

LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA
JOBSHEET XI



NAMA : Handino Asa Galih r

NIM : 244107020237

KELAS : 1-E

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLINEMA

2024

Percobaan 1

Class Mahasiswa12.java

```
public class Mahasiswa12 {  
    String nim;  
    String nama;  
    String kelas;  
    double ipk;  
  
    public Mahasiswa12() {  
    }  
  
    public Mahasiswa12(String nm, String name, String kls, double ip) {  
        nim = nm;  
        nama = name;  
        kelas = kls;  
        ipk = ip;  
    }  
  
    public void tampilInformasi() {  
        System.out.println(nama + "\t\t" + nim + "\t\t" + kelas + "\t\t"  
+ ipk);  
    }  
}
```

Class NodeMahasiswa12.java

```
public class NodeMahasiswa12 {  
    Mahasiswa12 data;  
    NodeMahasiswa12 next;  
  
    public NodeMahasiswa12(Mahasiswa12 data, NodeMahasiswa12 next) {  
        this.data = data;  
        this.next = next;  
    }  
}
```

Class SingleLinkedList12.java

```
public class SingleLinkedList12 {  
    NodeMahasiswa12 head;  
    NodeMahasiswa12 tail;  
  
    boolean isEmpty() {  
        return (head == null);  
    }  
  
    public void print() {  
        if (!isEmpty()) {  
            NodeMahasiswa12 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(Mahasiswa12 input) {
    NodeMahasiswa12 ndInput = new NodeMahasiswa12(input, null);
    if (isEmpty()) {
        head = ndInput;
        tail = ndInput;
    } else {
        ndInput.next = head;
        head = ndInput;
    }
}

public void addLast(Mahasiswa12 input) {
    NodeMahasiswa12 ndInput = new NodeMahasiswa12(input, null);
    if (isEmpty()) {
        head = ndInput;
        tail = ndInput;
    } else {
        tail.next = ndInput;
        tail = ndInput;
    }
}

public void insertAfter(String key, Mahasiswa12 input) {
    NodeMahasiswa12 ndInput = new NodeMahasiswa12(input, null);
    NodeMahasiswa12 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, Mahasiswa12 input) {
    if (index < 0) {
        System.out.println("indeks salah");
    } else if (index == 0) {
        addFirst(input);
    } else {
        NodeMahasiswa12 temp = head;
        for (int i = 0; i < index - 1; i++) {
            temp = temp.next;
        }
        temp.next = new NodeMahasiswa12(input, temp.next);
        if (temp.next.next == null) {
            tail = temp.next;
        }
    }
}
```

Class SLLMain12.java

```
import java.util.Scanner;

public class SLLMain12 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        SingleLinkedList12 sll = new SingleLinkedList12();

        Mahasiswa12 mhs1 = new Mahasiswa12("24212200", "Alvaro", "1A",
4.0);
        Mahasiswa12 mhs2 = new Mahasiswa12("23212201", "Bimon", "2B",
3.8);
        Mahasiswa12 mhs3 = new Mahasiswa12("22212206", "Cintia", "3C",
3.5);
        Mahasiswa12 mhs4 = new Mahasiswa12("21212203", "Dirga", "4D",
3.6);

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

Output

```
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         22212206          3C          3.5
Bimon          23212201          2B          3.8
Alvaro         24212200          1A          4.0
```

Pertanyaan :

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

Karena, memanggil method print dulu dengan kondisi data yang masih kosong.

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

Kegunaan umum variable temp (temporary) sebagai berikut:

- **pointer traversal**: Untuk berpindah dari satu node ke node berikutnya dalam linked list.
- **Menyimpan referensi sementara**: Untuk menyimpan alamat node yang sedang diproses.
- **Navigasi linked list**: untuk Membantu proses penelusuran (traversing) dari head hingga tail.
- **Operasi insert/delete**: Untuk melakukan operasi penambahan atau penghapusan node dengan menyimpan posisi node yang tepat.
- **Pencarian data**: Digunakan untuk mencari node dengan data tertentu dengan cara berpindah dari node satu ke node lainnya.

3. Lakukan modifikasi agar data dapat ditambahkan dari keyboard!

