PRAKTIKUM 19
DOUBLE LINKED LIST 2

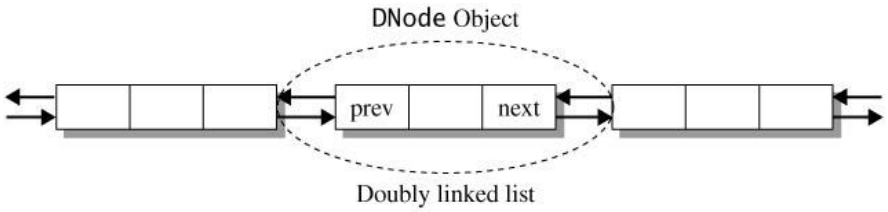
A. TUJUAN PEMBELAJARAN

1. Memahami konsep Double Linked List.
2. Mengetahui cara membuat sebuah Node pada Double Linked List.
3. Mampu membuat Double Linked List sendiri.
4. Memahami operasi-operasi yang terdapat pada Double Linked List.

B. DASAR TEORI

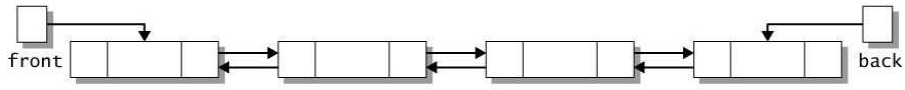
Double Linked List

Double Linked List terdiri dari tiga bagian yaitu untuk menyimpan nilai dan dua reference yang menunjuk ke node selanjutnya (**next** node) dan node sebelumnya (**previous** node). Untuk bergerak maju dan mundur pada double linked list menggunakan link **next** dan **prev** pada node.



Gambar 19.1 Representasi Sebuah Node di Double Linked List

Double Linked List mempunyai reference **front** untuk menandai awal node dan reference **back** untuk menandai akhir list



Gambar 19.2 Double Linked List

Pembacaan pada Double Linked List

Double Linked List dapat dibaca melalui dua arah.

- Pembacaan maju (forward scan) yaitu membaca double linked list dimulai dari reference **front** dan berakhir pada reference **back**.
- Pembacaan mundur (backward scan) yaitu membaca double linked list dimulai dari reference **back** dan berakhir pada reference **front**.

Representasi Node

Node pada Double Linked List direpresentasikan dengan class Dnode. Kumpulan object DNode membentuk sebuah list disebut double linked list. Object DNode mempunyai tiga variabel:

- nodeValue untuk menyimpan nilai
- prev untuk menandai node sebelumnya
- Next untuk menandai node sesudahnya.

Class mempunyai dua constructor.

- Default constructor

  membuat object DNode dengan nodeValue bernilai null, sedangkan prev dan next diset dengan nilai this (link yang menunjuk ke dirinya sendiri) .
- Constructor dengan argumen

  untuk memberikan nilai pada nodeValue, sedangkan untuk variabel prev dan next diset dengan nilai this.

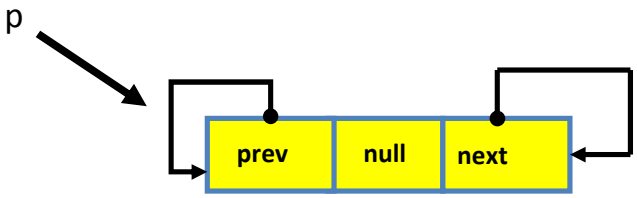
```
public class DNode<T>
{
   public T nodeValue;    // data value of the node
   public DNode<T> prev;  // previous node in the list
   public DNode<T> next;  // next node in the list

   // default constructor; creates an object with
   // the value set to null and whose references
   // point to the node itself
   public DNode()
   {
      nodeValue = null;
      // the next node is the current node
      next = this;
      // the previous node is the current node
      prev = this;
   }
// creates object whose value is item and
```

```
   // whose references point to the node itself
   public DNode(T item)
   {
      nodeValue = item;
      // the next node is the current node
      next = this;
      // the previous node is the current node
      prev = this;
   }
}
```

