MODUL I
LINKED LIST

1.1 PERMASALAHAN

1.1.1 *Single Linked list*

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 *Single Linked List*

1. Algoritma
   a. Buat objek daftar (linked list) bernama OrangViral yang dimulai dengan keadaan kosong (tidak ada orang).
   b. Minta pengguna memasukkan nama, hobi, skill, dan umur.
   c. Buat objek Orang baru dengan data tersebut.
   d. Periksa apakah daftar kosong.
   e. Jika daftar tidak kosong, buang orang pertama dari daftar, dan jadikan orang kedua sebagai orang pertama.
   f. Program akan terus berulang (*loop*) hingga pengguna memilih opsi Keluar dari menu, lalu program berhenti.
2. *Source Code*

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

       // Constructor
   public Orang(String nama, String hobi, String skill, int umur) {
           this.nama = nama;
           this.hobi = hobi;
           this.skill = skill;
```

```
        this.umur = umur;
        this.next = null;
    }
}

class OrangViral {
    Orang head;

    public OrangViral() {
        head = null;
    }

    // Menambahkan orang di depan list
    public void addFirst(String nama, String hobi, String skill,
int umur) {
        Orang orangBaru = new Orang(nama, hobi, skill, umur);
        if (head == null) {
            head = orangBaru;
        } else {
            orangBaru.next = head;
            head = orangBaru;
        }
    }

    // Menambahkan orang di akhir list
    public void addLast(String nama, String hobi, String skill, int
umur) {
        Orang orangBaru = new Orang(nama, hobi, skill, umur);
        if (head == null) {
            head = orangBaru;
        } else {
            Orang temp = head;
            while (temp.next != null) {
                temp = temp.next;
            }
            temp.next = orangBaru;
        }
    }

    // Menghapus orang dari depan list
    public void deleteFirst() {
        if (head == null) {
  System.out.println("List kosong, tidak ada yang bisa dihapus.");
        } else {
            System.out.println("Menghapus: " + head.nama);
            head = head.next;
        }
    }

    // Menghapus orang dari akhir list
    public void deleteLast() {
        if (head == null) {
            System.out.println("List kosong, tidak ada yang bisa
dihapus.");
        } else if (head.next == null) {
            System.out.println("Menghapus: " + head.nama);
            head = null;
```

```
            } else {
                Orang temp = head;
                while (temp.next.next != null) {
                    temp = temp.next;
                }
                System.out.println("Menghapus: " + temp.next.nama);
                temp.next = null;
            }
        }

        // Menampilkan seluruh orang yang ada di dalam list
        public void displayList() {
            if (head == null) {
                System.out.println("List kosong.");
            } else {
                Orang current = head;
                while (current != null) {
                    System.out.println("Nama: " + current.nama);
                    System.out.println("Hobi: " + current.hobi);
                    System.out.println("Skill: " + current.skill);
                    System.out.println("Umur: " + current.umur);
                    System.out.println("--------------------------");
                    current = current.next;
                }
            }
        }

        public static void main(String[] args) {
            OrangViral list = new OrangViral();
            Scanner input = new Scanner(System.in);
            boolean lanjut = true;

            while (lanjut) {
                System.out.println("Pilih tindakan:");
                System.out.println("1. Tambah orang di depan");
                System.out.println("2. Tambah orang di belakang");
                System.out.println("3. Hapus orang dari depan");
                System.out.println("4. Hapus orang dari belakang");
                System.out.println("5. Tampilkan daftar orang");
                System.out.println("6. Keluar");
                System.out.print("Masukkan pilihan: ");
                int pilihan = input.nextInt();
                input.nextLine(); // menangkap newline

                switch (pilihan) {
                    case 1:
                        // Tambah orang di depan
                        System.out.print("Masukkan nama: ");
                        String namaDepan = input.nextLine();
                        System.out.print("Masukkan hobi: ");
                        String hobiDepan = input.nextLine();
                        System.out.print("Masukkan skill: ");
                        String skillDepan = input.nextLine();
                        System.out.print("Masukkan umur: ");
                        int umurDepan = input.nextInt();
    list.addFirst(namaDepan, hobiDepan, skillDepan, umurDepan);
                        break;
```

