Nama: Arya Raihan Hanif
Kelas: Praktikum Pemrograman C
NPM: 233040101
Link Repo: https://github.com/aryahanif/PP12025_C_233040101

List (Operasi Add Middle)

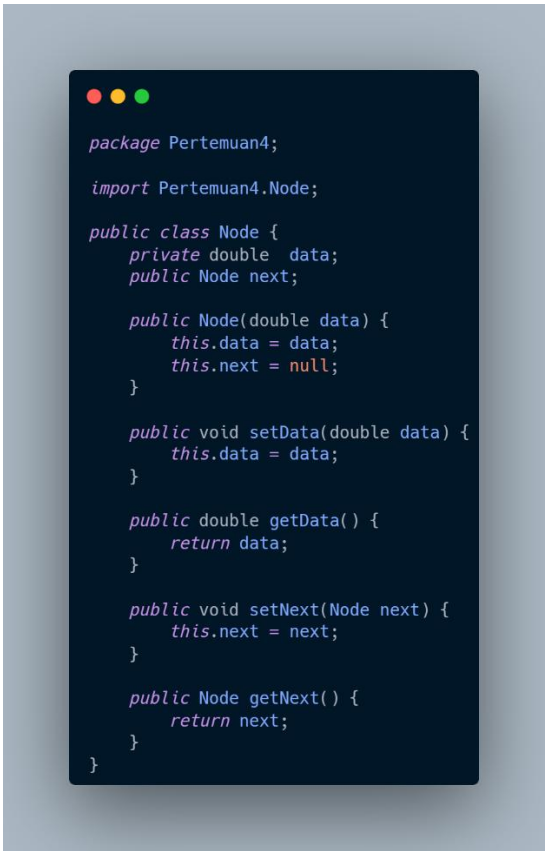
Latihan 1

Latihan ini sudah dilakukan dipertemuan sebelumnya yaitu membuatkan kelas **Node** sebagai representasi dari elemen **Node** List. Berikut kode program kelas **Node** menggunakan Bahasa Java

Pseudocode	Bahasa Pemrograman

`class Node {`
`  data: tipe data (T)`
`  next: Node`
`}`
	`public class Node {`
	`  private int data;`
	`  private Node next;`
	
	`  /** Inisialisasi atribut node */`
	
	`  public Node(int data) {`
	`  this.data = data;`
	`  }`
	
	`  /** Setter & Getter */`
	
	`}`

Jawab:



```java
package Pertemuan4;

import Pertemuan4.Node;

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

    public Node(double data) {
        this.data = data;
        this.next = null;
    }

    public void setData(double data) {
        this.data = data;
    }

    public double getData() {
        return data;
    }

    public void setNext(Node next) {
        this.next = next;
    }

    public Node getNext() {
        return next;
    }
}
```

Penjelasan Kode:

Kelas **Node** merupakan bagian dari paket **Pertemuan4** dan digunakan untuk merepresentasikan sebuah node dalam struktur linked list. Setiap node memiliki nilai bertipe double dan referensi ke node berikutnya dalam daftar.

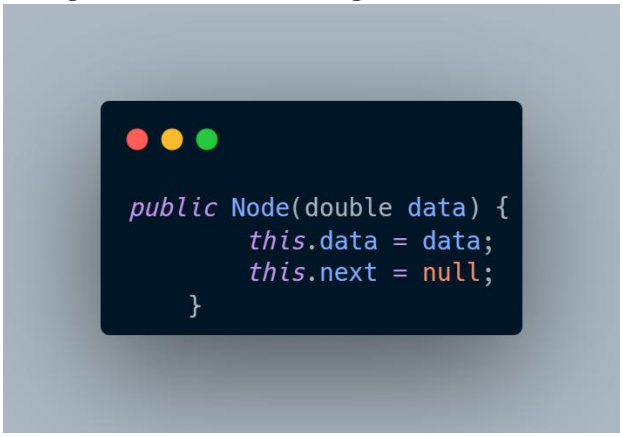
Struktur Kelas

Kelas ini memiliki dua atribut utama:

1. data (double) – menyimpan nilai dalam node.
2. next (Node) – menyimpan referensi ke node berikutnya.

Konstruktor

Kelas ini memiliki konstruktor yang menerima satu parameter data bertipe double dan menginisialisasi next dengan null.



```java
public Node(double data) {
        this.data = data;
        this.next = null;
    }
```

Metode Utama

1. **Setter dan Getter untuk data**
   - setData(double data): Mengubah nilai dalam node.
   - getData(): Mengembalikan nilai dalam node.
2. **Setter dan Getter untuk next**
   - setNext(Node next): Mengatur referensi ke node berikutnya.
   - getNext(): Mengembalikan referensi ke node berikutnya.

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

Algoritma Fungsi addMid

```
procedure addMid(data: integer, position: integer)
deklarasi
  posNode, curNode: Node {current node}
  i: integer
deskripsi
  newNode ← new Node(data)
  IF (HEAD = null) THEN
    HEAD ← newNode
  ELSE
    curNode ← HEAD;
    IF (position = 1) THEN        {tambah di awal}
      newNode.next ← curNode
      HEAD ← newNode
      {slide berikutnya}
```

```
  ELSE
    i ← 1
    WHILE(curNode <> null AND i < position) DO
      posNode ← curNode
      curNode ← curNode.next
      i++
    ENDWHILE
    posNode.next ← newNode
    newNode.next ← curNode
  ENDIF
ENDIF
```

