

LAPORAN PRAKTIKUM STRUKTUR DATA  
IMPLEMENTASI SINGLE LINKEDLIST PADA  
PEMROGRAMAN JAVA



Oleh :

DERIEL CHAERAHMAN

NIM 2411533007

DOSEN PENGAMPU : DR. WAHYUDI, S.T, M.T  
ASISTEN PRAKTIKUM : RAHMAT DWIRIZKI OLDERS

FAKULTAS TEKNOLOGI INFORMASI  
DEPARTEMEN INFORMATIKA  
UNIVERSITAS ANDALAS

2025

## **A. Pendahuluan**

Praktikum ini dilakukan untuk membuat program yang dapat menyimpan/mengelola data dengan implementasi struktur data pada java yaitu Single LinkedList. Digunakan untuk situasi yang butuh manipulasi data secara dinamis (fleksibel menambah dan menghapus data dalam struktur datanya) tanpa harus menggeser elemen seperti pada array. Pengaplikasiannya dalam dunia nyata berupa navigasi maju dalam browser, undo/redo dalam aplikasi.

### **1. Single LinkedList (Senarai Berantai)**

Merupakan struktur data berantai di mana setiap elemen (node) menyimpan referensi ke elemen berikutnya. Terdapat 2 bagian utama yaitu data dan referensi ke berikutnya(next). Pada single linkedlist setiap nodenya menunjuk ke node berikutnya, dan node terakhir menunjuk ke null sebagai tanda akhir dari daftar. Cocok untuk operasi penambahan/penghapusan di tengah list, karena node-nodenya tidak disimpan secara berdekatan dalam memori.

Method :

- add() : Menambah elemen pada node
- remove() : Menghapus elemen pada node
- contains() : Mengecek isi node
- size() : Mengembalikan jumlah elemen dalam list
- display() : Menampilkan seluruh isi linkedlist.
- isEmpty() : Mengecek apakah list kosong

## **B. Tujuan**

Tujuan dari dilakukannya praktikum ini adalah :

1. Memahami dan mengaplikasikan Single LinkedList dalam program java untuk menyimpan data.
2. Implementasi method Single LinkedList pada program java.
3. Dapat mengaplikasikan statement (if, while), class (Scanner), dll. yang dipelajari sebelumnya ke dalam program.
4. Membuat class OOP(Object Oriented Programming) dengan implementasi Single LinkedList.

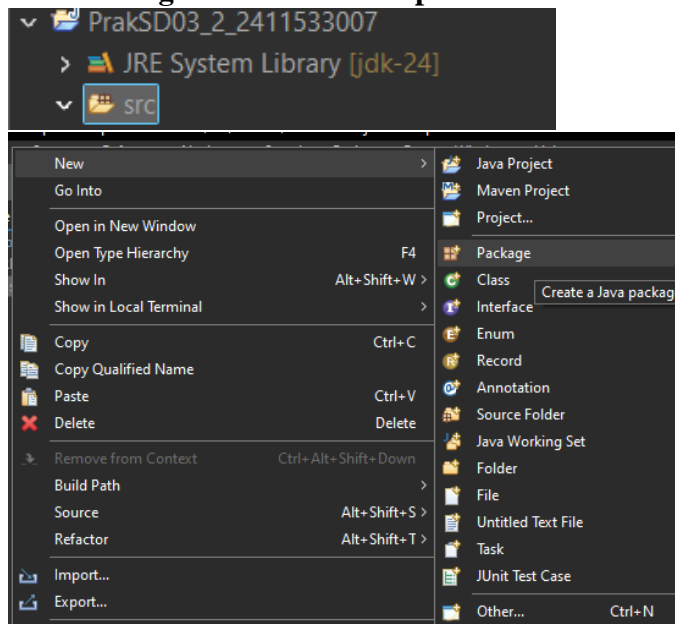
## **C. Langkah kerja praktikum**

### **a. Alat dan Bahan**

1. Perangkat computer atau laptop
2. Jaringan internet
3. IDE (Integrated Development Environment) direkomendasikan Eclipse IDE
4. Java JDK (Java Development Kit)

## b. Package pekan 5

1. Buat new package. **Buka java project** yang telah dibuat sebelumnya, lalu **klik kanan** pada folder 'src', setelahnya akan muncul list option, pilih 'new', lalu 'Package' dan beri nama 'pekan5'.



