

JOBSHEET 11
ALGORITMA STRUKTUR
DATA



Burhnauddin ihsan
244107020189
TI 1E/06

PROGRAM STUDI D-IV TEKNIK INFORMATIKA JURUSAN
TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
2025

PERCOBAAN 1

1. Kode program

Mahasiswa06

```
public class Mahasiswa06 {  
  
    String nim, nama, kelas;  
    double ipk;  
  
    Mahasiswa06() {  
  
    }  
  
    Mahasiswa06(String nama, String nim, String kelas, double ipk)  
{  
        this.nama = nama;  
        this.nim = nim;  
        this.kelas = kelas;  
        this.ipk = ipk;  
    }  
  
    public void tampilInformasi() {  
        System.out.printf("%-10s %-10s %-4s %.1f\n", nama, nim,  
kelas, ipk);  
    }  
}
```

NodeMahasiswa06

```
public class NodeMahasiswa06 {  
    Mahasiswa06 data;  
    NodeMahasiswa06 next;  
  
    public NodeMahasiswa06(Mahasiswa06 data, NodeMahasiswa06 next) {  
        this.data = data;  
        this.next = next;  
    }  
}
```

SingleLinkedList

```
void insertAt(int index, Mahasiswa06 input) {
    if (index < 0) {
        System.out.println("index salah");
    } else if (index == 0) {
        addFirst(input);
    } else {
        NodeMahasiswa06 tmp = head;
        for(int i = 0; i < index-1; i++) {
            tmp = tmp.next;
        }
        tmp.next = new NodeMahasiswa06(input, tmp.next);
        if (tmp.next.next == null) {
            tail = tmp.next;
        }
    }
}

} else {
    System.out.println("Linked list kosong");
}
}

void addFirst (Mahasiswa06 input) {
    NodeMahasiswa06 newNode = new NodeMahasiswa06(input, null);
    if (isEmpty()) {
        head = newNode;
        tail = newNode;
    } else {
        newNode.next = head;
        head = newNode;
    }
}

void addLast (Mahasiswa06 input) {
    NodeMahasiswa06 newNode = new NodeMahasiswa06(input, null);
    if (isEmpty()) {
        head = newNode;
        tail = newNode;
    } else {
        tail.next = newNode;
        tail = newNode;
    }
}

void insertAfter (String key, Mahasiswa06 input) {
    NodeMahasiswa06 newNode = new NodeMahasiswa06(input, null);
    NodeMahasiswa06 tmp = head;
    do {
        if (tmp.data.nama.equalsIgnoreCase(key)) {
            newNode.next = tmp.next;
            tmp.next = newNode;
            if (newNode.next == null) {
                tail = newNode;
            }
            break;
        }
        tmp = tmp.next;
    } while (tmp != null);
}
```

SllMain06

```
public class SllMain06 {  
    public static void main(String[] args) {  
        SingleLinkedList sll = new SingleLinkedList();  
        Mahasiswa06 mhs1 = new Mahasiswa06("Alya", "123456", "TI-1A",  
3.75);  
        Mahasiswa06 mhs2 = new Mahasiswa06("Bagas", "123457", "TI-1B",  
3.40);  
        Mahasiswa06 mhs3 = new Mahasiswa06("Citra", "123458", "TI-1A",  
3.90);  
        Mahasiswa06 mhs4 = new Mahasiswa06("Dimas", "123459", "TI-1C",  
3.20);  
  
        sll.print();  
        sll.addFirst(mhs4);  
        sll.print();  
        sll.addLast(mhs1);  
        sll.print();  
        sll.insertAfter("Dimas", mhs3);  
        sll.insertAt(2, mhs2);  
        sll.print();  
  
        System.out.println("data index 1 : ");  
        sll.getData(1);  
  
        System.out.println("data mahasiswa an Dimas berada pada index : " +  
sll.indexOf("Dimas"));  
        System.out.println();  
  
        sll.removeFirst();  
        sll.removeLast();  
        sll.print();  
        sll.removeAt(0);  
        sll.print();  
  
    }  
}
```

2. Hasil dari kode program

```

Linked list kosong
isi linked list:
Alvaro      24212200    1A    4,0

isi linked list:
Alvaro      24212200    1A    4,0
Cintia      22212202    3C    3,5

isi linked list:
Alvaro      24212200    1A    4,0
Cintia      22212202    3C    3,5
Bimon       23212201    2B    3,8

