LAPORAN PRAKTIKUM 6

STRUKTUR DATA

Oleh:
Ibrahim Mousa Dhani
2411532010

Dosen Pengampu : Dr. Wahyudi S.T, M.T.

Mata Kuliah : Struktur Data

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

PADANG

A. Pendahuluan

Linked list merupakan struktur data linear yang terdiri dari simpul-simpul (node) yang saling terhubung satu sama lain melalui pointer. Setiap node pada linked list biasanya terdiri dari dua bagian, yaitu bagian data dan bagian pointer (penunjuk) ke node berikutnya.

Pada pratikum sebelumnya kita sudah belajar tentang Single Linked List (SLL), sekarang kita akan mempelajari tentang Double Linked List (DLL). DLL adalah salah satu jenis struktur data linier yang terdiri dari rangkaian node di mana setiap node terhubung ke dua node lainnya melalui dua pointer, yaitu pointer ke node berikutnya dan pointer ke node sebelumnya. Berbeda dengan Single Linked List yang hanya memiliki satu arah koneksi, DLL memungkinkan penelusuran data dari dua arah yaitu dari awal ke akhir dan dari akhir ke awal. Karena itulah, DLL sering digunakan dalam aplikasi yang memerlukan kemampuan navigasi bolak-balik seperti browser history, editor teks, dan sistem navigasi data lainnya.

Struktur dasar dari setiap node pada DLL terdiri dari tiga komponen utama, yaitu data (informasi yang disimpan), pointer ke node berikutnya (next), dan pointer ke node sebelumnya (prev). Dengan dua pointer ini, operasi seperti penyisipan (insertion) dan penghapusan (deletion) node di posisi manapun menjadi lebih fleksibel dan efisien, karena tidak perlu melakukan traversal dari awal list sebagaimana pada Single Linked List. Hal ini membuat DLL unggul dalam hal efisiensi waktu pada operasi tertentu, terutama jika lokasi node sudah diketahui.

Namun, DLL juga memiliki kelemahan dibandingkan struktur data lainnya. Karena setiap node memiliki dua pointer, penggunaan memori menjadi lebih besar dibanding Single Linked List. Selain itu, pengelolaan pointer harus sangat hati-hati, karena kesalahan dalam memperbarui pointer saat menambah atau menghapus node dapat menyebabkan list menjadi rusak atau tidak dapat diakses. Meskipun demikian, dengan pengelolaan yang baik, DLL merupakan struktur data yang sangat powerful dan fleksibel untuk menangani berbagai jenis permasalahan dalam pemrograman dan pengolahan data.

B. Tujuan Praktikum

1. Mengetahui dan memahami konsep dasar struktur data Double Linked List (DLL) dan komponennya.
2. Mengimplementasikan operasi dasar DLL dalam bahasa pemrograman Java.
3. Memahami cara penambahan, dan penghapusan, node dalam linked list pada berbagai posisi (depan, belakang, dan posisi tertentu), serta penelusuran menggunakan traversal dan mencetak isi DLL
4. Meningkatkan pemahaman konsep pointer dan hubungan antar node dalam DDL

C. Langkah Langkah

a. NodeDLL

1. Pertama buat package baru dan beri nama pekan6. Buat kelas baru dan beri nama NodeDLL. Kelas ini adalah struktur dasar yang nantinya akan digunakan oleh struktur Double Linked List yang lengkap

```
package pekan6;

public class NodeDLL {
// Nama : Ibrahim Mousa Dhani
// NIM  : 2411532010
```

2. Deklarasikan atribut ‘data’ bertipe int, atribut ini akan menyimpan nilai atau data yang dimiliki oleh node
3. Setelah itu deklarasikan juga atribut ‘next’. Tipe datanya adalah NodeDLL, artinya menunjuk ke objek lain dari kelas NodeDLL. Atribut ini adalah pointer ke node berikutnya, dan deklarasikan juga atribut ‘prev’ yang akan menjadi pointer ke node sebelumnya

```
// mendefinisakan kelas Node
int data;         // data
NodeDLL next;     // Pointer ke next node
NodeDLL prev;     // Pointer ke previous node
```

