**LAPORAN HASIL PRAKTIKUM ALSD**

**JOBSHEET 11 LINKED LIST**



Faiva Puspa Sahara

244107020036

TI – 1E

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

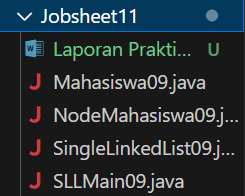
POLITEKNIK NEGERI MALANG

2025

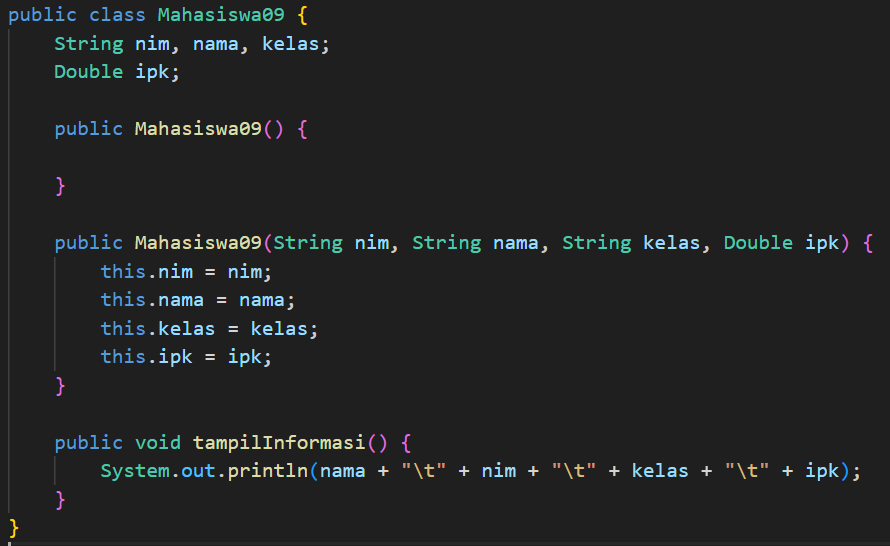
**HASIL PRAKTIKUM**

* 1. **Percobaan 1 (Pembuatan Single Linked List)**

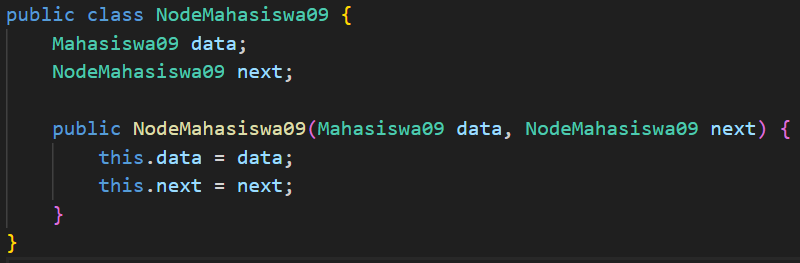
1. Buat folder baru bernama **Jobsheet11**. Buat file baru, tambahkan class **Mahasiswa09.java, NodeMahasiswa09.java, SingleLinkedList09.java, SLLMain09.java**



1. Class **Mahasiswa09.java**



1. Class **NodeMahasiswa09.java**



1. Class **SingleLinkedList09.java**

public class SingleLinkedList09 {

NodeMahasiswa09 head;

NodeMahasiswa09 tail;

boolean isEmpty() {

return head == null;

}

public void print() {

if (!isEmpty()) {

System.out.println("Isi Linked List:");

NodeMahasiswa09 tmp = head;

while (tmp != null) {

tmp.data.tampilInformasi();

tmp = tmp.next;

}

System.out.println("");

} else {

System.out.println("Linked List kosong");

}

}

public void addFirst(Mahasiswa09 input) {

NodeMahasiswa09 ndInput = new NodeMahasiswa09(input, null);

if (isEmpty()) {

head = ndInput;

tail = ndInput;

} else {

ndInput.next = head;

head = ndInput;

