MAKALAH IMPLEMENTASI

STRUKTUR DATA LINKEDLIST

Dosen Pengampu:

Rizqi Putri Nourma Budiarti, S.T., M.T

Nama: Galeh Ariya Irwana

NIM: 3130021003

PRODI SISTEM INFORMASI

FAKULTAS EKONOMI BISNIS DAN TEKNOLOGI DIGITAL

UNIVERSITAS NAHDLATUL ULAMA' SURABAYA

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Linked List atau dikenal juga dengan sebutan senarai berantai adalah struktur data yang terdiri dari urutan record data, dimana setiap record memiliki field yang menyimpan alamat/referensi dari record selanjutnya. Element data yang dihubungkan dengan link pada Linked List disebut Node. Biasanya di dalam LinkedList terdapat istilah Head, Tail, Next, Previous. Dalam setiap Bahasa pemrograman mungkin sudah ada atau sudah disediakan implementasi struktur data Linked List ini, namun kita juga perlu tahu cara untuk mengimplementasikannya secara manual, agar dapat memahami bagaimana cara kerja atau perilaku dari struktur data Linked List ini.

1.2 Rumusan Masalah

1.2.1 Bagaimana contoh implementasi Struktur Data LinkedList?

1.2.2 Bagaimana contoh implementasi Struktur Double LinkedList?

1.2.3 Bagaimana contoh implementasi Struktur Data Circular LinkedList?

1.2.4 Bagaimana contoh implementasi Struktur Data Multiple LinkedList?

1.3 Tujuan

1.3.1 Mengetahui cara kerja Struktur Data LinkedList

1.3.2 Mengetahui cara kerja Struktur Data Double LinkedList

1.3.3 Mengetahui cara kerja Struktur Data Circular LinkedList

1.3.4 Mengetahui cara kerja Struktur Data Multiple LinkedList

1.4 Manfaat

1.4.1 Memberikan pemahaman mengenai logika dan juga cara kerja dari Struktur Data LinkedList

1.4.2 Mampu mengimplementasikan Struktur Data LinkedList

BAB II
PEMBAHASAN

2.1 Implementasi Linked List

Untuk membuat sebuah Implementasi LinkedList atau SingleLinkedList kita akan membuat tiga class yaitu Node, LinkedListImpl, dan SingleListApp. Dimana class Node ini akan berfungsi sebagai tempat datanya, mulai dari mengambil data dan mengubah data, atau istilahnya bisa kita sebut setter dan getter pada class Node ini.Tidak lupa kita juga harus membuat constructor untuk memasukkan data ke dalam sebuah field atau attribute.

```
3   public class Node {
4       private String data;
5       private Node link;
6
```

2.1 Gambar Class

```
7
8       public Node() {
9           link = null;
10          data = null;
11      }
12
13      public Node(String data, Node link) {
14          this.data = data;
15          this.link = link;
16      }
```

2.2 Gambar Constructor

```
21
22      public void setData(String data) { this.data = data; }
25
26      public Node getLink() { return link; }
29
30      public void setLink(Node link) { this.link = link; }
33  }
34
```

2.3 Gambar Setter and Getter

Kemudian untuk melakukan operasi tambah data, remove data, dan yang lainnya kita bisa menggunakan class LinkedListImpl. Class inilah yang akan berfungsi sebagai tempat semua logic operasi dilakukan.

```java
public class LinkedListImpl {
    private Node start;
    private Node end;
    private int size;

    public LinkedListImpl() {
        start = null;
        end = null;
        size = 0;
    }
```

2.4 Contoh code class LinkedListImpl

Kita tahu sebelumnya dalam class ini akan berisi operasi untuk menyimpan semua logic jadi kita akan menambahkan beberapa function, yang pertama kita akan tambahkan function isEmpty dan juga getSize, dimana isEmpty ini akan mengecek apakah data di linkedlist itu ada isinya atau tidak dan function ini akan mengembalikan nilai Boolean. Kemudian function getSize berfungsi sebagai pengambilan ukuran data yang ada dalam LinkedList ini.

```java
    public boolean isEmpty() { return start == null; }

    public int getSize() { return size; }
```

