LAPORAN PRAKTIKUM ALGORITMA DAN STRUKTUR

SINGLE LINKED LIST

ABIM MUSTAWA

244107020078

KELAS TI-1B

PRODI D-IV TEKNIK

INFORMATIKA JURUSAN

TEKNOLOGI INFORMASI

POLITEKNIK NEGERI

MALANG 2025

1. Percobaan Praktikum

1.1 Praktikum 1 – Pembuatan Single Linked List

Didalam praktikum ini, kita akan mempraktikkan bagaimana membuat Single Linked List dengan representasi data berupa Node, pengaksesan linked list dan metode penambahan data.

Langkah-langkah Percobaan

1. Pada Project yang sudah dibuat pada Minggu sebelumnya. Buat folder atau package baru bernama Jobsheet11 di dalam repository Praktikum ASD.
2. Tambahkan class-class berikut:
   a. Mahasiswa00.java
   b. Node00.java
   c. SingleLinkedList00.java
   d. SLLMain00.java Ganti 00 dengan nomer Absen Anda
3. Implementasikan Class Mahasiswa00 sesuai dengan diagram class berikut ini :
   Mahasiswa
   - nim: String
   - nama: String
   - kelas: String
   - ipk: double
   - Mahasiswa()
   - Mahasiswa(nm: String, name: String, kls: String, ip: double)
   - tampilInformasi(): void

```java
public class Mahasiswa01 {
    String nim, nama, kelas;
    double ipk;

    public Mahasiswa01(){

    }

    public Mahasiswa01(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(nim + " - " + nama + " - " + kelas + " - " + ipk);
    }
}
```

4. Implementasi class Node seperti gambar berikut ini

```java
public class Node01 {
    Mahasiswa01 data;
    Node01 next;

    public Node01(Mahasiswa01 data, Node01 next) {
        this.data = data;
        this.next = next;
    }
}
```

5. Tambahkan attribute head dan tail pada class SingleLinkedList

```java
public class SingleLinkedList01 {
    Node01 head;
    Node01 tail;
```

6. Sebagai langkah berikutnya, akan diimplementasikan method-method yang terdapat pada SingleLinkedList.
7. Tambahkan method isEmpty().

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

8. Implementasi method untuk mencetak dengan menggunakan proses traverse.

```java
public void print() {
    if (!isEmpty()) {
        Node01 tmp = head;
        System.out.println(x:"Isi Linked List:\t ");
        while (tmp != null) {
            tmp.data.tampilInformasi();
            tmp = tmp.next;
        }
        System.out.println(x:"");
    }else {
        System.out.println(x:"Linked List Kosong");
    }
}
```

9. Implementasikan method addFirst().

```
public void addFirst(Mahasiswa01 input){
    Node01 ndinput = new Node01(input, next:null);
    if (isEmpty()) {
        head = ndinput;
        tail = ndinput;
    }else {
        ndinput.next = head;
        head = ndinput;
    }
}
```

10. Implementasikan method addLast().

```
public void addLast(Mahasiswa01 input){
    Node01 ndinput = new Node01(input, next:null);
    if (isEmpty()) {
        head = ndinput;
        tail = ndinput;
    }else {
        tail.next = ndinput;
        tail = ndinput;
    }
}
```

11. Implementasikan method insertAfter, untuk memasukkan node yang memiliki data input setelah node yang memiliki data key.

```java
public void insertAfter(String key, Mahasiswa01 input) {
    Node01 ndinput = new Node01(input, next:null);
    Node01 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);
}
```

12. Tambahkan method penambahan node pada indeks tertentu.

```java
public void insertAt(int index, Mahasiswa01 input){
    if (index < 0) {
        System.out.println(x:"indeks salah");
    }else if (index == 0) {
        addFirst(input);
    }else {
        Node01 temp = head;
        for (int i = 0; i < index - 1; i++) {
            temp = temp.next;
        }
        temp.next = new Node01(input, temp.next);
        if (temp.next.next == null) {
            tail = temp.next;
        }
    }
}
```

13. Pada class SLLMain00, buatlah fungsi main, kemudian buat object dari class SingleLinkedList.

```
public class SLLMain01 {
    Run | Debug
    public static void main(String[] args) {
        SingleLinkedList01 sll = new SingleLinkedList01();
```