}

}

public void addLast(Mahasiswa09 input) {

NodeMahasiswa09 ndInput = new NodeMahasiswa09(input, null);

if (isEmpty()) {

head = ndInput;

tail = ndInput;

} else {

tail.next = ndInput;

tail = ndInput;

}

}

public void insertAfter(Mahasiswa09 input, String key) {

NodeMahasiswa09 ndInput = new NodeMahasiswa09(input, null);

NodeMahasiswa09 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);

}

public void insertAt(int index, Mahasiswa09 input) {

if (index < 0) {

System.out.println("Indeks salah");

} else if (index == 0) {

addFirst(input);

} else {

NodeMahasiswa09 temp = head;

for (int i = 0; i < index - 1; i++) {

temp = temp.next;

}

temp.next = new NodeMahasiswa09(input, temp.next);

if (temp.next.next == null) {

tail = temp.next;

}

}

}

}

public void insertAt(int index, Mahasiswa09 input) {

if (index < 0) {

System.out.println("Indeks salah");

} else if (index == 0) {

addFirst(input);

} else {

NodeMahasiswa09 temp = head;

for (int i = 0; i < index - 1; i++) {

temp = temp.next;

}

temp.next = new NodeMahasiswa09(input, temp.next);

if (temp.next.next == null) {

tail = temp.next;

