

JOBSHEET 11
LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN STRUKTUR
DATA



MUHAMMAD DAFFI FIROS ZAIDAN

244107020182

TI 1E

PROGRAM STUDI D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2025

Percobaan : Pembuatan Single Linked List

Kode Program

Class Mahasiswa

```
public class Mahasiswa17 {  
    String nim, nama, kelas;  
    double ipk;  
  
    public Mahasiswa17(String nim, String nama, String kelas,  
double ipk) {  
        this.nim = nim;  
        this.nama = nama;  
        this.kelas = kelas;  
        this.ipk = ipk;  
    }  
  
    public void tampilkanInformasi() {  
        System.out.printf("%-15s %-10s %-5s %-5.1f%n", nama, nim,  
kelas, ipk);  
    }  
}
```

Class SingleLinkedList17

```
public class SingleLinkedList17 {
    NodeMahasiswa17 head;
    NodeMahasiswa17 tail;

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

    public void print() {
        if (!isEmpty()) {
            NodeMahasiswa17 tmp = head;
            System.out.println("Isi Linked List:\t");
            while (tmp != null) {
                tmp.data.tampilkanInformasi();
                tmp = tmp.next;
            }
            System.out.println("");
        } else {
            System.out.println("Linked list kosong");
        }
    }

    public void addFirst(Mahasiswa17 input) {
        NodeMahasiswa17 ndInput = new NodeMahasiswa17(input, null);
        if (isEmpty()) {
            head = ndInput;
            tail = ndInput;
        } else {
            ndInput.next = head;
            head = ndInput;
        }
    }

    public void addLast(Mahasiswa17 input) {
        NodeMahasiswa17 ndInput = new NodeMahasiswa17(input, null);
        if (isEmpty()) {
            head = ndInput;
            tail = ndInput;
        } else {
            tail.next = ndInput;
            tail = ndInput;
        }
    }

    public void insertAfter(String key, Mahasiswa17 input) {
        NodeMahasiswa17 ndInput = new NodeMahasiswa17(input, null);
        NodeMahasiswa17 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 insertAt(int index, Mahasiswa17 input) {
        if (index < 0) {
            System.out.println("Indeks Salah");
        } else if (index == 0) {
            addFirst(input);
        } else {
            NodeMahasiswa17 temp = head;
            for (int i = 0; i < index - 1; i++) {
                temp = temp.next;
            }
            temp.next = new NodeMahasiswa17(input, temp.next);
            if (temp.next.next == null) {
                tail = temp.next;
            }
        }
    }
}
```

Class NodeMahasiswa17

```
public class NodeMahasiswa17 {
    Mahasiswa17 data;
    NodeMahasiswa17 next;

    public NodeMahasiswa17(Mahasiswa17 data, NodeMahasiswa17 next)
    {
        this.data = data;
        this.next = next;
    }
}
```

Class SLLMain17

```
public class SLLMain17 {
    public static void main(String[] args) {
        SingleLinkedList17 sll = new SingleLinkedList17();

        Mahasiswa17 mhs1 = new Mahasiswa17("24212200", "Alvaro", "1A",
4.0);
        Mahasiswa17 mhs2 = new Mahasiswa17("23212201", "Bimon", "2B",
3.8);
        Mahasiswa17 mhs3 = new Mahasiswa17("22212202", "Cintia", "3C",
3.5);
        Mahasiswa17 mhs4 = new Mahasiswa17("21212203", "Dirga", "4D",
3.6);

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

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     22212202  3C  3,5
Bimon      23212201  2B  3,8
Alvaro     24212200  1A  4,0

PS D:\Tugas Kuliah\SEMESTER 2\ASLD\Praktikum ASD\Praktikum-ASD\Jobsheet 11>
```

Pertanyaan

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

Jawaban: Karena pada saat baris pertama dijalankan, linked list belum memiliki data sama sekali (belum ada node yang dimasukkan), sehingga kondisi isEmpty() bernilai true dan program menampilkan "Linked list kosong".

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

Jawaban: Variabel temp digunakan sebagai penunjuk (pointer) sementara untuk menelusuri (traversal) node-node pada linked list. Dengan temp, kita bisa membaca atau memodifikasi data pada setiap node tanpa mengubah posisi head atau tail, sehingga data pada linked list tetap aman dan tidak terhapus.

3. Lakukan modifikasi agar data dapat ditambahkan dari keyboard!

Jawaban:

```
import java.util.Scanner;

public class SLLMain17 {
    public static void main(String[] args) {
        SingleLinkedList17 sll = new SingleLinkedList17();
        Scanner input = new Scanner(System.in);

        System.out.print("Masukkan jumlah mahasiswa: ");
        int n = input.nextInt();
        input.nextLine(); // membersihkan buffer

        for (int i = 0; i < n; i++) {
            System.out.println("Data Mahasiswa ke-" + (i+1));
            System.out.print("NIM : ");
            String nim = input.nextLine();
            System.out.print("Nama : ");
            String nama = input.nextLine();
            System.out.print("Kelas : ");
            String kelas = input.nextLine();
            System.out.print("IPK : ");
            double ipk = input.nextDouble();
            input.nextLine(); // membersihkan buffer

            Mahasiswa17 mhs = new Mahasiswa17(nim, nama, kelas, ipk);
            sll.addLast(mhs);
        }

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

Percobaan 2: Modifikasi Elemen pada Single Link List

Penambahan Kode Program

Class SingleLinkedList17

```
public Mahasiswa17 getData(int index) {
    if (index < 0 || index >= size()) {
        return null;
    }
    NodeMahasiswa17 tmp = head;
    for (int i = 0; i < index; i++) {
        tmp = tmp.next;
    }
    return tmp.data;
}

public int indexOf(String key) {
    NodeMahasiswa17 tmp = head;
    int index = 0;
    while (tmp != null && !tmp.data.nama.equalsIgnoreCase(key)) {
        tmp = tmp.next;
        index++;
    }
    if (tmp == null) {
        return -1;
    }
    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 {
        head = head.next;
    }
}

public void remove (String key) {
    if (isEmpty()) {
        System.out.println("Linked List masih Kosong, tidak dapat dihapus!");
    } else {
        NodeMahasiswa17 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 {
        NodeMahasiswa17 temp = head;
        for (int i = 0; i < index - 1; i++) {
            temp = temp.next;
        }
        temp.next = temp.next.next;
        if (temp.next == null) {
            tail = temp;
        }
    }
}
```

Class SLLMain17

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

    sc.close();
}
}
```

Output

```
data index 1 :
Cintia      22212202   3C   3,5

data mahasiswa an Bimon berada pada index : 2

Isi Linked List:
Bimon       23212201   2B   3,8
Dirga       21212203   4D   3,6

Isi Linked List:
Dirga       21212203   4D   3,6

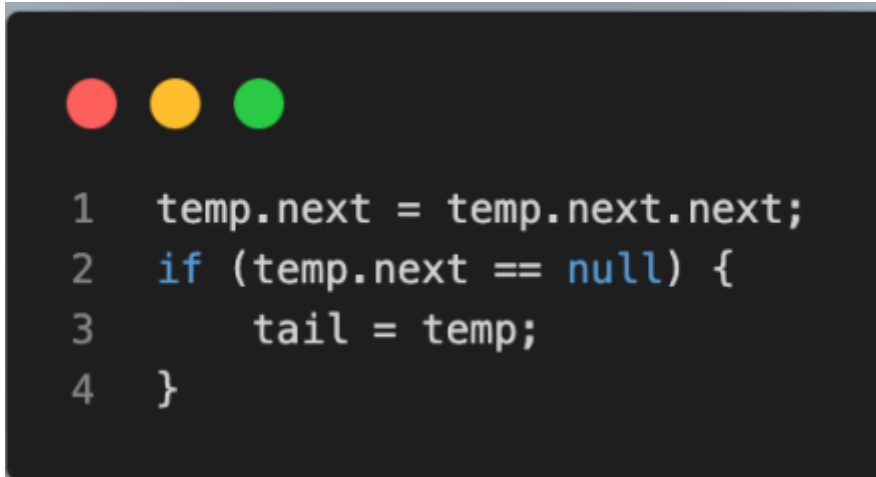
'S D:\Tugas Kuliah\SEMESTER 2\ASLD\Praktikum ASD\Praktikum-ASD\Jobsheet 11> |
```

Pertanyaan

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

Jawaban: Dalam implementasi fungsi remove pada struktur data seperti *linked list*, kata kunci break digunakan untuk menghentikan eksekusi loop pencarian elemen yang ingin dihapus. Loop tersebut biasanya diiterasi untuk menelusuri daftar hingga elemen yang dituju ditemukan. Setelah elemen yang sesuai berhasil ditemukan dan proses penghapusan node dilakukan, penggunaan break akan segera menghentikan iterasi loop. Hal ini penting untuk optimasi kinerja, karena tidak ada lagi kebutuhan untuk melanjutkan pencarian di sisa elemen daftar setelah target operasi (penghapusan) tercapai.

2. Jelaskan kegunaan kode dibawah pada method remove!



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

Jawaban

- Baris `temp.next = temp.next.next;` melakukan pembaruan pada pointer `next` dari node `temp`. Pointer `next` yang tadinya menunjuk ke node yang akan dihapus (`temp.next`) dialihkan untuk menunjuk ke node berikutnya lagi (`temp.next.next`). Dengan demikian, node yang berada di antara `temp` dan `temp.next.next` secara logis terputus dari rantai daftar, yang secara efektif merupakan operasi penghapusan.
- Blok kondisional `if (temp.next == null) { tail = temp; }` berfungsi untuk menangani kasus khusus. Setelah baris pertama dieksekusi, jika `temp.next` sekarang bernilai `null`, ini menandakan bahwa node yang baru saja dihapus adalah node terakhir dari daftar asli. Oleh karena itu, pointer `tail` (yang menandai akhir daftar) perlu diperbarui untuk menunjuk ke node `temp`, karena `temp` kini menjadi node terakhir dalam daftar setelah penghapusan.