```

PERTANYAAN

1. Pesan “Linked List Kosong” muncul karena saat program dijalankan, linked list belum memiliki data sehingga pointer head masih bernilai null. Saat isi list ditampilkan, program mendeteksi bahwa list kosong dan menampilkan pesan tersebut.
2. Secara singkat, variabel temp digunakan sebagai penunjuk sementara untuk menelusuri node-node dalam linked list tanpa mengubah posisi pointer utama seperti head. Ini membantu dalam proses seperti menampilkan, mencari, atau menghapus data secara aman.
3. Modifikasi SllMain06

```

        System.out.print("Berapa jumlah data mahasiswa yang ingin
ditambahkan? ");
        int jumlah = sc.nextInt();
        sc.nextLine();

        for (int i = 0; i < jumlah; i++) {
            System.out.println("\nData 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();

            Mahasiswa25 mhs = new Mahasiswa25(nama, nim, kelas, ipk);
            sll.addLast(mhs);
        }

        System.out.println("\nData Mahasiswa dalam Linked List:");
        sll.print();
    }

```

PERCOBAAN 2

1. kode program

SingleLinkedList

```
public class SingleLinkedList {
    NodeMahasiswa06 head;
    NodeMahasiswa06 tail;

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

    void print() {
        if (!isEmpty()) {
            NodeMahasiswa06 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");
        }
    }

    void addFirst (Mahasiswa06 input) {
        NodeMahasiswa06 newNode = new NodeMahasiswa06(input, null);
        if (isEmpty()) {
            head = newNode;
            tail = newNode;
        } else {
            newNode.next = head;
            head = newNode;
        }
    }

    void addLast (Mahasiswa06 input) {
        NodeMahasiswa06 newNode = new NodeMahasiswa06(input, null);
        if (isEmpty()) {
            head = newNode;
            tail = newNode;
        } else {
            tail.next = newNode;
            tail = newNode;
        }
    }

    void insertAfter (String key, Mahasiswa06 input) {
        NodeMahasiswa06 newNode = new NodeMahasiswa06(input, null);
        NodeMahasiswa06 tmp = head;
        do {
            if (tmp.data.nama.equalsIgnoreCase(key)) {
                newNode.next = tmp.next;
                tmp.next = newNode;
                if (newNode.next == null) {
                    tail = newNode;
                }
                break;
            }
            tmp = tmp.next;
        } while (tmp != null);
    }
}
```

```

void insertAt(int index, Mahasiswa06 input) {
    if (index < 0) {
        System.out.println("index salah");
    } else if (index == 0) {
        addFirst(input);
    } else {
        NodeMahasiswa06 tmp = head;
        for(int i = 0; i < index-1; i++) {
            tmp = tmp.next;
        }
        tmp.next = new NodeMahasiswa06(input, tmp.next);
        if (tmp.next.next == null) {
            tail = tmp.next;
        }
    }
}

void getData(int index) {
    NodeMahasiswa06 tmp = head;

    for (int i = 0; i < index; i++) {
        tmp = tmp.next;
    }

    if (tmp == null) {
        System.out.println("");
    }
    tmp.data.tampilInformasi();
}

int indexOf(String key) {
    NodeMahasiswa06 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;
    }
}

void removeFirst() {
    if (isEmpty()) {
        System.out.println("Linked List masih kosong, tidak dapat
dihapus");
        return;
    } else if (head == tail) {
        head = tail = null;
    } else {
        head = head.next;
    }
}