}

}

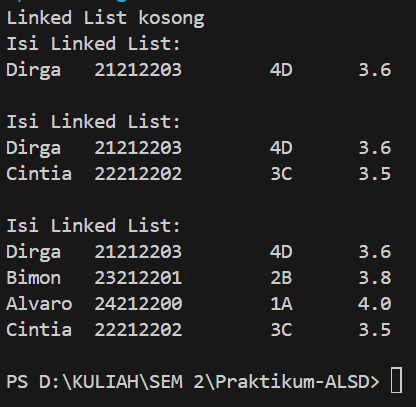
}

}

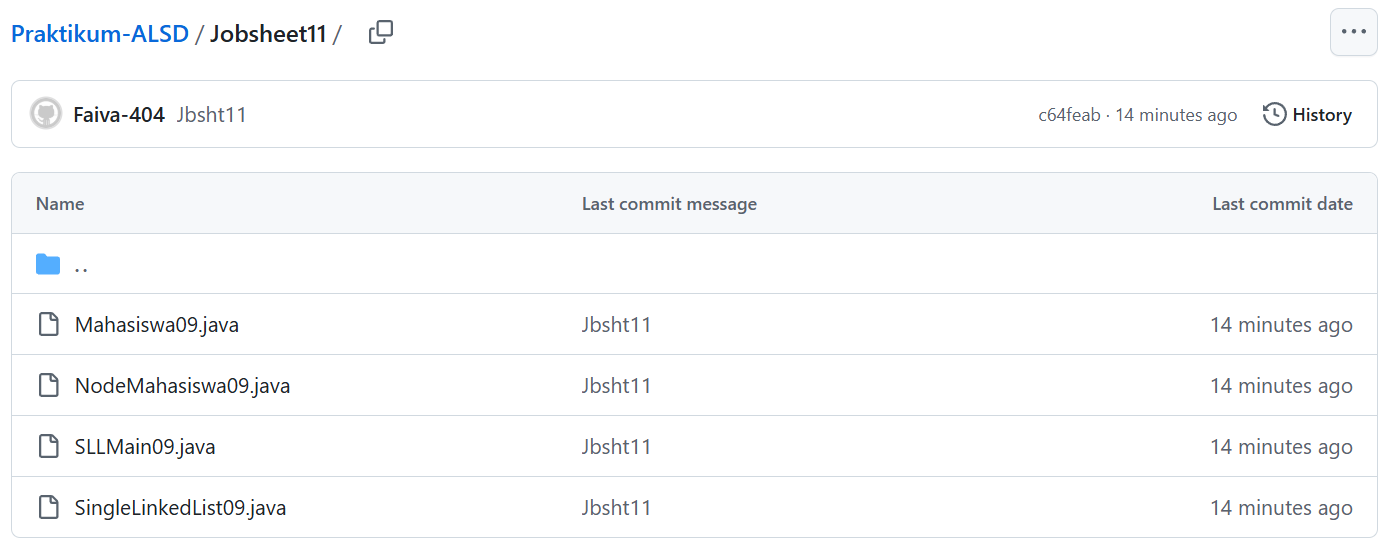
1. Class **SLLMain09.java**



1. Run Program



1. Commit dan push ke Github



* **Pertanyaan**

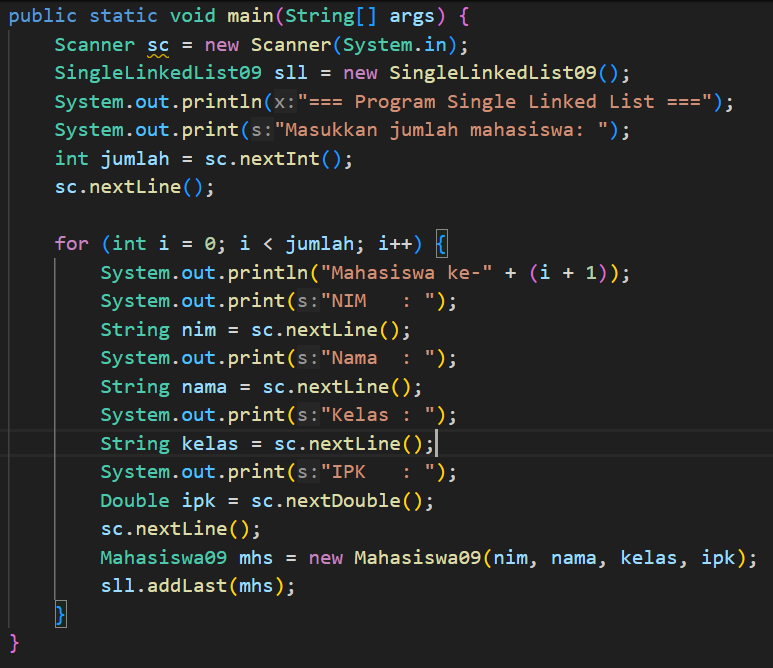
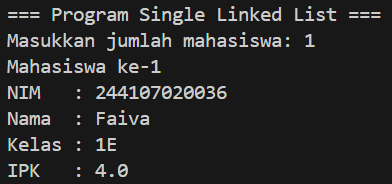
1. Mengapa hasil compile kode program di baris pertama menghasilkan “Linked List Kosong”?

* Saat pertama kali metode print() dipanggil dalam SLLMain09, tidak ada data yang telah ditambahkan ke dalam linked list. Head masih null, sehingga kondisi if (!isEmpty()) menjadi false, dan pesan **"Linked List Kosong"** ditampilkan.

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

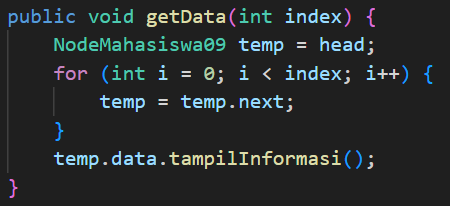
* Berfungsi sebagai pointer sementara yang digunakan untuk menjelajahi atau memodifikasi daftar terhubung **(linked list).**
* **print(),** temp untuk menelusuri seluruh elemen dari head hingga null.
* **insertAfter(),** temp untuk membantu mencari node dengan nama sesuai key, agar dapat menyisipkan node baru setelahnya.
* **insertAt()**, temp untuk mencapai posisi tertentu sebelum melakukan penyisipan elemen baru.