4. Lalu buat Konstruktor, konstruktor merupakan method khusus yang dipanggil saat objek dibuat. Konstruktor ini menerima satu parameter yaitu ‘int data’

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

this.data = data; berfungsi untuk menyimpan nilai ke dalam atribut data milik objek
this.next = null; artinya node ini belum menunjuk ke node lain/selanjutnya, karena saat dibuat biasanya node berdiri sendiri dulu
this.prev = null; artinya sama seperti yang next, tetapi ini belum menunjuk ke node yang laim/sebelumnya

5. Berikut kode lengkapnya

```
1  package pekan6;
2
3  public class NodeDLL {
4  // Nama : Ibrahim Mousa Dhani
5  // NIM  : 2411532010
6      // mendefinisakan kelas Node
7      int data;         // data
8      NodeDLL next;     // Pointer ke next node
9      NodeDLL prev;     // Pointer ke previous node
10
11     // konstruktor
12     public NodeDLL(int data) {
13         this.data = data;
14         this.next = null;
15         this.prev = null;
16     }
17 }
```

b. InsertDLL

1. Buat kelas baru dan beri nama InsertDLL. Gunakan NodeDLL yang sudah kamu buat sebelumnya sebagai struktur node nya

```
package pekan6;

public class InsertDLL {
// Nama : Ibrahim Mousa Dhani
// NIM  : 2411532010
```

1. Buat fungsi insertBegin() untuk menambahkan node baru di bagian depan dari double linked list. Buat node baru menggunakan konstruktor NodeDLL yang telah dibuat sebelumnya. Setelah node dibuat, pointer next dari node baru diarahkan ke head. Lalu lakukan pengecekan, jika head tidak kosong/null maka pointer prev dari node lama yang sebelumnya menjadi head, arahkan ke node baru. Sehingga hubungan dua arah antar node tetap terjaga. Return node baru sebagai head yang baru.

```
// 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;
    // jadikan pointer prev head ke new_node
    if (head != null) {
        head.prev = new_node;
    }
    return new_node;
```

2. Lalu buat juga fungsi insertEnd() untuk menambahkan node baru di bagian akhir dari double linked list. sama seperti sebelumnya buat node baru, lalu jika linked list null/kosong, maka node baru langsung menjadi head. Jika linked list sudah berisi, maka dilakukan iterasi dari head hingga node terakhir. Setelah itu, node terakhir akan mengarahkan pointer next nya ke node baru, dan sebaliknya, node baru akan mengarahkan pointer prev-nya ke node terakhir. List akan tetap terhubung secara dua arah, dan head takan tetap dikembalikan.

```
// fungsi menambahkan node di akhir
public static NodeDLL insertEnd(NodeDLL head, int newData) {
    // Buat node baru
    NodeDLL newNode = new NodeDLL(newData);
    // Jika DLL nul, jadikan newNode sebagai 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;
```

3. Buat fungsi insertAtPosition() yang bertujuan untuk menambahkan node di posisi tertentu berdasarkan nomor posisi yang dimasukkan oleh user. Buat node baru, jika posisi yang diinginkan adalah 1, maka artinya node baru akan menjadi head baru. Jika node lama tidak null, maka pointer prev dari node lama dihubungkan ke node baru. Setelah itu, node baru menjadi head baru. Jika posisi yang diinginkan lebih dari 1, maka dilakukan iterasi dari head untuk mencari node di posisi ke pos - 1. Jika posisi tidak ditemukan (misalnya melebihi panjang list), maka dicetak pesan bahwa posisi tidak tersedia

```
fungsi menambahkan node di posisi tertentu
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;
```

4. Buat fungsi printList() untuk menampilkan isi DLL. Telusuri node satu per satu mulai dari head. Selama masih ada node berikutnya (curr.next != null), cetak data + <->

```
public static void printList(NodeDLL head) {
    NodeDLL curr = head;
    while (curr != null) {
        System.out.print(curr.data + " <-> ");
        curr = curr.next;
    }
    System.out.println();
```

