MODUL I
LINKED LIST

1.1 PERMASALAHAN

1.1.1 List Manusia Viral

Oke singkat saja buatkan 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. 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 List Manusia Viral

1. Algoritma
   a. Inisialisasi struktur Node untuk menyimpan data manusia viral, lalu inisialisasi kelas LL dengan referensi ke node pertama (head).
   b. Buat metode TambahFirst, cek apakah list kosong, jika kosong, jadikan node baru sebagai head, jika tidak, arahkan next node baru ke head dan jadikan node baru sebagai head.
   c. Buat metode TambahLast, cek apakah list kosong, jika kosong, jadikan node baru sebagai head, jika tidak, iterasi hingga node terakhir dan tambahkan node baru setelahnya.
   d. Buat metode DeleteFirst, cek apakah list kosong, jika tidak kosong, pindahkan head ke node berikutnya.
   e. Buat metode DeleteLast, cek apakah list kosong atau hanya memiliki satu node, jika hanya satu node, kosongkan list dengan mengubah head menjadi null, jika lebih dari satu node, iterasi hingga node kedua terakhir dan hapus node terakhir.
   f. Buat metode Tampilkan, mulai dari head dan iterasi ke setiap node hingga akhir, tampilkan nama, skill, umur, dan hobi setiap node.
   g. Tambahkan "Vadel" menggunakan TambahLast.
   h. Tambahkan "Loli" menggunakan TambahLast.
   i. Tambahkan "Agus" dan "Fufa Fafa" menggunakan TambahLast.
   j. Tambahkan "Kak Gem" menggunakan TambahFirst, hapus "Fufa Fafa" menggunakan DeleteLast, hapus "Kak Gem" menggunakan DeleteFirst, dan tampilkan list manusia viral yang tersisa menggunakan Tampilkan.
   k. Akhiri algoritma.
2. *Source Code*

```
class Node {
    String nama;
    String skill;
    int umur;
    String hobi;
    Node next;
```

```
    public Node (String nama,String skill,int umur , String hobi ){
        this.nama=nama;
        this.skill=skill;
        this.umur=umur;
        this.hobi=hobi;
    }
    @Override
    public String toString(){
        return nama + umur + hobi;
    }
}


class LL {

    Node head;



   public void TambahFirst (String nama,String skill,int umur, String
hobi){
        Node baru = new Node(nama,skill,umur,hobi);

        if(head == null){
            head = baru;



        }else{
            Node current = head;
            while (current.next != null){
                current = current.next;
            }
            current.next=baru;


        }
    }
    public void TambahLast (String nama,String skill,int umur, String
hobi){
        Node baru = new Node(nama,skill,umur,hobi);
```

```
        if(head == null){
            head = baru;


        }else{
            baru.next=head;
            head=baru;

        }
    }


    public void Tampilkan (){
        Node current = head;
        while (current != null){
            System.out.println("nama : "+current.nama);
            System.out.println("skill : "+ current.skill);
            System.out.println("Umur : "+current.umur);
            System.out.println("Hobi : "+current.hobi);
            System.out.println("==============================");
            current = current.next;
        }
    }
    public Node DeleteFisrt(){

        if (head!= null){
            head = head.next;
        }
        return null;

    }
    public Node DeleteLast (){
        if (head.next == null){
            head = null;
        }else {
            Node temp = head;
            while(temp.next.next!=null){
                temp=temp.next;
            }
            temp.next =null;
```

```
        }
        return null;
    }
}

public class Main {


    public static void main(String[] args) {
        LL n = new LL();

        n.TambahLast("Vadel;", "Dancer geter", 19, "dance");
        n.TambahLast("Loli", "ATM Berjalan", 16, "bola");
        n.TambahLast("Agus", "Agus sakit", 35, "donasi");
        n.TambahLast("Fufa Fafa", "Roasting", 30, "Buku Kaskus");
        n.TambahFirst("kak Gem", "Kasi faham", 32, "Bernafas");



        n.DeleteLast();
        n.DeleteFisrt();
        n.Tampilkan();

//        System.out.println ("hai");

    }
}
```

1.3 ANALISIS DATA

1.3.1 List Manusia Viral

```
class Node {
```

Script "`class Node {`" berfungsi untuk mendefinisikan sebuah kelas bernama Node, yang merupakan bagian dari struktur data dalam program. Kelas ini digunakan untuk merepresentasikan node dalam linked list, di mana node ini akan menyimpan data dan referensi ke node berikutnya.

