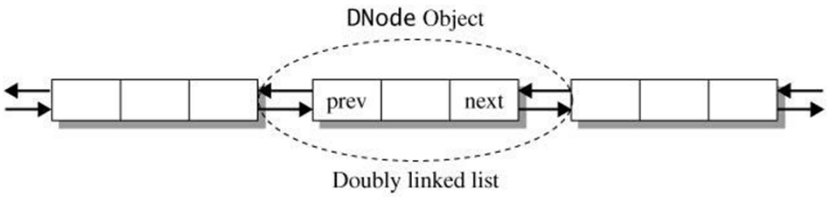
Double Linked List

Arna Fariza

Yuliana Setiowati

Double Linked List

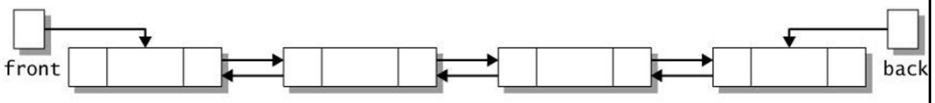
- Sama seperti single linked list, double linked list mempunyai struktur sequential.
- Double Linked List terdiri dari dua reference yang menunjuk ke node selanjutnya (**next** node) dan node sebelumnya (**previous** node)
- Untuk bergerak maju dan mundur pada double linked list menggunakan link **next** dan **prev** pada node.



Doubly linked list with nodes having two reference fields.

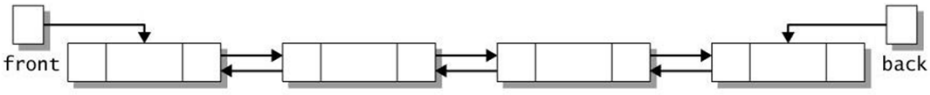
Double Linked List

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



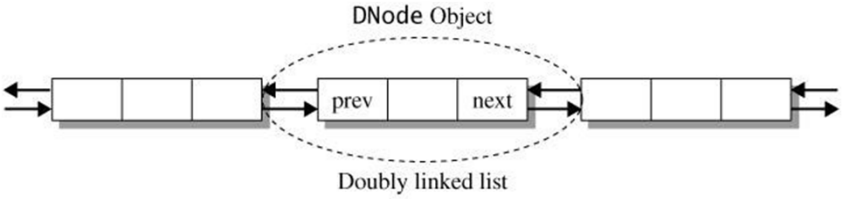
Pembacaan pada Double Linked List

- 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**.



Class DNode

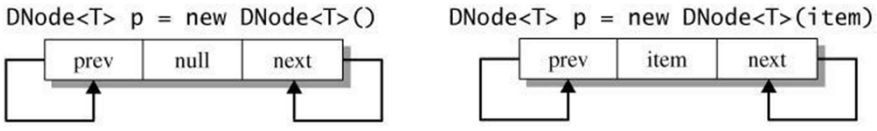
- Node pada Double Linked List direpresentasikan dengan class DNode
- Kumpulan object DNode membentuk sebuah list disebut **double linked list**



Doubly linked list with nodes having two reference fields.

Class DNode

- Object DNode mempunyai tiga variabel:
  - **nodeValue** untuk menyimpan nilai
  - **prev** untuk menandai node sebelumnya
  - **Next** untuk menandai node sesudahnya.



Creating DNode objects with a null data field or item of generic type T as the data field.

Constructor Class DNode

- 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**.

Class DNode

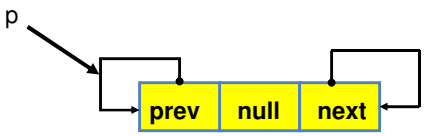
```
public class DNode<T>
{
   public T nodeValue;    // data value of the node
   public DNode<T> prev;  // previous node in the list
   public DNode<T> next;  // next node in the list

   // 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;
   }
```

Class DNode

```
    // 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;
    }
}
```

