LAPORAN PRATIKUM

"PEKAN 6"

Disusun Untuk Memenuhi Tugas Mata Kuliah Struktur Data

DOSEN PENGAMPU:

Wahyudi, Dr. S.T. M.T.

DISUSUN OLEH:

Karimah Irsyadiyah (2411533018)

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

T.A 2024/2025

Daftar Pustaka

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam dunia pemrograman, struktur data merupakan elemen penting yang berfungsi untuk mengelola dan memanipulasi data secara efisien. Salah satu struktur data yang sering digunakan adalah *Doubly Linked List* (DLL), yaitu daftar berantai ganda yang memungkinkan penelusuran dua arah (maju dan mundur). Melalui praktikum ini, mahasiswa diharapkan dapat memahami konsep dasar, implementasi, dan operasi dasar seperti penambahan, penghapusan, dan penelusuran elemen dalam DLL.

1.2 Tujuan

- Memahami konsep dan implementasi Doubly Linked List.
- Mengimplementasikan operasi dasar pada DLL: penambahan, penghapusan, dan penelusuran elemen.
- Menganalisis alur dan hasil eksekusi program dari masing-masing class yang digunakan.

1.3 Alat dan Bahan

- Perangkat Keras: Laptop/PC
- Perangkat komputer dengan IDE (contoh: IntelliJ IDEA / Eclipse).
- Java Development Kit (JDK).

BAB II
PEMBAHASAN

2.1 Langkah-langkah Praktikum dan Pembahasan Program

A. Class : NodeDLL.java

Sebagai struktur node dasar dari Doubly Linked List. Masing-masing node memiliki tiga atribut utama: data, next (penunjuk ke node berikutnya), dan prev (penunjuk ke node sebelumnya).

```
package pekan6;

public class NodeDLL {
	int data;
	NodeDLL next;
	NodeDLL prev;

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

	// Karimah Irsyadiyah

	// 2411533018
}
```

- int data: menyimpan nilai node.
- NodeDLL next: menunjuk ke simpul berikutnya.
- NodeDLL prev: menunjuk ke simpul sebelumnya.
- Constructor NodeDLL(int data) digunakan untuk menginisialisasi simpul baru dengan nilai tertentu, dan pointer next serta prev bernilai null.

B. Class : InsertDLL.java

Melakukan penambahan simpul ke dalam DLL pada berbagai posisi: awal, akhir, dan posisi tertentu.

Langkah Program

1. Membuat DLL awal: 2 <-> 3 <-> 5.
2. Menambahkan node 1 di depan.
3. Menambahkan node 4 di posisi ke-4.

Penjelasan Kode Penting

- Menambahkan node di awal:

```
package pekan6;

public class InsertDLL {
	static NodeDLL insertBegin(NodeDLL head, int data) {
		NodeDLL new_node = new NodeDLL(data);
		new_node.next = head;
		if(head != null) {
			head.prev = new_node;
		}
		return new_node;
	}
```

Membuat simpul baru, menautkan ke head, dan mengubah prev dari head.

- Menambahkan di akhir list:

```
	public static NodeDLL insertEnd(NodeDLL head, int newData) {
		NodeDLL newNode = new NodeDLL(newData);
		if(head == null) {
			head = newNode;
		} else {
			NodeDLL curr = head;
			while(curr.next != null) {
				curr = curr.next;
			}
			curr.next = newNode;
			newNode.prev = curr;
		}
		return head;
	}
```

- Menambahkan di posisi tertentu:

```
public static NodeDLL insertAtPosition(NodeDLL head, int pos, int
new_data) {
		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;
      }
```

- Menampilkan list:

- Menyisipkan node di awal.
- Menyisipkan node di posisi tertentu.

```
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("Karimah Irsyadiyah");
            System.out.println("2411533018");

            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;

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

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

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

Hasil Eksekusi (Contoh Output):

```
DLL Awal: 2 <-> 3 <-> 5 <->

Simpul 1 ditambah di awal: 1 <-> 2 <-> 3 <-> 5 <->

Tambah node 4 di posisi 4: 1 <-> 2 <-> 3 <-> 4 <-> 5 <->
```

C. Class : PenelusuranDLL.java

1. Program ini berada dalam package bernama pekan6 dan digunakan untuk menelusuri data dalam struktur Doubly Linked List (DLL), baik dari depan ke belakang (maju) maupun sebaliknya (mundur).
2. Metode forwardTraversal digunakan untuk mencetak data node dari head ke tail dengan menggunakan pointer next.
3. Metode backwardTraversal mencetak data dari tail ke head menggunakan pointer prev.
4. Dalam fungsi main, dibuat tiga node dengan data 1, 2, dan 3, lalu disambungkan sehingga membentuk sebuah DLL yang utuh: 1 <-> 2 <-> 3.
5. Program menampilkan nama dan NIM praktikan sebagai identitas.
6. Output dari program menunjukkan dua hasil penelusuran: penelusuran maju mencetak 1 <-> 2 <-> 3 <->, dan penelusuran mundur mencetak 3 <-> 2 <-> 1 <->.

