

LAPORAN PRAKTIKUM
MATA KULIAH PRAKTIKUM ALGORITMA STRUKTUR DATA

Dosen Pengampu : Triana Fatmawati, S.T, M.T

PERTEMUAN 11 : LINKED LIST



Nama : Yonanda Mayla Rusdiaty

NIM : 2341760184

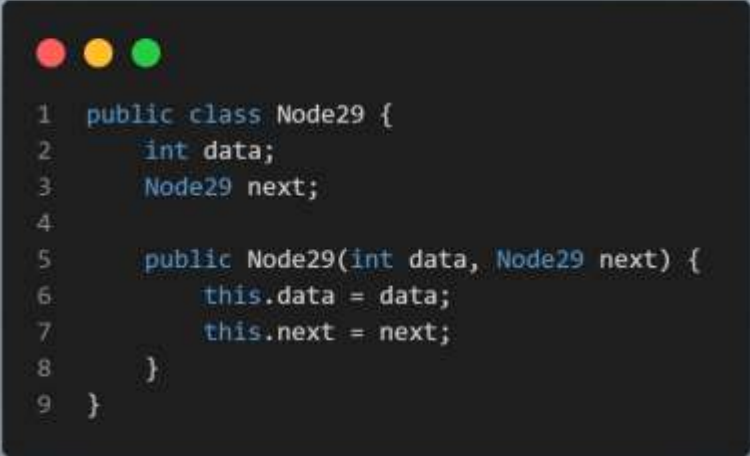
Prodi : D-IV Sistem Informasi Bisnis

JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG

2024

PRAKTIKUM 1

Kode program class Node29 :

A screenshot of a code editor window with a dark background and light-colored text. The code is written in Java and defines a class named Node29. The code is as follows:

```
1 public class Node29 {  
2     int data;  
3     Node29 next;  
4  
5     public Node29(int data, Node29 next) {  
6         this.data = data;  
7         this.next = next;  
8     }  
9 }
```

The code is numbered from 1 to 9 on the left side of the editor. The class has two attributes: an integer named 'data' and a reference to another Node29 object named 'next'. It also has a constructor that takes an integer 'data' and a Node29 object 'next' as parameters and initializes the attributes using 'this'.

Code program class LinkedList29 :

```
1 public class LinkedList29 {
2     Node29 head;
3
4     public boolean isEmpty() {
5         return (head == null);
6     }
7
8     public void print() {
9         if (!isEmpty()) {
10             System.out.print("Isi linked list : ");
11             Node29 currentNode = head;
12
13             while (currentNode != null) {
14                 System.out.print(currentNode.data + "\t");
15                 currentNode = currentNode.next;
16             }
17
18             System.out.println(" ");
19         } else {
20             System.out.println("linked list kosong");
21         }
22     }
23
24     public void addFirst(int input) {
25         Node29 newNode = new Node29(input, null);
26
27         if (isEmpty()) {
28             head = newNode;
29         } else {
30             newNode.next = head;
31             head = newNode;
32         }
33     }
34
35     public void addLast(int input) {
36         Node29 newNode = new Node29(input, null);
37
38         if (isEmpty()) {
39             head = newNode;
40         } else {
41             Node29 currentNode = head;
42
43             while (currentNode.next != null) {
44                 currentNode = currentNode.next;
45             }
46
47             currentNode.next = newNode;
48         }
49     }
50
51     public void insertAfter(int key, int input) {
52         Node29 newNode = new Node29(input, null);
53
54         if (!isEmpty()) {
55             Node29 currentNode = head;
56
57             do {
58                 if (currentNode.data == key) {
59                     newNode.next = currentNode.next;
60                     currentNode.next = newNode;
61                     break;
62                 }
63                 currentNode = currentNode.next;
64             } while (currentNode != null);
65         } else {
66             System.out.println("linked list kosong");
67         }
68     }
69 }
70 }
71 }
```

Kode program class SSLMain29 :

```
1 public class SSLMain29 {
2     public static void main(String[] args) {
3         LinkedList29 myLinkedList29 = new LinkedList29();
4         myLinkedList29.print();
5         myLinkedList29.addFirst(800);
6         myLinkedList29.print();
7         myLinkedList29.addFirst(700);
8         myLinkedList29.print();
9         myLinkedList29.addLast(500);
10        myLinkedList29.print();
11        myLinkedList29.insertAfter(700, 300);
12        myLinkedList29.print();
13    }
14 }
15
```

Output :

```
Linked list kosong
Isi linked list : 800
Isi linked list : 700    800
Isi linked list : 700    800    500
Isi linked list : 700    300    800    500
PS D:\KULIAH\college\smt 2\29_yonanda_asd\P11>
```

Pertanyaan :

1. Mengapa class LinkedList tidak memerlukan method isFull() seperti halnya Stack dan Queue?
➔ Karena dalam LinkedList tidak memiliki batasan seperti queue dan stack yang menggunakan array (ukurannya telah ditentukan sebelumnya). Stack dan queue memiliki kapasitas maksimum yang telah ditentukan sebelumnya, ketika jumlah elemen di dalamnya telah mencapai kapasitas maksimum, struktur data tersebut

dianggap penuh , oleh karena itu method “isFull()” penting untuk memeriksa apakah stack dan queue telah mencapai kapasitas maksimum sehingga tidak dapat menambahkan elemen baru lagi.

