

Nama : Roy Martin Silaban  
NPM : 233040130  
Kelas : IF-A  
Matkul : Praktikum Pemograman I

## PERTEMUAN 4

### Latihan 1

Latihan ini sudah dilakukan di pertemuan sebelumnya yaitu membuat kelas Node sebagai representasi dari elemen Node List. Berikut kode program kelas Node menggunakan bahasa Java.

```
package latihan;

public class Node {

    private int data;
    private Node next;

    // Constructor
    public Node(int data) {
        this.data = data;
        this.next = null;
    }

    // Getter untuk data
    public int getData() {
        return data;
    }

    // Setter untuk data
    public void setdata(int data) {
        this.data = data;
    }

    // Getter untuk next
    public Node getNext() {
        return next;
    }

    // Setter untuk next
    public void setNext(Node next) {
        this.next = next;
    }
}
```

#### Penjelasan Code ;

- **package latihan;** → Menyatakan bahwa kelas ini berada dalam paket **latihan**.
- **public class Node** → Mendefinisikan kelas **Node**, yang merepresentasikan satu elemen dalam linked list.
- **private int data;** → Menyimpan nilai dalam node.

- **private Node next;** → Menyimpan referensi ke node berikutnya dalam linked list.
- **public Node(int data)** → Konstruktor untuk membuat node baru dengan nilai tertentu.
- **this.data = data;** → Menginisialisasi atribut **data** dengan nilai yang diberikan.
- **this.next = null;** → Mengatur **next** menjadi **null**, karena awalnya belum terhubung ke node lain.
- **public int getData()** → Mengembalikan nilai **data** dari node.
- **public void setData(int data)** → Mengubah nilai **data** dalam node. (Catatan: Nama metode sebaiknya **setData**, bukan **setdata** agar konsisten dengan konvensi penamaan Java).
- **public Node getNext()** → Mengembalikan referensi ke node berikutnya dalam linked list.
- **public void setNext(Node next)** → Mengatur referensi ke node berikutnya dalam linked list.

## Latihan 2

Latihan ini akan memberikan implementasi operasi penambahan/sisipan elemen list di tengah/middle dengan notasi algoritma. Operasi ini direpresentasikan dengan fungsi **addMid** dengan parameter data yaitu node dan indeks yang akan ditambahkan ke List.

- Tambahkan fungsi dibawah ini di kelas **StrukturList**. Fungsi addMid di bawah dikonversi ke dalam bahasa pemrograman.

```

package latihan;

public class StrukturList {

    Node head;

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

    // Tambah elemen di akhir (tail)
    public void addTail(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            return;
        }
        Node temp = head;
        while (temp.getNext() != null) {
            temp = temp.getNext();
        }
        temp.setNext(newNode);
    }

    // Tambah elemen di awal (head)
    public void addHead(int data) {
        Node newNode = new Node(data);
        newNode.setNext(head);
        head = newNode;
    }

    // Tambah elemen di posisi tertentu (middle)
    public void addMid(int data, int position) {
        Node newNode = new Node(data);
        if (position == 1 || head == null) {
            addHead(data);
            return;
        }

        Node temp = head;
        for (int i = 1; temp != null && i < position - 1; i++) {
            temp = temp.getNext();
        }

        if (temp == null) {
            System.out.println("Posisi melebihi panjang list.");
            return;
        }

        newNode.setNext(temp.getNext());
        temp.setNext(newNode);
    }

    // Menampilkan isi linked list
    public void display() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.getData() + " ");
            temp = temp.getNext();
        }
        System.out.println();
    }
}

```

### Penjelasan Code ;

- **package latihan;** → Menyatakan bahwa kelas ini berada dalam paket **latihan**.
- **public class StrukturList** → Mendefinisikan kelas **StrukturList**, yang merepresentasikan struktur **Linked List**.
- **Node head;** → Menyimpan referensi ke elemen pertama (head) dalam linked list.

- **public StrukturList()** → Konstruktor untuk menginisialisasi linked list dengan **head = null**.