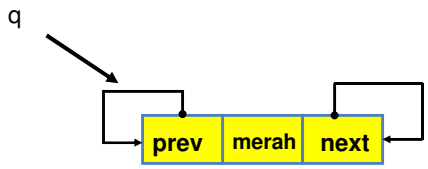

Membuat Node p

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

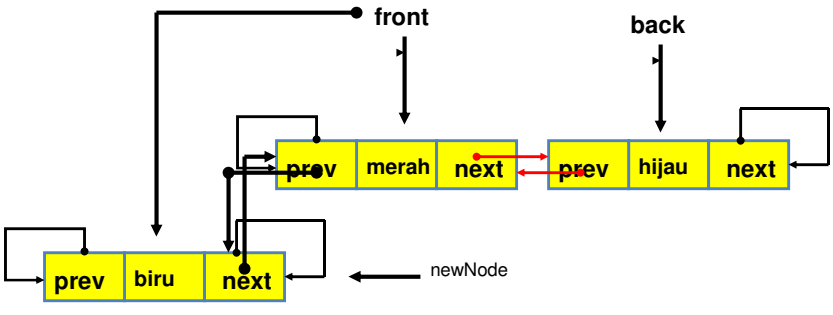


Membuat Node q

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

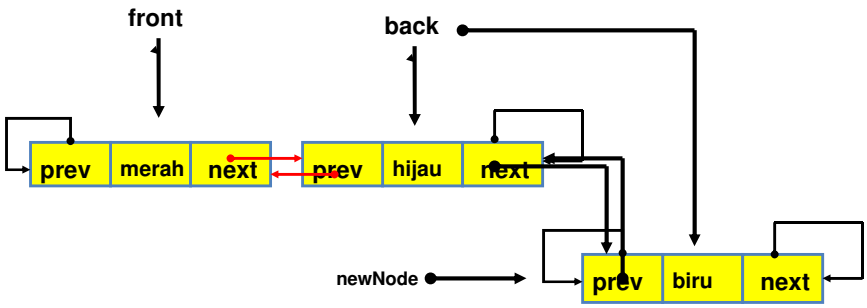


Menyisipkan Node di Depan List



- DNode<String> newNode = new DNode<String>(“biru”);
- newNode.next = front ;
- front.prev = newNode ;
- front = newNode ;

Menyisipkan Node di Belakang List



- DNode<String> newNode = new DNode<String>("biru");
- back.next = newNode ;
- newNode.prev = back ;
- back = newNode

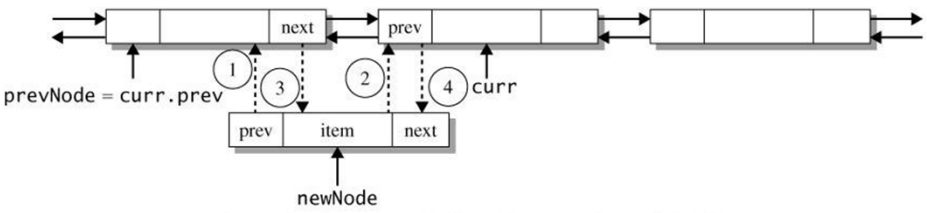
Menyisipkan Node di Double Linked List

- 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**.

