MODUL I
LINKEDLIST

1.1 PERMASALAHAN

Seseorang ingin dibuatkan *List* Manusia Viral, dimana dalam folder tersebut didalamnya harus ada method addFirst, addLast, deleteFirst, deleteLast, dan method untuk menampilkan manusia-manusia itu. Dalam pembuatan *List* Manusia Viral, buat parameternya dengan nama, Nama Manusia, Skills, 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 Fafaa dari *List* dengan method deleteLast, terakhir, hapus Kak Gem dengan method deleteFist dan tampilkan *List* manusia-manusia yang tersisa.

Tabel 1.1 Permasalahan

Nama:Kak Gem	Nama:Vadel	Nama:Loli	Nama:Agus	Nama:Fufu Fafa
Skill: Kasi faham	Skill:Dance Geter	Skill:ATM berjalan	Skill:Agus sakit	Skill:Roasting
Umur:32	Umur:19	Umur:16	Umur:35	Umur:30
Hobi:Bernafas	Hobi:Dance	Hobi:Liat Vadel	Hobi:Donasi	Hobi:Buka Kaskus

1.1.1 PROGRAM SINGLE LINKEDLIST

1. Algoritma
   a. Membuat sebuah objek yang merepresentasikan manusia viral.
   b. Menentukan beberapa atribut untuk manusia.
   c. Menambahkan sebuah atribut tambahan yang akan menyimpan referensi ke manusia viral.
   d. Membuat fungsi yang digunakan untuk membangun objek.
   e. Menggunakan atribut tambahan untuk membuat hubungan antar manusia viral.
   f. Menginisialisasi sebuah daftar terhubung tunggal yang berisi objek manusia viral.
   g. Menambahkan manusia viral ke awal daftar dengan membuat objek baru.
   h. Menambahkan manusia viral ke akhir daftar.
   i. Menghapus manusia viral pertama dalam daftar.
   j. Menghapus manusia viral terakhir dalam daftar. Jika hanya ada satu elemen, jadikan kepala null.
   k. Menampilkan daftar manusia viral .
   l. Menguji operasi dengan menambahkan beberapa elemen ke daftar.
2. Source Code

```
public class ManusiaViral{
    String nama;
    String hobi;
    Orang berikutnya;
    int umur;
    String skills;
    public ManusiaViral(String nama, String skills, int umur, String
hobi){
        this.nama = nama;
        this.hobi = hobi;
        this.berikutnya = null;
        this.umur = umur;
        this.skills = keterampilan;
    }
}

Public class Singlelinkedlist {

    private ManusiaViral head;

    public ViralList() {

        this.head = null;
    }
    public void addFirst(String nama, String skills, int umur, String
hobi){
        ManusiaViral newManusiaViral = new ManusiaViral(, skills,
umur, hobi);
        newManusiaViral.next = head;
        head = newManusiaViral;
    }
```

```
    public void addLast(String nama, String skills, int umur, String
hobi){
        ManusiaViral   newManusiaViral   =   new   ManusiaViral(nama,
skills, umur, hobi);
        if (head == null) {
            head = newManusiaViral;
        } else {
            ManusiaViral temp = head;
            while (temp.next != null) {\}
        }
        temp.next = newManusiaViral;
    }
}
public void deleteFirst(){
    if (head != null) {
        head = head.next;
    }
}
public void deleteLast(){
    if (head != null){
        if (head.next == null){
            head = null;
        } else {
            ManusiaViral temp = head;
            while (temp.next.next != null){
                temp = temp.next;
            }
            temp.next = null;
        }
    }
}
public void displayList(){
    ManusiaViral temp = head;
    while (temp != null){
        System.out.println("Nama: " + temp.nama + ", Skills: " +
temp.skills + ", Umur: " + temp.umur + ", Hobi: " + temp.hobi);
        temp = temp.next;
    }
}
public static void main(String[] args) {
    Singlelinkedlist list = new Singlelinkedlist();
    list.addFist("Kak Gem", "Kasi Faham", 32, "Bikin Meme");
    list.addLast("Vadel", "Dance Getter", 18, "Dance");
    list.addLast("Loli", "ATM Berjalan", 35, "Donasi");
    list.addLast("Agus", "Roasting", 35, "Donasi");
    list.addLast("Fufu Fafa", "Buka Kaskus", 30, "Buka Kaskus");
    list.deleteLast();
    list.deleteFirst();
    list.displayList();
}
```

2.1 ANALISIS DATA

2.1.1 Program Single LinkedList

```
public class ManusiaViral{
    String nama;
    String hobi;
    Orang berikutnya;
    int umur;
    String skills;
```

Script "`public class ManusiaViral`:" adalah kode public class ManusiaViral: mendefinisikan sebuah kelas bernama "`ManusiaViral`" yang dapat diakses secara publik di seluruh program. Kelas ini merepresentasikan manusia viral sebagai objek dengan beberapa atribut: "`String nama`" adalah atribut menyimpan nama manusia viral dalam bentuk teks. "`String skills`" atribut ini digunakan untuk menyimpan keterampilan manusia viral, juga dalam bentuk teks. "`int umur`" atribut ini menyimpan umur manusia viral dalam bentuk bilangan bulat (integer). "String hobi" atribut ini menyimpan hobi manusia viral, juga sebagai teks. "`Orang berikutnya`" atribut ini adalah referensi ke objek lain. Atribut ini digunakan untuk menunjuk ke manusia viral berikutnya dalam struktur data seperti linked list.

```
public ManusiaViral(String nama, String skills, int umur, String hobi){
        this.nama = nama;
        this.hobi = hobi;
        this.berikutnya = null;
        this.umur = umur;
        this.skills = keterampilan;
    }
}
```

