Nama : Dicky Darmawan
Kelas : TI 1B
No. Absen : 08
NIM : 244107020037

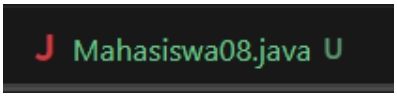
Pembuatan Single Linked List

Waktu percobaan: 30 menit

Langkah-langkah:

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

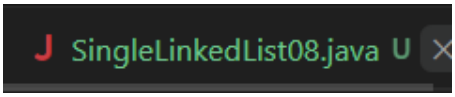
1. Pada Project yang sudah dibuat pada Minggu sebelumnya. Buat folder atau package baru bernama Jobsheet11 di dalam repository Praktikum ASD.
2. Tambahkan class-class berikut:
   a. Mahasiswa00.java

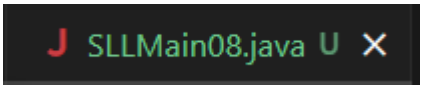
   b. Node00.java

   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

```
String nim, nama, kelas;
double ipk;

public Mahasiswa08() {

}

public Mahasiswa08(String nm, String name, String kls, double ip) {
    nim = nm;
    nama = name;
    kelas = kls;
    ipk = ip;
}

public void tampilInformasi() {

}
```

4. Implementasi class Node seperti gambar berikut ini

```
public class Node08 {

    Mahasiswa08 data;
    Node08 next;

    public Node08(Mahasiswa08 data, Node08 next) {
        this.data = data;
        this.next = next;
    }
}
```

5. Tambahkan attribute head dan tail pada class SingleLinkedList

```
public class SingleLinkedList08 {

    Node08 head;
    Node08 tail;

}
```

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

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

8. Implementasi method untuk mencetak dengan menggunakan proses traverse.

```
public void print() {
    if (!isEmpty()) {
        Node08 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(Mahasiswa08 input) {
    Node08 ndInput = new Node08(input, next:null);
    if (isEmpty()) {
        head = ndInput;
        tail = ndInput;
    } else {
        ndInput.next = head;
        head = ndInput;
    }
}
```

10. Implementasikan method addLast().

```
public void addLast(Mahasiswa08 input) {
    Node08 ndInput = new Node08(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.

```
public void insertAfter(String key, Mahasiswa08 input) {
    Node08 ndInput = new Node08(input, next:null);
    Node08 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.

```
public void insertAt(int index, Mahasiswa08 input) {
    if (index < 0) {
        System.out.println(x:"Index salah");
    } else if (index == 0) {
        addFirst(input);
    } else {
        Node08 temp = head;
        for (int i = 0; i < index - 1; i++) {
            temp = temp.next;
        }
        temp.next = new Node08(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 static void main(String[] args) {
    SingleLinkedList08 sll = new SingleLinkedList08();
```

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

```
Mahasiswa08 mhs1 = new Mahasiswa08(nm:"244107040390", name:"Alvaro", kls
Mahasiswa08 mhs2 = new Mahasiswa08(nm:"244107040391", name:"Kemal", kls:
Mahasiswa08 mhs3 = new Mahasiswa08(nm:"244107040392", name:"Ayu", kls:"
Mahasiswa08 mhs4 = new Mahasiswa08(nm:"244107040393", name:"Dirga", kls:
```

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   244107040393    2A      3.8

Isi Linked List:
Dirga   244107040393    2A      3.8
Alvaro  244107040390    2A      3.8

Isi Linked List:
Dirga   244107040393    2A      3.8
Ayu     244107040392    2A      3.7
Kemal   244107040391    2A      3.5
Alvaro  244107040390    2A      3.8
```

Pertanyaan

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

   Karena method yang pertama kali dilakukan adalah menampilkan isi dari linked list, sementara belum dilakukan method penambahan data pada linked list tersebut.

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

   Variabel temp adalah variabel sementara yang digunakan untuk menduplikasi head. Ketika ingin menambahkan, menghapus, mencetak data dari linked list, maka yang perlu digerakkan adalah variabel temp. Hal ini untuk mengatasi pergerakan head, jika head yang digerakkan maka data yang ada di head sebelumnya akan langsugn terhapus.

3. Lakukan modifikasi agar data dapat ditambahkan dari keyboard!

```
Tambahkan Data Mahasiswa
NIM   : 244107020037
Nama  : Dicky Darmawan
Kelas : 1B
IPK   : 4,0
Ingin menambah data lagi? (Y/N): n
Isi Linked List:
Dicky Darmawan  244107020037    1B      4.0
```

Modifikasi Elemen pada Single Linked List

Waktu Percobaan: 30 menit

Langkah-langkah:

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) {
    Node08 temp = head;
    for (int i =0 ; i < index; i++) {
        temp = temp.next;
    }
    temp.data.tampilInformasi();
}
```

3. Implementasikan method indexOf.

```
public int indexOf(String key) {
    Node08 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 di
    } 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 diha
    } else if (head == tail) {
        head = tail = null;
    } else {
        Node08 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 dih
    } else {
        Node08 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 {
        Node08 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 Dicky berada pada index: " + sll.indexOf(key:"dicky"));
System.out.println();

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

9. Jalankan class SLLMain

Verifikasi Hasil Percobaan