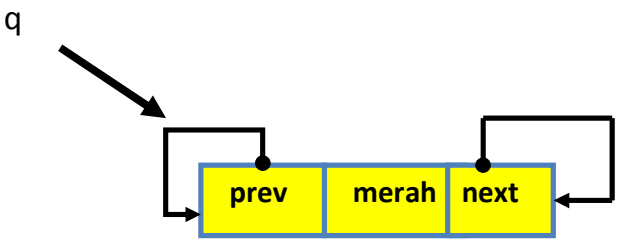
Membuat Node p

DNode<String> p=new DNode<String>();



Gambar 19.3 Membuat Node p

DNode<String> q=new DNode<String>(“merah”);



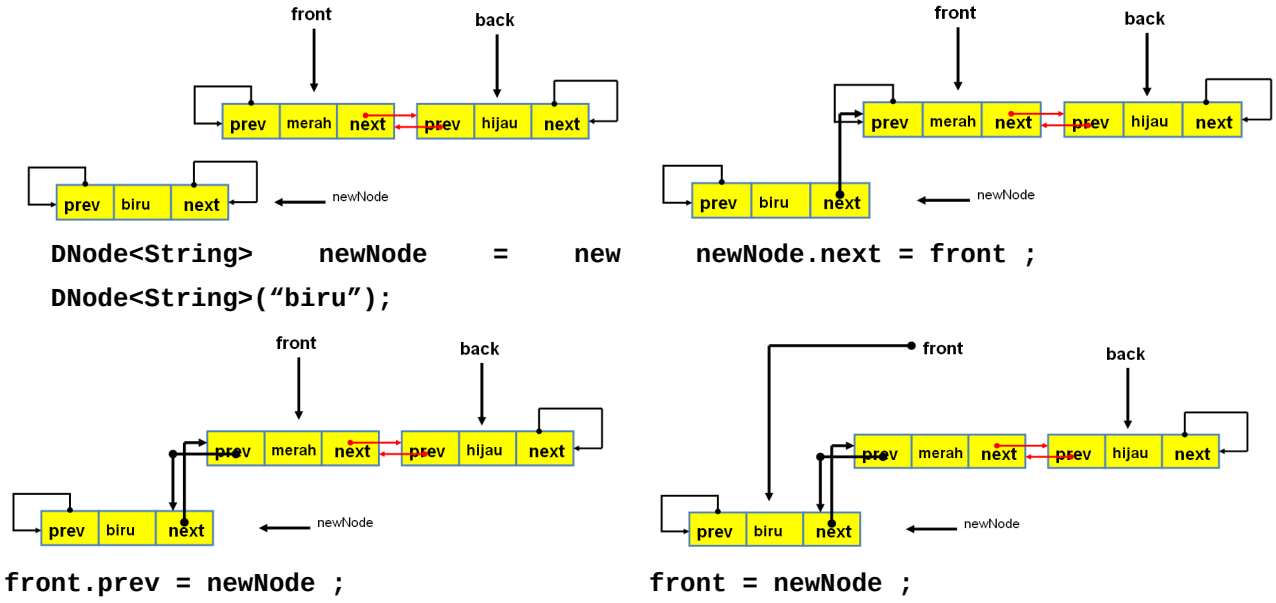
Gambar 19.4 Membuat Node q

Operasi pada Double Linked List

1. Menyisipkan Node

Menyisipkan Node di Depan List

Menyisipkan Node di Depan List, perlu memindahkan variabel reference front menunjuk ke Node newNode.



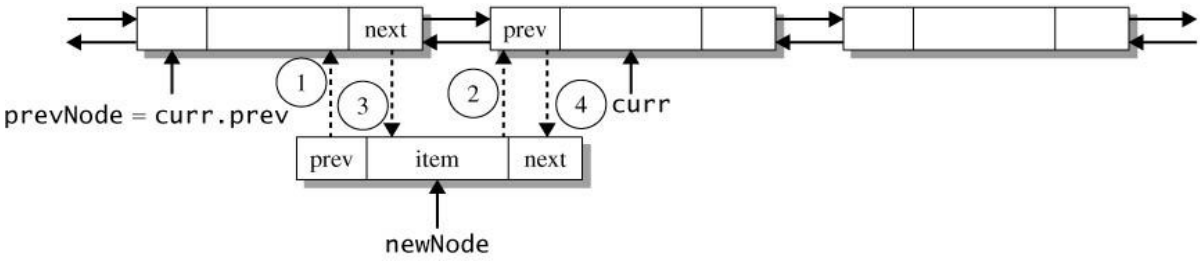
Gambar 19.5 Menambahkan Node di depan Double Linked List

Menyisipkan Node di Double Linked List

Untuk menyisipkan Node diperlukan dua variabel reference yaitu:

- curr : menandai node saat ini
- prevNode : menandai node sebelum curr

Menyisipkan node dilakukan sebelum curr dan sesudah prevNode.



