JOBSHEET
XI LINKED
LIST

1. Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Membuat struktur data linked list
2. Membuat linked list pada program
3. Membedakan permasalahan apa yang dapat diselesaikan menggunakan linked list

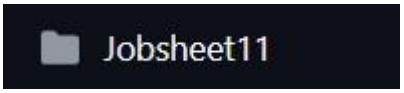
2. Praktikum

2.1 Pembuatan Single Linked List

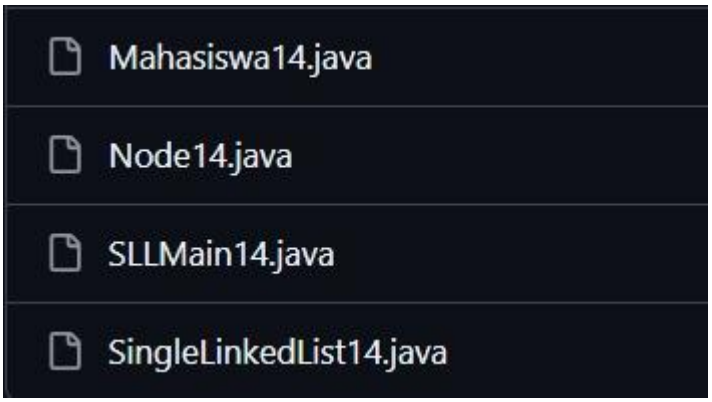
Waktu percobaan : 30 menit

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

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 Mahasiswa14 {
    String nim, nama, kelas;
    double ipk;

    Mahasiswa14(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(nama + "\t" + nim + "\t" + kelas + "\t" + ipk);
}

}
```

4. Implementasi class Node seperti gambar berikut ini

```java
public class Node14 {
    Mahasiswa14 data;
    Node14 next;

    public Node14(Mahasiswa14 data, Node14 next) {
        this.data = data;
        this.next = next;
    }
}
```

5. Tambahkan attribute **head dan tail** pada class SingleLinkedList

```java
public class SingleLinkedList14 {

    Node14 head;
    Node14 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()) {
        Node14 tmp = head;
        System.out.println(x:"isi linked lis :\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()`**.

```java
public void addFirst(Mahasiswa14 input) {
    Node14 NdInput = new Node14(input, null);
    if (isEmpty()) {
        head = NdInput;
        tail = NdInput;
    } else {
        NdInput.next = head;
        head = NdInput;
    }
}
```

10. Implementasikan method **`addLast()`**.

```java
public void addLast(Mahasiswa14 input) {
    Node14 NdInput = new Node14(input, 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, Mahasiswa14 input) {
    Node14 NdInput = new Node14(input, null);
    Node14 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, Mahasiswa14 input) {
    if (index < 0) {
        System.out.println(x:"index salah");
    }else if (index == 0) {
        addFirst(input);
    } else {
        Node14 temp = head;
        for (int i = 0; i < index - 1; i++) {
            temp = temp.next;
        }
        temp.next = new Node14(input, temp.next);
        if (temp.next.next == null) {
            tail = temp.next;
        }
    }
}
```

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

```
SingledLinkedList14 sll = new SingledLinkedList14();
```

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

```
Mahasiswa14 mhs1 = new Mahasiswa14(nim:"21212203", nama:"Dirga", kelas:"4D", ipk:3.6);
Mahasiswa14 mhs2 = new Mahasiswa14(nim:"24212200", nama:"Alvaro", kelas:"1A", ipk:4.0);
Mahasiswa14 mhs3 = new Mahasiswa14(nim:"22212202", nama:"Cintia", kelas:"3C", ipk:3.5);
Mahasiswa14 mhs4 = new Mahasiswa14(nim:"23212201", nama:"Bimon", kelas:"2B", ipk:3.8);
```

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

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

2.1.1 Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.

```
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
```

```
Linked list kosong.
isi linked lis :
Dirga   21212203        4D      3.6

isi linked lis :
Dirga   21212203        4D      3.6
Alvaro  24212200        1A      4.0

isi linked lis :
Dirga   21212203        4D      3.6
Cintia  22212202        3C      3.5
Bimon   23212201        2B      3.8
Alvaro  24212200        1A      4.0
```

2.1.2 Pertanyaan

1. Mengapa hasil compile kode program di baris pertama menghasilkan “Linked List Kosong”?
2. Jelaskan kegunaan variable temp secara umum pada setiap method!
3. Lakukan modifikasi agar data dapat ditambahkan dari keyboard!

Jawaban Tugas

1. karena fugsi print() sudah di panggil sebelum data masuk ke lingked listnya

2. karena temp adalah variabel sementara yang digunakan untuk menelusuri setiap node di lingked list , jika langsung menggunakan head atau tail maka data akan hilang karena tidak aanya akses dikarenakan head/tail berpindah

3.

2.2 Modifikasi Elemen pada Single Linked List

Waktu percobaan : 30 menit

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

2.2.1 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) {
    Node14 tmp = head;
    for (int i = 0; i < index; i++) {
        tmp = tmp.next;
        }
        tmp.data.tampilInformasi();
    }
```

3. Implementasikan method **`indexOf`**.

