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

4. Implementasi class Node seperti gambar berikut ini

```java
public class NodeMahasiswa00 {
    Mahasiswa00 data;
    NodeMahasiswa00 next;

    public NodeMahasiswa00(Mahasiswa00 data, NodeMahasiswa00 next) {
        this.data = data;
        this.next = next;
    }
}
```

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

```java
NodeMahasiswa00 head;
NodeMahasiswa00 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()) {
        NodeMahasiswa00 tmp = head;
        System.out.print(s:"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()`**.

```java
public void addFirst(Mahasiswa00 input) {
    NodeMahasiswa00 ndInput = new NodeMahasiswa00(input, next:null);
    if (isEmpty()) {
        head = ndInput;
        tail = ndInput;
    } else {
        ndInput.next = head;
        head = ndInput;
    }
}
```

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

```java
public void addLast(Mahasiswa00 input) {
    NodeMahasiswa00 ndInput = new NodeMahasiswa00(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, Mahasiswa00 input) {
    NodeMahasiswa00 ndInput = new NodeMahasiswa00(input, next:null);
    NodeMahasiswa00 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, Mahasiswa00 input) {
    if (index < 0) {
        System.out.println(x:"indeks salah");
    } else if (index == 0) {
        addFirst(input);
    } else {
        NodeMahasiswa00 temp = head;
        for (int i = 0; i < index - 1; i++) {
            temp = temp.next;
        }
        temp.next = new NodeMahasiswa00(input, temp.next);
        if (temp.next.next == null) {
            tail = temp.next;
        }
    }
}
```

13. Pada class SLLMain00, buatlah fungsi **`main`**, kemudian buat object dari class SingleLinkedList.
14. Buat empat object mahasiswa dengan nama mhs1, mhs2, mhs3, mhs4 kemudian isi data setiap object melalui konstruktor.
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();
```

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
```

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!

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) {
    NodeMahasiswa00 tmp = head;
    for (int i = 0; i < index; i++) {
        tmp = tmp.next;
    }
    tmp.data.tampilInformasi();
}
```

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

```
public int indexOf(String key) {
    NodeMahasiswa00 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 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 {
        NodeMahasiswa00 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 {
        NodeMahasiswa00 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 {
        NodeMahasiswa00 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
```

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   }
```

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.