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

Script di atas berfungsi untuk mendeklarasikan atribut-atribut yang dimiliki oleh kelas Node. Atribut "`String nama`", "`String skill`", dan "`String hobi`" digunakan untuk menyimpan informasi berupa kalimaat dari node. Sementara itu, "`int umur`" digunakan untuk menyimpan angka dari node tersebut. Atribut "`Node next`" berfungsi sebagai referensi untuk menunjuk ke node berikutnya dalam linked list, penghubungan antar node.

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

Script di atas berfungsi sebagai konstruktor untuk kelas Node. Konstruktor ini digunakan untuk menginisialisasi objek Node dengan nilai-nilai yang diberikan. Parameter "`String nama`", "`String skill`", "`int umur`", dan "`String hobi`" digunakan untuk menerima input saat membuat node baru dari kelas Node. Di dalam konstruktor, kata kunci "`this`" digunakan untuk membedakan atribut kelas dengan parameter yang memiliki nama yang sama, sehingga setiap atribut diisi dengan nilai yang sesuai.

```
@Override
    public String toString(){
        return nama + umur + hobi;
    }
```

Script di atas berfungsi untuk mengoverride metode "`toString()`" dari kelas induknya. Metode ini mengembalikan representasi string dari objek Node. Di dalam metode ini, nilai dari atribut "`nama, umur, dan hobi`" akan digabungkan menjadi satu string tanpa pemisah, yang memungkinkan pengguna untuk melihat informasi tersebut secara langsung saat objek Node di tampilkan.

```
class LL {
```

Script di atas berfungsi untuk mendefinisikan sebuah kelas bernama LL, yang digunakan untuk merepresentasikan struktur data linked list. Kelas ini akan menyimpan dan mengelola node-node yang terhubung, memungkinkan operasi seperti penambahan, penghapusan, dan penampilan data yang tersimpan dalam bentuk daftar.

```
Node head;
```

Script di atas berfungsi untuk mendeklarasikan atribut bernama "`head`" dalam kelas LL. Atribut ini bertipe Node dan digunakan untuk menyimpan referensi ke node pertama dalam linked list. Dengan adanya atribut "`head`", kelas LL dapat mengelola dan mengakses seluruh node yang ada dalam linked list mulai dari node pertama.

```
public void TambahFirst (String nama,String skill,int umur, String hobi){
        Node baru = new Node(nama,skill,umur,hobi);

        if(head == null){
            head = baru;


        }else{
            Node current = head;
            while (current.next != null){
                current = current.next;
            }
            current.next=baru;


        }
    }
```

Script di atas berfungsi untuk menambahkan node baru di akhir linked list. Dalam metode ini, objek baru dari kelas `Node` dibuat dengan parameter "`nama`", "`skill`", "`umur`", dan "`hobi`". Jika atribut "`head`" bernilai "`null`", artinya linked list kosong, sehingga node baru akan menjadi node pertama. Jika tidak, metode ini akan mencari node terakhir dengan melakukan iterasi melalui linked list menggunakan variabel "`current`". Setelah mencapai node terakhir, referensi "`next`" dari node tersebut diupdate untuk menunjuk ke node baru yang ditambahkan.

```
public void TambahLast (String nama,String skill,int umur, String hobi){


        Node baru = new Node(nama,skill,umur,hobi);


        if(head == null){
            head = baru;


        }else{
            baru.next=head;
            head=baru;
```

```
        }
    }
```

Script di atas berfungsi untuk menambahkan node baru di awal linked list. Dalam metode ini, objek baru dari kelas Node dibuat dengan parameter "`nama`", "`skill`", "`umur`", dan "`hobi`". Jika atribut "`head`" bernilai "`null`", artinya linked list kosong, sehingga node baru akan menjadi node pertama. Jika tidak, node baru akan ditambahkan di depan linked list dengan mengupdate referensi "`next`" dari node baru untuk menunjuk ke node pertama yang ada saat ini, kemudian atribut "`head`" diupdate untuk menunjuk ke node baru, menjadikannya sebagai node pertama dalam linked list.

```
public void Tampilkan (){
        Node current = head;
        while (current != null){
            System.out.println("nama : "+current.nama);
            System.out.println("skill : "+ current.skill);
            System.out.println("Umur : "+current.umur);
            System.out.println("Hobi : "+current.hobi);
            System.out.println("================================");
            current = current.next;
        }
    }
```