Menyisipkan Node di Double Linked List

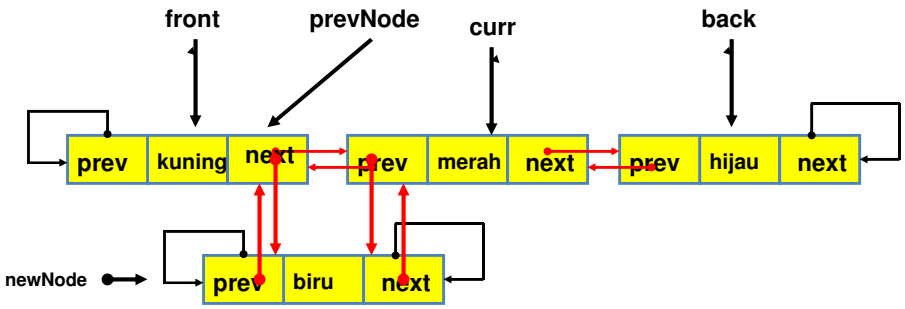
```
// declare the DNode reference variables newNode and prevNode
DNode<T> newNode, prevNode;
// create a new node and assign prevNode to reference the
// predecessor of curr
newNode = new DNode<T>(item);
prevNode = curr.prev;

// 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 Double Linked List



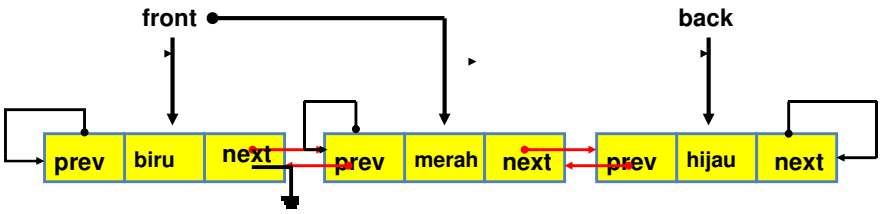
Inserting a node with value item before node curr in a doubly linked list.

Menyisipkan Node di Double Linked List



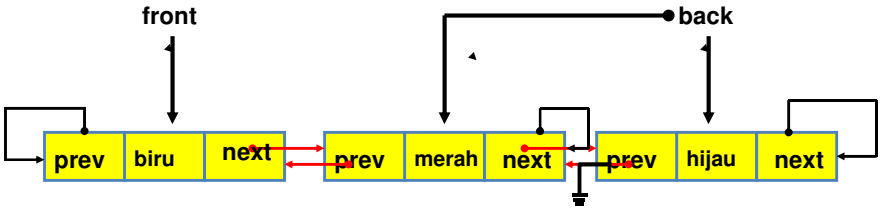
- newNode.prev = prevNode; // statement 1
- newNode.next = curr; // statement 2
- prevNode.next = newNode; // statement 3
- curr.prev = newNode; // statement 4

Menghapus Node di Depan List



- Front = front.next ;
- Front.prev.next = null ;
- Front.prev = this ;

Menghapus Node di Belakang List



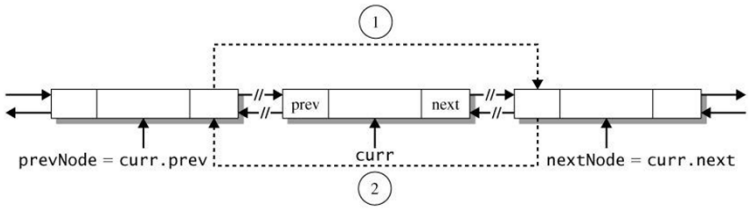
- Back = back.prev ;
- back.next.prev = null ;
- Back.next = this ;

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.**

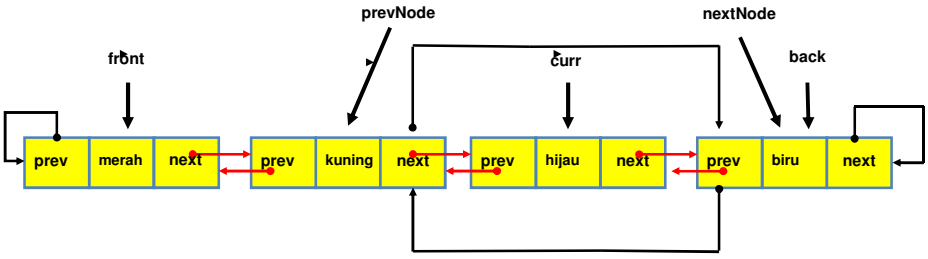
Menghapus Node Sesuai Target

```
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
```



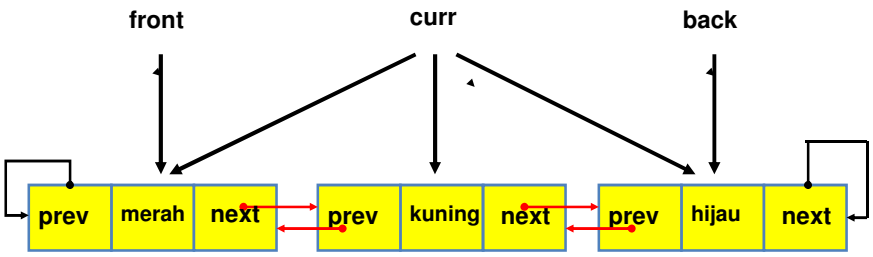
Deleting node curr in a doubly linked list.

Menghapus Node Sesuai Target



- **DNode<T> prevNode = curr.prev, nextNode = curr.next;**
- **prevNode.next = nextNode; // statement 1**
- **nextNode.prev = prevNode; // statement 2**

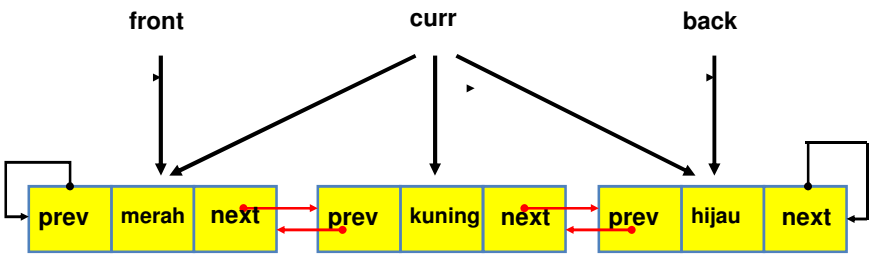
Membaca Maju Double Linked List




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

[merah ,kuning ,hijau]

Membaca Mundur Double Linked List




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

[hijau ,kuning ,merah]