1. Lakukan modifikasi agar data dapat ditambahkan dari keyboard!

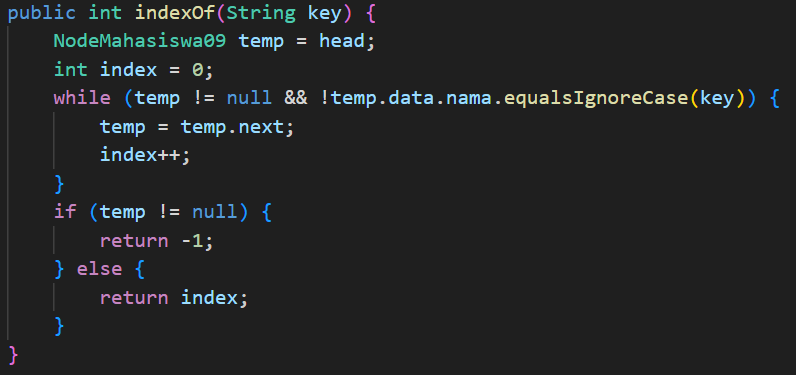
* 
* 

* 1. **Percobaan 2 (Modifikasi Elemen pada Single Linked Lszzz)**

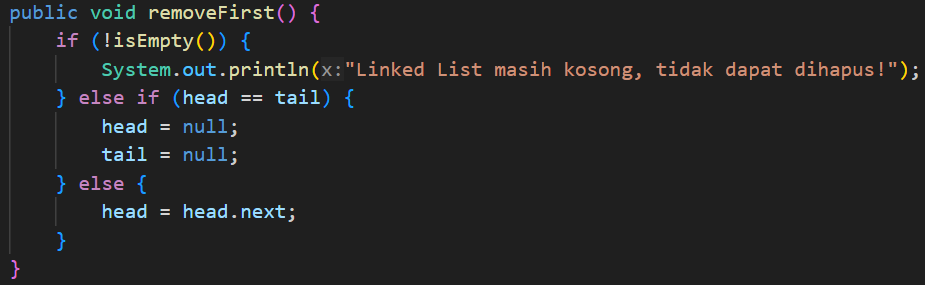
1. Tambahkan method untuk mendapatkan data pada indeks tertentu pada class **SingleLinkedList09.java**



