JOBSHEET IX LINKED LIST

Satria Wiguna/Ti 1D/Absen 26

# 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

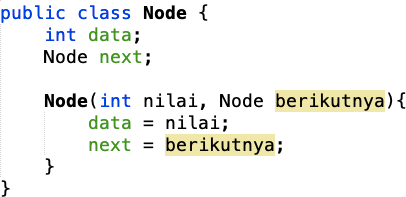
# Praktikum

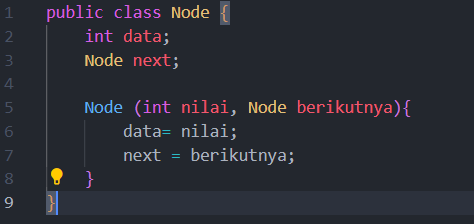
* 1. **Pembuatan Single Linked List**

## Waktu percobaan : 30 menit

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

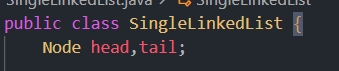
1. Pada Project **StrukturData** yang sudah dibuat pada Minggu sebelumnya, buat package dengan nama **minggu11**
2. Tambahkan class-class berikut:
   1. Node.java
   2. SingleLinkedList.java
   3. SLLMain.java
3. Implementasi class Node





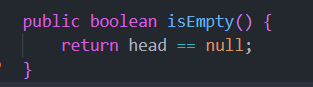
1. Tambahkan atribut pada class SingleLinkedList





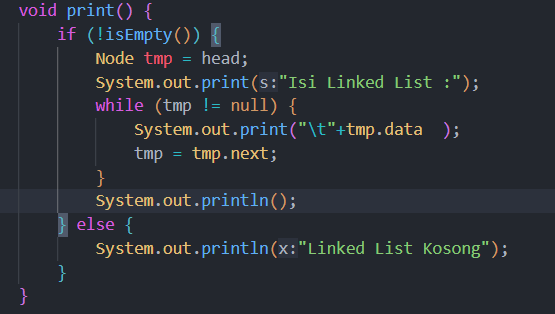
1. Sebagai langkah berikutnya, akan diimplementasikan method-method yang terdapat pada SingleLinkedList.
2. Tambahkan method **isEmpty()**.



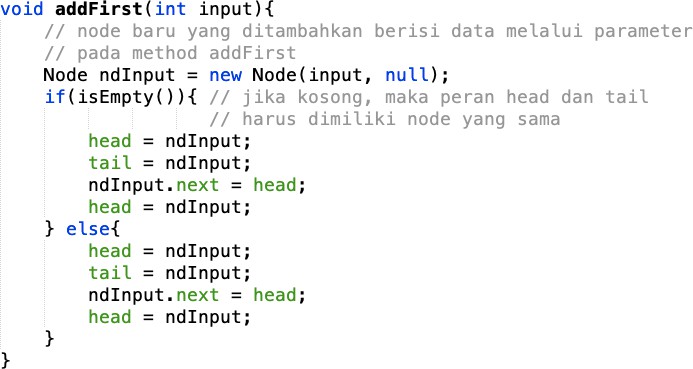
'

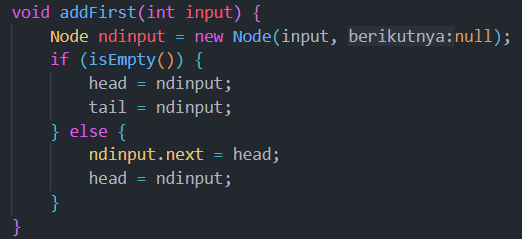
1. Implementasi method untuk mencetak dengan menggunakan proses traverse.



