(mhs4);
        sll.print();
        sll.addLast(mhs1);
        sll.print();
        sll.insertAfter("Dirga", mhs3);
        sll.insertArt(2, mhs2);
        sll.print();
    }
}
```

1.2 Verifikasi

```
Linked List kosong
Isi Linked List:
Dirga    21212225    4D    3.6

Isi Linked List:
Dirga    21212225    4D    3.6
Alvaro   24212200    1A    4.0

Isi Linked List:
Dirga    21212225    4D    3.6
Cintia   22212202    3C    3.5
Bimon    23212201    2B    3.8
Alvaro   24212200    1A    4.0
```

1.3 Pertanyaan

1. Mengapa hasil compile kode program di baris pertama menghasilkan “Linked List Kosong”?
Karena saat program dijalankan pertama kali, belum ada node yang ditambahkan ke dalam linked list (atribut head masih null, sehingga saat proses traverse dilakukan, tidak ada node untuk ditampilkan)
2. Jelaskan kegunaan variable temp secara umum pada setiap method!
***temp** digunakan sebagai penunjuk sementara (pointer) ke node yang sedang diproses saat melakukan traversing atau manipulasi linked list. temp, bisa digunakan untuk berpindah-pindah node tanpa mengubah nilai asli head atau tail.*

3. Lakukan modifikasi agar data dapat ditambahkan dari keyboard!

Kode Program untuk SLLMain25.java

```
import java.util.Scanner;

public class SLLMain25 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        SingleLinkedList25 list = new SingleLinkedList25();

        int jumlah = 2;
        System.out.println("jumlah data mahasiswa: " +
jumlah);
        for (int i = 0; i < jumlah; i++) {
            System.out.println("Data Mahasiswa ke-" + (i +
1));

            System.out.print("Nama : ");
            String nama = sc.nextLine();
            System.out.print("NIM : ");
            String nim = sc.nextLine();
            System.out.print("Kelas : ");
            String kelas = sc.nextLine();
            System.out.print("IPK : ");
            double ipk = sc.nextDouble();
            sc.nextLine();

            Mahasiswa25 mhs = new Mahasiswa25(nim, nama,
kelas, ipk);
            list.addLast(mhs);
        }
        System.out.println("\n === Linked LIST ===");
        list.print();
    }
}
```

```
jumlah data mahasiswa: 2
Data Mahasiswa ke-1
Nama : zaraa
NIM : 244102
Kelas : 1E
IPK : 3,8
Data Mahasiswa ke-2
Nama : angga
NIM : 244103
Kelas : 1E
IPK : 4,0

=== Linked LIST ===
Isi Linked List:
zaraa    244102  1E    3.8
angga    244103  1E    4.0
```

2.1 Modifikasi Elemen pada Single Linked List

SingleLinkedList25.java

```
public class SingleLinkedList25 {
    Node25 head;
    Node25 tail;

    boolean isEmpty() {
        return (head == null);
    }

    public void print() {
        if (!isEmpty()) {
            Node25 tmp = head;
            System.out.println("Isi Linked List:");
            while (tmp != null) {
                tmp.data.tampilInformasi();
                tmp = tmp.next;
            }
            System.out.println("");
        } else {
            System.out.println("Linked List kosong");
        }
    }

    public void addFirst(Mahasiswa25 input) {
        Node25 ndInput = new Node25(input, null);
        if (isEmpty()) {
            head = ndInput;
            tail = ndInput;
        } else {
            ndInput.next = head;
            head = ndInput;
        }
    }

    public void addLast(Mahasiswa25 input ) {
        Node25 ndInput = new Node25(input, null);
        if (isEmpty()) {
            head = ndInput;
            tail = ndInput;
        } else {
            tail.next = ndInput;
            tail = ndInput;
        }
    }

    public void insertAfter(String key, Mahasiswa25 input) {
        Node25 ndInput = new Node25(input, null);
        Node25 tmp = head;
        do {
            if (tmp.data.nama.equalsIgnoreCase(key)) {
                ndInput.next = tmp.next;
                tmp.next = ndInput;
                if (ndInput.next == null) {
                    tail = ndInput;
                }
            }
        } while (tmp.next != null);
        break;
    }
}
```