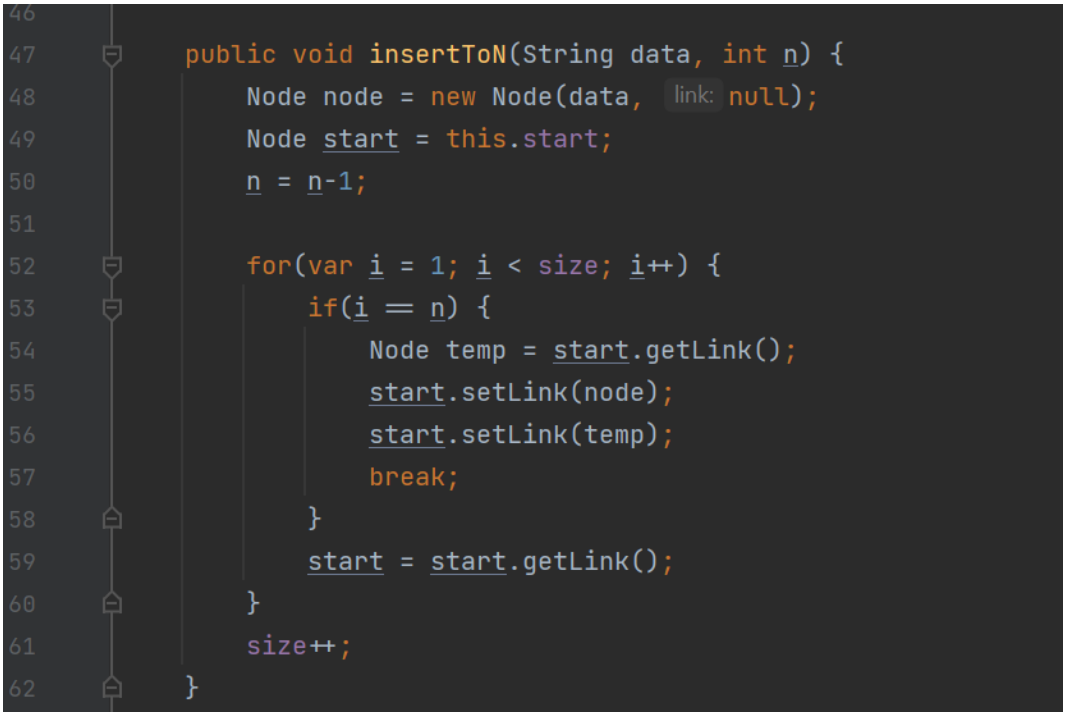
2.5 Gambar kode function isEmpty and getSize

Kemudian selanjutnya kita juga butuh sebuah function untuk menambah data dari posisi depan dan juga posisi akhir, jadi datanya yang kita masukkan nanti dapat diletakkan di bagian awal sendiri dan bisa juga di bagian paling akhir dalam sebuah node.

```java
    public void insertStart(String data) {
        Node node = new Node(data, link: null);
        size++;

        if(start == null) {
            start = node;
            end = start;
        }
    }
```