14. Buat empat object mahasiswa dengan nama mhs1, mhs2, mhs3, mhs4 kemudian isi data setiap object melalui konstruktor.

```
Mahasiswa01 mhs1 = new Mahasiswa01(nim:"24212200",nama:"Alvaro", kelas:"1A", ipk:4.0);
Mahasiswa01 mhs2 = new Mahasiswa01(nim:"23212201",nama:"Bimon", kelas:"2B", ipk:3.8);
Mahasiswa01 mhs3 = new Mahasiswa01(nim:"22212202",nama:"Cintia", kelas:"3C", ipk:3.5);
Mahasiswa01 mhs4 = new Mahasiswa01(nim:"21212203",nama:"Dirga", kelas:"4D", ipk:3.6);
```

15. Tambahkan Method penambahan data dan pencetakan data di setiap penambahannya agar terlihat perubahannya

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

Verifikasi Hasil Percobaan

```
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
```

Pertanyaan

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

Jawab:
Karena ada kode sll.print() paling atas sebelum ada data yang ditambahkan.

2. Jelaskan kegunaan variable temp secara umum pada setiap method!
   Jawab:
   Variabel temp digunakan sebagai point untuk menelusuri linked list dari head sampai tail.

3. Lakukan modifikasi agar data dapat ditambahkan dari keyboard!
   Jawab:

```
do {
    System.out.println(x:"==== MENU ===");
    System.out.println(x:"1. Tambah Data di Awal");
    System.out.println(x:"2. Tambah Data di Akhir");
    System.out.println(x:"3. Tambah Data Setelah Node Tertentu");
    System.out.println(x:"4. Tambah Data Pada Index Tertentu");
    System.out.println(x:"5. Tampilkan Data");
    System.out.println(x:"6. Exit");
    System.out.print(s:"Pilih Menu : ");
    pilihan = sc.nextInt();
    sc.nextLine();
    switch (pilihan) {
        case 1:
            System.out.println(x:"-- Tambah Data di Awal --");
            System.out.print(s:"Nama : ");
            nama = sc.nextLine();
            System.out.print(s:"NIM : ");
            nim = sc.nextLine();
            System.out.print(s:"Kelas : ");
            kelas = sc.nextLine();
            System.out.print(s:"IPK : ");
            ipk = sc.nextDouble();
            sc.nextLine();

            Mahasiswa01 mhsFirst = new Mahasiswa01(nim, nama, kelas, ipk);
            sll.addFirst(mhsFirst);
            System.out.println(x:"Data Berhasil Ditambahkan di Awal!");
            break;
        case 2:
            System.out.println(x:"-- Tambah Data di Akhir --");
            System.out.print(s:"Nama : ");
            nama = sc.nextLine();
            System.out.print(s:"NIM : ");
            nim = sc.nextLine();
            System.out.print(s:"Kelas : ");
```

```
        kelas = sc.nextLine();
        System.out.print(s:"IPK : ");
        ipk = sc.nextDouble();
        sc.nextLine();

        Mahasiswa01 mhsFirst = new Mahasiswa01(nim, nama, kelas, ipk);
        sll.addFirst(mhsFirst);
        System.out.println(x:"Data Berhasil Ditambahkan di Awal!");
        break;
    case 2:
        System.out.println(x:"-- Tambah Data di Akhir --");
        System.out.print(s:"Nama : ");
        nama = sc.nextLine();
        System.out.print(s:"NIM : ");
        nim = sc.nextLine();
        System.out.print(s:"Kelas : ");
        kelas = sc.nextLine();
        System.out.print(s:"IPK : ");
        ipk = sc.nextDouble();
        sc.nextLine();

        Mahasiswa01 mhsLast = new Mahasiswa01(nim, nama, kelas, ipk);
        sll.addLast(mhsLast);
        System.out.println(x:"Data Berhasil Ditambahkan di Akhir!");
        break;
    case 3:
        System.out.println(x:"-- Tambah Data Setelah Node Tertentu --");
        sll.print();

        System.out.print(s:"Masukakan Nama Mahasiswa setelahnya : ");
        String key = sc.nextLine();

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

        Mahasiswa01 mhsAfter = new Mahasiswa01(nim, nama, kelas, ipk);
        sll.insertAfter(key, mhsAfter);
        System.out.println("Data Berhasil Ditambahkan Setelah " + key );
        break;
    case 4:
        System.out.println(x:"-- Tambah Data Setelah Node Tertentu --");
        sll.print();

        System.out.print(s:"Masukakan Index (mulai dari 0) : ");
        int index = sc.nextInt();
        sc.nextLine();

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

        Mahasiswa01 mhsAtIndex = new Mahasiswa01(nim, nama, kelas, ipk);
        sll.insertAt(index, mhsAtIndex);
        System.out.println("Data Berhasil Ditambahkan di Index " + index );
        break;
```

```
    case 5:
        System.out.println(x:"-- Data Mahasiswa --");
        sll.print();
        break;
    default:
        if (pilihan == 6) {
            System.out.println(x:"Terima Kasih");
        }else {
            System.out.println(x:"Pilihan Tidak Valid");
        }
        break;
}
ile (pilihan != 6);
```