Gambar 19.6 Menyisipkan Node Sebelum Node Target

```
// declare the DNode reference variables newNode and prevNode
DNode<T> newNode, prevNode;
// create a new node and assign prevNode to reference the
// predecessor of curr
newNode = new DNode<T>(item);
```

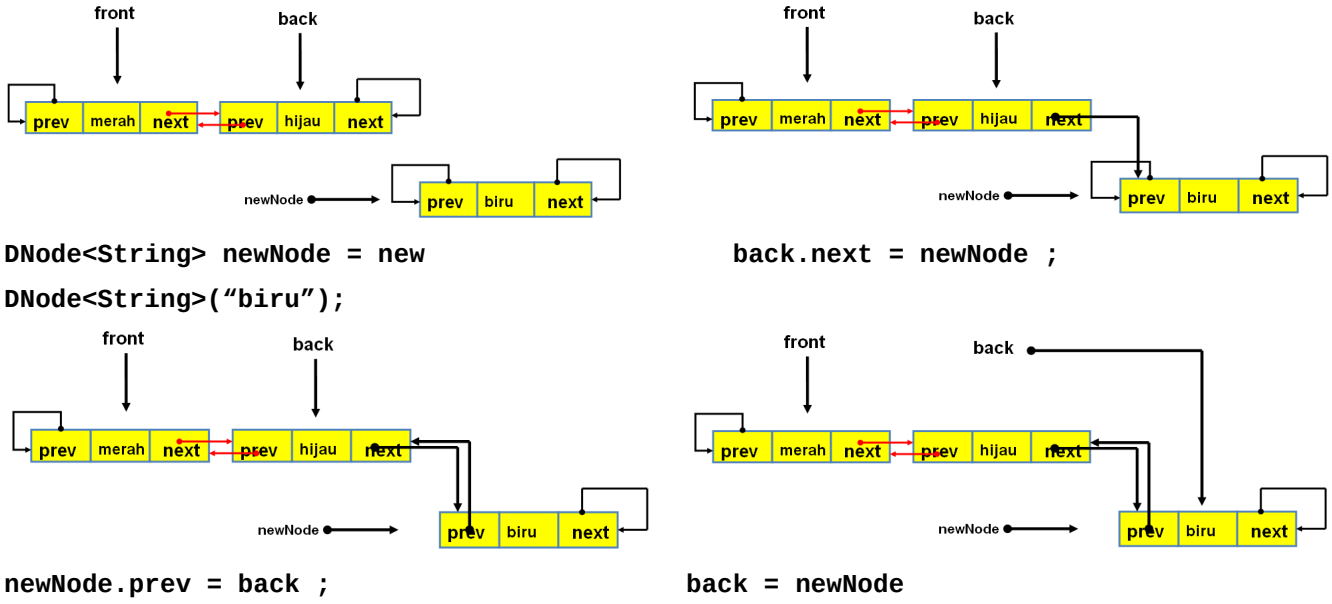
```
prevNode = curr.prev;

// update reference fields in newNode
newNode.prev = prevNode;      // statement 1
newNode.next = curr;          // statement 2

// update curr and its predecessor to point at newNode
prevNode.next = newNode;      // statement 3
curr.prev = newNode;          // statement 4
```

Menyisipkan Node di Belakang List

Menyisipkan Node di Belakang List, perlu memindahkan variabel reference back menunjuk ke Node newNode.