```

        }
        tmp = tmp.next;
    }while (tmp != null);
}

public void insertArt(int index, Mahasiswa25 input) {
    if (index < 0 ) {
        System.out.println("indeks salah");
    } else if (index == 0){
        addFirst(input);
    } else {
        Node25 tmp = head;
        for (int i = 0; i < index - 1; i++) {
            tmp = tmp.next;
        }
        tmp.next = new Node25(input, tmp.next);
        if (tmp.next.next == null) {
            tail = tmp.next;
        }
    }
}

public void getData(int index) {
    Node25 tmp = head;
    for (int i = 0; i < index; i++) {
        tmp = tmp.next;
    }
    tmp.data.tampilInformasi();
}

public int indexOf(String key){
    Node25 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 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 kosong, tidak dapat
dihapus");
    } else if (head == tail) {
        head = tail = null;
    }
}

```

```

        } else {
            Node25 tmp = head;
            while (tmp.next != tail) {
                tmp = tmp.next;
            }
            tmp.next = null;
            tail = tmp;
        }
    }

    public void remove(String key) {
        if (isEmpty()) {
            System.out.println("Linked List kosong, tidak dapat
dihapus");
        } else {
            Node25 tmp = head;
            while (tmp != null) {
                if ((tmp.data.nama.equalsIgnoreCase(key)) &&
                    (tmp == head)) {
                    removeFirst();
                    break;
                } else if
                    (tmp.data.nama.equalsIgnoreCase(key)) {
                    tmp.next = tmp.next.next;
                    if (tmp.next == null) {
                        tail = tmp;
                    }
                    break;
                }
                tmp = tmp.next;
            }
        }
    }

    public void removeAt(int index) {
        if (index == 0) {
            removeFirst();
        } else {
            Node25 tmp = head;
            for (int i = 0; i < index - 1; i++) {
                tmp = tmp.next;
            }
            tmp.next = tmp.next.next;
            if (tmp.next == null) {
                tail = tmp;
            }
        }
    }
}

```

SLLMain25.java

```

public class SLLMain25 {
    public static void main(String[] args) {
        SingleLinkedList25 sll = new SingleLinkedList25();

        Mahasiswa25 mhs1 = new Mahasiswa25("24212200", "Alvaro", "1A",
4.0);
        Mahasiswa25 mhs2 = new Mahasiswa25("23212201", "Bimon", "2B",
3.8);
    }
}

```



```

        Mahasiswa25 mhs3 = new Mahasiswa25("22212202", "Cintia", "3C",
3.5);
        Mahasiswa25 mhs4 = new Mahasiswa25("21212203", "Dirga", "4D",
3.6);

        sll.print();
        sll.addFirst(mhs4);
        sll.print();
        sll.addLast(mhs1);
        sll.print();
        sll.insertAfter("Dirga", mhs3);
        sll.insertArt(2, mhs2);
        sll.print();

        System.out.println("data index 1 :");
        sll.getData(1);
        System.out.println("data mahasiswa an Bimon berada pada index :"+ sll.indexOf("Bimon"));
        System.out.println();
        sll.removeFirst();
        sll.removeLast();
        sll.print();
        sll.removeAt(0);
        sll.print();
    }
}

```

2.2 Verifikasi

```

Linked List kosong
Isi Linked List:
Dirga    21212203      4D      3.6

Isi Linked List:
Dirga    21212203      4D      3.6
Alvaro   24212200      1A      4.0

Isi Linked List:
Dirga    21212203      4D      3.6
Cintia   22212202      3C      3.5
Bimon    23212201      2B      3.8
Alvaro   24212200      1A      4.0

data index 1 :
Cintia   22212202      3C      3.5
data mahasiswa an Bimon berada pada index :2

