

JOBSHEET 11
LINKED LIST
Praktikum Algoritma Struktur Data



Krisnahadi Jayawardana
NIM. 244107060001

Kelas 1C
Jurusan Teknologi Informasi
Sistem Informasi Bisnis
POLITEKNIK NEGERI MALANG
TAHUN 2025

2.1 Pembuatan Single Linked List

Link GitHub: <https://github.com/KrisnahadiJ/Praktikum-ASD.git>

Pengerjaan praktikum menggunakan folder yang sama dengan praktikum kemarin

1. Code Class Mahasiswa11.java

```
package Pertemuan12;
.....

public class Mahasiswa11 {
    public String nim;
    public String nama;
    public String kelas;
    public double ipk;

    public Mahasiswa11(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.printf(format:"%-10s %-10s %-5s %.2f\n", nama, nim, kelas, ipk);
    }
}
```

2. Code Class NodeMahasiswa11.java

```
package Pertemuan12;
.....

class NodeMahasiswa11 {
    Mahasiswa11 data;
    NodeMahasiswa11 next;

    public NodeMahasiswa11(Mahasiswa11 data, NodeMahasiswa11 next) {
        this.data = data;
        this.next = next;
    }
}
```

3. Code Class SingleLinkedList11.Java

```
package Pertemuan12;
.....

class SingleLinkedList11 {

    NodeMahasiswa11 head;
    NodeMahasiswa11 tail;
    .....

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

    public void print() {
        | if (!isEmpty()) {
        |     NodeMahasiswa11 tmp = head;
        |     System.out.println(x:"Isi Linked List:");
        |     while (tmp != null) {
        |         | if (tmp.data != null) {
        |         |     tmp.data.tampilInformasi();
        |         | }
        |         tmp = tmp.next;
        |     }
        |     System.out.println(x:"");
        | } else {
        |     System.out.println(x:"Linked list kosong");
        | }
    }
}
```

```
package Pertemuan12;
.....

class SingleLinkedList11 {

    NodeMahasiswa11 head;
    NodeMahasiswa11 tail;
    .....

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

    public void print() {
        | if (!isEmpty()) {
        |     NodeMahasiswa11 tmp = head;
        |     System.out.println(x:"Isi Linked List:");
        |     while (tmp != null) {
        |         | if (tmp.data != null) {
        |         |     tmp.data.tampilInformasi();
        |         | }
        |         tmp = tmp.next;
        |     }
        |     System.out.println(x:"");
        | } else {
        |     System.out.println(x:"Linked list kosong");
        | }
    }
}
```

4. Code Class SLLMain11.java

```
package Pertemuan12;
.....

class SingleLinkedList11 {

    NodeMahasiswa11 head;
    NodeMahasiswa11 tail;
    .....

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

    public void print() {
        | if (!isEmpty()) {
        |     NodeMahasiswa11 tmp = head;
        |     System.out.println(x:"Isi Linked List:");
        |     while (tmp != null) {
        |         | if (tmp.data != null) {
        |         |     tmp.data.tampilInformasi();
        |         | }
        |         tmp = tmp.next;
        |     }
        |     System.out.println(x:"");
        | } else {
        |     System.out.println(x:"Linked list kosong");
        | }
    }
}
```

2.1.1 Verifikasi Hasil Percobaan

```
Users\Krisnahadi\AppData\Roaming\Code\User\workspaceStorage\raktikum-ASD_e0841313\bin Pertemuan12.SLLMain11 "
Linked list kosong
Isi Linked List:
Dirga      21212203  4D    3,60

Isi Linked List:
Dirga      21212203  4D    3,60
Alvaro     24212200  1A    4,00

Isi Linked List:
Dirga      21212203  4D    3,60
Alvaro     24212200  1A    4,00
Cintia     22212202  3C    3,50
Bimon      23212201  2B    3,80
```

2.1.2 Pertanyaan

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

Pada baris pertama, method `sll.print()` dipanggil sebelum ada data yang ditambahkan ke dalam linked list. Karena linked list baru saja diinisialisasi, atribut head dan tail dari objek `SingleLinkedList11` masih bernilai null, yang berarti linked list kosong.

Method `print()` memeriksa kondisi ini menggunakan method `isEmpty()`, dan jika `isEmpty()` mengembalikan true, maka akan mencetak pesan "Linked List Kosong". Oleh karena itu, output pertama adalah "Linked List Kosong".

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

Variabel temp digunakan sebagai pointer atau penunjuk sementara untuk menavigasi atau mengiterasi elemen-elemen dalam linked list. Dalam linked list, setiap node memiliki referensi ke node berikutnya melalui atribut next. Variabel temp biasanya diinisialisasi dengan nilai head (node pertama) dan digunakan untuk berjalan melalui linked list dengan cara mengupdate `temp = temp.next` hingga mencapai node terakhir (atau node yang memenuhi kondisi tertentu).