```

```

void removeLast() {
    if (isEmpty()) {
        System.out.println("Linked List masih kosong, tidak dapat
dihapus");
    } else if (head == tail) {
        head = tail = null;
    } else {
        NodeMahasiswa06 tmp = head;
        while (tmp.next != tail) {
            tmp = tmp.next;
        }
        tmp.next = null;
        tail = tmp;
    }
}

void remove(String key) {
    if (isEmpty()) {
        System.out.println("Linked List masih kosong, tidak dapat
dihapus");
    } else {
        NodeMahasiswa06 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;
        }
    }
}

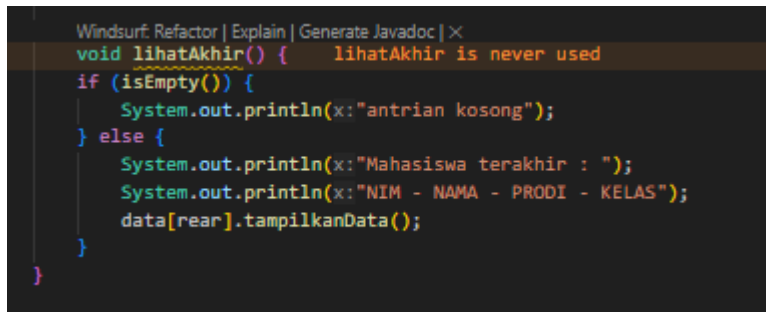
void removeAt(int index) {
    if (index == 0) {
        removeFirst();
    } else {
        NodeMahasiswa06 temp = head;
        for (int i = 0; i < index - 1; i++) {
            temp = temp.next;
        }

        temp.next = temp.next.next;
        if (temp.next == null) {
            tail = temp;
        }
    }
}
}

```


PERTANYAAN

1. keyword break pada method remove digunakan untuk keluar dari perulangan ketika salah satu dari kondisi tersebut sudah terpenuhi
2. modifikasi program



```
Windsurf: Refactor | Explain | Generate Javadoc | X
void lihatAkhir() { lihatAkhir is never used
if (isEmpty()) {
    System.out.println(x:"antrian kosong");
} else {
    System.out.println(x:"Mahasiswa terakhir : ");
    System.out.println(x:"NIM - NAMA - PRODI - KELAS");
    data[rear].tampilkanData();
}
}
```

TUGAS

1. kode program

QueueLinkedList

```
public class QueueLinkedList {
    NodeMahasiswa06 head;
    int capacity;
    int slot;

    QueueLinkedList(int capacity) {
        this.capacity = capacity;
        slot = capacity;
    }

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

    void print() {
        if (!isEmpty()) {
            NodeMahasiswa06 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");
        }
    }

    void addQueue(Mahasiswa06 input) {
        if (isEmpty()) {
            NodeMahasiswa06 newQueue = new NodeMahasiswa06(input,
null);
            head = newQueue;
        } else {
            NodeMahasiswa06 newQueue = new NodeMahasiswa06(input,
null);
            NodeMahasiswa06 tmp = head;
            while (tmp.next != null) {
```

```

tmp = tmp.next;
    }
    tmp.next = newQueue;
}
slot--;
}

Mahasiswa06 getQueue() {
    if (isEmpty()) {
        System.out.println("Antrian kosong");
        return null;
    } else {
        NodeMahasiswa06 mhs = head;
        head = head.next;
        slot++;
        return mhs.data;
    }
}

boolean isFull () {
    return slot == 0;
}

void clearQueue() {
    head = null;
}

int countMhs() {
    return capacity - slot;
}

Mahasiswa06 firstQueue() {
    return head.data;
}

Mahasiswa06 lastQueue() {
    NodeMahasiswa06 tmp = head;
    while (tmp.next != null) {
        tmp = tmp.next;
    }

    return tmp.data;
}
}

```

QueueMahasiswaMain

```
import java.util.Scanner;

public class QueueMahasiswaMain {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Masukkan kapasitas antrian: ");
        int kapasitas = sc.nextInt();

        QueueLinkedList queue = new QueueLinkedList(kapasitas);

        int pilihan;
        do {
            System.out.println("\n=== Menu Antrian Mahasiswa ===");
            System.out.println("1. Tambah antrian");
            System.out.println("2. Panggil antrian");
            System.out.println("3. Cek antrian kosong");
```

```

        System.out.println("4. Cek antrian penuh");
        System.out.println("5. Tampilkan antrian terdepan");
        System.out.println("6. Tampilkan antrian terakhir");
        System.out.println("7. Tampilkan semua antrian");
        System.out.println("8. Tampilkan jumlah mahasiswa dalam
antrian");
        System.out.println("9. Kosongkan antrian");
        System.out.println("0. Keluar");
        System.out.print("Pilih menu: ");
        pilihan = sc.nextInt();
        sc.nextLine();

        switch (pilihan) {
            case 1:
                if (!queue.isFull()) {
                    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();

                    Mahasiswa06 mhs = new Mahasiswa06(nama, nim,
kelas, ipk);

                    queue.addQueue(mhs);
                    System.out.println("Mahasiswa berhasil ditambahkan
ke antrian.");
                } else {
                    System.out.println("Antrian penuh!");
                }
                break;
            case 2:
                Mahasiswa06 dipanggil = queue.getQueue();
                if (dipanggil != null) {
                    System.out.println("Memanggil mahasiswa:");
                    dipanggil.tampilInformasi();
                }
                break;
            case 3:
                System.out.println(queue.isEmpty() ? "Antrian kosong."
: "Antrian tidak kosong.");
                break;
            case 4:
                System.out.println(queue.isFull() ? "Antrian penuh." :
"Antrian belum penuh.");
                break;
            case 5:
                if (!queue.isEmpty()) {
                    System.out.println("Mahasiswa di antrian
terdepan:");

                    queue.firstQueue().tampilInformasi();
                } else {
                    System.out.println("Antrian kosong.");
                }
                break;
        }
    }
}