Isi Linked List:
Cintia   22212202      3C      3.5
Bimon    23212201      2B      3.8

Isi Linked List:
Bimon    23212201      2B      3.8

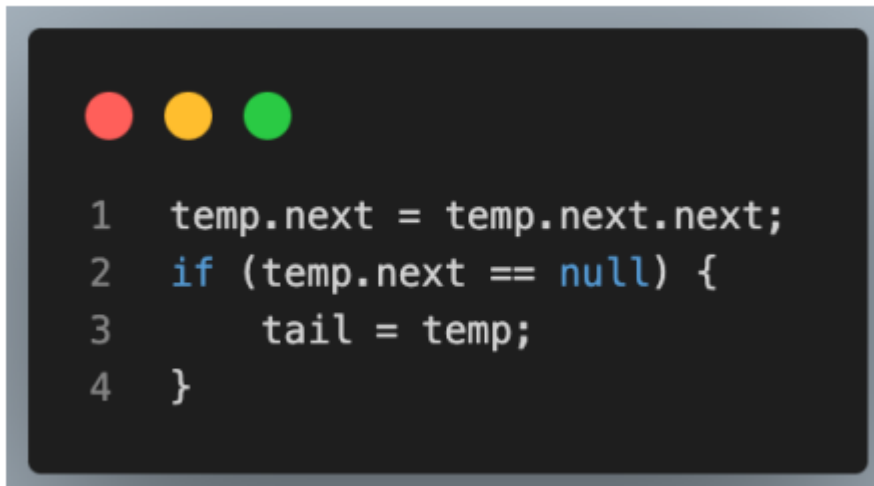
```

2.3 Pertanyaan

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

break digunakan untuk menghentikan perulangan ketika node yang ingin dihapus telah ditemukan. break memastikan bahwa hanya satu node pertama yang ditemukan akan dihapus, lalu program keluar dari loop.

2. Jelaskan kegunaan kode dibawah pada method remove



```
1  temp.next = temp.next.next;
2  if (temp.next == null) {
3      tail = temp;
4  }
```

- *Temp.next = temp.next.next;*
 - *ini menghapus node setelah temp, dengan cara: mengarahkan temp.next ke node setelah node yang akan dihapus. Node yang "dilewati" tidak lagi direferensikan dan akan otomatis dihapus oleh garbage collector.*
- *If (temp.next == null)*
 - *mengecek apakah node yang dihapus adalah node terakhir (tail).*
- *Tail = temp;*
 - *jika benar, maka update pointer tail ke node temp, karena sekarang temp adalah node terakhir dalam list.*

3.1 TUGAS

TugasMhs25.java

```
public class TugasMhs25 {
    String nama, nim;

    public TugasMhs25(String nama, String nim) {
        this.nama = nama;
        this.nim = nim;
    }

    public void tampil() {
        System.out.println("NIM: " + nim + ", Nama: " +
nama);
    }
}
```

TugasNode25.java

```
public class TugasNode25 {
    TugasMhs25 data;
    TugasNode25 next;

    public TugasNode25(TugasMhs25 data) {
        this.data = data;
        this.next = null;
    }
}
```

QueueLinkedList25.java

```
public class QueueLinkedList25 {
    TugasNode25 front, rear;
    int size = 0;

    public boolean isEmpty() {
        return front == null;
    }

    public void enqueue(TugasMhs25 mhs) {
        TugasNode25 newNode = new TugasNode25(mhs);
        if (isEmpty()) {
            front = rear = newNode;
        } else {
            rear.next = newNode;
            rear = newNode;
        }
        size++;
    }

    public void dequeue() {
        if (isEmpty()) {
            System.out.println("Antrian kosong!");
        } else {
            System.out.print("Memanggil: ");
            front.data.tampil();
            front = front.next;
            size--;
            if (front == null) rear = null;
        }
    }

    public void peekFront() {
        if (!isEmpty()) {
```



```

        String nama = sc.nextLine();
        System.out.print("Masukkan NIM: ");
        String nim = sc.nextLine();
        queue.enqueue(new TugasMhs25(nama,
nim));

        break;
    case 2:
        queue.dequeue();
        break;
    case 3:
        queue.peekFront();
        break;
    case 4:
        queue.peekRear();
        break;
    case 5:
        queue.printQueue();
        break;
    case 6:
        queue.countQueue();
        break;
    case 7:
        queue.clear();
        break;
    case 0:
        System.out.println("Terima kasih!");
        return;
    default:
        System.out.println("Menu tidak
tersedia.");
    }
}
}

```

3.2 VERIFIKASI OUTPUT