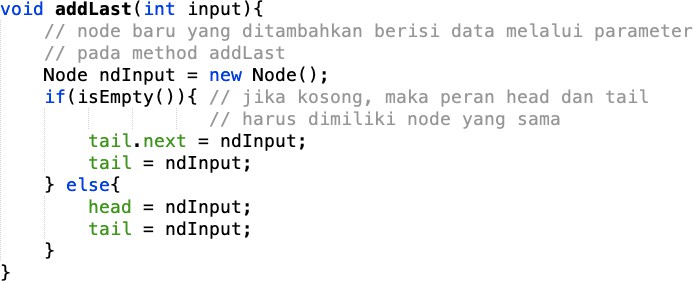


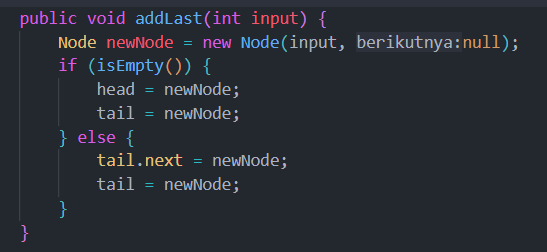
1. Implementasikan method **addFirst()**.





1. Implementasikan method **addLast()**.

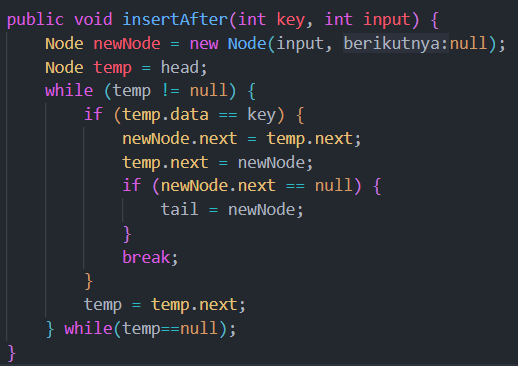




1. Implementasikan method **insertAfter**, untuk memasukkan node yang memiliki data

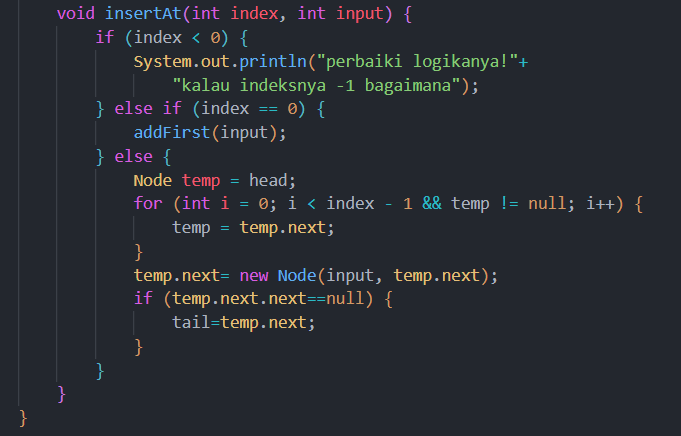
input setelah node yang memiliki data key.





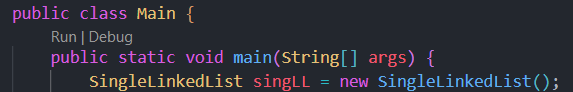
1. Tambahkan method penambahan node pada indeks tertentu.



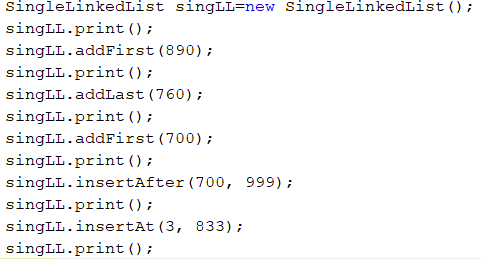


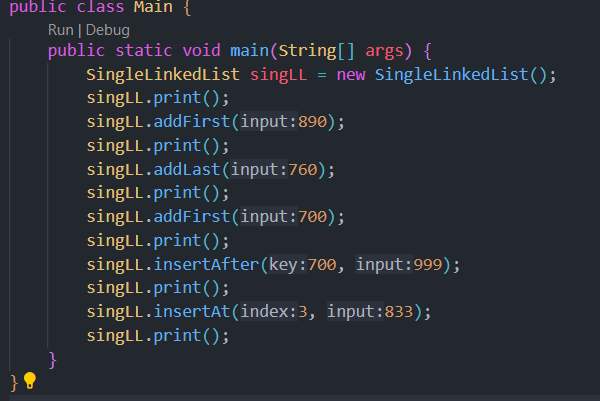
1. Pada class SLLMain, buatlah fungsi **main**, kemudian buat object dari class SingleLinkedList.





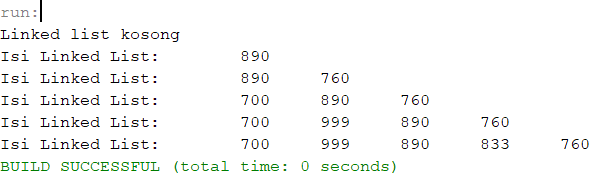
1. Tambahkan Method penambahan data dan pencetakan data di setiap penambahannya agar terlihat perubahannya.

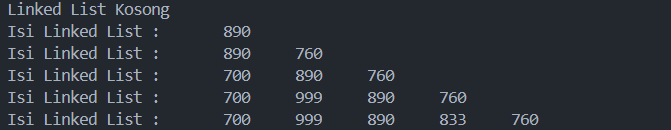




## Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.





## Pertanyaan

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

Pesan “Linked List Kosong” muncul saat method print() dipanggil karena isEmpty() mengembalikan true, menandakan bahwa head dari linked list adalah null. Ini menunjukkan bahwa linked list belum berisi node apa pun, sehingga disimpulkan bahwa linked list kosong

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

temp adalah penunjuk yang digunakan untuk melacak posisi saat melakukan operasi pada linked list, seperti mencari, menambah, atau menyisipkan node pada posisi tertentu dalam struktur data linked list

1. Perhatikan class **SingleLinkedList**, pada method **insertAt** Jelaskan kegunaan kode berikut



Kode diatas berguna memeriksa apakah node baru (temp.next) adalah node terakhir dalam linked list. Jika ya, maka tail diperbarui untuk menunjuk ke node baru ini, karena node baru telah disisipkan di akhir linked list.

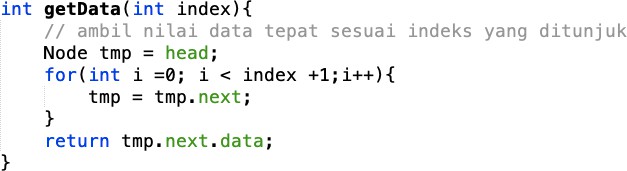
# Modifikasi Elemen pada Single Linked List

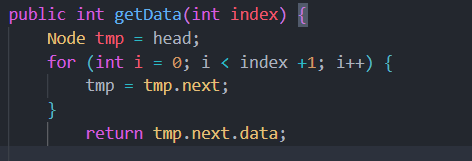
## Waktu percobaan : 30 menit

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

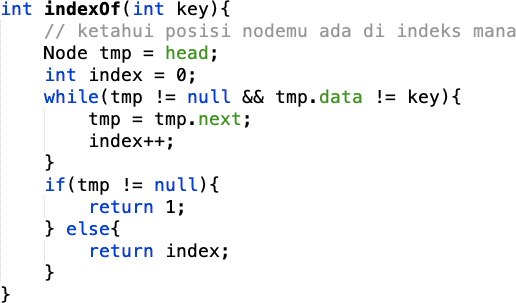
## Langkah-langkah Percobaan

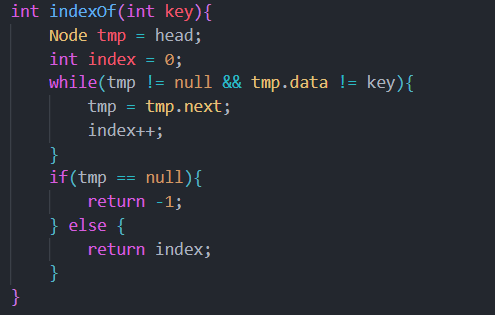
* + - 1. Implementasikan method untuk mengakses data dan indeks pada linked list
      2. Tambahkan method untuk mendapatkan data pada indeks tertentu pada class Single Linked List



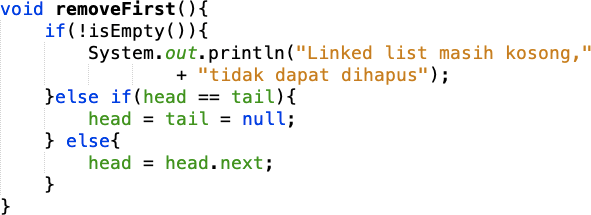


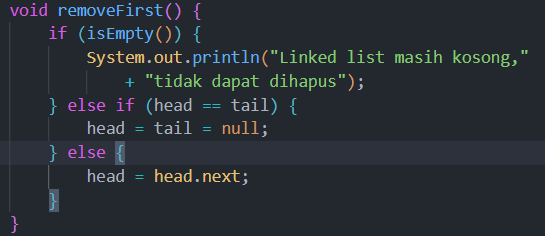
* + - 1. Implementasikan method **indexOf**.



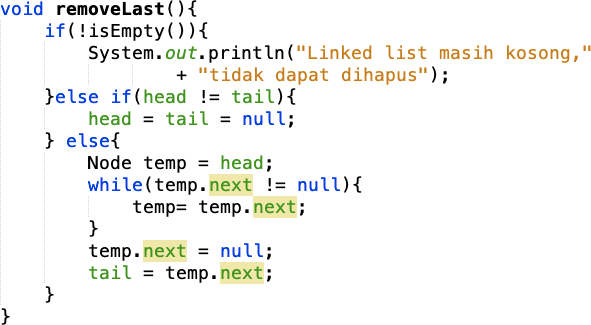


* + - 1. Tambahkan method removeFirst pada class SingleLinkedList



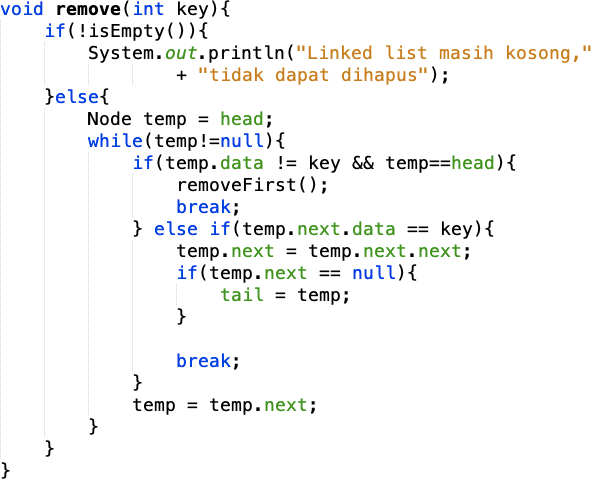


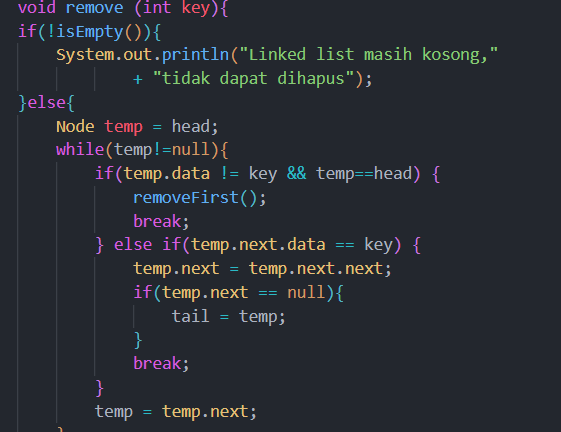
* + - 1. Tambahkan method untuk menghapus data pada bagian belakang pada class SingleLinkedList



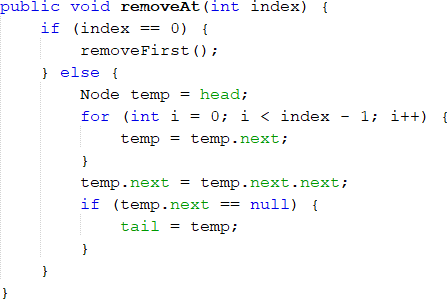


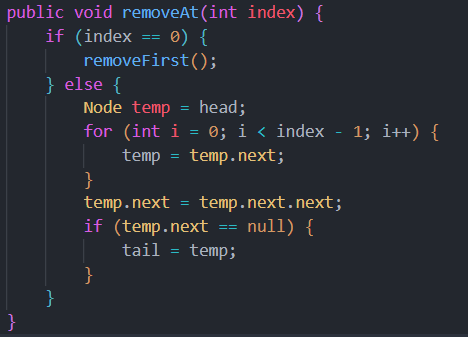
* + - 1. Sebagai langkah berikutnya, akan diimplementasikan method remove



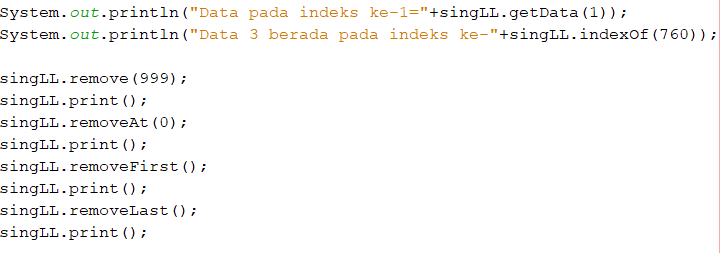


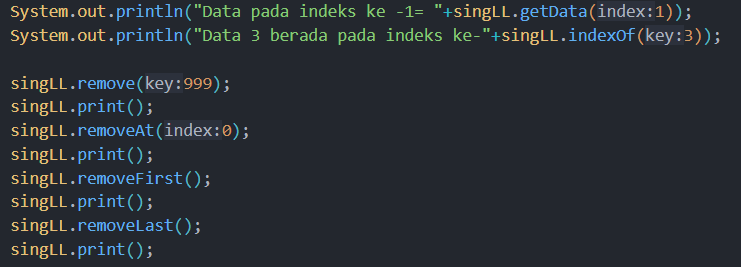
* + - 1. Implementasi method untuk menghapus node dengan menggunakan index.

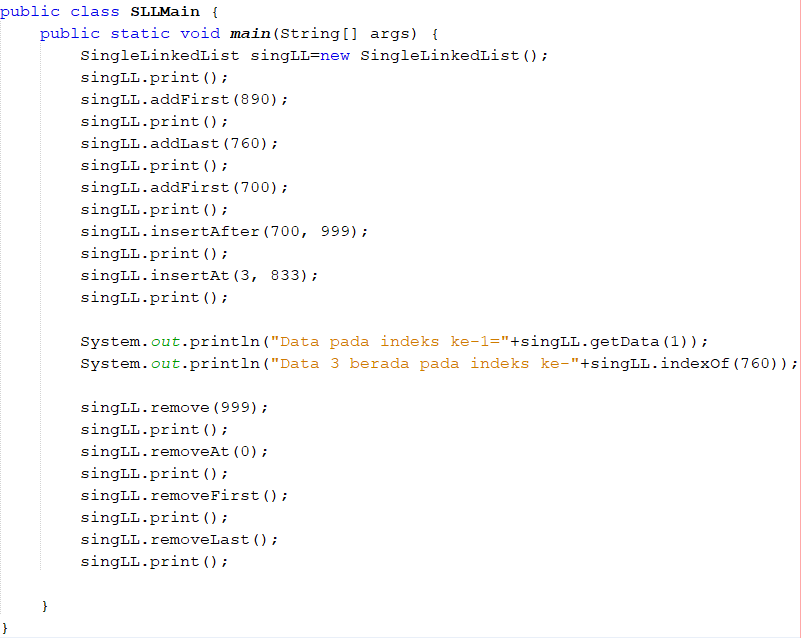




* + - 1. Kemudian, coba lakukan pengaksesan dan penghapusan data di method main pada class SLLMain dengan menambahkan kode berikut