#### Metode dalam StrukturList

##### 1. **addTail(int data)** → Menambah elemen di akhir linked list

- **Node newNode = new Node(data);** → Membuat node baru dengan nilai **data**.
- **if (head == null) head = newNode;** → Jika list kosong, node baru menjadi head.
- **Node temp = head;** → Memulai traversal dari head.
- **while (temp.getNext() != null) temp = temp.getNext();** → Mencari node terakhir.
- **temp.setNext(newNode);** → Menambahkan node baru di akhir list.

##### 2. **addHead(int data)** → Menambah elemen di awal linked list

- **Node newNode = new Node(data);** → Membuat node baru dengan nilai **data**.
- **newNode.setNext(head);** → Node baru menunjuk ke head lama.
- **head = newNode;** → Memperbarui head ke node baru.

##### 3. **addMid(int data, int position)** → Menambah elemen di posisi tertentu dalam linked list

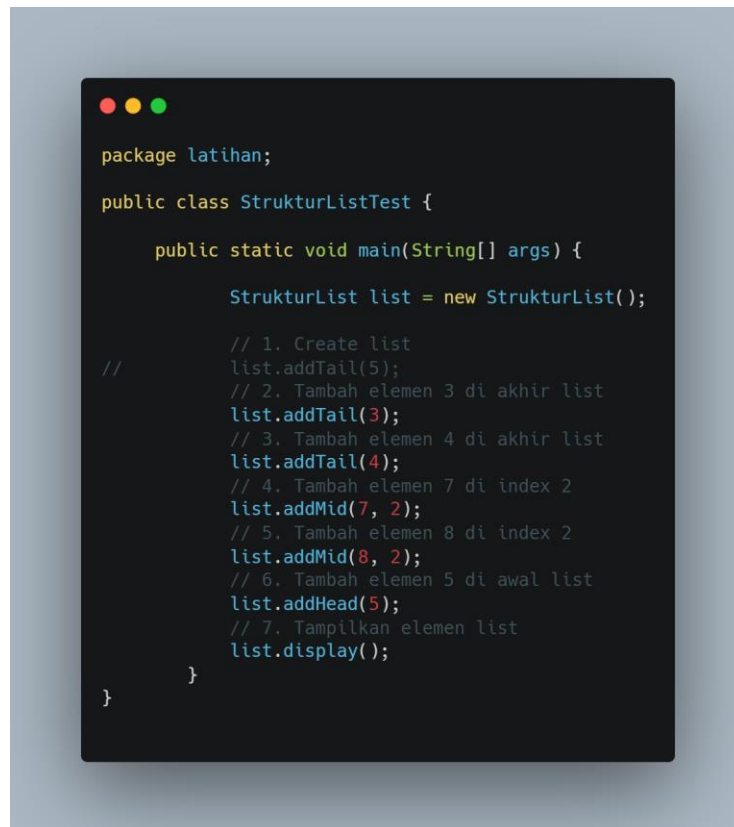
- **if (position == 1 || head == null) addHead(data);** → Jika posisi 1 atau list kosong, tambahkan di head.
- **Node temp = head;** → Memulai traversal dari head.
- **for (int i = 1; temp != null && i < position - 1; i++) temp = temp.getNext();** → Mencari posisi yang diinginkan.
- **if (temp == null)** → Jika posisi melebihi panjang list, tampilkan pesan error.
- **newNode.setNext(temp.getNext());** → Node baru menunjuk ke node setelahnya.
- **temp.setNext(newNode);** → Node sebelumnya menunjuk ke node baru.

##### 4. **display()** → Menampilkan isi linked list

- **Node temp = head;** → Memulai traversal dari head.
- **while (temp != null) { System.out.print(temp.getData() + " "); temp = temp.getNext(); }** → Cetak semua elemen.
- **System.out.println();** → Baris baru setelah pencetakan selesai.