2. Mengapa class LinkedList hanya memiliki atribut head yang menyimpan informasi node pertama? Bagaimana informasi node kedua dan lainnya diakses?

➔ 1 node terdiri dari atribut data dan next. Linkedlist hanya memiliki atribut head untuk menyimpan node pertama, karena head adalah titik awal atau pintu gerbang untuk mengakses seluruh linked list. Untuk mengakses node kedua, pada node head, terdapat atribut data yang berisi elemen, dan next yang berisi alamat node selanjutnya dalam linked list

3. Pada langkah, jelaskan kegunaan kode berikut

```
if (currentNode.data == key) {  
    newNode.next = currentNode.next;  
    currentNode.next = newNode;  
    break;  
}
```

➔ Kode tersebut digunakan untuk method insertAfter(), dimana method tersebut digunakan untuk menambahkan node baru pada posisi setelah node yang berisi data tertentu (key). Jika linked list tidak kosong, maka program akan menjalankan insertAfter(). Jika data currentNode menjadi key (yaitu node sebelumnya). Lalu, next dari node key akan diisi alamat dari node yang akan dimasukkan, kemudian insert data berhasil dilakukan.

4. Implementasikan method insertAt(int index, int key) dari tugas mata kuliah ASD (Teori)

➔ Impelmentasi method insertAfter terjadi pada program berikut :

```

public void insertAfter(int key, int input) {
    Node29 newNode = new Node29(input, next:null);

    if (!isEmpty()) {
        Node29 currentNode = head;

        do {
            if (currentNode.data == key) {
                newNode.next = currentNode.next;
                currentNode.next = newNode;
                System.out.println(x:"Insert data is succeed");
                break;
            }

            currentNode = currentNode.next;
        } while (currentNode != null);
    } else {
        System.out.println(x:"Linked list kosong");
    }
}

```

```

Linked list kosong
Isi linked list : 800
Isi linked list : 700    800
Isi linked list : 700    800    500
Insert data is succeed
Isi linked list : 700    300    800    500
PS D:\KULIAH\college\smt 2\29_yonanda_asd\P11>

```

PRAKTIKUM 2 : Mengakses dan Menghapus Node pada Linked List

Kode program class LinkedList29 :

```
1 public int getData(int index) {
2     Node29 currentNode = head;
3
4     for (int i = 0; i < index; i++) {
5         currentNode = currentNode.next;
6     }
7
8     return currentNode.data;
9 }
10
11 public int indexOf(int key) {
12     Node29 currentNode = head;
13     int index = 0;
14
15     while (currentNode != null && currentNode.data != key) {
16         currentNode = currentNode.next;
17         index++;
18     }
19
20     if (currentNode == null) {
21         return -1;
22     } else {
23         return index;
24     }
25 }
26
27 public void removeFirst() {
28     if (!isEmpty()) {
29         head = head.next;
30     } else {
31         System.out.println("Linked list kosong");
32     }
33 }
34
35 public void removeLast() {
36     if (isEmpty()) {
37         System.out.println("Linked list kosong");
38     } else if (head.next == null) {
39         head = null;
40     } else {
41         Node29 currentNode = head;
42
43         while (currentNode.next != null) {
44             if (currentNode.next.next == null) {
45                 currentNode.next = null;
46                 break;
47             }
48         }
49
50         currentNode = currentNode.next;
51     }
52 }
53
54 public void remove(int key) {
55     if (!isEmpty()) {
56         System.out.println("Linked list kosong");
57     } else if (head.data == key) {
58         removeFirst();
59     } else {
60         Node29 currentNode = head;
61
62         while (currentNode.next != null) {
63             if (currentNode.next.data == key) {
64                 currentNode.next = currentNode.next.next;
65                 break;
66             }
67
68             currentNode = currentNode.next;
69         }
70     }
71 }
```