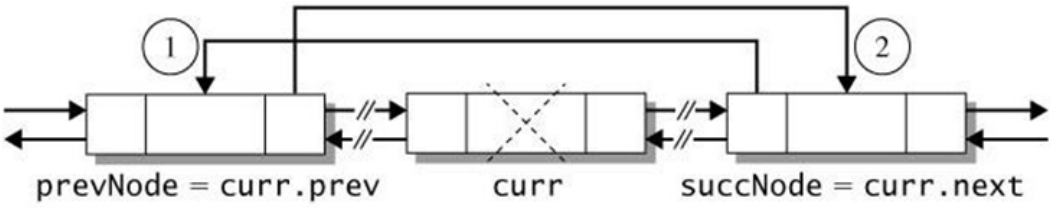
Gambar 19.7 Menyisipkan Node di belakang Double Linked List

2. Menghapus Node Sesuai Target.

Untuk menghapus Node diperlukan dua variabel reference yaitu:

- curr : menandai node yang akan di hapus
- prevNode : menandai node sebelum curr

Menghapus node dilakukan di curr.



Gambar 19.8 Menghapus Node Sesuai Target

```
DNode<T> prevNode = curr.prev, nextNode = curr.next;
// update the reference variables in the adjacent nodes.
prevNode.next = nextNode;     // statement 1
nextNode.prev = prevNode;     // statement 2
```

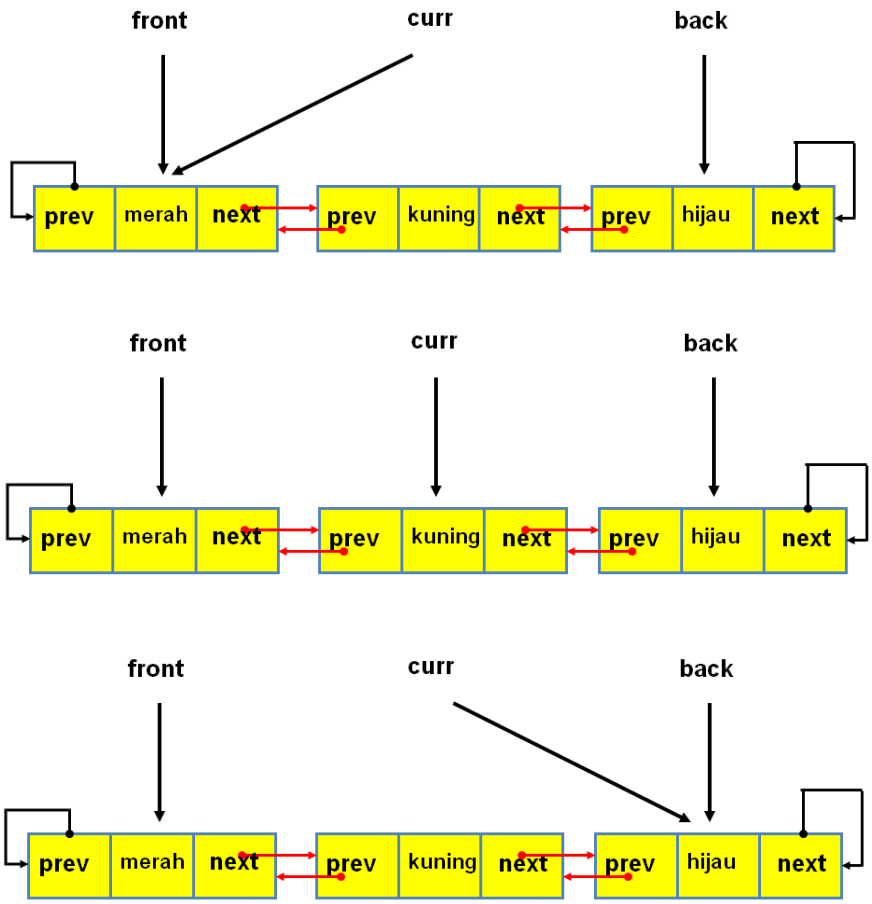
3. Pembacaan Maju Double Linked List

Pembacaan maju (forward scan) yaitu membaca double linked list dimulai dari reference **front** dan berakhir pada reference **back**. Diperlukan variable reference **curr** untuk menunjuk ke node yang sedang dibaca. Misal terdapat double linked list seperti di bawah ini.