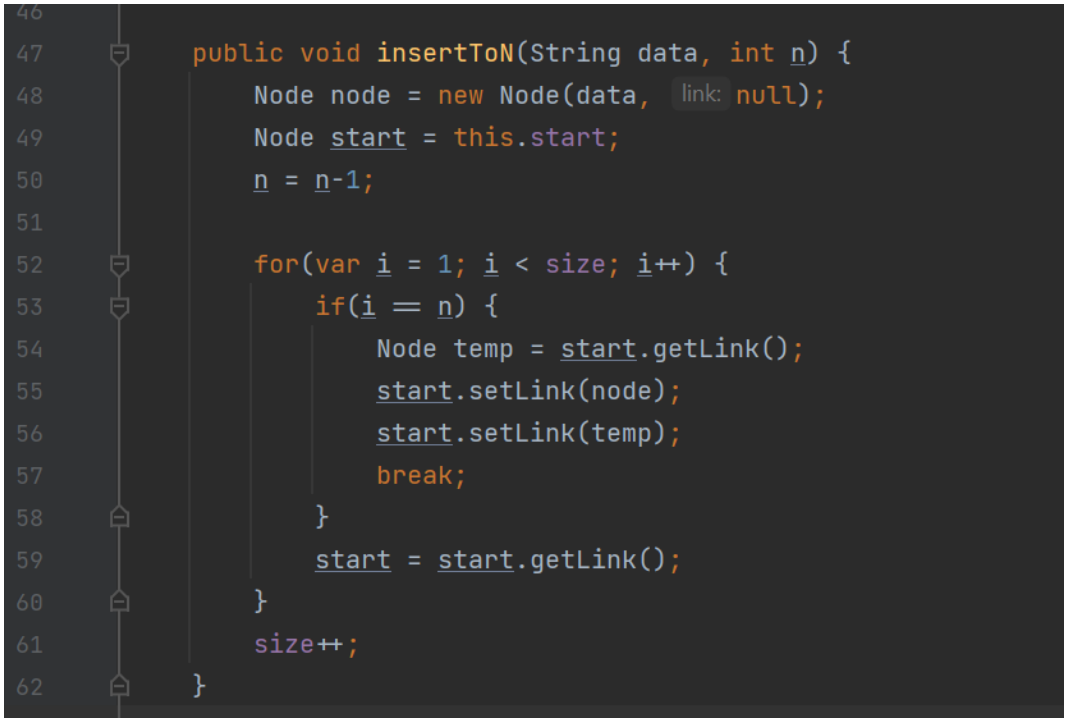
2.6 Gambar Code addFirst



```
47      public void insertToN(String data, int n) {
48          Node node = new Node(data,  link: null);
49          Node start = this.start;
50          n = n-1;
51
52          for(var i = 1; i < size; i++) {
53              if(i == n) {
54                  Node temp = start.getLink();
55                  start.setLink(node);
56                  start.setLink(temp);
57                  break;
58              }
59              start = start.getLink();
60          }
61          size++;
62      }
```

2.7 Gambar Kode addLast

Selain dari insert atau memasukkan data dari awal ataupun dari akhir, tentu kita juga butuh memasukkan data di tengah-tengah atau datanya itu disisipkan. Kita akan membuat function insertToN dimana function ini akan menerima dua parameter yang pertama adalah datanya dan yang kedua adalah data tersebut mau dimasukan ke dalam urutan yang mana.



```
47      public void insertToN(String data, int n) {
48          Node node = new Node(data,  link: null);
49          Node start = this.start;
50          n = n-1;
51
52          for(var i = 1; i < size; i++) {
53              if(i == n) {
54                  Node temp = start.getLink();
55                  start.setLink(node);
56                  start.setLink(temp);
57                  break;
58              }
59              start = start.getLink();
60          }
61          size++;
62      }
```