## c. Program NodeSLL

1. **Buat** terlebih dahulu new **class** dengan klik kanan pada package pekan5 dan beri nama "NodeSLL".

```
1 package pekan5;
2
3 public class NodeSLL {
```

2. Deklarasikan **variabel instance** integer "data", berfungsi untuk **menyimpan nilai/data** dari node.

```
4 //Node bagian data
5 int data;
```

3. Deklarasikan **variabel instance** bertipe **NodeSLL** "next". Berfungsi sebagai **pointer/referensi** ke node berikutnya dalam linkedlist.

```
6 //Pointer ke node berikutnya
7 NodeSLL next;
```

4. Membuat **konstruktor**, dengan parameter int data. Blok kodenya akan berjalan ketika objek dipanggil. 'this' sebagai setter untuk merujuk pada variabel instance pada kelas ini, yang menginisiasi variabel instance data dengan nilai dari parameter, dan variabel instance next dengan nilai null, artinya node ini tidak terhubung ke node lain.

```
8 //Konstruktor menginisialisasi node dengan data
9 public NodeSLL (int data) {
10     this.data = data;
11     this.next = null;
12 }
13 }
```

5. Penjelasan program : Merupakan program OOP. Mendefinisikan sebuah kelas NodeSLL yang merupakan blok pembangun (Building block) untuk struktur data single linkedlist. Setiap node punya 2 komponen utama yaitu data (untuk menyimpan nilai/data node) dan next (sebagai referensi/pointer ke node berikutnya). Terdapat konstruktor untuk membuat node baru dengan menginisialisasi data node dengan nilai yang diberikan, lalu pointer “next” di set ke null (artinya node ini awalnya tidak terhubung ke node lain).

#### d. Program TambahSLL

1. **Buat new class** dengan klik kanan pada package pekan5 dan beri nama public class “**TambahSLL**”.

```
1 package pekan5;
2
3 public class TambahSLL {
```

2. Membuat method insertAtFront, berfungsi untuk menambahkan node baru di awal linkedlist. Memiliki parameter NodeSLL dari class yang sebelumnya dibuat dengan atributnya “head” sebagai node pertama linkedlist, parameter value berfungsi sebagai nilai untuk node baru. Instansiasi node dari class NodeSLL dengan identifier “new\_node” dengan nilai nya diberikan ‘value’. Memanggil object new\_node untuk membuat node baru, lalu mengarahkan pointer ‘next’ ke node baru ‘head’. Lalu mengembalikan node baru sebagai ‘head’.

```
4 public static NodeSLL insertAtFront(NodeSLL head, int value) {
5     NodeSLL new_node = new NodeSLL(value);
6     new_node.next = head;
7     return new_node;
8 }
```

3. Membuat method insertAtEnd berfungsi untuk menambahkan node baru di akhir linked list. Memiliki parameter sama seperti insertAtFront. Membuat sebuah node dengan identifier “newNode” dengan sebuah nilai ‘value’, lalu memeriksa apakah list kosong (head sama dengan null) menggunakan if statement, yang lalu mengembalikan nilai newNode jika head tidak memiliki pointer saat ini.

```
10 //Fungsi menambahkan node di akhir SLL
11 public static NodeSLL insertAtEnd(NodeSLL head, int value) {
12     //buat sebuah node dengan sebuah nilai
13     NodeSLL newNode = new NodeSLL(value);
14     //jika list kosong maka node jadi head
15     if (head == null) {
16         return newNode;
17     }
```

4. Menyimpan head pointer ke variabel sementara, menelusuri hingga node akhir menggunakan while loop, lalu mengarahkan pointer ‘next’ ke node terakhir di node baru.

```

19     //Simpan head ke variabel sementara
20     NodeSLL last = head;
21     // Telusuri ke node akhir
22     while (last.next != null) {
23         last = last.next;
24     }
25
26     //ubah pointer
27     last.next = newNode;
28     return head;
29 }

