

**LAPORAN PRAKTIKUM PEMOGRAMAN I**  
**PERTEMUAN 3**



Disusun oleh :

Ghani Aliyandi

C 233040049

**JURUSAN INFORMATIKA FAKULTAS**  
**TEKNIK**  
**UNIVERSITAS PASUNDAN**  
**BANDUNG**  
**2025**

## Latihan 1

NamaKelas : Node

Kode :

```
package Pertemuan3;

public class Node {
    private int data;
    private Node next;

    // Konstruktor
    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 :

Class **Node** ini merupakan dasar untuk membuat **Singly Linked List**. Dengan class ini, kita bisa:

- Menyimpan data dalam setiap node.
- Menghubungkan satu node ke node berikutnya menggunakan next.
- Menggunakan metode **getter dan setter** untuk mengakses atau mengubah nilai data dan referensi.

## Latihan 2

NamaKelas : StrukturList

Kode :

```
package Pertemuan3;

public class StrukturList {
    private Node HEAD;

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

    // Fungsi untuk mengecek apakah list kosong
    public boolean isEmpty() {
        return HEAD == null;
    }

    // Fungsi untuk menambahkan elemen di akhir list (addTail)
    public void addTail(int data) {
        Node posNode=null, curNode=null;
        Node newNode = new Node(data);
        if (isEmpty()) {
            HEAD = newNode;
        }
        else
        {
            curNode = HEAD;
            while (curNode != null) {
                posNode = curNode;
                curNode = curNode.getNext();
            }
            posNode.setNext(newNode);
        }
    }
}
```

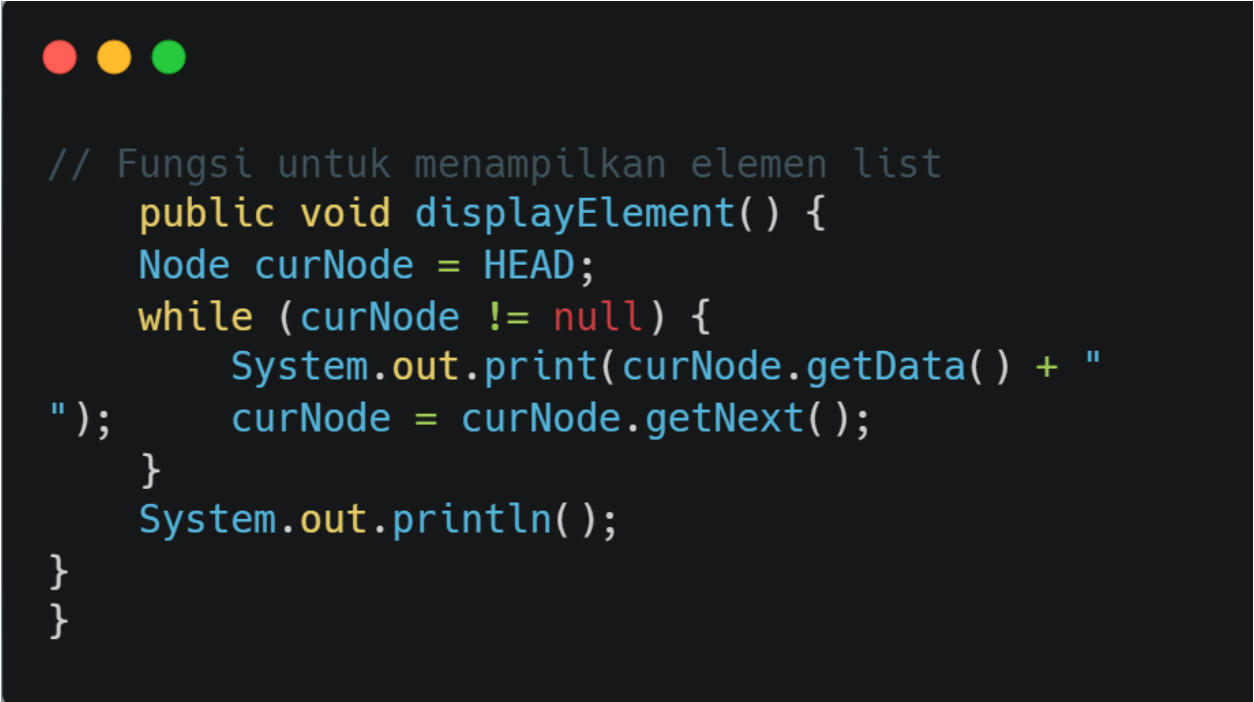
### Penjelasan :

- Konstruktor: Menginisialisasi HEAD dengan null, yang berarti daftar masih kosong.
- isEmpty(): Mengecek apakah daftar kosong dengan memeriksa apakah HEAD bernilai null.
- addTail(int data): Menambahkan node baru di akhir list.
- Jika list kosong, node baru langsung menjadi HEAD.
- Jika tidak, program mencari node terakhir dan menghubungkannya dengan node baru menggunakan setNext().

### Latihan 3

NamaKelas : StrukturList

Kode :



