

# MODUL I

## LINKED LIST

### 1.1 PERMASALAHAN

#### 1.1.1 MANUSIA VIRAL

Gunakan *singe linked list*. Oke singkat saja buat *list* manusia viral, harus ada *method addFIRST, addLAST, deleteFIRST, deleteLAST* dan *method* untuk menampilkan manusia-manusia itu. buat *list* manusia viral dengan parameter, nama manusia, skill, umur, dan hobi.

*List* manusia viral

Nama : Kak Gem	Nama : Vadel	Nama : Loli	Nama : Agus	Nama : Fufu fafa
Skill : Kasi Faham	Skill : Dance Gaerr	Skill : ATM Berjalan	Skill : Agus Sakit	Skill : Roasting
Umur : 32	Umur : 19	Umur : 16	Umur : 35	Umur : 35
Hobi : Bernafas	Hobi : Dance	Hobi : Liat Vadel	Hobi : Donasi	Hobi : Buka Kaskus

Pertama-tama tambahkan Vadel dalam *list*, diikuti dengan Loli dan Agus serta Fufu Fafa. kemudian tambah kak Gem menggunakan *method addFirst*. setelah itu hapus Fufu Fafa dari *list* dengan *method deleteLast*. terakhir, hapus kak Gem dengan *method deleteFirst* dan tampilkan *list* manusia-manusia yang tersisa.

---

## 1.2 HASIL PERCOBAAN

### 1.2.1 Program Manusia Viral

#### 1. Algoritma

- a. Kelas *Node*
  - i. *Start*.
  - ii. Mendeklarasi atribut nama, skill, umur, hobi dan *Node next*.
  - iii. Mengkonstruksi atribut nama, skill, umur, hobi berdasarkan *input*-an dari *user* dan *Node next* diinisialisasi dengan *null*.
  - iv. *Stop*.
- b. Kelas *Linked List*
  - i. *Start*.
  - ii. Mendeklarasi atribut *head* sebagai objek *Node*.
  - iii. Mengkonstruksi *head* dengan nilai *null*.
  - iv. Menambahkan data pada urutan pertama dengan deklarasi dan inisialisasi objek *Node* baru.
    1. Jika *head* sama dengan *null* maka masukkan *head* dengan *Node* baru.
    2. Jika tidak, *head* menjadi *next* dari *Node* baru, lalu *Node* baru menjadi *head*.
  - v. Menambahkan data pada urutan terakhir dengan deklarasi dan inisialisasi objek *Node* baru.
    1. Jika *head* sama dengan *null*, maka masukkan *head* dengan *Node* baru.
    2. Jika tidak, lakukan langkah berikut:
      - a. Deklarasi variabel sementara *temp* yang diinisialisasi dengan *head*.
      - b. Lakukan perulangan hingga *temp.next* sama dengan *null* untuk menemukan *Node* terakhir.
      - c. Setelah *Node* terakhir ditemukan, atur *temp.next* dengan *Node* baru, sehingga *Node* baru ditempatkan di akhir.
  - vi. Menghapus data pada urutan pertama.
    1. Jika *head* sama dengan *null*, cetak pesan "DATA KOSONG".
    2. Jika tidak, atur *head* menjadi *head.next*, sehingga *Node* pertama dihapus dari daftar.
  - vii. Menghapus data pada urutan terakhir.
    1. Jika *head* sama dengan *null*, cetak pesan "DATA KOSONG".
    2. Jika tidak, lakukan langkah berikut:
      - a. Deklarasi variabel sementara *temp* yang diinisialisasi dengan *head*.

- b. Lakukan perulangan hingga *temp.next.next* sama dengan *null* untuk menemukan *Node* sebelum *Node* terakhir.
- c. Setelah *Node* sebelum *Node* terakhir ditemukan, atur *temp.next* menjadi *null*, sehingga *Node* terakhir dihapus.

viii. Menampilkan semua data dalam *linked list*.

1. Deklarasi variabel sementara *temp* yang diinisialisasi dengan *head*.
2. Selama *temp* tidak sama dengan *null*, mencetak data pada *linked list*.
3. Ulangi proses sampai semua *Node* ditampilkan.

c. Kelas *Main*

- i. Buat objek *LinkedList* bernama *list*.
- ii. Tambahkan *Node* baru di urutan pertama dengan data milik "Vadel".
- iii. Tambahkan *Node* baru di urutan terakhir dengan data milik "Loli".
- iv. Tambahkan *Node* baru di urutan terakhir dengan data milik "Agus".
- v. Tambahkan *Node* baru di urutan terakhir dengan data milik "Fufu Fafa".
- vi. Tambahkan *Node* baru di urutan pertama dengan data milik "Kak Gem".
- vii. Hapus *Node* terakhir dari *linked list*.
- viii. Hapus *Node* pertama dari *linked list*.
- ix. Tampilkan seluruh data yang tersisa dalam *linked list*.