```

5. Method GetNode, untuk membuat node baru dengan nilai tertentu. Memiliki parameter integer data. Akan mengembalikan node baru dengan nilai 'data'.

```

31● static NodeSLL GetNode (int data) {
32     return new NodeSLL(data);
33 }

```

6. Method insertPos, berfungsi untuk menyisipkan node baru pada posisi tertentu. Memiliki parameter 'headNode' yaitu Node pertama pada linkedlist, 'position' yaitu posisi penyisipan (dimulai dari 1), dan lalu 'value' yaitu nilai untuk node baru.

If statement untuk memeriksa posisi kurang dari 1, lalu if statement untuk memeriksa apakah posisi sama dengan 1 dan if statement untuk memeriksa jika posisi tidak sama dengan 1. Akhir dari node ini akan mengembalikan head.

```

35● static NodeSLL insertPos(NodeSLL headNode, int position, int value) {
36     NodeSLL head = headNode;
37     if (position < 1) {
38         System.out.print("Invalid position");
39     }
40     if (position == 1) {
41         NodeSLL new_node = new NodeSLL(value);
42         new_node.next = head;
43         return new_node;
44     } else {
45         while (position-- != 0) {
46             if (position == 1) {
47                 NodeSLL newNode = GetNode(value);
48                 newNode.next = headNode.next;
49                 headNode.next = newNode;
50                 break;
51             }
52             headNode = headNode.next;
53         }
54         if (position != 1) {
55             System.out.print("Posisi di luar jangkauan");
56         }
57     }
58     return head;
59 }

```

7. Membuat method printList, berfungsi untuk mencetak seluruh isi linked list. Memiliki parameter NodeSLL (dari class sebelumnya) head (atribut dari class NodeSLL) yang berfungsi sebagai node pertama linked list. Menelusuri linked list dari head, lalu menampilkan/sysout nilai dari setiap node dengan tambahan string.

```

61 public static void printList(NodeSLL head) {
62     NodeSLL curr = head;
63     while (curr.next != null) {
64         System.out.print(curr.data + "-->");
65         curr = curr.next;
66     }
67     if (curr.next == null) {
68         System.out.print(curr.data);
69     }
70     System.out.println();
71 }

```

8. Main method, Membuat object linked list dari class NodeSLL dengan identifier "head", terdiri dari node awalnya yaitu 2,3,4,5 yang setiap node nya dihubungkan atau pointer ke next node. Menampilkan/sysout list asli, lalu menambahkan node baru bernilai integer 1 diposisi depan menggunakan method insertAtFront. Menambahkan node baru dibelakang menggunakan method insertAtEnd. Menambahkan node pada posisi ke-4 menggunakan method insertPos. Menampilkan hasil akhir dari node yang tersimpan.

```

73 public static void main (String[] args) {
74     //buat linked list 2->3->4->5
75     NodeSLL head = new NodeSLL(2);
76     head.next = new NodeSLL(3);
77     head.next.next = new NodeSLL(5);
78     head.next.next.next = new NodeSLL(6);
79     //cetak list asli
80     System.out.print("Senarai berantai awal : ");
81     printList(head);
82     //tambahkan node baru di depan
83     System.out.print("Tambah 1 simpul di depan : ");
84     int data = 1;
85     head = insertAtFront(head, data);
86     //cetak update list
87     printList(head);
88     //Tambahkan node baru di belakang
89     System.out.print("Tambah 1 simpul di belakang : ");
90     int data2 = 7;
91     head = insertAtEnd (head, data2);
92     //cetak update list
93     printList(head);
94     System.out.print("Tambah 1 simpul ke data 4 : ");
95     int data3 = 4;
96     int pos = 4;
97     head = insertPos (head, pos, data3);
98     //cetak update list
99     printList(head);
100 }
101 }