5. Selanjutnya yang terakhir buat fungsi main(). Di dalam bagian ini, pertama cetak nama dan NIM sebagai identitas. Selanjutnya buat DLL awal terlebih dahulu secara manual yang berisikan tiga node yaitu 2, 3, dan 5. Lalu cetak/tampilkan DLL awal ke layer menggunakan printList

```
public static void main(String[] args) {
    System.out.println("Nama : Ibrahim Mousa Dhani");
    System.out.println("NIM  :2411532010\n");
    // 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.print("DLL Awal: ");
    printList(head);
```

6. Tambahkan node baru (1) di bagian depan menggunakan insertBegin dan tampilkan ke layar

```
head = insertBegin(head, 1);
System.out.print(
        "simpul 1 ditambah di awal: ");
printList(head);
```

7. Tambahkan node baru (6) di bagian belakang menggunakan insertEnd lalu tampilkan

```
System.out.print(
        "simpul 6 ditambah di akhir: ");
int data = 6;
head = insertEnd(head, data);
printList(head);
```

8. Terakhir tambahkan node baru di posisi ke-4 menggunakan insertAtPosition lalu tampilkan hasilnya

```
// menambah node 4 di posisi 4
System.out.print("tambah node 4 di posisi 4: ");
int data2 = 4;
int pos = 4;
head = insertAtPosition(head, pos, data2);
printList(head);
```

9. Setelah semua benar dan tidak ada eror, maka jalankan program, lalu akan menghasilkan output sebagai berikut

```
Nama : Ibrahim Mousa Dhani
NIM  :2411532010

DLL Awal: 2 <-> 3 <-> 5 <->
simpul 1 ditambah di awal: 1 <-> 2 <-> 3 <-> 5 <->
simpul 6 ditambah di akhir: 1 <-> 2 <-> 3 <-> 5 <-> 6 <->
tambah node 4 di posisi 4: 1 <-> 2 <-> 3 <-> 4 <-> 5 <-> 6 <->
```

10. Berikut kode program versi lengkapnya

```
package pekan6;

public class InsertDLL {
// Nama : Ibrahim Mousa Dhani
// NIM  : 2411532010

    // 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;
        // jadikan pointer prev head ke new_node
        if (head != null) {
            head.prev = new_node;
        }
        return 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 newNode sebagai 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;
    }
  // fungsi menambahkan node di posisi tertentu
    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;
    }
    public static void printList(NodeDLL head) {
        NodeDLL curr = head;
        while (curr != null) {
            System.out.print(curr.data + " <-> ");
            curr = curr.next;
        }
        System.out.println();
    }
    public static void main(String[] args) {
        System.out.println("Nama : Ibrahim Mousa Dhani");
        System.out.println("NIM  :2411532010\n");
        // 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.print("DLL Awal: ");
        printList(head);
        // tambah 1 di awal
        head = insertBegin(head, 1);
        System.out.print(
                "simpul 1 ditambah di awal: ");
        printList(head);
        // tambah 6 di akhir
        System.out.print(
                "simpul 6 ditambah di akhir: ");
        int data = 6;
        head = insertEnd(head, data);
        printList(head);
        // menambah node 4 di posisi 4
        System.out.print("tambah node 4 di posisi 4: ");
        int data2 = 4;
        int pos = 4;
        head = insertAtPosition(head, pos, data2);
        printList(head);
```

c. PenelusuranDLL

1. Buat kelas baru pada package pekan5 dan beri nama PenelusuranDLL. Gunakan NodeDLL yang sudah kamu buat sebelumnya sebagai struktur node nya
2. Selanjutnya buat fungsi forwardTraversal() untuk menelusuri dan mencetak isi

```
package pekan6;

public class PenelusuranDLL {
// Nama : Ibrahim Mousa Dhani
// NIM  : 2411532010
```

dari double linked list dari depan ke belakang. NodeDLL curr = head; artinya traversal dimulai dari node pertama/head, lalu selama curr tidak sama dengan null lanjutkan sampai akhir/tail, print data dan cetak baris baru

```
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();
```