2.8 Gambar kode addAtN

Setelah memberikan function menambahkan pada class ini, kita juga akan menambahkan dua function lagi yaitu remove dan juga display. Function remove akan digunakan untuk menghapus data yang ada dalam node dan fungsi hapus ini juga akan menerima sebuah parameter, dimana data dari parameter ini akan digunakan untuk menentukan urutan keberapa data yang akan dihapus. Kemudian juga function display, function display digunakan sebagai informasi dari data yang ada dalam node. Gambarannya apabila data nya kosong apa yang ditampilkan atau informasi yang lain mengenai data yang ada dalam node.

```
64
65      public void removeToN(int n) {
66          if(n == 1) {
67              start = start.getLink();
68              size--;
69              return;
70          }
71
72          if(n == size) {
73              Node s = start;
74              Node e = start;
75
76              while (s ≠ end) {
77                  e = s;
78                  s = s.getLink();
79              }
80
81              end = e;
82              end.setLink(null);
83              size--;
84              return;
85          }
```

```
86
87          Node node = start;
88          n = n-1;
89
90          for(var i = 1; i < size; i++) {
91              if(i == n) {
92                  Node temp = node.getLink();
93                  temp = temp.getLink();
94                  node.setLink(temp);
95                  break;
96              }
97
98              node.getLink();
99          }
100
101         size--;
102     }
```

2.9 Gambar kode removeToN

```
104     public void display() {
105         if(size == 0) {
106             System.out.println("Kosong\n");
107             return;
108         }
109
110         if(start.getLink() == null) {
111             System.out.println(start.getData());
112             return;
113         }
114
115         Node srt = start;
116         System.out.println(start.getData());
117         srt = start.getLink();
118
119         while (srt.getLink() != null) {
120             System.out.println(srt.getData());
121             srt.getLink();
122         }
123         System.out.println(srt.getData() + "\n");
124     }
```

2.10 Gambar kode function display

Sampai disini kita sudah membuat dua class yaitu Node sebagai representasi datanya dan juga LinkedListImpl sebagai business logic dari operasi yang akan kita lakukan terhadap datanya. dan yang terakhir kita butuh yang sebuah class lagi yaitu SingleListApp untuk menjalankan semua class yang akan kita buat, dalam istilah ini class ini disebut main class. Didalam class ini hanya akan ada satu method atau function yang bernama main.

```java
import java.util.Scanner;

public class SingleListApp {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        LinkedListImpl list = new LinkedListImpl();
        System.out.println("Let's try Single List\n");

        char code;

        do {
            System.out.println("Perintah\n");
            System.out.println("1. Masukkan dari awal");
            System.out.println("2. Masukkan dari akhir");
            System.out.println("3. Masukkan dari ke N");
            System.out.println("4. Hapus posisi");
            System.out.println("5. Cek Kosong");
            System.out.println("6. Cek Size");

            System.out.print("\nMasukkan nomor:");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1 -> {
                    System.out.println("Masukkan Data");
                    list.insertStart(scanner.nextLine());
                    break;
                }
                case 2 -> {
                    System.out.println("Masukkan Data");
                    list.insertEnd(scanner.nextLine());
                    break;
                }
                case 3 -> {
                    System.out.println("Masukkan Data");
                    String data = scanner.nextLine();
                    System.out.println("Enter Position");
                    int n = scanner.nextInt();

                    if(n < 1 || n > list.getSize()) {
                        System.out.println("Posisi tidak valid\n");
                    } else {
                        list.insertToN(data, n);
                        break;
                    }
                }
```

```
            case 4 -> {
                System.out.println("Masukkan Posisi");
                int p = scanner.nextInt();

                if(p < 1 || p > list.getSize()) {
                    System.out.println("Position tidak valid\n");
                } else  {
                    list.removeToN(p);
                    break;
                }
            }
            case 5 -> {
                System.out.println("Cek kosong: " + list.isEmpty());
                break;
            }
            case 6 -> {
                System.out.println("Size: " + list.getSize() + "\n");
                break;
            }
        }
        list.display();
        System.out.println("\nMau Lanjut? (type y or n)");
        code = scanner.next().charAt(0);

    } while (code == 'Y' || code == 'y');
```