```

9. Output program : Program ini menampilkan cara memanipulasi linkedlist dengan menggunakan method yang sudah dibuat sebelumnya, yaitu untuk menambahkan node di awal, di akhir dan di posisi yang spesifik. Mengabungkan konsep OOP dari class NodeSLL sebelumnya dibuat, object dan method untuk mengelola single linked list, yaitu memiliki pointer satu arah ke node selanjutnya.

```
Console x
<terminated> TambahSLL [Java Application] C:\Program Files\Java\jdk-24\bin\j
Senarai berantai awal : 2-->3-->5-->6
Tambah 1 simpul di depan : 1-->2-->3-->5-->6
Tambah 1 simpul di belakang : 1-->2-->3-->5-->6-->7
Tambah 1 simpul ke data 4 : 1-->2-->3-->4-->5-->6-->7
```

#### e. Program PencarianSLL

1. Buat new **class** dengan klik kanan pada package pekan 5 dan beri nama “PencarianSLL”.

```
1 package pekan5;
2
3 public class PencarianSLL {
```

2. Membuat method `searchKey`, berfungsi untuk mencari apakah suatu nilai terdapat dalam linkedlist, dengan mengembalikan nilai boolean. Memiliki parameter `NodeSLL head` dan integer `key`. Memulai pencarian dari `head`, menelusuri tiap node, membandingkan nilai ‘data’ tiap node dengan `key`, mengembalikan ‘true’ jika sama. Jika tidak ditemukan sampai akhir list maka akan dikembalikan ‘false’.

```
4 static boolean searchKey(NodeSLL head, int key) {
5     NodeSLL curr = head;
6     while (curr != null) {
7         if (curr.data == key) {
8             return true;
9         }
10        curr = curr.next;
11    }
12    return false;
13 }
```

3. Membuat method `traversal`, berfungsi untuk menelusuri dan mencetak semua nilai dalam linkedlist, memiliki parameter `NodeSLL head` sebagai node pertama linked list. Memulai dari `head`, `sysout` nilai dari tiap node, pindah ke node berikutnya lewat pointer ‘next’, berhenti ketika mencapai akhir list, yaitu `curr` sama dengan `null`.

```
15 public static void traversal (NodeSLL head) {
16     //mulai dari head
17     NodeSLL curr = head;
18     //Telusuri sampai pointer null
19     while (curr != null) {
20         System.out.print(" " + curr.data);
21         curr = curr.next;
22     }
23     System.out.println();
24 }
```

4. Main method, membuat node awal `head` bernilai 14, disambung dengan `nnode` berikutnya diikuti pembuatan ‘next pointer’ yaitu node 21, 13, 30 dan terakhir

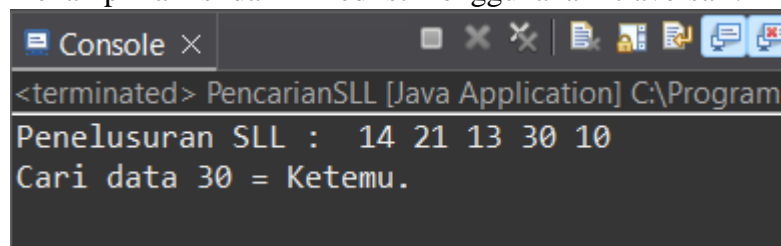
10. Sysout isi dari linkedlist menggunakan traversal. Memasukan data yang ingin dicari pada variabel key yaitu nilai 30, menggunakan method searchKey dengan parameter head dan key pada if statement, jika true maka ketemu dan jika if stament false maka tidak ada data tersebut yang ingin dicari tidak terdapat pada linkedlist.

```

26● public static void main (String[] args) {
27     NodeSLL head = new NodeSLL(14);
28     head.next = new NodeSLL(21);
29     head.next.next = new NodeSLL(13);
30     head.next.next.next = new NodeSLL(30);
31     head.next.next.next.next = new NodeSLL(10);
32     System.out.print("Penelusuran SLL : ");
33     traversal(head);
34     //data yang akan dicari
35     int key = 30;
36     System.out.print("Cari data " + key + " = ");
37     if (searchKey(head, key)) {
38         System.out.println("Ketemu.");
39     } else {
40         System.out.println("Tidak ada.");
41     }
42 }
43 }