## Latihan 3

Latihan ini akan memberikan penggunaan operasi penambahan elemen list (head, tail dan middle) dan kemudian menampilkan setiap elemen yang terdapat di list. Buatlah kelas **StrukturListTest** berikut fungsi main untuk mengeksekusi program. Konversikan urutan instruksi berikut di bawah ini ke fungsi tersebut!



```
package latihan;

public class StrukturListTest {

    public static void main(String[] args) {

        StrukturList list = new StrukturList();

        // 1. Create list
        list.addTail(5);
        // 2. Tambah elemen 3 di akhir list
        list.addTail(3);
        // 3. Tambah elemen 4 di akhir list
        list.addTail(4);
        // 4. Tambah elemen 7 di index 2
        list.addMid(7, 2);
        // 5. Tambah elemen 8 di index 2
        list.addMid(8, 2);
        // 6. Tambah elemen 5 di awal list
        list.addHead(5);
        // 7. Tampilkan elemen list
        list.display();
    }
}
```

### Penjelasan Code ;

- **package latihan;** → Menyatakan bahwa kelas ini berada dalam paket **latihan**.
- **public class StrukturListTest** → Mendefinisikan kelas **StrukturListTest** untuk menguji implementasi linked list.
- **public static void main(String[] args)** → Metode utama untuk menjalankan program.

### Langkah-langkah dalam Program

1. **StrukturList list = new StrukturList();** → Membuat objek **list** dari kelas **StrukturList**.
2. **list.addTail(3);** → Menambahkan elemen **3** di akhir linked list.
3. **list.addTail(4);** → Menambahkan elemen **4** di akhir linked list setelah **3**.
4. **list.addMid(7, 2);** → Menambahkan elemen **7** di posisi ke-2.
5. **list.addMid(8, 2);** → Menambahkan elemen **8** di posisi ke-2, sehingga **8** menggantikan posisi **7**.
6. **list.addHead(5);** → Menambahkan elemen **5** di awal linked list.

7. **list.display();** → Menampilkan semua elemen dalam linked list.

## Tugas

1. Buatlah Struktur list untuk menambahkan data /node di awal, menengah dan akhir dengan tipe data valuenya adalah bilangan pecahan!

```
package Tugas;

public class StrukturList {

    private Node head;

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

    // Tambah elemen di akhir (tail)
    public void addTail(double data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            return;
        }
        Node temp = head;
        while (temp.getNext() != null) {
            temp = temp.getNext();
        }
        temp.setNext(newNode);
    }

    // Tambah elemen di awal (head)
    public void addHead(double data) {
        Node newNode = new Node(data);
        newNode.setNext(head);
        head = newNode;
    }

    // Tambah elemen di posisi tertentu (middle)
    public void addMid(double data, int position) {
        Node newNode = new Node(data);
        if (position == 1 || head == null) {
            addHead(data);
            return;
        }

        Node temp = head;
        for (int i = 1; temp != null && i < position - 1; i++) {
            temp = temp.getNext();
        }

        if (temp == null) {
            System.out.println("Posisi melebihi panjang list.");
            return;
        }

        newNode.setNext(temp.getNext());
        temp.setNext(newNode);
    }

    // Menampilkan isi linked list
    public void display() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.getData() + " ");
            temp = temp.getNext();
        }
        System.out.println();
    }
}
```

