NAMA: Fahri Rizqon Arsiansyah

NRP: 233040062

KELAS: A

MATA KULIAH: Praktikum Pemrograman I

LATIHAN

1. Latihan 1

Latihan ini akan memberikan implementasi pembuatan elemen list. Elemen list direpresentasikan dengan Node. Sebuah node terdiri dari atribut data/nilai dan atribut *next*. Atribut *next* akan menunjuk ke node yang lain.

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 */`
	
	`}`

a. Contoh setter: `public void setNama(String nama) { this.nama = nama; }`
b. Contoh getter: `public String getNama() { return nama; }`

```java
package PERTEMUAN4.latihan;

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

  // Inisialisasi atribut node

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

  }

  // Setter & Getter
  public void setnilai(int data) {
    this.data = data;
  }

  public int getdata() {
    return data;
  }

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

  public Node getNext() {
    return next;
  }

}
```

Kode ini adalah implementasi kelas Node dalam Java yang digunakan dalam Linked List. Setiap Node menyimpan sebuah angka (data) dan referensi ke Node berikutnya (next). Saat pertama kali dibuat, next diset ke null, menandakan bahwa node ini berdiri sendiri. Kelas ini juga menyediakan metode untuk mengubah dan mengambil nilai data, serta mengatur dan mendapatkan referensi ke Node berikutnya. Intinya, kelas ini adalah blok penyusun dasar dari Linked List, memungkinkan penyimpanan data yang bisa bertambah secara dinamis tanpa batasan ukuran tetap seperti array.

2. Latihan 2

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}
```

AS@IF-UNPAS

```
  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
```

```java
// ADD MID
public void addMid(int data, int position) {
  Node posNode = null, curNode = null;
  Node newNode = new Node(data);
  if (HEAD == null) {
    HEAD = newNode;
    return;
  } else {
    curNode = HEAD;
    if (position == 1) {
      newNode.setNext(HEAD);
      HEAD = newNode;

    } else {
      int i = 1;
      while (curNode != null && i < position) {
        posNode = curNode;
        curNode = curNode.getNext();
        i++;
      }
    }
    posNode.setNext(newNode);
    newNode.setNext(curNode);
  }
}

}
```

Metode addMid(int data, int position) digunakan untuk menambahkan node baru pada posisi tertentu dalam Linked List. Pertama, node baru dibuat dengan nilai data. Jika Linked List masih kosong, node baru langsung dijadikan HEAD. Jika tidak, maka ada dua kemungkinan, yaitu jika posisi yang diminta adalah 1, maka node baru akan dimasukkan di awal dengan cara menghubungkannya ke HEAD saat ini, lalu HEAD diperbarui agar menunjuk ke node baru. Jika posisi lebih dari 1, metode ini akan menelusuri Linked List menggunakan curNode hingga mencapai posisi yang diinginkan, dengan posNode menyimpan node sebelum posisi tersebut. Setelah sampai, posNode akan dihubungkan ke node baru, dan node baru akan dihubungkan ke curNode, sehingga node baru masuk di tengah tanpa mengganggu urutan yang ada. Dengan cara ini, metode ini memungkinkan penambahan node di mana saja dalam Linked List, baik di awal, tengah, maupun sebelum node terakhir.

3. Latihan 3

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

```java
package PERTEMUAN4.latihan;

public class StrukturListTes {
    Run | Debug
    public static void main(String[] args) {
        StrukturList list = new StrukturList();
        list.addTail(data:3);
        list.addTail(data:4);
        list.addMid(data:7, position:2);
        list.addMid(data:8, position:2);
        list.addHead(data:5);

        list.displayElement();

    }

}
```

Output:

```
53874
PS C:\PP12025_A_233040062>
```

Kode ini membuat dan menguji Linked List menggunakan kelas StrukturList. Elemen-elemen ditambahkan ke dalam list, baik di awal, tengah, maupun akhir, lalu isi list ditampilkan. Tujuannya adalah memastikan bahwa semua operasi penambahan dan tampilan data berfungsi dengan benar.

Tugas

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

1.

```java
package PERTEMUAN4.Tugas;

public class StrukturList {
    public Node HEAD;

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

    // ADD TAIL
    public void addTail(double 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);
      }
    }

    // display
    public void displayElement() {
      Node curNode = HEAD;
      while (curNode != null) {
        System.out.println(curNode.getdata() + "");
        curNode = curNode.getNext();
      }
    }

    // ADD HEAD
    public void addHead(double data) {
      Node newNode = new Node(data);
      if (isEmpty()) {
        HEAD = newNode;
      } else {
        newNode.setNext(HEAD);
        HEAD = newNode;
      }
    }
```

```java
    // ADD MID
    public void addMid(double data, int position) {
      Node posNode = null, curNode = null;
      Node newNode = new Node(data);
      if (HEAD == null) {
        HEAD = newNode;
        return;
      } else {
        curNode = HEAD;
        if (position == 1) {
          newNode.setNext(HEAD);
          HEAD = newNode;
          return;

        } else {
          int i = 1;
          while (curNode != null && i < position) {
            posNode = curNode;
            curNode = curNode.getNext();
            i++;
          }
        }
        // if (posNode != null) {
        posNode.setNext(newNode);
        newNode.setNext(curNode);
        // }

      }
    }

}
```

Kode ini mengimplementasikan struktur Linked List sederhana dalam Java melalui kelas StrukturList. Linked list ini menggunakan node yang berisi nilai bertipe double dan referensi ke node berikutnya. Kelas ini memiliki beberapa metode utama:

- isEmpty() – Mengecek apakah linked list kosong dengan memeriksa apakah HEAD bernilai null.
- addHead(double data) – Menambahkan elemen baru di awal linked list dengan menjadikannya HEAD dan menghubungkannya ke node sebelumnya.
- addTail(double data) – Menambahkan elemen di akhir linked list dengan menelusuri sampai node terakhir, lalu menautkan node baru di ujungnya.
- addMid(double data, int position) – Menyisipkan node baru di posisi tertentu dengan menavigasi hingga lokasi yang diinginkan, lalu menyesuaikan referensi antar-node agar tetap terhubung dengan benar.
- displayElement() – Menampilkan semua elemen dalam linked list dengan menelusuri setiap node dari HEAD hingga node terakhir.

Kelas ini memungkinkan manipulasi linked list secara dinamis, termasuk penambahan elemen di awal, tengah, dan akhir, serta menampilkan isi list dengan mudah.

2. a.

```
package PERTEMUAN4.Tugas;

public class StrukturListTes {
    Run | Debug
    public static void main(String[] args) {
      StrukturList list = new StrukturList();
      //list.addTail(2.1);
      //list.addTail(5.5);
      //list.addMid(4.5, 2);
      //list.addMid(1.1, 2);
      //list.addHead(3.4);

      //list.displayElement();

       list.addTail(data:3.4);
       list.addMid(data:4.5, position:2);
       list.addHead(data:2.1);

       list.displayElement();
    }
  }
```

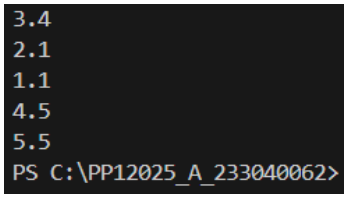
Output:

```
2.1
3.4
4.5
PS C:\PP12025_A_233040062>
```

b.

```
package PERTEMUAN4.Tugas;

public class StrukturListTes {
    Run | Debug
    public static void main(String[] args) {
      StrukturList list = new StrukturList();
      list.addTail(data:2.1);
      list.addTail(data:5.5);
      list.addMid(data:4.5, position:2);
      list.addMid(data:1.1, position:2);
      list.addHead(data:3.4);

      list.displayElement();

      //list.addTail(3.4);
      //list.addMid(4.5, 2);
      //list.addHead(2.1);

      //list.displayElement();
    }

  }
```

Output:



Kode ini bertujuan untuk mengujikan metode dalam StrukturList, yaitu menambahkan elemen di awal (addHead), di tengah (addMid), dan di akhir (addTail), serta menampilkan hasilnya menggunakan displayElement(). Dengan urutan eksekusi yang diberikan, hasil akhir yang ditampilkan adalah angka yang tersusun dalam linked list sesuai dengan operasi yang dilakukan.