1.2 Praktikum 2 – Modifikasi Elemen pada Single Linked List

Didalam praktikum ini, kita akan mempraktekkan bagaimana mengakses elemen, mendapatkan indeks dan melakukan penghapusan data pada Single Linked List.:

Langkah-langkah Percobaan

1. Implementasikan method untuk mengakses data dan indeks pada linked list
2. Tambahkan method untuk mendapatkan data pada indeks tertentu pada class Single Linked List

```
public void getData(int index){
    Node01 tmp = head;
    for (int i = 0; i < index; i++) {
        tmp = tmp.next;
    }
    tmp.data.tampilInformasi();
}
```

3. Implementasikan method indexOf.

```
public int indexOf(String key){
    Node01 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;
    }
}
```

4. Tambahkan method removeFirst pada class SingleLinkedList

```
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;
    }
}
```

5. Tambahkan method untuk me nghapus data pada bagian belakang pada class SingleLinkedList

```
public void removeLast(){
    if (isEmpty()) {
        System.out.println(x:"Linked List masih kosong, tidak dapat dihapus!");
    }else if (head == tail) {
        head = tail = null;
    }else {
        Node01 temp = head;
        while (temp.next != tail) {
            temp = temp.next;
        }
        temp.next = null;
        tail = temp;
    }
}
```

6. Sebagai langkah berikutnya, akan diimplementasikan method remove

```
public void remove(String key){
    if (isEmpty()) {
        System.out.println(x:"Linked List masih kosong, tidak dapat dihapus!");
    }else {
        Node01 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;
        }
    }
}
```

7. Implementasi method untuk menghapus node dengan menggunakan index.

```java
public void removeAt(int index) {
    if (index == 0) {
        removeFirst();
    }else {
        Node01 temp = head;
        for (int i = 0; i < index; i++) {
            temp = temp.next;
        }
        temp.next = temp.next.next;
        if (temp.next == null) {
            tail = temp;
        }
    }
}
```

8. Kemudian, coba lakukan pengaksesan dan penghapusan data di method main pada class SLLMain dengan menambahkan kode berikut

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

System.out.println("data mahasiswa an Bimon berada pada index : " + sll.indexOf(key:"bimon"));
System.out.println();

sll.removeFirst();
sll.removeLast();
sll.print();
sll.removeAt(index:0);
sll.print();
```

9. Jalankan class SLLMain

Verifikasi Hasil Percobaan

```
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!
   Jawab:
   Untuk menghentikan pencarian ketika node ditemukan dan dihapus. Apabila tidak ada break maka perulangan terus berjalan dan bisa terjadi menghapus node lain dengan nama yang sama.

2. Jelaskan kegunaan kode dibawah pada method remove
   ```
   temp.next = temp.next.next;
   if (temp.next == null) {
      tail = temp;
   }
   ```
   Jawab:
   temp.next = temp.next.next;
   Kode ini menjelaskan node setelah temp akan di lewati, sehingga node tersebut dihapus dari linked list
   if (temp.next == null) {
   tail = temp;
   }
   Kode ini mengecek apakah node yang baru saja dihapus adalah node terakhir atau bukan, jika iya maka temp akan menjadi tail

2. Latihan Praktikum

2.1 Latihan 1

Buatlah implementasi program antrian layanan unit kemahasiswaan sesuai dengan berikut ini :

a. Implementasi antrian menggunakan Queue berbasis Linked List!
b. Program merupakan proyek baru bukan modifikasi dari percobaan
c. Ketika seorang mahasiswa akan mengantri, maka dia harus mendaftarkan datanya
d. Cek antrian kosong, Cek antrian penuh, Mengosongkan antrian.
e. Menambahkan antrian
f. Memanggil antrian
g. Menampilkan antrian terdepan dan antrian paling akhir
h. Menampilkan jumlah mahasiswa yang masih mengantre.

Jawab: