JOBSHEET IX LINKED LIST

Nama: Rizqi Bagus Andrean

Absen: 25

Kelas: Ti-1D

# 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

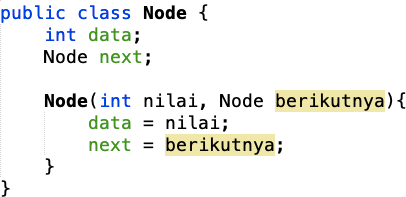
# Praktikum

* 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.

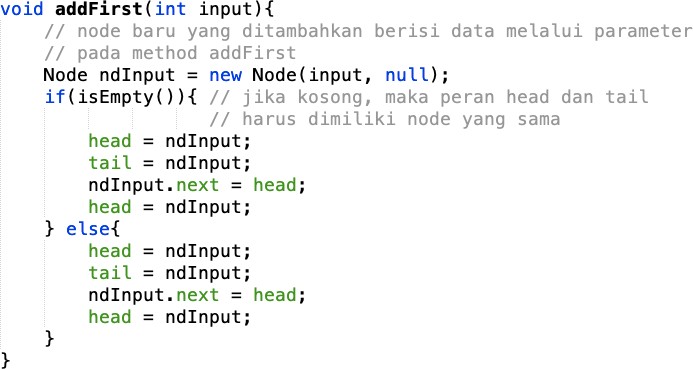
* + 1. Tambahkan method **isEmpty()**.



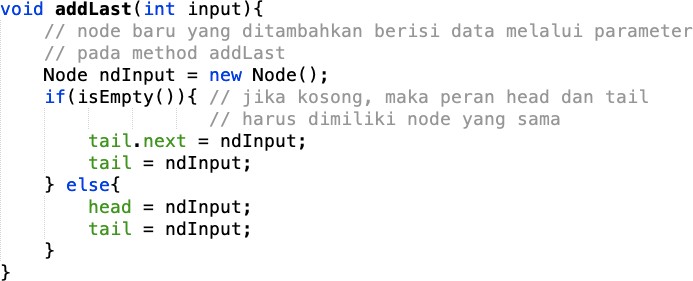
* + 1. Implementasi method untuk mencetak dengan menggunakan proses traverse.



* + 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.



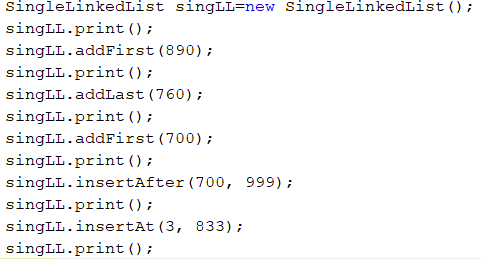
* + 1. Tambahkan method penambahan node pada indeks tertentu.



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

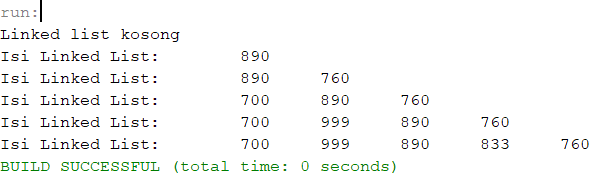


* + 1. Tambahkan Method penambahan data dan pencetakan data di setiap penambahannya agar terlihat perubahannya.



## 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”?

Karena belum ada yang ditambahkan atau headnya == null

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

Secara umum, variabel bernama temp berfungsi sebagai referensi sementara di setiap method dari class SingleLinkedList. Variabel ini digunakan untuk menelusuri linked list tanpa mengubah pointer head atau tail secara langsung. Ini memungkinkan method untuk iterasi melalui list, menemukan node tertentu, atau memasukkan node baru pada posisi yang diinginkan.

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

Potongan kode ini berfungsi untuk memperbarui pointer tail agar mengarah ke node setelah temp **tetapi hanya jika** node tersebut memang merupakan node terakhir di linked list. Situasi ini biasanya terjadi ketika kita sedang menambahkan node baru di akhir linked list.

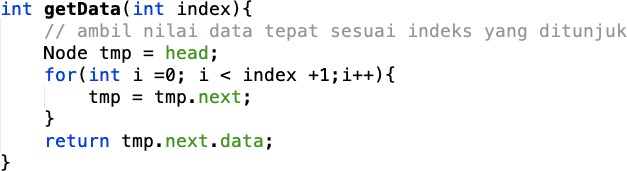
# Modifikasi Elemen pada Single Linked List

## Waktu percobaan : 30 menit

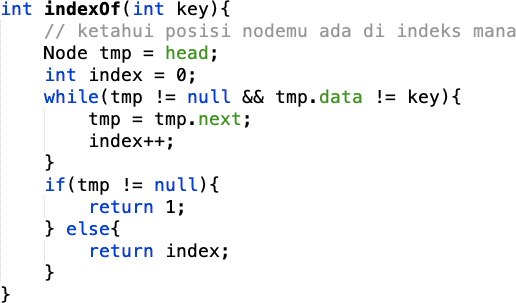
Didalam praktikum ini, kita akan mempraktekkan bagaimana mengakses elemen, mendapatkan indeks dan melakukan penghapusan data pada Single Linked List.:

## Langkah-langkah Percobaan

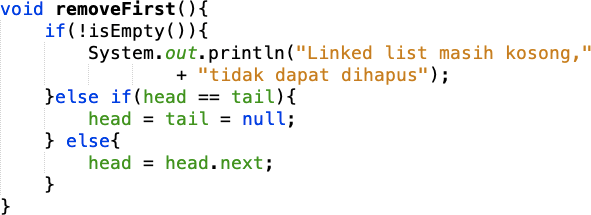
* + - 1. Implementasikan method untuk mengakses data dan indeks pada linked list
      2. Tambahkan method untuk mendapatkan data pada indeks tertentu pada class Single Linked List



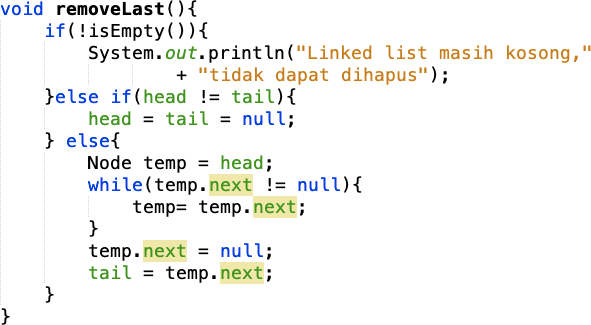
* + - 1. Implementasikan method **indexOf**.



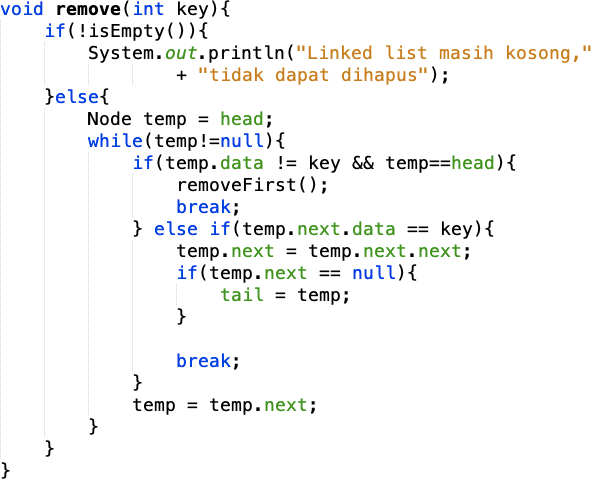
* + - 1. Tambahkan method removeFirst pada class SingleLinkedList



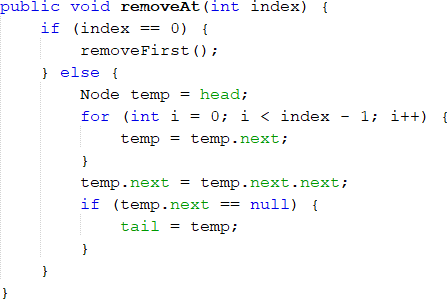
* + - 1. Tambahkan method untuk menghapus data pada bagian belakang pada class SingleLinkedList



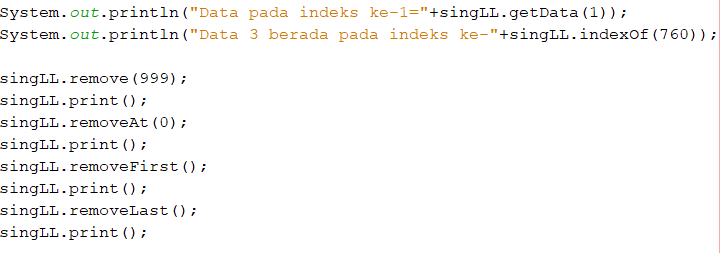
* + - 1. Sebagai langkah berikutnya, akan diimplementasikan method remove

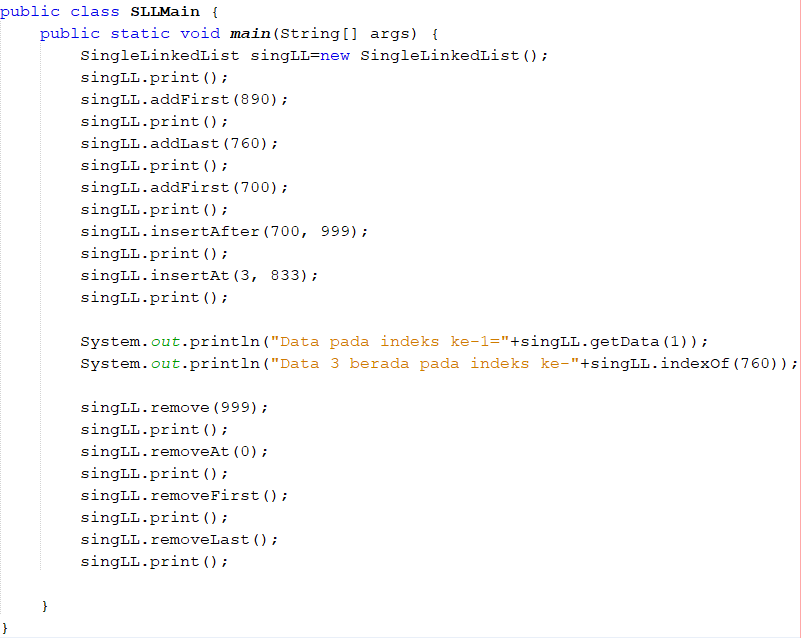


* + - 1. Implementasi method untuk menghapus node dengan menggunakan index.



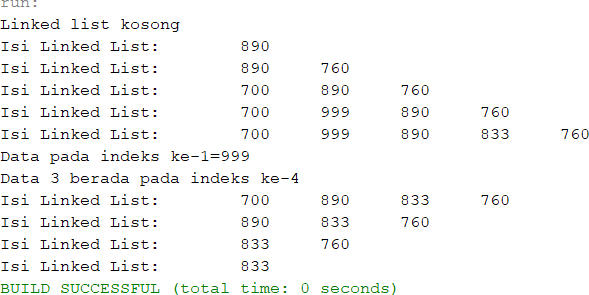
* + - 1. Kemudian, coba lakukan pengaksesan dan penghapusan data di method main pada class SLLMain dengan menambahkan kode berikut



* + - 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!

Break digunakan untuk menghentikan while saat satu kondisi terpenuhi;

1. Jelaskan kegunaan kode dibawah pada method remove

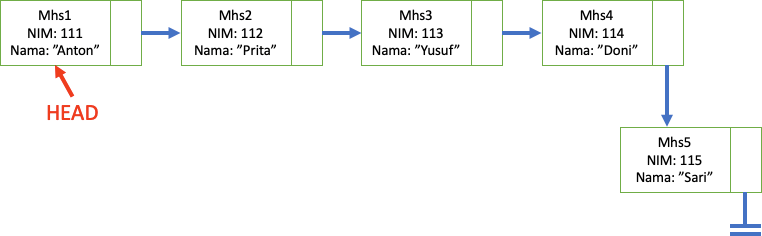


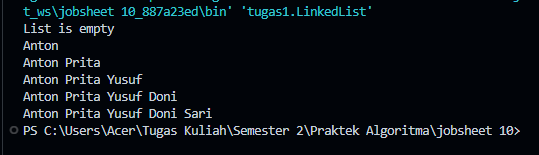
potongan kode ini melakukan bypass pada node yang ingin dihapus dengan memperbarui referensi pada node sebelumnya agar terhubung langsung ke node setelahnya. Ini secara efektif menghapus node yang ditargetkan dari linked list.

# Tugas

**Waktu pengerjaan : 50 menit**

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





package tugas1;

public class LinkedList {

    Node head, tail;

    boolean isEmpty() {

        return head == null;

    }

    void print() {

        if (isEmpty()) {

            System.out.println("List is empty");

            return;

        }

        Node temp = head;

        while (temp != null) {

            System.out.print(temp.data.nama + " ");

            temp = temp.next;

        }

        System.out.println();

    }

    void addFirst(Mahasiswa data) {

        Node newNode = new Node(data, null);

        if (isEmpty()) {

            head = tail = newNode;

        } else {

            newNode.next = head;

            head = newNode;

        }

    }

    void addLast(Mahasiswa data) {

        Node newNode = new Node(data, null);

        if (isEmpty()) {

            head = tail = newNode;

        } else {

            tail.next = newNode;

            tail = newNode;

        }

    }

    void insertAfter(Mahasiswa key, Mahasiswa input){

        Node temp = head;

        while (temp != null && temp.data != key) {

            temp = temp.next;

        }

        if (temp == null) {

            System.out.println("Key not found");

        } else {

            Node newNode = new Node(input, temp.next);

            newNode.next = temp.next;

            temp.next = newNode;

            if (temp == tail) {

                tail = newNode;

            }

        }

    }

    void insertAt(int index, Mahasiswa input) {

        Node ndInput = new Node();

        if (index > 0) {

            Node temp = head;

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

                if (temp == null) {

                    System.out.println("Index out of bound");

                    return;

                }

                temp = temp.next;

            }

            ndInput.data = input;

            ndInput.next = temp.next;

            temp.next = ndInput;

            if (temp == tail) {

                tail = ndInput;

            }

        } else if (index == 0) {

            ndInput.data = input;

            ndInput.next = head;

            head = ndInput;

            if (tail == null) {

                tail = ndInput;

            }

        } else {

            Node temp = head;

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

                temp = temp.next;

            }

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

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

                tail = temp.next;

            }

            System.out.println("Invalid index");

        }

    }

    public static void main(String[] args) {

        LinkedList list = new LinkedList();

        list.print();

        list.addFirst(new Mahasiswa("Anton", 111));

        list.print();

        list.addLast(new Mahasiswa("Prita", 124));

        list.print();

        list.addLast(new Mahasiswa("Yusuf", 124));

        list.print();

        list.addLast(new Mahasiswa("Doni", 124));

        list.print();

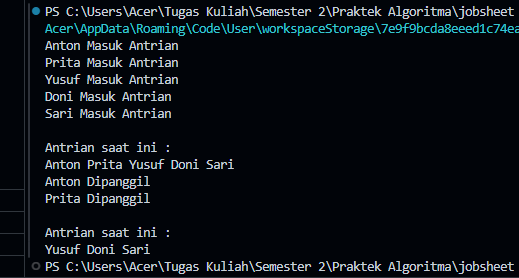
        list.addLast(new Mahasiswa("Sari", 124));

        list.print();

    }

}

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!



package tugas1;

public class LinkedList {

    Node head, tail;

    boolean isEmpty() {

        return head == null;

    }

    void print() {

        if (isEmpty()) {

            System.out.println("List is empty");

            return;

        }

        Node temp = head;

        while (temp != null) {

            System.out.print(temp.data.nama + " ");

            temp = temp.next;

        }

        System.out.println();

    }

    void addLast(Mahasiswa data) {

        Node newNode = new Node(data, null);

        if (isEmpty()) {

            head = tail = newNode;

        } else {

            tail.next = newNode;

            tail = newNode;

        }

        System.out.println(data.nama + " Masuk Antrian");

    }

    void removeFirst() {

        if (isEmpty()) {

            System.out.println("List is empty");

            return;

        }

        System.out.println(head.data.nama + " Dipanggil");

        head = head.next;

        if (head == null) {

            tail = null;

        }

    }

    public static void main(String[] args) {

        LinkedList list = new LinkedList();

        // Menambah Antrian

        list.addLast(new Mahasiswa("Anton", 111));

        list.addLast(new Mahasiswa("Prita", 124));

        list.addLast(new Mahasiswa("Yusuf", 124));

        list.addLast(new Mahasiswa("Doni", 124));

        list.addLast(new Mahasiswa("Sari", 124));

        System.out.println();

        System.out.println("Antrian saat ini : ");

        list.print();

        // Memanggil Antrian

        list.removeFirst();

        list.removeFirst();

        System.out.println();

        System.out.println("Antrian saat ini : ");

        list.print();

    }

}