```
package pekan6;

public class PenelusuranDLL {
	static void forwardTraversal(NodeDLL head) {
		NodeDLL curr = head;

		while(curr != null) {
			System.out.print(curr.data + " <-> ");
			curr = curr.next;
		}
		System.out.println();
	}

	static void backwardTraversal(NodeDLL tail) {
		NodeDLL curr = tail;

		while(curr != null) {
			System.out.print(curr.data + " <-> ");
			curr = curr.prev;
		}
		System.out.println();
	}

	public static void main(String[] args) {
		System.out.println("Karimah Irsyadiyah");
		System.out.println("2411533018");
		System.out.println();

		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);
    }
}
```

Hasil Output:

```
Karimah Irsyadiyah

2411533018

Penelusuran maju:

1 <-> 2 <-> 3 <->

Penelusuran mundur:

3 <-> 2 <-> 1 <->
```

D. Class HapusSLL.java

Berikut penjelasan program HapusDLL.

Tujuan Program:

1. Program ini menunjukkan cara menghapus node dalam Doubly Linked List (DLL), baik dari awal, akhir, maupun posisi tertentu.
2. Metode delHead:
3. Menghapus simpul (node) pertama dalam list. Jika list kosong, langsung mengembalikan null. Jika tidak, pointer head dipindah ke node berikutnya dan prev node baru diatur menjadi null.
4. Metode delLast:
5. Menghapus simpul terakhir. Traverse dilakukan hingga node terakhir, lalu koneksi ke node sebelumnya diputus dengan mengatur next dari node sebelumnya menjadi null.
6. Metode delPos:

7. Menghapus simpul di posisi tertentu. Traverse list hingga mencapai posisi yang dimaksud. Jika node ditemukan, koneksi antar node diperbarui agar node tersebut keluar dari list. Jika node yang dihapus adalah head, maka head digeser ke node berikutnya.
8. Metode printList:
9. Mencetak semua data node dalam list dari head hingga akhir.
10. Fungsi main:

- Membuat DLL dengan lima node berisi data 1 sampai 5.
- Mencetak list awal.
- Menghapus node pertama (head), lalu mencetak list.
- Menghapus node di posisi ke-2 (dalam list yang sudah diubah), lalu mencetak list hasil akhirnya.

```
package pekan6;

public class 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;
	}

	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;
		}
		if(curr.prev != null) {
			curr.prev.next = null;
		}
		return head;
	}

	public static NodeDLL delPos(NodeDLL head, int pos) {
		if(head == null) {
			return head;
```

```
            }
            NodeDLL curr = head;
            for(int i = 1; curr != null && i < pos; ++i) {
                  curr = curr.next;
            }
            if(curr == null) {
                  return head;
            }
            if(curr.prev != null) {
                  curr.prev.next = curr.next;
            }
            if(curr.next != null) {
                  curr.next.prev = curr.prev;
            }
            if(head == curr) {
                  head = curr.next;
            }
            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("Karimah Irsdyiyah");
            System.out.println("2411533018");
            System.out.println();

            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("Menghapus node ke-2: ");
            head = delPos(head, 2);
            printList(head);
      }
}
```

Output:

```
Karimah Irsdyiyah

2411533018

DLL Awal: 1 2 3 4 5

Setelah head dihapus: 2 3 4 5

Menghapus node ke-2: 2 4 5
```

BAB III
PENUTUP

3.1 Kesimpulan

Melalui rangkaian praktikum yang mencakup pembuatan, penelusuran, penambahan, dan penghapusan simpul dalam Doubly Linked List (DLL), dapat disimpulkan bahwa:

1. DLL memiliki struktur dua arah, memungkinkan traversal maju (dari head ke tail) maupun mundur (dari tail ke head) dengan memanfaatkan pointer next dan prev.
2. Penambahan simpul dapat dilakukan dengan fleksibel, baik di awal, akhir, maupun posisi tertentu, selama pointer antar simpul disesuaikan dengan benar untuk menjaga integritas list.
3. Penghapusan simpul juga dapat dilakukan di berbagai posisi dengan memutus dan menghubungkan kembali pointer next dan prev dari simpul yang berdekatan.
4. Pemahaman logika pointer sangat penting karena kesalahan dalam mengatur next dan prev dapat menyebabkan list menjadi rusak (inaccessible node, loop tak berujung, atau null pointer error).
5. Setiap operasi manipulasi pada DLL memiliki dampak langsung terhadap struktur dan urutan data dalam list, sehingga harus dilakukan secara hati-hati dan sistematis.
6. Praktikum ini memberikan pemahaman mendalam terhadap implementasi dasar struktur data dan pentingnya logika traversal dan manipulasi node dalam pemrograman berorientasi objek.