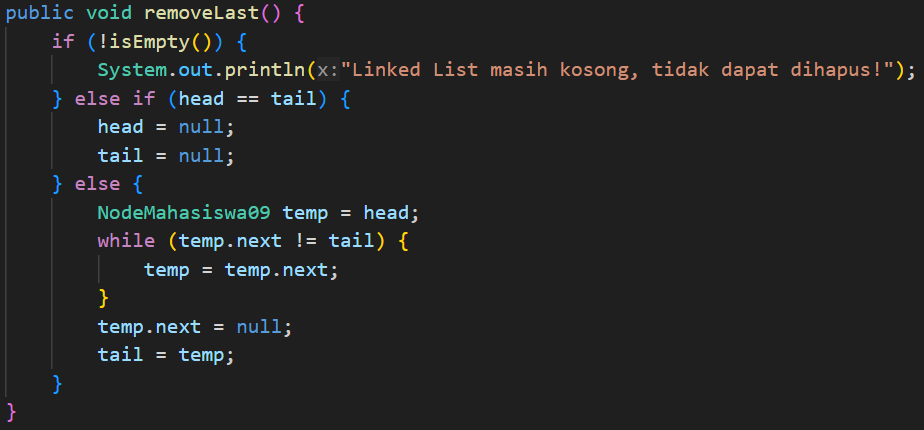
1. Implementasikan method **indexOf**



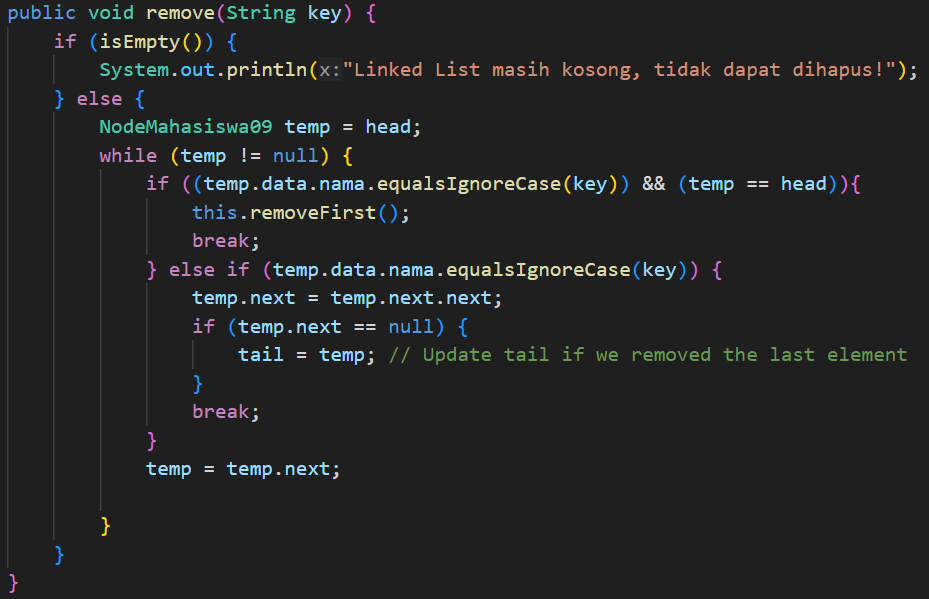
1. Tambahkan method **removeFirst** pada class **SingleLinkedList09.java**



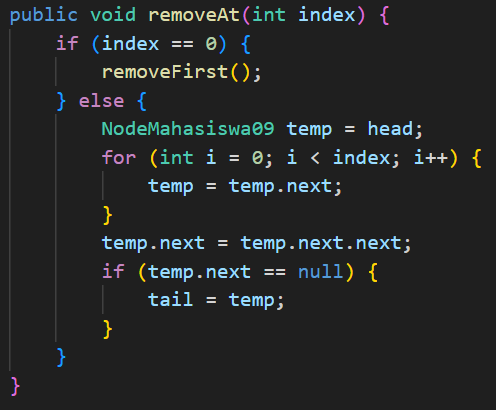
1. Tambahkan method untuk menghapus data pada bagian belakang **removeLast** pada class **SingleLinkedList09.java**



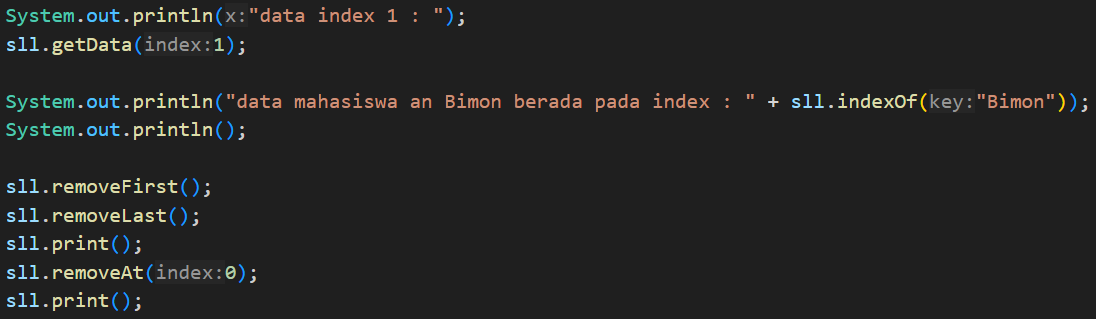
1. Sebagai langkah berikutnya, akan diimplementasikan method **remove**