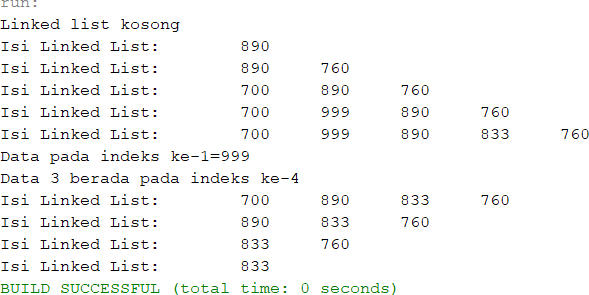


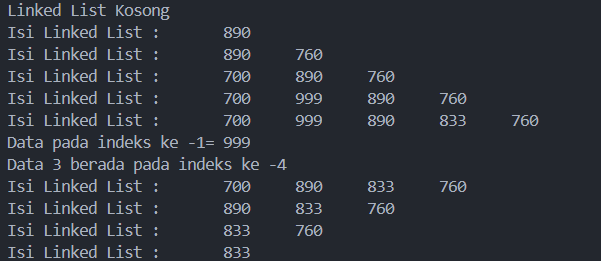


* + - 1. Method SLLMain menjadi:
      2. Jalankan class SLLMain

## Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.





# Pertanyaan

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

digunakan untuk menghentikan loop setelah menemukan dan menghapus node yang sesuai. Ini menghindari pemeriksaan berlebihan terhadap sisa elemen dalam linked list setelah operasi penghapusan selesai

1. Jelaskan kegunaan kode dibawah pada method remove

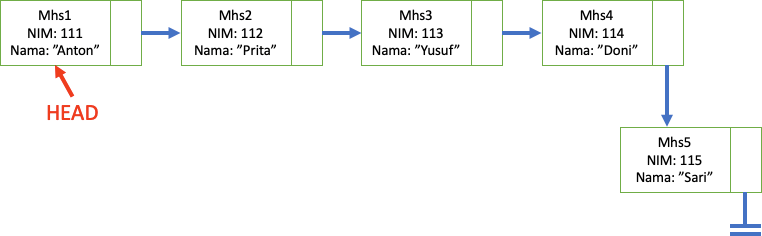


Metode ini mengambil bilangan integer sebagai input dan menghapus node pada indeks yang sesuai dari daftar.

# Tugas

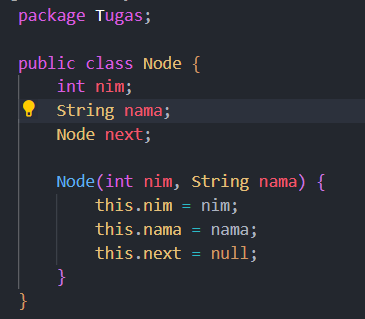
**Waktu pengerjaan : 50 menit**

1. Implementasikan ilustrasi Linked List Berikut. Gunakan 4 macam penambahan data yang telah dipelajari sebelumnya untuk menginputkan data.



1. Buatlah implementasi program antrian layanan unit kemahasiswaan sesuai dengan kondisi yang ditunjukkan pada soal nomor 1! Ketentuan
   1. Implementasi antrian menggunakan Queue berbasis Linked List!
   2. Program merupakan proyek baru, bukan modifikasi dari soal nomor 1!