TUGAS

Kode Program

MainQueue17

```
import java.util.Scanner;

public class MainQueue17 {

    public static Mahasiswa17 getMahasiswaData(Scanner input) {
        System.out.print("NIM   : ");
        String nim = input.nextLine();
        System.out.print("Nama   : ");
        String nama = input.nextLine();
        System.out.print("Kelas : ");
        String kelas = input.nextLine();

        double ipk;
        while (true) {
            System.out.print("IPK   : ");
            try {
                ipk = input.nextDouble();
                input.nextLine();
                break;
            } catch (java.util.InputMismatchException e) {
                System.out.println("Input tidak valid! Masukkan nilai angka untuk IPK
(contoh: 3.5).");
                input.nextLine();
            }
        }

        return new Mahasiswa17(nim, nama, kelas, ipk);
    }

    public static void main(String[] args) {
        QueueMahasiswa17 queue = new QueueMahasiswa17();
        Scanner sc = new Scanner(System.in);
        int choice;

        do {
            System.out.println("\n==== Menu Layanan Unit Kemahasiswaan =====");
            System.out.println("1. Cek Antrian Kosong");
            System.out.println("2. Cek Antrian Penuh");
            System.out.println("3. Tambah Antrian (Enqueue)");
            System.out.println("4. Panggil Antrian (Dequeue)");
            System.out.println("5. Lihat Antrian Terdepan");
            System.out.println("6. Lihat Antrian Paling Akhir");
            System.out.println("7. Lihat Jumlah Mahasiswa Mengantre");
            System.out.println("8. Tampilkan Seluruh Antrian");
            System.out.println("9. Kosongkan Antrian");
            System.out.println("0. Keluar");
            System.out.print("Masukkan pilihan: ");

            try {
                choice = sc.nextInt();
                sc.nextLine();

                switch (choice) {
                    case 1:
                        if (queue.isEmpty()) {
                            System.out.println("Antrian kosong.");
                        } else {
                            System.out.println("Antrian tidak kosong.");
                        }
                        break;
                    case 2:
                        System.out.println("Antrian tidak pernah penuh (berbasis linked
list).");
                        break;
                    case 3:
                        System.out.println("\nMasukkan data mahasiswa:");
                        Mahasiswa17 newMahasiswa = getMahasiswaData(sc);
                        queue.enqueue(newMahasiswa);
                        break;
                    case 4:
                        queue.dequeue();
                        break;
                    case 5:
                        queue.peekFront();
                        break;
                    case 6:
                        queue.peekBack();
                }
            }
        } while (choice != 0);
    }
}
```

```

case 6:
            queue.peekRear();
            break;
        case 7:
            System.out.println("Jumlah mahasiswa dalam antrian: " +
queue.size());
            break;
        case 8:
            queue.displayQueue();
            break;
        case 9:
            queue.clear();
            break;
        case 0:
            System.out.println("Keluar dari program.");
            break;
        default:
            System.out.println("Pilihan tidak valid.");
    }
} catch (java.util.InputMismatchException e) {
    System.out.println("Input tidak valid! Masukkan angka untuk pilihan menu.");
    sc.nextLine();
    choice = -1;
}

} while (choice != 0);

sc.close();
}
}

```

QueueMahasiswa17

```

public class QueueMahasiswa17 {
    NodeMahasiswa17 front;
    NodeMahasiswa17 rear;
    int size;

    public QueueMahasiswa17() {
        front = null;
        rear = null;
        size = 0;
    }

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

    public boolean isFull() {
        return false;
    }

    public void enqueue(Mahasiswa17 data) {
        NodeMahasiswa17 newNode = new NodeMahasiswa17(data, null);
        if (isEmpty()) {
            front = newNode;
            rear = newNode;
        } else {
            rear.next = newNode;
            rear = newNode;
        }
        size++;
        System.out.println("Mahasiswa " + data.nama + " ditambahkan ke antrian.");
    }

    public Mahasiswa17 dequeue() {
        if (isEmpty()) {
            System.out.println("Antrian kosong, tidak dapat memanggil layanan.");
            return null;
        } else {
            Mahasiswa17 served = front.data;
            front = front.next;
            if (front == null) {
                rear = null;
            }
            size--;
            System.out.println("Memanggil mahasiswa: " + served.nama);
            return served;
        }
    }
}

```

z