Metode "`public ManusiaViral (String nama, String skills, int umur, String hobi)`" adalah konstruktor untuk kelas "`ManusiaViral`" yang digunakan untuk menginisialisasi objek baru dengan atribut-atribut nama, keterampilan, umur, dan hobi. Nilai yang diterima oleh parameter nama akan diatur ke atribut nama, sementara skills, umur, dan hobi juga ke atribut masing-masing. Selain itu, atribut berikutnya diinisialisasi sebagai null, menandakan bahwa objek tersebut belum memiliki referensi ke objek lain dalam konteks linked list.

```
Public class Singlelinkedlist {
    private ManusiaViral head;

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

Kelas "`Singlelinkedlist`" mendefinisikan struktur data linked *list* tunggal (singly linked *list*) dengan atribut *head* yang menyimpan referensi ke node pertama dalam daftar,

bertipe "`ManusiaViral`". Konstruktor "`ViralList()`" menginisialisasi head sebagai null, menunjukkan bahwa daftar tersebut kosong ketika pertama kali dibuat.

```
public void addFirst(String nama, String skills, int umur, String hobi){
        ManusiaViral newManusiaViral = new ManusiaViral(, skills, umur,
hobi);
        newManusiaViral.next = head;
        head = newManusiaViral;
    }
```

Metode "`addFirst`" adalah sebuah fungsi yang menambahkan node baru di awal daftar terhubung. Node baru, "`newManusiaViral`", dibuat menggunakan informasi yang diberikan (nama, keterampilan, umur, dan hobi). Referensi *next* pada node baru kemudian diatur untuk menunjuk ke *head* saat ini. Terakhir, head diperbarui untuk menunjuk ke node baru tersebut, menjadikannya node pertama dalam daftar.

```
public void addLast(String nama, String skills, int umur, String hobi){
        ManusiaViral newManusiaViral = new ManusiaViral(nama, skills,
umur, hobi);
        if (head == null) {
            head = newManusiaViral;
        } else {
            ManusiaViral temp = head;
            while (temp.next != null) {\}
        }
        temp.next = newManusiaViral;
    }
}
```

Metode "`addLast`" adalah fungsi yang menambahkan node baru di akhir daftar terhubung. Node baru, "`newManusiaViral`", dibuat menggunakan informasi yang diberikan (nama, keterampilan, umur, dan hobi). Jika head adalah null, maka node baru akan menjadi node pertama dalam daftar. Jika tidak, sebuah variabel sementara bernama temp diinisialisasi untuk menunjuk ke head. Metode ini kemudian menggunakan *loop while* untuk mencari node terakhir dengan memeriksa temp.next hingga mencapai null. Setelah loop selesai, temp.next diatur untuk menunjuk ke node baru, sehingga menambahkannya di akhir daftar.

```
public void deleteFirst(){
    if (head != null) {
        head = head.next;
    }
}
```

Metode "`deleteFirst`" adalah sebuah fungsi yang menghapus node pertama dari daftar terhubung. Jika head tidak bernilai null, maka head diperbarui untuk menunjuk ke node berikutnya "`head.next`", sehingga node pertama dihapus dari daftar.

```
public void deleteLast(){
    if (head != null){
        if (head.next == null){
            head = null;
        } else {
```

```
            ManusiaViral temp = head;
            while (temp.next.next != null){
                temp = temp.next;
            }
            temp.next = null;
        }
    }
}
```

Metode "deleteLast" adalah fungsi yang menghapus node terakhir dari daftar terhubung. Jika head tidak bernilai null dan hanya ada satu node, maka head diatur menjadi null. Namun, jika terdapat lebih dari satu node, variabel temp digunakan untuk menemukan node sebelum terakhir dengan menggunakan *loop*, kemudian "temp.next" diatur menjadi null, sehingga node terakhir dihapus dari daftar.

```
public void displayList(){
    ManusiaViral temp = head;
    while (temp != null){
        System.out.println("Nama: " + temp.nama + ", Skills: " +
temp.skills + ", Umur: " + temp.umur + ", Hobi: " + temp.hobi);
        temp = temp.next;
    }
}
```

Metode "displayList" adalah fungsi yang menampilkan semua node dalam daftar terhubung. Variabel *temp* diatur untuk menunjuk ke *head*, dan kemudian menggunakan *loop* while untuk mencetak informasi dari setiap *node* hingga mencapai akhir daftar, yaitu ketika temp menjadi null.

```
public static void main(String[] args) {
    Singlelinkedlist list = new Singlelinkedlist();
    list.addFist("Kak Gem", "Kasi Faham", 32, "Bikin Meme");
    list.addLast("Vadel", "Dance Getter", 18, "Dance");
    list.addLast("Loli", "ATM Berjalan", 35, "Donasi");
    list.addLast("Agus", "Roasting", 35, "Donasi");
    list.addLast("Fufu Fafa", "Buka Kaskus", 30, "Buka Kaskus");
    list.deleteLast();
    list.deleteFirst();
    list.displayList();
}
```

Fungsi "main" adalah fungsi utama yang menjalankan keseluruhan program. Di dalam metode "main", sebuah *instance* dari "Singlelinkedlist" dibuat, dan beberapa node ditambahkan menggunakan metode "addFirst" dan "addLast". Setelah itu, node terakhir dihapus dengan "deleteLast", diikuti oleh penghapusan node pertama menggunakan deleteFirst. Terakhir, semua node yang tersisa ditampilkan dengan memanggil "displayList".