3. Lakukan modifikasi agar data dapat ditambahkan dari keyboard!

Memodifikasi menggunakan scanner agar menerima input dari keyboard berikut modifikasi yang saya lakukan pada class SSLMain11:

```
package Pertemuan12;

import java.util.Scanner;

public class SSLMain11 {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        SingleLinkedList11 sll = new SingleLinkedList11();
        Scanner sc = new Scanner(System.in);
        System.out.print(s:"Masukkan jumlah data yang ingin ditambahkan:");
        int jumlahData = sc.nextInt();
        sc.nextLine();
        for (int i = 0; i < jumlahData; i++) {
            System.out.println("Masukkan data ke-" + (i + 1) + ":");
            System.out.print(s:"NIM: ");
            String nim = sc.nextLine();
            System.out.print(s:"Nama: ");
            String nama = sc.nextLine();
            System.out.print(s:"Kelas: ");
            String kelas = sc.nextLine();
            System.out.print(s:"IPK: ");
            double ipk = sc.nextDouble();
            sc.nextLine();

            Mahasiswa11 mhs = new Mahasiswa11(nim, nama, kelas, ipk);
            sll.addLast(mhs);
        }

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

        sc.close();
    }
}
```

Output:

```
Masukkan jumlah data yang ingin ditambahkan:2
Masukkan data ke-1:
NIM: 1234
Nama: Krisnahadi Jayawardana
Kelas: SIB 1C
IPK: 4,89
Masukkan data ke-2:
NIM: 2345
Nama: Daffa Putra
Kelas: SIB 1C
IPK: 4,90

Data dalam Linked List:
Isi Linked List:
Krisnahadi Jayawardana 1234      SIB 1C 4,89
Daffa Putra 2345      SIB 1C 4,90
```

2.2 Modifikasi Elemen pada Single Linked List

Code Modifikasi pada Class SingleLinkedList11.java

A. Public void getData11

```
public void getData11(int index) {  
    NodeMahasiswa11 tmp = head;  
    for (int i = 0; i < index; i++) {  
        tmp = tmp.next;  
    }  
    tmp.data.tampilInformasi();  
}
```

B. Public int indexOf

```
public int indexOf(String key) {  
    NodeMahasiswa11 tmp = head;  
    int index = 0;  
    while (tmp != null && (tmp.data == null || !tmp.data.nama.equalsIgnoreCase(key))) {  
        tmp = tmp.next;  
        index++;  
    }  
  
    if (tmp == null) {  
        return -1;  
    } else {  
        return index;  
    }  
}
```

C. Public void removeFirst

```
public void removeFirst() {  
    if (isEmpty()) {  
        System.out.println(x:"Linked List masih Kosong, tidak dapat dihapus!");  
    } else if (head == tail) {  
        head = tail = null;  
    } else {  
        head = head.next;  
    }  
}
```

D. Public void removeLast

```
public void removeLast() {  
    if (isEmpty()) {  
        System.out.println(x:"Linked List masih Kosong, tidak dapat dihapus!");  
    } else if (head == tail) {  
        head = tail = null;  
    } else {  
        NodeMahasiswa11 temp = head;  
        while (temp.next != tail) {  
            temp = temp.next;  
        }  
        temp.next = null;  
        tail = temp;  
    }  
}
```

E. Public void remove

```
public void remove(String key) {
    if (isEmpty()) {
        System.out.println(x:"Linked List masih Kosong, tidak dapat dihapus!");
    } else {
        NodeMahasiswa11 temp = head;
        while (temp != null) {
            if ((temp.data.nama.equalsIgnoreCase(key)) && (temp == head)) {
                this.removeFirst();
                break;
            }
            else if (temp.next.data.nama.equalsIgnoreCase(key)) {
                temp.next = temp.next.next;
                if (temp.next == null) {
                    tail = temp;
                }
                break;
            }
            temp = temp.next;
        }
    }
}
```

F. Public void removeAt

```
public void removeAt(int index) {
    if (index == 0) {
        removeFirst();
    } else {
        NodeMahasiswa11 temp = head;
        for (int i = 0; i < index - 1; i++) {
            temp = temp.next;
        }
        temp.next = temp.next.next;
        if (temp.next == null) {
            tail = temp;
        }
    }
}
```

2.2.2 Verifikasi Hasil Percobaan

```
ptionMessages -cp C:\Users\Krisnahadi\AppData\Roaming\Co
n Pertemuan12.SLLMain11 "
Data pada indeks 1:
Cintia    22212202    3C    3,50
Data mahasiswa bernama Bimon berada pada indeks: 2

Isi Linked List:
Cintia    22212202    3C    3,50
Bimon     23212201    2B    3,80

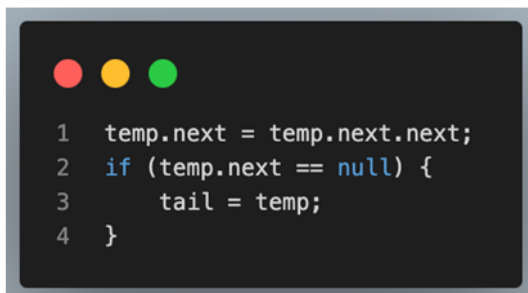
Isi Linked List:
Bimon     23212201    2B    3,80
```

2.2.3 Pertanyaan

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

Keyword break digunakan dalam fungsi remove untuk menghentikan iterasi pada loop setelah node yang sesuai dengan kriteria (berdasarkan key) ditemukan dan dihapus. Tanpa break, iterasi akan terus berjalan meskipun node yang dicari sudah ditemukan dan dihapus, yang dapat menyebabkan kesalahan logika atau penghapusan node yang tidak diinginkan. Dengan menggunakan break, program memastikan bahwa hanya node pertama yang cocok dengan key yang akan dihapus, dan proses iterasi dihentikan segera setelah operasi penghapusan selesai.

2. Jelaskan kegunaan kode dibawah pada method remove



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

Kode temp.next = temp.next.next; digunakan untuk melewati node yang akan dihapus dengan menghubungkan node saat ini (temp) langsung ke node setelah node yang dihapus. Hal ini secara efektif menghapus node dari linked list. Kondisi if (temp.next == null) memeriksa apakah node yang dihapus adalah node terakhir. Jika iya, pointer tail diperbarui untuk menunjuk ke node saat ini (temp), memastikan konsistensi struktur linked list.