```
                    case 2:
                        // Tambah orang di belakang
                        System.out.print("Masukkan nama: ");
                        String namaBelakang = input.nextLine();
                        System.out.print("Masukkan hobi: ");
                        String hobiBelakang = input.nextLine();
                        System.out.print("Masukkan skill: ");
                        String skillBelakang = input.nextLine();
                        System.out.print("Masukkan umur: ");
                     int umurBelakang = input.nextInt();
list.addLast(namaBelakang, hobiBelakang, skillBelakang, umurBelakang);
                        break;
                    case 3:
                        // Hapus orang dari depan
                        list.deleteFirst();
                        break;
                    case 4:
                        // Hapus orang dari belakang
                        list.deleteLast();
                        break;
                    case 5:
                        // Tampilkan daftar orang
                        list.displayList();
                        break;
                    case 6:
                        // Keluar
                        lanjut = false;
                        break;
                    default:
                        System.out.println("Pilihan tidak valid.");
                }}
            input.close();
        }
       }
```

1.3 Analisis Data

1.3.1 *Single Linked List*

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

Script kelas "`Orang`" adalah sebuah kelas yang digunakan untuk merepresentasikan sebuah "`node`" dalam struktur data *linked list*. Setiap objek dari kelas ini menyimpan informasi pribadi seseorang, yaitu "`nama`", "`hobi`", "`skill`", dan "`umur`". Selain itu, kelas ini memiliki atribut "`next`" yang merupakan referensi (pointer) ke objek "`Orang`" berikutnya dalam daftar (*linked list*). Dengan demikian, setiap objek "`Orang`" dapat dihubungkan dengan objek "`Orang`" lainnya,

membentuk rantai atau daftar orang. Atribut "`next`" akan bernilai null jika objek tersebut adalah orang terakhir dalam daftar.

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

class OrangViral {
    Orang head;

    public OrangViral() {
        head = null;
    }
```

Script "`Orang`" digunakan untuk menginisialisasi objek baru dengan atribut "`nama`", "hobi", "`skill`", dan "`umur`" yang diberikan sebagai parameter, serta mengatur "`next`" menjadi "`null`" karena objek baru belum terhubung ke objek lain dalam *linked list*. Sementara itu, kelas "`OrangViral`" berfungsi sebagai struktur *linked list* itu sendiri, di mana atribut "`head`" menunjukkan elemen pertama dari daftar. Konstruktor "`OrangViral`" menginisialisasi daftar sebagai kosong dengan mengatur "`head`" menjadi "`null`".

```
// Menambahkan orang di depan list
    public void addFirst(String nama, String hobi, String skill, int umur)
{
        Orang orangBaru = new Orang(nama, hobi, skill, umur);
        if (head == null) {
            head = orangBaru;
        } else {
            orangBaru.next = head;
            head = orangBaru;
        }
    }
```

Script "`addFirst`" digunakan untuk menambahkan objek "`Orang baru`" di bagian depan dari *linked list*. Metode ini pertama-tama membuat objek "`Orang baru`" menggunakan data yang diterima sebagai parameter ("`nama`", "`hobi`", "`skill`", dan "`umur`").

```
   // Menambahkan orang di akhir list
   public void addLast(String nama, String hobi, String skill, int umur) {
       Orang orangBaru = new Orang(nama, hobi, skill, umur);
       if (head == null) {
           head = orangBaru;
       } else {
           Orang temp = head;
           while (temp.next != null) {
```

```
                temp = temp.next;
            }
            temp.next = orangBaru;
        }
    }
```

Script "`addLast`" menambahkan objek Orang baru di akhir *linked list*. Pertama, objek baru dibuat dengan data yang diberikan. Jika "`head`" kosong, objek baru menjadi elemen pertama. Jika tidak, metode ini menggunakan variabel sementara "`temp`" untuk mencari elemen terakhir dengan *loop* "`while`". Setelah menemukan elemen terakhir (yang memiliki "`next`" bernilai "`null`"), objek baru dihubungkan dengan mengatur "`temp.next`" menjadi objek baru, sehingga objek baru kini berada di akhir list.

```
// Menghapus orang dari depan list
    public void deleteFirst() {
        if (head == null) {
            System.out.println("List kosong, tidak ada yang bisa
        dihapus.");
        } else {
            System.out.println("Menghapus: " + head.nama);
            head = head.next;
        }
    }

    // Menghapus orang dari akhir list
    public void deleteLast() {
        if (head == null) {
            System.out.println("List kosong, tidak ada yang bisa
        dihapus.");
        } else if (head.next == null) {
            System.out.println("Menghapus: " + head.nama);
            head = null;
        } else {
            Orang temp = head;
            while (temp.next.next != null) {
                temp = temp.next;
            }
            System.out.println("Menghapus: " + temp.next.nama);
            temp.next = null;
        }
    }
```

Script "`deleteFirst`" menghapus objek "`Orang`" dari depan *linked list*. Jika "`head`" kosong, yang berarti daftar juga kosong, metode ini menampilkan pesan bahwa tidak ada yang bisa dihapus. Jika tidak, metode ini mencetak nama orang yang dihapus dan memperbarui "`head`" menjadi elemen berikutnya dalam daftar.Sementara itu, metode "`deleteLast`" menghapus objek "`Orang`" dari akhir *linked list*. Jika daftar kosong, metode ini juga

menampilkan pesan yang sama. Jika hanya ada satu elemen (di mana "`head.next`" bernilai "`null`")

```
// Menampilkan seluruh orang yang ada di dalam list
    public void displayList() {
        if (head == null) {
            System.out.println("List kosong.");
        } else {
            Orang current = head;
            while (current != null) {
                System.out.println("Nama: " + current.nama);
                System.out.println("Hobi: " + current.hobi);
                System.out.println("Skill: " + current.skill);
                System.out.println("Umur: " + current.umur);
                System.out.println("--------------------------");
                current = current.next;
            }
        }
    }
```

