LAPORAN PRAKTIKUM STRUKTUR DATA

DOUBLE LINKED LIST

OLEH:

AHMAD ZAHRAN

2411532004

DOSEN PENGAMPU:

DR. WAHYUDI, S.T, M.T

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

2024

A. Pendahuluan

Struktur data Double Linked List (DLL) merupakan salah satu struktur data dinamis yang setiap node-node nya memiliki dua pointeer yaitu ‘next’ yang menunjuk ke node berikutnya dan ‘prev’ yang menunjuk ke node sebelumnya.

Karena memiliki dua buah pointer ini, DLL memungkinkan penelusuran data secara maju dan mundur.

B. Tujuan

Tujuan dilakukannya praktikum ini adalah sebagai berikut:

1. Memahami Struktur Data DoubleLinkedList pada java
2. Mengimplementasikan DoubleLinkedList

C. Langkah Kerja Praktikum

a. Kelas NodeDLL

```
public class NodeDLL {
    int data;   // node bagian data
    NodeDLL next;   // pointer ke node berikutnya
    NodeDLL prev;
```

1. Menyimpan nilai data kedalam node, dan masing-masing pointer ‘next’, ‘prev’ untuk menuju node setelah dan sebelumnya.

```
    // konstruktor menginisialisasi node dengan data
    public NodeDLL(int data) {
        this.data = data;
        this.next = null;
    }
}
```

2. Construktor untuk mengatur data dan pointer next ke nilai “null”.

b. Kelas InsertDLL

```
// fungsi menambahkan node di awal DLL
static NodeDLL insertBegin(NodeDLL head, int data) {
    // buat node baru
    NodeDLL new_node = new NodeDLL(data);
    // jadikan pointer next-nya ke head
    new_node.next = head;
    // jika list tidak kosong, set prev head ke new_node
    if (head != null) {
        head.prev = new_node;
    }
    return new_node;
}
```

1. Menambah node di depan, Membuat Node baru dengan data yang diberikan, Pointer Node baru menjadi head, dan jika list tidak kosong, set prev head ke new_node dan kembalikan new_node.

```
// fungsi menambahkan node di akhir
public static NodeDLL insertEnd(NodeDLL head, int newData) {
    // buat node baru
    NodeDLL newNode = new NodeDLL(newData);
    // jika dll null jadikan head
    if (head == null) {
        head = newNode;
    } else {
        NodeDLL curr = head;
        while (curr.next != null) {
            curr = curr.next;
        }
        curr.next = newNode;
        newNode.prev = curr;
    }
    return head;
}
```

2. Menambah node di akhir, Buuat node baru, dan jika list kosong, head menjadi node baru, dan else list tidak kosong maka, cari node terakhir dan gunakan node prev untuk mentautkan node baru setelahnya.

```
public static NodeDLL insertAtPosition(NodeDLL head, int pos, int new_data) {
 // buat node baru
 NodeDLL new_node = new NodeDLL(new_data);
 if (pos == 1) {
     new_node.next = head;
     if (head != null) {
         head.prev = new_node;
     }
     head = new_node;
     return head;
 }
 NodeDLL curr = head;
 for (int i = 1; i < pos - 1 && curr != null; ++i) {
     curr = curr.next;
 }
 if (curr == null) {
     System.out.println("Posisi tidak ada");
     return head;
 }
 new_node.prev = curr;
 new_node.next = curr.next;
 curr.next = new_node;
 if (new_node.next != null) {
     new_node.next.prev = new_node;
 }
 return head;
}
```

3. Menyisipkan node di posisi tertentu, if posisi sama dengan 1, lakukan penambahan di awal, dan untuk seterusnya kita buat perulangan untuk menelusuri node sebelum posisi yang ingin dituju, lalu jika posisi sama dengan null atau tidak ada, maka langsung print"posissi tidak ada"

```
public static void main (String [] args) {
    System.out.println("AhmadZahran");
    System.out.println("2411532004");
    // membuat dll 2 <->3<->5
    NodeDLL head = new NodeDLL(2);
    head.next = new NodeDLL(3);
    head.next.prev = head;
    head.next.next = new NodeDLL(5);
    head.next.next.prev = head.next;
    //cetak DLL awal
    System.out.println("DLL Awal: ");
    printList(head);
    head = insertBegin(head, 1);
    System.out.print("Simpul1 tambah diawal: ");
    printList(head);
    head = insertBegin(head, 6);
    System.out.print("Simpul 6 tambah diakhir: ");
    int data = 6;
    head = insertEnd(head,data);
    printList(head);
    System.out.println("tambah node 4 di posisi 4: ");
    int data2 = 4;
    int pos = 4;
    head = insertAtPosition(head, pos, data2);
    printList(head);
    }
}
```

4. Fungsi main untuk mengecek apakah DLL kita berjalan atau tidak, Menambahkan node kedua dengan data 3, dan menyambungkannya sebagai next dari head.Menetapkan prev node 3 menunjuk kembali ke node 2 (membentuk hubungan dua arah).Menetapkan prev node 5 menunjuk ke node 3 (lengkap sudah hubungan 2, 3, 5).Menampilkan seluruh isi list dari head ke tail menggunakan fungsi printList().Menambahkan node dengan data 1 di awal list, mencetak hasilnya. Membuat variabel data = 6 dan menambahkan 6 ke akhir list secara benar dengan insertEnd(). Cetak hasil list.Menambahkan node dengan data 4 pada posisi ke-4 dalam list. Setelah itu mencetak isi list.