```

```

        case 6:
            if (!queue.isEmpty()) {
                System.out.println("Mahasiswa di antrian
terakhir:");
                queue.lastQueue().tampilInformasi();
            } else {
                System.out.println("Antrian kosong.");
            }
            break;
        case 7:
            queue.print();
            break;
        case 8:
            System.out.println("Jumlah mahasiswa dalam antrian: "
+ queue.countMhs());
            break;
        case 9:
            queue.clearQueue();
            System.out.println("Antrian telah dikosongkan.");
            break;
        case 0:
            System.out.println("Terima kasih.");
            break;
        default:
            System.out.println("Pilihan tidak valid!");
    }
} while (pilihan != 0);
}
}

```

2. hasil dari kode program

```

=== Menu Antrian Mahasiswa ===
1. Tambah antrian
2. Panggil antrian
3. Cek antrian kosong
4. Cek antrian penuh
5. Tampilkan antrian terdepan
6. Tampilkan antrian terakhir
7. Tampilkan semua antrian
8. Tampilkan jumlah mahasiswa dalam antrian
9. Kosongkan antrian
0. Keluar
Pilih menu: 7
Isi linked list
burhan 121 1e 3,0
udin 2 1b 2,0
ihсан 145 1d 4,0

=== Menu Antrian Mahasiswa ===
1. Tambah antrian
2. Panggil antrian
3. Cek antrian kosong
4. Cek antrian penuh
5. Tampilkan antrian terdepan
6. Tampilkan antrian terakhir
7. Tampilkan semua antrian
8. Tampilkan jumlah mahasiswa dalam antrian
9. Kosongkan antrian
0. Keluar
Pilih menu: 5
Mahasiswa di antrian terdepan:
burhan 121 1e 3,0

=== Menu Antrian Mahasiswa ===
1. Tambah antrian
2. Panggil antrian
3. Cek antrian kosong
4. Cek antrian penuh
5. Tampilkan antrian terdepan
6. Tampilkan antrian terakhir
7. Tampilkan semua antrian
8. Tampilkan jumlah mahasiswa dalam antrian
9. Kosongkan antrian
0. Keluar
Pilih menu: 6
Mahasiswa di antrian terakhir:
ihسان 145 1d 4,0

=== Menu Antrian Mahasiswa ===
1. Tambah antrian
2. Panggil antrian
3. Cek antrian kosong
4. Cek antrian penuh
5. Tampilkan antrian terdepan
6. Tampilkan antrian terakhir
7. Tampilkan semua antrian
8. Tampilkan jumlah mahasiswa dalam antrian
9. Kosongkan antrian
0. Keluar
Pilih menu: 8
Jumlah mahasiswa dalam antrian: 3