3. Selanjutnya buat fungsi backwardTraversal() cara kerjanya kurang lebih sama seperti sebelumnya tetapi ini kebalikann nya, yaitu menelusuri dan mencetak isi dari double linked list dari belakang ke depan. NodeDLL curr = tail; artinya traversal dimulai dari node terakhir/tail, lalu selama curr tidak sama dengan null lanjutkan sampai awal/head print data dan cetak baris baru

```
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();
```

4. Selanjutnya buat fungsi main(). Di dalam bagian ini, pertama cetak nama dan NIM sebagai identitas. Lalu buat node awal terlebih dahulu secara manual yaitu 1 sebagai head, 2 sebagai second, dan 3 sebagai third

```
public static void main(String[] args) {
    System.out.println("Nama : Ibrahim Mousa Dhani");
    System.out.println("NIM  : 2411532010\n");
        // cetak DLL
        NodeDLL head = new NodeDLL(1);
        NodeDLL second = new NodeDLL(2);
        NodeDLL third = new NodeDLL(3);
```

5. Sambungkan node tersebut menjadi DLL seperti pada gambar berikut:
   head.next menjadi node kedua/second
   second.prev menjadi node awal/head
   second.next menjadi node ketiga/third
   third.prev menjadi node kedua/second

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

6. Setalah itu cetak DLL kita menggunakan/memanggil fungsi forwardTraversal dan backwardTraversal yang telah dibuat tadi lalu tampilkan hasilnya

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

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

7. Setelah semua benar dan tidak ada eror, maka jalankan program, lalu akan menghasilkan output sebagai berikut

```
Nama : Ibrahim Mousa Dhani
NIM  : 2411532010

Penelusuran maju:
1 <-> 2 <-> 3 <->
Penelusuran mundur:
3 <-> 2 <-> 1 <->
```

8. Berikut kode program versi lengkapnya

```
package pekan6;

public class PenelusuranDLL {
// Nama : Ibrahim Mousa Dhani
// NIM  : 2411532010
    //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();
        }
        //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();
        }
        public static void main(String[] args) {
            System.out.println("Nama : Ibrahim Mousa Dhani");
            System.out.println("NIM  : 2411532010\n");
               // cetak DLL
               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);
           }
}
```

d. HapusDLL

1. Buat kelas baru dan beri nama HapusDLL. Gunakan NodeDLL yang sudah kamu buat sebelumnya sebagai struktur node nya

```
package pekan6;

public class HapusDLL {
//Nama : Ibrahim Mousa Dhani
//NIM  : 2411532010
```

2. Buat fungsi delHead yang berfungsi untuk menghapus node paling depan dari DLL. Pertama, fungsi mengecek apakah head bernilai null jika iya, maka list kosong dan langsung mengembalikan null. Jika tidak, node pertama/head disimpan di variabel sementara temp, kemudian head dipindahkan ke node berikutnya (head = head.next). Jika node baru setelah penghapusan tidak null, maka pointer prev dari node baru ini diset menjadi null, karena sekarang ia menjadi node pertama. Lalu kembalikan node baru tersebut sebagai head.

```
//fungsi menghapus node awal
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;
```

3. Selanjutnya buat fungsi delLast() untuk menghapus node terakhir dari DLL. Sama seperti sebelumnya, dilakukan pengecekan awal jika head bernilai null, atau jika head.next == null artinya hanya ada satu node dalam DLL, maka fungsi langsung mengembalikan null, karena tidak ada atau hanya ada satu node. Jika terdapat lebih dari satu node, traversal dilakukan dari head hingga node terakhir dengan menggunakan while (curr.next != null). Setelah ditemukan node terakhir (curr), pointer next dari node sebelumnya (curr.prev) diatur menjadi null, dan kembailkan head

```
//fungsi menghapus di akhir
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;
```

4. Buat fungsi delPos() untuk menghapus node pada posisi tertentu. Pertama, jika head bernilai null, maka lansung kembalikan. Selanjutnya dilakukan penelusuran dari head sampai ke posisi yang ditentukan menggunakan for loop. Jika posisi tidak ditemukan (curr == null), maka DLL dikembalikan tanpa perubahan. Jika node ditemukan, maka dilakukan pengaturan pointer: jika node tersebut memiliki node sebelumnya, maka prev.next dari node sebelumnya diarahkan ke node berikutnya dan jika node memiliki node berikutnya, maka next.prev dari node berikut diarahkan ke node sebelumnya. Jika node yang dihapus adalah head, maka head diperbarui menjadi curr.next. Setelah itu, head dikembalikan.

```
//fungsi menghapus node posisi tertentu
public 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 adalah head
    if (head == curr) {
        head = curr.next; }
    return head;
```