Kode program class SSLMain29 :

```
1 public class SSLMain29 {
2     public static void main(String[] args) {
3         LinkedList29 myLinkedList29 = new LinkedList29();
4
5         myLinkedList29.print();
6         myLinkedList29.addFirst(800);
7         myLinkedList29.print();
8         myLinkedList29.addFirst(700);
9         myLinkedList29.print();
10        myLinkedList29.addLast(500);
11        myLinkedList29.print();
12        myLinkedList29.insertAfter(700, 300);
13        myLinkedList29.print();
14
15        System.out.println("Data pada index ke-1 : " + myLinkedList29.getData(1));
16        System.out.println("Data 300 berada pada index ke-" + myLinkedList29.indexOf(300));
17        myLinkedList29.remove(300);
18        myLinkedList29.print();
19        myLinkedList29.removeFirst();
20        myLinkedList29.print();
21        myLinkedList29.removeLast();
22        myLinkedList29.print();
23    }
24 }
25
```

Output :

```
Linked list kosong
Isi linked list : 800
Isi linked list : 700    800
Isi linked list : 700    800    500
Isi linked list : 700    300    800    500
Data pada index ke-1 : 300
Data 300 berada pada index ke-1
Isi linked list : 700    800    500
Isi linked list : 800    500
Isi linked list : 800
PS D:\KULIAH\college\smt 2\29_yonanda_asd\P11>
```


Pertanyaan

1. Jelaskan maksud potongan kode di bawah pada method remove()!

```
if (currentNode.next.data == key) {  
    currentNode.next = currentNode.next.next;  
    break;  
}
```

➔ Method remove digunakan untuk menghapus node dengan data sesuai key. `currentNode.next.data == key` digunakan untuk memeriksa apakah data dari node berikutnya yaitu `currentNode.next.data` sama dengan nilai key? Jika iya maka kita telah menemukan node yang ingin dihapus. `currentNode.next = currentNode.next.next`, berarti jika data dari node berikutnya adalah sama dengan nilai key, hal tersebut berarti node berikutnya harus dihapus. Setelah node yang ingin dihapus ditemukan dan dihapus, program akan berhenti dengan break

2. Jelaskan maksud if-else block pada method indexOf() berikut

```
if (currentNode == null) {  
    return -1;  
} else {  
    return index;  
}
```

➔ Kode program tersebut bertujuan untuk mengembalikan indeks dari node yang memiliki nilai data yang sama dengan key. Pengecekan `currentNode == null` dilakukan setelah loop while selesai berjalan. Jika `currentNode` adalah null, maka loop telah mencapai akhir linkedlist tanpa menemukan node dengan nilai yang sama dengan key. Dalam hal ini, kita mereturnkan -1, yang mengindikasikan bahwa node dengan key tersebut tidak ditemukan dalam linked list. Jika `currentNode` tidak null, yang menunjukkan bahwa node dengan key telah ditemukan dalam linked list, method akan emngembalikan nilai index yang mrupakan indeks dari node tersebut dalam linked list

3. Error apa yang muncul jika argumen method `getData()` lebih besar dari jumlah node pada linked list? Modifikasi kode program untuk handle hal tersebut.

➔ Jika argumen yang diberikan kepada metode `getData()` lebih besar dari jumlah node pada linked list, maka akan terjadi error `NullPointerException`. Hal ini terjadi karena pada saat iterasi melalui linked list, mencoba untuk mengakses `currentNode` yang bernilai null pada iterasi tertentu.

4. Apa fungsi keyword break pada method remove()? Bagaimana efeknya jika baris tersebut dihapus?

➔ Keyword break pada metode remove() digunakan untuk menghentikan iterasi ketika sebuah node dengan nilai data yang sesuai dengan kunci ditemukan dan dihapus. Dengan menghapus break, kita kehilangan mekanisme untuk menghentikan iterasi setelah node yang dihapus ditemukan,

TUGAS

1. Implementasikan method-method berikut pada class LinkedList:
 - a. insertBefore() untuk menambahkan node sebelum keyword yang diinginkan
 - b. insertAt(int index, int key) untuk menambahkan node pada index tertentu
 - c. removeAt(int index) untuk menghapus node pada index tertentu
2. Dalam suatu game scavenger hunt, terdapat beberapa point yang harus dilalui peserta untuk menemukan harta karun. Setiap point memiliki soal yang harus dijawab, kunci jawaban, dan pointer ke point selanjutnya. Buatlah implementasi game tersebut dengan linked list.