Langkah-langkah pembacaan adalah :

1. Variabel reference curr menunjuk ke node yang ditunjuk oleh variable reference front.
2. Lakukan pembacaan pada node yang ditunjuk oleh curr.
3. Lakukan pengecekan apakah curr.next != curr. Jika ya lakukan langkah 4, jika tidak maka pembacaan double linked list selesai.
4. Arahkan curr ke curr.next
5. Lakukan pembacaan pada node yang ditunjuk oleh curr. Kembali ke langkah 3.

```
DNode<T> curr = front ;
String str = "[" + curr.nodeValue;
while(curr.next != this)
{     curr = curr.next;
            str += ", " + curr.nodeValue; }
str += "]";
```



Gambar 19.9 Pembacaan Maju di Double Linked List

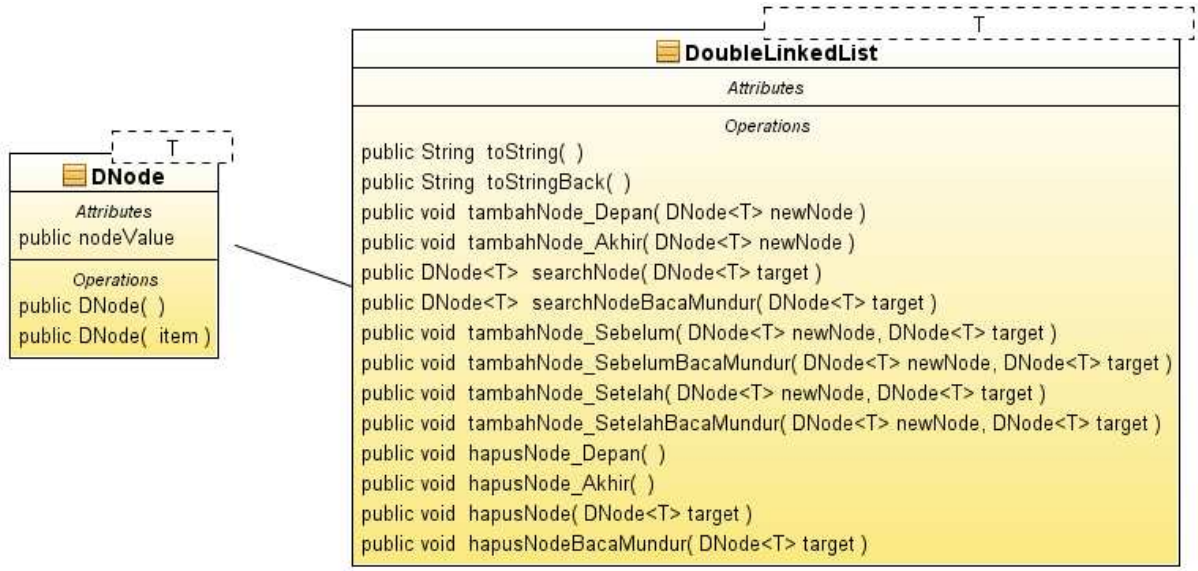
C. TUGAS PENDAHULUAN

Buatlah review mengenai :

- Double Linked List
- Representasi Node pada Double Linked List.
- Cara menghapus node di depan Double Linked List.

D. PERCOBAAN

Pada percobaan Double Linked List ini menerapkan UML seperti dibawah ini, dengan membuat class DNode dan Classs Double Linked List.



Gambar 19.10 Class Double Linked List

Percobaan 1 : Menghapus Node di awal Double Linked List dengan membuat method

public void hapusNode_Depan()

```
public void hapusNode_Depan() {
        if (front != null) {
            front = front.next;
            front.prev.next = null;
            front.prev = front;
        }
    }
```

Percobaan 2 : menghapus Node di akhir Double Linked List

public void hapusNode_Akhir()

```
public void hapusNode_Akhir() {
        if (back != null) {
            back = back.prev;
            back.next.prev = null;
            back.next = back;
        }
    }
```

E. LATIHAN

Pada Class DoubleLinkedList tambahkan method - method di bawah ini !