Script "`displayList`" berfungsi untuk menampilkan semua objek "`Orang`" yang ada dalam *linked list*. Pertama, metode ini memeriksa apakah "`head`" kosong, yang menandakan bahwa daftar juga kosong. Jika demikian, ia akan mencetak pesan "`List kosong`." Jika ada elemen dalam daftar, metode ini menggunakan variabel "`current`" yang diatur ke "`head`".

```
public static void main(String[] args) {
        OrangViral list = new OrangViral();
        Scanner input = new Scanner(System.in);
        boolean lanjut = true;

        while (lanjut) {
            System.out.println("Pilih tindakan:");
            System.out.println("1. Tambah orang di depan");
            System.out.println("2. Tambah orang di belakang");
            System.out.println("3. Hapus orang dari depan");
            System.out.println("4. Hapus orang dari belakang");
            System.out.println("5. Tampilkan daftar orang");
            System.out.println("6. Keluar");
            System.out.print("Masukkan pilihan: ");
            int pilihan = input.nextInt();
            input.nextLine(); // menangkap newline
```

Script "`main`" adalah titik masuk program yang mengelola interaksi pengguna dengan daftar orang. Di dalamnya, objek "`OrangViral`" dan "`Scanner`" dibuat untuk menangani daftar dan *input*. Sebuah *loop* "`while`" menampilkan "`menu`" untuk menambah atau menghapus orang, menampilkan daftar, dan keluar dari aplikasi. Setelah meminta pengguna untuk memasukkan pilihan, program memproses tindakan yang sesuai. Baris "`input.nextLine()`" digunakan

untuk menangkap karakter "`newline`" yang tersisa, memastikan input berikutnya dapat dibaca dengan benar, sehingga memungkinkan pengguna berinteraksi hingga memilih untuk keluar.

```
switch (pilihan) {
                case 1:
                    // Tambah orang di depan
                    System.out.print("Masukkan nama: ");
                    String namaDepan = input.nextLine();
                    System.out.print("Masukkan hobi: ");
                    String hobiDepan = input.nextLine();
                    System.out.print("Masukkan skill: ");
                    String skillDepan = input.nextLine();
                    System.out.print("Masukkan umur: ");
                    int umurDepan = input.nextInt();
                    list.addFirst(namaDepan, hobiDepan, skillDepan,
      umurDepan);
                    break;
                case 2:
                    // Tambah orang di belakang
                    System.out.print("Masukkan nama: ");
                    String namaBelakang = input.nextLine();
                    System.out.print("Masukkan hobi: ");
                    String hobiBelakang = input.nextLine();
                    System.out.print("Masukkan skill: ");
                    String skillBelakang = input.nextLine();
                    System.out.print("Masukkan umur: ");
                    int umurBelakang = input.nextInt();
                    list.addLast(namaBelakang, hobiBelakang,
      skillBelakang, umurBelakang);
                    break;
```

Script ini menggunakan "`switch`" untuk menangani pilihan pengguna. Jika pengguna memilih "`1`", program meminta *input* untuk menambahkan orang di depan daftar, mencakup nama, hobi, skill, dan umur, lalu memanggil metode "`addFirst`" untuk menambahkan orang tersebut. Jika pilihan yang diberikan adalah "`2`", program meminta input yang sama untuk menambahkan orang di belakang daftar, kemudian menggunakan metode "`addLast`". Keduanya memastikan bahwa data pengguna ditangani dengan baik untuk memperbarui daftar orang.

```
 case 3:
                    // Hapus orang dari depan
                    list.deleteFirst();
                    break;
                case 4:
                    // Hapus orang dari belakang
                    list.deleteLast();
                    break;
                case 5:
                    // Tampilkan daftar orang
                    list.displayList();
                    break;
```

```
                case 6:
                    // Keluar
                    lanjut = false;
                    break;
                default:
                    System.out.println("Pilihan tidak valid.");
            }
        }
        input.close();
    }
}
```

Script ini menangani pilihan tindakan pengguna untuk menghapus orang dari daftar atau menampilkan daftar yang ada. Jika pengguna memilih "`3`", program memanggil metode "`deleteFirst`" untuk menghapus orang dari depan daftar. Pilihan "`4`" memanggil metode "`deleteLast`" untuk menghapus orang dari belakang daftar. Pilihan "`5`" akan menampilkan seluruh daftar orang dengan memanggil metode "`displayList`". Jika pengguna memilih "`6`", variabel lanjut diatur ke *false*, yang menghentikan loop dan mengakhiri program. Jika pilihan yang dimasukkan tidak valid, program memberi tahu pengguna dengan pesan "`Pilihan tidak valid`".