```
int jml = 50;

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

    System.out.println("Tambahkan data mahasiswa 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();

}
```

```

        Mahasiswa12 mhs = new Mahasiswa12(nim, nama, kelas, ipk);
        sll.addFirst(mhs);
        System.out.println();
        System.out.print("Apakah ada yang ingin ditambahkan lagi?
(y/n) : ");
        String jawab = sc.nextLine();
        if (jawab.equalsIgnoreCase("n")) {
            System.out.println("Terima kasih. Have a Nice Day");
            break;
        }
    }

```

Percobaan 2

Penambahan kode program Class SingleLinkedList12

```

public class SingleLinkedList12 {
    NodeMahasiswa12 head;
    NodeMahasiswa12 tail;

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

    public void print() {
        if (!isEmpty()) {
            NodeMahasiswa12 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(Mahasiswa12 input) {

    NodeMahasiswa12 ndInput = new NodeMahasiswa12(input, null);

    if (isEmpty()) {

        head = ndInput;

        tail = ndInput;

    } else {

        ndInput.next = head;

        head = ndInput;

    }

}

public void addLast(Mahasiswa12 input) {

    NodeMahasiswa12 ndInput = new NodeMahasiswa12(input, null);

    if (isEmpty()) {

        head = ndInput;

        tail = ndInput;

    } else {

        tail.next = ndInput;

        tail = ndInput;

    }

}

public void insertAfter(String key, Mahasiswa12 input) {

    NodeMahasiswa12 ndInput = new NodeMahasiswa12(input, null);

    NodeMahasiswa12 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, Mahasiswa12 input) {
    if (index < 0) {
        System.out.println("indeks salah");
    } else if (index == 0) {
        addFirst(input);
    } else {
        NodeMahasiswa12 temp = head;
        for (int i = 0; i < index - 1; i++) {
            temp = temp.next;
        }
        temp.next = new NodeMahasiswa12(input, temp.next);
        if (temp.next.next == null) {
            tail = temp.next;
        }
    }
}
}

```

Kode Program Class SSLMain

```
System.out.println("data index 1 :");

    sll.getData(1);

    System.out.println();

    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();
```

Output

```
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

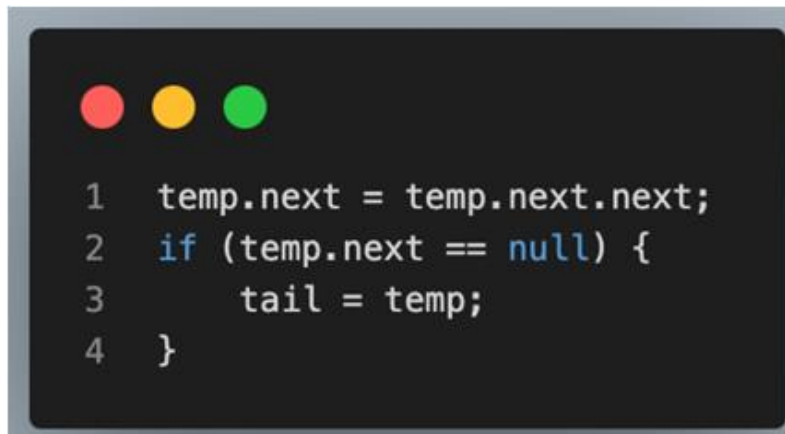
Isi Linked List:
Bimon       23212201   2B          3,8
```

Pertanyaan

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

Untuk menghentikan Loop dan mencegah eror Ketika penghapusan multiple node atau operasi pada node yang tidak valid.

2. Jelaskan kegunaan kode dibawah pada method remove



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

Untuk mengupdate pointer Ketika ingin mengubah referensi node sebelumnya agar menunjuk ke kode setelah node yang dihapus. Kemudian , jika kode referensi berisi null maka node (temp) merupakan tail

Tugas

Class Mahasiswa

```
public class Mahasiswa {
    String nim;
    String nama;
    String kelas;
    String prodi;

    public Mahasiswa(String nim, String nama, String prodi, String
kelas) {
        this.nim = nim;
        this.nama = nama;
        this.prodi = prodi;
        this.kelas = kelas;
    }
}
```

```

    }

    public void tampilData() {
        System.out.println(nama + "\t\t" + nim + "\t\t" + kelas + "\t\t"
+ prodi);
    }
}

```

Class NodeMhs

```

public class Node {
    Mahasiswa data;
    Node next;

    public Node(Mahasiswa data, Node next) {
        this.data = data;
        this.next = next;
    }
}

```

Class QueueLinkedListLayananUnit

```

public class Queue {
    Node front;
    Node rear;
    int size;
    int max;

    public Queue(int max) {
        this.max = max;
        this.size = 0;
        this.front = null;
        this.rear = null;
    }
}

```