1. Buatlah method untuk mencari Node tertentu dengan pembacaan Maju

public DNode<T> searchNode(DNode<T> target)

2. Buatlah method untuk mencari Node tertentu dengan pembacaan Mundur

   public DNode<T> searchNodeBacaMundur(DNode<T> target)

3. Buatlah method untuk menghapus Node tertentu di Double Linked List(pembacaan List menggunakan pembacaan maju)

   public void hapusNode (DNode<T> target)

4. Buatlah method untuk menghapus Node tertentu di Double Linked List(pembacaan List menggunakan pembacaan Mundur)

   public **void hapusNodeBacaMundur (DNode<T> target)**

Buatlah class DoubleLinkedListDemo2 untuk menguji semua method yang terdapat pada class DoubleLinkedList yang telah dikerjakan pada praktikum ini.

```
public class DoubleLinkedListDemo2 {

    public static void main(String[] args) {
        DoubleLinkedList<String> Dlist = new DoubleLinkedList<String>();
        System.out.println(Dlist.toString());
        Dlist.tambahNode_Akhir(new DNode<String>("ungu"));
        System.out.println("Tambah Node Akhir[LIst Kosong] : " + Dlist.toString());

        Dlist.tambahNode_Depan(new DNode<String>("merah"));
        System.out.println("Tambah Node di Depan : " + Dlist.toString());

        Dlist.tambahNode_Depan(new DNode<String>("ungu"));
        System.out.println("Tambah Node di Depan : " + Dlist.toString());

        Dlist.tambahNode_Akhir(new DNode<String>("kuning"));
        System.out.println("Tambah Node di Akhir : " + Dlist.toString());

        Dlist.hapusNode_Depan();
        System.out.println("Hapus Node di Depan : " + Dlist.toString());

        Dlist.hapusNode_Akhir();
        System.out.println("Hapus Node di Akhir : " + Dlist.toString());
```

```
        Dlist.hapusNode(new DNode<String>("oranye"));
        System.out.println("Hapus  Node  Sesuai  Target(Target  di  Awal  List)  :  "  +
Dlist.toString());

        Dlist.hapusNode(new DNode<String>("pink"));
        System.out.println("Hapus  Node  Sesuai  Target(Target  di  Akhir  List)  :  "  +
Dlist.toString());
    }
}
```

Output :

```
Double Linked List Kosong
Tambah Node Akhir[LIst Kosong] : [ungu]
Tambah Node di Depan : [merah, ungu]
Tambah Node di Depan : [ungu, merah, ungu]
Tambah Node di Akhir : [ungu, merah, ungu, kuning]
Hapus Node di Depan : [merah, ungu, kuning]
Hapus Node di Akhir : [merah, ungu]
```

DOUBLE LINKED LIST UNTUK POLINOMIAL

1. Masalah aritmatika polinom adalah membuat sekumpulan subrutin manipulasi terhadap polinom simbolis (symbolic Polynomial). Misalnya: $P1 = 6x^8 + 8x^7 + 5x^5 + x^3 + 15$

   $P2 = 3x^9 + 4x^7 + 3x^4 + 2x^3 + 2x^2 + 10$

   $P3 = x^2 + 5$

   Representasikan bilangan polinom dengan menggunakan linked list dan buatlah prosedur-prosedur untuk :

   - Menyisipkan simpul di awal jika pangkat yang dimasukkan lebih dari pangkat tertinggi dari bilangan polinomial.
   - Menyisipkan simpul di tengah jika pangkat dari bilangan yang kita sisipkan berada di tengah.
   - Menyisipkan simpul di akhir jika pangkat dari bilangan yang disisipkan adalah 0.
   - Menghapus simpul, baik di awal, di tengah, ataupun di akhir.
2. Lakukan pejumlahan dan pengurangan pada aritmatika polinom

   Contoh :

   $P1 + P2 = 3x^9 + 6x^8 + 12x^7 + 5x^5 + 3x^4 + 3x^3 + 2x^2 + 25$

   $P1 – P2 = 6x^8 + 8x^7 + 5x^5 + x^3 + 15 – (3x^9 + 4x^7 + 3x^4 + 2x^3 + 2x^2 + 10) = -3x^9 + 6x^8 + 4x^7 + 5x^5 - x^3 – 5.$

F. LAPORAN RESMI

Kerjakan hasil percobaan(D) dan latihan(E) di atas dan tambahkan analisa.