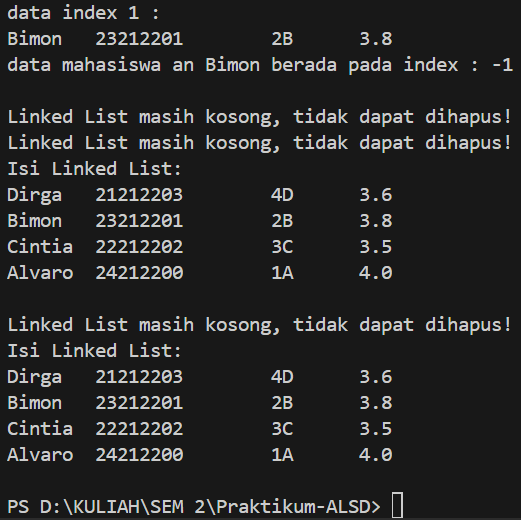
1. Implementasikan method untuk menghapus node dengan menggunakan index



1. Kemudian, coba lakukan pengaksesan dan penghapusan data di method main pada class **SLLMain09.java**



1. Run Program



* **Pertanyaan**

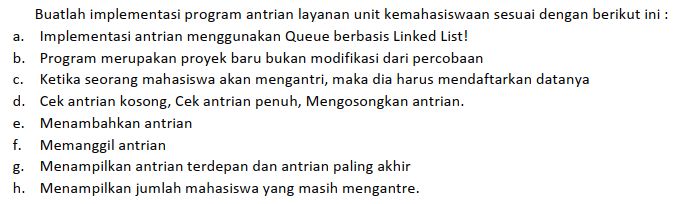
1. Mengapa digunakan keyword break pada fungsi remove? Jelaskan!

* Untuk menghentikan perulangan setelah elemen yang dicari berhasil ditemukan dan dihapus.

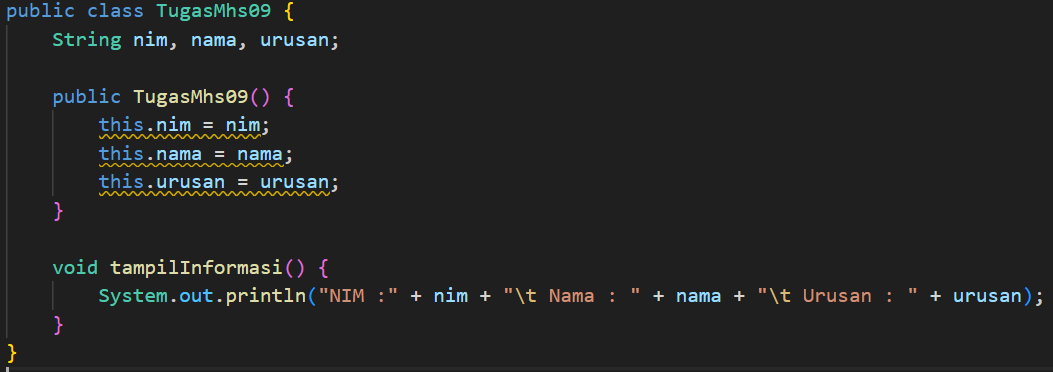
1. Jelaskan kegunaan kode dibawah pada method remove



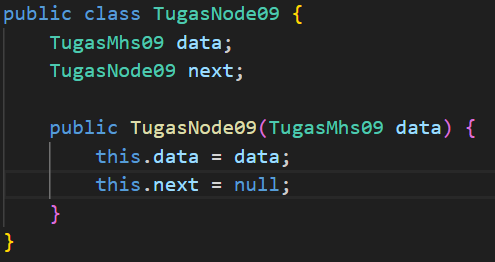
* **temp.next = temp.next.next;**  untuk menghapus node setelah node temp. Dan menyambungkan node temp ke node setelah **temp.next**
* **if (temp.next = = null)** untuk mengecek apakah node setelah temp sudah tidak ada (node yg dihapus adalah node terakhir).
* **tail = temp;** menunjukkan jika benar, maka node temp saat ini adalah node terakhir.
  1. **Tugas**



* **Class TugasMhs09.java**



* **Class TugasNode.java**



* **Class**
* Commit dan push ke Github