```
        this.rear = null;
    }

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

    public boolean isFull() {
        return size == max;
    }

    public void clear() {
        front = null;
        rear = null;
        size = 0;
        System.out.println("Antrian berhasil dikosongkan!");
    }

    public void enqueue(Mahasiswa data) {
        if (isFull()) {
            System.out.println("Antrian sudah penuh!");
            return;
        }
        Node newNode = new Node(data, null);
        if (isEmpty()) {
            front = newNode;
            rear = newNode;
        } else {
            rear.next = newNode;
            rear = newNode;
        }
    }
}
```

```

        size++;

        System.out.println("Mahasiswa berhasil ditambahkan ke antrian");
    }

    public Mahasiswa dequeue() {
        if (isEmpty()) {
            System.out.println("Tidak ada antrian");
            return null;
        }
        Mahasiswa data = front.data;
        front = front.next;
        size--;
        if (isEmpty()) {
            rear = null;
        }
        return data;
    }

    public void peek() {
        if (!isEmpty()) {
            System.out.println("Antrian terdepan:");
            front.data.tampilData();
        } else {
            System.out.println("Tidak ada antrian");
        }
    }

    public void peekBelakang() {
        if (!isEmpty()) {
            System.out.println("Antrian terakhir:");
            rear.data.tampilData();
        } else {

```

```

        System.out.println("Tidak ada antrian");
    }
}

public void print() {
    if (!isEmpty()) {
        Node tmp = front;
        int i = 0;
        System.out.println("Isi antrian: ");
        while (tmp != null) {
            System.out.println("Mahasiswa ke-" + (i + 1) + ":");
            tmp.data.tampilData();
            System.out.println();
            tmp = tmp.next;
            i++;
        }
    } else {
        System.out.println("Tidak ada antrian");
    }
}

public int getSize() {
    return size;
}
}

```


Class QueueMainLayananUnitKemahasiswaan

```
import java.util.Scanner;

public class QueueMain {
    public static void menu() {
        System.out.println("\n+==== MENU LAYANAN UNIT KEMAHASISWAAN  
====+");
        System.out.println("| No |  
Menu |");
        System.out.println("+----+-----+");
        System.out.println("| 1 | Tambah  
Antrian |");
        System.out.println("| 2 | Panggil  
Antrian |");
        System.out.println("| 3 | Cek Antrian  
Terdepan |");
        System.out.println("| 4 | Cek Antrian  
Terakhir |");
        System.out.println("| 5 | Tampilkan Semua  
Antrian |");
        System.out.println("| 6 | Cek Jumlah  
Antrian |");
        System.out.println("| 7 | Kosongkan  
Antrian |");
        System.out.println("| 0 |  
Keluar |");
        System.out.println("+----+-----+");
    }

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

        System.out.print("Masukkan kapasitas maksimal antrian: ");
        int n = sc.nextInt();
        sc.nextLine();

        Queue queue = new Queue(n);
    }
}
```

```

        int pilih;
        do {
            menu();
            System.out.print("Pilih menu (0-7): ");
            pilih = sc.nextInt();
            sc.nextLine();

            switch (pilih) {
                case 1:
                    if (!queue.isFull()) {
                        System.out.println("\nMasukkan data
mahasiswa:");

                        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();

                        Mahasiswa mhs = new Mahasiswa(nim, nama, prodi,
kelas);

                        queue.enqueue(mhs);
                    } else {
                        System.out.println("Antrian sudah penuh!");
                    }
                    break;
                case 2:
                    Mahasiswa mhs = queue.dequeue();
                    if (mhs != null) {
                        System.out.println("\nMahasiswa yang
dipanggil:");

                        mhs.tampilData();
                    }
                    break;
            }
        } while (true);
    }
}

```

```

        case 2:
            Mahasiswa mhs = queue.dequeue();
            if (mhs != null) {
                System.out.println("\nMahasiswa yang
dipanggil:");
                mhs.tampilData();
            }
            break;
        case 3:
            queue.peek();
            break;
        case 4:
            queue.peekBelakang();
            break;
        case 5:
            queue.print();
            break;
        case 6:
            System.out.println("Jumlah mahasiswa dalam antrian:
" + queue.getSize());
            break;
        case 7:
            queue.clear();
            break;
        case 0:
            System.out.println("Terima kasih telah menggunakan
layanan ini.");
            break;
        default:
            System.out.println("Menu tidak valid!");
    }
} while (pilih != 0);
sc.close();
}
}

```

Output