```

5. Penjelasan program : Program ini terdiri dari 2 fungsi utama yaitu seacrhKey untuk mencacri nilai tertentu dalam linkes list dengan menelusuri tiap node, dan fungsi traversal untuk mencetak semua data dalam linkedlist. Pada method main dibuat linked list berisi nilai 14, 21, 13, 30, dan 10, lalu menggunakan method ‘seacrhKey’ untuk mencari suatu nilai tersimpan pada linked list dan menampilkan isi dari linkedlist menggunakan ‘traversal’.



```

<terminated> PencarianSLL [Java Application] C:\Program
Penelusuran SLL : 14 21 13 30 10
Cari data 30 = Ketemu.

```

#### f. Program HapusSLL

1. Buat new **class** dengan klik kanan pada package pekan5 dan beri nama “**HapusSLL**”.
2. Membuat **method deleteHead**, bergungsi untuk menghapusnode pertama (head) dari linked list. Memiliki parameter NodeSLL head untk node pertama linkedlist. Mengembalikan node baru yang menjadi head (node kedua sebelumnya), memeriksa jila list kosong maka dikembalikan null, jika tidak pindahkan head ke node berikutnya, lalu kembalikan head yang baru.



```

1 package pekan5;
2
3 public class HapusSLL {
4     //Fungsi untuk menghapus head
5     public static NodeSLL deleteHead(NodeSLL head) {
6         //jika SLL kosong
7         if (head == null) {
8             return null;
9         }
10        //pindahkan head ke node berikutnya
11        head = head.next;
12        //Return head baru
13        return head;
14    }

```

3. Membuat method removeLastNode, berfungsi untuk menghapus node terakhir dari linkedlist, memiliki parameter 'head' sebagai node pertama linked list. Mengembalikan head list yang sama, jika list kosong atau hanya 1 node, mengembalikan null. Menemukan node terakhir ke dua, maka set pointer next node kedua terakhir ke null.

```

16 //Fungsi menghapus node terakhir SLL
17 public static NodeSLL removeLastNode(NodeSLL head) {
18     //jika list kosong, return null
19     if (head == null) {
20         return null;
21     }
22     //jika list satu node, hapus node dan return null
23     if (head.next == null) {
24         return null;
25     }
26     //temukan node terakhir ke dua
27     NodeSLL secondLast = head;
28     while (secondLast.next.next != null) {
29         secondLast = secondLast.next;
30     }
31     //Hapus node terakhir
32     secondLast.next = null;
33     return head;
34 }

```

4. Membuat method deleteNode, berfungsi untuk menghapus node pada posisi tertentu. Memiliki parameter head dan position. Mengembalikan head list. Menghandle list kosong, lalu menghapus head pada posisi sama dengan 1, menelusuri list sampai posisi yang ditentukan, menghubungkan node sebelumnya dengan node setelahnya/prev, jika posisi tidak ditemukan maka sysout eror message.

```

36 //fungsi menghapus node di posisi tertentu
37● public static NodeSLL deleteNode(NodeSLL head, int position) {
38     NodeSLL temp = head;
39     NodeSLL prev = null;
40     //jika linked list null
41     if (temp == null) {
42         return head;
43     }
44     //kasus 1 : head dihapus
45     if (position == 1) {
46         head = temp.next;
47         return head;
48     }
49     //kasus 2 : menghapus node yang ditengah
50     //telusuri ke node yang dihapus
51     for (int i = 1; temp != null && i < position; i++) {
52         prev = temp;
53         temp = temp.next;
54     }
55     //jika ditemukan, hapus node
56     if (temp != null) {
57         prev.next = temp.next;
58     } else {
59         System.out.println("Data tidak ada.");
60     }
61     return head;
62 }

```

5. Membuat method printList, berfungsi untuk mencetak isi linked list. Memiliki parameter head sebagai node pertama linked list. Menghasilkan sysout semua node dengan tambahan tanda string.

```

64 //Fungsi mencetak SLL
65● public static void printList (NodeSLL head) {
66     NodeSLL curr = head;
67     while (curr.next != null) {
68         System.out.print(curr.data + "-->");
69         curr = curr.next;
70     }
71     if (curr.next == null) {
72         System.out.print(curr.data);
73     }
74     System.out.println();
75 }