2.11 Gambar kode Main

2.2 Implementasi DoubleLinkedList

DoubleLinkedList adalah list yang memiliki dua variable pointer, yaitu pointer yang menuju ke node selanjutnya, ini bisa kita sebut next dan juga pointer yang menuju ke node sebelumnya, ini juga bisa kita sebut preview, dimana head dan tailnya menunjukkan ke NULL. Kali ini kita akan mencoba mengimplementasikan struktur data Double linkedlist dimana kita akan membuat implementasi ini, menerapkan konsep Object Oriented Programming (OOP). Pertama kita akan butuh tiga kelas, kelas yang pertama sebagai kelas representasi dari sebuah data, kemudian yang kedua kita butuh kelas untuk menampung semua business logic untuk melakukan operasi terhadap data. Dan yang terakhir kita juga butuh sebuah class sebagai tempat menjalankan semua aplikasi kita atau bisa kita sebut main program. Kita akan buat class NodeDouble sebagai representasi data yang kita miliki, dimana class ini akan berisi attribute data, linknext, dan juga linkprevious, dan tidak lupa kita tambahkan constructor didalamnya.

```java
public class NodeDouble {
    private String data;
    private NodeDouble linkNext;
    private NodeDouble linkPrev;

    public NodeDouble(String data) {
        this.data = data;
    }
```

2.12 Gambah kode class NodeDouble

Di dalam class ini juga kita akan buat sebuah function setter dan getter dari semua field yang kita miliki.

```java
    public String getData() { return data; }

    public void setData(String data) { this.data = data; }

    public NodeDouble getLinkNext() { return linkNext; }

    public void setLinkNext(NodeDouble linkNext) { this.linkNext = linkNext; }

    public NodeDouble getLinkPrev() { return linkPrev; }

    public void setLinkPrev(NodeDouble linkPrev) { this.linkPrev = linkPrev; }
}
```

2.13 Gambar kode Setter and getter

Sampai sini kita telah selesai membuat class untuk representasi data yang kita punya, selanjutnya kita akan buat class untuk logic memanipulasi data seperti menambah, dari data yang kita punya. Disini kita buat class DoubleLinkedListImpl, dalam class ini selain menyimpan function untuk logic manipulasi data, dalam class kita juga butuh 2 field head dan tail.

```java

public class DoubelLinkedListImpl {
    NodeDouble head, tail = null;

```

2.14 Gambar Class DoubleLinkedListImpl

Setelah kita buat class kita akan membuat dua function, yang pertama adalah untuk memasukkan data dan yang kedua adalah function printNoudes function ini berfungsi untuk print info terkait nodes apakah ketika dijalankan node atau datanya kosong, atau jika ada maka akan ditampilkan dengan tersusun rapi.

```java
    public void insertNode(String data) {
        NodeDouble node = new NodeDouble(data);

        if(head == null) {
            head = tail = node;
            head.setLinkPrev(null);
            tail.setLinkNext(null);
        }
        else {
            tail.setLinkNext(node);
            node.setLinkPrev(tail);

            tail = node;
            tail.setLinkNext(null);
        }
    }
```

2.15 Gambar function Insert

```java
    public void printNodes() {
        NodeDouble current = head;
        if(head == null) {
            System.out.println("Doubly linked list is empty");
            return;
        }
        System.out.println("Nodes of doubly linked list: ");
        while(current != null) {
            System.out.print(current.getData() + " ");
            current = current.getLinkNext();
        }
    }
}
```

2.16 Gambar Function Print

Dari class DoubleLinkedListImpl ini kita hanya akan menambahkan function ini saja, selanjutnya Langkah terakhir yaitu membuat sebuah class dan didalamnya terdapat sebuah main method untuk menjalankan kode program yang telah kita buat. dan didalam mian function atau main method ini kita tinggal membuat object dari class DoubleLinkedListImpl setelah itu kita bisa melakukan pemanggilan method dari class DoubleLinkedListImpl untuk melakukan sebuah operasi tambah data.

```java
public class DoubleLingkedListApp {
    public static void main(String[] args) {

        DoubelLinkedListImpl doubleList = new DoubelLinkedListImpl();

        doubleList.insertNode( data: "Galeh");
        doubleList.insertNode( data: "Ariya");
        doubleList.insertNode( data: "Irwana");

        doubleList.printNodes();

    }
}
```