```
+==== MENU LAYANAN UNIT KEMAHASISWAAN =====+
| No | Menu
+-----+
| 1 | Tambah Antrian
| 2 | Panggil Antrian
| 3 | Cek Antrian Terdepan
| 4 | Cek Antrian Terakhir
| 5 | Tampilkan Semua Antrian
| 6 | Cek Jumlah Antrian
| 7 | Kosongkan Antrian
| 0 | Keluar
+-----+
Pilih menu (0-7): 1

Masukkan data mahasiswa:
NIM   : 1233214
Nama  : Handino
Prodi : TI
Kelas : 1E
Mahasiswa berhasil ditambahkan ke antrian
```

```
Pilih menu (0-7): 1

Masukkan data mahasiswa:
NIM   : 45665478
Nama  : Asa Galih
Prodi : TI
Kelas : 1E
Mahasiswa berhasil ditambahkan ke antrian
```

```
Pilih menu (0-7): 1

Masukkan data mahasiswa:
NIM   : 12345678
Nama  : Romadhon
Prodi : TI
Kelas : 1D
Mahasiswa berhasil ditambahkan ke antrian
```

```
Pilih menu (0-7): 1

Masukkan data mahasiswa:
NIM   : 09876543
Nama  : Akbar
Prodi : TI
Kelas : 1E
Mahasiswa berhasil ditambahkan ke antrian
```

```
+==== MENU LAYANAN UNIT KEMAHASISWAAN ====+
| No | Menu |
+-----+
| 1 | Tambah Antrian |
| 2 | Panggil Antrian |
| 3 | Cek Antrian Terdepan |
| 4 | Cek Antrian Terakhir |
| 5 | Tampilkan Semua Antrian |
| 6 | Cek Jumlah Antrian |
| 7 | Kosongkan Antrian |
| 0 | Keluar |
+-----+
Pilih menu (0-7): 3
Antrian terdepan:
Handino          1233214          1E          TI

+==== MENU LAYANAN UNIT KEMAHASISWAAN ====+
| No | Menu |
+-----+
| 1 | Tambah Antrian |
| 2 | Panggil Antrian |
| 3 | Cek Antrian Terdepan |
| 4 | Cek Antrian Terakhir |
| 5 | Tampilkan Semua Antrian |
| 6 | Cek Jumlah Antrian |
| 7 | Kosongkan Antrian |
| 0 | Keluar |
+-----+
Pilih menu (0-7): 2
Mahasiswa yang dipanggil:
Handino          1233214          1E          TI
```

```

+==== MENU LAYANAN UNIT KEMAHASISWAAN ====+
| No | Menu |
+-----+
| 1 | Tambah Antrian |
| 2 | Panggil Antrian |
| 3 | Cek Antrian Terdepan |
| 4 | Cek Antrian Terakhir |
| 5 | Tampilkan Semua Antrian |
| 6 | Cek Jumlah Antrian |
| 7 | Kosongkan Antrian |
| 0 | Keluar |
+-----+
Pilih menu (0-7): 6
Jumlah mahasiswa dalam antrian: 3

+==== MENU LAYANAN UNIT KEMAHASISWAAN ====+
| No | Menu |
+-----+
| 1 | Tambah Antrian |
| 2 | Panggil Antrian |
| 3 | Cek Antrian Terdepan |
| 4 | Cek Antrian Terakhir |
| 5 | Tampilkan Semua Antrian |
| 6 | Cek Jumlah Antrian |
| 7 | Kosongkan Antrian |
| 0 | Keluar |
+-----+
Pilih menu (0-7): 4
Antrian terakhir:
Akbar          09876543          1E          TI

```

```

+==== MENU LAYANAN UNIT KEMAHASISWAAN ====+
| No | Menu |
+-----+
| 1 | Tambah Antrian |
| 2 | Panggil Antrian |
| 3 | Cek Antrian Terdepan |
| 4 | Cek Antrian Terakhir |
| 5 | Tampilkan Semua Antrian |
| 6 | Cek Jumlah Antrian |
| 7 | Kosongkan Antrian |
| 0 | Keluar |
+-----+
Pilih menu (0-7): 5
Isi antrian:
Mahasiswa ke-1:
Asa Galih          45665478          1E          TI

Mahasiswa ke-2:
Romadhon          12345678          1D          TI

Mahasiswa ke-3:
Akbar          09876543          1E          TI

```