```
==== Menu Antrian Layanan ====
```

1. Tambah Antrian
2. Panggil Antrian
3. Lihat Antrian Terdepan
4. Lihat Antrian Terakhir
5. Lihat Semua Antrian
6. Jumlah Antrian
7. Kosongkan Antrian
0. Keluar

Pilih menu: 1

Masukkan Nama: rani

Masukkan NIM: 24412

```
==== Menu Antrian Layanan ====
```

1. Tambah Antrian
2. Panggil Antrian
3. Lihat Antrian Terdepan
4. Lihat Antrian Terakhir
5. Lihat Semua Antrian
6. Jumlah Antrian
7. Kosongkan Antrian
0. Keluar

Pilih menu: 1

Masukkan Nama: miftah

Masukkan NIM: 24413

```
==== Menu Antrian Layanan ====
```

1. Tambah Antrian
2. Panggil Antrian
3. Lihat Antrian Terdepan
4. Lihat Antrian Terakhir
5. Lihat Semua Antrian
6. Jumlah Antrian
7. Kosongkan Antrian
0. Keluar

Pilih menu: 5

Daftar Antrian:

NIM: 24412, Nama: rani

NIM: 24413, Nama: miftah

```
==== Menu Antrian Layanan ====
```

1. Tambah Antrian
2. Panggil Antrian
3. Lihat Antrian Terdepan
4. Lihat Antrian Terakhir
5. Lihat Semua Antrian
6. Jumlah Antrian
7. Kosongkan Antrian
0. Keluar

Pilih menu: 6

Jumlah antrian: 2

```
==== Menu Antrian Layanan ====
```

1. Tambah Antrian
2. Panggil Antrian
3. Lihat Antrian Terdepan
4. Lihat Antrian Terakhir
5. Lihat Semua Antrian
6. Jumlah Antrian
7. Kosongkan Antrian
0. Keluar

Pilih menu: 3

Antrian terdepan: NIM: 24412, Nama: rani

```
==== Menu Antrian Layanan ====
```

1. Tambah Antrian
2. Panggil Antrian
3. Lihat Antrian Terdepan
4. Lihat Antrian Terakhir
5. Lihat Semua Antrian
6. Jumlah Antrian
7. Kosongkan Antrian
0. Keluar

Pilih menu: 4

Antrian terakhir: NIM: 24413, Nama: miftah

```
==== Menu Antrian Layanan ====
1. Tambah Antrian
2. Panggil Antrian
3. Lihat Antrian Terdepan
4. Lihat Antrian Terakhir
5. Lihat Semua Antrian
6. Jumlah Antrian
7. Kosongkan Antrian
0. Keluar
Pilih menu: 2
Memanggil: NIM: 24412, Nama: nani

==== Menu Antrian Layanan ====
1. Tambah Antrian
2. Panggil Antrian
3. Lihat Antrian Terdepan
4. Lihat Antrian Terakhir
5. Lihat Semua Antrian
6. Jumlah Antrian
7. Kosongkan Antrian
0. Keluar
Pilih menu: 7
Antrian dikosongkan.

==== Menu Antrian Layanan ====
1. Tambah Antrian
2. Panggil Antrian
3. Lihat Antrian Terdepan
4. Lihat Antrian Terakhir
5. Lihat Semua Antrian
6. Jumlah Antrian
7. Kosongkan Antrian
0. Keluar
Pilih menu: 5
Antrian kosong.
```