5. Buat fungsi printList() dibuat untuk menampilkan isi DLL. Fungsi ini menerima head sebagai parameter dan melakukan penelusuran dengan while loop. Setiap data dari node dicetak ke layar dengan format spasi antar data. Fungsi diakhiri dengan System.out.println() untuk mencetak baris baru setelah seluruh node ditampilkan.

```
//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();
```

6. Selanjutnya buat fungsi main(). Di dalam bagian ini, pertama cetak nama dan NIM sebagai identitas. Lalu buat DLL dengan lima node yaitu 1 sampai 5. Setiap node dihubungkan dua arah, next ke node setelahnya, dan prev ke node sebelumnya. Setelah list terbentuk, list awal dicetak ke layar dengan printList().

```
public static void main(String[] args) {
    System.out.println("Nama : Ibrahim Mousa Dhani");
    System.out.println("NIM  : 2411532010\n");
    //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);
```

7. Lalu hapus node di bagian depan/head (1) dengan menggunakan fungsi delHead yang telah dibuat tadi dan tampilkan

```
System.out.print("Setelah head dihapus: ");
head = delHead(head);
printList(head);
```

8. Hapus node bagian akhir/tail (5) dengan menggunakan fungsi delLast lalu

```
System.out.print("Setelah node terakhir dihapus: ");
head = delLast(head);
printList(head);
```

tampilkan

9. Terakhir hapus node di posisi ke-2 (yaitu 3) dan tampilkan hasilnya

```
System.out.print("menghapus node ke 2: ");
head = delPos(head, 2);

printList(head);
```

10. Setelah semua benar dan tidak ada eror, maka jalankan program, lalu akan menghasilkan output sebagai berikut

```
Nama : Ibrahim Mousa Dhani
NIM  : 2411532010

DLL Awal: 1 2 3 4 5
Setelah head dihapus: 2 3 4 5
Setelah node terakhir dihapus: 2 3 4
menghapus node ke 2: 2 4
```

11. Berikut kode program versi lengkapnya

```java
package pekan6;

public class HapusDLL {
//Nama : Ibrahim Mousa Dhani
//NIM  : 2411532010

    //fungsi menghapus node awal
    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;
    }
    //fungsi menghapus di akhir
    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;
        }
    //fungsi menghapus node posisi tertentu
    public 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 adalah head
        if (head == curr) {
            head = curr.next; }
        return head;
    }
    //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();
    }
    public static void main(String[] args) {
        System.out.println("Nama : Ibrahim Mousa Dhani");
        System.out.println("NIM  : 2411532010\n");
        //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);
    }
}
```

D. Kesimpulan

Berdasarkan praktikum yang telah dilakukan kita dapat disimpulkan bahwa struktur data Double Linked List (DLL) merupakan salah satu bentuk struktur data dinamis yang sangat berguna dalam pengelolaan data yang memerlukan fleksibilitas tinggi, terutama ketika dibutuhkan akses dua arah terhadap elemen-elemen di dalam list.

Pada pratikum ini kita mengetahui dan memahami serta mengimplementasikan konsep Double Linked List menggunakan bahasa Java. Implementasi dilakukan secara bertahap, mulai dari mendefinisikan kelas NodeDLL sebagai struktur node ganda, hingga membuat berbagai fungsi untuk memanipulasi list seperti: menambahkan node di awal, akhir, dan posisi tertentu, menghapus node dari awal, akhir, dan posisi tertentu, serta melakukan penelusuran maju dan mundur.

Karena memiliki dua arah akses/pointer, DLL lebih fleksibel dibandingkan SLL. DLL memungkinkan penelusuran data ke depan maupun ke belakang, serta memudahkan dalam operasi penyisipan dan penghapusan node di tengah-tengah list tanpa perlu traversal dari awal.