```
// Fungsi untuk menampilkan elemen list
public void displayElement() {
    Node curNode = HEAD;
    while (curNode != null) {
        System.out.print(curNode.getData() + "
");
        curNode = curNode.getNext();
    }
    System.out.println();
}
}
```


#### Penjelasan :

- curNode mulai dari HEAD (node pertama).
- Menggunakan while loop, setiap data dari node dicetak ke layar.
- curNode diperbarui ke node berikutnya hingga mencapai null (akhir list).
- Setelah semua elemen dicetak, System.out.println(); digunakan untuk pindah ke baris baru.

## Latihan 4

NamaKelas : ListMain

Kode :



```
// Tes-2 (Output: 3 4 5)
    StrukturList list3 = new
StrukturList();
    list3.addTail(3);
    list3.addTail(4);
    list3.addTail(5);
    System.out.print("Tes-2 (a): ");
    list3.displayElement();
```

### Penjelasan :

- Membuat objek list dari class StrukturList.
- Menambahkan tiga elemen (3, 4, 5) ke dalam linked list menggunakan addTail(), sehingga elemen disusun secara berurutan.
- Mencetak teks "Elemen: " ke layar.
- Memanggil metode display() untuk menampilkan isi linked list, yaitu 3 → 4 → 5.

## Tes 1

NamaKelas : ListMain

Kode :

```
package Pertemuan3;

public class ListMain {
    public static void main(String[] args) {
        // Tes-1 (Output: 3 2 1)
        StrukturList list1 = new
StrukturList();
        list1.addHead(1);
        list1.addHead(2);
        list1.addHead(3);
        System.out.print("Tes-1 (a): ");
        list1.displayElement();

        // Tes-1 (Output: 1 4 5 7)
        StrukturList list2 = new
StrukturList();
        list2.addTail(1);
        list2.addTail(4);
        list2.addTail(5);
        list2.addTail(7);
        System.out.print("Tes-1 (b): ");
        list2.displayElement();
    }
}
```

Penjelasan :

Tes-1 (a)

- Menambahkan elemen (**1, 2, 3**) di **head** menggunakan addHead().
- Karena setiap elemen baru dimasukkan di depan, hasil akhir adalah **3 → 2 → 1**.

Tes-1 (b)

- Menambahkan elemen (**1, 4, 5, 7**) di **tail** menggunakan addTail().
- Elemen ditambahkan di belakang secara berurutan, sehingga hasilnya **1 → 4 → 5 → 7**.

## Latihan 5

NamaKelas : StrukturList

Kode :

```
// Fungsi untuk menambahkan elemen di awal list (addHead)
public void addHead(int data) {
    Node newNode = new Node(data);
    if (isEmpty()) {
        HEAD = newNode;
    } else {
        newNode.setNext(HEAD);
        HEAD = newNode;
    }
}
```

### Penjelasan :

Metode addHead(int data) menambahkan elemen baru di awal linked list. Jika list kosong, node baru langsung menjadi HEAD. Jika tidak, node baru akan menunjuk ke HEAD lama, lalu HEAD diperbarui ke node baru. Dengan cara ini, elemen terbaru selalu berada di depan list.

## Tes 2

NamaKelas : ListMain

Kode :

```
// Tes-2 (Output: 3 4 5)
        StrukturList list4 = new
StrukturList4);addHead(5);
        list4.addHead(4);
        list4.addHead(3);
        System.out.print("Tes-2 (b): ");
        list4.displayElement();
```

### Penjelasan :

Kode ini menguji metode addHead() dengan menambahkan elemen (5, 4, 3) ke awal linked list. Karena addHead() menambahkan elemen di depan, urutan penyimpanan menjadi 3 → 4 → 5.

Metode displayElemen() kemudian mencetak daftar ini ke layar sebagai output

### Tes 3

NamaKelas : StrukturList

Kode :

```
// Tes-3 (Output: 3 2 1 dan 1 4 5 7)
    StrukturList list5 = new
StrukturList();
    list5.addHead(1);
    list5.addHead(2);
    list5.addHead(3);
    System.out.print("Tes-3 (a): ");
    list5.displayElement();

    StrukturList list6 = new
StrukturList();
    list6.addTail(1);
    list6.addTail(4);
    list6.addTail(5);
    list6.addTail(7);
    System.out.print("Tes-3 (b): ");
    list6.displayElement();
}
```

### Penjelasan :

Tes-3 (a):

- Elemen 1, 2, 3 ditambahkan ke awal menggunakan addHead(), sehingga urutan akhirnya menjadi 3 → 2 → 1.

Tes-3 (b):

- Elemen 1, 4, 5, 7 ditambahkan ke akhir menggunakan addTail(), sehingga urutan akhirnya menjadi 1 → 4 → 5 → 7.