c. Kelas HapusDLL

```
public static NodeDLL delHead(NodeDLL head) {
    if (head == null) {
        return null;
    }
    NodeDLL temp = head;
    head = head.next;
    if (head != null) {
        head.prev = null;
    }
    return head;
}
```

1. Menghapus Node pertama, Memeriksa apakah list kosong, jika kosong, maka langsung kembalikan nilai null, dan geser head menjadi node berikutnya, dan jika node pertama dihapus, periksa lagi apakah head bukan null, untuk menjaga konsistensi list.

```
public static NodeDLL delLast(NodeDLL head) {
    if (head == null) {
        return null;
    }
    if (head.next == null) {
        return null;
    }
    NodeDLL curr = head;
    while (curr.next != null) {
        curr = curr.next;
    }
    // update pointer previous node
    if (curr.prev != null) {
        curr.prev.next = null;
    }
    return head;
}
```

2. Menghapus Node Terakhir, seperti biasa cek dulu apakah list kosong kalu kosong langsung kembalikan nilai null, telusuri list dari node pertama, buat loop untuk mengecek jika belum kondisi null maka cari ke node selanjutnya, dan seletalh menemukna node terakhir, buat node next menjadi null

```
/fungsi menghapus node posisi tertentu
ublic static NodeDLL delPos(NodeDLL head, int pos
// jika DLL kosong
if (head == null)
    return head;
NodeDLL curr = head;
// telusuri sampai ke node yang akan dihapus
for (int i = 1; curr != null && i < pos; i++) {
    curr = curr.next;
}
// jika posisi tidak ditemukan
if (curr == null)
    return head;
// update pointer
if (curr.prev != null)
    curr.prev.next = curr.next;
if (curr.next != null)
    curr.next.prev = curr.prev;
// jika yang dihapus head
if (head == curr)
    head = curr.next;
return head;
```

3. Menghapus node di posisi tertentu buat perulangan untuk mencari node yang akan dihapus dan perbarui pointer prev dan next dari node sebelahnya agar list tetap terhubung, dan update nilai head jika diperlukan.

```
//fungsi mencetak DLL
public static void printList(NodeDLL head) {
 NodeDLL curr = head;
 while (curr != null) {
     System.out.print(curr.data + " ");
     curr = curr.next;
 }
 System.out.println();
}
```

4. Menelusuri dan mencetak isi linked list dari depan ke belakang

```
ublic static void main(String[] args) {
    System.out.println("AhmadZahran");
    System.out.println("2411532004");
    // buat sebuah DLL
    NodeDLL head = new NodeDLL(1);
    head.next = new NodeDLL(2);
    head.next.prev = head;

    head.next.next = new NodeDLL(3);
    head.next.next.prev = head.next;

    head.next.next.next = new NodeDLL(4);
    head.next.next.next.prev = head.next.next;

    head.next.next.next.next = new NodeDLL(5);
    head.next.next.next.next.prev = head.next.next.next;

    System.out.print("DLL Awal: ");
    printList(head);

    System.out.print("Setelah head dihapus: ");
    head = delHead(head);
    printList(head);
    System.out.print("Setelah node terakhir dihapus: ");
    head = delLast(head);
    printList(head);
    System.out.print("Menghapus node ke-2: ");
    head = delPos(head, 2);
    printList(head);
}
```

5. Kelas main yang mana proses-prosesnya ada, membuat list awal, menghapus node pertama, menghapus node treakhir, dan menghapus node di posisi ke-2 lalu cetak.

d. Kelas PenelusuranDLL

```
// fungsi penelusuran maju
static void forwardTraversal(NodeDLL head) {
    // memulai penelusuran dari head
    NodeDLL curr = head;
    // lanjutkan sampai akhir
    while (curr != null) {
        // print data
        System.out.print(curr.data + " <-> ");
        // pindah ke node berikutnya
        curr = curr.next;
    }
    // print spasi
    System.out.println();
}
```

1. Fungsi Penelusuran maju, digunakan untuk menelusuri DLL dari awal menggunakan pointer next

```
// fungsi penelusuran mundur
static void backwardTraversal(NodeDLL tail) {
    // mulai dari akhir
    NodeDLL curr = tail;
    // lanjut sampai head
    while (curr != null) {
        // cetak data
        System.out.print(curr.data + " <-> ");
        // pindah ke node sebelumnya
        curr = curr.prev;
    }
    // cetak spasi
    System.out.println();
}
```

2. Fungsi Penelusuran mundur, untuk menelusuri DLL dari akhir menggunakan pointer prev.

```java
public static void main (String[]args) {
// cetakDLL
    NodeDLL head = new NodeDLL(1);
    NodeDLL second = new NodeDLL(2);
    NodeDLL third = new NodeDLL(3);

    head.next = second;
    second.prev = head;
    second.next = third;
    third.prev = second;

    System.out.println("Penelusuran maju: ");
    forwardTraversal(head);

    System.out.println("Penelusuran mundur: ");
    backwardTraversal(third);
}
```

3. Fungsi main, Membuat linked list secara manual, menyambungkan semua list agar biida di telusuri secara maju maupun mundur, dan cetak keduanya baik penelusuran maju maupun mundur sekian…

D. Kesimpulan

Melalui praktikum ini, mahasiswa memperoleh pemahaman yang lebih dalam tentang bagaimana struktur data Double Linked List bekerja. Dengan membagi implementasi ke dalam beberapa kelas terpisah, program menjadi lebih modular dan mudah dikelola. Operasi penambahan, penghapusan, dan pencarian telah berhasil diimplementasikan dan diuji menggunakan Java.