```
Isi Linked List:
Ahmad   55555555        1D      3.7
Amin    44444444        3D      3.8
Ana     33333333        1A      3.6
Dicky   22222222        1B      3.9
Abel    11111111        2A      3.9

Data index 1 :
Amin    44444444        3D      3.8
Data mahasiswa an Dicky berada pada index: 3

Isi Linked List:
Amin    44444444        3D      3.8
Ana     33333333        1A      3.6
Dicky   22222222        1B      3.9

Isi Linked List:
Ana     33333333        1A      3.6
Dicky   22222222        1B      3.9
```

Pertanyaan

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

   Karena break memastikan program keluar dari loop tepat saat penghapusan selesai. Tanpa break, loop while akan berlanjut ke node berikutnya meskipun data sudah ditemukan dan dihapus

2. Jelaskan kegunaan kode dibawah pada method remove

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

   Kegunaan dari kode di atas adalah untuk melewati 1 node atau menghapus node pada index tertentu. Jika variabel next pada suatu node menunjukkan null, maka node tersebut akan dijadikan sebagai node.

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.

   Output antrian penuh (Saya set maksimal 5 mahasiswa):

```
== LAYANAN MAHASISWA ==
1. Tambahkan Data Antrian
2. Proses Antrian
3. Tampilkan Antrian Terdepan
4. Tampilkan Antrian Terbelakang
5. Tampilkan Semua Antrian
6. Tampilkan Jumlah Antrian
7. Kosongkan Antrian
0. Keluar
Pilih: 1
Antrian telah penuh!
```

   Mengosongkan antrian:

```
== LAYANAN MAHASISWA ==
1. Tambahkan Data Antrian
2. Proses Antrian
3. Tampilkan Antrian Terdepan
4. Tampilkan Antrian Terbelakang
5. Tampilkan Semua Antrian
6. Tampilkan Jumlah Antrian
7. Kosongkan Antrian
0. Keluar
Pilih: 7
Berhasil Mengosongkan Antrian
```

```
== LAYANAN MAHASISWA ==
1. Tambahkan Data Antrian
2. Proses Antrian
3. Tampilkan Antrian Terdepan
4. Tampilkan Antrian Terbelakang
5. Tampilkan Semua Antrian
6. Tampilkan Jumlah Antrian
7. Kosongkan Antrian
0. Keluar
Pilih: 5
Antrian Kosong!
```

e. Menambahkan antrian:

```
== LAYANAN MAHASISWA ==
1. Tambahkan Data Antrian
2. Proses Antrian
3. Tampilkan Antrian Terdepan
4. Tampilkan Antrian Terbelakang
5. Tampilkan Semua Antrian
6. Tampilkan Jumlah Antrian
7. Kosongkan Antrian
0. Keluar
Pilih: 1
Masukkan Data
Nama  : Rara
NIM   : 678
Kelas : 2F
Data Berhasil Ditambahkan!
```

f. Memanggil antrian:

```
== LAYANAN MAHASISWA ==
1. Tambahkan Data Antrian
2. Proses Antrian
3. Tampilkan Antrian Terdepan
4. Tampilkan Antrian Terbelakang
5. Tampilkan Semua Antrian
6. Tampilkan Jumlah Antrian
7. Kosongkan Antrian
0. Keluar
Pilih: 2
Berhasil memproses antrian dari Dicky
```

```
== LAYANAN MAHASISWA ==
1. Tambahkan Data Antrian
2. Proses Antrian
3. Tampilkan Antrian Terdepan
4. Tampilkan Antrian Terbelakang
5. Tampilkan Semua Antrian
6. Tampilkan Jumlah Antrian
7. Kosongkan Antrian
0. Keluar
Pilih: 5
Antrian:
Nama    NIM     Kelas
--------------------
Ahmad   234     1A
Marsha  345     2A
Bayu    567     2C
Rara    678     2F
```

g. Menampilkan antrian terdepan dan antrian paling akhir:

```
== LAYANAN MAHASISWA ==
1. Tambahkan Data Antrian
2. Proses Antrian
3. Tampilkan Antrian Terdepan
4. Tampilkan Antrian Terbelakang
5. Tampilkan Semua Antrian
6. Tampilkan Jumlah Antrian
7. Kosongkan Antrian
0. Keluar
Pilih: 3
Antrian terdepan adalah
Ahmad   234     1A
```

```
== LAYANAN MAHASISWA ==
1. Tambahkan Data Antrian
2. Proses Antrian
3. Tampilkan Antrian Terdepan
4. Tampilkan Antrian Terbelakang
5. Tampilkan Semua Antrian
6. Tampilkan Jumlah Antrian
7. Kosongkan Antrian
0. Keluar
Pilih: 4
Antrian terbelakang adalah
Rara    678     2F
```

h. Menampilkan jumlah mahasiswa yang masih mengantre:

```
== LAYANAN MAHASISWA ==
1. Tambahkan Data Antrian
2. Proses Antrian
3. Tampilkan Antrian Terdepan
4. Tampilkan Antrian Terbelakang
5. Tampilkan Semua Antrian
6. Tampilkan Jumlah Antrian
7. Kosongkan Antrian
0. Keluar
Pilih: 6
Jumlah antrian: 4 mahasiswa
```