MODUL I

LINKED LIST

1.1 PERMASALAHAN

1.1.1 List Manusia Viral

Kak Gem ingin dibuatkan 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 node dengan atribut seperti nama, skill, umur, hobi, dan *pointer next*.
   b. Inisialisasi list dengan membuat sebuah list kosong dengan head yang menunjuk ke *null*.
   c. Membuat fungsi yang bisa menambah node di depan dengan cara mengarahkan *pointer next* dari node baru ke head, kemudian perbarui head menjadi node baru.
   d. Membuat fungsi yang bisa menambah node di belakang dengan cara menyambungkan node baru di akhir.
   e. Membuat fungsi yang bisa menghapus node pertama dengan cara memindahkan *head* ke node berikutnya.
   f. Membuat fungsi yang bisa menghapus node terakhir dengan cara mengatur pointer *next* dari node kedua terakhir menjadi *null.*
   g. Membuat fungsi yang bisa menampilkan semua node lengkap dengan atributnya.
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;
    }
}
class Single {
    Node head;
    public Single() {
      this.head = null;
    }
    public void addFirst (String nama, String skill, int umur, String
hobi) {
      Node newNode = new Node(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) {
      Node newNode = new Node(nama, skill, umur, hobi);
      if (head == null) {
```

```
        head = newNode;
      } else {
        Node temp = head;
        while (temp.next != null) {
          temp = temp.next;
        }
        temp.next = newNode;
      }
    }
    public void deleteFirst () {
      if (head == null) {
        System.out.println("list kosong");
      } else {
        head = head.next;
      }
    }
    public void deleteLast () {
      if (head == null) {
        System.out.println("list kosong");
      } else {
        Node temp = head;
        while (temp.next.next != null) {
          temp = temp.next;
        }
        temp.next = null;
      }
    }
    public void display () {
      Node 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 JurnalEasy {
    public static void main(String[] args) {
        Single list = new Single();
        list.addLast("Vadel", "Dance Geter", 19, "Dance");
        list.addLast("Agus", "Agus Sakit", 35, "Donasi");
        list.addLast("Loli", "ATM Berjalan", 16, "Liat Vadel");
        list.addLast("Fufu Fafa", "Roasting", 30, "Buka Kaskus");
        System.out.println("\nSebelum add kak gem");
        list.display();
        list.addFirst("Kak Gem", "Kasi Faham", 32, "Bernafas");
        System.out.println("\nSetelah add kak gem");
        list.display();
        list.deleteLast();
        System.out.println("\nSetelah hapus fufufafa");
        list.display();
        list.deleteFirst();
        System.out.println("\nSetelah hapus kak gem / manusia yang
tersisa");
        list.display();
    }
}
```

1.3 ANALISIS DATA

1.3.1 Nama Program

```
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;
    }
}
```

Kode ini menjelaskan kelas "`Node`". Kelas ini memiliki 4 atribut: "`nama`", "`skill`", "`umur`", "`hobi`". Karena ini *single link list*, makanya hanya ada 1 *pointer* yaitu "`Node next`". Kelas ini juga memiliki konstraktor yang menginisialisasi atribut atribut tadi.

```
class Single {
    Node head;
    public Single() {
      this.head = null;
    }
```

Kode ini adalah kelas "`Single`" yang mengimplementasikan *single link list*. Konstruktor "`Single()`" menginisialisasi list dengan menetapkan "`head`" ke "`null`", yang berarti bahwa *linked list* tersebut dimulai dalam keadaan kosong.

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

Kode di atas adalah metode untuk menambahkan node baru ke awal *linked list*. Jika list kosong, node baru menjadi "`head`". Jika tidak, node baru diarahkan ke "`head`" saat ini dan kemudian "`head`" diperbarui untuk menunjuk ke node baru tersebut.

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

```
            temp = temp.next;
          }
          temp.next = newNode;
        }
      }
```

Kode di atas merupakan untuk menambahkan sebuah node baru ke akhir dari sebuah *linked list*. Metode ini menggunakan variabel "`temp`" menemukan node terakhir. Setelah menemukan node terakhir, "`temp.next`" akan diatur untuk mengarah ke node baru, sehingga node baru tersebut ditambahkan di akhir *linked list*.

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

Kode di atas digunakan untuk menghapus elemen pertama dari sebuah *linked list* dengan cara mengatur "`head`" untuk menunjuk ke node berikutnya, sehingga menghapus elemen pertama dari list.

```
public void deleteLast () {
      if (head == null) {
        System.out.println("list kosong");
      } else {
        Node temp = head;
        while (temp.next.next != null) {
          temp = temp.next;
        }
        temp.next = null;
      }
    }
```

Kode di atas digunakan untuk menghapus node terakhir dari *linked list* dengan cara mencari node kedua terakhir dan mengatur *pointer* "`next`"-nya menjadi "`null`", sehingga menghapus node terakhir dari *linked list*.

```
public void display () {
      Node 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;
        }
    }
}
```

Kode ini adalah metode "`display()`" yang digunakan untuk menampilkan isi dari *linked list*. Metode ini dimulai dari "`head`" kemudian bergerak ke node berikutnya dengan "`next`" sampai mencapai akhir dari *linked list*.

```
public class JurnalEasy {
    public static void main(String[] args) {
        Single list = new Single();

        list.addLast("Vadel", "Dance Geter", 19, "Dance");
        list.addLast("Agus", "Agus Sakit", 35, "Donasi");
        list.addLast("Loli", "ATM Berjalan", 16, "Liat Vadel");
        list.addLast("Fufu Fafa", "Roasting", 30, "Buka Kaskus");

        System.out.println("\nSebelum add kak gem");
        list.display();
        list.addFirst("Kak Gem", "Kasi Faham", 32, "Bernafas");
        System.out.println("\nSetelah add kak gem");
        list.display();
        list.deleteLast();
        System.out.println("\nSetelah hapus fufufafa");
        list.display();
        list.deleteFirst();
        System.out.println("\nSetelah hapus kak gem / manusia yang
tersisa");
        list.display();
    }
}
```

Kode di atas merupakan metode main dari keseluruhan kode. Sebuah *linked list* kosong dibuat melalui objek "`Single`", dan beberapa node ditambahkan di akhir list menggunakan "`addLast`" untuk menyimpan informasi individu manusia viral seperti "`Vadel`", "`Agus`", "`Loli`", dan "`Fufu Fafa`". Setelah itu, list ditampilkan, kemudian node "`Kak Gem`" ditambahkan menggunakan "`addFirst`", dan hasilnya ditampilkan lagi. Node terakhir dihapus menggunakan "`deleteLast`", dan setelahnya node pertama dihapus dengan "`deleteFirst`" kemudian ditampilkan manusia manusia yang tersisa pada *list*.