Jawab:

```java
package Pertemuan4;

public class StrukturList {
    private Node HEAD;

    public boolean isEmpty() {
        return HEAD == null;
    }

    public void addHead(double data) {
        Node newNode = new Node(data);
        newNode.setNext(HEAD);
        HEAD = newNode;
    }

    public void addMid(double data, int position) {
        Node newNode = new Node(data);

        if (HEAD == null || position == 1) {
            addHead(data);
            return;
        }

        Node curNode = HEAD;
        int i = 1;

        while (curNode.next != null && i < position - 1) {
            curNode = curNode.next;
            i++;
        }

        newNode.next = curNode.next;
        curNode.next = newNode;
    }

    public void addTail(double data) {
        Node newNode = new Node(data);

        if (isEmpty()) {
            HEAD = newNode;
            return;
        }

        Node curNode = HEAD;
        while (curNode.next != null) {
            curNode = curNode.next;
        }

        curNode.next = newNode;
    }

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

Penjelasan Kode:

Kelas **StrukturList** merupakan bagian dari paket **Pertemuan4** dan digunakan untuk mengelola struktur linked list secara dinamis. Kelas ini memungkinkan manipulasi elemen dalam daftar melalui berbagai metode seperti menambahkan di awal, tengah, dan akhir, serta menampilkan elemen.

Atribut Utama

1. **HEAD** (Node) – Menunjuk ke elemen pertama dalam linked list.

Metode Utama

1. isEmpty()
- Mengembalikan true jika linked list kosong, false jika tidak.

2. addHead(double data)
- Menambahkan node baru di awal linked list.
- Node baru akan menjadi HEAD, sementara HEAD lama menjadi node berikutnya.

3. addMid(double data, int position)
- Menambahkan node baru di posisi tertentu dalam linked list.
- Jika posisi adalah 1 atau linked list kosong, node baru akan ditambahkan di awal menggunakan addHead().
- Jika posisi lebih besar, node akan disisipkan di antara dua node yang ada.

4. addTail(double data)
- Menambahkan node baru di akhir linked list.
- Jika linked list kosong, node baru akan menjadi HEAD.
- Jika tidak, node baru akan ditambahkan setelah node terakhir.

5. displayElement()
- Menampilkan semua elemen dalam linked list secara berurutan.
- Mencetak nilai dari setiap node diikuti dengan spasi, kemudian baris baru.

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!

Urutan Instruksi	Output
1. Create list dengan keyword new 2. Tambah elemen 3 di akhir list 3. Tambah elemen 4 di akhir list 4. Tambah elemen 7 di index 2 5. Tambah elemen 8 di index 2 6. Tambah elemen 5 di awal list 7. Tampilkan elemen list	5 3 8 7 4

Tugas

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

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

a. 2.1 3.4 4.5

b. 3.4 2.1 1.1 4.5 5.5

Jawab:

```java
package Pertemuan4;

public class StrukturListTest {
    public static void main(String[] args) {
        StrukturList list = new StrukturList();

        // Pengujian untuk menghasilkan deret 5 3 8 7 4
        list.addTail(3);
        list.addTail(4);
        list.addMid(7,2);
        list.addMid(8,2);
        list.addHead(5);
        System.out.println("Output untuk deret Latihan 3:");
        list.displayElement(); // Output: 5.0 3.0 8.0 7.0 4.0

        // Pengujian untuk menghasilkan deret (a) 2.1 3.4 4.5
        list = new StrukturList(); // Reset list
        list.addTail(4.5);
        list.addMid(3.4, 1);
        list.addHead(2.1);
        System.out.println("Output untuk deret Tugas a:");
        list.displayElement(); // Output: 2.1 3.4 4.5

        // Pengujian untuk menghasilkan deret (b) 3.4 2.1 1.1 4.5 5.5
        list = new StrukturList(); // Reset list

        list.addTail(4.5);
        list.addTail(5.5);
        list.addMid(1.1, 1);
        list.addMid(2.1, 1);
        list.addHead(3.4);
        System.out.println("Output untuk deret Tugas B:");
        list.displayElement(); // Output: 3.4 2.1 1.1 4.5 5.5
    }
}
```

Penjelasan Kode:

Kelas **StrukturListTest** digunakan untuk menguji implementasi StrukturList dengan berbagai skenario penambahan elemen ke dalam linked list.

Uji Coba

1. Pengujian Deret 5 3 8 7 4

Langkah-langkah:

- Menambahkan 3 di akhir.
- Menambahkan 4 di akhir.
- Menambahkan 7 di posisi ke-2.
- Menambahkan 8 di posisi ke-2.
- Menambahkan 5 di awal.

Hasil yang diharapkan:

```
Output untuk deret Latihan 3:
5.0 3.0 8.0 7.0 4.0
```

2. Pengujian Deret (a) 2.1 3.4 4.5

Langkah-langkah:

- Menambahkan 4.5 di akhir.
- Menambahkan 3.4 di posisi ke-1.
- Menambahkan 2.1 di awal.

Hasil yang diharapkan:

```
Output untuk deret Tugas a:
2.1 3.4 4.5
```

3. Pengujian Deret (b) 3.4 2.1 1.1 4.5 5.5

Langkah-langkah:

- Menambahkan 4.5 di akhir.
- Menambahkan 5.5 di akhir.
- Menambahkan 1.1 di posisi ke-1.
- Menambahkan 2.1 di posisi ke-1.
- Menambahkan 3.4 di awal.

Hasil yang diharapkan:

```
Output untuk deret Tugas B:
3.4 2.1 1.1 4.5 5.5
```

Output untuk keseluruhan kode:

```
Output untuk deret Latihan 3:
5.0 3.0 8.0 7.0 4.0
Output untuk deret Tugas a:
2.1 3.4 4.5
Output untuk deret Tugas B:
3.4 2.1 1.1 4.5 5.5
```