## 2. Source Code

```
public class NodeEasy {
    String nama;
    String skill;
    int umur;
    String hobi;
    NodeEasy next;

    public NodeEasy (String nama, String skill, int umur, String hobi)
    {
        this.nama = nama;
        this.skill = skill;
        this.umur = umur;
        this.hobi = hobi;
        this.next = null;
    }
}

public class LinkedListEasy {
    NodeEasy head;
```

```

public LinkedListEasy() {
    this.head = null;
}

public void addFirst (String nama, String skill, int umur, String
hobi) {
    NodeEasy newNode = new NodeEasy(nama, skill, umur, hobi);
    if (head == null) {
        head = newNode;
    } else {
        newNode.next = head;
        head = newNode;
    }
}

public void addLast (String nama, String skill, int umur, String
hobi) {
    NodeEasy newNode = new NodeEasy(nama, skill, umur, hobi);
    if (head == null) {
        head = newNode;
    } else {
        NodeEasy temp = head;
        while (temp.next != null) {
            temp = temp.next;
        }
        temp.next = newNode;
    }
}

public void deleteFirst () {
    if (head == null) {
        System.out.println("DATA KOSONG");
    } else {
        head = head.next;
    }
}

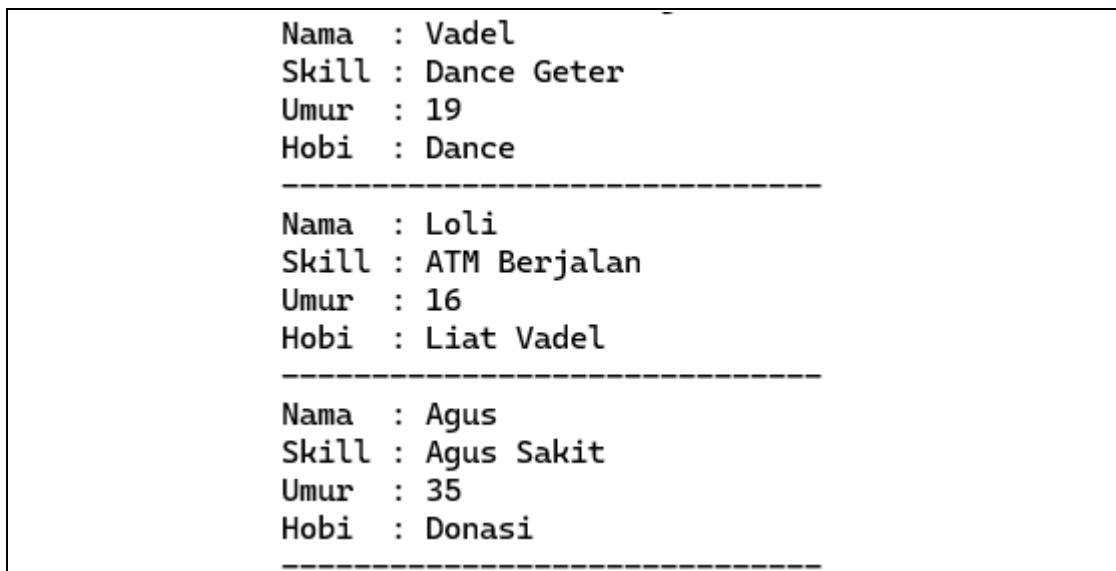
public void deleteLast () {
    if (head == null) {
        System.out.println("DATA KOSONG");
    } else {
        NodeEasy temp = head;
        while (temp.next.next != null) {
            temp = temp.next;
        }
        temp.next = null;
    }
}

public void display () {
    NodeEasy temp = head;
    while (temp != null) {
        System.out.println("Nama   : " + temp.nama);
        System.out.println("Skill  : " + temp.skill);
        System.out.println("Umur   : " + temp.umur);
        System.out.println("Hobi   : " + temp.hobi);
        System.out.println("-----");
        temp = temp.next;
    }
}