JAWAB

1. Berikut adalah kode program penerapannya pada class LinkedList29 :

```

1 public void insertBefore(int key, int input) {
2     Node29 newNode = new Node29(input, null);
3
4     if (isEmpty()) {
5         if (head.data == key) {
6             addFirst(input);
7             return;
8         }
9
10        Node29 currentNode = head;
11
12        while (currentNode.next != null) {
13            if (currentNode.next.data == key) {
14                newNode.next = currentNode.next;
15                currentNode.next = newNode;
16                break;
17            }
18            currentNode = currentNode.next;
19        }
20    } else {
21        System.out.println("Linked list kosong");
22    }
23 }
24
25 public void insertAt(int index, int key) {
26     if (index == 0) {
27         addFirst(key);
28         return;
29     }
30
31     Node29 newNode = new Node29(key, null);
32     Node29 currentNode = head;
33
34     for (int i = 0; i < index - 1; i++) {
35         if (currentNode == null) {
36             System.out.println("Indeks diluar batas");
37             return;
38         }
39         currentNode = currentNode.next;
40     }
41
42     if (currentNode == null) {
43         System.out.println("Indeks diluar batas");
44         return;
45     }
46
47     newNode.next = currentNode.next;
48     currentNode.next = newNode;
49 }
50
51 public void removeAt(int index) {
52     if (isEmpty()) {
53         System.out.println("Linked list kosong");
54         return;
55     }
56
57     if (index == 0) {
58         removeFirst();
59         return;
60     }
61
62     Node29 currentNode = head;
63
64     for (int i = 0; i < index - 1; i++) {
65         if (currentNode == null || currentNode.next == null) {
66             System.out.println("Indeks diluar batas");
67             return;
68         }
69         currentNode = currentNode.next;
70     }
71
72     if (currentNode.next == null) {
73         System.out.println("Indeks diluar batas");
74         return;
75     }
76
77     currentNode.next = currentNode.next.next;
78 }

```

2. Berikut adalah kode programnya :
Kode program class ScavengerHunt :

```
1  import java.util.Scanner;
2
3  public class ScavengerHunt {
4      ScavengerHuntNode head;
5
6      public ScavengerHunt() {
7          this.head = null;
8      }
9
10     public void addPoint(String question, String answer) {
11
12         ScavengerHuntNode newNode = new ScavengerHuntNode(question, answer);
13
14         if (head == null) {
15             head = newNode;
16         } else {
17             ScavengerHuntNode current = head;
18             while (current.next != null) {
19                 current = current.next;
20             }
21             current.next = newNode;
22         }
23     }
24
25     public void startHunt() {
26
27         Scanner scanner = new Scanner(System.in);
28         ScavengerHuntNode current = head;
29
30         while (current != null) {
31
32             System.out.println("Pertanyaan : " + current.question);
33             System.out.print("Jawaban : ");
34             String userAnswer = scanner.nextLine();
35
36             if (userAnswer.equalsIgnoreCase(current.answer)) {
37                 System.out.println("AMDA BENAR!! Silahkan lanjut ke pertanyaan selanjutnya");
38                 current = current.next;
39             } else {
40                 System.out.println("Sayang sekali jawabarmu salah :( Silahkan coba lagi!!");
41             }
42
43             System.out.println();
44
45         }
46
47         System.out.println("SELAMATTT! Kamu Berhasil Menemukan Harta Karun!");
48         scanner.close();
49
50     }
51 }
52 }
```

Kode program class ScavengerHuntNode :

```

1  public class ScavengerHuntNode {
2      String question;
3      String answer;
4      ScavengerHuntNode next;
5
6      public ScavengerHuntNode(String question, String answer) {
7          this.question = question;
8          this.answer = answer;
9          this.next = null;
10     }
11 }

```

Kode program class ScavengerHuntMain :

```

1  public class ScavengerHuntMain {
2      public static void main(String[] args) {
3
4          ScavengerHunt scavengerHunt = new ScavengerHunt();
5
6          scavengerHunt.addPoint("Apa yang bisa terbang tanpa sayap dan bisa menangis tanpa mata?", "Awan");
7          scavengerHunt.addPoint("Apa yang selalu mengikuti Anda, namun tidak bisa Anda raba?", "Bayangan");
8          scavengerHunt.addPoint("Apa yang bisa Anda lihat tetapi tidak bisa Anda dengar?", "Mimpi");
9
10         scavengerHunt.startHunt();
11     }
12 }
13 }

```

Output :

```

Pertanyaan : Apa yang bisa terbang tanpa sayap dan bisa menangis tanpa mata?
Jawaban    : Awan
ANDA BENAR!! Silahkan lanjut ke pertanyaan selanjutnya

Pertanyaan : Apa yang selalu mengikuti Anda, namun tidak bisa Anda raba?
Jawaban    : Bayangan
ANDA BENAR!! Silahkan lanjut ke pertanyaan selanjutnya

Pertanyaan : Apa yang bisa Anda lihat tetapi tidak bisa Anda dengar?
Jawaban    : Percakapan
Sayang sekali jawabanmu salah :( Silahkan coba lagi!)

Pertanyaan : Apa yang bisa Anda lihat tetapi tidak bisa Anda dengar?
Jawaban    : 

```