Script di atas berfungsi untuk menampilkan informasi dari setiap node dalam linked list. Metode ini dimulai dengan mendeklarasikan variabel "`current`" yang diinisialisasi dengan atribut "`head`". Dengan menggunakan "`while`", metode ini akan terus berjalan selama "`current`" tidak bernilai "`null`", yang menunjukkan bahwa masih ada node yang perlu ditampilkan. Untuk setiap node, informasi tentang "`nama`", "`skill`", "`umur`", dan "`hobi`" dicetak, diikuti dengan garis pemisah. Setelah menampilkan informasi setiap node, variabel "`current`" diupdate untuk menunjuk ke node berikutnya, sehingga proses ini berlanjut hingga semua node ditampilkan.

```
public Node DeleteFisrt(){

        if (head!= null){
            head = head.next;
        }

        return null;

    }
```

Script di berfungsi untuk menghapus node pertama dari linked list. Metode ini memeriksa apakah atribut "`head`" tidak bernilai "`null`", yang berarti linked list tidak kosong. Jika tidak kosong, atribut "`head`" diupdate untuk menunjuk ke node berikutnya dalam linked list, efektif menghapus node pertama. Metode ini kemudian mengembalikan "`null`",

menunjukkan bahwa tidak ada nilai yang dikembalikan setelah penghapusan. Jika linked list kosong, tidak ada yang diubah.

```
public Node DeleteLast (){
        if (head.next == null){
            head = null;
        }else {
            Node temp = head;
            while(temp.next.next!=null){
                temp=temp.next;
            }
            temp.next =null;
        }
        return null;
    }
```

Script berfungsi untuk menghapus node terakhir dari linked list. Metode ini pertama-tama memeriksa apakah hanya ada satu node dalam linked list dengan mengecek jika "`head.next`" bernilai "`null`". Jika benar, maka linked list menjadi kosong dengan mengatur "`head`" menjadi "`null`". Jika ada lebih dari satu node, variabel sementara "`temp`" diinisialisasi dengan "`head`" dan digunakan untuk menelusuri linked list hingga mencapai node kedua dari terakhir , yaitu node yang memiliki "`next`" menunjuk ke "`null`". Setelah mencapai node tersebut, referensi "`next`" dari node tersebut diatur menjadi "`null`", yang menghapus node terakhir dari linked list.

```
public class Main {
   }
```

Script berfungsi untuk mendeklarasikan kelas bernama Main. Kelas ini biasanya digunakan sebagai titik awal eksekusi program Java. Dalam konteks program Java, kelas Main akan berisi metode "`main`", yang merupakan metode utama yang dijalankan ketika program dieksekusi. Kelas ini dapat menyimpan logika dan struktur lainnya yang diperlukan untuk menjalankan program.

```
public static void main(String[] args) {
     }
```

Script di atas berfungsi untuk mendeklarasikan metode "`main`", yang merupakan titik masuk utama untuk eksekusi program Java. Metode ini diawali dengan kata kunci "`public`", yang menandakan bahwa metode ini dapat diakses dari mana saja, termasuk di luar kelas yang mendeklarasikannya.

```
LL n = new LL();
```

Script di atas berfungsi untuk mendeklarasikan dan menginisialisasi objek n dari kelas LL. Proses ini menggunakan kata kunci "`new`" untuk membuat instance baru, yang

memungkinkan objek n untuk mengakses metode dan atribut dalam kelas LL, seperti menambahkan atau menghapus node dalam linked list.

```
        n.TambahLast("Vadel;", "Dancer geter", 19, "dance");
        n.TambahLast("Loli", "ATM Berjalan", 16, "bola");
        n.TambahLast("Agus", "Agus sakit", 35, "donasi");
        n.TambahLast("Fufa Fafa", "Roasting", 30, "Buku Kaskus");
        n.TambahFirst("kak Gem", "Kasi faham", 32, "Bernafas");


        n.DeleteLast();
        n.DeleteFisrt();
        n.Tampilkan();
```

Script di atas berfungsi untuk mengelola linked list menggunakan objek n dari kelas LL. Pada baris pertama hingga keempat, metode "`TambahLast()`" dipanggil untuk menambahkan beberapa node baru ke dalam linked list dengan informasi yang berbeda, seperti nama, skill, umur, dan hobi. Metode "`TambahFirst()`" kemudian digunakan untuk menambahkan node baru di awal linked list. Setelah itu, metode "`DeleteLast()`" dan "`DeleteFisrt()`" dipanggil untuk menghapus node terakhir dan node pertama dari linked list. Akhirnya, metode "`Tampilkan()`" digunakan untuk menampilkan semua node yang tersisa dalam linked list, mencetak informasi setiap node, termasuk nama, skill, umur, dan hobi.