```

```
}  
  
public class MainEasy {  
    public static void main(String[] args) {  
        LinkedListEasy list = new LinkedListEasy();  
  
        list.addFirst("Vadel", "Dance Geter", 19, "Dance");  
        list.addLast("Loli", "ATM Berjalan", 16, "Liat Vadel");  
        list.addLast("Agus", "Agus Sakit", 35, "Donasi");  
        list.addLast("Fufu Fafa", "Roasting", 30, "Buka Kaskus");  
        list.addFirst("Kak Gem", "Kasi Faham", 32, "Bernafas");  
  
        list.deleteLast();  
        list.deleteFirst();  
  
        list.display();  
    }  
}
```

### 3. Hasil Program



```
Nama : Vadel  
Skill : Dance Geter  
Umur : 19  
Hobi : Dance  
-----  
Nama : Loli  
Skill : ATM Berjalan  
Umur : 16  
Hobi : Liat Vadel  
-----  
Nama : Agus  
Skill : Agus Sakit  
Umur : 35  
Hobi : Donasi  
-----
```

Gambar 1.1 Hasil Run Program

Berdasarkan **Gambar 1.1**, dapat diketahui bahwa data manusia viral seperti Vadel, Loli, Agus, Fufu Fafa, dan kak Gem berhasil di tambahkan sesuai dengan metode yang digunakan seperti *addFirst* dan *addLast*. Lalu, data kak Gem dan Fufu Fafa berhasil dihapus dari *list* dengan kak Gem terhapus menggunakan metode *deleteFirst* dan Fufu Fafa terhapus menggunakan metode *deleteLast*.

## 1.3 ANALISIS DATA

### 1.3.1 Nama Program

```
public class NodeEasy
```

*Script* “public class NodeEasy” adalah deklarasi kelas untuk digunakan sebagai data yang dapat memiliki alamat selanjutnya. Kata kunci “public” berfungsi untuk kelas “NodeEasy” dapat diakses di kelas utama.

```
String nama;  
String skill;  
int umur;  
String hobi;  
NodeEasy next;
```

*Script* di atas adalah kumpulan deklarasi variabel dan objek yang disebut sebagai atribut seperti nama, skill, umur, hobi dan next. Atribut tersebut nantinya dapat diakses di dalam fungsi yang sama ataupun di luar fungsi dengan menggunakan metode yang ada pada fungsi.

```
public NodeEasy (String nama, String skill, int umur, String hobi) {  
    this.nama = nama;  
    this.skill = skill;  
    this.umur = umur;  
    this.hobi = hobi;  
    this.next = null;  
}
```

*Script* di atas merupakan konstruktor untuk melakukan inisialisasi data pada atribut ketika objek “NodeEasy” dideklarasikan. Hal ini dapat mempermudah dalam menetapkan data awal pada objek “NodeEasy”.

```
public class LinkedListEasy
```

*Script* “public class LinkedListEasy” adalah deklarasi kelas untuk digunakan sebagai *linkedlist* yang dapat memiliki atribut dan metode yang dapat digunakan untuk menambah data, menghapus data, dan menampilkan data.

```
NodeEasy head;
```

*Script* “NodeEasy head” adalah deklarasi objek untuk digunakan sebagai data yang dapat memiliki alamat ke data selanjutnya. Di mana “head” memiliki atribut nama, skill, umur, hobi, dan next.

```
public LinkedListEasy() {  
    this.head = null;  
}
```

*Script* di atas merupakan konstruktor untuk melakukan inisialisasi data pada atribut ketika objek “LinkedListEasy” dideklarasikan. Hal ini dapat mempermudah dalam menetapkan data awal pada objek “LinkedListEasy”.

```
public void addFirst (String nama, String skill, int umur, String hobi)
{
    NodeEasy newNodeEasy = new NodeEasy(nama, skill, umur, hobi);
    if (head == null) {
        head = newNodeEasy;
    } else {
        newNodeEasy.next = head;
        head = newNodeEasy;
    }
}
```

*Script* di atas merupakan metode yang berfungsi untuk menambahkan data baru di awal *linked list*. Metode ini menerima empat parameter nama, skill, umur, dan hobi, yang akan digunakan untuk membuat objek “NodeEasy”.

```
public void addLast (String nama, String skill, int umur, String hobi) {
    NodeEasy newNodeEasy = new NodeEasy(nama, skill, umur, hobi);
    if (head == null) {
        head = newNodeEasy;
    } else {
        NodeEasy temp = head;
        while (temp.next != null) {
            temp = temp.next;
        }
        temp.next = newNodeEasy;
    }
}
```