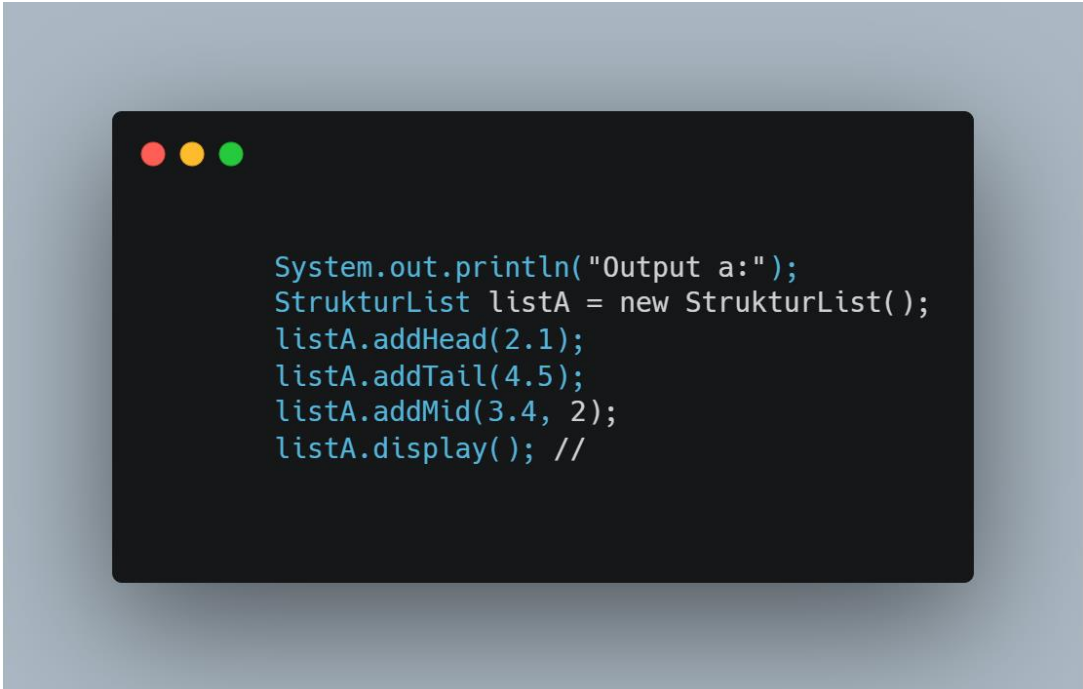
**Penjelasan Code:**

## Penjelasan Singkat Kode: StrukturList.java

1. **StrukturList()**  
→ Konstruktor untuk menginisialisasi linked list kosong dengan **head = null**.
2. **addTail(double data)**  
→ Menambahkan elemen di akhir linked list. Jika kosong, elemen pertama menjadi **head**.
3. **addHead(double data)**  
→ Menambahkan elemen di awal linked list dan menjadikannya **head** baru.
4. **addMid(double data, int position)**  
→ Menambahkan elemen di posisi tertentu dalam linked list. Jika posisi tidak valid, tampilkan pesan error.
5. **display()**  
→ Menampilkan semua elemen dalam linked list dari **head** ke **tail**.

2. Lakukan pengujian terhadap operasi tersebut seperti pada latihan 3 sehingga membentuk deret bilangan seperti dibawah ini:

- a. 2.1 3.4 4.5



```
System.out.println("Output a:");
StrukturList listA = new StrukturList();
listA.addHead(2.1);
listA.addTail(4.5);
listA.addMid(3.4, 2);
listA.display(); //
```


### Penjelasan:

1. **StrukturList listB = new StrukturList();**  
→ Membuat objek **listB** dari kelas **StrukturList**.
2. **listB.addHead(2.1);**  
→ Menambahkan elemen **2.1** di awal linked list.
3. **listB.addHead(3.4);**  
→ Menambahkan elemen **3.4** di awal, sehingga **3.4** menjadi kepala baru.



4. `listB.addTail(4.5);`  
→ Menambahkan elemen `4.5` di akhir linked list.
5. `listB.addTail(5.5);`  
→ Menambahkan elemen `5.5` di akhir setelah `4.5`.
6. `listB.addMid(1.1, 3);`  
→ Menambahkan elemen `1.1` di posisi ke-3 dalam linked list.
7. `listB.display();`  
→ Menampilkan semua elemen dalam linked list.

b. 3.4 2.1 1.1 4.5 5.5



```
System.out.println("\nOutput b:");
StrukturList listB = new StrukturList();
listB.addHead(2.1);
listB.addHead(3.4);
listB.addTail(4.5);
listB.addTail(5.5);
listB.addMid(1.1, 3);
listB.display();
```

#### Penjelasan Code:

1. `StrukturList listB = new StrukturList();`  
→ Membuat objek `listB` dari kelas `StrukturList`.
2. `listB.addHead(2.1);`  
→ Menambahkan elemen `2.1` di awal linked list.
3. `listB.addHead(3.4);`  
→ Menambahkan elemen `3.4` di awal, sehingga `3.4` menjadi kepala baru.
4. `listB.addTail(4.5);`  
→ Menambahkan elemen `4.5` di akhir linked list.
5. `listB.addTail(5.5);`  
→ Menambahkan elemen `5.5` di akhir setelah `4.5`.

6. `listB.addMid(1.1, 3);`

→ Menambahkan elemen `1.1` di posisi ke-3 dalam linked list.

7. `listB.display();`

→ Menampilkan semua elemen dalam linked list.

3. Link GitHub