```

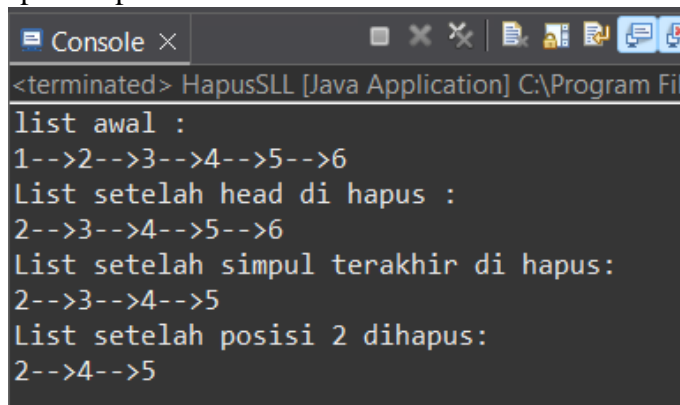
6. Main method, inialisasi linked list dengan nilai node awal 1, dilanjutkan dengan node selanjutnya yaitu 2 sekaligus dengan pemberian pointer, dan terus berlanjut sampai node dengan nilai 6. Menampilkan isi dari node menggunakan method printList dengan parameter head untuk pengurutan penampilan node dimulai dari head/ node awal. Menggunakan method deleteHead dengan parameter head untuk menghapus node paling depan. Menggunakan method removeLastNode parameter head, untuk menghapus node pada posisi paling belakang. Menggunakan method deleteNode dengan spesifik node ke 2 setelah head dihapus.

```

77● public static void main (String[] args) {
78     //buat SLL 1->2->3->4->5->6 -> null
79     NodeSLL head = new NodeSLL(1);
80     head.next = new NodeSLL (2);
81     head.next.next = new NodeSLL (3);
82     head.next.next.next = new NodeSLL (4);
83     head.next.next.next.next = new NodeSLL(5);
84     head.next.next.next.next.next = new NodeSLL (6);
85     // cetak list awal
86     System.out.println("list awal : ");
87     printList(head);
88     // hapus head
89     head = deleteHead(head);
90     System.out.println("List setelah head di hapus : ");
91     printList(head);
92     //hapus node terakhir
93     head = removeLastNode (head);
94     System.out.println("List setelah simpul terakhir di hapus: ");
95     printList (head);
96     // Deleting node at position 2
97     int position = 2;
98     head = deleteNode (head, position);
99     // Print list after deletion
100    System.out.println("List setelah posisi 2 dihapus: ");
101    printList(head);
102 }
103 }

```

7. Output program : Program ini terdapat 3 method yaitu deleteHead untuk menghapus ndoe pertama, removeLastNode untuk mengpaus node terakhir dan deleteNode untuk meghapus node pada posisi tertentu, serta method printList untuk menampilkan semua node pada linked list. Pada main method dibuat 6 buah node, lalu menggunakan masing-masing method untuk didemontarasikan. Mencangkup penghapusan diawal, di akhir, dan lokasi spesifik pada node.



```

Console x
<terminated> HapusSLL [Java Application] C:\Program Fil
list awal :
1-->2-->3-->4-->5-->6
List setelah head di hapus :
2-->3-->4-->5-->6
List setelah simpul terakhir di hapus:
2-->3-->4-->5
List setelah posisi 2 dihapus:
2-->4-->5

```

#### D. Kesimpulan

Setelah melakukan praktikum ini dapat memahami dan mengimplementasikan Single LinkedList (Senarai Berantai tunggal) dan methodnya. Membuat program dengan konsep OOP seperti NodeSLL, memabut method dengan implementasi unutk memanipulasi node pada single linkedlist. Dapat diterapkan seperti pencatatan data

dinamis, dimana kemudahan penambahan dan penghapusan data. Mempelajari cara menambahkan data dalam single linked list berupa mengarahkan pointer 'next'.