*Script* di atas merupakan metode yang berfungsi untuk menambahkan data baru di akhir *linked list*. Metode ini menerima empat parameter nama, skill, umur, dan hobi, yang digunakan untuk membuat objek “NodeEasy”.

```
public void deleteFirst () {
    if (head == null) {
        System.out.println("DATA KOSONG");
    } else {
        head = head.next;
    }
}
```

*Script* di atas merupakan metode yang berfungsi untuk menghapus data pertama dari *linked list*. Jika *linked list* kosong atau ketika “head” tidak berisi maka metode akan menampilkan pesan "DATA KOSONG". Namun, jika *linked list* berisi data, metode ini akan mengubah “head” menjadi data berikutnya, sehingga data pertama yang lama dihapus dari *list*.

```
public void deleteLast () {
    if (head == null) {
        System.out.println("DATA KOSONG");
    } else {
        NodeEasy temp = head;
        while (temp.next.next != null) {
            temp = temp.next;
        }
    }
}
```

```

    }
    temp.next = null;
  }
}

```

*Script* di atas merupakan metode yang berfungsi untuk menghapus data terakhir dari *linked list*. Jika *linked list* kosong, maka metode akan menampilkan pesan "DATA KOSONG". Namun, jika terdapat data di dalam *linked list*, metode ini akan melakukan iterasi dari "head" hingga menemukan *Node* yang merupakan data kedua terakhir, lalu menghapus data terakhir dengan mengatur "temp.next" menjadi *null*, sehingga *Node* terakhir dihilangkan dari *list*.

```

public void display () {
    NodeEasy temp = head;
    while (temp != null) {
        System.out.println("Nama : " + temp.nama);
        System.out.println("Skill : " + temp.skill);
        System.out.println("Umur : " + temp.umur);
        System.out.println("Hobi : " + temp.hobi);
        System.out.println("-----");
        temp = temp.next;
    }
}

```

*Script* di atas merupakan metode *display* yang berfungsi untuk menampilkan semua data dalam *linked list*. Metode ini menggunakan variabel "temp" untuk melakukan iterasi mulai dari *Node* pertama hingga *Node* terakhir. Setiap *Node* dalam *linked list* akan dicetak pada layar, termasuk atribut nama, skill, umur, dan hobi dari masing-masing *Node*, diikuti dengan garis pemisah. Iterasi akan berhenti ketika *Node* terakhir telah ditampilkan.

```

public class MainEasy {
    public static void main(String[] args) {

```

*Script* di atas merupakan deklarasi kelas "MainEasy" dan metode utama, yang merupakan titik awal eksekusi program Java. Metode *main* menggunakan array *String[] args* sebagai parameter untuk menerima argumen dari *command line* saat program dijalankan. Semua logika program akan dituliskan dalam blok *main*, dan metode ini akan dieksekusi pertama kali ketika program "MainEasy" dijalankan.

```

LinkedListEasy list = new LinkedListEasy();

```

*Script* di atas merupakan inisialisasi objek *list* dari kelas "LinkedListEasy". Ketika objek *list* dibuat, konstruktor "LinkedListEasy()" akan dipanggil untuk menginisialisasi atribut-atribut yang ada dalam objek tersebut. Objek *list* ini nantinya akan digunakan untuk mengakses dan memanipulasi data di dalam *linked list* sesuai dengan metode-metode yang ada pada kelas "LinkedListEasy".



```
list.addFirst("Vadel", "Dance Geter", 19, "Dance");  
list.addLast("Loli", "ATM Berjalan", 16, "Liat Vadel");  
list.addLast("Agus", "Agus Sakit", 35, "Donasi");  
list.addLast("Fufu Fafa", "Roasting", 30, "Buka Kaskus");  
list.addFirst("Kak Gem", "Kasi Faham", 32, "Bernafas");  
  
list.deleteLast();  
list.deleteFirst();  
  
list.display();
```

*Script* di atas merupakan serangkaian perintah untuk memanipulasi objek *list* dari kelas “LinkedListEasy”. Pertama, beberapa data ditambahkan ke dalam *linked list* menggunakan metode “addFirst” dan “addLast”, di mana “addFirst” digunakan untuk menambahkan data pertama dengan data "Vadel" dan "Kak Gem" di awal, sementara “addLast” digunakan untuk menambahkan data "Loli", "Agus", dan "Fufu Fafa" di akhir *linked list*. Selanjutnya, data terakhir dihapus menggunakan metode “deleteLast”, diikuti dengan penghapusan data pertama menggunakan metode “deleteFirst”. Terakhir, metode display digunakan untuk menampilkan semua data yang tersisa dalam *linked list* setelah operasi penambahan dan penghapusan.