2.17 Gambar Main function DoubleLinkedListApp

2.3 Implementasi CircularLinkedList

Jika kita tahu bahwa SingleLinkedList dan juga DoubleLinkedList tailnya mengarah ke null, berbeda dengan CircularLinkedList, tail dari CircularLinkedList ini mengarah ke head. Kita akan mencoba implementasi dari CircularLinkedList ini, konsep yang akan digunakan adalah konsep Object Oriented Programming atau (OOP). Pertama kita akan buat class sebagai class yang merepresentasikan sebuah data, didalam class ini terdapat beberapa field diantaranya data yang bertipe data String dan juga next. Di dalam class ini juga akan ada sebuah constructor dan function setter dan getter dari field yang ada.

```java
public class CircularNode {
    String data;
    CircularNode next;

    public CircularNode(String data) {
        this.data = data;
    }
```

2.18 Gambar kode Class CircularNode

```
    public String getData() { return data; }

    public void setData(String data) { this.data = data; }

    public CircularNode getNext() { return next; }

    public void setNext(CircularNode next) { this.next = next; }
}
```

2.19 Gambar kode setter and getter CircularNode

Selanjutnya kita akan buat class sebagai tempat logic kita memanipulasi data yang kita punya, class ini akan berisi field head dan tail, kemudian ada dua function yang akan kita buat dalam class ini, yang pertama adalah function insert untuk menambahkan data kedalam node kemudian ada function display sebagai fungsi menampilkan data dalam node, dalam fungsi display ini juga terdapat sebuah pengecekan apabila node nya kosong atau tidaknya sehingga ketika kita menjalankan program kita tahu apakah node nya kosong atau sudah terisi.

```
public class CircularDoubleLinkedListImpl {
    private CircularNode head, tail = null;
```

2.20 Gambar class CirucularDoubleLinkedListImpl

```
    public void insert(String data) {
        CircularNode cnode = new CircularNode(data);

        if(head == null) {
            head = cnode;
            tail = cnode;
            cnode.setNext(head);
        } else  {
            tail.setNext(cnode);
            tail = cnode;
            tail.setNext(head);
        }
    }
```

2.21 Gambar function insert

```
    public void display() {
        CircularNode current = head;

        if(head == null) {
            System.out.println("List is Empty");
        } else {
            System.out.println("Nodes of the circular linked list: ");

            do{
                System.out.print(current.getData());
                current = current.getNext();
            }while(current != head);
            System.out.println();
        }
    }

}
```

2.22 Gambar kode function display

Langkah terakhir kita buat sebuah class dimana di dalam class ini terdapat satu buah method atau function namanya main function atau main method. Di dalam main method ini kita akan menginstansiasi atau membuat object dari class CircularDoubleLinkedlist yang sudah kita buat di atas. Setelah membuat object tersebut kita akan melakukan pemanggilan function insert melalui object yang telah kita buat dan terakhir kita juga panggil function display untuk menampilkan data yang telah kita masukkan.

```
public class CircularDoubleLinkedListApp {
    public static void main(String[] args) {

        CircularDoubleLinkedListImpl cnode = new CircularDoubleLinkedListImpl();

        cnode.insert( data: "Galeh");
        cnode.insert( data: "Ariya");
        cnode.insert( data: "Irwana");

        cnode.display();

    }
}
```

2.23 Gambar main fincton CircularDoubleLinkedListApp

2.4 Implementasi MultipleLinkedList

Kita akan mencoba untuk implementasi MultipleLinkedList. MultipleLinkedlist merupakan list yang memiliki lebih dari dua pointer. Dalam kasus ini kita akan membuat sebuah class untuk implementasi MultipleLinkedList ini yang isinya

adalah sebuah inner class yang bernama Node dan inner class ini memiliki field data, next dan child.

```
class MultiNode {

    public static class Node {
        String data;
        Node next;
        Node child;
    }
```