Node:



SingleLinkedList:

package Tugas;

public class SingleLinkedList {

    Node head, tail;

    boolean isEmpty() {

        return head == null;

    }

    void print() {

        if (!isEmpty()) {

            Node temp = head;

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

            while (temp != null) {

                System.out.print("\tNIM: " + temp.nim + ", Nama: " + temp.nama);

                temp = temp.next;

            }

            System.out.println();

        } else {

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

        }

    }

    void addFirst(int nim, String nama) {

        Node newNode = new Node(nim, nama);

        if (isEmpty()) {

            head = newNode;

            tail = newNode;

        } else {

            newNode.next = head;

            head = newNode;

        }

    }

    void addLast(int nim, String nama) {

        Node newNode = new Node(nim, nama);

        if (isEmpty()) {

            head = newNode;

            tail = newNode;

        } else {

            tail.next = newNode;

            tail = newNode;

        }

    }

    void insertAfter(int keyNim, int nim, String nama) {

        Node newNode = new Node(nim, nama);

        Node temp = head;

        while (temp != null) {

            if (temp.nim == keyNim) {

                newNode.next = temp.next;

                temp.next = newNode;

                if (newNode.next == null) {

                    tail = newNode;

                }

                break;

            }

            temp = temp.next;

        }

    }

    void insertAt(int index, int nim, String nama) {

        if (index < 0) {

            System.out.println("Indeks tidak boleh negatif");

        } else if (index == 0) {

            addFirst(nim, nama);

        } else {

            Node temp = head;

            for (int i = 0; i < index - 1 && temp != null; i++) {

                temp = temp.next;

            }

            if (temp != null) {

                temp.next = new Node(nim, nama);

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

                    tail = temp.next;

                }

            } else {

                System.out.println("Indeks melebihi jumlah elemen dalam list");

            }

        }

    }

    int getData(int index) {

        if (index < 0) {

            System.out.println("Indeks tidak boleh negatif");

            return -1;

        }

        Node temp = head;

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

            if (temp == null) {

                System.out.println("Indeks melebihi jumlah elemen dalam list");

                return -1;

            }

            temp = temp.next;

        }

        return temp != null ? temp.nim : -1;

    }

    int indexOf(int keyNim) {

        Node temp = head;

        int index = 0;

        while (temp != null && temp.nim != keyNim) {

            temp = temp.next;

            index++;

        }

        return temp == null ? -1 : index;

    }

    void removeFirst() {

        if (isEmpty()) {

            System.out.println("Linked list masih kosong, tidak dapat dihapus");

        } else if (head == tail) {

            head = tail = null;

        } else {

            head = head.next;

        }

    }

    void removeLast() {

        if (isEmpty()) {

            System.out.println("Linked list masih kosong, tidak dapat dihapus");

        } else if (head == tail) {

            head = tail = null;

        } else {

            Node temp = head;

            while (temp.next != tail) {

                temp = temp.next;

            }

            temp.next = null;

            tail = temp;

        }

    }

    void remove(int keyNim) {

        if (isEmpty()) {

            System.out.println("Linked list masih kosong, tidak dapat dihapus");

        } else if (head.nim == keyNim) {

            removeFirst();

        } else {

            Node temp = head;

            while (temp != null) {

                if (temp.next != null && temp.next.nim == keyNim) {

                    temp.next = temp.next.next;

                    if (temp.next == null) {

                        tail = temp;

                    }

                    break;

                }

                temp = temp.next;

            }

        }

    }

    void removeAt(int index) {

        if (index < 0) {

            System.out.println("Indeks tidak boleh negatif");

        } else if (index == 0) {

            removeFirst();

        } else {

            Node temp = head;

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

                if (temp == null) {

                    System.out.println("Indeks melebihi jumlah elemen dalam list");

                    return;

                }

                temp = temp.next;

            }

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

                temp.next = temp.next.next;

                if (temp.next == null) {

                    tail = temp;

                }

            } else {

                System.out.println("Indeks melebihi jumlah elemen dalam list");

            }

        }

    }

}

Main :



Hasil:  