```

```
=== Menu Antrian Mahasiswa ===
1. Tambah antrian
2. Panggil antrian
3. Cek antrian kosong
4. Cek antrian penuh
5. Tampilkan antrian terdepan
6. Tampilkan antrian terakhir
7. Tampilkan semua antrian
8. Tampilkan jumlah mahasiswa dalam antrian
9. Kosongkan antrian
0. Keluar
Pilih menu: 3
Antrian tidak kosong.
```

```
=== Menu Antrian Mahasiswa ===
1. Tambah antrian
2. Panggil antrian
3. Cek antrian kosong
4. Cek antrian penuh
5. Tampilkan antrian terdepan
6. Tampilkan antrian terakhir
7. Tampilkan semua antrian
8. Tampilkan jumlah mahasiswa dalam antrian
9. Kosongkan antrian
0. Keluar
Pilih menu: 4
Antrian penuh.
```

```
=== Menu Antrian Mahasiswa ===
1. Tambah antrian
2. Panggil antrian
3. Cek antrian kosong
4. Cek antrian penuh
5. Tampilkan antrian terdepan
6. Tampilkan antrian terakhir
7. Tampilkan semua antrian
8. Tampilkan jumlah mahasiswa dalam antrian
9. Kosongkan antrian
0. Keluar
Pilih menu: 2
Memanggil mahasiswa:
burhan      121      1e  3,0
```

```
=== Menu Antrian Mahasiswa ===
1. Tambah antrian
2. Panggil antrian
3. Cek antrian kosong
4. Cek antrian penuh
5. Tampilkan antrian terdepan
6. Tampilkan antrian terakhir
7. Tampilkan semua antrian
8. Tampilkan jumlah mahasiswa dalam antrian
9. Kosongkan antrian
0. Keluar
Pilih menu: 9
Antrian telah dikosongkan.
```

```
=== Menu Antrian Mahasiswa ===
1. Tambah antrian
2. Panggil antrian
3. Cek antrian kosong
4. Cek antrian penuh
5. Tampilkan antrian terdepan
6. Tampilkan antrian terakhir
7. Tampilkan semua antrian
8. Tampilkan jumlah mahasiswa dalam antrian
9. Kosongkan antrian
0. Keluar
Pilih menu: 9
Antrian telah dikosongkan.
```

```
=== Menu Antrian Mahasiswa ===
1. Tambah antrian
2. Panggil antrian
3. Cek antrian kosong
4. Cek antrian penuh
5. Tampilkan antrian terdepan
6. Tampilkan antrian terakhir
7. Tampilkan semua antrian
8. Tampilkan jumlah mahasiswa dalam antrian
9. Kosongkan antrian
0. Keluar
Pilih menu: 3
Antrian kosong.
```

```
=== Menu Antrian Mahasiswa ===
1. Tambah antrian
2. Panggil antrian
3. Cek antrian kosong
4. Cek antrian penuh
5. Tampilkan antrian terdepan
6. Tampilkan antrian terakhir
7. Tampilkan semua antrian
8. Tampilkan jumlah mahasiswa dalam antrian
9. Kosongkan antrian
0. Keluar
Pilih menu: █
```

Masukkan kapasitas antrian: 3

=== Menu Antrian Mahasiswa ===

1. Tambah antrian
2. Panggil antrian
3. Cek antrian kosong
4. Cek antrian penuh
5. Tampilkan antrian terdepan
6. Tampilkan antrian terakhir
7. Tampilkan semua antrian
8. Tampilkan jumlah mahasiswa dalam antrian
9. Kosongkan antrian
0. Keluar

Pilih menu: 1

Nama : burhan

NIM : 121

Kelas : 1e

IPK : 3

Mahasiswa berhasil ditambahkan ke antrian.

=== Menu Antrian Mahasiswa ===

1. Tambah antrian
2. Panggil antrian
3. Cek antrian kosong
4. Cek antrian penuh
5. Tampilkan antrian terdepan
6. Tampilkan antrian terakhir
7. Tampilkan semua antrian
8. Tampilkan jumlah mahasiswa dalam antrian
9. Kosongkan antrian
0. Keluar

Pilih menu: 1

Nama : udin

NIM : 2

Kelas : 1b

IPK : 2

Mahasiswa berhasil ditambahkan ke antrian.

=== Menu Antrian Mahasiswa ===

1. Tambah antrian
2. Panggil antrian
3. Cek antrian kosong
4. Cek antrian penuh
5. Tampilkan antrian terdepan
6. Tampilkan antrian terakhir
7. Tampilkan semua antrian
8. Tampilkan jumlah mahasiswa dalam antrian
9. Kosongkan antrian
0. Keluar

Pilih menu: 1

Nama : ihsan

NIM : 145

Kelas : 1d

IPK : 4

Mahasiswa berhasil ditambahkan ke antrian.