```
public int indexOf(String key) {
        Node14 temp = head;
        int index = 0;
        while (temp != null && !temp.data.nama.equalsIgnoreCase(key)) {
            temp = temp.next;
            index++;
        }
        if (temp == 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 menghapus 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 {
        Node14 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 {
    Node14 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.

```
public void removeAt(int index) {
    if (index == 0) {
        removeFirst();
    } else {
        Node14 temp = head;
        for (int i = 0; i < index - 1; 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

```
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

2.2.2 Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.

```
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
```

```
data index 1:
Cintia  22212202        3C      4.0
data mahasiswa an Bimon berada pada index: -1

isi linked lis :
Cintia  22212202        3C      4.0
```

2.2.3 Pertanyaan

1. Mengapa digunakan keyword break pada fungsi remove? Jelaskan!
2. Jelaskan kegunaan kode dibawah pada method remove

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

Jawaban Pertanyaan

1. supaya ketika menemukan node yang equals dengan key nya, maka perulangan akan berhenti agar tidak lanjut mencari ke node-node berikutnya yang tidak perlu

2. temp.next = temp.next.next itu di gunakan misalnya ada node A - B - C, terus kita mau hapus node B kita tinggal bilang ke A“Eh A, abis kamu langsung ke C aja ya, jangan ke B lagi”

temp adalah node sebelum yang mau dihapus. agar langsung loncat ke node setelahnya, jadi node yang mau dihapus otomatis nggak ikut nyambung lagi.

sedangkan baris setelahnya - akhir
berfungsi untuk memperbarui posisi tail. jika setelah penghapusan ternyata temp.next bernilai null itu berarti node yang baru saja dihapus adalah node terakhir. maka node temp sekarang menjadi node terakhir yang baru sehingga tail perlu diperbarui untuk menunjuk ke temp.

3. Tugas

Waktu pengerjaan : 50 menit

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.

Jawaban Tugas

//FILE MAIN

```
import java.util.Scanner;

public class main14 {
    public static void main(String[] args) {
        layanan14 ukm = new layanan14();
        Scanner sc = new Scanner(System.in);
        int pilih;
        String nama;

        do {
            System.out.println("layanan");
            System.out.println("1 Tambah antrian");
            System.out.println("2. Panggil antrian");
            System.out.println("3. Lihat terdepan");
            System.out.println("4. Lihat terakhir");
            System.out.println("5. Cek apakah antrian kosong");
            System.out.println("6. Kosongkan");
            System.out.println("7. jumlah");
            System.out.println("8. Tampil semua");
            System.out.println("0. Keluar");
            System.out.print("Pilih ");
            pilih = sc.nextInt(); sc.nextLine();
            switch (pilih) {
                case 1:
                    System.out.print("nama: ");
                    nama = sc.nextLine();
                    ukm.tambah(nama);
                    break;
```

```
                case 2:
                    ukm.panggil();
                    break;
                case 3:
                    ukm.tampilkanDepan();
                    break;
                case 4:
                    ukm.tampilkanBelakang();
                    break;
                case 5:
                    System.out.println(ukm.kosong() ? "Antrian
kosong" : "ada antrian.");
                    break;
                case 6:
                    ukm.kosongkan();
                    break;
                case 7:
                    ukm.tampilkanJumlah();
                    break;
                case 8:
                    ukm.tampilkanSemua();
                    break;
                case 0:
                    System.out.println("degradasi yahaha hayuuk emyu
gua kalah");
                    break;
                default:
                    System.out.println("tdk valid");
            }
        } while (pilih != 0);
    }
}
```

//FILE LAYANAN

```
public class layanan14 {
    nodee14 depan, belakang;
    int jumlah;
    public layanan14() {
        depan = belakang = null;
        jumlah = 0;
    }
```

```
    public boolean kosong() {
        return depan == null;
    }
    public boolean penuh() {
        return false;
    }
    public void tambah(String nama) {
        nodee14 baru = new nodee14(nama);
        if (kosong()) {
            depan = belakang = baru;
        } else {
            belakang.next = baru;
            belakang = baru;
        }
        jumlah++;
        System.out.println(nama + "  masuk antrian");
    }
    public void panggil() {
        if (kosong()) {
            System.out.println("Antrian koson");
        } else {
            System.out.println("Memanggil: " + depan.nama);
            depan = depan.next;
            jumlah--;
            if (depan == null) belakang = null;
        }
    }
    public void tampilkanDepan() {
        if (!kosong()) {
            System.out.println("Antrian terdepan: " + depan.nama);
        } else {
            System.out.println("Antrian kosong");
        }
    }
    public void tampilkanBelakang() {
        if (!kosong()) {
            System.out.println("Antrian terakhir: " + belakang.nama);
        } else {
            System.out.println("Antrian kosong");
        }
    }
    public void tampilkanJumlah() {
```

```
        System.out.println("Jumlah mengantre: " + jumlah);
    }
    public void kosongkan() {
        depan = belakang = null;
        jumlah = 0;
        System.out.println("kosongkan");
    }
    public void tampilkanSemua() {
        if (kosong()) {
            System.out.println("Antrian kosong.");
        } else {
            System.out.println("mhs dlm anri : ");
            nodee14 bantu = depan;
            while (bantu != null) {
                System.out.println("- " + bantu.nama);
                bantu = bantu.next;
            }
        }
    }
}
```

//FILE NODEE

```
public class nodee14 {
    String nama;
    nodee14 next;

    public nodee14(String nama) {
        this.nama = nama;
        this.next = null;
    }
}
```