2.24 Gambar class MultiNode dan inner class

Selanjutnya dalam class MultiNode kita akan buat sebuah function CreateList yang mereturn sebuah data Node dan juga function ini menerima sebuah parameter array dan n. sesuai dengan nama functionnya didalam function ini ada logic untuk membuat sebuah list.

```
    public static Node createList(String arr[], int n) {
        Node head = null;
        Node tmp = null;

        for (int i = 0; i < n; i++) {
            if (head == null) {
                tmp = head = new Node();
            }
            else {
                tmp.next = new Node();
                tmp = tmp.next;
            }
            tmp.data = arr[i];
            tmp.next = tmp.child = null;
        }
        return head;
    }
```

2.25 Gambar function createList

Kemudian kita akan menambahkan juga sebuah function print, gunanya adalah untuk print data dalam node ini. Isi dari function ini simple hanya mengecek apakah headnya itu kosong atau tidak jika kosong maka tidak akan ditampilkan apa apa.

```
28
29      public static void printMultiLevelList(Node head)
30      {
31
32          while (head ≠ null) {
33              if (head.child ≠ null) {
34                  printMultiLevelList(head.child);
35              }
36              System.out.print(head.data + " ");
37              head = head.next;
38          }
39      }
40
```

2.26 Gambar function print

Dan terakhir kita buat main function tempat kita menjalankan semua kode yang kita buat. isi dari main function ini adalah array tipe datanya string kemudian kita juga buat sebuah variable yang isinya kita panggil function createlist, karena function create list menerima parameter jadi kita kirimkan array dan juga angka n nya tadi. kemudian tinggal kita call function print

```
40
41      public static void main(String[] args)
42      {
43          String arr1[] = { "Galeh", "Ariya", "Irwana" };
44          String arr2[] = { "Dea", "Ayu", "Puspita" };
45          String arr3[] = { "Arcive" };
46          String arr4[] = { "Adek", "Citra", "Larasati" };
47
48          Node head1 = createList(arr1, n: 3);
49          Node head2 = createList(arr2, n: 2);
50          Node head3 = createList(arr3, n: 1);
51          Node head4 = createList(arr4, n: 3);
52
53          head1.child = head2;
54          head1.next.next.child = head3;
55          head2.next.child = head4;
56
57          Node head = null;
58          head = head1;
59
60          printMultiLevelList(head);
61      }
```

2.27 Gambar Main function untuk multiplelinkedlits

BAB 3
PENUTUP

3.1 Kesimpulan

Dalam struktur data ada empat jenis linkedlist diantaranya adalah SingleList, DoubleLinkedList, CircularLinkedList dan yang terakhir MultipleLinkedlist. Perbedaan dari setiap linkedlist ini adalah dari pointernya, misalnya singlelist hanya memiliki satu pointer yang mengarah ke null, kemudian DoubleLinkedList memiliki dua pointer, pointer yang pertama mengarah ke node selanjutnya ini bis akita sebut next dan jga pointer yang mengarah ke node sebelumnya yang bisa kita sebut preview. Kemudian CircularLinkedList, CircularLinkedList ini memiliki pointer tailnya akan mengarah ke headnya, dan yang terakhir adalah MutipleLinkedList, MutipleLinkedList ini memiliki lebih dari 2 buah pointer. Implementasi dari masing masing linkedlist ini tentu saja berbeda, namun pada intinya pada linkedlist ini kita hanya akan bermain main pada pointernya saja, bagaimana kita memanipulasi data menggunakan pointernya tersebut.

3.2 Saran

Kedepannya saya berharap agar materi mengenai MultipleLinkedlist ini semakin banyak, karena saya rasa dari sekian banyak saya browsing tentang materi MultipleLinkedList ini, pembahasan yang diberikan sangatlah sedikit.